

Keil C51 μ Vision4 的使用














和春技術學院 資工系 李鴻鵬老師編寫
教育人員或學生可任意下載使用或連結但不得有商業行為

Keil C51 μ Vision4 的使用

Keil C51 μ Vision4 是美國 Keil Software 公司出品的 51 系列相容單晶片 C 語言整合發展環境(Integrated Development Environment, IDE)，與組合語言相比，C 語言在功能上、架構性、可讀性、可維護性上有明顯的優勢，因而易學易用。用過組合語言後再使用 C 語言來開發，會更加體會深刻。目前最新的版本是 C51 Version 9.00 Release，也就是所謂的 μ Vision4。

Keil C51 軟體提供豐富的函數庫和功能強大的除錯工具，及全視窗界面。另外重要的一點，只要看一下編譯後生成的組合語言代碼，就能體會到 Keil C51 生成的目標代碼效率非常之高，多數語句生成的組合語言代碼很緊湊，容易理解。在開發大型軟體時更能體現高階語言的便利與優勢。

Keil C51 μ Vision4 較之前的版本，新增了下面幾個功能：

-  多重的監控和彈性的視窗管理系統。
-  系統檢視器(System Viewer) - 顯示裝置周邊暫存器的訊息。
-  除錯恢復檢視(Debug Restore Views) - 建立和儲存多重除錯視窗設計。
-  多專案工作區(Multi-Project Workspace) - 與許多專案簡化工作。
-  源碼和解組譯連結(Source and Disassembly Linking) - 解組譯視窗和源碼視窗完全同步使程式除錯和游標導航較容易。
-  記憶體視窗固定(Memory Window Freeze) - 儲存目前記憶體視窗檢視允許容易在不同的點及時比較。
-  設備模擬 - 更新支援很多新設備(例如 Infineon XC88x, SiLABS C8051Fxx, Atmel SAM7/9 和從 Luminary, NXP, and Toshiba 來的 Cortex M3 MCUs)。
-  支援硬體除錯轉接器(Support for Hardware debug adapters) - 包括 ADI, miDAS-Link, Atmel SAM-ICE, Infineon DAS, 和 ST-Link。
-  新資料和指令追蹤(New Data and instruction trace) - 對 ARM 和 Cortex MCUs。
-  基於 XML 的專案檔案(XML based Project files) - 建立, 檢視和修改專案如同容易可讀的 XML 本文檔案一樣。
-  串列的視窗 - 擴充到提供一個基本的 100-VT 終端機, ASCII 模式, 混合模式, 和十六進制模式檢視。
-  拖放檔案開啓(Drag & Drop File Opening) - 檔案拖進 μ Vision4 專案空間自動會被開啓。
-  監控點和邏輯分析儀(Watchpoints and Logic Analyzer) - 現在更容易設定變數。

下面詳細介紹 Keil C51 μ Vision4 IDE 基本的功能和使用。

第一章 建立第一個 Keil C51 程式-使用 C 語言

隨著單晶片技術的不斷發展，以單晶片 C 語言為主的高階語言 IDE，也不斷的被開發出來，而且受到許多的單晶片愛好者和工程師所喜愛，更在學校中被廣泛的使用在單晶片課程或微處理機課程教學上。Keil C51 μ Vision4 是眾多單晶片 IDE 軟體中優秀的軟體之一，它支援許多不一樣公司的 MCS-51 架構的晶片，它集編輯(Edit)，編譯(Compiler)，模擬(Simulation)等於一體，同時還支援，PLM，組合語言和 C 語言的程式設計，它的界面和微軟的 VC++的界面相似，易學易用，在程式除錯，軟體模擬方面也有很強大的功能。使用 C51 寫好 C 程式，然後用 C51 的編譯器把寫好的 C 程式編譯為機器碼，這樣單晶片才能執行編寫好的 C 程式。

下面結合 MCS-51 介紹單晶片 C 語言的優越性：

- ✚ 不須完全懂得單晶片的硬體架構，也能夠編寫出符合硬體實際的專業水準的程式。
- ✚ 不懂完全得單晶片的指令集，也能夠編寫單晶片程式。
- ✚ 不同函數的數據實行覆蓋，有效利用單晶片上有限的 RAM 空間。
- ✚ 提供 auto, static, 和 const 等存儲類型和專門針對 8051 單晶片的 data, idata, pdata, data, 和 code 等存儲類型，自動為變數合理地配置位址。
- ✚ C 語言提供複雜的數據類型（陣列(Array)、結構(Structure)、聯合(Union)、枚舉(Enumeration)、指標(Pointer)等），極大地增強了程式處理能力和靈活性。
- ✚ 提供 small, compact, 和 large 等編譯模式，以適應單晶片上記憶體的大小。
- ✚ 中斷服務程式的現場保護和恢復，中斷向量表的填寫，是直接與單晶片相關的，都是由 C 編譯器代辦。
- ✚ 程式具有堅固性：數據被破壞是導致程式執行異常的重要因素。C 語言對數據進行了許多專業性的處理，避免了執行中間不正常的破壞。
- ✚ 提供常用的標準函數庫，以供用戶直接使用。
- ✚ 有嚴格的句法檢查，錯誤很少。
- ✚ 可方便地接受多種實用程式的服務：如單晶片上資源的初始化有專門的實用程式自動生成，簡化用戶程式設計，提升執行的安全性等等。
- ✚ 表頭檔案(header)中定義、說明複雜數據類型和函數原型，有利於程式的移植和支援單晶片的系列化產品的開發。

以上簡單介紹了 Keil C51 軟體，要使用 Keil C51 軟體，必需先要安裝它，這也是學習單晶片 C 語言所要求的第一步的建立學習環境。

使用者可到Keil C51 的官方網站下載(<https://www.keil.com/demo/eval/c51.htm>)，在網頁中填妥個人資料，就可下載免費版，如圖 1-1 所示。此免費版有 2K ROM大小的限制，不過一般通常在學校使用，使用 2K ROM的大小就綽綽有餘了，若使用超過 2K ROM的大小的話，就必須購買正式版了。安裝方法很簡單，安裝時只要點選C51V900.exe，就可自動執行安裝了，其他後續版本也都一樣，這裡就不做介紹了。

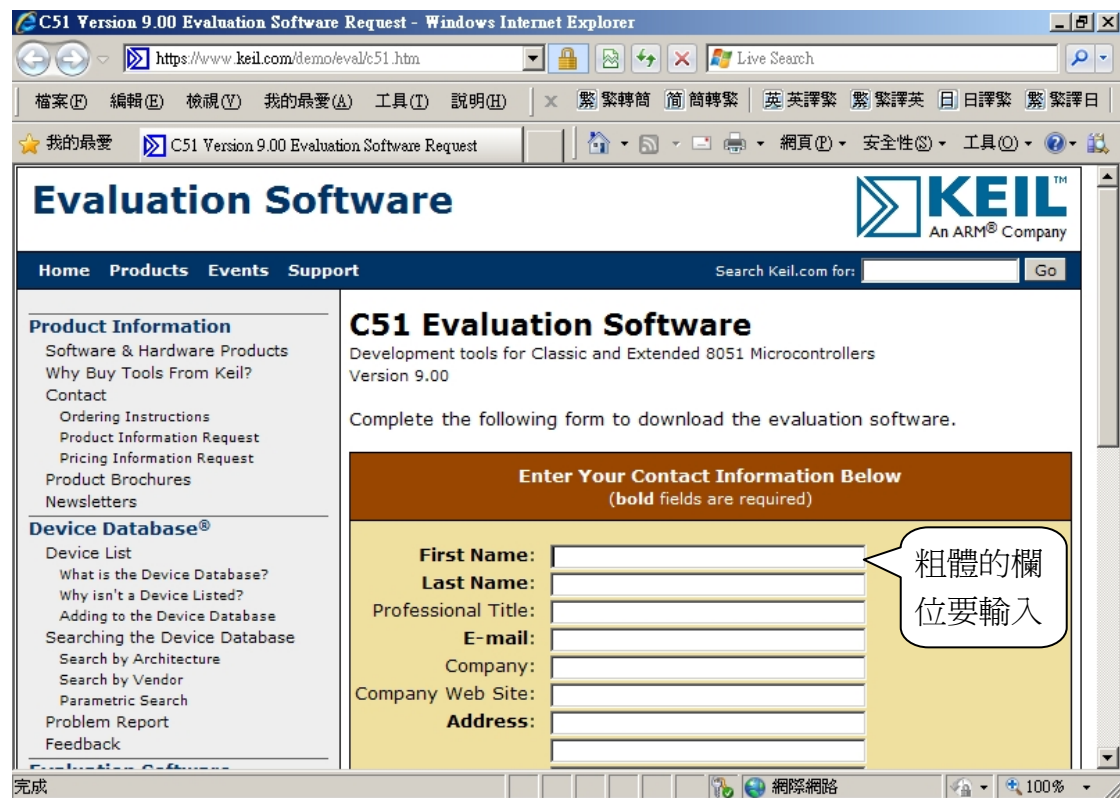


圖 1-1 Keil C51 μ Vision4 官方下載網站

安裝好 C51 後，C51 的初始主畫面如圖 1-2 所示。

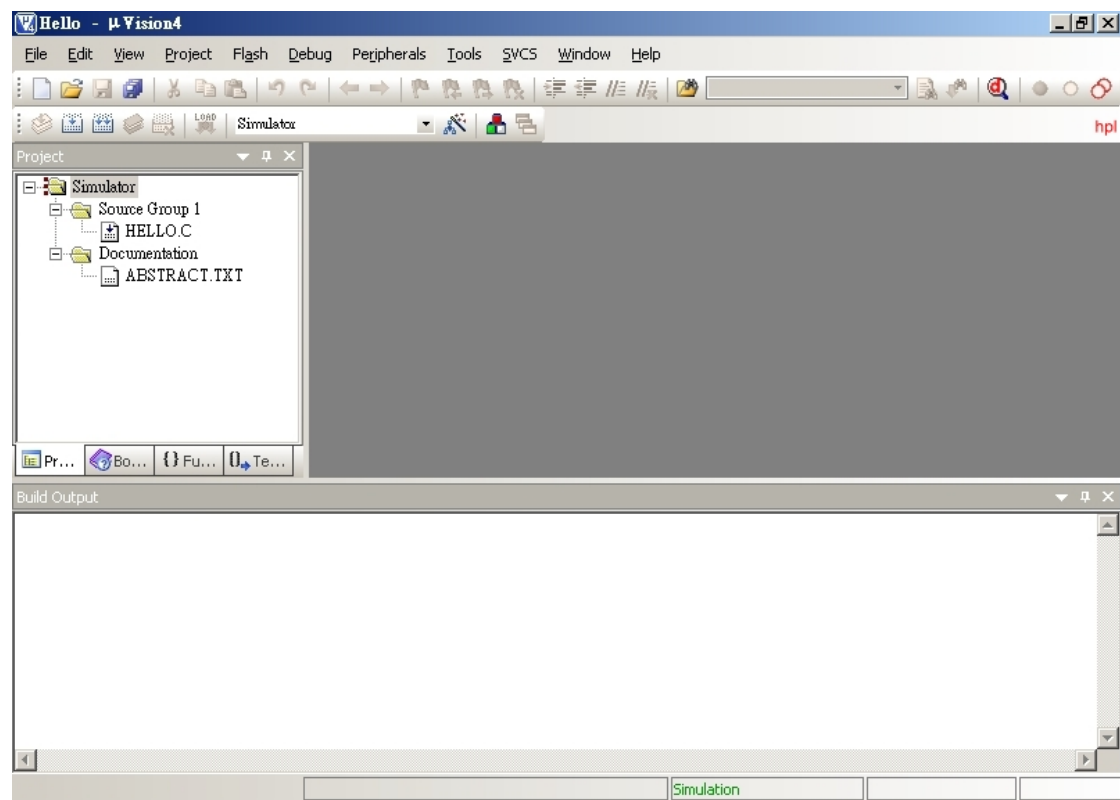


圖 1-2 Keil C51 初始主畫面

接著下面就讓我們一起來建立自己的第一個單晶片 C 語言程式吧。請根據下面步驟

一步步的來，你絕對可以在短時間內熟悉 C51 的使用。

(1) 點擊 Project(專案)選單，選擇彈出的下拉式選單中的“New μ Vision Project...”，如圖 1-3。接著彈出一個標準 Windows 檔案對話視窗，如圖 1-4。在“儲存於”中選擇您要存放的資料夾，一個專案最好存在一個資料夾內，若此資料夾不存在，請先建立它，或按“建立新資料夾”按鈕以建立新資料夾。在“檔名”中輸入您的第一個專案名稱，這裡我們用“test1”。“存檔類型”為 uvproj，這是 Keil μ Vision4 專案檔案預設的副檔名，以後只要直接點擊此專案檔案，即可打開此專案。

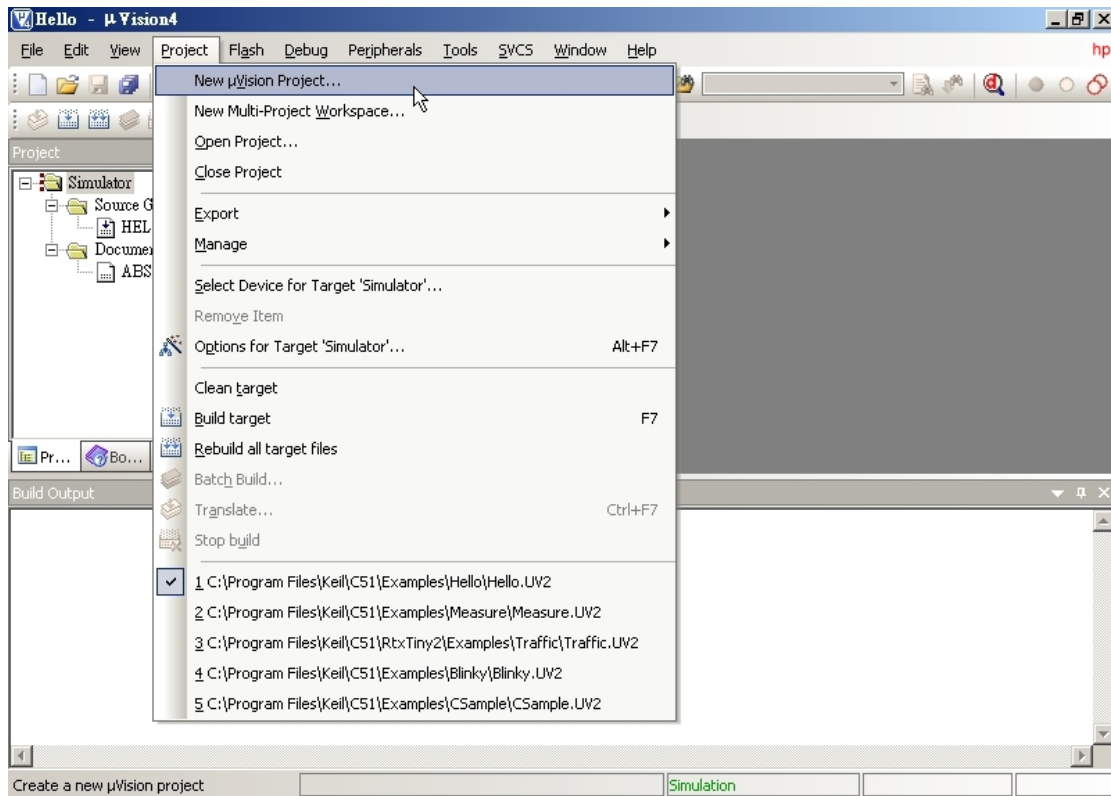


圖 1-3 New μ Vision Project 選單

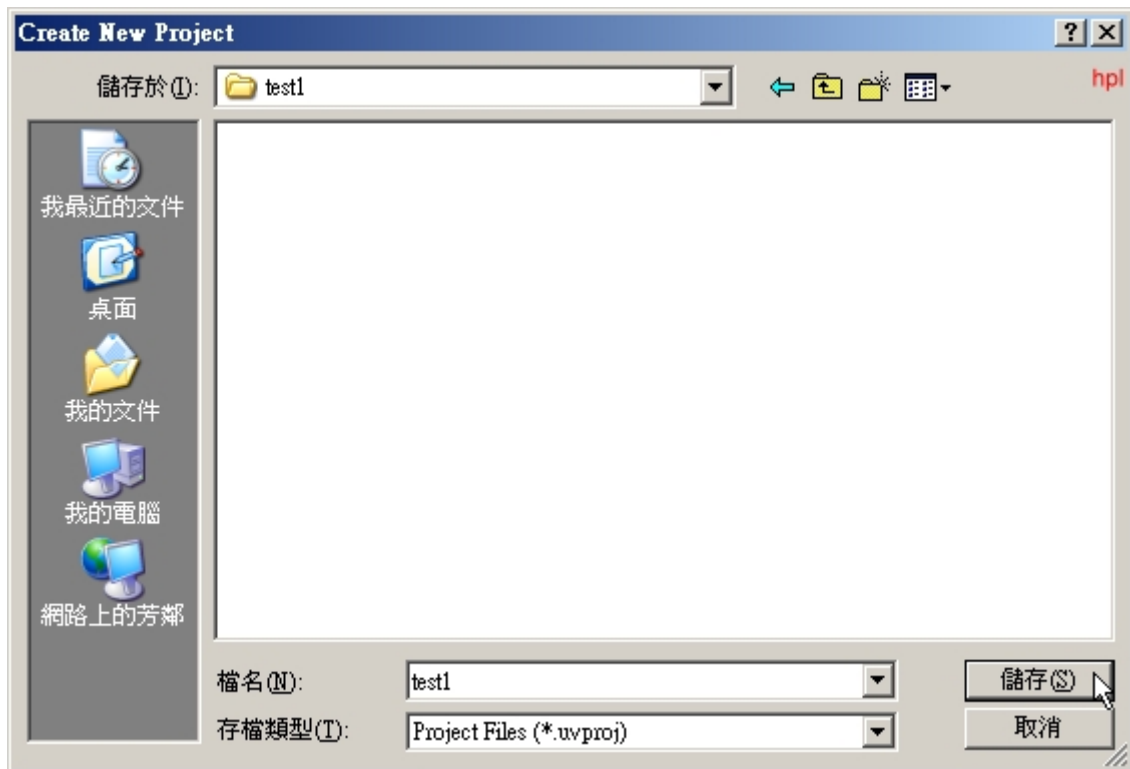


圖 1-4 New μ Vision Project 選單

若第一次使用，則會出現圖 1-5 的畫面，若您要將您之前建立的舊版的專案檔副檔名更名為新版的專案檔副檔名，則按確定，否則按取消，用戶可根據自己的需求選擇不變或更名。因為第一次使用，按取消即可。

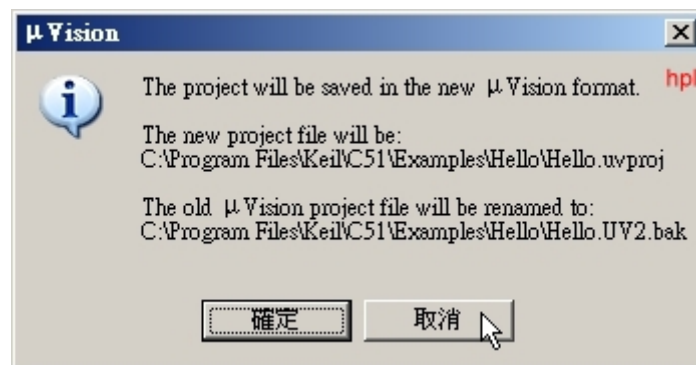


圖 1-5 選擇舊版或新版的專案檔副檔名

(2) 選擇所要的單晶片型號，這裡選擇常用的 Atmel 公司的 AT89S51。目前 Atmel 已經停產 AT89C51/AT89C52，改由 AT89S51/AT89S52 來取代之。AT89S 系列較 AT89C 系列更為便宜，工作頻率可高達 33 MHz，且重複燒錄次數至少可達一千次以上，非常適合學校學習或工程師開發產品之用。所以本講義裡的大部分程式都是基於此 AT89S51 晶片的，此時螢幕如圖 1-6 所示。在右邊圖中的“Description”方塊內，會簡單的介紹 AT89S51 有什麼功能及特點。點選 OK 按鈕後，會出現圖 1-7，詢問你是否需要拷貝標準的 8051 啟動碼程式(STARTUP.a51)到你的專案資料夾，並且將此檔案加入專案“Copy Standard 8051 Startup Code to Project Folder and Add File to Project”，點選“是”後，就可以進行程式的編寫了。

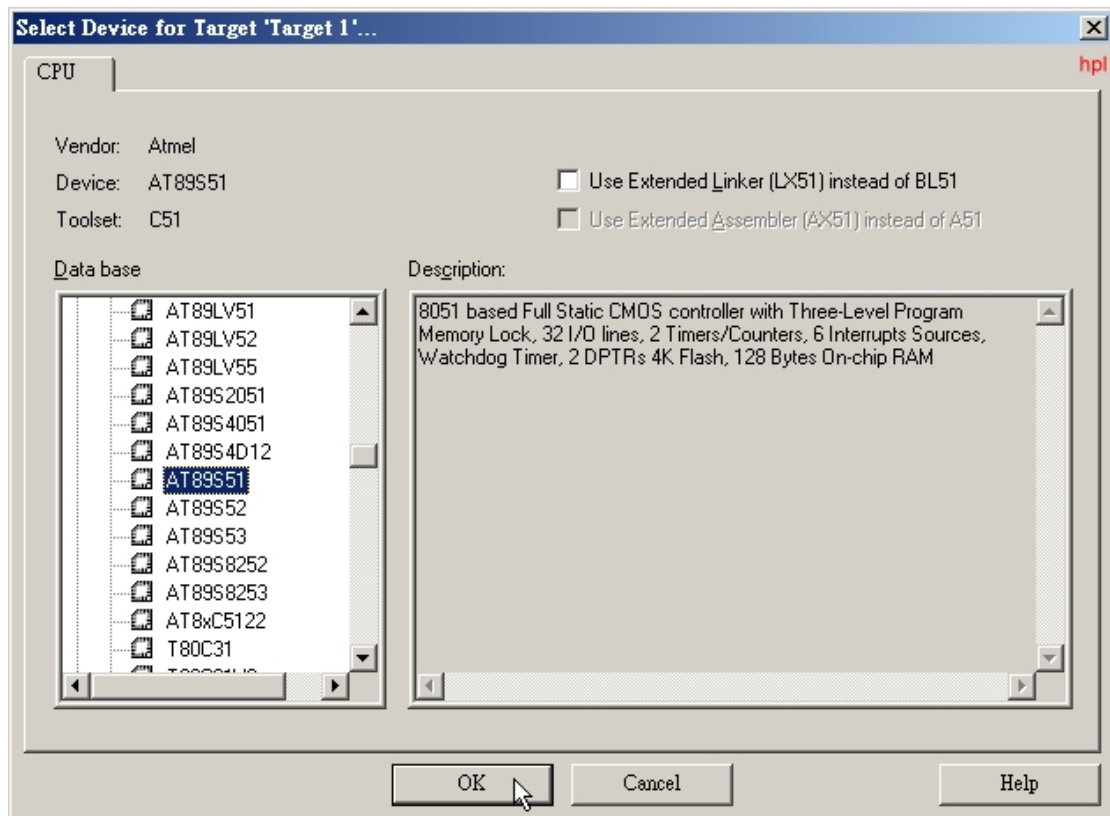


圖 1-6 選取晶片型號

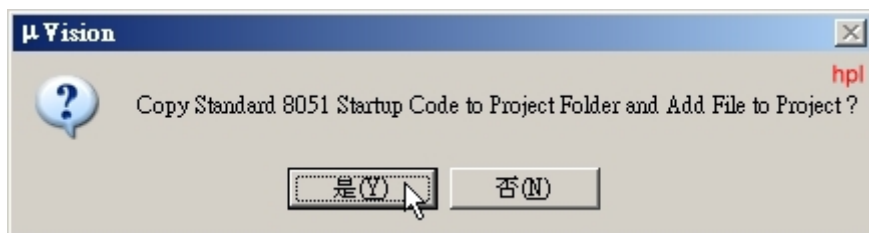


圖 1-7 詢問是否需要加入 8051 啟動碼

STARTUP.a51 的主要工作是把包含 `idata`、`xdata`、及 `pdata` 在內的記憶體區塊清除為 0，並且初始化遞迴指標。STARTUP.a51 的內容在附錄 1 中，用戶可自行參考。注意，若是編寫組合語言程式，則不需加入此啟動程式。在完成上述的初始化程序之後，8051 的控制權才會交給 `main()` 主程式開始執行用戶的程式。

(3) 首先在專案中建立新的程式檔案或加入舊程式檔案。如果您沒有現成的程式或是第一次使用，那麼就要新建一個 C 程式檔案。在 C51 中有一些程式的範例，但是在這裡我們還是以一個 C 程式為例介紹如何新建一個 C 程式，和如何加到您的第一個專案中吧。點擊圖 1-8 中 1 的新建文件的圖示按鈕，在 2 中出現一個新的文字編輯視窗，或是也能透過選單 `File/New` 或是按下快速鍵 `Ctrl+n` 來實現。接著現在就能編寫程式了。

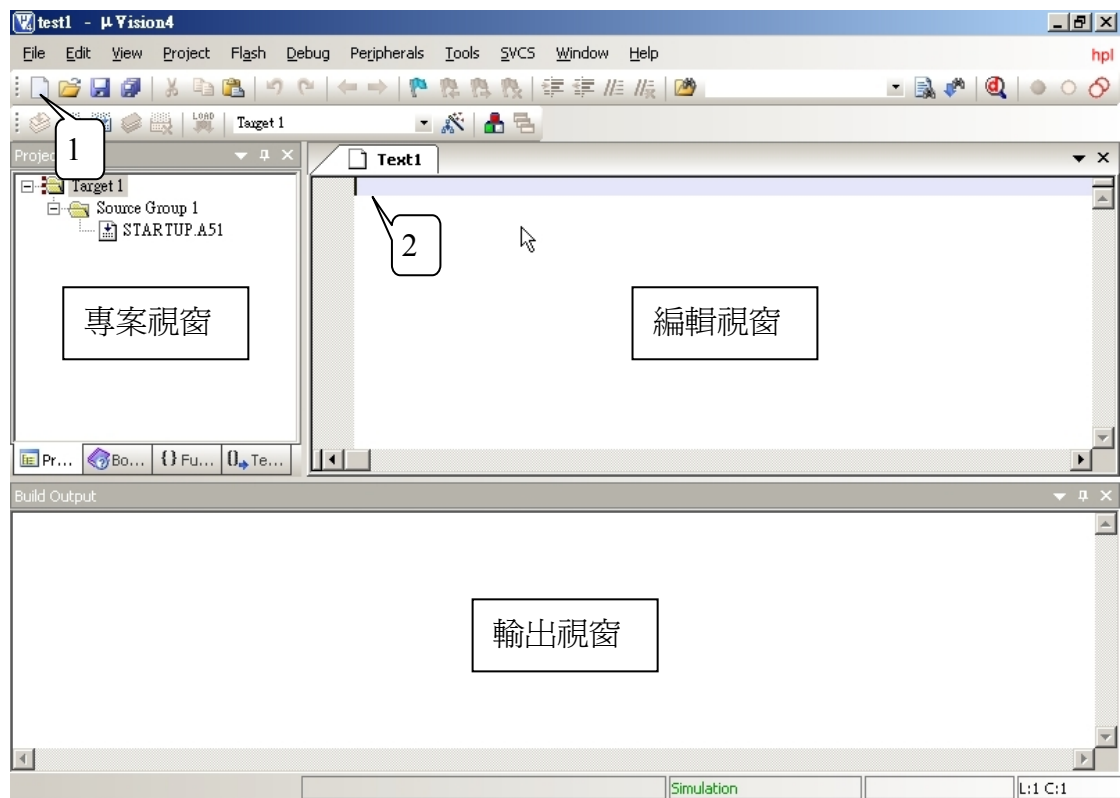


圖 1-8 新建文件

下面是一段一顆 LED 在閃爍的程式，在上圖 2 的文件編輯工作區內鍵入下面的程式，先不管程式的語法和意思，先看看如何把它存檔，加入到專案中存檔，和如何編譯及執行。AT89X51.h 的內容在附錄 2 中，若你用的單晶片是 AT89S51，則用<AT89X52.h>。

```
#include <AT89X51.h>
main()
{
    int i;
    while(1)
    {
        P0_0=1;
        for(i=1;i<20000;i++);
        P0_0=0;
        for(i=1;i<20000;i++);
    }
}
```

(4) 點擊圖 1-9 中的儲存檔案圖示按鈕，也能用選單 File/Save 或按快速鍵 Ctrl+S，則出現圖 1-10 的視窗。把此程式命名為 test1.c，儲存在專案所在的資料夾中，再按儲存鈕。這個時候您會發現程式單字有了不同的顏色，這表示 Keil 的 C 語言語法檢查開始作用了。

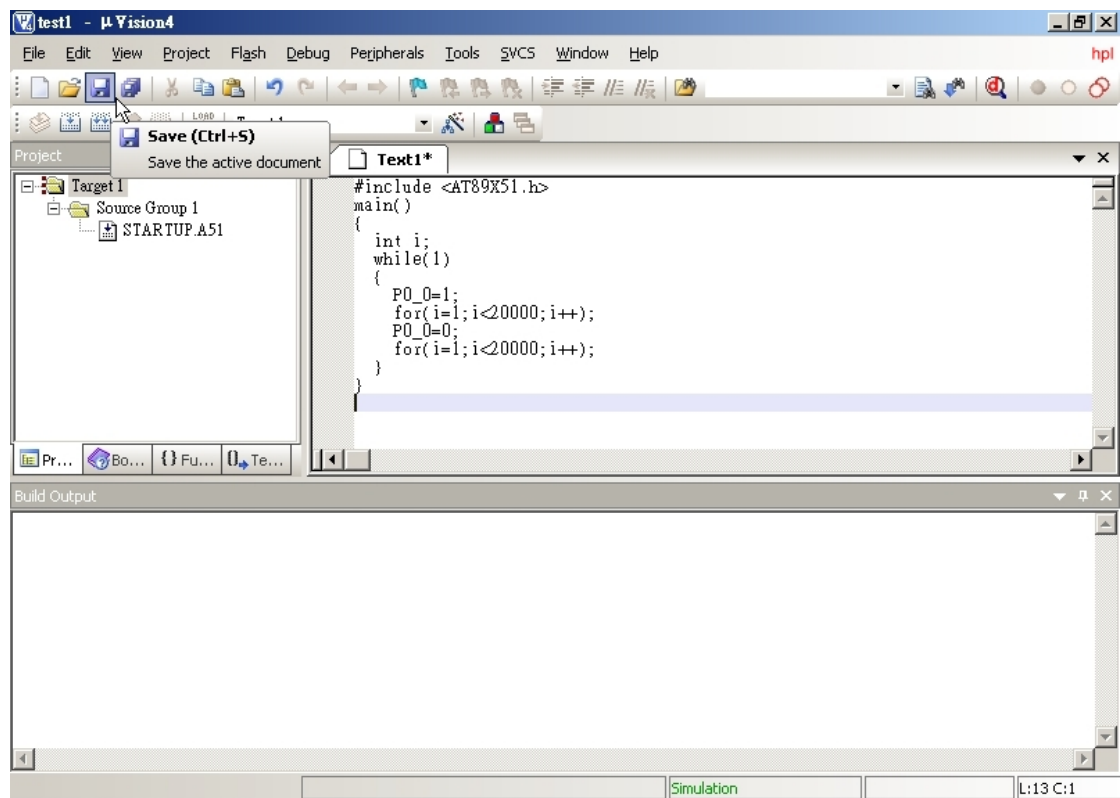


圖 1-9 儲存檔案



圖 1-10 儲存 c 檔案

(5) 滑鼠在螢幕左邊的 Source Group1 資料夾圖示上按右鍵，彈出一選單，如圖 1-11 所示，在這裡能做出專案中增加減少檔案等操作。選 “Add Files to Group ‘Source Group 1...’” 彈出檔案視窗，選擇剛剛儲存的檔案，按下 Add 按鈕，將此 c 檔案加入到

此專案中。按下 close 按鈕，關閉檔案視窗，如圖 1-12 所示，則此 test1.c 程式檔案已加到此專案中了，如圖 1-13 所示。

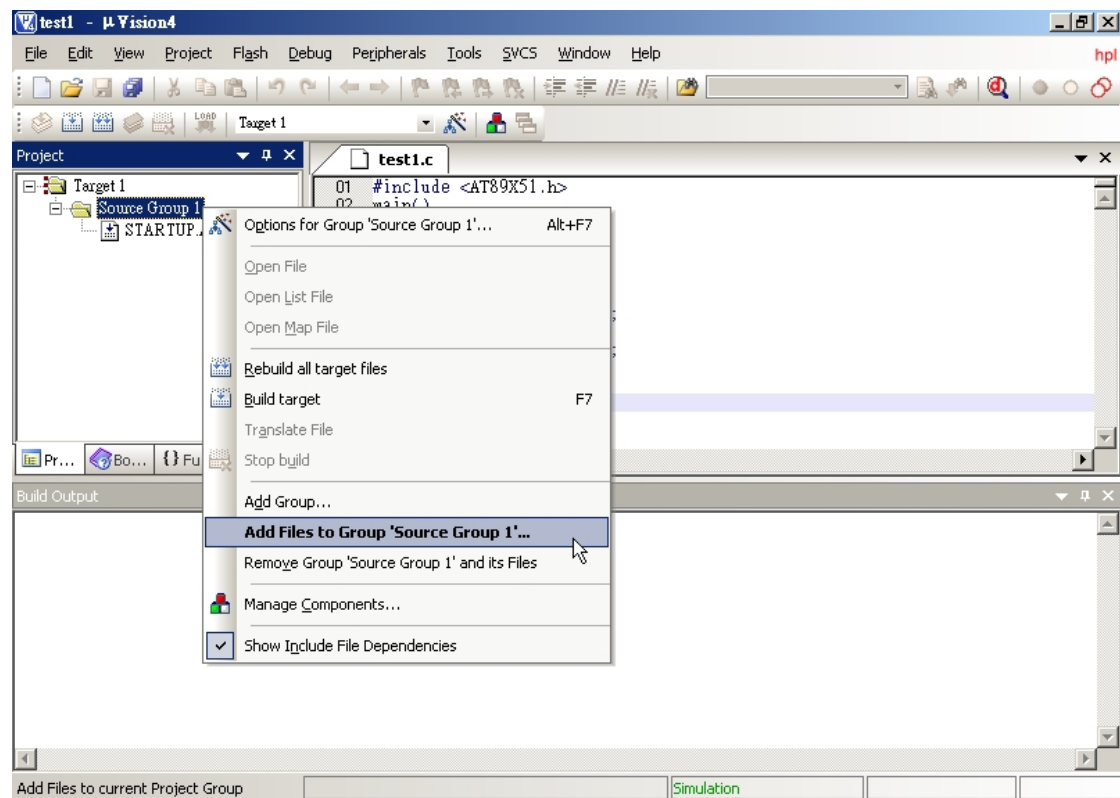


圖 1-11 選取 “Add Files to Group ‘Source Group 1’...”



圖 1-12 選取要加入到專案中的 c 檔案

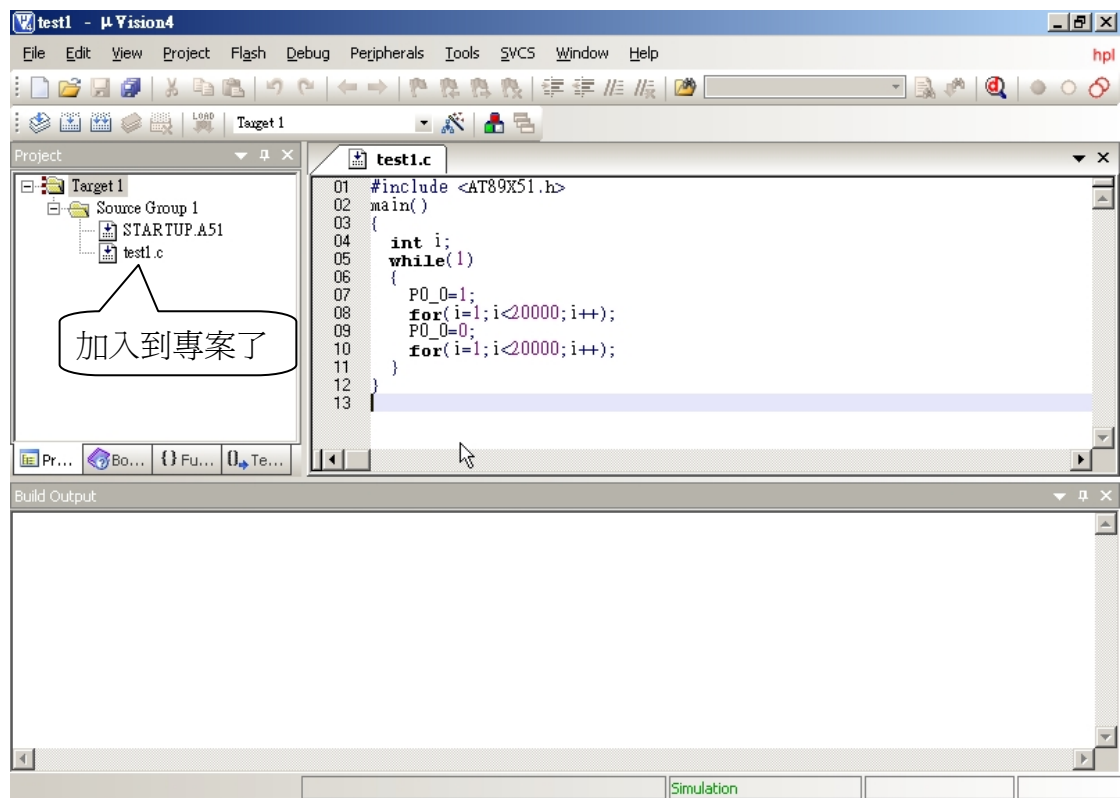


圖 1-13 把 c 檔案加入到專案中了

若用戶寫的是組合語言檔案，那必須存成*.a51 或*.asm 檔，然後將組合語言檔案加入到專案中。

(6) 接下來要來做一些基本的設定選項的工作。滑鼠在螢幕左邊的 Target 1 資料夾圖示上按右鍵，彈出一選單，如圖 1-14 所示，然後選取“Options for target ‘Target 1’...”。

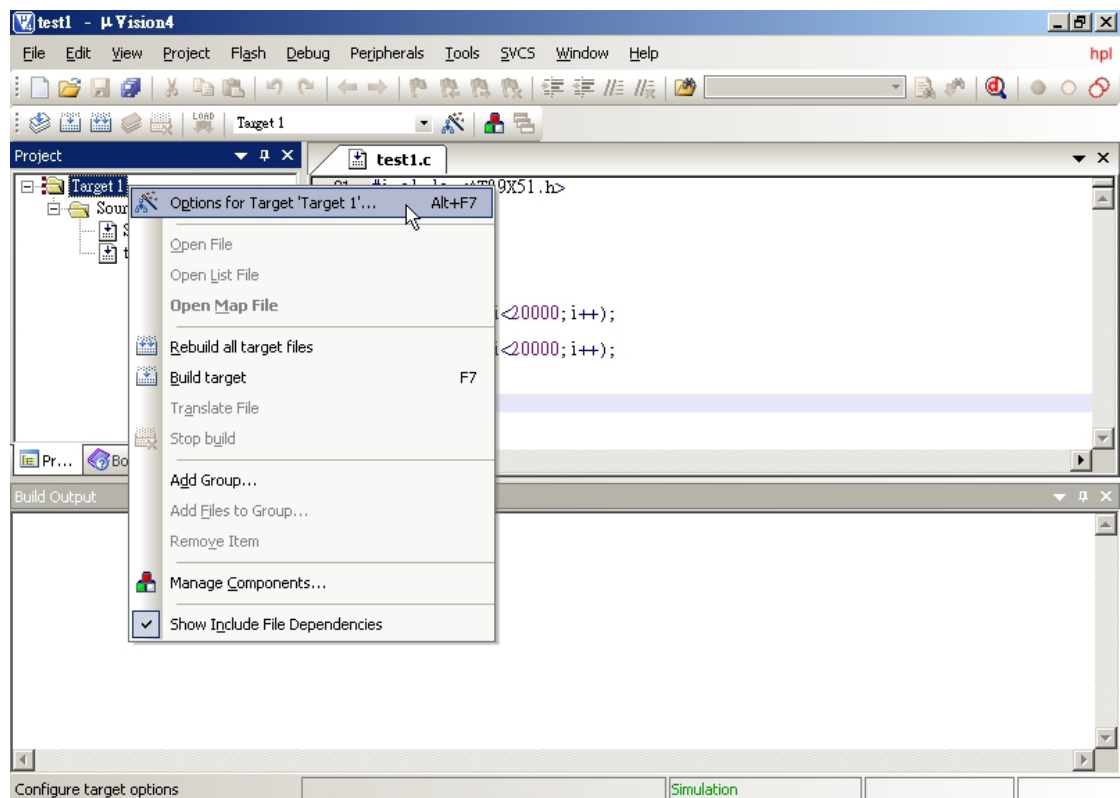


圖 1-14 選取“Options for target ‘Target 1’...”

在圖 1-15 的 Target 標籤頁中，更改所選用單晶片的工作頻率為 12。MCS-51 系列單晶片系統一般常選用 11.059 MHz 或 12 MHz。前者適用於產生各種標準的鮑率(baud rate)，後者的一個機器週期為 1 μ s，便於產生精確延遲時間。本程式中假設使用頻率為 12 MHz 的晶體振盪頻率。另外，勾選 Use On-Chip ROM(0x0-0xFFFF)，以使用單晶片上的 Flash ROM。

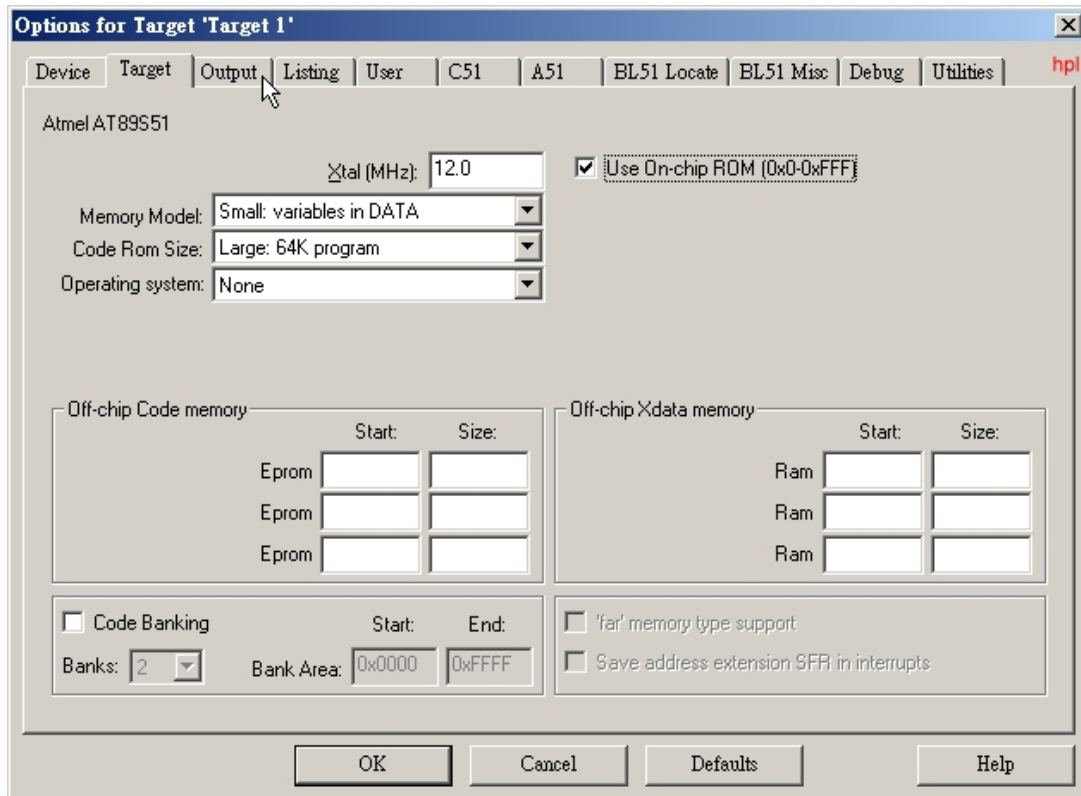


圖 1-15 在 Target 標籤頁中，更改所選用單晶片的工作頻率

再來切換到 Output 標籤頁，只要勾選“Create HEX File”就好了，以產生燒錄檔，如圖 2-13 所示。如果用戶只是單純的做練習，那就省略此步驟了。若要更改存放目的檔的資料夾，則點擊“Select Folder for Objects...”，若要更改編譯後的主檔名，則在“Name of Executable:”右邊的空格內輸入主檔名即可，一般而言，這 2 個選項都採用預設值，用戶不需更改他們。

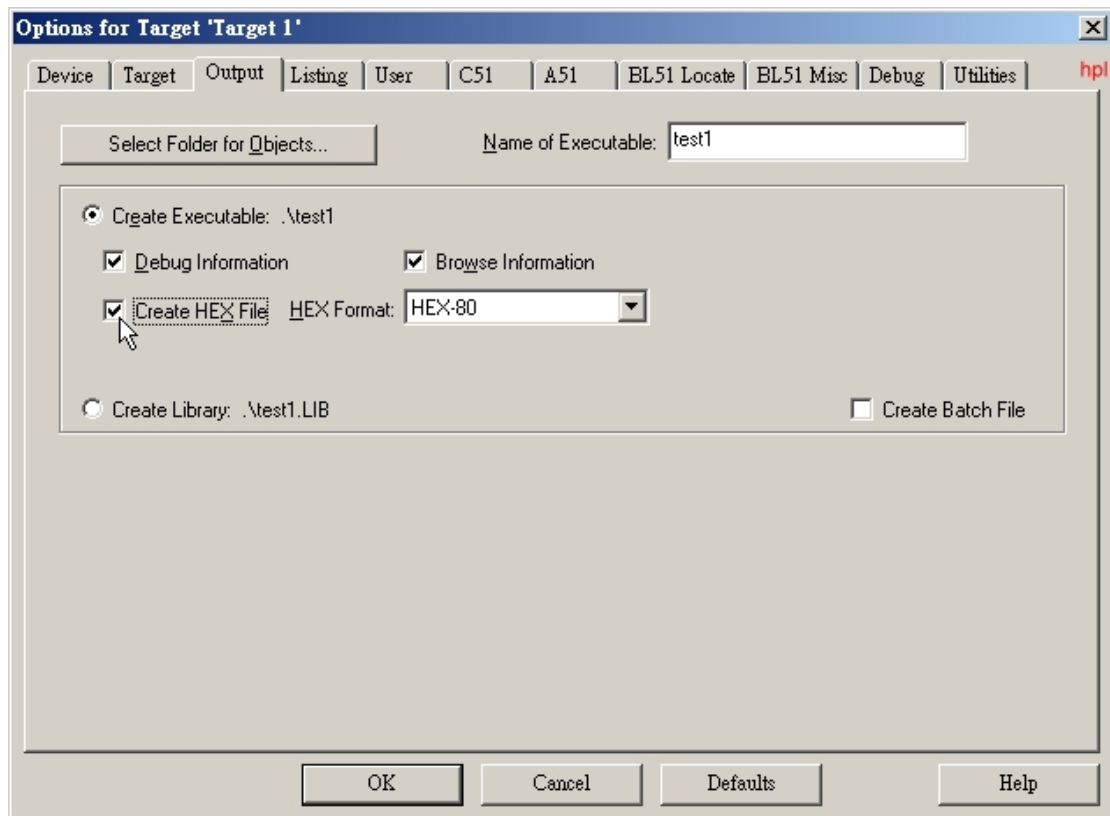







圖 1-16 勾選“Create HEX File”，以產生燒錄檔

(7) 完成基本的選項設定後，下面就剩下編譯執行了。先來看這三個按鈕 ，這三個都是編譯按鈕。 按鈕是用於編譯目前工作區的檔案但不做連結(Link)， 按鈕是用於編譯整個專案檔案並連結，如果之前編譯過一次之後檔案沒有做任何編輯的話，這個時候再點擊是不會再次重新編譯的。 按鈕是重新編譯整個專案檔案並連結，每點擊一次均會再次編譯連結一次，不管程式是否有改變。 是停止編譯按鈕，只有點擊了前三個中的任一個，停止按鈕才會生效。或是從選單 Project 中，也可執行編譯，在此按下“Build target”或 F7 快捷鍵，如圖 1-17 所示。

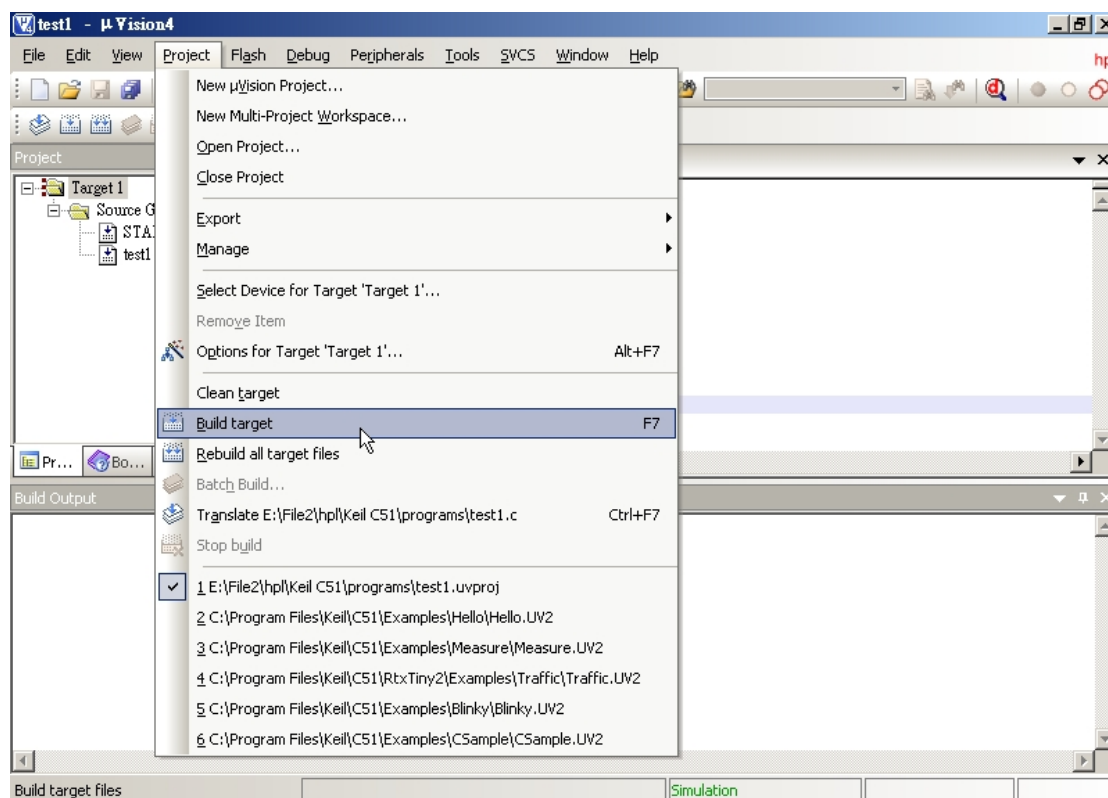


圖 1-17 編譯專案

編譯完成後，在下方的 Build Output 區域中，可看到編譯的訊息，如圖 1-18 所示。若有出現錯誤訊息，則再根據錯誤訊息，回到程式中修改，編譯完全正確後，才能產生正確的燒錄檔 test1.hex。

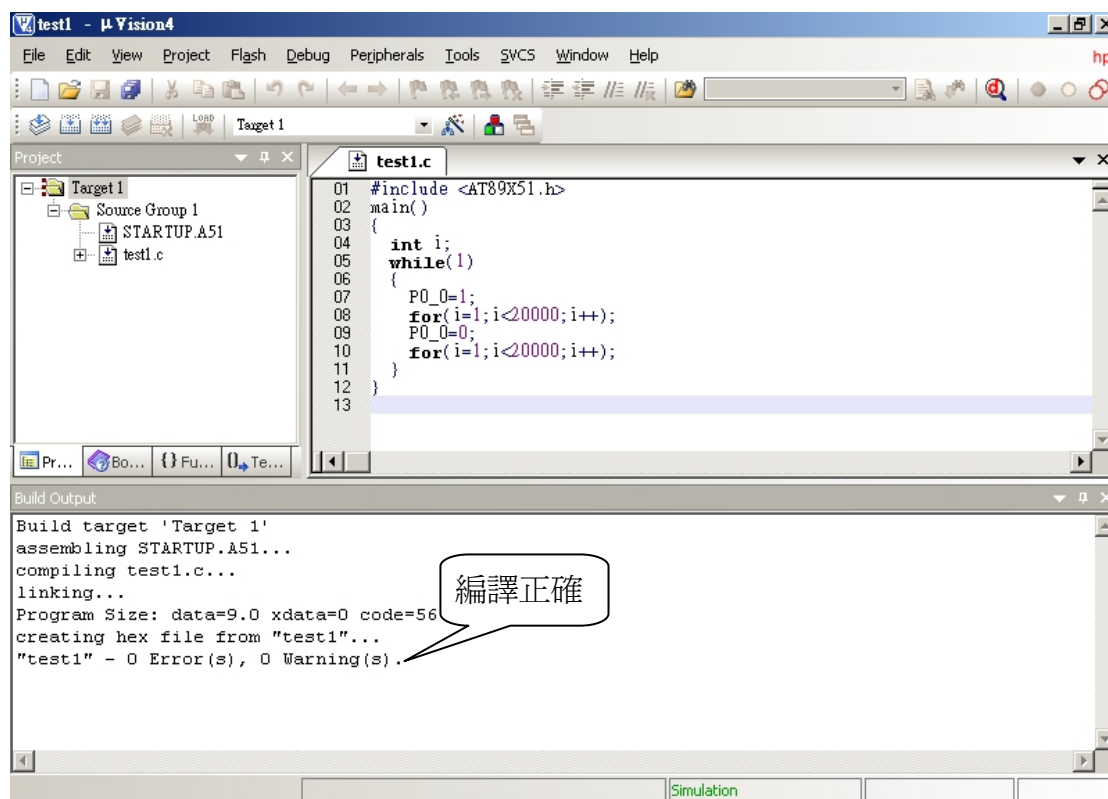










圖 1-18 專案編譯正確

(8) 選取選單 Debug/Start/Stop Debug Session，或按快速鍵 Ctrl+F5，或按在檔案工具列右邊有一個小放大鏡的按鈕，則進入除錯(Debug)模式，並顯示不同的工作視窗，如圖 1-19 所示。進入除錯模式之前，會先出現一個小視窗，告訴你目前用的版本是免費的評估版，有 2K ROM 大小的限制，點擊確定即可進入除錯模式。在除錯工具列中， (Reset) 按鈕表示重置單晶片，並使程式回到最開頭處執行。 (Run) 按鈕表示執行， (Stop) 按鈕表示停止，當程式處於執行狀態時，停止按鈕才有效。 (Step Into) 按鈕表示單步執行會進入函數內， (Step Over) 按鈕表示單步執行不會進入函數內， (Step Out) 按鈕表示離開函數， (Run to Cursor) 按鈕表示執行到游標所在處。

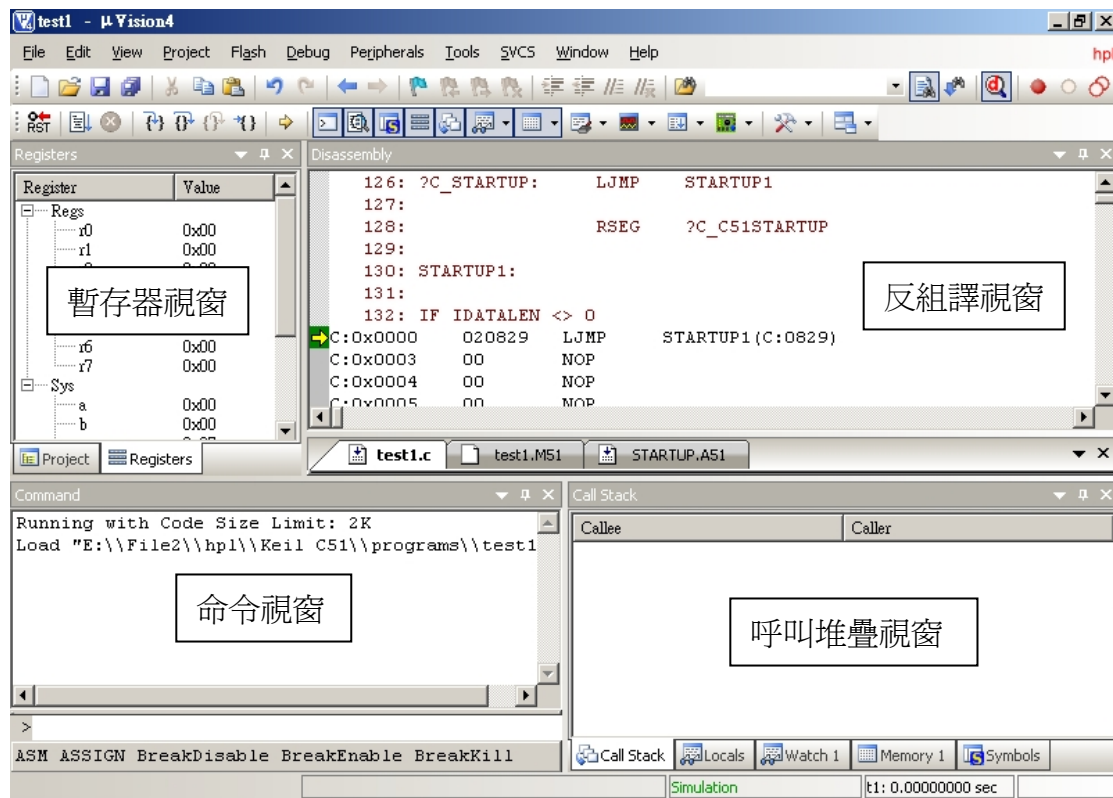


圖 1-19 程式除錯畫面

爲了要檢視輸出結果是否正確，則必須叫出P0輸出入埠觀察輸出結果。選取選單 Peripherals/I/O-Ports/Port 0，如圖1-20所示。

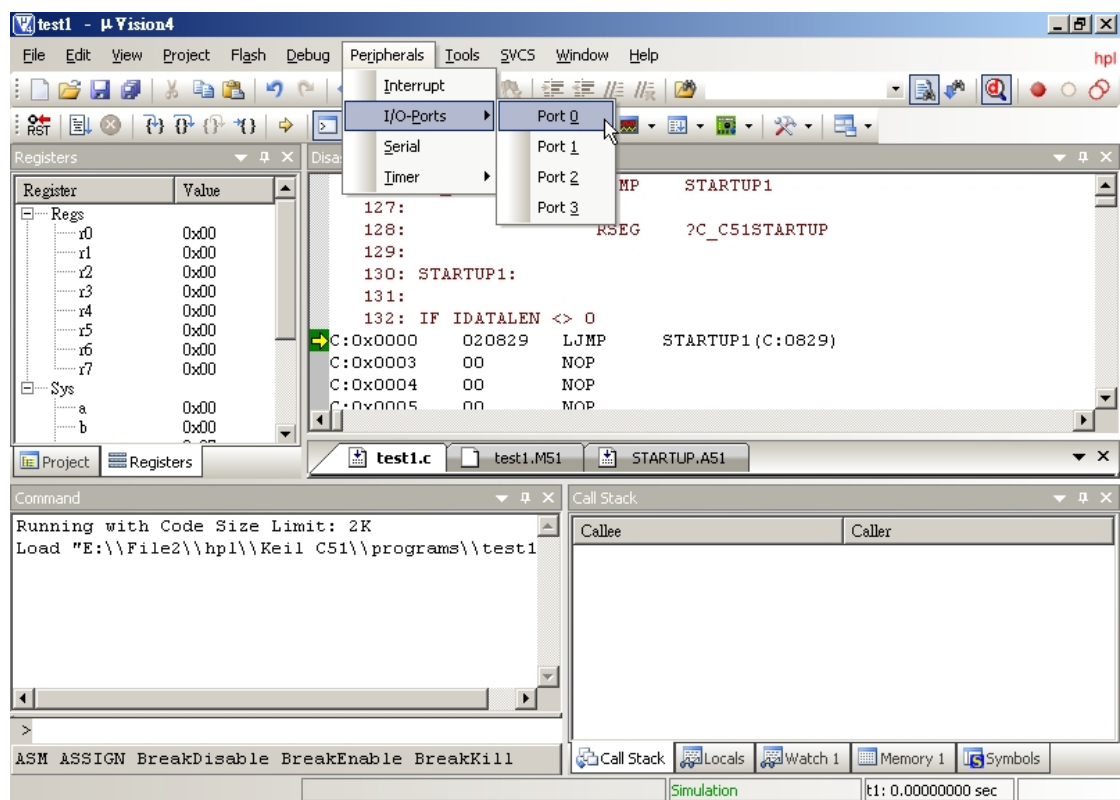


圖 1-20 選取 Port 0

出現 Parallel Port 0 小視窗，並顯示每一個位元的值，也可移動到其他位置觀察，如圖 1-21 所示。

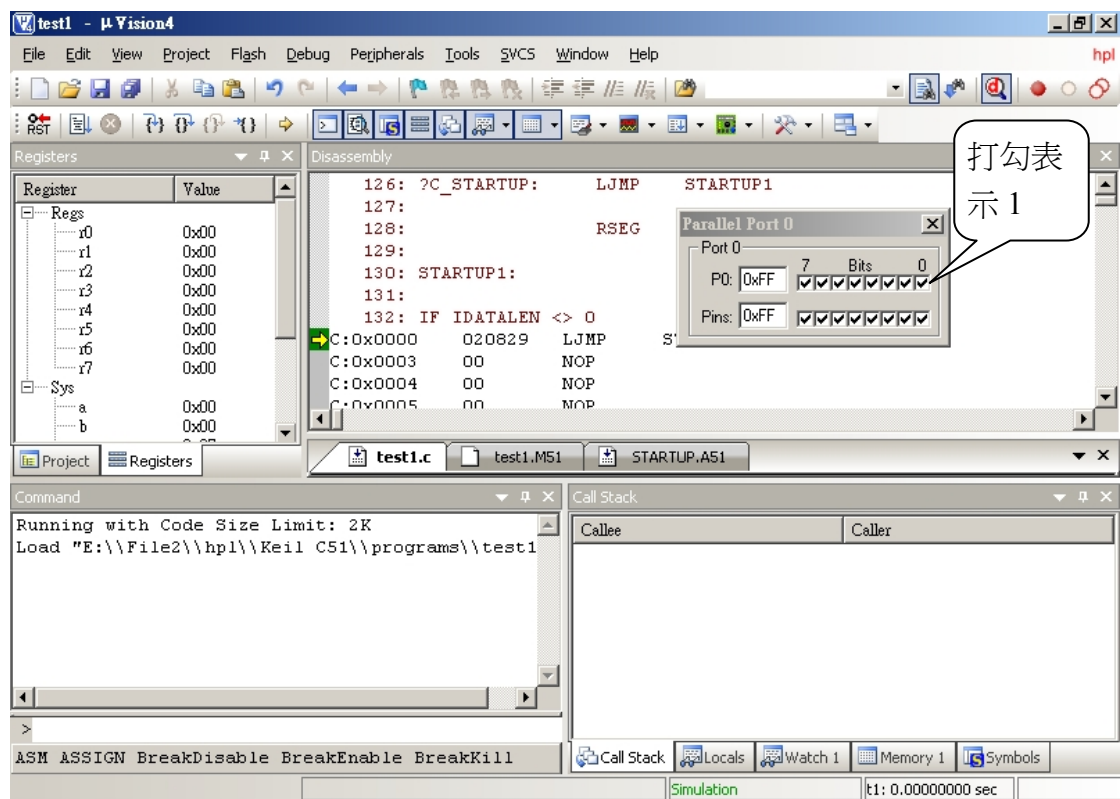




圖 1-21 顯示 Port 0 的視窗

最後要準備執行此程式了，先按一下重置  按鈕，讓單晶片及程式回到最初狀態，再按下執行  按鈕後，則程式開始執行。我們可以看到 Parallel Port 0 視窗中的 P0_0 位元不斷的被設定與清除。

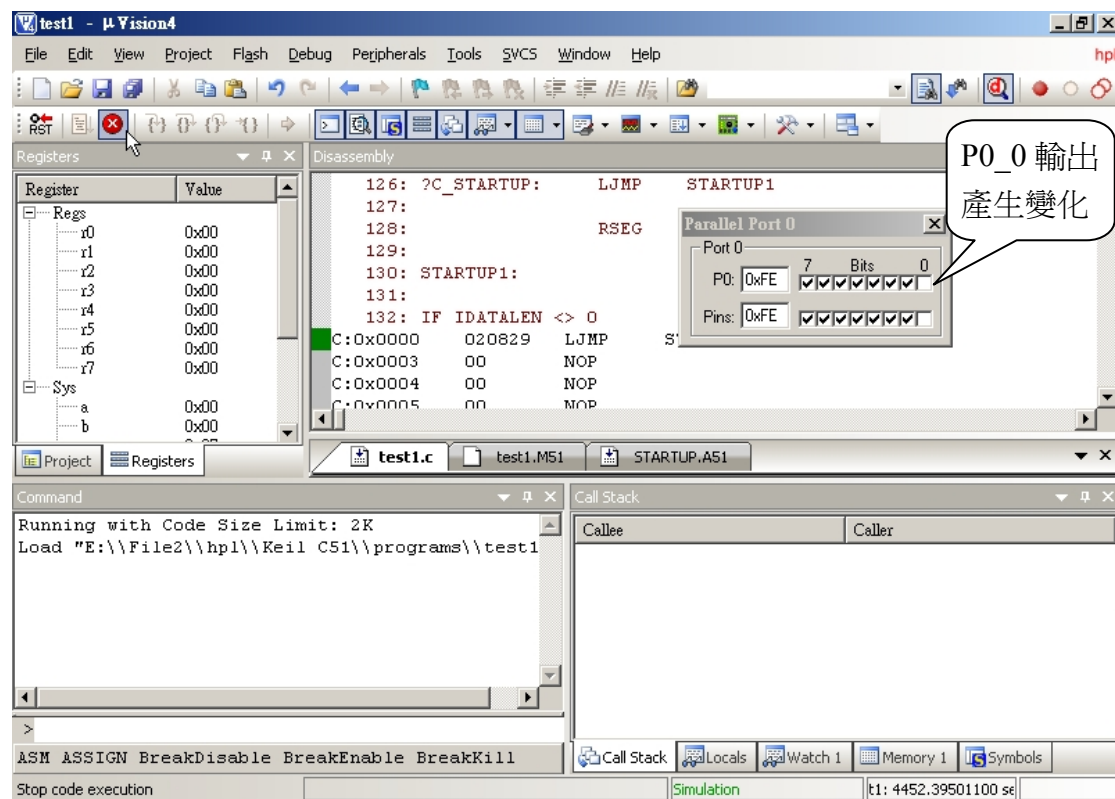


圖 1-22 顯示 Port 0 的內容及停止程式執行

若要觀察組合語言程式編譯後所產生的運算碼，在 ROM 的存放情形，則選取選單 View/Memory Windows/Memory 1，或按右下方的 Memory 1 按鈕，然後在 Address 欄位內輸入 0x800 或 0800h，則 Memory 1 視窗從 0x0800 開始顯示運算碼，如圖 1-23 所示。

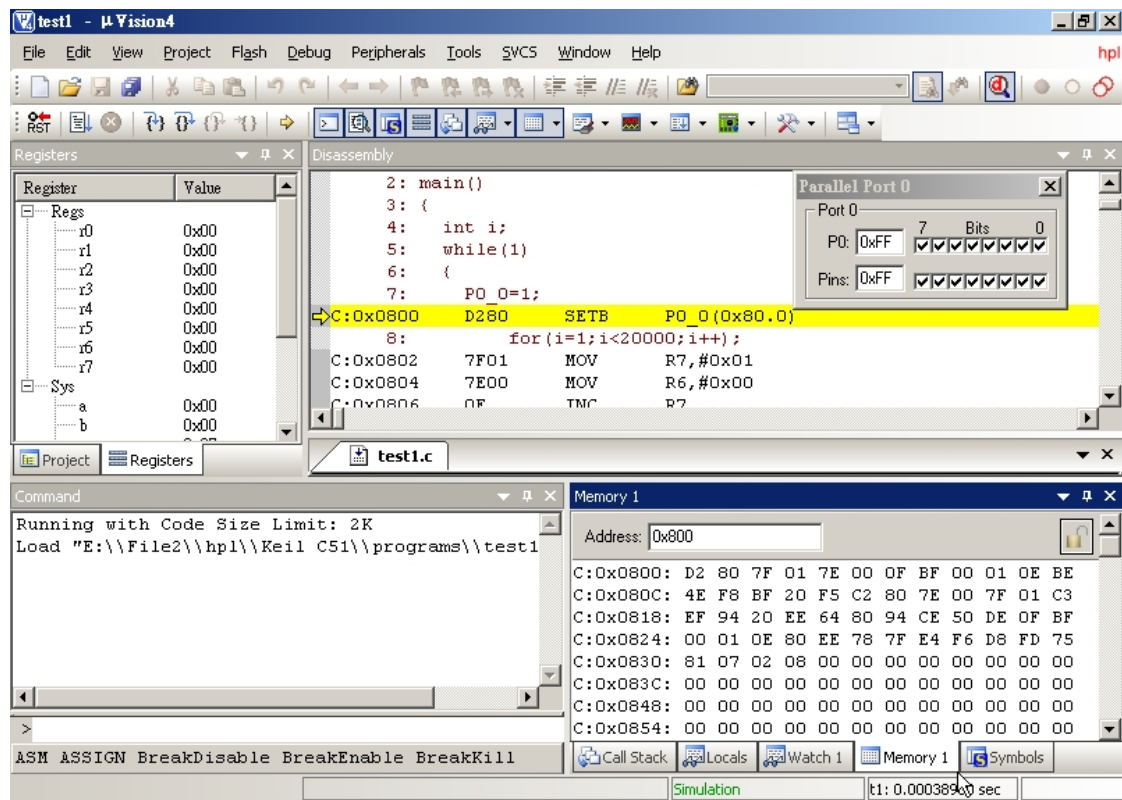




圖 1-23 Memory 1 視窗顯示運算碼

程式執行正確後，最後要停止程式執行回到檔案編輯模式中，就要先按停止  按鈕，再按開啓/關閉除錯模式  按鈕。若要關閉此專案，則選取選單 Project/Close Project，關閉此專案。Keil C51 μVision4 的 C 語言程式初次使用，到此告一段落，下一章我們看 Keil C51 μVision4 的組合語言程式的初次使用。

第二章 建立第二個 Keil C51 程式-使用組合語言

雖然以 C 語言或以組合語言撰寫 MCS-51 程式來相比，C 語言在功能上、架構性、可讀性、可維護性上有明顯的優勢，因而易學易用。然而組合語言所寫出來的程式經編譯(compiler)後，所產生出來的運算碼或機械語言碼所佔用的記憶體，會比 C 語言所寫出來的程式經編譯後，所產生出來的運算碼所佔用的記憶體少，因此執行效率較高。C 語言要經過編譯後轉成組合語言，而轉出來的組合語言的內容的寫法有時後讓人覺得有劃蛇添足的感覺。然而對程式如果要求要寫得簡單、直接、有效率的話，通常是會選擇組合語言。一般而言，對讀電子電機或資工系的學生，最好是 C 語言和組合語言都要學。

接著下面就讓我們一起來建立自己的第二個單晶片組合語言程式吧。若您對上一章的操作已經有了初步認識的話，那你對下面的步驟，就會覺得更容易。

(1) 點擊 Project(專案)選單，選擇彈出的下拉式選單中的“New μ Vision Project...”，如圖 2-1。接著彈出一個標準 Windows 檔案對話視窗，如圖 2-2。在“儲存於”中選擇您要存放的資料夾，一個專案最好存在一個資料夾內，若此資料夾不存在，請先建立它，或按“建立新資料夾”按鈕以建立新資料夾。在“檔名”中輸入您的第二個專案名稱，這裡我們用“test2”。“存檔類型”為 uvproj，這是 Keil μ Vision4 專案檔案預設的副檔名，以後只要直接點擊此專案檔案，即可打開此專案。

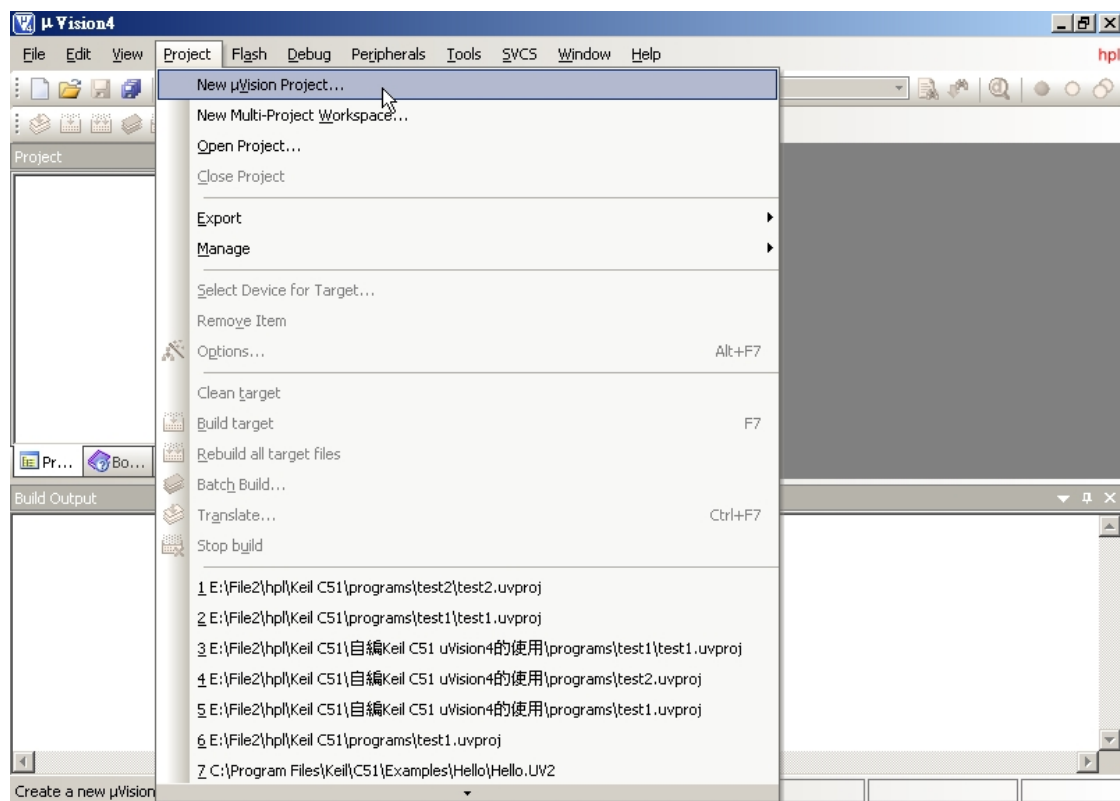


圖 2-1 New μ Vision Project 選單

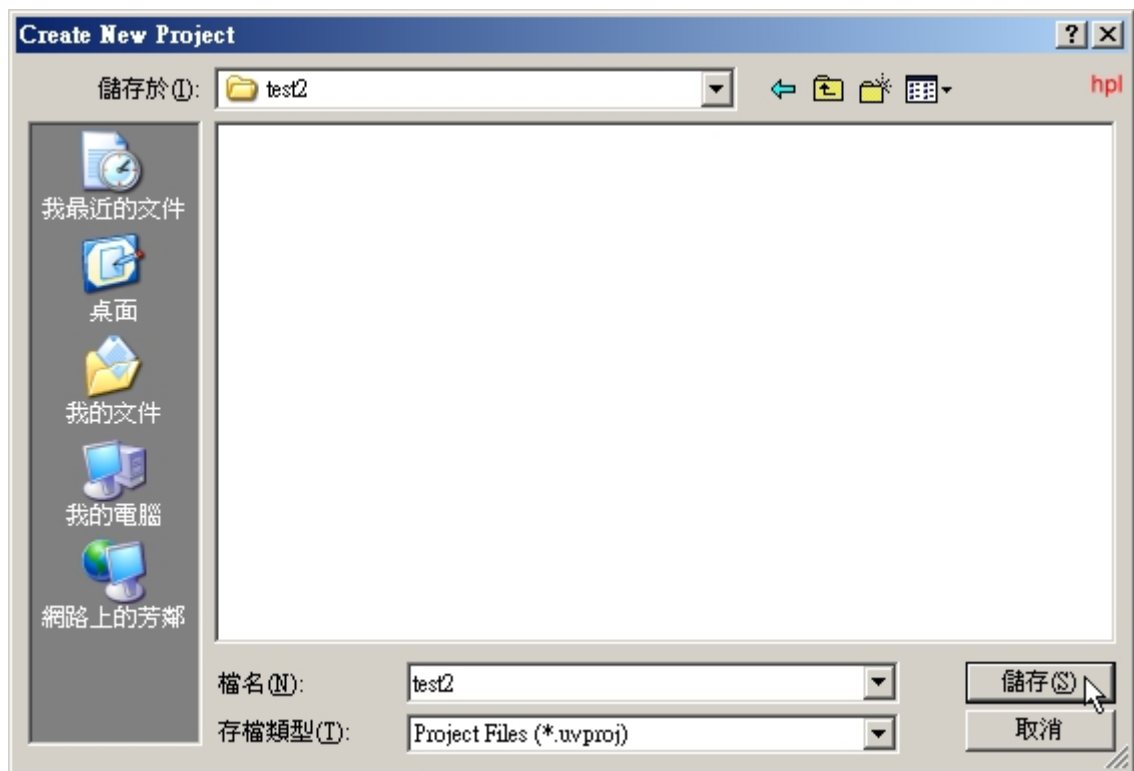


圖 2-2 New μ Vision Project 選單

因第二次使用，所以不會出現圖 1-5 的畫面。

(2) 選擇所要的單晶片型號，這裡仍然選擇常用的 Atmel 公司的 AT89S51，此時螢幕如圖 2-3 所示。在右邊圖中的“Description”方塊內，會簡單的介紹 AT89S51 有什麼功能及特點。點選 OK 按鈕後，會出現圖 2-4，詢問你是否需要拷貝標準的 8051 啟動碼程式到你的專案資料夾，並且將此檔案加入專案“Copy Standard 8051 Startup Code to Project Folder and Add File to Project”，因為組合語言不需要加入此 STARTUP.a51，點選“否”後，就可以進行程式的編寫了。

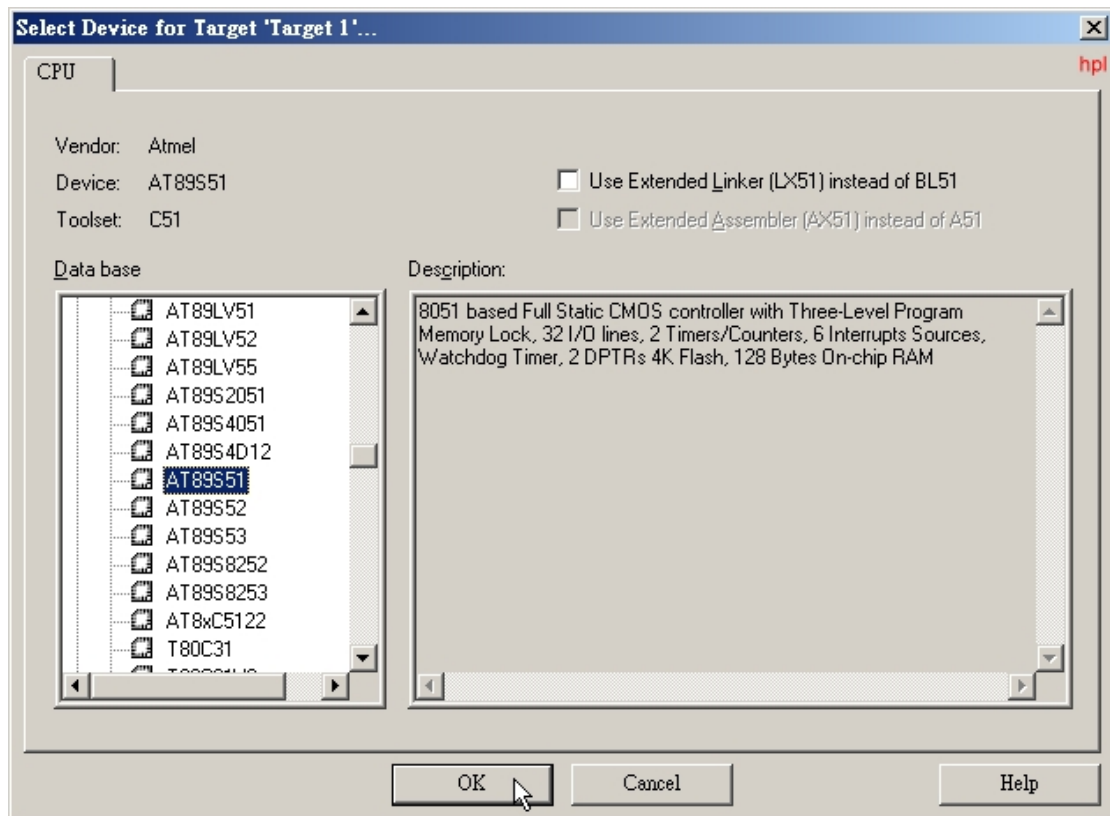


圖 2-3 選取晶片型號

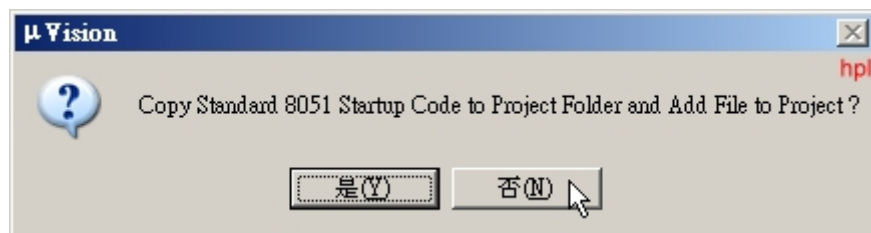


圖 2-4 詢問是否需要加入 8051 啟動碼

(3) 首先在專案中建立新的程式檔案或加入舊程式檔案。如果您沒有現成的程式或是第一次使用組合語言，那麼就要新建一個空白程式檔案。在 C51 中有一些程式的範例，但是在這裡我們還是以一個組程式為例介紹如何新建一個組程式，和如何加到您的專案中吧。點擊圖 2-5 中 1 的新建文件的圖示按鈕，在 2 中出現一個新的文字編輯視窗，或是也能透過選單 File/New 或是按下快速鍵 Ctrl+n 來實現。接著現在就能編寫程式了。

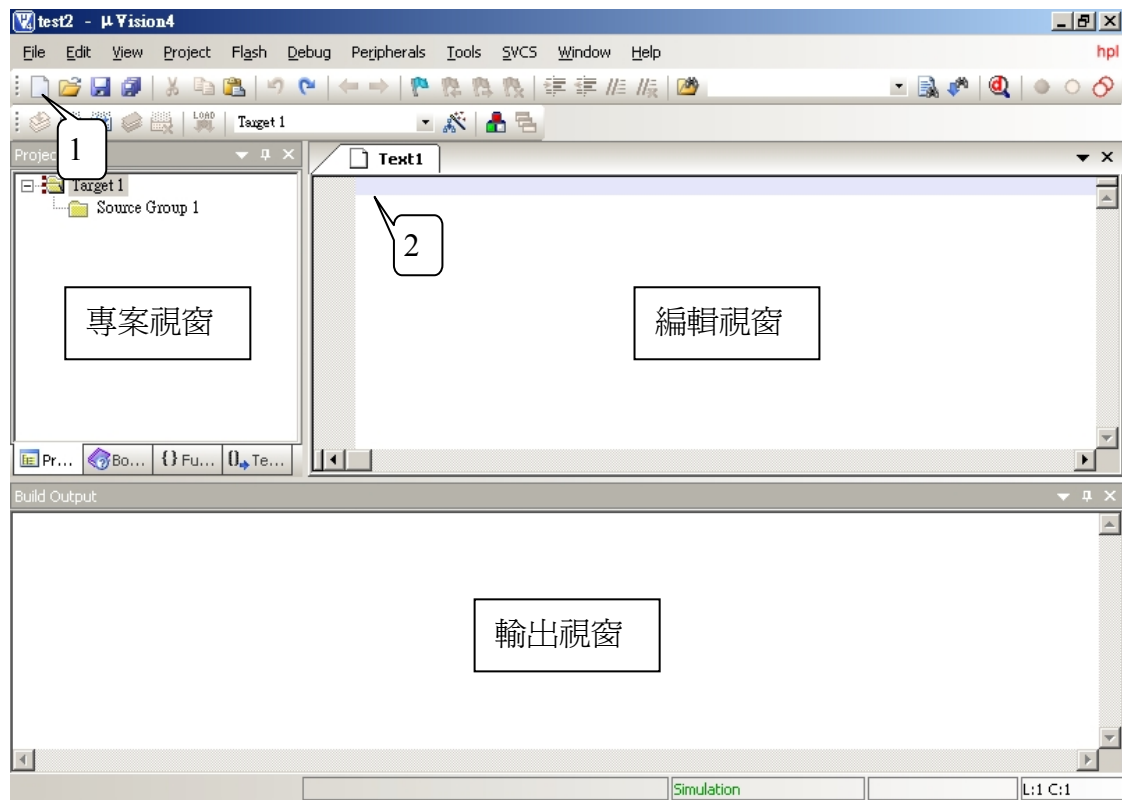


圖 2-5 新建文件

下面是一個跑馬燈的程式，在上圖 2 的文件編輯工作區內鍵入下面的程式，先不管程式的語法和意思，先看看如何把它存檔，加入到專案中存檔，和如何編譯及執行。

```

MOV      A,#0FEH
START:   MOV      P1,A
         RL       A
         ACALL    DELAY
         AJMP     START
DELAY:   MOV      R0,#248
D1:      MOV      R1,#200
D2:      DJNZ     R1,D2
         DJNZ     R0,D1
         RET
         END

```

(4) 點擊圖 2-6 中的儲存檔案圖示按鈕，也能用選單 File/Save 或按快速鍵 Ctrl+S，則出現圖 2-7 的視窗。把此程式命名為 test2.a51，儲存在專案所在的資料夾中，再按儲存鈕。這個時候您會發現程式單字有了不同的顏色，這表示 Keil 的組合語言語法檢查開始作用了。

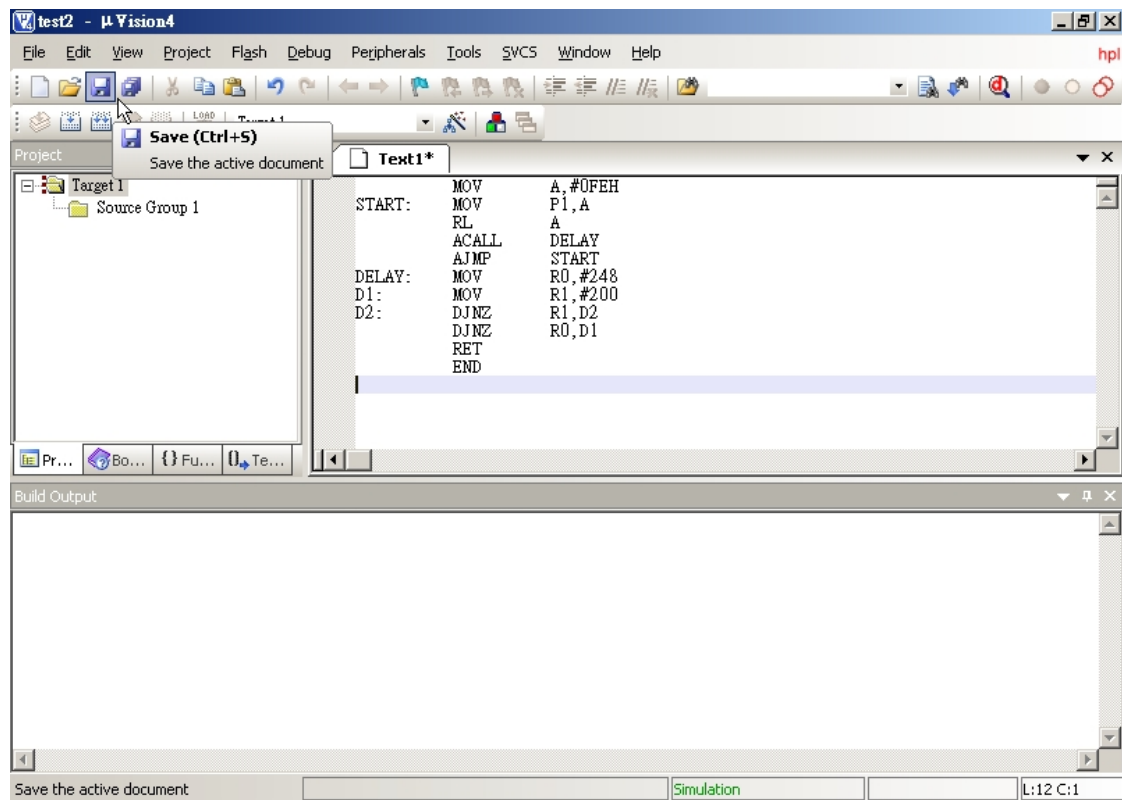


圖 2-6 儲存檔案



圖 2-7 儲存 a51 檔案

(5) 滑鼠在螢幕左邊的 Source Group1 資料夾圖示上按右鍵，彈出一選單，如圖 2-8 所示，在這裡能做出專案中增加減少檔案等操作。選 “Add Files to Group ‘Source Group 1...’” 彈出檔案視窗，選擇剛剛儲存的檔案，按下 Add 按鈕，將此.a51 檔案加入

到此專案中。按下 close 按鈕，關閉檔案視窗，如圖 2-9 所示，則此 test2.a51 程式檔案已加到此專案中了，如圖 2-10 所示。

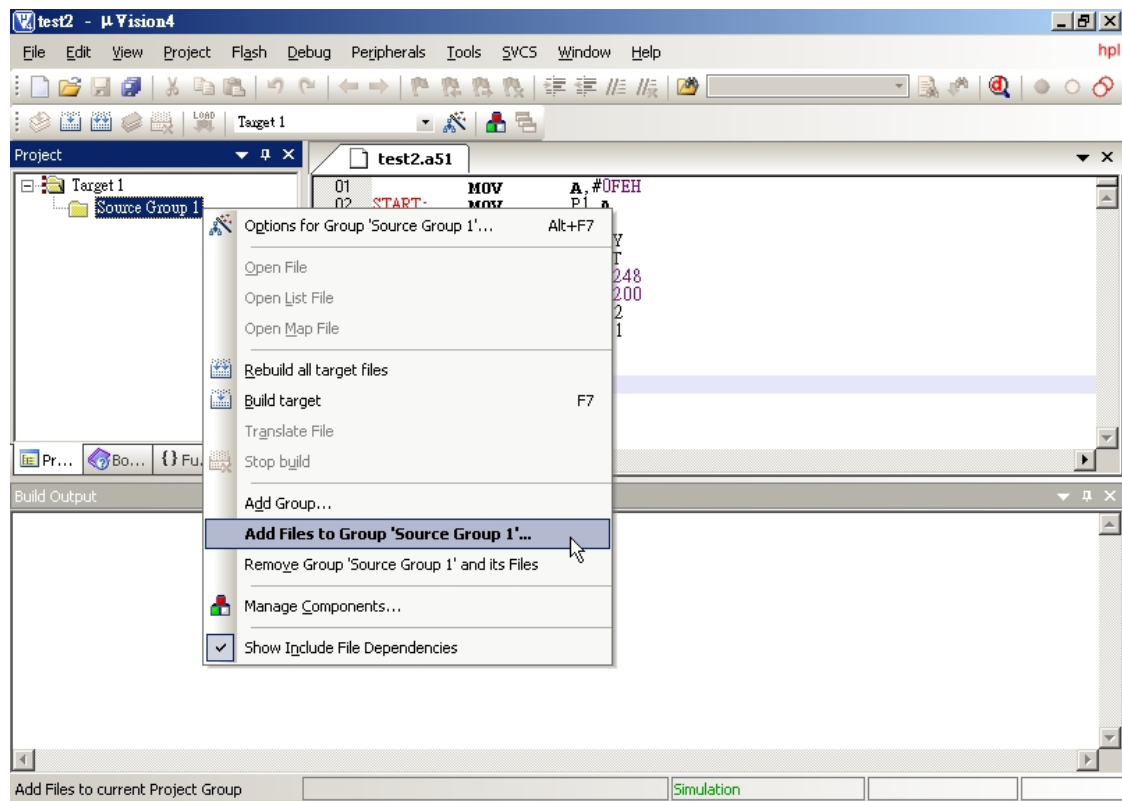


圖 2-8 選取 “Add Files to Group ‘Source Group 1’...”



圖 2-9 選取要加入到專案中的組合語言檔案

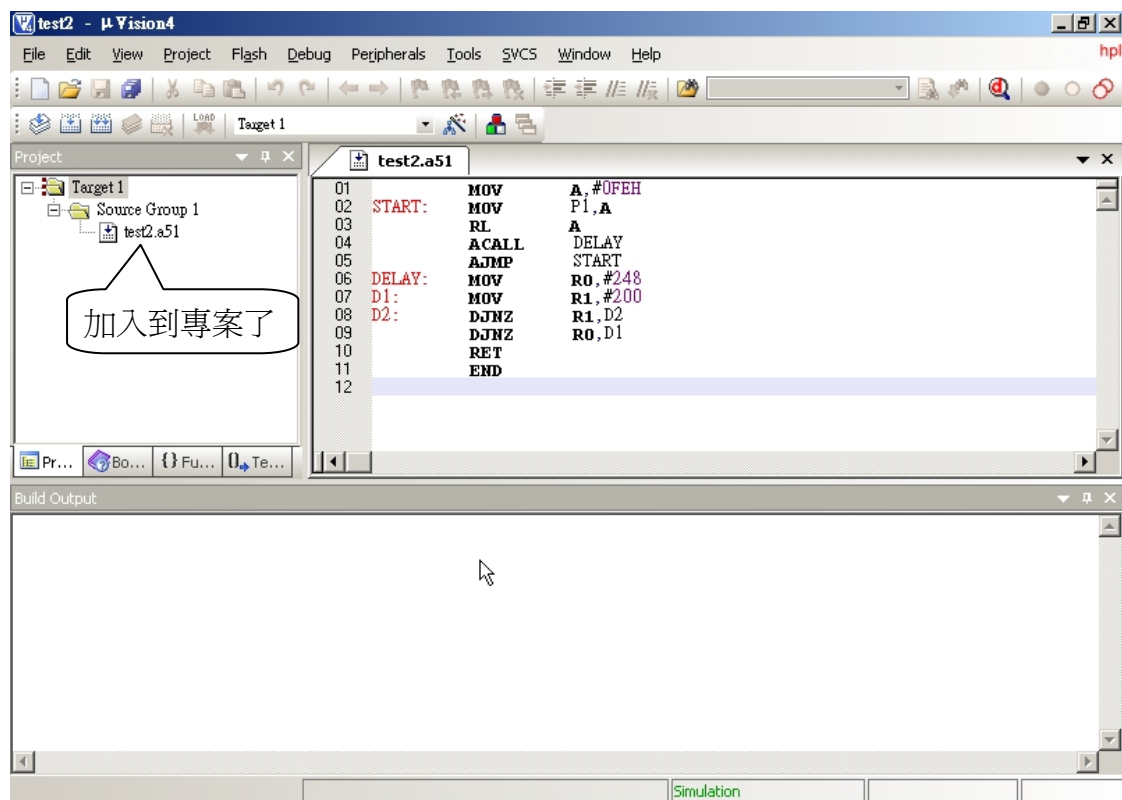


圖 2-10 把組合語言檔案加入到專案中了

(6) 接下來要來做一些基本的設定選項的工作，在每一個專案裡面都要設定一次。滑鼠在螢幕左邊的 Target 1 資料夾圖示上按右鍵，彈出一選單，如圖 2-11 所示，然後選取“Options for target ‘Target 1’...”，則出現圖 2-12 的視窗。

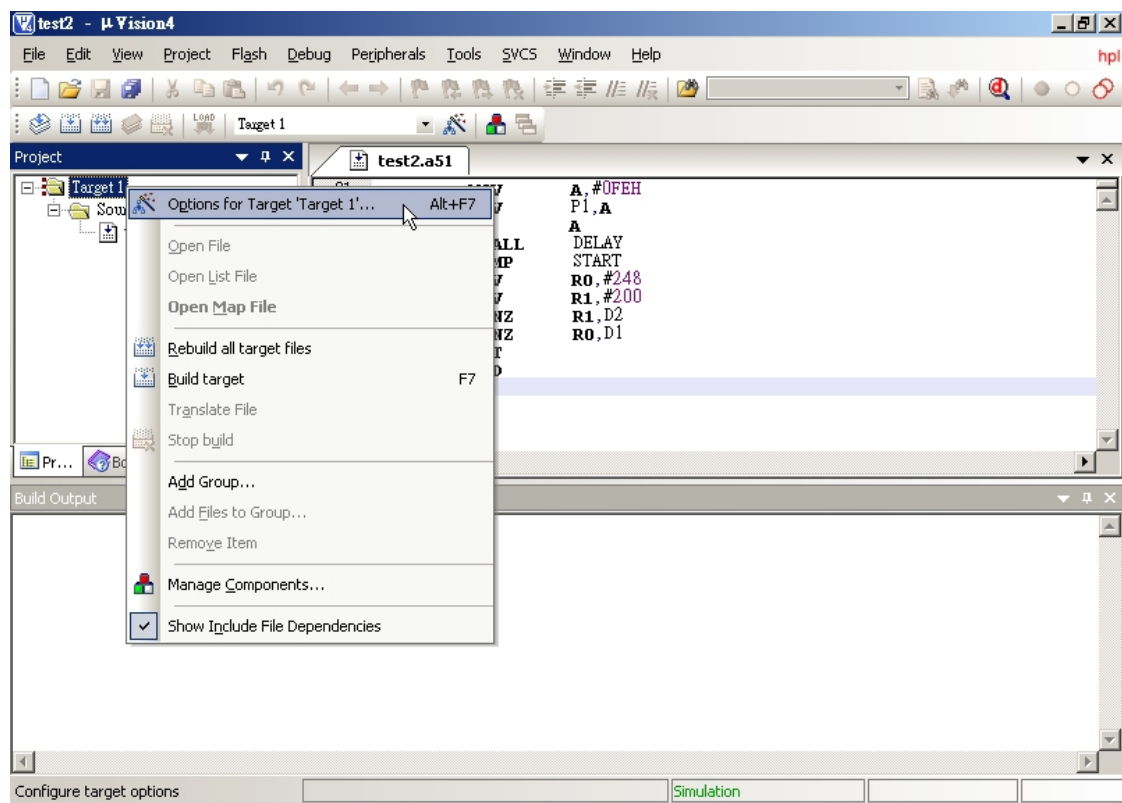


圖 2-11 選取“Options for target ‘Target 1’...”

在圖 2-12 的 Target 標籤頁中，如同圖 1-15，更改所選用單晶片的工作頻率為 12，並勾選 Use On-Chip ROM(0x0-0xFF)，以使用單晶片上的 Flash ROM。

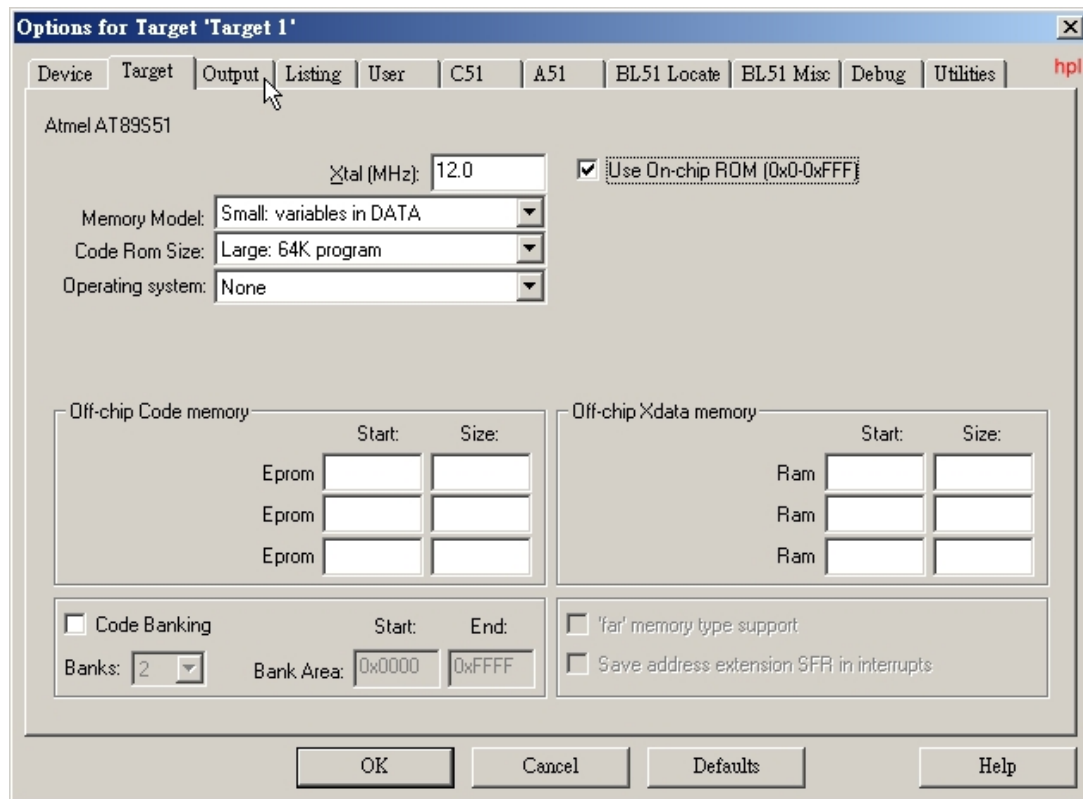


圖 2-12 在 Target 標籤頁中，更改所選用單晶片的工作頻率

再來切換到 Output 標籤頁，只要勾選“Create HEX File”就好了，以產生燒錄檔，如圖 2-13 所示。如果用戶只是單純的做練習，那就省略此步驟了。若要更改存放目的檔的資料夾，則點擊“Select Folder for Objects...”，若要更改編譯後的主檔名，則在“Name of Executable:”右邊的空格內輸入主檔名即可，一般而言，這 2 個選項都採用預設值，用戶不需更改他們。

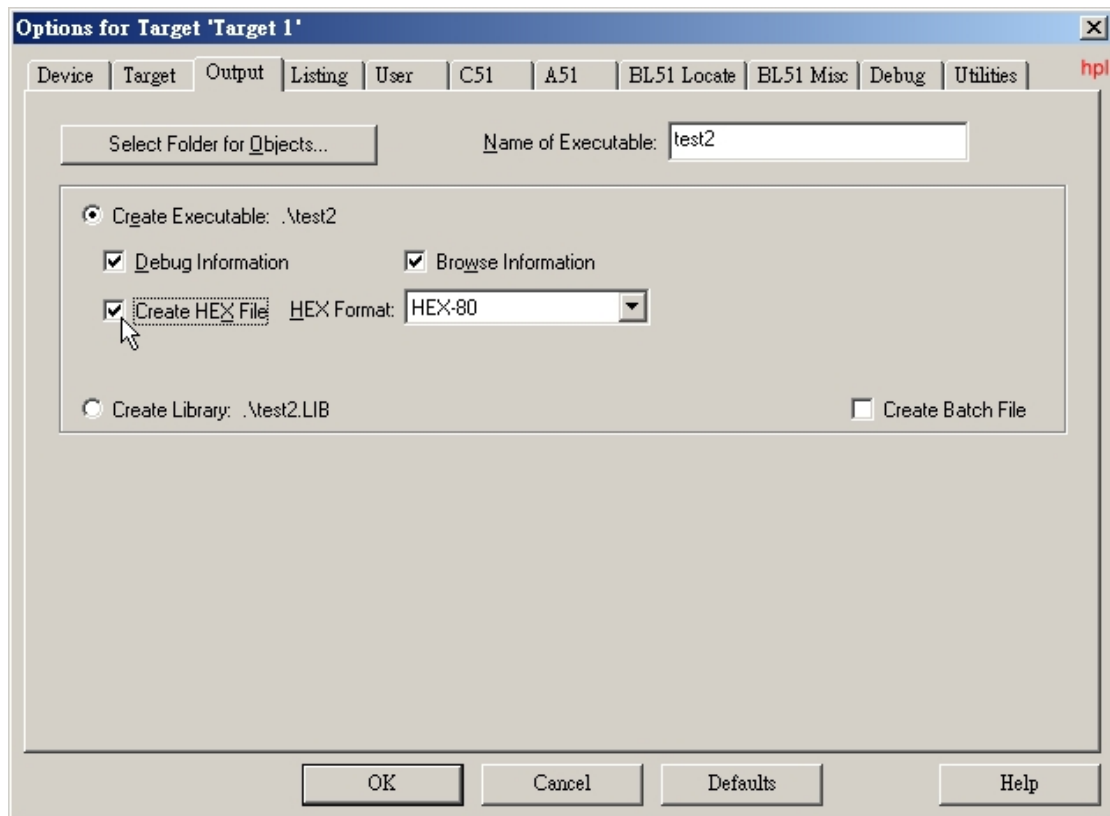



圖 2-13 勾選“Create HEX File”，以產生燒錄檔

(7) 完成基本的選項設定後，下面就剩下編譯執行了。在此按下  按鈕，“Build target”，或 F7 快捷鍵，如圖 2-14 所示。

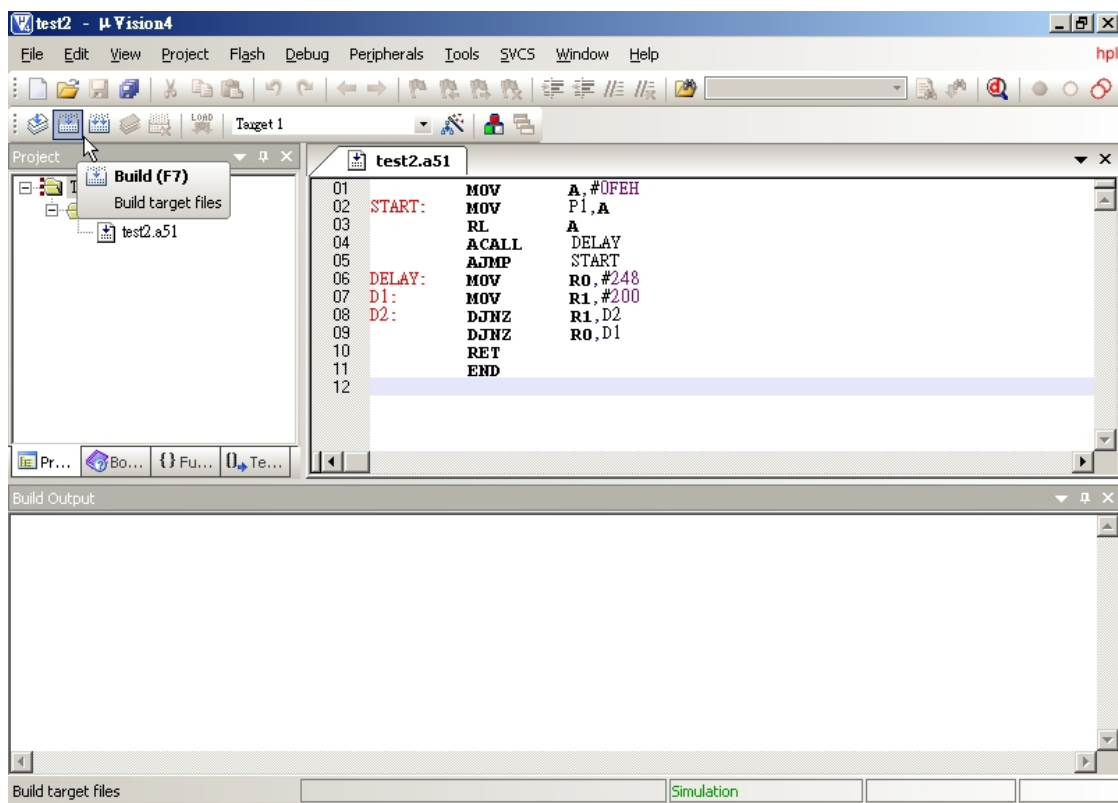


圖 2-14 編譯專案

編譯完成後，在下方的 Build Output 區域中，可看到編譯的訊息，如圖 2-15 所示。若有出現錯誤訊息，則再根據錯誤訊息，回到程式中修改，編譯完全正確後，才能產生正確的燒錄檔 test2.hex。

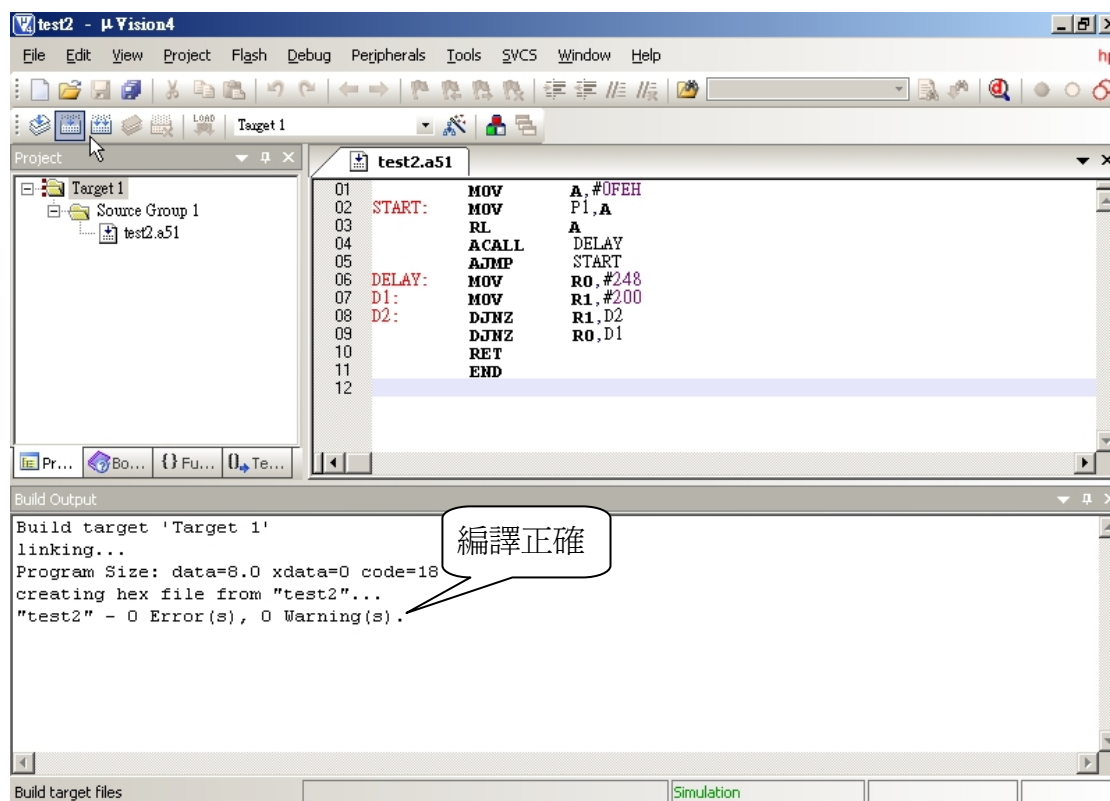










圖 2-15 專案編譯正確

(8) 選取選單 Debug/Start/Stop Debug Session，或按快速鍵 Ctrl+F5，或按在檔案工具列右邊有一個小放大鏡的按鈕，則進入除錯(Debug)模式，並顯示不同的工作視窗，如圖 2-16 所示。進入除錯模式之前，同樣地會先出現一個小視窗，告訴你目前用的版本是免費的評估版，有 2K ROM 大小的限制，點擊確定即可進入除錯模式。在除錯工具列中， (Reset) 按鈕表示重置單晶片，並使程式回到最開頭處執行。 (Run) 按鈕表示執行， (Stop) 按鈕表示停止，當程式處於執行狀態時，停止按鈕才有效。 (Step Into) 按鈕表示單步執行會進入函數內， (Step Over) 按鈕表示單步執行不會進入函數內， (Step Out) 按鈕表示離開函數， (Run to Cursor) 按鈕表示執行到游標所在處。

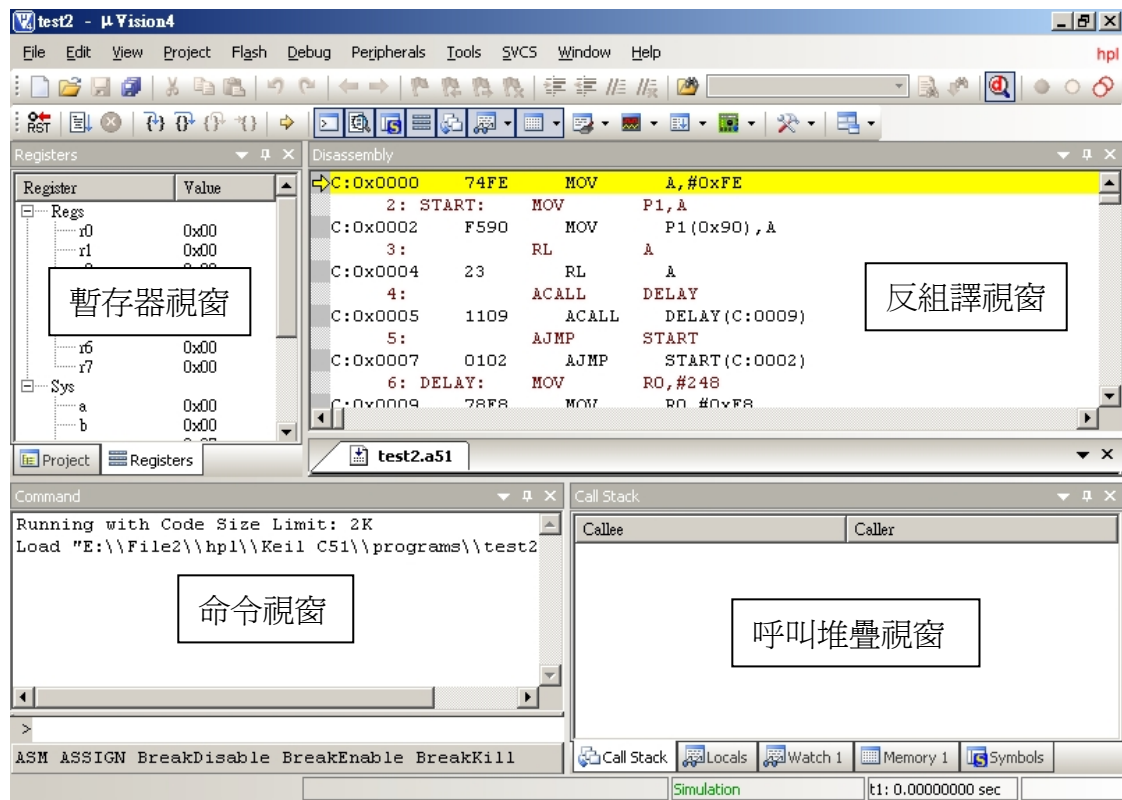


圖 2-16 程式除錯畫面

爲了要檢視輸出結果是否正確，則必須叫出 P1 輸出入埠觀察輸出結果。選取選單 Peripherals/I/O-Ports/Port 1，如圖 2-17 所示。

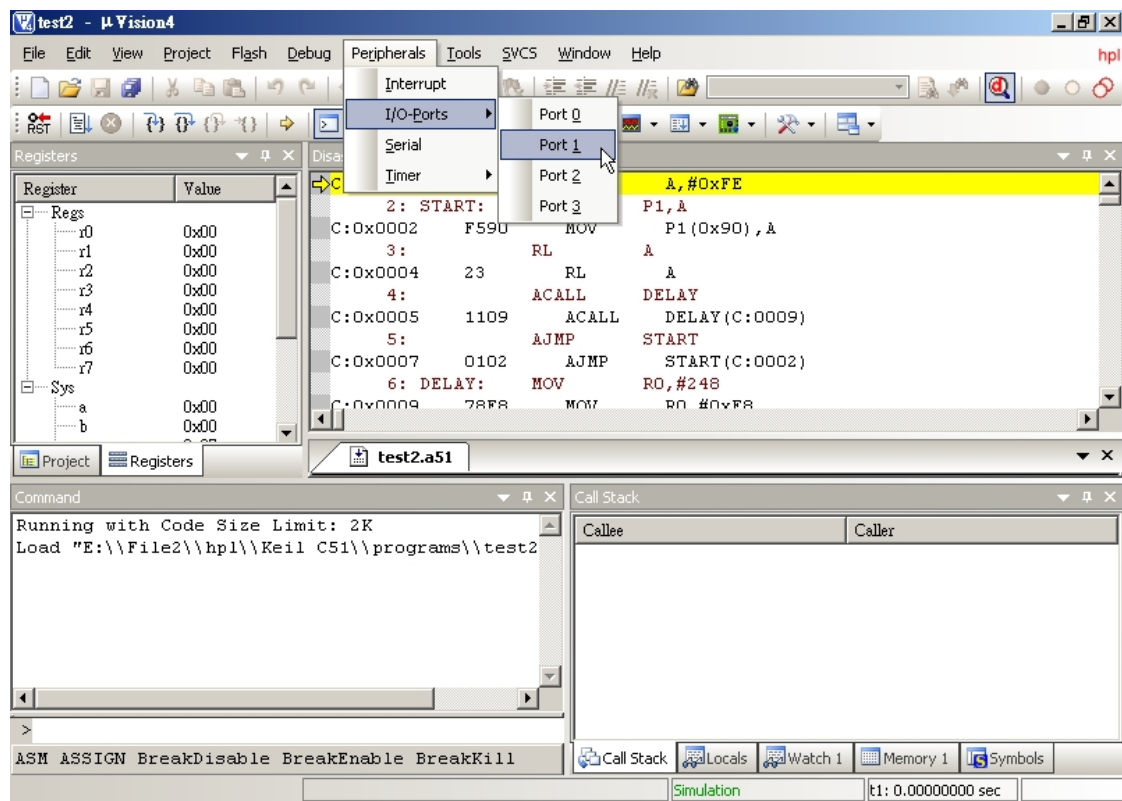


圖 2-17 選取 Port 1

出現 Parallel Port 1 小視窗，並顯示每一個位元的值，也可移動到其他位置觀察，如圖 2-18 所示。

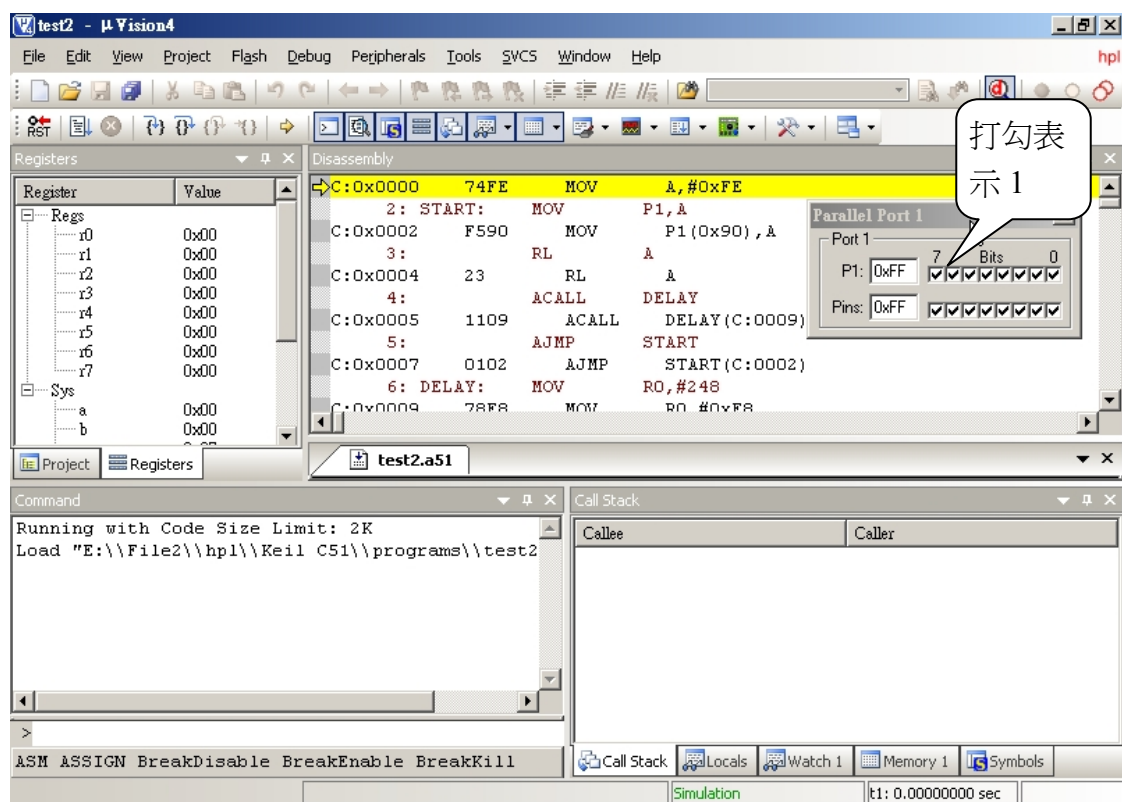


圖 2-18 顯示 Port 1 的視窗





最後要準備執行此程式了，先按一下重置  按鈕，讓單晶片及程式回到最初狀態，再按下執行  按鈕後，則程式開始執行。我們可以看到 Parallel Port 1 視窗中的 P1_0 位元到 P1_7 位元不斷的被設定與清除，如圖 2-19 所示。

圖 2-20 Memory 1 視窗顯示運算碼

程式執行正確後，最後要停止程式執行回到檔案編輯模式中，就要先按停止  按鈕再按開啓\關閉除錯模式  按鈕。若要關閉此專案，則選取選單 Project/Close Project，關閉此專案。Keil C51 μ Vision4 的組合語言程式使用，到此告一段落。其他進一步的使用，請參考相關書籍或 Keil C51 的 Help 說明。

附錄 1

STARTUP.a51

\$NOMOD51

```
;-----  
;  
; This file is part of the C51 Compiler package  
; Copyright (c) 1988-2005 Keil Elektronik GmbH and Keil Software, Inc.  
; Version 8.01  
;  
; *** <<< Use Configuration Wizard in Context Menu >>> ***  
;-----  
;  
; STARTUP.A51: This code is executed after processor reset.  
;  
;  
; To translate this file use A51 with the following invocation:  
;  
;  
; A51 STARTUP.A51  
;  
;  
; To link the modified STARTUP.OBJ file to your application use the following  
; Lx51 invocation:  
;  
;  
; Lx51 your object file list, STARTUP.OBJ controls  
;  
;-----  
;  
;  
; User-defined <h> Power-On Initialization of Memory  
;  
;  
; With the following EQU statements the initialization of memory  
; at processor reset can be defined:  
;  
;  
; <o> IDATALEN: IDATA memory size <0x0-0x100>  
; <i> Note: The absolute start-address of IDATA memory is always 0  
; <i> The IDATA space overlaps physically the DATA and BIT areas.  
IDATALEN EQU 80H  
;  
;  
; <o> XDATASTART: XDATA memory start address <0x0-0xFFFF>  
; <i> The absolute start address of XDATA memory  
XDATASTART EQU 0  
;  
;  
; <o> XDATALEN: XDATA memory size <0x0-0xFFFF>  
; <i> The length of XDATA memory in bytes.
```

```

XDATALEN      EQU      0
;
; <o> PDATASTART: PDATA memory start address <0x0-0xFFFF>
;   <i> The absolute start address of PDATA memory
PDATASTART    EQU      0H
;
; <o> PDATALEN: PDATA memory size <0x0-0xFF>
;   <i> The length of PDATA memory in bytes.
PDATALEN      EQU      0H
;
; </h>
;-----
;
; <h> Reentrant Stack Initialization
;
; The following EQU statements define the stack pointer for reentrant
; functions and initialized it:
;
; <h> Stack Space for reentrant functions in the SMALL model.
; <q> IBPSTACK: Enable SMALL model reentrant stack
;   <i> Stack space for reentrant functions in the SMALL model.
IBPSTACK      EQU      0          ; set to 1 if small reentrant is used.
; <o> IBPSTACKTOP: End address of SMALL model stack <0x0-0xFF>
;   <i> Set the top of the stack to the highest location.
IBPSTACKTOP   EQU      0xFF + 1   ; default 0FFH+1
; </h>
;
; <h> Stack Space for reentrant functions in the LARGE model.
; <q> XBPSTACK: Enable LARGE model reentrant stack
;   <i> Stack space for reentrant functions in the LARGE model.
XBPSTACK      EQU      0          ; set to 1 if large reentrant is used.
; <o> XBPSTACKTOP: End address of LARGE model stack <0x0-0xFFFF>
;   <i> Set the top of the stack to the highest location.
XBPSTACKTOP   EQU      0xFFFF + 1 ; default 0FFFFH+1
; </h>
;
; <h> Stack Space for reentrant functions in the COMPACT model.
; <q> PBPSTACK: Enable COMPACT model reentrant stack
;   <i> Stack space for reentrant functions in the COMPACT model.

```

PBPSTACK EQU 0 ; set to 1 if compact reentrant is used.

;

; <o> PBPSTACKTOP: End address of COMPACT model stack <0x0-0xFFFF>

;

<i> Set the top of the stack to the highest location.

PBPSTACKTOP EQU 0xFF +1 ; default 0FFH+1

; </h>

; </h>

;-----

;

; Memory Page for Using the Compact Model with 64 KByte xdata RAM

; <e>Compact Model Page Definition

;

; <i>Define the XDATA page used for PDATA variables.

; <i>PPAGE must conform with the PPAGE set in the linker invocation.

;

; Enable pdata memory page initialization

PPAGEENABLE EQU 0 ; set to 1 if pdata object are used.

;

; <o> PPAGE number <0x0-0xFF>

; <i> uppermost 256-byte address of the page used for PDATA variables.

PPAGE EQU 0

;

; <o> SFR address which supplies uppermost address byte <0x0-0xFF>

; <i> most 8051 variants use P2 as uppermost address byte

PPAGE_SFR DATA 0A0H

;

; </e>

;-----

; Standard SFR Symbols

ACC DATA 0E0H

B DATA 0F0H

SP DATA 81H

DPL DATA 82H

DPH DATA 83H

NAME ?C_STARTUP

?C_C51STARTUP SEGMENT CODE

?STACK SEGMENT IDATA

 RSEG ?STACK

 DS 1

 EXTRN CODE (?C_START)

 PUBLIC ?C_STARTUP

 CSEG AT 0

?C_STARTUP: LJMP STARTUP1

 RSEG ?C_C51STARTUP

STARTUP1:

IF IDATALEN <> 0

 MOV R0,#IDATALEN - 1

 CLR A

IDATALOOP: MOV @R0,A

 DJNZ R0,IDATALOOP

ENDIF

IF XDATALEN <> 0

 MOV DPTR,#XDATASTART

 MOV R7,#LOW (XDATALEN)

 IF (LOW (XDATALEN)) <> 0

 MOV R6,#(HIGH (XDATALEN)) +1

 ELSE

 MOV R6,#HIGH (XDATALEN)

 ENDIF

 CLR A

XDATALOOP: MOVX @DPTR,A

 INC DPTR

 DJNZ R7,XDATALOOP

 DJNZ R6,XDATALOOP

ENDIF

IF PPAGEENABLE <> 0


```

                MOV        PPAGE_SFR,#PPAGE
ENDIF

```

```

IF PDATALEN <> 0
                MOV        R0,#LOW (PDATASTART)
                MOV        R7,#LOW (PDATALEN)
                CLR        A
PDATALOOP:     MOVX        @R0,A
                INC        R0
                DJNZ       R7,PDATALOOP
ENDIF

```

```

IF IBPSTACK <> 0
EXTRN DATA (?C_IBP)

                MOV        ?C_IBP,#LOW IBPSTACKTOP
ENDIF

```

```

IF XBPSTACK <> 0
EXTRN DATA (?C_XBP)

                MOV        ?C_XBP,#HIGH XBPSTACKTOP
                MOV        ?C_XBP+1,#LOW XBPSTACKTOP
ENDIF

```

```

IF PBPSTACK <> 0
EXTRN DATA (?C_PBP)

                MOV        ?C_PBP,#LOW PBPSTACKTOP
ENDIF

```

```

                MOV        SP,#?STACK-1

```

; This code is required if you use L51_BANK.A51 with Banking Mode 4

; <h> Code Banking

; <q> Select Bank 0 for L51_BANK.A51 Mode 4

#if 0

; <i> Initialize bank mechanism to code bank 0 when using L51_BANK.A51 with Banking Mode 4.

EXTRN CODE (?B_SWITCH0)

```
CALL    ?B_SWITCH0    ; init bank mechanism to code bank 0
#endif
;=</h>
LJMP    ?C_START

END
```

附錄 2

AT89X51.H

/*-----

Header file for the low voltage Flash Atmel AT89C51 and AT89LV51.

Copyright (c) 1988-2002 Keil Elektronik GmbH and Keil Software, Inc.

All rights reserved.

-----*/

#ifndef __AT89X51_H__

#define __AT89X51_H__

/*-----

Byte Registers

-----*/

sfr P0 = 0x80;

sfr SP = 0x81;

sfr DPL = 0x82;

sfr DPH = 0x83;

sfr PCON = 0x87;

sfr TCON = 0x88;

sfr TMOD = 0x89;

sfr TL0 = 0x8A;

sfr TL1 = 0x8B;

sfr TH0 = 0x8C;

sfr TH1 = 0x8D;

sfr P1 = 0x90;

sfr SCON = 0x98;

sfr SBUF = 0x99;

sfr P2 = 0xA0;

sfr IE = 0xA8;

sfr P3 = 0xB0;

sfr IP = 0xB8;

sfr PSW = 0xD0;

sfr ACC = 0xE0;

sfr B = 0xF0;

/*-----

P0 Bit Registers

-----*/

```
sbit P0_0 = 0x80;
sbit P0_1 = 0x81;
sbit P0_2 = 0x82;
sbit P0_3 = 0x83;
sbit P0_4 = 0x84;
sbit P0_5 = 0x85;
sbit P0_6 = 0x86;
sbit P0_7 = 0x87;
```

```
/*-----
```

```
PCON Bit Values
```

```
-----*/
```

```
#define IDL_      0x01
```

```
#define STOP_     0x02
```

```
#define PD_       0x02    /* Alternate definition */
```

```
#define GF0_      0x04
```

```
#define GF1_      0x08
```

```
#define SMOD_     0x80
```

```
/*-----
```

```
TCON Bit Registers
```

```
-----*/
```

```
sbit IT0  = 0x88;
```

```
sbit IE0  = 0x89;
```

```
sbit IT1  = 0x8A;
```

```
sbit IE1  = 0x8B;
```

```
sbit TR0  = 0x8C;
```

```
sbit TF0  = 0x8D;
```

```
sbit TR1  = 0x8E;
```

```
sbit TF1  = 0x8F;
```

```
/*-----
```

```
TMOD Bit Values
```

```
-----*/
```

```
#define T0_M0_    0x01
```

```
#define T0_M1_    0x02
```

```
#define T0_CT_    0x04
#define T0_GATE_ 0x08
#define T1_M0_    0x10
#define T1_M1_    0x20
#define T1_CT_    0x40
#define T1_GATE_ 0x80
```

```
#define T1_MASK_ 0xF0
#define T0_MASK_ 0x0F
```

```
/*-----
```

```
P1 Bit Registers
```

```
-----*/
```

```
sbit P1_0 = 0x90;
sbit P1_1 = 0x91;
sbit P1_2 = 0x92;
sbit P1_3 = 0x93;
sbit P1_4 = 0x94;
sbit P1_5 = 0x95;
sbit P1_6 = 0x96;
sbit P1_7 = 0x97;
```

```
/*-----
```

```
SCON Bit Registers
```

```
-----*/
```

```
sbit RI    = 0x98;
sbit TI    = 0x99;
sbit RB8   = 0x9A;
sbit TB8   = 0x9B;
sbit REN   = 0x9C;
sbit SM2   = 0x9D;
sbit SM1   = 0x9E;
sbit SM0   = 0x9F;
```

```
/*-----
```

```
P2 Bit Registers
```

```
-----*/
```

```
sbit P2_0 = 0xA0;
sbit P2_1 = 0xA1;
```

```
sbit P2_2 = 0xA2;
sbit P2_3 = 0xA3;
sbit P2_4 = 0xA4;
sbit P2_5 = 0xA5;
sbit P2_6 = 0xA6;
sbit P2_7 = 0xA7;
```

```
/*-----
```

IE Bit Registers

```
-----*/
```

```
sbit EX0  = 0xA8;      /* 1=Enable External interrupt 0 */
sbit ET0  = 0xA9;      /* 1=Enable Timer 0 interrupt */
sbit EX1  = 0xAA;      /* 1=Enable External interrupt 1 */
sbit ET1  = 0xAB;      /* 1=Enable Timer 1 interrupt */
sbit ES   = 0xAC;      /* 1=Enable Serial port interrupt */
sbit ET2  = 0xAD;      /* 1=Enable Timer 2 interrupt */
```

```
sbit EA   = 0xAF;      /* 0=Disable all interrupts */
```

```
/*-----
```

P3 Bit Registers (Mnemonics & Ports)

```
-----*/
```

```
sbit P3_0 = 0xB0;
sbit P3_1 = 0xB1;
sbit P3_2 = 0xB2;
sbit P3_3 = 0xB3;
sbit P3_4 = 0xB4;
sbit P3_5 = 0xB5;
sbit P3_6 = 0xB6;
sbit P3_7 = 0xB7;
```

```
sbit RXD  = 0xB0;      /* Serial data input */
sbit TXD  = 0xB1;      /* Serial data output */
sbit INT0 = 0xB2;      /* External interrupt 0 */
sbit INT1 = 0xB3;      /* External interrupt 1 */
sbit T0   = 0xB4;      /* Timer 0 external input */
sbit T1   = 0xB5;      /* Timer 1 external input */
sbit WR   = 0xB6;      /* External data memory write strobe */
sbit RD   = 0xB7;      /* External data memory read strobe */
```

```
/*-----
```

```
IP Bit Registers
```

```
-----*/
```

```
sbit PX0  = 0xB8;
```

```
sbit PT0  = 0xB9;
```

```
sbit PX1  = 0xBA;
```

```
sbit PT1  = 0xBB;
```

```
sbit PS   = 0xBC;
```

```
sbit PT2  = 0xBD;
```

```
/*-----
```

```
PSW Bit Registers
```

```
-----*/
```

```
sbit P     = 0xD0;
```

```
sbit F1    = 0xD1;
```

```
sbit OV    = 0xD2;
```

```
sbit RS0   = 0xD3;
```

```
sbit RS1   = 0xD4;
```

```
sbit F0    = 0xD5;
```

```
sbit AC    = 0xD6;
```

```
sbit CY    = 0xD7;
```

```
/*-----
```

```
Interrupt Vectors:
```

```
Interrupt Address = (Number * 8) + 3
```

```
-----*/
```

```
#define IE0_VECTOR 0 /* 0x03 External Interrupt 0 */
```

```
#define TF0_VECTOR 1 /* 0x0B Timer 0 */
```

```
#define IE1_VECTOR 2 /* 0x13 External Interrupt 1 */
```

```
#define TF1_VECTOR 3 /* 0x1B Timer 1 */
```

```
#define SIO_VECTOR 4 /* 0x23 Serial port */
```

```
#endif
```