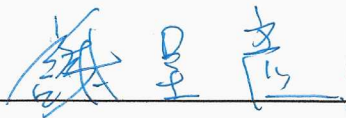


## 王彥傑 高職單晶片實習專題製作說明

名稱	以 AT89S52 單晶片設計之具有日曆功能之電子鬧鐘
說明	這是在讀台南高工資訊科製作的一份專題報告。 因應推甄，附上的報告是後來稍作精簡版，跟原本的不完全一樣。
指導老師	謝呈彥
組員的姓名	王彥傑、李宜禎、曾紋瑩、胡宇承
我負責的部份	C 語言程式撰寫、程式測試、報告製作

指導老師簽名



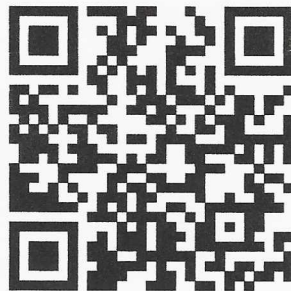
甄試委員您好：

這是我讀高職期間做的一份專題，也是我第一次把軟體跟硬體做結合的一份專案。這份專題最精髓的部份是程式的撰寫，最重要的邏輯就在程式裡。過程中思考如何在小小的 256 bytes 記憶體內儲存程式運作中所需用的資料。需要設計資料結構、演算法，此外因為與硬體配合，需要練習製作與硬體互動程式。請看以下列舉：

(括號內為對應 C 語言程式碼檔案)

- 資料結構的設計：使用整數儲存日期時間、儲存每一組鬧鐘設定 (alarm.c, date.c)
- 演算法：整數與日期時間的互轉、下次鬧鐘距離現在時間計算 (date.c)
- 與硬體互動：透過軟體操作 LCD、偵測鍵盤輸入 (lcd.c, input.c)
- 計時精準度改進：設計潤百分之一秒 (main.c, def.h)

C 語言的程式碼，放在 Github 上，可以掃描下面 QR Code 或點選網址觀看。

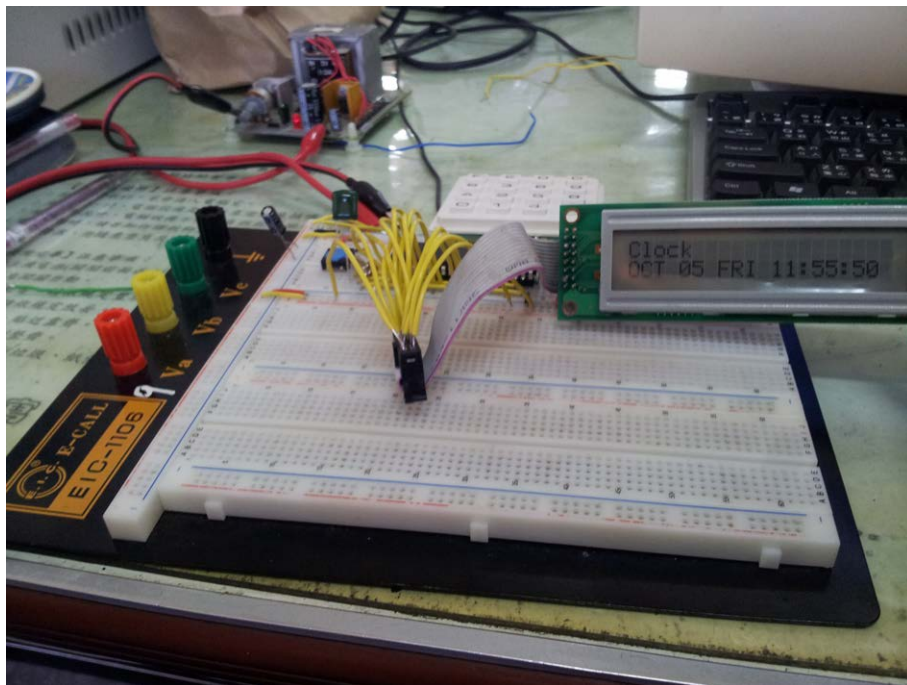


<https://github.com/bzeme/highschoolreport>

101 學年度國立臺南高工資訊科  
Department of Information  
National Tainan Industrial High School

單晶片實習專題製作報告 (2021 年精簡版)

以 AT89S52 單晶片設計之具有日曆功能之電子鬧鐘



**班級** 資訊科 三年級甲班

**組員**

03 李宜禎 07 曾紋瑩

09 王彥傑 17 胡宇承

**指導教師** 謝呈彥 老師

**繳交日期** 2013 年 1 月 4 日

電子鬧鐘在日常生活中很常見，為了驗證我們單晶片學習的成果。因此我們決定以單晶片製作一個具有日曆功能的電子鬧鐘。所有的日曆功能都是使用軟體製作，並沒有使用到任何現成的電子日曆時鐘 IC。

AT89S52 是一個 MCS-51 系列的單晶片，是 Atmel 公司出品的單晶片，其相容於 Intel 公司的 8052 單晶片，有 8KB 的 Flash ROM，還有 256 Bytes 的 RAM，有 6 個中斷源，以及 3 組計數器，還有 4 組 8bit 之輸入輸出埠。除此之外，還可以線上燒錄，意味著不必把單晶片從電路中移除即可燒錄程式，是一顆相當實用且常見的單晶片。

我們使用 AT89S52 單晶片製作電子鬧鐘，單晶片因為把輸入、輸出、記憶與處理單元融合至一顆晶片，因此製作出來的線路簡單，成本低，不必連接許多邏輯線路，完全以軟體控制週邊硬體，例如：蜂鳴器、LCD 顯示器等。但因為軟體的比重相當重，因此必須要花相當多的時間開發軟體，因此在這個專題中，C 語言的使用就相當重要了。

計時的部分，使用了 Timer 2 計數器，並且使用「16 位元自動重新載入模式」，因為有自動重新載入，所以定時相當精準。每當 Timer 2 計數器進位時，就會產生中斷及重新載入計數器數值，然後執行中斷副程式，這時只要在中斷副程式進行計時即可。

<b>1</b>	<b>簡介</b>	<b>1</b>
1.1	功能介紹 . . . . .	1
<b>2</b>	<b>時程表與工作分配</b>	<b>2</b>
2.1	時程表 . . . . .	2
2.2	工作分配 . . . . .	2
<b>3</b>	<b>電路圖及重點元件介紹</b>	<b>3</b>
3.1	電路圖 . . . . .	3
3.2	C 語言程式碼 . . . . .	3
3.3	SDCC 編譯器 . . . . .	4
3.4	AT89S52 單晶片 . . . . .	4
3.5	材料清單 . . . . .	4
3.6	工具 . . . . .	5
<b>4</b>	<b>實作</b>	<b>7</b>
4.1	程式撰寫 . . . . .	7
4.2	編譯程式碼 . . . . .	8
4.2.1	USB 燒錄程式 . . . . .	8

4.3	麵包板接線 . . . . .	9
4.4	實際操作 . . . . .	11
4.4.1	開始之畫面 . . . . .	11
4.4.2	設定時間 . . . . .	11
4.4.3	設定鬧鐘 . . . . .	12
4.4.4	鬧鐘鈴響 . . . . .	13
<b>5</b>	<b>測試、除錯、修正與改進</b>	<b>16</b>
5.1	測試過程 . . . . .	16
5.2	蜂鳴器電路修正 . . . . .	17
5.3	計時準確度修正 . . . . .	18
5.4	程式改進 . . . . .	21
5.5	Reset 電路修正 . . . . .	22
<b>6</b>	<b>結果與討論</b>	<b>23</b>
<b>7</b>	<b>參考資料</b>	<b>24</b>

相信大家都有睡過頭的經驗吧！我也不例外。為了不要睡過頭，通常會買一個鬧鐘放在家裡，然後設定好時間，只要時間到就會響了。但是傳統鬧鐘只能設定一組時間，且必須要手動開啟。

而電子鬧鐘就不同了，除了可以設定多組時間外，還可以依照星期幾來設定鬧鐘。此外這次專題又增加了日曆功能，可以隨時查看日期，並且加入了?床功能，如不想要起床，可以設定鬧鐘五分鐘後再次鈴響。為了讓使用者不會因為按了鬧鐘又睡著了，又精心設計了隨機的簡易加減法題目，使用者必須要答對題目，確認已經確實睡醒了，才可以中止鬧鐘。這種功能在其他的電子鬧鐘算是少見，不過很實用喔！

## 1.1 功能介紹

- 最多可設 20 組鬧鐘，每組鬧鐘可以設定一個時間，並可決定要在一星期的哪幾天鈴響。
- 具有電子日曆的功能，上面會顯示日期、星期與時間。
- 自動判斷星期以及閏年，使用者僅須設定年、月、日以及時間。
- 在設定時間後，若未來有誤差，不必 RESET，可按下重新調整鈕，即會進入日期時間調整之畫面，以調整日期時間。而原有設好的鬧鐘設定會保留。若使用 RESET，則所有設定會消失。
- 可計時範圍為 1972~2099 年。
- 具有賴床功能，只要按下賴床鈕，就會啟動 5 分鐘之倒數計時器，並於 5 分鐘後再度鈴響。
- 為了避免無人在家時，鬧鐘無限制鈴響，因此在鈴響開始時會自動啟動內部之計數器，若 1 小時沒有解除鬧鐘或按下賴床鈕，就會自動解除鬧鐘。

# 2

## 時程表與工作分配

### 2.1 時程表

週次	進度
1	討論題目並規畫進度
2	繪製電路圖、領取材料
3	裝配電路於麵包板上
4	撰寫程式，完成 LCD 與鍵盤輸入的程式
5	撰寫程式，完成日期時間輸入以及計時功能
6	撰寫程式，完成鬧鐘功能 修正電路，使蜂鳴器會鈴響
7	修正程式，把計時調整更準確
8	修正程式錯誤，使鬧鐘在正確的時間鈴響
9	修正程式，加入倒數計時功能

表 2.1: 每週時程表

### 2.2 工作分配

組員	工作
李宜禎	裝配電路、討論報告
曾紋瑩	裝配電路、剪單芯線
王彥傑	程式設計、燒錄、測試。報告製作
胡宇承	裝配電路、焊接 LCD 接腳

表 2.2: 工作分配表

### 3.1 電路圖

下頁圖 3.1 為使用 AT89S52 製作之具有日曆功能的電子鬧鐘。圖中各零件可以對照第5頁之表 3.1。

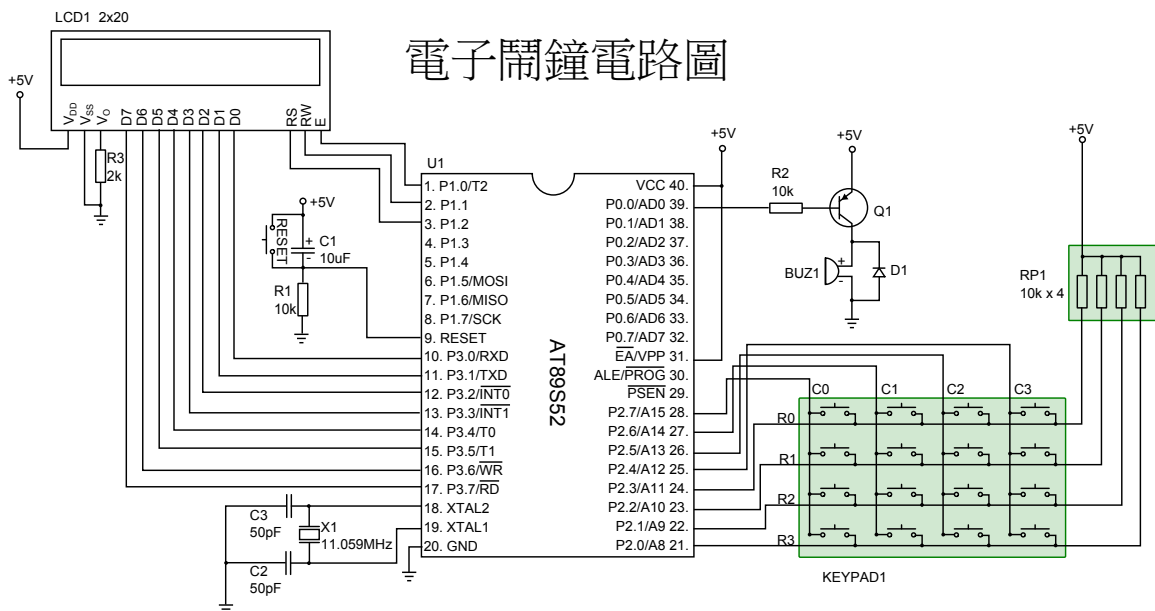


圖 3.1: 電路圖

### 3.2 C 語言程式碼

撰寫好的 C 語言程式放在 Github 內，連結: <https://github.com/bzeme/highschoolreport>

詳細實作介紹請看第 4 章。



### 3.3 SDCC 編譯器

我們使用 C 語言實作，使用 SDCC 作為編譯器，來製作 AT89S52 之程式。SDCC 全名為 Small Device C Compiler，是一個專門給小型晶片，像是基於 Intel MCS51 的晶片（我們所使用的單晶片 AT89S52 即是）和其他單晶片的 C 語言編譯器。

使用 SDCC 編譯器搭配 C 語言，可以免除撰寫組合語言的負擔，也讓撰寫複雜邏輯的程式更容易。

SDCC 編譯器的網站在此: <http://sdcc.sourceforge.net/>

### 3.4 AT89S52 單晶片

AT89S52，其指令碼和架構是基於 Intel MCS51 (或 8051) 系列的單晶片，它是整個專案所使用最重要的元件，所有的邏輯控制都在此。

它有 8KB 的 ROM 和 256 bytes 的 RAM。ROM 是儲存指令碼 (code) 和常數資料的位置，而 RAM 則是儲存程式運作時資料的位置，包含現在日期時間、鬧鐘設定、系統狀態和其他暫存變數。

它有 4 組 8-bit I/O Ports 和 3 組 Timer，我們使用其 I/O Port 來連結控制週邊零件，包含 LCD 液晶顯示器、鍵盤、蜂鳴器。另外使用 Timer 2 這一組 Timer 和與此 Timer 搭配之硬體中斷，來進行計時的工作。

### 3.5 材料清單

表 3.1 為我們所需要使用到的材料。

代號	名稱	規格	數量
U1	AT89S52		1
BUZ	蜂鳴器	電壓式 5V	1
R1	電阻	10k	1
R2	電阻	20k	1
R3	電阻	2k	1
X1	石英晶體	11.059MHz	1
KEYPAD1	4x4 鍵盤		1
RP1	排阻	10k * 8	1
LCD1	LCD 顯示器模組	2 * 20	1
Q1	電晶體	NPN 9013	1
D1	二極體	1N4004	1
C1	電解電容	10 $\mu$ F	1
C2	陶瓷電容	50pF	1
C3	陶瓷電容	50pF	1
	單芯線		適量
	排線	二排共 14pin	1
	排針	二排共 14pin	1
	焊錫		適量

表 3.1: 製作電子鬧鐘所需的材料清單

## 3.6 工具

表 3.2為我們所需要使用到的工具，包含製作報告所需的軟體。

名稱	用途	數量
電腦	用來設計軟體	1
USB 燒錄器	用來燒錄 AT89S52	1
麵包板	用來裝配電路	1
數位電表	用來檢測電路	1
尖嘴鉗	用來剪線、接線輔助用	1
剝線鉗	用來剝開單芯線	1
焊接工具	用來焊接 LCD 排針	1
SDCC	C 語言編譯器，用以編譯與連結程式	1
Notepad++	文字編輯軟體，用以撰寫程式	1
Windows 作業系統	電腦運作所需之作業系統	1
Inkscape	向量繪圖軟體，用以繪製電路圖	1
L <sup>A</sup> T <sub>E</sub> X	排版軟體，用來編輯這份報告	1
Libusb	為 USB 燒錄器所需的驅動程式	1
Progisp	電腦上的單晶片燒錄程式	1

表 3.2: 製作電子鬧鐘所需的工具

## 4.1 程式撰寫

按照所需的功能撰寫程式，程式碼因篇幅緣故不放此。完整撰寫好的程式請參閱 Github 連結: <https://github.com/bzeme/highschoolreport>

由於 AT89S52 只有 256 bytes 的 RAM 空間，可存放目前的日期時間和 20 組鬧鐘，因此在資料結構上有特別進行設計。

- 日期和時間的部份，使用 32-bits 的無號整數來儲存，程式會在需要顯示時把時間轉成年、月、日、時、分、秒，顯示在 LCD 上。相關的程式在 `date.c` 內。這樣的儲存方式不只減少所需的空間，也減少中斷程式的指令數。
- 每一個鬧鐘設定只需 3 bytes 的空間。因此 20 組鬧鐘共 60 bytes。「時、分」用 16-bits 無號整數儲存；星期設定和鬧鐘開關用 1 byte 儲存，其中 7 bits 代表一星期中的每一天，此外剩餘的 1 bit 則記錄鬧鐘開關。

上述有提到減少中斷指令數，這是因為每 1/100 秒就會呼叫一次中斷，要避免下次中斷被呼叫時，先前的中斷副程式尚未執行完成。此外，也因為中斷時間執行短，單晶片可以有更多時間是在省電模式狀態。

為求精準度，我們是用 Timer 2 計數器，並且使用「16 位元自動重新載入模式」，會在中斷發生時自動重新載入初始值，避免因透過指令碼重新載入初始值時，因延遲而發生計時延遲的情形。

程式中，有二個重要的硬體控制邏輯分別是 LCD 液晶顯示器 (輸出) 與鍵盤輸入。LCD 液晶顯示器的相關程式放置於 `lcd.c` 內，而鍵盤輸入的程式放置於 `input.c` 內。LCD 顯示器是透過傳送控制指令和資料給 LCD 顯示器來進行顯示；鍵盤輸入是透過傳送逐列掃描訊號，再去讀取每行的輸入訊號，來判定使用者按下的按鍵。

LCD 顯示器的指令與操作方式為參考書籍「89S51/89S52 單晶片與專題製作最佳範本, 黃慶璋、石博元著」製作。

## 4.2 編譯程式碼

### 安裝 SDCC 編譯器

上網搜尋 SDCC，並下載 SDCC 編譯器。

下載完後，點選  `sdcc-3.2.0a-setup.exe` 安裝。

### 編譯程式

為了方便編譯動作的進行，所以寫了 2 個批次檔，分別為：build.bat、clean.bat。

src/build.bat

```
1 set SDCC_FLAGS= --opt-code-size
2 set SDCC_LDFLAGS= --iram-size 256 --opt-code-size
3 call clean.bat
4 FOR %%i IN (*.c) DO sdcc %SDCC_FLAGS% -c %%i
5 sdcc *.rel -o clock.ihx %SDCC_LDFLAGS%
6 packihx clock.ihx > clock.hex
7 pause
```

src/clean.bat

```
1 del *.asm *.cdb *.rel *.hex *.ihx *.lst *.map *.rst *.sym *.lnk core *.dump* *.
   adb *.hashes *.lk *.mem *.omf
```

把這些檔案放在原來的目錄，點選 build.bat，就可以編譯了。

#### 4.2.1 USB 燒錄程式

1. 把 USB 燒錄器插上。

- 上網搜尋下載 Progisp1.72 並開啟，點選右邊 Load Flash，選擇 clock.hex，然後按下 Auto 即可燒錄。請看圖 4.1。

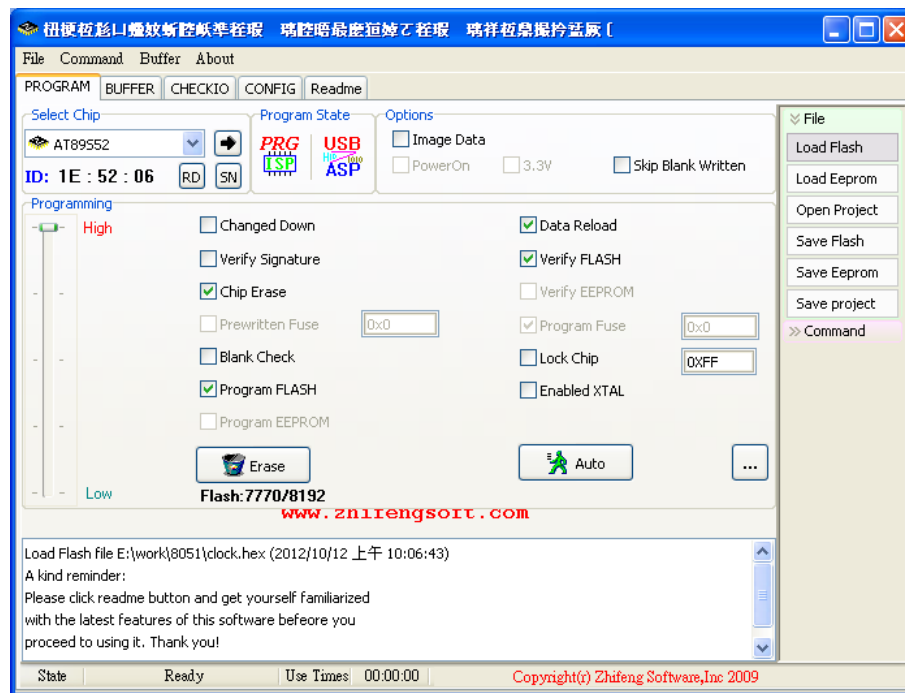


圖 4.1: Progisp 軟體介面

## 4.3 麵包板接線

接下來要把線路接上麵包板上，才能作進一步的測試。

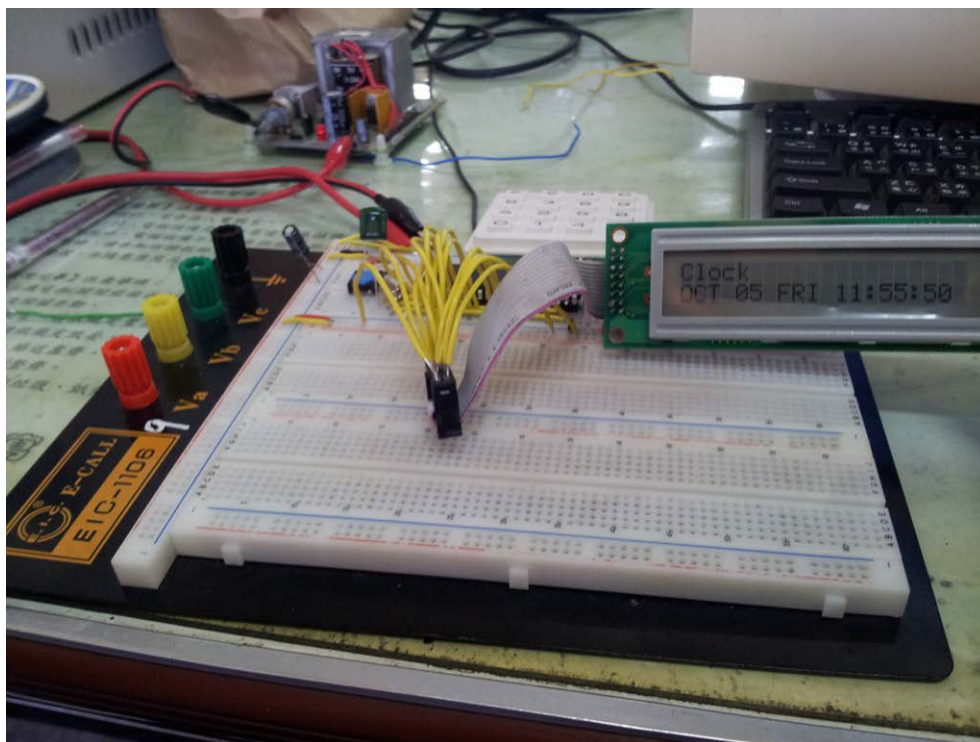


圖 4.2: 接線後的結果 1

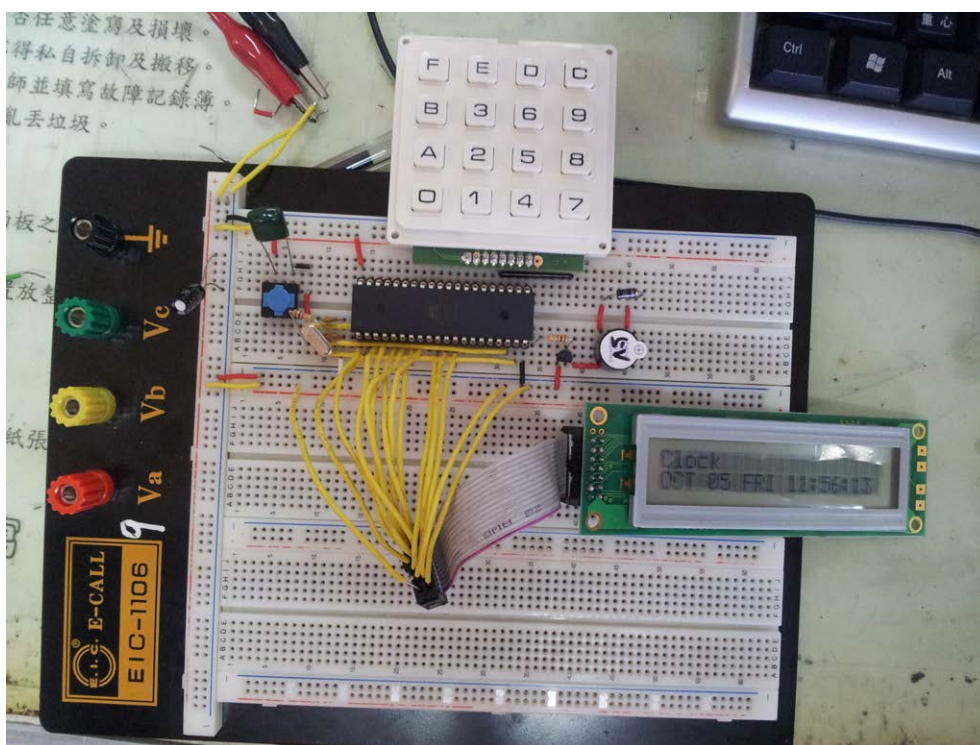


圖 4.3: 接線後的結果 2



## 4.4 實際操作

製作的過程中，會不斷地重覆修改程式，重新編譯、燒錄、測試。經歷不斷測試、修改過程，才作出成品。詳細電路與程式的修正請看第5章。

以下為完成後實際操作畫面。為了節省篇幅，所以只放 LCD 顯示器的圖片。

### 4.4.1 開始之畫面

當電源接上時，程式開始運作後，即會出現“Welcome! Alarm Clock”的字樣，如圖 4.4所示：



圖 4.4: 電源接上時之歡迎訊息

按下任意鍵就進入主畫面，開始計時，如圖 4.5。



圖 4.5: 鬧鐘主畫面

### 4.4.2 設定時間

進入主畫面後，按下 F 鍵，設定日期與時間，如圖 4.6所示：





圖 4.6: 設定日期與時間

設定完成後，按下 A 套用設定，如圖 4.7



圖 4.7: 日期與時間設定完成

#### 4.4.3 設定鬧鐘

這裡設了 2 組鬧鐘，間隔 10 分鐘，測試鬧鐘功能。其中一組設定為只鈴響一次，其中一組設定為星期三與星期五鈴響。如圖 4.8與4.9。



圖 4.8: 設定第 1 組鬧鐘為 20:49



圖 4.9: 設定第 2 組鬧鐘為 21:00，並且為星期三、星期五響

#### 4.4.4 鬧鐘鈴響

過了 5 分鐘後，鬧鐘鈴響了 (\*\* Wake Up! \*\* 閃爍)，按下賴床按鈕，測試賴床功能。



圖 4.10: 第一組鬧鐘響了



圖 4.11: 按下賴床鈕後進入賴床模式

5 分鐘賴床時間過後，又鈴響了 (\*\* Wake Up! \*\* 閃爍，圖為 \*\* Wake Up! \*\* 字樣閃爍消失時所拍的)，再按下解除鬧鐘按鈕，回答數學問題即可解除。



圖 4.12: 賴床模式結束，再次鈴響

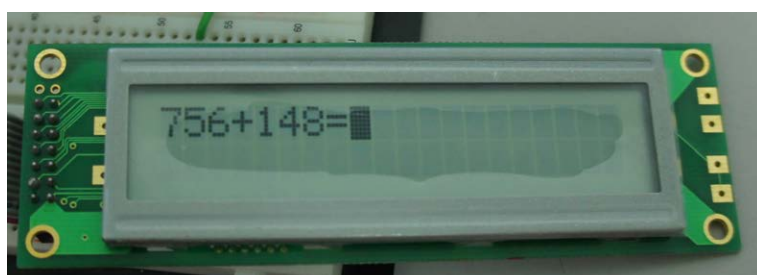


圖 4.13: 顯示數學問題



圖 4.14: 輸入數學問題



圖 4.15: 回答正確，解除鬧鐘

再過了幾分鐘後，第 2 組鬧鐘又響了，這次直接按下解除鬧鐘按鈕，回答數學問題解除鬧鐘。



圖 4.16: 第二組鬧鐘響了



圖 4.17: 回答數學問題



圖 4.18: 成功解除鬧鐘

任何的電腦軟體或電路，很難在第一次完成後就保證可以正常工作，所以必需先測試後，才能確定其有沒有問題。若電路或軟體有問題，則必需要進行改進與修正，此外，如果發現有需要增加或改進的地方也可以進行改進。

在我們的電子鬧鐘電路製作完成後，為了確保電路正常動作，所以我們進行測試與除錯的工作。經過我們組員的測試，電路確實有錯誤，且運作過程中也有問題，程式也有一些問題存在。經過修正後，這些問題已經解決了，電路也正常運作了。

以下就來介紹我們所遇到的問題，以及我們如何修正電路與程式，以解決問題。

## 5.1 測試過程

在接完電路並燒錄程式後，發現了些問題，分別為下列幾項：

- 蜂鳴器不響
- 時間有誤差
- 電源接上後，必需要先經過一次 RESET 電路才會正常

與軟體相關的問題有：

- 不該響的時候鈴響

為了測試時間的誤差，並且目標一日誤差 1 秒內，因此我們使用電腦的 NTP 軟體，先把電腦時間調準，然後看著時間，手動調整電子鬧鐘的時間，然後，再經過 2~3 日後，再次使用 NTP 軟體對時，並把電子鬧鐘之時間作比較。一剛開始會

發現慢了 10 幾秒，所以我們便調整程式中的參數，並且反復測試，把時間調到非常精準了。

此外我還實際把電子鬧鐘拿來使用，設定早上 2:30 分會響，結果真的有響。所以程式的部分應該沒有問題。

但後來發現，其實程式有問題。因為我發現它會在不該響的時後響。這時，檢查了一下程式碼，才發現我使用了 16bit 的整數來代表倒數的時間，但因為 16bit 太小了，結果造成溢位。後來改用 32bit 的整數，問題就解決了。

## 5.2 蜂鳴器電路修正

原本蜂鳴器的電路如圖 5.1，使用 CS9013 之 NPN 電晶體。

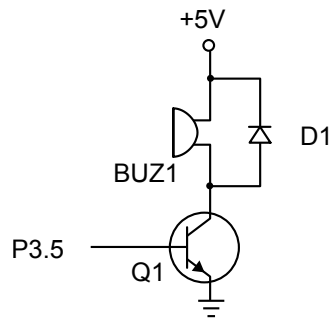


圖 5.1: NPN 電晶體線路

但因為 AT89S52 之輸出電流太小了，因此無法驅動電晶體使其進入飽和區，所以改用 CS9012 之 PNP 電晶體，並改接成 Pull up 的方式，並且把輸出腳由 P3 換到 P0，就變成了圖 5.2。

除此之外，還發現了蜂鳴器不會發出很穩定的聲音，而是有點小聲且斷斷續續的。經過幾次的嘗試，發現是因為電源有雜訊，加上濾波電容 10 $\mu$ F 於電源輸入端後，即可正常動作。



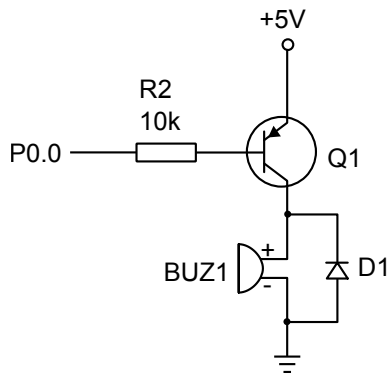


圖 5.2: PNP 電晶體線路

## 5.3 計時準確度修正

雖然我們使用的是 11.059MHz 的石英晶體，但是因為材料在製作過程會有誤差。為了修正計時器的準確度，所以特別在 def.h 檔案內加入了幾個有關計時的項目，只要修改 def.h 內的相關選項，包含了閏百分之一秒，以及計數的週期。共有 3 個項目，分別為 CLOCK\_D12、CLOCK\_LEAP、CLOCK\_LEAP\_MAX：

- CLOCK\_D12：代表為頻率除以 1200，也就是 AT89S52 經過  $\frac{1}{100}$  秒，計數器所上數之數字。當電路計數 CLOCK\_D12 次時，即會發生中斷，百分之一秒會加一 (程式內部之變數，不會顯示在 LCD 上)。
- CLOCK\_LEAP：每一次中斷為 leapCount 加上 CLOCK\_LEAP
- 當 leapCount 大於等於 CLOCK\_LEAP\_MAX 時，則把 leapCount 減去 CLOCK\_LEAP\_MAX 並且該次不計算百分之一秒。

下面為中斷副程式中的一小片段：

### main.c 片段

```

53 #if CLOCK_LEAP != 0
54     /* 為了調整使時間誤差減少，加上了閏百分之一秒之計算
55     */
56     leapCount += CLOCK_LEAP;
57     if (leapCount >= CLOCK_LEAP_MAX) {
58         leapCount -= CLOCK_LEAP_MAX;
59         return;
60     }
61 #endif

```

以及在 def.h 內的定義的一小片段：

def.h 片段

```
19 /*  
20  * 頻率調整  
21  * CLOCK_D12 頻率除以12  
22  * CLOCK_LEAP 每次閏百分之一秒計數  
23  * CLOCK_LEAP_MAX 計數超過此數字則閏百分之一秒  
24  */  
25 #define CLOCK_D12 9215  
26 #define CLOCK_LEAP 1  
27 #define CLOCK_LEAP_MAX 88465
```

為了測量時間的精準度，並量測實際 AT89S52 之工作頻率，所以我去下載了網路校時軟體 NTPClock，可以到 <http://www.stdtime.gov.tw/chinese/home.aspx> 下載，執行後出現下面的畫面，左下角顯示 N，代表時間已校正，並且顯示的是標準時間。如圖 5.3 所示。

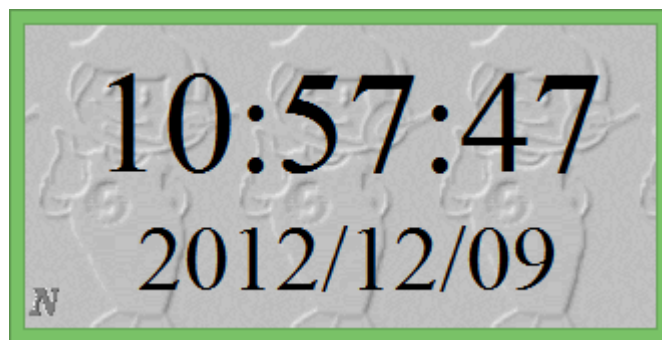


圖 5.3: NTPClock 軟體執行畫面

然後再把時間設定到電子鬧鐘上，如圖 5.4所示：

等候一天後，再次執行 NTPClock 軟體，比對時間，發現計時不準確了，用目測的觀察誤差幾秒。

**注意：**電腦內之時鐘其實並不準確，可能一天會誤差幾秒。所以為了要得到準確的時間，進行任何比對時間的動作前，一定要先執行 NTPClock 校時，才能得到最精確的時間。

用以下公式計算實際之工作頻率：公式中“預期工作頻率”為程式設定之 AT89S52 工作頻率，

$$\frac{\text{實際經過的秒數}}{\text{鬧鐘計時經過的秒數}} = \frac{\text{預期工作頻率}}{\text{實際的工作頻率}}$$



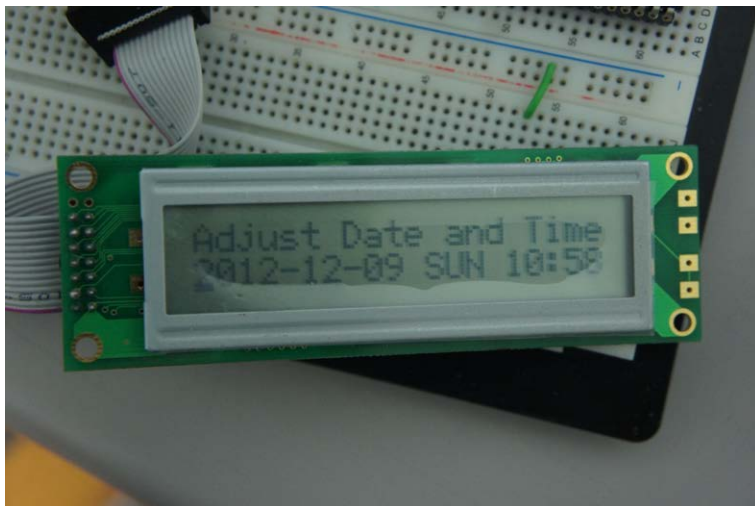


圖 5.4: 設定電子鬧鐘時間

因此，若我們在 `def.h` 中設定 `CLOCK_D12` 為 9216，而 `CLOCK_LEAP` 為 0，代表頻率為  $9216 * 1200 = 11059200$ ，若一天慢了 10.5 秒，則代入公式：

$$\frac{86400}{86389.5} = \frac{11059200}{x}$$

解出  $x$ ，為 11057856。再來要計算出 `CLOCK_D12` 的值，可用下面公式計算

$$\frac{\text{實際的工作頻率}}{1200}$$

設計算結果之整數為  $J = 9214$ ，小數為  $K = 0.88$ ，則 `CLOCK_D12` 就是  $J$  了。

因每一次計數都會少  $K(0.88)$  個週期，因此計時會變快，當時快過  $J + K(9214.88)$ ，即會多算了  $\frac{1}{100}$  秒，因此，我們要減少一次的計數。因此，`CLOCK_LEAP` 與 `CLOCK_LEAP_MAX`，的關係如下：

$$\frac{\text{CLOCK\_LEAP\_MAX}}{\text{CLOCK\_LEAP}} = \frac{J + K}{K}$$

只要把  $\frac{J + K}{K}$  化簡，即可得到 `CLOCK_LEAP_MAX` 與 `CLOCK_LEAP` 的值 (此處分別為 115186 與 11)。

## 5.4 程式改進

原本我們製作好的鬧鐘沒有顯示倒數計時，而且有時會在不該響時鈴響，後來查明原因，是因為使用了 16bit 有號整數的緣故，因為一週為  $86400 \times 7 = 604800$  秒，超過 16bit 之最大值 32767，所以會造成不準確，最後決定改成 32bit 整數。以下為原始程式碼與修改後程式碼的對照 (檔案位於 alarm.c 與 alarm.h)。

alarm.c 修改前：

alarm.c 片段

```
15 /* 距下次鈴響的時間 */
16 __idata volatile int next = 0;
```

alarm.c 修改後：

alarm.c 片段

```
15 /* 距下次鈴響的時間 */
16 __idata volatile long next = 0;
```

alarm.h 修改前：

alarm.h 片段

```
5 extern __idata volatile int next;
```

alarm.h 修改後：

alarm.h 片段

```
5 extern __idata volatile long next;
```

此外，還曾加了倒數計時之功能，增加下面程式碼至 main.c：

main.c 片段

```
216 /* 重新整理 LCD 上的時間 */
217 if (tmp > 0) {
218     if (tmp > 86399) {
219         lcd_show_num(tmp / 86400, 1, 0, 1);
220         lcd_puts("d ");
221     } else
222         lcd_erase(0, 3, 0);
223
224     tmp = tmp % 86400;
```

```
225         lcd_show_num(tmp / 3600, 2, 0, 1);
226         tmp = tmp % 3600;
227         lcd_putchar(':');
228         lcd_show_num(tmp / 60, 2, 1, 1);
229         lcd_putchar(':');
230         lcd_show_num(tmp % 60, 2, 1, 1);
231     }
```

## 5.5 Reset 電路修正

原本 Reset 電路是按照單晶片課本接的，使用的電容是使用 0.1 $\mu$ F 以及 10k 的電阻，但是後來發現電源接上時，似乎無法 Reset，所以有時會有一些奇特的事發生在電源接通時。後來發現是 Reset 電路的問題，於是把電容換成 10 $\mu$ F，電路就正常了。

經由這次製作有關電子鬧鐘的專題，我們了解到 AT89S52 單晶片的使用方式，還有如何撰寫程式來控制 LCD 還有 4x4 鍵盤，以及製作過程中，團隊合作的經驗。這次製作專題也算是增加同學之間的感情吧！

儘管製作過程中，偶爾會有些插曲，例如：麵包板接線完成後，發現電路不動作，檢查的時候，手碰到了 AT89S52 時，覺得被燙到，才發現 AT89S52 裝反了。雖然聞到似乎燒焦的味道，但是經過測試，並沒有燒壞零件，還是可以使用，不然就損失了 90 元了。

製作過程中，比較麻煩的是 LCD 顯示器的部分，因為學校購買的 2x20 的 LCD 顯示器，接腳是 2 列並排的，所以不能直接使用在麵包板上，因此我們把兩列共 14pin 之接腳焊上後，老師又去材料室幫我們找 14pin 之排線，並且使用排線連接，再用單芯線拉到麵包板上使用。雖然線很不美觀，但也只能這樣。

這次我們花了大部分的時間在寫程式上，因為學校單晶片的課程還沒教完的緣故，所以我們又自己研讀了有關單晶片的書籍，並且一邊參考書籍一邊寫程式。此外，也花了大部分的時間在日期與時間的演算法上，因為要考慮到閏年的問題，所以程式也會比較複雜一點。

這次的專題算是有成功，只是因為課程安排的因素，所以沒有多餘的時間把電路焊接到洞洞板上面，只能裝在麵包板上，可能要等到下學期統測考完後，比較有時間時，再來 Layout 與焊接電路了。

我們認為還有可以改良的地方。第一個是閏年的計算，因為程式是以 4 年為一次閏年計算，所以最多只能到 2099 年，因為 2100 年不是閏年。除此之外，還可以加上倒數計時器的功能，只是若要加入倒數計時器的功能，可能要想辦法再精簡程式，因為 8kB 的 ROM 已經快要用光了。

最後我們要感謝呈彥老師，指導我們製作專題，並給予我們意見，才讓這次專題可以完成。

- 89S51/89S52 單晶片與專題製作最佳範本, 黃慶璋、石博元著
- 電子學, 謝呈彥著
- SDCC - Small Device C Compiler, <http://sdcc.sourceforge.net/>
- Intel 8051, Wikipedia, [https://en.wikipedia.org/wiki/Intel\\_8051](https://en.wikipedia.org/wiki/Intel_8051)