邏輯設計實驗Lab6結報

105060012張育菘

**1 . Finish the time display function supporting 24-hour (00-23). 1.1 Support two modes: AM/PM and 24-hour.**

|  |  |  |  |
| --- | --- | --- | --- |
| hr\_ten | hr\_one | hr\_one | min\_one |
| R2,T1,U1,W2 | R3,T2,T3,V2 | W13,W14,V15,W15 | W17,W16,V16,V17 |

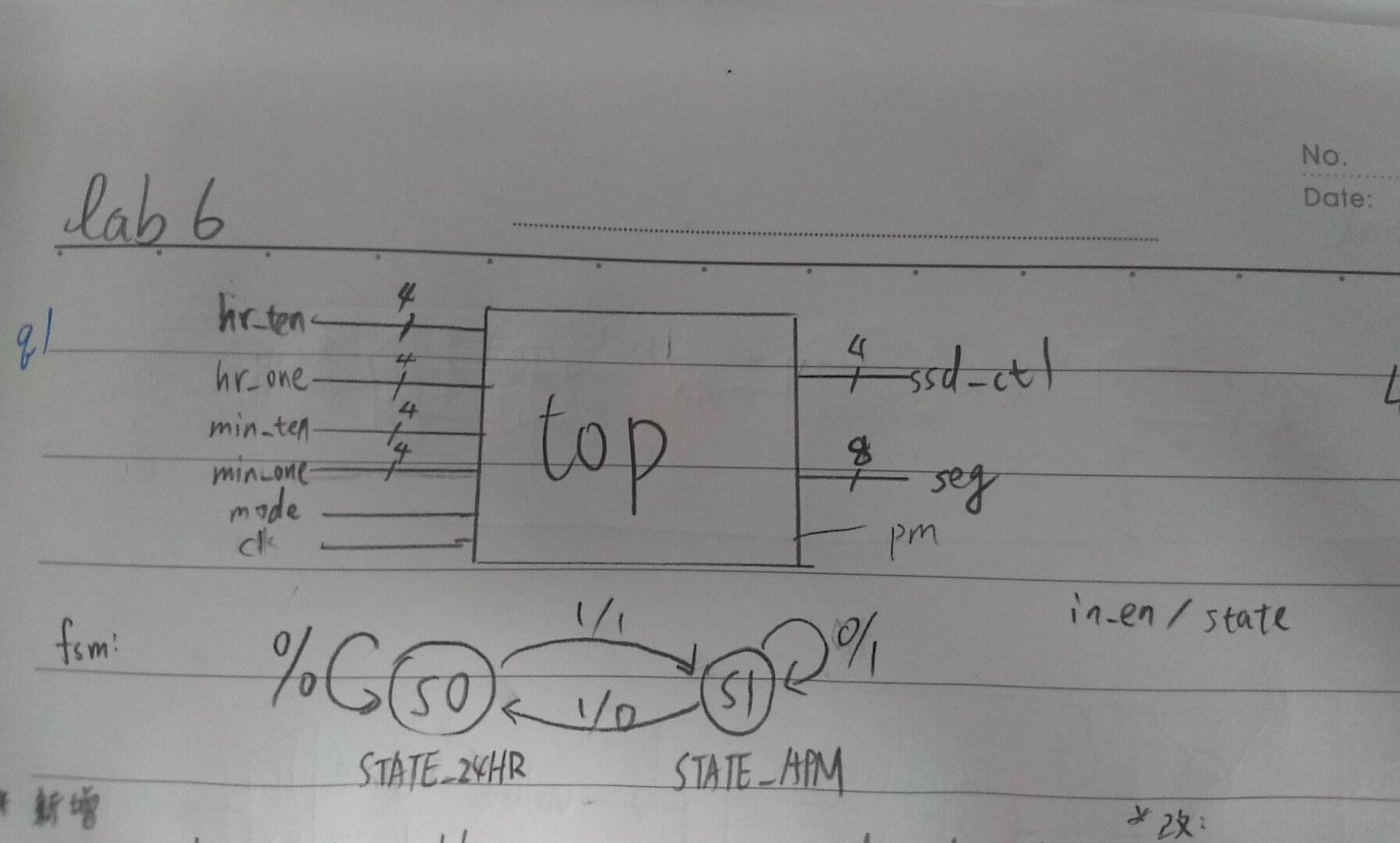
|  |  |  |
| --- | --- | --- |
| clk | mode | pm |
| W5 | T17 | U16 |

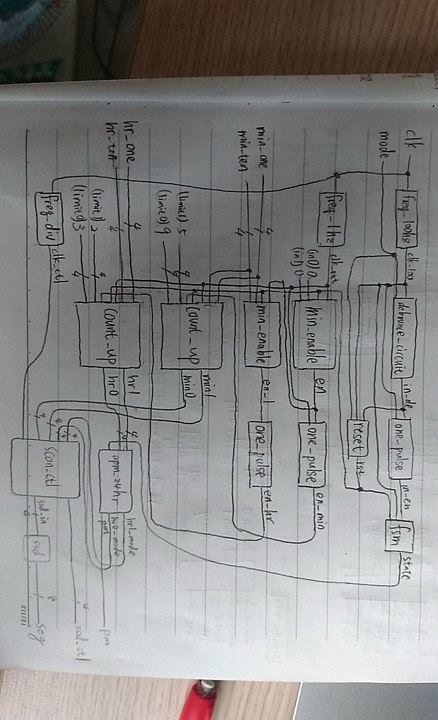
Design Specification

Input：clk,mode,hr\_ten[3:0], hr\_one [3:0], hr\_one [3:0], min\_one [3:0];

Output：ssd\_ctl[3:0],seg[7:0],pm;

block diagram :





Design Implementation

Logic function :

1. top：

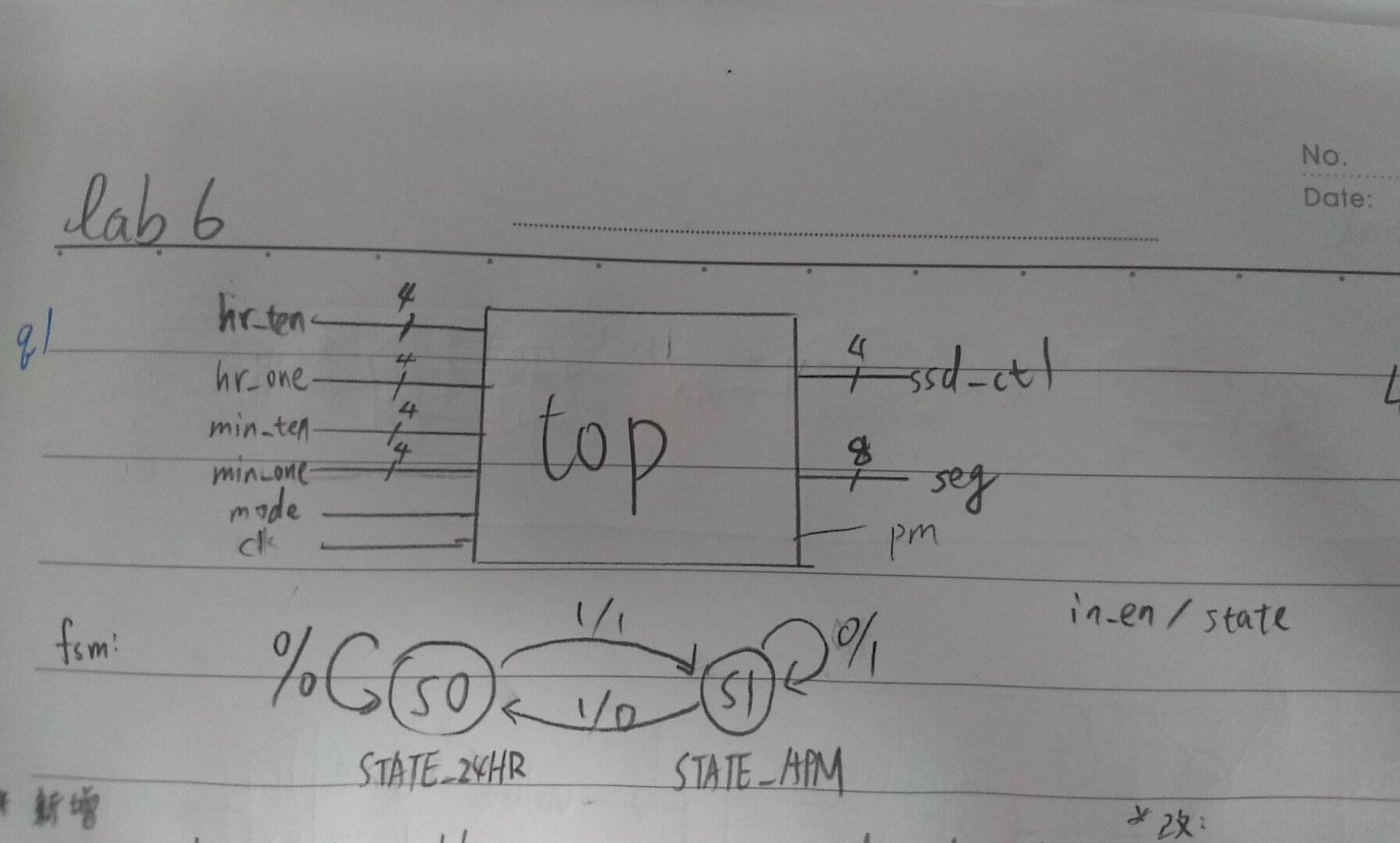
此top module，專門拿來呼叫其他小moudule的。

1. fsm：

此module做為finite state machine，以接出state至apm\_24hr決定顯示24hr抑或是A/PM的關鍵。

其input為經過one\_pulse處理過的訊號in\_en，因此當按下mode按鈕時，只會顯示一下的A/PM形式(1hz)。

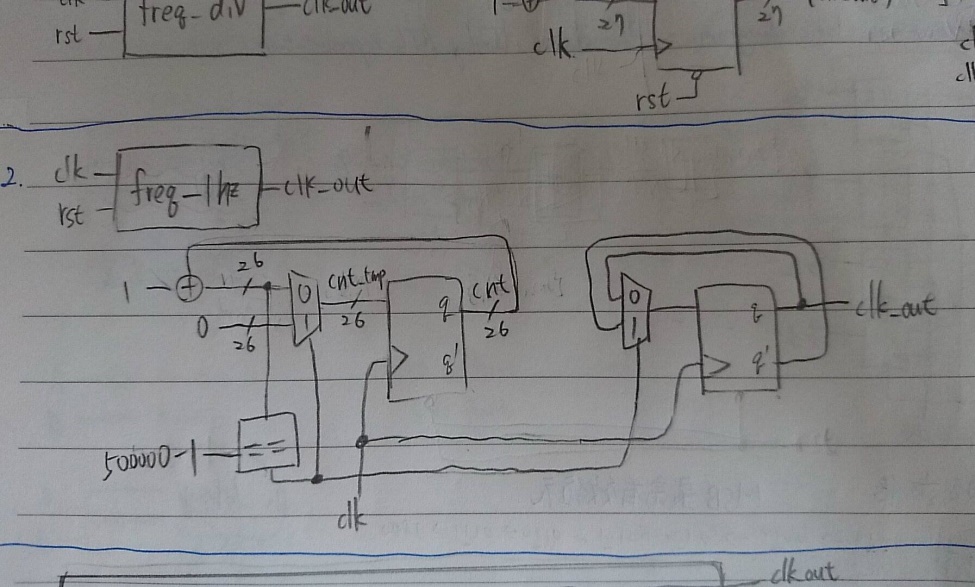
這裡的clk我是接100hz，因為clk在0🡪1時讀one\_pulse(其clk為100hz)處理過的值，因此top的clk不能太快。



1. freq\_1hz：

此為除頻的module，輸出1hz的clk\_out控制count\_down。

* 這裡不能接top的input rst，因為那裡的rst是希望count\_up重新開始數。當rst=0時，freq\_1hz輸出的clk持續為0🡪freq\_1hz不做事。



1. freq\_100hz：

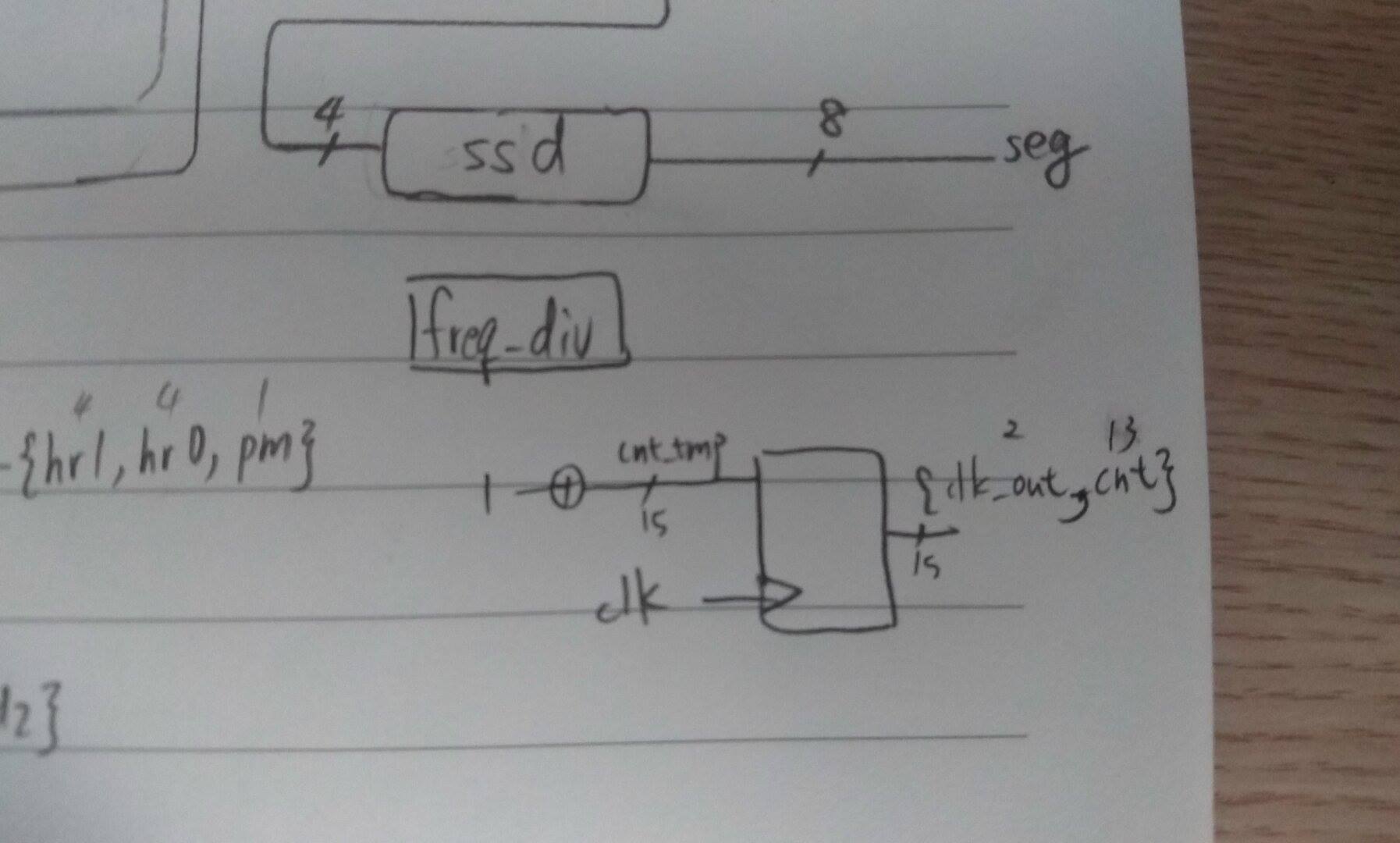
此為除頻的module，輸出100hz的clk\_out當作debounce\_circuit、one\_pulse以及fsm的clk。

* 這裡不能接top的input rst，因為那裡的rst是希望count\_up重新開始數。當rst=0時，freq\_100hz輸出的clk持續為0🡪freq\_100hz不做事。



1. freq\_div：

此為除頻的module，輸出的clk\_ctl當作scan\_ctl的clk，其頻率極高。

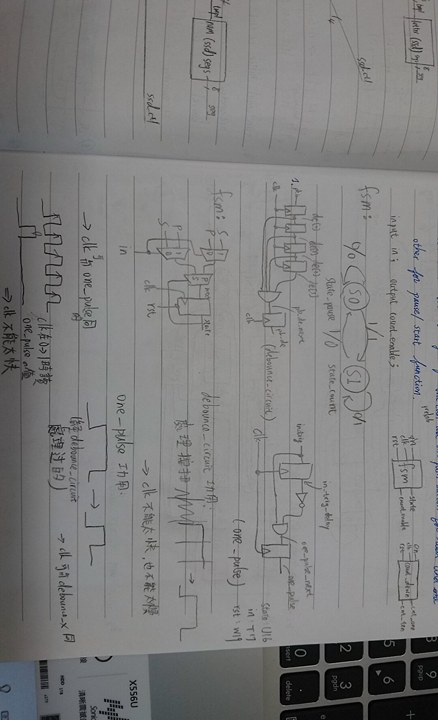


1. debounce\_circuit：

此為**除按鈕雜訊**的module，原理是，當debounce\_circuit收到連續4個1時才產生一個0🡪1的訊號，這樣可以避免收到前段起伏不定的雜訊，達到除雜訊的效果。

且要接100hz的clk，因為若接此的clk頻率太快就無法達到除雜訊的效果，太慢可能會超過按鈕維持穩定值的時間，因此我選擇100hz作為debounce\_circuit的clk。

* 這裡不能接top的input rst，因為那裡的rst是希望count\_up重新開始數。當rst=0時，freq\_100hz輸出的clk\_100持續為0🡪 debounce\_circuit不做事。

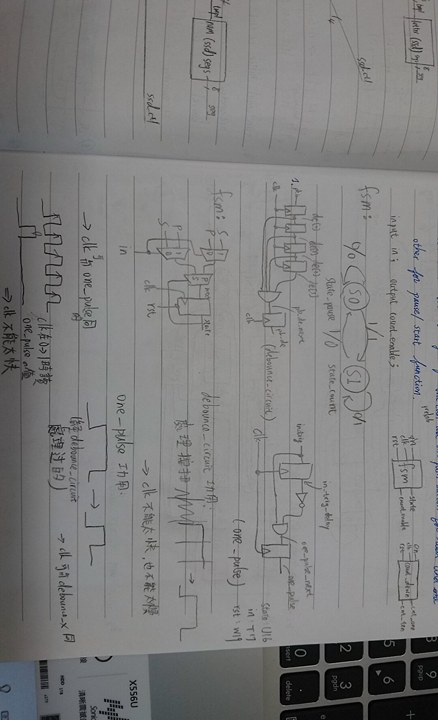


1. one\_pulse：

輸入訊號經過debounce\_circuit處理後，其週期可能會超過1個clk的時間，因此需要此module使訊號的週期為1個clk的時間。

這裡的clk我也是接與debounce\_circuit同的100hz。

* 這裡不能接top的input rst，因為那裡的rst是希望count\_up重新開始數。當rst=0時，freq\_100hz輸出的clk\_100持續為0🡪 one\_pulse不做事。



1. count\_up：

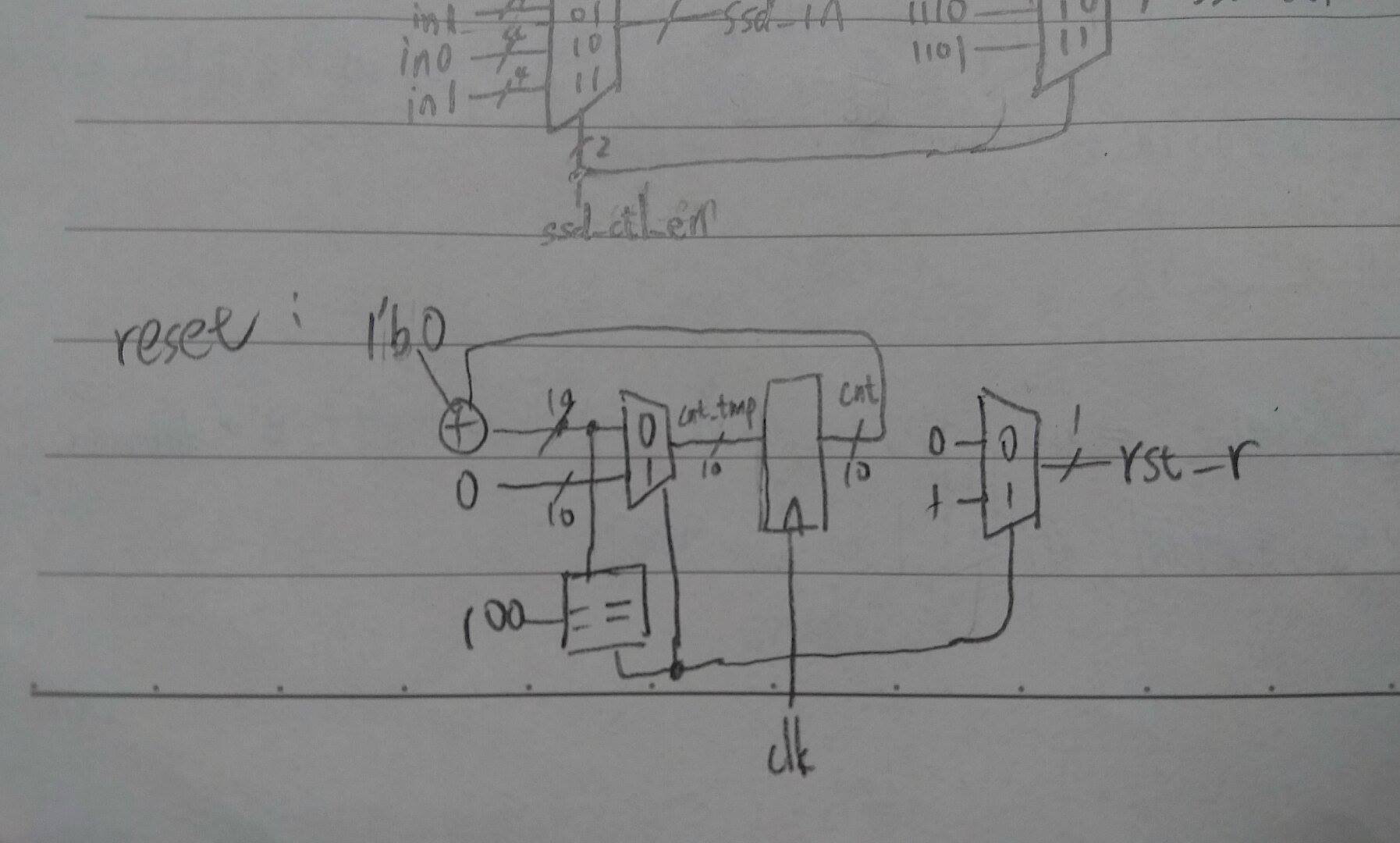
這是同時數兩個變數，個位數cnt\_one跟十位數cnt\_ten進行往下數。當cnt\_one=9時把cnt\_ten的數值加一以及cnt\_one的數值變成0；當cnt\_one跟cnt\_ten分別為limi0 & limit1(上界)時，讓兩個數變成0。而初始化(rst=0)就是把數設為cnt\_ten=in1 & cnt\_one=in0。

這裡還接一個en，作為是否能往下數的判斷值，若en=1，表示可以往上數；若en=0，則不能繼續往上數。

1. reset：

此module功能是為產生rst訊號。

而此input訊號rst來自只經過debounce\_circuit處理過的輸入訊號in\_de，因為此週期可能為多個週期的clk\_100，因此可以拿來作為判斷長按的依據。

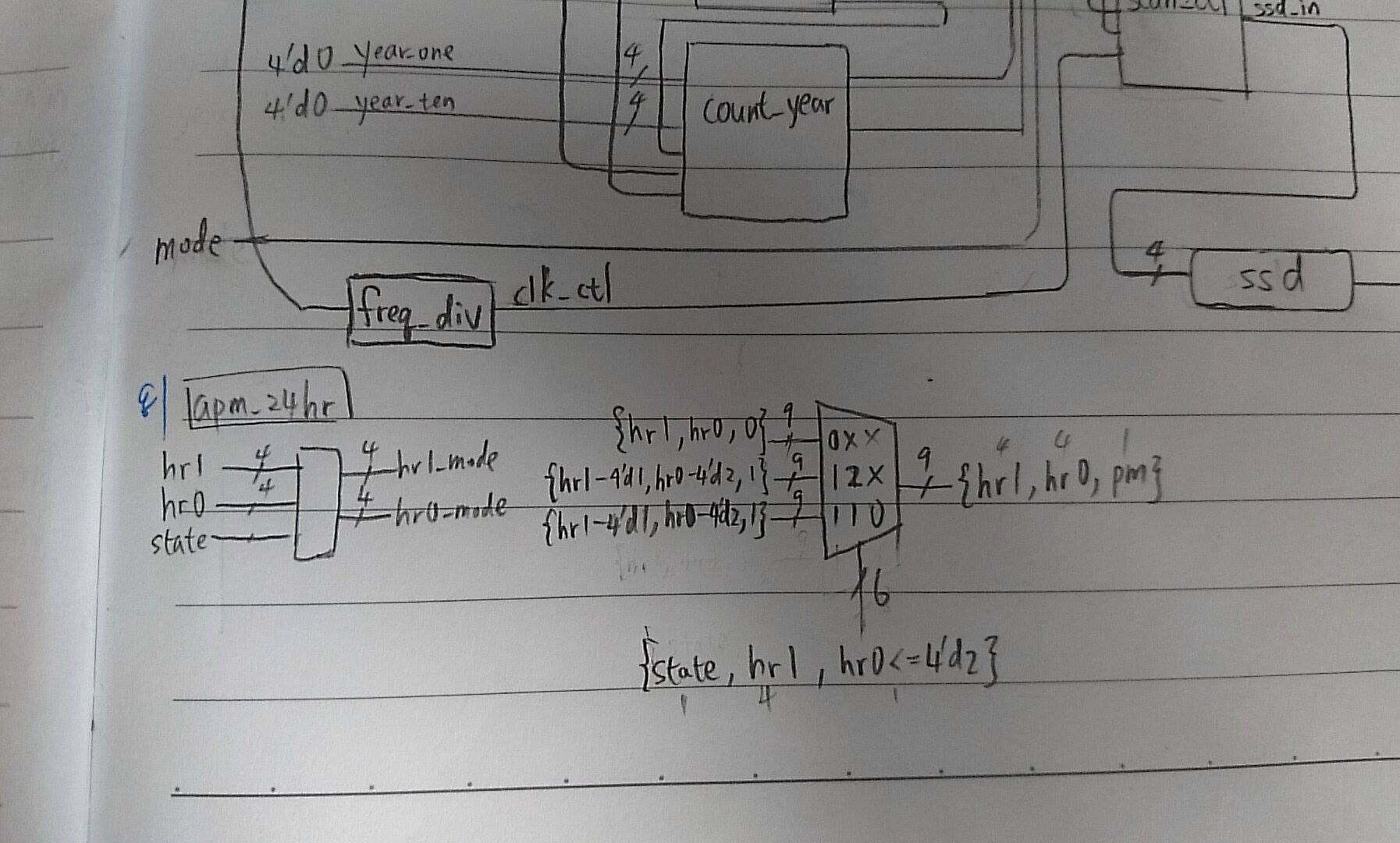
****

1. apm\_24hr：

此module是決定要顯示24hr還是A/PM。(STATE\_24HR：0, STATE\_APM：1)

作法就是判斷state,hr1的值&(hr0<=4’d2)此判斷式的對錯：

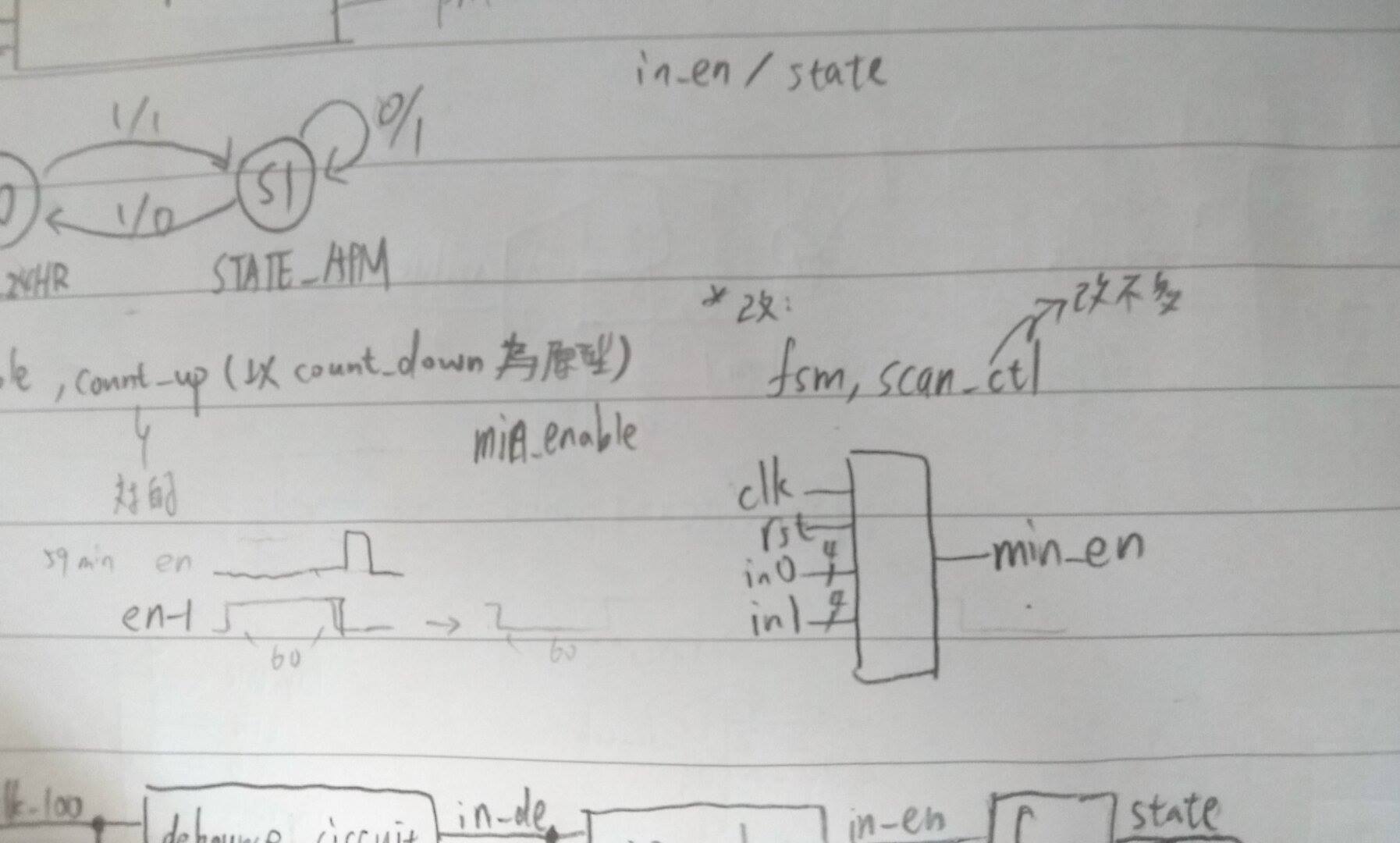
當(hr1==4'd2)&&(state==1) 或是 (hr1==4'd1)&&(~(hr0<=4'd2)) &&(state==1) 時，要從24HR的形式轉換成A/PM的形式，且顯示pm的燈要亮。



1. min\_enable：(從0數至60的counter)

此module是用來決定處理小時的counter和處理分鐘的counter能否開始加法。

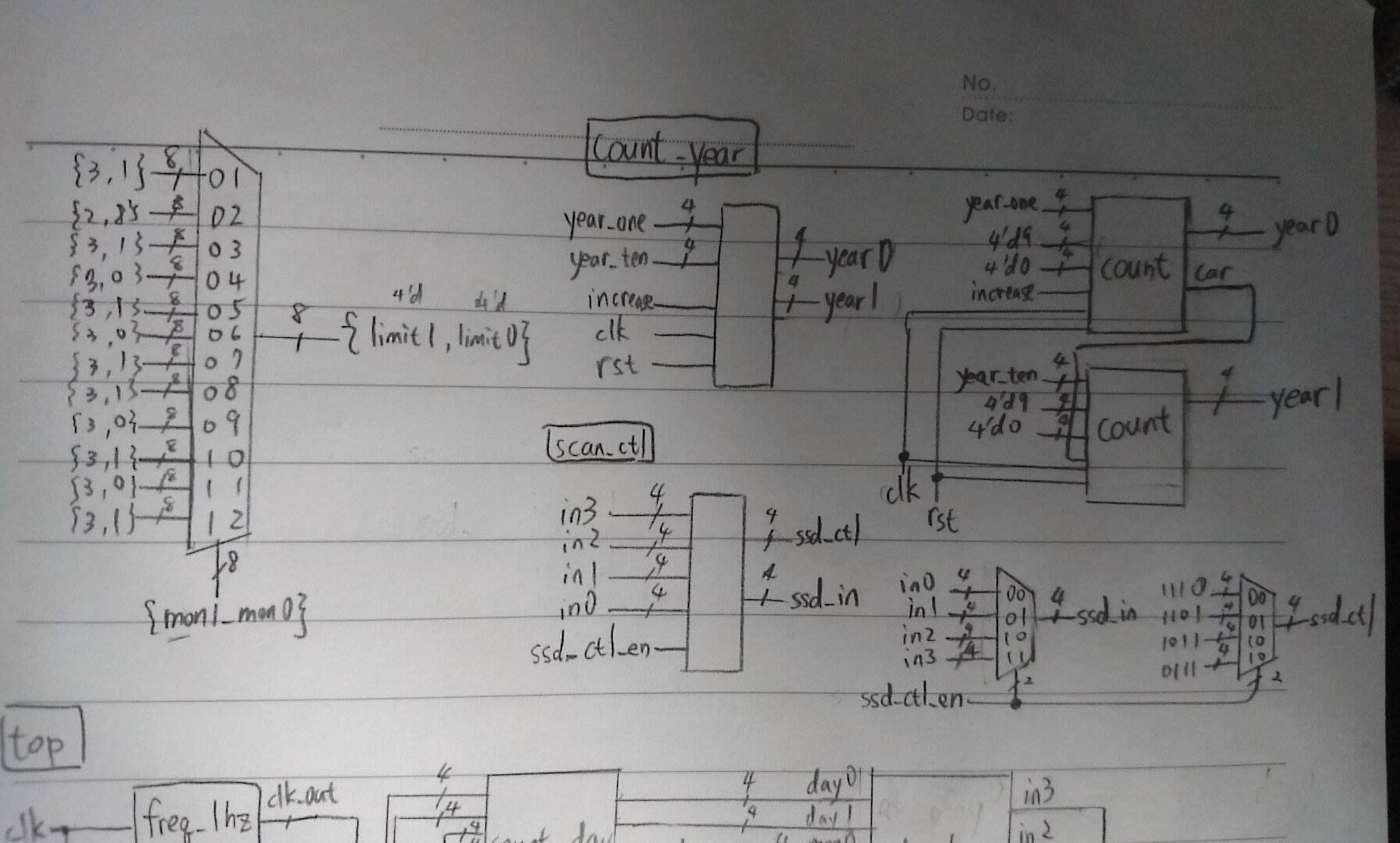
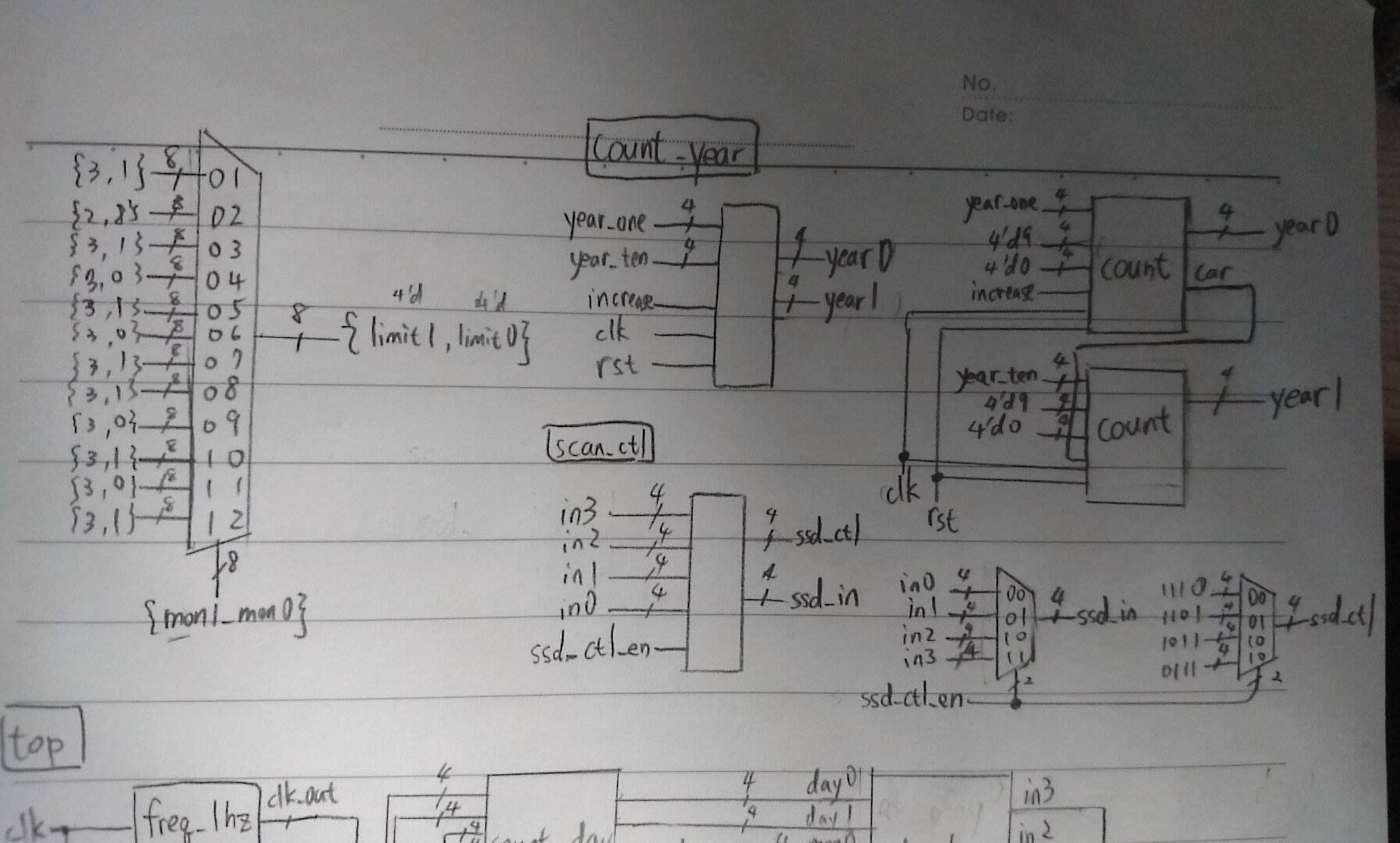
* 其產生的min\_en接出去後還要再接一個one\_pulse處理，因為其產生的min\_en訊號週期為為60倍的clk，因此要接一個one\_pulse處理成週期為clk的訊號。



1. scan\_ctl：

此module是讓七段顯示器同時顯示不同位數的module。

ssd\_ctl\_en(from freq\_div的clk\_ctl[2:0])的用意為顯示兩個不同的數，因為七段顯示器一次只能顯示一種數次，因此需要一個頻率介於內建時間到1hz的ssd\_ctl\_en產生同時顯示兩個數的錯覺。

1. ssd：

當bcd 為 4'd0: segs = 8'b00000011; //七段顯示器顯示”0”

當bcd 為 4'd1: segs = 8'b10011111; //七段顯示器顯示”1”

當bcd 為 4'd2: segs = 8'b00100101; //七段顯示器顯示”2”

當bcd 為 4'd3: segs = 8'b00001101; //七段顯示器顯示”3”

當bcd 為 4'd4: segs = 8'b10011001; //七段顯示器顯示”4”

當bcd 為 4'd5: segs = 8'b01001001; //七段顯示器顯示”5”

當bcd 為 4'd6: segs = 8'b01000001; //七段顯示器顯示”6”

當bcd 為 4'd7: segs = 8'b00011111; //七段顯示器顯示”7”

當bcd 為 4'd8: segs = 8'b00000001; //七段顯示器顯示”8”

當bcd 為 4'd9: segs = 8'b00001001; //七段顯示器顯示”9”

當bcd 為4'd10:segs=8'b11111111; //七段顯示器顯示” “

default: segs = 8'b00000000; //七段顯示器顯示”8.”

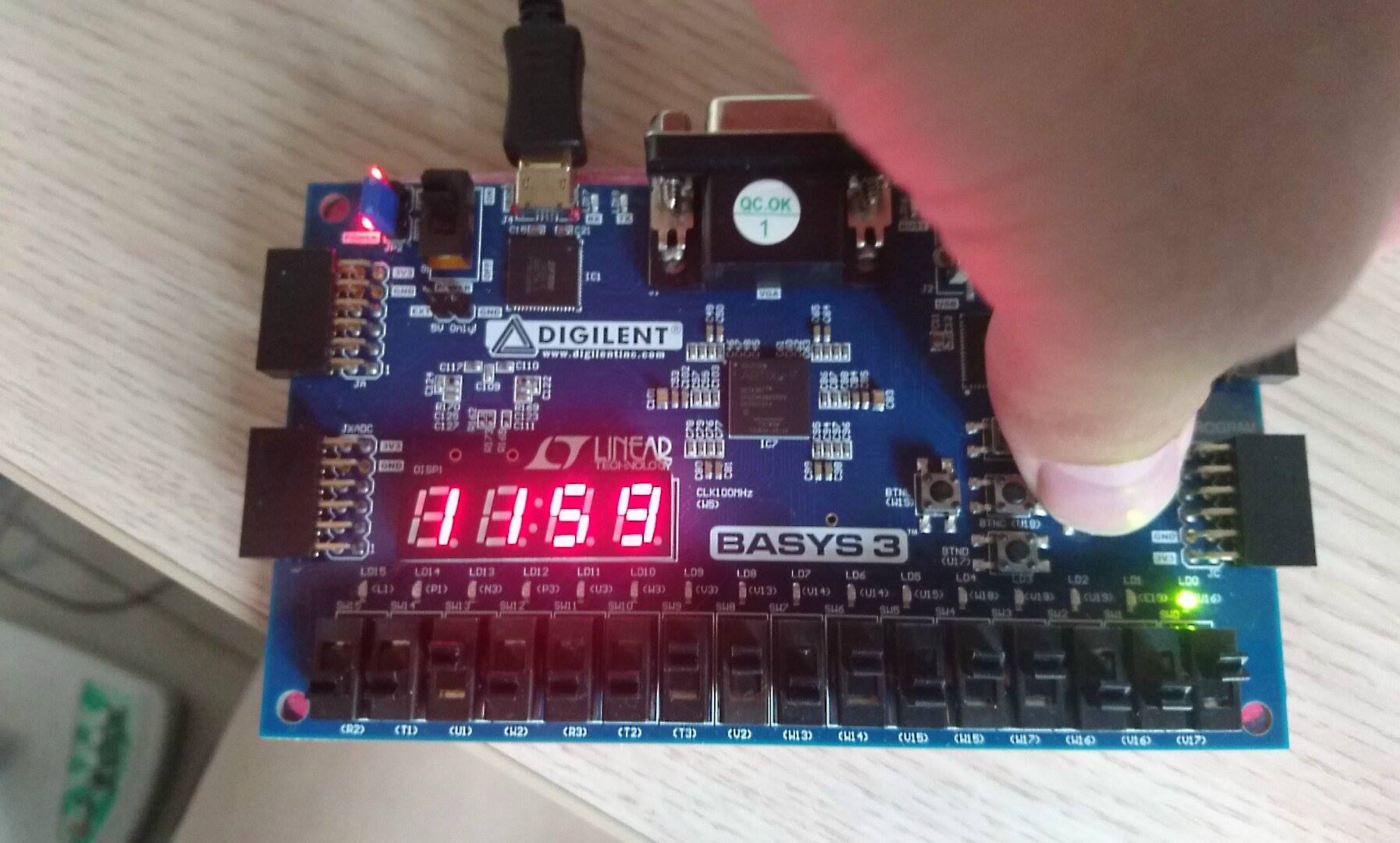
Result

1. 圖一：當長按V17，表示rst=1🡪數字顯示給定的初始值23:59。
2. 圖二：當按下V17，表示轉換mode🡪從顯示24HR變成顯示A/PM；且因為23:59變成11:59時表示此狀態為PM，因此U16亮。
3. 圖三：23:59經過一分鐘後會回至00:00。

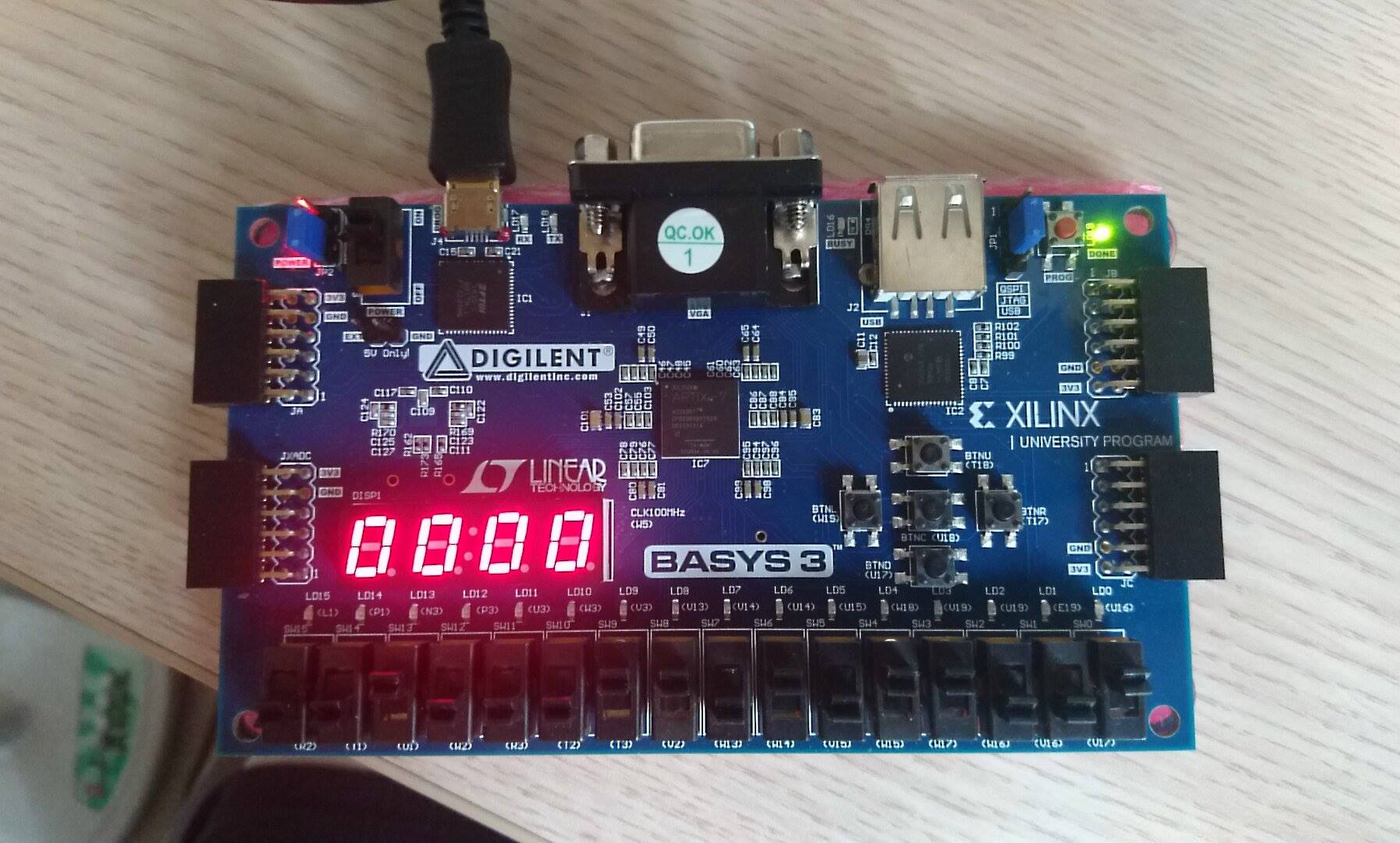
圖一



圖二



圖三



Discussion

1. 我在這設計的hr\_ten[3:0], hr\_one [3:0], hr\_one [3:0], min\_one [3:0]是用來作為初始值的設定，這樣的好處就是我有辦法決定我要從何處開始，也方便用來debug。
2. count\_up判斷rst要在rst從0🡪1時判斷，因為rst是一個按鈕，按下的過程是0🡪1，且要在rst==1時做出重置，這是跟以往實驗較不同之處。
3. min\_enable其產生的min\_en接出去後還要再接一個one\_pulse處理，因為其產生的min\_en訊號週期為為60倍的clk，因此要接一個one\_pulse處理成週期為clk的訊號。

**2. For the date functions in clock (no leap year), we have the following functions: o Day (Jan/March/May/July/Aug/Oct/Dec: 1-31, Feb: 28, Apr/June/Sept/Nov: 30), o Month (1-12), o Year (00-99).**

**Implement the following functions:**

**2.1 Month-Day function display in the 4 14-segment displays. 2.2 Combine the Year and 1.1 to finish a Year-Month-Day timer, and use one DIP switch to select the display of Year (2 Seven-Segment Displays, SSDs) or Month-Day (4 SSDs).)**

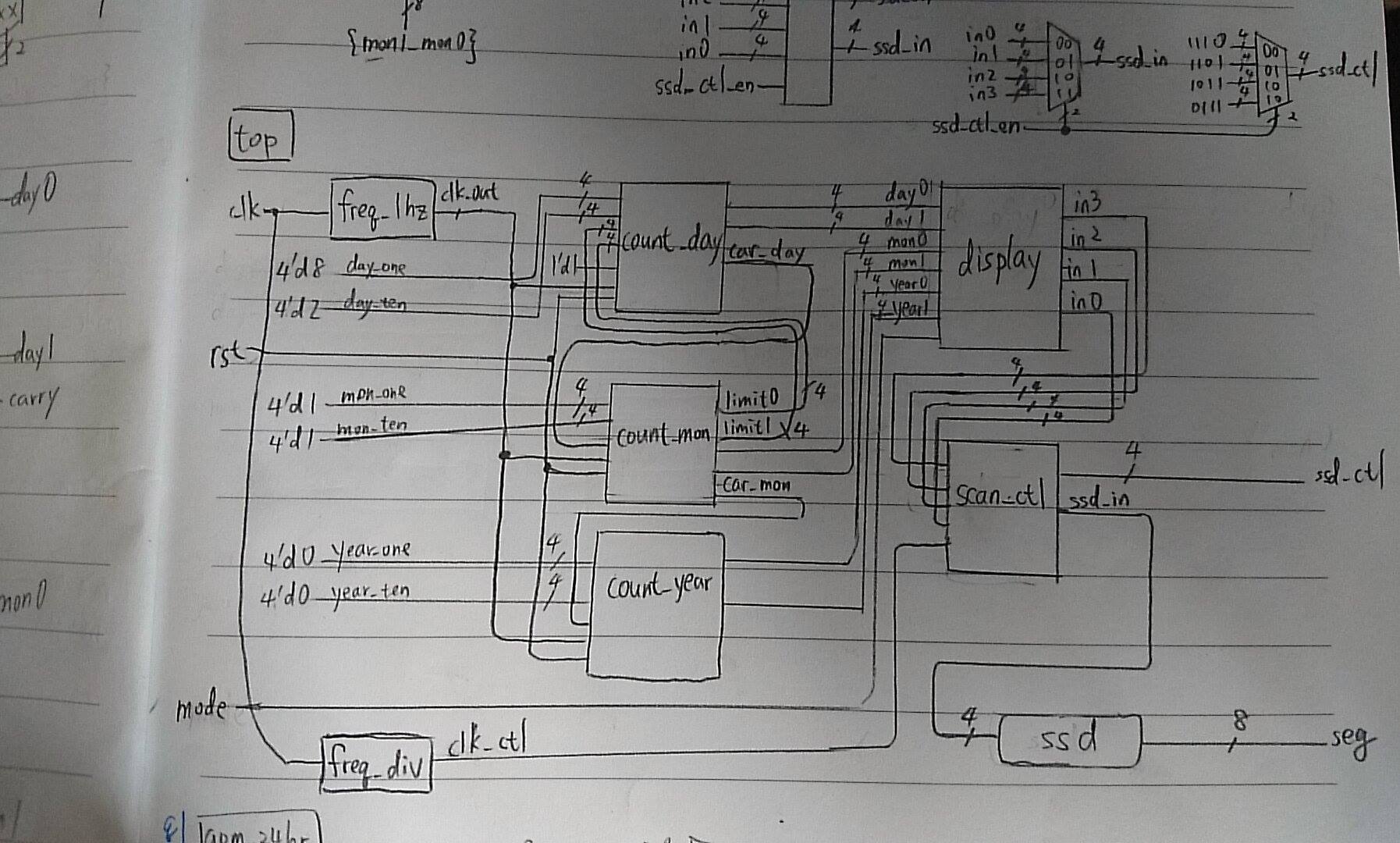
|  |  |  |
| --- | --- | --- |
| clk | rst | mode |
| W5 | V17 | R2 |

Design Specification

Input：clk,rst,mode;

Output：ssd\_ctl[3:0],seg[7:0],state;

block diagram :



Design Implementation

Logic function :

1. top：

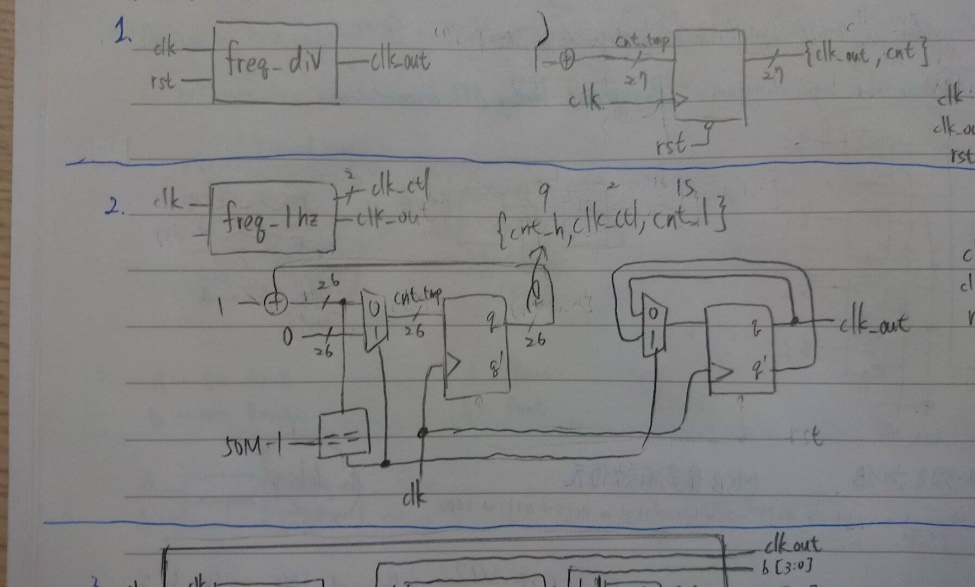
此top module，專門拿來呼叫其他小moudule的。

並給定初始值，年/月/日分別為00/11/28。

1. freq\_1hz：

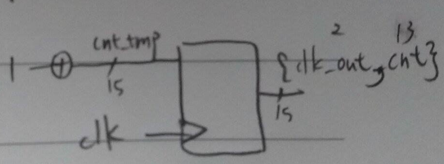
此為除頻的module，分別輸出1hz的clk\_out控制count\_down以及控制scan\_ctl的clk\_ctl。

* 這裡不能接display的input rst，因為那裡的rst是希望count\_down重新開始數。當rst=0時，freq\_1hz輸出的clk持續為0🡪freq\_1hz不做事。



1. freq\_div：

此為除頻的module，輸出的clk\_ctl當作scan\_ctl的clk，其頻率極高。

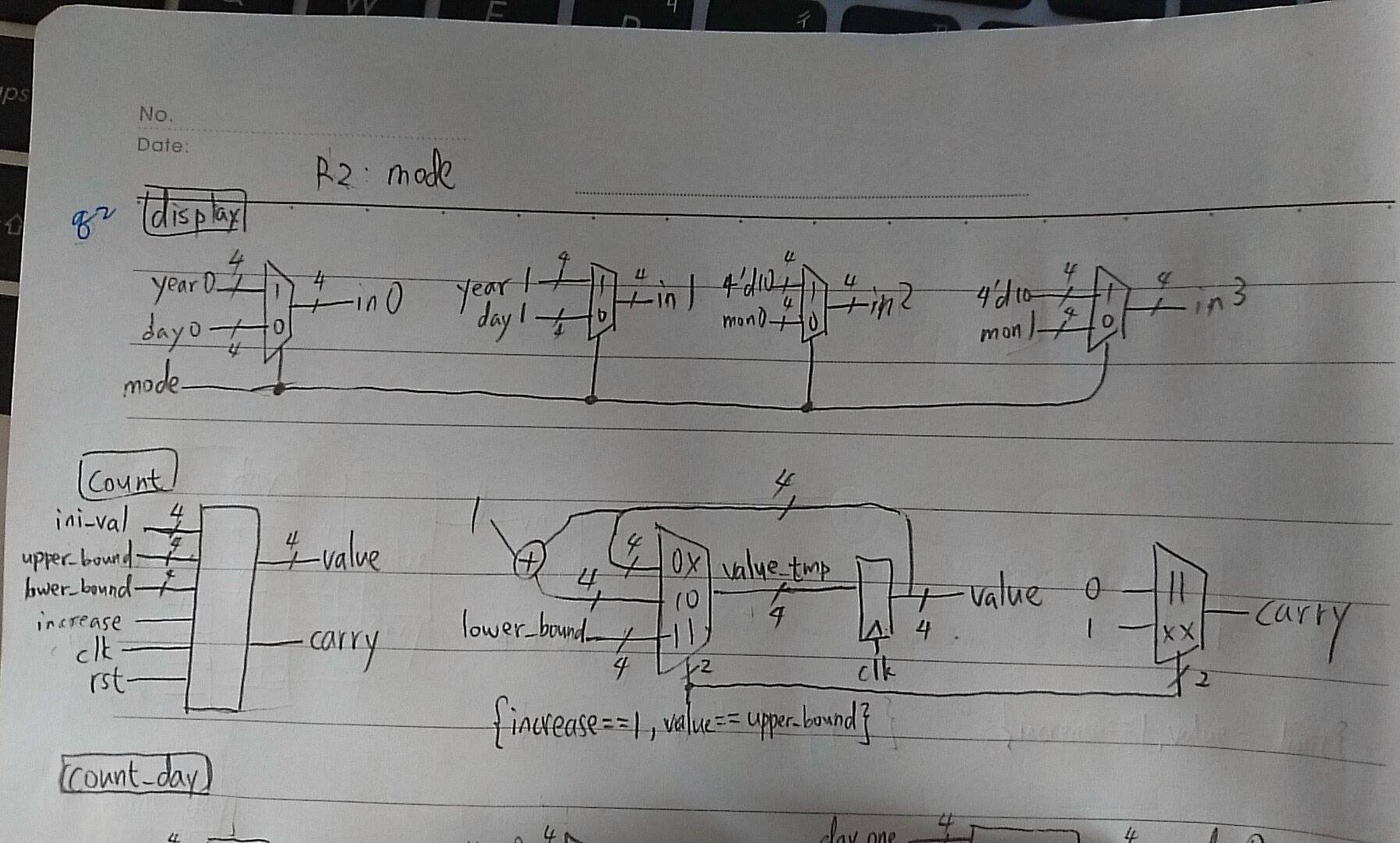


1. count：

此module是用來做一個數的加法。

carry表示進位；increase表示低一為數的進位；upper\_bound表示上界；lower\_bound表示下界；ini\_val表示初始值。

當increase=1表示有數近來，因此可以繼續進行加法。



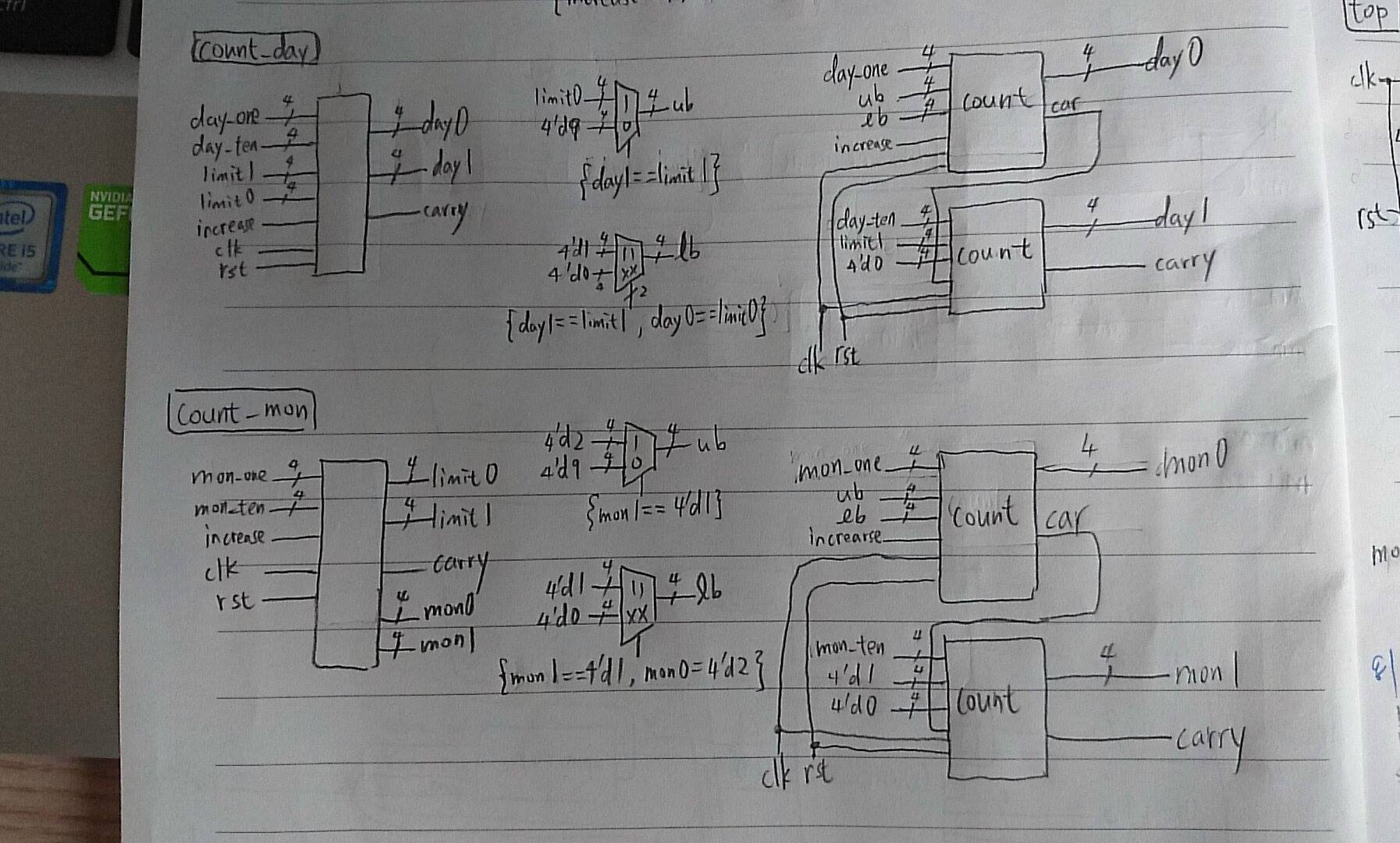
1. count\_day：

此module是用來做日期的顯示。

由於日期是一兩個數，因此內部接兩個count，分別做個位數及時位數的加法；而個位數的上界(ub)下界(lb)分別是用(day1==limit1)以及((day1==limit1) && (day0==limit0))來作判別給值。

此接出去的carry會作為count\_mon的increase。

* 個位數的carry即為十位數的increase，由於carry是在作暫存值時就給值了，因此當個位數要進位時，個位數以及十為數會同時變化。



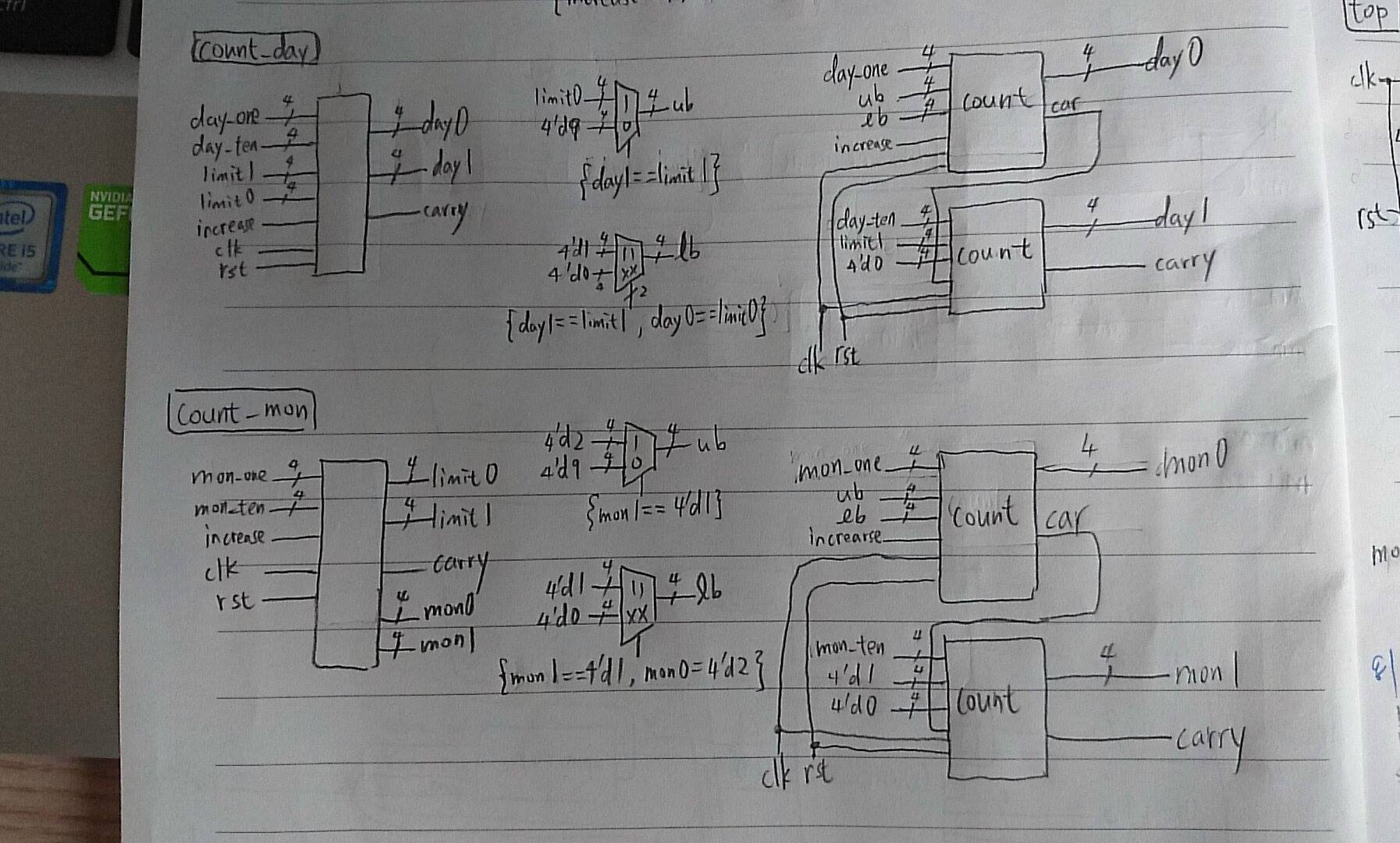
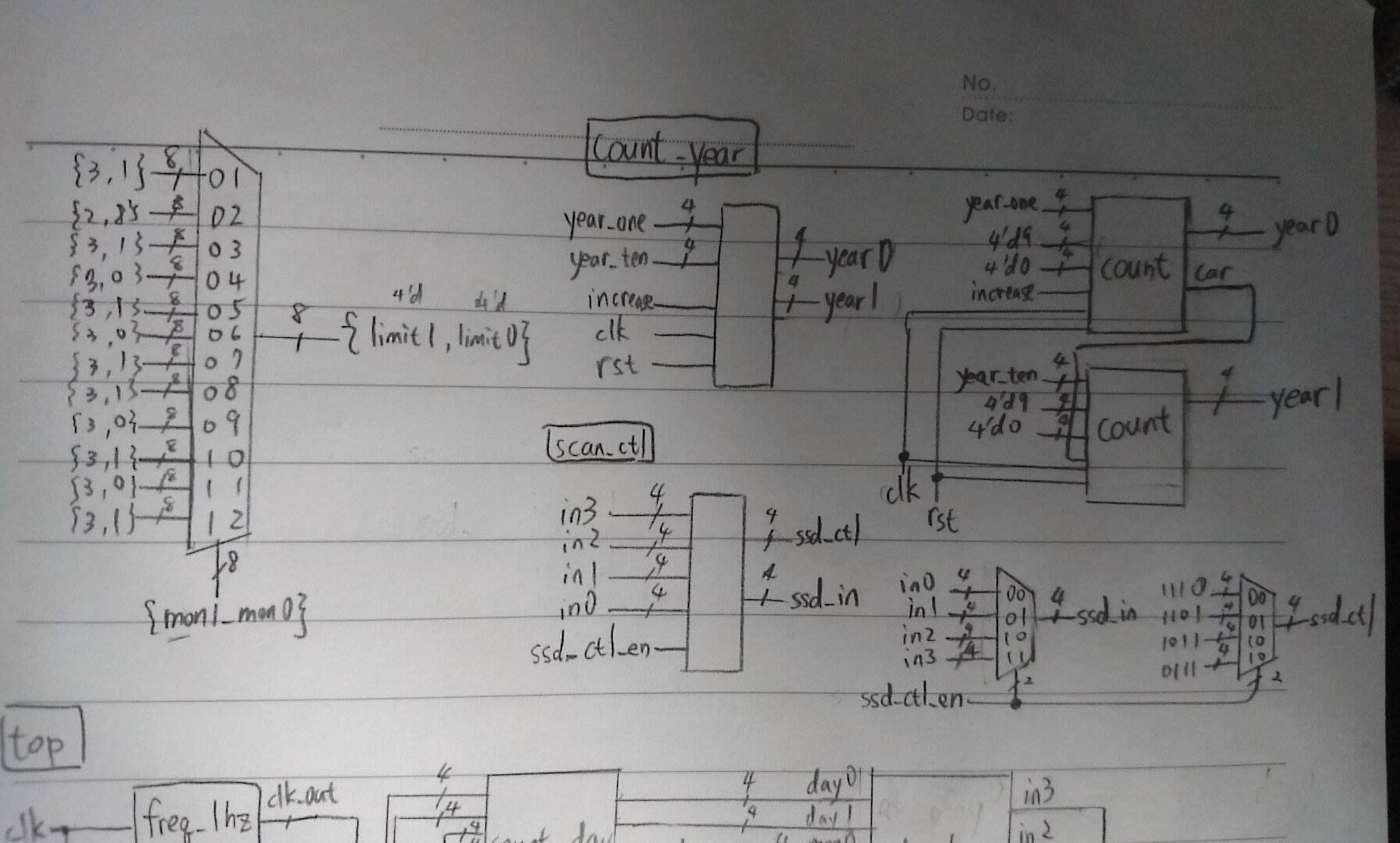
1. count\_mon：

此module是用來做月份的顯示，還有給count\_day上界的limit1(十位數的)和limit0(個位數的)。

而個位數的上界(ub)下界(lb)分別是用(mon1==4’d1)以及((mon1==4’d1) && (mon0==4’d2))來作判別給值。

此接出去的carry會作為count\_year的increase。

* 個位數的carry即為十位數的increase，由於carry是在作暫存值時就給值了，因此當個位數要進位時，個位數以及十為數會同時變化。

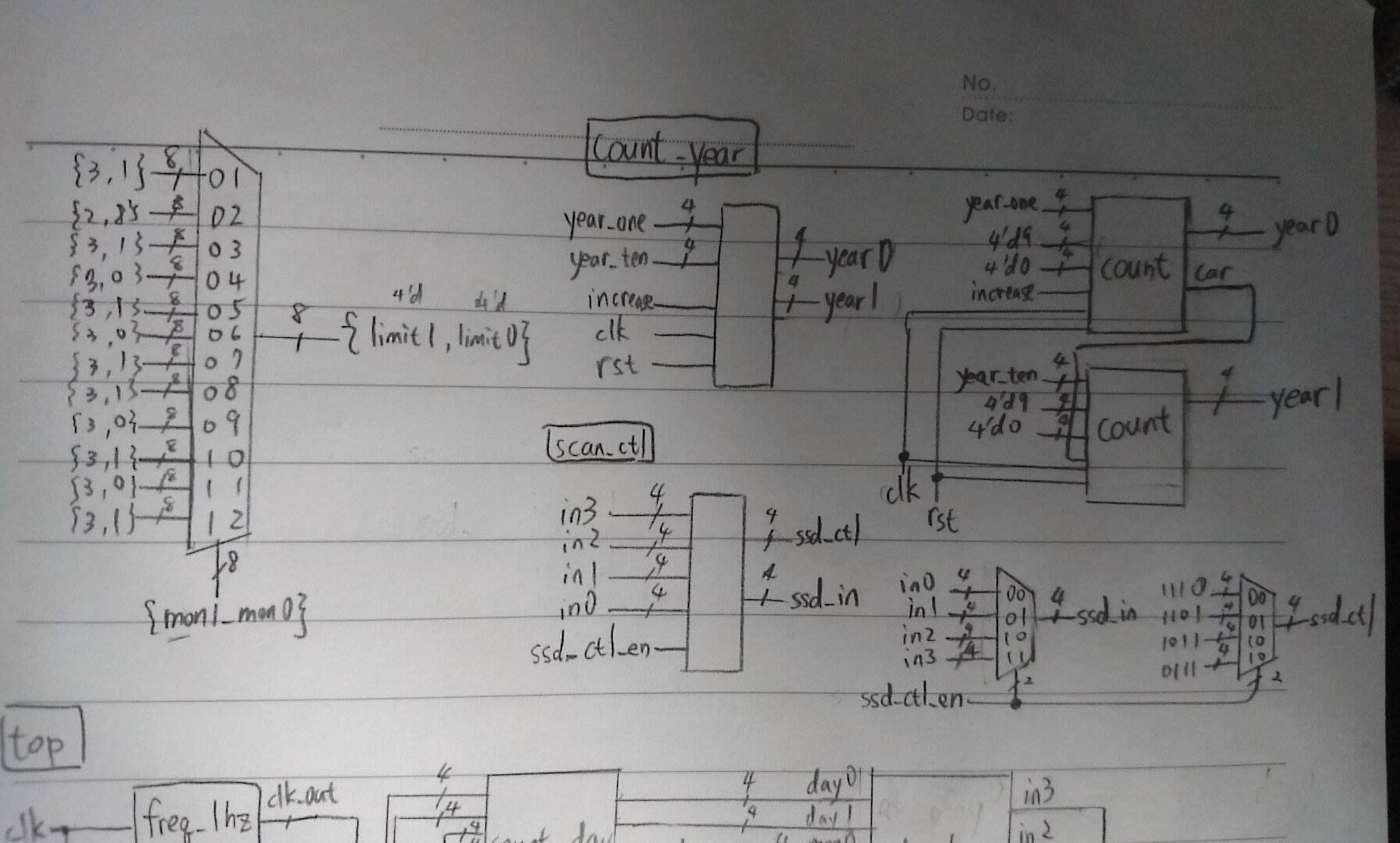
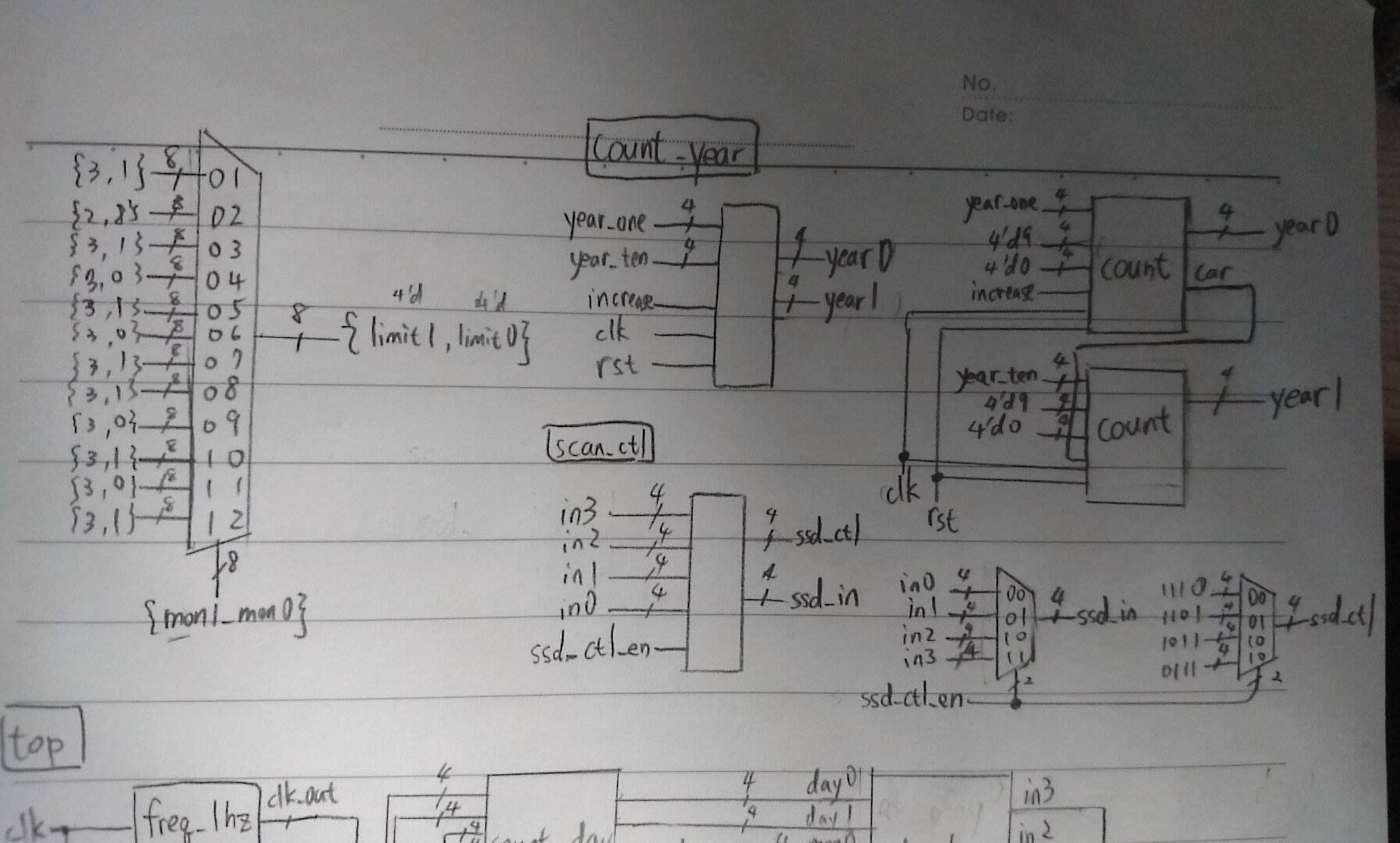


1. count\_year：

此module是用來做年份的顯示(00~99)。

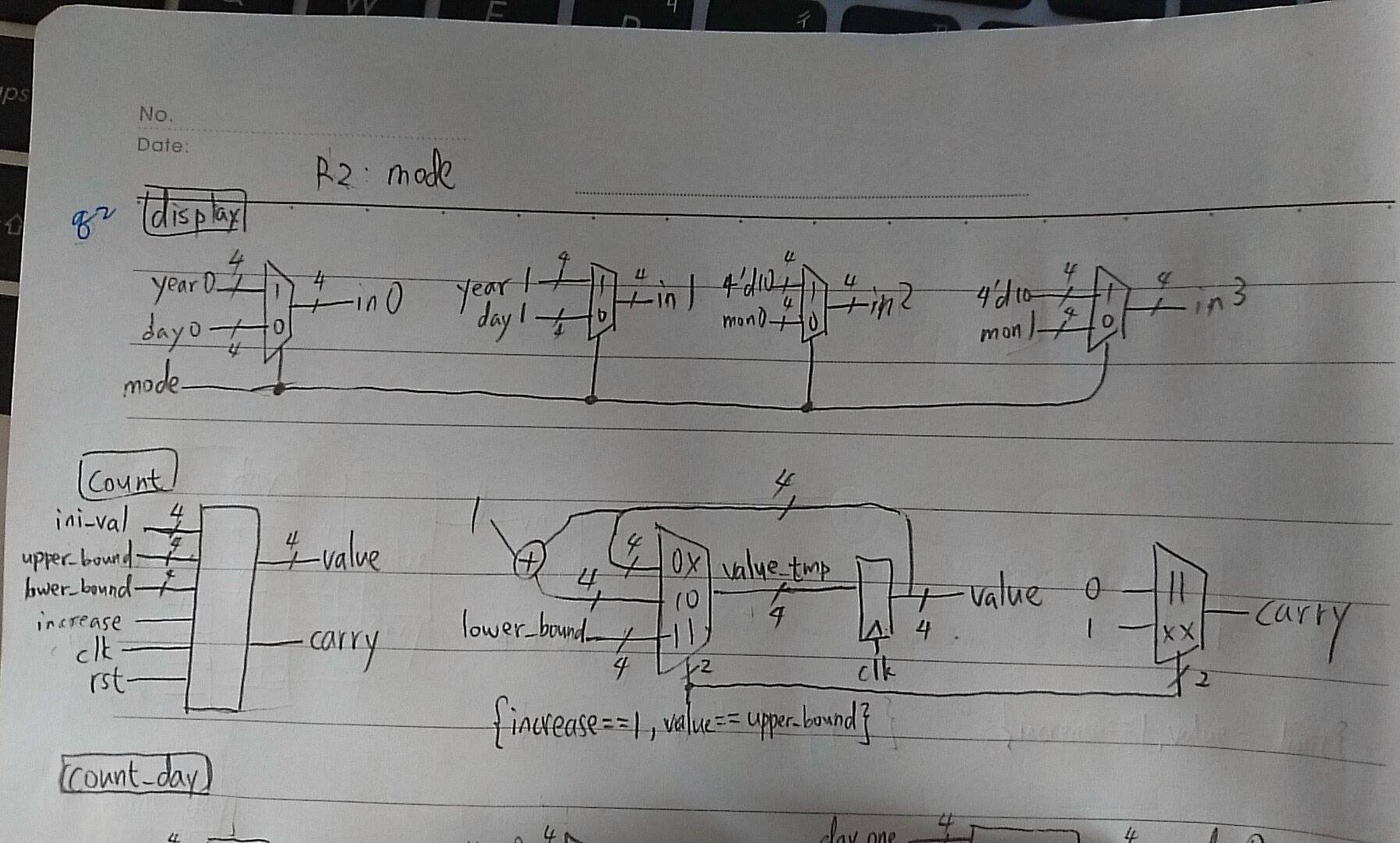
由於年份的十位數進位在top內沒有意義，因此就不把處理十位數的carry接出來。

* 個位數的carry即為十位數的increase，由於carry是在作暫存值時就給值了，因此當個位數要進位時，個位數以及十為數會同時變化。

1. display：

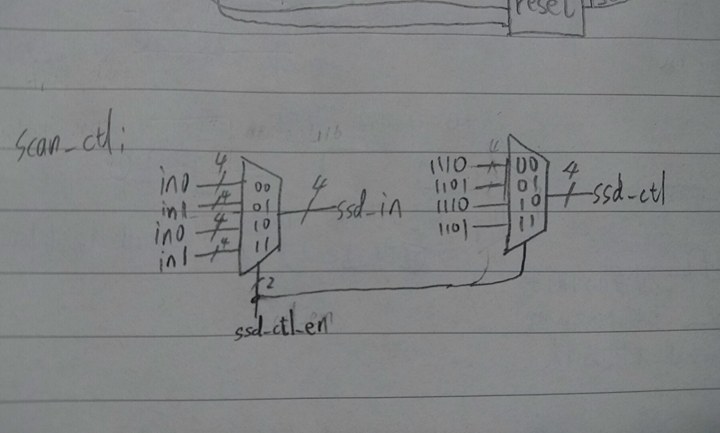
此module是用來決定scan\_ctl的input :in3, in2,in1,in0。



1. scan\_ctl：

此module是讓七段顯示器同時顯示不同位數的module。

ssd\_ctl\_en(from freq\_1hz的clk\_ctl[2:0])的用意為顯示兩個不同的數，因為七段顯示器一次只能顯示一種數次，因此需要一個頻率介於內建時間到1hz的ssd\_ctl\_en產生同時顯示兩個數的錯覺。



1. ssd：

當bcd 為 4'd0: segs = 8'b00000011; //七段顯示器顯示”0”

當bcd 為 4'd1: segs = 8'b10011111; //七段顯示器顯示”1”

當bcd 為 4'd2: segs = 8'b00100101; //七段顯示器顯示”2”

當bcd 為 4'd3: segs = 8'b00001101; //七段顯示器顯示”3”

當bcd 為 4'd4: segs = 8'b10011001; //七段顯示器顯示”4”

當bcd 為 4'd5: segs = 8'b01001001; //七段顯示器顯示”5”

當bcd 為 4'd6: segs = 8'b01000001; //七段顯示器顯示”6”

當bcd 為 4'd7: segs = 8'b00011111; //七段顯示器顯示”7”

當bcd 為 4'd8: segs = 8'b00000001; //七段顯示器顯示”8”

當bcd 為 4'd9: segs = 8'b00001001; //七段顯示器顯示”9”

當bcd 為4'd10:segs=8'b11111111; //七段顯示器顯示” “

default: segs = 8'b00000000; //七段顯示器顯示”8.”

Result

1. 圖一、二：當V17=1，表示rst=1 🡪顯示初始值。

圖一：mode=0 🡪顯示mon/day的初始值1128

圖二：mode=1🡪縣示year的初始值00

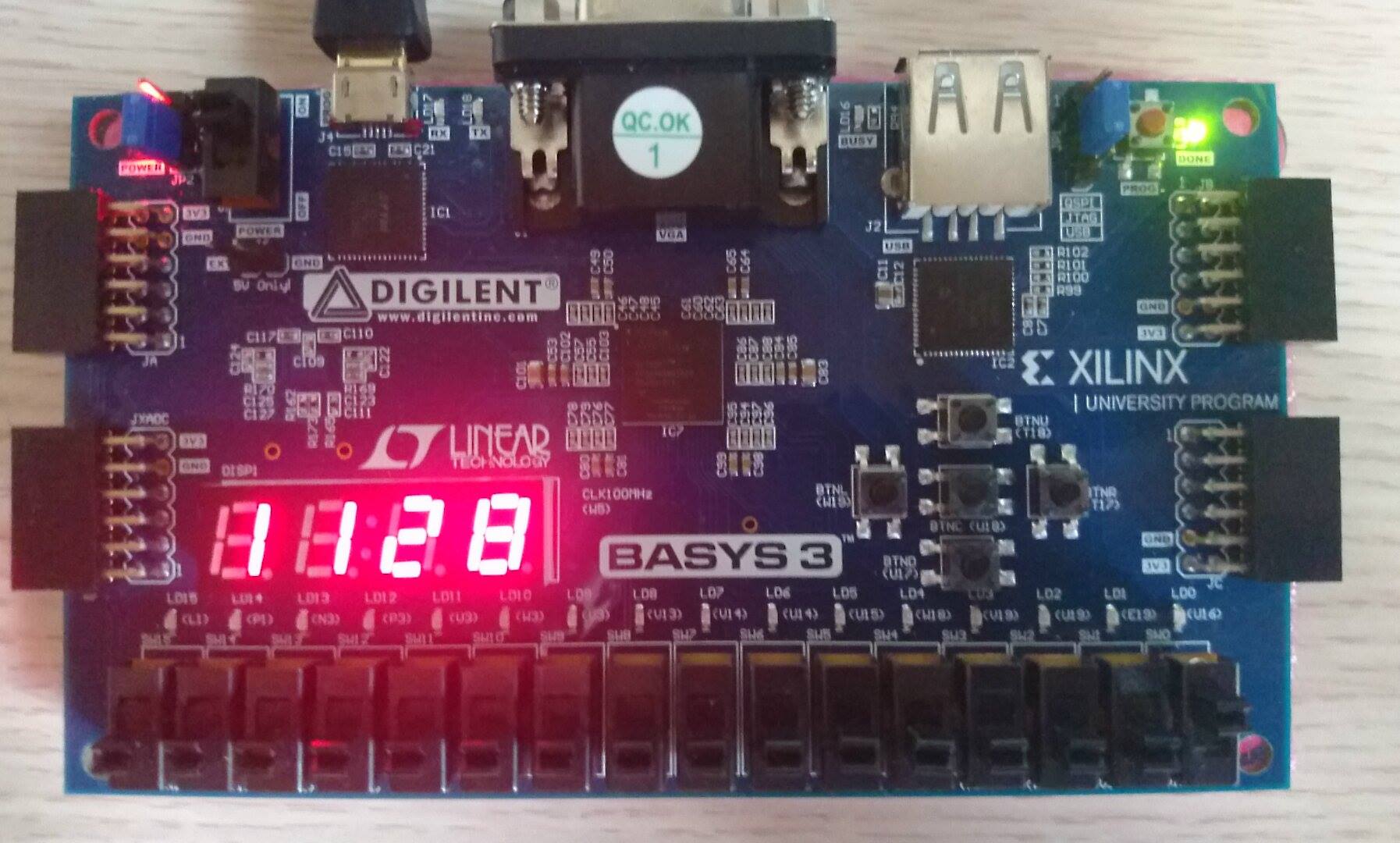
1. 圖三、四：當V17=0，表示rst=0。

圖三目前數到的日期為12/31，下一刻會變成01/01，而年份也會加一年(圖四)

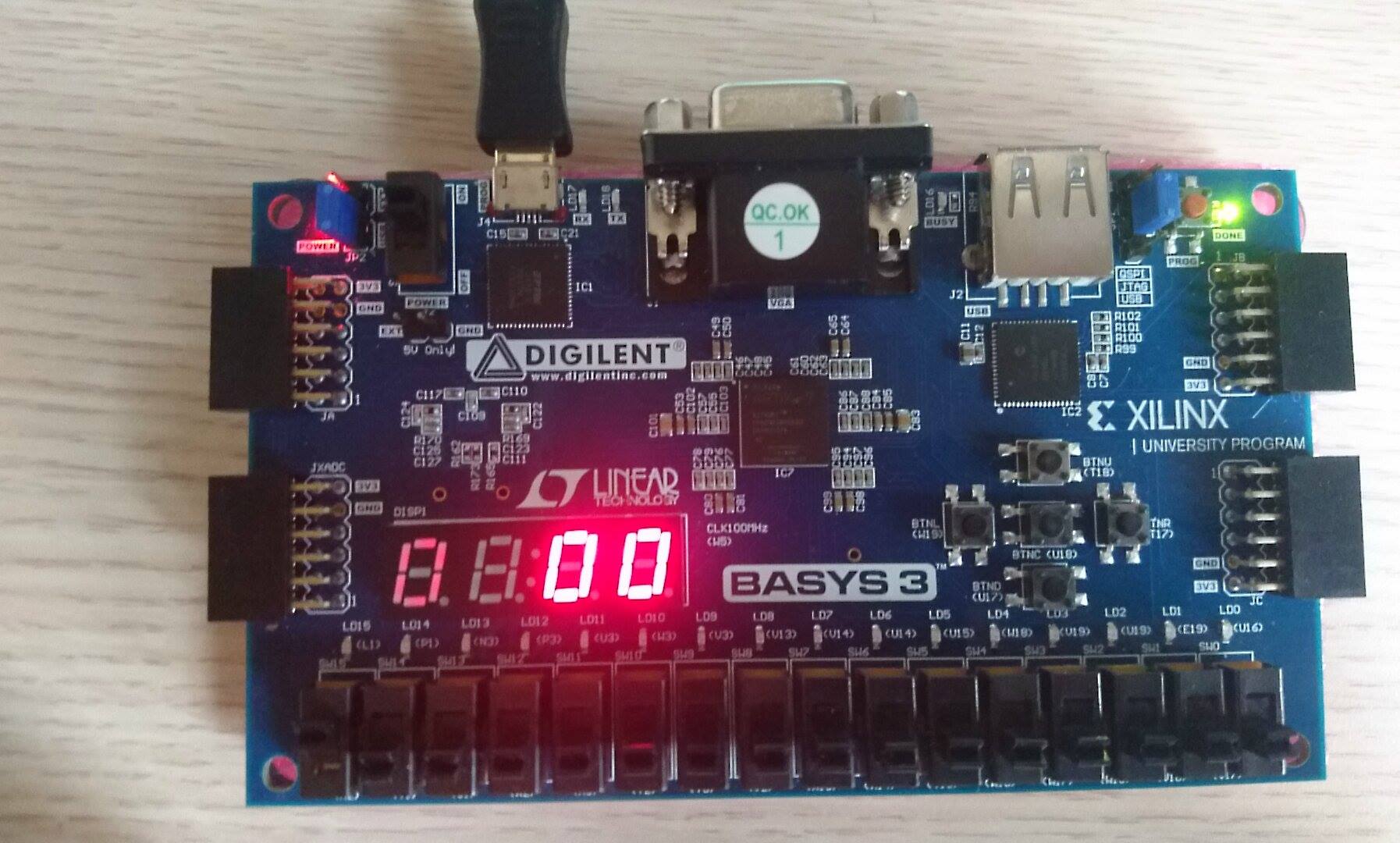
1. 圖五、六 ：當V17=0，表示rst=0。

圖五目前數到的日期為02/28，下一刻會變成03/01(圖六)

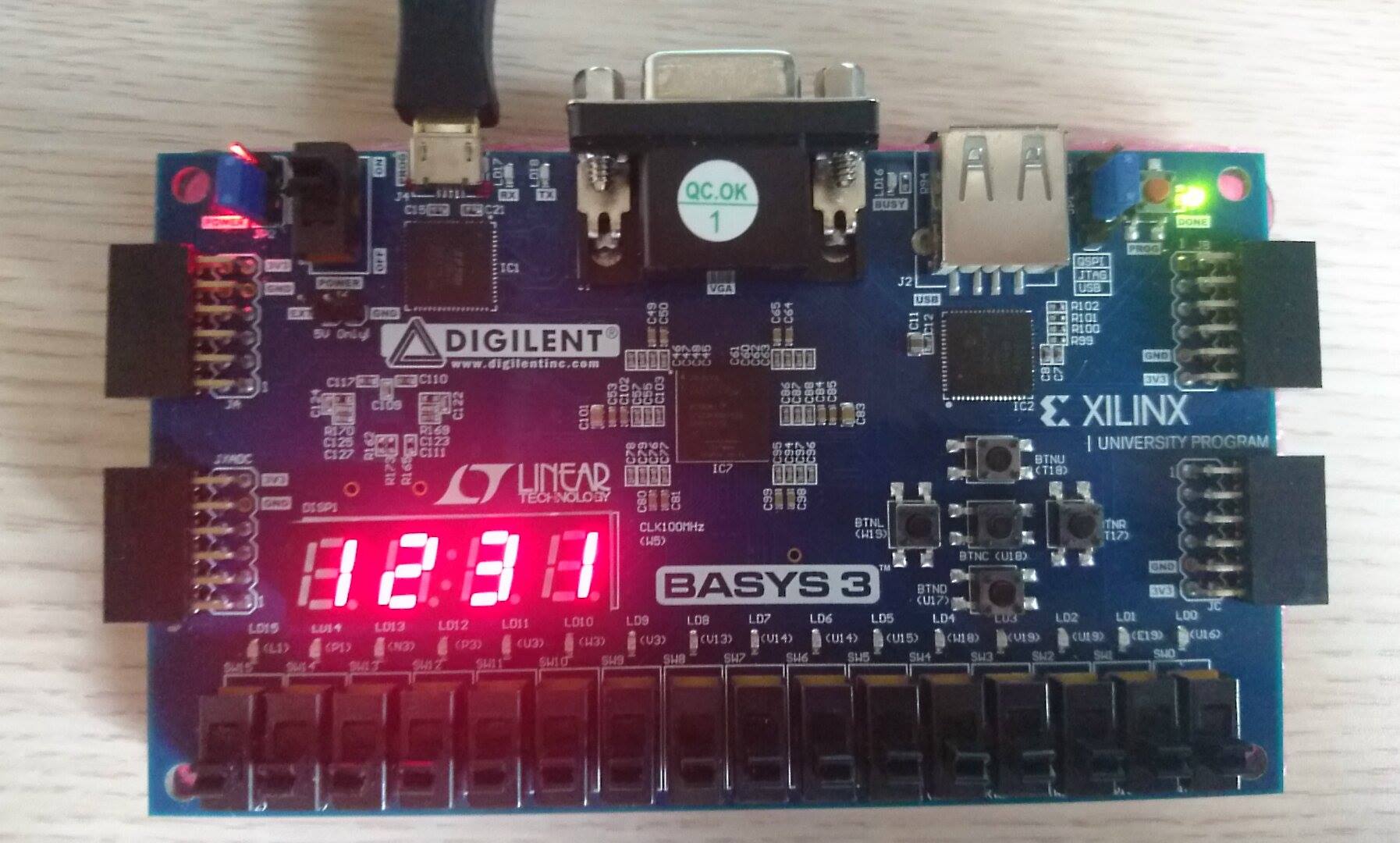
圖一



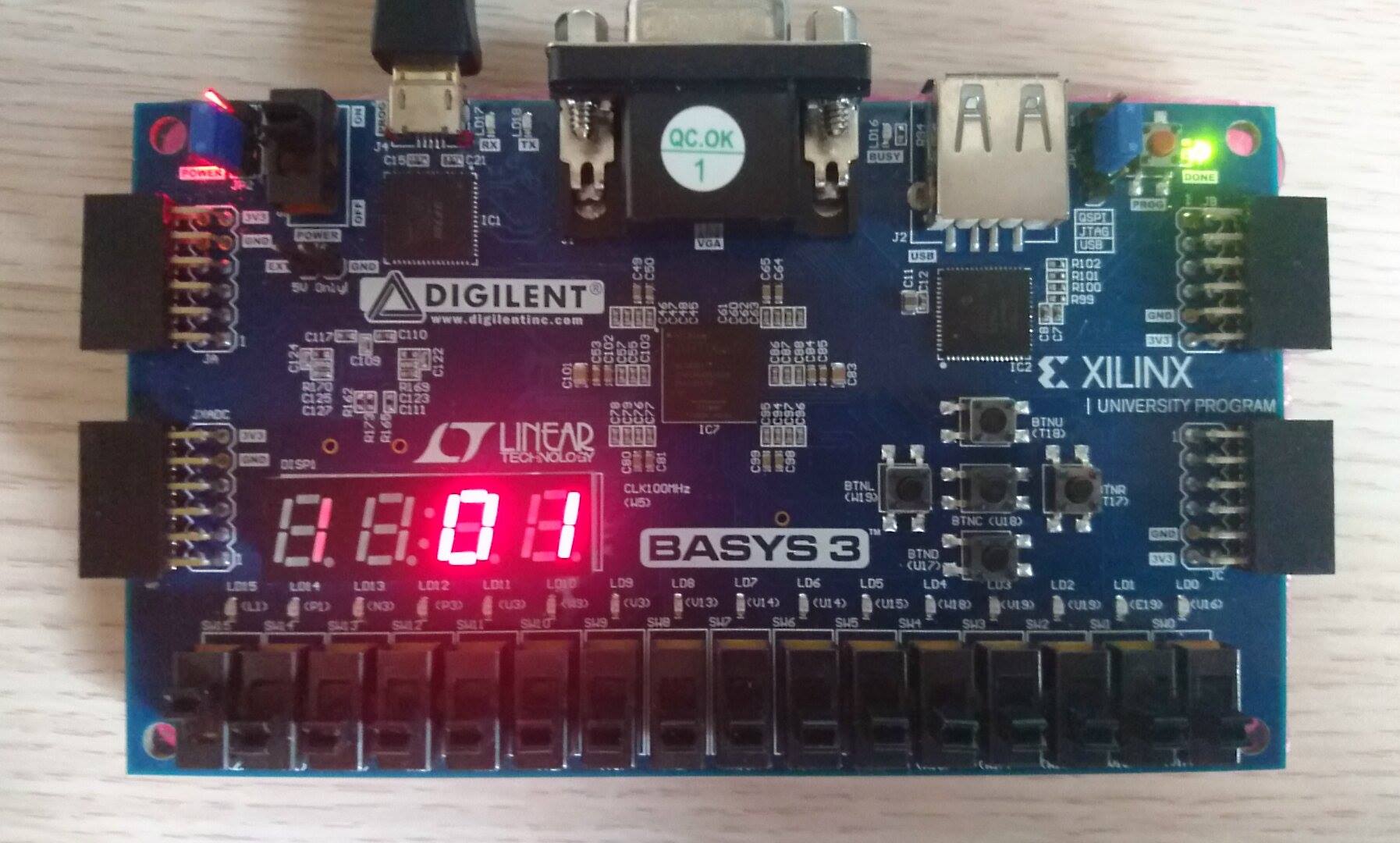
圖二



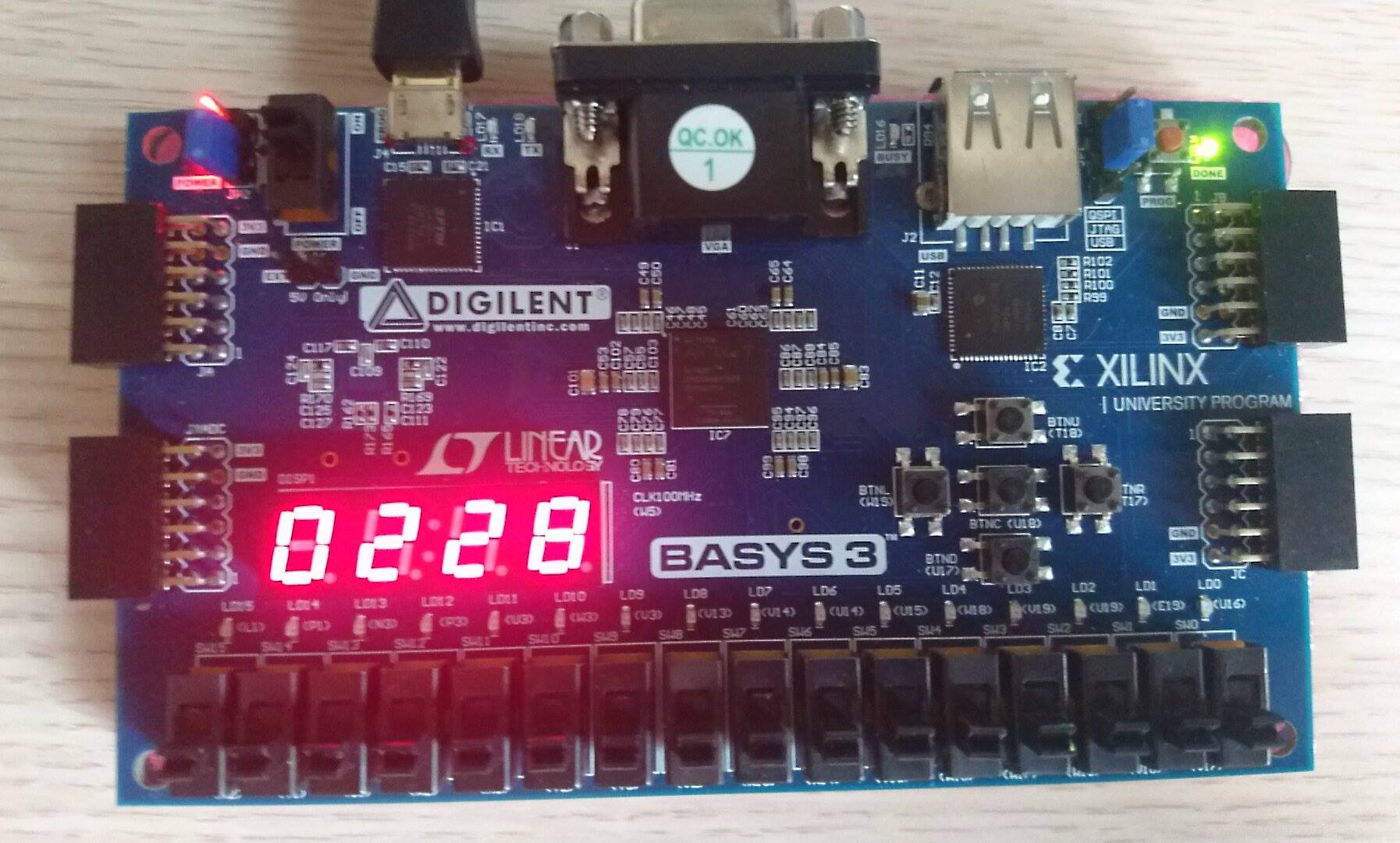
圖三



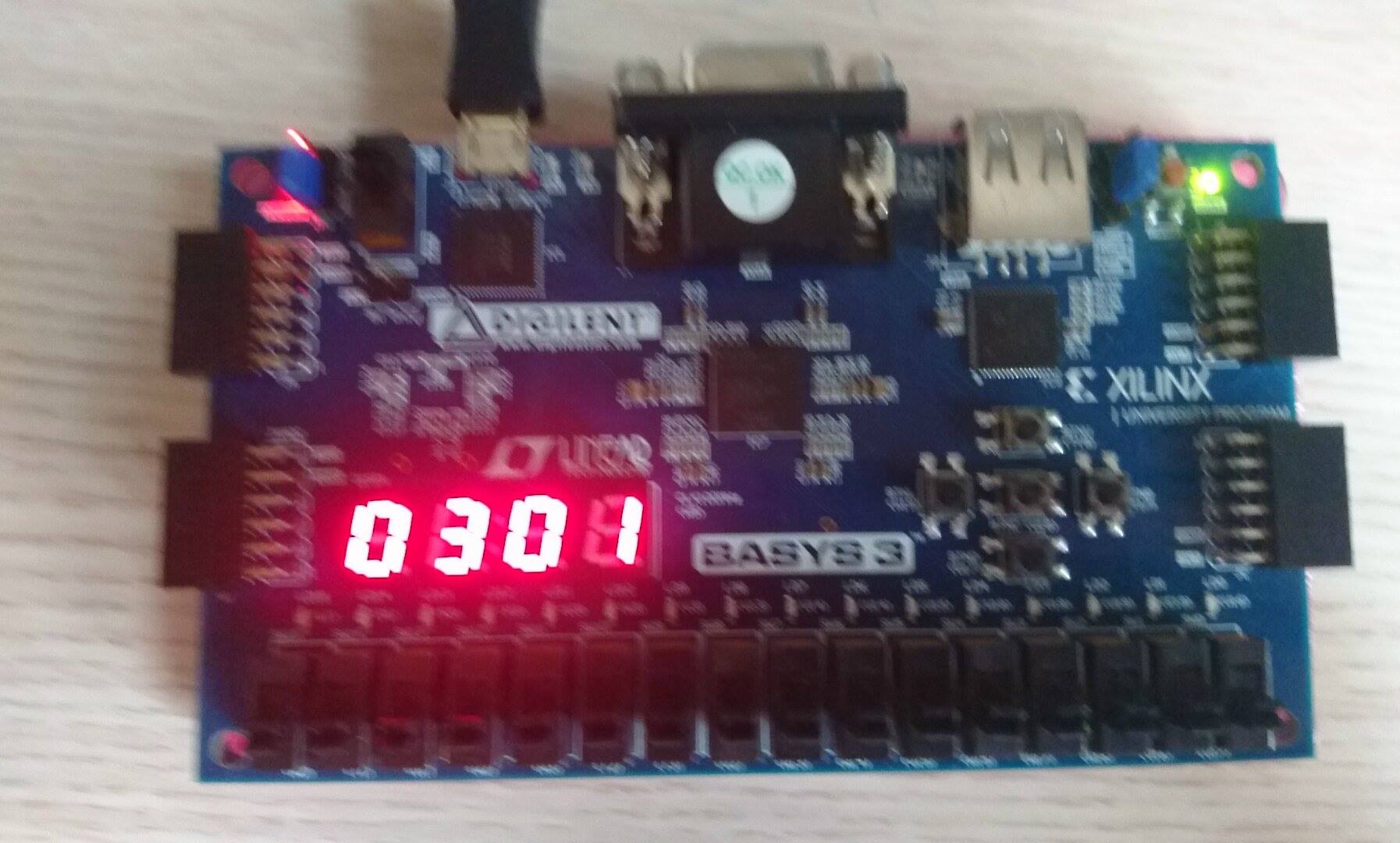
圖四



圖五



圖六



Discussion

1. 在做count時特地接upper\_bound和lower\_bound，這樣子就比較容易count有辦法做像00🡪12的循環加法。
2. 在做count\_day,count\_mon和count\_year時，個位數的carry即為十位數的increase，由於carry是在作暫存值時就給值了，因此當個位數要進位時，個位數以及十為數會同時變化。
3. display在in3以及in2上有用到4’d10是因為我在ssd內設4'd10:segs=8'b11111111，因此當要顯示的數是10，即其效果就像在七段顯示器上不會顯示任何圖案一樣。

**Conclusion**：

這次的lab讓我們做出一個時鐘，真的是超級猛的，而且也更新了我的count變成更有邏輯性的樣子，真的學到很多。