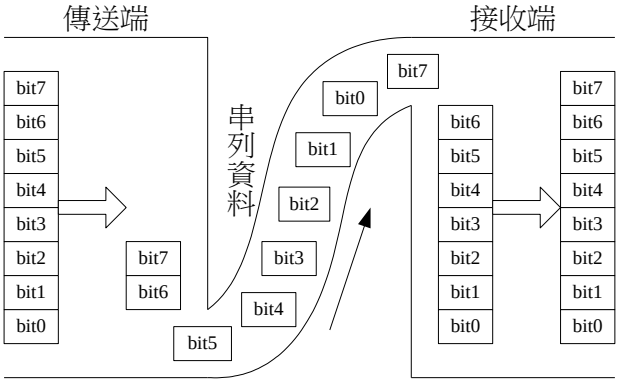
Lab 6　UART 實習

銘傳大學電腦與通訊工程學系

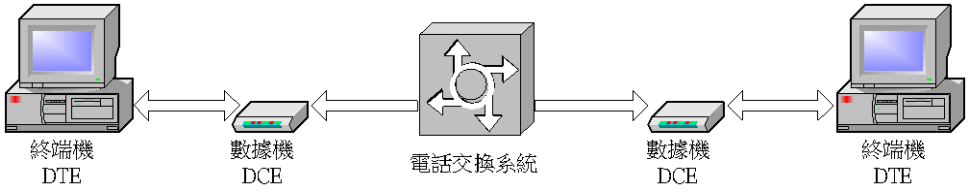
陳慶逸

一、背景知識

資料的傳輸方式可分為並列傳輸方式與串列傳輸方式，並列傳輸就是在同一時間內以數個位元為一個轉移單位的傳輸方式。這裡的數個位元一般是指 8 位元為一個單位。並列傳輸方式可以取得較大的資料傳輸寬度而且速度也較快，可是所花費的成本較高，適合於近距離的傳輸；所謂串列傳輸就是指傳輸的資料線只有一位元，將資料以一個位元接一個位元的方式傳送，接收到時再組合起來。而串列傳輸方式的特性恰好與並列傳輸方式相反。在成本的考量上，可以使用串列傳輸方式。下圖為串列傳輸示意圖。



為了要讓各家的電腦能夠互傳資料，所以必須訂定串列傳輸標準，在電話通信網路系統中，電腦這類的終端機(Data Terminal Equipment，DTE)與數據機(Data Communication Equipment，DCE)之間的數位介面通訊協定，由美國電機工業協會(EIA)與國際電報資詢委員會(CCITT)於 1969 年制定了 RS-232 介面標準。至今，RS-232 串列傳輸介面仍然是工業界常用的通訊介面之一。



RS-232 串列傳輸介面的接頭有 DB9 與 DB25 兩種，下表為 RS-232 的 DB9 與 DB25 腳位對照表，RS-232 串列傳輸主要是利用 RXD 與 TXD 來作為資料的接收與傳送線，以下為 RS-232 DB9 與 DB25 的腳位對照表以及連接頭。

表 9-1　RS-232 DB9 與 DB25 腳位對照表

信號名稱	DB9	DB25	信號說明
CD	1	8	載波偵測信號
RXD	2	3	資料接收端
TXD	3	2	資料傳送端
/DTR	4	20	終端機資料備妥信號
GND	5	7	接地端
/DSR	6	6	數據機資料備妥回應
/RTS	7	4	終端機要求傳送信號
/CTS	8	5	數據機傳送回應信號
RI	9	22	振鈴指示訊號

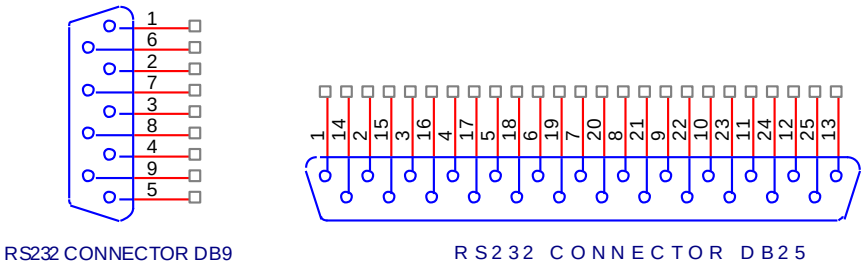


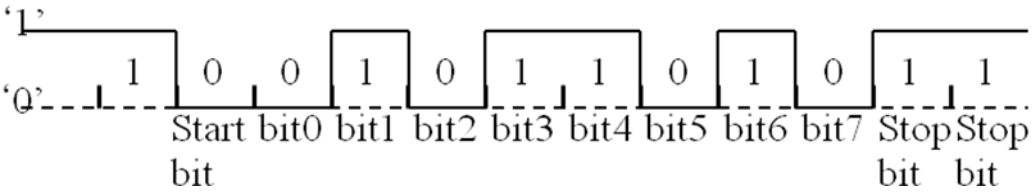
圖 9-3　RS-232 DB9 與 DB25 連接頭

了解到腳位功能之後，我們可以知道在兩台個人電腦(或裝置)之間的 RS-232 傳輸時，必須使用跳線，也就是說，第一台電腦(或裝置)的資料輸出腳位接至第二台電腦(或裝置)的資料輸入腳位，反之亦然，而接地線共接，參考下圖。

串列傳輸在傳送一個位元組時，必須要傳送 8 次，而 RS-232 的串列傳輸方式是在傳送 8 個位元資料之前加上一個起始位元，並在傳送 8 個位元資料之後加上一個停止位元，於是原先傳送一個位元組要傳送 8 次就增為 10 次。以下是 RS-232 串列傳輸的示意圖，傳輸時間順序由左至右，在 RS-232 的傳輸結構中，起始位元固定為 0，停止位元固定為 1，所以接收端的動作是一直不斷的檢查傳輸線的狀態。當傳輸線上的信號一直為 1 就表示沒有資料傳送；當傳輸線上的信號由 1 變為 0，即表示有資料將傳送，接收端就會開始準備接收 8 個位元資料，直到傳送完 8 個位元資料，

傳送端最後會送出停止位元，並使傳輸線的信號保持為 1，以等待下一次的資料傳輸。經由增加起始位元與停止位元方式，雖然會使串列傳輸效率更降低，但可解決位元資料傳輸的起始與停止之問題。另一串列傳輸協定為傳輸速度，通常以鮑率(Buat Rate)，即每秒傳輸的位元數來衡量，一般 RS-232 常使用的鮑率有 1200、2400、4800、9600 及 19200 等。兩種裝置在進行串列傳輸時，必須決定以何種鮑率來進行資料傳輸，當兩種裝置使用同一鮑率才能確保資料傳輸正確無誤。

以下資料串代表 ASCII 碼的 'Z' = 01011010B



1. UART 暫存器定義:

SOPC Builder 中提供一個 UART 的 IP Core，這個 UART Core 可以與 Nios CPU 相接。 Nios UART 暫存器定義如下:

Table 29. UART Register Map

A2..A0	Register Name	R/W	Description/Register Bits															
			15	. . .	12	11	10	9	8	7	6	5	4	3	2	1	0
0	rxdata	RO	RxData															
1	txdata	WO	TxData															
2	status[1]	RW			eop	cts	dcts	–	e[2]	rrdy	trdy	tmt	toe	roe	brk	fe	pe
3	control	RW			ieop	rts	idcts	trbk	ie	irrdy	itrdy	itmt	itoe	iroe	ibrk	ife	ipe
4	divisor	RW	Baud Rate Divisor (optional)															
5	endofpacket	RW	End-packet value															

Notes
(1) A write operation to the `status` register clears the `dcts`, `e`, `toe`, `roe`, `brk`, `fe`, and `pe` bits.
(2) `status` register bit 8 (`e`) is the logical OR of the `toe`, `roe`, `brk`, `fe`, and `pe` bits.

1-1 rxdata 與 txdata 暫存器

(1)當 UART 接收完一個資料時，會將接收到的位元組放在 rxdata 暫存器中(Read only)，以便處理器讀取。

(2)當 Nios II 處理器要發送資料到 UART 埠時，要先將位元組放入 txdata 暫存器中(Write only)，由 UART 發送出去。

Table 29. UART Register Map

A2..A0	Register Name	R/W	Description/Register Bits															
			15	. . .	12	11	10	9	8	7	6	5	4	3	2	1	0
0	rxdata	RO								RxData							
1	txdata	WO								TxData							
2	status[1]	RW			eop	cts	dcts	–	e^2	rrdy	trdy	tmt	toe	roe	brk	fe	pe
3	control	RW			ieop	rts	idcts	trbk	ie	irrdy	itrdy	itmt	itoe	iroe	ibrk	ife	ipe
4	divisor	RW	Baud Rate Divisor (optional)														
5	endofpacket	RW								End-packet value							

Notes
(1) A write operation to the `status` register clears the `dcts`, `e`, `toe`, `roe`, `brk`, `fe`, and `pe` bits.
(2) `status` register bit 8 (`e`) is the logical OR of the `toe`, `roe`, `brk`, `fe`, and `pe` bits.

1-2 status 狀態暫存器

(1) TRDY 狀態位元(R): TRDY 位元表示了 txdata 暫存器的當前狀態，當 txdata 暫存器為空(即 txdata 中的資料已經送入發送端移位暫存器)時，它準備接收新資料且 TRDY = 1；當 txdata 暫存器為滿時，TRDY 為 0。Avalon 主控制器在將新資料寫入 txdata 暫存器前，必須等待 TRDY 變為 1。

(2) RRDY 狀態位元(R): RRDY 位元表示了 rxdata 暫存器的當前狀態。當 rxdata 暫存器為空(即 UART 接收端未收到任何新資料時)，RRDY 位元為 0。而當一個接收到的新資料送入 rxdata 暫存器時，RRDY 位元為 1。Avalon 主控制器在讀 rxdata 暫存器之前，必須等待 RRDY 變為 1。

(3) e^2 位元: e^2 位元是其他幾個狀態位元(toe, roe, brk, fe 和 pe)的邏輯或結果，使用者可檢查 e^2 狀態位元來判斷 UART 模組工作時是否出現異常的情況。

Table 29. UART Register Map

A2..A0	Register Name	R/W	Description/Register Bits															
			15	. . .	12	11	10	9	8	7	6	5	4	3	2	1	0
0	rxdata	RO								RxData							
1	txdata	WO								TxData							
2	status[1]	RW			eop	cts	dcts	–	e^2	rrdy	trdy	tmt	toe	roe	brk	fe	pe
3	control	RW			ieop	rts	idcts	trbk	ie	irrdy	itrdy	itmt	itoe	iroe	ibrk	ife	ipe
4	divisor	RW	Baud Rate Divisor (optional)														
5	endofpacket	RW								End-packet value							

Notes
(1) A write operation to the `status` register clears the `dcts`, `e`, `toe`, `roe`, `brk`, `fe`, and `pe` bits.
(2) `status` register bit 8 (`e`) is the logical OR of the `toe`, `roe`, `brk`, `fe`, and `pe` bits.

1-3 控制暫存器

(1) ITMT：傳送暫存器空中斷致能。

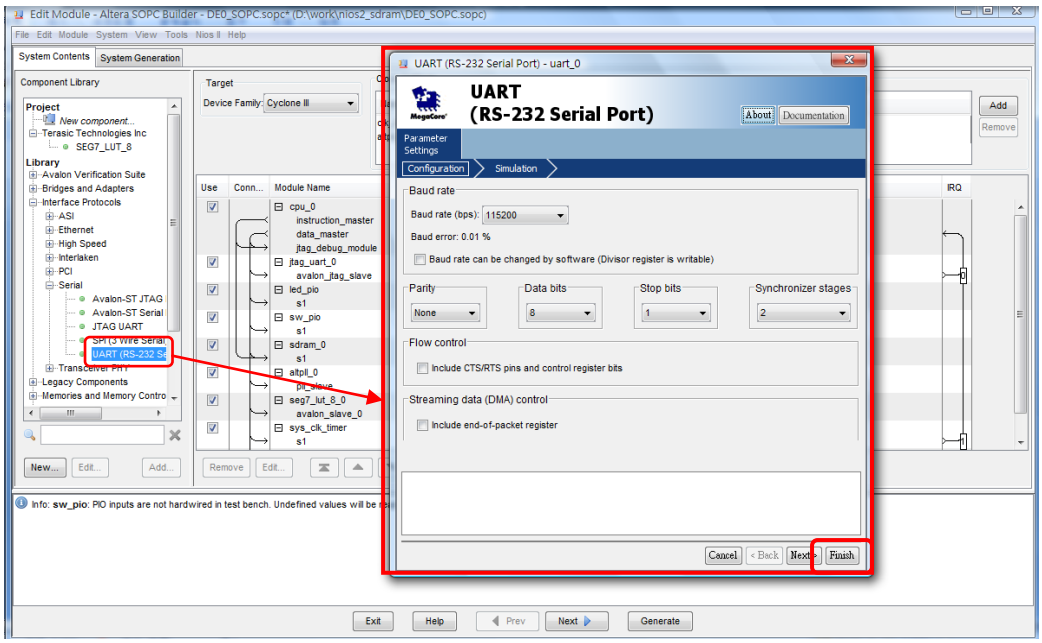
(2) ITRDY：傳送暫存器準備好中斷致能。

(3) IRRDY：接收暫存器準備好中斷致能。

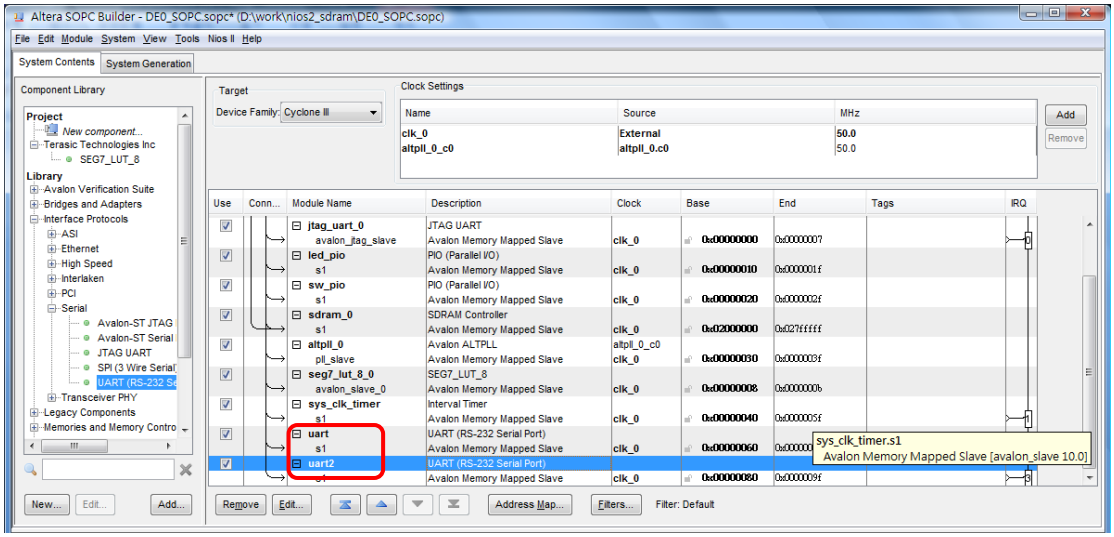
(4) IE：錯誤中斷致能。

二、硬體設計

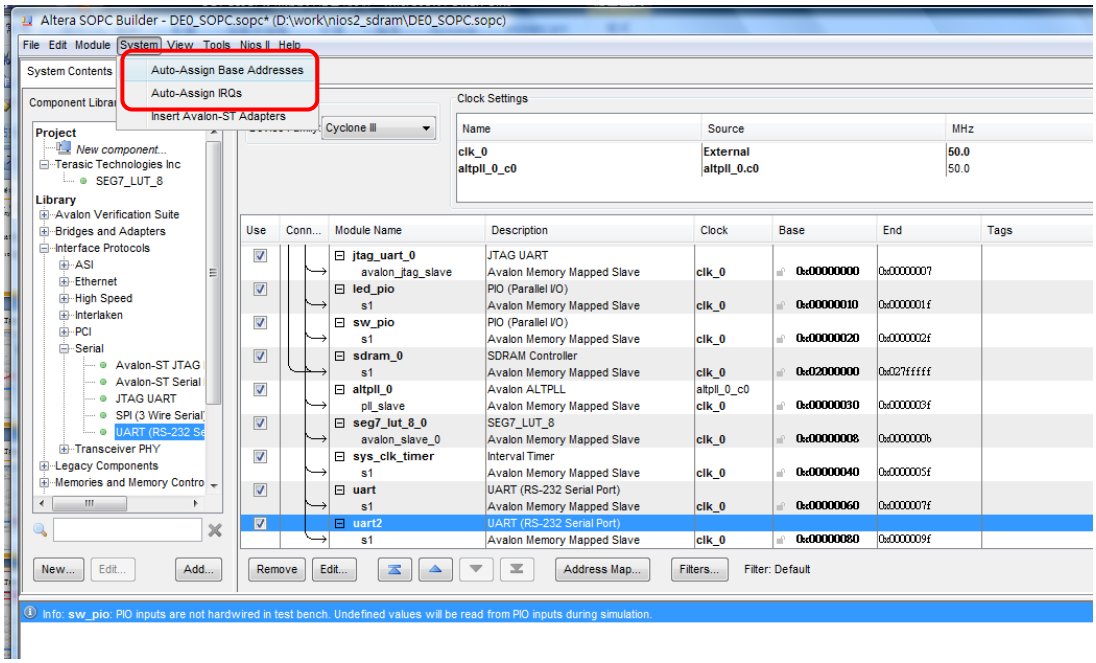
1. 在 SOPC Builder 中增加第一個 UART IP Core，重新命名為 uart。



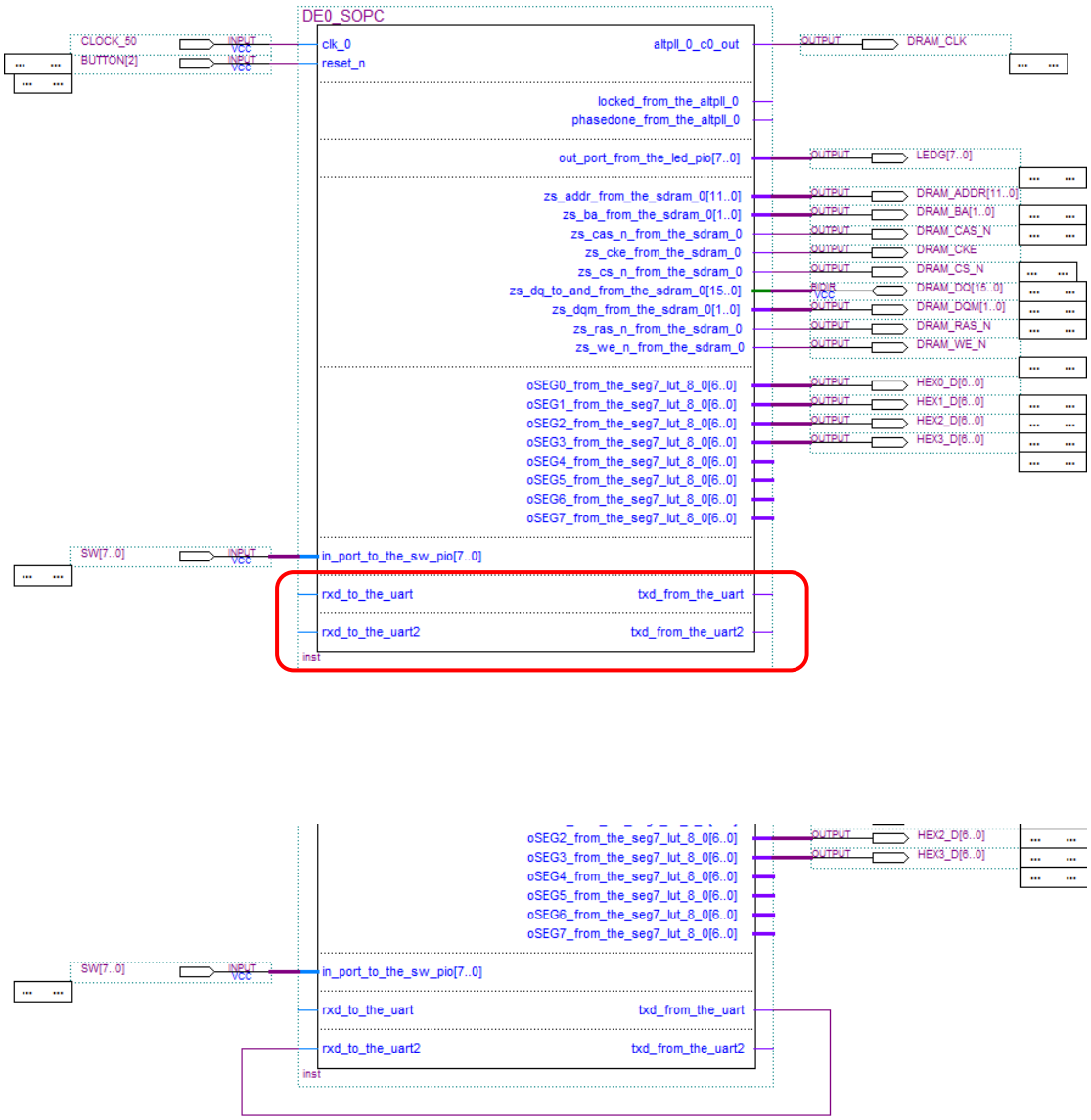
2. 依相同的程式建立第二個 UART IP Core，重新命名為 uart2。



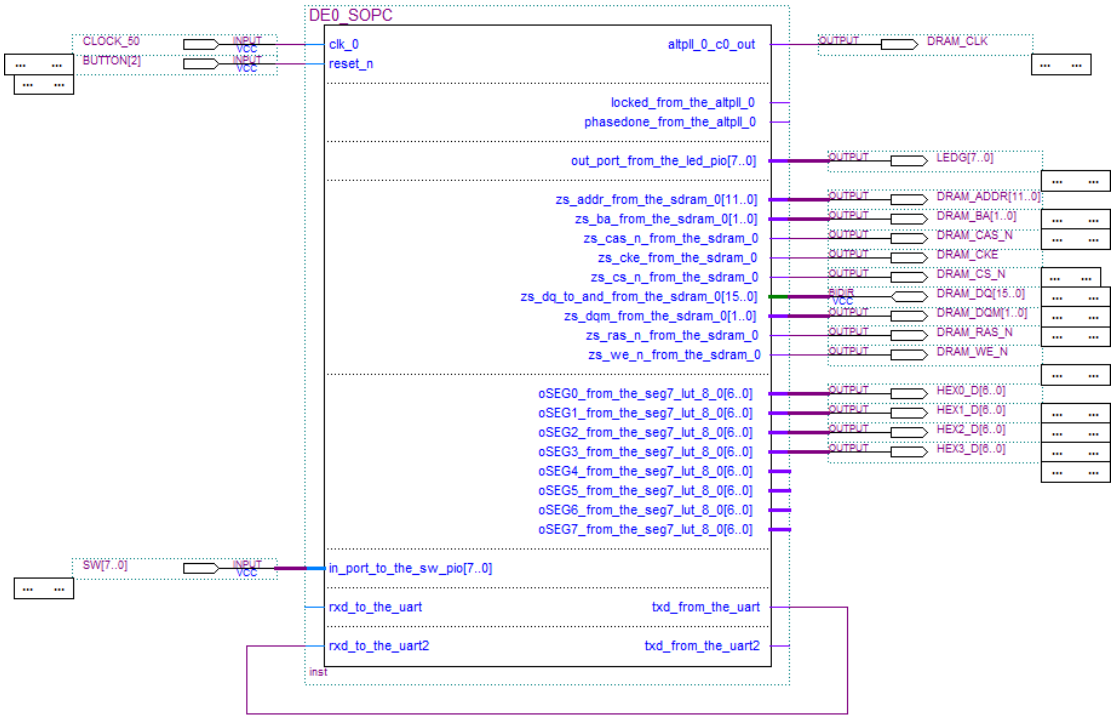
3. 若有位址或 IRQ 衝突的情況，則選擇 System/ Auto-Assign Base Addresses 和 System/ Auto-Assign IRQs 來更新之(若無則略過本步驟)。完成後點選 Generate 產生系統。



4. 更新 Nios II Symbol 後可以看到 Nios II 系統已多出兩個 UART 介面，此處我們將第一個 UART 的傳送輸連接到第二個 UART 的接收端。完成連線後點選 Start Compilation 編譯電路。


DE0_SOPC
CLOCK_50
BUTTON[2]
INPUT
VCC
clk_0
reset_n
altpll_0_c0_out
OUTPUT
DRAM_CLK
locked_from_the_altpll_0
phasedone_from_the_altpll_0
out_port_from_the_led_pio[7..0]
LEDG[7..0]
zs_addr_from_the_sdram_0[11..0]
DRAM_ADDR[11..0]
zs_ba_from_the_sdram_0[1..0]
DRAM_BA[1..0]
zs_cas_n_from_the_sdram_0
DRAM_CAS_N
zs_cke_from_the_sdram_0
DRAM_CKE
zs_cs_n_from_the_sdram_0
DRAM_CS_N
zs_dq_to_and_from_the_sdram_0[15..0]
DRAM_DQ[15..0]
zs_dqm_from_the_sdram_0[1..0]
DRAM_DQM[1..0]
zs_ras_n_from_the_sdram_0
DRAM_RAS_N
zs_we_n_from_the_sdram_0
DRAM_WE_N
oSEG0_from_the_seg7_lut_8_0[6..0]
HEX0_D[6..0]
oSEG1_from_the_seg7_lut_8_0[6..0]
HEX1_D[6..0]
oSEG2_from_the_seg7_lut_8_0[6..0]
HEX2_D[6..0]
oSEG3_from_the_seg7_lut_8_0[6..0]
HEX3_D[6..0]
oSEG4_from_the_seg7_lut_8_0[6..0]
oSEG5_from_the_seg7_lut_8_0[6..0]
oSEG6_from_the_seg7_lut_8_0[6..0]
oSEG7_from_the_seg7_lut_8_0[6..0]
SW[7..0]
in_port_to_the_sw_pio[7..0]
rxd_to_the_uart
txd_from_the_uart
rxd_to_the_uart2
txd_from_the_uart2
inst

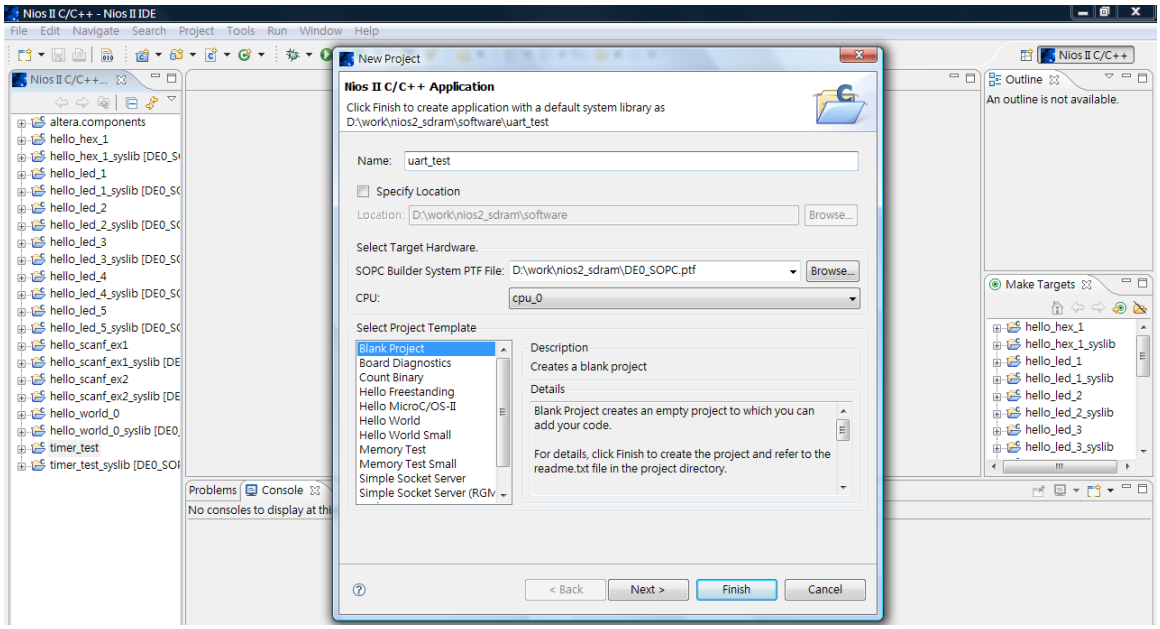
oSEG2_from_the_seg7_lut_8_0[6..0]
HEX2_D[6..0]
oSEG3_from_the_seg7_lut_8_0[6..0]
HEX3_D[6..0]
oSEG4_from_the_seg7_lut_8_0[6..0]
oSEG5_from_the_seg7_lut_8_0[6..0]
oSEG6_from_the_seg7_lut_8_0[6..0]
oSEG7_from_the_seg7_lut_8_0[6..0]
SW[7..0]
INPUT
VCC
in_port_to_the_sw_pio[7..0]
rxd_to_the_uart
txd_from_the_uart
rxd_to_the_uart2
txd_from_the_uart2
inst


DE0_SOPC
CLOCK_50
BUTTON[2]
INPUT
VCC
clk_0
reset_n
altpll_0_c0_out
OUTPUT
DRAM_CLK
locked_from_the_altpll_0
phasedone_from_the_altpll_0
out_port_from_the_led_pio[7..0]
LEDG[7..0]
zs_addr_from_the_sdram_0[11..0]
DRAM_ADDR[11..0]
zs_ba_from_the_sdram_0[1..0]
DRAM_BA[1..0]
zs_cas_n_from_the_sdram_0
DRAM_CAS_N
zs_cke_from_the_sdram_0
DRAM_CKE
zs_cs_n_from_the_sdram_0
DRAM_CS_N
zs_dq_to_and_from_the_sdram_0[15..0]
BIDIR
DRAM_DQ[15..0]
zs_dqm_from_the_sdram_0[1..0]
DRAM_DQM[1..0]
zs_ras_n_from_the_sdram_0
DRAM_RAS_N
zs_we_n_from_the_sdram_0
DRAM_WE_N
oSEG0_from_the_seg7_lut_8_0[6..0]
HEX0_D[6..0]
oSEG1_from_the_seg7_lut_8_0[6..0]
HEX1_D[6..0]
oSEG2_from_the_seg7_lut_8_0[6..0]
HEX2_D[6..0]
oSEG3_from_the_seg7_lut_8_0[6..0]
HEX3_D[6..0]
oSEG4_from_the_seg7_lut_8_0[6..0]
oSEG5_from_the_seg7_lut_8_0[6..0]
oSEG6_from_the_seg7_lut_8_0[6..0]
oSEG7_from_the_seg7_lut_8_0[6..0]
SW[7..0]
in_port_to_the_sw_pio[7..0]
rxd_to_the_uart
txd_from_the_uart
rxd_to_the_uart2
txd_from_the_uart2
inst

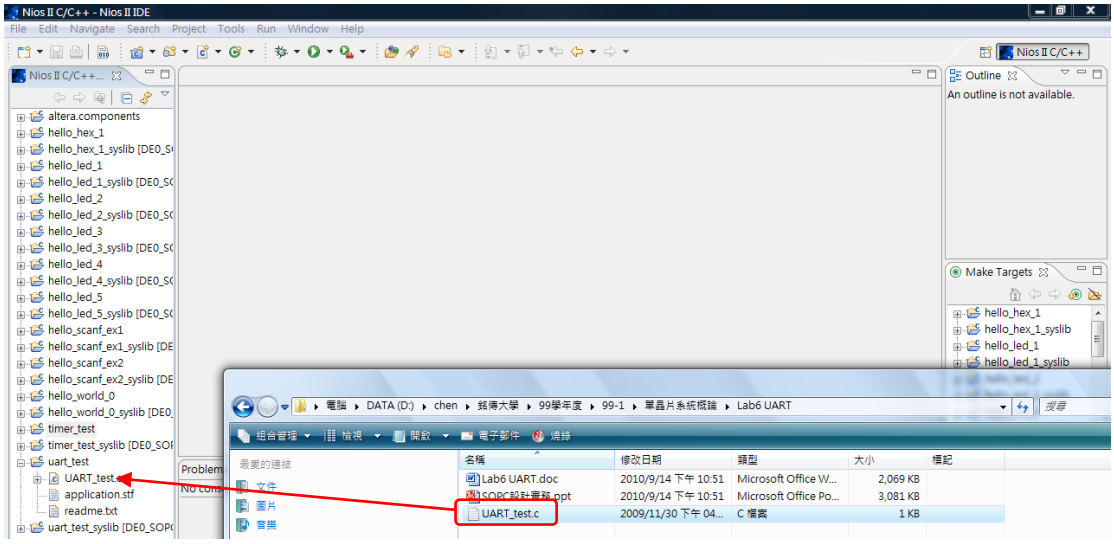
三、Nios II 軟體設計

1. 在 Nios II IDE 中建立新專案。

1-1 New/ Nios II C/C++ Application



1-2 將 UART_test.c 以滑鼠拖曳到新建的專案中。



程式內容(UART test)

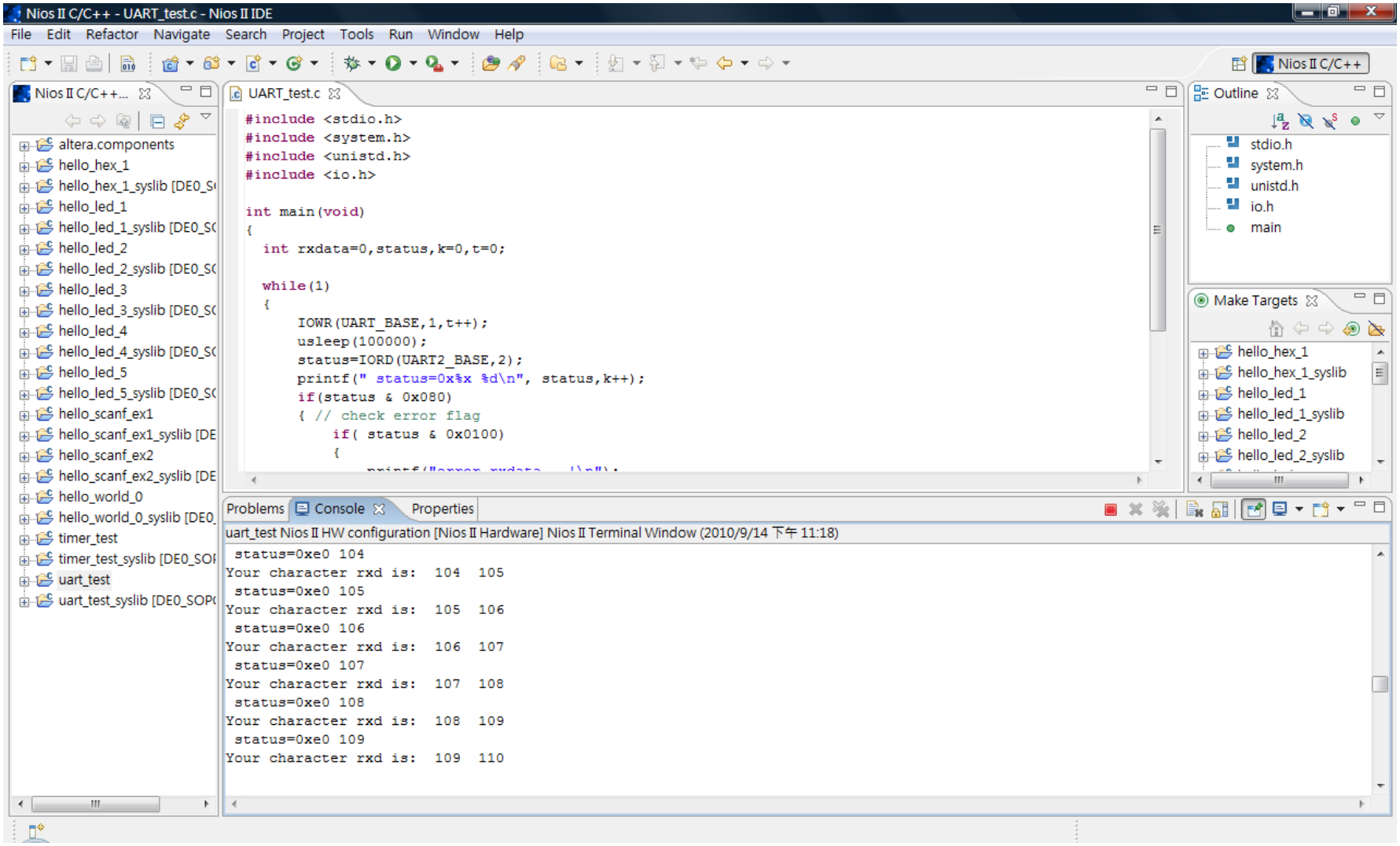
```
#include <stdio.h>
#include <system.h>
#include <unistd.h>
#include <io.h>

int main(void)
{
  int rxdata=0,status,k=0,t=0;

  while(1)
  {
     IOWR(UART_BASE,1,t++);
     usleep(100000);
     status=IORD(UART2_BASE,2);
     printf(" status=0x%x %d\n", status,k++);
     if(status & 0x080)
     { // check error flag
        if( status & 0x0100)
        {
           printf("error rxdata ..!\n");
           IOWR(UART2_BASE,2,0x00);
```

```
        } else {
            rxdata = IORD(UART2_BASE,0);
            printf("Your character rxd is:\t%x  %d\n", rxdata,k);
            IOWR(UART2_BASE,2,0x00);
        }
    }
  }
  return 0;
}
```

結果：前 64 資料無反應，第 64 筆資料之後才開始顯示接收的數值。


Nios II C/C++ - UART_test.c - Nios II IDE
File Edit Refactor Navigate Search Project Tools Run Window Help
Nios II C/C++
Nios II C/C++...
altera.components
hello_hex_1
hello_hex_1_syslib [DE0_S
hello_led_1
hello_led_1_syslib [DE0_SC
hello_led_2
hello_led_2_syslib [DE0_SC
hello_led_3
hello_led_3_syslib [DE0_SC
hello_led_4
hello_led_4_syslib [DE0_SC
hello_led_5
hello_led_5_syslib [DE0_SC
hello_scanf_ex1
hello_scanf_ex1_syslib [DE
hello_scanf_ex2
hello_scanf_ex2_syslib [DE
hello_world_0
hello_world_0_syslib [DE0_
timer_test
timer_test_syslib [DE0_SOF
uart_test
uart_test_syslib [DE0_SOPC
UART_test.c
#include <stdio.h>
#include <system.h>
#include <unistd.h>
#include <io.h>
int main(void)
{
int rxdata=0,status,k=0,t=0;
while(1)
{
IOWR(UART_BASE,1,t++);
usleep(100000);
status=IORD(UART2_BASE,2);
printf(" status=0x%x %d\n", status,k++);
if(status & 0x080)
{ // check error flag
if(status & 0x0100)
{
Outline
stdio.h
system.h
unistd.h
io.h
main
Make Targets
hello_hex_1
hello_hex_1_syslib
hello_led_1
hello_led_1_syslib
hello_led_2
hello_led_2_syslib
Problems
Console
Properties
uart_test Nios II HW configuration [Nios II Hardware] Nios II Terminal Window (2010/9/14 下午 11:18)
status=0xe0 104
Your character rxd is: 104 105
status=0xe0 105
Your character rxd is: 105 106
status=0xe0 106
Your character rxd is: 106 107
status=0xe0 107
Your character rxd is: 107 108
status=0xe0 108
Your character rxd is: 108 109
status=0xe0 109
Your character rxd is: 109 110

討論：

(1) 若要將接收字元改以整數顯示應在何處設定?

(2) UART2 的 Status register 值在未接收到的前 64 筆數值時為 0x60(RRDY=0, TRDY=1)，成功接收資料後變為 0xe0 (RRDY=1, TRDY=1)。 TRDY=1 代表傳送暫存器為空的時候，準備接收資料；當傳送暫存器滿時，TRDY=0。而 RRDY=0 時表接收暫存器空的時候，當新接收的值傳輸到接收暫存器時，RRDY=1。

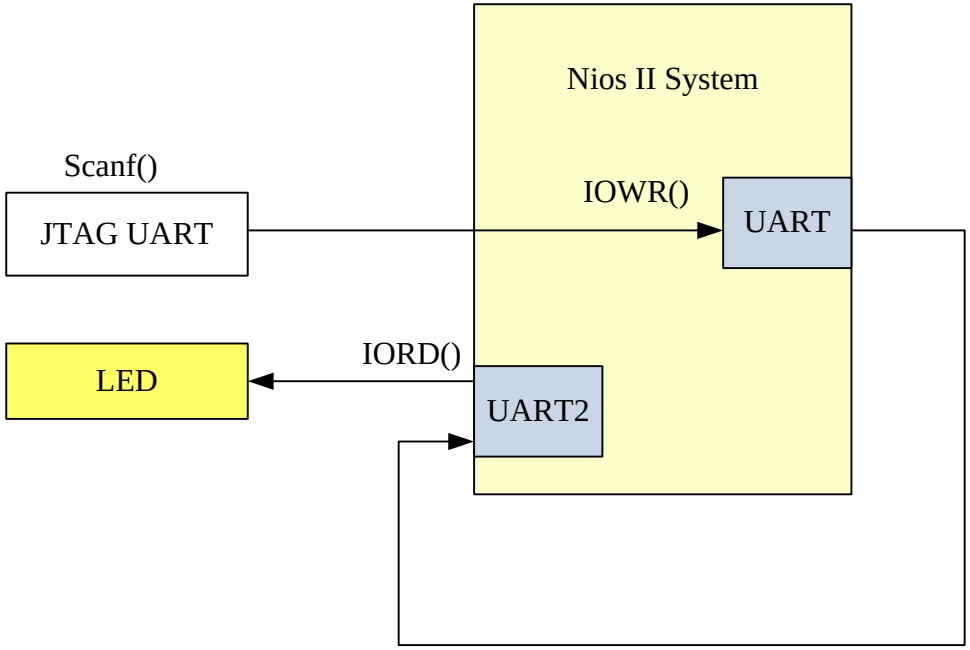
(3) Status register 中的 RRDY(Receive Character Ready)、TRDY(Transmit Ready)、TMT(Transmit Empty, 當傳送暫存器正將資料從 TXD 接腳送出時，TMT 設為 0；當傳送暫存器空閒時，TMT 為 1)不以人為方式重置(使用 IOWR(UART2_BASE,2,0x00);無法清除此三個數值)，暫存器會自動更新。因此在程式中 status register 讀出來的值為 0xe0，即使下 IOWR 指令仍無法令值變更為 0x00。

四、作業

作業一

1、題目：UART 控制 LED 亮滅

2、功能說明：利用 printf 及 scanf 指令功能，讓 Nios II 讀取鍵盤所輸入之 0~255 範圍的數值，將之寫入 UART 之傳送端，當 UART2 接收端接收到 UART 傳送端送過來的數值之後，再將之寫入 LED 的基底位址，以控制 LED 的亮滅(即利用 LED 顯示鍵盤所輸入的數值)。舉例而言，當鍵盤輸入數值為 15 時，則 LED = 0000 1111。



執行結果:

```
nios2-terminal: connected to hardware target using JTAG UART on cable
nios2-terminal: "USB-Blaster [USB-0]", device 1, instance 0
nios2-terminal: (Use the IDE stop button or Ctrl-C to terminate)
```

```
input your number : ( 0 -> 255 )
15
```

```
input your number : ( 0 -> 255 )
255
```

```
input your number : ( 0 -> 255 )
```

P.S.

取得使用者的輸入值可以使用「標準輸入」（Standard input）的 scanf()函式，並搭配格式指定字與&取址運算子指定給變數，例如：

```
#include <stdio.h>

int main(void) {
	int input;


	printf("請輸入數字：");

	scanf("%d", &input);

	printf("您輸入的數字：%d\n", input);
```

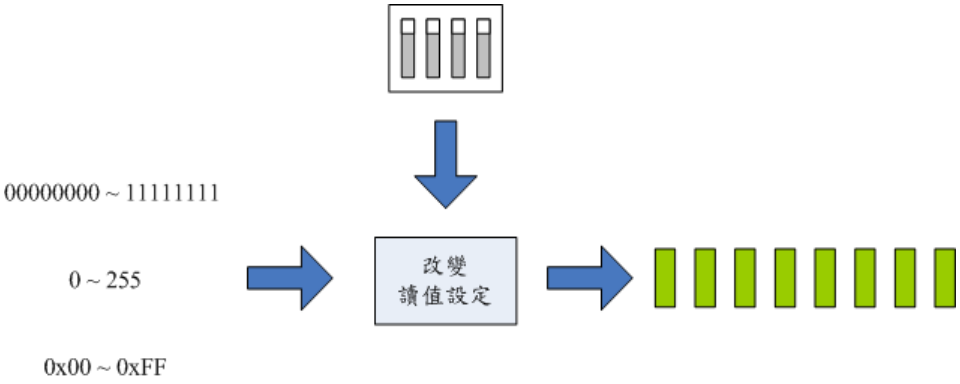
```
    return 0;
}
```

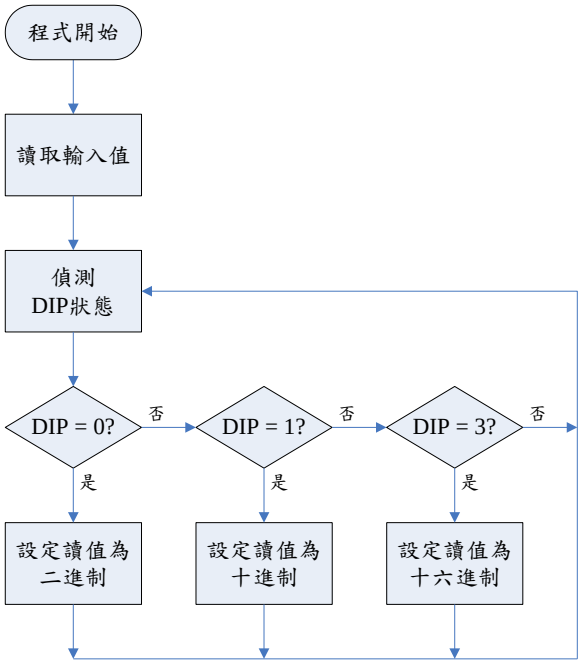
在程式中先宣告了一個整數變數 input，使用 scanf()函式時，若輸入的數值為整數，則使用格式指定字%d，若輸入的是其它資料型態，則必須使用對應的格式指定字，如果是 double，特別注意要使用%lf 來指定，而您必須告知程式儲存資料的變數位址，為此，必須使用&取址運算子，這會將變數的記憶體位址取出，則輸入的數值就知道變數的記憶體位址並儲存之。

作業二

1、利用 DE2 上的 switch 決定鍵盤輸入數值的型式(二進位、十進位或十六進位)，並以該數值控制 LED 亮滅。

2、說明：當 switch = 0 時，由鍵盤輸入 8 bit 之二進位值字串，LED 顯示相對應的輸出(ex: 鍵盤輸入 01010101，LED 顯示 01010101)；當 switch=1 時，鍵盤輸 0 ~ 255 之間的整數值，而 LED 則顯示其對應的二進位輸出(ex: 鍵盤輸入 85，LED 顯示 01010101)；當 switch=3 時，鍵盤輸入 00 -> FF 之間的十六進位數值，LED 則是顯示其對應的二進位輸出(ex: 鍵盤輸入 AA，LED 顯示 01010101)。





執行結果:

```
nios2-terminal: connected to hardware target using JTAG UART on cable
nios2-terminal: "USB-Blaster [USB-0]", device 1, instance 0
nios2-terminal: (Use the IDE stop button or Ctrl-C to terminate)


input your 8-bit binary code : ( H -> L )
00111100
input your 8-bit binary code : ( H -> L )
11001100
input your number : ( 0 -> 255 )
85
input your 2-byte hex number : ( 00 -> FF )
A7
```