uClinux 作業系統下利用應用程式驅動 LED(uClinux-2)

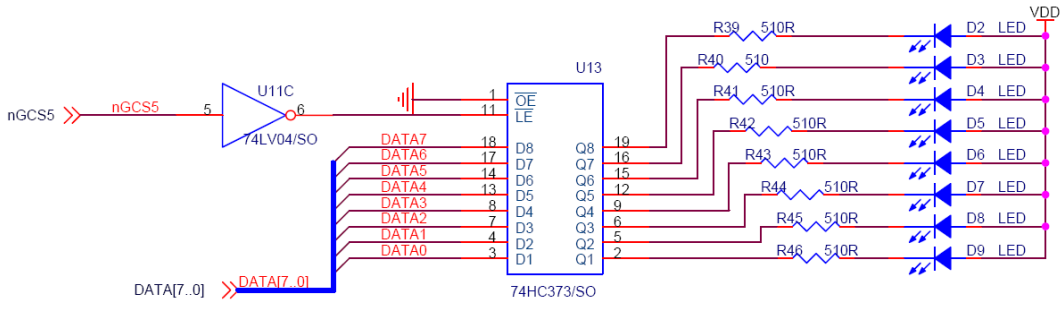
不管在視窗作業系統，或是 Linux 作業系統下，只要討論到輸出入控制，即便是小到一個發光二極體的亮或滅，就是要學習如何纂寫驅動程式，這也是許多控制界的人士，對於有作業系統的平台望之而卻步；主要的原因就是在於記憶體管理單元(Memory Management Unit)，大部分的作業系統為了防止使用者因纂寫程式的錯誤，而造成整個系統的損壞，所以在開機時，將所有硬體所相關的輸出入裝置，當然也包含了記憶體重新配置，除非是系統設計者，使用者無法直接接觸到硬體裝置，這就是為甚麼只要是針對硬體的控制，必須要纂寫驅動程式的原因。

uClinux 作業系統，為一針對無記憶體管理單元(Non-MMU)所纂寫的作業系統，所以可以直接控制硬體輸出入設備，雖然如此，整個系統相對的也暴露在風險中，所以，如果要直接控制輸出入設備，纂寫程式時務必要小心，不要造成系統的混亂。

基於 uClinux 作業系統的方便性，近幾年來筆者一直致力於推廣 uClinux 作業系統植入 NON-MMU 的平台上，使用 8 位元的使用者，可以仔細考慮看看，將平台改至 ARM7 為基礎的 32 位元，無記憶體管理單元的 uClinux 作業系統，如此可以快速的增強原有系統的功能。

以下將以驅動 ESD44B0_B 目標板內的 8 組發光二極體為例，說明 uClinux 作業系統下，用應用程式來控制輸出入設備的方便性：

1. 發光二極體原理：



- 74HC373 為資料閂積體電路，當 LE 為高電位及 OE 為低電位時，D[8:1]的狀態會被輸出至 Q[8:1]，其真值表如下：

Dn	LE	OE	On
H	H	L	H

L	H	L	L
X	L	L	Q0
X	X	H	Z

- 由電路得知，nGCS5 為低電位時，資料匯流排的資料會從輸出腳 Q[8:1]輸出，有關 S3C44B0X 之記憶體控制器的原理請參閱拙著。
- 發光二極體(D[2:9])的狀態為資料匯流排 DATA[7:0]狀態的反向。
- nGCS5 為目標晶片的記憶體區塊五所控制，所以只要存取記憶體位址 0xA00_0000 – 0xBFF_FFFF 區間，此致能腳位之狀態就會為低電位。

控制發光二極體，在 0xA00_0000 – 0xBFF_FFFF 記憶體區間內寫入控制資料，則發光二極體的狀態為寫入資料的反向。

2. 程式纂寫：使用者只要在記憶體 0xA00_0000 位址處寫入資料，相對應的 LED 即會有反向的動作，主要程式碼如下：

```
#include "hardware.h"

#define LED_MAP   (0xA000000)
.....................
void Display_Led( char Data )
{
     VPchar(LED_MAP) = Data;
}
```

- 含入檔 hardware.h 為系統定義檔，ESD44B0_B 目標版之定義於 linux-2.4.x/include/asm-armnommu/arch-esd 路徑下，此路徑必須在製作檔內定義。
- 呼叫 Display_Led 函式並帶入 Data 參數以控制發光二極體。

3. 製作檔纂寫：製作檔內除了含入 config.arch 外須加入 hardware.h 之路徑，這樣在編譯時才找的到，內容如下：

```
include config.arch
ESDPATH = -I$(ROOTDIR)/linux-2.4.x/include/asm-armnommu/arch-esd

EXEC = esdled
OBJS = esdled.o

all: $(EXEC)

$(EXEC): $(OBJS)
	$(CC) $(LDFLAGS) -o $@ $(OBJS) $(LDLIBS)

%.o : %.c
	$(CC) -c $(CFLAGS) $(ESDPATH) $< -o $@

clean:
	-rm -f $(EXEC) *.elf *.gdb *.o
```

只要執行 make 後即會產生執行檔，將此執行檔再目標板執行即可看見發光二極體以規律的方式閃爍。

發光二極體只是最簡單的輸出入控制，本人將會不定期由淺入深的發表，在 uClinux 作業系統下，以應用程式來控制不同的輸出入設備。

Victor 於加拿大