Low Level Assembly Language

6502 (維基百科)

MOS 6502 [編輯]

維基百科，自由的百科全書

MOS 6502是1975年由MOS科技所研發的8位元微處理器。當年6502剛問世時是當時效能最強的8位元CPU，且價格只有大型業者（如Motorola、Intel）相近產品的六分之一甚至更低；且除了Zilog公司的Z80外，6502幾乎快過多數業者的相近產品，進而激起一系列的的電腦專案[1]，並在之後的1980年代帶來一場個人電腦的革命。MOS科技僅授權兩家業者能相容研製6502，即是所謂的「第二供貨源」，此分別是洛克威爾國際公司（Rockwell International）與Synertek公司，更之後才有更多的業者獲得相容研製的授權，並仍持續在嵌入式系統的市場中供貨。

目錄 [隱藏]

一顆以DIL-40塑膠封裝的MOS 6502處理器。

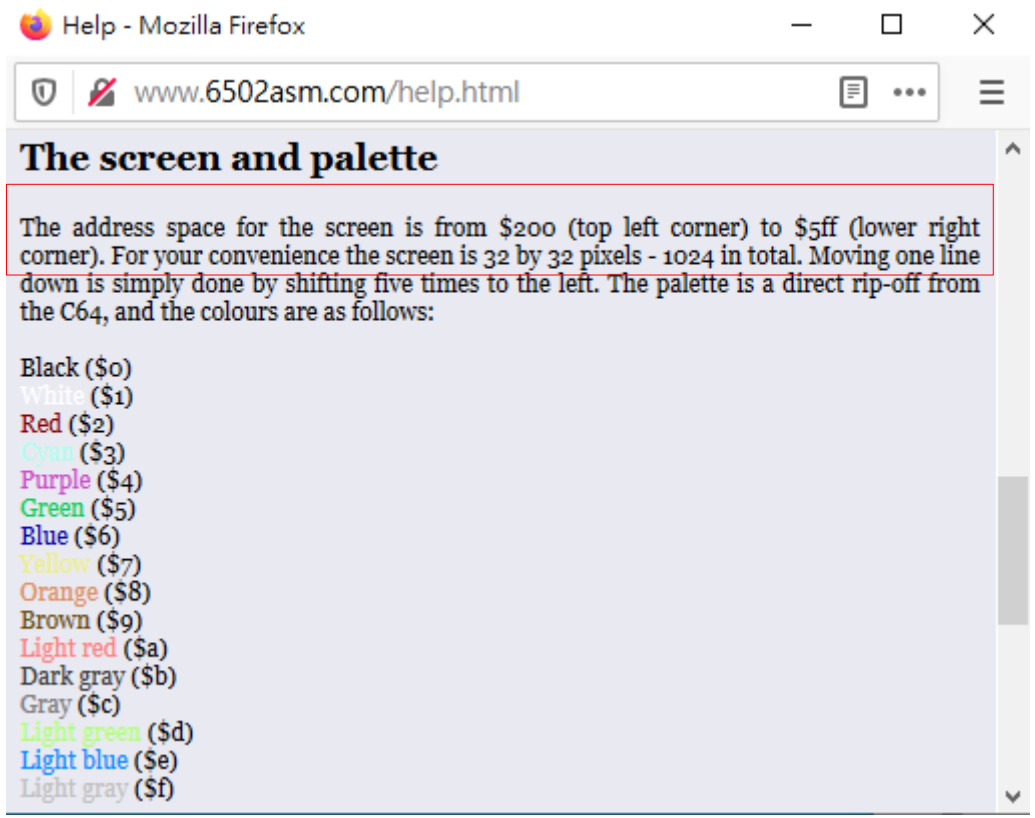
以6502系列為CPU的裝置 [編輯]

Apple II的採用使得6502成了廣為人知的CPU，歐美地區發售的Commodore系列8位元電腦也大量使用了6502系列CPU。

家用遊戲機紅白機（使用6502相容指令集的Ricoh 2A03）、文曲星電子詞典等也採用它。後來的「超級任天堂」使用了65C816（16位元版的6502）。

我直到今天都覺得
超任的遊戲很好玩

www.6502asm.com（顯示位址的範圍和其調色盤）



網頁提供的虛擬硬體的說明

$5ff - $200 + 1 = $400 = 2^10
32 x 32 = 2^5 x 2^5 = 2^10 = 1024

www.6502asm.com（測試對兩塊區域上色）

預設畫面

Compile & run code

https://www.masswerk.at/6502/6502_instruction_set.html

6502 Instruction Set

6502 (8-bit microprocessor) 指令集

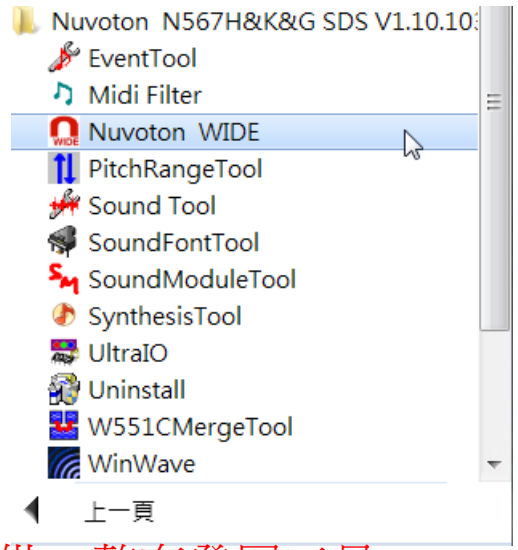
HI	LO-NIBBLE															
	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
00	BRK impl	ORA X,ind				ORA zpg	ASL zpg		PHP impl	ORA #	ASL A			ORA abs	ASL abs	
10	BPL rel	ORA ind,Y				ORA zpg,X	ASL zpg,X		CLC impl	ORA abs,Y				ORA abs,X	ASL abs,X	
20	JSR abs	AND X,ind			BIT zpg	AND zpg	ROL zpg		PLP impl	AND #	ROL A		BIT abs	AND abs	ROL abs	
30	BMI rel	AND ind,Y				AND zpg,X	ROL zpg,X		SEC impl	AND abs,Y				AND abs,X	ROL abs,X	
40	RTI impl	EOR X,ind				EOR zpg	LSR zpg		PHA impl	EOR #	LSR A		JMP abs	EOR abs	LSR abs	
50	BVC rel	EOR ind,Y				EOR zpg,X	LSR zpg,X		CLI impl	EOR abs,Y				EOR abs,X	LSR abs,X	
60	RTS impl	ADC X,ind				ADC zpg	ROR zpg		PLA impl	ADC #	ROR A		JMP ind	ADC abs	ROR abs	
70	BVS rel	ADC ind,Y				ADC zpg,X	ROR zpg,X		SEI impl	ADC abs,Y				ADC abs,X	ROR abs,X	
80		STA X,ind			STY zpg	STA zpg	STX zpg		DEY impl		TXA impl		STY abs	STA abs	STX abs	
90	BCC rel	STA ind,Y			STY zpg,X	STA zpg,X	STX zpg,Y		TYA impl	STA abs,Y	TXS impl			STA abs,X		
A0	LDY #	LDA X,ind	LDX #		LDY zpg	LDA zpg	LDX zpg		TAY impl	LDA #	TAX impl		LDY abs	LDA abs	LDX abs	
B0	BCS rel	LDA ind,Y			LDY zpg,X	LDA zpg,X	LDX zpg,Y		CLV impl	LDA abs,Y	TSX impl		LDY abs,X	LDA abs,X	LDX abs,Y	
C0	CPY #	CMP X,ind			CPY zpg	CMP zpg	DEC zpg		INY impl	CMP #	DEX impl		CPY abs	CMP abs	DEC abs	
D0	BNE rel	CMP ind,Y				CMP zpg,X	DEC zpg,X		CLD impl	CMP abs,Y				CMP abs,X	DEC abs,X	
E0	CPX #	SBC X,ind			CPX zpg	SBC zpg	INC zpg		INX impl	SBC #	NOP impl		CPX abs	SBC abs	INC abs	
F0	BEQ rel	SBC ind,Y				SBC zpg,X	INC zpg,X		SED impl	SBC abs,Y				SBC abs,X	INC abs,X	

上頁程式用到了 8 種指令

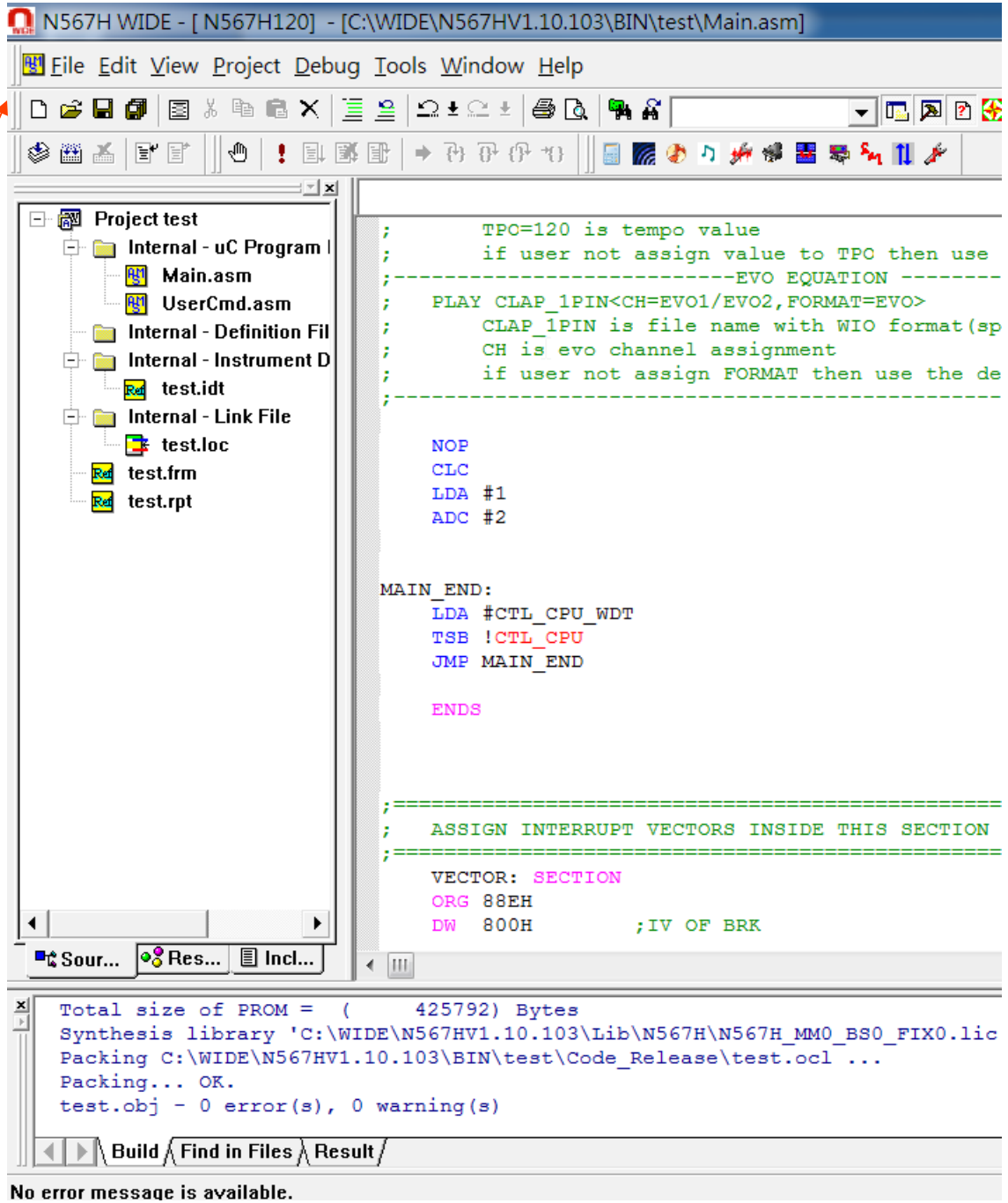
Address Modes:

A	.... Accumulator	OPC A	*operand is AC (implied single byte instruction)*
abs	.... absolute	OPC $LLHH	*operand is address $HHLL **
abs,X	.... absolute, X-indexed	OPC $LLHH,X	*operand is address; effective address is address incremented by X with carry ***
abs,Y	.... absolute, Y-indexed	OPC $LLHH,Y	*operand is address; effective address is address incremented by Y with carry ***
#	.... immediate	OPC #$BB	*operand is byte BB*
impl	.... implied	OPC	*operand implied*
ind	.... indirect	OPC ($LLHH)	*operand is address; effective address is contents of word at address: C.w($HHLL)*
X,ind	.... X-indexed, indirect	OPC ($LL,X)	*operand is zeropage address; effective address is word in (LL + X, LL + X + 1), inc. without carry: C.w($00LL + X)*
ind,Y	.... indirect, Y-indexed	OPC ($LL),Y	*operand is zeropage address; effective address is word in (LL, LL + 1) incremented by Y with carry: C.w($00LL) + Y*
rel	.... relative	OPC $BB	*branch target is PC + signed offset BB ****
zpg	.... zeropage	OPC $LL	*operand is zeropage address (hi-byte is zero, address = $00LL)*
zpg,X	.... zeropage, X-indexed	OPC $LL,X	*operand is zeropage address; effective address is address incremented by X without carry ***
zpg,Y	.... zeropage, Y-indexed	OPC $LL,Y	*operand is zeropage address; effective address is address incremented by Y without carry ***

WIDE (Winbond Intergrated Development Environment)

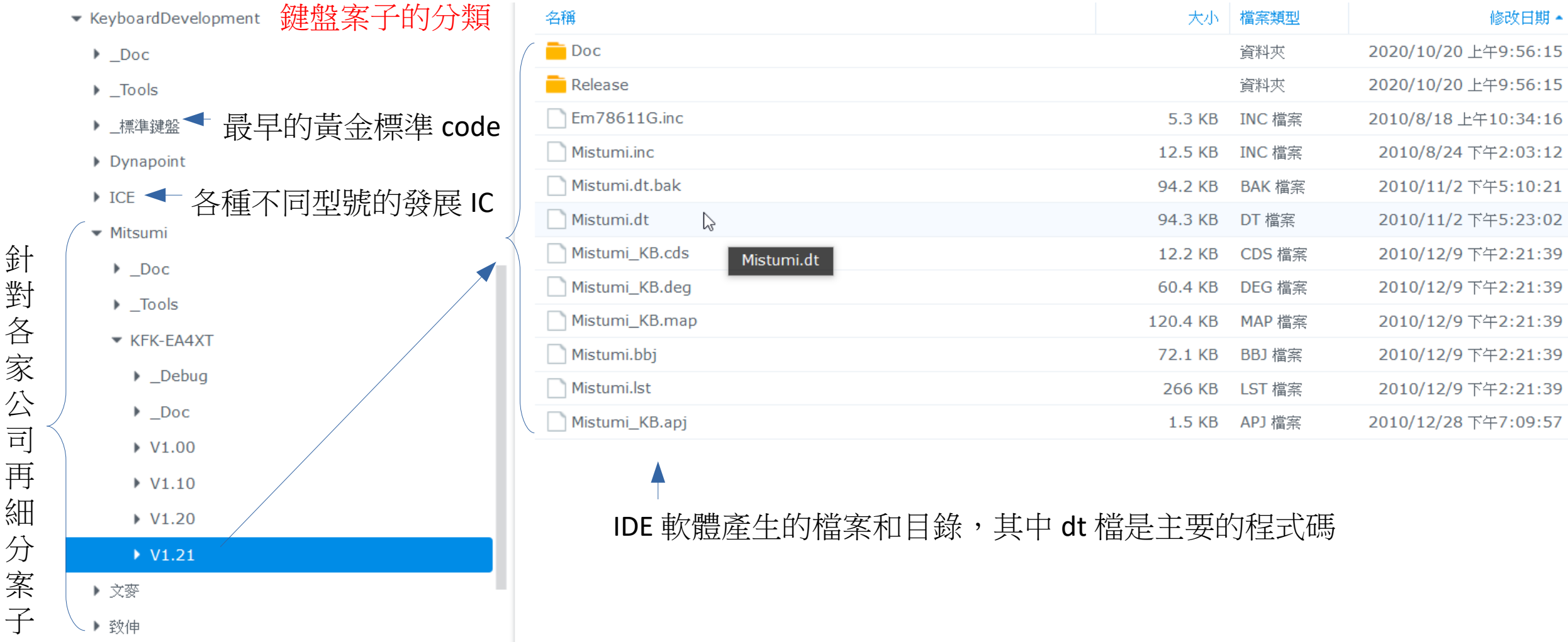


Winbond = 華邦



軟體部門提供一整套發展工具， WIDE 只是其中之一

Firmware 概觀 (以義隆電的鍵盤案子為例)



C:\Users\Atrisk\AppData\Local\Temp\Mistumi.dt - EmEditor

檔案(F) 編輯(E) 搜尋(S) 檢視(V) 工具(T) 視窗(W) 說明(H)

Mistumi.dt

```
        bc      R15, 0
        jbs     R10, 6
        bs      R15, 0
COM_NORMAL_KEY_1:
        bs      R10, 6
        jmp     CHECK_EP1
; COMBINE KEY FORMAT: NORAML KEY -> USB KEY CODE, BIT7 M/B, OTHER USB KEY CODE
;           COMBINE KEY -> BIT3 M/B OTHER USB KEY CODE


;============================== USB BREAK KEY ================================
CLR_BUFFER:
        mov     A, @0xB0
        mov     RegRSR, A
CLR_KEY_LOOP:
        mov     A, R0
        xor     A, R10
        jbc     RegStatus, bmStaZ
        jmp     CLR_NORMAL_KEY
        inc     RegRSR
        mov     A, RegRSR
        xor     A, @0xB8
        jbc     RegStatus, bmStaZ
        jmp     CLR_KEY_NOT_FIND
        mov     R0, R0   ;ERROR FOR NO MATCH KEY IN FIFO
        jbs     RegStatus, bmStaZ
        jmp     CLR_KEY_LOOP
CLR_KEY_NOT_FIND:
        bs      R17, bmEp1ReportReady;FLAG OF KEY REPORT STATUS
```

部分程式原始碼 (Mistumi.dat)

一個案子約有五千行類似左邊的程式碼，想像若沒文件和註解的話？

D:\Slides_2020\Low_Level_Assembly_Language\usb-rs232\Demo_MOD-USB-RS232\Demo\USB2RS232\main.c - EmEditor

檔案(F) 編輯(E) 搜尋(S) 檢視(V) 工具(T) 視窗(W) 說明(H)

usb_config.h | main.c

```
 *↓
 * Overview:        This routine initializes the UART to 19200↓
 *↓
 * Note:            ↓
 *↓
 ********************************************************************/↓
void InitializeUSART(void)↓
{↓
    #if defined(__18CXX)↓
        unsigned char c;↓
        #if defined(__18F14K50)↓
            //ANSELHbits.ANS11 = 0; // Make RB5 digital so USART can use pin for Rx↓
            ANSELH = 0;↓
            #ifndef BAUDCON↓
                #define BAUDCON BAUDCTL↓
            #endif↓
        #endif↓
        UART_TRISRx=1;              // RX↓
        UART_TRISTx=0;              // TX↓
        TXSTA = 0x24;           // TX enable BRGH=1↓
        RCSTA = 0x90;           // Single Character RX↓
        SPBRG = 0x71;↓
        SPBRGH = 0x02;          // 0x0271 for 48MHz -> 19200 baud↓
        BAUDCON = 0x08;         // BRG16 = 1↓
        c = RCREG;              // read ↓
    #endif↓
```

部分

microchip 提供的 usb 轉 rs232 解決方案

程式碼用 C 語言實現，主程式檔案有 1345 行，很多是註解，估計真正的 C 指令有幾百行

project 的檔案

rs232 › Demo_MOD-USB-RS232 › Demo › USB2RS232 ›

搜尋 USB2RS232

名稱	修改日期	類型	大小
Objects	2011/5/16 下午 04:44	檔案資料夾	
output	2011/5/16 下午 04:44	檔案資料夾	
18f14k50_g.lkr	2020/10/21 上午 08:52	LKR 檔案	2 KB
HardwareProfile - Low Pin Count USB Development Kit.h	2020/10/21 上午 08:52	C/C++ Header	7 KB
HardwareProfile.h	2020/10/21 上午 08:52	C/C++ Header	5 KB
main.c	2020/10/21 上午 08:52	C Source	47 KB
procdefs.ld.boot	2020/10/21 上午 08:52	BOOT 檔案	3 KB
rm18F14K50.lkr	2020/10/21 上午 08:52	LKR 檔案	2 KB
USB Device - CDC - Serial Emulator - C18 - Low Pin Count USB Development Kit.mcp	2020/10/21 上午 08:52	MCP 檔案	3 KB
USB Device - CDC - Serial Emulator - C18 - Low Pin Count USB Development Kit.mcs	2020/10/21 上午 08:52	MCS 檔案	1 KB
USB Device - CDC - Serial Emulator - C18 - Low Pin Count USB Development Kit.mcw	2020/10/21 上午 08:52	MCW 檔案	96 KB
USB Device - CDC - Serial Emulator - C18 - Low Pin Count USB Development Kit.tagsrc	2020/10/21 上午 08:52	TAGSRC 檔案	2 KB
usb_config.h	2020/10/21 上午 08:52	C/C++ Header	6 KB
usb_descriptors.c	2020/10/21 上午 08:52	C Source	13 KB