

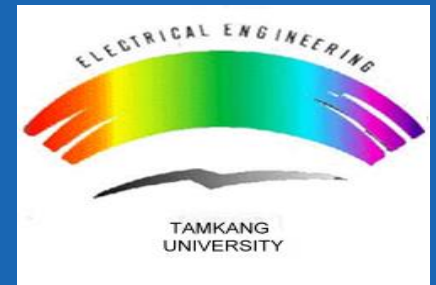
# 第05次實習課

學生：林培瑋

2024 Advanced Mixed-Operation System (AMOS) Lab.



**Tamkang University**  
**Department of Electrical and Computer Engineering**  
No.151, Yingzhuan Rd., Tamsui Dist., New Taipei City 25137, Taiwan (R.O.C.)





|                                 |    |
|---------------------------------|----|
| ❖ 微處理機概論 – HW1.....             | 3  |
| ❖ HW1 重點整理.....                 | 22 |
| ❖ UART技術手冊 p8、p30.....          | 23 |
| ❖ Keil Tool-P348、349(F11).....  | 24 |
| ❖ Keil Tool-P348、349(F10).....  | 27 |
| ❖ Keil Tool-P348、349(F5).....   | 31 |
| ❖ p347 poll & busy waiting..... | 32 |





# 微處理機概論 – HW1

繳交期限：4/9 23:59

(用 word 檔上傳至 iclass)

請提前上傳，以免 iclass 塞車

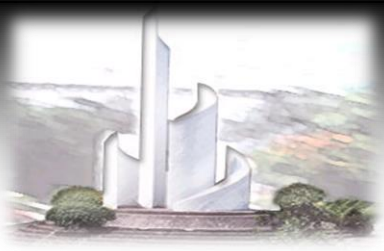
**PANEL**

Parallel Architectures & Networks Lab.

Dept. of Electrical Engineering 淡江大學電機系

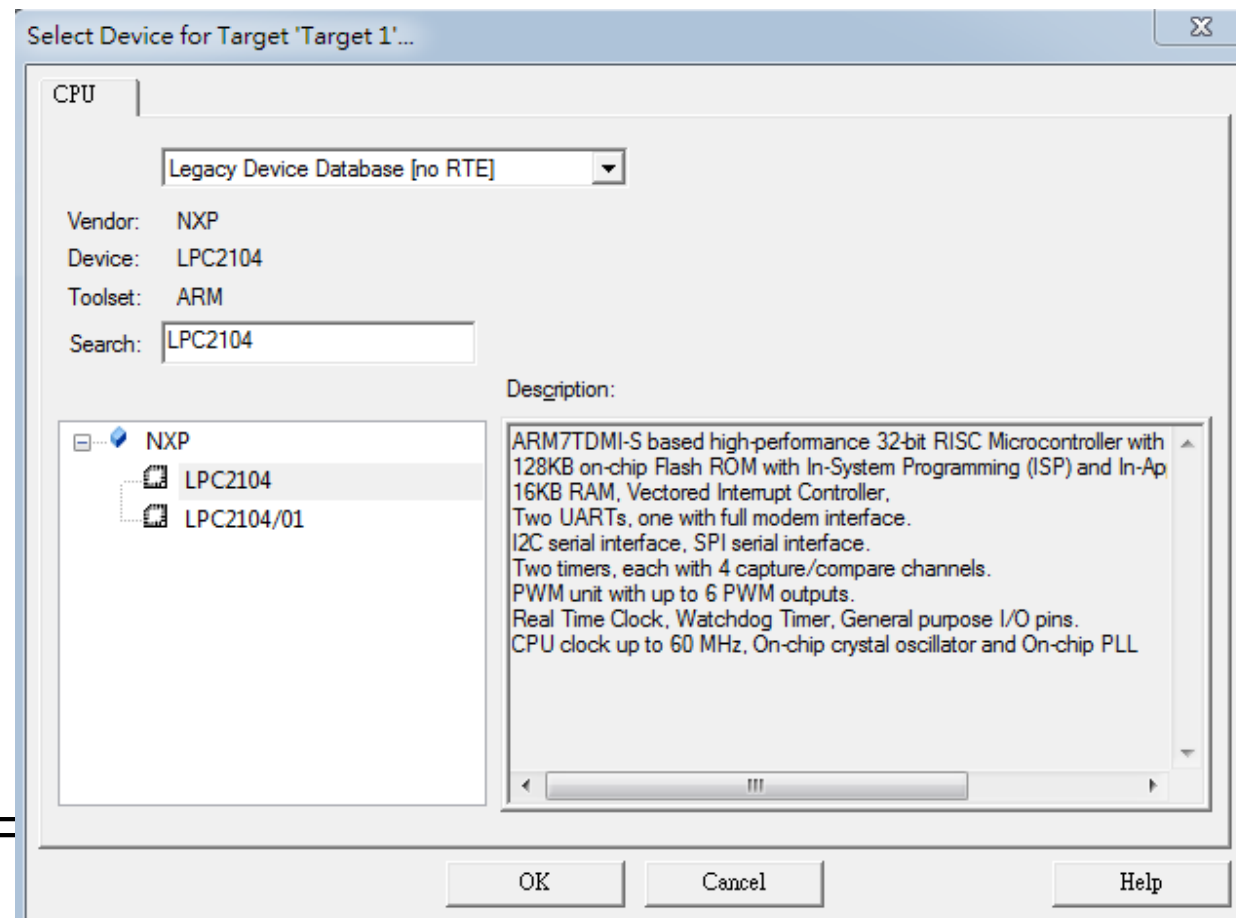


# 作業題目



課本p.348-349

ARM7TDMI: LPC2104

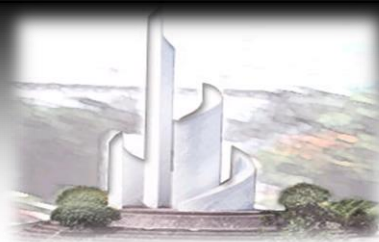


**PANEL**

Parallel Architectures & Networks Lab.

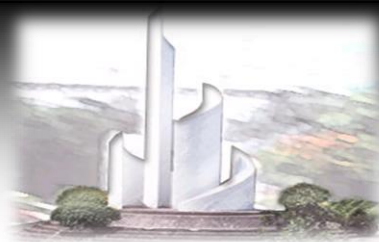


# 程式(1/3)



```
1 AREA UARTDEMO, CODE, READONLY
2 PINSEL0 EQU 0xE002C000 ; controls the function of pins
3 U0START EQU 0xE000C000 ; start of UART0 register
4 LCR0 EQU 0xC ; line control register for UART0
5 LSR0 EQU 0x14 ; line status register for UART0
6 RAMSTART EQU 0x40000000 ; start of onboard RAM for 2104
7 ENTRY
8 start
9 LDR sp, =RAMSTART ; set up stack pointer
10 BL UARTConfig ; initialize/configure UART0
11 LDR r1, =CharData ; starting address of characters
12 Loop
13 LDRB r0, [r1], #1 ; load character, increment address
14 CMP r0, #0 ; null terminated?
15 BLNE Transmit ; send character to UART
16 BNE Loop ; continue if not a '0'
17 done B done ; otherwise we're done
18
```

# 程式(2/3)

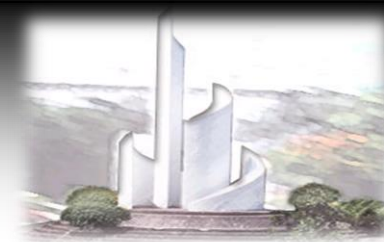


```
19
20 UARTConfig
21     STMIA    sp!, {r5, r6, LR}
22     LDR      r5, =PINSEL0      ; base address of register
23     LDR      r6, [r5]          ; get contents
24     BIC      r6, r6, #0xF      ; clear out lower nibble
25     ORR      r6, r6, #0x5      ; sets p0.0 to Tx0 and P0.1 to Rx0
26     STR      r6, [r5]          ; r/modify/w back to register
27     LDR      r5, =U0START
28     MOV      r6, #0x83         ; set 8 bits, no parity, 1 stop bit
29     STRB     r6, [r5, #LCR0]    ; write control byte to LCR
30     MOV      r6, #0x61         ; 9600 baud @15 MHz VPB clock
31     STRB     r6, [r5]          ; store control byte
32     MOV      r6, #3            ; set DLAB=0
33     STRB     r6, [r5, #LCR0]    ; Tx and Rx buffers set up
34     LDMDDB   sp!, {r5, r6, PC}
35
```





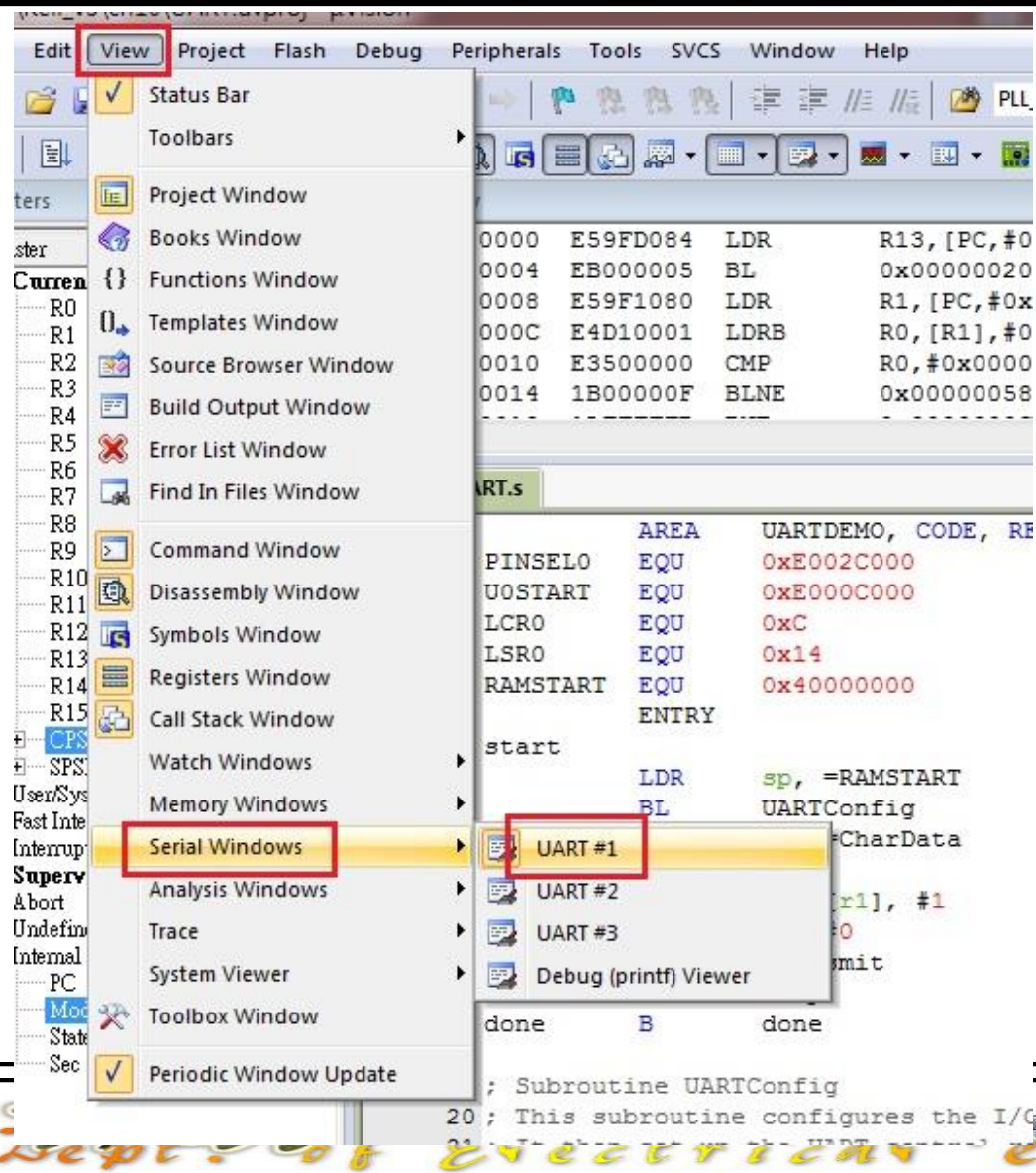
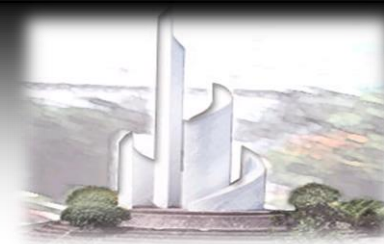
# 程式(3/3)



```
36
37 Transmit
38     STMIA    sp!, {r5, r6, LR}
39     LDR      r5, =U0START
40 wait
41     LDRB     r6, [r5, #LSR0]    ; get status of buffer
42     改成TST  CMP     r6, #0x20    ; buffer empty?
43     BEQ      wait              ; spin until buffer's empty
44     STRB     r0, [r5]
45     LDMDB    sp!, {r5, r6, PC}
46 CharData
47     DCB      "Watson. Come quickly!", 0
48     END
49
```



# 開啟UART視窗



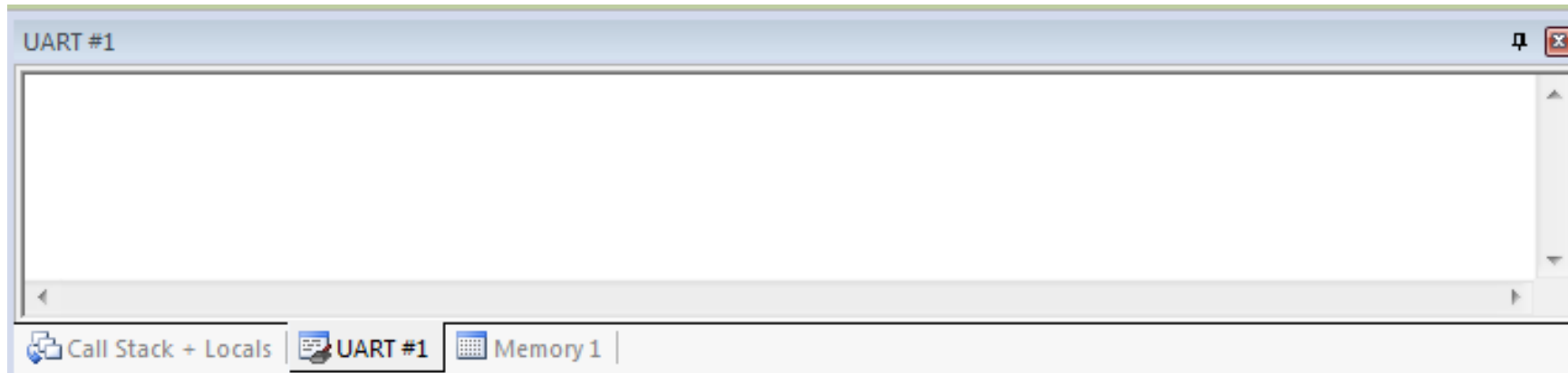
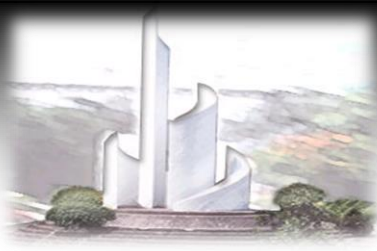
**PANEL**

Parallel Architectures & Networks Lab.





右下角可以看到UART視窗



**PANEL**

Parallel Architectures & Networks Lab.

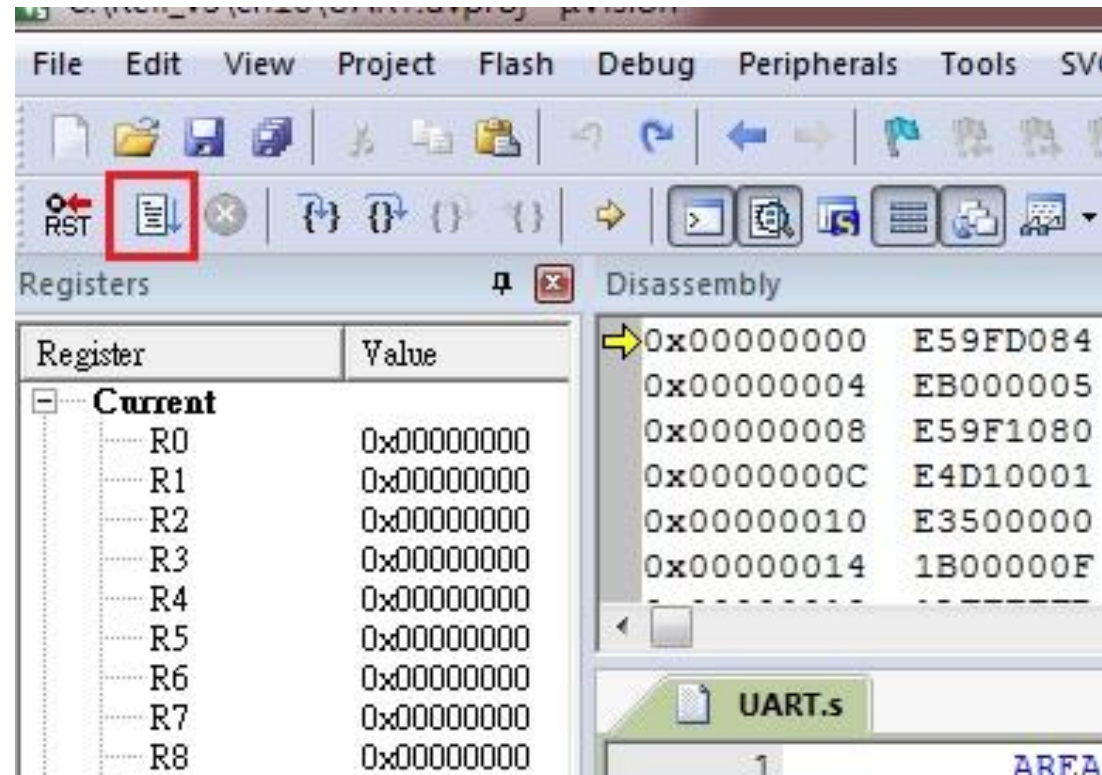
TJU of Electrical Engineering 淡江大學電機系



# 執行程式



► 執行程式使用”Run”或按”F5”



**PANEL**

Parallel Architectures & Networks Lab.

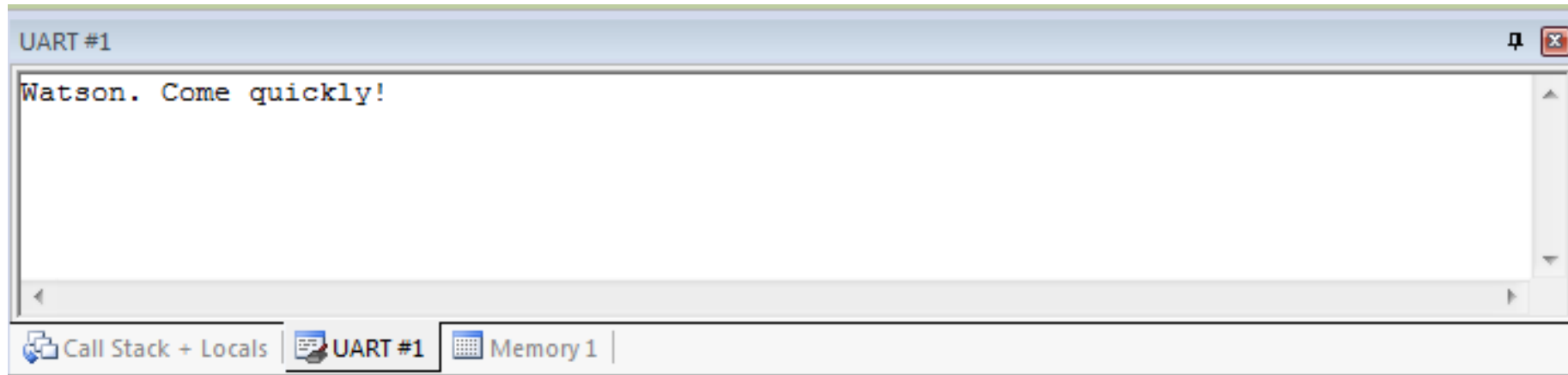
Dept. of Electrical Engineering 淡江大學電機系



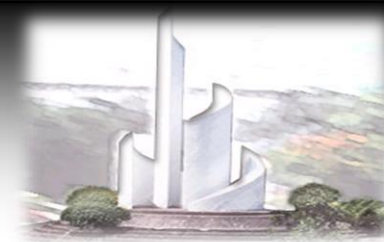
# 執行結果



結果圖成功印出字串



# 第一部分作業內容



- ▶ (1) 執行課本p.348-349程式

印出字串”**TKU-ECE+學號+英文名字**”

Ex: TKU-ECE 407443210 Michael

- (2) 改寫上述程式使能印出其反向字串

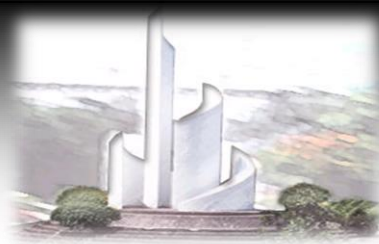
Ex: leahciM 012344704 ECE-UKT

- (3) Calculate Keil Tool System Clock

Frequency and rewrite the initialization to show  
about 9600 baud on the divisor latch window

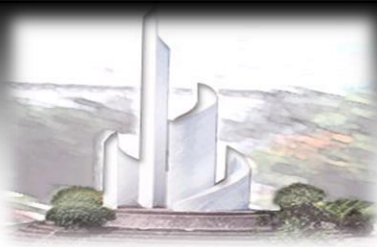


# 第一部分作業內容



- ▶ 截圖清楚，要截到UART#1及如下頁所示之UART0視窗圖。
- ▶ 請交完整結報word檔。(包括按F5、按F11及按F10執行效果的心得)





## UART0視窗

Universal Asynchronous Receive Transmit 0 (UART0)

|  |  |
|--|--|
| <b>Line Control</b><br>UOLCR: 0x00<br>Word Length: 5 bits<br>Stop Bits: 1<br>Parity: Odd Parity<br><input type="checkbox"/> DLAB<br><input type="checkbox"/> Break Control<br><input type="checkbox"/> Parity Enable | <b>Line Status</b><br>UOLSR: 0x60<br><input type="checkbox"/> Receiver Data Ready (RDR)<br><input type="checkbox"/> Overrun Error (OE)<br><input type="checkbox"/> Parity Error (PE)<br><input type="checkbox"/> Framing Error (FE)<br><input type="checkbox"/> Break Interrupt (BI)<br><input checked="" type="checkbox"/> Tx Holding Register Empty (THRE)<br><input checked="" type="checkbox"/> Transmitter Empty (TEMT)<br><input type="checkbox"/> Error in Rx FIFO (RXFE) |
| <b>Interrupt Enable</b><br>UIIER: 0x00<br><input type="checkbox"/> RBR IE<br><input type="checkbox"/> THRE IE<br><input type="checkbox"/> Rx Line Status IE  | <b>Interrupt ID &amp; FIFO Control</b><br>UIIIR/FCR: 0x01 <input type="checkbox"/> FIFO Enable<br>Interrupt: None<br>Rx Trigger: Level 0 (1)<br><input type="checkbox"/> Rx FIFO Reset <input type="checkbox"/> Tx FIFO Reset  |
| <b>Divisor Latch</b><br>UODLL: 0x01<br>UODLM: 0x00<br>Baudrate: 187500   | <b>Receiver &amp; Transmitter Registers</b><br>UORBR/THR: 0x00<br><b>Scratch Pad Register</b><br>UOSCR: 0x00   |

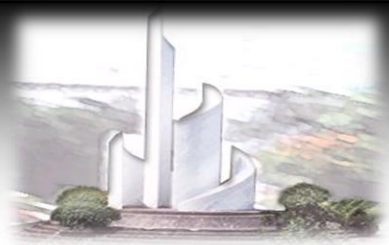
**PANEL**

Parallel Architectures & Networks Lab.

Dept. of Electrical Engineering



## 第二部分



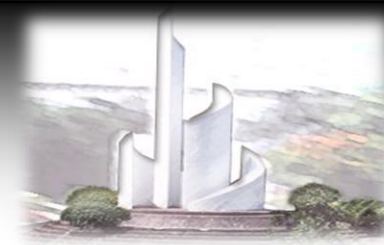
### 加入上課所講的Receive (主程式)

```
13 Loop
14     LDRB r0, [r1],#1 ; load character, increment address
15     CMP r0,#0 ; null terminated?
16     BLNE Transmit ; send character to UART
17     BNE Loop ; continue if not a ??
18
19     MOV r9,#20
20 Loop1
21     BL Receive
22     STRB r0, [r1],#1
23     SUBS r9,r9,#1
24     BNE Loop1
25 done B done
```

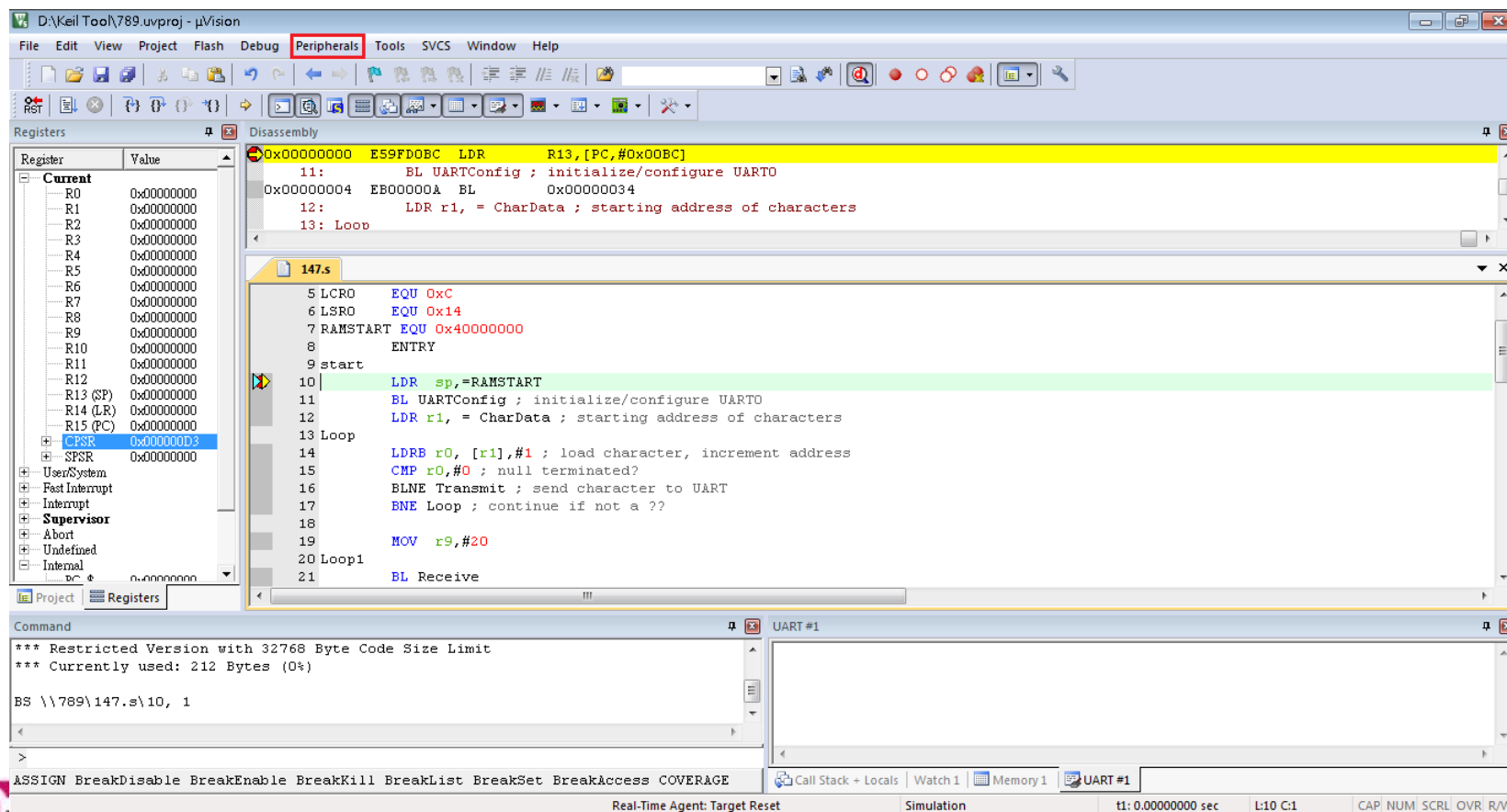


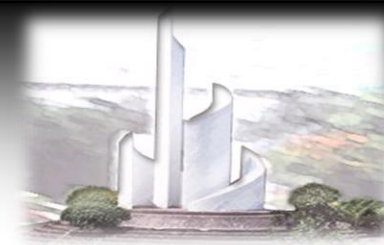
## 加入上課所講的Receive (副程式)

```
44 Transmit
45     STMIA sp!,{r5,r6,lr}
46     LDR r5, = UOSTART
47 wait
48     LDRB r6,[r5,#LSRO] ; get status of buffer
49     TST r6,#0x20 ; buffer empty?
50     BEQ wait ; spin until buffer empty
51     STRB r0,[r5]
52     LDMDB sp!,{r5,r6, pc}
53
54 Receive
55     STMIA sp!,{r5,r6,lr}
56     LDR r5, = UOSTART
57 wait1
58     LDRB r6,[r5,#LSRO] ; get status of buffer
59     TST r6,#1 ;data ready
60     BEQ wait1 ;until data ready
61     LDRB r0,[r5]
62     LDMDB sp!,{r5,r6, pc}
63
```

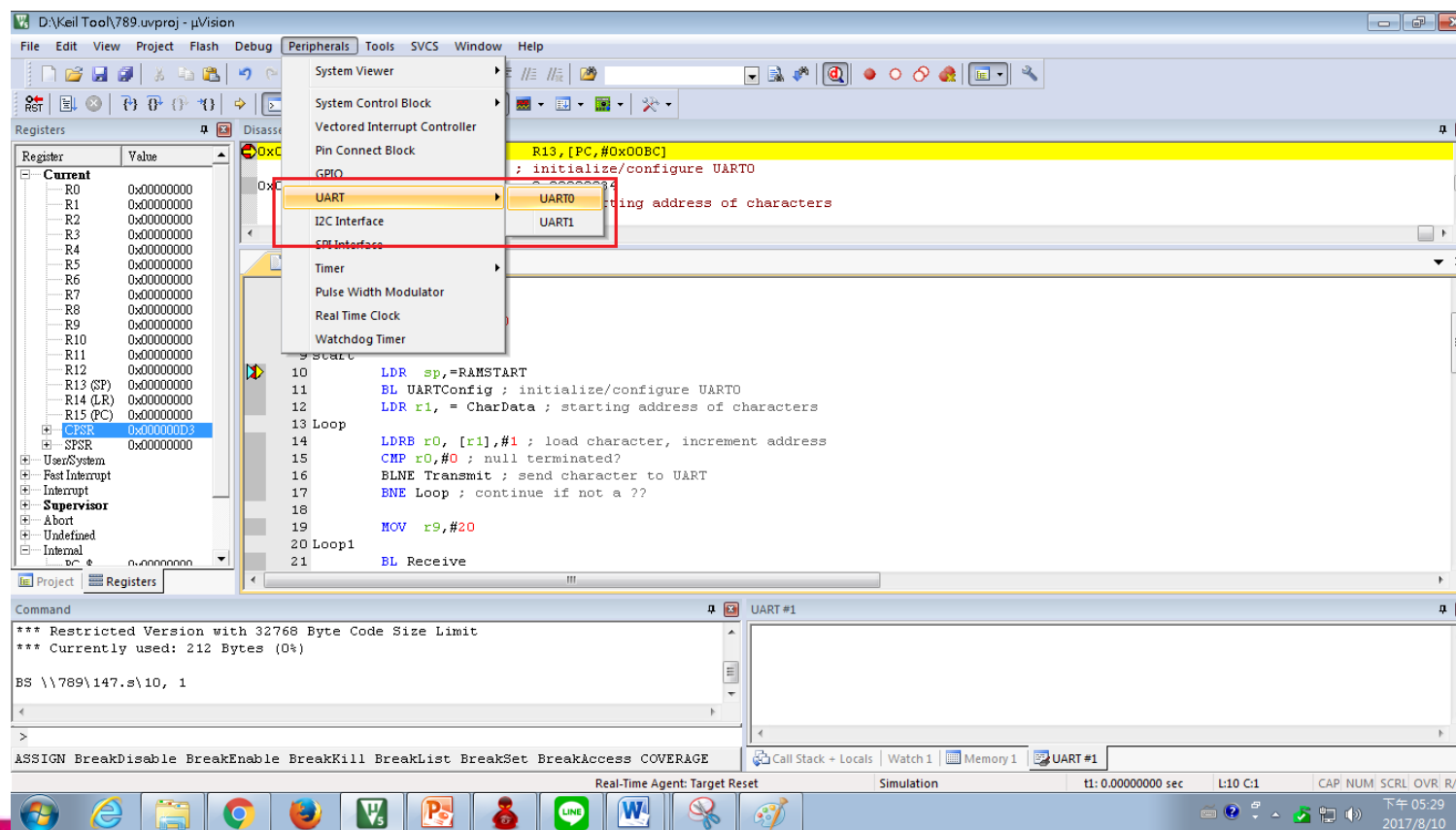


## ► 打開圖中的Peripherals

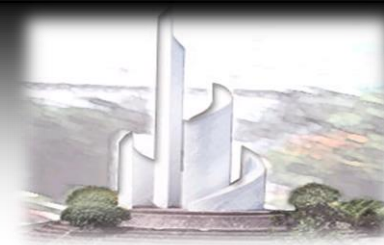




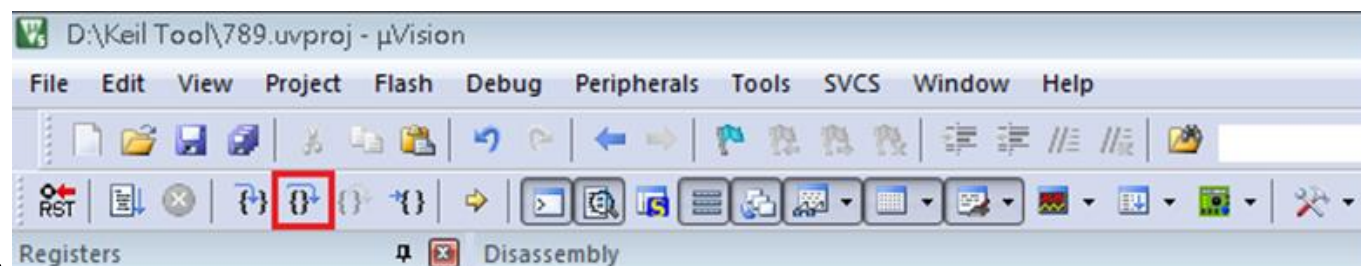
## 找到UART點選UART0







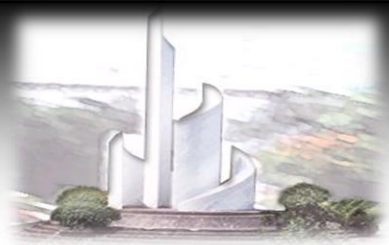
▶ 接著點選圖中的指令或是F10 and F11一直到結束



▶ 再點選的迴程中請注意UOLCK以及UOLSK的變化，並思考其中代表的意義。

▶ 加入三種錯誤的偵測，沒有錯誤的時候才可receive

## 第二部分作業內容



- ▶ 印出字串 ”**TKU-ECE+學號+英文名字**”
- ▶ 截圖清楚，要截到UART0視窗圖以及UART#1視窗圖。
- ▶ **請繳交包含第一部分與第二部分的完整結報word檔。**



◆ 繳交內容：按照結報格式編寫完整，把結報word檔上傳iclass對應作業位置。

◆ 繳交期限：4/9 23:59

◆ Word檔名：

**微處理機概論\_學號\_姓名\_HW1**

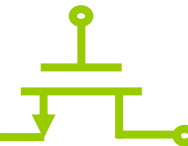
◆ 程式**需在Keil Tool程式視窗內展示**且展示程式與執行結果的**每一截圖需看的到學號姓名**否則不計分。

**PANEL**

Parallel Architectures & Networks Lab.

Dept. of Electrical Engineering 淡江大學電機系





## ❖ 第一部分：

- (3) Calculate Keil Tool **System Clock Frequency** and rewrite the initialization to **show about 9600 baud on the divisor latch window**
- 先算出**System Clock Frequency**，再改寫”設定速率”的程式碼，使其在**divisor latch window**上顯示9600 baud
- 須以註解或手算方式來呈現**System Clock Frequency**

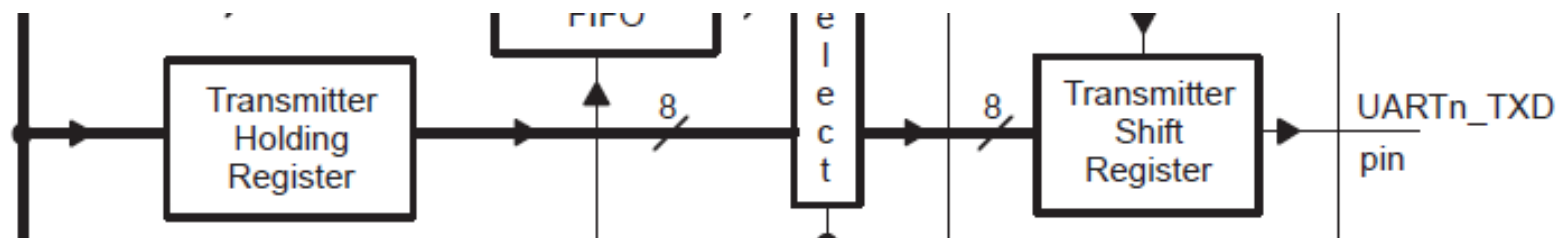
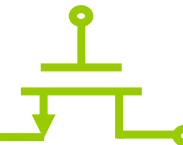
## ❖ 第二部分：

- Receive 主程式 → SUB **S** r9, r9, #1
- 代表運算後的結果會影響狀態暫存器 CPSR
  - 也可以寫成
  - SUB r9, r9, #1
  - CMP r9, #0

在ARM架構中，CPSR寄存器的NZCV標誌位佔有4個bit，分別代表以下意義：

- N ( Negative, 負數 )：第31位，如果計算結果是負數，則該位被設置為1，否則為0。
- Z ( Zero, 零 )：第30位，如果計算結果為零，則該位被設置為1，否則為0。
- C ( Carry, 進位 )：第29位，用於無符號算術操作，如果發生進位，則該位被設置為1，否則為0。
- V ( Overflow, 溢出 )：第28位，用於有符號算術操作，如果發生溢出，則該位被設置為1，否則為0。





|   |      |   |  |
|---|------|---|--|
| 6 | TEMT |   | Transmitter empty (TEMT) indicator.<br><b>In non-FIFO mode:</b>  |
|   |      | 0 | Either the transmitter holding register (THR) or the transmitter shift register (TSR) contains a data character.   |
|   |      | 1 | Both the transmitter holding register (THR) and the transmitter shift register (TSR) are empty.  |
|   |      | 0 | <b>In FIFO mode:</b><br>Either the transmitter FIFO or the transmitter shift register (TSR) contains a data character.   |
|   |      | 1 | Both the transmitter FIFO and the transmitter shift register (TSR) are empty.  |
| 5 | THRE |   | Transmitter holding register empty (THRE) indicator. If the THRE bit is set and the corresponding interrupt enable bit is set (ETBEI = 1 in IER), an interrupt request is generated.<br><b>In non-FIFO mode:</b> |
|   |      | 0 | Transmitter holding register (THR) is not empty. THR has been loaded by the CPU.   |
|   |      | 1 | Transmitter holding register (THR) is empty (ready to accept a new character). The content of THR has been transferred to the transmitter shift register (TSR).  |
|   |      | 0 | <b>In FIFO mode:</b><br>Transmitter FIFO is not empty. At least one character has been written to the transmitter FIFO. You can write to the transmitter FIFO if it is not full.                                 |
|   |      | 1 | Transmitter FIFO is empty. The last character in the FIFO has been transferred to the transmitter shift register (TSR).  |



## ❖ 第一次迴圈

→如果TSR是空的，則資料會自動從THR傳到TSR。

```

p348UART.s
5 LSRO      EQU      0x14      ; line status register for UART0
6 RAMSTART  EQU      0x40000000 ; start of onboard RAM for 2104
7
8 start
9          LDR      sp, = RAMSTART ; set up stack pointer
10         BL       UARTConfig ; initialize/configure UART0
11         LDR      r1, = CharData ; starting address of characters
12 Loop
13         LDRB     r0, [r1], #1 ; load character, increment address
14         CMP     r0, #0 ; null terminated?
15         BLNE    Transmit ; send character to UART
16         BNE     Loop ; continue if not a '0'
17 done      B       done ; otherwise we e done
18
19 ; Subroutine UARTConfig
20 ; This subroutine configures the I/O pins first. It
21 ; then sets up the UART control register. The
22 ; parameters
23 ; are set to 8 bits, no parity and 1 stop bit.
24 ; Registers used:
25 ; r5 - scratch register
26 ; r6 - scratch register
27 ; inputs: none
28 ; outputs: none
29
30 UARTConfig
31         STMIA    sp!, {r5, r6, lr}
32         LDR      r5, = PINSEL0 ; base address of register
33         LDR      r6, [r5] ; get contents
34         BIC     r6, r6, #0xF ; clear out lower nibble
35         ORR     r6, r6, #0x5 ; sets P0.0 to Tx0 and P0.1 to Rx0
36         STR     r6, [r5] ; r/modify/w back to register
37         LDR      r5, = UOSTART
38         MOV     r6, #0x83 ; set 8 bits, no parity, 1 stop bit
39         STR     r6, [r5, #1CR01] ; write control byte to LCR
          
```

Universal Asynchronous Receive Transmit 0 (UART0)

Line Control

UOLCR: 0x03

Word Length: 8 bits

Stop Bits: 1

Parity: Odd Parity

☐ DLAB

☐ Break Control

☐ Parity Enable

Line Status

UOLSR: 0x20

☐ Receiver Data Ready (RDR)

☐ Overrun Error (OE)

☐ Parity Error (PE)

☐ Framing Error (FE)

☐ Break Interrupt (BI)

☒ Tx Holding Register Empty (THRE)

☐ Transmitter Empty (TEMT)

☐ Error in Rx FIFO (RXFE)

Interrupt Enable

UOIER: 0x00

☐ RBR IE

☐ THRE IE

☐ Rx Line Status IE

Interrupt ID & FIFO Control

UOIRR/FCR: 0x01 ☐ FIFO Enable

Interrupt: None

Rx Trigger: Level 0 (1)

☐ Rx FIFO Reset ☐ Tx FIFO Reset

Divisor Latch

UODLL: 0x61

UODLM: 0x00

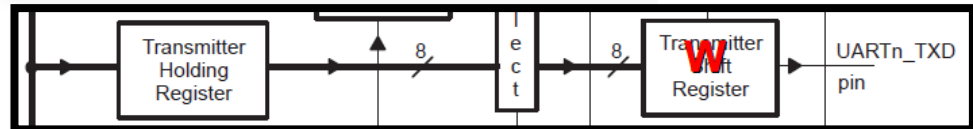
Baudrate: 1932

Receiver & Transmitter Registers

UORBR/THR: 0x00


Scratch Pad Register

UOSCR: 0x00



The diagram illustrates the data flow in a UART transmitter. It shows a 'Transmitter Holding Register' connected to an 8-bit 'Shift Register'. The 'Shift Register' is connected to an 8-bit 'Transmitter Register', which is then connected to the 'UARTn\_TXD pin'. The 'Transmitter Holding Register' is labeled with a red 'W' and the word 'Register'.

2024/3/20



2024 Tamkang University

24

## ❖ 第二次迴圈

```

p348UART.s
2 PINSEL0 EQU 0xE002C000 ; controls the function of the pins
3 UOSTART EQU 0xE000C000 ; start of UART0 registers
4 LCR0 EQU 0xC ; line control register for UART0
5 LSR0 EQU 0x14 ; line status register for UART0
6 RAMSTART EQU 0x40000000 ; start of onboard RAM for 2104
7
8 start
9 LDR sp, = RAMSTART ; set up stack pointer
10 BL UARTConfig ; initialize/configure UART0
11 LDR r1, = CharData ; starting address of characters
12 Loop
13 LDRB r0, [r1], #1 ; load character, increment address
14 CMP r0, #0 ; null terminated?
15 BLNE Transmit ; send character to UART
16 BNE Loop ; continue if not a '0'
17 done B done ; otherwise we e done
18
19 ; Subroutine UARTConfig
20 ; This subroutine configures the I/O pins first. It
21 ; then sets up the UART control register. The
22 ; parameters
23 ; are set to 8 bits, no parity and 1 stop bit.
24 ; Registers used:
25 ; r5 - scratch register
26 ; r6 - scratch register
27 ; inputs: none
28 ; outputs: none
29
30 UARTConfig
31 STMIA sp!, {r5, r6, lr}
32 LDR r5, = PINSEL0 ; base address of register
33 LDR r6, [r5] ; get contents
34 BIC r6, r6, #0xF ; clear out lower nibble
35 ORR r6, r6, #0x5 ; sets P0.0 to Tx0 and P0.1 to Rx0
36 STR r6, [r5] ; r/modifw back to register
                
```

Universal Asynchronous Receive Transmit 0 (UART0)

Line Control

U0LCR: 0x03

Word Length: 8 bits

Stop Bits: 1

Parity: Odd Parity

☐ DLAB

☐ Break Control

☐ Parity Enable

Line Status

U0LSR: 0x00

☐ Receiver Data Ready (RDR)

☐ Overrun Error (OE)

☐ Parity Error (PE)

☐ Framing Error (FE)

☐ Break Interrupt (BI)

☐ Tx Holding Register Empty (THRE)

☐ Transmitter Empty (TEMT)

☐ Error in Rx FIFO (RXFE)

Interrupt Enable

U0IER: 0x00

☐ RBR IE

☐ THRE IE

☐ Rx Line Status IE

Interrupt ID & FIFO Control

U0IIR/FCR: 0x01 ☐ FIFO Enable

Interrupt: None

Rx Trigger: Level 0 (1)

☐ Rx FIFO Reset ☐ Tx FIFO Reset

Divisor Latch

U0DLL: 0x61

U0DLM: 0x00

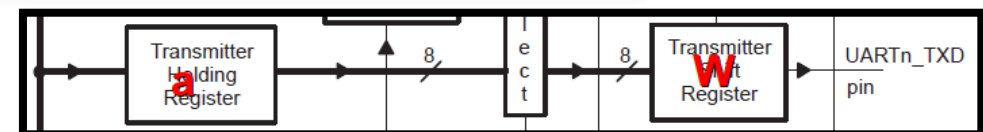
Baudrate: 1932

Receiver & Transmitter Registers

U0RBR/THR: 0x00


Scratch Pad Register

U0SCR: 0x00



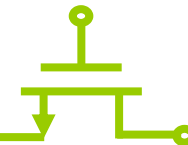
The diagram illustrates the UART0 hardware components and data flow. It shows a 'Transmitter Holding Register' (labeled 'a') connected to a 'Transmitter Register' (labeled 'W'). The data path is 8 bits wide, indicated by '8' and 'e c t'. The output of the Transmitter Register is connected to the 'UARTn\_TXD pin'.

2024/3/20



2024 Tamkang University

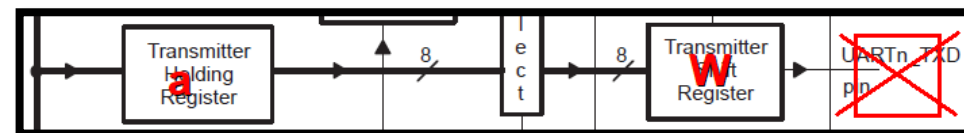
25

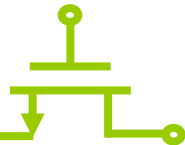


- ❖ 第三次迴圈-無窮迴圈(wait)。
- ❖ UART#1視窗無法顯示，因為使用F11**沒有模擬的輸出端**，使傳出去的值卡在Transmit holding register裡面，當在判斷THRE的狀態時，就會因為有值在裡面不斷的迴圈**(紅色框框)**。

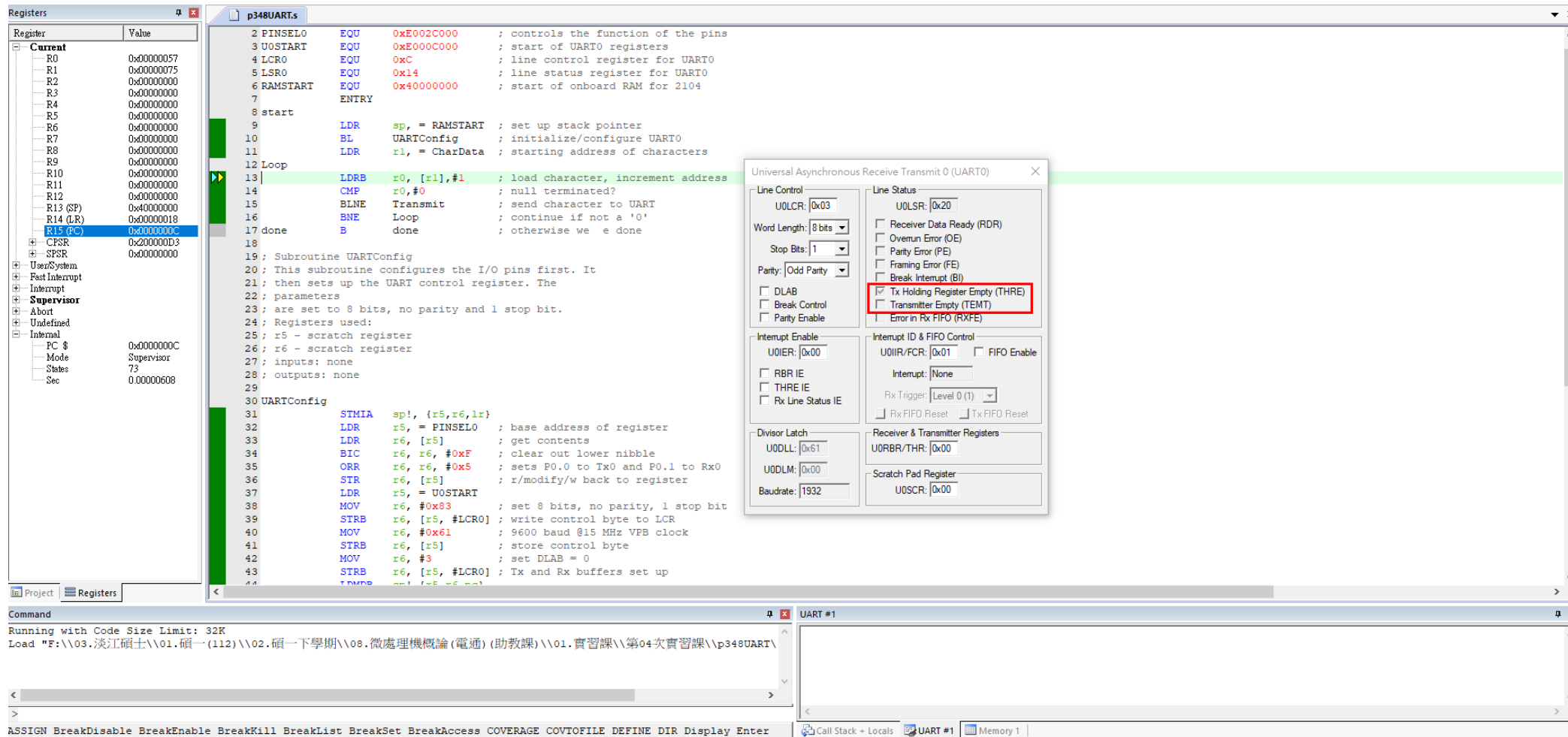
```

56 Transmit
57     STMIA    sp!, {r5, r6, lr}
58     LDR      r5, = U0START
59 wait  LDRB    r6, [r5, #LSR0] ; get status of buffer
60     TST      r6, #0x20         ; buffer empty?
61     BEQ      wait             ; spin until buffer's empty
62     STRB     r0, [r5]
63     LDMDB    sp!, {r5, r6, pc}
64 CharData
65     DCB      "Watson. Come quickly!", 0
66     END
    
```





## ❖ 第一次迴圈

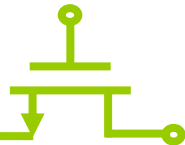


The screenshot displays the Keil MDK-ARM IDE interface. The main window shows the assembly code for the file `p348UART.s`. The code includes comments for pin selection, UART initialization, and a loop for transmitting data. The `UARTConfig` subroutine is highlighted, showing the configuration of the UART control register (LCR0) for 8 bits, no parity, and 1 stop bit, and the setting of the divisor latch (U0DLR) for 9600 baud.

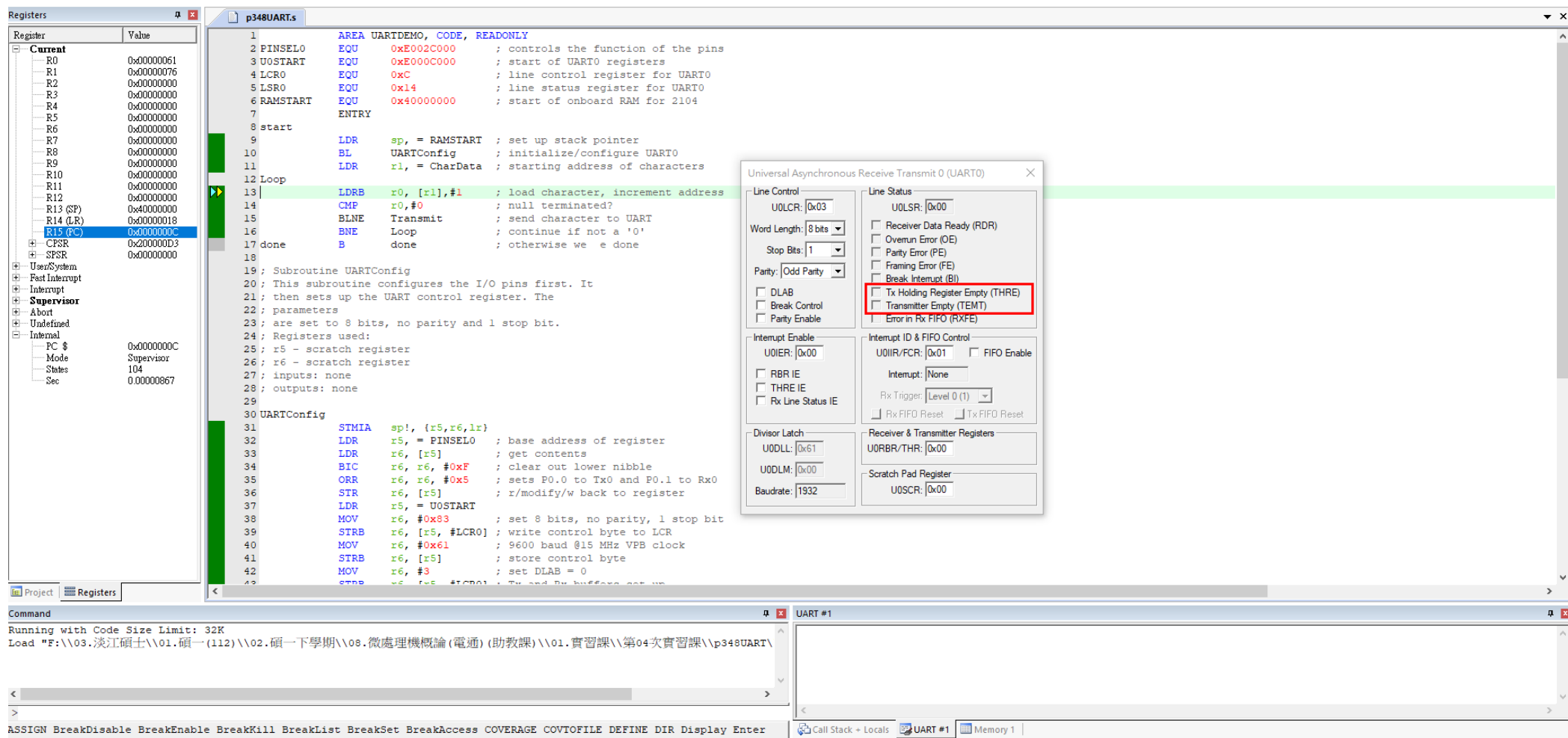
On the left, the **Registers** window shows the current state of the processor registers, with `R15 (PC)` highlighted at address `0x0000000C`.

On the right, the **Universal Asynchronous Receive Transmitter 0 (UART0)** configuration dialog is open. The **Line Control** tab is selected, showing the configuration for the UART0. The **Line Status** section is expanded, and the **Tx Holding Register Empty (THRE)** and **Transmitter Empty (TEMT)** options are checked, indicating that the UART is configured for interrupt-driven transmission.

At the bottom, the **Command** window shows the command `Running with Code Size Limit: 32K` and the load path for the assembly file. The **UART #1** window is also visible, showing the UART configuration details.

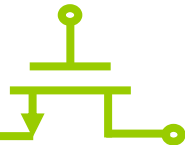


## ❖ 第二次迴圈

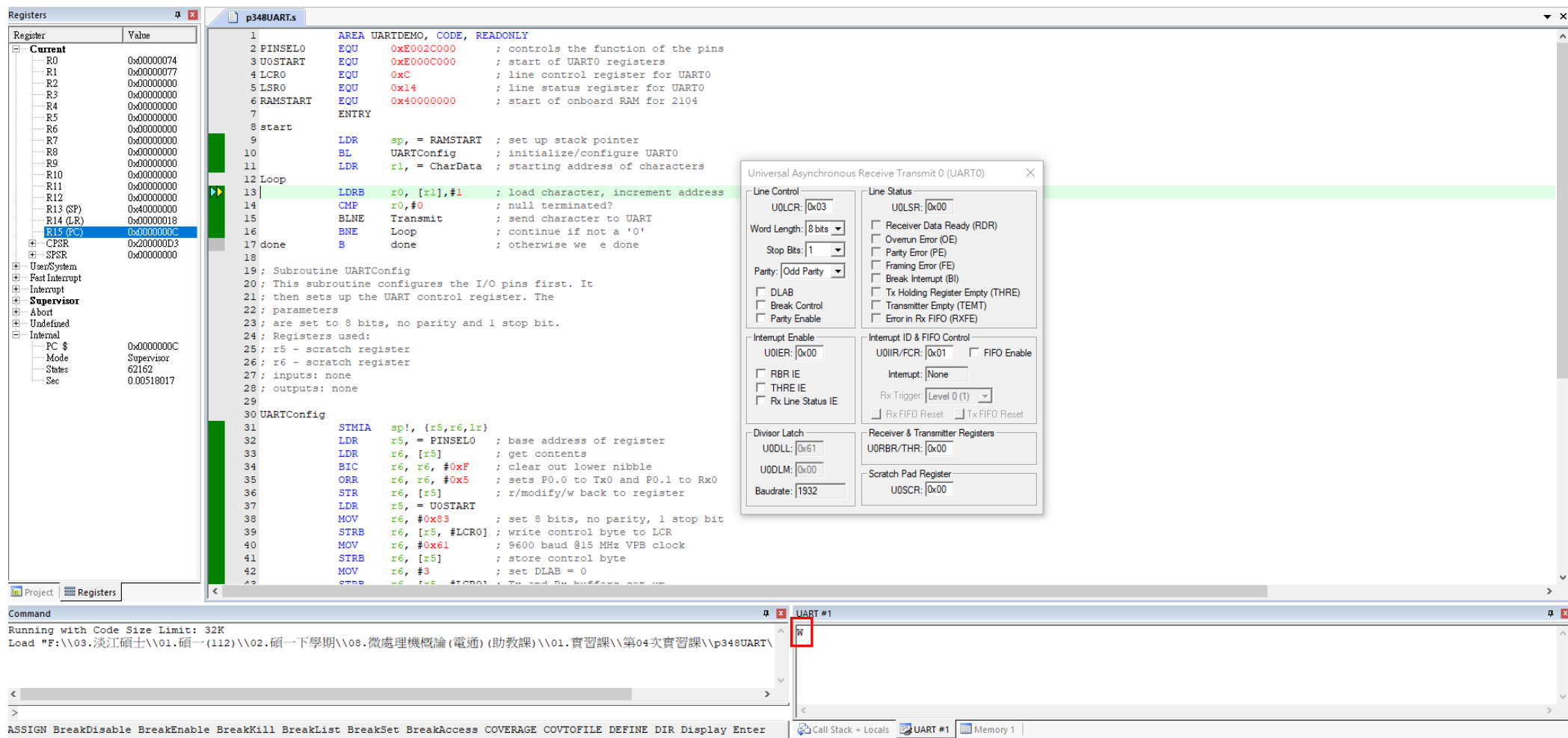


The screenshot displays the Keil MDK-ARM IDE interface. On the left, the 'Registers' window shows the current state of various registers, with R15 (PC) highlighted at address 0x0000000C. The main window shows the assembly code for 'p348UART.s'. The code is organized into sections: 'AREA UARTDEMO, CODE, READONLY', 'ENTRY', and 'Subroutine UARTConfig'. The 'Loop' section (lines 12-17) is highlighted in green, indicating the current execution point. The 'UARTConfig' subroutine (lines 19-42) configures the UART0 pins and registers. A 'Universal Asynchronous Receive Transmit 0 (UART0)' configuration dialog is open, showing settings for Line Control (UOLCR: 0x03), Line Status (UOLSR: 0x00), Word Length (8 bits), Stop Bits (1), Parity (Odd Parity), and Interrupt Enable (UOIER: 0x00). The 'Tx Holding Register Empty (THRE)' and 'Transmitter Empty (TEMT)' options are checked and highlighted with a red box. The 'Command' window at the bottom shows the command 'Running with Code Size Limit: 32K' and the load path for the project.





## ❖ 第三次迴圈



The screenshot displays the Keil IDE interface during the assembly of the file `p348UART.s`. The main window shows the assembly code, with the third loop iteration highlighted in green. The `Registers` window on the left shows the current state of the registers, with `R15 (PC)` highlighted. The `Universal Asynchronous Receive Transmitter 0 (UART0)` configuration dialog is open, showing various settings for the UART0 peripheral.

**Assembly Code (p348UART.s):**

```

1  AREA UARTDEMO, CODE, READONLY
2  PINSEL0 EQU 0xE002C000 ; controls the function of the pins
3  U0START EQU 0xE000C000 ; start of UART0 registers
4  LCR0 EQU 0xC ; line control register for UART0
5  LSR0 EQU 0x14 ; line status register for UART0
6  RAMSTART EQU 0x40000000 ; start of onboard RAM for 2104
7  ENTRY
8  start
9      LDR sp, = RAMSTART ; set up stack pointer
10     BL UARTConfig ; initialize/configure UART0
11     LDR r1, = CharData ; starting address of characters
12 Loop
13     LDRB r0, [r1], #1 ; load character, increment address
14     CMP r0, #0 ; null terminated?
15     BLNE Transmit ; send character to UART
16     BNE Loop ; continue if not a '0'
17 done B done ; otherwise we e done
18
19 ; Subroutine UARTConfig
20 ; This subroutine configures the I/O pins first. It
21 ; then sets up the UART control register. The
22 ; parameters
23 ; are set to 8 bits, no parity and 1 stop bit.
24 ; Registers used:
25 ; r5 - scratch register
26 ; r6 - scratch register
27 ; inputs: none
28 ; outputs: none
29
30 UARTConfig
31     SIMIA sp!, {r5, r6, lr}
32     LDR r5, = PINSEL0 ; base address of register
33     LDR r6, [r5] ; get contents
34     BIC r6, r6, #0xF ; clear out lower nibble
35     ORR r6, r6, #0x5 ; sets P0.0 to Tx0 and P0.1 to Rx0
36     STR r6, [r5] ; r/modify/w back to register
37     LDR r5, = U0START
38     MOV r6, #0x83 ; set 8 bits, no parity, 1 stop bit
39     STRB r6, [r5, #LCR0] ; write control byte to LCR
40     MOV r6, #0x61 ; 9600 baud @15 MHz VPB clock
41     STRB r6, [r5] ; store control byte
42     MOV r6, #3 ; set DLAB = 0
43     STRB r6, [r5, #LCR0] ; Tx and Rx buffers get up

```

**Registers Window:**

| Register       | Value      |
|----------------|------------|
| Current        | 0x00000074 |
| R0             | 0x00000077 |
| R1             | 0x00000000 |
| R2             | 0x00000000 |
| R3             | 0x00000000 |
| R4             | 0x00000000 |
| R5             | 0x00000000 |
| R6             | 0x00000000 |
| R7             | 0x00000000 |
| R8             | 0x00000000 |
| R9             | 0x00000000 |
| R10            | 0x00000000 |
| R11            | 0x00000000 |
| R12            | 0x00000000 |
| R13 (SP)       | 0x40000000 |
| R14 (LR)       | 0x00000018 |
| R15 (PC)       | 0x0000000C |
| CPSR           | 0x200000D3 |
| SFSR           | 0x00000000 |
| User/System    |            |
| Fast Interrupt |            |
| Interrupt      |            |
| Supervisor     |            |
| Abort          |            |
| Undefined      |            |
| Internal       |            |
| PC             | 0x0000000C |
| Supervisor     | 62162      |
| States         | 0.00518017 |
| Sec            |            |

**UART0 Configuration Dialog:**

- Line Control:** UOLCR: 0x03, Word Length: 8 bits, Stop Bits: 1, Parity: Odd Parity.
- Line Status:** UOLSR: 0x00.
- Interrupt Enable:** UOIER: 0x00, RBR IE, THRE IE, Rx Line Status IE.
- Interrupt ID & FIFO Control:** UOIR/FCR: 0x01, FIFO Enable, Interrupt: None, Rx Trigger: Level 0 (1).
- Divisor Latch:** UODLL: 0x61, UODLM: 0x00, Baudrate: 1932.
- Receiver & Transmitter Registers:** UORBR/THR: 0x00, UOSCR: 0x00.

**Command Window:**

```

Running with Code Size Limit: 32K
Load "F:\03.淡江碩士\01.碩一(112)\02.碩一下學期\08.微處理機概論(電通)(助教課)\01.實習課\第04次實習課\p348UART\

```

## ❖ 最後一次迴圈

**Registers**

| Register | Value      |
|----------|------------|
| R0       | 0x00000000 |
| R1       | 0x00000000 |
| R2       | 0x00000000 |
| R3       | 0x00000000 |
| R4       | 0x00000000 |
| R5       | 0x00000000 |
| R6       | 0x00000000 |
| R7       | 0x00000000 |
| R8       | 0x00000000 |
| R9       | 0x00000000 |
| R10      | 0x00000000 |
| R11      | 0x00000000 |
| R12      | 0x00000000 |
| R13 (SP) | 0x40000000 |
| R14 (LR) | 0x00000018 |
| R15 (PC) | 0x0000001C |
| CPSR     | 0x600000D3 |
| SFSR     | 0x00000000 |
| PC \$    | 0x0000001C |
| Mode     | 1179680    |
| States   | 0.09830667 |
| Sec      |            |

**p348UART.s**

```

4 LCR0 EQU 0xC ; line control register for UART0
5 LSR0 EQU 0x14 ; line status register for UART0
6 RAMSTART EQU 0x40000000 ; start of onboard RAM for 2104
7 ENTRY
8 start
9 LDR sp, = RAMSTART ; set up stack pointer
10 BL UARTConfig ; initialize/configure UART0
11 LDR r1, = CharData ; starting address of characters
12 Loop
13 LDRB r0, [r1], #1 ; load character, increment address
14 CMP r0, #0 ; null terminated?
15 BLNE Transmit ; send character to UART
16 BNE Loop ; continue if not a '0'
17 done B done ; otherwise we e done
18
19 ; Subroutine UARTConfig
20 ; This subroutine configures the I/O pins first. It
21 ; then sets up the UART control register. The
22 ; parameters
23 ; are set to 8 bits, no parity and 1 stop bit.
24 ; Registers used:
25 ; r5 - scratch register
26 ; r6 - scratch register
27 ; inputs: none
28 ; outputs: none
29
30 UARTConfig
31 STMIA sp!, {r5, r6, lr}
32 LDR r5, = PINSEL0 ; base address of register
33 LDR r6, [r5] ; get contents
34 BIC r6, r6, #0xF ; clear out lower nibble
35 ORR r6, r6, #0x5 ; sets P0.0 to Tx0 and P0.1 to Rx0
36 STR r6, [r5] ; r/modify/w back to register
37 LDR r5, = UOSTART
38 MOV r6, #0x83 ; set 8 bits, no parity, 1 stop bit
39 STRB r6, [r5, #LCR0] ; write control byte to LCR
40 MOV r6, #0x61 ; 9600 baud @15 MHz VPB clock
41 STRB r6, [r5] ; store control byte
42 MOV r6, #3 ; set DLAB = 0
43 STRB r6, [r5, #LCR0] ; Tx and Rx buffers set up
44 LMDB sp!, {r5, r6, pc}
45
46 ; Subroutine Transmit

```

**Universal Asynchronous Receive Transmit 0 (UART0)**

**Line Control**

UOLCR: 0x03

Word Length: 8 bits

Stop Bits: 1

Parity: Odd Parity

☐ DLAB

☐ Break Control

☐ Parity Enable

**Interrupt Enable**

UOIER: 0x00

☐ RBR IE

☐ THRE IE

☐ Rx Line Status IE

**Divisor Latch**

UODLL: 0x61

UODLM: 0x00

Baudrate: 1932

**Line Status**

UOLSR: 0x00

☐ Receiver Data Ready (RDR)

☐ Overrun Error (OE)

☐ Parity Error (PE)

☐ Framing Error (FE)

☐ Break Interrupt (BI)

☐ Tx Holding Register Empty (THRE)

☐ Transmitter Empty (TEMT)

☐ Error in Rx FIFO (RXFE)

**Interrupt ID & FIFO Control**

UOIR/FCR: 0x01 ☐ FIFO Enable

Interrupt: None

Rx Trigger: Level 0 (1)

☐ Rx FIFO Reset ☐ Tx FIFO Reset

**Receiver & Transmitter Registers**

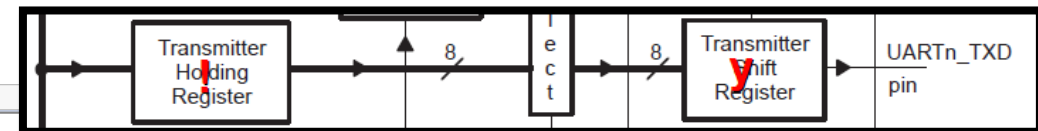
UOBR/THR: 0x00

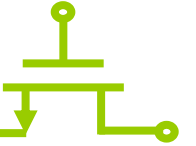
**Scratch Pad Register**

UOSCR: 0x00

**UART #1**

Watson. Come quick!





**Registers**

| Register | Value      |
|----------|------------|
| R0       | 0x00000000 |
| R1       | 0x00000000 |
| R2       | 0x00000000 |
| R3       | 0x00000000 |
| R4       | 0x00000000 |
| R5       | 0x00000000 |
| R6       | 0x00000000 |
| R7       | 0x00000000 |
| R8       | 0x00000000 |
| R9       | 0x00000000 |
| R10      | 0x00000000 |
| R11      | 0x00000000 |
| R12      | 0x00000000 |
| R13 (SP) | 0x00000000 |
| R14 (LR) | 0x00000000 |
| R15 (PC) | 0x00000000 |
| CPSR     | 0x000000D3 |
| SFSR     | 0x00000000 |

PC: 0x00000000  
Mode: Supervisor  
States: 0  
Sec: 0.00000000

**p348UART.s**

```

1 AREA UARTDEMO, CODE, READONLY
2 PINSEL0 EQU 0xE002C000 ; controls the function of the pins
3 UOSTART EQU 0xE000C000 ; start of UART0 registers
4 LCR0 EQU 0xC ; line control register for UART0
5 LSRO EQU 0x14 ; line status register for UART0
6 RAMSTART EQU 0x40000000 ; start of onboard RAM for 2104
7 ENTRY
8 start
9 LDR sp, = RAMSTART ; set up stack pointer
10 BL UARTConfig ; initialize/configure UART0
11 LDR r1, = CharData ; starting address of characters
12 Loop
13 LDRB r0, [r1], #1 ; load character, increment address
14 CMP r0, #0 ; null terminated?
15 BLNE Transmit ; send character to UART
16 BNE Loop ; continue if not a '0'
17 done
18 B done ; otherwise we are done
19 ; Subroutine UARTConfig
20 ; This subroutine configures the I/O pins first. It
21 ; then sets up the UART control register. The
22 ; parameters
23 ; are set to 8 bits, no parity and 1 stop bit.
24 ; Registers used:
25 ; r5 - scratch register
26 ; r6 - scratch register
27 ; inputs: none
28 ; outputs: none
29
30 UARTConfig
31 STMIA sp!, {r5, r6, lr}
32 LDR r5, = PINSEL0 ; base address of register
33 LDR r6, [r5] ; get contents
34 BIC r6, r6, #0xF ; clear out lower nibble
35 ORR r6, r6, #0x5 ; sets P0.0 to Tx0 and P0.1 to Rx0
36 STR r6, [r5] ; r/modify/w back to register
37 LDR r5, = UOSTART
38 MOV r6, #0x83 ; set 8 bits, no parity, 1 stop bit
39 STRB r6, [r5, #LCR0] ; write control byte to LCR
40 MOV r6, #0x61 ; 9600 baud @15 MHz VPB clock
41 STRB r6, [r5] ; store control byte
42 MOV r6, #3 ; set DLAB = 0
43 STRB r6, [r5, #LSR0] ; Tx and Rx buffers get up
          
```

Universal Asynchronous Receiver/Transmitter (UART) Configuration

**Line Control**

UOLCR: 0x03

Word Length: 8 bits

Stop Bits: 1

Parity: Odd Parity

☐ DLAB

☐ Break Control

☐ Parity Enable

**Interrupt Enable**

UOIER: 0x00

☐ RBR IE

☐ THRE IE

☐ Rx Line Status IE

**Divisor Latch**

UODLL: 0x61

UODLM: 0x00

Baudrate: 1932

**Line Status**

UOLSR: 0x60

☐ Receiver Data Ready (RDR)

☐ Overrun Error (OE)

☐ Parity Error (PE)

☐ Framing Error (FE)

☐ Break Interrupt (BI)

☒ Tx Holding Register Empty (THRE)

☒ Transmitter Empty (TEMT)

☐ Error in Rx FIFO (RXFE)

**Interrupt ID & FIFO Control**

UOIR/FCR: 0x01 ☐ FIFO Enable

Interrupt: None

Rx Trigger: Level 0 (1)

☐ Rx FIFO Reset ☐ Tx FIFO Reset

**Receiver & Transmitter Registers**

UORBR/THR: 0x00

**Scratch Pad Register**

UOSCR: 0x00

**Command**

Running with Code Size Limit: 32K

Load "F:\03.淡江碩士\01.碩一(112)\02.碩一下學期\08.微處理機概論(電通)(助教課)\01.實習課\第04次實習課\p348UART\

UART #1

Watson. Come quickly!

ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet BreakAccess COVERAGE COVTOFILE DEFINE DIR Display Enter

Call Stack + Locals | UART #1 | Memory 1

→兩種情況THR、TSR的值會被傳出來

1. 程式結束時(F5)
2. 重新存取UOSTART時(Receive)

ing this data? In the simulation tools, there is a serial window that can accept data from a UART, driving the necessary handshake lines that are normally attached to the receiver. The assembly code for our transmitter routine looks like the following:

只要狀態沒成立，CPU就要一直等待，沒辦法做其他事情。

```
wait: LDR    r5, =U0START
      LDRB   r6, [r5, #LSR0]
      CMP TST r6, #0x20
      BEQ    wait
      STRB   r0, [r5]
```

**BUSY WAITING**  
(CPU的狀態)

**LOOP**  
**POLL(CPU主動詢問周邊晶片)**

```
; get status of buffer
; buffer empty?
; spin until buffer's empty
```

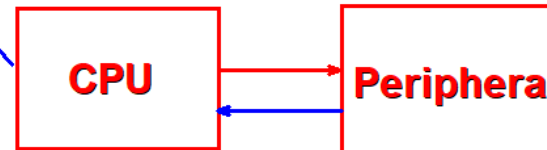
改進：周邊在任何狀態發生改變時，主動通知CPU，省去BUSY WAITING時間。

編號 → 讓CPU知道什麼情況發生時，執行哪一個程式。

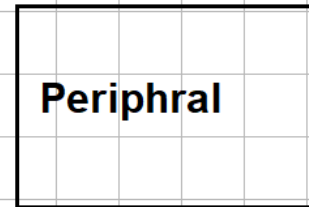
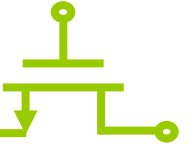
1. BUFFER EMPTY
2. DATA READY

**INTERRUPT**

例：門鈴壞掉→busy waiting



# p347 poll & busy waiting



INTERRUPT(並告知CPU編號)

Interrupt number

CPU

藉由Interrupt vector table找到對應執行程式的初始位置  
並在記憶體裡找到對應的程式並執行

- 1. first instruction address
- 2. address

Memory

TSR -> Terminate and Stay Resident(常駐程式)

只要是 ISR -> Interrupt Service Routine



# *Q&A*

***Thanks for your attention !!***