

embedded/USART

- USART簡介
- USART主要特性
- USART功能介紹
 - USART Block Diagram
 - USART 特性描述
 - 傳送器
 - 資料的傳送
 - 傳送器的設定
 - 傳送斷開符號
 - 傳送空間符號
 - 接收器
 - 起始位偵測
 - 資料的接收
 - 接收器的設定
 - 接收斷開符號
 - 接收空間符號
 - 溢出錯誤
 - 噪音錯誤
 - Frame錯誤
 - Fractional baud rate generation的設定
- Modes
 - USART using DMA
 - USART 用 DMA 讀取 DATA 再傳送
 - USART 接收端經過 DMA 存放DATA
 - SmartCard Mode
 - SmartCard
 - register 設定
 - Smartcard mode Stop bit
 - parity error in smartcard mode
 - IrDA SIR ENDEC
 - IrDA
 - register 設定
 - IrDA SIR 實體層
 - USART SYNCHRONOUS MODE
 - register 設定
 - 特性
 - USART 與 SPI
 - HARDWARE FLOW CONTROL
 - RTS Flow Control
 - CTS Flow Control
- PARITY CONTROL
 - Even/Odd parity
 - Even parity

- Odd parity
 - 接收後會做 parity checking
 - 傳送前會做 parity generation
- USART Register 總表
- CODE SECTION
 - 發送器測試
 - 接收器測試
- REFERENCE
- Q&A

USART簡介

串轉輸為CPU與周邊裝置或CPU與CPU間的資料傳輸方法之一，而USART(universal synchronous asynchronous receiver transmitter) 通用同步/非同步收發傳輸器，則常被用於一般的串轉輸應用中。

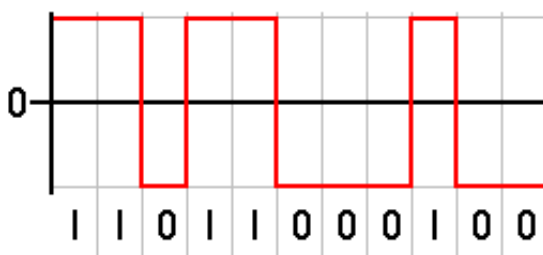
USART主要特性

- 全雙工的非同步通訊
- NRZ標準資料格式

NRZ(Nonreturn to Zero):不歸零編碼

這是一種傳送資訊的編碼方式，它以正脈波代表1，負脈波代表0，當訊號連續為'1'時，則保持正脈波，直到出現'0'為止

它的特色是編碼解碼較為簡單，但缺乏同步傳輸的能力，且無法提供較佳的訊號校正能力。



圖片來源：[wikipedia](#)

- 可程式化的資料長度 (8 or 8+1 bits)
- 可程式化的停止位元 (1 or 2 bits)
- 提供同步傳輸的CLK信號
- 藉由DMA的資料傳輸，每個USART都能用DMA發送和接收資料
- 獨立的發送器和接收器的Enable Bit(TE、RE)
- 傳輸檢測標誌

– 接收緩衝區滿(Receive buffer full, RXNE)

- 傳送緩衝區空(Transmit buffer empty, TXE)
- 傳輸結束(End of transmission flags, TC)
- 檢測控制
 - 發送檢測位(Transmits parity bit)
 - 對接收的資料進行檢測(Checks parity of received data byte)
- 4個錯誤檢測標誌
 - 溢出錯誤(Overrun error)
 - 噪音檢測(Noise detection)
 - Frame錯誤(Frame error)
 - 奇偶檢測錯誤(Parity error)
- 支援10種中斷
 - CTS改變(CTS changes, CTSIE)
 - LIN中斷檢測(LIN break detection, LBDIE)
 - 傳送緩衝區空(Transmit data register empty, TXEIE)
 - 傳送完成(Transmission complete, TCIE)
 - 接收緩衝區滿(Receive data register full, RXNEIE)
 - 空閒線路檢測(Idle line received, IDLEIE)
 - 溢出錯誤(Overrun error)
 - 在一般情況下，本身不產生中斷，在DMA情況下，則由EIE產生中斷，經檢驗USART_CR1的FE位可得知溢出錯誤
 - Frame錯誤(Framing error)
 - 在一般情況下，本身不產生中斷，而由RXNE產生中斷，經檢驗USART_CR1的FE位可得知Frame錯誤
 - 在DMA情況下，則由EIE產生中斷，經檢驗USART_CR1的FE位得知錯誤
 - 噪音錯誤(Noise error)
 - 在一般情況下，本身不產生中斷，而由RXNE產生中斷，經檢驗USART_CR1的NF位得知錯誤
 - 在DMA情況下，則由EIE產生中斷，經檢驗USART_CR1的NF位得知錯誤
 - 檢驗錯誤(Parity error, PEIE)
- 2種喚醒接收器的方式
 - Idle line 在接收端處於靜默(mute mode)時，可透過發送空閒符號(即所有位均為'1'的資

料)，喚醒接收端。

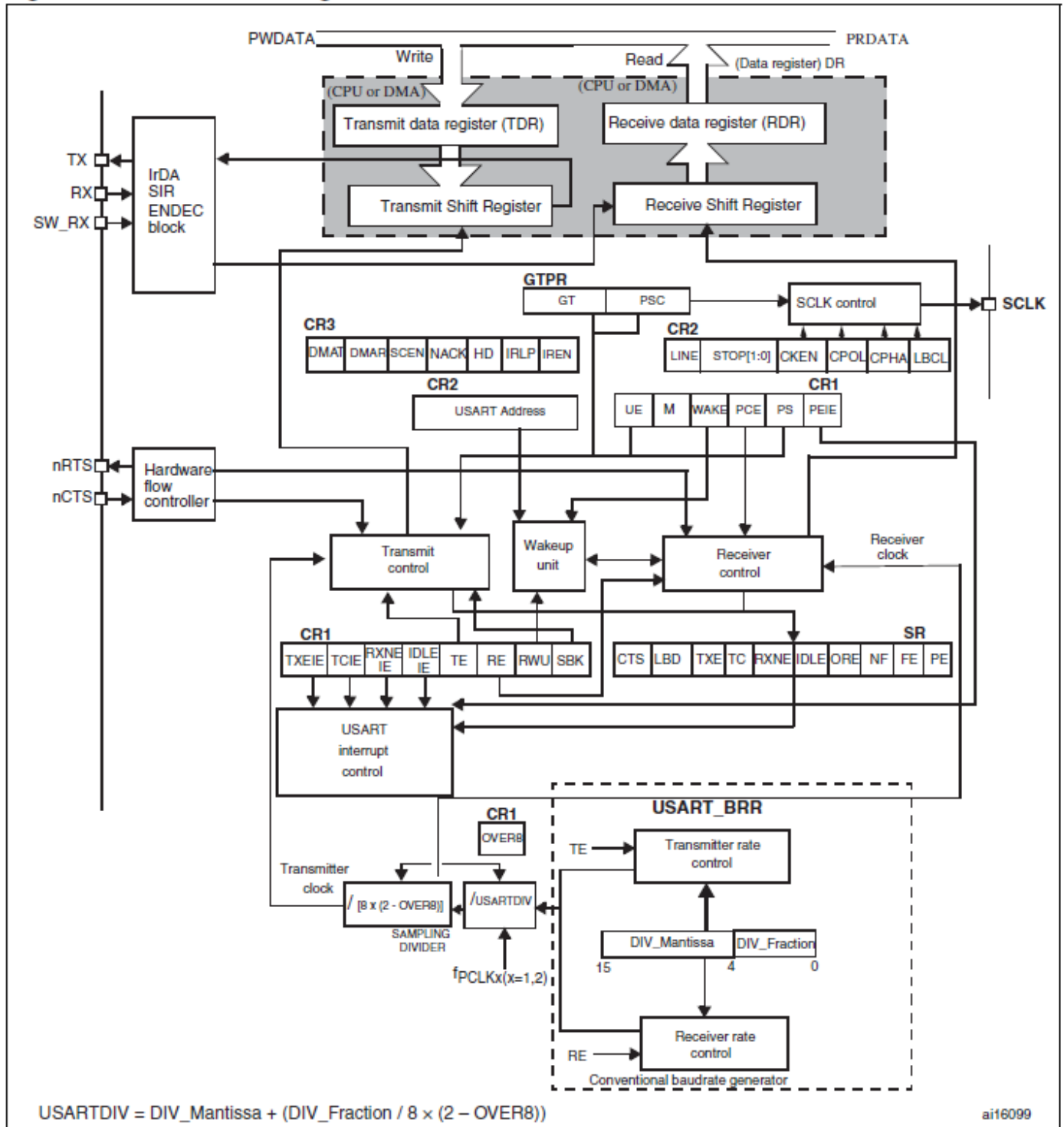
- Address bit MSB為'1'的資料被認為是地址，否則為一般資料。

在這資料中，接收端會將最後4bits與USART_CR2暫存器中的ADD位比較，若相同則清除RWU位，後面的資料將能正常接收。

USART功能介紹

USART Block Diagram

Figure 245. USART block diagram



Ref: RM0090 Reference Manual P.949

- 任何USART雙向通信至少需要兩個腳位：接收資料輸入(RX)和發送資料輸出(TX)
 - RX: 接收資料輸入，並藉由採樣的技術判斷資料及噪音
 - TX: 發送資料，當發送器被啟動時，如果沒有傳送數據，則TX保持高電位。在Single-wire half-duplex或Smartcard mode時，此I/O同時被用於資料的傳送和接收
- 根據USART_CR1暫存器中的M位選擇8或9位元決定資料長度(見圖)
- 使用fractional baud rate generator —— 12位整數和4位小數的表示方法，放在baud rate暫存器(USART_BRR)中
- 一個狀態暫存器(USART_SR)

- 資料暫存器(USART_DR)

另外在同步模式中，還需要其他腳位

- SCLK: 發送器的時鐘輸出，用於同步傳輸的時鐘

在Hardware flow control中:

- nCTS: 清除發送，若在高電位，則當目前資料傳送結束後，中斷下一次的資料傳送
- nRTS: 發送請求，若在低電位，則表示USART已經準備好接收資料

USART 特性描述

資料長度根據USART_CR1暫存器中的M位選擇8或9位元

9-bit word length (M bit is set), 1 stop bit



8-bit word length (M bit is reset), 1 stop bit

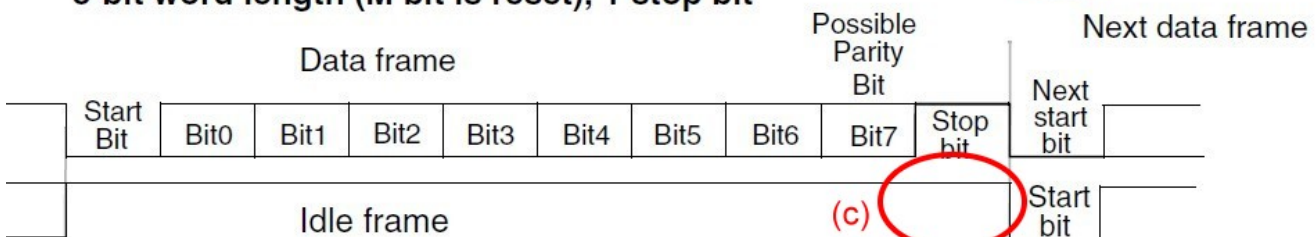


在起始位(start bit)期間，TX處於低電位，如圖中的(a)，在停止位期間，TX處於高電位，如圖中的(b)。

9-bit word length (M bit is set), 1 stop bit



8-bit word length (M bit is reset), 1 stop bit



另外空閒符號則全由'1'組成，包含資料的停止位元位數也是'1'，如圖中的(c)，後面接著下一個資料的開始位；

9-bit word length (M bit is set), 1 stop bit



8-bit word length (M bit is reset), 1 stop bit



中斷符號則全由'0'所組成，包含資料的停止位也是'0'，如圖中的(d)，

在中斷時，發送器會再插入1或2個停止位('1')以區分下一筆資料的起始位，如圖中的(e)

Ref: [RM0090 Reference Manual P.950](#)

傳送器

傳送器依據USART_CR1的M位狀態來決定發送8或9位元的資料。當transmit enable bit(TE)被設定時，資料放入transmit shift register後，經由TX腳位送出，同時，相對應的時鐘脈衝會由SCLK腳位輸出。

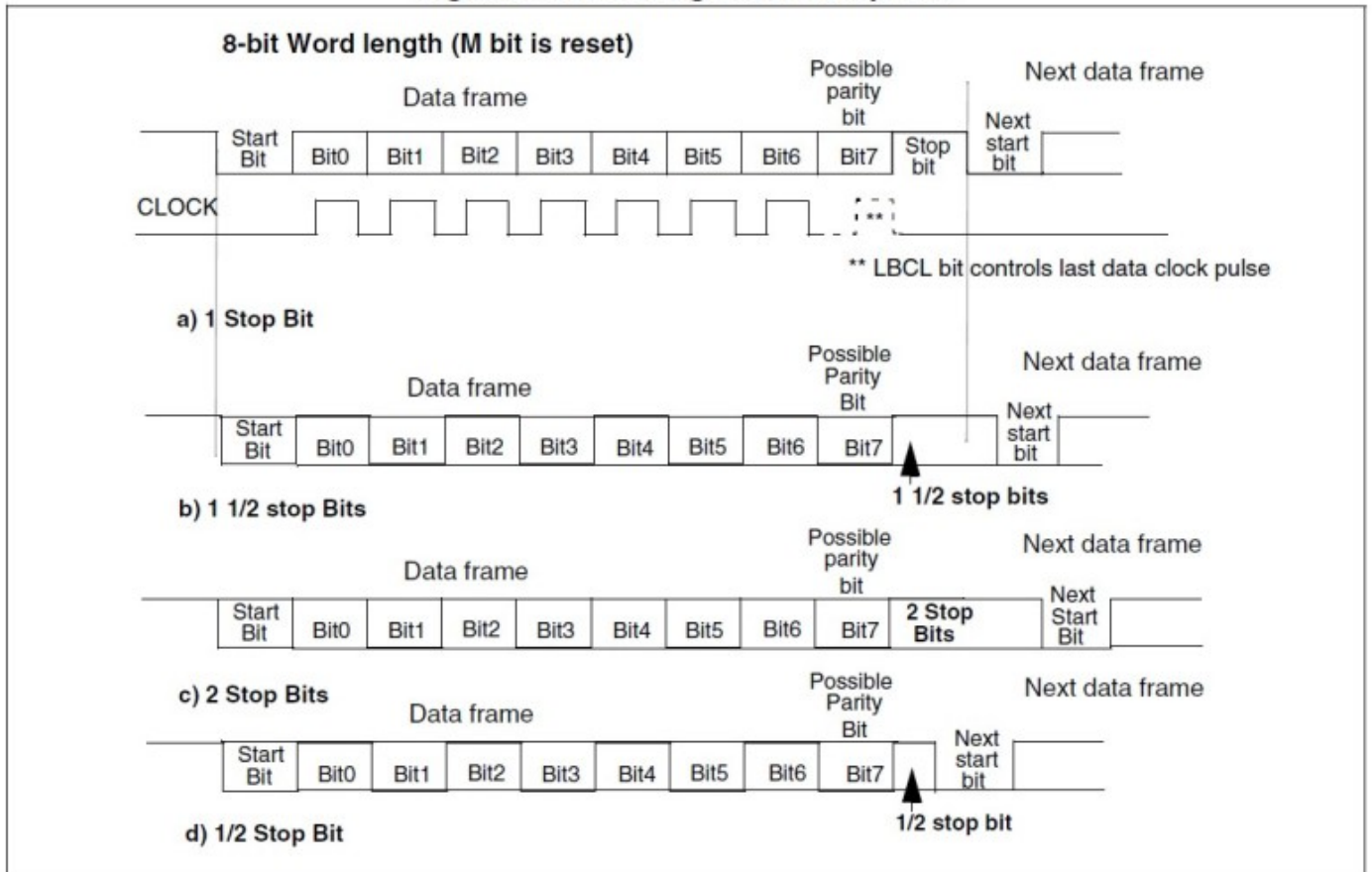
資料的傳送

在USART發送期間，TX首先傳送資料的最低有效位元(least significant bit)，因此在此模式中，USART_DR和transmit shift register之間包含一個緩衝器(TDR)。每個資料再傳送前都會有一個低電位的起始位；之後跟著的停止位元數目，則可由使用者決定0.5, 1, 1.5或2

- 1 bit的stop bit: 預設的默認停止位元數
- 2 bits的stop bit: normal USART, single-wire 和 modem modes
- 0.5 bits的stop bit: Smartcard mode接收數據用
- 1.5 bits的stop bit: Smartcard mode發送數據用

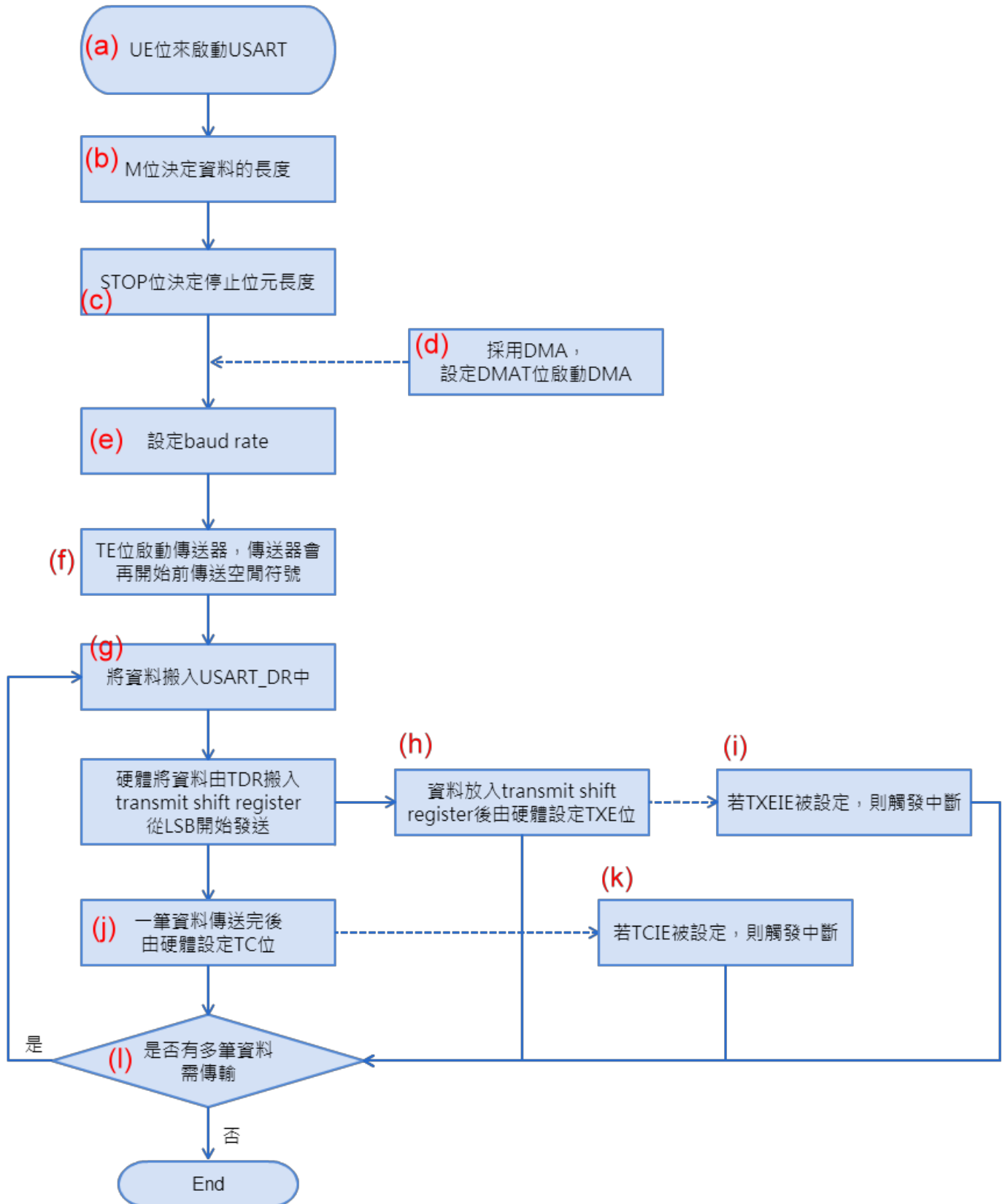
stop bits其實不算是個bit，他是傳輸結束後的一段時間(period)，用以區隔每個傳輸的資料，其功用是在非同步傳輸的時候可以告訴接收器，資料傳輸已經結束。透過增加stop bits的長度，可讓接收器能有足夠的時間可以處理該資料

另外，由於transmit shift register搬移到TDR中最少需要1/2 baud clock，因此在Smartcard mode的接收中，最少必須設定0.5 bit的stop bits

Figure 298. Configurable stop bits

Ref: [RM0090 Reference Manual P.952](#)

傳送器的設定



1. 設定USART_CR1暫存器的UE位來啟動傳輸，如圖中的(a)
2. 設定USART_CR1暫存器的M位決定資料長度，如圖中的(b)
3. 設定USART_CR2暫存器中的STOP位來決定停止位元的長度，如圖中的(c)
4. 採用多緩衝器的話，則須設定USART_CR3的DMAT啟動DMA，並設置DMA的暫存器，如圖中的(d)
5. 利用USART_BRR暫存器設定baud rate，如圖中的(e)
6. 設置USART_CR1的TE位，在第一筆資料傳送前，傳送一個空間的frame，如圖中的(f)
7. 將欲發送的資料放入USART_DR中，如圖中的(g)

8. 若有多筆資料要傳送，則重複步驟7.，如圖中的(l)

當資料放入USART_DR會由硬體清除TXE位，如圖中的(h)，則表示:

1. 資料已從TDR中進入transmit shift register，資料的發送已開始
2. TDR暫存器已被清空
3. 下一筆資料可放入USART_DR中

若TXEIE位的設置，則會產生一個中斷，如圖中的(i):

- 如果USART正在發送資料，對USART_DR的寫入會把資料移到TDR暫存器中，並在目前的資料傳送結束後把該筆資料移進transmit shift register中
- 如果USART沒有在發送資料，則對USART_DR的寫入會把資料直接放入transmit shift register中，並啟動傳送，當傳送開始時，硬體會立即設定TXE位

一個frame的資料發送完畢後，TC位會被設定，如圖中的(j)，如果USART_CR1中的TCIE有被設定，則會產生一個中斷，如圖中的(k)，先讀取USART_SR暫存器，再寫入USART_DR暫存器，則可清除TC位

傳送斷開符號

透過設定USART_CR1的SBK位，可以發送一個斷開符號，斷開符號的長度取決於M位。

如果SBK=1，則在目前的資料發送後，會再TX線上發送一個斷開符號，當傳送完成後，會由硬體恢復SBK位。

USART會由硬體在最後一個斷開符號的結束處插入一個'1'，確保能辨識下一個資料的起始位。

傳送空間符號

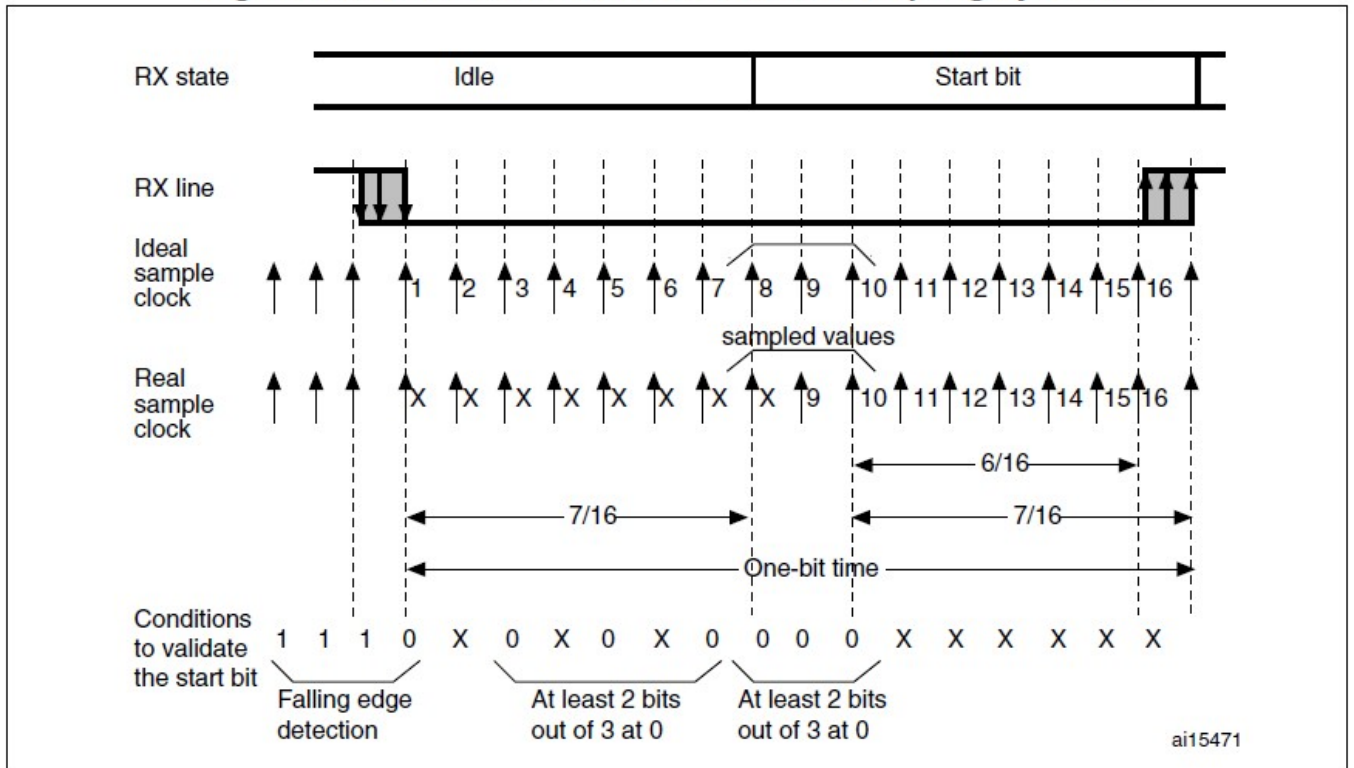
設置USART_CR1的TE位會使得USART在發送第一筆資料前，發送一個空間符號，喚醒接收端。

接收器

接收器依據USART_CR1 M位的狀態來決定接收8或9位元的資料。

起始位偵測

Figure 300. Start bit detection when oversampling by 16 or 8



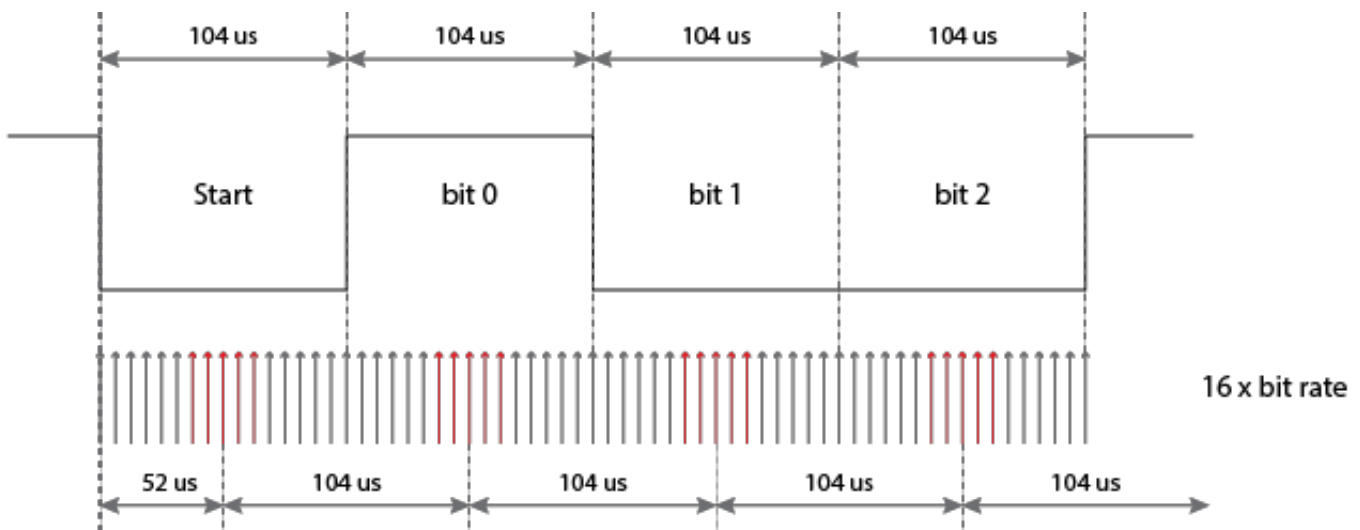
Ref: [RM0090 Reference Manual P.954](#)

在USART中，如果辨認出一個特殊的採樣序列(1 1 1 0 X 0 X 0 X 0 0 0 0)，則認定偵測到一個起始位。

如果該序列不完整，則接收端退回起始位偵測並回到空閒狀態，等待下一次的電壓下降。

如果三個採樣點上有僅有兩個是0(第3、5、7採樣點或8、9、10採樣點)，則依然判定為偵測到一個起始位，但NE(噪音標誌)會被設定

採樣的時間間隔



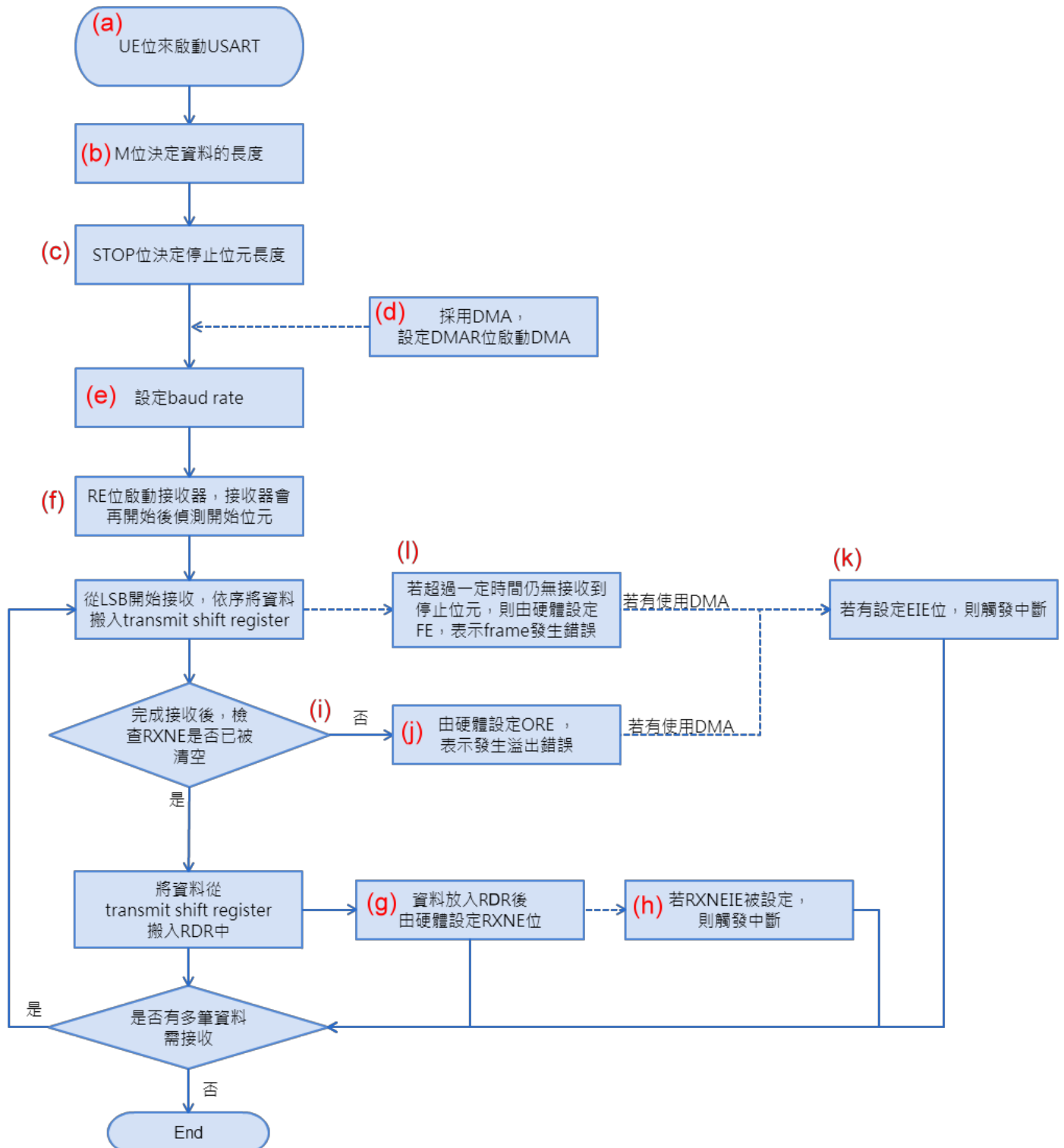
Ref: [UART receiver clock speed](#)

假設baud rate = 9600 bps，則一個bit的傳輸時間為104us，usart會在接收器啟動後的52us，開始採樣
若偵測到開始位元，則開始接收資料，反之則等待104us，再採樣一次

資料的接收

在USART接收期間，RX從資料最低有效位元(least significant bit)開始接收，因此在此模式中，USART_DR和received shift register之間包含一個緩衝器(RDR)。

接收器的設定



1. 設定USART_CR1暫存器的UE位來啟動USART接收，如圖中的(a)
2. 設定USART_CR1暫存器的M位決定資料長度，如圖中的(b)
3. 設定USART_CR2暫存器中的STOP位來決定停止位元的長度，如圖中的(c)

4. 採用多緩衝器接收資料，則須設定USART_CR3的DMAR啟動DMA，並設置DMA的暫存器，如圖中的(d)
5. 利用USART_BRR暫存器設定baud rate，如圖中的(e)
6. 設定USART_CR1暫存器中的RE位，啟動接收器，並開始偵測起始位，如圖中的(f)

當資料被接收到後:

1. 硬體會設定RXNE位，表示received shift register中的資料已移入RDR中，亦即資料已被接收並可被讀出，如圖中的(g)
2. 若USART_CR1中的RXNEIE被設定時，會產生一個中斷，如圖中的(h)
3. 資料接收期間如檢測到frame錯誤或是噪音、溢出錯誤等問題，相關的標誌將被設定(FE、NF、ORE)
4. 藉由讀取USART_DR可清除RXNE位，RXNE位必須要在下一資料接收前被清除，以免產生溢出錯誤
5. 在DMA接收時，RXNE在每個字元接收後被設置，並因DMA讀取RDR而被清除

接收斷開符號

USART在接收斷開符號後，可像處理frame錯誤一樣處理

接收空間符號

當空間符號被偵測到時，USART處理步驟如同一般資料一樣處理，但如果USART_CR1的IDLEIE被設置時，將會產生一個中斷

溢出錯誤

若RXNE沒有被覆位，此時又接收到一個新資料，則會發生溢出錯誤，如圖中的(i)

當溢出錯誤產生時:

1. USART_SR中的ORE位將被設置，如圖中的(j)
2. RDR中的內容將不會被清除，因此讀取USART_DR仍可以得到之前的資料
3. 若USART持續在接收中，則Received shift register中的資料將被覆蓋
4. 如果RXNEIE被設置，或是EIE(Error interrupt enable)和DMAR位被設定，則會產生一個中斷，如圖中的(k)
5. 依序讀取USART_SR和USART_DR暫存器，可清除ORE位

當ORE位被設置時，表示至少有一個資料已遺失，有以下兩種可能性:

1. 如果RXNE=1，表示之前的資料還在RDR中，且可被讀出
2. 如果RXNE=0，表示之前的資料已被讀走，RDR已無資料可被讀取，此種情況發生在讀取RDR中上一筆資料時，又接收到新的資料時發生。

噪音錯誤

透過不同的採樣技術，可以區分有效的輸入資料和噪音，並進行資料恢復。

透過設定USART_CR1中的OVER8位可選16或8次的採樣，見Fig. 250和Fig. 251:

- OVER8 = 1: 採用8次採樣，採樣的頻率較快(最高頻率為fPCLK/8)

- $OVER8 = 0$: 採用16次採樣，採樣的頻率最高為 $f_{PCLK}/16$

設定USART_CR3中的ONEBIT位可選則不同的採樣技術:

- $ONEBIT = 0$: 採樣資料中心的 3 bits，若此3 bits不相等，則NF位會被設定
- $ONEBIT = 1$: 只採樣中心的單一bit，此時NF的檢測將會被取消

當在資料接收中檢測到噪音時:

- NE會在RXNE位的升緣時被設定
- 無效的資料會從received shift register移入USART_DR暫存器中
- 在單一資料的接收下，不會有中斷產生，但透過NE和RXNE位的設置，由後者來產生中斷；

在多緩衝器的接收中，如果USART_CR3暫存器中的EIE位被設定，則會產生一個中斷

Figure 250. Data sampling when oversampling by 16

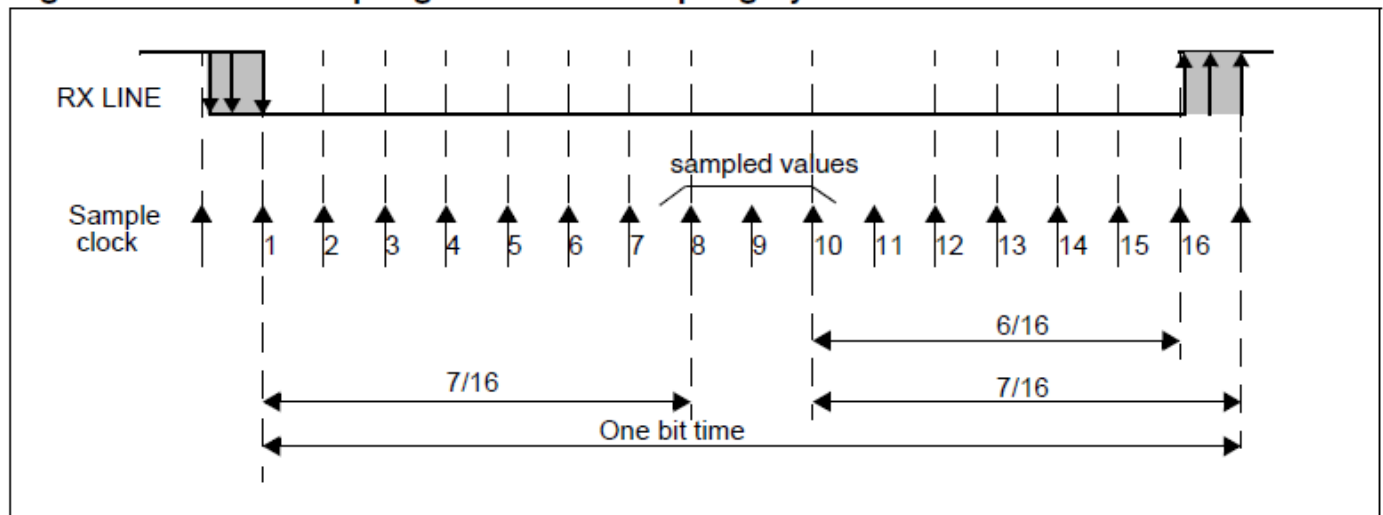


Figure 251. Data sampling when oversampling by 8

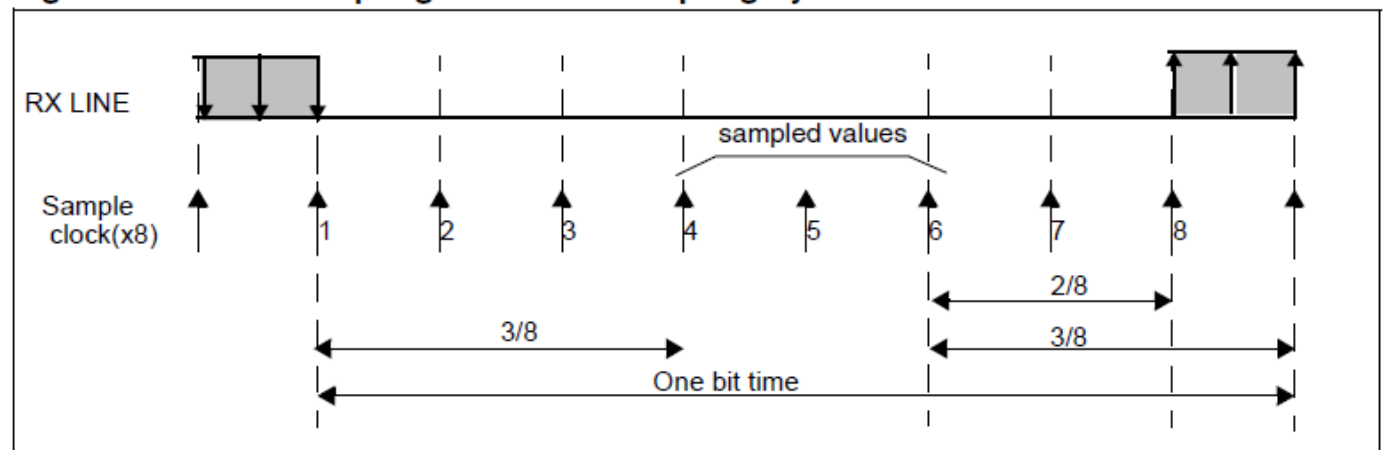


Table 105. Noise detection from sampled data

Sampled value	NE status	Received bit value
000	0	0
001	1	0
010	1	0
011	1	1
100	1	0
101	1	1

Table 105. Noise detection from sampled data (continued)

Sampled value	NE status	Received bit value
110	1	1
111	0	1

Ref: [RM0090 Reference Manual P.957~958](#)

Frame錯誤

由於沒有同步上或線路上大量的噪音，使得停止位沒有在預期的時間上接收和識別出來，則發生Frame錯誤

當Frame錯誤被檢測出時:

1. FE位被設定，如圖中的(l)
2. 無效的資料從received shift register移入USART_DR暫存器中
3. 在一般資料的接收下，不會有中斷產生，可藉由RXNEIE位的設置，在中斷中檢測FE位得知發生錯誤；

在DMA的接收中，如果USART_CR3暫存器中的EIE位被設定，則會產生一個中斷，如圖中的(k)

依序讀取USART_SR和USART_DR暫存器可恢復FE位

Fractional baud rate generation的設定

接收器和傳送器的Baud rate分別由USART_BRR設置USARTDIV的整數部分(Mantissa)及小數部分(Fraction)，計算方式如下所示:

Equation 1: Baud rate for standard USART (SPI mode included)

$$\text{Tx/Rx baud} = \frac{f_{\text{CK}}}{8 \times (2 - \text{OVER8}) \times \text{USARTDIV}}$$

Equation 2: Baud rate in Smartcard, LIN and IrDA modes

$$\text{Tx/Rx baud} = \frac{f_{\text{CK}}}{16 \times \text{USARTDIV}}$$

Ref: [RM0090 Reference Manual P.959](#)

其中USARTDIV為一個無號的定點數(unsigned fixed point number)，fCK為給周邊設備的時鐘。

- 當OVER8 = 0 時，小數部分佔USART_BRR的DIV_Fraction[3:0]，共 4 bits
- 當OVER8 = 1 時，小數部分佔USART_BRR的DIV_Fraction[2:0]，共 3 bits，其中 DIV_Fraction[3]應該保持'0'

USART_BRR被更新後，baud rate的計數器中的值也會同時被更新，因此在傳輸途中不應該更新USART_BRR中的值。另外，如果TE或RE被分別禁止，則baud rate的計數器也會停止計數

Modes

USART using DMA

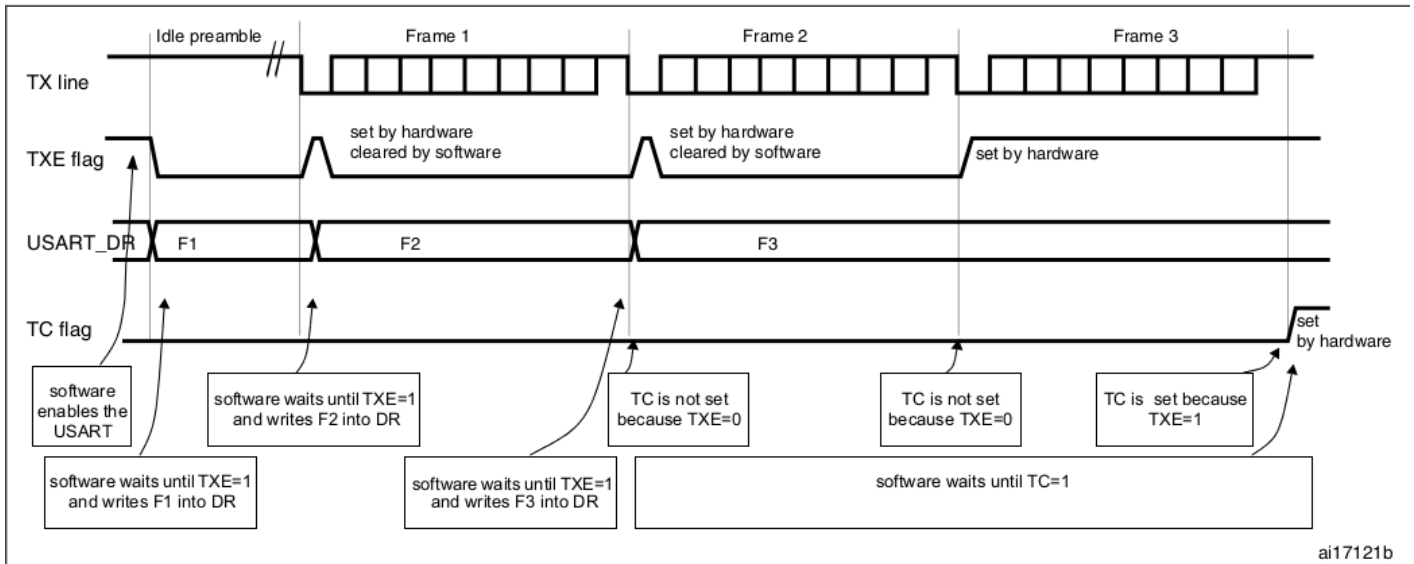
Ref: [Using The DMA controller on STM32](#)

利用UART傳送小量data的時候, 接收方可以利用interrupt 的方法, 令processor 把data 從recieve register 寫到 memory裡面; 但當連續傳送大量的data時, 如果能不使用到processor 做將data 寫入memory的動作, 可以減輕資源有限的微處理器的負擔.

Ref: [RM0090 Reference Manual P.981](#)

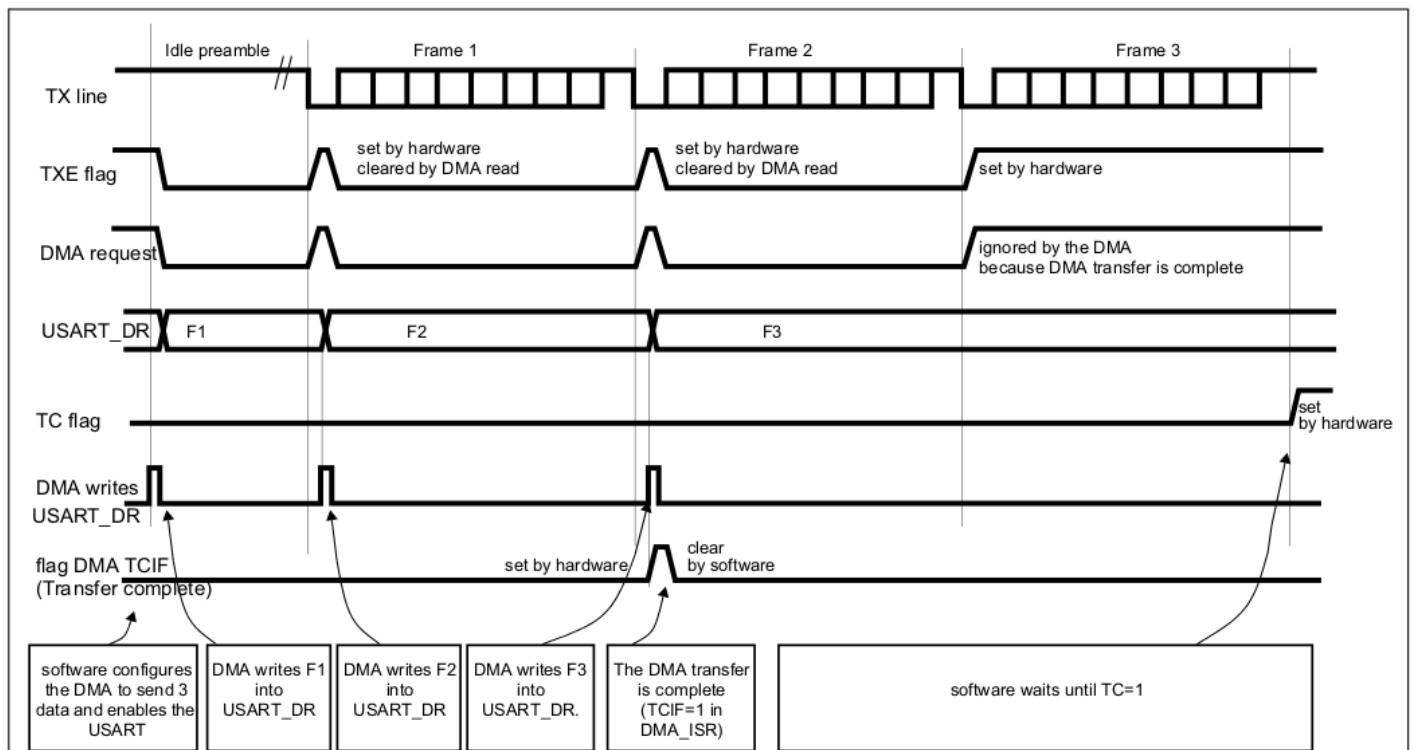
USART 用 DMA 讀取 DATA 再傳送

Figure 299. TC/TXE behavior when transmitting



(一般 USART 傳送)

Figure 315. Transmission using DMA

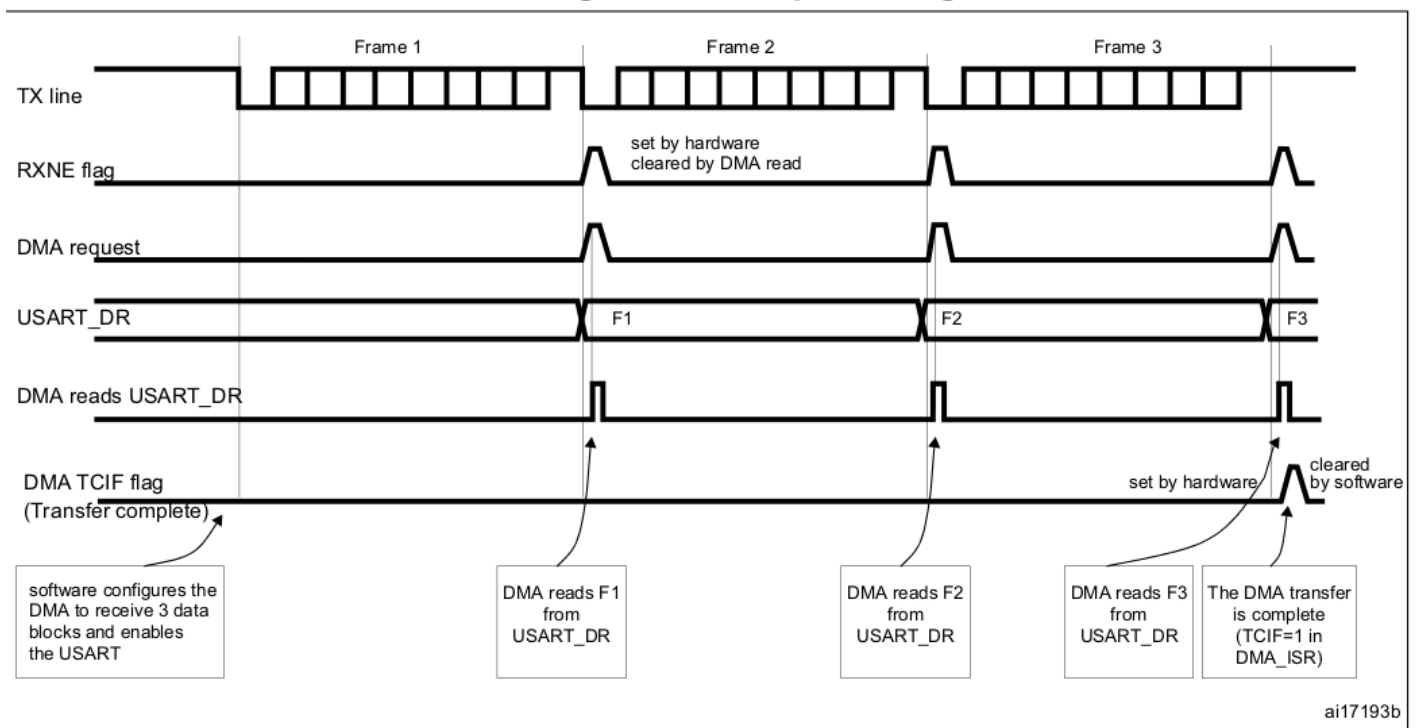


(DMA USART 傳送)

1. 剛開始DMA先把第一個DATA寫入USART_DR.
2. 當TXE(transmission enable), 第一個data(FRAME1)就從TX Line上傳出去;TXE的同時,DMA request觸發, 所以DMA把第二個data也寫進USART_DR.
3. 當FRAME1傳完以後, 再一次TXE, FRAME2 開始從TX line傳出去,同時DMA request再次觸發, FRAME3也開始寫入.
4. 因為DMA一開始要設定這次會傳幾個byte,當回寫入完最後一個byte到USART_DR以後,DMA transfer complete會觸發,之後USART就會等待TC觸發,以確保最後一個byte傳送完畢.
5. TC 會由硬體在最後一個frame傳送完會設為1.

USART 接收端經過 DMA 存放DATA

Figure 316. Reception using DMA



與transmission類似。

1. USART_CR3 的 DMAR 要enable.

- 當收到一個frame以後, 除了RXNE 以外, DMA request也會觸發, 表示DMA可以從USART_DR把DATA傳送到設定好的memory地址.
- 當預先設定好的接收data數量達到以後, DMA TCIF會觸發interrupt, DMAR會在這個interrupt routine裡清掉, 不就會產生DMA request.

SmartCard Mode

SmartCard

SmartCard Mode 是為了支援asynchronous protocol Smartcards(ISO 7816-3)[5]. Smartcard 是一種單線半雙工的通輸協定

register 設定

USART_CN3 : SCEN = 1; HDSEL, IREN : keep clean USART_CN2 : LINEN

CLKEN = 1 (有可能會用到 Clock)

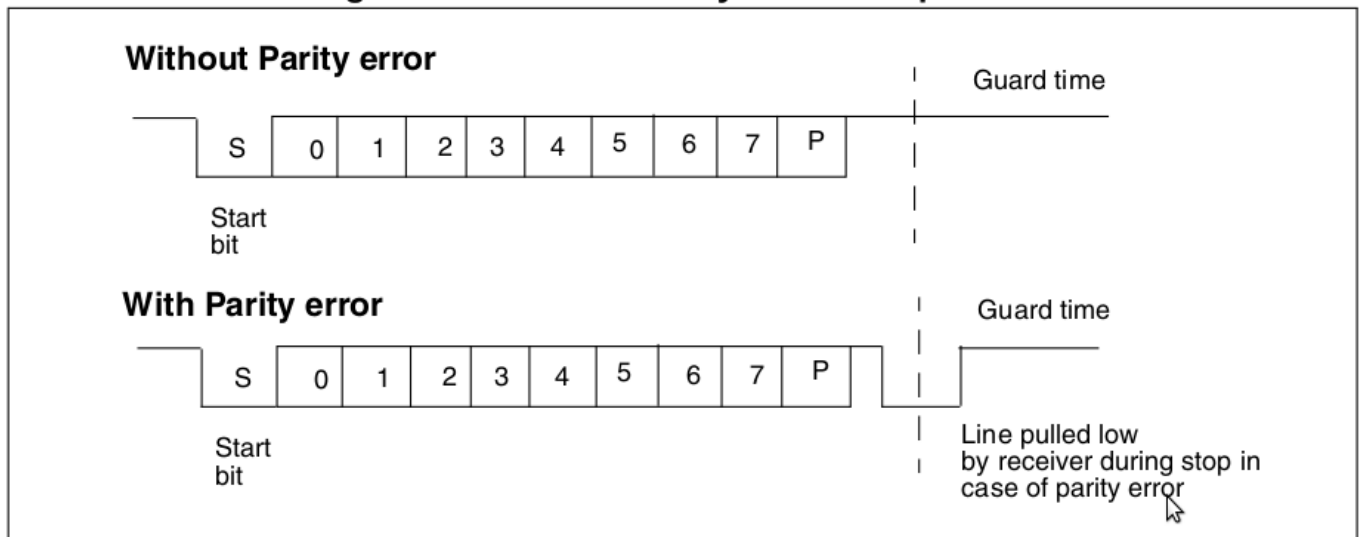
Smartcard mode Stop bit

預設為 transmitter / receiver 兩端都是 1.5 bit, 但是 receiver 可以使用 0.5 bit 的 stop bit.

因為 smartcard mode 是利用 TX 單線傳接, 所以如果設定 0.5 bit stop bit 的話, 在傳送, 接收轉換是需要修改 control register.

parity error in smartcard mode

Figure 311. ISO 7816-3 asynchronous protocol



當發現 parity error, receiver 會把 TX 立刻設為 0 維持一個 baud clock (這個動作稱為 "NACK"). 這樣會做成 framing error 讓 transmitter 端知道, 然後重新傳一次.

IrDA SIR ENDEC

Ref: [RM0090 Reference Manual P.979](#)

IrDA

IrDA 是一種標準化紅外線數據傳送方式; STM32F4 支援利用 USART 輸出訊號配合轉換電路, 轉換成符合 IrDA 標準的訊號, 實現紅外線傳輸.

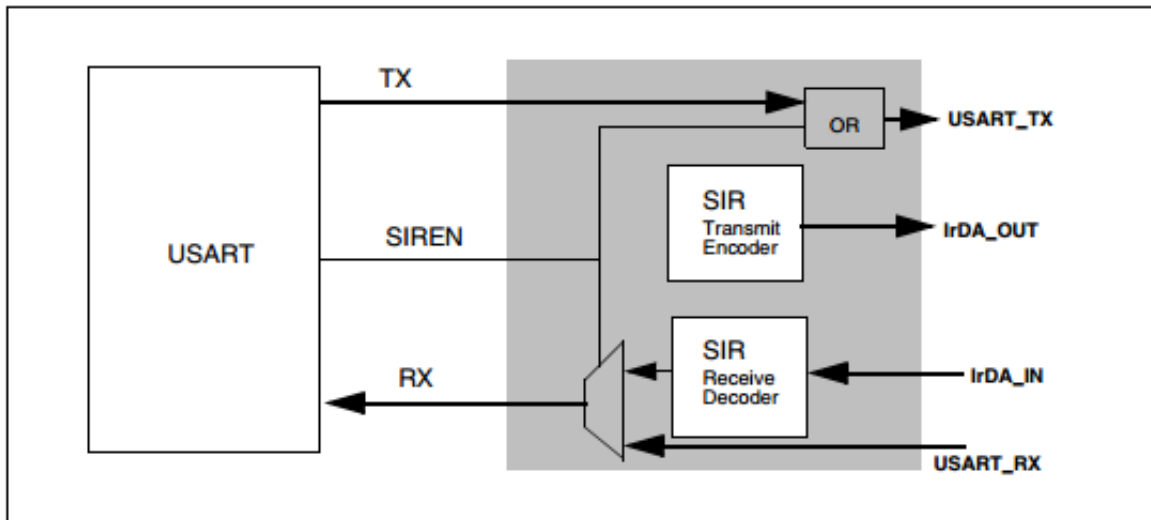
低速紅外線其傳輸數率上限為 115.2 Kbps, 相較於低速紅外線, 高速紅外線的傳輸速率達到 4 Mbps, 更適合傳送較大的檔案, 圖片等; 而由於紅外線無法穿越牆壁, 所以也適合在一個安全保密的封閉環境裡作網路傳輸.

register 設定

USART_CR3 : IREN = 1; SCEN, HDSEL : keep clean.

USART_CR2 : LINEN, STOP, CLKEN : keep clean.

IrDA SIR 實體層

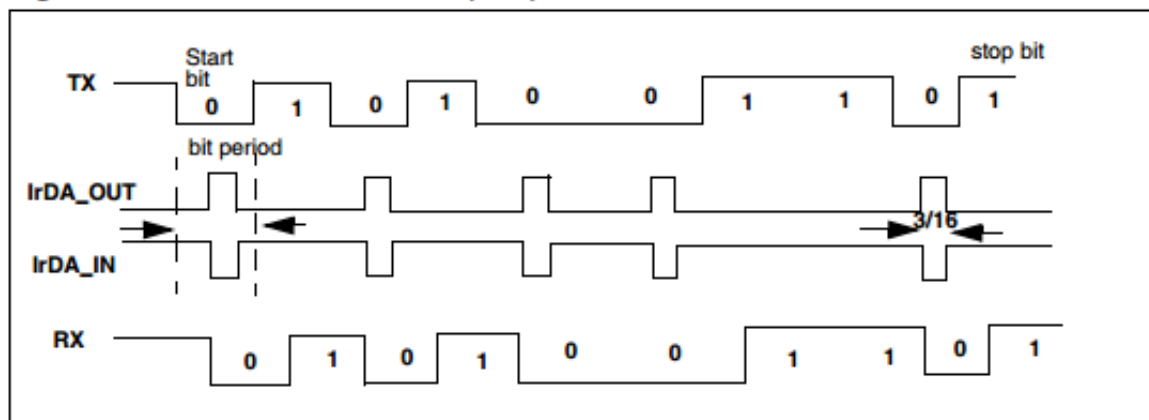
Figure 262. IrDA SIR ENDEC- block diagram

SIR Transmit encoder 會把 USART 的 Non Return to Zero 轉換成 IrDA 輸出訊號。

IrDA 輸出是一種 Return-To-Zero, Inverted 形式的訊號, 用一個紅外線 pulse 表示一個 logic 0.

每一個紅外線 pulse 相當於 3/16 的 bit period, 而 IrDA mode 下的 USART 的 bit rate 最高為 115.2 Kbps.

(下圖 IrDA_OUT).

Figure 263. IrDA data modulation (3/16) -Normal mode

USART SYNCHRONOUS MODE

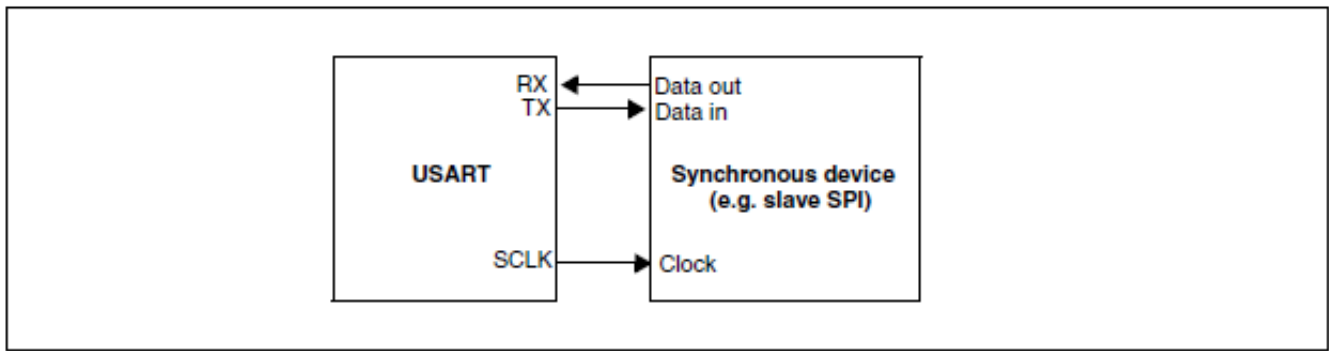
Ref: [RM0090 Reference Manual P.974](#)

register 設定

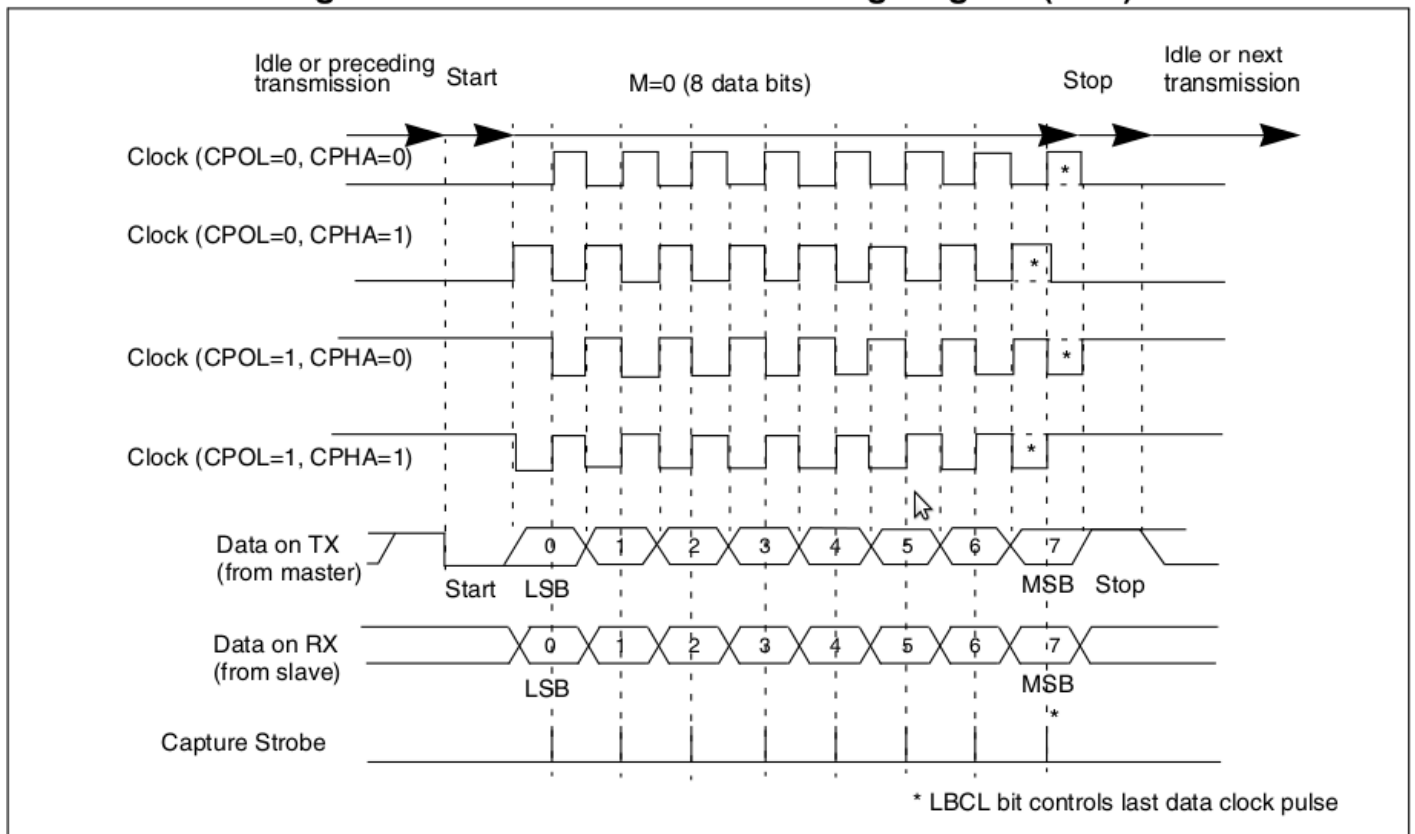
USART_CR2 : CLKEN = 1; LINEN : keep cleared;

USART_CR3 : SCEN, HDSEL, IREN : keep cleared;

特性

Figure 256. USART example of synchronous transmission

Synchronous 指的是master 跟 slave 用同一個 Clock, 這個clock(SCLK)由 master 端送出, 同時需要TE = 1, clock 才會產生; 這個時候只有master 方會接收或傳送data. (UART 4,5 不支援 Synchronous mode)

Figure 308. USART data clock timing diagram (M=0)

在傳送 start bit, stop bit 時不會有 clock pulse 從 SCLK 送出, clock 處於 logic 0.

Last bit clock pulse (LBCL), 確定第8(9, 如果 M=1) bit 的 clock pulse 會不會輸出.

CPOL = 0: high 為 logic 1 ; CPOL = 1: high 為 logic 0.

CPHA = 0: clock pulse 發生在 bit period 的後半段中; CPHA = 1: clock pulse 發生在 bit period 的前半段; 差別在於 bit period 中間的 edge 是正/負緣觸發.

master mode: 不能夠用 slave (不是產生 SCLK 的一方, 非 usart 周邊) 輸入進來的 input clock 做接收或送去 data.

因為同步, 所以不需要 oversampling ($W_s \gg 2W_m$)

USART 與 SPI

SPI：它是主從式的架構，通常一個主設備和多個從設備

由四條信號線組成：SCLK、MISO、MOSI、CS

MOSI：master output, slave input，主設備輸出

MISO：master input, slave output，主設備輸入

SCLK：serial clock，clock信號，由主設備產生

CS：chip select (optional)

USART在全雙工的模式(特別是同步模式)下，也有類似的訊號

TX：傳送訊號給周邊

RX：週邊設備傳送給主設備的訊號

SCLK：由主設備產生的clock訊號

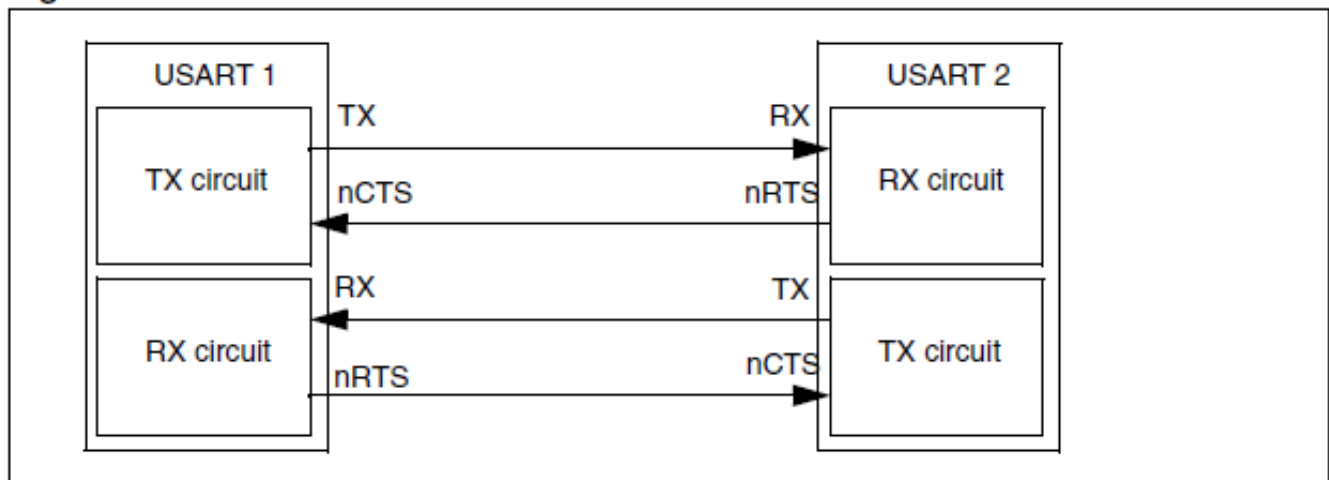
以上是兩者有類似的地方

HARDWARE FLOW CONTROL

Ref: [RM0090 Reference Manual P.983](#)

利用 nCTS 跟 nRTS 控制 TX, RX 要不要再傳送或接收 data

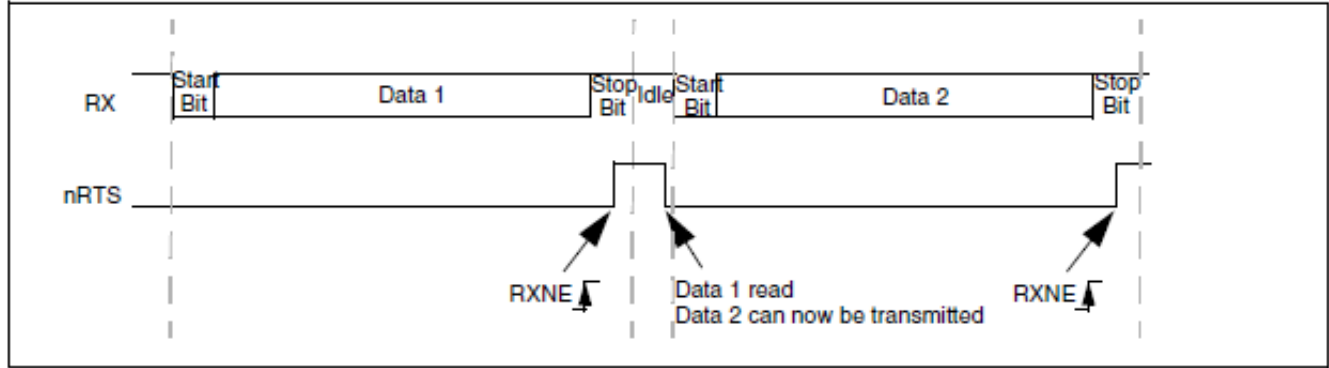
Figure 266. Hardware flow control between 2 USARTs



RTS Flow Control

RTSE(RTS flow control enable), CTSE 為1才開啟hw flow control;

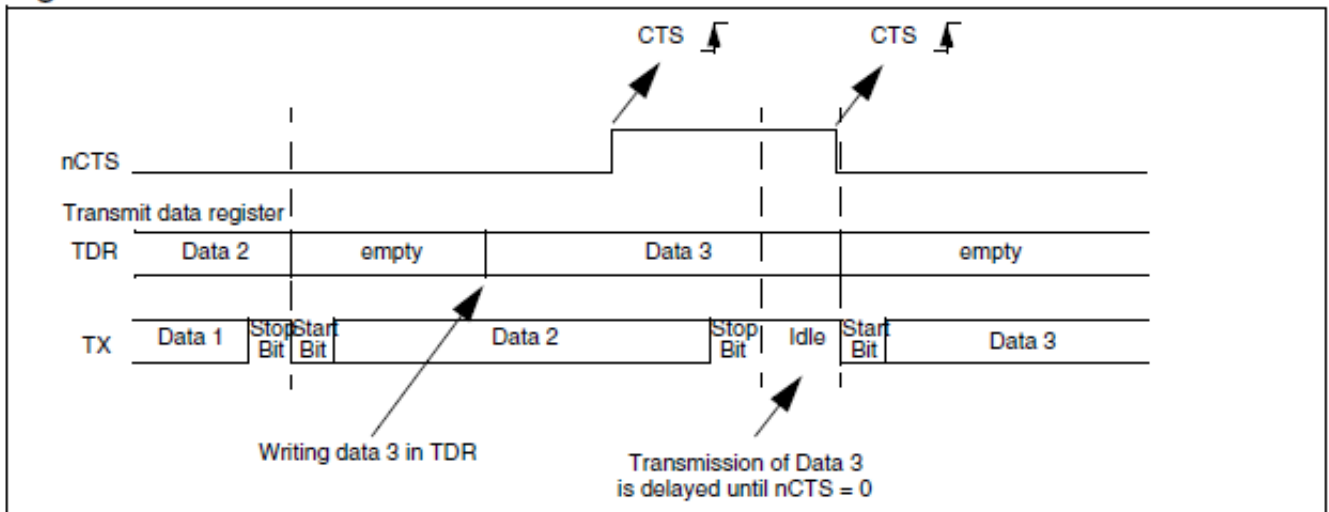
Figure 318 在receive端register 滿了以後, nRTS轉為 1, 表示接收完這個data以後就暫時不接收Data;

Figure 267. RTS flow control

CTS Flow Control

跟上面是對應的, 當nRTS為1, transmitter 端的nCTS就會是1, 表示他知道reciever 目前無法接收data, TX就會在傳送完當前的data以後 idle.

nCTS為1 的時候, 會由硬體去設定CTSIE bit (USART_CR3).

Figure 268. CTS flow control

PARITY CONTROL

Parity control是用來確保傳輸資料的正確性。其原理是在傳輸端產生一個parity bit，然後在接收端可以重新計算parity bit以確保在傳輸過程沒有發生錯誤。在STM32，它可以透過設定USART_CR1 register的PCE bit來打開。STM32的frame長度是由M bit所決定，所以USART的frame有以下這些可能格式：

Table 118. Frame formats

M bit	PCE bit	USART frame ⁽¹⁾
0	0	SB 8 bit data STB
0	1	SB 7-bit data PB STB
1	0	SB 9-bit data STB
1	1	SB 8-bit data PB STB

Even/Odd parity

parity依算方式的不同分成兩種方式，even parity和odd parity

Even parity

如果一個frame內1的數量是偶數，則在 even parity的情況下會把parity bit設為0。

E.g.: 假設 data=00110101; 因為有 4 bits被設為1，而且我們選擇的是 even parity(PS bit in USART_CR1 = 0)，所以 parity bit被設為0。

Odd parity

如果一個frame內1的數量是奇數，則在 odd parity的情況下會把parity bit設為0。

E.g.: 假設 data=00110101; 因為有 4 bits被設為1，而且我們選擇的是 odd parity(PS bit in USART_CR1 = 1)，所以 parity bit被設為1。

接收後會做 parity checking

如果 parity check 失敗了，USART_SR register的PE flag會被設立，然後如果USART_CR1 register的PEIE bit也有被設立的話，還會產生中斷。PE flag最後在軟體執行(a read from the status register followed by a read or write access to the USART_DR data register)時被清除。

傳送前會做 parity generation

如果USART_CR1的PCE bit被設立，那麼MSB會被改成parity bit (PS=0 是even parity, PS=1 是odd parity)

小知識：

MSB: Most Significant Bit，代表位數最大的那個bit

LSB: Least Significant Bit，代表位數最小的那個bit

USART Register 總表

- Status register(USART_SR)
- Data register(USART_DR)
- Baud rate register(USART_BRR)
- Control Register 1(USART_CR1)
- Control Register 2(USART_CR2)
- Control Register 3(USART_CR3)
- Guard time and prescaler register(USART_GTPR)

Table 148. USART register map and reset values																																	
Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	USART_SR	Reserved																						CTS	LBD	TXE	TC	RXNE	IDLE	ORE	NF	FE	PE
	Reset value																							0	0	1	1	0	0	0	0	0	0
0x04	USART_DR	Reserved																						DR[8:0]									
	Reset value																							0	0	0	0	0	0	0	0	0	0
0x08	USART_BRR	Reserved														DIV_Mantissa[15:4]										DIV_Fraction [3:0]							
	Reset value															0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
0x0C	USART_CR1	Reserved														OVER8	Reserved	UE	M	WAKE	PCE	PS	PEIE	TXEIE	TCIE	RXNEIE	IDLEIE	TE	RE	RWU	SBK		
	Reset value															0	Reserved	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x10	USART_CR2	Reserved														LINEN	STOP [1:0]	CLKEN	CPOL	CPHA	LBCL	Reserved	LBDIE	LBDL	Reserved	ADD[3:0]							
	Reset value															0	0	0	0	0	0	0	Reserved	0	Reserved	0							
0x14	USART_CR3	Reserved																				ONEBIT	CTSIE	CTSE	RTSE	DMAT	DMAR	SCEN	NACK	HDSEL	IRLP	IREN	EIE
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	0
0x18	USART_GTPR	Reserved														GT[7:0]							PSC[7:0]										
	Reset value															0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		

CODE SECTION

發送器測試

Download sample code :

```
git clone https://github.com/wujiheng/stm32f407.git
cd stm32f407/USART
```

To compile code

```
make
make flash <-- 記得把USB連上去
```

這裡採用minicom 超級終端機來接收USART字串

```
ls /dev <-- 找device，見圖，這裡找到/dev/ttyUSB0
```

```

File Edit View Search Terminal Help
wujiheng@edalab-SH560 ~ $ ls /dev
agpgart      cpu_dma_latency loop0      network_throughput ram15  sda2      stlinkv2_1 tty20  tty34  tty48  tty61  ttyS16  ttyS3  vcs2  vcsa63
alarm        disk            loop1      null           ram2   sda3      tty        tty21  tty35  tty49  tty62  ttyS17  ttyS30  vcs3  vcsa7
ashmem       dri             loop2      oldmem          ram3   sda4      tty0       tty22  tty36  tty5   tty63  ttyS18  ttyS31  vcs4  vcsa8
autofs       dvd            loop3      port            ram4   sda5      tty1       tty23  tty37  tty50  tty7   ttyS19  ttyS4  vcs5  vga_arbiter
binder       dvdrw          loop4      ppp             ram5   sda6      tty10      tty24  tty38  tty51  tty8   ttyS2  ttyS5  vcs6  vhost-net
block        ecryptfs        loop5      psaux           ram6   serial    tty11      tty25  tty39  tty52  tty9   ttyS20  ttyS6  vcs63  video0
bsg          fb0            loop6      ptmx            ram7   sg0       tty12      tty26  tty4   tty53  ttyprintk ttyS21  ttyS7  vcs7  zero
btrfs-control fd              loop7      pts             ram8   sg1       tty13      tty27  tty40  tty54  ttyS0   ttyS22  ttyS8  vcs8
bus          full           loop-control ram0            ram9   shm       tty14      tty28  tty41  tty55  ttyS1   ttyS23  ttyS9  vcsa
cdrom        fuse           mapper      ram1            random snapshot tty15      tty29  tty42  tty56  ttyS10  ttyS24  ttyUSB0 vcsa1
cdrw         hpet           mcelog      ram10           rkill  snd        tty16      tty3   tty43  tty57  ttyS11  ttyS25  uinput  vcsa2
char         input          mei         ram11           rtc    sr0        tty17      tty30  tty44  tty58  ttyS12  ttyS26  urandom vcsa3
console      kmsg           mem         ram12           rtc0   stderr     tty18      tty31  tty45  tty59  ttyS13  ttyS27  v4l     vcsa4
core         kvm            net         ram13           sda    stdin      tty19      tty32  tty46  tty6   ttyS14  ttyS28  vcs    vcsa5
cpu          log            network_latency ram14          sda1   stdout     tty2       tty33  tty47  tty60  ttyS15  ttyS29  vcs1   vcsa6

```

```
sudo apt-get install minicom
sudo minicom -s <-- 不一定要用root，不過使用者必須要device node 讀寫的權限，-s表
進入setup
```

```

+-----[configuration]-----+
| Filenames and paths          |
| File transfer protocols      |
| Serial port setup            |
| Modem and dialing            |
| Screen and keyboard          |
| Save setup as dfl             |
| Save setup as..              |
| Exit                         |
| Exit from Minicom            |
+-----+

```

```

File Edit View Search Terminal Help

+-----+
| A -   Serial Device       | /dev/ttyUSB0 |
| B - Lockfile Location    | : /var/lock |
| C -   Callin Program     | :           |
| D -   Callout Program    | :           |
| E -   Bps/Par/Bits       | : 9600 8N1  |
| F - Hardware Flow Control | : No        |
| G - Software Flow Control | : No        |
|                           |             |
| Change which setting?    |             |
+-----+
|                           |             | | |
| | Screen and keyboard   | |           |
| | Save setup as dfl    | |           |
| | Save setup as..     | |           |
| | Exit                 | |           |
| | Exit from Minicom   | |           |
| |                     | |           |
+-----+

```

選擇第三個 "Serial port setup"，設定接收的模式及port
先按 'A' 選擇 device，並輸入 /dev/ttyUSB0 (由剛剛的 ls /dev) 決定

```

+-----[Comm Parameters]-----+
|                               |
| Current: 9600 8N1           |
| Speed      Parity    Data   |
| A: <next>   L: None   S: 5   |
| B: <prev>   M: Even   T: 6   |
| C: 9600     N: Odd    U: 7   |
| D: 50400    O: Mark   V: 8   |
| E: 115200   P: Space                |
|                               |
| Stopbits   |
| W: 1        Q: 8-N-1                |
| X: 2        R: 7-E-1                |
|                               |
| Choice, or <Enter> to exit? |
+-----+

```

再來按 'E' 設定接收的參數，選擇

'C' --> Baud rate 9600

'L' --> None parity check

'V' --> 資料長度 8 bits

'W' --> 停止位元數 1 bit

輸入完後按Enter離開

```
+-----[configuration]-----+
| Filenames and paths          |
| File transfer protocols      |
| Serial port setup           |
| Modem and dialing            |
| Screen and keyboard          |
| Save setup as dfl             |
| Save setup as..              |
| Exit                         |
| Exit from Minicom            |
+-----+
```

回到原本的畫面，選擇 'Save setup as default'，然後選擇 'Exit from Minicom' 離開

```
File Edit View Search Terminal Help

Welcome to minicom 2.6.2

OPTIONS: I18n
Compiled on Feb  8 2013, 06:27:51.
Port /dev/ttyUSB0, 14:52:04

Press CTRL-A Z for help on special keys
```

```
File Edit View Search Terminal Help

Welcome to minicom 2.6.2

OPTIONS: I18n
Compiled on Feb  8 2013, 06:27:51.
Port /dev/ttyUSB0, 14:52:04

Press CTRL-A Z for help on special keys
```



按照圖上的接法，白色線接PA2、綠色線接PA3

** 白色線為USB的RX所以要接上板子的TX(PA2) **

** 綠色線為USB的TX所以要接上板子的RX(PA3) **

此時畫面中的Minicom會顯示出結果，不斷的印出“Init complete! Hello World!”


```
File Edit View Search Terminal Help  
Init complete! Hello World!  
Init complete! Hello World!  
Init complete! Hello World!  
Init complete! Hello World!  
Init complete! Hello World!  
Init complete! Hello World!  
Init complete! Hello World!  
Init complete! Hello World!  
Init complete! Hello World!  
Init complete! Hello World!  
Init complete! Hello World!  
Init complete! Hello World!  
Init complete! Hello World!  
Init complete! Hello World!  
Init complete! Hello World!  
Init complete! Hello World!  
Init complete! Hello World!  
Init complete! Hello World!  
Init complete! Hello World!  
Init complete! Hello World!  
Init complete! Hello World!  
Init complete! Hello World!  
Init complete! Hello World!  
Init complete! Hello World!  
Init complete! Hello World!  
Init complete! Hello World!  
Init complete! Hello World!  
Init complete! Hello World!  
Init complete! Hello World!  
Init complete! Hello World!  
Init complete! Hello World!  
Init complete! Hello World!  
Init complete! Hello World!  
Init complete! Hello World!  
Init complete! Hello World!  
Init complete! Hello World!  
Init complete! Hello World!  
Init complete! Hello World!
```

CTRL-A Z for help | 9600 8N1 | NOR | Minicom 2.6.2

按Ctrl+a，再按x可離開Minicom

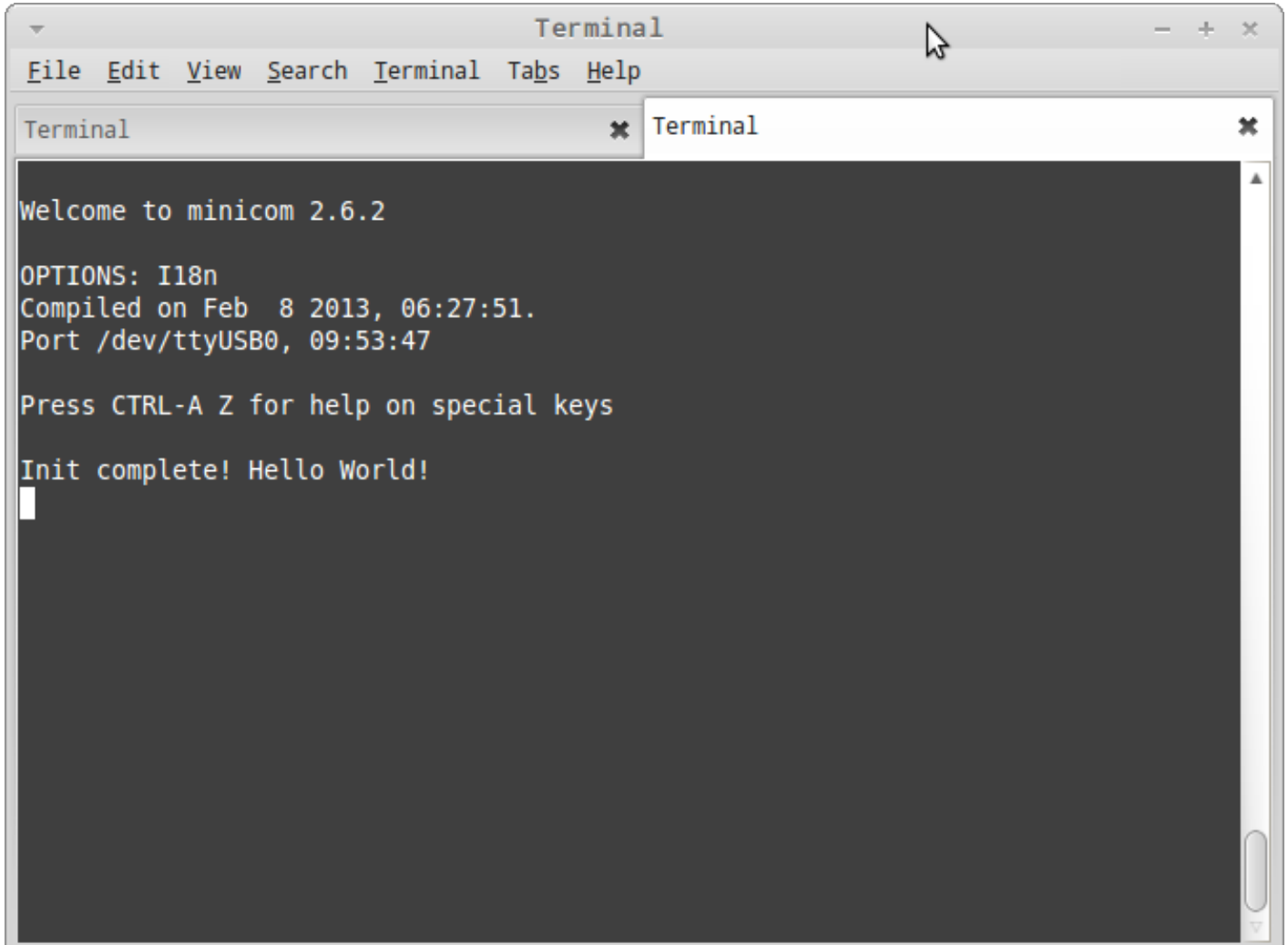
實驗波形圖，採用USBee AX作訊號分析



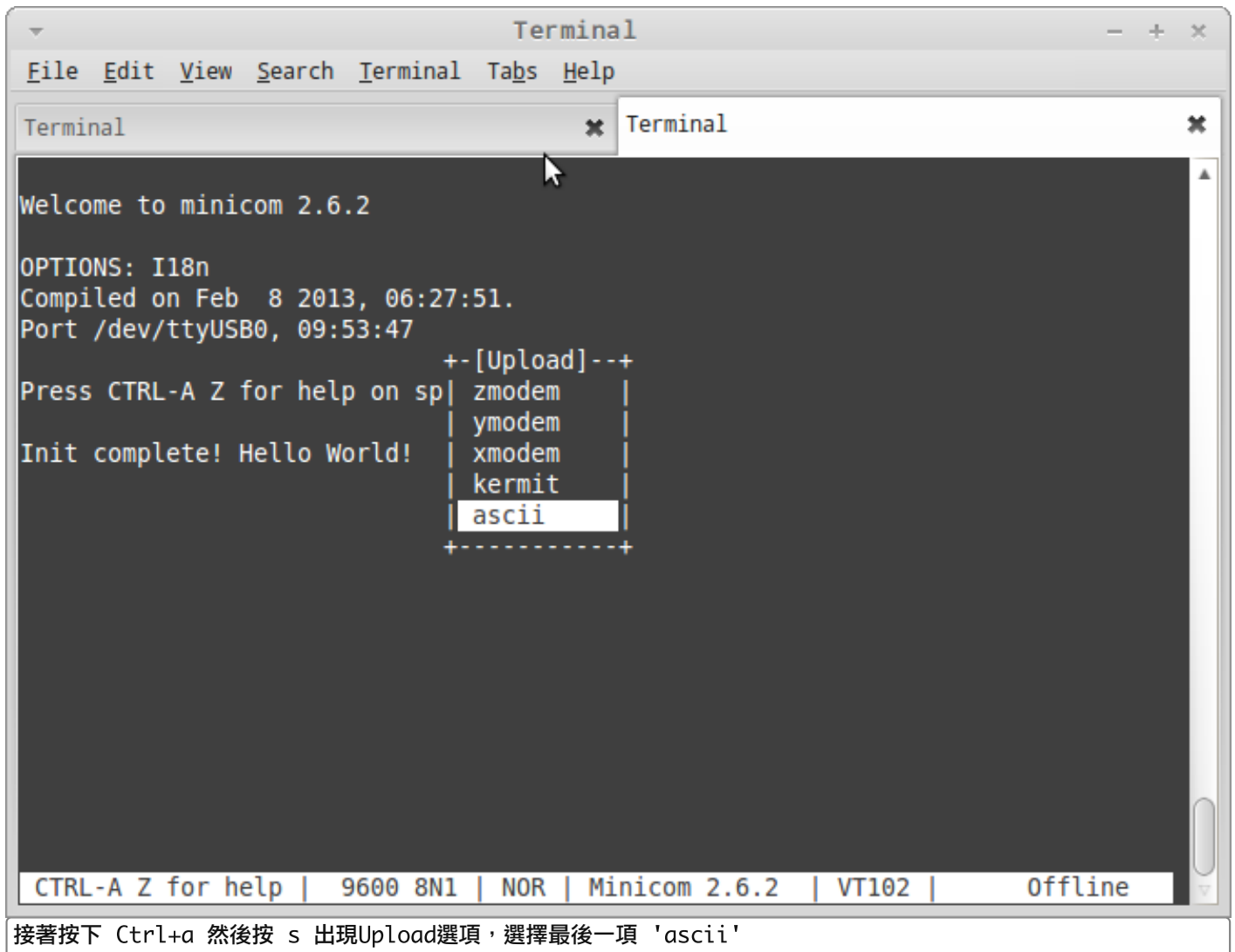
接收器測試

To compile code

```
// 切換到branch USART_Receive  
git checkout USART_Receive  
  
// 編譯並上傳程式  
make; make flash
```



打開Minicom只會看到一行字串



The screenshot shows a terminal window titled "Terminal" with a menu bar containing "File", "Edit", "View", "Search", "Terminal", "Tabs", and "Help". The terminal output is as follows:

```
Welcome to minicom 2.6.2

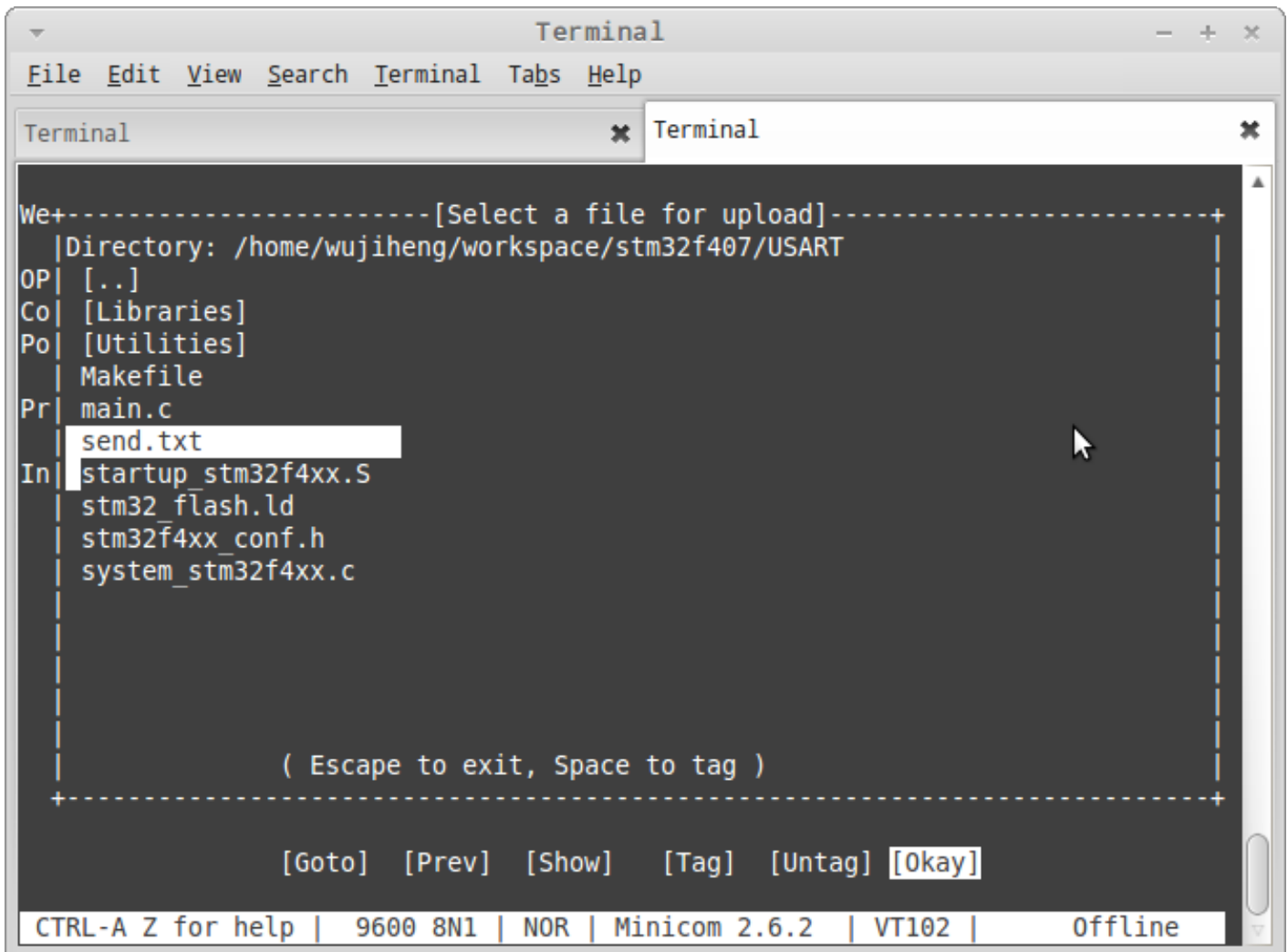
OPTIONS: I18n
Compiled on Feb  8 2013, 06:27:51.
Port /dev/ttyUSB0, 09:53:47

Press CTRL-A Z for help on sp|
Init complete! Hello World!

+-[Upload]--+
| zmodem    |
| ymodem    |
| xmodem    |
| kermit    |
| ascii    |
+-----+

CTRL-A Z for help | 9600 8N1 | NOR | Minicom 2.6.2 | VT102 | Offline
```

Below the terminal window, a text instruction reads: 接著按下 Ctrl+a 然後按 s 出現Upload選項，選擇最後一項 'ascii'.

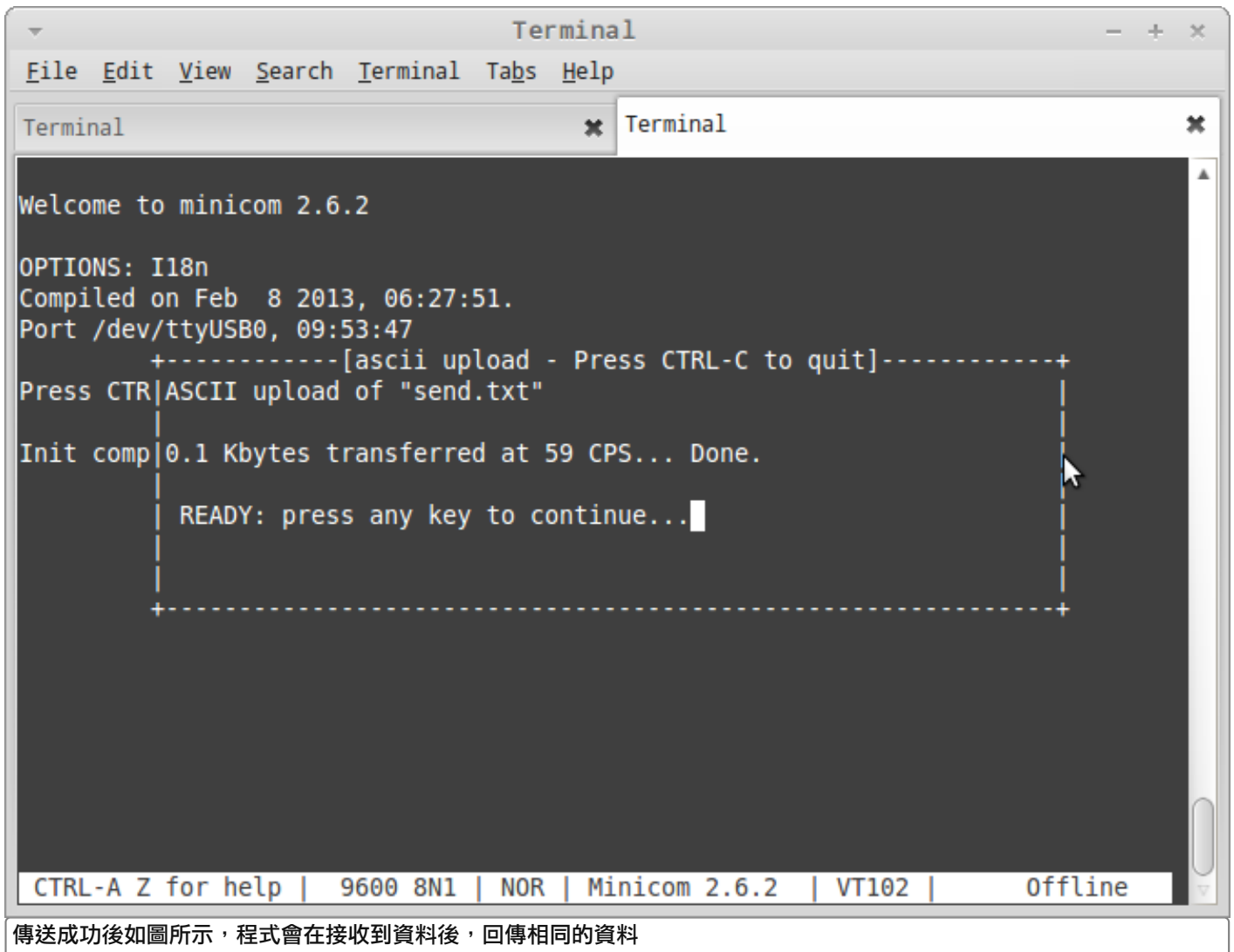


```
Terminal
File Edit View Search Terminal Tabs Help

Terminal x Terminal x

We+-----[Select a file for upload]-----+
|Directory: /home/wujiheng/workspace/stm32f407/USART|
OP| [..]|
Co| [Libraries]|
Po| [Utilities]|
| Makefile|
Pr| main.c|
| send.txt|
In| startup_stm32f4xx.S|
| stm32_flash.ld|
| stm32f4xx_conf.h|
| system_stm32f4xx.c|
|
| ( Escape to exit, Space to tag )|
+-----+
|
| [Goto] [Prev] [Show] [Tag] [Untag] [Okay]|
|
| CTRL-A Z for help | 9600 8N1 | NOR | Minicom 2.6.2 | VT102 | Offline|
```

找到要發送過去的檔案，按空白鍵選取，按Enter開始傳送



```
Terminal
File Edit View Search Terminal Tabs Help

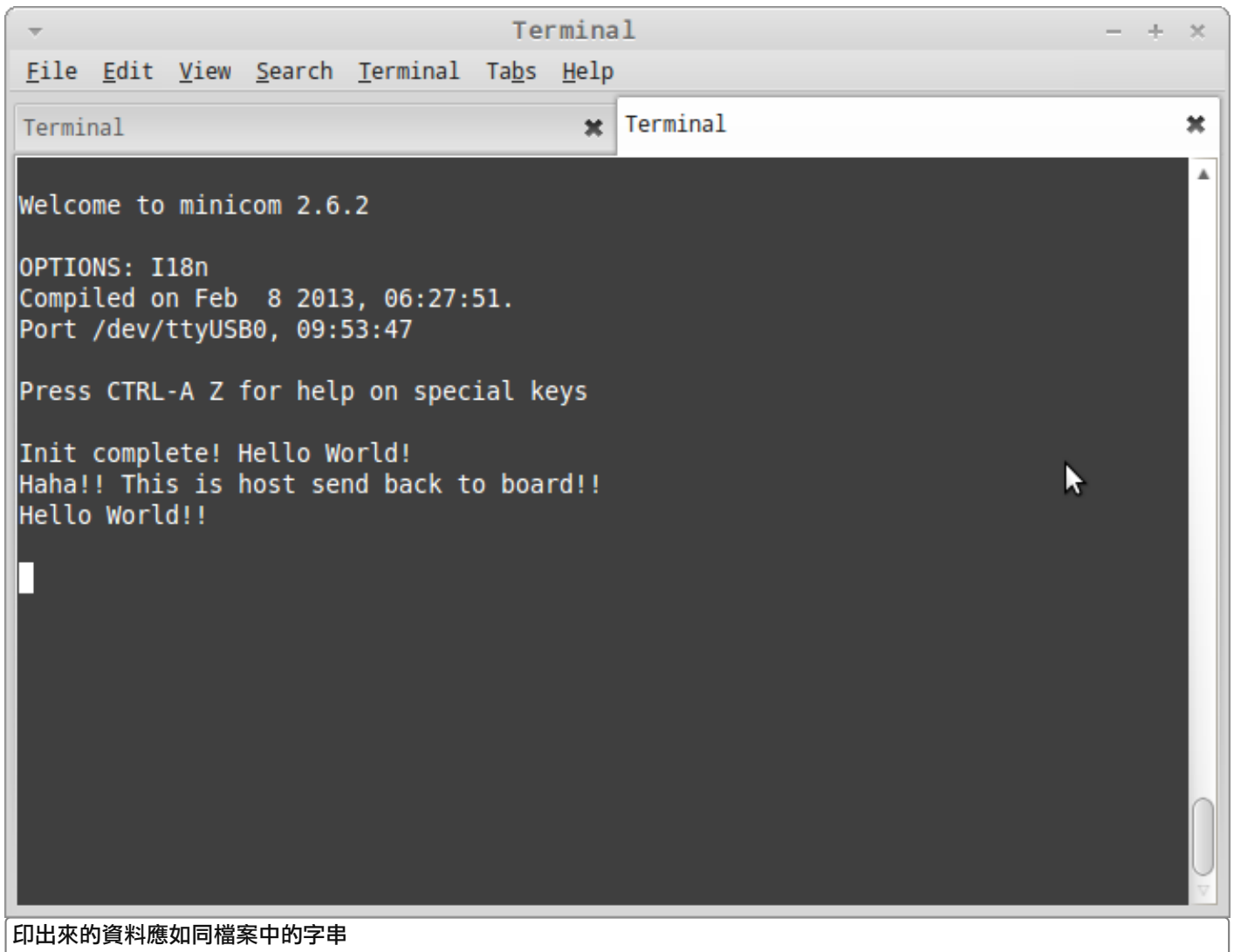
Terminal x Terminal x

Welcome to minicom 2.6.2

OPTIONS: I18n
Compiled on Feb  8 2013, 06:27:51.
Port /dev/ttyUSB0, 09:53:47
+-----[ascii upload - Press CTRL-C to quit]-----+
Press CTRL|ASCII upload of "send.txt"
Init comp|0.1 Kbytes transferred at 59 CPS... Done.
| READY: press any key to continue...|
+-----+

CTRL-A Z for help | 9600 8N1 | NOR | Minicom 2.6.2 | VT102 | Offline
```

傳送成功後如圖所示，程式會在接收到資料後，回傳相同的資料



The screenshot shows a terminal window titled "Terminal" with a menu bar (File, Edit, View, Search, Terminal, Tabs, Help). The terminal output is as follows:

```
Welcome to minicom 2.6.2

OPTIONS: I18n
Compiled on Feb  8 2013, 06:27:51.
Port /dev/ttyUSB0, 09:53:47

Press CTRL-A Z for help on special keys

Init complete! Hello World!
Haha!! This is host send back to board!!
Hello World!!
```

Below the terminal window, a caption reads: 印出來的資料應如同檔案中的字串

REFERENCE

- [1]Universal asynchronous receiver/transmitter wikipedia.
- [2]STM32F407xx Reference Manual P.946 ~ P.997
- [3]IrDA and RS-232
- [4]Using The DMA controller on STM32
- [5]ISO 7816-3

Q&A

1.baud rate為何不是2的倍數？幾個baud rate數字的關聯？為何設定會有小數點？

baud rate 是單位時間內傳輸資訊的個數，單位是bits/sec

最早的發明是用來測量電報傳輸速率，現在用來作為網路兩節點的傳輸速率

常用的有300、1200、2400、9600、115200、19200等bits/sec

這些規格是來自歷史因素，最早的baud rate用在電傳，是75baud

後來每次擴充都是兩倍，75->150->300->600->1200....

而我們為了要得到這個值，必須從系統的clock做分頻，因此要設定USARTDIV

這也是USARTDIV會是小數的原因

2. 為什麼USART可以選擇8 bit/9 bit

由於歷史因素，所以usart可以選擇8,9bit的傳送

最一開始的ASCII code是使用7 bit來做編碼，而第8個位元常會被拿來做各種應用

像是加上parity bit來驗證資料的正確性

但是當以8為單位的電腦系統興起後，開始用8bit來做傳輸

所以一開始7 bit ASCII code也被擴展為8 bit

3. 為何stop bit有0.5 bit、1.5 bit的設計？

stop bit其實不算是bit，他是傳輸結束後的一段時間(period)，區隔每個傳輸的資料

它的功用是在非同步傳輸的時候可以告訴receiver資料傳輸已經結束

stop bit有0.5, 1, 1.5, 2bits，共四種

一些比較老的teletype machine可能需要不只一個stop bit

如果stop bit越長，可以增加多一點點的處理時間

另一個原因是長一點的stop bit可以提供長一點的同步時間

若是在環境比較不好的情況下(例如，長距離傳輸)，較長的同步時間會可以有效減少錯誤的發生

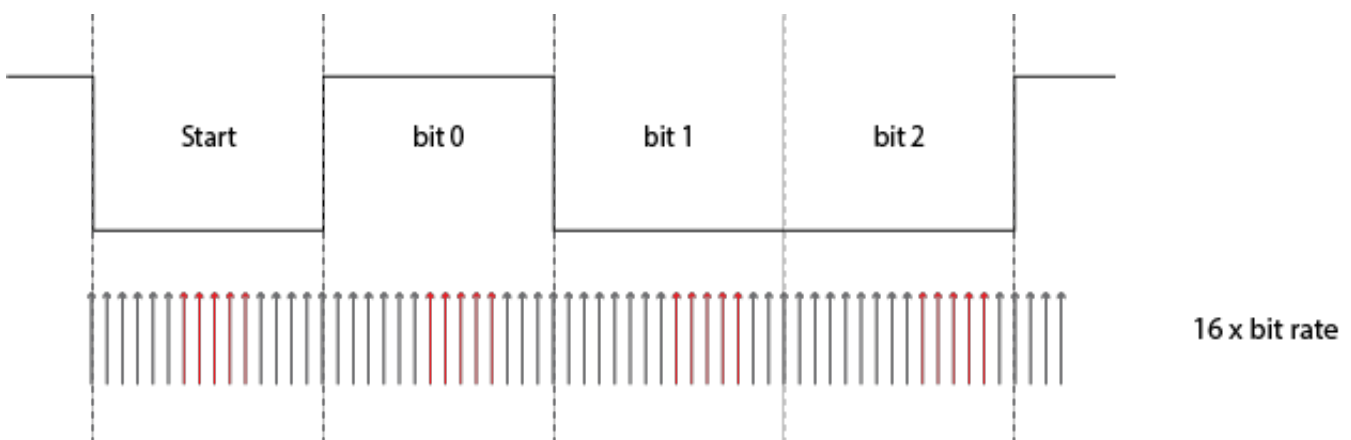
不過缺點是會導致throughput的降低

4. 為何取樣是看8,9,10這幾個bit？

因為8,9,10是在整個start bit的正中間

由於接收端和傳送端的取樣頻率可能會有些微誤差

如果我們在中間取樣，可以容許一定程度的取樣頻率誤差



圖片來源