

Dispositivo Intel 8251A / USART

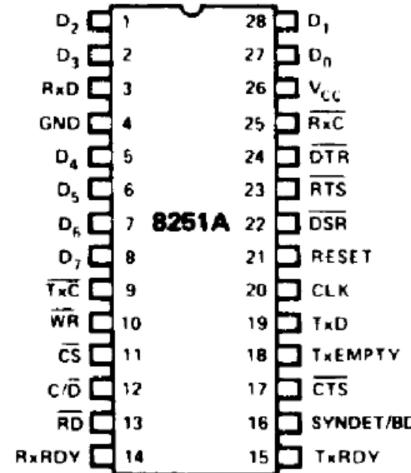
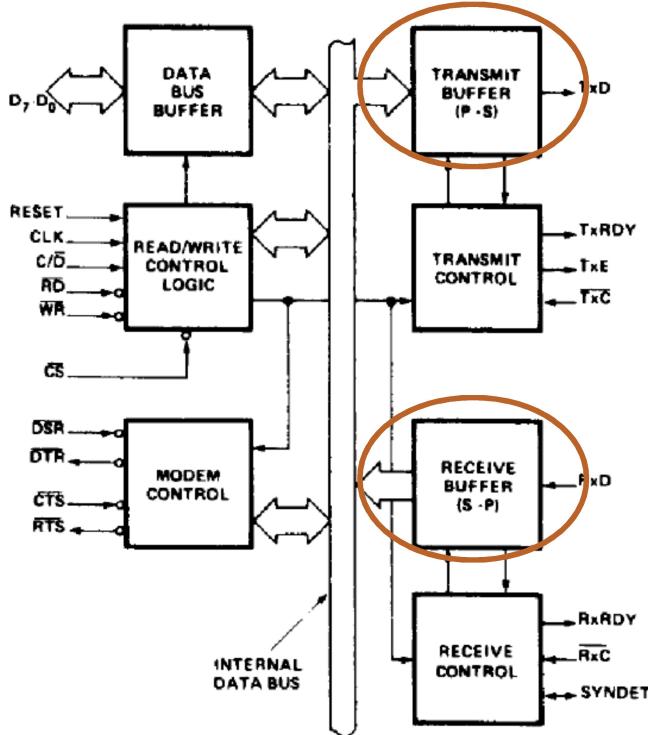
Configurazione e sincronizzazione

Intel 8251A

Il device Intel 8251A è un dispositivo programmabile che nasce per la comunicazione fra microprocessori della famiglia Intel e i dispositivi seriali. Esso realizza lo standard industriale USART e fornisce i segnali di controllo previsti dallo standard di comunicazione seriale RS232.

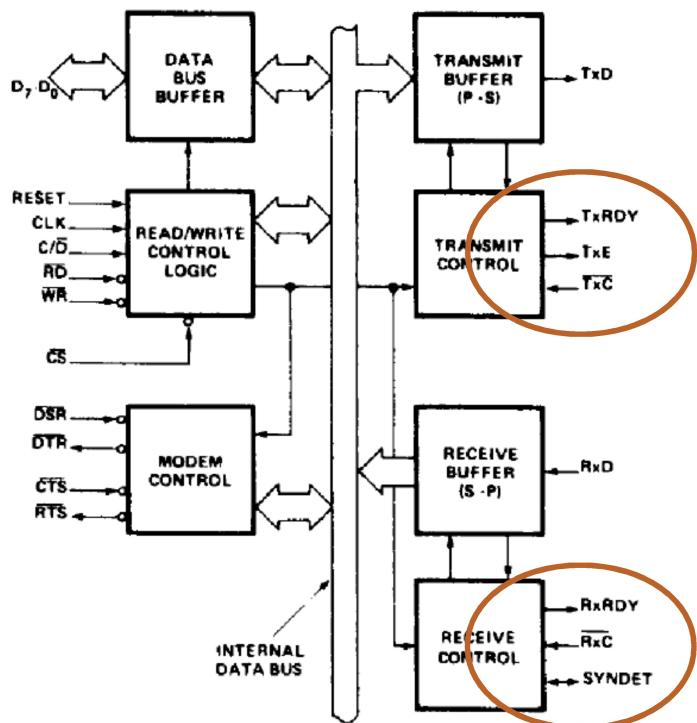
Il device può essere usato per trasmettere e ricevere contemporaneamente:

- In trasmissione, il dispositivo riceve in un buffer i dati dalla CPU (in formato parallelo) e li converte in uno stream di dati seriali attraverso uno shift register;
- In ricezione, i dati seriali vengono ricevuti attraverso uno shift register e inviati (in parallelo) alla CPU tramite un buffer.



Intel 8251A - sincronizzazione

Attraverso opportuni segnali di controllo, il dispositivo segnala alla CPU se può accettare un nuovo carattere da trasmettere ($\text{TxRDY}=1$), oppure se ha ricevuto un nuovo carattere che può comunicare alla CPU ($\text{RxRDY}=1$)



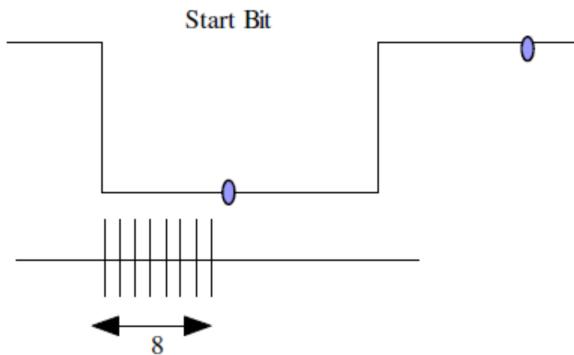
I segnali TxRDY e RxRDY possono essere usati come segnali di interruzione
Essi vengono mascherati da due bit del registro di controllo, TxEnable e RxEnable

Intel 8251A – comunicazione asincrona

In modalità **asincrona**, la sincronizzazione è gestita a livello del singolo carattere trasmesso, e quindi sono necessarie delle informazioni aggiuntive per mantenere il sincronismo.

Ogni volta che la CPU invia un carattere, il dispositivo crea un frame contenente:

- un bit di start (livello logico basso)
- i bit di dato
- un bit di parità (pari o dispari)
- uno o due stop bit (livello logico alto)



La linea di ricezione è posta in IDLE (=1) e la comunicazione inizia appena si rileva lo start bit

Il clock del RX viene settato ad una velocità multipla (16x o 64x) rispetto a quella del TX in modo da non perdere lo start bit

Successivamente, il RX rifà il campionamento (es. con un shift di 8 impulsi se la velocità di RX è 16 volte quella di TX) posizionandosi al centro del bit e parte il normale campionamento alla frequenza del trasmettitore



Intel 8251A – comunicazione sincrona

La comunicazione sincrona viene usata per aumentare le prestazioni e l'efficienza:

- Il frame inviato contiene uno stream di caratteri e la sincronizzazione avviene solo all'inizio oppure dopo un numero **PREDETERMINATO** di caratteri, inviando una sequenza di caratteri speciali (BYTE SYNC)
- Per la correzione degli errori si applica un codice ridondante sull'intero frame (stesso numero di bit indipendentemente dalla lunghezza del msg).

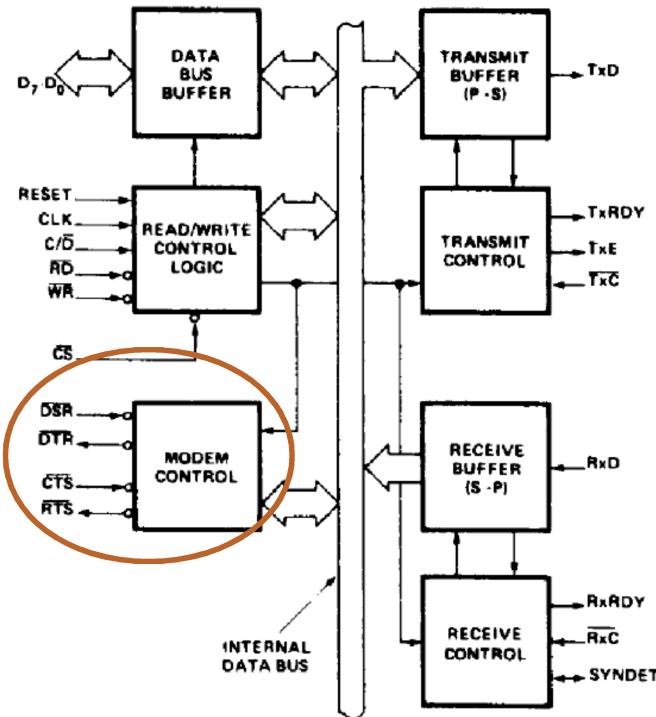


Intel 8251A - modem control

Lo standard RS232 prevede due coppie di segnali hardware per il controllo della comunicazione fra un **DTE** (data terminal equipment) e un **DCE** (data communication equipment, es. *modem*):

- **DTR**(data terminal ready)/**DSR**(data set ready)
- **RTS**(request to send)/**CTS**(clear to send)

Non è definito dallo standard un protocollo per l'utilizzo di tali segnali. L'effettivo utilizzo dipende dall'implementazione specifica dei device considerati.



DTR/DSR sono tipicamente usati per stabilire la connessione fra terminale e modem:

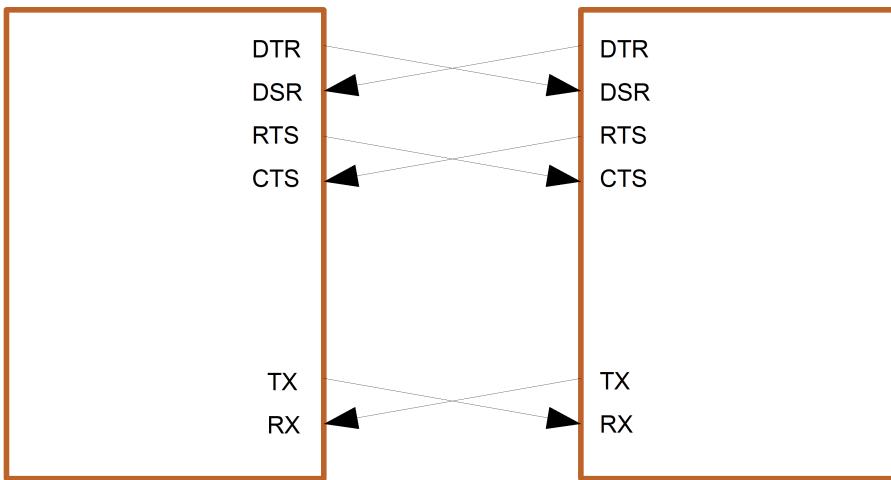
- Il DTE asserisce **DTR** per comunicare al DCE che è ON e pronto a ricevere, iniziare o continuare una comunicazione
- Il DCE asserisce **DSR** per comunicare al DTE che è ON e pronto a ricevere comandi o dati

RTS/CTS sono stati definiti in origine per disciplinare la comunicazione half-duplex:

- Il DTE asserisce **RTS** quando vuole trasmettere dati
- Il DCE asserisce **CTS** se è pronto a ricevere

Intel 8251A - Architettura di riferimento

La configurazione utilizzata negli esercizi proposti in ASIM è di tipo “NULL MODEM”, e prevede che i segnali di handshaking vengano direttamente scambiati fra due terminali senza la presenza di un modem. L’architettura di riferimento è quella mostrata di seguito.



- Il DSR di un terminale è collegato con il DTR dell’altro terminale e viceversa
- Il RTS di un terminale è collegato con il CTS dell’altro e viceversa
- In questa configurazione RTS serve a comunicare la capacità di ricevere messaggi più che una vera richiesta di invio.
- Il TX di un terminale è collegato con il RX dell’altro e viceversa

NOTA: nella configurazione usata negli esempi, i sistemi sono entrambi inizializzati ponendo DTR=1 e RTS=1, per cui l’handshaking è stabilito in partenza.



Usart – registri

All’interfaccia seriale vengono assegnati due indirizzi consecutivi di memoria, attraverso cui è possibile accedere ai suoi registri interni:

- MODE: primo accesso in scrittura dopo RESET all’indirizzo *dispari*
- SYNC1/SYNC2: secondo e terzo accesso in scrittura all’indirizzo *dispari* se è stata settata una comunicazione sincrona (dipende da quanti caratteri di sincronizzazione sono stati specificati)
- CNTRL: successive scritture all’indirizzo *dispari*
- STATUS: lettura all’indirizzo *dispari*
- DATO: lettura/scrittura all’indirizzo *pari*



MODE register

MODE |0 |1 |0 |1 |1 |1 |0 |1 | *
* | | | | | | | Trasmissione Asincrona
* | | | | | | Non utilizzato
* | | | | | 8 bit per dato
* | | | | | * bit di parità
* | | | | tipo di parità dispari
* | | | | * 2 bit di stop
* | | | | #bit di sync in trasmissione asincrona

CONTROL register

CNTRL	1		1		1		1		0		1		1		1		-	*
*																Abilita trasmittitore (TxEN)	*	
*																Attiva DTR (Data Terminal Ready): pronto a comunicare	*	
*																Attiva ricevitore (RxEN)	*	
*																Non utilizzato	.	
*																Azzera bits di errore in STATUS	*	
*																Attiva RTS (Request to Send): indica intenzione a trasmettere;	.	
*																Resetta la periferica	*	
*																Ricerca sincronizzazione (HUNT MODE)	.	



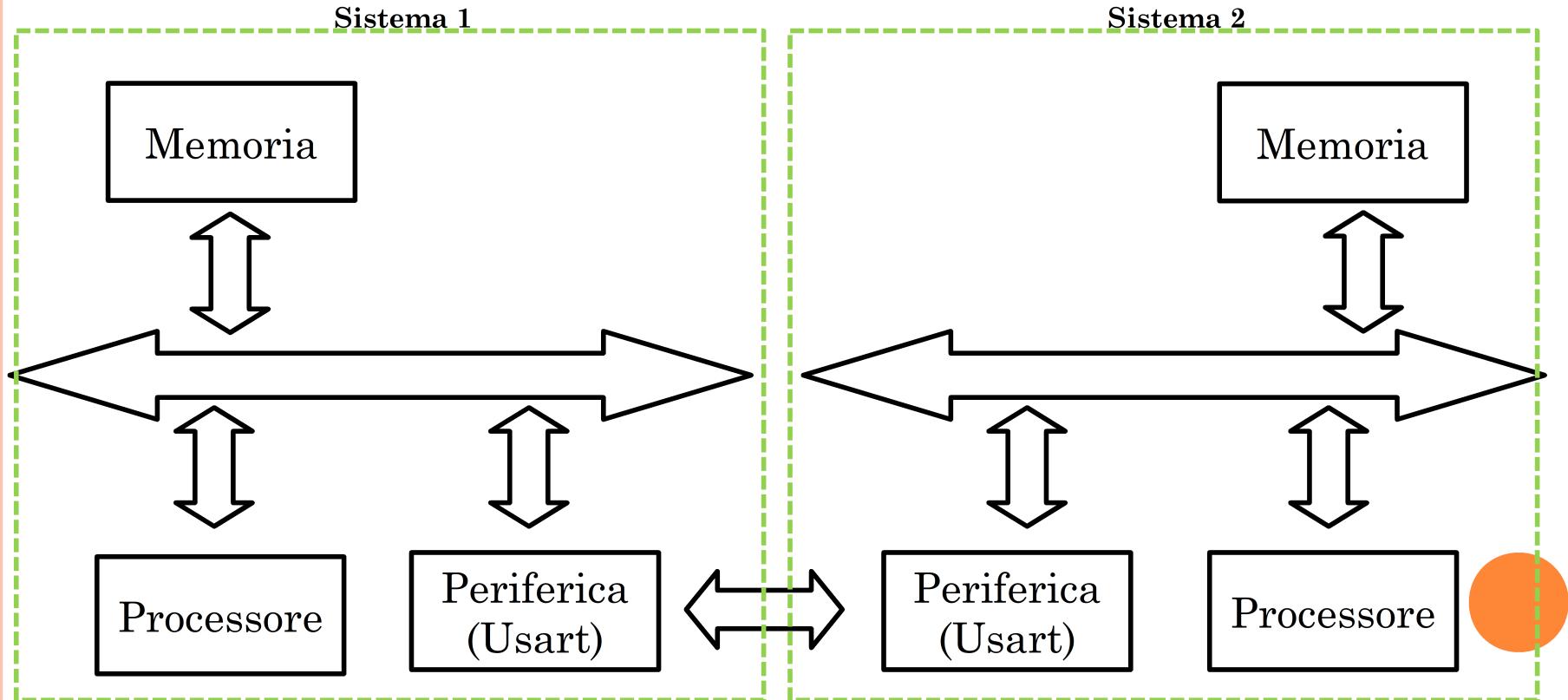
STATUS register

STATUS |1|0|0|0|0|0|0|0|0|

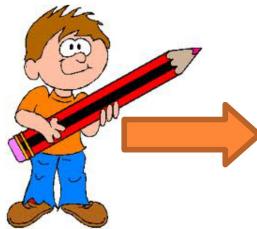
- * | | | | | | | b0 (TxRDY) diviene alto quando DATA OUT viene copiato in TSHIFT REG., torna basso quando il processore copia un nuovo carattere in DATA OUT
- * | | | | | | b1 (RxRDY) diviene alto quando RSHIFT REG. viene copiato in ^{*}DATA IN, torna basso in seguito a lettura da DATAIN
- * | | | | | TxEMPTY (non ci sono caratteri da inviare)
- * | | | | Errore di parità
- * | | | | Errore di overrun
- * | | | Errore di framing
- * | | Rilevati bit di sincronismo
- * | DSR attivo

Configurazione

Consideriamo due sistemi S1 ed S2 che comunicano tramite due periferiche seriali secondo una trasmissione asincrona.



Operazione di scrittura (senza interrupt in trasmissione)



Il sistema 1 vuole scrivere un dato sulla seriale

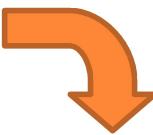
Controlla che DSR sia alto



Controlla che il **bit 0** del registro di stato sia alto (no int in tx)



Scrive il carattere in DATA OUT: il **bit 0** del registro di stato si abbassa



Alla fine della trasmissione del carattere, il **bit 0** del registro di stato si alza e viene attivato il segnale di interruzione **TxRDY**.



Il sistema 2 legge il dato

Ipotesi di partenza:
DTR e RTS sono attivi
(configurazione del registro CNTRL)
DSR e CTS sono attivi perché collegati ai segnali DTR e RTS del sistema gemello.

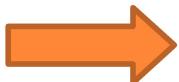
Il **bit 0** del registro di stato è alto per segnalare che la periferica è pronta a trasmettere

NB: Il segnale di interruzione **TxRDY** si alza alla fine della trasmissione di un carattere, quando il dato in DATA OUT viene copiato in Tshift per una nuova trasmissione.

Per il primo carattere la scrittura in DATA OUT è subito seguita dalla copia in Tshift: ciò attiva il segnale TxRDY e mette a 1 il bit0 del registro di stato prima che il carattere sia stato trasmesso. Un nuovo carattere viene scritto in DATA OUT ma verrà copiato in Tshift solo alla fine della trasmissione precedente.

Operazione di lettura

Il bit 1 del registro
di stato, che
corrisponde a
RxRDY, torna a
zero



Controlla eventuali
errori



Il sistema 2
legge il dato
(il segnale
RxRDY è
alto)



NB: Il segnale di interruzione RxRDY si
alza alla ricezione di un carattere, che
viene rilevata tramite opportuna rete di
controllo (macchina sequenziale), non
visibile nel modello di programmazione.

