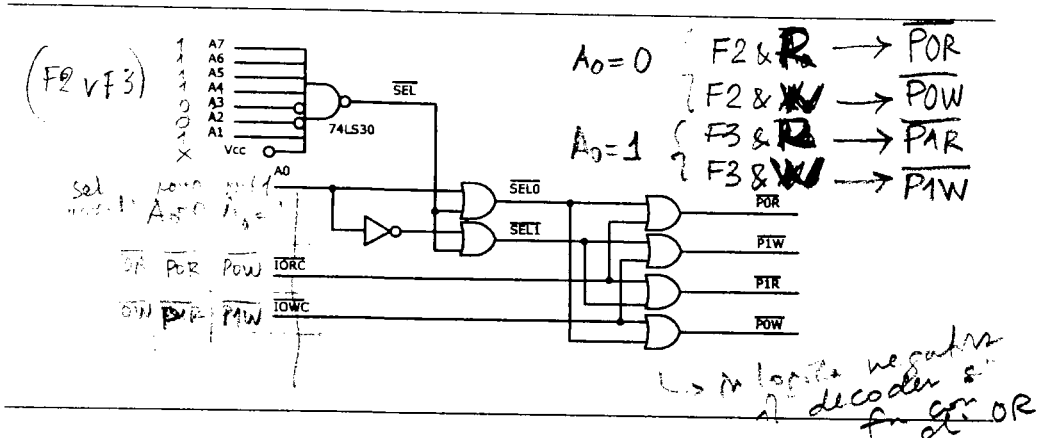

Figura 8.3 Esempio di decodifica degli indirizzi per una interfaccia che presenta due porte di ingresso e due di uscita. La linea $\overline{SEL}$ rappresenta l'uscita del decodificatore dell'*indirizzo di base* assegnato all'interfaccia (per semplicità si suppone che l'indirizzo sia dato su soli 8 bit). Si noti che la linea $\overline{SEL}$ risulta asserita in presenza dell'indirizzo F2 e dell'indirizzo F3. A_0 serve a discriminare la porta all'indirizzo relativo 0 da quella all'indirizzo relativo 1.

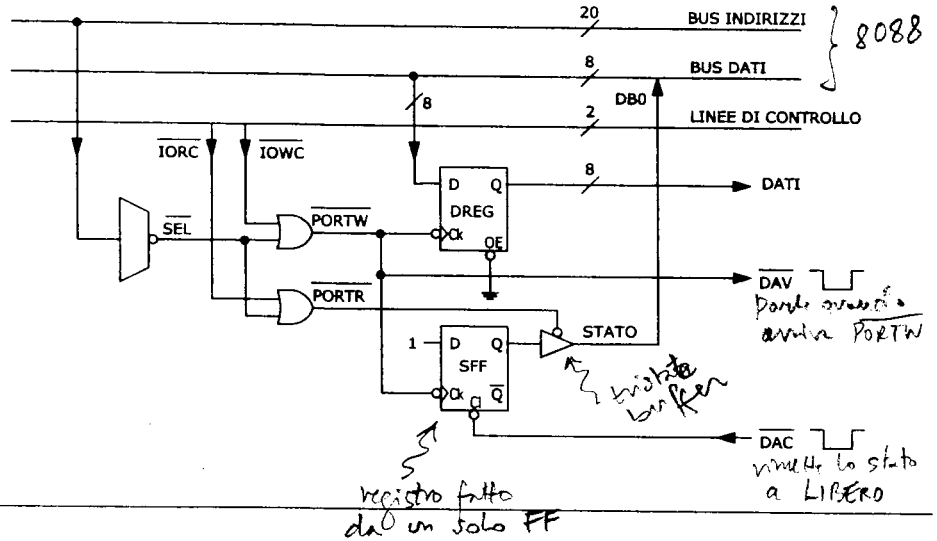

Figura 8.7 Esempio di interfaccia di uscita. La linea $\overline{SEL}$ risulta asserita solo in presenza dell'indirizzo assegnato all'interfaccia. Il registro dati dell'interfaccia (DREG) è costituito da un latch di 8 bit, la cui uscita è tenuta sempre abilitata.

si stabilisce che il sottoprogramma STAMPA venga chiamato secondo questa convenzione[4]:

```
MOV   SI,<offset BUFFER>   ;DS:SI= indirizzo di BUFFER
MOV   CX,<n>               ;CX= n
CALL  STAMPA
```

Si stabilisce inoltre che STAMPA effettui per intero il trasferimento degli n caratteri e restituisca il controllo all'istruzione successiva a quella di chiamata solo a trasferimento concluso.

Il prelievo e il trasferimento di un singolo carattere richiede questa semplice sequenza di istruzioni:

```
MOV   AL,[SI]
OUT   PORT,AL
```

Dove con PORT si è indicato l'indirizzo assegnato all'interfaccia. Come già osservato, l'istruzione OUT copia il contenuto di AL in DREG dell'interfaccia, fa passare SFF a 1 e trasmette il $\overline{DAV}$ alla stampante.

La trasmissione degli n caratteri richiede l'iterazione delle due precedenti istruzioni in un ciclo che mantenga SI aggiornato in modo da puntare al prossimo carattere da stampare. È però necessario sincronizzare l'esecuzione del programma con l'attività del periferico. A tal fine basta introdurre un ciclo di attesa del cambiamento di stato di SFF (conseguente alla risposta della stampante):

```
ATTESA:  IN    AL,PORT
         AND   AL,1
         JNZ   ATTESA
```

Terminata l'attesa, è necessario conteggiare il carattere trasmesso e, a seconda del fatto che questo sia o non sia l'ultimo:

- concludere e tornare al chiamante;
- prelevare e trasmettere il prossimo carattere.

Tenuto conto delle precedenti osservazioni, il sottoprogramma STAMPA assume questa forma finale[5]:

```
STAMPA:  MOV   AL,[SI]
         OUT   PORT,AL
ATTESA:  IN    AL,PORT
         AND   AL,1
         JNZ   ATTESA
         INC   SI
         LOOP  STAMPA
         RET
```

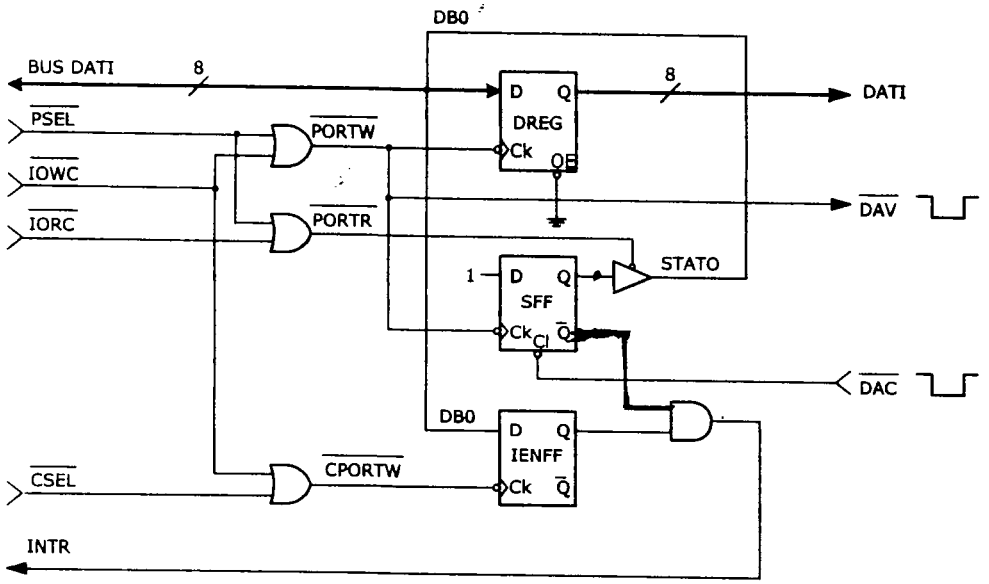

Figura 8.10 Interfaccia di uscita modificata con l'aggiunta della richiesta di interruzione.

I programmi vedranno solo la routine di inizializzazione. A questa daremo ancora il nome di STAMPA e manterremo la precedente convenzione di chiamata[9]:

```
MOV   SI, <offset BUFFER> ; DS:SI -> BUFFER
MOV   CX, <n>             ; CX = n
CALL  STAMPA
```

Sezione di inizializzazione Con le precedenti assunzioni e convenzioni risulta agevole costruire la sezione di inizializzazione:

```
STAMPA:  MOV   BUSY,1          ; Driver occupato!
         MOV   IND,SI          ; Salvataggio indirizzo
         MOV   CONT,CX         ; ....e contatore
         MOV   AL,[SI]         ; AL <- primo carattere
         OUT   PORT,AL         ; Trasmissione carattere
         MOV   AL,1            ; Abilita l'interfaccia
         OUT   CPORT,AL        ; a generare interruzioni
         RET                   ; Ritorno al chiamante
```

Nello scrivere il precedente codice si è assunto che l'interfaccia fosse nello stato in cui non genera interruzioni (IENFF in stato 0). (e che il suo stato fosse "LIBERO")

Routine di servizio dell'interruzione Come abbiamo già esposto in precedenza, la routine di servizio della interruzione inizia con le istruzioni di salvataggio nello stack dei registri utilizzati dalla routine e della parola di stato del processore[10]:

```
INTSTAMP: PUSH  PSW  (PUSHF)   ; Salvataggio PSW
          PUSH  AX             ; Salvataggio dei
          PUSH  CX             ; ...tre registri
          PUSH  SI             ; .....usati in INTSTAMP
```

A questo punto c'è da conteggiare il carattere trasmesso e prelevare il prossimo. A tal fine si deve recuperare il contenuto di IND e CONT e aggiornarli.

```
MOV   SI,IND          ; SI<-Offset prossimo
INC   SI              ; Puntamento al prossimo
MOV   CX,CONT         ; CX<-restanti + 1
DEC   CX              ; Conteggio
```

[9]Si noti che la sequenza di chiamata presuppone che il driver sia libero, ovvero che BUSY valga 0. In pratica, prima di effettuare la chiamata è necessario verificare il rispetto di questa condizione.

[10]Nei microprocessori della famiglia 8086, di cui si è preso a prestito la notazione assembler, l'operazione PUSH PSW viene eseguita come parte del ciclo di interruzione, prima di portare a 0 IE. PSW salvata contiene quindi 1 per IE (altrimenti non ci sarebbe stata l'interruzione). L'istruzione IRET ha l'effetto di rimettere PSW salvata sullo stack e quindi di riabilitare il sistema di interruzione. Nel caso della nostra ipotetica CPU, IRET ha solo l'effetto di riportare a 1 IE (oltre che tornare al programma interrotto); dunque IE non fa parte di PSW.

Prima di procedere occorre stabilire se ci sono o meno altri caratteri da stampare. L'ultima istruzione è funzionale a tale scopo. In sintesi, risulta il seguente codice[11]:

```
INTSTAMP: PUSH  PSW            ; Sal-
          PUSH  AX             ; --va-
          PUSH  CX             ; ----tag-
          PUSH  SI             ; ------gio
          MOV   SI,IND         ; SI<-Offset prossimo
          INC   SI             ; Puntamento al prossimo
          MOV   CX,CONT        ; CX<-restanti + 1
          DEC   CX             ; Conteggio
          JZ    FINE           ; Era l'ultimo?
          MOV   AL,[SI]        ; NO! Si prende il prossimo
          OUT   PORT,AL
          MOV   IND,SI         ; Salvataggio puntatore e
          MOV   CONT,CX        ; ..contatore aggiornati
ESCI:     POP   SI             ; Ri-
          POP   CX             ; --pri-
          POP   AX             ; ----sti-
          POP   PSW            ; ------no
          IRET                 ; Al punto di interruzione
FINE:     MOV   BUSY,0         ; Tutti trasmessi
          MOV   AL,0           ; Driver libero !
          OUT   CPORT,AL       ; Disabilita intr interfaccia
```