Interrupt nell'Apple II

di Francesco Meschia - Asti

Sul numero 36 della rivista ho letto la prova dell'Amstrad CPC 464 in cui si parlava delle possibilità di gestione degli interrupt del suddetto computer (AFTER ed EVERY).

Incoraggiato anche dalla routine HEX-BASIC comparsa sullo stesso numero mi sono deciso a scrivere una routinetta che possa gestire gli interrupt sul mio Apple.

Per chi non lo sapesse, dirò che gli interrupt, che si possono abbreviare con IRQ (Interrupt ReQuest) sono segnali prodotti via hardware o via software che forzano il microprocessore a interrompere il lavoro che sta eseguendo per saltare ad una particolare routine.

Le routine che ora presento servono proprio a forzare l'interprete a interrompere l'esecuzione del programma Basic per saltare a una routine.

Dimenticavo di dire che questi interrupt vengono eseguiti ogni tanto, e in questo particolare caso più di dieci volte al secondo, a seconda della lunghezza della routine che deve venire periodicamente eseguita.

```
0300-   A9 4C      LDA   #$4C
0302-   A2 18      LDX   #$18
0304-   A0 03      LDY   #$03
0306-   85 BA      STA   $BA
0308-   86 BB      STX   $BB
030A-   84 BC      STY   $BC
030C-   A9 4C      LDA   #$4C
030E-   A2 2B      LDX   #$2B
0310-   A0 03      LDY   #$03
0312-   85 0A      STA   $0A
0314-   86 0B      STX   $0B
0316-   84 0C      STY   $0C
0318-   C9 3A      CMP   #$3A
031A-   B0 0C      BCS   $0328
031C-   20 4A FF   JSR   $FF4A
031F-   20 40 03   JSR   $0340
0322-   20 3F FF   JSR   $FF3F
0325-   4C BE 00   JMP   $00BE
0328-   4C C8 00   JMP   $00C8
032B-   A9 C9      LDA   #$C9
032D-   A2 3A      LDX   #$3A
032F-   A0 B0      LDY   #$B0
0331-   85 BA      STA   $BA
0333-   86 BB      STX   $BB
0335-   84 BC      STY   $BC
0337-   60         RTS
```

Figura 1 - Disassemblato della routine che attiva una specie di interrupt che forza l'interprete Applesoft ad eseguire una nostra routine ogni volta che preleva un carattere dal programma in esecuzione. La routine che deve essere eseguita si trova alla locazione $300 il cui salto è indicato dalla freccia. Se si sposta la routine ciclica si deve modificare il relativo JSR.

La routine del listato 1 è il manipolatore di interrupt vero e proprio. Questa routine deve venire copiata e salvata su disco. Per attivarla è necessario caricare la routine "periodica", caricare il manipolatore, modificare il JSR ROUTINE se la routine "periodica" non è allocata alla $340, operazione eseguibile da Basic con:
*POKE 780, indirizzo INT (indirizzo/256)*256*
POKE 781, INT(indirizzo/256)
e infine attivare l'interrupt con CALL 768. Per disattivare gli interrupt basta dare A = USR(0). Da questo momento l'interprete non interromperà più il suo lavoro finché non verrà data nuovamente CALL 768.

Il listato 2 è un altro programmino in Assembler chiamato IRQ SPRITE (nome dato pensando con nostalgia alla CALL SPRITE del fu-TI99), che è in grado di far muovere un pixel per lo schermo indipendentemente dal programma.

N.B. Il manipolatore va salvato su disco con il nome di IRQ.OBJ0, A$ 300, L$ 39 e il listato 2 va salvato con il nome di SPRITE.OBJ0, A$ 340, L$ 64.

Per sperimentare queste routine basta fare girare il programmino Basic del listato 3 e vedere il risultato, oppure copiare il listato 4. Questo programma è un gioco che usa la routine SPRITE e consiste nel centrare un pixel che si muove per lo schermo. Poiché il pixel va ad interrupt il suo movimento è velocissimo e difficile da seguire.

Un'ultima avvertenza: gli interrupt non vengono più eseguiti quando l'interprete incontra una WAIT, una INPUT o una GET o quando si salta ad una routine in linguaggio macchina.

Naturalmente questi sono solo dei piccolissimi esempi delle possibilità degli interrupt. Altri usi sono: orologi in tempo reale, musica indipendente dai programmi e tante altre cosette.

Quindi chiunque abbia una conoscenza del linguaggio Assembler 6502 può modificare le routine o crearne di nuove. Per questi puntualizzo che prima di eseguire routine grafiche ad interrupt, da Assembler è consigliabile salvare nello stack e poi riprendere tutti i puntatori grafici (come nella routine SPRITE).

Chiunque abbia sviluppato routine inte-

```
0340-   A6 E4      LDX   $E4
0342-   86 07      STX   $07
0344-   A5 E0      LDA   $E0
0346-   48         PHA
0347-   A5 E1      LDA   $E1
0349-   48         PHA
034A-   A5 E2      LDA   $E2
034C-   48         PHA
034D-   A5 1D      LDA   $1D
034F-   48         PHA
0350-   A5 26      LDA   $26
0352-   48         PHA
0353-   A5 27      LDA   $27
0355-   48         PHA
0356-   A5 52      LDA   $52
0358-   48         PHA
0359-   A5 E5      LDA   $E5
035B-   48         PHA
035C-   A5 1C      LDA   $1C
035E-   48         PHA
035F-   A5 30      LDA   $30
0361-   48         PHA
0362-   A2 00      LDX   #$00
0364-   20 F0 F6   JSR   $F6F0
0367-   A6 06      LDX   $06
0369-   A0 00      LDY   #$00
036B-   A9 41      LDA   #$41
036D-   20 57 F4   JSR   $F457
0370-   E6 06      INC   $06
0372-   A2 07      LDX   #$07
0374-   20 F0 F6   JSR   $F6F0
0377-   A6 06      LDX   $06
0379-   A0 00      LDY   #$00
037B-   A9 41      LDA   #$41
037D-   20 57 F4   JSR   $F457
0380-   A5 07      LDA   $07
0382-   85 E4      STA   $E4
0384-   68         PLA
0385-   85 30      STA   $30
0387-   68         PLA
0388-   85 1C      STA   $1C
038A-   68         PLA
038B-   85 E5      STA   $E5
038D-   68         PLA
038E-   85 52      STA   $52
0390-   68         PLA
0391-   85 27      STA   $27
0393-   68         PLA
0394-   85 26      STA   $26
0396-   68         PLA
0397-   85 1D      STA   $1D
0399-   68         PLA
039A-   85 E2      STA   $E2
039C-   68         PLA
039D-   85 E1      STA   $E1
039F-   68         PLA
03A0-   85 E0      STA   $E0
03A2-   60         RTS
```

Figura 2 - Routine ciclica che muove un punto sullo schermo di una posizione ogni volta che viene richiamata.

```
100  REM  PROVA INTERRUPT
110  PRINT  CHR$ (4)"BLOAD SPRITE.OBJO"
120  PRINT  CHR$ (4)"BRUN IRQ.OBJO"
130  HGR
140  CALL 768: REM  ATTIVA L'INTERRUPT
150  FOR LOOP = 1 TO 10000
160 A =  SQR (1)
170  NEXT
180 A =  USR (0): REM  DISATTIVA INTERRUPT
190  REM  LA RIGA 160 SERVE COME RITARDO
```

Figura 3 - Listato di un programmino di prova della routine Interrupt.

ressanti ad interrupt può contattarmi telefonandomi o scrivendomi. Il mio indirizzo è:
Francesco Meschia,
89, Viale Partigiani - 14100 Asti
Tel. (0141) 217475

Un po' di chiarezza...

Nonostante il funzionamento del programma sia corretto, ci sono nel testo dell'autore alcune inesattezze che è il caso di precisare, soprattutto per non ingenerare confusione nei lettori meno esperti di programmazione in linguaggio macchina.

Ci sono due tipi di interrupt nel 6502: l'NMI e l'IRQ.

Il primo (Non Maskerable Interrupt) è un comando brutale dato al microprocessore da una periferica perché, appena terminata l'istruzione in corso (sei microsecondi nel caso peggiore), venga dato corso all'apposita routine di servizio dell'interrupt non mascherabile (in genere è successo qualcosa di grave tipo una mancanza di corrente o la raccolta di un dato da una periferica che non può assolutamente attendere).

Il secondo (Interrupt Request) è una richiesta di interruzione che una periferica rivolge via hardware al microprocessore e che viene da questi eseguita appena possibile. In questo 'appena possibile' è sottintesa una capacità software di gestione dell'IRQ sia per quanto riguarda il momento dell'esecuzione che per quanto riguarda la priorità tra periferiche che chiedessero contemporaneamente di essere ascoltate.

È in genere l'IRQ che si usa per effettuare delle operazioni cicliche senza dover tenere impegnato il microprocessore per tutto il tempo del servizio. Ad esempio se si collega il filo del READY della stampante all'IRQ si può continuare a lavorare mentre la stampante scrive (naturalmente con un programmino di gestione e un'area di memoria dedicata al buffer di stampa).

L'IRQ può essere attivato anche da una istruzione in linguaggio macchina: il BREAK (codice 0); ma in questo caso viene gestito dall'Apple in modo diverso e si usa principalmente per il debug dei programmi in Assembler.

Risulta chiaro perciò che, essendo l'interrupt una richiesta di servizio hardware, non si può provocare da Basic (a meno che questo non sia stato previsto dal sistema operativo).

```
100  REM  PIXEL HUNTER
110  TEXT : HOME
120  RESTORE : FOR DA = 32768 TO 32822: READ A: POKE
     DA,A: NEXT
130  POKE 232,0: POKE 233,128: SCALE= 1: ROT= 0
140  IF  PEEK (768) = 169 THEN 170
150  PRINT  CHR$ (4)"BLOAD SPRITE.OBJO"
160  PRINT  CHR$ (4)"BRUN IRQ.OBJO"
170  REM  PRESENTAZIONE
180 A =  USR (0): REM  INTERRUPT DISABLE
190  HOME
200  VTAB 3: PRINT "     ******** PIXEL HUNTER  *
     ********"
210  VTAB 6: PRINT "IN QUESTO GIOCO IL CACCIATORE
      DI PIXEL__DEVE COLPIRE, CON LA SUA ASTRONAV
     E, IL__PIXEL CHE CORRE A META' SCHERMO PRIMA
     ___CHE IL TIMER VADA A ZERO."
220  PRINT "   MAN MANO CHE CATTURERAI I PIXEL IL
     ___TUO TEMPO UTILE DIMINUIRA'."
230  PRINT "   USA IL PADDLE 0 PER MUOVERE L'ASTR
     O-_NAVE E IL BOTTONE PER SPARARE."
240  PRINT : PRINT : PRINT "____$$$$$$$$  BUONA C
     ACCIA  $$$$$$$$": VTAB 22: HTAB 20: PRINT "-
      PREMI UN TASTO -"
250  POKE  - 16368,0: WAIT  - 16384,128: REM  ASP
     ETTA UN TASTO
260  POKE  - 16368,0
270  REM  INIZIALIZZA VARIABILI & PAGINA ZERO
280  POKE 6,0
290 TI = 30:SP =  - 16336
300  HOME : HGR
310  REM  PLOTTA STELLE
320  HCOLOR= 7
330  FOR I = 1 TO 50
340 X =  RND (1) * 275:Y =  RND (1) * 140
350  DRAW 2 AT X,Y: NEXT
360  CALL 768: REM  INTERRUPT ENABLE
370  REM  LOOP PRINCIPALE DEL GIOCO
380  FOR LO = 1 TO 10000
390 X =  PDL (0):BU =  PEEK ( - 16287)
400 TI = TI - .5
410  IF TI < 0 THEN 600
420  HCOLOR= 7
430  DRAW 1 AT X,150
440  IF BU > 127 THEN 490
450  VTAB 21: PRINT "----------------------------
     ------------": VTAB 22: PRINT "   TIME LEFT:"
      INT (TI);" ": TAB( 25);"SCORE:"SC
470  HCOLOR= 0: DRAW 1 AT X,150
480  NEXT
490  REM  SPARO E SUONO
500  HPLOT X + 4,135 TO X + 4,65: REM  LASER
510  FOR SO = 1 TO 5:H =  PEEK (SP) +  PEEK (SP):
      NEXT
530  HCOLOR= 0
540  HPLOT X + 4,135 TO X + 4,65
550  IF  PEEK (6) > X - 12 AND  PEEK (6) < X + 20
      THEN SC = SC + 200:TI = 30 - (SC / 200): CALL
     64477:LO = 1: GOTO 450
560  GOTO 450
570  REM  DATA SHAPE TABLE
580  DATA  2,0,6,0,51,0,45,45,45,45,228,28,28,28,
```

(40)

(continua a pagina 130)

Il programma dell'autore infatti non utilizza affatto gli interrupt del microprocessore, ma si aggancia alla funzione GETCHAR ($B1) già vista nello scorso numero, che l'interprete Applesoft utilizza per leggere dalla memoria il programma che sta eseguendo. L'aggancio è fatto in modo tale che ogni volta che l'Applesoft preleva un carattere viene costretto anche ad eseguire una nostra routine (locata a $340). Questo spiega perché l'interruzione non viene eseguita durante certe istruzioni (INPUT, GET ecc) e durante le routine in linguaggio macchina chiamate con una normale CALL. Gli interrupt hardware (IRQ) invece, se non disabilitati con una apposita istruzione, vengono sempre eseguiti al momento della richiesta. MC

(segue da pagina 129)

```
      36,36,52,54,54,30,30,30,30,77,9,24,52,54,62,
      63,47,45,45,45,53,63,63,63,63,36,100,77,36,3
      6,0,200,200,144,18,36,0,0,5,0,0
590   REM  TIME=0
600   HCOLOR= 5
610   FOR DR = 0 TO 15
620   FOR SD = 1 TO 3:H =  PEEK (SP): NEXT
630   HPLOT X,150 TO  RND (1) * 270 + 2, RND (1) *
      100 + 40: NEXT
640   HOME
650   FOR I = 1 TO 10
660   POKE  - 16303,0
670   VTAB 10: HTAB 15: INVERSE : PRINT "GAME OVER
      "
680   FOR T = 1 TO 30: NEXT
690   POKE  - 16304,0: POKE  - 16297,0: POKE  - 16
      300,0: NEXT
700   NORMAL : FOR I = 1 TO 100: NEXT
710   GOTO 170
```

Figura 4 - Listato di un gioco realizzato con la routine di Interrupt. Si deve colpire un punto che corre sullo schermo sotto controllo della routine in Linguaggio Macchina di figura 2. I trattini di sottolineatura dentro le PRINT vanno sostituiti con altrettanti spazi.

Le routine dell'Applesoft

La routine di questa volta permette di passare uno o più parametri, di un solo byte, ad un nostro programma in linguaggio macchina. Serve in pratica ad eliminare gli antiestetici richiami tipo:

```
100 POKE 10,45:POKE 11,30:CALL 768
```

che vengono sostituiti da una semplice:

```
100 CALL 768,45,30
```

I valori dei parametri possono anche essere delle variabili o delle espressioni complesse tipo:

```
100 CALL 769,PEEK(243)/16+1
```

In questo caso viene calcolata tutta l'espressione dopo la virgola e il risultato, se minore di 255, sarà passato alla nostra routine.

Essendo essenzialmente un sottoprogramma del Basic Applesoft, la routine controlla anche la validità sintattica dell'espressione e tutti gli altri tipi di errore che potrebbero verificarsi (ad esempio un vettore sovradimensionato o una stringa al posto di un numero).

Il nome della routine è GETPARM (almeno così la chiamo io) in quanto corrisponde alla seconda parte della routine di POKE, quella cioè che si occupa del calcolo del parametro da depositare nell'indirizzo desiderato.

Descrizione

GETPARM - Calcola il valore di una espressione preceduta dalla virgola e deposita il risultato nel Registro X.

Se il risultato è maggiore di 255 segnala errore. Esegue inoltre tutti i normali controlli del Basic su una espressione. Si può usare ripetutamente per prelevare più di un valore.

Il ritorno al Basic avviene tramite l'entry B7 della routine CHARGET ($B1), per cui nell'accumulatore si trova il carattere successivo della riga Basic; di norma $00 se la riga è finita o $3A (:) se ci sono altre istruzioni. Si può utilizzare l'entry $E74F che scavalca il controllo di presenza della virgola, ma in questo caso bisogna far precedere il JSR GETPARM da un JSR CHARGET. Il carattere di separazione può così essere uno dei seguenti:

:;'"()[]!?$≠&_@\ e le lettere da A a Z.

GETPARM $ E74C

Parametri in entrata	Parametri in uscita
nessuno	nessuno

Registri in entrata		in uscita
Accumulatore	qualsiasi	car. succ. del Basic
Reg. X	"	valore del parametro
Reg. Y	"	zero
STATUS	"	come da JSR $B1
Stack point	"	non modificato

Entry: $E74F - non controlla la virgola ma deve essere preceduto da un JSR $B1 .

Note: Se il valore dell'espressione è maggiore di 255 segnala ILLEGAL QUANTITY ERROR, o SYNTAX ERROR se non riesce a decifrare 1,espressione. Rientra al Basic tramite la routine GETCHAR ($B1).

Al rientro della GETPARM il registro Y risulta azzerato, X contiene ovviamente il valore del parametro mentre tutti gli altri registri si trovano come al rientro da una GETCHAR.

Esempio:

```
0300-   20 B1 00    JSR   $00B1  salta la parentesi aperta
0303-   20 4F E7    JSR   $E74F  legge il primo valore
0306-   86 0B       STX   $0B    lo deposita in B
0308-   20 B1 00    JSR   $00B1  salta la parentesi chiusa
030B-   20 4C E7    JSR   $E74C  legge il secondo numero
030E-   86 0C       STX   $0C    lo deposita in C
0310-   A5 0B       LDA   $0B    recupera il carattere
0312-   20 ED FD    JSR   $FDED  e lo stampa
0315-   C6 0C       DEC   $0C    finché C
0317-   D0 F9       BNE   $0312  non va a zero
0319-   4C B7 00    JMP   $00B7  poi rientra al Basic.

]CALL 768(170),30
**********************************
]
```

Stampa il carattere corrispondente al valore tra parentesi tante volte quanto c'è scritto dopo la virgola.

Elenco del software disponibile su cassetta o minifloppy

Per ovviare alle difficoltà incontrate da molti lettori nella digitazione dei listati pubblicati nelle varie rubriche di software sulla rivista, MCmicrocomputer mette a disposizione i programmi più significativi direttamente su supporto magnetico. Riepiloghiamo qui a fianco i programmi disponibili per le varie macchine, ricordando che i titoli non sono previsti per computer diversi da quelli indicati.
Il numero della rivista su cui viene descritto ciascun programma è riportato nell'apposita colonna; consigliamo gli interessati di procurarsi i relativi numeri arretrati, eventualmente rivolgendosi al nostro Servizio Arretrati utilizzando il tagliando pubblicato in fondo alla rivista.

Per l'ordinazione inviare l'importo (a mezzo assegno, c/c o vaglia postale) alla Technimedia srl, Via Valsolda 135, 00141 Roma.

Le cassette utilizzate sono Basf C-60 Compusette II; i minifloppy sono Basf singola faccia singola densità.

Codice	Titolo programma	MC n.	Prezzo	Note
	APPLE II			
DA2/00	Shape Tablet	22	15000	
DA2/01	Motomuro	26	15000	
DA2/02	&DEBUG	28	15000	
DA2/03	EDIT + INPUT	29	15000	
DA2/04	Basic modulare	34	15000	
DA2/05	ANNA Animation Lang.	35/37	15000	
DA2/06	Miniset + Leva-DOS	37	15000	
DA2/07	27 programmi grafici	38	30000	
DA2/08	Adventure Editor	38	15000	
	COMMODORE 64			
C64/01	Briscola	25	17000	
C64/02	Serpentone	29	17000	
C64/03	Othello	29	17000	
C64/04	Chase	33	17000	
C64/05	Spreadsheet	34	30000	
C64/06	Bilancio familiare	35	17000	
C64/07	The dark wood	36	17000	
C64/08	Totocalcio: sis.rid.	37	17000	
C64/09	Orchetes	37	17000	
C64/10	Wordprocessor	38	17000	
C64/11	Helicopt	38	17000	
C64/12	Finestra grafica	39	17000	
C64/13	Paroliamo	39	17000	
C64/14	Scarabeo	40	17000	
C64/15	Magazzino	41	17000	
D64/01	Spreadsheet	34	15000	
D64/02	ADP Basic	da 35 a 39	15000	
D64/03	Wordprocessor	38	15000	
D64/04	Paroliamo	39	15000	
D64/05	Data base Galileo	40/41	15000	
D64/06	Magazzino	41	15000	
	COMMODORE VIC-20			
CVC/01	VIC-Maze	19	17000	Config. base
CVC/02	Pic-Man	23	17000	Config. base
CVC/03	Briscola	25	17000	Config. base
CVC/04	Grand Prix	28	17000	Config. base
CVC/05	Frogger	26	17000	RAM: almeno + 3 K
CVC/06	Invaders	29	23000	RAM: + 16 K
CVC/07	Othello	29	17000	RAM: + 16 K
CVC/08	SKI	31	17000	Config. base
CVC/09	VIC-quiz	32	17000	RAM: almeno + 8 K
CVC/10	Zigurat	33	17000	Config. base
CVC/11	Extended Basic	36	17000	RAM: + 16 K
CVC/12	Fireman	36	17000	Config. base
CVC/13	Accordi' per chitarra	39	17000	RAM: almeno + K
CVC/14	Piramide di Iunnuh	39	17000	RAM: almeno + K
CVC/15	Il castello	40	17000	RAM: + 16 K
DVC/01	EXMA	27/28	15000	RAM: + 16 K
	SINCLAIR SPECTRUM			
CSS/01	TRILAB	28	17000	
CSS/02	SET di caratteri	27/29	17000	
CSS/03	Grafica TREDIM	29	17000	
CSS/04	Ippica	30	17000	
CSS/05	Graphic-Comp	32	17000	
CSS/06	Macchina del tempo	34	17000	
CSS/07	Piramide di Iunnuh	35	17000	
CSS/08	Over Basic	37	17000	
CSS/09	Prospettiva	38	17000	
CSS/10	Motomuro	39	17000	48 K RAM
CSS/11	Othello	40	17000	
CSS/12	The dark wood	40	17000	
CSS/13	Musica	41	17000	
	TEXAS TI-99/4A			
CT9/01	Macchina del tempo	27	17000	
CT9/02	Simon	29	17000	
CT9/03	Babilonia	30	17000	
CT9/04	Labirinto 3D	31	17000	
CT9/05	Piramide di Iunnuh	33	17000	Extended Basic
CT9/06	Scrabble	34	17000	
CT9/07	Morphy	35	17000	
CT9/08	Equo canone	37	17000	
CT9/09	Scopa	39	17000	
CT9/10	Montecarlo	39	17000	Extended Basic
CT9/11	Totocalcio	41	30000	

Nota:
l'iniziale del codice e' C per le cassette, D per i minifloppy