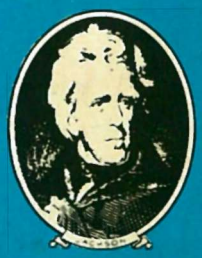
Vogliamo
Incominciare
Così ?

IMPARIAMO A PROGRAMMARE IN BASIC CON IL VIC™/CBM™

Rita
Bonelli

GRUPPO
EDITORIALE
JACKSON



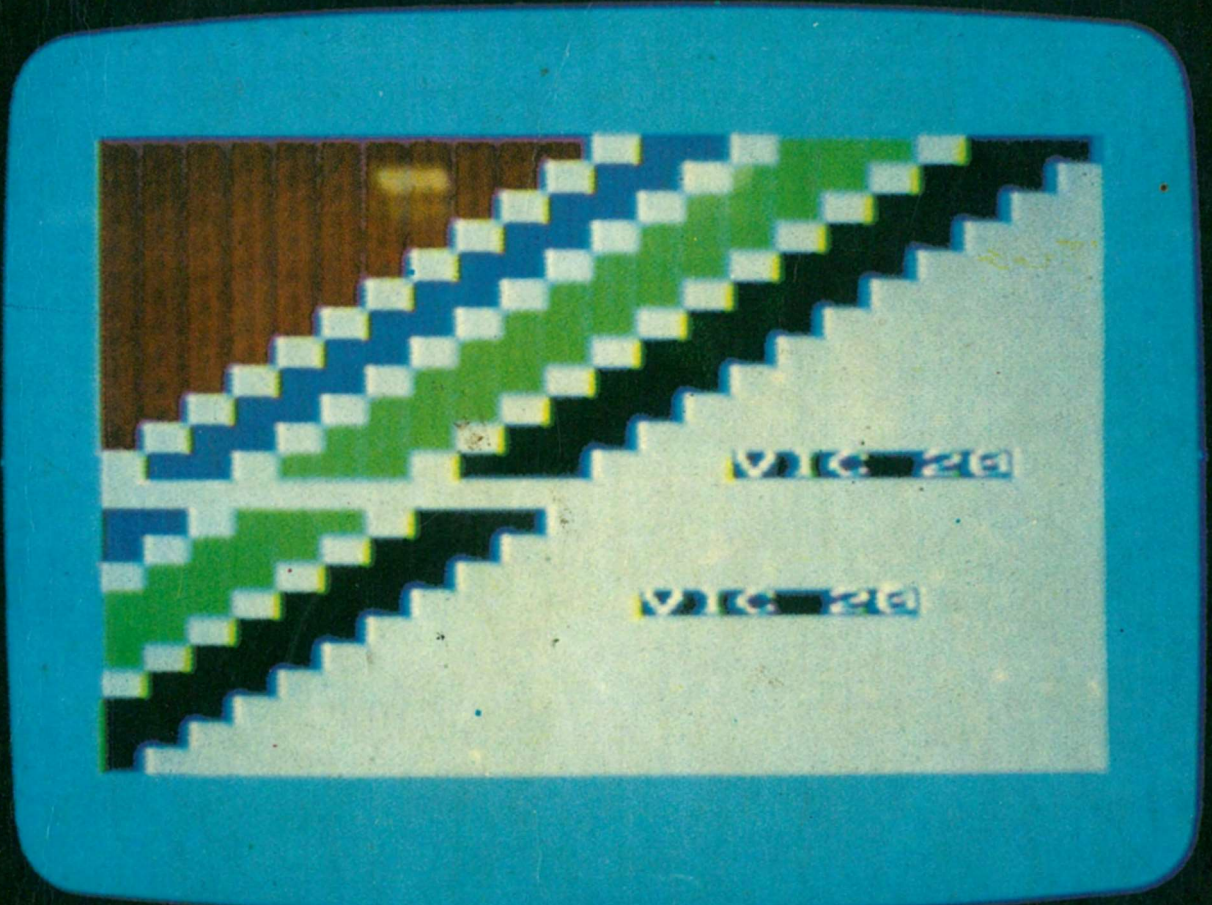


TABELLA PAROLE RISERVATE

ABS	GET	ON	SPC
AND	GET#	OPEN	SQR
ASC	GOSUB	OR	ST
ATN	GOTO	PEEK	STEP
CHR$	IF	POKE	STOP
CLOSE	INPUT	POS	STR$
CLR	INPUT#	PRINT	SYS
CMD	INT	PRINT#	TAB
CONT	LEFT$	READ	TAN
COS	LEN	REM	THEN
DATA	LET	RESTORE	TI
DEF	LIST	RETURN	TI$
DIM	LOAD	RIGHT$	TO
END	LOG	RND	USR
EXP	MID$	RUN	VAL
FN	NEW	SAVE	VERIFY
FOR	NEXT	SGN	WAIT
FRE	NOT	SIN	

(GOTO puo' essere scritto anche GO TO)

2.7. FACCIAMO DEI CALCOLI

Proviamo ora ad usare il VIC per fare dei calcoli. Scrivete usando il calcolatore in modo immediato:

```
?27*18
```

otterrete:

```
486
```

infatti il calcolatore ha eseguito il prodotto di 27 per 18 ed ha stampato il risultato.

Provate ora:

```
?1324/(54*2)
```

otterrete: 12.2592593

se ora provate:

```
?1324/54*2
```

otterrete: 49.037037

infatti nel primo calcolo, a causa delle parentesi prima e' stato calcolato il prodotto di 54 per 2 e poi e' stato

diviso 1324 per il risultato della moltiplicazione; nel secondo invece prima e' stato moltiplicato 1324 per 54 e poi il risultato e' stato diviso per 2.

Vediamo di dedurre alcune regole:

. le operazioni sono svolte da sinistra a destra, ma hanno la precedenza quelle racchiuse in parentesi. Sono cioe' mantenute le solite regole della matematica;

. l'asterisco si usa per il simbolo di moltiplicazione;

. la barra trasversale a destra si usa per il simbolo di divisione.

Il VIC calcola quindi anche complicate espressioni con l'uso degli operatori aritmetici e delle parentesi. Inoltre si possono usare gli operatori relazionali e gli operatori logici o booleani.

Le espressioni vengono calcolate muovendosi da sinistra verso destra, ma dando alle diverse operazioni una precedenza che chiamiamo priorita'. Vengono eseguite prima le operazioni che hanno la priorita' piu' alta considerando anche la priorita' delle parentesi.

Gli operatori relazionali possono entrare nelle espressioni aritmetiche e alla relazione viene sostituito il valore -1 se essa e' verificata ed il valore 0 se essa non e' verificata. Esempio:

```
?27*(23<6)  da' come risultato 0
```

infatti 23<6 non e' una relazione vera e quindi gli viene sostituito 0.

Invece:

```
?27*(6<23)  da' come risultato -27
```

infatti 6<23 e' una relazione vera e quindi gli viene sostituito -1.

Riepiloghiamo quindi la regola:

. se una espressione relazionale e/o logica e' vera essa vale -1;

. se una espressione relazionale e/o logica e' falsa essa vale 0.

Per quanto riguarda gli operatori logici ne parleremo piu' diffusamente in seguito.

Riportiamo uno schema di tutti gli operatori segnalando la loro priorita' e le modalita' operative.

TIPO	PRIORITA'	OPERATORE	COMMENTI
	9	()	parentesi che racchiudono espressioni
ARITMETICI	8	freccia su	elevamento a potenza
	7	-	segno numeri negativi
	6	*	moltiplicazione
	6	/	divisione
	5	+	addizione
	5	-	sottrazione
RELAZIONALI	4	=	uguale
	4	<>	non uguale
	4	<	minore
	4	>	maggiore
	4	<= o =<	minore o uguale
	4	>= o =>	maggiore o uguale
LOGICI	3	NOT	complemento logico
	2	AND	prodotto logico
	1	OR	somma logica

2.8. ESTRAIAMO DEI NUMERI A CASO

Provate a scrivere:

```
?RND(1);RND(0);RND(-1)
```

vedrete comparire sullo schermo 3 numeri quasi-a-caso. Essi potranno essere scritti o in notazione decimale o in

notazione esponenziale.

La funzione RND fornisce un numero compreso tra 0 e 1. Il sistema usa una formula matematica per calcolare i numeri a caso, da cui il "quasi-a-caso" usato prima. Questa formula genera una sequenza di numeri a caso partendo da un valore iniziale, detto seme (seed). La funzione si scrive RND(X), dove X puo' essere un numero positivo, negativo o nullo. Il numero X influisce sulla sequenza dei numeri generati in questo modo:

. se X=0 la sequenza inizia prendendo come origine il contenuto del contatore dei fotogrammi dello schermo;

. se X>0 la sequenza prosegue e il valore del numero non influisce sul numero generato;

. se X<0 viene preso come origine della sequenza il numero X e quindi per ogni X si ottiene un numero a caso diverso, ma sempre lo stesso per lo stesso X. Cioe' se volete che un vostro programma tratti sempre la stessa sequenza di numeri a caso dovete usare lo stesso numero X negativo con la funzione RND.

Potete fare questa prova: spegnete il VIC, riaccendetelo dopo qualche secondo, scrivete ed eseguite in modo immediato:

```
?RND(5)
```

prendete nota del numero ottenuto. Ripetete di nuovo la prova usando un numero diverso da 5 ma positivo, vedrete che il numero ottenuto e' sempre lo stesso. Questo significa che il calcolatore appena acceso si trova nelle stesse condizioni e non e' ancora stato generato alcun numero a caso, per cui per qualunque X positivo viene lo stesso numero random, dato che la partenza e' la stessa.

Se ripetete questa prova scrivendo invece RND(0) vedrete ogni volta un numero diverso.

Provate ad usare il programma che segue:

```
10 REM NUMERI RANDOM
20 INPUT"ARGOMENTO RND ";X
30 FOR K=1TO 10
40 PRINT RND(X)
50 NEXTK
60 PRINT
70 GOTO 20
```

esso chiede un numero X come argomento per la funzione RND e

modo come appaiono le linee di testo.

Per distinguere il colore del bordo si ha un numero che va da 0 a 7, mentre per distinguere il colore dello sfondo si ha un numero che va da 0 a 15. Nella tabella che segue sono riportati i numeri distintivi dei colori ed i loro significati.

NUMERI DISTINTIVI DEI COLORI

COLORE	NUM. PER SCHERMO S	NUM. PER BORDO B
NERO	0	0
BIANCO	1	1
ROSSO	2	2
AZZURRO	3	3
VIOLA	4	4
VERDE	5	5
BLU	6	6
GIALLO	7	7
ARANCIONE	8	
GIALLO/ARANCIO	9	
ROSA	10	
CELESTE	11	
VIOLA CHIARO	12	
VERDE CHIARO	13	
BLU INTERMEDIO	14	
GIALLO CHIARO	15	

Il numero da scrivere con la POKE nel byte di indirizzo 36879 si puo' calcolare con la seguente formula:

```
K = S * 16 + 8 + B
```

e si ottengono le 128 combinazioni di colore sfondo/bordo con il testo in campo diretto. Se non si aggiunge 8, cioe' si scrive:

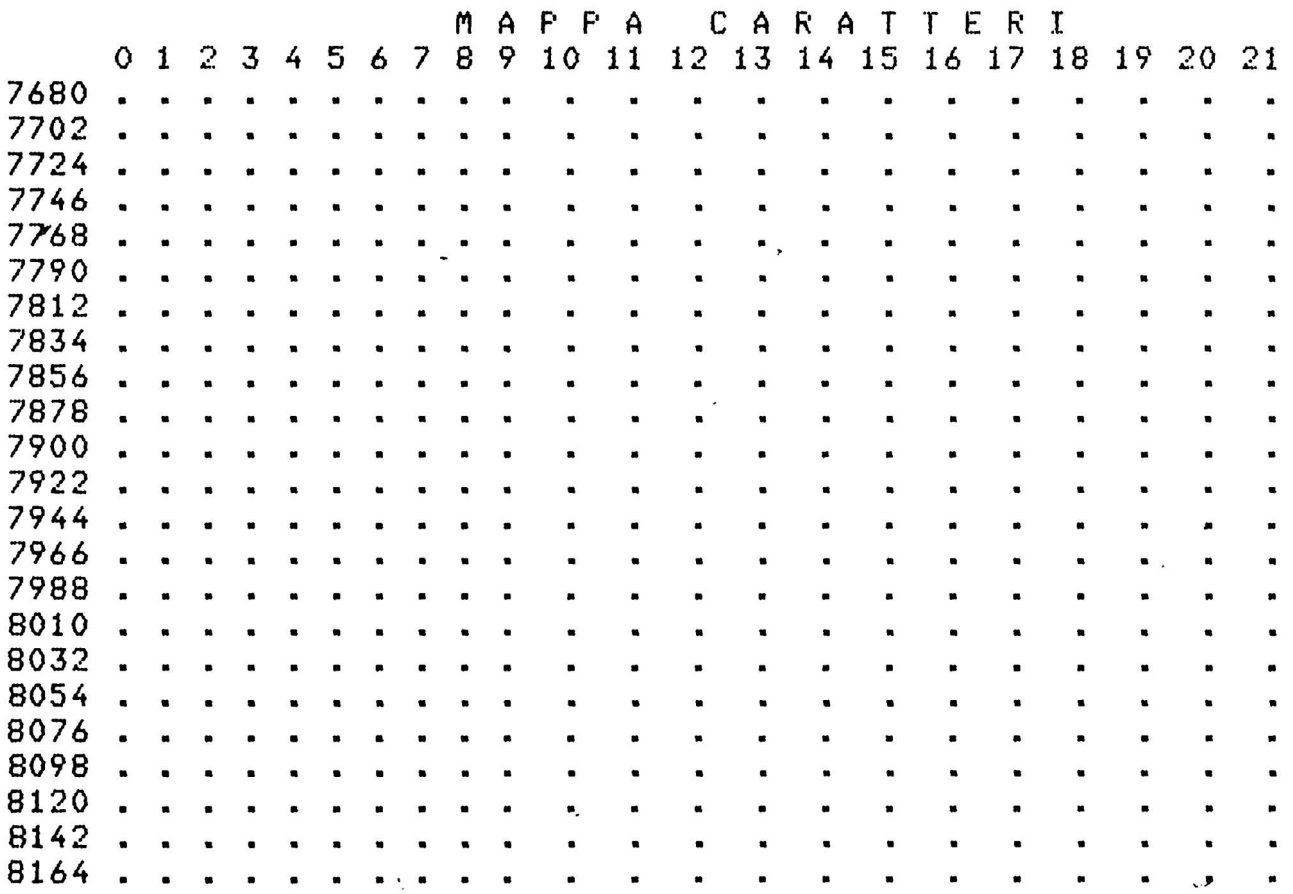
```
K = S * 16 + B
```

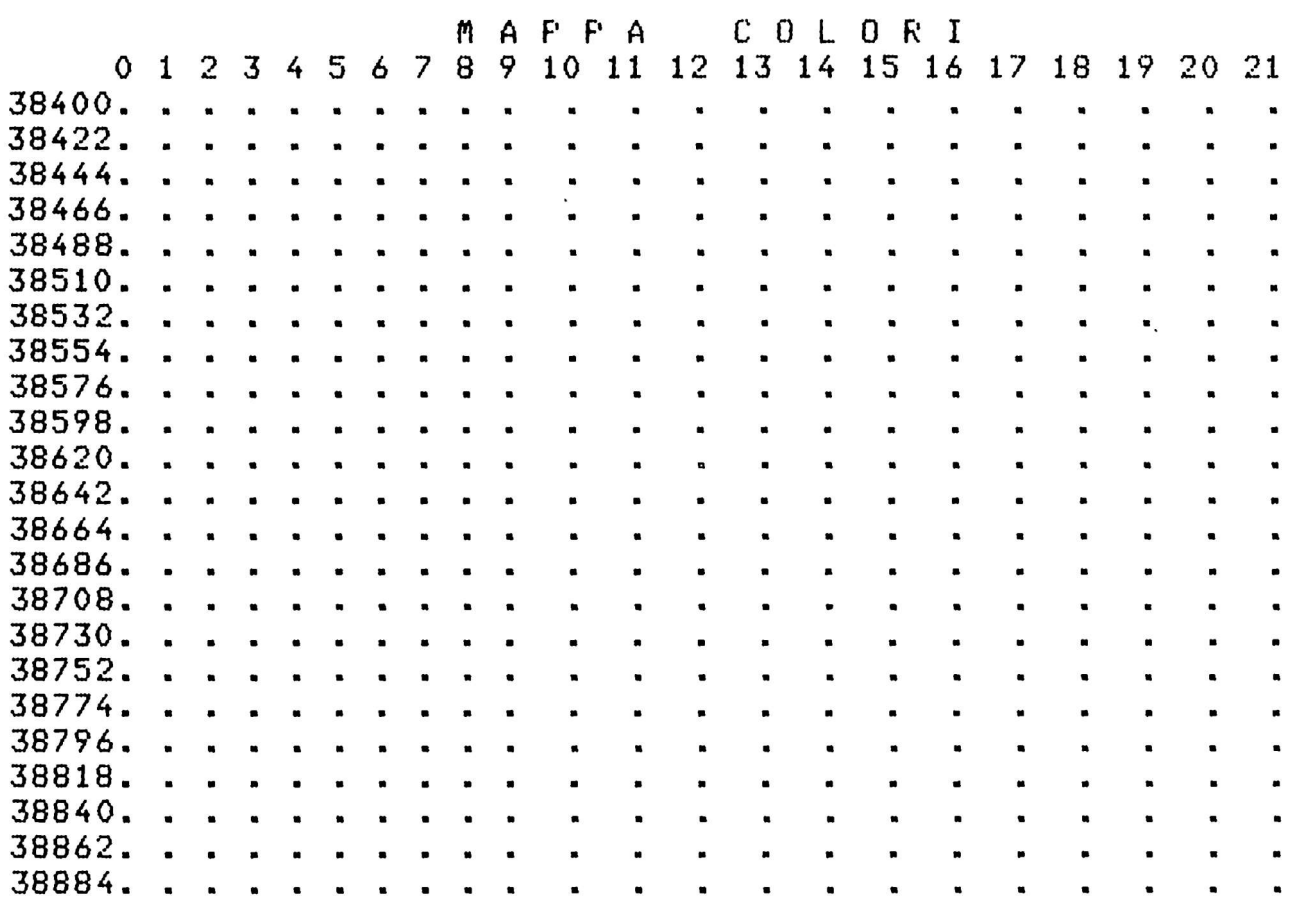
si ottengono le 128 combinazioni di colore sfondo/bordo con il testo in campo inverso.

3.2. IL COLORE DEL TESTO

Nel precedente paragrafo abbiamo imparato a modificare i colori dello sfondo e del bordo. Ora dobbiamo imparare a modificare il colore del testo.

Nella parte in alto a destra della tastiera ci sono i


MAPPA CARATTERI
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21
7680
7702
7724
7746
7768
7790
7812
7834
7856
7878
7900
7922
7944
7966
7988
8010
8032
8054
8076
8098
8120
8142
8164


MAPPA COLORI
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21
38400.
38422.
38444.
38466.
38488.
38510.
38532.
38554.
38576.
38598.
38620.
38642.
38664.
38686.
38708.
38730.
38752.
38774.
38796.
38818.
38840.
38862.
38884.

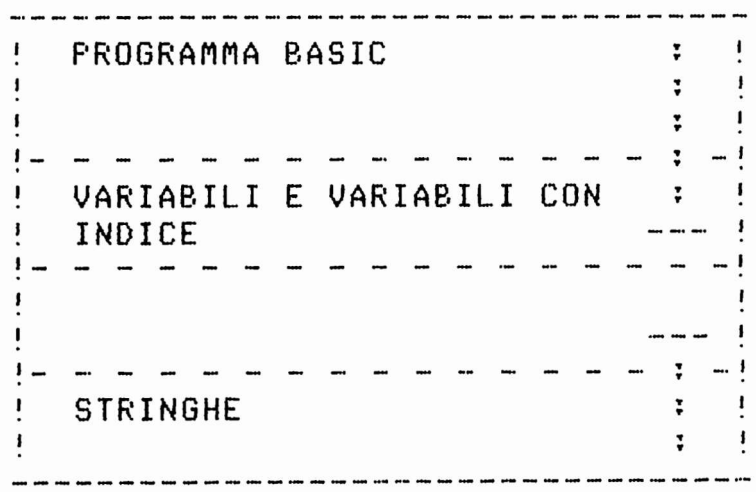
Indirizzi decimali		Indirizzi esadecimali
0	AREA DI LAVORO PER SISTEMA	0000
1023	OPERATIVO E BASIC RAM1K	03FF
1024	ESPANSIONE RAM	0400
4095	RAM3K	0FFF
4096	AREA RAM UTENTE DISPONIBILE	1000
7679	NELLA VERSIONE 5K RAM3.5K	1DFF
7680	MAPPA CARATTERI MEMORIA	1E00
8191	DI SCHERMO RAM0.5K	1FFF
8192	ESPANSIONE RAM/ROM	2000
16383	RAM/ROM 8K	3FFF
16384	ESPANSIONE RAM/ROM	4000
24575	RAM/ROM 8K	5FFF
24576	ESPANSIONE RAM/ROM	6000
32767	RAM/ROM 8K	7FFF
32768	IMMAGINE DEI CARATTERI	8000
36863	4 BLOCCHI DA 1K ROM4K	8FFF
36864	INDIRIZZI DI SISTEMA	9000
	ROM	
37136	INPUT/OUTPUT 0	9110
	RAM	
38400	MAPPA COLORI DEL VIDEO	9600
	RAM 0.5K	
38912	INPUT/OUTPUT 2	9800
	RAM	
39936	INPUT/OUTPUT 3	9C00
	RAM	
40960	ESPANSIONE ROM	A000
	ROM 8K	
49152	INTERPRETE BASIC	C000
	ROM 8K	
57344	SISTEMA OPERATIVO 8K	E000
	ROM	
65535		FFFF

Vediamo ora l'utilizzo dei bytes da 0 a 1023:

Indirizzi decimali		Indirizzi esadecimali
0	AREA DI LAVORO BASIC	0000
144	AREA DI LAVORO SISTEMA OP.	0090
256	AREA STACK BASIC E SIST. OP.	0100
512	AREA LAVORO BASIC E SIST. OP.	0200
828	BUFFER CASSETTA	033C
1023		0400

L'area stack e' usata con indirizzi decrescenti.

Vediamo ora l'utilizzo dei 3.5K utente (4096-7679):



il programma inizia al byte 4096 per indirizzi di bytes crescenti; subito sotto il programma vengono memorizzate le variabili semplici e poi quelle con indice, per le stringhe, sia semplici che con indice, in questa zona stanno solo i nomi, le dimensioni e i puntatori; il corpo delle stringhe sta nell'area a indirizzi piu' alti e viene memorizzato per indirizzi decrescenti.

Quando le stringhe crescono troppo si puo' avere il messaggio OUT OF MEMORY.

Vediamo l'utilizzo dei 4K per l'immagine dei caratteri (32868-36863):

. da 32768 a 33791 si hanno 1024 byte per i 128 possibili

caratteri del set grafico da far apparire sullo schermo (confrontate con Appendice B). Ogni carattere e' rappresentato in un blocco di 8 byte, cioe' si hanno a disposizione 8*8=64 bit per rappresentare un carattere. Si dice che il carattere e' rappresentato a punti in una matrice 8x8. Se moltiplichiamo 128x8 otteniamo 1024. Vediamo come e' rappresentato il carattere A:

```
0 0 0 1 1 0 0 0      e togliendo          1 1
0 0 1 0 0 1 0 0      gli zeri per      1      1
0 1 0 0 0 0 1 0      vedere meglio  1              1
0 1 1 1 1 1 1 0                     1 1 1 1 1 1 1
0 1 0 0 0 0 1 0                     1              1
0 1 0 0 0 0 1 0                     1              1
0 0 0 0 0 0 0 0
```

Il carattere A che corrisponde al codice 1 per la memoria di schermo si trova all'indirizzo:

32768 + 8 * codice e quindi 32768 + 8 * 1 = 32776

se andate a scrivere PRINT PEEK(32776) vedrete il numero 24 che in binario e': 00011000, se scrivete PRINT PEEK(32777) vedrete 36 che in binario e': 00100100, e cosi' via. Quindi nella mappa della memoria di schermo si trovano i codici dei caratteri, ma essi sono usati dal sistema come puntatori per andare a prelevare, dopo averli moltiplicati per 8 ed avere aggiunto l'indirizzo base, che in questo caso e' 32768, il disegno per punti del carattere. Questo disegno per punti e' quello che appare sul video. Il video ha quindi una risoluzione per punti di (22x8)x(23x8), cioe' di 176x184.

. da 32792 a 34815 si trovano i caratteri del set grafico in campo inverso. Se provate ad aggiungere 1024 all'indirizzo usato prima per vedere l'immagine di A ed usate ancora il comando PEEK vedrete una immagine nella quale gli zeri sono stati scambiati con gli uno.

. da 34816 a 35839 si trovano i caratteri del set maiuscolo/minuscolo.

. da 35840 a 36863 si trovano i caratteri del set maiuscolo/minuscolo in campo inverso.

Nella prima parte della memoria sono localizzati i puntatori per mezzo dei quali si puo' vedere dove si trovano le diverse zone in cui e' divisa la memoria utente; passiamo ad elencarli:

Indirizzi decimali byte basso	alto	Contenuto
43	44	indirizzo inizio programma
45	46	indirizzo inizio variabili
47	48	indirizzo inizio variabili con indice
49	50	indirizzo fine variabili con indice
51	52	indirizzo inizio stringhe (la prima caricata)
53	54	indirizzo fine stringhe
55	56	indirizzo fine memoria
65	66	indirizzo DATA

Di norma 43 e 44 contengono l'indirizzo 4096, dove inizia il programma.

Per calcolare gli indirizzi dovete procedere cosi':

```
PRINT(PEEK(44))*256 + PEEK(43)
```

e similmente per gli altri.

6.6. MEMORIZZAZIONE DELLE ISTRUZIONI E DEI DATI

Le linee di programma Basic sono memorizzate una dopo l'altra con il seguente formato:

. due bytes per il LINK, cioe' per puntare alla prossima linea, se i 2 bytes di LINK sono zero il programma termina; per calcolare l'indirizzo dove e' memorizzata la prossima linea si deve calcolare secondo byte moltiplicato 256 + primo byte;

. due byte per il numero di linea: il primo byte da' la parte meno significativa del numero linea ed il secondo la piu' significativa;

. segue la linea Basic e vengono usati i codici dell'Appendice J per le parole chiave ed i simboli ed i codici ASCII per i dati del programmatore;

. la linea termina con un byte a zero binario (tutti bit zero).

Se la linea di programma contiene spazi questi vengono conservati, con spreco di memoria. Non viene conservato lo spazio tra il numero di linea e il primo carattere della frase. In fase di LIST questo spazio viene aggiunto.

CARATTERI STANDARD	CARATTERI ALTERNATIVI	CODICE ASCII
		0
		1
		2
		3
		4
WHT	WHT	5
		6
		7
		8
		9
		10
		11
		12
RETURN	RETURN	13
SWITCH TO LOWER CASE	SWITCH TO LOWER CASE	14
		15
		16
CRSR ⇓	CRSR ⇓	17
RVS ON	RVS ON	18
CLR HOME	CLR HOME	19
INST DEL	INST DEL	20
		21
		22
		23

CARATTERI STANDARD	CARATTERI ALTERNATIVI	CODICE ASCII
		24
		25
		26
		27
RED	RED	28
CRSR ⇒	CRSR ⇒	29
GRN	GRN	30
BLU	BLU	31
SPACE	SPACE	32
!	!	33
"	"	34
#	#	35
$	$	36
%	%	37
&	&	38
'	'	39
(	(	40
)	)	41
*	*	42
+	+	43
,	,	44
-	-	45
		46
/	/	47

CARATTERI STANDARD	CARATTERI ALTERNATIVI	CODICE ASCII
0	0	48
1	1	49
2	2	50
3	3	51
4	4	52
5	5	53
6	6	54
7	7	55
8	8	56
9	9	57
:	:	58
;	;	59
<	<	60
=	=	61
>	>	62
?	?	63
@	@	64
A	a	65
B	b	66
C	c	67
D	d	68
E	e	69
F	f	70
G	g	71

CARATTERI STANDARD	CARATTERI ALTERNATIVI	CODICE ASCII
H	h	72
I	i	73
J	j	74
K	k	75
L	l	76
M	m	77
N	n	78
O	o	79
P	p	80
Q	q	81
R	r	82
S	s	83
T	t	84
U	u	85
V	v	86
W	w	87
X	x	88
Y	y	89
Z	z	90
[	[	91
£	£	92
]	]	93
↑	↑	94
←	←	95

CARATTERI STANDARD	CARATTERI ALTERNATIVI	CODICE ASCII
		96
	A	97
	B	98
	C	99
	D	100
	E	101
	F	102
	G	103
	H	104
	I	105
	J	106
	K	107
	L	108
	M	109
	N	110
	O	111
	P	112
	Q	113
	R	114
	S	115
	T	116
	U	117
	V	118
	W	119

CARATTERI STANDARD	CARATTERI ALTERNATIVI	CODICE ASCII
	X	120
	Y	121
	Z	122
		123
		124
		125
π		126
		127
		128
		129
		130
		131
		132
f1	f1	133
f3	f3	134
f5	f5	135
f7	f7	136
f2	f2	137
f4	f4	138
f6	f6	139
f8	f8	140
SHIFT RETURN	SHIFT RETURN	141
SWITCH TO UPPER CASE	SWITCH TO UPPER CASE	142
		143

CARATTERI STANDARD	CARATTERI ALTERNATIVI	CODICE ASCII
BLK	BLK	144
↑ CRSR	↑ CRSR	145
RVS OFF	RVS OFF	146
CLR HOME	CLR HOME	147
INST DEL	INST DEL	148
		149
		150
		151
		152
		153
		154
		155
PUR	PUR	156
← CRSR	← CRSR	157
YEL	YEL	158
CYN	CYN	159
SPACE	SPACE	160
		161
		162
		163
		164
		165
		166
		167

CARATTERI STANDARD	CARATTERI ALTERNATIVI	CODICE ASCII
		168
		169
		170
		171
		172
		173
		174
		175
		176
		177
		178
		179
		180
		181
		182
		183
		184
		185
		186
		187
		188
		189
		190
		191

CARATTERI STANDARD	CARATTERI ALTERNATIVI	CODICE ASCII
		192
	A	193
	B	194
	C	195
	D	196
	E	197
	F	198
	G	199
	H	200
	I	201
	J	202
	K	203
	L	204
	M	205
	N	206
	O	207
	P	208
	Q	209
	R	210
	S	211
	T	212
	U	213
	V	214
	W	215

CARATTERI STANDARD	CARATTERI ALTERNATIVI	CODICE ASCII
	X	216
	Y	217
	Z	218
		219
		220
		221
		222
		223
		224
		225
		226
		227
		228
		229
		230
		231
		232
		233
		234
		235
		236
		237
		238
		239

CARATTERI STANDARD	CARATTERI ALTERNATIVI	CODICE ASCII
		240
		241
		242
		243
		244
		245
		246
		247
		248
		249
		250
		251
		252
		253
		254
π		255

APPENDICE B

IL CODICE PER LA MEMORIA DI SCHERMO

In questa tabella sono riportati i codici numerici che si trovano nella mappa dei caratteri della memoria di schermo. La mappa dei caratteri e' formata da 506 bytes di indirizzo da 7680 a 8185.

Se si legge con la funzione PEEK il contenuto dei bytes della mappa dei caratteri si ritrovano questi valori numerici. Se si vuole adoperare il comando POKE per scrivere direttamente nella mappa dei caratteri si devono adoperare questi codici e non quelli ASCII della Appendice A.

Nella tabella sono riportati i codici appartenenti ai due set di caratteri. Tali codici vanno da 0 a 127. Se si aggiunge 128 ad un codice si ottiene lo stesso carattere in campo inverso. I caratteri stampabili sono quindi 128. Se osservate bene i due set di caratteri vi accorgerete che il set alternativo ha meno caratteri grafici, dato che ha al posto di un gruppo di caratteri grafici le lettere maiuscole, pero' ha 4 caratteri grafici che non sono compresi nel primo set, quelli che corrispondono ai codici 94,95,105 e 122.

I due set di caratteri non sono mescolabili, si puo' usare l'uno o l'altro. Nella Appendice A si elencano tutti i modi per selezionare il set di caratteri desiderato.

La relazione matematica che lega i codici dell'appendice B con i codici ASCII e' la seguente, chiamando A il codice ASCII e D il codice della memoria di schermo (Display Code):

- se 32<=A<= 63 allora 32<=D<= 63 quindi D=A
- se 64<=A<= 95 allora 0<=D<= 31 quindi D=A-64
- se 96<=A<=127 allora 64<=D<= 95 quindi D=A-32
- se 160<=A<=191 allora 96<=D<=127 quindi D=A-64.

Segue la tabella dei codici per i caratteri della memoria di schermo.

CARATTERI STANDARD	CARATTERI ALTERNATIVI	CODICE MEMORIA DI SCHERMO	
		NORMALE	CAMPO INVERSO
@	@	0	128
A	a	1	129
B	b	2	130
C	c	3	131
D	d	4	132
E	e	5	133
F	f	6	134
G	g	7	135
H	h	8	136
I	i	9	137
J	j	10	138
K	k	11	139
L	l	12	140
M	m	13	141
N	n	14	142
O	o	15	143
P	p	16	144
Q	q	17	145
R	r	18	146
S	s	19	147
T	t	20	148
U	u	21	149
V	v	22	150
W	w	23	151

CARATTERI STANDARD	CARATTERI ALTERNATIVI	CODICE MEMORIA DI SCHERMO	
		NORMALE	CAMPO INVERSO
X	x	24	152
Y	y	25	153
Z	z	26	154
[	[	27	155
£	£	28	156
]	]	29	157
↑	↑	30	158
←	←	31	159
SPACE	SPACE	32	160
!	!	33	161
"	"	34	162
#	#	35	163
$	$	36	164
%	%	37	165
&	&	38	166
'	'	39	167
(	(	40	168
)	)	41	169
*	*	42	170
+	+	43	171
,	,	44	172
-	-	45	173
	.	46	174
/	/	47	175

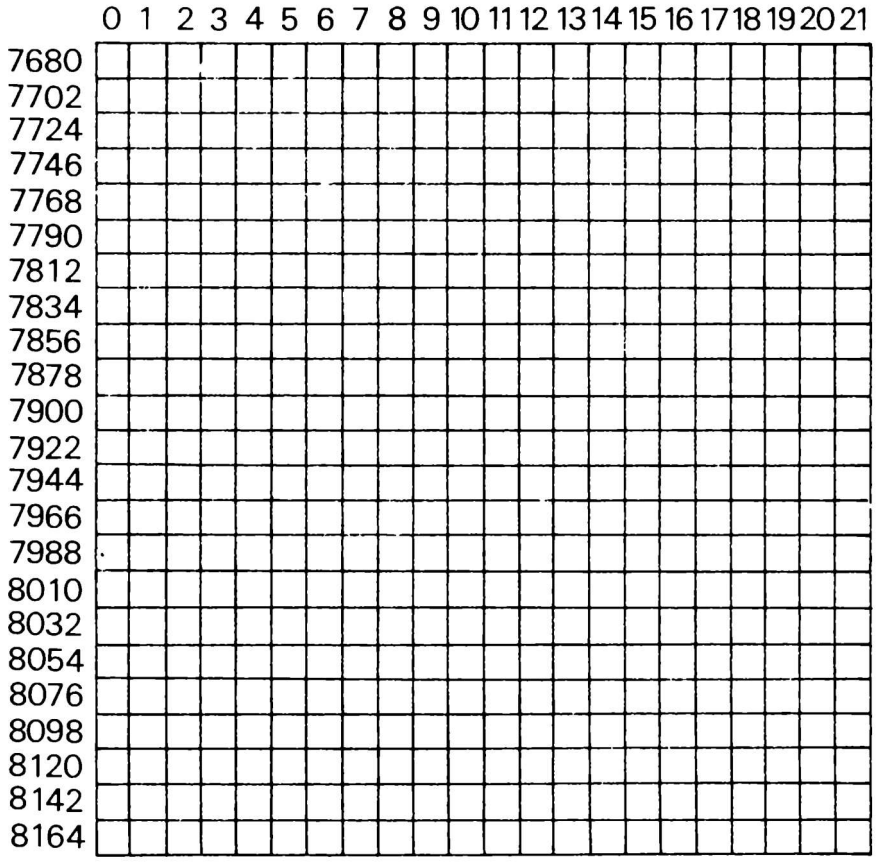
CARATTERI STANDARD	CARATTERI ALTERNATIVI	CODICE MEMORIA DI SCHERMO	
		NORMALE	CAMPO INVERSO
0	0	48	176
1	1	49	177
2	2	50	178
3	3	51	179
4	4	52	180
5	5	53	181
6	6	54	182
7	7	55	183
8	8	56	184
9	9	57	185
		58	186
;	;	59	187
<	<	60	188
=	=	61	189
>	>	62	190
?	?	63	191
		64	192
	A	65	193
	B	66	194
	C	67	195
	D	68	196
	E	69	197
	F	70	198
	G	71	199

CARATTERI STANDARD	CARATTERI ALTERNATIVI	CODICE MEMORIA DI SCHERMO	
		NORMALE	CAMPO INVERSO
	H	72	200
	I	73	201
	J	74	202
	K	75	203
	L	76	204
	M	77	205
	N	78	206
	O	79	207
	P	80	208
	Q	81	209
	R	82	210
	S	83	211
	T	84	212
	U	85	213
	V	86	214
	W	87	215
	X	88	216
	Y	89	217
	Z	90	218
		91	219
		92	220
		93	221
		94	222
		95	223

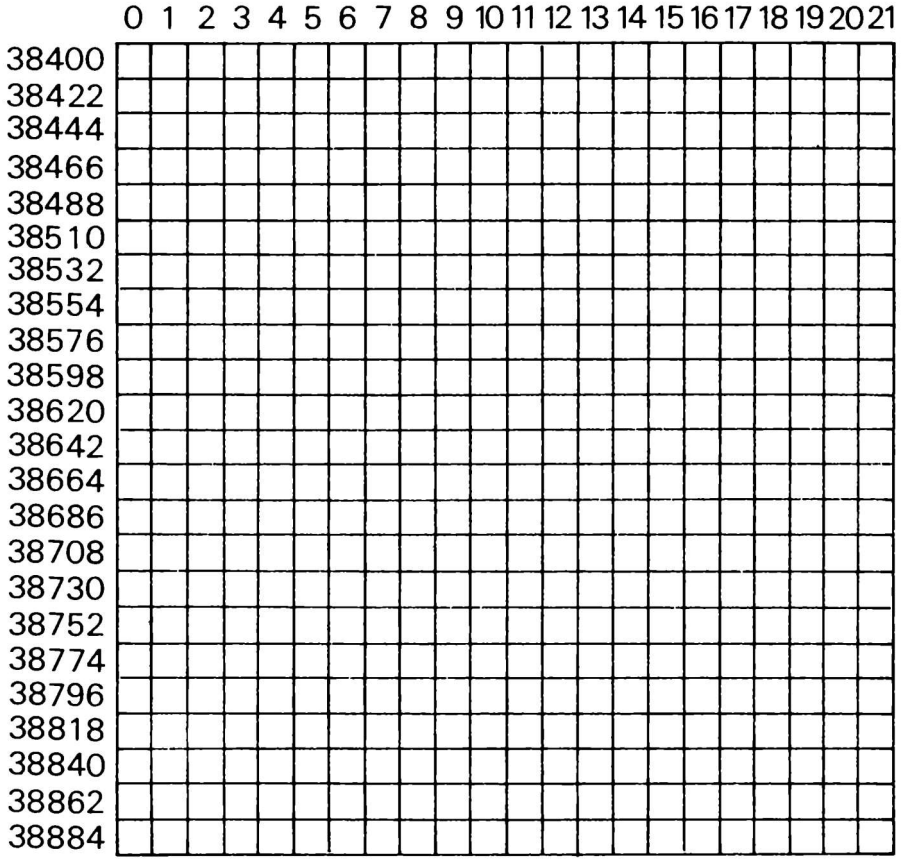
CARATTERI STANDARD	CARATTERI ALTERNATIVI	CODICE MEMORIA DI SCHERMO	
		NORMALE	CAMPO INVERSO
SPACE	SPACE	96	224
		97	225
		98	226
		99	227
		100	228
		101	229
		102	230
		103	231
		104	232
		105	233
		106	234
		107	235
		108	236
		109	237
		110	238
		111	239
		112	240
		113	241
		114	242
		115	243
		116	244
		117	245
		118	246
		119	247

CARATTERI STANDARD	CARATTERI ALTERNATIVI	CODICE MEMORIA DI SCHERMO	
		NORMALE	CAMPO INVERSO
		120	248
		121	249
		122	250
		123	251
		124	252
		125	253
		126	254
		127	255

MAPPA DEI CARATTERI



MAPPA DEI COLORI



APPENDICE D

I MESSAGGI DI ERRORE

Il sistema segnala gli errori con dei messaggi, segue l'elenco di questi con le spiegazioni sul loro significato.

. BAD DATA - Il programma era in attesa di ricevere un input numerico da una periferica e sono arrivati caratteri invalidi.

. BAD SUBSCRIPT - Nel programma e' stata definita una variabile con indice con un definito numero di dimensioni e con definite dimensioni e una istruzione si riferisce ad una variabile con lo stesso nome, ma non corrispondente alla definizione. Esempio:

```
DIM X(2,3)          variabile X con 2 dimensioni, avente 3
                    righe e 4 colonne

X(1,1,1)=8          riferimento ad un elemento X avente 3
                    dimensioni invece di due
?BAD SUBSCRIPT ERROR  messaggio di errore

X(5,3)=8            riferimento ad un elemento X avente 2
                    dimensioni, ma la prima fuori dalla
                    definizione
?BAD SUBSCRIPT ERROR messaggio di errore
```

. CAN'T CONTINUE - e' stato dato un comando CONT, ma il programma non puo' proseguire per una delle seguenti ragioni:

1) non esiste un programma;
2) si e' appena scritta una nuova linea di programma;
3) il programma non ha ricevuto il comando di RUN;
4) e' appena stato segnalato un errore.

. DEVICE NOT PRESENT - la periferica richiesta per una operazione del tipo: OPEN, CLOSE, CMD, INPUT#, GET#, PRINT#, non e' presente.

. DIVISION BY ZERO - si e' tentato di dividere per zero e questo non e' consentito.

. EXTRA IGNORED - si sono scritti piu' dati di quanti richiesti in risposta ad un comando INPUT. Vengono accettati tanti dati quante sono le variabili presenti nel comando e

gli altri vanno persi, ma il programma continua.

. FILE NOT FOUND - non si trova il file sulla periferica: per il nastro si incontra una segnalazione di END OF TAPE dopo aver fatto scorrere il nastro, per il disco non si trova il nome del file nell'indice.

. FILE NOT OPEN - si e' usato un comando come: CLOSE, CMD, INPUT#, GET#, PRINT# per un file che non e' stato aperto.

. FILE OPEN - si e' tentato di aprire un file che e' gia' stato aperto.

. FORMULA TOO COMPLEX - si e' usata una formula di calcolo troppo complicata ed il sistema non riesce a calcolarla. Si deve spezzare il calcolo in due parti e calcolarle separatamente.

. ILLEGAL DIRECT - si e' tentato di usare in modo immediato o il comando INPUT o il comando GET o il comando DEF di definizione di una funzione e questo non e' consentito.

. ILLEGAL QUANTITY - si e' tentato di usare una funzione con un argomento fuori dai limiti consentiti. Questo puo' avvenire nei seguenti casi:

1) gli indici di una variabile con indice sono fuori dall'intervallo 0-32767, per esempio si e' usato un numero negativo;

2) per la funzione LOG si e' usato un argomento negativo o nullo;

3) per la funzione SQR si e' usato un argomento negativo;

4) A elevato B, dove A < 0 e B non intero;

5) si usa il comando USR, ma il programma in linguaggio macchina non e' presente in memoria;

6) si usano le funzioni di stringa MID$, LEFT$, RIGHT$ con parametri fuori dall'intervallo 1-255;

7) si usa il comando ON...GOTO con indici non validi;

8) si usano i comandi: PEEK, POKE, WAIT, SYS con parametri fuori dall'intervallo 0-65535;

8) si usano i comandi: WAIT, POKE, TAB, SPC con parametri fuori dall'intervallo 0-255.

. LOAD - si sta caricando un programma da nastro e si sono trovati piu' di 31 errori nel primo blocco, oppure ci sono errori in ambedue i blocchi.

. NEXT WITHOUT FOR - si puo' avere un errore nel concatenamento dei FOR/NEXT, oppure la variabile citata in NEXT non corrisponde a quella del FOR.

. NOT INPUT FILE - se un file e' stato aperto per scrivere non si puo' fare una operazione di INPUT.

. NOT OUTPUT FILE - si tenta di scrivere su un file che e' stato aperto per INPUT, oppure che e' solo di INPUT come la tastiera.

. OUT OF DATA - si tenta di leggere dall'interno del programma con un READ piu'.dati di quanti ne sono stati forniti con le frasi DATA; a volte si ha questo errore premendo il RETURN dopo il READY. del video, che viene interpretato come READ Y.

. OUT OF MEMORY - puo' essere determinato dalle seguenti cause:

1) si sta scrivendo un programma troppo lungo;

2) si sta facendo girare un programma e le variabili che si creano occupano troppa memoria; le stringhe occupano spazio di memoria che varia in dipendenza della loro lunghezza;

3) nel programma il numero di FOR/NEXT o di GOSUB/RETURN supera la capacita' dell'area stack.

Per decidere se dipende dall'ultima causa 3), basta scrivere:

?FRE(0) e vedere se la risposta e' un numero diverso da zero.

. OVERFLOW - si sono eseguiti dei calcoli che hanno dato risultati troppo grandi rispetto alle possibilita' del calcolatore.

. REDIM'D ARRAY - le variabili con indice possono essere dimensionate una volta sola in un programma.

. REDO FROM START - si e' risposto ad una richiesta di INPUT numerico con caratteri non validi. Dopo il messaggio compare il punto interrogativo come nuova richiesta di INPUT ed il programma prosegue se si risponde in modo corretto.

. RETURN WITHOUT GOSUB - si e' incontrato un RETURN, ma prima non c'e' stato un GOSUB.

. STRING TOO LONG - si e' cercato di concatenare delle stringhe e la stringa risultante sarebbe piu' lunga di 255 caratteri.

. SYNTAX - si e' scritto un comando con qualche errore ed il VIC non lo riconosce.

. TYPE MISMATCH - si e' usato un tipo di dato errato, numero al posto di stringa o viceversa.

. UNDEF'D FUNCTION - si fa riferimento ad una funzione che non e' stata definita precedentemente.

. UNDEF'D STATEMENT - si e' usato un comando come: GOSUB, GOTO, THEN con riferimento ad un numero di linea inesistente.

. VERIFY - ci sono delle differenze nella comparazione tra il contenuto della memoria ed un file su disco o nastro.

Quando si ha un messaggio di errore il programma si ferma, le variabili mantengono il loro valore e possono essere analizzate con dei comandi in modo immediato, i riferimenti di programma nella stack area vengono persi (GOSUB e FOR) e quindi non si puo' proseguire nell'esecuzione. Si puo' pero ripartire con un GOSUB ad una linea opportuna, ed in tale modo non si perdono i valori delle variabili gia' calcolate. Oppure si puo' ripartire con RUN.

Gli unici errori che non fermano il programma sono:

REDO FROM START
EXTRA IGNORED.

Tra i messaggi riportati i seguenti sono errori segnalati dal Sistema Operativo:

BAD DATA
DEVICE NOT PRESENT
FILE NOT FOUND
FILE NOT OPEN
FILE OPEN
LOAD
NOT INPUT FILE
NOT OUTPUT FILE
VERIFY

tutti gli altri sono errori segnalati dall'interprete BASIC.

GLI OPERATORI

Si hanno 4 tipi di operatori:

- aritmetici;
- relazionali;
- logici;
- funzionali.

Nelle espressioni possono essere usati tutti i tipi di operatori, se gli operandi sono numerici, solo alcuni se gli operandi sono stringhe. In una espressione aritmetica si possono usare tutti gli operatori che producono risultati numerici. Per le stringhe si possono usare gli operatori relazionali e logici e l'operatore aritmetico + che serve a concatenare tra loro due stringhe, cioe' a scriverle una vicino all'altra. La stringa risultato di una operazione di concatenamento non puo' superare i 255 caratteri.

Nel linguaggio BASIC esistono delle funzioni gia' intrinsicamente definite e quando esse vengono usate in espressioni danno luogo ad un valore corrispondente; inoltre l'utente puo' definire delle funzioni con la frase DEF FN. Se in una espressione e' presente una funzione essa viene calcolata con precedenza su tutto il resto.

Le espressioni vengono calcolate da sinistra a destra dando la precedenza alle operazioni racchiuse in parentesi e rispettando le priorita' degli operatori. Le operazioni con priorita' maggiore vengono eseguite per prime.

Si riporta la tabella degli operatori, omettendo le funzioni:

TIPO	PRIORITA'	OPERATORE	COMMENTI
	9	()	parentesi che racchiudono espressioni
ARITMETICI	8	freccia su	elevamento a potenza
	7	-	segno numeri negativi
	6	*	moltiplicazione
	6	/	divisione
	5	+	addizione
	5	-	sottrazione

TIPO	PRIORITA'	OPERATORE	COMMENTI
RELAZIONALI	4	=	uguale
	4	<>	diverso
	4	<	minore
	4		maggiore
	4	<=	minore o uguale
	4	>=	maggiore o uguale
LOGICI	3	NOT	complemento logico (negazione)
	2	AND	prodotto logico
	2	OR	somma logica

Gli operatori relazionali e gli operatori logici servono a formare delle espressioni il cui valore e' una variabile, diciamo di tipo logico, cioe' un VERO o un FALSO; questa variabile assume valore aritmetico -1 se vale VERO e valore aritmetico 0 se vale FALSO.

Gli operatori logici servono a collegare due condizioni, le tabelle della verita' per essi sono:

AND

A	B	RISULTATO
vero	vero	vero
vero	falso	falso
falso	vero	falso
falso	falso	falso

OR

A	B	RISULTATO
vero	vero	vero
vero	falso	vero
falso	vero	vero
falso	falso	falso

NOT

A	RISULTATO
vero	falso
falso	vero

Nel confronto tra stringhe valgono per i caratteri alfabetici le normali regole dell'ordinamento alfabetico,

per gli altri caratteri l'ordinamento dipende dal valore dei corrispondenti codici ASCII.

I COMANDI

Si espongono qui i comandi del BASIC, che di norma vengono usati in modo immediato, per ognuno di essi si descrivera' l'effetto se usati all'interno di un programma. Tutti questi comandi lavorano quando si preme RETURN.

CONT serve per far proseguire un programma che si e' fermato. Esso non ha effetto se :

1) non esiste un programma;
2) si e' scritta una nuova linea di programma;
3) non si e' dato RUN;
4) c'e' stato un messaggio di errore.

Non ha senso usarlo in un programma.

LIST serve per listare sul video il programma che e'in memoria. Se usato da solo produce la lista di tutto il programma con eventuale scrolling. Lo scrolling puo' essere rallentato premendo CTRL o fermato premendo RUN STOP. Si puo' usare anche nei seguenti modi:

LIST n-	lista il programma dalla linea n in poi;
LIST n	lista solo la linea n;
LIST -n	lista dall'inizio fino alla linea n;
LIST n-m	lista dalla linea n alla linea m.

Se si usa in un programma, produce la lista e termina l'esecuzione del programma stesso.

LOAD serve per caricare in memoria un programma dal nastro o dal disco. Se scrivete solo LOAD il sistema cerca il primo programma che trova sul nastro e lo carica in memoria. Se scrivete LOAD "nome-programma" il sistema cerca sul nastro il programma avente quel nome e lo carica in memoria. Se scrivete LOAD "nome-programma",8 il sistema cerca sul disco il programma con quel nome e lo carica. Se non esiste sul dispositivo un programma con quel nome viene segnalato che non e' stato trovato. Il dispositivo cassetta corrisponde al numero di periferica #1, mentre il disco corrisponde al numero #8. In mancanza di virgola e numero dopo il nome del programma viene assunto 1 per il dispositivo.

Se il comando LOAD viene usato in un programma si ha il caricamento del nuovo programma al posto del vecchio e se questo non e' piu' lungo del precedente non vengono toccate le variabili che si trovano in memoria. Inoltre il nuovo programma va in esecuzione appena caricato. Per questa ragione si usa il LOAD da programma per ottenere il concatenamento tra programmi. E' quindi possibile segmentare

programmi lunghi per farli entrare in memoria.

NEW serve per cancellare il contenuto della memoria. E' buona norma usare questo comando prima di cominciare a scrivere un nuovo programma in memoria. Se usato da programma lascia la memoria pulita. Si deve fare attenzione e non usare NEW fuori tempo, si rischia di distruggere un lungo lavoro di scrittura di un programma in memoria.

RUN serve per mandare in esecuzione un programma che si trova in memoria. Prima dell'inizio del programma vengono azzerate le variabili. Se si scrive solo RUN il programma parte dalla linea con numero minore. SE si scrive RUN n, il programma parte dalla linea n, ma si ha lo stesso l'azzeramento delle variabili. Se si vuole far partire da una linea n un programma senza toccare le variabili, si deve usare il comando GOTO n. Non ha senso usare questo comando in un programma.

SAVE serve per memorizzare un programma che si trova in memoria o sulla cassetta o sul disco. Se si scrive solo SAVE il programma viene memorizzato sul nastro senza nome. Dovete fare attenzione e tenere la cassetta inattiva in modo che sia il sistema a chiedervi di avviarla, altrimenti rischiate di non aver posizionato bene un nastro e di cancellare altre registrazioni interessanti. Se scrivete SAVE seguito dal nome di un programma, viene registrato sul nastro il programma con il nome e questo e' sempre consigliabile. In tale modo poi le letture dei programmi si ottengono fornendo il nome del programma. Il comado puo' essere completato cosi':

```
SAVE "nome-programma",1,0
SAVE "nome-programma"          questi due comandi sono e=
                               quivalenti e memorizzano
                               su cassetta;
SAVE "nome-programma",1,1      dopo aver memorizzato sulla
                               cassetta scrive il carat=
                               tere di FINE NASTRO;
SAVE "nome-programma",8        memorizza su disco .
```

Non ha senso dare questo comando da programma.

VERIFY serve per verificare quanto appena memorizzato comparandolo con il contenuto della memoria. Il comando va completato con il nome del programma e con il numero della periferica, a meno che non si tratti del nastro, nel qual caso si puo' tralasciare il numero e se si vuole anche il nome. Il comando VERIFY puo' essere usato con un nome non esistente per far scorrere il nastro e farlo posizionare dopo l'ultimo programma registrato. Non ha senso dare questo comando da programma.

LE ISTRUZIONI

Si espongono le istruzioni del BASIC. Tutte le istruzioni salvo,: INPUT, INPUT#, GET, GET# e DEFFN, possono essere usate sia in modo immediato che differito. Si ricorda che una linea di programma inizia con il numero di linea e puo' contenere una i struzione o piu' istruzioni e in quest'ultimo caso le istruzioni devono essere separate dai due punti.

CLOSE questo comando si scrive facendo seguire la parola chiave da un numero: CLOSE n. Dove n e' il numero assegnato logicamente al file nella OPEN iniziale. Per una corretta gestione dei files e' necessario chiudere tutti i file aperti prima di terminare un programma.

CLR si scrive senza parametri e serve per azzerare tutte le variabili di un programma, comprese le aree di lavoro come la stack area. Quando si fa RUN di un programma si ottiene automaticamente un CLR.

CMD con questo comando viene trasferito quanto normalmente esce sul video alla periferica appena aperta con lo stesso numero logico di file che segue il comando CMD. Esempio:

```
OPEN 1,4    apre la stampante che e' la periferica 4
CMD 1       trasferisce l'uscita al file di numero
            logico 1, appena aperto sulla stampante
            e quindi predispone la scrittura sulla
            stampante
LIST        manda sulla stampante la lista.
```

Per rimandare l'uscita al video si deve scrivere:

```
PRINT#1:CLOSE 1.
```

La struttura della frase e' uguale a quella della PRINT#.

DATA questa parola chiave deve essere seguita da una lista dei dati, separati da virgole, che si vogliono memorizzare nella zona apposita. In tale modo viene creato un magazzino di dati, dove i dati vengono memorizzati uno dopo l'altro con la stessa sequenza nella quale compaiono nei DATA. I dati numerici possono contenere il punto decimale. Le stringhe si possono scrivere senza apici delimitatori; e' necessario usare gli apici delimitatori se una stringa deve contenere i seguenti caratteri: spazio, virgola, due punti. Se nella lista dei dati compaiono due virgole (caratttere separatore dei dati) vicine il sistema assume che in mezzo sia o il carattere 0 o la stringa nulla. Non e' importante dove nel programma siano localizzate le frasi DATA, esse non sono frasi esecutive; e' consigliabile porle tutte insieme in fondo al programma. Esempio:

```
10DATA34,8.7,,PIPPO,6578,"SEGUONO: PRIMO, SECONDO","CASA   "
```

con questa frase DATA vengono memorizzate le seguenti costanti:

```
34                  8.7                  0                  PIPPO
6578                SEGUONO: PRIMO, SECONDO
CASA + 3 spazi
```

Il sistema dopo aver preparato il blocco di dati posiziona un puntatore interno prima del primo dato; ogni volta che con la frase READ si preleva (legge) un dato dal blocco il puntatore viene spostato al dato seguente. Se si leggono piu' dati di quanti disponibili si ha un messaggio di errore. Il puntatore puo' essere riposizionato all'inizio del blocco dati con la frase RESTORE.

DEF FN serve a definire una funzione che puo' essere richiamata ed eseguita in un programma chiamandola per nome. Il nome della funzione deve seguire i caratteri FN ed essere seguito da una coppia di parentesi contenenti l'argomento della funzione, il simbolo = e poi la formula di calcolo; cosi':

```
10 DEF FNA(X)=13*(X-3)/(X+1)
```

Se nel programma si scrive:

```
50 PRINT FNA(9)
```

viene sostituito 9 ad X, effettuato il calcolo e stampato il valore risultante. Se invece si scrive:

```
80 Y = 57 + FNA(B)
```

viene sostituito ad X nella formula di calcolo il valore della variabile B ed Y viene posto al valore della funzione calcolata + 57.

DIM serve per definire le variabili con indice. Alla parola DIM segue la lista di definizione delle variabili, cosi':

```
DIM A1(4,5),B$(6,7,2)
```

Per calcolare il numero di elementi di ogni variabile si deve aggiungere 1 alle dimensioni esposte e moltiplicarle tra loro. Nell'esempio visto sopra A1 e' formato da (4+1)*(5+1)=30 elementi, mentre B$ e' formato da (6+1)*(7+1)*(2+1) elementi. In una frase DIM si possono definire tante variabili quante ne entrano nella linea. Non

esiste un limite teorico al numero delle dimensioni di una variabile con indice, ma tale limite e' dato dalla capacita' di memoria del calcolatore. Se il programma richiama una variabile con indice che non e' stata definita con una DIM, si ha un dimensionamento implicito a 10 (cioe' 11 elementi per ogni dimensione), se poi si pone la frase DIM piu' avanti nel programma si ha errore di ridimensionamento. Per questa ragione e' consigliabile scrivere tutte le frasi DIM all'inizio del programma.

Per quanto riguarda il dimensionamento delle stringhe si deve tener presente che al momento del dimensionamento il sistema crea solo le tabelle di riferimento alle stringhe con lo spazio per il puntatore al contenuto della stringa. Il contenuto della stringa occupa dinamicamente lo spazio riservato alle stringhe in memoria al momento dell'esecuzione del programma.

END fa terminare l'esecuzione del programma, senza messaggio di segnalazione. Il programma puo' continuare se si scrive CONT e se alla frase END seguono altre frasi nel programma.

FOR...TO...STEP serve per controllare l'esecuzione ripetitiva di un pezzo di programma. A questo comando e' legato il comando NEXT che serve per chiudere (delimitare) il ciclo da percorrere. Se si vuole creare un ciclo di attesa di alcuni secondi si puo' scrivere:

```
FOR K=1 TO N: NEXT K
```

e il tempo di attesa dipende dal valore di N.

Il formato della frase e':

```
FOR var.controllo = val.iniz. TO val.fin. STEP val.incr.
```

dove : var.controllo e' il nome della variabile che serve per controllare il ciclo (il contatore del ciclo);
val.iniz. e' il valore iniziale per la variabile di controllo del ciclo;
val.fin e' il valore finale per la variabile di controllo del ciclo;
val.incr. e' il numero da sommare algebricamente alla variabile di controllo ad ogni ciclo.

Questi ultimi 3 dati possono anche essere variabili. Se l'incremento e' 1, si puo' tralasciare lo STEP.

Vengono eseguite in modo ciclico tutte le linee di programma comprese tra la frase FOR e la frase NEXT. Quando viene eseguita la frase FOR viene creata la variabile di controllo e le viene assegnato il valore iniziale, inoltre viene memorizzato il valore finale e l'incremento. Questo

significa che se questi due dati sono variabili, essi possono anche venir modificati nel ciclo. Nel ciclo invece non deve essere modificata la variabile di controllo. Poi il programma esegue le frasi che seguono FOR fino alla linea dove si trova NEXT. Quando viene incontrato il NEXT il sistema somma algebricamente lo STEP alla variabile di controllo e poi controlla se essa ha superato il valore finale. Se il valore finale non e' stato superato allora il programma torna ad eseguire le linee di programma che iniziano subito dopo il FOR. Se provate questo esempio vedete come funziona il FOR:

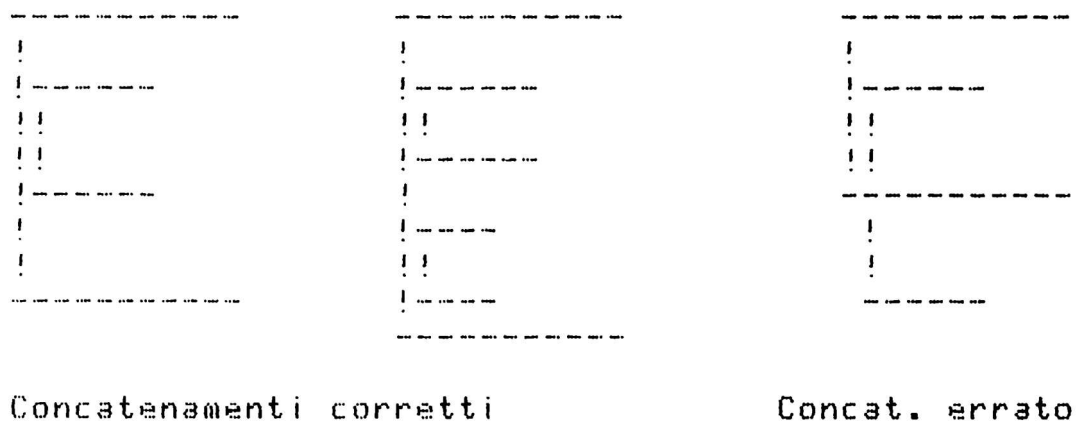
```
10 FOR K=1 TO 10
20 PRINT K;
30 NEXTK
35 PRINT
40 PRINT"VALORE DI K DOPO 10 GIRI: ";K
```

infatti vedrete sul video i numeri interi da 1 a 10 e poi a capo:

```
VALORE DI K DOPO 10 GIRI: 11
```

e questo sta a dimostrare che all'uscita dal ciclo K contiene un valore con il quale il ciclo non e' stato percorso.

Le frasi FOR possono essere concatenate tra loro, ma i cicli non possono intrecciarsi tra loro; segue uno schema di concatenamenti giusti e di concatenamento errato:



Concatenamenti corretti Concat. errato

Questi schemi tradotti in frasi di programma corrispondono a:

```
10 FOR K=1 TO X      10 FOR K=1 TO Y      10 FOR K=1 TO Z
......               ......               ......
40 FOR J=1 TO N      40 FOR J=1 TO M      40 FOR J=1 TO R
......               ......               ......
80 NEXT J            70 NEXT J            90 NEXT K
......               ......               ......
90 NEXT K            80 FOR I= 1 TO       94 NEXT J
                     ......
                     90 NEXT I
                     99 NEXT K
```

Si possono avere fino a 9 FOR concatenati, questa limitazione dipende dalle dimensioni dell' area stack.

GET in questa frase la parola chiave deve essere seguita dal nome di una variabile e serve per leggere un carattere dalla tastiera, senza dare il RETURN. Quando il programma esegue questa istruzione inizia immediatamente a leggere e quindi se non e' stato premuto alcun tasto legge una stringa nulla o uno zero a seconda del tipo di variabile che segue GET. Viene usata questa istruzione facendola seguire dall'analisi del carattere ricevuto, per creare delle attese nel programma ed operare delle scelte. Esempio:

```
10 GET A$
20 IF A$ = "" THEN 10
```

queste due istruzioni fanno proseguire il programma solo se viene premuto un qualunque tasto; infatti la stringa nulla non corrisponde ad alcun tasto.

GET# al carattere # deve seguire il numero logico di un file precedentemente aperto con OPEN e legge un carattere dal file.

GOSUB serve per saltare ad un sottoprogramma, la parola chiave deve essere seguita da un numero di linea. Il sistema ricorda il numero di linea della istruzione GOSUB e quando nel sottoprogramma incontra la frase RETURN torna alla linea dopo il GOSUB. Il sistema consente di avere al massimo 23 GOSUB attivi contemporaneamente.

GOTO serve per saltare ad una linea qualunque nel programma, e' l'istruzione di salto incondizionato. Si puo' anche scrivere GO TO.

IF.....THEN e' la frase che consente di operare delle scelte. Tra le parole chiave IF e THEN si pone una relazione condizionale; se essa e' vera il programma esegue quello che

viene chiesto dopo il THEN, se essa e' falsa il programma prosegue dalla linea seguente. Sono possibili 3 tipi di IF; li descriviamo facendo degli esempi:

```
. 100 IF A = B THEN 300          se il valore di A e' uguale
  110........                    al valore di B, cioe' la
  ...                            condizione e' vera il pro=
  ...                            gramma prosegue dalla linea
  300.......                     300, se no prosegue da 110;

. 150 IF A<=B THEN PRINT "A<=B":GOTO 300
  160 PRINT "A>B"                se A<=B stampa A<=B e poi
  .....                          rimanendo sulla stessa li=
  .....                          nea esegue GOTO300, se no
  300....                        stampa A>B e prosegue;

. 100 IF A = B GOTO 100          e' analoga al primo esempio
                                 e si comporta allo stesso
                                 modo.
```

IF A THEN equivale a IF A <> 0 THEN.

Se dopo il THEN si vogliono eseguire piu' istruzioni esse devono essere separate dai due punti e vengono eseguite tutte se la condizione e' vera.

INPUT la parola chiave e' seguita dai nomi delle variabili nelle quali si vuole memorizzare quanto si legge dalla tastiera, i nomi delle variabili devono essere separati dalla virgola. Il programma si ferma ed evidenzia un punto interrogativo sul video. L'utente risponde con i dati e se ne scrive meno di quanti richiesti il programma si ferma ancora ed evidenzia 2 punti interrogativi per segnalare che e' ancora in attesa di dati. I dati che si scrivono devono essere separati dalla virgola e si deve terminare premendo RETURN. Se l'utente fornisce un numero di dati maggiore del numero delle variabili il sistema segnala EXTRA IGNORED ed il programma prosegue; i dati in piu' sono persi. Se la risposta e' errata per il tipo dei dati, cioe' a richiesta numerica si risponde con stringhe, il sistema segnala REDO FROM START e ripropone il punto interrogativo restando in attesa di INPUT.

Se si vuole rendere piu' comprensibile la richiesta di dati si puo' scrivere la frase ponendo dopo la parola INPUT una stringa tra apici recante una frase opportuna, scrivere un punto e virgola e poi la lista di variabili; cosi':

```
20 INPUT "SCRIVI 2 NUMERI";A,B
```

La frase tra apici non puo' superare i 20 caratteri.

La frase di INPUT puo' essere interrotta premendo contemporaneamente SHIFT destro e STOP RUN, oppure premendo contemporaneamente RUN STOP e RESTORE.

INPUT# lavora come la INPUT e legge dati dal file aperto avente lo stesso numero logico. Non ha senso in questo caso usare il formato con la frase tra apici.

LET questa parola chiave puo' essere usata nelle frasi di assegnazione e di calcolo, ma e' opzionale. Le frasi esempio che seguono hanno lo stesso significato:

```
10 LET A = 5      corrisponde a   10 A = 5
60 X = (X-3)*7    corrisponde a   60 LET X = (X-3)*7
```

NEXT questa frase serve per chiudere il ciclo di FOR. Alla parola chiave NEXT si puo' far seguire il nome della variabile di controllo del FOR o scrivere solo NEXT. Il sistema se manca la variabile chiude l'ultimo FOR aperto, se c'e' la variabile controlla che essa corrisponda a quella dell'ultimo FOR aperto. Si puo' anche scrivere NEXT L,K per chiudere due FOR di cui il primo aperto riguarda L ed il secondo K.

ON....GOTO.... questo comando consente le scelte multiple. Esso si scrive cosi':

```
ON K GOTO N1,N2,N3,....
```

e si comporta cosi:

. K, che puo' anche essere una espressione, viene considerata come numero intero;

. se K=1 va ad eseguire la linea numero N1, cioe' salta a N1, essendo N1 il primo numero nella lista;

. se K=2 va ad eseguire la linea N2, essendo N2 il secondo numero nella lista;

. e cosi' via tenendo conto del numero dei numeri di linea presenti;

. se K=0, K<0 o K maggiore del consentito il programma prosegue dalla linea seguente.

Si puo' usare anche: ON...GOSUB... e valgono le stesse regole viste per GOTO.

OPEN questa frase serve per permettere al VIC di collegarsi con le periferiche. Il formato completo della frase e':

```
OPEN LF,D,SA,FN
```

dove:

. LF e' il numero logico del file che si desidera aprire ed e' il numero da usare dopo # nelle frasi di lettura e scrittura, esso puo' andare da 1 a 255;
. D e' il numero che individua la periferica e puo' essere:

0 per il video;
1 per il registratore a nastro;
4 per la stampante;
8 per il disco;

. SA e' un indirizzo secondario che specifica a seconda della periferica usata il tipo di operazione, il numero del buffer, il canale usato. Per la cassetta:
SA=0 significa lettura, SA=1 significa scrittura, SA=2 significa scrittura con carattere finale di FINE FILE.
. FN e' il nome del file interessato all'operazione, non puo' essere omesso per il disco.
Esempi:

```
10 OPEN 1,0        apre il video come file numero 1;

15 OPEN 2,1,0,"DATI"   apre sulla cassetta il file
                       numero 2 di nome DATI per
                       leggere;

30 OPEN 3,4          apre sulla stampante il file 3;

40 OPEN 2,8,15     apre il file 2 sul canale 15 del
                   disco;

50 OPEN 2,8,2,"ANAGRAFE,S,W"
                   apre sul disco il file 2 sul buf=
                   fer 2 per scrivere (W) un file di
                   tipo sequenziale S.
```

POKE la parola chiave deve essere seguita da 2 numeri (o variabili o espressioni numeriche) separati da virgola. Il primo si riferisce all'indirizzo di una locazione di memoria e deve essere compreso tra 0 e 65535, mentre il secondo deve poter essere contenuto in un byte e quindi deve essere compreso tra 0 e 255. Il comando serve per scrivere nel byte di indirizzo "primo numero" il "secondo numero".

PRINT serve per fare apparire dati sul video. La parola chiave puo' essere seguita da:

. frasi tra apici;
. nomi di variabili;
. nomi di funzioni;
. simboli di punteggiatura come virgola e punto e virgola.

Gli elementi da stampare appaiono in sequenza sul video, andando a capo se la riga e' piena. La virgola fa saltare

per il prossimo elemento alla prossima zona di stampa se precedentemente sono stati stampati al massimo 10 caratteri, altrimenti viene saltata un'altra zona di stampa. La zona di stampa e' di 12 caratteri. Il punto e virgola fa stampare gli elementi vicini nel loro formato di stampa. Il formato di stampa per le stringhe non prevede aggiunte di caratteri, mentre per i numeri pone uno spazio o il segno prima e uno spazio dopo il numero.
Possono essere usate le funzioni TAB e SPC per posizionarsi dove si vuole.
Se la lista dei dati termina senza virgola o punto e virgola si ha una spaziatura verticale, se termina con uno dei due caratteri citati non si ha spaziatura verticale.

PRINT# il simbolo # deve essere seguito dal numero logico di un file precedentemente aperto. Per alcune apparecchiature periferiche non si possono usare le funzioni sopra viste.

READ la parola chiave e' seguita da una lista di variabili separate da virgole. Il programma va a prelevare dal blocco dati, partendo dalla posizione del puntatore interno i dati in sequenza e li assegna alle variabili della lista; si ha errore se si tenta di leggere un numero e si trova una stringa, o si tenta di leggere piu' dati di quelli disponibili.

REM alla parola chiave possono seguire commenti fino alla lunghezza massima di una linea di programma. Essa viene saltata durante l'esecuzione del programma.

RESTORE serve per riposizionare il puntatore interno all'inizio del blocco dei dati preparato dalle frasi DATA.

RETURN serve per chiudere logicamente un sottoprogramma e far ritornare alla linea dopo l'ultimo GOSUB eseguito.

STOP fa fermare il programma con il messaggio BREAK IN LINE ....; si puo' continuare il programma usando il comando CONT.

SYS deve essere seguito da un numero compreso tra 0 e 65535. Fa saltare il programma a questo indirizzo, dove deve essere stato memorizzato un programma in linguaggio macchina. Il programma in linguaggio macchina deve terminare con un opportuno comando per tornare al BASIC.

WAIT fa fermare il programma fino a quando il contenuto di una determinata locazione di memoria si e' modificato nel modo voluto. La frase si scrive cosi':

WAIT I,J,K analizza lo stato della locazione I;

```
esegue l'OR esclusivo con K;
esegue l'AND del risultato con J;
se il risultato e' diverso da 0
il programma prosegue, se no
ricomincia da capo.
```

LE FUNZIONI

Si riportano le funzioni del BASIC in ordine alfabetico divise nei 3 gruppi: numeriche, stringa, varie.

Numeriche:

ABS(X) fornisce il valore assoluto di x.

ATN(X) fornisce l'angolo in radianti la cui tangente e' X.

COS(X) fornisce il coseno dell'angolo X. X deve essere in radianti.

EXP(X) calcola il valore della costante e=2.71827183 elevata a X.

FNYY(X) calcola il valore della funzione YY, definita con DEF FNYY, con argomento X.

INT(X) ritorna il valore di X troncato dei decimali senza arrotondamento.

LOG(X) calcola il logaritmo naturale, cioe' in base e, di X. Per ottenere il log in base 10 basta dividere per LOG(10).

PEEK(X) fornisce il contenuto della locazione di memoria di indirizzo X. Deve essere X compreso tra 0 e 65535.

RND(X) questa funzione fornisce un numero pseudo-a-caso compreso tra 0 e 1. La sequenza dei numeri viene generata prendendo come origine un valore iniziale che dipende da X. Se X=0 la sequenza prende come origine il contatore dei fotogrammi dello schermo al momento. Se X<0 esso viene preso come origine e quindi ogni volta che si fa girare il programma si ottiene la stessa sequenza di numeri. Se X>0 il valore di X non influisce sull'origine della sequenza, che dipende dallo stato del calcolatore.

SGN(X) fornisce 0 per X=0, 1 per X>0 e -1 per X<0.

SIN(X) calcola il seno dell'angolo X, che deve essere espresso in radianti.

SQR(X) calcola la radice quadrata di X, che deve essere positivo, altrimenti si ha errore.

TAN(X) calcola la tangente dell'angolo X che deve essere in radianti.

USR(X) fa saltare il programma ad una sequenza di istruzioni in linguaggio macchina il cui indirizzo iniziale si deve trovare nei bytes 1 e 2. Il valore di X viene passato al programma in linguaggio macchina e questo ritorna un valore al programma Basic.

Stringa:

ASC(X$) ritorna il codice ASCII del primo carattere della stringa X$.

CHR$(X) e' la funzione opposta di ASC, fornisce il carattere il cui codice ASCII e' X.

LEFT$(X$,X) ritorna una stringa formata dagli X caratteri di sinistra della stringa X$.

LEN(X$) fornisce un numero che e' uguale al numero di caratteri della stringa.

MID$(X$,S,X) ritorna una stringa formata da X caratteri della stringa X$ partendo dalla posizione S.

RIGHT$(X$,X) ritorna gli X caratteri piu' a destra della stringa X$.

STR$(X) ritorna il numero X trasformato in stringa, preceduto da uno spazio o dal segno meno.

VAL(X$) e' l'inverso della funzione precedente. La stringa X$ viene trasformata in un numero. Se la stringa contiene caratteri non numerici la conversione avviene per i caratteri numerici validi che precedono i non validi senza segnalazione di errore.

Varie:

FRE(X) non ha importanza cosa contiene X, la funzione fornisce un numero uguale al numero di bytes liberi nella memoria del VIC.

POS(X) ritorna un numero compreso tra 0 e 21 che rappresenta la colonna sulla quale si trova il cursore dello

schermo; X puo' essere qualunque.

SPC(X) fa saltare X spazi senza cancellare eventuali caratteri presenti.

TAB(X) X rappresenta la colonna dove si vuole sia stampato il prossimo carattere. Se X supera 21 si passa alla riga successiva.

APPENDICE F

LE POKE DI USO COMUNE E ALTRE COSE UTILI

Per comodita' riportiamo un elenco dei comandi POKE piu' usati normalmente:

POKE 36869,240	fa passare al set di caratteri grafici, quello attivo al momento dell'accensione
POKE 36869,242	fa passare al set di caratteri maiuscolo/minuscolo
POKE 36879,X	modifica in base a X la combinazione dei colori dello sfondo, del bordo e del testo
POKE 36879,27	attiva la combinazione di colori che si ha all'accensione
POKE 36878,X	predispone il volume per il sonoro, 0<=X<=15
POKE 36877,X	serve per la tonalita' del rumore bianco, 128<=X<=255
POKE,36876,X	serve per la tonalita' alta, 128<=X<=255
POKE 36875,X	serve per la tonalita' media, 128<=X<=255
POKE 36874,X	serve per la tonalita' bassa, 128<=X<=255

Riportiamo un elenco delle CHR$ piu' usate con il comando PRINT:

CHR$(5)	predispone il testo in bianco
CHR$(13)	sostituisce il RETURN
CHR$(14)	fa passare al set maiuscolo/minuscolo
CHR$(17)	fa abbassare il cursore di una riga

CHR$(18)	attiva il RVS ON
CHR$(19)	sposta il cursore nell' angolo in alto a sinistra
CHR$(20)	corrisponde al tasto DEL
CHR$(28)	predispone il testo in rosso
CHR$(29)	sposta il cursore a destra di una colonna
CHR$(30)	predispone il testo in verde
CHR$(31)	predispone il testo in blu
CHR$(32)	produce uno spazio
CHR$(141)	corrisponde a SHIFT e RETURN
CHR$(142)	fa passare al set grafico
CHR$(144)	predispone il testo in nero
CHR$(145)	fa alzare il cursore di una riga
CHR$(146)	disattiva il RVS ON cioe' corrisponde a RVS OFF
CHR$(147)	pulisce lo schermo e manda il cursore in alto a sinistra
CHR$(148)	corrisponde al tasto INST
CHR$(156)	predispone il colore in viola
CHR$(157)	fa spostare il cursore a sinistra di una posizione
CHR$(158)	predispone il testo in giallo
CHR$(159)	predispone il testo in azzurro
CHR$(160)	produce uno spazio + shift

Ricordiamo inoltre che:

SPC(X)	fa spaziare di X posizioni senza cancellare

TAB(X) fa andare alla colonna X, la prima a sinistra corrisponde a zero

Per interrompere un programma che chiede INPUT si devono premere contemporaneamente i tasti RUN STOP e RESTORE, oppure rispondere solo con RETURN.

RUN STOP e SHIFT fa caricare il primo programma che si trova sulla cassetta.

Per ottenere in stampa numeri interi incolonnati a destra si deve procedere cosi':

- se i numeri sono interi trasformarli in stringa con la funzione STR$(K), la quale pone davanti al numero uno spazio o il segno meno;
- preparare una stringa di spazi shiftati, cioe' dopo gli apici premere un certo numero di spazio e SHIFT insieme e chiudere gli apici;
- concatenare la stringa di spazi con la stringa del numero e prendere con la funzione RIGHT$ il numero di caratteri desiderato.

Esempio:
```
A%=546
B%= -3458
A$=STR$(A%)
B$=STR$(B%)
S$=" 6 spazi + SHIFT"
PRINT RIGHT$(S$+A$,9)
PRINT RIGHT$(S$+B$,9)
```
vedrete apparire i numeri incolonnati a destra.

Per ottenere lo stesso risultato con i numeri decimali dovete moltiplicarli per una opportuna potenza di 10 (1000 se volete 3 decimali, 100 se ne volete 2), poi renderli interi con la funzione INT. A questo punto i numeri possono essere trasformati in stringa mediante la STR$, la stringa puo' essere manipolata separando la parte decimale dalla intera, si puo' inserire il punto o la virgola decimale come stringa e operare come sopra per incolonnarli.

Ricordate che in fase di INPUT gli spazi a sinistra del dato vengono cancellati, se volete conservarli dovete usare spazio + SHIFT.

Per avere stringhe tutte della stessa lunghezza si deve creare una stringa di spazi o di spazi + SHIFT e operare dei concatenamenti, poi con le funzioni di stringa si prendono le parti che interessano.

APPENDICE G

I COMANDI ABBREVIATI

Abbiamo gia' sperimentato che invece della parola PRINT si puo' usare il punto interrogativo; questo e' un modo abbreviato di scrivere il comando. Esistono possibilita' analoghe anche per altri comandi BASIC e sono riportate nella tabella che segue. Per abbreviare i comandi si deve scrivere o solo la prima lettera seguita (o le prime due lettere seguite) dalla prossima lettera premuta insieme al tasto SHIFT. Nel programma viene conservato il codice corrispondente alla parola chiave completa, come spiegato nella Appendice J. Se si chiede la lista del programma con il comando LIST le parole chiave appaiono complete. In fase di scrittura quando si preme una lettera insieme a SHIFT appare il corrispondente carattere grafico.

COMANDO	ABBREVIAZIONE	COMANDO	ABBREVIAZIONE
AND	A + SHIFT e N	SAVE	S + SHIFT e A
NOT	N + SHIFT e O	STEP	ST + SHIFT e E
CLOSE	CL + SHIFT e O	STOP	S + SHIFT e T
CHR	C + SHIFT e L	SYS	S + SHIFT e Y
CMD	C + SHIFT e M	THEN	T + SHIFT e H
CONT	C + SHIFT e O	VERIFY	V + SHIFT e E
DATA	D + SHIFT e A	WAIT	W + SHIFT e A
DEF	D + SHIFT e E	ABS	A + SHIFT e B
DIM	D + SHIFT e I	ASC	A + SHIFT e S
END	E + SHIFT e N	ATN	A + SHIFT e T
FOR	F + SHIFT e O	CHR$	C + SHIFT e H
GET	G + SHIFT e E	EXP	E + SHIFT e X
GOSUB	GO + SHIFT e S	FRE	F + SHIFT e R
GOTO	G + SHIFT e O	LEFT$	LE + SHIFT e F
INPUT#	I + SHIFT e N	MID$	M + SHIFT e I
LET	L + SHIFT e E	PEEK	P + SHIFT e E
LIST	L + SHIFT e I	RIGHT$	R + SHIFT e R
LOAD	L + SHIFT e O	RND	R + SHIFT e N
NEXT	N + SHIFT e E	SGN	S + SHIFT e G
OPEN	O + SHIFT e P	SIN	S + SHIFT e I
POKE	P + SHIFT e O	SPC(	S + SHIFT e P
PRINT	?	SQR	S + SHIFT e Q
PRINT#	P + SHIFT e R	STR$	ST + SHIFT e R
READ	R + SHIFT e E	TAB(	T + SHIFT e A
RESTORE	RE + SHIFT e S	USR	U + SHIFT e S
RETURN	RE + SHIFT e T	VAL	V + SHIFT e A
RUN	R + SHIFT e U		

APPENDICE H

COME CALCOLARE ALCUNE FUNZIONI MATEMATICHE

Le funzioni che seguono non sono fornite intrinsecamente dal BASIC, ma possono essere calcolate con le seguenti formule:

Funzione	Formula di calcolo
SECANTE	SEC(X)=1/COS(X)
COSECANTE	CSC(X)=1/SIN(X)
COTANGENTE	COT(X)=1/TAN(X)
ARCOSENO	ARCSIN(X)=ATN(X/SQR(-X*X+1))
ARCOCOSENO	ARCCOS(X)=-ATN(X/SQR(-X*X+1)) +<PI>/2
ARCOSECANTE	ARCSEC(X)=ATN(X/SQR(X*X-1))
ARCOCOSECANTE	ARCCSC(X)=ATN(X/SQR(X*X-1)) +(SGN(X)-1)*<PI>/2
ARCOCOTANGENTE	ARCOT(X)=ATN(X)+<PI>/2
SENOIPERBOLICO	SINH(X)=(EXP(X)-EXP(-X))/2
COSENOIPERBOLICO	COSH(X)=(EXP(X)+EXP(-X))/2
TANGENTEIPERBOLICA	TANH(X)=EXP(-X)/(EXP(X)+EXP(-X)) *2+1
ARCOSENOIPERBOLICO	ARCSINH(X)=LOG(X+SQR(X*X+1))
ARCOCOSENOIPERBOLICO	ARCCOSH(X)=LOG(X+SQR(X*X-1))
ARCOTANGENTEIPERBOLICA	ARCTANH(X)=LOG((1+X)/(1-X))/2
ARCOSECANTEIPERBOLICA	ARCSECH(X)=LOG((SQR(-X*X+1)+1)/X)
ARCOCOSECANTEIPERBOLICA	ARCCSCH(X)=LOG((SGN(X)*SQR(X* X+1)/X)
ARCOCOTANGENTEIPERBOLICA	ARCCOTH(X)=LOG((X+1)/(X-1))/2

APPENDICE I

COLORI SFONDO/BORDO E TESTO

Al momento dell'accensione del VIC il colore dello sfondo e' bianco, il colore del bordo e' azzurro ed il cursore e' blu.

Se volete modificare la combinazione dei colori dovete scrivere ed eseguire o in modo immediato o in una linea di programma:

```
POKE 36879,X
```

dove X deve essere un numero compreso tra 0 e 255.

Per facilitarvi nella scelta dei colori potete servirvi della tabella che segue, dove sono riportati i numeri corrispondenti alle combinazioni dei colori sfondo/bordo, da usare come X nel comando POKE. Nelle colonne della tabella sono riportati gli 8 possibili colori del bordo e nelle righe i 16 possibili colori dello sfondo.

	BORDO							
SFONDO	BLK	WHT	RED	CYAN	PUR	GRN	BLU	YEL
BLACK	8	9	10	11	12	13	14	15
WHITE	24	25	26	27	28	29	30	31
RED	40	41	42	43	44	45	46	47
CYAN	56	57	58	59	60	61	62	63
PURPLE	72	73	74	75	76	77	78	79
GREEN	88	89	90	91	92	93	94	95
BLUE	104	105	106	107	108	109	110	111
YELLOW	120	121	122	123	124	125	126	127
ORANGE	136	137	138	139	140	141	142	143
LT. ORANGE	152	153	154	155	156	157	158	159
PINK	168	169	170	171	172	173	174	175
LT. CYAN	184	185	186	187	188	189	190	191
LT. PURPLE	200	201	202	203	204	205	206	207
LT. GREEN	216	217	218	219	220	221	222	223
LT. BLUE	232	233	234	235	236	237	238	239
LT. YELLOW	248	249	250	251	252	253	254	255

Come potete vedere i numeri colore vanno di 8 in 8 in