# Kit di programmazione S.O.G.A.R. 128

*di Ugo Boccardi - Trani (BA)*

Con la nascita del Commodore Amiga la maggior parte degli utenti Commodore ha preferito provvedere alla permuta del vecchio 64 con il più potente fratellino. Nonostante ciò ho potuto notare, almeno da queste parti, che questa corsa all'acquisto si è parzialmente calmata e parecchi non hanno sacrificato al «Dio Amiga» tutto il parco software del buon 64 per passare ad una macchina più potente.

Di contro, ho natato che coloro i quali avevano acquistato il Commodore 128 si sono visti «abbandonati» al proprio destino: la macchina si presta molto bene ad applicazioni semiprofessionali, ma chi realizza i programmi?

In un primo atto impulsivo, anch'io (utente di un Commodore 128D) sarei passato all'Amiga poiché, visitando le vetrine dei rivenditori Commodore, rimanevo incantato dalla splendida grafica e dai suoni entusiasmanti.

Ciò che mi ha colpito è soprattutto l'estrema semplicità d'uso dei programmi: con tutte quelle finestre, se ben disposte, e quei menu a tendina, si riesce ad usare un programma senza aver letto il manuale.

A questo punto ho cercato di realizzare le stesse cose in HIRES a 40 colonne con risultati sufficienti, ma il Commodore 128 ha le 80 colonne a colori che lo rendono per questo un computer «serio». Ho provato anche a realizzarle in 80 colonne, ma la lentezza e la enorme difficoltà in Basic mi hanno scoraggiato. Poi, in un programma che si rispetti si hanno grafici di HIRES e chi me li dava in 80 colonne? Utilizzare un Expander 128 per la grafica poteva essere una soluzione (quello pubblicato da MCmicrocomputer andava bene), ma scrivere in HIRES una paginata era un guaio e la stampa di parte dello schermo assurda; e la cosa più importante: I COLORI NON C'ERANO.

Per risolvere queste situazioni decido di realizzare un nuovo Sistema Operativo che permetta varie possibilità di utilizzo, ma i problemi sono molti, il più difficile è la lentezza del Basic interpretato: è necessario compilare il Basic per ottenere risultati soddisfacenti.

Così è nato S.O.G.A.R. 128: il sistema Operativo Grafico ad Altissima Risoluzione. Cosa ben importante, come vedremo, è la compilabilità dei programmi mediante una semplice procedura.

## La filosofia di utilizzo

In un computer la grafica utilizzabile tramite Basic, a mio parere, non deve essere utilizzata per realizzare sofisticatissimi programmi di controllo grafico poiché la lentezza del Basic (anche compilato), sminuirebbe tutte le possibili potenzialità di una macchina: questo tipo di software, di solito, viene realizzato in Assembler.

La grafica da Basic, dunque, serve per realizzare programmi professionali che permettano di avere grafici illustrativi e paginate selettive piuttosto sofisticate, tali da rendere utilizzabile il programma nel modo più intuitivo possibile (vedi finestre su Amiga).

Per risolvere questo problema è necessario avere una certa interazione tra la grafica e il testo, un metodo ideale è simile a quello utilizzato sui sistemi MS-DOS: nel momento in cui si seleziona il modo grafico tutto l'editing di schermo va in alta risoluzione. In altre parole, si scrive o si corregge il proprio listato sopra la paginata grafica. Questo è molto comodo ed utile al programmatore che può sfruttare le istruzioni grafiche per realizzare una bella paginata selettiva o introduttiva assieme al testo vero e proprio.

Il S.O.G.A.R. riprende questa filosofia d'utilizzo in modo migliore e più sofisticato.

Il computer appena acceso può lavorare in 80 o in 40 colonne; abbandonando le 40 colonne, analizziamo solo lo schermo ad 80 colonne.

In questa modalità, normalmente, si può usare su 25 righe il cursore per scrivere i propri programmi cioè, se lo paragoniamo alle 40 colonne, siamo in modo testo, per poter disegnare anche un solo punto bisogna ricorrere ad espansioni Basic in grado di sfruttare i 640×200 pixel della altissima risoluzione ottenibile sul 128 però ad un solo colore.

La difficoltà di queste soluzioni è che non si riesce ad ottenere un programma serio che sfrutti la massima risoluzione senza fare continuamente avanti e indietro tra HIRES e modo testo.

Il S.O.G.A.R. riesce ad ovviare a ciò modificando l'editor di schermo. Vediamo in che modo.

## Come funziona

Una volta digitato RUN«S.O.G.A.R. 128» il computer farà apparire una paginata celeste con su scritto che il S.O.G.A.R. è presente ed infine, appare «READY.» con una lineetta sotto. Questa lineetta è il nuovo cursore il quale sta ad indicare che siamo in SuperHIRES ovvero possiamo scrivere, editare un listato, usare PRINT, PRINTUSING, INPUT ecc. tutto in Alta Risoluzione in 640×176 a 16 colori in un quadrato di 8×8 pixel.

Si potrebbe obiettare: «Solo 640×176» ma, tenendo presente che si hanno a disposizione solo 16K di ram video, questo è il massimo per una risoluzione a 16 colori. Comunque, andando più avanti si vedrà che ciò non è un problema: si potrà selezionare anche una grafica in 640×192 o 640×200 in monocrome.

Il nuovo Sistema Operativo, dunque, permette una interazione tra ciò che si disegna e il modo testo, nella descrizione tecnica viene spiegato come avviene ciò; per il momento passiamo a spiegare come utilizzare nei propri programmi questo S.O.

*È disponibile, presso la redazione, il disco con il programma pubblicato in questa rubrica. Le istruzioni per l'acquisto e l'elenco degli altri programmi disponibili sono a pag. 231.*

Non è necessario alcuno strano accorgimento per la digitazione dei programmi: basta entrare nell'ordine di idee che tutto ciò che si scrive lo si fa in SuperHIRES per cui ciò che si andrà a disegnare si sovrapporrà al testo scritto; ma si possono anche creare finestre diverse di testo e di grafica con il comando WINDOW.

Iniziamo con lo spiegare una cosa importante: la grafica in 40 colonne non esiste più! Tutti i vecchi comandi grafici che normalmente venivano utilizzati per la grafica in 320×200 ora funzionano in 80 colonne con la risoluzione selezionata.

Premesso ciò, passo ad elencare ed a spiegare i comandi e le loro variazioni rispetto a quelli tradizionali.

### *I comandi grafici*

Da notare che i parametri tra parentesi quadre possono essere omessi.

**GRAPHIC modalità** [,cancellazione]

Questa istruzione imposta il Commodore 128 su una delle sei modalità grafiche:

0 — testo a 40 colonne
1 — grafica 640×200 monocrome e testo 80×25 monocrome
2 — grafica 640×192 16 colori in 16×8 e testo 80×24 a colori
3 — grafica 640×176 16 colori in 8×8 e testo 80×22 a colori
4 — grafica 640×160 16 colori in 4×8 e testo 80×20 a colori
5 — testo ad 80 colonne.

Il parametro di cancellazione, specifica se lo schermo deve essere cancellato (uguale a 1) oppure lasciato intatto (uguale a 0).

Il valore di Default è 0.

Quando si passa da una modalità all'altra, le dimensioni della finestra video vengono cambiate (es. se si passa dal modo 1 al modo 3 le righe da 25 diventano 22) per cui è necessario, al fine di porre i valori effettivi, premere due volte il tasto HOME, la stessa procedura che si usa per cancellare una finestra.

Da notare che quando si seleziona la grafica il cursore diventa una lineetta in basso che non lampeggia; lo scrolling dei colori nella modalità 2 è impreciso poiché la matrice colore è 16×8 mentre lo scrolling avviene per una matrice 8×8.

**DRAW**

Serve per disegnare punti e linee ed è esattamente uguale all'analogo delle 40 colonne (vedi manuale per ulteriori informazioni).

L'unica differenza, che si riscontra anche nei successivi comandi, è che la sorgente colore può essere pari a 2 e, in questo caso, si ha una inversione fra quello che è presente sullo schermo e il punto disegnato.

**BOX**

Anche questo comando è uguale a quello delle 40 colonne ed è inutile quindi soffermarsi, basta vedere il manuale del Commodore 128 per ottenere tutte le informazioni.

**CIRCLE**

Comando per disegnare cerchi, ellissi o archi; non presenta differenze da quello delle 40 colonne. Unica nota è che la risoluzione è raddoppiata in X per cui i cerchi sono delle ellissi e per ottenere cerchi bisogna raddoppiare la coordinata X.

**LOCATE x,y**

Posiziona le coordinate della penna grafica in X e Y.

**COLOR** numero sorgente, numero colore.

Questa istruzione assegna un colore ad una delle sette aree di colore:

Area— sorgente
0 — sfondo - 40 colonne
1 — sfondo testo in SuperHIRES 80 colonne
2 — pen grafica in SuperHIRES 80 colonne
3 — sfondo grafica in SuperHIRES 80 colonne
4 — pen e testo in SuperHIRES modalità 1 - 80 colonne
5 — carattere (schermo 80 o 40 colonne)
6 — sfondo (80 colonne modalità 5) o bordo.

Quando si passa in SuperHIRES a colori, la gestione dei colori è diversa poiché si hanno due sfondi diversi: uno dei testi (area 1) e l'altro dei disegni (area 3). È importante sottolineare che lo sfondo del modo testo si comporta in modo diverso dal tradizionale, infatti, questo viene cambiato carattere per carattere e così è possibile creare, con la WINDOW, finestre con sfondi di colore diverso (cosa non possibile normalmente). Questa maggiore flessibilità dei colori porta, però, a sacrificare il lampeggio e la sottolineatura, quest'ultima risolvibile.

**PAINT** [sorgente colore],x,y

Riempie un'area di colore: l'area viene definita da un perimetro chiuso. In questo caso a differenza dei precedenti comandi la sorgente colore potrà essere solo 0 o 1 (il valore 2 non avrebbe senso).

**CHAR** [sorgente colore],x,y,stringa[,ingrx,ingry]

La CHAR, nel vecchio modo di operare, serviva per scrivere stringhe nello schermo bit map in 40 colonne o funzionava solo come PRINT più sofisticata in modo testo. Poiché la funzione di scrivere in HIRES non è più necessaria, come sappiamo, ho pensato di rendere questo comando leggermente diverso e più potente.

Nel modo testo esso assume le caratteristiche normali (vedi manuale per informazioni), ma nel modo grafico la sua sintassi è quella descritta:

sorgente colore — può assumere i valori 0, 1 o 2 ed ha la stessa funzione della DRAW.

y,y — sono le coordinate del punto da cui deve iniziare a scrivere la stringa, esse vanno da 0 a 640 per la x e da 0 a 200 per la y.

stringa — contiene le parole da scrivere.

ingrx e ingry — indicano il fattore di ingrandimento di ogni singola lettera: ponendo ingrx=2 si avrà una stringa raddoppiata in x e se si pone ingry=3 si avrà una stringa triplicata in y. I valori che possono assumere sono da 1 a 255 (Default) X=1,Y=1).

Nota: Se durante un programma si digita: "POKE6576,3:POKE4588,64" il set di caratteri dovrà essere posto da $4000 a $4800, in questo modo si potranno utilizzare diversi caratteri prelevati da disco.

**OFF**

Comando non presente sul manuale serve per cancellare una finestra in SuperHIRES.

**WINDOW** col sup sin,rig sup sim,col inf des, rig inf des[,tipo]

Questo comando definisce una finestra logica all'interno dello schermo a 40 o 80 colonne. Le coordinate devono essere nella gamma 0-39/79 per i valori di colonna e 0-24/23/21/19 (a seconda della modalità) per i valori di riga. Il flag tipo assume i seguenti significati.

# — spiegazione
0 — finestra 40 o 80 colonne testo.
1 — finestra 40 o 80 colonne testo, cancellata.
2 — finestra 80 colonne grafica.
3 — finestra 80 colonne grafica, cancellata.

In questo modo si possono definire 2 finestre diverse, una di testo e una grafica: tutto ciò che si disegna sarà all'interno della finestra grafica e le coordinate verranno automaticamente aggiornate.

Per riportare la finestra grafica a tutto schermo si deve semplicemente digitare GRAPHIC seguito dal numero indicante la modalità in cui ci si trova.

Nota: Le coordinate nei tipi 2 e 3 possono essere 0-80 per le colonne e 0-25/24/22/20 per le righe.

**COPY** comando[,numero finestra][,x,y]

L'istruzione COPY, come tutti sanno, svolge la funzione di ricopiare su disco un file; poiché questo comando è scarsamente usato, ho deciso di modificarlo per poter sfruttare una serie di funzioni interessanti.

Questa istruzione, in abbinamento alla WINDOW, può permettere tecniche

di programmazione sofisticate. Vediamo come si usa.

Il comando assume diversi significati:
0 — legge una finestra indicata dal numero
1 — legge una finestra indicata dal numero a partire dalle coordinate indicate in X e Y
2 — inverte una finestra indicata dal numero con il contenuto dello schermo
3 — inverte una finestra indicata dal numero con il contenuto dello schermo a partire dalle coordinate X e Y
4 — memorizza una finestra di schermo con il numero indicato.
5 — cancella l'ultima finestra memorizzata
6 — stampa il contenuto di una finestra sulla stampante (tipo MPS 803) in maniera diretta.
7 — stampa il contenuto di una finestra sulla stampante (tipo MPS 803) in modo inverso.

Da ciò si deduce la potenza di COPY, ma bisogna sottolineare un particolare: la finestra a cui si fa riferimento è SOLO quella grafica, cioè definita dalla WINDOW con tipo 2 o 3. Da notare che la stampa avviene ruotata di 90 gradi poiché la stampante non ha 640 punti in orizzontale. Voglio ricordare che la stampa è solo della finestra attuale e infatti, giocando con la WINDOW si può stampare o solo parte o tutto lo schermo.

Un punto importante utile in fase di programmazione è la locazione 4438: in questa si ha lo stato della funzione COPY e assume i diversi valori:
0 — tutto bene
1 — non può leggere una finestra in memoria: è un errore dovuto alla funzione 1 o 3 quando si usano coordinate troppo alte perché la finestra rientri nello schermo
2 — esiste già: si ha quando si vuole memorizzare una finestra con un numero già esistente
3 — non esiste finestra: nelle funzioni 0,1,2,3 quando si richiama una finestra non memorizzata precedentemente
255 — fine memoria: la memoria buffer disponibile per memorizzare le finestre è finita.

A questo punto bisogna specificare che la memoria buffer nella quale vengono memorizzate le finestre si trova nel BANCO 1 da $0400 a $4000, ma è possibile porla (sempre nel BANCO 1) in qualsiasi posto si desideri: i puntatori di questa zona sono posti a $0b10-$0b11 (rispettivamente byte basso-alto) per l'inizio e a $0b12-$0b13 (rispettivamente byte basso-alto) per la fine.

Con questo ultimo comando si conclude la schiera di funzioni utilizzabili con il S.O.G.A.R. 128, ma le potenzialità sono enormi, proprio per questo ho scritto un semplice programma che illustri nelle linee generali ciò che si può fare con questo nuovo Sistema Operativo.

Un'ultima nota importante: dopo l'istruzione IF..THEN è necessario porre ":" fra THEN e un'istruzione grafica di quelle citate.

## *Il programma dimostrativo*

Per illustrare meglio le varie possibilità del sistema S.O.G.A.R., ho preferito non dilungarmi in inutili esempi e consigli al programmatore che fanno perdere tempo, ma considero più veloce scrivere un programma semplice e senza alcuna finalità, ma che utilizzi tutti i comandi nuovi.

In questo modo, metto il programmatore di fronte alla applicazione diretta e permetto di valutare se è necessario ricorrere ad un tale sistema operativo o no. Più avanti si parlerà della compilazione.

Il programma proposto è il classico Studio di Funzioni, ovvero il Match Pack 128 già pubblicato, ma qui in veste grafica diversa: più accattivante.

È inutile dilungarsi ulteriormente nella spiegazione dello stesso, ma voglio solo chiarire che questo è solo un esempio: non è necessario seguire questo metodo di programmazione (il passaggio da un modo grafico e testo) ed infatti ho realizzato un programma, BIBLIO 128 che sfrutta questa grafica in maniera totalmente diversa (rimane sempre nel modo 3).

## *Descrizione tecnica*

Essendo un Sistema Operativo che deve essere utilizzato da programmatori, è necessario dare una descrizione tecnica del tutto per permetterne un corretto uso.

Il S.O.G.A.R. si pone nel BANCO 0 nella zona di memoria da $1300 a $4800, e nel BANCO 1 a partire da $0400 per finire a $4050. Da ciò che si vede, «ruba» molta memoria, ma nel 128 non è un sacrificio.

Utilizza inoltre diversi puntatori posti da $0b00 fino $0b14, e sfrutta la memoria da $03e4 a $03f0. Lascia libera però la zona degli sprite ($0e00-$1000) utile per altre funzioni dato che gli sprite in 80 colonne non esistono.

Ora svelo un trucco.

Per poter far funzionare l'intero edit video in altissima risoluzione, ho ricopiato tutto l'edit vecchio da $c000 a $d000 in memoria RAM a partire da $2000 fino a $3000 e l'ho modificato con le routine «EDITOR» poste da $3020 fino a $3311. Esse controllano se siamo in modo testo o in SuperHIRES e di conseguenza si regolano stampando le lettere nei due modi diversi. In questo modo, modificando tutti i puntatori delle routine del Kernal, il computer funziona con il nuovo editor e così anche il Basic e l'Edit di schermo.

Accanto a ciò ho realizzato le varie routine in Assembler che permettono di utilizzare le varie risoluzioni. Queste si pongono da $1300 a $1e9e e possono essere attivate da comuni POKE e SYS (ricordarsi di ciò).

Inizialmente, infatti, il S.O.G.A.R. era utilizzato in questo modo: POKE e SYS in ogni punto del programma e... ricovero al manicomio da parte mia. È praticamente assurdo ricordarsi tutte quelle SYS e, cosa peggiore, per disegnare un punto bisognava stare 10 minuti per pensare ai comandi da scrivere: non è proprio un vantaggio.

Dimesso dal manicomio, ho realizzato un'interfaccia più comoda e quindi ecco l'attuale S.O.G.A.R.. Questa interfaccia è posta da $3312 a $3800 e provvede a riconoscere i comandi Basic grafici ed a sfruttare le routine S.O.G.A.R..

Da $3800 a $4000 è posta una copia dello schermo in 80 colonne quando si lavora in SuperHIRES. In altre parole l'editor di schermo per funzionare deve leggere le varie istruzioni da ogni riga, cioè quando premo il tasto RETURN esso legge dallo schermo 2 righe e le pone nel buffer da $0200 in poi. In seguito le elabora e esegue le istruzioni (se ci sono). Quando siamo in SuperHIRES è praticamente impossibile far decifrare l'alta risoluzione e quindi tradurre un'insieme di punti in codici ASCII e porli nel buffer a $0200. Per ovviare a ciò ho usato uno schermo «virtuale» posto a $3800 che funziona da schermo testo vero e proprio che si va a sovrapporre a quello in Altissima Risoluzione.

Da $4000 a $4800 si ha memoria libera: serve solo per il nuovo set di caratteri utilizzabile dalla CHAR.

Da $1ea0 a $1f7c c'è lo START per il S.O.G.A.R. e la nuova routine di RUN STOP/RESTORE che evita (con il RESTORE) di rimettere a posto i puntatori del S.O.G.A.R..

## *La fase di compilazione*

Nell'utilizzare questo Sistema Operativo ci si accorge che esso è utile in diverse situazioni, sia in quelle in cui si richiede l'uso della grafica, sia quando quest'ultima non è necessaria. Il problema maggiore si riscontra quando, dopo la realizzazione, un programma risulta lento. A questo punto un programmatore furbo COMPILA. Come TUTTI sanno, però, non è possibile compilare una espansione Basic poiché il compilatore non è in grado di riconoscere le nuove istruzioni. E anche in questo caso, che non ci sono nuove istruzioni, ma sono

solo cambiate le routine a cui fanno riferimento, il compilatore non le riconosce.

Per ovviare a questo problema si potrebbe scivere un programma mediante POKE e SYS: così facendo potrebbe essere compilato. Ma non è umano scrivere un programma in questi termini.

La situazione rimane invariata poiché, l'unica soluzione è quella appena proposta, però... se lo facesse il computer, al nostro posto, sarebbe più comodo!

L'idea è proprio questa: scrivere un programma normalmente e alla fine, quando si è sicuri della sua funzionalità, lo sottoponiamo ad una procedura che chiamiamo di «PRECOMPILAZIONE» e poi alla compilazione vera e propria.

Accanto al S.O.G.A.R. ho realizzato delle subroutine che sfruttano lo stesso mediante POKE e SYS e, quindi rendendo un programma compilabile, ed insieme ho realizzato un programma chiamato «PRECOMPILER» che, svolge la funzione di tradurre le istruzioni grafiche del S.O.G.A.R. in un insieme di variabili e GOSUB che fanno riferimento alle subroutine sopra citate.

Non descrivo il programma «PRECOMPILER» poiché non è necessario, ma passo a dare le indicazioni necessarie per una buona e corretta compilazione.

La prima cosa da fare è, ovviamente, scrivere il programma. Una volta realizzato e controllato in tutti i suoi punti, si procede alla renumerazione a partire dalla linea 1000. Questo è necessario poiché saranno poste in testa al programma le subroutine di controllo S.O.G.A.R.. Dopo ciò, preparare un disco contenente il programma realizzato assieme al programma «PRECOM.LIB». Leggere il programma «PRECOMPILER» e, dopo averlo fatto partire, inserire il disco preparato in precedenza. Il computer inizierà la traduzione del programma facendo scorrere tutti i numeri di linea; al termine uscirà la scritta: FINE (che fantasia!).

Nel disco sarà presente un programma che avrà il nome preceduto da «P/»: questo è il programma precompilato, infatti leggendolo avrà una struttura diversa (tutte POKE e SYS) e sarà più lungo dell'originale.

Questo programma potrà essere compilato con opportuni accorgimenti bisogna limitare le zone di memoria del BANCO 0 e del BANCO 1. Ogni compilatore ha determinate direttive che permettono di informare lo stesso compilatore che non deve usare determinate zone di memoria. Queste zone sono:

BANCO 0 da $1300 a $4800
BANCO 1 da $0400 a $4050 (se si amplia il buffer bisogna proteggere la zona utilizzata).

Prima di concludere è importante tenere presente i seguenti consigli durante la realizzazione del proprio programma:

1) Non bisogna usare le seguenti variabili:

SO%,A%,B%,A1%,B1%,XR%,YR%,IA%, FA%,RO%,MO%,K%,SR$

Poiché vengono utilizzate dalle subroutine poste in testa al programma.

2) I comandi grafici possono avere, come da manuale, anche coordinate polari (es. DRAW1,45#60). Con il S.O.G.A.R. questo metodo funziona, però in compilazione non darà alcun risultato, per cui se un programma dovrà essere compilato NON dovrà avere questo tipo di scrittura.

3) L'istruzione SCALE funziona, ma una volta compilato il programma non avrà alcun effetto, quindi non considerarne l'uso.

4) Il programma compilato una volta fatto partire necessita che venga caricato da disco il file «S.O.G.A.R./C» il quale contiene le routine S.O.G.A.R. in L.M. in versione per i programmi compilati. La differenza dalle routine originali consiste solo nella parte di START.

5) È importante non riunire troppe istruzioni grafiche sulla stessa linea. Il motivo è semplice: in fase di PRECOMPILAZIONE i comandi grafici vengono sostituiti da un insieme di assegnazioni di variabili e da GOSUB che allungano la linea, se vi sono troppe istruzioni sulla stessa linea questa può diventare più lunga di 255 caratteri e quindi non idonea.

6) Al termine della fase di PRECOMPILAZIONE può essere necessario rileggere il programma precompilato in memoria e poi salvarlo di nuovo, al fine di rimettere a posto i puntatori di linea.

## *Conclusioni*

Siamo giunti al termine e credo di non aver omesso nulla.

Spero che la lunghezza dell'articolo non abbia scoraggiato i più abili programmatori poiché non era questa la mia intenzione, infatti desidero solo fornire tutte le spiegazioni possibili per utilizzare al meglio questo Sistema Operativo. Se quindi ho dato l'impressione che, utilizzare il S.O.G.A.R. è impresa da ingegneri elettronici non è assolutamente vero. Provate infatti a vedere il programma Math Pack 128 e ricavate le conclusioni.

Sul disco è presente il S.O.G.A.R. suddiviso in due file «S.O.G.A.R. 128» e «S.O.G.A.R./2» che rappresenta il S.O. vero e proprio il quale, deve essere caricato ogni qual volta si desidera scrivere o usare un programma interpretato.

È presente anche il file «S.O.G.A.R./C» che dovrà essere messo nel disco assieme al programma compilato, e non è necessario, quando si fa funzionare un programma compilato, caricare il S.O.G.A.R. 128.

Accanto a questi vi sono una serie di file programma con vari nomi (R1 1300-164F,R2 183B-1859, ecc.) i quali sono le varie routine S.O.G.A.R. distaccate che unite formano «S.O.G.A.R./2»; le ho inserire accanto ai file non assemblati (non tutti i file sono presenti in Assembler: alcuni sono scritti con il monitor del 128) poiché potrebbero servire a qualche programmatore per modificarle.

Poi si ha ancora un file chiamato «PREROUTINE» che contiene le diverse subroutine per la fase di compilazione. Lo stesso programma con eliminati gli ultimi due byte è presente sul disco sotto il nome «PRECOM.LIB», ed è questo quello da usare in fase di PRECOMPILAZIONE. La eliminazione dei due byte è dovuta alla necessità di unire queste subroutine con un altro programma.

Abbiamo il programma «PRECOMPILER» che è, appunto, il precompilatore (compilato a sua volta per essere più veloce). La versione non compilata è anch'essa presente con il nome «PRECOM».

Vi è anche il Math Pack 128 realizzato con il S.O.G.A.R. 128.

Per concludere non poteva mancare una «delizia», un programma chiamato «DOODLE CONV» il quale legge un file DOODLE del 64, lo traduce nel formato GRAPHIC3 e lo salva sotto forma di finestra con il numero 64.

Per usare questo schermo appena salvato basta inserire nel programma le seguenti istruzioni:

BLOAD"NOME",B1:COPY0,64

ed apparirà l'immagine tradotta che potrà essere utilizzata nei vostri programmi.

Un'ultima cosa, nel disco sono presenti due nuovi set di caratteri con il nome «FEDERATION» e «SCRIPT», vengono utilizzati dal Math Pack 128 ma, possono essere utilizzati da qualunque altro programma.

A questo punto non posso far altro che augurarvi buon lavoro.

# 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 sotto 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 Carlo Perrier 9, 00157 Roma.

| Codice | Titolo Programma | MC n. | Prezzo |
|---|---|---|---|
| **APPLE II** | | | |
| DA2/06 | Miniset + LevaDOS | 37 | 15000 |
| DA2/07 | 27 programmi grafici | 38 | 30000 |
| DA2/08 | Adventure Editor | 38 | 15000 |
| DA2/09 | Animazione Funzioni | 42 | 15000 |
| DA2/12 | Routine Grafiche Estese | 44 | 15000 |
| DA2/13 | Scroll 300 linee | 46 | 15000 |
| DA2/14 | Assembler in Basic | 50 | 15000 |
| DA2/15 | G-Basic II | 53 | 15000 |
| DA2/16 | Disk Editor | 54 | 15000 |
| DA2/17 | Latino | 57 | 15000 |
| DA2/18 | Battaglia | 61 | 15000 |
| DA2/19 | Catalogo | 64 | 15000 |
| DA2/20 | Apple Puzzle II | 65 | 15000 |
| DA2/21 | Precisione Multipla | 66 | 15000 |
| DA2/22 | Sistema 2 + Toto 5.3 IIGS | 68 | 15000 |
| DA2/23 | Operazione Apokalypsis | 71 | 30000 |
| DA2/24 | Classifiche di Formula 1 | 72 | 15000 |
| DA2/25 | Programmabile RPN | 73 | 15000 |
| DA2/26 | Supercircle + Poligonale | 74 | 15000 |
| DA2/27 | Hard Copy OKI 83/A | 76 | 15000 |
| DA2/28 | ProDOS Utility | 77 | 15000 |
| DA2/29 | Modulo Base | 78 | 15000 |
| DA2/30 | List db | 79 | 15000 |
| DA2/31 | Bioritmi | 80 | 15000 |
| **COMMODORE AMIGA** | | | |
| DAM/01 | F-15 | 63 | 15000 |
| DAM/02 | Gest. liste programmi | 64 | 15000 |
| DAM/03 | Studio di Funzioni | 66 | 15000 |
| DAM/04 | Math Pack | 68 | 15000 |
| DAM/05 | Redcode & Mars (Core Wars) | 68 | 15000 |
| DAM/06 | Life | 69 | 15000 |
| DAM/07 | Rubrica Telefonica | 70 | 15000 |
| DAM/08 | Piramidi | 70 | 15000 |
| DAM/09 | Regolazione dei colori | 71 | 15000 |
| DAM/10 | Analitica | 71 | 15000 |
| DAM/11 | Grafici | 72 | 15000 |
| DAM/12 | Traduttore | 73 | 15000 |
| DAM/13 | La Borsa | 74 | 15000 |
| DAM/14 | DMA Music Compiler | 74 | 15000 |
| DAM/15 | Poker | 78 | 15000 |
| DAM/16 | Programmi per il Copper | 79 | 15000 |
| DAM/17 | Mandelbrot mania | 81 | 15000 |
| **MS-DOS** | | | |
| DMS/01 | Plotter + Morse | 67 | 15000 |
| DMS/02 | Melole + Spawn | 68 | 15000 |
| DMS/03 | Pretty + Scritte scorrevoli + Compute | 69 | 15000 |
| DMS/04 | Emulatore CGA per Hercules | 70 | 15000 |
| DMS/05 | Turbo Directory | 71 | 15000 |
| DMS/06 | Math Tool S | 72 | 15000 |
| DMS/07 | Bioritmi + Routine | 72 | 15000 |
| DMS/08 | Salvavideo + Scritte scorrev. + PG151 | 73 | 15000 |
| DMS/09 | Optimizer + Indenter dBase III | 74 | 15000 |
| DMS/10 | Joystick Controller | 75 | 15000 |
| DMS/11 | BootSlow & SlowDown + Turbo Utility | 76 | 15000 |
| DMS/12 | Redcode & Mars (Core Wars) | 76 | 15000 |
| DMS/13 | Gestione Errori Critici Disco + PosCur | 77 | 15000 |
| DMS/14 | Finestre & Desk | 78 | 15000 |
| DMS/15 | General Manager | 78 | 15000 |
| DMS/16 | Tool 05 | 79 | 15000 |
| DMS/17 | Pulldown Menu + Retrace | 80 | 15000 |
| DMS/18 | Righe | 81 | 15000 |
| DMS/19 | La spada di Krall | 82 | 15000 |
| DMS/20 | Regressione | 82 | 15000 |
| **ATARI ST** | | | |
| DST/01 | Virus Killer | 74 | 15000 |
| DST/02 | Mandelbrot + Proiez. Ort. + Bilancio | 78 | 15000 |
| DST/03 | Diagrammi di Henon | 81 | 15000 |
| **COMMODORE 128** | | | |
| D28/01 | MMCalc | 53 | 15000 |
| D28/02 | Hardcopy 128 | 55 | 15000 |
| D28/03 | Sheet It | 57 | 15000 |
| D28/04 | Star Quest | 58 | 15000 |
| D28/05 | Family Budget | 60 | 15000 |
| D28/06 | La Casa Stregata | 61 | 15000 |
| D28/07 | Strutture 80/33 | 63 | 15000 |
| D28/08 | Bas 80 V. 2.0a | 64 | 15000 |
| D28/09 | Paint 80 1.0 | 65 | 15000 |
| D28/10 | Bas 80 V. 2.11 | 66 | 15000 |
| D28/11 | Calendario Perpetuo + Montecarlo | 67 | 15000 |
| D28/12 | Disegna Circuiti | 68 | 15000 |
| D28/13 | Mark's Data Base | 70 | 15000 |
| D28/14 | Label Disk + Disk Editor + Dem DOS | 71 | 15000 |
| D28/15 | Pulldown 128HR + Menu + Drawer | 72 | 15000 |
| D28/16 | Prospettive | 73 | 15000 |
| D28/17 | Char 80 V. 1.0 | 74 | 15000 |
| D28/18 | Italia 128 | 75 | 15000 |
| D28/19 | Super Sprite | 77 | 15000 |
| D28/20 | Othello | 80 | 15000 |
| D28/21 | Expert System Shell 128 | 81 | 15000 |
| D28/22 | Kit di programmazione S.O.G.A.R. 128 | 82 | 15000 |
| **COMMODORE 64** | | | |
| D64/11 | Anno Domini | 57 | 15000 |
| D64/12 | The Disk Editor | 54/6/7 | 15000 |
| D64/13 | Boz's Adventure | 57 | 15000 |
| D64/14 | Link-64 | 57 | 15000 |
| D64/15 | New Char 2.2 | 58 | 15000 |
| D64/16 | Music 64 | 59 | 15000 |
| D64/17 | TRX-MEM | 59 | 15000 |
| D64/18 | WOS + WBasic | 60 | 15000 |
| D64/19 | Strange Basic + Dracula | 63 | 15000 |
| D64/20 | File Rescue | 64 | 15000 |
| D64/21 | La Casa | 64 | 15000 |
| D64/22 | Digital Voice | 65 | 15000 |
| D64/23 | Vita 3D | 65 | 15000 |
| D64/24 | Corso di Linguistica | 66 | 15000 |
| D64/25 | Archiplus | 66 | 15000 |
| D64/26 | Math Pack Plus | 66 | 15000 |
| D64/27 | Scroll + Multitask + Classifica | 67 | 15000 |
| D64/28 | Calend. Perpetuo + Effetto Telecamera | 68 | 15000 |
| D64/29 | Listing Plus + Utility Data | 69 | 15000 |
| D64/31 | Trucchi e Routine per programmatori | 71 | 15000 |
| D64/32 | Flow-Chart + Flower's Love | 73 | 15000 |
| D64/33 | Sprite Editor | 76 | 15000 |
| D64/34 | Portfolio 64 + Elimin. bordi schermo | 77 | 15000 |
| D64/35 | Alfabeto Morse + Locate + Menu/Driver | 78 | 15000 |
| D64/36 | Schedario Gare | 80 | 15000 |
| D64/37 | Intonatore | 81 | 15000 |
| D64/38 | Gendata 64 | 82 | 15000 |
| **MSX** | | | |
| DMX/01 | Toto 13 | 60 | 15000 |
| DMX/02 | Painter | 62 | 15000 |
| DMX/03 | MSX Bank | 63 | 15000 |
| DMX/04 | Grafica 3D + Hard Copy | 65 | 15000 |
| DMX/05 | Easy Disk | 66 | 15000 |
| DMX/06 | Classifiche | 67 | 15000 |
| DMX/07 | Magic Paint | 67 | 15000 |
| DMX/08 | Autogest | 68 | 15000 |
| DMX/09 | Compilatore v. 1.01 | 69 | 15000 |
| DMX/10 | Diskmap | 70 | 15000 |
| DMX/11 | Mini dBase MSX | 71 | 15000 |
| DMX/12 | Grafica in Turbo Pascal | 72 | 15000 |
| DMX/13 | Math Pack Plus 3.20 | 73 | 15000 |
| DMX/14 | RGB CAD | 75 | 15000 |
| DMX/15 | Simple Desk | 76 | 15000 |
| DMX/16 | The MSX2 Super Print | 77 | 15000 |
| DMX/17 | Grafica in Turbo Pascal (Graph 1&2) | 77 | 15000 |
| DMX/18 | Hard Copy | 78 | 15000 |
| DMX/19 | HEXDUMP | 79 | 15000 |
| DMX/20 | Utilities in Turbo Pascal | 80 | 15000 |
| DMX/21 | dBase MSX Plus | 81 | 15000 |
| DMX/22 | Turbo Pascal Turtle Graphics | 82 | 15000 |

Nota:
l'iniziale del codice è C per le cassette, D per i floppy.