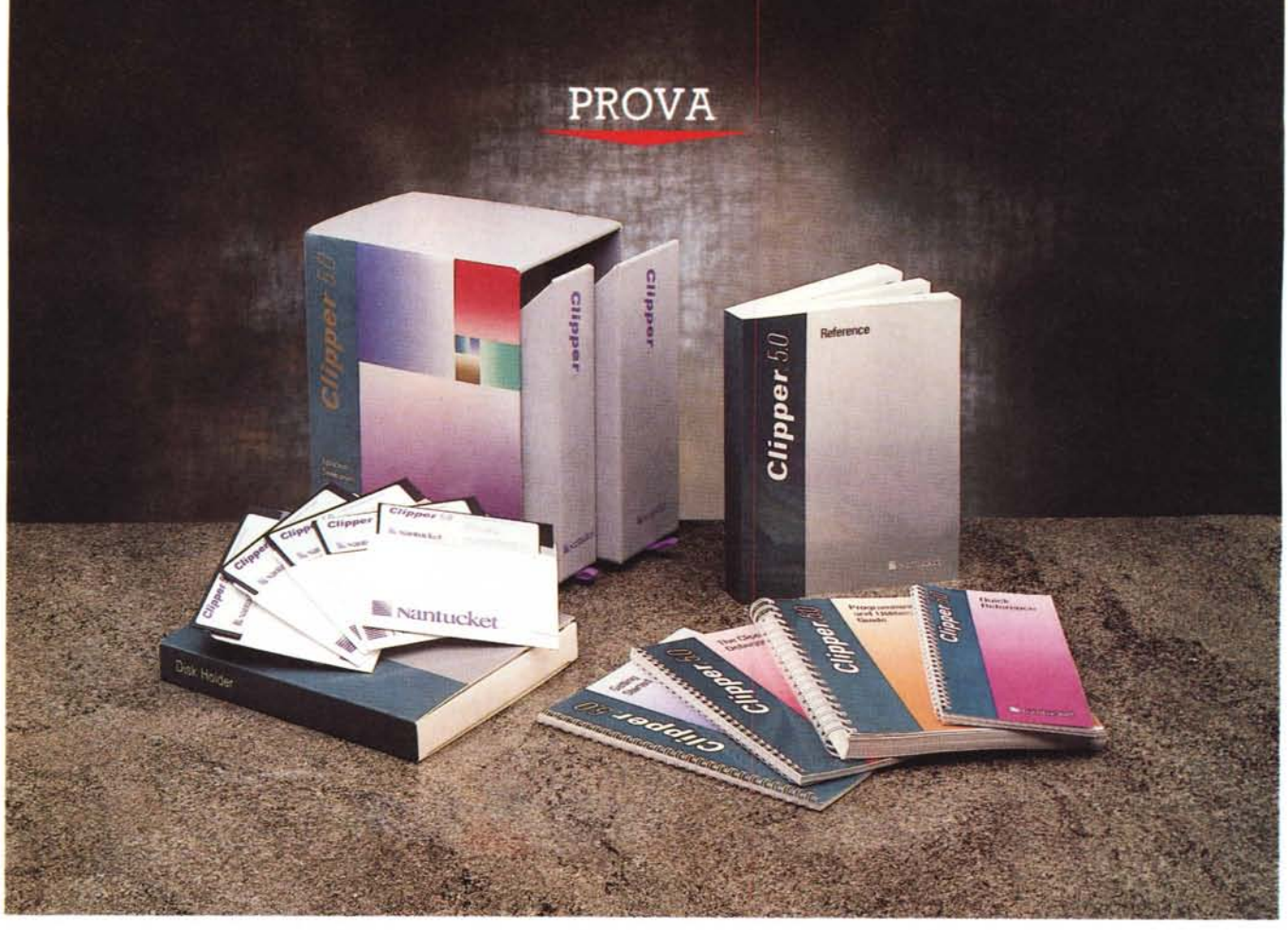


Clipper 5.0

di Gabriele Romanzi

In occasione dell'ultima edizione dello SMAU la Soc. Algol, distributrice per l'Italia dei prodotti Nantucket, ha presentato la nuova versione del compilatore Clipper, intorno alla quale si era creata un'attesa spasmodica; l'annuncio di questa nuova versione risale infatti a circa un anno fa, ma l'effettiva uscita sul mercato è stata ritardata dalla necessità di correggere alcuni problemi riscontrati nella versione di test rilasciata ad alcuni sviluppatori Clipper negli Stati Uniti.

Fino ad oggi, con la precedente versione «Summer 87», il Clipper era un compilatore per il linguaggio dBase a cui aggiungeva delle proprie funzionalità che lo rendevano adatto per lo sviluppo di applicazioni commerciali anche molto complesse, mentre adesso ha assunto una sua fisionomia ben precisa staccandosi dalla compatibilità a tutti i costi con il dBase e, vista la fama che è riuscito a guadagnarsi «sul campo», si propone come il nuovo standard nel campo dei prodotti per DBMS in ambiente MsDos.

Clipper 5.0

Produttore:
Nantucket Corporation
Distributore:
Algol S.p.A.
Via Feltre, 28/6 - 20132 Milano
Tel. 02/26411411
Prezzi *(IVA esclusa):*

Clipper 5.0	*L. 1.590.000*
Upgrade da versione Summer 87 a 5.0	*L. 430.000*

Il pacchetto e l'installazione

Il Clipper 5 arriva in una scatola di cartone rigido che contiene due custodie estraibili, la prima destinata ai manuali e la seconda alla custodia dei floppy e di una busta con delle brochure riguardanti offerte promozionali relative ad alcuni servizi aggiuntivi della Nantucket (hot-line, riviste, ecc.).

La manualistica si presenta più corposa rispetto a quella della versione precedente; i manuali sono cinque (più un «Quick Reference» di formato tascabile), rilegati ad anelli ad eccezione del «Reference» che invece ha una rilegatura a colla come un libro di tipo tradizionale. Quest'ultimo è il vero e proprio manuale di riferimento del linguaggio, comprendente oltre alla sintassi completa dei comandi e delle funzioni anche

una parte introduttiva in cui vengono esposte le principali caratteristiche della programmazione con il Clipper 5; capitoli specifici sono inoltre dedicati agli «statement» del linguaggio, alle classi ed alle direttive per il pre-processore (su questi ultimi punti torneremo in dettaglio nel seguito).

Una prima importante novità di questa nuova versione è la presenza, all'interno del pacchetto, delle Norton Guides con i relativi database per il Clipper 5, in cui sono duplicate le informazioni contenute nella manualistica cartacea; per chi non conoscesse questo prodotto ricordiamo che si tratta di un programma TSR (Terminate and Stay Resident) realizzato dalla Peter Norton Computing che permette di avere a disposizione, alla semplice pressione di un tasto funzione, informazioni utili durante lo sviluppo di un programma, quali la sintassi dei comandi, delle funzioni, ecc..

Storia del dBASE
Riassunto delle puntate precedenti

di Francesco Petroni

Quando nacque il PC IBM, nei primissimi anni '80, sembrava che il mondo che stava per nascere attorno ai PC sarebbe ruotato attorno al linguaggio Basic, l'unico che riusciva a vivere su un hardware, all'epoca, così limitato.

Fortunatamente, già dopo pochi mesi, vennero alla luce alcuni prodotti che da una parte sfruttavano al meglio le caratteristiche hardware, ancora non ben comprese dagli utilizzatori (i primi PC IBM disponevano di 64 kbyte, indirizzabili dal Basic, ma erano espandibili fino a 640), e dall'altra erano utilizzabili molto più facilmente e direttamente anche da utenti alle prime armi (che cominciavano ad essere molto numerosi dato che anche le aziende si erano «accorte» che esistevano i PC).

Uno dei primi prodotti che ebbero notevole successo fu il dBASE III, sviluppato dalla Ashton Tate, sulla base di un precedente dBASE II, nato un paio di anni prima, per piattaforme 8 bit, e riadattato, dapprima come dBASE II e poi appunto come dBASE III, anche al PC quando questo apparve.

Pressoché contemporanea è la nascita del PC classe XT, e quindi anche il disco rigido (inizialmente si partì da misure 10 mega, mentre oggi la misura minima è come noto di 20 mega) cominciò ad essere considerato, per alcune attività, uno standard.

Il dBASE si diffuse moltissimo non solo tra gli utilizzatori finali, che ne apprezzavano la facilità d'uso, sia dall'ambiente «puntino», sia dall'ambiente ASSIST, ma anche dai tecnici sviluppatori, che riuscivano a realizzare, con le sue funzionalità, procedure di tipo gestionale (che richiedono numerosi archivi e calcoli non di tipo scientifico), in un tempo sensibilmente inferiore a quello richiesto dagli altri linguaggi.

Il successo del dBASE fu tale che numerosi altri produttori svilupparono decine di pacchetti ausiliari che si affiancavano al dBASE per migliorarne funzionalità o prestazioni.

Una delle categorie di prodotti che ebbe più successo fu quella costituita dai «compilatori», che fondamentalmente risolvevano due problemi legati all'utilizzo del dBASE.

Il primo è quello della velocità, che è ovviamente sensibilmente maggiore in un compilatore che non nel dBASE intepretato. Il secondo è quello della diffusione del pacchetto applicativo, che, se sviluppato sotto dBASE comporta la necessità per l'utente finale, di acquistare anche la copia del dBASE.

Tra i compilatori quello che ha avuto più successo, soprattutto tra i programmatori, è stato il Clipper della Nantucket, distribuito in Italia dalla Algol.

Si è trattato di un successo crescente cha ha portato il Clipper ad occupare il primo posto come strumento per la programmazione di applicativi di tipo gestionale, soppiantando via via il Basic, che all'inizio non aveva alternative, ma che, come detto, era poco adatto per le applicazioni che lavorano su archivi, e il Cobol, che fu all'inizio molto utilizzato dai programmatori che provenivano dal mondo «mainframe» ma che non è stato mai molto a suo agio sui PC.

E il successo è dimostrato anche dal progressivo distacco del Clipper dal dBASE. Nella prima versione Clipper riconosceva gli statement dBASE e archivi in formato dBASE.

Nella seconda, denominata Summer 87 (ma i file sono datati dicembre 1987), vennero introdotti funzionalità e comandi del tutto autonomi, che ne consentono un uso indipendente dal dBASE. In particolare fu apprezzata la disponibilità di comandi «di rete», presenti direttamente nel pacchetto base. E questo accadeva in un momento in cui hardware e software di rete cominciavano ad essere affidabili e quindi adatti ad accogliere applicazioni gestionali multipostazione.

Altro fenomeno connesso al successo del Clipper fu la proliferazione delle Librerie aggiuntive, che in pratica forniscono direttamente soluzioni ai vari problemi «a fattore comune» che un programmatore Clipper si trova ad affrontare. Ad esempio la gestione del mouse, la generazione di menu pop-down, l'aggancio a routine grafiche.

La disponibilità e la facile reperibilità delle librerie, unita alla possibilità per il programmatore di svilupparne di proprie, anche in linguaggio C, hanno definitivamente consolidato la posizione di predominio del Clipper in quelle software house che sviluppano programmi applicativi per PC o per reti per PC e che lo fanno in maniera «industriale».

Con la nuova versione 5 il distacco dal dBASE si completa.

Il Clipper non solo arricchisce ulteriormente il proprio set di istruzioni e di funzioni, ma ora può addirittura utilizzare file dati organizzati in un formato definito autonomamente e quindi non necessariamente DBF.

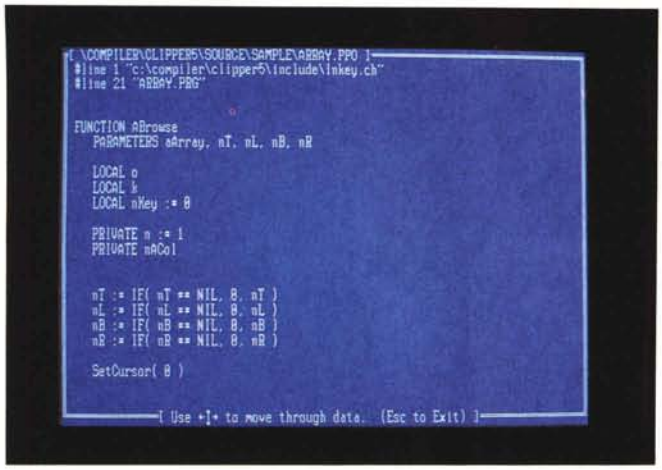
Si tratta quindi di un prodotto ambizioso, e nella prova sono ampiamente descritte le sue caratteristiche, che si vuole scrollare di dosso l'appellativo di «compilatore per il dBASE» che lo ha sempre accompagnato, e vuole vivere di vita autonoma in un settore tecnologico, quello dei DBMS evoluti, dove incontra numerosi e agguerriti concorrenti.

Tra questi citiamo il dBASE IV, giunto alla sospirata versione 1.1, il Paradox 3.5 (ma andiamo per quattro), il DataEase 4.2 (ottimo il suoi multiform), il FoxBase Pro (non molto noto in Italia), per non parlare della valanga dei nuovi DBMS attesi sotto Windows.

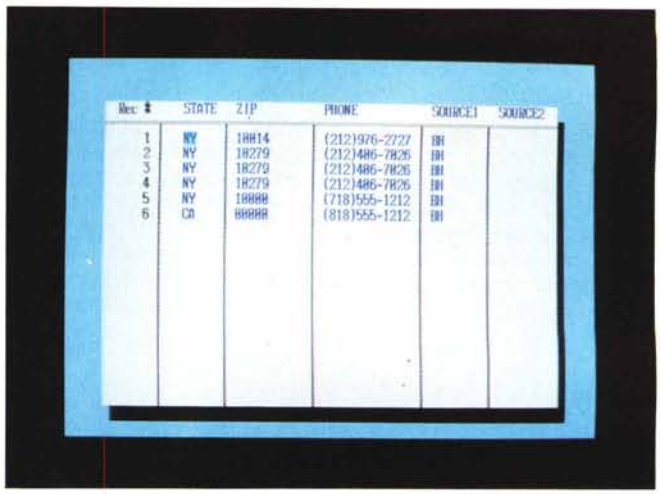
Si tratta di prodotti «double-face» nel senso che dispongono sia di ambienti interattivi, ad uso degli utenti, sia di ambienti programmabili, per gli utenti evoluti o i programmatori, mentre il Clipper si rivolge dichiaratamente solo a questi ultimi.

La nostra prova ha, necessariamente, un taglio tecnico, è cioè rivolta a chi già conosce il «vecchio» Clipper o perlomeno il dBASE.

Data l'importanza del prodotto e l'interesse che suscita tra gli addetti ai lavori, a questa prova seguiranno sia articoli più divulgativi, che partendo dall'inizio spieghino anche ai principianti come programmare con Clipper, sia articoli di taglio tecnico, nei quali vengano suggeriti trucchi e approfonditi temi avanzati.



◀ *Listing di un file di testo.*



Browse di un file DBF.

Nei manuali ad anelli vengono trattati alcuni argomenti specifici del pacchetto; troviamo così:
il «Getting Started», contenente informazioni sull'installazione, sui concetti fondamentali della programmazione in Clipper, sulle novità di questa nuova versione, sulla documentazione on-line e sull'utility DBU per la gestione interattiva degli archivi e degli indici ad essi associati;
il «Programming and Utilities Guide», contenente la documentazione di riferimento del Ccmpilatore, del Linker, del Make e di altre utility fornite nel pacchetto, una trattazione specifica sulla programmazione in rete locale con il Clipper ed una sezione relativa alla programmazione con l'Extend System per l'interfacciamento con routine in C ed Assembler;
il «Clipper Debugger», contenente il manuale di riferimento del nuovo Clipper Debugger;
il «Product and Services Guide», contenente informazioni di vario tipo sui servizi ausiliari offerti dalla Nantucket;
il «Quick Reference», per la consultazione rapida di comandi e funzioni durante lo sviluppo.

L'installazione del pacchetto denota come con questa nuova versione il Clipper sia diventato un vero e proprio sistema di sviluppo, completo delle funzionalità tipiche di questi prodotti; dalla directory unica in cui copiare «a mano» i file contenuti nei vari dischetti siamo passati ad un programma di installazione, che crea una serie di directory in cui vengono memorizzati i vari file dopo opportuna decompressione (i file arrivano infatti compattati in una serie di archivi sui dischetti).

Anche il Clipper, quindi, ha ora bisogno del settaggio di una serie di variabili di ambiente per permettere l'individuazione, da parte di compilatore e linker, delle librerie e degli altri moduli del sistema di sviluppo; proprio questa necessità, unitamente ad altri problemi legati all'environment del Dos, è stata la causa di alcuni malfunzionamenti riscontrati nell'installazione del pacchetto peraltro in via di risoluzione da parte della Algol.

Completata l'installazione troviamo, nella directory prescelta, una serie di subdirectory con nomi abbastanza simili a quelli tipici di un compilatore C; abbiamo infatti una directory BIN per gli eseguibili, una directory INCLUDE per gli header (novità assoluta per il Clipper) ed una directory LIB per le LIBrerie, affiancata da un'altra directory PLL per le Pre-Linked Library di cui parleremo nel seguito.

Passiamo ora ad analizzare le principali componenti di questo pacchetto.

Il compilatore ed il linker

Nei mesi che precedono l'uscita di una nuova versione di un prodotto si intrecciano sempre le congetture su quali possano essere le novità che verranno introdotte; nel caso del Clipper, poi, di cui pregi e difetti erano ben noti, si ipotizzavano modifiche sostanziali soprattutto nel linker, in quanto il Plink86, nella versione specifica per Clipper, aveva da sempre denotato dei limiti sia per quanto riguardava la velocità operativa che per quanto riguardava la gestione della memoria (quest'ultimo argomento vero incubo per gli sviluppatori di applicazioni in rete).

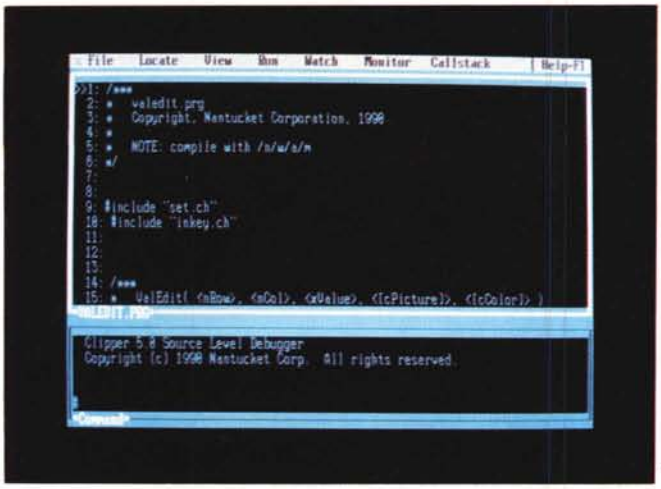
Per la versione 5 la Nantucket si è affidata alla PocketSoft Inc. che ha realizzato una versione specifica per il Clipper del suo linker RTlink; è ora possibile realizzare degli overlay dinamici e caricare in memoria, durante l'esecuzione del programma, soltanto le parti di codice effettivamente necessarie in un dato momento oltre che creare delle librerie di run-time contenenti parti del programma. Queste librerie, dette PLL (Pre-Link Library), possono contenere parti di un programma eseguibile a cui possono accedere anche moduli EXE differenti e vengono create con un processo di link a passi successivi.

È possibile finalmente superare la barriera dei «fatidici» 640 Kbyte in cui il Dos costringe a far girare le applicazioni; questo viene ottenuto sfruttando l'eventuale memoria espansa presente nel sistema (purché conforme alle specifiche LIM) oppure effettuando lo swapping su disco delle parti di applicazione attualmente non necessarie.

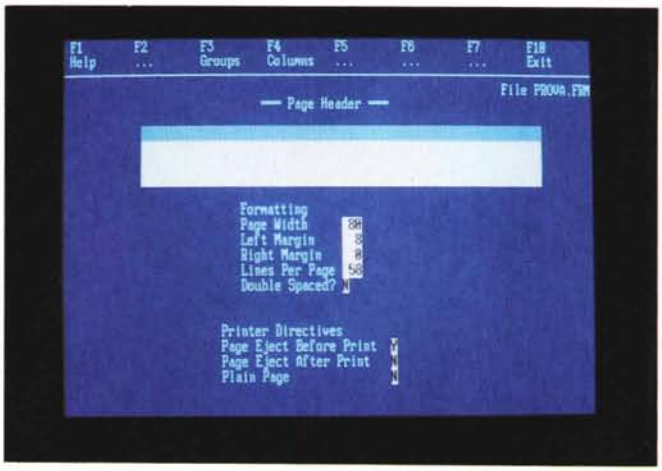
È possibile effettuare un link incrementale del codice prodotto e compilato; in questa modalità operativa il linker controlla quali moduli sono cambiati dall'ultima volta ed opera di conseguenza solo su questi riducendo quindi il tempo necessario a portare a termine l'operazione; un'altra maniera di ridurre il tempo di link è quello di raggruppare le parti di codice già stabilizzate in una PLL da rendere disponibile al programma al momento dell'esecuzione.

Per quanto riguarda la velocità operativa del linker, purtroppo, non è che il nuovo RTlink sia... un fulmine; rispetto a linker quali il Microsoft o il Tlink della Borland la velocità operativa è sensibilmente inferiore, anche perché il linker del Clipper effettua degli «swap» su disco che inevitabilmente rallentano le operazioni (è consigliabile l'impiego di PC con hard disk non troppo lenti per non penalizzare eccessivamente le prestazioni).

Anche il compilatore è stato riprogettato ed ottimizzato, comunque più nella velocità di esecuzione che per quello che riguarda la dimensione dell'eseguibile prodotto; questo perché è stato introdotto l'utilizzo delle PLL che permet-



Il Debugger.



Il Label Generator. ►

te di ridurre a piacimento la dimensione del file EXE contenente il modulo principale del programma.

A questo proposito, per permettere lo sviluppo di applicazioni in rete, viene fornita una versione PLL della libreria Clipper che può essere memorizzata in unica copia sul server mentre nelle singole workstation basta memorizzare soltanto l'eseguibile senza che ad esso venga aggiunto il «fardello» della libreria, producendo quindi un file EXE di dimensioni ridotte.

Ad esempio, il seguente semplice programma:

```
CLEAR
@ 10,10 SAY "CIAO, MONDO !"
QUIT
```

che compilato e linkato nella maniera tradizionale produce un eseguibile di più di 140 Kbyte, se lo compiliamo e linkiamo con l'opzione </PLL> produce un eseguibile di poco più di 7 Kbyte!

Anche per l'utilizzo della PLL contenente la libreria del Clipper il programmatore non è tenuto a pagare alcun diritto alla Nantucket, così come è sempre stato nella filosofia di questo prodotto.

È stata migliorata in maniera sensibile l'ottimizzazione del codice, soprattutto per quanto riguarda parti di codice che possono non venir mai eseguite, e la gestione delle espressioni; inoltre le opzioni che il programmatore utilizza costantemente nella linea di comando del compilatore possono essere memorizzate in una variabile dell'environment.

La nuova versione del compilatore Clipper è compatibile a livello di sorgenti con le precedenti, permettendo in questo modo al programmatore di ricompilare le applicazioni già sviluppate sfruttando le nuove potenzialità offerte dalla versione 5; la stessa compatibilità non è purtroppo assicurata verso le librerie di terze parti (DGE, Artful, ecc...) che richiederanno al programmatore l'upgrade verso le versioni specifiche, che comunque cominciano già ad essere disponibili sul mercato.

In ogni caso la Nantucket è già proiettata verso il futuro che si presenta sempre più distaccato rispetto alla compatibilità finora mantenuta con il dBase della Ashton Tate; ci sono infatti alcuni comandi ed alcune funzioni, contraddistinte nella manualistica da un asterisco, che la Nantucket afferma di aver lasciato solo per compatibilità con la versione precedente del Clipper e di cui sconsiglia l'uso da ora in poi se si vuole che le applicazioni scritte con la versione 5 siano compatibili con quelle future.

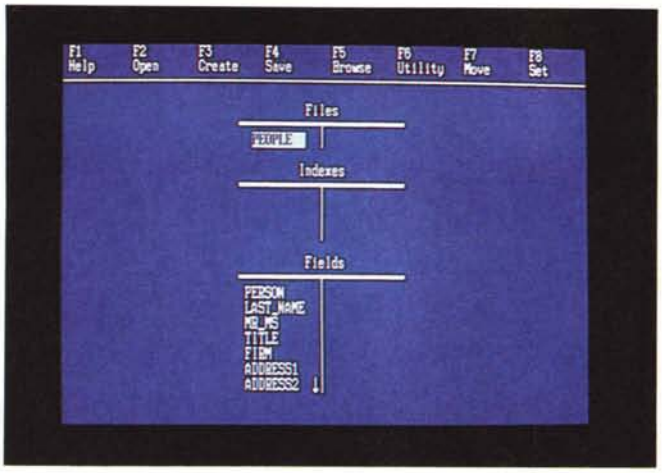
A proposito dei comandi disponibili, una delle più grosse novità di questa versione 5 è la possibilità da parte dell'utente di... ridefinirli; il compilatore, infatti, quando viene lanciato va a cercare un file denominato STD.CH in cui sono definiti tutti i comandi del linguaggio (a livello di sintassi) e l'utente ha la possibilità sia di modificare questo file sia di crearne di nuovi (indicandoli con un'opportuna opzione al compilatore): in questo modo oltre alle User-Function delle precedenti versioni, il programmatore ha a disposizione uno strumento per la creazione di User-Command, strumento da usare comunque con oculatezza (ci vuole poco a trasformare un comando GET in una APPEND) ma che può offrire grosse potenzialità in fase di creazione di applicazioni complesse.

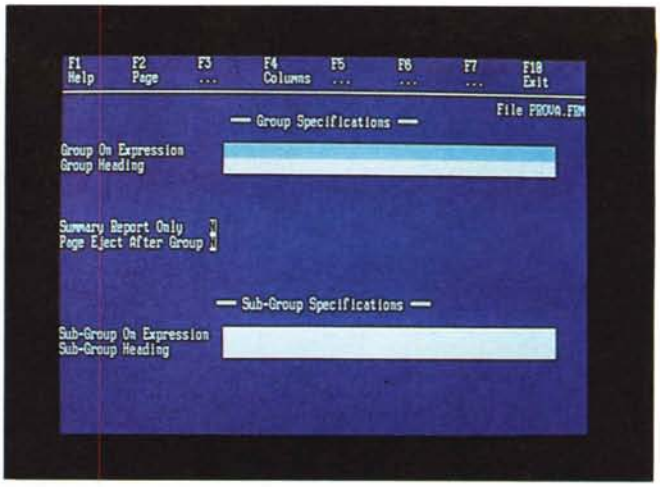
Il file STD.CH è composto da una serie di direttive di compilazione: è questa un'altra delle novità del Clipper 5, che comprende ora un pre-processore simile a quello del linguaggio C; tramite degli «Header-file» (altra analogia con il C) è possibile definire dei nuovi comandi da parte del programmatore (#COMMAND), definire delle costanti (#DEFINE), inglobare un file esterno nel sorgente attualmente in compilazione (#INCLUDE) e compilare in maniera selettiva sezioni di codice al verificarsi o meno di una determinata condizione (#IFDEF...#IFNDEF...).

Il compilatore ha quindi ora due distinte fasi: la fase del pre-processore e la fase del compilatore vero e proprio; nella prima di queste due fasi il pre-processore scandisce il sorgente alla ricerca delle direttive di compilazione e le trasforma in codice Clipper pronto per essere compilato, insieme al resto del programma, nella seconda fase; tramite l'opzione </P> della linea di comando del compilatore è possibile ottenere un file con suffisso .PPO (Pre-Processor Output) contenente l'output della fase di pre-processamento in modo da poter controllare cosa viene passato al modulo che effettua la compilazione vera e propria.

In un sistema di sviluppo come quello del Clipper 5 non poteva mancare un Make, ovvero una utility per la gestione di programmi composti da molti moduli sorgenti, header e librerie; il Make riduce il tempo richiesto per la compilazione di grosse applicazioni in quanto, basandosi su una serie di regole di dipendenze tra file (il cosiddetto make-file), compila di volta in volta solo le parti di codice che sono cambiate dall'ultima compilazione effettuata (oltre a quelle ad esse legate): in congiunzione con la possibilità vista in precedenza del link incrementale per mezzo del RTlink ciò comporta un notevole aiuto al programmatore in termini di risparmio di tempo di lavoro.

RMAKE è il nome del Make fornito con il Clipper 5; è molto simile a quello dello Unix a cui aggiunge delle caratteristiche specifiche per l'ambiente Clipper. Quando viene lanciato opera in due





L'utility DBU con il Browse di un archivio.

fasi: nella prima (parsing) viene analizzato il make-file alla ricerca delle regole di dipendenza da applicare, mentre nella seconda (making) queste regole vengono applicate ai file la cui data (o ora) è cambiata dall'ultima compilazione.

Le utility

Con il Clipper 5 vengono fornite una serie di utility per facilitare la fase di sviluppo e creazione di un'applicazione; alcune di queste sono già note ai programmatori Clipper come, ad esempio, il DBU (Data Base Utility): si tratta di un programma (scritto in Clipper) per la creazione e la gestione interattiva degli archivi dati, degli indici e dei file con le varie «viste» sui dati.

Tramite dei menu a tendina l'utente può selezionare i file su cui intende operare (anche su più aree di lavoro) ed aprirli per la consultazione o la modifica (sia dei dati che delle strutture); è possibile impartire comandi di ricerca (Seek, Locate, Goto, Skip), stabilire relazioni tra file (Set Relation), settare dei filtri (Set Filter) e stabilire una lista di campi del database attualmente selezionato su cui operare.

Oltre che come strumento per la gestione dei file, DBU è utile a chi è al primo approccio con il Clipper come tutorial per l'utilizzo delle funzioni ACHOICE() e DBEDIT() con le quali sono stati realizzati i menu ed il meccanismo di visualizzazione del contenuto degli archivi; in maniera analoga, per chi viene dal mondo dBase ed è abituato a lavorare con le funzionalità CREATE MODIFY REPORT e LABEL, può essere di ausilio l'utility RL (Report and Label), anch'essa scritta in Clipper.

Con RL è possibile creare in maniera interattiva file di report (.FRM) e label (.LBL) da utilizzare poi con i comandi REPORT FORM e LABEL FORM per la generazione di tabulati di stampa o etichette; con un'interfaccia utente simile a quella vista per DBU l'utente può definire i parametri di formattazione della pagina (o dell'etichetta) e definire gruppi per unire informazioni provenienti da uno o più database in relazione tra di loro.

Dell'utility per la documentazione online (Norton Guide) abbiamo già parlato nella descrizione del programma di installazione; rimane da dare solo un cenno al Program Editor (PE), un semplice editor che non ha certo le potenzialità per proporsi come strumento per la stesura di programmi complessi (al massimo per la modifica «al volo» di qualche linea) ma che può essere considerato, dal momento che ne vengono forniti i sorgenti, come un esempio di realizzazione di un semplice editor da inserire nei propri programmi (ad esempio per permettere all'utente finale di modificare il Config.sys) basato sulla funzione MEMOEDIT() del Clipper.

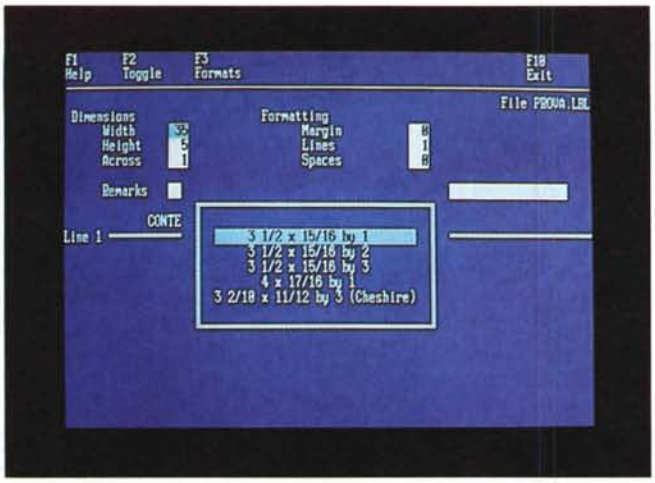
Le altre novità del linguaggio

Il linguaggio Clipper disponeva già di una nutritissima serie di comandi e funzioni che ne consentivano l'uso non solo per lo sviluppo di applicativi per la gestione di basi di dati (suo compito principale) ma anche per altri settori, essendo dotato di tutte le principali strutture di un moderno linguaggio di programmazione; in questa nuova versione sono state aggiunte delle importanti novità, frutto soprattutto del «feedback» ricevuto dal popolo dei programmatori Clipper che spesso lamentavano qualche carenza in alcune componenti del linguaggio.

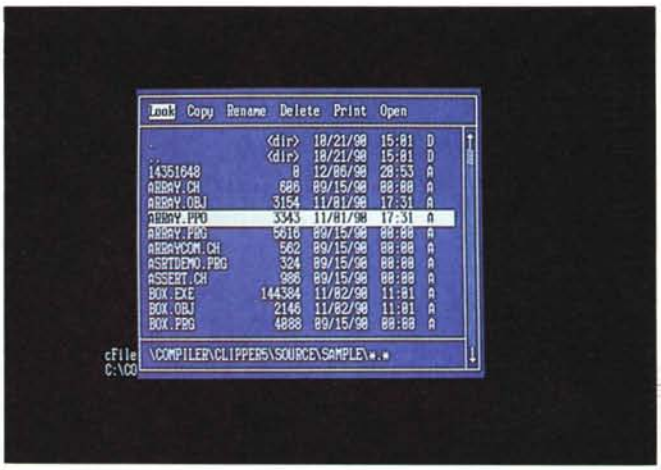
Innanzitutto alcune significative novità sono state introdotte tra gli operatori, che hanno ereditato dal linguaggio C l'assegnazione «in-line» (:=) utilizzabile dove è consentito l'uso di un'espressione, l'incremento ed il decremento (++, --) sia prefisso che postfisso oltre alla possibilità di combinazione degli operatori matematici con quello di assegnazione (+=,-=,*=,/=). Il passaggio dei parametri ad una subroutine può avvenire ora sia per valore che per riferimento (tramite l'operatore @) e non è più necessario che il numero di parametri di una funzione coincida con il numero di argomenti passati (è possibile, ad esempio, chiamare una generica funzione con un comando come TEST(«PAR1»,,,«PAR2»)): il parametro mancante viene sostituito con il valore NIL.

Per quanto riguarda le variabili è stato introdotto il concetto di <lexical scope> in contrapposizione al tradizionale <dynamic scope>; i riferimenti alle variabili che appartengono al primo tipo (locali e statiche) vengono risolti al momento della compilazione rendendole così di accesso più veloce rispetto a quelle del secondo tipo (private e pubbliche). Le variabili locali e statiche, inoltre, sono accessibili soltanto nella procedure in cui vengono dichiarate e mentre le prime «vivono» per tutto il periodo di esistenza dell'entità dichiarante, le seconde esistono per l'intera durata del programma.

Oltre al tipo di dato NIL citato in precedenza (assegnato automaticamente a tutte le variabili non pubbliche non inizializzate) è stato introdotto il Code Block, un tipo di dato che permette di trattare codice compilato come un dato; questi blocchi possono essere assegnati a variabili, passati come parametri e ritornati da funzioni oltre che essere calcolati (cioè eseguito il codice relativo e ritornato il relativo risultato): è un po' il concetto di macro ben noto ai programmatori Clipper ma notevolmente più potente e più veloce in quanto i Code Block



Il Form Generator.



Selezione di file. ►

vengono compilati non come le macro durante l'esecuzione del programma ma al momento stesso della compilazione di quest'ultimo.

Gli array, con il Clipper 5, possono ora essere anche multidimensionali, nidificati (array di array), possono essere gestiti per riferimento permettendo a più variabili di riferirsi allo stesso array (l'area di memoria occupata dall'array viene deallocata soltanto quando l'ultimo riferimento ad esso cessa di esistere) e possono essere dimensionati dinamicamente permettendo così la gestione di strutture di dati quali liste e code; per la inizializzazione e la creazione degli array possono ora essere utilizzate delle costanti (anche di tipo carattere) e nel caso di funzioni definite dall'utente gli array possono essere ritornati come risultato.

Tra le operazioni possibili sugli array vi segnaliamo la possibilità di metterli a confronto con l'operatore di uguaglianza (=), scandirli con la funzione AEVAL() che calcola un Code Block per ogni elemento, duplicarli, copiarli ed ordinarli (con la funzione ASORT()); un determinato valore può essere ricercato in un array con la funzione ASCAN().

Nella gestione dei file è stato fornito al programmatore un nuovo strumento che permette di svincolarsi, volendo, dalla struttura dei .DBF retaggio del dBase; è infatti ora possibile sviluppare dei propri driver tramite i quali gestire, in maniera trasparente, diversi «engine» di interfaccia verso formati nuovi e definibili dall'utente.

Una importante innovazione del Clipper 5, sebbene in forma ancora limitata, è l'introduzione della programmazione per oggetti; come si affretta a precisare il manuale, nel capitolo relativo a questo argomento, il Clipper 5 non è un linguaggio object-oriented, ma ha al suo interno una implementazione di strutture basate su questa teoria (si vocifera comunque di successive versioni del Clipper specifiche per OOP).

Un <oggetto>, in Clipper, è un insieme complesso di dati con una struttura predefinita e caratteristiche comuni; con questa versione del compilatore vengono forniti un numero limitato di questi oggetti, dette <classi>, realizzati allo scopo di supportare particolari operazioni del Clipper (non e purtroppo possibile definirne di nuovi).

Il tipo di un oggetto è formalmente la sua classe e le informazioni contenute in un oggetto, insieme alle operazioni ad esso applicabili, variano in funzione della classe dell'oggetto; per ogni classe è possibile, tramite una speciale funzione di creazione, generare nuovi oggetti (istanza della classe) che possono essere manipolati per riferimento (come gli array).

Abbiamo visto che ogni classe definisce un insieme di operazioni possibili sui propri oggetti; queste operazioni vengono eseguite tramite l'invio di messaggi utilizzando l'operatore <:>; ad esempio, per inviare un messaggio all'oggetto «Generic» la sintassi è:

```
Generic:PageDn()
```

In questo caso «Generic» è il nome della variabile che contiene il riferimento all'oggetto, mentre «PageDn()» è il <selettore>, cioè l'operatore che specifica l'operazione che deve essere eseguita sull'oggetto (in questo caso uno scroll di una pagina).

Le classi predefinite presenti in questa versione sono:

— Error Class, che fornisce oggetti contenenti informazioni sugli errori di runtime;

— Get Class, che fornisce oggetti per l'editing interattivo di campi di database e variabili (può essere utilizzato per implementare i comandi @...GET e READ);

— TBrowse Class, che fornisce oggetti per la visualizzazione e consultazione di dati in forma tabellare (con l'utilizzo di code-block è possibile creare degli strumenti per l'acquisizione, consultazione e gestione interattiva di insiemi di dati);

— TBColumn Class, che fornisce oggetti a «colonna» per oggetti TBrowse, ovvero oggetti contenenti le informazioni necessarie a definire una colonna di dati di un oggetto TBrowse.

Tramite queste classi (e relativi oggetti) il lavoro del programmatore viene enormemente semplificato permettendo, una volta acquisita una certa dimestichezza con queste nuove strutture di dati, di creare interfacce utenti complesse con poco sforzo.

In conclusione un accenno all'Extend System, lo strumento tramite il quale è possibile interfacciare routine in C ed Assembler ai propri programmi Clipper; ha mantenuto la potenza ed efficacia della precedente versione con una importante novità: sono state aggiunte nuove funzioni che permettono di riassegnare variabili di memoria passate per riferimento da un programma Clipper ad una routine C o Assembler.

Il debugger

Il debugger del Clipper 5 viene trattato in questa specifica sezione, vuoi per l'importanza che ormai ha assunto uno strumento di questo tipo nell'attività quotidiana di un programmatore, vuoi perché si tratta di un elemento che ha subito sostanziali modifiche rispetto alle versioni precedenti, adeguandosi allo standard di prestazioni tipiche di prodotti per lo sviluppo di applicativi.

Con questo debugger non si raggiungono certo le potenzialità di un Codeview o di un Turbo Debugger ma rispet-

to a quel semplice strumento offerto con la precedente versione il passo in avanti è certamente notevole; ora il debugger è un programma separato, in grado di processare file .EXE prodotti con l'opzione </B> del compilatore, attraverso il quale è possibile seguire passo passo la sequenza di istruzioni eseguite, visualizzare in un'apposita finestra il contenuto di variabili (o anche crearne di nuove durante l'esecuzione), eseguire procedure o funzioni definite dall'utente (purché compilate e linkate nell'applicativo corrente) ed esaminare la struttura di un database.

Una volta lanciato il debugger (con il comando CLD seguito dal nome del file eseguibile) lo schermo appare diviso in due finestre: la prima è la Code Window e viene utilizzata per la visualizzazione del codice mentre la seconda è la Command Window tramite la quale è possibile impartire i comandi al debugger (è possibile spostare la linea di demarcazione tra queste due finestre in modo da rendere visibile una parte maggiore di una delle due); nella parte alta dello schermo una serie di menu a tendina permettono di impartire i principali comandi senza doverli digitare sulla linea di comando.

Tramite i tasti funzione (la cui disposizione dei comandi ricorda quella del Codeview) è possibile eseguire il programma una istruzione alla volta oppure in modalità <Animate> (esecuzione delle istruzioni in maniera consecutiva con un certo intervallo di tempo tra una e l'altra), settare dei Breakpoint su linee di programma in cui il debugger deve fermarsi per permettere un'esecuzione passo passo, settare dei Watchpoint o dei Tracepoint su cui il debugger deve fermarsi al variare del valore di una variabile o di un'espressione; è inoltre possibile definire dei Passpoint, ovvero dei contatori indicanti il numero di volte che una data linea (istruzione) viene eseguita, cosa che risulta molto utile per il controllo di loop e cicli nidificati.

Una grossa comodità offerta all'utente è quella di poter definire degli <script> in cui memorizzare la situazione attuale dei settaggi relativi a Breakpoint, Watchpoint e Tracepoint di un programma in fase di debug, in modo da poter ripristinare la situazione corrente, nella successiva sessione di lavoro con lo stesso programma, tramite il comando INPUT.

Purtroppo la stessa comodità operativa non è stata prevista per la visualizzazione delle finestre che sono non sovrapponibili; si può avere qualche problema infatti quando, oltre alle due finestre viste in precedenza, si decide di aprire anche la Watch Window (per i Watchpoint ed i Tracepoint) e la Callstack Window che visualizza nella parte destra dello schermo la sequenza delle funzioni chiamate fino a quel momento: in questa situazione operativa non rimane che espandere a tutto schermo le singole finestre (tramite il tasto F2) per una consultazione migliore ma purtroppo limitata ad un solo elemento tra quelli componenti la sessione di debug.

Direttamente in modalità full-screen viene invece visualizzata la Status Window, dedicata alla consultazione delle aree di lavoro attualmente attive unitamente ai file DBF in esse aperti.

Tutto questo stando a quanto riportato dalla manualistica; se invece ci si diverte a «trafficare» un po' con i file si scopre che:

— sulla linea di comando si possono definire degli switch per lavorare a 43 (EGA) o 50 (VGA) linee e richiamare in automatico un file contenente lo script desiderato;

— nella Status Window viene visualizzata la sola struttura del file di database attualmente selezionato e non c'è modo di ottenere la lista dei settaggi come previsto dai manuali;

— nel passare da una all'altra delle Window richiamabili, la Command Window viene «sporcata» dal contenuto dell'ultima finestra aperta e ritorna al suo stato originario solo dopo aver premuto varie volte il tasto Enter in modo da far «affiorare» linee pulite (un problema analogo si presenta anche per le finestre della Status Window).

Le divergenze riscontrate non sono segnalate neanche nel file Readme dove invece si parla della possibilità di visualizzare il contenuto di un array nella Watch Window semplicemente evidenziandone il nome e premendo il tasto Enter.

Comunque il debugger si è mostrato all'altezza del pacchetto, fornendo finalmente uno strumento efficiente al programmatore che, se vuole, può anche integrarlo nel programma eseguibile collegando con il linker la libreria CLD.LIB e richiamandolo tramite la funzione ALTD(); come ulteriore possibilità si può lanciare il programma EXE (purché compilato con l'opzione </B>) e richiamare il debugger durante il normale funzionamento tramite la «vecchia» combinazione di tasti ALT-D.

Conclusioni

Sicuramente il Clipper, con questa nuova release, ha assunto una veste marcatamente professionale e specialistica: non segue infatti la filosofia di alcuni suoi concorrenti nel campo dei DBMS per Ms-Dos quali il FoxBase o il Paradox, in cui ad un ambiente di lavoro con interfaccia utente precostituita viene aggiunto un linguaggio di programmazione per la realizzazione di proprie applicazioni custom, ma piuttosto si allinea alla categoria dei compilatori per specifici linguaggi.

La sua origine «C» è stata maggiormente marcata dall'introduzione di una serie di costrutti tipici di questo linguaggio e le nuove potenzialità offerte dal compilatore e dal linker richiedono una certa dimestichezza con le nozioni base della gestione della memoria in ambiente Ms-Dos; chi si aspettava le nuove versioni di questi due moduli in veste ... «turbo» rimarrà probabilmente deluso è sicuramente preferibile spendere qualche attimo in più durante la fase di realizzazione di un programma avendo come contropartita un sistema per la gestione della memoria e degli overlay adeguato alle esigenze di oggi, dove i 640 Kbyte di RAM sono ormai «acqua fresca» per un PC.

La manualistica è corposa ma presenta alcuni disallineamenti, peraltro risolti nella versione per Norton Guide della documentazione, sicuramente più facilmente manipolabile dal programmatore.

Il prodotto è sicuramente all'avanguardia per quanto riguarda la realizzazione di sistemi per DBMS in ambiente Ms-Dos e questa nuova versione del compilatore non potrà che rafforzare la forza e l'influenza della Nantucket in questo mercato.

Ci sono comunque delle cose di questo Clipper 5 che ci hanno fatto un po' riflettere.

L'affermazione dello standard Clipper (perché di questo ormai si può parlare) ha fatto sì che proliferasse il mercato delle librerie di terze parti che colmassero i vuoti del linguaggio (grafica, comunicazioni, ecc...).

La Nantucket ha invece inaspettatamente adottato lo spirito del «se c'è già qualcuno che si preoccupa di fare una certa cosa perché preoccuparmi di farla anch'io ?» riproponendo anche con questa nuova versione il problema degli add-on, che costringe l'acquirente a doversi sobbarcare una doppia spesa (compilatore + add-on).

Per chi ha già un consolidato di applicazioni scritte in Clipper il problema probabilmente non si pone ma per chi ne deve sviluppare di nuove la problematica è completamente differente, anche perché a questo punto i concorrenti del Clipper, anche dal punto di vista economico, si allargano a dismisura.

I prossimi mesi ci diranno quali di questi fattori avrà maggiormente influenzato le scelte del mercato; per il momento godiamoci le novità del Clipper 5 e togliamoci finalmente la soddisfazione di far girare quel programma che avevamo buttato in un cassetto perché ogni volta ci diceva «Not enough memory»!.

MC

AXXON
PROPONE

STUDIOSEGRE

PICCOLI, MICRO, VELOCISSIMI

CONNER

HARD DISK

La vasta gamma di hard disk da 3" 1/2
prodotta dalla CONNER è distribuita in tutta
Italia dalla AXXON.
Molto affidabili e veloci, questi hard disk
progettati per workstation, PC e laptop
presentano caratteristiche di notevole rilievo:

- capacità: 20 - 40 - 80 - 100 e 200 Mb
- tempo-medio di accesso: da 24 a 16 millisecondi
- consumi bassissimi: 4,2 a 2,9 watt
- sopportano urti fino a 50 G.

Sono disponibili modelli con interfaccia PC/AT
oppure SCSI embedded.

CHI CERCA CONNER TROVA AXXON

BERGAMO e **BRESCIA - AXXON** spa
Tel. (02) 95.30.06.31
BOVISO MASCIAGO (MI) - ARTAX srl
Tel. (0362) 55.87.61
TORINO - MI.RA.MA sas
Tel. (011) 27.35.561
PADOVA - ELCOM srl
Tel. (049) 80.70.319
PIEVE MODOLENA (RE) - DE PIETRI
Tel. (0522) 79.26.94
FIRENZE - T.D.A. sas
Tel. (02) 95.300.631
AMELIA (TR) - MARIO GARDANO
Tel. (0744) 98.37.91
MOGLIANO (MC) - SYSTEM HOUSE ELIA
Tel. (0733) 29.27.76
ROMA - ADDEL srl
Tel. (06) 60.95.881
NAPOLI - EXPO TRADING CO sas
Tel. (081) 68.20.39
BARI - DBS Sist. Inf. Aziendali
Tel. (080) 22.84.30
CATANIA - ADVANTAGE srl
Tel. (095) 22.11.80

ASEM GROUP

FUTURO PRESENTE AXXON

AXXON spa
Via Roma, 108
20060 Cassina De' Pecchi (MI)
Tel. (02) 95.30.06.31 - Fax (02) 95.30.07.21