

THE CLARION® Call

VOL. 1 NO. 1 BARRINGTON SYSTEMS INC., POMPANO BEACH, FLORIDA SUMMER 1987

CLARION CHECKS IN AT THE WORLD'S LARGEST RESORT HOTEL

Guests at the Las Vegas Hilton may be in a gambling mood, but management doesn't want to gamble with their comfort.

Maintenance of 3400 guest rooms and the giant convention facilities could be a hassle, but a CLARION-developed management system helps keep the magnificent inn in top shape.

Systems Associates of Bowling Green, Ohio, recently installed its network-based maintenance management system that provides record keeping and active control of over 100,000 work orders and trip tickets, 12,000 inventory parts, 5,000 pieces of equipment, 15,000 other items, and personnel data for over 500 employees.

A leader in software development for the energy management field, Systems Associates chose CLARION to develop its SAI5000 Maintenance Management System (MMS) for the Hilton Corporation. MMS, used with the company's Energy 2.0C Energy Management System, provides a complete answer to the needs of a typical hotel engineering department.

Major functions provided in MMS include:

- work order and trip ticket creation and control
- preventive maintenance scheduling and tracking
- parts and equipment inventory management
- job costing
- equipment histories
- personnel data control

See "LAS VEGAS HILTON" continued on page 2



CLARION APPLICATIONS NOW READ AND WRITE dBASE III FILES

Want to create applications that end-users love? Can't because you have to keep a file in dBase III format?

Not anymore.

Now application programs written in CLARION can read and write dBase III files.

Here's what that means to you.

- Your CLARION programs can use a dBase III file that is simultaneously compatible with dBase language programs.

- You can convert your dBase III application programs to CLARION, one at a time.

- You can use the Translator option

to produce CLARION .EXE programs that formerly required a dBase III program and its run-time system and license to produce.

In short, you can still enjoy all of the productivity gains and attractive

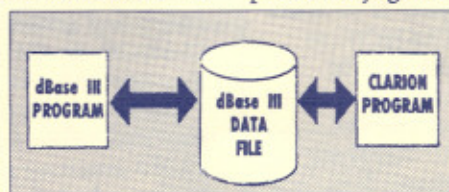
end-user features of CLARION, even if it's not practical to convert a dBase III file to CLARION right now.

Bruce Barrington,

founder and CEO of Barrington Systems, explains:

"This isn't just another dBase add-on. What we are announcing is a new strategy for the dBase user who

See "DATA BASE THREE LEM" continued on page 8



APPLICATION PROTOTYPING OPENS THE DOOR, CUTS COSTS, FOR TECHNOLOGICS

"Using the Screener, we were able to prototype the entire application in a few days and let our client see it before we began serious coding. We have been able to install segments of the application by leaving the prototype stubs in place and adding functions as they become available."

This is how Roger C. Jones, President of TechniLogics of Gresham, Oregon, describes a key benefit of his company's use of CLARION.

TechniLogics, a software development and mainframe consulting firm, reports excellent reviews for TL/RENTS, its new inventory management system for the equipment rental and sales industry.

Jones says, "Originally, we were going to use C because no development tools were on the market that would satisfy our screen and database handling requirements without

unacceptable expense to us and to our customers. Thanks to CLARION's development tools and transportable run-time system, we have reduced our development time and cost by a conservative 50 percent over the C approach."

TL/RENTS features "action keys" for entering data, pop-up windows, menu selection, and help facilities. The action keys dramatically reduce keystrokes and achieve keyboard simplicity. The system contains dozens of screens, windows, menus, and a "second normal form" data base.

The use of MENU screen types rather than ENTRY types allows users to point to valid values; little or no logical editing is necessary.

The F2 key has been defined as the "quick entry" key. This

key fills in all default values known to a transaction.

Screens contain limited amounts of logically-grouped information to prevent a cluttered appearance. When a transaction contains more than one screen of information, windows present the next logical data group, either automatically or on command. The F9 and F10 keys page forward and backward through windows.

Jones explains further: "If a required field is left empty, a pop-up window allows the user to scroll through a related file and find the needed information. For example, suppose you

want to enter a contract to rent a piece of equipment. You know the customer has rented from you before, but you don't know anything more than the customer's name. Skipping over the customer number and entering the first few

characters of the customer's name in the name field causes the system to present a window with several customer records close to the customer's name. When you find the right customer, one control key fills in the contract with all of the customer's information."

The HELP facility provides on-line documentation. The user only needs to know about the F1 key to get started.

Jones also expressed appreciation for the support received from the technical staff at Barrington Systems. He says, "Since we are a relatively new user, we have had the usual learning curves to climb over. We have received excellent support from your technical staff during our startup. We definitely appreciate their help." ☞



THE CLARION[®] Call

The CLARION Call is a periodic newsletter published for CLARION users by Barrington Systems, Inc., 150 East Sample Road, Pompano Beach, Florida 33064. Its purpose is to share techniques for using CLARION, to announce enhancements, and to provide a forum for CLARION users to exchange application experiences.

If you would like to contribute to The CLARION Call, please drop us a line and include an application programming story or tip you think others will find helpful. ☞

© 1987 Barrington Systems, Inc.
CLARION is a registered trademark of Barrington Systems, Inc.
THE CLARION Call is a trademark of Barrington Systems, Inc.

LAS VEGAS HILTON (Continued from front page)

According to Steven W. Shurts, Project Engineer, the maintenance system had two major design criteria. The system had to have the same "look and feel" as the existing energy application, and it had to have instant access over a network to large files.

Shurts says, "We found that, with CLARION, we could accomplish all of our desired goals plus some that we hadn't even thought of at the time." He adds that the first requirement is more than satisfied by CLARION's extensive screen design and generation capabilities — "Some of the really strong features of CLARION are the pop-up help screens, the ability to define keys, [and] the ability to create and edit reports..."

As for the second requirement, processing speed, several data base products were evaluated before CLARION was chosen. Shurts continues, "We have a file of over 12,000 inventory parts and can access any record off the network instantaneously. We have had the manager of MIS at a very large client comment very favorably on the speed of the system when compared to his mainframe system. Response time is under two seconds for all applications."

Shurts is pleased with the reception that MMS is getting from his prominent client.

At this hotel, thanks to Systems Associates and to CLARION, energy management worries and maintenance problems check out fast. ☞

John Herron, a product designer, has thirteen years programming and design experience including management experience at Peachtree Software. Herron is a specialist on keyboard/user interfaces and screen processing techniques.



BEHIND THE SCREENS

by John Herron

Face it. Beauty is only screen deep. Your user may spend hours looking at a screen that, in reality, is doing nothing, while your program is busy doing some sophisticated file handling or complex math calculations. More often than not, an opinion about the quality of your entire program is based only on what the user sees.

The following tips will help you create professional screens worthy of your CLARION application programs.

Horizontal Scrolling

Don't feel obligated to limit the size of a field to the number of columns available on the screen. CLARION supports fields that are up to 255 characters long simply by allowing a field to scroll horizontally. All you have to do is specify the size of the field and the size of the data area that will receive the field input. The rest is handled by CLARION. Note the following example:

```

.
.
OMIT('***-END-***')
.
Company Name: *****
.
** -END- **
.
.
ROW(3,6)  STRING('Company Name:')
COL(20)  ENTRY(2522),USE(CO_NAME)
.
.
CO_NAME  STRING(40)
.
.

```

If the company name BARRINGTON SYSTEMS, INC. is typed into the previous field, it looks like this:

```

BARRINGTON SYSTEMS, IN      after the first 22 characters have been typed.
ARRINGTON SYSTEMS, INC     after the 23rd character is typed, and
RRINGTON SYSTEMS, INC.    like this after the period is typed.

```

Later, if the field is initialized, the first 22 characters are displayed.

The Update Procedure

The UPDATE procedure is very handy when you need to get all the current information from the screen into the USE variables. Suppose you are on a network and you need to re-read and change a record from a file after all changes have been entered and verified. You can read the record, do an UPDATE, and then put the record back into the file.

Since the purpose of UPDATE is to update the record from the screen area, you should not use UPDATE for fields that are larger than the allocated screen area (as described in Horizontal Scrolling). For larger fields, you can accomplish the same thing by using a temporary variable for the USE variable and then copying the temporary variable to the record field before putting the record back into the file.

In the following example, you can see that fields A, B, D, and E are all 10 bytes long and that field C is 40 bytes long. The field areas described on the screen, however, are all 10 bytes long. The statements shown in the CODE section below will UPDATE the record with current information from the screen.

```

FILE
RECORD
A      STRING(10)
B      STRING(10)
C      STRING(40)
D      STRING(10)
E      STRING(10)
.
.
SCREEN
ENTRY(2510),USE(A)
ENTRY(2510),USE(B)
ENTRY(2510),USE(LOCAL_C)
ENTRY(2510),USE(D)
ENTRY(2510),USE(E)
.
LOCAL_C  STRING(40)
.
CODE
GET(RECORD)          IGET RECORD DO DISPLAY ON SCREEN
LOCAL_C = C          IMOVE FIELD C TO LOCAL VARIABLE
DISPLAY              IDISPLAY ALL FIELDS ON SCREEN
.
.
IFIELD EDITING DONE HERE
.
GET(RECORD)          IGET RECORD FOR UPDATE
UPDATE(7A,7B)        IUPDATE FIELDS A AND B FROM SCREEN
C = LOCAL_C          IUPDATE FIELD C WITH LOCAL VARIABLE
UPDATE(7D,7E)        IUPDATE FIELDS D AND E FROM SCREEN
PUT(RECORD)          IUPDATE FILE WITH NEW DATA

```

Overlay Prompts and Fields

Feel free to overlay prompts and fields. Overlaying allows you to stack questions and answers while retaining the ability to back up to a previous question. The user can press the Esc key to back up through the fields. In the following example, dummy fields are used to gain control between fields when the user presses Esc. Notice the ESC attribute on the two dummy fields. Without this attribute, CLARION will skip over the same fields going backward that it skipped over going forward.

```

.
.
.
PROMPT  ROW(6,10)  STRING(14)
COL(25)  ENTRY(2520),USE(NAME)
COL(25)  ENTRY,USE(7DUMMY1)
COL(25)  ENTRY(2520),USE(ADDRESS),ESC(7DUMMY1)
COL(25)  ENTRY,USE(7DUMMY2)
COL(25)  ENTRY(2520),USE(CITY),ESC(7DUMMY2)
.
.
NAME     STRING(20)
ADDRESS  STRING(20)
CITY     STRING(20)

```

(Continued on next page)


```

CODE
OPEN(SCREEN)
PROMPT = 'Enter Name  : '
LOOP
ACCEPT
CASE FIELD( )
OF ?NAME
OROF ?DUMMY2
  PROMPT = 'Enter Address: '
  SELECT(?ADDRESS)
OF ?DUMMY1
  PROMPT = 'Enter Name  : '
  SELECT(?NAME)
OF ?ADDRESS
  PROMPT = 'Enter City  : '
  SELECT(?CITY)

```

Highlighting a Menu Item

CLARION automatically highlights a menu item when a user presses the first letter of the item. If two or more items start with the same letter, the one closest to the current item is highlighted. The user can then press Enter to actually select the highlighted item.

The user can also move among the Menu choices with the cursor arrow keys. 

Gary Liming, a product designer, has thirteen years programming and design experience including management experience at Digital Equipment Corporation. Liming is a specialist on data base management, communications, and computer hardware.



MAKING MEMORY SERVE YOU RIGHT

by Gary Liming

There is a point in the application development cycle when your application is bug-free and works correctly, but you wonder if you've done everything you can to make the application run as efficiently as possible. Although there are many ways to fine-tune an application, some of the simplest and most dramatic involve file processing statements. This article explores two sets of file processing statements that can help you noticeably speed up your CLARION files.

STREAM and FLUSH

The first set includes the STREAM and FLUSH statements. These two statements are used to toggle on and off a condition that guarantees the data integrity of your file. To understand these statements, let's first take a look at the nature of disk I/O.

When a record is added to a file, several things must happen. The data must be written out to disk, and the directory that keeps track of that data's position must be updated. Because disk reads and writes are done in fixed length blocks, DOS buffers this process in memory. The amount of buffer area is set in your CONFIG.SYS file at boot time. Consequently, your data and directory information remain in memory until a full disk block can be written out. Because this usually results in fewer I/O operations, file processing is faster. The bad news is that it leaves your data vulnerable to a power failure.

Consider the following scenario. Your application program

has just added several new records to a file, but the directory that keeps track of the positions of the new records is still in memory. Now, suppose a power failure occurs, or the operator inadvertently turns off the power. Then the positions of the new records are lost and, conceivably, the entire file becomes unusable.

This is why CLARION, by default, writes out all information currently in memory to complete a file operation. This tends to slow down file processing, but your records are safe.

What if your application needs the speed and you don't care about the power failure problem? Say, for instance, you have an application that does file processing in batch mode. If a power failure occurs you can always rerun the application. Or perhaps power failures are rare in your area. For cases like these, CLARION's STREAM statement allows you to toggle on the buffering feature that DOS provides.

You can then use the FLUSH statement to terminate the STREAM condition, in essence toggling CLARION back into its default mode. The FLUSH statement writes out all of the data in a file whenever that file changes (after an ADD, PUT, or DELETE). Using the STREAM statement can decrease the amount of time it takes to process a file by quite a bit.

BUFFER and CACHE

The other set of file statements includes BUFFER and CACHE. Both of these statements speed up processing time by utilizing free memory to hold records or keys. The buffers or caches created by these statements are "write-through" in nature, which means that if the record currently being written to the file is in memory as well as on disk, the memory version is updated as well.

Use the BUFFER statement to allocate part or all of free memory reserved for the next read operation. For instance, if you create a buffer for 500 records and then open a file with 900 records, 500 records will be read into the buffer with the first read operation. From this point, a read operation that requests one of the records in the buffer will not need to use disk I/O to retrieve that record. After the first 500 records have been processed, a read operation for the 501st record causes the next 400 records to be read into the buffer. Because 500 records read all at once is faster than one record read 500 times, processing time is decreased.

Note: Use the BUFFER statement only when processing a file in sequential record order, e.g., using SET with NEXT or PREVIOUS. Do not use the BUFFER statement with keyed access.

The CACHE statement is similar to the BUFFER statement except that it reads specific records into memory and does not refill the buffer once those records have been processed. If the requested record is in the buffer, no disk activity needs to occur. If not, it reads the disk after searching the buffer.

CACHEing records will also significantly speed up file processing in sequential order, but it could slightly slow down the processing of volatile files. For example, say you have a file with 750 records, the first 500 of which are cached. If most of your file activities involve adding new records to the end of the file and reading and updating those records, checking the cache for the new records turns into unnecessary overhead; however, if you have a control file of 100 records, you may cache

all of these records into memory. This effectively places the file in a "RAM disk" for record retrieval.

The CACHE statement can also be called in keyed mode which, in effect, stores keys instead of records in memory. During a key access to a file, the key file is searched for a match. A pointer is then used to retrieve the desired record. CACHEing keys, then, reduces the number of disk accesses involved in searching the key file. The overhead associated with keyed retrieval can be overcome by CACHEing the entire key file, provided you have enough room in memory. If you have room for the records as well, you can do keyed record retrieval with no disk operations. This is ideal for applications with small, frequently accessed files.

One more consideration about the BUFFER and CACHE statements: the buffers and caches may shrink as your application runs. If free memory is required for a memory table and none is available, any memory used for buffers will be freed until the table is satisfied. If that is not enough, or if there are no buffers, record cache memory is freed and then key cache memory is freed. Once the memory table has been freed, buffers and caches can be restored by executing the BUFFER or CACHE statement again.

The BUFFER, CACHE, STREAM, and FLUSH statements provide you with the ability to write programs that can automatically adapt to a variety of environments and allow you to make your own trade-offs between performance and data integrity. Best of all, the effects of these statements can be measured simply and quickly by adding or deleting a single line of code. ☞

Monty Shaw, who has ten years programming and design experience, is the lead product designer at Barrington Systems. Shaw is a specialist on operating systems and microcomputer internals.



ENTERING DIFFERENT DATE FORMATS

by Monty Shaw

You can use the CLARION ENTRY statement to input two date formats, MM/DD/YY and MM/DD/YYYY. You can also use the picture token @p<<. <<. <<p to input the popular international format, DD.MM.YY.

To input the DD.MM.YY format, use @p<<. <<. <<p for the ENTRY, a GROUP as the USE variable, and another GROUP to rearrange the month and the day. Then call DEFORMAT. If DEFORMAT returns zero, the date entered is invalid.

In the following sample program, the function EURODATE is used to convert from a DD.MM.YY format to a CLARION standard date format.

```

1          PROGRAM
2
3 1        MAP
4 2        PROC(MAIN)
5 2        FUNC(EURODATE),LONG
6 2
7 1        CODE
8 1        MAIN
9
10 1       MAIN PROCEDURE
11

```

```

12 1       DATESCR      SCREEN      HUE(7,1,1)
13 2         ROW(4,32)  STRING('Test European Date')
14 2         ROW(12,32) STRING('CLARION date:')
15 2         ROW(10,30) STRING('Enter DD.MM.YY:')
16 2         COL(45)    ENTRY(@p<<. <<. <<p),USE(EDATE),1MM
17 2       CDATE       ROW(12,45)  STRING(@D1)
18 2       MSG         ROW(7,31)   STRING(20),ENH,BLK
19 2
20
21 1       EDATE GROUP
22 2       EDAY        STRING(2)
23 2         STRING(1)
24 2       EMON        STRING(2)
25 2         STRING(1)
26 2       EYEAR       STRING(2)
27 2
28
29 1       USDATE LONG
30
31 1       CODE
32
33 1       OPEN(DATESCR)
34
35 1       LOOP
36 2         ACCEPT
37 2         MSG = ''
38 2         USDATE = EURODATE(EDATE)
39 2         IF USDATE = 0
40 3           MSG = 'INVALID DATE'
41 3           MSG = CENTER(MSG)
42 3
43 2         CDATE = USDATE
44 2
45
46 1       EURODATE FUNCTION(EDATE)
47 1       EDATE GROUP
48 2       EDAY        STRING(2)
49 2         STRING(1)
50 2       EMON        STRING(2)
51 2         STRING(1)
52 2       EYEAR       STRING(2)
53 2
54 1       VDATE GROUP
55 2       VNON        STRING(2)
56 2         STRING('/')
57 2       VDAY        STRING(2)
58 2         STRING('/')
59 2       VYEAR       STRING(2)
60 2
61 1       CODE
62
63 1       VNON = EMON
64 1       VDAY = EDAY
65 1       VYEAR = EYEAR
66 1       RETURN(DEFORMAT(VDATE,@D1))
67

```

A \$195 "THANK YOU" OFFER

CLARION has just completed its first full year in the marketplace and has been a smashing success because of the help and enthusiasm of our early customers. Many of you have offered positive and constructive feed-back that has been incorporated into updates, LEMs, Translator, and the new *Getting Started* Guide.

Also, you have been most generous in sharing your CLARION results with prospective CLARION users — and you are our best sales force!

As an expression of our appreciation, we are offering an additional copy of CLARION with Translator for the special price of \$295.00 to all customers on record as of August 1, 1987 — a \$195 "Thank You" for making CLARION such a success.

Just call 1-800-354-5444 or drop us a letter, and we will have this additional copy on the way to you. This offer is open through August 31st, so if you're ready for your next copy of CLARION, now's the time to call. ☞

TECHNICAL SUPPORT QUESTIONS AND ANSWERS

Technical Support Questions and Answers is a regular feature of The CLARION Call. In each issue, we will publish our Technical Support Group's answers to questions received by phone from CLARION programmers.

* Why do I get a PLINK 86 error when trying to load a utility?

CLARION is not in the path. Add the CLARION directory to your PATH statement in your AUTOEXEC.BAT file. For example:

```
PATH C:\CLARION;C:\DOS;C:\COMMANDS
```

* How can I position a pop-up window?

Pop-up windows "float" based on the last accessed location on the screen. Use the SHOW command with a null parameter to set the position that is to be the upper left corner of the pop-up. The SHOW command must precede the OPEN statement. For example:

```
SHOW(10,15, '')      IPOSITION WINDOW AT ROW 10, COLUMN 15
OPEN(WINDOW)        IOPEN WINDOW
```

will set the upper left corner of the window at Row 10, Column 15. The pop-up will now float from this position.

* What should I do if I get the error message TOO MANY OPEN FILES?

If your program uses more files than your system is set for, the Processor displays the following error message:

```
TOO MANY OPEN FILES
```

To remedy the situation, change the default number in your CONFIG.SYS file. For example:

```
FILES=n
```

* What does the question mark (?) in front of a field name in the code section mean?

Screen fields within a SCREEN structure are automatically assigned a field number by the Compiler, with the first field in the structure being field 1, the next field 2, and so forth. This number sequence is used to automatically progress in order from field to field through a CLARION screen, accepting input for each field. The FIELD() function may be used at any time to return the current field number.

When a CLARION program is compiled, field equate labels are automatically generated for each field. A field equate label may then be used in place of the actual field number to reference a particular field. The field equate label is formed by placing a question mark in front of the USE variable name.

For example, an entry field defined as

```
ENTRY(25,25),USE(ADDRESS)
```

would have a field equate label of

```
?ADDRESS
```

Assuming that this is the third field in the screen, ?ADDRESS is equated to 3. A CASE statement can then be used to check for this field in either of the following forms:

```
CASE FIELD()
OF 3
IF ADDRESS = '' THEN BEEP.
```

OR

```
CASE FIELD()
OF ?ADDRESS
IF ADDRESS = '' THEN BEEP.
```

The advantage of using the field equate label is that if a new field is inserted which changes the automatic field numbering, the Compiler will update the value of ?ADDRESS. See page 6-3 of the *Reference Manual* for a more detailed explanation of field equate labels.

* How can I put a new screen into an existing program?

Use the Editor to create the SCREEN structure by typing a label in column 1, followed by the keyword SCREEN and a period. Then use the Screener to design your screen. The Screener will automatically place the code for your screen design in your source file following the SCREEN structure label.

* How do I use Ctrl-S in Screener to move the order of a field?

1. After you have pressed Ctrl-S to access the screen Summary window, press Enter to display the **Change type** menu at the bottom of the screen. The selected item's current type will be reversed.

2. Change the selected item's relative position in the source code by using Cursor Up or Cursor Down to move the selected item to a new position. If a structure is selected, the entire structure — everything from the MENU or REPEAT statement to the terminating period — is also moved.

3. Using Cursor Right or Cursor Left, change the screen item's type designation, if necessary. The Screener won't let you choose a type that is incompatible with the selected item.

4. Press Enter to update the selected screen item. The appropriate window, with information on the item selected, will replace the Summary window.

5. Press Enter to step through the window, updating as necessary. Press Ctrl-Enter to return to the structure screen and save any changes. Press Ctrl-Esc to return to the screen without changing the selected item.

*** What should I do if I get the DOS error message Bad command or file name after entering a linker name in Translator's Linker name field?**

Chances are you forgot to install your linker on your disk drive or you installed it in a directory that is not specified in your AUTOEXEC.BAT file's PATH command. Some users found this confusing, so Translator has been modified as follows.

When you enter the **Linker name** field, Translator verifies that a file by that name actually exists on your disk drive. If the linker cannot be found, the path is searched for any occurrence of a file by that name. If the linker is found in the path, the correct full path name of the linker is automatically entered into the field for you. If the linker still cannot be located, the error message **Linker not in path** is displayed, the error beep is sounded, and the cursor remains in the **Linker name** field. To install your linker at this point, press Ctrl-Esc to exit Translator and install your linker as directed in the linker documentation.

*** When I issue a RUN command, part of my screen scrolls off the monitor and DOS writes a message on the bottom line of the screen. How can I issue a RUN command without having to reopen my screen to overwrite the DOS message?**

After processing the RUN command, DOS writes a message to the screen confirming that the command has been processed. For example, after the command

```
RUN('COPY FILE1.EXT FILE2.EXT')
```

has been processed, DOS scrolls the entire screen up one line and displays the message

```
1 File(s) copied
```

making it necessary to reopen the window or screen that was open when the RUN command was issued.

To prevent DOS from displaying the above message on the screen, enter the following:

```
RUN('COPY FILE1.EXT FILE2.EXT > NUL')
```

"GETTING STARTED" GUIDE ADDED TO CLARION DOCUMENTATION

You asked for it, and here it is. A *Getting Started* guide is now included in each CLARION package.

Getting Started answers the questions most frequently asked by programmers using CLARION for the first time. It briefly describes the utility programs, the format and structure of a CLARION program, and the steps in writing a CLARION program. *Getting Started* also includes a topical guide to the more advanced programming concepts illustrated in the *Examples & Exercises* manual. ☞

MORE PROGRAMMING TIPS AND TECHNIQUES AVAILABLE

Members of our technical staff have written a number of articles describing various programming tips and techniques. If you would like to have one or more of the articles abstracted below, just call 1-305/785-4555.

CREATING A BATCH FILE IN DIRECTOR

by Kim Kemp

Instructions for creating a batch file in Director by redirecting input and output devices.

USING THE INIT MACRO

by Stephen Kemp

Discusses when and how to use the INIT (Initialization) macro.

SWITCHING BETWEEN COLOR AND MONOCHROME MONITORS

by Monty Shaw

Discusses switching between color and monochrome monitors and provides code for activating either monitor.

ENTERING A STRING WITH NO ECHO

by Monty Shaw

Provides code for a function called GETPASSWORD, which pops up a small window and allows the user to enter a password that is not echoed back.

PEEK INTO VIDEO MEMORY

by Monty Shaw

Discusses changing the foreground or background attributes to display a message and then removing the message and displaying the original foreground and background attributes again. Provides code for the functions that make this possible.

CONTROLLING THE KEYBOARD LOCKS

by Monty Shaw

Provides code for turning keyboard locks on and off and for checking the current state of the locks.

CREATING STRUCTURED FILES IN CLARION

by Monty Shaw

Discusses simulating a structured file by using an empty file created with Filer and copying it to create a new file with the same format. Provides code for creating a new log file.

REGISTER YOUR APPLICATIONS WITH US

Many CLARION users have suggested that Barrington Systems maintain a registry of CLARION-developed applications. We are frequently asked if other developers have done certain types of applications and if that information can be exchanged.

If you have an application you would like us to reference, please call and ask for an Application Registration form.

Relevant information in the Registry will be provided only to other licensed CLARION users (or, with your approval, to prospective CLARION users). A complete list will not be made available.

Barrington Systems will not test or evaluate applications registered and cannot, of course, make any endorsements of them. ☞

CLARION CONTINUES VIGOROUS GROWTH

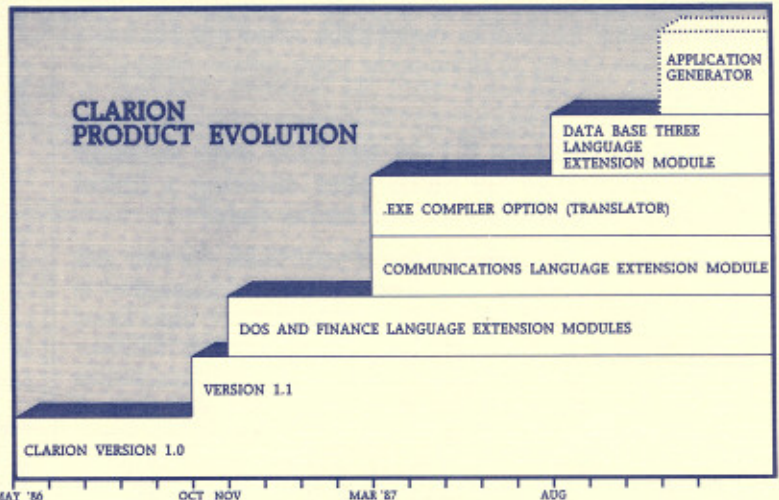
The Data Base Three LEM is the latest in a series of enhancements since CLARION's introduction in 1986.

The COMMI LEM, which became available in April, provides the capability of performing advanced asynchronous communications from within a CLARION-written application program. Other LEMs available are FILE1, DOS1, and FINANCE1.

Executable (.EXE) programs also became an option in April with the compiler addition, Translator.

The current CLARION version 1.1, which was introduced last November, features file-processing enhancements and the Converter and Crossrefer utility programs. Free, unlimited run-time modules were also announced at that time.

CLARION customers can expect further enhancements in application generation facilities this year. Significant productivity gains have been proven by features such as Screener, Reporter, Helper, and the overall compile/test process. Active development is underway to provide additional support. ☞



DATA BASE THREE LEM

(Continued from front page)

finds himself more and more restricted by the limitations of data base languages. Extensions to existing dBase applications can now be written in CLARION in a fraction of the time required for other languages. At the same time, developers can incorporate the exciting user-interface that has made CLARION so popular."

The new Data Base Three Language Extension Module (\$49.50), which allows you to process dBase III files as if they are CLARION files, provides a set of file-processing commands for .DBF files that parallels CLARION file-processing commands.

Record keys can be the same as those used in dBase programs or can be unique to the requirements of a CLARION-written application. During processing, the CLARION program updates and maintains indexes, which are then valid for subsequent processing by .PRG programs. ☞

UPDATES AND FUTURE VERSIONS

CLARION is periodically updated. These updates are available upon request to licensed CLARION customers for just \$19.50. Also, if you are the first to report a bug, your correction update is free. Information about updates is available from our Technical Support Center at (305) 785-4555 and on the Byte Information Exchange.

As new versions of CLARION are made available, credit will be given to current users for applicable CLARION products already purchased. So be sure that Barrington Systems has your current address correctly noted in its files.

If you received this newsletter at your correct address, you don't need to contact us. ☞

CLARION CONFERENCE NOW AVAILABLE ON BIX

Barrington Systems is now participating in the Byte Information Exchange (BIX) conferencing system. BIX provides product announcements, new update releases, and programming tips and techniques 24 hours a day.

You can also ask technical questions about CLARION through BIX and receive a reply within 24 hours, Monday through Friday.

If you are interested in subscribing to BIX and joining the CLARION conference, contact:

Byte Information Exchange
One Phoenix Mill Lane
Peterborough, NH 03458
(603) 924-9281

WITH ALL DUE RESPECT

dBase and dBase III are registered trademarks of Ashton-Tate.

TechniLogics and TL/RENTS are trademarks of TechniLogics, Inc.

SAI 5000, Energy 2.0C and SAI are trademarks of Systems Associates, Inc.

CLARION is a registered trademark of Barrington Systems, Inc.