

Top Story...

GI/TERM launched at Worldspan — slated to replace VISTA

Worldspan begins the migration of ALC users from the VISTA green screen to the new graphical user interface product by TPF Software.

As of early 2002, Worldspan has begun migrating its employees who are ALC users from the VISTA green-screen interface to a more powerful product with a graphical user interface: GI/TERM.

GI/TERM also supports the ACTIVE and TCP/IP protocols.

GI/TERM is TPF Software's ALC terminal emulation product that brings the convenience and power of a graphical user interface to long-suffering ALC users. A graphical user interface (GUI) provides users with the productivity enhancements expected on a modern PC: tool buttons; drop-down and right click menus; copy, cut and paste; point and click; multiple windows; and customizable colors, among others.

But the advantages of GI/TERM are not limited to a more attractive user interface. According to Patsy Ferguson, Senior Quality Control Analyst at Worldspan, GI/TERM has brought advantages in decreased training time, reduced resource use, and increased testing efficiency.

Training Advantages

Moving from a command-driven, green-screen interface to a GUI product usually makes it easier to train users, and Worldspan is finding this to be true with GI/TERM as well.

"GI/TERM will serve as a great time saver to our area not only for testing but also in the area of training," Patsy Ferguson says.

"We used to spend one and a half days teaching test system functionality and Xedit commands for working with our files. That time has been decreased to less than half a

Continued on next page

In this issue...

- 1 **GI/TERM launched at Worldspan — slated to replace VISTA**
- 4 **TPF/IDE: A new integrated development environment for TPF**
- 6 **Watch expressions debut in TPF/GI**
- 6 **TPF/GI Local Variables enhanced**
- 8 **Data Event Control Block support debuts in TPF/GI**
- 8 **More TPF Software enhancements**

day. We can sit with someone one on one and in about 30 minutes have them using basic functions. After less than two hours, they are proficient.”

Using GI/TERM

GI/TERM is easy to use from beginning to end. When a user starts GI/TERM, the first thing he or she sees is the GI/TERM Control Center (item 1 in the figure). This Control Center contains the list of hosts the user can connect to.

To begin using ALC terminals, the user simply double clicks a host in the Control Center. GI/TERM connects to that host, and all the CRT windows available for this user on that host automatically appear (item 4 in the figure). GI/TERM has the ability to connect to multiple hosts, with more than one terminal per host.

Resource Use Advantages

According to Jeff Longwell of Worldspan, GI/TERM’s ability to connect to multiple hosts with multiple terminals inspired him to make internal improvements to the VISTA code to work with this feature.

As a result, Worldspan has discovered that moving to GI/TERM will help them reduce the number of user IDs that each ALC tester requires.

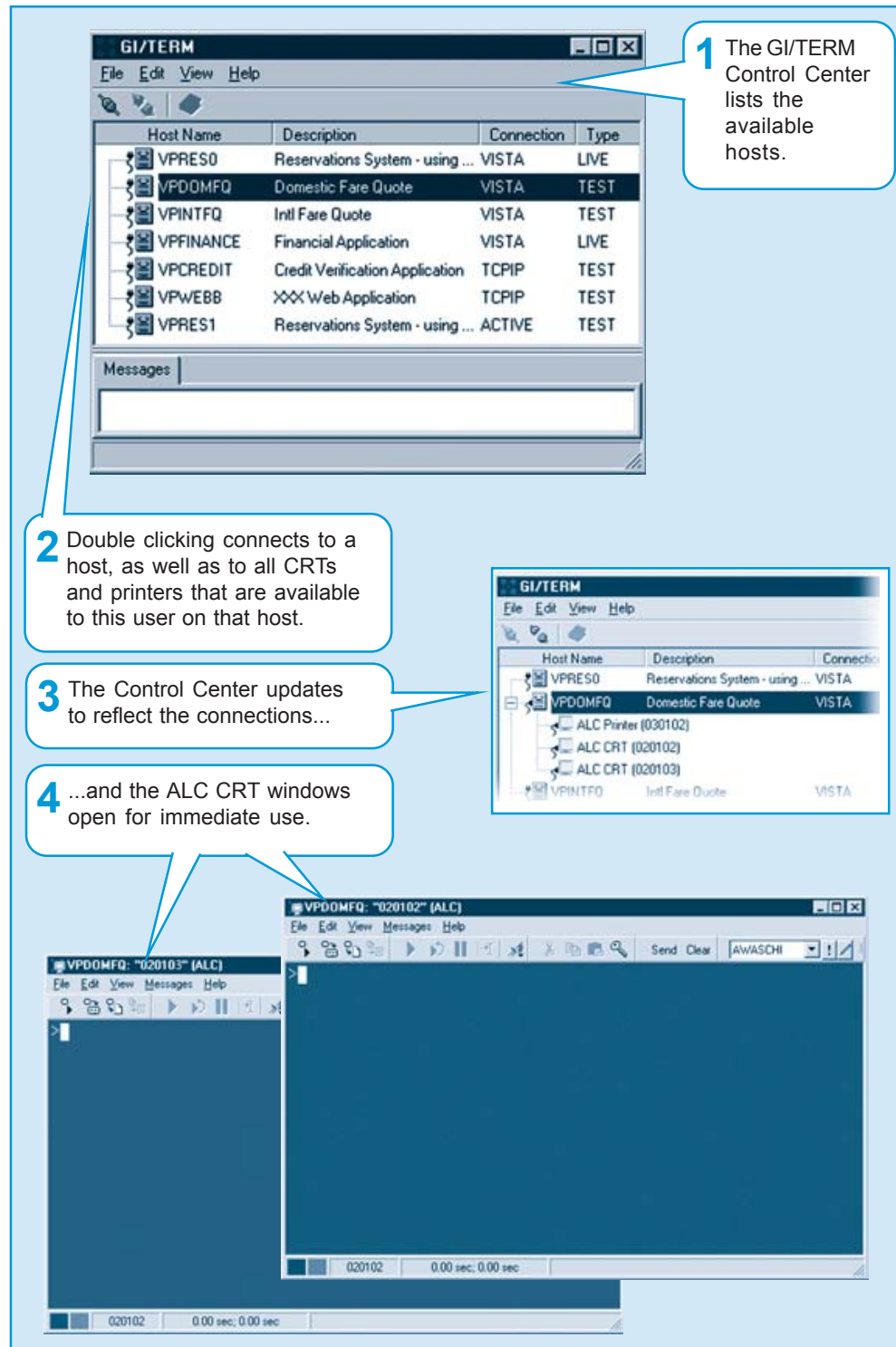
“I used to have four user IDs for accessing our test systems,” says Patsy Ferguson. “Within a week of using GI/TERM, I discontinued two of them as they are no longer required. If another session is needed, it is just added.”

Testing Advantages

Although GI/TERM is a great tool for *any* ALC user, it has features that make it especially convenient and powerful for ALC programmers and testers.

“GI/TERM has proven in the short time we have had it, to be a great tool for accessing our test systems and filing log/capture files,” Patsy Ferguson says. “We have all the functionality plus more than we had using VM4 and the cumbersome Xedit entries.”

The most important of GI/TERM’s testing features is the recording, playback, and logging of messages. In GI/TERM, these



three concepts are grouped together under the name “Message Run.”

Message Run

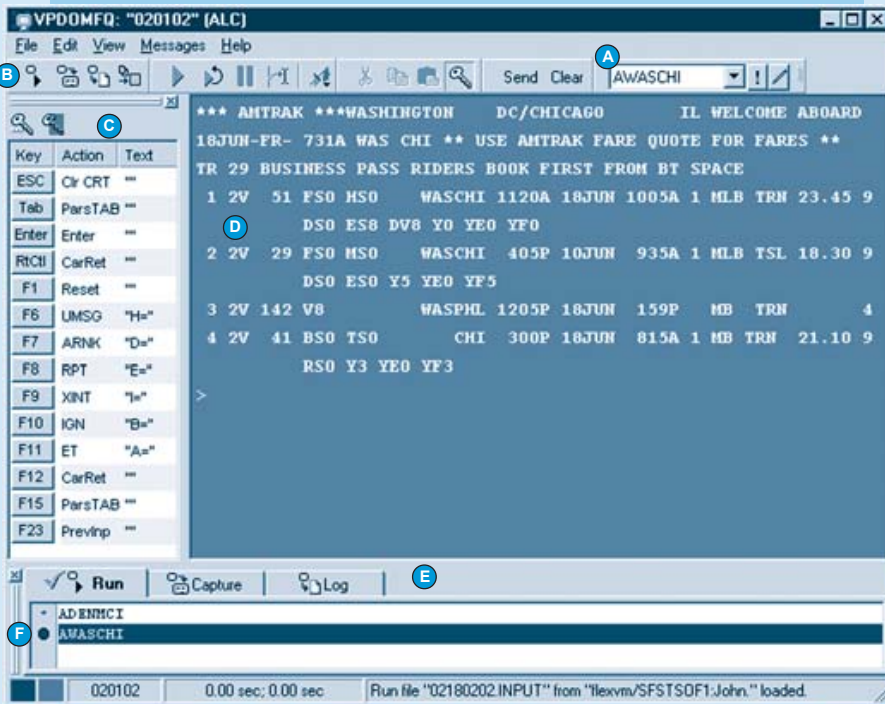
Every ALC CRT window in GI/TERM has its own Message Run window, which can either “dock” (attach itself to) the CRT window or “float” (be a separate window) (item E in the figure).

Each Message Run window contains three pages.

- The Run page displays the current input file for that CRT. This is the file of messages

The CRT Window In Detail

The CRT window's GUI features make using GI/TERM much more efficient than using the green-screen VISTA or ACTIVE interfaces.



- A The retrieve list is maintained in a drop-down box, and the contents of the retrieve list are editable.
- B Tool buttons allow users to open a Run file or start recording or logging with one click.
- C A customizable shortcut key window can dock, float, or automatically appear when the mouse hovers over it.
- D Text colors and fonts are customizable, and input can be a different color than output.
- E The Message Run window displays the contents of the current Run, Capture and Log files. This window can dock or float.
- F Breakpoints can be set in the message run file with one click.

ently caches host files on the PC.

The Run, Capture and Log facilities are extremely simple to control. For example, clicking the Capture tool button once will start recording input to a temporary file. Clicking the tool button a second time will stop recording and prompt the user to save to a permanent file.

Worldspan is finding GI/TERM's handling of files to be very convenient. "Now we point and click to access run files or any file," Patsy Ferguson says. And Joe Patton of Worldspan says, "GI/TERM improves my productivity by providing log files that I can easily send off-site."

Can GI/TERM Work for You?

The answer is almost certainly yes. You don't need CMSTPF, TPF/GI, or any other TPF Software products to use GI/TERM. All you need is TPF 4.1, and GI/TERM can be running on your system in a couple of days.

***You don't need
CMSTPF, TPF/GI, or
any other TPF
Software products.
All you need is TPF
4.1, and GI/TERM
can be running on
your system in a
couple of days.***

From an administrator's point of view, according to Jeff Longwell, GI/TERM is easy to work with. "I am happy with how simple the install and maintenance of GI/TERM is," he says. GI/TERM allows administrators to add company-specific items to its menus, and Jeff Longwell says that is another factor in making it easy for him to maintain.

Call Thiru at 919-676-5501 for information or for a trial run of GI/TERM on your system. ❖

that can be played back into the terminal.

- The Capture page displays the input currently being captured to a file.
- The Log page displays the input and output currently being logged to a file.

Any of these three files can be edited using a built-in PC editor and then easily saved to the host. In fact, it is a seamless process to open and use files on the host, edit them on the PC, then save them back to the host. To minimize download time, GI/TERM transpar-

TPF/IDE: A new integrated development environment for TPF

TPF/IDE is an integrated development environment (IDE) for editing, compiling and assembling TPF applications from the PC.

Although PC programmers have had full-blown integrated development environments for years, IDEs for TPF are a recent development.

TPF/IDE, a new integrated development environment by TPF Software, seeks to do something that no other IDE for TPF has done before.

Quite simply, TPF/IDE is a *legacy-friendly* IDE that works with your existing EXECs and library systems.

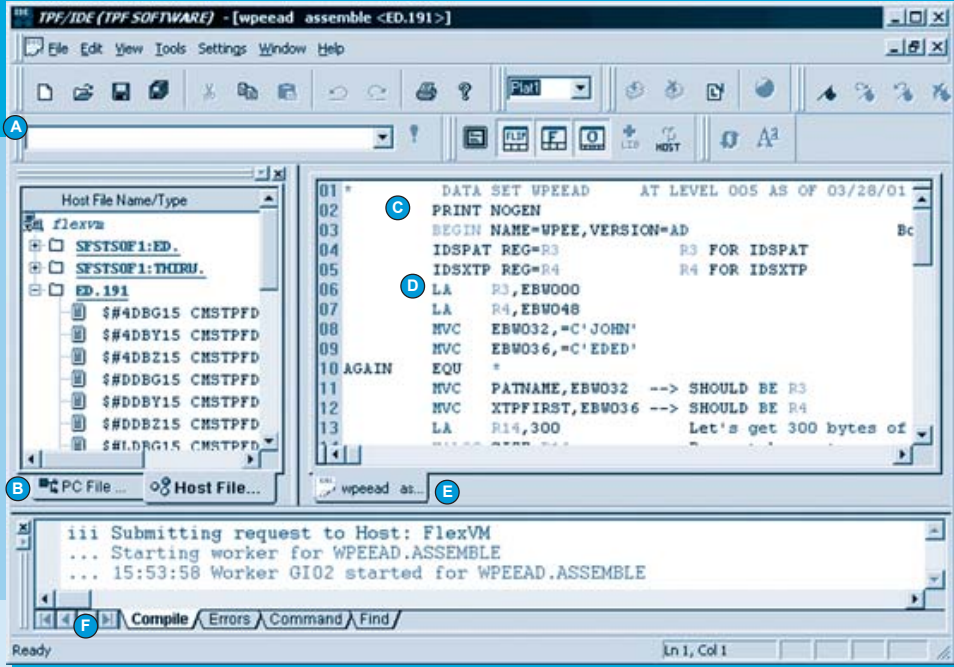
TPF/IDE is a legacy-friendly IDE that works with your existing EXECs and library systems.

The purpose of TPF/IDE is to allow programmers to upload and download source files in any language, edit them with modern graphical user interface conveniences, and compile or assemble them — all without requiring TPF shops to make several monstrously expensive purchases and completely change the way they do things.

Editing

TPF/IDE provides a modern PC editing environment with copy, cut and paste, syntax highlighting, and PC-style editing conventions that will be

TPF/IDE In Detail



The screenshot shows the TPF/IDE interface with several key components labeled A through F:

- A**: A drop-down menu for CMS commands.
- B**: File view windows for PC and Host files.
- C**: Syntax highlighting in the code editor.
- D**: Context-sensitive help available for keywords.
- E**: Multiple files opened for editing.
- F**: A status window showing errors and messages.

Code Editor Content:

```
01 * DATA SET WPEEAD AT LEVEL 005 AS OF 03/28/01
02 PRINT NOGEN
03 BEGIN NAME=WPEE,VERSION=AD
04 IDSPAT REG=R3 R3 FOR IDSPAT
05 IDSXTP REG=R4 R4 FOR IDSXTP
06 LA R3,EBW000
07 LA R4,EBW048
08 MVC EBW032,=C'JOHN'
09 MVC EBW036,=C'EDED'
10 AGAIN EQU =
11 MVC PATNAME,EBW032 --> SHOULD BE R3
12 MVC XTPFIRST,EBW036 --> SHOULD BE R4
13 LA R14,300 Let's get 300 bytes of
```

Status Window Content:

```
iii Submitting request to Host: FlexVM
... Starting worker for WPEEAD.ASSEMBLE
... 15:53:58 Worker GI02 started for WPEEAD.ASSEMBLE
```

- A** CMS commands can be entered here. A drop-down list remembers commands.
- B** File view windows make it easy to load remote or local files for editing.
- C** Syntax highlighting in the editor makes the meaning of code clear.
- D** Context-sensitive help is available: just click the keyword and press F1. C++ and Assembler supported.
- E** Multiple files can be opened for editing.
- F** Errors, messages, and other output is available in this Visual C++ style window.

familiar to programmers coming out of college.

Syntax highlighting makes code easier to understand.

To edit a remote file, programmers first simply double click the file in the Host File window (item A in figure). The file is

downloaded and automatically loaded in the TPF/IDE editor.

TPF/IDE applies syntax highlighting to editor files. *Syntax highlighting* means coloring or formatting language constructs so that the meaning of source code can be more easily understood. In TPF/IDE, syntax highlighting is applied according to the file type and is customizable so that new file types can be added. Current file types supported include *assemble*, *sabr*, *exec*, *bsc/bscr*, *lsc/lscr*, *c*, *cpp*, and *h*.

Who needs TPF/IDE? Your entire enterprise...

If you are currently using VM/CMS for your program development, and you are using EXECs to do your compiles and assemblies, your entire team will benefit from moving to TPF/IDE.

Application Programmers. They need to shorten the edit-compile/assemble-debug cycle by having a modern PC editor with integrated access to library systems and EXECs.



Systems Programmers. They need to implement IDEs for application programmers without throwing away everything that is already working.



Managers. Given today's tight budgets, they need to increase programming efficiency and application reliability without breaking the bank by having to buy everything new.



After the file is edited, it can be saved back to the host by clicking the Save button.

Multiple files can be edited at the same time, with full capability of copying and pasting between them.

Context-Sensitive Help

At some point during editing sessions, most programmers need help using a language feature or keyword.

TPF/IDE has *context-sensitive* language

help. All a programmer has to do is click a keyword in the source code and press F1. That term will be looked up and help will be shown.

Context-sensitive help is available for both C/C++ and Assembler.

Compiling or Assembling

TPF/IDE compiles or assembles programs by calling EXECs that you specify.

For example, if you already have an EXEC named TPFASM that assembles a TPF

program, TPF/IDE will use it.

This means that if you're currently using VM/CMS for your program development, and you're using EXECs to do your assemblies or compiles, TPF/IDE will easily integrate with those facilities.

Getting TPF/IDE

To learn more about TPF/IDE — and to find out how easy and cost-effective it would be to install at your company — contact Thiru at 919-676-5501. ❖

Watch Expressions debut in TPF/GI

The new Watch Expressions window allows C/C++ programmers using TPF/GI to monitor and edit expressions containing absolutely any variable. Expressions can include literals and operators and can be quite complex.

TPF/GI has broken a new barrier in its quest to deliver greater power into the hands of TPF programmers.

The Watch Expressions window, new in TPF/GI 2.6.1, allows C and C++ programmers to view and edit the values of ALL variables while they debug their TPF programs.

That's right: we said *all variables*.

Previous versions of TPF/GI allowed programmers to view and edit the values of local variables only (local variables are variables that are declared inside of functions).

Now, however, the new Watch Expressions window allows the viewing and editing of any variable that can be manipulated by the current statement. This includes *extern* and *static* variables.

Any type of variable can be viewed and edited, including classes, structs, arrays, pointers, and null-terminated strings as well as ints, chars, doubles, and other basic C/C++ types.

The values for these variables are displayed by the Watch Expressions window in human-friendly formats: numbers can be viewed in decimal as well as hex, char can be viewed as characters, and enum values can be viewed by their symbolic names rather than their ordinal values only.

Expressions, Not Just Variables

But the description so far just scratches the surface of the Watch Expressions window's power. The window actually allows you to look at *expressions*, not just variables. Expressions can be made up of *more than one* variable as well as literals and operators (see figure).

Among other things, the power to use expressions rather than simple variable names means that you can view and edit individual elements in an array and individual member variables of a struct or class without viewing the entire array, struct, or class.

For example, if *a* is an array of int, you can enter the expression *a[0]* into the Watch Expressions window to view and edit only the first integer in the array.

Even more advanced, if *i* is an int variable that your code is using to loop through the values in *a*, you can enter the expression *a[i]* into the Watch Expressions window. Each time *i* changes value, a different int from *a* will be displayed.

Other expression examples:

- **Structs and Classes.** If *s* is a struct and *m* is a member variable of *s*, *s.m* monitors the value of one member variable.
- **Pointers.** If *pn* is a pointer to an int, then the expression **pn* will dereference that

Local Variables window renamed Variables and enhanced

The TPF/GI window formerly known as the "Local Variables Window" has been enhanced and renamed the "Variables Window" in TPF/GI 2.6.1.

This new Variables Window has two pages. The "Locals" page displays all local and parameter variables that are visible in the current scope (to view all variables, programmers can use the new Watch Expressions window — see page 1).

The new "this" page displays the value of the hidden *this* variable when it is available. Although the Locals page also displays the *this* variable, the "this" page will display it with its member variables already conveniently expanded. ❖

pointer and display the value of the int. If *pc* is a pointer to a class instance and *m* is a member variable of that class, *pc->m* monitors the value of one member variable.

• **Logical Operators.** *n == i* returns 1 if the values of *n* and *i* are equal, 0 if they are not. *<*, *>*, *>=*, *<=*, *!=*, *&&*, *||*, and *!* behave as expected.

• **Mathematical and Bitwise Operators.** Operators such as *+*, *-*, ***, */*, *%*, *&*, *|*, *<<*, *>>*, and *!* work as you would expect. Parentheses group subexpressions, or the normal rules of C/C++ precedence apply. For example, *(x + n) * a[i % 3] / j * i* is handled easily by the Watch Expressions

C and C++ programmers deal with nested structures, classes, and arrays that can be quite complex.



```
class c_Rolodex
{
private:
    t_ContactArray    m_TwoBestCustomers;
    t_Contact         * m_AllCustomers;
    int               m_CustomerCount;
    t_FullNames       m_FullNames;
public:
    c_Rolodex();      /* constructor */
    ~c_Rolodex();     /* destructor */
    t_Contact * getCustomer(int Index);
    int addCustomer(t_Name MiddleName,
                  t_Phone PhoneNumber);
    void getCustomerCount();
    /* The next member function is the
    assignment operator */
};
```

That's why they need features such as TPF/GI's Watch Expressions window. It can show the value of expressions of any complexity.

Name	Value
pcs2_3->c2double	999.999
(*pcs2_3).c2short	77
local_int	2
local_double1	11.1
local_int - local_double1	-9.1
contact3.FullName[2]	"Smith"
pcs2_3->c2double/local_double1	90.09
contact3.FullName[local_int]	"Smith"
(local_int+1)*local_double1	33.3

window.

Using Watch Expressions

The Watch Expressions window (see figure) is made up of four pages — Watch1, Watch2, Watch3, and Watch4. Each page contains a table with expressions in the left-hand column and the values for those expressions in the right-hand column.

As you step through your program, values that change are highlighted in the Watch Expression window, and values known to be invalid take on a grayed appearance.

Expressions can be added to the window in several ways. To place the name of one variable in the window, you can right click on that variable in your code and choose *Add to Watch* from the local menu that pops up.

To place an entire expression in the window (not just a single variable name), you can select the entire expression in your source code before right clicking. You can also drag and drop a selected expression into the window.

At any time, you can type a new value into the right-hand column or a new expression into the left-hand column. Typing a new value into the right-hand column will actually change the value of that variable on the host, allowing you to interactively test how your program behaves under different conditions. Typing a new expression into the left-hand column will change which expression is being watched.

Expressions can be removed from the window by hitting the Delete key or by right clicking and selecting Delete. ❖

DECB support debuts in TPF/GI

TPF PUT 13 and higher support the concept of data event control blocks (DECBs). And now TPF/GI 2.6.1 provides powerful graphical tools that help programmers debug code that uses DECBs.

What is a DECB? DECBs act like data levels that can be created and released dynamically, that can be named, and that can grow as necessary. IBM created the DECB concept to address the archaic limitations of having only 16 data levels that cannot easily expand.

To accommodate the DECB concept, TPF/GI has redesigned its graphical ECB window. In the process, support for traditional data levels has been enhanced as well.

Redesigned ECB window

A new DECB area has been added to the graphical page of the ECB window. In this DECB area, each DECB is represented by a rectangle. The name of the DECB appears on the front of the rectangle. If a DECB contains a block, the rectangle representing it is red; otherwise, the rectangle is blue.

The size of the block owned by the DECB is indicated by a single character in



Figure 1: A new DECB area has been added to the ECB window.

parentheses beside the DECB name: for example, *FARE QUOTE (4)* means a 4K block is on the DECB named *FARE QUOTE*. *(H)* means a high speed block, *(L)* a large block, and *(S)* a small block.

The presence of detached blocks is signalled by a smaller red rectangle within the data level rectangle.

Using the DECB area

To edit the contents of detached blocks, programmers simply double click the smaller red rectangle. Similarly, to edit the DECB block itself, programmers can double click the main DECB rectangle.

Other operations on the DECB can be performed by right clicking or by using the memory chip, trash, and database

icons in the lower right corner of the ECB window.

For example, dragging the memory chip and dropping it on a DECB level will get core for the DECB. Dragging a DECB to the trash will release the core block associated with DECB. Dragging the database icon to the DECB will do a FINDC.

Selecting a DECB level causes information about that level to be displayed in the one-line detail area below the DECB rectangles. The fields in this detail area can be double clicked or right clicked to explore DECB information in even more detail.

A similar one-line detail area has been added for the data level area as well.

More TPF Software enhancements

ECB Contents Customizable

The TPF/GI ECB window now has buttons that allow users to determine what information is shown. Users can elect not to view data levels, DECBs, or machine trace.

New Trace Statistics Window

The new Trace Statistics window in TPF/GI displays statistics and charts detailing program and file I/O, pool usage, memory and heap usage, macros,

and instructions.

TPF/GI Scripting Enhancements

TPF/GI 2.6.1: (1) Adds scripting objects to make it easy for users to manipulate and edit core on the host; (2) Takes steps to ensure that the Microsoft script debugger will automatically appear to handle programmer errors in scripts written for TPF/GI; (3) Lets users supply their own script templates to be used when they create new scripts from within TPF/GI.

View Fixed Files enhanced

The View Fixed Files dialog box now accepts either equate names or record IDs.

Visual Log enhanced

Users can now reopen previous Visual Logs and append information to them.

Trace Output viewer enhanced

The Trace Output viewer search filter that contains the names of all programs and ECB IDs is now sorted in alphabetical order. ❖

www.tpfsoftware.com ♦ info@tpfsoftware.com ♦ 919-676-5501
TPF Software, Inc. ♦ 8729 Gleneagles Drive ♦ Raleigh, NC USA 27613-5419
Copyright © TPF Software, Inc. ♦ Editor: Ed Jordan

