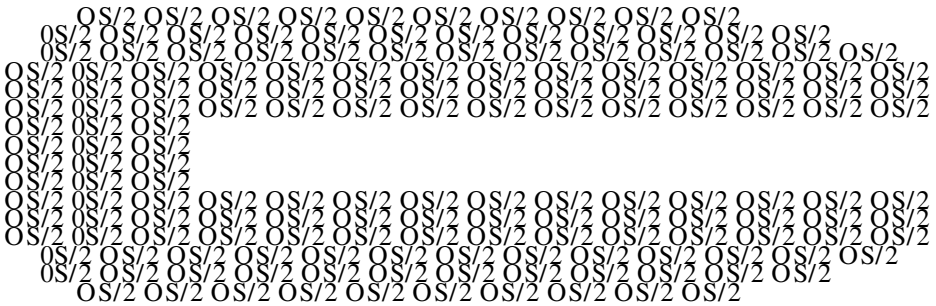

IBM OS/2 Compiler News

April 92

In this issue

They're here! OS/2,[1] C Set/2, ™ WorkFrame/2, ™ the products you've been hearing so much about, and that many of you have been testing. Our lead article highlights the key features of OS/2 2.0, and we follow that with the C Set/2 story. Also in this issue:

- Tooling around COMDEX
- Support for our products
- Branding
- New functions in C Set/2
- Technical Q&As
- WorkFrame/2 editor APIs
- News of Japanese versions of our products

OS/2 .... Just what you've been waiting for.

On March 31st 1992 IBM shipped OS/2 Version 2.0. The first 32-bit Operating System geared to the end-user. One of the most anticipated Operating Systems in the history of the PC. With a Workplace Shell ™ that's a state of the art user interface, OS/2 2.0 also has features developers can truly appreciate. Combined with the growing number of development tools, from IBM and others, OS/2 is

[1] ™ OS/2 and products marked ™ are trademarks or registered trademarks of IBM Corporation

destined to be the development platform of choice that will attract developers from mainframe environments, DOS, Windows,[2] and UNIX,[3]

Much has been said about the abilities of OS/2 to run DOS and Windows software programs unmodified. This is usually the first thing you hear about OS/2, and it is a very important feature of the Operating System. After all, the level of compatibility achieved borders on the miraculous. But behind the miracle is a lot of really cool technology, made possible by OS/2's 32-bit architecture.

The Features You Need to Know about:

32-Bit Memory

Version 2.0 exploits the 32-bit architecture of the Intel 386/486 family of microprocessors (including the SX and SLC). Version 2.0 will let access up to 4GB Gigabytes of Real memory and up to 64TB (terabytes) of virtual memory!

Performance

The expected performance improvement achieved by migrating a 16-bit program to a 32-bit version ranges from 5-60%, depending on how an application is written, and what 32-bit features it uses. It also means that the 32-bit instructions unique to the 386 (et al) chip can be used for better memory management. The file systems have been optimized with 2.0 as well. The FAT has already been converted to 32-bit. HPFS still has to be re-written. But both have improved disk caching (Reading and Writing).

Virtual DOS Machines

The 32-bit instruction set allows OS/2 to create Multiple Virtual DOS Machines; which means in essence that each DOS session runs independently of and protected from one another. So if one DOS application hangs, nothing else does. The user just closes the application and restarts it, And his/her system is left intact throughout the entire process.

Another advantage 2.0 gives us is the ability to load all device drivers above the 640K DOS memory limit. So even with a mouse, EMM, and network drivers loaded a DOS session can have in excess of 630K available.

---

2 ™ Registered trademark of Microsoft Corporation

3 ™ UNIX is a registered trademark of UNIX System Laboratories Inc

If 630K+ is not enough, DOS EMS drivers can be used to access up to 32MB of EMS 4.0 memory; and programs that incorporate "DOS memory extenders" that use DPMI (DOS Protected Mode Interface) technology are also supported. In fact a DOS application can access up to 512 MB of extended memory.

This magic is done through Virtual Device Drivers (VDD) for the Screen mouse, Netbios, Memory, etc.

Windows 3.0

OS/2 2.0 will run most Windows 3.0 applications. The exceptions are those few apps that access the special 32-bit DLL provided with Windows. However any Windows app that runs in REAL or STANDARD mode should run without modification.

Window applications have the advantage of being protected from each other. This will greatly reduce the chance of UAEs (Unrecoverable Application Errors). Since the Windows Applications can be run in separate sessions, they're not limited to the finite resources available in a single Windows session (another common cause of UAEs). If you have a particularly ill-behaved Windows application and it generates a UAE all by itself, you don't have to shut down your system as is required by Windows.

You might expect that Windows software running under OS/2 would perform poorly compared to running under Windows 3.0 itself. In fact, many applications run just as fast or faster under OS/2 as they do under native Windows.

WorkPlace Shell

OS/2's new desktop metaphor is called the Workplace Shell. The new metaphor incorporates file drawers and folders. The drag-and-drop capabilities are greatly enhanced over what currently exists in earlier versions. The environment is highly customizable, and user friendly. The user need never go to a command prompt unless so desired.

What's Ahead:

With state of the art Development Tools and Applications from from IBM and others, including Borland, Lotus, Wordperfect and Describe, OS/2 is destined to be a rich robust user environment that will be adopted by novice and expert users alike.

Today from one desktop a user can access DOS, OS/2, Windows, and Remote X Windows Application, seamlessly....truly the Integrating Platform for the nineties.

WorkFrame/2 and C Set/2... Countdown to Launch

The story of these products is a tale of teamwork, dedication, and belief in a goal. The goal was the delivery of the core set of IBM OS/2 2.0 development tools by the teams of Toronto, Lexington, Boca, and Yorktown.

The way things work is that the team receives a "golden master" of the operating system, which in turns allows us to finalise our products. We then ship our golden masters to the manufacturing plants. You're the next in line: our customers.

The Golden Master arrives and everyone swings into action. Finalizing the products on the OS/2 2.0 Golden Master from Boca means build, test, and fix, to ensure total integration. In the preceding weeks and months, the product has been rigorously tested with ongoing releases of OS/2 2.0, so by now, people are familiar with every subtle nuance of the code, and we're at fever pitch. We've worked so hard for such a long time, and now our "moment in history" approaches.

One idea obsesses everyone: this is our first completely in-house compiler and we want our drivers to be of the highest quality. Everything else is irrelevant, including eating and sleeping. The teamwork is amazing: everyone's on board...it's like a contagious disease - everyone catches it. We're hard at work, and then suddenly - we're there! The last test case checks out ok, and a resounding shout of triumph (or maybe it's relief) echoes through the Lab. We've met the previously announced availability dates of March 31, 1992 for WorkFrame/2 and April 7, 1992 for IBM C Set/2.

Now what? We can finally relax, chat about the product, and relive what has been an incredible experience .... do you remember when ....? wasn't it amazing when ....? did you see them when they ....? And we laugh and joke and smile, because we did it: we got our product out the door on time. Way to go, team!

IBM C Set/2

Using the technologies of our C/6000, C/370, C/400 and the recently announced C+ + /6000, Toronto Lab is delighted to bring to you IBM's first fully in-house developed compiler for the PC. This compiler is a coupling of a state of the art, optimizing 32-bit Back End, and a common C Front End shared across IBM's C compilers. Thus, not only does it allow your applications to truly exploit the power of 32-bit processors, it also delivers the promise of SAA ™ for C on OS/2.

Along with the C compiler, the IBM C Set/2 product comes complete with run-time libraries and a fully interactive, full function, source-level PM debugger.

IBM WorkFrame/2

Boasting a truly open development environment, the IBM WorkFrame/2 product allows you to take advantage of OS/2 2.0 as your application development platform of choice - whether you are building native OS/2, DOS or Windows applications.

Highly configurable, IBM WorkFrame/2 has a graphical user interface and is multi-threaded. It's project-oriented and allows you to seamlessly plug in different and/or multiple Edit/Compile/Debug/Link tools.

Beta program

Perhaps a better name for beta would be 'beaten'! Many of you have given C Set/2 a good workout, throwing everything you could think of at the product. Since its first beta code drop last July 1991, more than four million lines of internal IBM as well as external customer C code have been processed through C Set/2. The beta program for OS/2 2.0 development tools has also contributed to the continued refinement of this product through both problem and requirements reports.

We've fixed and will continue to fix bugs reported to us. To the extent that we could, we have also implemented some of the requirements you've made known to us. A list of these late functional additions can be found in this newsletter. Those requirements that have not been implemented are currently under consideration for the next release of the product.

What about C++ ?

This is undoubtedly the most popular question that we have received - both internally and externally - over the course of our C Set/2 presentations and briefings.

It's not our nature to discuss unannounced products, especially in a forum such as this, but we can say that as we have announced the RISC System/6000 C++ product, it is logical to assume that, given customer demand, we are actively pursuing this possibility.

J Versions

On May 29, 1992 IBM will make available the J (for Japanese) versions of our products. IBM WorkFrame/2 J V1.0 and IBM C Set/2 J V1.0 will run on OS/2 J 2.0 and will support both SBCS (Single-Byte Character Sets) and DBCS (Double-Byte Character Sets). Both these products will be fully translated into Japanese.

Have you driven our C Set lately?

Like our friends at Ford, we're aiming to constantly improve our products, and since the announce of C Set/2 and WF/2, we've added some new function you may not know about.

- Support for 16 bit callback functions

  These are functions that exist in 32-bit modules but are callable from 16 bit code. They can be used as callback functions for 16 bit subsystems such as ES.

- Support for user named data segments i.e. #pragma data_seg
- Support for user named text segments i.e. #pragma alloc_text
- Unix-style low-level I/O functions
- Linkage keywords for easier & more compatible specification of function linkage
- /sp option and #pragma pack which allow MS compatible structure packing
- The new /gh option allows you to generate special prolog sequences which, when used with vendor profiling tools, enables profiling of C Set/2 generated code

The first six are detailed in the User's Guide. The /gh option is covered in the README file.

Making a Good Thing Better

Let's talk about something that's probably been on your minds since the moment you heard that IBM was about to offer a brand-new set of application development tools for OS/2 2.0; how are we going to support our new products?

We don't often talk about this aspect of our business, perhaps because like everyone else we take it for granted, but it's something that we believe we do well. And in the case of our OS/2 2.0 applica-

tion development tools we've gone even further in the US and Canada by adding on-line defect support to our more traditional support methods.

Until now, we've asked you to either contact your dealer or IBM representative, or mail your problem to us. Now, you'll be pleased to know, if you call 1-800-237-5511 you'll get direct access to the IBM Support Center. Instant defect support.

How does all this work? When you call this number you are connected to one of three dispatch centres - Tampa, Boulder or Chicago. From there, as an OS/2 tools licensee, the operator routes you to the IBM Support Center in Boca Raton, Florida, where a fully-trained tools support person works on diagnosing and solving your problem. If more specialised knowledge is required, the problem is sent to the lab responsible for originally developing the tool, where it is examined and corrected.

Here's another IBM support fact that you may not have known: once we've checked out your problem and corrected it, we send you your code fix by courier. We don't keep you waiting until a new release comes out; we send you your fix there and then.

And the best part is this: all this is included with your license for any IBM OS/2 2.0 application development tool!!

Product Branding

Branding:[4] when you see the ***Golden Arches***, you likely recognize the the "brand" of McDonald's. A brand provides instant recognition of both the company and product that it represents. Brands that survive today do so because the images they portray differentiate their product from their competitors. The dictionary defines a "brand" as a mark put upon an object to indicate:

1. Ownership
2. A set of product attributes
    - Character of the goods or product
    - Quality
    - Dependability
    - Consistency over time

[4] This is the first of a series of articles on product branding, and how it will apply to WorkFrame/2.

Yes, a brand is still the same mark those cowboy heroes of the american west used - a 'mark with red-hot iron'.

In customers' minds a brand creates a preference and reinforces an existing company image or product strength. Also the brand may command a premium if what it conveys to customers is top quality. Inside the company, a brand acts as a framework for operational strategies, provides leverage for the company assets, focuses the business and motivates employees.

Let's look at how and where brands fit into marketing today.

1. The ***offering*** is the physical product and/or service that company markets.
2. Next is the ***trademark***: any word, name, symbol or device, any combination. It is used to identify the company's products and to distinguish them from those manufactured or sold by others.
3. The ***brand*** is the set of attributes that differentiates the offerings from those of its competitors in the minds of the customers.
4. The ***image*** is the image generated by the sum of all experience with and information about, a company in the minds of all. This information includes all the company's ***brands*** and as well as the qualities of its employees.

The product offerings **C Set/2** and **WorkFrame/2** are trademarked to protect IBM from being used by any other manufacturer. They could also be used as brands.

One of IBM's central strategies of today is openness and being able to integrate other vendors products. That is exactly what WorkFrame/2 does for application development tools: it allows developers to use what they want. We are already part of and thus enjoy the benefits of both the **IBM** and the **OS/2** brands. WorkFrame/2 provides a clearly different set of features and/or functions from other competitors which we can use. Tool manufacturers who want to enable their product to WorkFrame/2 and who also want to use the WorkFrame/2 brand will be able to reap the inherent benefits of both the IBM and OS/2 brands as well as the value of the WorkFrame/2 brand. In the next newsletter we'll discuss where and how you can look for the **WorkFrame/2** brand as well as how to become "branded".

Tooling around COMDEX

What takes up a lot of room, attracts many thousands of visitors, has lots to see, and is known all over the world? Ask anyone in the street and they'll say DisneyWorld. Ask anyone in the computer business and without hesitation they'll reply: COMDEX!

Spring Comdex/92 is over, and the response was nothing short of incredible. IBM had a 10,000 square foot booth, and C Set/2 and WorkFrame/2 were being shown to anyone interested. And people sure were interested! The booth was busy non-stop. Our team demo'd, chatted, gave away literature, and answered technical questions.

There was a head to head running in the booth, showing 16- and 32-bit versions of PMGLOBE (a compute intensive graphics program) running simultaneously on two identical computers. The 32-bit version (which was compiled from the original 16 bit source by C Set/2) completed drawing in one third the time of the 16 bit code.

Because we developed C Set/2 and WorkFrame/2, and we use them too, you'd expect us to be very enthusiastic about them, especially considering the excellent feedback from our beta test. But even we were astounded by the unqualified support booth visitors gave us. Here are some comments:

- We wouldn't have been able to finish our prototype without Workframe/2. It really increased our productivity.
- We are very happy with the excellent code generated by your compiler.
- This is much better than any other development environment we have seen.
- This is exactly the kind of tool we require to do our development.
- This is a very nice debugger. The whole environment looks very powerful.
- How hard will it be to port my code from MSC 6.0 to your compiler?
- Very impressive!
- What does it cost and when is it available?
- I didn't think IBM could make tools like this.

Over 150 people came by the C-Set/WorkFrame booth to see demos, and entered our draw for a copy of Workset/2 (which includes the C Set/2 compiler and debugger, the WorkFrame/2 development environment, and the OS/2 developers toolkit). We are pleased to announce that the winner of the draw is Michael Pittard of HealthQuest, Atlanta, Georgia. Congratulations, Michael.

During the show, many visitors asked questions we weren't able to answer there and then. Other folks wrote questions on business cards and left them with us. We thought we'd share these questions with all our readers, so if you were at COMDEX, read on ....

Topic	**Question/Answer**
Q: Makefiles	How can you include other objects in a generated makefile?
Answer	When you run the makefile creation utility, you select the source files, and the actions you wish to execute. If you have extra object files you wish to be linked in, simply select those objects (in addition to the other source files), and ensure the LINK action has been selected. WorkFrame/2 will then build a makefile that will link the selected objects into the final executable.
Q: MicroFocus COBOL	Is Microfocus COBOL supported?
Answer	Microfocus have publically stated that they do intend to provide WorkFrame/2 support. WorkFrame/2 itself is a language independent development environment - the external interfaces have been made publically available, and any vendor that wishes to provide WorkFrame/2 support is free to do so.
Q. Capture Tools	Why does "Capture Tools" not work on VIEW?
Answer	VIEW is actually a front end to the help manager which starts a new program and then exits immediately. WF captures only the tools that it starts. It is not aware of any tools started by an already executing process. Hence, the behaviour that you're seeing cannot be avoided.
Q: Making mods to project make files	Can I have WF/2 remember my modifications to the project make files?
Answer.	Yes. When you try to save your automatically created make file, you will be asked whether you wish to overwrite your existing make file. If you answer no, then it will only write out the dependency file. Hence, any changes you've made to the make file will be intact.
Q: Make file creation steps	Can you add a preprocessor step to the make file creation process?
Answer	Currently, no. This version of the utility was intended to cover the most common functions and a fixed set of actions. As mentioned earlier, you can modify the make file and be assured that it will not be overwritten when dependencies are refreshed.

Topic	Question/Answer
Q: WF/2 function	Does the WF provide source/revision control?
Answer	The current version does not provide an architected source/revison control method. Our goal was to allow the end user to select the best set of tools and integrate these tools into the WF rather than providing everything ourselves. Hence in this version, you can either install some sort of logout and login command in the tools pulldown, or have icons representing the login and logon commands to which you can drag files. Providing an architected method for tightly integrating a SCS/RCS is our priority in the upcoming version.
Q: Command files	Can I run a command file as my compiler, linker. or other action tool?
Answer	Yes, but in order to do so, you must create a language profile that will call cmd.exe instead of the tool directly. However, if you proceed in this manner, then you may not be able to use the shipped options dialogs. Instead you must use the default dialogs which will allow you to pass the compiler name as a parameter to cmd.exe. If you're doing this to set up some environment variables that the compiler requires, then the best solution is to have them set properly prior to invoking the WF. This is in fact, one of the documented requirements.
Q: Migration	What's available to help me migrate from Microsoft C 6.0?
Answer	A migration guide is included with the C Set/2 product, that documents most of the changes required when porting from other compilers (including Microsoft C 6.0). Furthermore, a migration library is selectable, which will support many of the non-ANSI functions provided by other compilers.
Q: Profilers	Are there any supported profilers? Are any planned?
Answer	C Set/2 supports the generation of profiling hooks in the code generated. This support was developed in consultation with several profiler vendors. Although multiple profilers are being developed to support C Set/2, none have been announced to date.
Q: MicroFocus COBOL	Is calling from/to Microfocus COBOL supported? How?
Answer	Any language (including COBOL) that either uses or supports the standard system calling convention will be able to call and be called by C Set/2 programs. Support of this calling convention is the responsibility of the individual compiler vendors.
Q: Cfront	Is there a Cfront available?
Answer	C Set/2 has been successfully tested with internal versions of Cfront. IBM does not provide a Cfront.

Topic	Question/Answer
Strings display	Q: How do I change the strings display default?
Answer	There is no capability to change this default setting in the first version of the debugger. The requirement is well understood, and will be provided in a future release.
Q: Kernel debugging	What is available to help me with kernel debugging?
Answer	A kernel debugger is provided with the OS/2 Toolkit. This includes a debug enabled version of the kernel, the kernel debugger itself, and some other related tools.
Q: Writing device drivers?	What is available to help me write device drivers?
Answer	An IBM publication "Writing OS/2 2.0 Device Drivers In C" is available (The pub number is G362-0006). Further samples and documentation are also provided in the OS/2 Toolkit.
Q: Multiple windows	Can you run the debugger in one window and the app in another window?
Answer	Absolutely. This is the normal method of operation.
Q: Prices	Are educational prices available?
Answer	A National Education License Fee (NELF) is available. The NELF represents a 30% discount off the single unit price, and is not subject to any further discount or allowance. Contact your local IBM branch office for details.
Q: Upgrades from C/2	Are there upgrades available from C/2?
Answer	No upgrade discount is currently planned for C Set/2.
Q: Canadian availability?	What is the Canadian availability? Where & when can I purchase?
Answer	C Set/2 is generally available now in Canada (as well as elsewhere). You can order via your local IBM office, or by calling the toll free number 1-800-465-7999 (In the US 1-800-3IBM-OS2). Due to production constraints, it may be several weeks before you actually receive the product, although every effort is being made to ensure prompt delivery.

Technical corner

C Set/2 Multithreaded Libraries (the /Gm+ compile option)

Using DosCreateThread() vs _beginthread()

These two functions are similar, but not interchangeable. Either may be used by a C Set/2 multithreaded application, noting the following differences:

DosCreateThread()	_beginthread()
Thread function must have linkage type _System.	Thread function must have linkage type _Optlink.
Does not register a C Set/2 exception handler for the new thread. You must do this yourself with #pragma handler()	Registers a C Set/2 exception handler for you on the new thread.
Thread should never terminate by a return. It should always terminate with the function _endthread(). See the note on DosExit(), below.	Thread may terminate by either returning, or by calling _endthread().

Figure 1. Differences

Using DosExit() vs _endthread()

These functions are similar, but DosExit() should never be used to exit a thread. _endthread() frees storage allocated by the C Set/2 library for the thread.

Mixed 32-bit and 16-bit code

Problems can arise with the C libraries when code generated with C/2 is combined with code generated by C Set/2, because only one of the libraries can be initialized by an EXE. There are two possible solutions, with different limitations:

1. Write the main() function using C/2, and place all C Set/2 code in DLLs. You may use either the static or dynamic C/2 libraries. The C Set/2 librray is initialized by the DLL, and may be either sttically or dynamically linked. Use the 16:32 thunk generating features of C Set/2 to provide the entry points to the C Set/2 code from 16 bit code. This approach provides complete

library support for both compilers, but limits the stack size of thread 1 to 64 Kbytes.

2. Write the main() function using C Set/2, and place the C/2 code in DLLs. You must statically link LLIBCDLL.LIB to the C/2 code, which will initialize the C/2 libraries in the DLLs. Use the 32:16 thunk features of C Set/2 to call the 16 bit code. This removes the limits on the stack size for thread 1, but does not copy the parent process's environment to where the 16 bit DLLs can access it.

You ask, we answer

These questions are culled from a variety of sources: from internal and external fora; from customer calls; and from the reply forms mailed in by our readers.

Topic	Question/Answer
Q: Debugger	Will the debugger use the 486 hardware facilities, unlike MS CodeView?
Answer	Yes. The debugger provides hardware support for detecting when the value at an address (or range of addresses) changes value. • select Breakpoint/Set from within IPMD • select Change-Address from the Type listbox • specify the address range to watch in the Parameters field When you let the program run, the debugger will stop execution when the values at that memory location change. The monitoring of the memory location is done in hardware, so the performance impact on your application is minimal.
Q: Graphics support	Will the compiler provide graphics capability for protected-mode programs, unlike MS 6.0?
Answer	No. Currently the only graphics support which C Set/2 provides is via the Presentation Manager interface.
Q: Obj file format	Can someone please tell me the format of .obj files created by C-SET2 ? Is it the same as Microsoft C (32-bit) compiler?
Answer	Basically, the OBJ format for both MSC and IBM C Set/2 is the same. Both compilers produce Intel OMF files with various extensions. If your tool handles MSC today, it will probably handle C Set/2 as well.

Topic	Question/Answer
Q: C2 migration	A question from a first-time 1.3-2.0 application migrator. Our 1.3 application (compiled via IBM C/2) has a number of macros of the form (for example): #define DosDelete(a,b) DosDelete_(a,b) where DosDelete_ calls the actual API function and provides some common error notification and recovery functions. C/2 does not complain about this. C Set/2 produces a warning EDC0423 - "Macro name DosDelete cannot be redefined." Is this not an ANSI-approved type of thing to do?
Answer	Check your header files. I'll bet you'll discover that DosDelete has been #defined to DOS32Delete or something similar. In the 1.3 header files, this does not happen. Stick the line #undef DosDelete before your #define, and you should be OK.
Q: Named segments	How do I make named segments via C-SET/2?
Answer	Creating a named data segment is very easy. Just use #pragma data_seg(globseg) before your global data and #pragma data_seg( ) after your global data. Then put the following lines in your DEF file: SEGMENTS globseg CLASS 'DATA' SHARED globseg can be almost any name you want.
Q: /Ls /Lj	What other flag(s) should I use to replace the /Ls /Lj so that the only relevant source is written into the .lst file? In addition, if I use the /Ls /Lj, The file of .lst is too large and there is too much useless info which makes harder to find what the problem is.
Answer	The options for controling include information are as follows: /Ls - includes source problem /Li - includes user include files /Lj - includes both user and system include files So you may want to use either /Ls or /Ls /Li instead of /Ls /Lj.
Q: MS C 6.0 migration	I am migrating some code from MSC60 to C/Set2 and am having a problem with C runtime functions in the DLL. The functions in particular are atoi and toupper. I keep getting access violations. It appears that the toupper fails when trying to use the _ctype look-up table. I noticed that linking toupper with test program and NOT a DLL it works fine. In Microsoft 6.0 we used /Alfu option which forced a new DS register in the DLL routine. I am compiling the DLL with the following options: /G3 /Gs- /Gd+ /Gn+ /Se /Ss+ /Sn+ /c /Ti+ Link options: /CO /ST:8192 Linking with the dde4sbs and os2386 library

Topic	Question/Answer
Answer	You are probably doing one of the following: 1. Your .DEF file specifies your DLL as INITGLOBAL, either explicitly, or as the default. You must specify INITINSTANCE for most C Set/2 DLL's. 2. You have not specified DATA MULTIPLE NONSHARED. C Set/2 can't share the data in DGROUP.
Q: Pascal language	Pascal code is linked in object format to object C code before mutual compilation. Now if we upgrade the old C code to 32-bit, using say C-Set/2, how do we cope with the old 16bit Pascal code? Will the 32-bit compiler cope with the old Pascal code? Will it allow mutual compilation of 32-bit C and 32 bit Pascal?
Answer	Using C Set/2, you can convert the C code to 32-bit and leave the Pascal part in 16-bit, and use our 32-> 16 call support to interface between them.
Q: EPM not finding errors	How do I get EPM to locate errors? I define the editor with the /w option just like the documentation says. The WorkFrame/2 tutorial mentions something about a Compiler pulldown. I assume this refers to a pulldown from EPM. I do not have this.
Answer	Did you remember to select the 'Send compiler errors' checkbox in the configure editor dialog?
Q: LAN Install of C SET/2	I would like to install C SET/2 on the LAN so that all users of the LAN would have access to it. Is there a way after it is installed that each user can update their CONFIG.SYS and add the the necessary ICONS? I would assume that the function to do this is some where in the C SET/2 install.
Answer	Sorry, There is no utility to do this yet. You have to ask your user to update their CONFIGS.SYS and Desktop manually.
Q: msize macro	Anybody know how to get the size of a buffer allocated via "calloc" like the IBM C/2 1.1 compiler could via the "msize" macro? It was very useful to ensure you didn't memcpy something past the end of a allocated field; we use it in many places in our code and I can't find a replacement.
Answer	No we didn't provide the msize macro, but we've taken note of the requirement. Please think about using our debug allocation routines that we supply in our libraries. They will try to catch problems like the one you described. The routines are documented in our User's Guide and online information. Basically, all you have to do is compile with __DEBUG_ALLOC__ defined and all your allocation calls are changed to the debug ones. The Microsoft memory allocation routines keep the size of the block in a magic cookie at the front of the allocated block, so it's easy for a macro to check this value. C Set/2 uses more complex data structures (which allow us to malloc/free faster than Microsoft), but this will make it more difficult to find the size of a block. It certainly won't be possible with a macro. Maybe a function call. But using _debug_malloc will be much better.

Topic	Question/Answer
Q: Forcing system linkage	I have a routine in a DLL which is failing. I pulled the code into a short EXE to debug it and it "magically" worked. The problem with the DLL routine seems to be in the linkage I'm using to get to WinMessageBox. When I have icc generate ASM files, the calls look similar, except that for the DLL there is an extra "ADD ESP,18H" instruction after the call. (The parameters seem to be getting set up before the call in the same way for both routines: There are six 4-byte parms being passed, and each ASM file shows six PUSHes.) 1. does the ADD prove that optlink is being used in the DLL? 2. the two linkages look similar. In the case of six parms being passed, can the called routine tell the difference between system and optlink linkages? Here's my relevant source code: #pragma linkage(pfnWinMsgBox, system) typedef ULONG _System pfnWinMsgBox(HWND, HWND, PSZ, PSZ, ULONG, ULON ... pfnWinMsgBox *WinMsgBox; /* WinMessageBox routine */ ... DosQueryProcAddr(dll_handle, /* WinMessageBox */ ORD_WIN32MESSAGEBOX, NULL, (PFN *)&WinMsgBox); ... rc = (*WinMsgBox)(HWND_DESKTOP, (HWND) 0, text, title, (ULONG)0, sty Is there something wrong this this indirect way of using #pragma linkage?
Answer	To tell if _Optlink or _System linkage is being used, you have to look at the .ASM code to see if the parameters are being put into/pulled out of EAX/EDX/ECX or the stack. There is no magical way for a called routine to figure it out. Since you are seeing six PUSHes in each case, that means that both routines have _System linkage. The ADD ESP statement is the stack being cleaned up. Sometimes we clean up the stack for several function calls at once, but this should only happen at /O+ . Are you building one of your programs at /O+ ? Look around the rest of the function to see whether the stack is being cleaned up elsewhere. Otherwise, the code looks fine. Using Using #pragma linkage on a function pointer like that is perfectly ok.
Q: Warning Levels	Is there somewhere within the WorkFrame environment where I can set the Warning Level for my compiles?
Answer	If it's a compiler option you wish to change, then select Compile from the Options pulldown and then find the correct dialog to change the level (goes for any other option too).
Q: C Set/2 Performance	I am developing code that is pure 32-bit, Single Thread, Statically Linked and uses a few Kernel calls and no PM. I have recently increased the memory on my system and created a large VDISK. I would like to put All programs and data required for compilation and link edit into my VDISK so that I don't access my hard disk at all. Would someone please supply me with the list of programs and data that I should copy to my VDISK.

Topic	Question/Answer
Answer	Put the following: • Toolkit header files • C header files • The libraries you work with • The various exe's - Compiler/linker/RC-compiler/Make Most important - the TMP directory should be located on the VDISK! Make sure you run the header files through an 'Uncommenter' that removes the comments from them, you don't really need the comments and it'll cut down on your compile time. I have a 4 Meg VDISK and I have all the above fit in it with no problem. Compile time went down by about 50-60% when I've made the move. Make sure that your paths point to the VDISK. (Include/Lib/Path)
Q: Passing /I	How do you pass the /I parm from WKFRAME? I changed the Debug options to include this with no luck
Answer	You just select 'debug' under the 'options' pulldown. The ordering of the parameters you pass should be something like: 1. options to IPMD.exe 2. name of executable to debug (or %o/%f/...) 3. parameters to executable You could also not put any options at all and then PMD would put up an initialization dialog to get the required information. Parameters for IPMD are: • /i start debugging with application at the OS/2 initialization of the application. This will allow one to debug the initialization of global variables. Note: when main() is is reached this initialization is finished. The default is to start debugging at main(). • /n Do not use the restart information. As much restart information is saved as possible. Open windows, breakpoints,... Note: only thread 1 information is saved since other threads do not exist when the IPMD is started at main(). The default is to use restart information. This information is in the Debugger Tutorial, page 2 "Starting the Debugger from the Operating System/2 Command Prompt".

Topic	Question/Answer
Q: Pascal/2.	Can WorkFrame work with Pascal/2? I created a language profile for IBM Pascal/2 but when invoking the compiler I get the following in the compile window: invocation of < pascal.exe> begins invocation string: TEST.PAS < ---------->< Compile - TEST.PAS> ended tc= 0 rc= 8 at... What does rc= 8 mean? The compiler is not even executed.
Answer	The path to the Pascal compiler must be in the first 132 characters of PATH. When WFENV.CMD changed PATH, it moved the compiler's path beyond 132 chars.
Q: Invoking EPM from monitor box	When I invoke EPM by double-clicking on an error message line in the monitor box, the editor is invoked with the file containing the error, but I do not get the highlighted lines or the "Compiler" selection on the action bar as I do when I drag the error onto an already open edit window. Also, when the editor is invoked in this way, the drag drop interface from the monitor box appears to be disabled. Can someone please tell me what I am doing wrong? I did select the "Send-Compiler Errors" checkbox in the Configure Editor dialoge.
Answer	Add a /W to the front of the editor options string. For eg., if you had '%a %z' make it '/W %a %z'.

WorkFrame/2 interfaces for editor vendors

Introduction

When people use something every day, they want it to look 'just so'. Developers are no different. They want their integrating environment to be customizable and open, so they may continue to work with their favourite editor, so they can use several compilers at the same time, and so they can plug in other tools from different vendors.

WorkFrame/2 is an Edit/Compile/Debug environment, designed to be independent of any specific editor, compiler or debugger. A number of programming interfaces have been defined to facilitate this. Through these interfaces, the editor vendors can tightly integrate their product with the WorkFrame/2 product. The user then has the choice of plugging in his preferred existing or new 16-bit and 32-bit tools, making WorkFrame/2 totally customized.

Integration can be accomplished at various levels. Support at any of the levels assumes that all previous levels of support are provided.

1. At the simplest level, the editor does not provide any specific support for the WorkFrame/2. The user is responsible for specifying the editor invocation parameters. WorkFrame/2 invokes the editor in all situations for which editor invocation is appropriate. For example, double clicking on an entry in a files list or on one of the Make File Creation Utility's list boxes will cause the Editor to be invoked with the name of the file(s) to be processed. However, it cannot take advantage of any of the other advanced features of the WorkFrame/2. The OS/2 system editor participates in this manner.
2. The editor can also support the WorkFrame/2 drag & drop protocols. Rather than double clicking on a selected list of files, the user may drag that list of files to an active editor window or icon. The editor can then respond by opening up one or more edit windows. The conventions for dragging files to and from WorkFrame/2 are described later in this article. The OS/2 Enhanced Editor provides this support.
3. The editor can also support the WorkFrame/2 Dynamic Data Exchange (DDE) protocols for processing compiler error messages. If the editor provides this support, then double clicking on a compiler error message in the output window will not only cause the editor to be invoked, but information about the source lines and columns of errors can be communicated to the editor. The OS/2 Enhanced Editor provides this support.
4. The editor may also support the dragging of compiler error messages from the output window to an active edit window or icon. This allows the user to edit a file, save, compile, drag errors to the editor, and repeat the cycle. The OS/2 Enhanced Editor provides this support.

This article discusses the following topics:

1. defining an editor to the WorkFrame/2,
2. dragging files from and to the editor,
3. compiler error message handling,
4. dragging of error messages.

A sample program illustrating all the interfaces described in this document is available from the author. Please mail in the reader reply form at the end of this newsletter to receive this information.

Defining the Editor to WorkFrame/2

WorkFrame/2 provides a dialog that allows the user to configure the editor (see Figure 2). The user can specify the name of the executable file as well as its parameters. The parameters should contain any invocation flags and positions where the name of the selected file or files should be placed. The user can take advantage of the substitution symbol %f or the symbol pair %a %z to define where the names of the selected files should be placed in the invocation parameter.

%f is substituted with the name of the first file selected

%a %z is substituted with the names of all the selected files. Each of the names will be separated by the string which appears between the %a and the %z symbols.

The field entitled "Send compiler errors" should be checked if the editor supports the processing of error messages as described in "Compiler Error Message Handling" on page 15.

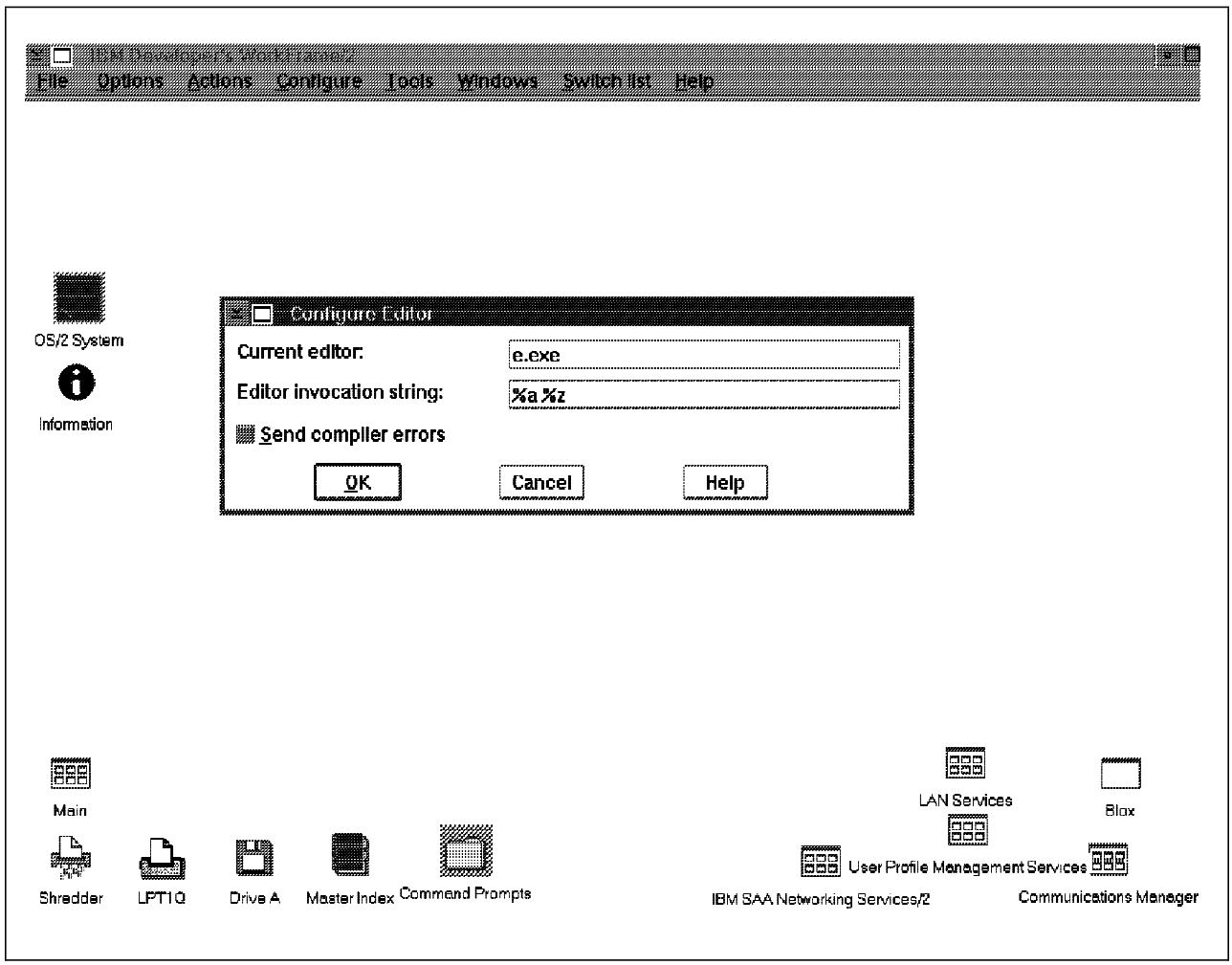

Figure 2. Editor Configuration Dialog

Dragging Files from and to the Editor

WorkFrame/2 can be the source or the target of a drag operation. WorkFrame/2 supports the system interfaces defined by the "Drg" prefixed set of system APIs. The conventions of use are discussed in the following sections.

Target Conventions

Valid Target windows:

1. Project File List

   This semantic is to copy the file into the directory associated with the project.

2. File Control

   This semantic is to copy the file or directory into the directory indicated as the current directory in that window.

WorkFrame/2 will support operations as contained in the DRAGINFO structure of DO_COPY or DO_DEFAULT. Any other operations will be flagged as DOR_NODROPOP in response to DM_DRAGOVER messages.

For each item provided, WorkFrame/2 will only accept a drag if all items are DO_COPYABLE, if not DOR_NODROPOP will be returned. WorkFrame/2 will only support a rendering mechanism of DRM_OS2FILE with a format of either DRF_TEXT or DRF_UNKNOWN. Any others will return DOR_NEVERDROP to DM_DRAGOVER messages. WorkFrame/2 will examine the control flags and reject the drag if DC_REF or DC_GROUP is on. It will interpret a DC_CONTAINER request as a directory. That is the container name plus the source name will be a directory name. WorkFrame/2 will eventually issue a DosCopy for the source file or directory.

If a drag structure passes the criteria described above, WorkFrame/2 will assume the following conventions with respect to the contents of each DRAGITEM structure.

Target Rendering

If the source handle is provided, WorkFrame/2 will attempt to build a fully qualified name for the source and target of the file copy operation. If the container handle is not NULL, the source name will be concatenated to the container name to create a fully qualified source name, otherwise the sourcename will be assumed to be fully qualified. If the target name is provided it will be assumed to be unqualified and it will be concatenated to the directory name associated with

the target window. If it is not provided, then the sourcename will be assumed to be unqualified and used as the target name. This string will be concatenated to the project directory name to create a fully qualified name. A copy operation will take place. After the copy completes, a DM_ENDCONVERSATION message will be sent to the source, MP1 containing the item number supplied in the DRAGITEM structure and MP2 will contain DMFL_TARGETSUCCESSFUL or DMFL_TARGETFAIL as appropriate.

Source Rendering

If the source handle is not provided, WorkFrame/2 will attempt to communicate with the source to complete the rendering. WorkFrame/2 will make this determination by examining the first item in the array of DRAGITEM structures. If the first item requires source rendering then it will be assumed that all items require source rendering. That is, WorkFrame/2 will NOT support a mixture of source and target rendering items in the same DRAGINFO structure.

A transfer structure will be built with a rendering mechanism of DRM_OS2FILE,DRF_UNKNOWN. The rendertoname will be the current project directory name concatenated with the target name, if one is provided. If one is not provided then the directory name followed by a backslash will be provided. The operation will be DO_COPY. If the source requested DC_PREPARE a DM_RENDERPREPARE message will be sent first. If the DM_RENDERPREPARE message returns FALSE then the DM_RENDER message will not be sent (a DM_ENCONVERSATION message will NOT be sent either). A DM_RENDER message will be sent. If the reply flags are set WorkFrame/2 will not support retry. Therefore in this case if DMFL_RENDERRRETRY is set, a DM_RENDERCOMPLETE followed by a DM_ENDCONVERSATION message will be sent. If the DMFL_NATIVERENDER bit is set alone, then WorkFrame/2 will attempt a NATIVERENDER if the source file handle has now been provided. In any case a DM_ENDCONVERSATION message will be sent to the source with DM_TARGETSUCCESSFUL or DM_TARGETFAIL as appropriate. If the flags are not set then the WorkFrame/2 will send a DM_ENDCONVERSATION message.

To net it out, if DM_RENDER returns TRUE, WorkFrame/2 will expect a DM_RENDERCOMPLETE message. If DM_RENDER returns FALSE WorkFrame/2 will take the appropriate action based on the flags BUT will NOT expect a DM_RENDERCOMPLETE

message. In any rendering scenario BOTH sides must free their access to the Drag transfer structure. WorkFrame/2 will do it when receiving a DM_RENDERCOMPLETE message or when FALSE is returned from DM_RENDER.

WorkFrame/2 will keep track of a count of the items being rendered (either natively or with cooperation) and will free the drag structures once all are processed.

Messages Processed by Source

1. DM_DRAGOVER

   Workframe/2 will determine if the objects are droppable (as described above). Only if all operations and all rendering mechanisms are supported will DOR_DROP be returned.

2. DM_DROP

   Workframe/2 will support rendering as described above, both Target and Source rendering are supported.

3. DM_DROPHELP

   A message box will be displayed describing what kind of operation is supported.

4. DM_RENDERCOMPLETE

   This will occur only when the source has not provided a source file name. WorkFrame/2 will examine the source handle, if the handle is NULL it will assume that the file has been completely processed by the source and will simply return a DM_ENDCONVERSATION. If it is provided then it is assumed that the source has updated the container and source string fields and WorkFrame/2 will perform the copy operation as described above.

Source Conventions

Valid Source Windows

1. Project File List
2. File Control

WorkFrame/2 will build a Draginfo structure containing one drag-item structure for each file selected on the project file list. The DRAGINFO structures operation field will be set to DO_COPY. WorkFrame/2 will not support a move operation, i.e. it will assume its list of files is still valid after a successful drag operation.

Each Dragitem structure will be set up as follows:

- item

  Handle of the file list window. Parent is the desktop, owner is the WorkFrame primary window.

- itemid

  listbox item being dragged (number in listbox for future deselection).

- type

  Will always be DRT_UNKNOWN

- rendering mechanism and format

  DRM_OS2FILE,DRF_UNKNOWN

- containername

  WorkFrame/2 will place the handle of the source directory of the file being dragged. The directory name will always contain a trailing backslash.

- sourcename

  WorkFrame/2 will place the handle of the UNQUALIFIED source filename. To fully qualify the filename, concatenate sourcename to containername.

- suggested targetname

  Workframe/2 will place the sourcename handle here. As a result the targetname = sourcename = unqualified name.

- control

  Will always be zero.

- supported operations

  DO_COPYABLE only.

Messages Processed by Source

For each file processed by the Target, WorkFrame/2 will expect a DM_ENDCONVERSATION message. MP1 must contain the item number provided by WorkFrame/2 in the DRAGITEM structure. MP2 will contain DMFL_TARGETSUCCESSFUL if the operation worked or DMFL_TARGETFAIL if the operation failed.

Compiler Error Message Handling

Overview: The WorkFrame/2 will communicate compilation errors to the editor by using Dynamic Data Exchange (DDE) calls. An overview of the interaction between compiler, WorkFrame/2 and editor are depicted in Figure 3

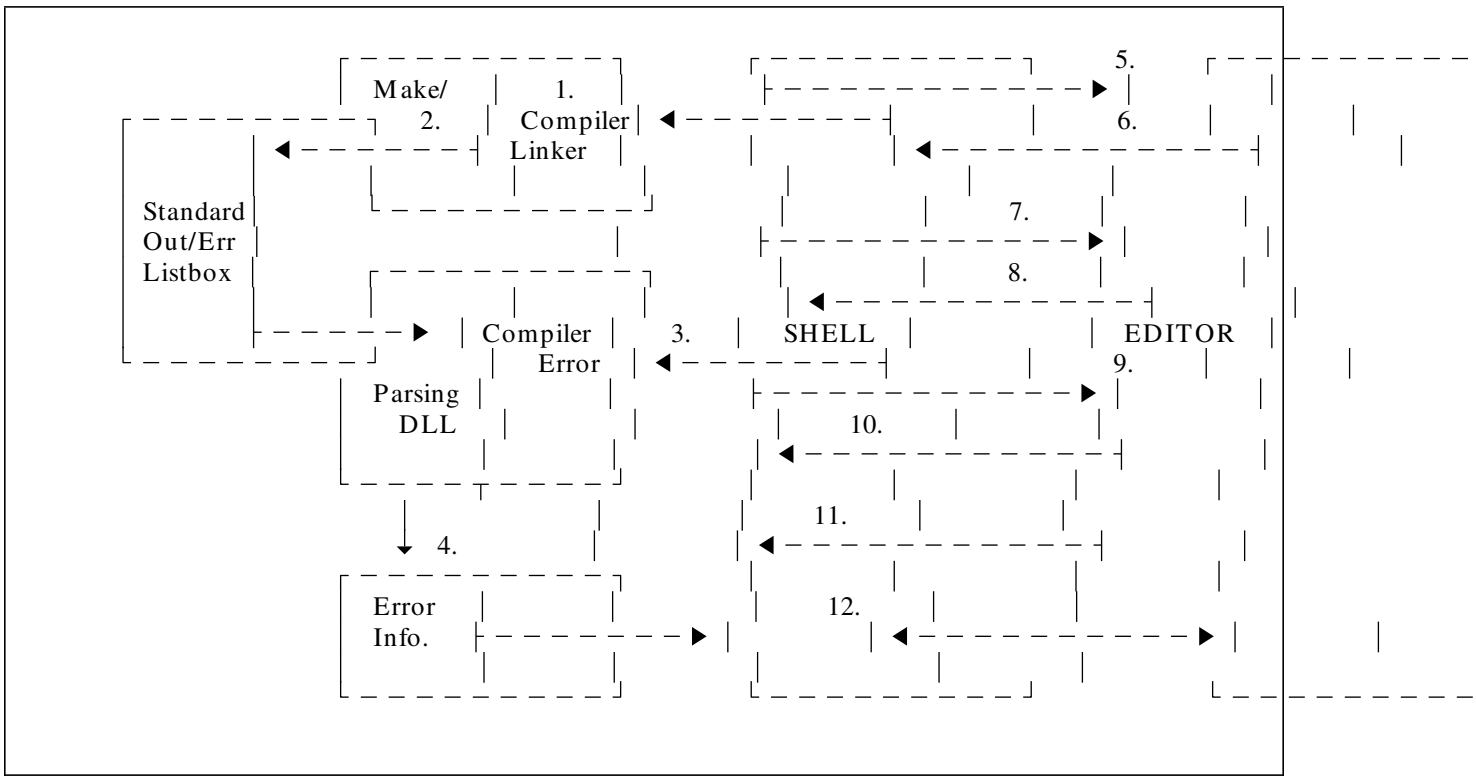

Figure 3. Overview of Compiler/WorkFrame/2/Editor Interaction

1. In response to a Make, Compile, or Build Action, the WorkFrame/2 invokes the Make utility, the Compiler, or the Linker via DosExecPgm.

2. The utility writes its output to one of the Standard I/O handles (Stdout or Stderr). These handles have been redirected back to the WorkFrame/2 and the output is displayed in a listbox.

3. When the user double clicks on an error line[5] in the listbox for the first time, the WorkFrame/2 will invoke the compiler DLL to find all error messages for all files in the listbox.

[5] This process can also be initiated by selecting a line and beginning a drag operation.

4. The compiler DLL returns this information to the WorkFrame/2, at that point the WorkFrame/2 will invoke the DLL to determine the name of the error message help file, if one exists.

5. The WorkFrame/2 then starts the Editor for the source file identified in the error message clicked upon, (via DosStartSession)[6] and issues a WinDdeInitiate call to establish a conversation with the Editor.

6. Once the Editor is initialized (having identified itself as the server for that file), it will receive the connection request for that file. It should accept the request by sending a WinDdeRespond message.

7. Once the acceptance of the connection is received, the WorkFrame/2 will send a WM_DDE_EXECUTE message to tell the Editor to Initialize, passing it information about all source lines in error, and the name of the help file (if one is provided). This message also implies a WM_DDE_ADVICE message, confirming that the Editor may send WM_DDE_DATA requests (see below).

   The EPM editor, for example, takes this information, highlights all lines in error and builds a pulldown menu that allows the user to move from error to error in the source file (see Figure 4 on page 28).

8. The Editor should send a WM_DDE_ACK message back confirming acceptance of the command.

9. As the user continues to double click on different error messages in the listbox, the WorkFrame/2 will send a WM_DDE_EXECUTE message to tell the Editor to "Goto" a particular line (ie original source line), passing it information about that particular line.

10. The Editor should send a WM_DDE_ACK message back confirming acceptance of the command.

---

[6] If a drag operation is in progress then the WorkFrame/2 will issue a DrgDrag request and if a drop is accepted, enable itself for DDE communication

11. The Editor may send a WM_DDE_DATA request to ask for information relating to a particular error line. This will cause the WorkFrame/2 to send a WM_DDE_EXECUTE "Goto" message (see point 9 above).

12. If the listbox dialog is terminated by the user, the WorkFrame/2 will send the Editor a WM_DDE_TERMINATE message. If the user terminates the Edit session the Editor will send the WorkFrame/2 a WM_DDE_TERMINATE message.

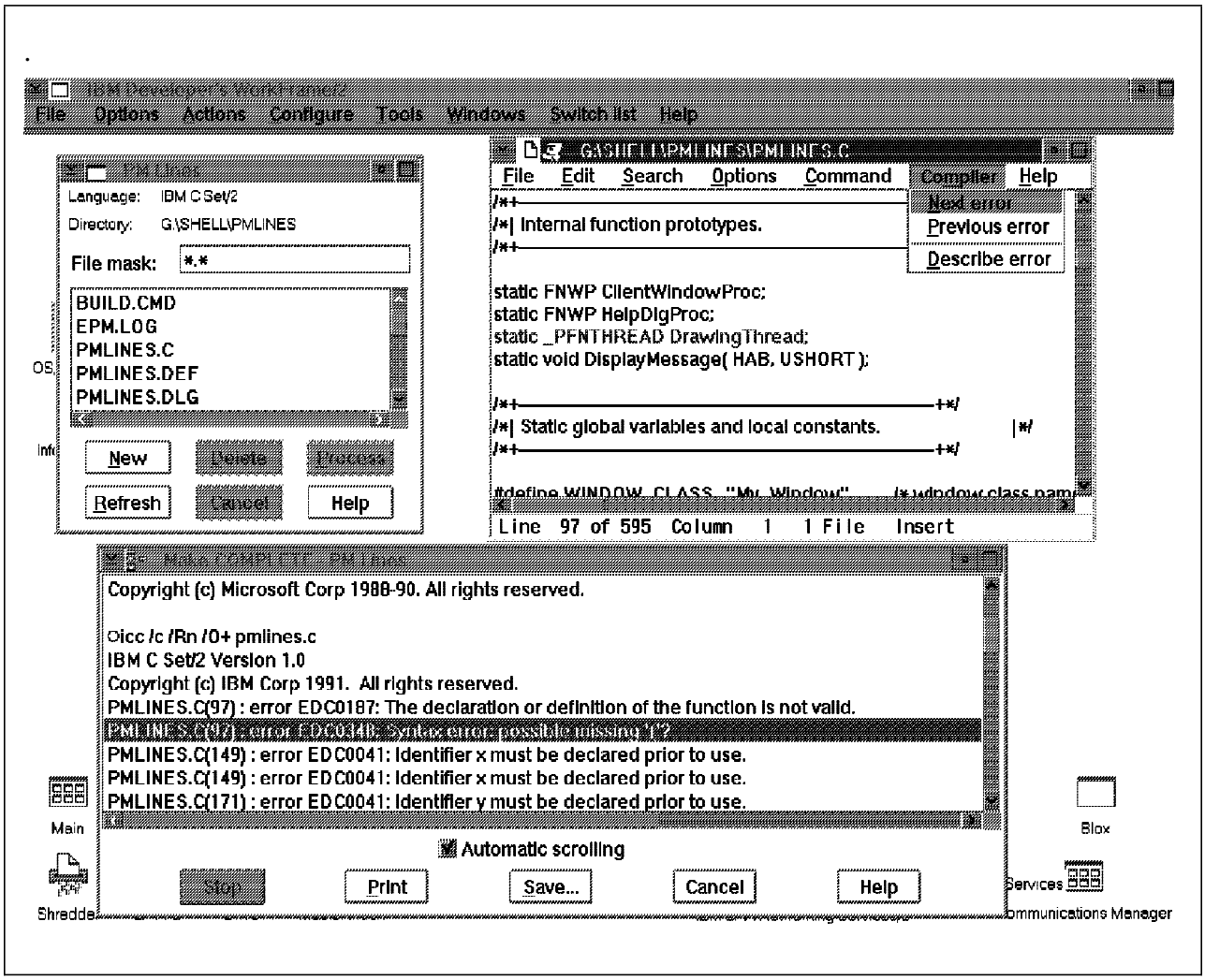

Figure 4. Support for Compiler errors in EPM

WorkFrame/2 - Editor Interaction for Compiler Error Messages

A description of the interaction and the format of the messages follows:

1. In response to a user action, the WorkFrame/2 will start the editor via DosStartSession, passing it the name of the file to be edited.

2. The Editor must register itself as a Server for that file specifying an application name of "WB Server" and using the filename as the topic name.

   That is

```
WinDdeInitiate(hwnd, "WB Editor", szBuf);

where:
        hwnd   - Editor window handle
        szBuf - Name of the file being edited.
```

3. The WorkFrame/2 will then try to establish a DDE session. The request will come to the Editor as a WM_DDE_INITIATE message. Logic can look as follows:

```
case  WM_DDE_INITIATE :
{
  PDDEINIT pddei =  NULL;              /* DDE init struct ptr    */
     /* is this our own broadcast? then don't process it */
  if ((HWND)mp1 = =  hwnd)
    break;

  pddei =  (PDDEINIT)mp2;

  if (!fConnected)   /* could keep track of whether you are connected
                        already and handle only one connection */
    {

/* check to see if the request is from the WorkFrame/2 */
     if ((strcmp("WB Editor", pddei-> pszAppName) = =  0) &&
/* check to see if the request is for the filename stored prev */
       (strcmp(szBuf, pddei-> pszTopic) = =  0))
        {
          hwndDDEClient =  (HWND)mp1; /* store client window handle */

          /* reply that the connection is accepted by passing filename
             as the topic and "WB Editor" as the application name */
          WinDdeRespond((HWND)mp1, hwnd,  "WB Editor", szBuf);
          fConnected =  TRUE;  /* conversation established    */
        }
     }
  DosFreeSeg(PDDEITOSEL(pddei));   /* REQUIRED see msj v4n3 may 89 */
  break;
}
```

4. Once the WorkFrame/2 receives the acknowledgement it will send an initiate message, passing a list of all the lines in error. The Editor could do such things as initializing variables to keep track of original line numbers or colorizing all lines-in-error.

   The message will come in as a WM_DDE_EXECUTE. The format of the message is as follows:

```
ULONG    cbData;         /* size of structure +  item +  data */
USHORT   fsStatus;       /* can be ignored */
USHORT   usFormat;       /* can be ignored */
USHORT   offszItemName; /* offset to message type "Initialize" */
USHORT   offabData;      /* offset to data */
CHAR     message[];      /* "Initialize"   - ASCIIZ */
ULONG    errorcount      /* count of number of errors to follow */
ERRSTR   errors[] ;      /* list of source lines in error */
ULONG    textlength   ;  /* length of library name   (not including */
                         /* the terminating NULL)                    */
char     libraryname[]; /* Asciiz text of library name           */
```

where ERRSTR is a structure consisting of:

```
typedef _errstr
   {
    ULONG errorline;
    ULONG offset;
    ULONG length;
    ULONG magic_cookie;
   } ERRSTR;
```

The "magic cookie" field is really the first line in the error listbox relating to this error. This field can be used in a message from the Editor to the WorkFrame/2 (a WM_DDE_DATA message) to ask the shell to send a "Goto" message for this error. This mechanism allows the Editor to implement a "Next Error" function amd get the error message text related to that error. The errorline and offset fields are provided by the compiler error message parser. In the first release the errorline will be in the range 1 through 65535 and the offset will be in the range 0 through 255. The length field provides information to aid in error highlighting. It tells the editor to highlight the line starting at offset for length. In this release the length will always be zero. The Editor should position the cursor at the offset provided within that line (if possible) and highlight portions of the line beginning at that offset (possibly until the next blank).

If textlength is not zero then a library name is available. The editor should associate this help file with the current help instance. The resource ids, by convention, contained in this file are greater than 20000, leaving the lower 20000 resource ids for use by the editor.

The offszItemName field points to the message array and the offabData field points to the errorcount field. Two macros are provide by OS/2 to convert these offsets into pointers. The pointers can then be used to access the message type and data areas of the message.

```
#define DDES_PSZITEMNAME(pddes) \
        (((PSZ)pddes) +  ((PDDESTRUCT)pddes)-> offszItemName)

#define DDES_PABDATA(pddes)       \
        (((PBYTE)pddes) +  ((PDDESTRUCT)pddes)-> offabData)
```

Sample processing might be:

```
case  WM_DDE_EXECUTE :            /* one time DDE data request        */
  pddeIn =  (PDDESTRUCT)mp2;
  if (pddeIn = =  NULL)
    return ;

     /* Do we have an established conversation with this client? */

  if (hwndDDEClient = =  (HWND)mp1)
  {

        /* is it an initialize request */
    if (strcmp(DDES_PSZITEMNAME(pddeIn), "Initialize") = =  0)
    {
      ULONG  * worklines;

      worklines=  (ULONG *)DDES_PABDATA(pddeIn);
      errorcount= *worklines;
      + + worklines;

      /* now use the count of errors and each error line number
        to initialize the environment for subsequent goto messages */

      /* finally send the client an acknowledgement, it is the
         client's (ie WorkFrame/2's responsibility to free storage)

      WinDdePostMsg(hwndDDEClient,hwnd,WM_DDE_ACK,pddeIn);
    }

  return 0;
  break;                          /* dde_request                      */
```

5. After the conversation has started and the Editor has been initialized for that file, every time the User double clicks on a line in error in the Errors listbox a "Goto" message will be sent to the Editor. The format of the "Goto" message is as follows:

```
ULONG   cbData;        /* size of structure +  item +  data */
USHORT  fsStatus;      /* can be ignored */
USHORT  usFormat;      /* can be ignored */
USHORT  offszItemName; /* offset to message type "Goto" */
USHORT  offabData;     /* offset to data */
CHAR    message[];     /* "Goto"   - ASCIIZ */
ULONG   errorline;     /* original source line in error       */
ULONG   errorsoffset;  /* offset of start of error within line */
ULONG   resourceid  ;  /* resourceid of help in the help file  */
                       /* corresponding to this message       */
ULONG magic_cookie;
ULONG   textlength  ;  /* length of text to follow (not including */
                       /* the terminating NULL)                */
char    errortext[];   /* Asciiz text of error message        */
```

The Editor should scroll the Edit session to the line in error and place the mouse pointer at the offset within that line in error (the

Editor should allow for any tabs expansion since this offset will make no assumption about tabs, ie a tab character is just another character on the line). The Editor is also passed the text of the message so that it could be displayed in an information area. The resource id (if it is not zero) can be used to display help for that error message (via WinPostMsg(hwndHelp,HM_DISPLAY_HELP...))

Sample code is as follows:

```
case  WM_DDE_EXECUTE :            /* one time DDE data request        */
  pddeIn =  (PDDESTRUCT)mp2;
  if (pddeIn = =  NULL)
    return ;

     /* Do we have an established conversation with this client? */

  if (hwndDDEClient = =  (HWND)mp1)
  {

         /* is it a goto request */
     if (strcmp(DDES_PSZITEMNAME(pddeIn), "Goto") = =  0)
     {
       USHORT * work;

       work =  (USHORT *)DDES_PABDATA(pddeIn);
       currenterror =  *work;
       currentoffset =  *(work+ 1);
       currentresid  =  *(work+ 2);
       currentmagic  =  *(work+ 3);
       strcpy(szErrorLine,(PSZ)(work+ 5));
       /* then use the error line, offset and text to position
          the mouse pointer and scroll */

      /* finally send the client an acknowledgement, it is the
         client's (ie WorkFrame/2's responsibility to free storage)

      WinDdePostMsg(hwndDDEClient,hwnd,WM_DDE_ACK,pddeIn);
     }
  }
  return 0;
  break;                          /* dde_request                      */
```

In response to a "Next Error" action the Editor can tell the WorkFrame/2 to Send a "Goto" message containing the text of a particular message. The format of the message is as follows:

```
ULONG   cbData;         /* size of structure +  item +  data */
USHORT  fsStatus;       /* can be ignored */
USHORT  usFormat;       /* can be ignored */
USHORT  offszItemName; /* offset to message type "Goto" */
USHORT  offabData;      /* offset to data */
CHAR    message[];      /* "SendGoto"   - ASCIIZ */
ULONG   magiccookie;    /* identification of which error message */
                        /* information is required                */
```

The message must be "SendGoto" and the data must be the magiccookie. This information is used to initialize the DDE structure and the WM_DDE_DATA message is sent.

```
SEL sel =  NULL;
PDDESTRUCT pdde =  NULL;
USHORT rc =  0;
USHORT usSegSize ;
USHORT totalsize;

/*************************************************************************/
/* 1) Allocate a givable memory segment                                  */
/*************************************************************************/
usDataSize =  4;   /* the size of the magic cookie */
usSegSize =  (USHORT)strlen("SendGoto")+ 1;
totalsize =  sizeof(DDESTRUCT)+ usDataSize+ usSegSize;
rc =  DosAllocSeg(totalsize,            // ddestruct +  topic
                  &sel, SEG_GIVEABLE);
if (rc && sel)                    // check api return code and selector
  return  NULL;

/*************************************************************************/
/* 2) Fill in the new DDE structure                                      */
/*************************************************************************/

pdde =  SELTOPDDES(sel);
memset((PVOID)pdde, 0, (size_t)usSegSize); // zero out memory
pdde-> usFormat =  usFormat;/* the WorkFrame/2 doesn't care in R1) */
pdde-> offszItemName =  (USHORT)sizeof(DDESTRUCT);
memcpy(DDES_PSZITEMNAME(pdde), "SendGoto",usSegSize);
pdde-> offabData =  (USHORT)sizeof(DDESTRUCT)+ usSegSize;
memcpy(DDES_PABDATA(pdde), magiccookie,usDataSize);
pdde-> fsStatus =  0;           // set status flags
pdde-> cbData =  (ULONG)totalsize;

WinDdePostMsg(hwndDDEClient,hwnd,WM_DDE_DATA,pdde);
```

When the user terminates the error message listbox the shell will send a terminate message to the Editor, so that the session can be disestablished. The request will come in as a WM_DDE_TERMINATE message

```
case  WM_DDE_TERMINATE :          /* DDE conversation termination
                                     request                    */
  if (hwndDDEClient !=  (HWND)mp1)
                  break;
  hwndDDEClient =  NULL;        /* indicate that session is over */
                                /* agree on termination */
  WinDdePostMsg((HWND)mp1, hwnd, WM_DDE_TERMINATE, NULL, TRUE);
  fConnected =  FALSE;
  break;
```

If the user terminates the Edit session while the link is still active the Editor must send the WorkFrame/2 a message.

```
case  WM_CLOSE :
  if (fConnected)
    WinDdePostMsg(hwndDDEClient, hwnd, WM_DDE_TERMINATE, NULL,
      TRUE);
```

Dragging of Compiler Error Messages

When a drag operation is started from a monitor window, (see Figure 4 on page 28) WorkFrame/2 will invoke the compiler-provided error parsing interface. If the parsing is successful, WorkFrame/2 will begin a drag operation setting up the Draginfo structure to specify DO_DEFAULT and to contain exactly one DRAGITEM structure. The DRAGITEM structure will be set up as follows:

item	Monitor window handle
itemid	0
type	DRT_TEXT
rmf	< DRM_DDE,DDE3ERRORS>
control	0
supportedops	DO_COPYABLE
containername	handle of string containing project directory with a trailing backslash.
sourcename	handle of string containing file in error, this name will be fully qualified. It is the name extracted from the error message and is the name the receiver of the drag should use as the topic for the subsequent DDE handshaking.
targetname	0
xoffset	Not set
yoffset	Not set

Once the target accepts the drag, it should post itself a message to perform the DDE conversation. That is the DDE conversation should not be started until you return from receiving the DM_DROP message. The mechanics of the DDE conversation are discussed in "Compiler Error Message Handling" on page 15.

Ouch!

Our apologies to our many readers outside of North America. We didn't mean to leave you in the cold by printing only US order numbers for C Set/2 and associate products. A sincere sorry in all your languages, and here is the missing data.

Product Summary

Product	Order No.	Part No.
IBM C Set/2 V1.0	5621-303	01G8923
IBM WorkFrame/2 V1.0	5621-302	01G8927
IBM OS/2 2.0 Developer's Toolkit	5621-078	01G8925

A word from your editor

If you didn't receive this newsletter mailed direct from IBM, and you would like regular hardcopy, then let us know. Mail in the reply form at the back with your request to join us, plus your full mailing address, and we'll add you to our mailing list, wherever you are. Yes, wherever you are...from Argentina to Zaire, if you have a mail service, we'll mail you our newsletter.

For the many of you sending in the reply forms with your comments, we've often had the need to call you to discuss your comments further. It's a great help if you include your phone number. Thanks!

Next issue? Customer satisfaction .... WorkFrame/2 API's for compiler vendors .... more technical Q&As ....and more.

This newsletter was produced by the OS/2 C Planning department of the IBM Canada Toronto Lab. For further information on any of the products mentioned, please contact your local IBM office, or an authorised IBM Business Partner.

This newsletter was created and marked for processing using IBM BookMaster (Program Number 5688-015) and IBM Document Composition Facility (DCF)™ (Program Number 5748-XX9).

The final copy was printed on an IBM 3825 Page Printer, an Advanced Function Printer.

This newsletter is © IBM Corporation 1992.

IBM

OS/2 Compiler News April 92 edition

Reader's Comment Form

1. *Did you find this newsletter useful?*
2. *Is there any way you think we could improve this newsletter?*
3. *Is there any compiler-related subject you would like to see addressed in this newsletter?*

Please note:

- *IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you, and all such information will be considered non-confidential.*
- *Do not use this form to report compiler problems or to request copies of publications. Instead, contact your IBM representative or an authorised IBM Business Partner.*
- *If you wish, you may include your name, address, and company name if applicable, and your phone number.*

_____

_____

_____

_____

_____

Thank you for your cooperation and help. You can either mail this form to us, or hand it into an IBM office for forwarding.

OS/2 Compiler News April 92 edition

Reader's Comment Form

Fold here and tape.....fold here and tape.....fold here and tape.....fold here and tape.....fold here and tape.....fold here and tape.....fold here and tape.....

IBM

IBM Canada Ltd
Workstation Languages Planning
Dept 394
22/394/844/TOR
844 Don Mills Road
North York
Ontario, M3C 1V7
Canada

Fold here and tape.....fold here and tape.....fold here and tape.....fold here and tape.....fold here and tape.....fold here and tape.....fold here and tape.....