SHELL PROGRAM INTERFACE

WordPerfect Corporation

Version 1.0
February, 1986

SHELL PROGRAM INTERFACE -- TECHNICAL SPECIFICATION

This specification contains the information necessary to write progr interact with WordPerfect's Shell program manager. The following su the information contained in each chapter of this document.

Chapter 1 -- Introduction to the Shell

This chapter introduces the Shell program manager in general, it's c and the benefits to be gained by writing programs that interact dire Shell.

Chapter 2 -- Writing programs that interact with the Shell

This chapter gives some basic guidelines for writing programs that i the Shell. It contains the general steps necessary for a program to the Shell.

Chapter 3 -- Testing for the presence of the Shell

This chapter describes the method that programs should use to determ Shell is present in the system. An example procedure (in assembly l given to illustrate the method.

Chapter 4 -- Shell Functions

This chapter details each of the available Shell functions, and give parameters, results, possible error codes, and examples in assembly each.

Chapter 5 -- User Interface guidelines and suggestions

This chapter gives some basic guidelines on implementing the Shell i that the user will be able to take advantage of all of the Shell's c

Appendix

A - File Formats for the Clipboard
B - Memory Considerations

CHAPTER 1 -- INTRODUCTION TO THE SHELL

The Shell is a program manager/integrator developed by WordPerfect C to help integrate WordPerfect, MathPlan, and other programs.

The Shell provides the user with a program menu from which programs, commands, batch files, and utilities can be executed. The user can menu to include whatever programs he or she wishes.

The user can load several programs into memory concurrently, and swi between those programs at will under the Shell. The Shell also prov "clipboard", or buffer, for moving data (text, numbers, graphs, etc. between programs.

Included with the Shell in the WordPerfect Library package are sever utility programs, such as a calculator, a calendar, a notebook, a DO manager, and others -- all of which can cut and paste information to the clipboard.

A program that can interact directly with the Shell can take advanta following features and capabilities:

1. Suspend operation and transfer control to the Shell manage allows the user to load other programs concurrently and sw and forth between programs with the press of a key. For e the user is able to pop into any of the desktop utilities calendar, notebook, etc.) that are included with the Shell

2. Write or append ASCII text to the Shell's clipboard. This user to easily move data from one program to another.

3. Retrieve a copy of text in the Shell's clipboard.

4. Take of advantage of Expanded Memory, if available. If th an Intel AboveBoard (c), or other Expanded Memory board th the Lotus/Intel/Microsoft Specification, the Shell will au swap programs out to the the expanded memory above 640K. allows the user to load up many more programs at once than normally fit in 640K of conventional memory.

All programs, even those that do not interact directly with the Shel advantage of the following Shell features and capabilities:

1. Start the program from the Shell menu with the press of a user can specify which directory to make the new default d before starting the program. Command line arguments can a specified for each program on the Shell menu. The user ca create sub-menus for organizing their hard disk and progra logical groupings.

2. Use the Shell keyboard macro facility. This allows the us a series of keystrokes and assign them to either an ALT-SH combination or to a filename on disk. The keystrokes that recorded can be played back from any program. There is no (except disk space) to the number or length of Shell macro be created.

3. Execute DOS commands, batch files, or Shell utilities from menu.

CHAPTER 2 -- WRITING PROGRAMS THAT INTERACT WITH THE SHELL

In order for a program to interact directly with the Shell, it must following basic steps:

1. Test to see whether or not the Shell is present in the sys method for doing this is described in detail in Chapter 3.

2. If the Shell is present, use Function 50h to request a uni ID code. This code is then used in subsequent function ca Shell.

3. Release any memory that is not needed. The program should determine what memory it needs for data, buffers, etc., an any extra memory using DOS function 4Ah.

   When DOS loads a program into memory, all available memory assigned to that program. In order for the Shell to be ab other programs concurrently into memory, programs that int the Shell must release extra memory by "shrinking" down wi function 4Ah. This should be done as soon as the program that it has been loaded under the Shell.

   Most WordPerfect programs use less memory for buffers, etc loaded under the Shell than they do when loaded normally f This makes it possible to load several program concurrentl Shell.

4. Use Shell Function 57h to determine if the Shell is loadin program as part of a "start-up" procedure. If so, the pro do any initialization, then transfer control back to the S Function 50h) before modifying the screen or asking for an input.

5. Check user input where applicable for the "Shell" key (Ctr use the Shell functions to interact with the Shell (i.e., temporarily, write or append data to the clipboard, get da clipboard, etc.)

6. Before terminating, the program should issue Function 55h permission to exit) so that the Shell can insure that all in the reverse order that they were loaded into memory. T necessary for DOS to operate properly.

Chapters 3 and 4 describe how to test for the Shell and make functio the Shell.

CHAPTER 3 -- TESTING FOR THE PRESENCE OF THE SHELL

This chapter describes how to use the DOS "get interrupt vector" tec test whether or not the Shell is present. Any program can use these test for the presence of the Shell.

1. Use the DOS "get interrupt vector" command (DOS function 35h) t contents of interrupt vector number 1Ah (addresses 0000:0068 th 0000:006B).

   The Shell uses this interrupt vector to perform all interaction application programs and the Shell manager. The OFFSET of the service routine address is stored in the word located at addres the SEGMENT address is stored in the word located at address 00

2. Compare the contents of an ASCII string which starts at the add specified by the contents of the interrupt vector 1Ah plus a fi 2 bytes to the string "SHELL001".

   If the result of the string comparison is positive, the Shell i can be called via interrupt 1Ah.

Example

The following code is an example of the technique described above to if the Shell is present.

```
;*****************************************************************
;
;    The following procedure determines whether or not the Shell
;    is present in the system.
;
;    RESULT:   CARRY FLAG is SET   if the Shell is present
;              CARRY FLAG is CLEAR if the Shell is not present
;
;*****************************************************************

Test_For_Shell PROC NEAR

     PUSH      AX
     PUSH      BX
     PUSH      CX
     PUSH      SI
     PUSH      DI
     PUSH      DS
     PUSH      ES

     PUSH      CS
     POP       DS              ; prepare DS for string compare

     MOV       AH,35h
```

```
     MOV       AL,1Ah          ; issue "get interrupt vector"
     INT       21h

     MOV       DI,BX
     ADD       DI,2

     MOV       SI,OFFSET ascii_name

     MOV       CX,8
     CLD
     REPE      CMPSB
     JNE       Test_No

Test_Yes:
     STC
     JMP       Test_Exit

Test_No:
     CLC

Test_Exit:
     POP       ES
     POP       DS
     POP       DI
     POP       SI
     POP       CX
     POP       BX
     POP       AX
     RET

ascii_name: DB "SHELL001"

Test_For_Shell ENDP
```

CHAPTER 4 -- SHELL FUNCTIONS

All interaction with the Shell manager is performed by issuing inter
The contents of the registers before and after issuing the interrupt
depending on the Shell function being called. The function number i
passed in register AH. For all functions except Function 50h (Get P
the program ID code is passed in register AL.

For all functions, register AH returns an error status code. The po
status codes are listed for each function. In general, if AH is zer
executed normally. If AH is non-zero, an error of some kind occurre

For all functions, if AH is 0FFh on return, an invalid Function code
passed in AH.

For each function, the input parameters, results, possible error cod
example in assembly language are given.

Function 50h -- Get Program ID

This function is used to return a unique program ID code to the call should be the first function used by the calling program, and should soon as the program determines that the Shell is present.

For all subsequent function calls, the program ID code should be pas register AL.

IN: AH contains the function number (50h)

OUT: AL contains the Program ID code
AH contains an error code

EXAMPLE:

```
MOV  AH,50h
INT  1Ah
OR   AH,AH
JNZ  error_handler
```

ERROR CODES:

```
AH = 0    No error.
AH = 1    No Program ID code available.
```

Function 51h -- Go to Shell

This function is used to temporarily suspend the calling program and control to the Shell manager. The Shell menu will be displayed and allowed to select from among the entries on the menu.

When the user elects to return to the calling program, the Shell tra control back to the caller by returning from the interrupt.

It is very important that the calling program check the error status on return from this interrupt. The user may have elected to exit al that are loaded under the Shell, in which case the caller will need shop" and terminate.

IN: AH contains the function number (51h)
AL contains the Program ID (returned by Function 50h)

OUT: AH contains an error code

EXAMPLE:

```
MOV  AH,51h
MOV  AL,program_ID
INT  1Ah
OR   AH,AH
JZ   continue_with_program
DEC  AH
JZ   unable_to_transfer_to_shell
DEC  AH
JZ   exit_program
DEC  AH
JZ   exit_program_but_save_info_first
```

ERROR CODES:

AH = 0	No error, continue with program.
AH = 1	Unable to transfer control to Shell -- continue program or issue error message.
AH = 2	User wishes to exit all programs that are loaded Shell. Close up files, etc., and terminate.
AH = 3	User wishes to save any information (documents, etc.) that has been modified, and then exit all are loaded under the Shell. Save any updated in (this may require user interaction), and termina

Note: When this function is called, the Shell saves the following in about the caller's "state" on the caller's stack:

1. ALL registers except AX
2. All flags
3. The current cursor position and mode

4. The current DOS DTA address
5. The following interrupt vectors:
   - Divide overflow
   - Critical error handler

The above "state" is restored before the Shell returns control to th

If AH contains a 2 or 3 (terminate) code on return from this functio calling program should still issue Function 55h (Request permission before actually terminating.

Function 52h -- Write Data to Clipboard

Use this function to write the data specified by DS:SI and CX regist
clipboard. Any previous information contained in the clipboard is l
Function 53h to append or add data to the clipboard.

The clipboard is virtual in nature, so if the data is too large for
buffer reserved for the clipboard, a temporary file will automatical
on disk, and the additional information will be stored there.

IN: AH contains the function number (52h)
AL contains the Program ID code
BL contains a format type code as follows:
1 = data is in WP text format
2 = data is in WP Merge File format
3 = data is in ASCII text format
(see Appendix for explanation of format codes)

CX contains the number of bytes to write
Note: Calling programs should write or append no more
bytes at a time to the clipboard. The amount of data
be stored in the clipboard is limited only by disk sp
maximum "chunk" that the Shell can handle per call is
DS:SI contains the segment and offset of the location in m
where the data to be written is stored.

OUT: AH contains an error code

EXAMPLE:

```
MOV  AH,52h
MOV  AL,program_ID
MOV  BL,format_type
MOV  CX,number_of_bytes
MOV  SI,offset_of_data        ; DS is segment of data
INT  1Ah
OR   AH,AH
JNZ  error_handler
```

ERROR CODES:

AH = 0 No error.
AH = 1 No room to add data to clipboard (Disk Full erro
AH = 2 Invalid format type.

Function 53h -- Append Data to Clipboard

This function is similar to Function 52h, but appends the data to th clipboard.

IN: AH contains the function number (53h)
AL contains the Program ID code
BL contains a format type code as follows:
1 = data is in WP text format
2 = data is in WP Merge File format
3 = data is in ASCII text format
(see Appendix for explanation of format codes)

CX contains the number of bytes to write
No more than 5K bytes at a time (see note under Funct - Write to Clipboard)
DS:SI contains the segment and offset of the location in m where the data to be appended is stored.

OUT: AH contains an error code

EXAMPLE:

```
MOV  AH,53h
MOV  AL,program_ID
MOV  BL,format_type
MOV  CX,number_of_bytes
MOV  SI,offset_of_data        ; DS is segment of data
INT  1Ah
OR   AH,AH
JNZ  error_handler
```

ERROR CODES:

AH = 0 No error.
AH = 1 No room to add data to clipboard (Disk Full erro
AH = 2 Invalid format type.

Function 54h -- Retrieve Data from Clipboard

Use this function to retrieve data from the clipboard. The number o returned in CX. The calling program should continue making this fun until CX equals 0, meaning there is no more data to be retrieved fro clipboard. If CX is zero on the first call, the clipboard is empty.

Function 56h can be used to determine whether the data in the clipbo text format (WP or ASCII) or in a mail-merge format (WP Merge File).

IN: AH contains the function number (54h).
AL contains the Program ID code.
BL contains a format type code as follows:
1 = request data in WP text format
2 = request data in WP Merge File format
3 = request data in ASCII text format
(see Appendix for explanation of format codes)

OUT: AH contains an error code
CX contains number bytes of data returned (0 = empty)
Note: The Shell will hand back approximately 128 byte
The program should continue calling Function 54h unti
ES:DI contains the segment and offset of the buffer where
located.

EXAMPLE:

```
Read_clipboard:
          MOV  AH,54h
          MOV  AL,program_ID
          MOV  BL,format_type
          INT  1Ah
          OR   AH,AH
          JNZ  error_handler
          JCXZ no_more_data

          ; PROCESS CX BYTES AT ES:DI

          JMP  Read_clipboard
no_more_data:
```

ERROR CODES:

```
          AH = 0    No error.
          AH = 1    Disk I/O error -- can't retrieve data
          AH = 2    Invalid format type request (i.e., requested dat
                    Merge File format when data in clipboard was WP
                    ASCII text)
```

Function 55h -- Request Permission to Exit

Because the Shell uses DOS to load programs and manage memory, progr
be exited in the reverse order that they were loaded into memory. T
that a program request permission from the Shell just prior to termi
Shell will return control to the program when it is time to exit.

IN: AH contains the function number (55h)
AL contains the Program ID code

OUT: AH contains an error code

EXAMPLE:

```
MOV  AH,55h
MOV  AL,program_ID
INT  1Ah
OR   AH,AH
JNZ  go_ahead_and_terminate
JMP  restart_program
```

ERROR CODES:

```
AH = 0    Restart program -- user wishes to re-enter progr
AH <> 0   Go ahead and terminate program.
```

Note: The calling program should use DOS function 4Ch to terminate.

Function 56h -- Check Clipboard Format

Use this function to return a format type code indicating what type currently in the clipboard (text or mail merge format).

IN: AH contains the function number (56h)
AL contains the Program ID code

OUT: AH contains the format type code:
0 = data is text (WP or ASCII)
1 = data is mail-merge format (WP Merge)

EXAMPLE:

```
MOV  AH,56h
MOV  AL,program_ID
INT  1Ah
```

ERROR CODES:

No errors are defined for this function

Function 57h -- Check if Shell is installing program resident

This function should be issued by the calling program after it has r program ID code, and after it has released any extra memory, but bef modifies the screen or does any user interaction.

If the function returns a 1 in AL, the user wants to load this progr memory as he (she) is starting the Shell, but does not wish to begin program until they select it from the Shell menu. If this is the ca program should go ahead and complete any initialization required, an Function 1 to transfer control to the Shell manager.

IN: AH contains the function number (57h)
AL contains the Program ID code

OUT: AH contains an error code
AL = 1 if program should transfer control to Shell immedia starting.

EXAMPLE:

```
MOV  AH,57h
MOV  AL,program_ID
INT  1Ah
OR   AH,AH
JNZ  error_handler
OR   AL,AL
JZ   continue_with_program

MOV  AH,51h              ; transfer to Shell
MOV  AL,program_ID
INT  1Ah
OR   AH,AH
JZ   continue_with_program
DEC  AH
JZ   continue_with_program
DEC  AH
JAE  exit_program

continue_with_program:
```

ERROR CODES:

```
AH = 0    No error
AH = 1    Invalid program ID code
```

Function 58h -- Test if Shell is using Expanded Memory

This function allows programs to determine whether or not the Shell Expanded Memory. If so, the programs do not need to release memory they start, but can take as much or all of the 640K conventional mem wish.

IN: AH contains the function number (58h)

OUT: AL contains 1 if Expanded Memory is being used.

EXAMPLE:

```
MOV  AH,58h
INT  1Ah
OR   AL,AL
JZ   no_expanded_memory
JMP  can_use_all_conventional_memory
```

ERROR CODES:

No errors are defined for this function

CHAPTER 5 -- USER INTERFACE GUIDELINES AND SUGGESTIONS

All programs (to date) that interact with the Shell use Ctrl-F1 as t key. We highly recommend that all programs that interact with the S Ctrl-F1 to go to the Shell, interact with the clipboard, etc.

One of the features of the Shell is the capability for the user to s programs by pressing an ALT-SHIFT-letter key combination. This feat implemented as a Shell macro, and assumes that the normal procedure the Shell consists of pressing Ctrl-F1 and then 1. For this reason, recommend that a menu be displayed when the user presses Ctrl-F1, wh first option is Go to Shell. This will allow the user to use the "s method of switching to and from your program.

We suggest that programs which intend to support the Shell directly through the Shell User's Manual (particulary the Program Profiles in Appendix) to get a feel for how WordPerfect programs interact with t

APPENDIX

A. File Formats

Information can be passed to and retrieved from the clipboard in one following formats:

1. WP Text -- WordPerfect Text format. Data in this format con characters interspersed with WordPerfect function codes. WordP function codes range from 128 to 255 decimal. "Extended charac would normally occupy this range are preceeded and followed by "gate" -- the function code 0E1h. Newlines are indicated by a linefeed (0Ah). Only programs which are familiar with WordPerf function codes should attempt to read or write to the clipboard format.

2. WP Merge File -- WordPerfect mail-merge format. This is the WP Text (text interspersed with WP function codes), but the tex formatted into a field & record orientation. A ^R and linfeed each field and a ^E and linefeed (0Ah) follow each record. Thi should only be used by programs which are able to strip out Wor function codes.

3. ASCII Text -- ASCII Text format. Data in this format consis normal ASCII text. Newlines are represented by a carriage retu (0Dh)/linefeed (0Ah) combination. Codes above 128 are assumed extended characters (foreign, line-drawing characters, etc.) T and should be used by all programs which need to communicate wi clipboard, but do not wish to worry about WordPerfect function

B. Memory Considerations

As explained in Chapter 2, programs which interact with the Shell ne release any memory that they do not need when they start. Many prog normally take all available memory. In this case, the program shoul exception when being loaded under the Shell, take a reasonble amount (or allow the user to specify how much to use) and release the rest other Shell programs.

The Shell itself takes about 25K of memory.

The Shell supports the Lotus/Intel/Microsoft Expanded Memory Specifi The Shell will automatically sense if Expanded Memory is availabe an programs out to Expanded Memory when necessary. When the Shell is r Expanded Memory, each program can use all or as much of the 640K conventional memory as it desires. When programs are not currently they are swapped out to Expanded Memory by the Shell.