

CUSTOMIZATION GUIDE

Custom Commands

Command Override

XyWrite's command override feature lets you create your own set of commands. You can redefine the functions associated with existing immediate commands CALL and DIR and function calls (such as CL and YD), or you can create new command names.

Custom command sets are stored in a separate customization file and loaded into memory with the LOAD command. Once the command set has been defined and loaded, you activate it with the OO (Override On) and OX (Override Off) commands.

CUSTOMIZATION GUIDE

Custom Commands

Creating a Custom Command Set

NOTE Due to the limitations of the software used to create this manual, some of the ASCII characters used in XyWrite could not be reproduced. In such instances, the convention used to indicate those characters is `[ascii#n]` – where `n` is the number, used in conjunction with `[Ctrl]/[Alt]`, which produces the appropriate special ASCII character. It will be colored blue to indicate that it links to a [table](#) which shows what the character looks like.

To create a custom command set:

1. Create the customization file. For example:

Type: [F5] new command.set

2. On the first line, type the label that identifies this file as a command set. Be sure to type it in uppercase.

Type: ;U2;

3. Create the first custom command. The format of each immediate command in the set is:

`<<5,^name>>` (or `<<5,^name>>` – see [Note #1](#))

`[ascii#2]code[ascii#2]`

Where *name* is the name of the immediate command or function call you are redefining or the name of the immediate command you are creating and *code* is the XPL (XyWrite Programming Language) instruction you want the command to execute.

Let's take a look at a simple example (a more complex example is presented at [the end of this section](#)). We'll create a new command named GET that calls a frequently referenced appointment file named DATES.

- a. Type the command label followed by a carriage return.

Type: `<<5,^get>>` [↵]

- b. Type ASCII character 2.

Type: [Ctrl][Alt]2 (on the numeric keypad)

- c. Write the program for command you want to execute (refer to [“User Programming”](#) for instructions on writing XPL programs). We recommend that you start all programs with instructions to disable command override. For our example, the program would look like this:

BX oxQ2 **BX** call c:\xy4\docs\datesQ2 **BX** ooQ2 `<<ex>>`

Note that the last two commands turn on command override and exit the program.

- d. Type an ASCII character 2 to mark the end of the code for this command.

4. Repeat step 3 for every command you want to redefine or create.
5. Store the file.

Custom Commands Using the Custom Command Set

NOTE Due to the limitations of the software used to create this manual, some of the ASCII characters used in XyWrite could not be reproduced. In such instances, the convention used to indicate those characters is [ascii#n] – where n is the number; used in conjunction with [Ctrl][Alt], which produces the appropriate special ASCII character. It will be colored blue to indicate that it links to a [table](#) which shows what the character looks like.

To use the custom command set:

1. Load it into memory. For example:

Type: [F5] load command.set

2. Turn on command override.

Type: [F5] oo [↵]

3. Issue any of the commands in the custom set. If you redefined a standard XyWrite immediate command or function call, the new definition will be applied.

4. To restore the standard XyWrite functions and commands, turn off command override.

Type: [F5] ox [↵]

NOTE #1 – Command Labels: If you are redefining or creating a new immediate command, type two carets (^ ^) before name. If you are redefining a function call, type one caret before name.

NOTE #2 – Function Calls: You should use caution when redefining function calls or you may not get the results you expect. We recommend that you avoid redefining the function calls **BC**, **BX**, **Q2**, and **XC**.

NOTE #3 – VA Command: You can check the status of command override with VA\$OO. A value of 0 means it is off; a value of 1 means it is on.

NOTE #4 – Command Line: If you need the contents of the command line, your XPL program should include a command to save it before you execute any other BX or BC functions. For example, to save the contents of the command line to macro 01, insert the following instructions at the beginning of the XPL code that defines your custom command:

```
<<sx01, <<va$cl>> >>
```

CUSTOMIZATION GUIDE

Custom Commands

Using the Custom Command Set (Cont.)

Example

```
{{5,GetArg}}
```

```
[ascii#2] <<sv01,^>> <<XS40,01,02,03,04>> <<SX10, <<VA$CL>> >> <<XS10,04,05,06,07>>  
<<SX40, <<IS07>> >>[ascii#2]
```

```
←
```

```
{{5,^open,^o}}
```

```
[ascii#2] <<SX40, <<VA$FR>> >> JM(2.GetArg) BX oxQ2 BX call <<PV40>> Q2 BX ooQ2[ascii#2]
```

```
←
```

```
{{5,^write,^w}}
```

```
[ascii#2] <<SX40, <<VA$FR>> >> JM(2.GetArg) BX oxQ2 BX save <<PV40>> Q2 BX ooQ2[ascii#2]
```

This example defines two new commands: Open (abbreviated “O”) and Write (abbreviated “W”). In both commands, VA\$FR captures the name of the command override routine that was accessed (^open, ^o, ^write, or ^w) and stores it to macro 40.

The program then jumps to a routine called GetArg, which captures any argument that was typed after the custom command, including any switches, such as /NV. Notice that the JM function call uses “2.” at the beginning of the jump. The “2.” indicates that “GetArg” is in the U2 customization file.

GetArg parses the ^^ from the command, storing the result in macro 04. It then use VA\$CL to capture the text from the entire command line in macro 10. Next, macro 04 is parsed out of macro 10, and the results are saved in macro 07. For example, if you type the command:

```
w/nv newfile.doc
```

The string “/nv newfile.doc” is saved in macro 07. Finally, macro 07 is saved back to macro 40.

In the original routine, macro 40 is put out as part of the XyWrite command. In the example, W/NV NEWFILE.DOC is executed as SAVE/NV NEWFILE.DOC.