

CUSTOMIZATION GUIDE

User Programming

User programming involves using XyWrite Programming Language (XPL) to write a variety of customized application programs. You can make your programs as simple or as sophisticated as you want - you can record any sequence of keystrokes and create a program for later execution or you can use XPL commands to add more power and functionality to your programs.

NOTE Programming Background *This section contains advanced programming material and is not recommended for beginners. You need some programming background to get the full benefits of the programming features.*

User Programming extends the power and functionality of XyWrite. You can create new functions based on combinations of existing functions to bring convenience and speed to your daily work.

You can create procedures that accept keyboard input, such as an order entry system. You can also write user programs to accomplish complex editing tasks that conditionally require a change to be made. For example, you can create a program that removes list entries dated prior to a certain date.

User Programming commands enable you to:

- Save values to 1000 macros (00-999) during program execution
- Save strings, subroutines, or expressions in a macro
- Use the contents of a macro in an expression
- Divide the contents of a macro into parts, extract the parts, and then save them to other macros
- Insert the contents of a macro at the cursor position
- Clear macros from memory
- Branch on a condition (IF Statement)
- Test for errors
- Stop to wait for keyboard input any time during program execution
- Pass values to a program as it starts
- Determine the current cursor position
- Make use of the current filename, path, page number, line number, and other XyWrite values and settings
- Label a point in the program
- Jump to a label
- Switch to the window that contains a specific file
- Use operators and functions to perform various tasks in your program
- Produce an audible signal at a point in your program
- Use the XPL debugger when testing your program to pause execution at certain points
- Execute a command from within a program without clearing the command line

CUSTOMIZATION GUIDE

User Programming

Procedures for User Programming

Creating And Running An XPL Program

The following provides an overview of the basic procedure for creating a user program. An explanation of most of the commands and functions described here are provided elsewhere in this section.

Step 1 - Plan the Program

Analyze the steps you'd take if you manually performed the task. You can then prepare a list of instructions to tell your computer how to do the same task automatically. If the task is complex, you may find it helpful to draw a flowchart. Consider all possibilities and try to anticipate problems.

Step 2 - Create or Call a Program File

Use the NE (New) command to create a program file or the CA (Call) command to open one.

Step 3 - Write the Program

Turn on program mode and type the exact keystrokes for the procedure you want. [Scroll Lock] turns program mode on and off. When program mode is on, XyWrite displays an "S" at the top right of the screen, displays a message on the status line, and records all your keystrokes as function calls in your program file.

You use the **PFUNC (Put Function Call)** command to insert function calls not assigned to keys into your program. (Be sure to turn off program mode before using the PFUNC command.)

When you are writing XPL programs, you can use macros to save strings or values. XyWrite provides **one thousand programming macros, numbered 00 to 999**, for use in XPL programs.

Record the keystrokes that represent the actions you want to occur when the program begins execution. When you reach a point in the program where you want to insert an embedded command, turn off program mode by pressing [Scroll Lock]. Type in the desired embedded commands. Each time, press [F5] to move to the command line, enter the command, and press [F9].

You can make your programs easier to read by including notes or comments throughout the program file; for example, to explain the program's purpose or to identify different sections of the program. Use the string **;** in a program file to indicate that the text that follows is a comment. A comment can be as long as you want; and you can include as many different comments in a file as you need. A carriage return marks the end of a comment.

When you are finished entering the program, be sure to turn off program mode.

Step 4 - Store the Program

When the program is as you want it, store it on disk. You cannot run your program until it is stored on disk.

Step 5 - Test and Debug the Program

Try out your program to see if you get the results you expected. Use the RUN command to execute the program.

Default **DB (Debugger)** enables you to execute your program one element at a time so you can verify that the program is doing what you want. Turn debugging on before you run your program by setting default DB to one of the values from 1 to 7. Refer to "Default Settings" for more information.

Step 6 - Load the Program to a Key

This step is optional, but it makes using your program easier. You use the **LDPM (Load Program)** command to load a program created in program mode to a key.

If you want to keep the program file loaded on the macro key for use after you quit XyWrite, use the **STSGT (Store Macro)** command.

User Programming

Procedures for User Programming (Cont.)

Entering XPL Commands

Many XPL commands are embedded commands, and you enter them in your program as you would any other embedded XyWrite commands. Before inserting an embedded command, be sure program mode is off. After you press [F9] to enter certain of these embedded commands in your program, XyWrite opens a command window so you can include an argument for the command. Press [Shift][F1] to close the window when you have entered the appropriate argument. (To abort the command window, press [Esc].)

For example, to enter the command: **«SX01,«RC»»**

1. Enter the SX command:

Type: [F5] sx [↵]

2. Type the rest of the command in the command window:

Type: 01,[F5]rc [↵]

Press: [Shift][F1]

You can also use expanded view for typing or modifying commands.

For example, to enter the command: **«SX01,«RC»»**

1. Press: [Ctrl][F8] to switch to expanded view.
2. Move the cursor where you want to insert the command, and:

Press: [Ctrl][<]

Type: SX01,

Press: [Ctrl][<]

Type: RC

Press: [Ctrl][>]

Press: [Ctrl][>]

3. Press: [F8] to return to draft view.

Revising XPL Programs

To revise a user program, you need to switch to expanded view to reveal all embedded commands:

1. Call up the program file you want to revise using the CA (Call) command. For example, to revise the file EXERCISE.PGM:

Type: [F5] ca exercise.pgm [↵]

2. Press: [Ctrl][F8] to switch to expanded view.

Result: The markers no longer appear – the commands are displayed expanded within double angle brackets.

3. Insert and delete any commands you want removed from the program including their surrounding double angle brackets.

4. Press: [F8] to switch to draft view.

Result: The commands that were expanded on the screen are now shown as markers.

CUSTOMIZATION GUIDE

User Programming

Function Calls

Function calls are two-letter instructions that represent basic XyWrite activities. Some function calls are assigned to keys and some are not.

In a program, XyWrite function calls look like two bold letters followed by a space; however, they are unique, single-unit representations that must be created either by using program mode ([Scroll Lock]) or by using the **PFUNC (Put Function Call)** command.

After you press [Scroll Lock] to turn program mode on, every key you press that is not an alphanumeric character causes XyWrite to put (paste) the function call assigned to that key into your program file. For example, if you press [F5], XyWrite enters the function call BC (Blank the Command Line) into your program.

You can use the PFUNC command to insert function calls into your program even if they are not assigned to keys. Be sure to turn off program mode before using the PFUNC command. For example, to enter the NM (No Markers) function call, which hides all embedded command markers and paragraph-end arrows.

Type: [F5] pfunc nm [↵]

Refer to the section "**Function Calls**" elsewhere in this document for a complete list of all XyWrite function calls. You can use all of those function calls in programs except TS (Toggle Scroll Lock). TS cannot be entered into a program file, since you use it to begin and end the recording of a program.

Two function calls especially useful in XPL programs are **BC (Blank the Command Line)** and **BX (Blind Execute)**.

The BC function call clears the command line and moves the cursor to the start of the command line.

Example: **BC** ca test**XC**

The BX function call executes a command from within a program without moving the cursor to or clearing the command line. If the command and any arguments are enclosed in parentheses (), square brackets [], or curly braces {}, XyWrite executes the command automatically.

Example: **BX** (ca test)

Otherwise, if they are not so enclosed, you must use the **Q2 (Execute Too)** function call with the BX function call to execute the command. Q2 is useful for executing a command string that may contain parentheses, square brackets, or curly braces.

Example: **BX** ca test**Q2**

User Programming

Macros

Macros are storage areas, or buffers, you can use to save strings or values. XyWrite provides several types of macros: ordinary, programming, and additional programming.

- **Ordinary Macros: A-Z, 0-9** – These are the macros that you usually access with the [F2] keys. They are “permanent” in that their contents remain intact when the program stops running.
- **Programming Macros: 00-999** – These one-thousand macros can be used only within an XPL program. The contents of programming macros 00-99 are saved only while the program is running and vanish at the end of each program. The contents of programming macros 100-999 remain intact until either they are overwritten or you quit XyWrite. Ordinary macros 0 through 9 are distinct and separate from programming macros 00 through 09.
- **Additional Programming Macros: &A-&Z, &0-&9** – These can be used only with the [LDPM \(Load Program\)](#) and the [RUN](#) commands. Refer to the sections that describe these commands for more information. Like ordinary macros, these are “permanent” and remain intact until either you overwrite them or quit XyWrite.

If you use programming macros exclusively for programs, you can reserve your ordinary macros for on-the-fly use; since programming macros are not accessible from the keyboard.

There are several embedded commands you can use with programming macros. There are three kinds of information you can store in a programming macro: a numeric expression (value), a literal string of characters, or a program. The kind of information you are saving determines the command you use to save it and the command you use to access it. Refer to “[XPL Commands](#)” for more information on macro commands.

You can use the [PR \(Prompt\)](#) command in a program to display the contents of a macro on the status line. To do this you specify the number of the macro, preceded by an @ character in the message argument of the PR command. Example: `«prThe file is @98»`

NOTE Reserved Macros *There are also 1000 programming macros, numbered 1000 through 1999 that are reserved for use by XyWrite menus.*

CUSTOMIZATION GUIDE

User Programming

XPL Commands

There are four categories of XPL commands: Commands that Return Values, Flow Control Commands, Macro Commands, and Miscellaneous XPL Commands. Most of the commands are embedded commands and appear as markers in draft view. Arguments (values) to embedded commands can include macros, logical expressions, numeric expressions, or additional embedded commands.

Commands That Return Values

The following commands insert values into your program from XyWrite, from the user, or from another program. You can use any of these commands as values in expressions.

AS

Argument Insert

Takes the string passed in from the RUN *filename,string* command and saves it to macro 00. You can then use the **PV (Put Variable)** command to access the contents of macro 00 on the screen, in the text, or in an expression. For example, if you start the program with RUN EXERCISE.PGM, 1234, then macro 00 contains the string 1234 (not the number) within the program. If you load the program to a key with LDPM and then run that program from a macro key, macro 00 will contain any string you enter on the command line. Since AS returns a string rather than a number, use IS00 when you want to use AS within SX; for example, `<<SX01,<<IS00>>>>`.

Example #1: The new variable is `<<AS>>`

Example #2: `<<IF <<AS>>==<<PV01>>>>`

CL

Column Location of Cursor

Takes on the value of the current column position of the cursor. The columns on the display are numbered starting from the left at column 0. (In graphic view, CL displays the pixel position of the cursor rather than the column location.)

Example: `<<SX01,<<CL>>>>`

CP

Character Position

Takes on the value of the current character position in number of characters from the beginning of the file.

Example: `<<SX01,<<CP>>>>`

Once you have saved the cursor position and moved the cursor, you can use the **JMP (Jump)** command to return to that previous position:

Example: `[F5]jmp <<pv01>> [↵]`

ER

Error

Takes on a logical value of either TRUE or FALSE. ER is TRUE if there was an error in the previously executed command; otherwise, it has the value FALSE. Frequently used with the SEARCH command – ER is TRUE if the search returns NOT FOUND.

Example: `<<IF<<ER>>>>`

User Programming

XPL Commands (Cont.)

EXIST *filename*

Check For Filename

Checks to see if a filename already exists in the directory you specify and takes on a logical value of either TRUE or FALSE. EXIST is TRUE if the file is not found; otherwise, it has the value FALSE.

Example: **BX** exist <<pv01 >>Q2

RC

Read Character

Stops the program and waits for the user to press a key. When a key is pressed, RC takes on the value of that key and the program continues with the next step. Use RC in an expression; typically you would assign RC to a macro.

Example: <<SX01, <<RC >> >>

VA *nm*

Value of Variable

Obtains the current value (or string) of the variable you request (*nm*). You can request any XyWrite default setting (for example, VA RM obtains the current value of the right margin) or any environment setting (for example, VA SPA obtains the current drive and path). Unlike the other commands in this section (except IS), you can use this command either in a program or directly in text. Refer to "[VA Settings](#)" for a list of all the variables you can use with the VA command.

Example: <<VALM >>

You can use the /NV switch with the VA command to display the value on the status line.

Example: [F5]va/nv lm [-]

Flow Control Commands

The following commands control the flow of the program.

EX

Exit and Continue

EX and EX1 are two different ways to exit a program. EX is a subroutine return. When used at the main level of a program, the EX command stops the program. When used in a subroutine, EX exits from that subroutine and continues execution at the point the subroutine returns to (a subroutine is any program within a program).

Example: <<EX >>

EX1

Exit and Stop

EM stops the program altogether. Unlike EX, it stops regardless of whether execution is in the main program or in a subroutine.

Example: <<EX1 >>

CUSTOMIZATION GUIDE

User Programming

XPL Commands (Cont.)

GL *label*

Go to Label

Causes execution to jump to the label specified by the LB command. The label can be any length string.

Example: <<GLstart>>

To make your programs easier to read, type two carriage returns after each GL statement. This will not affect operation of the program, since the GL command skips all statements and goes directly to the specified label. Label names are case sensitive; therefore, <<glend>> does not jump to <<lbEND>>.

IF *expression trueaction* EI

IF Condition

This command evaluates a boolean expression and determines whether the expression is TRUE or FALSE. If the expression is TRUE, the commands specified in *trueaction* are executed and then execution continues with the next statement after the EI command (End If) command. If the expression is FALSE, execution jumps to EI and continues from there. Examples:

Comparing Values: <<IF <<PV01 >> == <<PV02 >> >> <<GLA >> <<EI >> <<GLB >>

Comparing Strings: <<IF <<IS01 >> == <<IS99 >> >> <<GLA >> <<EI >> <<GLB >>

LB *label*

Label

Marks a spot in the program that you can jump to with the GL command. The label can be inserted at any point in a program, and can be any length string. LB can also be used simply as a comment in a program. (When writing a comment that contains spaces, though, the comment must be typed in expanded view.)

Example: <<LBstart>>

Macro Commands

In some of the following commands, you specify #, which is a macro: either an ordinary macro (A-Z, 0-9) or a programming macro (00-999). Macros &A-&Z and &0-&9 do not work with those commands.

CLRASG

Clear All Macros

Clears all macros – A-Z, 0-9, &A-&Z, &0-&9 and 100-1999. CLRASG is an immediate command.

Example: [F5]clrasg [↵]

CLRSgt

Clear Ordinary Macros

Clears all ordinary macros A-Z and 0-9, but leaves programs assigned to programming macros (100-1999) and additional programming macros (&A-&Z and &0-&9) in memory. CLRSgt is an immediate command.

Example: [F5]clrsgt [↵]

User Programming

XPL Commands (Cont.)

CLRSGT

Clear Additional Macros

Clears only the additional programming macros (10-999). CLRSGT is an immediate command.

Example: [F5]clrsgt [↵]

GT

Get Macro

Inserts the text or invokes the program or subroutine assigned to the specified macro (#).

GT # is similar to the PV (Put Variable) command, except GT inserts text all at once, while PV inserts the text one character at a time. If overstrike mode is on, the text output by the GT command does not overwrite existing text.

Example: <<GT98>>

IS

Insert macro

In User Programming, you can use IS within an IF statement to compare the string contents of one macro either to another macro or to a literal string enclosed in double quotes. (When used outside of XPL, IS operates as just another regular formatting command.)

Example #1: <<IF <<IS01 >> == <<IS99 >> >>

Example #2: <<IF <<IS02 >> == "XyWrite"

PV

Put Variable

Inserts the characters one at a time from the specified macro to the current location. May be used in two ways:

- When used outside of an expression, PV inserts the text or runs the program assigned to the macro - either on the command line or in text. If overstrike mode is on, the text output by the PV command overwrites existing text.

Example: <<SX01, <<VA\$fi >> >> <<PV01 >>

- When used inside an expression, XyWrite interprets the content of the macro as a number and combines it with the rest of the expression to give a numeric result.

Example: <<IF <<PV800 >> < <<PV801 >> >>

REMOVE

Remove From Memory

Clears any single macro # (or user program) from memory. REMOVE is an immediate command.

Example: [F5]remove x [↵]

SAVE %#,filename

Save To Another File

Saves the contents of a macro # to a new file on disk *filename*, without opening the file. SAVE is an immediate command.

Example: [F5]save %b,myfile [↵]

CUSTOMIZATION GUIDE

User Programming

XPL Commands (Cont.)

SU #,string

Save Subroutine

Saves string as a program to the specified macro # (this is similar to SV, except the string is marked as a program). You can run this subroutine within another program using either the GT or the PV command. The following example saves to macro 98 a subroutine that executes the SAVE command.

Example: <<SU98,BC saveXC>> >>

SV #,stringexp

Save String

Saves a literal string to the specified macro #. You can specify the string expression (*stringexp*); or if you omit the comma and *stringexp*, XyWrite saves any block of text that is selected when you run the program. If you omit *stringexp*, XyWrite clears the macro or initializes it with a size of 0. SV can be used only for saving literals. You can compare this string to contents of any other macro or to a literal string.

Example #1 puts the string YES into macro 99; Example #2 initializes or clears macro 32; Example #3 saves the currently selected block to macro 33.

Example #1: <<SV99,YES>>

Example #2: <<SV32,>>

Example #3: <<SV33>>

SX #,numericexp

Save Expression

Evaluates a character or series of characters and saves the result to the specified macro #. The character(s) can be a number (such as 88), a numeric expression (such as the result of 2 + 2), or a XyWrite variable (such as the current left margin setting).

Example #1: <<SX01,25>>

Example #2: <<SX02,<<IS99>>>>

User Programming

XPL Commands (Cont.)

XS #1,#2,#3,#4,#5

Extract String

Divides the contents of a macro into parts (often called parsing), then extracts and saves the parts to other macros. The XS command works by searching the specified macro for the character(s) you designate. If it finds a match, XS creates three new macros: one for storing the information that precedes the matched string; one for the matched string itself; and one for the information that follows the matched string.

#1 is the macro that contains the string you want to search.

#2 is the macro that contains the string you want to match (you can use wild cards in this string).

#3 is the macro where XS will store the information that precedes the matched string.

#4 is the macro where XS will store the matched string (unless you use wild cards in #2, the contents of this macro will be identical to the contents of #2).

#5 is the macro where XS will store the information that follows the matched string.

For example, you may want to extract a filename extension to determine what printer file to use for a file or to what subdirectory to send the file, you could use the XS command in program like this:

```
<<XSa,b,x,y,z>>
```

If macro A contains the string CHAPTER.DOC, macros B and Y contain a period, macro X contains the string CHAPTER, and macro Z contains the string DOC.

NOTE Wild Cards *XyWrite accepts wild card characters in the string you want to match. Use the same wild cards as you use with the Search command. To enter them into the program file, press [Shift][Esc] followed by the wild card letter you want.*

CUSTOMIZATION GUIDE

User Programming

Miscellaneous XPL Commands

You can use the following commands for various tasks in your XPL programs:

- **BEEP** – To produce an audible signal
- **GOFILE** – To switch windows to an open file
- **LDPM** – To load a program
- **PFUNC** – To enter a program function call
- **RUN** – To run a program
- **STSGT** – To store macro keys to disk

BEEP

Producing an Audible Signal

Format:

[C:\XY4]BEEP

This is an immediate command. BEEP produces an audible signal at a point in your program.

Menu: Not a menu item.

Using BEEP

To produce an audible signal:

1. Switch to expanded view:

Press: [Ctrl][F8]

2. Use **PFUNC** to enter either the BC (Blank the Command Line) or the BX (Blind Execute) function call:

Type: [F5] pfunc bx [↵]

3. Type the command:

Type: beep

4. Use **PFUNC** to enter either the XC (Execute) or the Q2 (Execute Too) function call, depending on the command you used in Step 2:

Type: [F5] MpfuncQ2 [↵]

Result: When you run the program, the computer emits an audible signal at the point where the command appears.

GOFILE

Switching Windows to an Open File

Format:

[C:\XY4]GOFILE *filename*

filename is the name of the file to which you want to move. This is an immediate command.

Menu: Not a menu item.

Using GOFILE

To switch to the window that contains a file named MYFILE:

1. Put the cursor at the point in your program where you want to switch windows.
2. Use either the BC (Blank the Command Line) or the BX (Blind Execute) function call:

Type: [F5] pfunc bc [↵]

CUSTOMIZATION GUIDE

User Programming

Miscellaneous XPL Commands (Cont.)

3. Type the command:

Type: `gofile myfile [-]`

4. Use either the XC (Execute) or the Q2 (Execute Too) function call, depending on the command you used in Step 2:

Type: `[F5] pfuncXC [-]`

Result: When you run the program, execution switches to the window that contains the file named MYFILE at the point where the command appears.

LDPM

Loading a Program

Format:

Option 1

`[C:\XY4]LDPM d:programfile,#`

Option 2

`[C:\XY4]LDPM d:programfile`

d: is the letter of the drive that contains *programfile*.

programfile is the existing program file you want to load.

(*optional*) is the single letter (A-Z) or number (0-9) or two characters &A-@Z or &0-&9 where you want to save the program file.

This is an immediate command.

Menu:

ADVANCED → PROGRAMMING → SAVE TO MACRO KEY

Using LDPM

This command loads a program file either onto the specified macro key (*Option 1*) or into memory (*Option 2*). Option 1 enables you to run the program file with an [F2] key (rather than with the RUN command). Option 2 enables you to RUN a program directly from memory rather than from disk.

You can assign programs to two different kinds of macros:

- **Macros A-Z and 0-9** – You normally run this kind of program from the keyboard using [F2]# (which accesses the function calls @A-@Z or @O-@9).
- **Macros &A-&Z and &0-&9** – You run this kind of program from any key where you have assigned the corresponding function call &A-&Z or &0-&9, or with the FUNC command (such as FUNC &A).

Loading a Program into a Macro (*Option 1*)

To load a program into a macro:

1. **Load the program** To load the program file EXERCISE.PGM into macro X:

Type: `[F5] ldpm exercise.pgm,x [-]`

Result: The program file is copied to the macro (in memory). You can now run the program file EXERCISE.PGM by pressing [F2]x.

2. **Verify** (Optional) To verify that the file has indeed been loaded onto that key:

Press: `[Ctrl][F2]`

CUSTOMIZATION GUIDE

User Programming

Miscellaneous XPL Commands (Cont.)

After viewing the text, press [Esc] to return to the document.

3. Store the Macro (Optional) If you want to keep this program file loaded on the macro key for use at future editing sessions (after you quit), use the **STSGT (Store Macros)** command.

NOTE Additional Macros You can load programs onto any of up to 72 keys. This includes the 36 ordinary macros [F2]A through [F2]Z (Function Calls @A-@Z) and [F2]O through [F2]9 (Function Calls @0-@9). It also includes additional programming macros assigned to function calls &A through &Z and &0 through &9.

Loading a Program into Memory (Option 2)

To load a program directly into memory (rather than onto a macro), use LDPM with only the filename of the program. For example:

Type: [F5] ldpm exercise.pgm [↵]

Result: The program file EXERCISE.PGM is now stored in memory. When you use the RUN command, XyWrite checks to see if the program you specify is stored in memory before going to the disk.

PFUNC

Program Function Calls

Format: [C:\XY4:]PFUNC ##

is a two-character function call. This is an immediate command.

Menu: Not a menu item.

The PFUNC (Put Function Call) command lets you enter a XyWrite function call into your file when not in program mode or when the function call is not assigned to a key.

Using the PFUNC Command

Let's suppose you are creating a program in which you want to clear the command line, but leave the cursor in the text area. The function call for this action is CH (Clear Header), which is not typically assigned to a key. To enter it in your program:

Type: [F5] pfunc ch [↵]

Result: A bold CH appears in your program file. When you run the program, XyWrite will clear the command line without moving the cursor there.

RUN

Running a Program

Format: [C:\XY4]RUN *d:programfile,string*

d: is the letter of the drive that contains *programfile*.

programfile (optional) is the name of the program file you want to run.

string (optional) is any information you want to pass to the program.

This is an immediate command.

Menu:

ADVANCED→ **PROGRAMMING**→ **RUN MACRO**

RUN causes the specified program file to execute. This means the commands (and text) stored in the program file are executed automatically, as if typed from the keyboard. When you include information

User Programming

Miscellaneous XPL Commands (Cont.)

as an argument to the RUN command, XyWrite automatically stores the information in macro 00. You can access it by including «is00» or «pv00» in the program, depending on the type of information and how you want to use it.

Running A Program

To run a program file – for example, EXERCISE.PGM:

Type: [F5] run exercise.pgm [↵]

Result: This command runs the program file named EXERCISE.PGM – the keystrokes stored in that file are automatically executed.

To stop the program (if necessary):

Press: [Ctrl][Break]

NOTE Related Commands You can also run a program file with an [F2] key. Refer to [LDPM \(Load Program\)](#) to load the program file onto a macro key.

NOTE #1 – Shortcuts XyWrite remembers the name of the last file run. Therefore, if you enter the RUN command without specifying the name of a program file, XyWrite reruns the most recently run file. If want to run a new program but aren't sure of its name, you can build a directory to locate the file and then execute the RUN command while pointing at the program name.

NOTE #2 – Running a Program from STARTUP.INT If you place a program into STARTUP.INT to be run on startup (with RUN filename), end that program with «EX» rather than ending it with «EX1». Since STARTUP.INT is itself a program, your program is a subroutine within it. EX1 in the subroutine stops execution after the subroutine without returning to STARTUP.INT.

STSGT

Store Macro Keys to Disk

Format: [C:\XY4]STSGT *filename*

filename identifies the file on disk to which the macro keys will be stored. This is an immediate command.

Menu:

INSERT → TEXT FROM A MACRO → OPTIONS...

STSGT stores the set of current macro keys to the specified file on disk. This enables you to reload the keys for use at a later editing session. You can also save several sets of macro keys and load each one for a different purpose.

Storing Macro Keys to Disk

1. View the Macro Keys (Optional) To view the text that will be saved to disk:

Press: [Ctrl][F2]

After viewing the text, press [Esc] to return to the document.

CUSTOMIZATION GUIDE

User Programming

Miscellaneous XPL Commands (Cont.)

2. Disk Save (Long-Term Save) To save to disk all of the keys viewed in Step 1:

Type: stsgt proposal.sgt [-]

In this case, PROPOSAL.SGT is the filename to which the keys are stored. You can save to any filename you want, but we recommend that you use the SGT extension, and store the file in the same directory as SIG.EXE. If you follow those conventions, XyWrite will include the new macro file when it creates a list of macros.

User Programming

XPL Operators and Functions

XyWrite provides operators and functions in the following categories:

- Arithmetic operators
- Logical operators
- Relational operators
- String operators
- String functions

Arithmetic Operators

Use these operators to perform arithmetic on numeric values.

- + Addition
- Subtraction
- * Multiplication
- / Division

Example: `<<SX99,<<PV01>>*10+<<PV02>>>>`

Logical Operators

Logical operators perform logical (or boolean) operations on numeric or string expressions. You use logical operators within IF statements.

- ! Or
- & And
- @XOR Exclusive Or
- @NOT Not

Example: `<<IF(<<IS99>>==<<IS01>>)!(<<IS99>>==<<IS02>>)>>`

Relational Operators

These operators let you compare two numeric expressions (with PV) or two string expressions (with IS).

- < Less Than
- > Greater Than
- <= Less Than or Equal (same as =<)
- >= Greater Than or Equal (same as =>)
- <> Not Equal
- == Equal

Example: `<<IF(<<PV01>><<PV02>>)>>`

String Functions

String functions operate on string expressions and return a value or another string.

- @UPR Converts letters to uppercase (parentheses are required)
- @CNV Converts a function call to its two-letter mnemonic
- @SIZ Returns a value equal to the number of characters in a string (parentheses are required)
- @NUM Strips all punctuation and letters, leaving only digits
- @INT Eliminates fractions after decimal point

Example: `<<sx03,@UPR(<<is03>>)>>`

CUSTOMIZATION GUIDE

User Programming

XPL Operators and Functions (Cont.)

String Operators

A string operator operates on two string expressions.

+ **Concatenation**
Links two or more separate strings

Example: <<SX99,<<IS01 >> + <<IS02 >> >>

[\[ascii#238\]](#) **Contains**

Determines if a string exists within another string. If the first string is found to be part of the second string, XyWrite returns a numeric value equal to the number of characters that precede the occurrence of the first string within the second: returns a value of 0 if the characters of the first string are at the beginning of the second string; returns a value of *n* if the first string appears within the second string but not at the beginning (*n* is the number of characters in the second string that precede the first string; returns a value of -1 if the first string is not in the second string.

Example: <<IF(<<IS01 >>[\[ascii#238\]](#)<<IS02 >>==O) >>

. . . returns 0 if macro 01 = A and macro 02 = ABCDE.

User Programming Examples

This section contains examples of the following procedures:

- Creating and running a user program
- Testing for a carriage return
- Evaluating keyboard input and branching

Creating and Running a User Program

As an example, the following procedure writes a program to count the number of characters in a file and display the result.

1. Plan the Program In order to count the number of characters in a file, you press [Ctrl][End] to move the cursor to the end of the file, use the **CP (Character Position)** command to count the number of characters from the beginning of the file, and save the result in a macro key. Since you are asking XyWrite to make an evaluation, you use the **SX (Save Expression)** command to save the result to a macro and the **PV (Put Variable)** command to display the contents on the screen.

2. Create the Program File Use the NE (New) command to create a new program file. For example:

Type: [F5] ne count.pgm [↵]

Result: XyWrite creates a new (empty) file with the name COUNT.PGM.

3. Write the Program After turning on program mode, type all the keystrokes necessary to perform the routine. When you are finished, turn program mode off.

Press: [Scroll Lock] to turn program mode on.

Press: [Ctrl][End]

Result: A bold **BF** appears on the screen (for Bottom of File).

Press: [Scroll Lock] to turn program mode off.

Type: [F5] sx[F9]

Result: A text entry screen opens.

01,[F5]cp[F9]

Press: [Shift][F1] to close the window.

(If you want to abort the command window, press [Esc])

Result: The SX command displays in expanded view as **«SX01,«CP»»**.

Press: [Scroll Lock] to turn program mode on again.

Press: [F5]

Result: **BC** appears in the program file, which moves the cursor to the command line.

Press: [Scroll Lock] to turn program mode off again.

Type: [F5] pv 01[F9]

Result: This will display the contents of macro 01 (the value acquired by the CP command) on the command line when the program runs.

In draft view, the program file looks something like this:

BF ▲ BC ▲

CUSTOMIZATION GUIDE

User Programming

Examples (Cont.)

In expanded view, the program file contains code that looks like this:

```
BF <<sx01,<<cp>>>>BC <<pv01>>
```

The program code functions as follows:

BF (Bottom of File) moves the cursor to the end of the file.

<<sx01,<<cp>>>> saves the value of the current cursor location to macro 01.

The CP (Character Position) command returns the number of characters from the beginning of the file.

BC (Blank Command Line) clears the command line.

<<pv01>> outputs the contents of macro 01 to the screen.

4. Store the Program

Type: [F5] store [-]

Result: The program file COUNT.PGM is stored on the disk and disappears from the screen.

5. Test the Program Test the program by calling a file in the current directory and counting the number of characters in it. To execute the program, call up any file and:

Type: [F5] run count.pgm [-]

Result: The program moves the cursor to the end of the file, counts the number of characters in the file, and displays the number on the command line.

6. Load the Program onto a macro Key If you want to load COUNT.PGM onto a text key, refer to "[LDPM \(Loading a Program\)](#)" for information.

Testing for a Carriage Return

Testing for a carriage return requires two steps: (1) save a carriage return to a macro and (2) make an IF statement. The following example saves the carriage return to macro 91; and uses an IF statement to compare it to macro 00. Type this in with [Scroll Lock] turned off.

Type: [F5] sv [-]

Type: 91, [-]

Press: [Shift][F1]

Result: The carriage return is saved to macro 91. In expanded view, the macro you just typed in looks rather odd, but nonetheless is correct:

```
<<SV91,←
```

```
>>
```

Then type in the following IF statement:

```
<<IF <<IS00>>[ascii#238] <<IS91>>==0>>
```

If the content of macro 00 is a carriage return, this statement will be true.

User Programming

Examples (Cont.)

Evaluating Keyboard Input and Branching

You can write programs that will pause in the middle of execution, ask a question and wait for you to respond. Therefore, you can stop and make choices. For example, you can modify your STARTUP.INT file to choose which printer file to load. The following program demonstrates the **RC (Read Character)** command and the IF statement. RC causes the program to pause, allowing you to select a printer by pressing a letter (D or L).

```
<<LBA>> <<SV01,D>> <<SV02,L>> BC Dot matrix (D) or Laser (L)?
```

```
<<SX99,<<RC>> >> <<SX99,@UPR(<<IS99>>) >>
```

```
<<IF <<IS99>> == <<IS01 >> >> <<GLDOTMATRIX>>
```

```
<<EI >> <<IF <<IS99 >> == <<IS02 >> >> <<GLLASER >>
```

```
<<EI >> BC p Press 'D' or 'L' XC <<GLA >>
```

```
<<LBDOTMATRIX >> BC ldpm eplq500.prn XC <<EX >>
```

```
<<LBLASER >> BC ldpm hplj-2.prn XC <<EX >>
```