TSO/E REXX Execution and Buffer Control Commands

Objectives

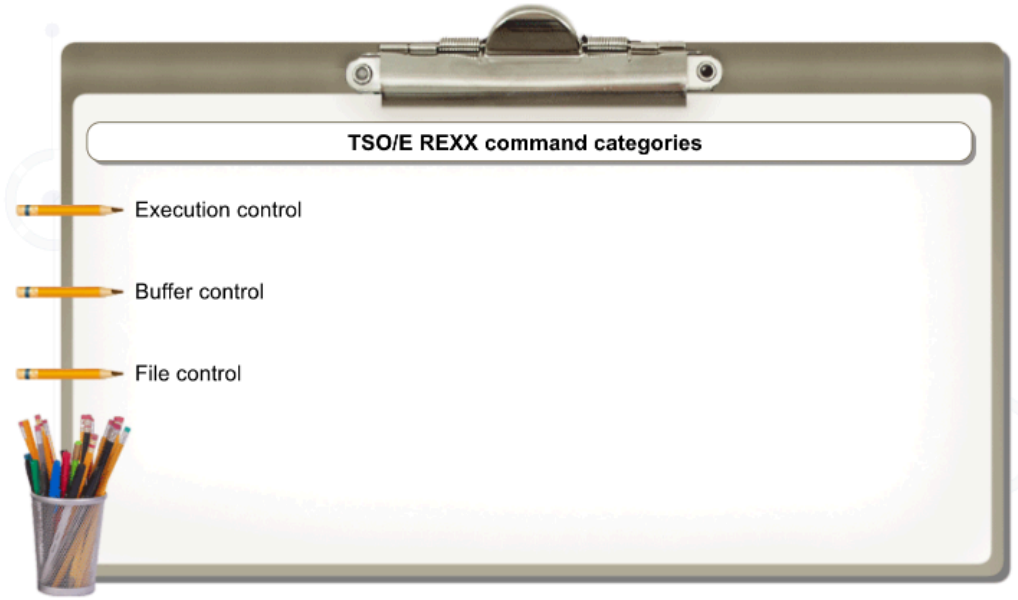
TSO/E REXX Execution and Buffer Control Commands

In this module, you will look at the TSO/E commands that are available when running REXX in the TSO/E environment.

You will review the execution control commands to interrupt a REXX program and the buffer control commands to manipulate and control the external data queue.

After completing this module, you will be able to:

- Identify How to Use TSO/E REXX Execution Control Commands
- Identify How to Control the External Data Queue with Buffer Control Commands



When REXX was developed on the IBM mainframe virtual machine (VM) operating system, many VM-specific commands were available to control REXX processing.

After REXX was ported to the MVS operating system environment in 1988, TSO/E was modified to enable these commands to be executed, but only from an executing REXX program.

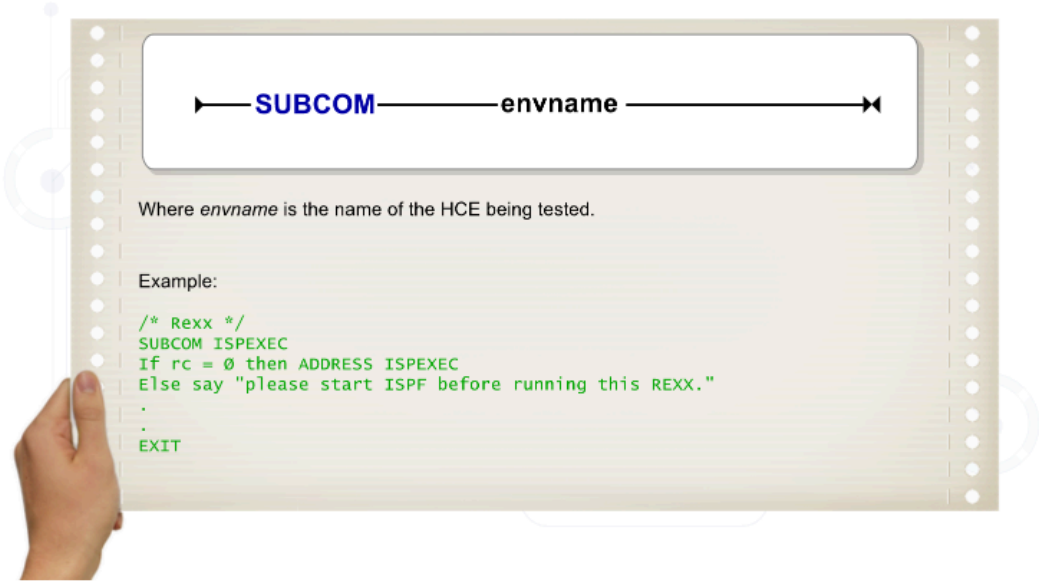
These commands are now referred to as TSO/E REXX commands. For the purposes of this course, they are grouped into the three categories listed above. You will now focus on execution control and buffer control commands.

TSO/E command	Description
SUBCOM	SUBCOM command can be used to determine if a host command environment (HCE) is available.
Immediate commands	
HE	Halt Execution command stops execution of all execs currently running. This command is useful if a procedure or function is in an endless loop or is stuck in runaway processing. HI is normally recommended as HE does not set the HALT condition.
HI	Halt Interruption command stops execution of the exec currently running. This command is useful if an exec is in an endless loop or is stuck in runaway processing.
HT	Halt Typing command stops output from an exec being typed to the user terminal. Processing of the exec continues but all output is suppressed, except output generated by TSO/E commands. This command is useful if an exec must display a large number of lines during execution of a loop.
RT	Resume Typing command resumes typing that has been previously suppressed with the HT command.
TE	Trace End command resumes normal execution of an exec that is being debugged by the use of the Trace Start (TS) command.
TS	Trace Start command starts tracing execs to enable a programmer to control the execution of an exec and identify faults in the exec. All commands, including the results of expressions, are displayed on the screen.

Execution control commands are used to determine how a REXX program should process. SUBCOM interrogates the environment that REXX is running under. The other commands, which are commonly used for debugging, are called immediate commands. These are usually executed when the PA1 or the ATTN key has been pressed while a REXX program is running. This causes the following message to be displayed:

```
IRX0920I ENTER HI TO END, A NULL LINE TO CONTINUE, OR AN IMMEDIATE COMMAND+
```

At this point, immediate commands can be used. They can also be used as a parameter of the EXECUTIL command, for example, EXECUTIL TS in a REXX program, or while in interactive debug that has been started by using the TRACE ? instruction.



As more products incorporate and build interfaces to the REXX interpreter, more HCEs become available for use from a standard REXX.

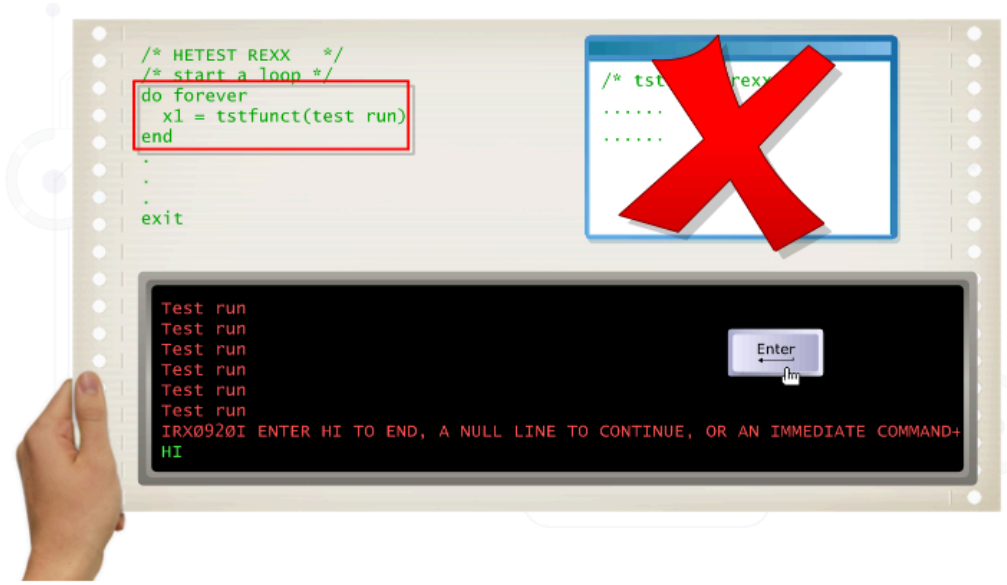
For example, Computer Associates CA-7 now has a HCE for REXX programs to execute CA-7 commands. Many of these HCEs require the product to be active or a special command to be executed before they can be used.

The SUBCOM command is used to determine if a particular HCE is available for use. When executed, the return code determines whether the HCE can be used; 0 = available, 1 = unavailable.

The HI immediate command is used to stop interpretation of the current processing exec and set the HALT condition so any SIGNAL ON HALT traps can be processed.

Should a program be looping or be required to be terminated for any reason, the PA1 or ATTN key can be pressed and HI can be entered to stop the program.

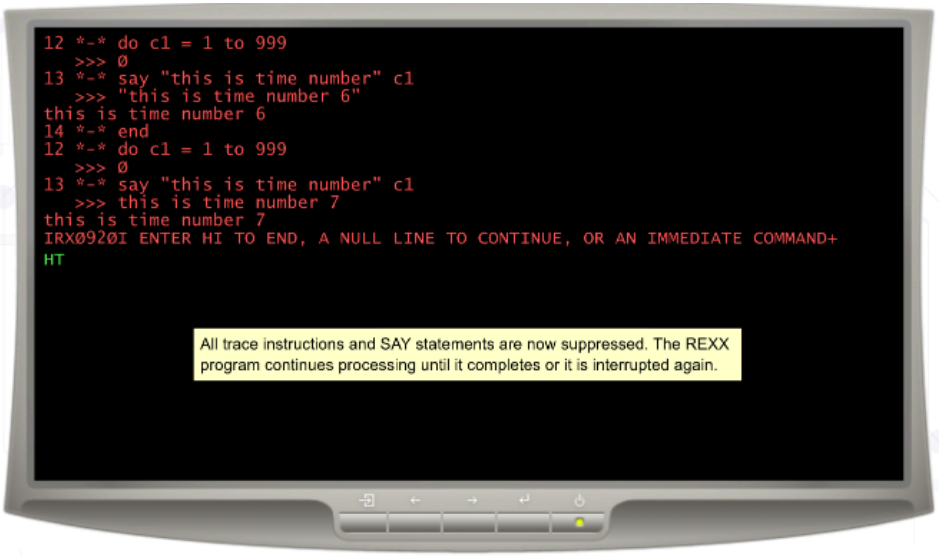
Click **Play** to see how the HI command works.



Occasionally, a REXX may loop and continually call another REXX program or function. This is often difficult to terminate as the PA1 key is more likely to be pressed while the function is being called.

If the HI immediate command is used, only the function will terminate. This allows the main REXX to continue looping and call the function again. The solution is to use the HE immediate command, which will terminate all executing REXX programs.

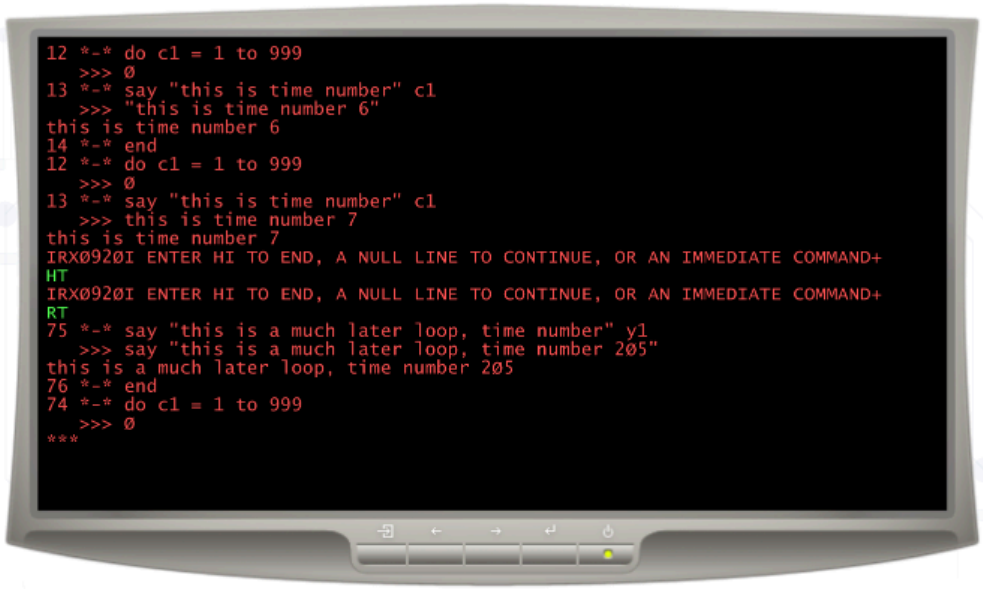
Unlike the HI command, HE will not cause a HALT condition so no SIGNAL ON HALT traps will be processed.



When tracing a REXX program through the TRACE instruction, large sections of code or long loops are sometimes accidentally included in the trace.

The REXX program may be terminated or the HT immediate instruction can be used. This stops all statements written from the program from being displayed on the screen, including output from SAY instructions. Output from TSO commands will still be displayed.

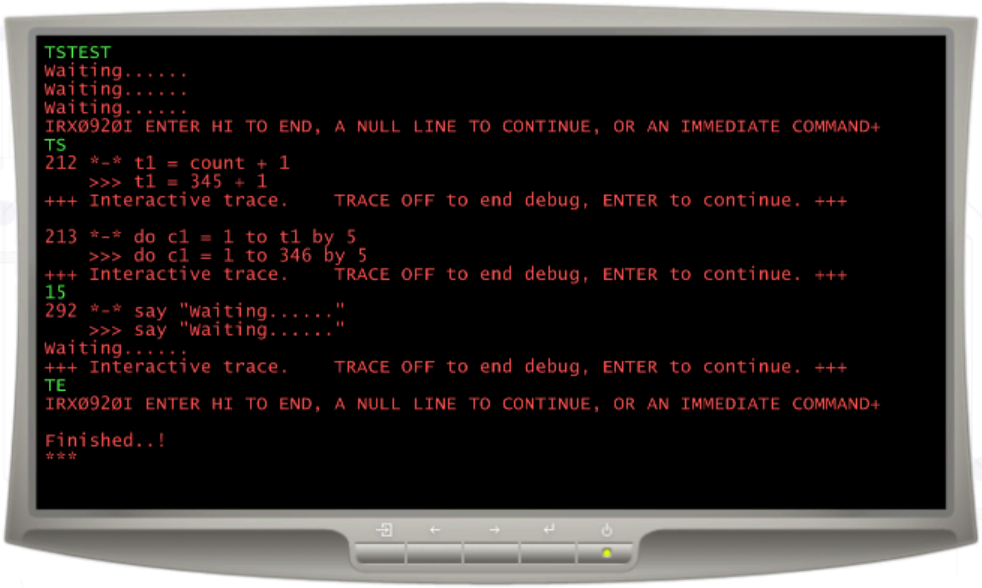
Click **Play** to see how the HT command works.



The RT instruction, which is the opposite of the HT instruction, enables traced lines and SAY instructions to be displayed on the screen.

This instruction is typically used when HT has been issued for a long-running program, and typing is switched back on to determine how far program processing has progressed.

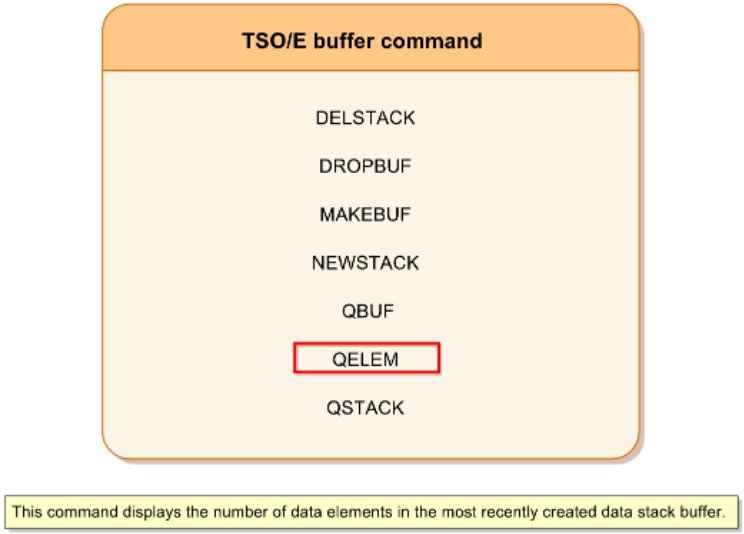
Click **Play** to see how the RT command works.



The TS and TE immediate commands enable interactive debug to be started and stopped as if the TRACE ?R keyword instruction had been used.

After pressing PA1 and entering TS, the system enters interactive debug mode, tracing one line and then pausing with a prompt. The user then presses Enter to trace the next line, enters a numeric value to skip trace for a number of lines, and types = to repeat the last instruction, or TE to end the interactive debug facility.

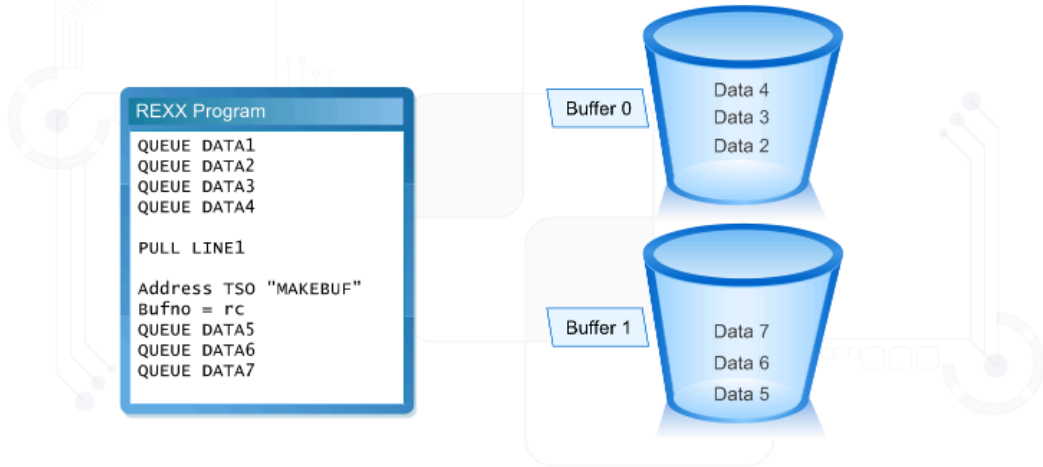
Click Play to see how the TS and TE commands works.



The external data queue, also known as "the stack", is a temporary data storage area where data is kept as records. These are read by using the PULL keyword instruction, or by other command environments that are looking for a prompt while a REXX program is running. Although it is not often used in the z/OS environment, the stack is useful for interrogating products, such as ACF2, or for executing commands that require prompts, such as FTP. Some REXX programmers also use the stack in preference to compound variables.

The commands listed above have been developed for use in the z/OS environment to help manage and maintain the stack. Because these are essentially TSO commands, the only information returned is a return code.

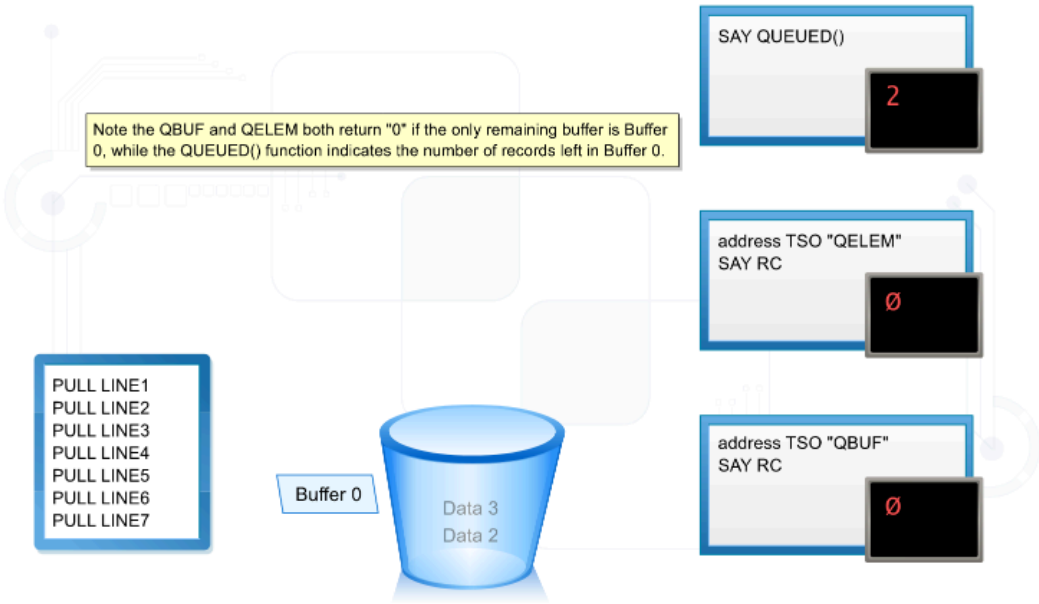
Mouse-over each command for a description.



The stack consists of a single buffer called Buffer 0. Any lines that are queued to the stack are placed into and retrieved from this buffer on a first-in, first-out (FIFO) basis. However, it may be necessary to have a second set of data placed in the stack that needs to be retrieved before the original set.

Extra buffers can be built by using the MAKEBUF command. The return code received from MAKEBUF indicates which buffer has been built: 1 for Buffer 1, 2 for Buffer 2, and so forth. The MAKEBUF command sets a new entry point into the stack for queued data.

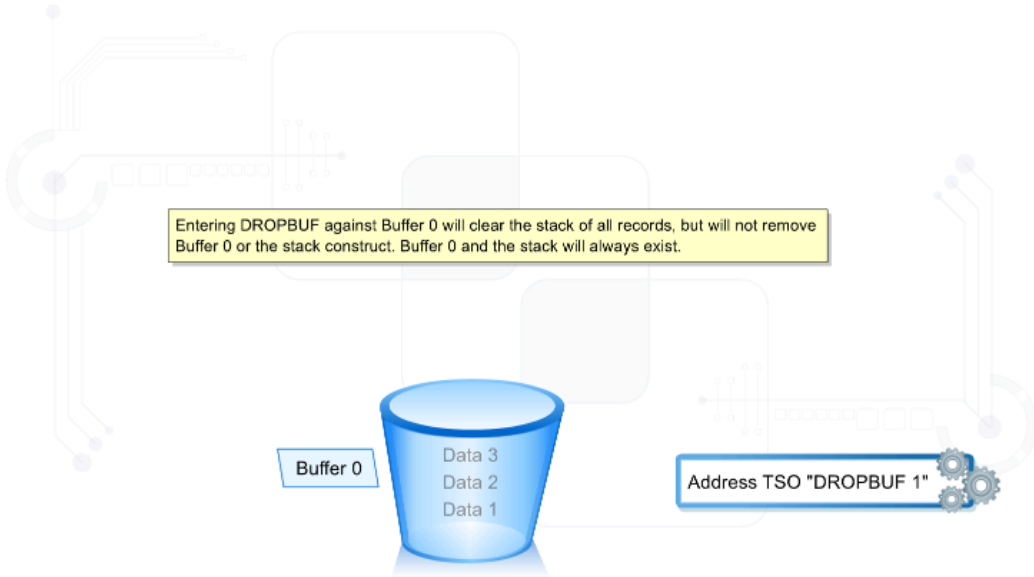
Click **Play** to see an example of the effects of the MAKEBUF command. In this example, the QUEUE instruction would place data into the top of the buffer and the PULL instruction would take from the bottom.



With multiple stacks, the PULL command extracts data from the latest buffer. If the buffer is emptied, it remains until another line is extracted. The current buffer is then destroyed and data is extracted from the previous buffer. The QUEUED() function indicates the number of elements in the entire stack. The return code from the QELEM command indicates the number of elements in the most recently created buffer. If no buffers have been created, QELEM returns 0.

You can use the QBUF command to determine how many buffers exist with the stack. The return code from the QBUF command indicates the number of the most recently created buffer. If the return code is 1, two buffers exist, the standard Buffer 0 and Buffer 1.

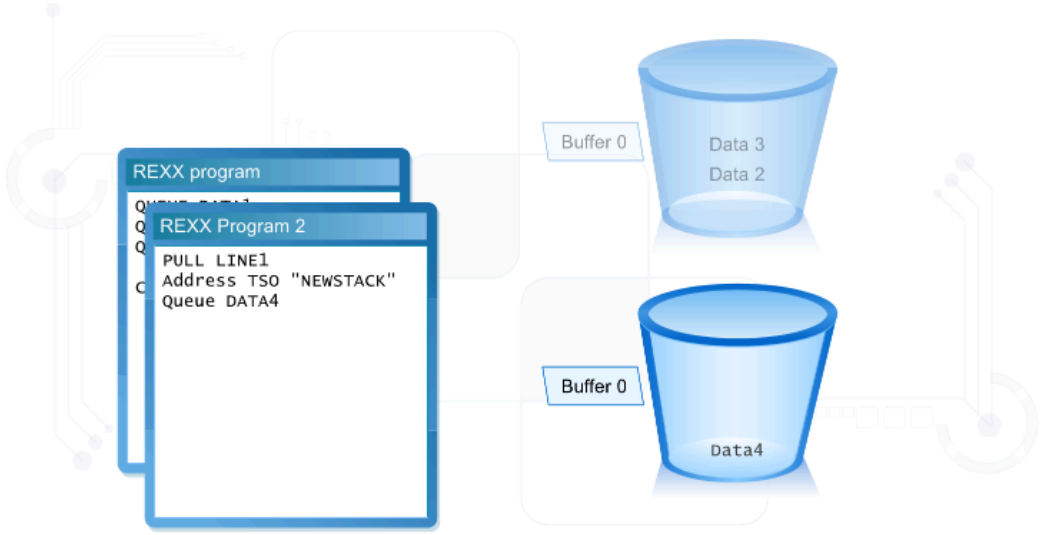
Click **Play** to see how multiple buffers can be used.



The DROPBUF command enables buffers and their records to be removed without having to read them. Without parameters, the DROPBUF command removes the current buffer. The DROPBUF n command removes Buffer n and any buffers that were created after it. The error codes for the DROPBUF command are:

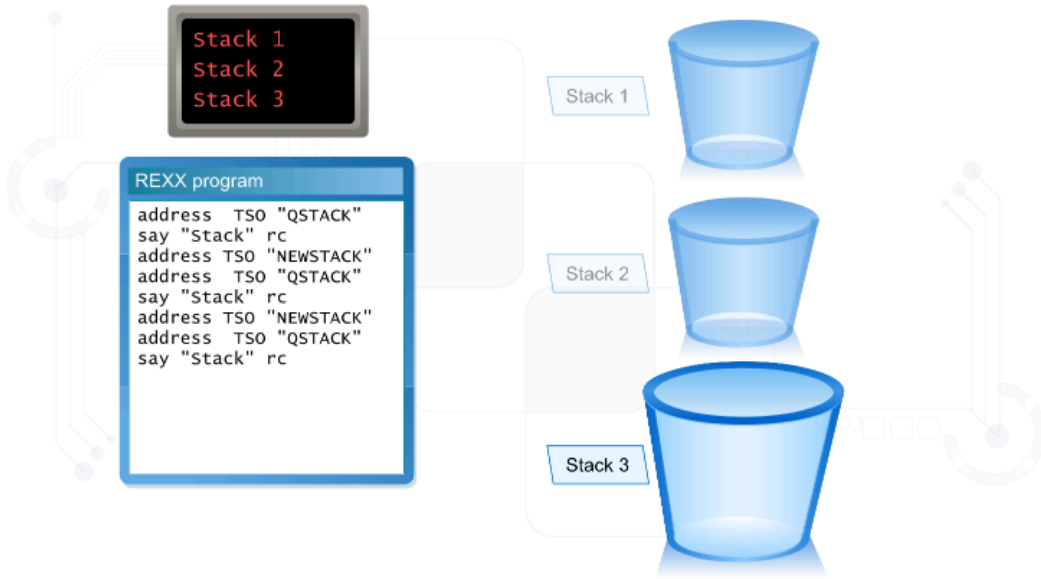
0 - Buffer was successfully removed.
1 - The DROPBUF command was invalid, for example, DROPBUF A.
2 - The specified buffer did not exist.

Click **Play** to see how the DROPBUF command works.



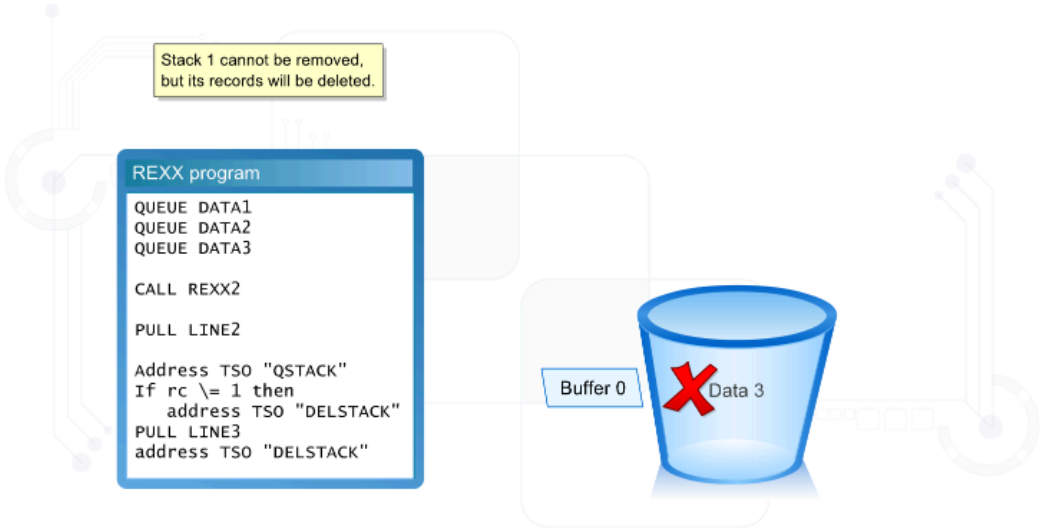
The stack is automatically available when a REXX begins processing and is referred to as Stack 1. As the stack is allocated at the address space level, it shares the same stack if another REXX is called. If records are already in Stack 1 and the second level REXX must use it, confusion could occur around which records belong to which REXX program. The NEWSTACK command enables an entirely new stack to be built. Unlike buffers, when a new stack is built, the previous stack is still in the background but is unavailable for use and cannot be seen until the new stack is manually removed.

You can build as many stacks as are required. If a PULL instruction is executed on the empty Stack 2, data is read from the keyboard. The QUEUED() function returns only the number of records in the latest stack. **Click** **Play** to see how the NEWSTACK command works.



Stacks can be built in any REXX procedure in an application so it is often necessary to determine whether any new stacks have been built. The return code from the QSTACK command can be used to determine the number of the current stack.

Click **Play** to see how the QSTACK command works.



In most cases, it is advisable to remove any stacks that have been created in a REXX program before the program completes execution. The DELSTACK command will remove the newest created stack and all the records in it.

As STACK 1 must always be available, only its records will be removed if DELSTACK is executed against it. Stack 1 and Buffer 0 are effectively indestructible in the z/OS TSO/E environment.

Click Play to see how DELSTACK can be used.