

CLIST PROGRAMMING

*Get on the
Fast Track!*



TM

SYS-ED®

PO Box 1213

New York, NY 10156

212-564-9147,48,49

FAX: 212-967-3498

www.mainframetrainingbysysed.us

1 Introduction

The CLIST language is an interpretive programming language; it does not need to be compiled and link edited. A CLIST can be used for performing a wide range of application tasks.

1.1 Function of a CLIST

A CLIST can be used to:

- C execute a series of TSO commands and subcommands to save time and effort.
- C create procedures using symbolic variables.
- C invoke one CLIST which in turn can invoke another CLIST; CLISTs can be nested to the desired level.
- C display panels which in turn can execute other CLISTs based on the selections made on the panel.
- C CLISTs can invoke programs in either the foreground or the background.
- C execute programs written in other programming languages.

1.2 The CLIST Language

The CLIST language has arithmetic and logical operators. It also has string handling and built-in functions.

CLIST statements are used for structuring programs to:

- C perform I/O.
- C define and use symbolic variables.
- C handle error and attention interrupts.
- C communicate with users at a terminal using Dialogue Manager defined panels.
- C CLIST statements can be used in conjunction with TSO commands and JCL statements to write applications.

2 Application Tasks

CLISTS that Perform Routine Tasks

CLISTS can be written which reduce the amount of time that a user has to spend on routine tasks such as checking on status of data sets, allocating data sets for particular programs, and printing files.

By grouping together in a CLIST, the instructions required to complete a task, it is possible to reduce the time, number of keystrokes, and errors involved in performing the task.

These CLISTS may consist entirely of TSO commands or of a combination of TSO commands and CLISTS statements.

CLISTS that are Self-contained Applications

A CLIST can invoke another CLIST; this is known as a nested CLIST.

The GLOBAL statement provides for the definition of common data among CLISTS and the PROC statement provides for the passing of parameters to a CLIST.

CLISTS can issue ISPF commands, such as ISPEXEC to display full-screen panels. Conversely, ISPF panels can invoke CLISTS based upon input provided by the user. When the user changes a value on a panel, the change to the variable on the associated panel definition also applies to the value of the variable in the CLIST that displayed the panel.

CLISTS that Manage Applications Written in Other Languages

CLISTS can be written to act as intermediaries between a user and applications written in other languages.

A CLIST can send messages to, and receive messages from, the terminal to determine what the user wants to do. Then, based on this information, the CLIST can set up the environment and create the commands required to invoke the application that performs the requested tasks.

3 Types of CLISTs

There are two types of CLISTs:

- C simple CLIST.
- C programmable CLIST.

Simple CLIST

A simple CLIST contains TSO commands and subcommands.

Programmable CLIST

A programmable CLIST contains CLIST statements in addition to the TSO commands and subcommands.

A programmable CLIST can bypass a command or group of commands based on certain information supplied by the user when the CLIST was executed or based on certain conditions.

It allows input/output to files and terminals.

3.1 Contents of CLISTs

A CLIST contains the following:

- C List of TSO commands and subcommands.
- C CLIST statements which are much like higher level languages such as COBOL, PL/I, etc.
- C Commands to invoke ISPF dialogue management services to display full screen panels.
- C JCL statements.

A CLIST should be given a name so that one CLIST may execute a series of TSO commands.

4 Data Sets and CLIST Libraries

CLISTS are programs that reside in either sequential or partitioned data sets (PDSs).

A sequential CLIST data set consists of only one CLIST, while a PDS may contain one or more CLISTS. In a PDS, each CLIST is a member and has a unique member name.

When a PDS consists entirely of CLISTS, it is called a CLIST library. A CLIST library may also consist of a concatenation of individual CLIST libraries.

CLISTS are stored and catalogued like other TSO data sets.

- C Your installation can allocate a PDS to be used as a production CLIST library.
- C You may create your own CLIST library by making your own CLISTS member of a PDS.

4.1 Creating and Editing CLIST Data Sets

There are two ways to create and edit CLIST data sets.

Using options 2 (EDIT) and 3 (UTILITIES) of ISPF/PDF:

1. Allocate a CLIST data set using the allocate panel in ISPF.
2. Create your CLIST in the full-screen environment using the ISPF/PDF editor.
3. Modify the CLIST by making corrections directly to the data on the screen.

Using the TSO EDIT command and its subcommands:

1. Include the CLIST keyword on the EDIT command to inform TSO that you are creating a CLIST data set.
2. Enter and save your CLIST statements, TSO commands, and TSO subcommands.
3. Use subcommands of the EDIT command to modify the CLIST.

Rules and Guidelines

- C When creating a partitioned dataset for CLISTs, it is preferable to make the library type CLIST such as 'userid.name.CLIST'. CLIST libraries may be concatenated.
- C The library with the largest blocksize must be the first one in the concatenation list.
- C New versions of the mainframe operating system do not require the first data set to contain the largest blocksize.
- C A CLIST data set can only be either Variable Blocked or Fixed Record Format. Use the same format as your production CLIST library.
- C Be careful about record formats when copying a CLIST dataset to another CLIST dataset.
 - Variable blocked records have line numbers in columns 1 through 8.
 - Fixed block records have sequence numbers in columns 72 through 80. These numbers must be removed manually when Variable Blocked is copied to Fixed Record Format or vice versa.

5 Executing CLISTs

When executing CLISTs use the:

- C EXECute command when the CLIST is under option 6 of ISPF.
- C EXECute subcommand when the CLIST is under the TSO EDIT command.

Unless the user terminates the EDIT mode first with the End Subcommand, only the EDIT subcommands can be used in the executing CLIST and other CLIST statements.

After EDIT mode has been terminated, all other TSO commands may be included in the CLIST.

5.1 Background Submission

Foreground CLIST:

```
SUBMIT(*) END(XX)
//BATCHJB JOB ...
//SUBMEET EXEC PGM=IKJEFT01
.
.
.
//SYSTSPRT DD SYSOUT=*
//SYSTSIN DD *
EXEC `USERID.LIBNAME.CLIST(CLISTNM)`
XX
```

5.2 Foreground Job Submission

A foreground job submission executes a CLIST in foreground.

The CLIST will often execute a CALL command to execute a program in foreground.

```
CALL `USERID.NAME.LOAD(PROGNAME)`
```

```
CALL LIBNAME(PROGNAME)  
(TSO will add USERID & LOAD)
```

6 EXECUTE Command

The EXEC command has two forms:

- C Explicit
- C Implicit

Explicit Form

Enter "exec or "ex" followed by one of the following:

1. The name of the data set and the member name, with the member name enclosed in parentheses.
2. The name of the CLIST enclosed in parentheses.
3. The name of the CLIST.
4. The fully qualified name of a data set enclosed in single quotes.

Implicit Form

Enter only the name of the CLIST, optionally preceded by a percent sign - %.

The implicit form works only when executing CLISTs that are members of a PDS allocated to the file SYSPROC.

The two implicit forms are:

1. Enter only the member name

When using this form, TSO searches several libraries before it searches the SYSPROC file to ensure that the name you entered is not a TSO command.

2. Enter the member name prefixed with a percent sign (%).

When you use this form, called the extended implicit form, TSO searches only the SYSPROC file for the name, thus reducing the amount of search time.

Some of the commands in a CLIST may have symbolic variables for operands. When you specify the EXEC command, you may supply actual values for the CLIST to use in place of the symbolic variables.

The EXEC subcommand of EDIT performs the same basic functions as the EXEC command. However, a CLIST which is executed with the EXEC subcommand of EDIT can only execute CLIST statements and EDIT subcommands.

6.1 Explicit Form Examples

C EXEC LIBNAME (CLISTX)

CLISTX is a member in USERID.LIBNAME.CLIST.

C EXEC (CLISTX)

CLISTX is a member in USERID.CLIST.

C EXEC CLISTX

where the sequential data set USERID.CLISTX.CLIST contains the CLIST statements.

C EXEC 'FIRST QUALIFIER.CLISTPGM'

where the fully qualified data set name contains CLIST statements.

6.2 Implicit Form Example

CLISTPGM or %CLISTPGM

Where CLISTPGM is a member in a library allocated to SYSPROC.

If no % is used, TSO searches many other libraries before searching SYSPROC.

7 Methods for Allocating Data Sets to SYSPROC

There are three methods of allocating partitioned CLIST data set to SYSPROC.

1. Allocate them by a CLIST that you execute when you log on.
2. Allocate the data sets at the terminal using the ALLOCATE command.
3. Have your logon procedure modified to allocate them.

When using method 1 or 2, include the REUSE operand on the ALLOCATE command. The RESUE operand enables you to use an already allocated filename without having to free it.

The following commands allocate libraries to SYSPROC.

```
ALLOCATE FILE(SYSPROC) -  
  DSNAME('XXXX.YYYY.CLIST' -  
  'AAA.BBBB.CLIST' -  
  'CCC.DDD.CLIST') -  
  SHR-  
  REUSE
```

LIBRARY 'XXXX.YYYY.CLIST' must have the largest block size.

A Variable Blocked format library should not be concatenated with Fixed Block format.