75-231

Cornell University

Department of Computer Science -- Office of Computer Services

PL/CT - A Terminal Version of PL/C
Release 1

User's Guide to the Cornell-TSO Version

R. W. Conway, C. G. Moore III and S. L. Worona

PL/CT is a special version of PL/C designed to permit programs to be run interactively from a typewriter terminal. It is completely compatible with normal PL/C -- that is, the source language accepted by PL/C and PL/CT is identical and the result of execution is exactly the same. Hence a program can be developed and tested under PL/CT and subsequently run under normal PL/C, or vice-versa. Release 1 of PL/CT is a preliminary version; the design of subsequent releases will be guided by user experience with Release 1. Comments and suggestions are invited.

Release 1 of PL/CT permits the user to interact with the program during its _execution_. Output will be printed on the terminal and input data may be requested from the terminal. The course and rate of execution can be controlled from the terminal. It is also possible to interrupt execution and display and alter the values of variables. However, Release 1 does not permit any change in the source program. It receives a complete program, compiles it, and then executes it in interactive mode. But to make any change in the program it is necessary to leave PL/CT, make the change under some editing system, and then present the modified program to PL/CT for complete recompilation. The high compilation speed of PL/C may make this mode of operation tolerable. If not, future releases can permit interactive compilation, and various forms of program modification, but the implementation of this ability is non-trivial and we are reluctant to undertake it until it is clear that it is really necessary.

This Guide provides only minimal information about TSO, perhaps sufficient for very straightforward programming tasks. For additional information see the OCS _TSO Primer_ or the IBM _TSO Command Language Reference_ manual (GC28-6732).

Contents

Sign-on Procedure and Terminal Usage (2741 terminal)

To initiate a TSO session from a dial-up 2741 terminal with an acoustic coupler, perform the following steps:

1. Turn on the terminal (switch at right of keyboard)
2. Place telephone handset in coupler.
3. Turn on the coupler.
4. Dial (9-) 257-1410
5. Wait for terminal keyboard to unlock, then enter popo in lower case (no shift) and press "return" to transmit the entry to the system. The terminal will then prompt you to "logon".
6. Enter logon act non (again in lower case) where "act" is your three-character account number and "non" suppresses printing of system notices.
7. The terminal will then instruct you to enter your four-character password (and overprint four spaces so that the password will not be legible on the listing).
8. The terminal will then reply "READY" indicating that it is in command mode and ready to receive a TSO command.

After you enter a line on the terminal it is transmitted to the system by pressing "return". You must then wait for the system to process the line before entering another line. In many cases the terminal will indicate its readiness with some prompting message, but in some cases there will only be a faint click as the keyboard unlocks ready for your next entry.

To correct a typing error on the terminal simply backspace over the incorrect characters and enter new characters. Only the last character typed in a given column position will be transmitted to the computer. In order to delete an entire line and start fresh, hit "attention" (rather than "return").

Normal practice is to enter letters in lower case, since this helps to distinguish your input from the terminal response on the listing.

Levels and Modes

The most complicated aspect of using TSO-PL/CT is understanding that you are communicating with the system at several different levels. Sometimes you are entering commands telling the TSO command processor what to do, sometimes you are entering lines that are PL/I source statements, and sometimes you are entering data required for the execution of your PL/I program. It is essential that you understand the difference between these levels, and that you understand the means by which you indicate the proper level to the system.

TSO handles this problem of levels by establishing different "modes" of communication. The highest level is called _command mode_. When the system is in command mode it assumes that anything entered is a command (and not a program line or data). Two of these commands change the mode of the system. EDIT causes the system to enter _EDIT mode_; PLC causes it to enter _PL/CT mode_. When the system is in EDIT mode it assumes that anything entered is an EDIT sub-command. One of these sub-commands causes the system to shift to _INPUT mode_ in which it assumes that everything entered is a line to be stored in a TSO dataset. Similarly there are modes within PL/CT that determine whether input from the terminal is considered to be source program lines, execution data, or execution debugging sub-commands. It is difficult to describe, but it works fairly naturally in practice. The relationship between levels, modes, commands and sub-commands can be summarized in the following table:

Mode: Command
  Actions: (commands)
    EDIT to enter EDIT mode
    DELETE to delete a dataset
    LISTCAT to list the names of datasets
    LOGOFF to leave TSO
    PLC to enter PL/CT mode

Mode: EDIT
Actions: (sub-commands)
INPUT to enter INPUT mode
CHANGE to change part of a line
SAVE to save a copy of a dataset
LIST to display all or part of a dataset
DELETE to delete lines of a dataset
RENUM to renumber lines of a dataset
END to return to command mode

Mode: INPUT
Actions:
Enter source program line(s)
Enter empty line to return to EDIT mode

Mode: PL/CT
Actions:
Enter source program lines
Enter execution data
Debug sub-commands
Returns to command mode when program is completed or stopped, or two consecutive attentions.

TSO Command Mode

The system indicates that it is in command mode with the prompting message 'READY'.

Commands (optional abbreviation below full form)

EDIT dataset-name NEW PLC
E OLD

The dataset-name is a string of not more than 8 characters, beginning with a letter.

Specify either NEW or OLD. NEW is specifed if you do not already have a dataset by this name. OLD is specified if you wish to modify an existing dataset.

The execution of this command shifts the system to EDIT mode. But if NEW is specified the editor automatically supplies an initial INPUT sub-command and goes directly into INPUT mode.

DELETE dataset-name
D

Delete the indicated dataset.

LISTCAT
LISTC

List the contents of the dataset catalog -- the names of all your datasets. (The names will appear in "full" form, rather than the simple form that is sufficient for the purposes described here.)

LOGOFF

Indicates the end of your terminal session with TSO. The system will reply with a line giving the cost of the session. System leaves command mode and will accept no further commands. Turn off the terminal and the coupler and replace the telephone handset.

PLC sp-list, DATA(d-list), OPTIONS(op-list), PAUSE

Causes the system to enter PL/CT mode to compile and execute a PL/C program. "Sp-list" specifies the source program; "d-list" specifies the source of input data for execution of the program; "op-list" specifies PL/C options; PAUSE causes a return to the terminal before beginning execution of the program. All of the phrases after PLC are optional; their order is immaterial except that sp-list (if given) must come first.

The source-program dataset (specified by sp-list) contains lines equivalent to the source program cards submitted to batch-PL/C -- including PL/C control cards (*PL/C, *PROCESS, etc.) Sp-list is in one of the following forms:

1. a TSO dataset name

2. an asterisk, indicating program will be entered from the terminal. This is the default assumption if no source-program-list is given.

3. a list of dataset names, separated by commas and enclosed in parentheses. The datasets listed are "concatenated" -- the first line of one dataset follows the last line of the predecessor dataset on the list -- and presented as a single dataset to PL/CT. At most one asterisk may be given as an item on this list, indicating that one portion of the concatenated dataset is to come from the terminal.

D-list is in one of the following forms:

1. If the concatenated source-program dataset contains a *DATA line, input data will be drawn from that dataset. The DATA option on the PLC command should be omitted.

2. If there is no *DATA line in the source-program dataset and the DATA option is omitted, then DATA(*) is assumed and PL/CT will return to the terminal for input data.

3. If there is no *DATA line in the source-program dataset and the DATA option is given on the PLC command then input data will be drawn from the dataset concatenated from the items given in the datafile-list. Items may be TSO datasets; one item on the list may be an asterisk (if no asterisk was given in sp-list).

When a dataset is entered from the terminal, its end must be indicated by entering an end-of-file line consisting of "/*" from the terminal.

Op-list specifies PL/C options to be applied _after_ any options that may be given on a *PL/C card in the source-program dataset. (This may be used to override the *PL/C options. A *PL/C card need not be present.)

Examples:

PLC
: Both source program and input data are to come from the terminal. The source program lines must conclude with a *DATA line to initiate execution, just as in batch-PL/C.

PLC PROB1
: Source program is to come from PROB1. If this includes a *DATA card it will also supply data, otherwise data will come from the terminal.

PLC DATA(P1DATA)
: Source program is to come from the terminal; input data from P1DATA.

PLC PROX4, DATA(XDATA), OPTIONS(ATR, XREF)
: Source program is to come from PROX4, data from XDATA. The cross-reference and attribute listing is to be printed.

PLC (*,PRG1), PAUSE
: Source program is to come first from the terminal (perhaps to supply a *PLC card) and then from PRG1; input data is to come from the terminal (unless PRG1 includes a *DATA line). Return to the terminal before beginning execution.

PLC (CS104,LIBR), DATA(INIT,*)
: Source program is to come from CS104 followed by LIBR, data is initially to come from INIT and from the terminal when that is exhausted.

TSO EDIT Mode

The TSO EDIT facility permits the creation and modification of TSO datasets. The following describes a portion of the full EDIT, assuming you are working with a relatively small dataset and that you have a listing of that dataset available (to determine line numbers). The full EDIT is much more powerful and flexible than what is described here -- see the IBM _TSO Command Language Reference Manual_.

Lines are automatically numbered by the editor, starting at 10 and increasing by 10. You can subsequently insert additional lines in these intervals, and there is a command to renumber the entire dataset to restore the uniform 10-increment numbering.

EDIT Sub-commands (optional abbreviation below full form)

INPUT
I

Add lines to end of dataset. Enter empty line to return to EDIT mode.

INPUT n R
I

Replace lines beginning with line number n with text given, line by line, at the terminal. Enter empty line to return to EDIT mode.

INPUT n1 n2 I
I

Insert lines from the terminal after existing line n1. The first inserted line will be numbered n1+n2, the second n1+2*n2, etc. The insert will be terminated whenever the next insertion would overwrite an existing line. Enter empty line to return to EDIT mode.

CHANGE n 'string1' 'string2'
C

In line number n replace string1 by string2. If string1 occurs more than once in line n only the first (leftmost) occurrence will be replaced by string2. String1 and string2 do not need to be of the same length.

SAVE
S

Save the dataset currently being edited, in its current form, destroying any previous form under the same name.

LIST n1 n2
L

List the lines from n1 to n2. If n2 is omitted list only n1. If both n1 and n2 are omitted list the entire dataset.

DELETE n1 n2
D

Delete the lines from n1 to n2. If n2 is omitted delete only line n1.

RENUM
REN

Renumber the entire dataset starting with 10 and with an increment of 10.

END

Return from EDIT to command mode. Note the working copy of the dataset will be destroyed on leaving EDIT mode, so if you want to keep it you must specify SAVE before END.

PL/CT Mode

The PL/CT source language is identical is PL/C Release 7. The PL/CT default options are different (to reduce the amount of printing). The following default options are different from those of batch PL/C:

CMPRS	NOHDRPG
NODUMP, NODUMPE, NODUMPT	NOOPLIST
FLAGE	NOSOURCE

PL/CT Source Program Entry

Programs can be entered in one of two ways:
1. directly to PL/CT from the terminal, or
2. by preparing a TSO dataset which is then presented to PL/CT.

For all but trivial programs the TSO dataset method should be used, since it provides a means of saving the source program for subsequent reuse and/or modification. If a program is presented directly to PL/CT it is not saved in the system and must be reentered to be reused.

Note that once a source line has been presented to PL/CT (either from the terminal or from a dataset) there is no way within PL/CT to change that line. You must leave PL/CT, change the program, and then re-invoke PL/CT.

Disposition of Output

Under PL/CT the terminal serves in the same role as the printer under batch PL/C -- it receives the SYSPRINT lines. While it is possible to route output under PL/CT to a TSO dataset rather than the terminal, and subsequently print such a dataset on one of the high-speed printers, this involves the use of TSO commands that are beyond the scope of this Guide.

Terminal Use during Execution

There are four situations in which PL/CT returns to the terminal and waits for input:

1. An "attention" interrupt from the terminal.

2. The completion of a command line from the terminal.

3. The occurrence of a non-fatal error during execution of the program (following printing of normal PL/C error message). Note this applies only to <u>execution</u> errors. Errors detected during translation will be printed but will not return control to the terminal.

4. Data is requested by execution of a GET statement in the program and not supplied by the D-list dataset.

In the first three cases PL/CT will explain why it has returned to the terminal, and request entry of a debug sub-command by printing "DBC:". In the fourth case there will be no prompting message supplied by PL/CT. (It is often desirable to print your own "prompt" immediately before each GET.) A data request _must be satisfied_; that is, you cannot enter a debug sub-command when data is expected. In each case, enter your response, and transmit it to PL/CT by pressing "carriage return". If PAUSE is specified on the PLC command, PL/CT will return to the terminal for a debug sub-command immediately before execution of the first _executable_ statement of the MAIN procedure. (Declarations in the MAIN procedure will already have been processed.)

PL/CT will deliver output to the terminal (normally from SYSPRINT, see paragraph above on "disposition of output") resulting from the execution of PUT statements in the program. In this case there is no return of control to the terminal and program execution continues. An attention interrupt during output will suppress further lines of output from that statement and return to the terminal for a debug sub-command.

Two consecutive attention interrupts (without an intervening debug sub-command) will terminate PL/CT and return to TSO command mode.

PL/CT Debug Sub-commands

The sub-command format rules are the following:

1. Format is "free-field" as in PL/C. Many blanks are equivalent to a single blank (except in a string constant).

2. A sub-command may not be continued onto another line.

3. Several sub-commands on the same line will be executed in order, left-to-right. However, the execution of the first RUN or STOP sub-command terminates the scan of the line -- anything to the right of the leftmost RUN or STOP is ineffective.

4. Whenever parentheses enclose a single argument in a sub-command they can be omitted. If two or more arguments are to be given then the parentheses must be given.

The Release 1 PL/CT sub-commands are:

PAUSE (label, ...);
: Insert a "pause marker" immediately before the statement whose label is given on the list. The label must be unsubscripted, and must be accessible under normal PL/C scope rules at the point where sub-command is given.

NOPAUSE (label, ...);
Remove pause markers at statements indicated. Labels must be accessible.

RUN options ;
Options are:

FROM (label) -- resume execution from the statement whose label is given. This statement must be accessible under normal PL/C scope rules. This is equivalent to executing a GOTO in the program followed by a RUN sub-command.

TO (label) - run no farther than up to the statement whose label is given. This is not inclusive; execution will pause before executing the specified statement. A line consisting solely of a label L is interpreted as RUN TO (L);.

STEP (n) - execute at most n statements. The default if STEP is specified without an argument is STEP(1). If the STEP option is omitted STEP(no limit) is implied. A line consisting solely of an integer n is interpreted as RUN STEP (n);.

NOPAUSE (n) - ignore the first n pause markers during execution. If the NOPAUSE option is omitted NOPAUSE(0) is implied. If NOPAUSE is given without an argument NOPAUSE(32767) is assumed.

These options may be given in any combination, in any order. Execution of the program resumes at the point of interrupt (unless the FROM option has been specified). Execution continues until interrupted or the completion of the sub-command as specified by any one of the TO, STEP and NOPAUSE options is achieved.

A null sub-command (an empty line - carriage return) is considered a "rerun" sub-command. It will repeat the previous RUN command.

Variable = constant ;
The target variable must be a scalar or a subscripted variable with constant subscript(s). In Release 1 it cannot be an array, a structure or an element of a structure.

PUT LIST (variable, ...);
Variable is a scalar, an array, a structure or a subscripted variable with a constant subscript. In Release 1 it cannot be an element of a structure or an expression. The output format is such that each variable on the list begins a new line. That is PUT LIST(X, Y); is treated as if it were given as PUT SKIP LIST(X); PUT SKIP LIST(Y);

The keyword LIST is optional and for a single variable the parentheses may be omitted. That is, PUT X is equivalent to PUT LIST(X);.

The following sub-commands are equivalent to the corresponding PL/C statement executed at the point of interrupt:

STOP;	NOFLOW;	PUT ALL;
CHECK;	PUT OFF;	PUT SNAP;
NOCHECK;	PUT ON;	PUT ARRAY;
FLOW;	PUT FLOW;	

PL/CT Errors

When errors are detected during compilation of a program the usual PL/C action is taken. That is, a message is printed, some repair is automatically effected, compilation continues and execution will be attempted. (There are a few cases in which these errors are "fatal" and execution is suppressed.) If the repair is not satisfactory you must leave PL/CT mode, alter the source program, and then re-submit it to PL/CT.

Similarly, during execution of the program PL/CT gives the standard PL/C response -- message and repair -- but then returns control to the terminal and requests a debug sub-command.

An error in entering a debug sub-command will cause an error message to be printed followed by a request for re-entry of the sub-command. The complete sub-command must be re-entered -- not just from the point of error. However, if several sub-commands were given on the line containing the error, sub-commands to the left of the erroneous command will have been executed and need not be re-entered.