

DLG -- current line count
ROBOYS?
TAB?
Formal
APLSV

18 July 1974
Sheldon Linker
Working Document

ATTENTION ALL APL USERS:

This document is designed for those people who have already been using this system. It is not intended for those who have not used CRMS APL before.

The following CHANGES TO APL are in effect.

All programs must be recompiled to run under the new system. To run programs under the old system, type:

OLDARS
SR OLDARUN

to recompile under the old system, type

ATRA
>SR OLDARUN

This will be available for a week or so; after that, the new system must be used.

[1] ? no longer asks for a random seed. As a seed, it uses 93117.

OK [2] Random seeds can be set using ~~□RANDSET SEED~~.

RL [3] ~~□RAND~~ is the current random seed.

TS [4] ~~□TIME~~ returns an 8-element vector containing:

- (1) The Year
- (2) Month (in numbers)
- (3) Date
- (4) Hour ~~(0 to 23)~~ 1 to 24
- (5) Minute
- (6) Second
- (7) Milliseconds
- (8) Microseconds

(more) *g* [5] ~~□SET_ALARM~~[BOX] TIME sets the alarm-clock to go off in TIME ~~milli~~-seconds. The alarm-clock goes off by placing a character scalar in the mailbox BOX. If no box is specified, then the box used is ~~□ALARM~~.

g [6] ~~TIME~~+~~□RESET_ALARM~~[BOX] resets the alarm-clock so that it will not go off. If it has, then BOX or ~~□ALARM~~ is cleared. See Section 5 for conditions on this. This function returns the time left on the clock, if any, in ~~milli~~seconds.

g [7] BOX+~~□BOX_ALARM~~ creates a new alarm-clock and returns its box.

CA/RA? [8] OUTONE no longer exists.

[9] There is no prompting on ~~□INNUM~~ or ~~□INMIX~~.

- [10] *DO NAME OTN* now starts the specified function at terminal # 17 with pre-created I/O boxes and alarm-clock box. If a terminal number is specified than that terminal, instead of #17, is used.
- [11] Master programs should no longer have dummy arguments. This means that master programs should be niladic. This is also true for the new *STAND ALONE* programs (programs which are neither Masters nor Subjects, but run multiply under control of *MULTIPLE*).
- [12] *[TABSET ARG]* can now be used to set teletype tabs. An argument of the null vector will clear all the tabs; zero will set tabs every 6 spaces; and a vector will set a tab at every column specified in the vector. *[TAB]* is defined as the tab character.
- [13] *[START]* no longer exists.
- [14] *[KILL]* no longer exists.
- [15] *[ID]* is no longer the process ID. It is the slave number for slave programs and the robot number for robot programs and has been changed from *[ID]* to *[PROC]*.
- [16] Robot numbers are negative.
- [17] *[CR_SUBJECTS[OARG;OTN_RN] FN]* creates subject processes. *ARG*, if given is passed to the subject process. If it is not given, then the null vector is passed. Therefore, all Subject processes must still be monadic. *TN_RN* is a vector containing terminal and/or robot numbers. As each subject process is created, it is attached either to a terminal or to a robot process. This is transparent to the subject programs, and they have no way of ever finding out. If *TN_RN* is omitted, then this function will ask for the vector on the experimenter's teletype. The subject process starts execution at function number *FN*. *[CR_SUBJECTS FN]* is equivalent to *[START X, FN]*.
- [18] *[SLAVES]* no longer exists.
- [19] *[SUBJECTS]* always exists in the master process and is an N by 7-element matrix. It contains one row per subject. The columns are as follows:
- (1) process number
 - (2) terminal or robot number
 - (3) input box
 - (4) output box
 - (5) master to subject box
 - (6) subject to master box
 - (7) alarm-clock box
- [20] *CR_ROBOTS[OARG;ON] FN* creates robot processes. *ARG*, if present, is passed to the created process. If it is not there, the null vector is passed. *FN* is the function number. If *N* is omitted, then it is asked for on the experimenter's terminal.

[21] $\square ROBOTs$ always exist and is an N by 6-element matrix. It has one row per robot process. The columns are as follows:

- (1) process number
- (2) robot to subject box
- (3) subject to robot box
- (4) robot to master box
- (5) master to robot box
- (6) alarm-clock box

[22] $X \leftarrow \square CR_BOX[N] M$ acts like n occurrences of $\square CR_BOX M$. The result is made into a vector and placed in X.

[23] $MESSAGE \leftarrow \square GET$ receives a message from the master.

[24] $\square GIVE$ MESSAGE sends a message to the master.

[25] $MESSAGE \square GIVE$ TO SN_RM sends a message to the subjects and robots specified in SN_RM.

[26] $MESSAGE \leftarrow \square GET$ FROM ID=SN_RM receives message from some subject or robot process and stores the senders process-id in ID.

[27] $\square CRLF$ is defined as the two characters "carriage return" and "line feed".

[28] $\square DE_SUBJ$ SN_RM deletes all the subject and robot processes specified and everything connected with them except $\square ROBOTs$ and $\square SUBJECTs$. (?)

[29] PI no longer exists.

[30] $\alpha START$ is now ~~MULTIPLE~~.

NI IP (multiple identical independent processes)

[31] $\square SINGLE$ is a niladic function. It creates a process which starts execution at your first function (function #2). This function must be a monadic function (function 2).

[32] The STATE vector of $\square CR_PROC$ has been changed. It is as follows;

- (1) $\square WIDTH$
- (2) $\square DIGITS$
- (3) $\square PROC$
- (4 5) $\square IO_BOXES$
- (6 7) $\square IPC_BOXES$
- (8) $\square ALARM$
- (9) $\square RAND$

[33] $\square WAIT$ TIME will suspend execution for TIME ~~milliseconds~~.

seconds

TIME need not be integral return not delay

[34] $\square QUOTE$ is equivalent to APL\360's ' operator.

[35] $\square CR_LOG[ONAME]$ creates a logging file named NAME, or if no name is given, named 'NONAME'. It also creates two mailboxes and the vector $\square LOG_BOXES$.