ES RSU 1A

OPERATING SYSTEM

ERAMCO SYSTEMS
RSU-1A Module
Quick Reference Guide

In this quick reference guide for the ES RSU-1A operating system the input for, and result of, each function is shown in a similiar format:

FN (Function Name deriviation)
Input : data, format, etc.
Output or function results, etc.

CLDF (CLear DataFile)
Alpha : Filename
All registers in the named file will be set to zero. The working file remains unchanged.

CLEM (CLear Extended Memory)
No user input required
All Extended Memory is cleared.

CLPR (CLear Program in Rsu)
At prompt : Name of program
The named program in an RSU block is deleted.

CLRSU (CLear RSU)
No user input required
The contents of all connected RSUs are cleared.

CLSEC (CLear SECured status)
X-reg : RSU block number

The specified RSU program block is unsecured.

CRDF (**CR**eate a **D**ata**F**ile)
Alpha : Filename
X-reg : Filelength in regs
A datafile of the specified length and name is created in an RSU data block. The newly created file becomes the working file.

DELFL (**DEL**ete **F**i**L**e)
Alpha : Filename
The specified file is deleted.
The "working file" is unspecified.

DF+ (**DF**-reg addition)
X-reg : Value to add
The contents of the X-reg are added to the DF-reg pointed to by the DF-pointer

DF- (**DF**-reg subtraction)
X-reg : Value to subtract
The contents of the X-reg are subtracted from the DF-reg pointed to by the DF-pointer.

DF* (**DF**-reg multiplication)
X-reg : Value to multiply
The DF-reg pointed to by the DF-pointer is multiplied by the contents of the X-reg.

DF/ (**DF**-reg divide)
X-reg : Value to divide by

The DF-reg pointed to by the DF-pointer is divided by the value of the X-reg.

DPTDF (Decrement PoinTer DataFile)
No user input required
The DF-pointer is decremented by one. If already zero, it remains zero.

EXALL (EXchange ALL main memory)
Alpha : Write-all filename
The entire contents of the calculator's main memory and the named file are swapped.
If flag 11 was set when the DF write-all file was recorded then program execution will start automatically at the line where the program counter was positioned.

EXEM (EXchange Extended Memory)
Alpha : Extended memory filename
The entire contents of the Extended Memory and the named file are swapped.

EXREG (EXchange REGisters)
No user input required
All main memory data storage registers are swapped with DF-registers, starting at the DF-pointer. The DF-pointer is left set to the first DF-reg after the last swapped DF-reg.

Partial exchange may occur when there are fewer DF-registers available than there are main memory registers. This is indicated by a cleared X-reg after the operation.

EXREGX (**EX**change **REG**isters by **X**)
X-reg : BBB.EEE
Exchanges main memory registers with DF-registers, starting at BBB and ending at EEE. See EXREG for the behaviour of the DF-pointer.

EXST (**EX**change **ST**ack registers)
No user input required
Exchanges the stack, in T-Z-Y-X-LastX order, with DF-registers, starting at the DF-pointer.
The DF-pointer is left set to the first register after the last-swapped register.

FIND (**FIND** target value)
X-reg : Target value
Y-reg : DF-pointer step size
The working datafile is searched for a match with the numeric or string value in the X-register. The entire value is compared: embedded substrings are ignored.
The Y-reg controls the searched register step-size: a negative value specifies that the search steps down by the associated number of DF-registers, a positive value specifies stepping up.

When a program is running the next program step is executed if the target value is found, otherwise not ("do if true").
If the function is executed from the keyboard the message "**YES**" will be displayed if the target string is found, otherwise "**NO**".
If the target value is found, the DF-pointer is set to that DF-register, otherwise it is left unchanged.

FINDI (**FIND** target value **I**nverse)
X-reg : Target value
Y-reg : DF-pointer step size
As for FIND, except that when a program is running it will **skip** the next step **if** the target value is **found**, else execute it.
When executed from the keyboard the message "**NO**" is displayed **if** the target string is **found**, otherwise "**YES**".
See also FIND, above.

FLDIR (**FiLe DIR**ectory)
No user input required
Displays filetype, length and name of all RSU-files.
When the directory ends the X-register will contain the number of free registers left in the RSU data block. If multiple data blocks exist the X-register will refer to the block with the largest number of free registers
If a printer is attached, the directory will be printed. This can be stopped with **R/S**.

If the **ON** key is pressed the calculator is switched off. Pressing any other key will pause the directory until the key is released.
The last displayed directory entry remains in the alpha register.

INIDATA (**INI**tialize **DATA** block)
X-reg : Block to initialize
The contents of the specified RSU block will be cleared and the block will be internally identified as an RSU data block (this identification is invisible to the user).
The number of free registers in a newly initialized RSU data block is 679.
After execution of INIDATA the "working file" becomes unspecified.

INIPRGM (**INI**tialize **PRoGraM** block)
Alpha : Name of RSU program block
X-reg : Block to initialize
At prompt: XROM number
The specified RSU block will be cleared and identified as a program storage block (This identification appears in CAT 2: its XROM number will be the number input at the prompt, and the CAT 2 "header" is set to the name in the alpha register).
After execution of INIPRGM the "working file" becomes unspecified.

IPTDF (**I**ncrement **P**oin**T**er **D**ata**F**ile)
No user input required

The DF-pointer will be incremented by one.

LOADP (LOAD Program to prgm block)
At prompt: Name of program
The named program in main memory will be loaded into an RSU block, initialized with INIPRGM. If an RSU program block is secured it is not possible to load programs into this block (see SETSEC & CLSEC).
If sufficient RSU memory is not available the (highest) number of free program registers in the RSU program block (or blocks) is returned to the X-reg, otherwise the X-register is not altered.

MPTDF (Move PoinTer DataFile)
X-reg : Step size DF-pointer
The DF-pointer will be moved by the number of registers specified in the X-reg.
The sign of the X-reg determines if the DF-pointer is decremented or incremented. A negative value will decrement the DF-pointer and a positive value will increment the DF-pointer.
If an attempt is made to move the DF-pointer below zero the pointer is set to zero.

RCLALL (ReCaLl ALL main memory)
Alpha : Write-all filename
The entire contents of the calculator's main memory are replaced by the contents of the named file.

If flag 11 was set at the time of recording the specified file then program execution will start automatically at the line where the program counter was positioned.

RCLDF (ReCaLl DataFile register)
No user input required
The contents of the DF-reg pointed to by the DF-pointer will be transferred to the X-reg.
If the stack lift is enabled the stack will be raised and the T-register lost, if not the X-register will be overwritten.

RCLDFX (ReCaLl DF-reg by X)
X-reg : DF-reg to recall
The DF-pointer is set to the DF-reg indicated and the RCLDF function is performed.

RCLEM (ReCaLl Extended Memory)
Alpha : X-Memory filename
The contents of the named Extended Memory file in an RSU block is transferred to Extended Memory.

RCLIDF (ReCaLl df-reg Increment DF-pointer)
No user input required
The contents of the DF-reg pointed to by the DF-pointer is transferred to the X-register and the DF-pointer is incremented by one. See RCLDF for remarks on stacklift.

RCLREG (ReCaLl to REGisters)

No user input required

All main memory storage registers are filled with the contents of the DF-registers in the working file. Transfer starts from the DF-reg pointed to by the DF-pointer. See EXREG for behaviour of the DF-pointer.

RCLREGX (ReCaLl REGisters by X)

X-reg : BBB.EEE

The contents of the DF-registers, starting at the DF-pointer, are transferred to the main memory storage registers starting with register BBB and ending at register EEE. See EXREG for behaviour of the DF-pointer.

RCLST (ReCaLl to STack registers)

No user input is required

The contents of the five DF-registers starting at the DF-pointer are transferred to the stack registers T,Z,Y,X and Lastx. The DF-pointer is set to the next DF-register after that containing the new Last-X.

READRAM (READ a RAMfile from tape)

Alpha : Filename to read
X-reg : RSU block to read file to

The named RSU file is read from mass storage and transferred to the RSU block indicated by the X-register.
The "working file" becomes unspecified.

RENFL (RENname a FiLe)

Alpha : Oldname,Newname

The file with "oldname" will be renamed "newname". The new filename must be unique.

RESDF (RESize DataFile)

Alpha : Filename

X-reg : New size of datafile

The named datafile will be resized as specified by the X-register. When resized data storage blocks are packed to leave as much space as possible.
The "working file" becomes unspecified.

RLEFT? (Room LEFT?)

No user input required

After execution the X-reg will contain the number of free registers of the datastorage block with the most registers free.
If the stacklift is enabled the stack will be raised and the T-register lost, if not the X-register will be overwritten.

RPTDF (Recall PoinTer DataFile)

No user input required

The value of the DF-pointer is recalled to the X-reg. For remarks on stacklift see RLEFT?.

RSTDF (ReSeT DataFile)

Alpha : Filename

The named file will become the working file.
The DF-pointer is set to zero.

RSTDFX (ReSeT DataFile to X)

Alpha : Filename
X-reg : DF-pointer value

As for RSTDF except that after the "working file" is respecified the DF-pointer is set to the value in the X-reg.

SETSEC (SET prgm block SECured)

X-reg : Block to secure

The specified program block will be secured to prevent the loading or deleting of programs in the specified programstorage block.

SPTDF (Set PoinTer DataFile to x)

X-reg : New pointer value

The DF-pointer of the working file is set to the value in the X-reg.

STOALL (STOre ALL main memory)

Alpha : Filename

The entire contents of the calculator's main memory will be stored in a file with the specified name.
If flag 11 is set at storage time program execution will resume automatically after EXALL or RCLALL.

STODF (STOre x in DataFile)

X-reg : Value to store

The X-reg will be stored in the datafile at the current DF-pointer position.

STODFY (STOre x in DataFile at Y)

X-reg : Value to store
Y-reg : DF-reg to store in

The DF-pointer will be set to the DF-reg indicated by the Y-reg and the contents of the X-register stored in that DF-reg.

STOEM (STOre the Extended Memory)

Alpha : Filename

The entire contents of the Extended Memory are stored in a file named in alpha.

STOIDF (STOre x at df-pointer and Increment the DF-pointer)

X-reg : Value to store

The value in the X-reg is stored in the DF-reg pointed to by the DF-pointer and the DF-pointer is incremented to the next DF-reg.

STOREG (STOre REGisters)

No user input required

All main memory datastorage registers are stored in the working datafile starting at the current DF-pointer position.
After execution the DF-pointer is set to the next not used datafile register.
See remarks at EXREG for behaviour when a partial store is done.

STOREGX (STOre REGisters by X)

X-reg : BBB.EEE

Main memory datastorage registers starting at BBB and ending with EEE are stored in the working datafile starting at the current DF-pointer position.
See EXREG for partial store and DF-pointer behaviour.

STOST (STOre STack registers)
No user input required
The registers T,Z,Y,X and Lastx are stored in the datafile starting at the current DF-pointer. For further remarks see also EXST.

SZ?DF (SiZe ? DataFile)
No user input required
Returns the size of the working datafile to the X-reg. For behaviour of stacklift see RLEFT?.

WRTRAM (WRiTe RAM to tape)
Alpha : Filename
X-reg : Block number
The contents of the specified RSU block is stored in a special filetype on the selected mass storage medium. A file check byte is computed and placed at the end of the file.
This special filetype is automatically secured and will show up in the mass storage directory as follows:

NAME	TYPE	REGS
TSTFILE	??,S	640

X<>DF (exchange X with DF-reg)
X-reg : Value to exchange
The value of the X-reg is exchanged with the contents of the DF-reg pointed to by the DF-pointer.

X<>DFY (exchange X and DF-reg at Y)
X-reg : Value to exchange
Y-reg : DF-reg to exchange
The DF-pointer is set to the DF-reg indicated by the Y-register and the contents of the DF-reg pointed to by the new DF-pointer and the X-reg are exchanged.

X=DF? (X-reg = DF-reg ?)
X-reg : Comparing value
The X-register and the DF-reg pointed to by the DF-pointer are compared. If they are either alphabetic or numerical equal and when a program is running the next step is executed, otherwise it is skipped ("do if true").
If executed from the keyboard and the result is true "YES" is displayed, otherwise "NO".

X<>DF? (X-reg <> DF-reg ?)
X-reg : Comparing value
The X-register and the DF-register pointed to by the DF-pointer are compared for either alphabetic or numerical inequality. See X=DF? for an explanation of the result.

X<DF? (X-reg < DF-reg ?)
X-reg : Comparing value

The X-register is tested for being either alphabetic or numerical less than the DF-reg pointed to by the DF-pointer. See X=DF? for an explanation of the result.

X<=DF? (X-reg <= DF-reg ?)
X-reg : Comparing value
The X-register is tested for being either alphabetic or numerical less than or equal to the DF-reg pointed to by the DF-pointer. See X=DF? for an explanation of the results.

X>DF? (X-reg > DF-reg ?)
X-reg : Comparing value
A test is performed on the X-register and the DF-reg pointed to by the DF-pointer to see if the X-register is either alphabetic or numerical greater than the DF-reg. See X=DF? for an explanation of the result.

X>=DF? (X-reg >= DF-reg ?)
X-reg : Comparing value
The X-register is tested for being either alphabetic or numerical greater than or equal to the value in the DF-reg pointed to by the DF-pointer. See X=DF? for an explanation of the result.

X=PT? (X-reg = DF-pointer ?)
X-reg : Comparing value

The x-register is numerically compared with the value of the DF-pointer. See X=DF? for an explanation of the result.

ERROR MESSAGES

ALPHA DATA Alpha data was input when numerical data was expected.

BAD PAGENO The specified RSU "block" number is illegal.

BAD ROMFILE An RSU file on mass storage has been misread, or corrupted or was misrecorded.

BAD XROMNO The specified Xrom number is already present in the system, or is greater than 31, or is 0.

CONFIG ERR The number of Main or Extended Memory modules plugged in when an RSU main or extended memory "ALL" file is being recalled is different from the number at storage time.

DATA ERROR Illegal input data has been entered.

DUP FL NAME The specified file-name already exists.

END OF FL	An attempt is made to set the DF-pointer beyond the end of the file.
FL TYP ERR	An attempt has been made to use an incorrect file-type for the required function.
NAME ERROR	The specified name is too long.
NO EXT MEM	The present system does not contain an Extended functions memory module.
NO FL NAME	The alpha register does not contain an RSU filename.
NO HPIL	No HP-IL module is plugged in.
NONEXISTENT	The ES RSU-1A system is not present or function parameters greater than 999 have been input.
NO PRGMROM	No RSU program block is initialized or all present are secured.
NO ROOM	None of the RSU data blocks has enough free registers to perform the function.

NO RSU PAGE	The specified "block" is not an RSU page, or it has been switched off.
NOT A FILE	The name in the alpha register is not that of an RSU file.
NO WORKFL	There is no working file specified.
PACKING, TRY AGAIN	There is not enough room in main memory to hold the working file information, or no register is free for holding an END.
PRGMROM SEC	Programs cannot be deleted from, or added to, a secured RSU program block.
PRIVATE	An attempt has been made to load a private program into an RSU program block.
RAM	An attempt has been made to use CLPR to clear a program which exists in main memory but not in an RSU program block.
ROM	A program of the same name already exists in an RSU program block (or ROM).

RUN-TIME MESSAGES

CLEAR RSU	The contents of all connected RSUs are being deleted.
LOADING PRGM	A program is being loaded from main memory into an RSU program block.
PACKING	The program being loaded is packed to use as little space as possible.
READRAM	An RSU block is being read from mass storage.
WRTRAM	An RSU block is being written to mass storage.

PROGRAMMABILITY

All functions are programmable except: **CLPR INIPRGM LOADP**

ERAMCO SYSTEMS
W. van Alcmade str. 54
1785 LS Den Helder
The Netherlands