

## -ML ROM FUNCTION REFERENCE

ROM Entry Point functions:

**PROMPT LENGTHENER:** While executing a prompting function, pressing the [ON] key will lengthen the prompt to enter more digits. To enter RCL M for example: RCL [ON], display shows RCL \_ \_ \_, enter 117 and the function will be executed or inserted in the current program (in PRGM mode).

**ROM CATALOG:** Pressing ENTER while switching on the calculator will show a list of plugged modules.

For the following descriptions double quotes will mean a hexadecimal (non-normalized, NNN) number in a register (like X): "04FACDB". Text in alpha is shown in single quotes: 'This Is Text', numbers without quotes are regular numbers in a register.

**-ML ROM**      **XROM 21,00**      **ROM name, misc. functions**  
**Normal mode:** Shows the full mantissa of X in the display. <- displays X again  
**PRGM mode:** Goes to the beginning of the last program in memory  
**FIX 15:**      Extended Tone, in X "LLLT", LLL is length, T is pitch

**AND**      **XROM 21,01**      **Bitwise AND**  
Bitwise AND of X and Y, result in X.

**OR**      **XROM 21,02**      **Bitwise OR**  
Bitwise OR of X and Y.

**FATN**      **XROM 21,03**      **Function Address Table Name**  
If X contains the address of a FAT entry "AAAA" will result in the actual function address in X, address of next FAT entry will be in L. If the program is usercode the result is "2AAAA".

**CLBL**      **XROM 21,04**      **Clear Block microcode**  
In X "BBBBEEEE", erases a block of MLDL memory to zero, BBBB is the start address and EEEE the end address. If X contains only a page number "P" that whole page will be erased.

**MOVE**      **XROM 21,05**      **MOVE block microcode**  
In X "BBBBEEEEDDDD", moves a block of microcode starting at BBBB and ending at EEEE to destination DDDD. Useful for copying entire ROM images. Destination must be MLDL ram.

**AFAT**      **XROM 21,06**      **Append Function Address Table**  
In X "BOPAAA", ads a function to the FAT in page P, starting at address AAA. B=0 for a microcode program, B=2 for a usercode program. O=offset, should be 0 in almost all cases. If the number of functions in the FAT is larger than 64 there will be an error message.

**DFAT**      **XROM 21,07**      **Delete Function Address Table**  
Like AFAT: in X "OPAAA". (B is not used), removes the function starting at PAAA from the FAT. Error message if address is not in the FAT.

**D>B**      **XROM 21,08**      **Decimal to Binary**  
Converts a normal number to binary NNN, for example: 153 becomes "099". Maximum 12 bits are supported, X is saved in L.

**B>D**      **XROM 21,09**      **Binary to Decimal**  
Inverse function of D>B.

**^XCAT**      **XROM 21,10**      **Extended Catalog**  
Usercode program, prompts for an XROM number and prints a catalog listing of that ROM with XROM numbers and start addresses of the functions. XCAT will first execute ROMCHK, the catalog will not be printed if the checksum is bad, but will print it anyway after R/S.

**^PDS**      **XROM 21,11**      **Print Disassembly listings**  
Usercode program to prints a disassembly listing using the DAVID Assembler. Prompts for start and end address, but end address not required. SIZE 001 is always set. Do not use PDS to print listings of the system ROMs, but write a simple program as described in the DAVID Assembler manual.

**SRCH**      **XROM 21,12**      **Search microcode word**  
In X "BBBBEEEEWWW", finds the microcode word WWW between addresses BBBB and EEEE. Address of the found word is put in alpha, X will contain the correct starting address for the next find operation. If the word is not found, display will show NO, and the next program line will be skipped if used in a running program.

**CODE**      **XROM 21,13**      **Code alpha to X**  
Codes alpha to a NNN in X: 'HHHHHH' in alpha becomes "HHHHH" in X. Characters in alpha must be valid hex digits.

**DECODE XROM 21,14 Decode X to alpha**

Inverse of CODE: X will be put in alpha and display.

**COMPILE XROM 21,15 Compile User Code**

Compiles a usercode program by calculating all GTO's. 2-byte GTO's may be replaced by 3-byte GTO's. Name of program must be in alpha. If a label does not exist there will be an error message. COMPILE is not programmable.

**NOPS XROM 21,16 find NOPS in microcode**

In X "BBBBEEEE", finds blocks of at least two microcode NOPS between BBBB and EEEE. Address and number of NOPS found will be in alpha, X will contain the correct starting address for the next NOPS operation. If no NOPS found display will show NO, and the next program line will be skipped if used in a running program.

**CVIEW XROM 21,17 see and VIEW alpha**

Alpha will be printed if a printer is connected else shown in the display. Flag 21 is ignored and a running program will not stop.

**VIEWA XROM 21,18 VIEW Alpha**

Shows the alpha register in the display.

**BYTEC XROM 21,19 BYTE Count**

Shows the length in bytes of a usercode program. If executed in normal mode the number of used registers will be in X. Program name must be in alpha otherwise the current program will be used. BYTEC is non-programmable.

**GETPC XROM 21,20 GET Program counter**

Puts the current Program Counter in C according to the MM-format, to be used with PUTPC, INSBYTE, RCLBYTE, STOBYTE.

**PUTPC XROM 21,21 PUT Program Counter**

Puts the program counter that is in MM-format in X, in the program counter in the b-register.

**HEXKB XROM 21, 22 Hexadecimal Keyboard**

Alpha is put in the display with a prompt and a hexadecimal keyboard is now active. Digits may be removed by pressing <- and R/S ends the input and continues the program, [shift] R/S stops the program. Digits entered will be in alpha 'HHHH' and X: "HHHH".

**INSBYTE XROM 21,23 Insert BYTE**

Inserts a byte in the current program with the byte in X and the address in MM-format in Y. For example to create a RCL M: go to the instruction before which you want the RCL M, leave PRGM mode and do GETPC, 144, INSBYTE, 177, INSBYTE. INSBYTE drops the stack and increments the pointer that was in Y and is in X after the function. The old X value will be in L.

**RCLBYTE XROM 21,24 Recall BYTE**

In X an address in MM-format, will put in X the byte at the given address. Y will contain the correct pointer to the next byte in memory.

**STOBYTE XROM 21,25 Store BYTE**

Like INSBYTE, but overwrites the byte at the pointer location. See listing of PIOKE.

**LOADP XROM 21, 26 Load User Program to MLDL**

Puts a usercode program from the HP41 to MLDL ram. Name of the program must be in alpha, otherwise the current program will be used. X must contain "AAAA", the first free address of the program in the MLDL. It is the users responsibility to have enough free space in the MLDL. The following flags are used:

Flag 0 Flag 1

clear	clear	program not PRIVATE
clear	set	program will be PRIVATE after a COPY command
set	clear	program will be PRIVATE, can not be copied
set	set	program will be PRIVATE, can not be copied

Flag 2: if set the FAT will be updated, if clear the FAT will not be changed.

After LOADP the first useable address in MLDL ram will be in X for loading the next user program.

**FNM XROM 21,27 Function Name**

Given the address of a function in a ROM "AAAA" the function name will be appended to the alpha register. For a usercode program the contents of X should be "2AAAA", this would be the result of FATN for a usercode program.

**LB XROM 21,28 Load Bytes**

Non-programmable function, prompts for a decimal number that will be insterted in the current program as a byte before the current instruction. For example use LB 144, LB 117 to create a RCL M. LB may disturb the line numbering of the user program, which can be solved by executing PACK.

**NRCL XROM 21,29 Non normalized Recall**

Recalls a register with the register number in X to X as a non-normalized number. If X < 0 the address is an absolute address.

**NSTO XROM 21,30 Non normalized Store**

Stores Y to the register indicated in X, like NRCL.

**K> XROM 21,31 Key suspend**

Prompts for a key, after pressing that key the key assignment to that key is suspended until K< is executed. K> is not programmable.

**K< XROM 21,32 Key reactivate**

Reactivates all key assignments that were suspended with K> or any other operation like 0 STO e. Assignments that are really erased cannot be reactivated.

**?FR XROM 21,33 Number of Free Registers**

Returns the number of free registers available for programming to X.

**ROMSUM XROM 21,34 ROM Checksum calculation**

Calculates the checksum of a ROM page in X "P" and writes it to the last address of that page if it is MLDL ram.

**ROMCHK XROM 21,35 ROM data Check**

Verifies the checksum of a ROM with the XROM number in X. Display shows the ROM Revision code during calculations and shows OK or BAD, depending on the result. If the result is BAD the next line of a running program will be skipped.

**ROM? XROM 21,36 ROM information**

Returns information about a ROM with the XROM number in X. The result is:

X	FAT address of first function
Y	Number of functions
L	XROM number

If the XROM is not found the display will show NO, and the next program line will be skipped if used in a running program.

**RLN XROM 21,37 Rotate Left N bits**

RLN rotates the X register nn bits to the left, where nn is contained in the text string: F1.nn (text string with length 1 and character nn in hex). To rotate X 23 bits to the left use the argument F1.17.

RLN is an Alpha Argument (AA) function and is only useful in a program. The function argument is a synthetic text string in alpha. The AA functions can be used only in a running program. In a program listing, the AA function is immediately followed by a (synthetic) text string containing the argument. Program execution will continue with the instruction after the text string. Note that a check on a valid text string is not done, which could result in unpredictable results if the string is not present.

**J2 XROM 21,38 Jump 2 lines**

Skip two lines in a program

**J3 XROM 21,39 Jump 3 lines**

Skip 3 lines in a program

**J4 XROM 21,40 Jump 4 lines**

Skip 4 lines in a program

**PKA XROM 21,41 Programmable Key Assignments**

AA function (see description of RLN). This function enables programmable key assignments by using the argument F3.pp.oo.kk, where

pp	prefix (see hex table)
oo	postfix (see hex table)
kk	keycode (as found in the key assignment registers, see Wickes))

**X+Y XROM 21,42 Binary Add of X and Y**

Binary addition of X and Y, original X is saved in L.

**Y-X XROM 21,43 Binary Subtract of X and Y**

Binary subtraction of X and Y, original X is saved in L.

**1CMP XROM 21,44 1's Complement of X**

Binary inverse of X, original X is saved in L.

**2CMP XROM 21,45 2's Complement of X**

2's Complement of X, original X is saved in L.

**1D, 2D, 3D, 4D XROM 21,46 to 21,49 1,2,3 or 4 Decode**

Decode of last 1, 2 3 or 4 digits of X, result is appended to alpha.

**TF XROM 21,50 Toggle Flag**

AA function (see description of RLN). Toggles the status of a flag, flag in the argument F1.ff.

**NNN XROM 21,51 Non Normalized Number load**

AA function (see description of RLN). Load X with a NNN from the argument: Fn.HH.HH.HH will result in X: "HHHHHHHH", n must have the correct number of character.

**I>A XROM 21,52 Integer TO Alpha**

Like ARCL X, but only adds the integer part to alpha while ignoring the current FIX setting.

**ASG XROM 21,53 Assign Key**

Works like ASN, but supports assignment of synthetic functions by typing the full function name, for example ASG [ALPHA] RCL M [ALPHA]. Also supports entry like STO 99 and TONE 25. Use an I for indirect functions, for example STO IND e is typed as STO Ie. All status register postfixes are supported, use the + sign for register 10. ASG cannot assign functions with a space in the name.

**MSG XROM 21,54 Message in Display**

AA function (see description of RLN), shows the text of the argument in the display without disturbing the alpha register.

**SUB XROM 21,55 call microcode SUBroutine**

AA function (see description of RLN), calls a microcode program as a subroutine starting at the address indicated in the argument in two possible ways:

F3.AA.AA.00 AAAA is the absolute address of the routine

F3.AA.A0.XX XX is the XROM number, AAA the address within the page (see PDS listing)

The processor registers will be loaded from the stack:

X	->	A
Y	->	B
Z	->	C
T	->	M
L	->	N

**ND XROM 21,56 N digits Decode**

AA function (see description of RLN), like 1D to 4D, but argument contains number of digits to be decoded F1.0n.

**FADR XROM 21,57 Function Address**

Prompts for a function name and puts its startaddress and hex code in alpha. Non-programmable.

**BFCN XROM 21,58 Buffer Function**

AA function (see description of RLN), that operates on a specified buffer. Argument looks like F1.NB, where B is the identification of the buffer, N is the function number:

N=0 puts the absolute register address of the buffer header in X

N=1 puts the buffer length in X

N=2 copies the buffer header to X

N=3 replaces the buffer header by X

N=4 copies the last buffer register to X

N=5 replaces the last buffer register by X

**ML XROM 21,59 -ML ROM function**

Prompts for a function number in the - ML ROM and executes that function. The function executed is calculated as the entered number plus 9, for example ML 00 executes B>D, ML 05 is DECODE. This is done to facilitate entry of often used functions by pressing on of the top row keys. Assignment of ML to the LN key for example executes the DECODE function by pressing LN twice.

**VRG XROM 21,60 View Register**

Prompts for an absolute register address, decodes the contents of it and puts it in alpha and the display. For example VRG 013 shows the C register.

**SRM XROM 21,61 Save ROM on cassette**

Stores a full 4K ROM or MLDL ram on the cassette drive. Name of file is in alpha, page number in X (decimal). The file will be protected. A file with the same name will be overwritten if it is of the same type.

**GROM XROM 21,62 Get ROM from cassette**

Restores a complete 4K ROM image (stored with GROM) from cassette to MLDL memory with the filename in alpha, page number decimal in X.

**^PIOKE XROM 21,63 PEEK and POKE programs**

Usercode program to list and modify usercode programs. At the prompt enter an address in MM-format and the usercode program will be shown byte by byte. R/S will show the next byte, the current byte will be overwritten by entering a hex number.

Some general remarks:

Some functions require other modules:

DAVID Assembler for:	PDS
Extended Functions Module for:	PDS
IL Module + cassette recorder for:	SROM, GROM
IL or Extended Functions for:	K<

The -ML ROM is made by Frits Ferwerda with contributions from Meindert Kuipers. A number of functions have been used from ASSEMBLER 3 and the Eramco Operating System.

Contact [meindert\(at\)kuipers\(dot\)to](mailto:meindert(at)kuipers(dot)to) for further information.