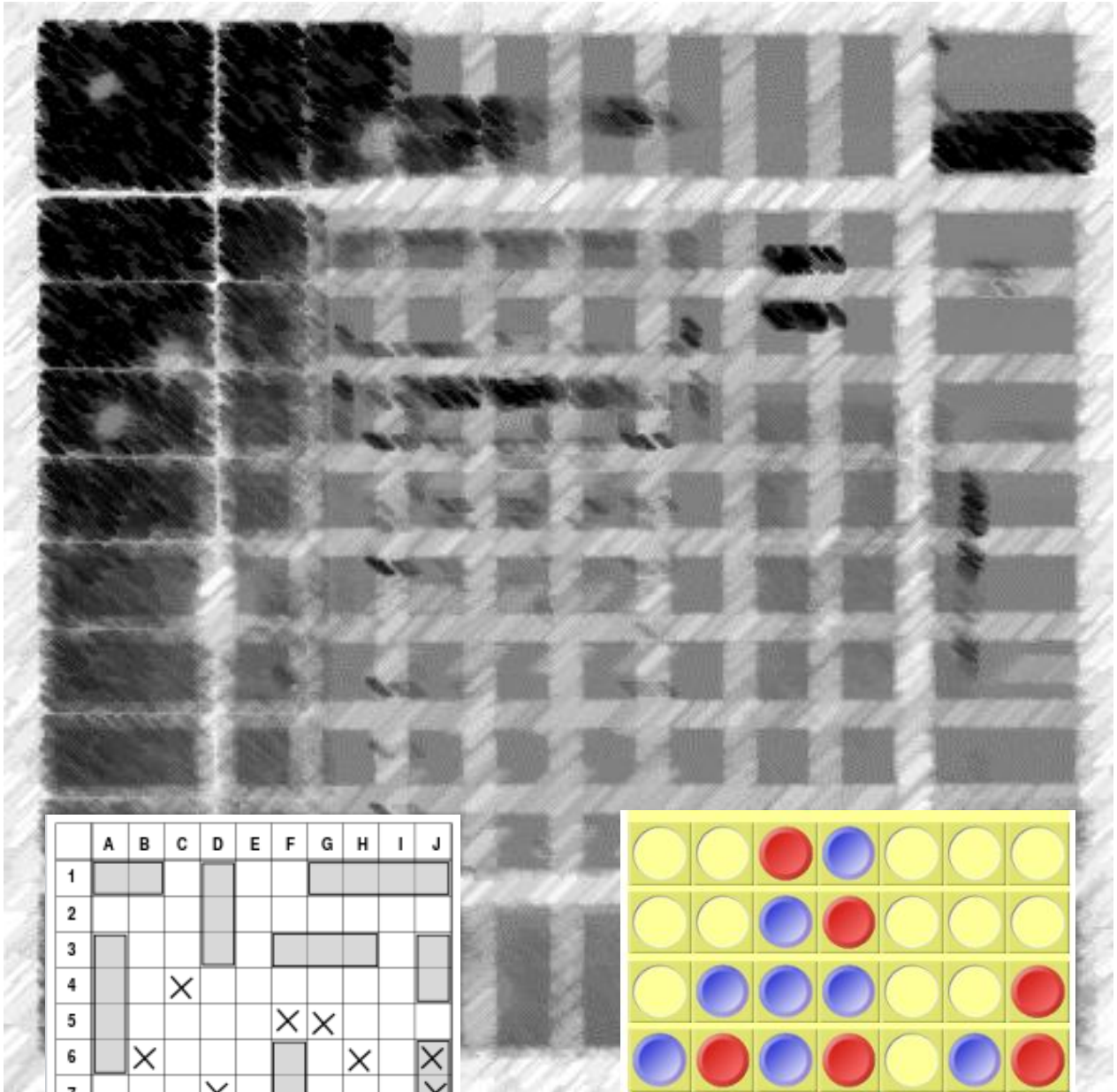
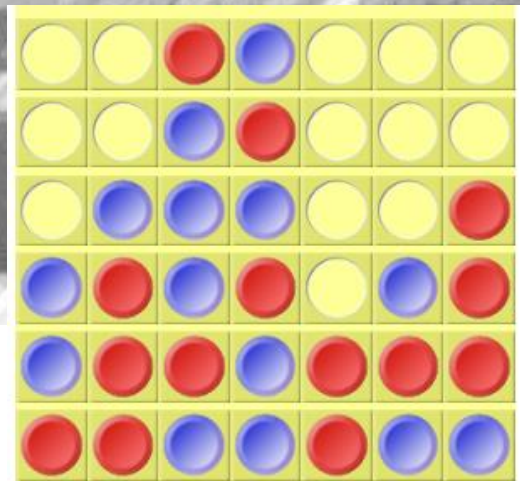


# VINTAGE GAMES VOLUME-1

## COLLECTION FOR THE HP-41



	A	B	C	D	E	F	G	H	I	J
1										
2										
3										
4			X							
5						X	X			
6		X						X		X
7				X						X
8	X	X						X		
9										
10										



Compiled by Ángel M. Martín

May 2019

This compilation revision 1.2.2

Copyright © 2017-19 Ángel Martín

Published under the *GNU* software license agreement.

Original authors retain all copyrights and should be mentioned in writing by any part utilizing this material. No commercial usage of any kind is allowed.

Screen captures taken from V41, Windows-based emulator developed by Warren Furlow.  
See [www.hp41.org](http://www.hp41.org)

## Introduction.

---

This compilation includes a large amount of information on the subject of Games for the HP-41. Most of the games described here (and then some more!) are included in one of the Games Modules available in the CL Library; you can refer to the CL Modules Reference section for a relationship of the specific programs of interest.

The sources of the material included are very diverse: HP Museum, hp41.org Archive, PPC Calculator Journal, Data File Issues, Prisma Magazine, HP User program Library, Swap Disks (that huge heap of mostly undocumented stuff...), and finally different individual contributors' web sites. Collecting all this material hasn't been trivial, and necessarily introduces small inconsistencies in form and structure – but nevertheless the descriptions should be sufficient for a working implementation of the games.

The compilation is divided into three main sections, plus a final CL module reference as detailed below. You can use the hyperlinks to go directly to your area of interest, or use the index in the next pages to access the individual program.

1. [MCODE Games](#)
2. [FOCAL Games](#)
3. Adventure Games
4. CL Modules Reference

With the exception of the Adventure games (a class on their own for obvious reasons), there's no category, subject, topic or other criteria used to structure the compilation (Brain teasers, Casino, Puzzles, Board, Mazes, Simulations, etc). Broadly speaking, the games are structured in a "from more simple to more complex" sequence. This of course is not the same as "from worse to better", as this is largely a personal choice. Some of the most enjoyable games are small in size but require quick reflexes; and conversely some of the more elaborate simulations are a tad too long and the player may become desinterested after a few runs.

The way original authors documented their programs varies immensely, some are really minimalistic in the sketches, whereas others use a more verbose description of the game instructions with prolific program details.

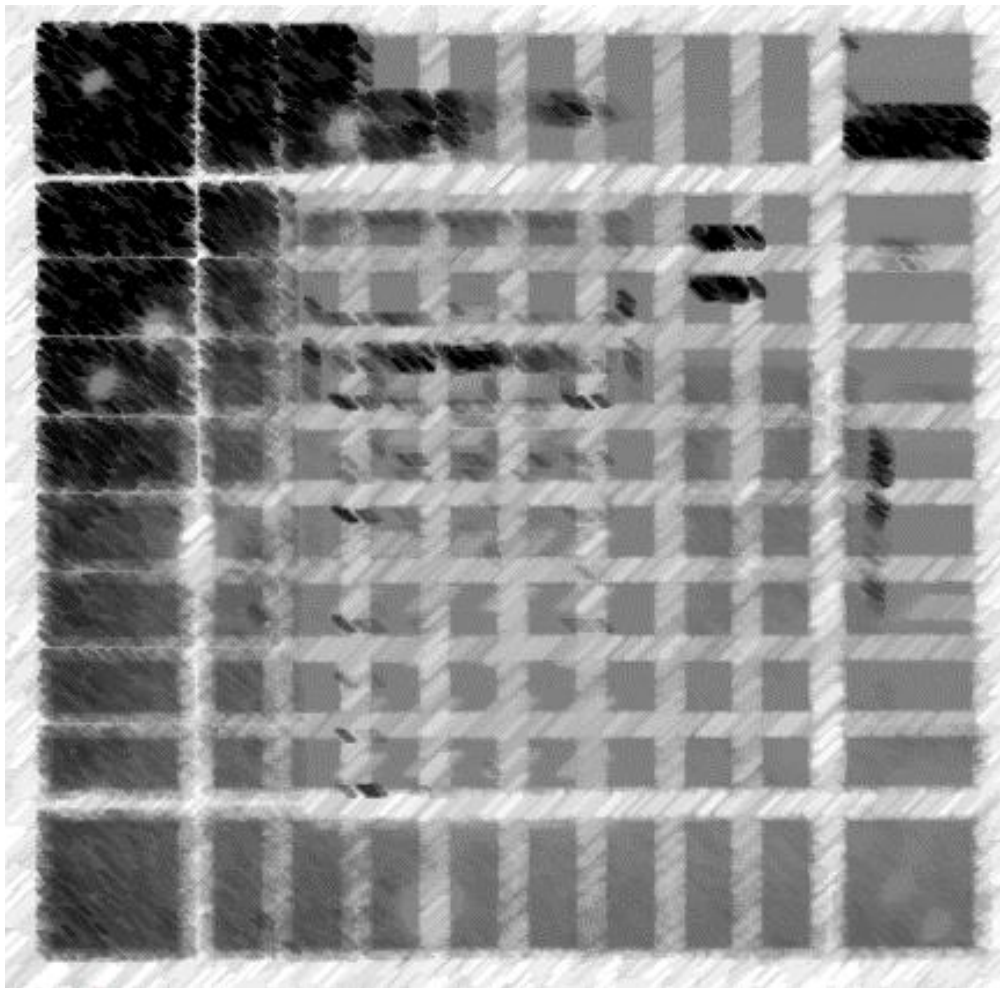
So all in all, here you have a comprehensive representation of the Games on the HP-41 platform that can provide many days of enjoyment to the user. At the very least, it should contribute by providing easy access to a ton of documentation only available using obscure sources difficult to locate.

## Table of Contents (in alphabetical order)

### Part I : MCODE Games

1	<a href="#">Four MCODE Games</a> – G. Vallance; DataFile V11N2, p23 . . . . .	8
#1:	<a href="#">Explosion</a> . . . . .	9
#2:	<a href="#">One to Five</a> . . . . .	14
#3:	<a href="#">FlipX</a> . . . . .	18
#4:	<a href="#">Reverse</a> . . . . .	22
2	<a href="#">High Rollers (MCODE version)</a> – Ross Cooling; PPCCJ . . . . .	60
3	<a href="#">“Le Compte est Bon”</a> – JM Baillard; Web resource . . . . .	40
4	<a href="#">Master Mind (w/ MCODE)</a> - JM Baillard; Web resource . . . . .	56
5	<a href="#">Metronome</a> – Mark Power, DataFile V9N4 p18 . . . . .	33
6	<a href="#">Pattern Twister</a> – Guillaume Tello, Web Source . . . . .	37
7	<a href="#">Poker (w/ MCODE)</a> – JM Baillard; Web resource . . . . .	73
8	<a href="#">Roman Numerals</a> – Martin/Baillard; Web resource . . . . .	99
9	<a href="#">Sudoku Solver</a> – Martin/Baillard; Web resource . . . . .	87
10	<a href="#">Splash Screen</a> – Nelson F. Crowle; NFC ROM . . . . .	104

# Part I - MCODE Games



## Four MCODE Games for the HP 41.

### G. Vallance- DataFile V11N2, p23-30 ; (March 1992)

The idea for these games came from BASIC programs for the TI99/4A home computer (1,2,3,4). These had the advantage of a color TV display but were exasperatingly slow. The MCODE versions are satisfyingly fast and the one-line display adds a further challenge.

The listings given here use the ZENROM mnemonics and occupy FBA7 to FF43. The only address dependent parts are the four FAT entries FCF7, FD70, FE04, and FE94, and the two-word class 1 subroutines calls to **RANL**, **RAN**, **WAITKEY**, **MARK**, **STODISP**, **T ESTEXP**,  **EVAL** and  **DISP**. Three of the programs start with a class 3 jump around a keycode table. At the start of the XROM function (5) the entry address is in C(6:3) thus the address of the keycode table can be stored.

All the programs use a random number generator; the one given here is Ken Emery's (6). You may prefer to use your own or one based on Mark Power's (7). Note the two entry points (**RAN** and **RANL**) depending on whether X is to be saved in L. This allows any game to be repeated (or restarted after pressing the back arrow) by recalling LASTx even if the program calls RAN several times.

All the programs use **WAITKEY**, a general purpose routine for keyboard input, this exists to standby if back arrow is pressed or the battery goes flat; pressing the ON key causes all necessary housekeeping to be done before turning off. Any other keypress returns to the calling program with the keycode in A[1:0].

The first three games use a 3 by 3 board represented on the display as three groups of three separated by colons, with the number of moves on the right. For keyboard input the board corresponds to the 3 by 3 grid formed by the nine digit keys 1 to 9. For example PI FIVE displays 202:334:510 representing:

2 0 2	corresponding to keys	7 8 9
3 3 4		4 5 6
5 1 0		1 2 3

#### References.

- (1). Hal. Renko and Sam Edwards. Tantalizing Games for your TI99/4A. Addison-Wesley 1983. Explosion, pp. 70-76
- (2). Ibid. One to Five, pp 46-49
- (3). Scott Vincent. Dynamic Games for your TI99/4A. Interface 1983. Flippa, pp. 105-108
- (4). Ibid. Number Juggle, pp. 144-146
- (5). Ken Emery. MCode for beginners. Synhetix 1985. What's up on entry p.132
- (6). Ibid. "RN", pp. 100-102
- (7). Mark Power. Mcode Random Numbers. Datafile V6N8p9
- (8). Ken Emery. Op. cit. XOR, pp. 80-82
- (9). Richard Kendon. Reverse the order of digits of an integer. DataFile V10N4p24



## **EXPLOSION.**(Reference [#1])

---

Two players place counters on empty squares or their own counters on a 3 by 3 grid. If a pile of counters becomes critical it explodes and takes over neighboring (not diagonal) squares, which may in turn explode. The critical capacity is the number of non-diagonal adjacent squares i.e. 2 for corners, 3 for the centers of the sides, and 4 for the center. The winner is the one starting a self-sustaining chain reaction.

At the prompt "C/M" press C for a game against the calculator with the calculator having the first go; M to play against the calculator with me making the first move; or press any other key (except ON or back arrow) for a game between two people. It is possible to reach stalemate where neither side can move. For example, 13521343435 displays AAA:BBA:AAA: 5- press back arrow to escape.

The calculator takes about half a second to evaluate a move at the beginning of a game, compared to 18 sec on the TI99. Slowest evaluation later in the game is about 2 sec. The calculator only evaluates the next move and it is not difficult to defeat it. Incidentally, the original BASIC program (1) has an undocumented feature (not in this MCODE version) which sometimes causes the evaluation procedure to go haywire, so that a simple scheme to defeat it did not always work.

Flag	Clear	Set
F0	first player	calculator or second player
F1	no explosion	explosion
F2	<16 explosions	>15 explosions (self-sustaining)
F3	actual move	evaluating (don't store or display)
F4	2 players	against calculator (needs to evaluate)

### Data Storage:

M[12:4]critical capacity;	M[2:0] number of turns	
N[13] pointer;	N[12:4] board;	N[2:0] evaluation
Q[13] explosions;	Q[6:3] table address;	Q[2:0] random

The status of the board is stored as the nine digits N[12:4]. Each position is stored as a single digit and is displayed as a single character. The positions of the first player are stored as positive digits and displayed as A, B, etc; moves by the calculator or second player are stored as negative digits (F, E, etc.) and displayed as a, b, etc.

The program starts with a jump to FD02 where the keycode table address is stored in Q[6:3]. FD03 to FD11 gets a random seed, multiplies it by 1,000 converts to binary (max 3E7) shifts two bits left, adds one for luck and stores it in Q[2:0]. This provides 12 more or less random bits for the EVAL subroutine. FD12-E initializes the board and stores the capacity of each position. FD1F-29 prompts for the type of game required. FD2A-35 tests whether C or M was pressed and sets or clears flags 0 and 4. FD36-8 displays the empty board at the start of the game.

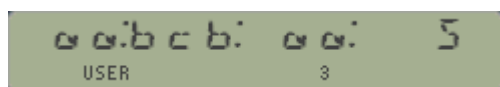
The main program loop starts at FD39. FD39-43 uses the keycode table to test for valid keys and at the same time sets the pointer to the corresponding digit. FD44-50 tests that the position is occupied by the player's counter(s) or is empty and either rejects the move or adds or subtracts one to the board in A[12:4]. FD51-6 adds one to the number of moves on alternate moves. The rest of the main program stores the move, tests for explosions, tests for end of game (flag 2 set), changes player by toggling flag 0, evaluates if necessary, and loops back for the next move.

The **STODISP** subroutine takes the new board from A[12:4] and stores it in N[12:4], displays the new board and number of moves, and beeps.

The **TESTEXP** subroutine tests whether the board in A[12:4] has a critical position and determines the effect of subsequent explosions. FC4C-62 tests whether A[PT] is critical, and if so subtracts the critical value. FC65-76 adds one to the adjacent positions and takes them over by giving them the current player's sign. These positions may have become critical, so if flag 1 is set the program loops back to test again. FC7A-E stores the number of explosions in Q[13] and sets flag 2 if there are more than 15 – this seems to be a sufficient criterion for a continuous chain reaction.

The **MARK** subroutine loads a 9 bit number and expands it into 9 digits in C[12:4] with ones at the potential adjacent to the current pointer. For example PT=11 corresponds to key 8; the non-diagonal adjacent keys are 7, 9, and 5, indicated by C[12:4] = 101010000. A loop counter, 8, is left in C[13] for use by the calling program. These loops, FC65-76 and FCC0-D0, change the pointer, but exit with the pointer unchanged because they are controlled by the counter.

The **EVAL** subroutine evaluates each possible move as a number (biased by 100h to take care of negative numbers) in A[2:0] and stores the best (lowest) in N[2:0]. FC94-8 subtracts one for the center square if it's not already taken (not in original program); FCA4-B3 adds the value of each position; FCB7-D subtracts 2 for each of the calculator's subcritical positions; but FCBE-D0 adds 10 if an adjoining opponent's position is also subcritical. If the evaluation is equal to the value already stored in N[2:0] the random number in Q[2:0] is used to choose between them. FCAB tests the leftmost bit and if it is set it's replaced at the right at FCDF so the 12 bits are rotated and can be used indefinitely. FCE3-8 converts the current pointer (i.e. move being stored) to a constant and stores it in N[13]. EVAL exits with PT=P set to the optimum value. EVAL uses PT=Q to indicate the move being evaluated; TEXTEXP uses PT=P to evaluate each move but STODSP, which also uses PT=Q, is not called during evaluation.





Header AA58	085	"E"	
Header AA59	004	"D"	Explosion
Header AA5A	00F	"O"	
Header AA5B	00C	"L"	
Header AA5C	010	"P"	DataFile V11N2p23
Header AA5D	018	"X"	G. Vallance
<b>XPLODE</b>	<b>05B</b>	<b>JNC +11d</b>	
AA5F	086	"3"	
AA60	076	"2"	
AA61	036	"1"	
AA62	085	"6"	
AA63	075	"5"	keycodes table
AA64	035	"4"	
AA65	084	"9"	
AA66	074	"8"	
AA67	034	"7"	
AA68	000		end of table
LB_AA67	268	WRIT 9(Q)	
AA6A	379	PORT DEP:	Random number
AA6B	03C	XQ	sets DEC mode
AA6C	104	->A904	[RANL]
AA6D	226	C=C+1 S&X	
AA6E	226	C=C+1 S&X	
AA6F	226	C=C+1 S&X	
AA70	260	SETHX	
AA71	38D	?NC XQ	
AA72	008	->02E3	[BCDBIN]
AA73	1E6	C=C+C S&X	
AA74	1E6	C=C+C S&X	
AA75	226	C=C+1 S&X	
AA76	106	A=C S&X	
AA77	278	READ 9(Q)	
AA78	0A6	A<>C S&X	
AA79	268	WRIT 9(Q)	
AA7A	04E	C=0 ALL	
AA7B	070	N=C ALL	
AA7C	35C	PT= 12	
AA7D	090	LD@PT- 2	
AA7E	0D0	LD@PT- 3	
AA7F	090	LD@PT- 2	2,32343232
AA80	0D0	LD@PT- 3	
AA81	110	LD@PT- 4	
AA82	0D0	LD@PT- 3	
AA83	090	LD@PT- 2	
AA84	0D0	LD@PT- 3	
AA85	090	LD@PT- 2	
AA86	158	M=C ALL	
AA87	3C1	?NC XQ	Enable & Clear LCD
AA88	0B0	->2CF0	[CLLCDE]
AA89	3BD	?NC XQ	
AA8A	01C	->07EF [	MESSL]
AA8B	003	"C"	

AA8C	02F	"/"	"C/M"
AA8D	20D	"M"	
AA8E	149	?NC XQ	
AA8F	024	->0952	[ENCP00]
AA90	379	PORT DEP:	
AA91	03C	XQ	
AA92	124	->A924	[WAITKY]
AA93	048	SETF 4	
AA94	388	SETF 0	
AA95	130	LDI S&X	070
AA96	070	CON:	
AA97	366	?A#C S&X	
AA98	1D3	JNC +58d	[LB_AAD0]
AA99	384	CLRF 0	
AA9A	130	LDI S&X	082
AA9B	082	CON:	
AA9C	366	?A#C S&X	
AA9D	013	JNC +02	[LB_AA9D]
AA9E	044	CLRF 4	
LB_AA9DF	1A0	A=B=C=0	
AAA0	379	PORT DEP:	
AAA1	03C	XQ	
AAA2	16D	->A96D	[STODSP]
LB_AAA1	379	PORT DEP:	
AAA4	03C	XQ	
AAA5	124	->A924	[WAITKY]
AAA6	01C	PT= 3	
AAA7	278	READ 9(Q)	
LB_AAA6	23A	C=C+1 M	
AAA9	330	FETCH S&X	
AAAA	2E6	?C#0 S&X	
AAAB	3C3	JNC -08	[LB_AAA1]
AAAC	3DC	PT=PT+1	
AAAD	366	?A#C S&X	
AAAE	3D7	JC -06	[LB_AAA6]
LB_AAAD	0B0	C=N ALL	
AAB0	10E	A=C ALL	
AAB1	38C	?FSET 0	
AAB2	02F	JC +05	[LB_AAB5]
AAB3	1E2	C=C+C @PT	
AAB4	37F	JC -17d	[LB_AAA1]
AAB5	162	A=A+1 @PT	
AAB6	033	JNC +06	[LB_AABA]
LB_AAB5	2E2	?C#0 @PT	
AAB8	01B	JNC +03	[LB_AAB9]
AAB9	1E2	C=C+C @PT	
AABA	34B	JNC -23d	[LB_AAA1]
LB_AAB9	1A2	A=A-1 @PT	
LB_AABA	198	C=M ALL	
AABD	2F6	?C#0 XS	
AABE	013	JNC +02	[LB_AABE]
AABF	226	C=C+1 S&X	

LB_AABE	2B6	C=-C-1 XS	
AAC1	158	M=C ALL	
AAC2	379	PORT DEP:	
AAC3	03C	XQ	
AAC4	16D	->A96D	[STODSP]
AAC5	004	CLRF 3	
AAC6	379	PORT DEP:	
AAC7	03C	XQ	
AAC8	1A9	->A9A9	[TSTEXP]
AAC9	20C	?FSET 2	
AACA	067	JC +12d	[LB_AAD4]
AACB	38C	?FSET 0	
AACC	01B	JNC +03	[LB_AACD]
AACD	384	CLRF 0	
AACE	2AB	JNC -43d	[LB_AAA1]
LB_AACD	388	SETF 0	
AAD0	04C	?FSET 4	
AAD1	293	JNC -46d	[LB_AAA1]
LB_AAD0	379	PORT DEP:	
AAD3	03C	XQ	
AAD4	1E7	->A9E7	[EVAL]
AAD5	2D3	JNC -38d	[LB_AAAD]
LB_AAD4	2ED	?NC XQ	
AAD7	040	->10BB	[BEEP]
AAD8	1F9	?NC GO	
AAD9	00E	->037E	[STMSGF]

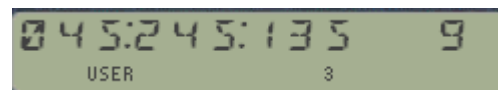
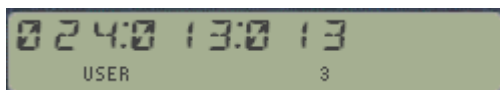
## ONE to FIVE. (Reference [#2])

Pressing one of the digit keys adds one to each of the numbers in the same row and same column, but adding one to five gives zero. The puzzle is solved by reducing all the numbers to zero. The method of solving is similar to that used with Rubik's Cube, in that one develops algorithms for changing one set of numbers while leaving the relations between some others unchanged.

The nine digits on the board are stored in N[13:5], the number of goes in N[2:0]. This program does not use a keycode table, instead FD92-A2 tests the keycode to see if it is in the range of keycodes for the digit keys.

The **DISP** subroutine requires the keycode for one of the digit keys in A[1:0], 034 in C[2:0], and PT=1. FDB3-D generates the number 100100100 and may shift it right 1 or 2, depending on which column the keycode belongs to. FDBF-C9 makes 111000000 and shifts it by 3 or 6 depending on the row. These two numbers are combined with OR and added to the numbers on the board. FDCF-D8 changes sixes to zero and stores the new board. The rest of DISP displays the new board and beeps.

A random distribution of numbers cannot usually be solved – the game is started by making ten random moves on an empty board. FD76-B puts a random digit in the range 0 to 8 in A[S]; the JC -1 at FD7B takes care of the rare case when **RAN** returns a zero. The digit in A[S] is divided by 3 and the quotient and remainder are used to make a random keycode which is passed to the **DISP** subroutine.



Header AADB	085	"E"	
Header AADC	016	"V"	
Header AADD	009	"I"	One to Five
Header AADE	006	"F"	
Header AADF	00F	"O"	DataFile V11N2p23
Header AAEO	014	"T"	G. Vallance
<b>TOFIVE AAE1</b>	<b>0F8</b>	<b>READ 3(X)</b>	
AAE2	128	WRIT 4(L)	
AAE3	04E	C=0 ALL	
AAE4	130	LDI S&X	009
AAE5	009	CON:	
LB_AAE4	070	N=C ALL	
AAE7	000		
AAE8	375	PORT DEP:	Random number
AAE9	03C	XQ	sets DEC mode
AAEA	106	->A906	[RAN]
LB_AAE8	2FC	RCR 13	
AAEC	11E	A=C MS	

LB_AAEA	1BE	A=A-1 MS	
AAEE	3FF	JC -01	[LB_AAEA]
AAEF	04E	C=0 ALL	
AAF0	130	LDI S&X343	
AAF1	343	CON:	
AAF2	33C	RCR 1	
AAF3	106	A=C S&X	
AAF4	31C	PT= 1	
LB_AAF2	31E	?A<C MS	
AAF6	027	JC +04	[LB_AAF7]
AAF7	1DE	A=A-C MS	
AAF8	162	A=A+1 @PT	
AAF9	3E3	JNC -04	[LB_AAF2]
LB_AAF7	362	?A#C @PT	
AAFB	013	JNC +02	[LB_AAFA]
AAFC	142	A=A+C @PT	
LB_AAFA	1BE	A=A-1 MS	
AAFE	01F	JC +03	[LB_AAFE]
AAFF	166	A=A+1 S&X	
AB00	3EB	JNC -03	[LB_AAFA]
LB_AAFE	158	M=C ALL	
AB02	375	PORT DEP:	
AB03	03C	XQ	
AB04	328	->AB28	[DISP]
AB05	266	C=C-1 S&X	
AB06	303	JNC -32d	[LB_AAE4]
LB_AB04	375	PORT DEP:	
AB08	03C	XQ	
AB09	124	->A924	[WAITKY]
AB0A	130	LDI S&X	097
AB0B	097	CON:	
AB0C	31C	PT= 1	
AB0D	302	?A<C @PT	
AB0E	2EB	JNC -35d	[LB_AAE8]
AB0F	39C	PT= 0	
AB10	302	?A<C @PT	
AB11	3B3	JNC -10d	[LB_AB04]
AB12	130	LDI S&X	034
AB13	034	CON:	
AB14	302	?A<C @PT	
AB15	397	JC -14d	[LB_AB04]
AB16	31C	PT= 1	
AB17	302	?A<C @PT	
AB18	37F	JC -17d	[LB_AB04]
AB19	0F0	C<>N ALL	
AB1A	226	C=C+1 S&X	
AB1B	289	?C GO	
AB1C	003	->00A2	[ERROF]
AB1D	0F0	C<>N ALL	
AB1E	158	M=C ALL	
AB1F	375	PORT DEP:	
AB20	03C	XQ	

	AB21	328	->AB28	[DISP]
	AB22	2FA	?C#0 M	
	AB23	327	JC -28d	[LB_AB04]
	AB24	2ED	?NC XQ	
	AB25	040	->10BB	[BEEP]
	AB26	1F9	?NC GO	
	AB27	00E	->037E	[STMSGF]
DISP	AB28	198	C=M ALL	
	AB29	1C6	A=A-C S&X	
	AB2A	222	C=C+1 @PT	
	AB2B	0E2	B<>C @PT	
	AB2C	04E	C=0 ALL	
	AB2D	226	C=C+1 S&X	
	AB2E	23A	C=C+1 M	
	AB2F	2BC	RCR 7	
	AB30	23E	C=C+1 MS	
	AB31	182	A=A-B @PT	
	AB32	02F	JC +05	[LB_AB34]
	AB33	33C	RCR 1	
	AB34	1A2	A=A-1 @PT	
	AB35	017	JC +02	[LB_AB34]
	AB36	33C	RCR 1	
LB_AB34		0EE	B<>C ALL	
	AB38	04E	C=0 ALL	
	AB39	130	LDI S&X	111
	AB3A	111	CON:	
	AB3B	03C	RCR 3	
	AB3C	39C	PT= 0	
	AB3D	1A2	A=A-1 @PT	
	AB3E	02F	JC +05	[LB_AB40]
	AB3F	03C	RCR 3	
	AB40	1A2	A=A-1 @PT	
	AB41	017	JC +02	[LB_AB40]
	AB42	03C	RCR 3	
LB_AB43		06E	A<>B ALL	
	AB44	370	C=C OR A	
	AB45	10E	A=C ALL	
	AB46	0B0	C=N ALL	
	AB47	14E	A=A+C ALL	
	AB48	190	LD@PT- 6	
LB_AB46		33C	RCR 1	
	AB4A	362	?A#C @PT	
	AB4B	017	JC +02 [	LB_AB4A]
	AB4C	002	A=0 @PT	
LB_AB4A		3D4	PT=PT-1	
	AB4E	054	?PT= 4	
	AB4F	3D3	JNC -06	
	AB50	0AE	A<>C ALL	
	AB51	070	N=C ALL	
	AB52	3D9	?NC XQ	
	AB53	01C	->07F6	[ENLCD]
LB_AB51		130	LDI S&X	003



	AB55	003	CON:	
	AB56	154	?PT= 6	
	AB57	01F	JC +03	[LB_AB57]
	AB58	254	?PT= 9	
	AB59	01B	JNC +03	
LB_AB57		130	LDI S&X	00B
	AB5B	00B	CON:	
LB_AB59		2FC	RCR 13	
	AB5D	3E8	WRIT 15(e)	
	AB5E	3DC	PT=PT+1	
	AB5F	2D4	?PT= 13	
	AB60	3A3	JNC -12d	[LB_AB51]
	AB61	0B0	C=N ALL	
	AB62	03C	RCR 3	
	AB63	130	LDI S&X	002
	AB64	002	CON:	
	AB65	358	ST=C XP	
LB_AB63		2FE	?C#0 MS	
	AB67	013	JNC +02	[LB_AB66]
	AB68	388	SETF 0	
LB_AB66		398	C=ST XP	
	AB6A	2FC	RCR 13	
	AB6B	3E8	WRIT 15(e)	
	AB6C	3DC	PT=PT+1	
	AB6D	214	?PT= 2	
	AB6E	3C3	JNC -08	[LB_AB63]
	AB6F	149	?NC XQ	
	AB70	024	->0952	[ENCP00]
	AB71	130	LDI S&X	059
	AB72	059	CON:	
	AB73	358	ST=C XP	
	AB74	379	?NC XQ	
	AB75	058	->16DE	[XTONE]
	AB76	2A0	SETDEC	
	AB77	0B0	C=N ALL	
	AB78	3E0	RTN	

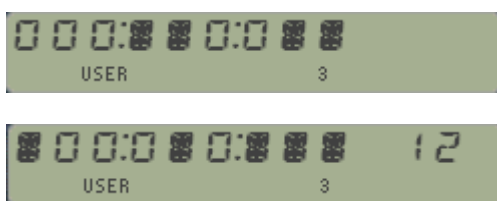
## FLIPX(Reference [#3])

There are 9 counters on a 3 by 3 grid. They are white on one side and black on the other. You can turn over a black counter but not a white one. However, if you turn over a black counter some of its neighbors are flipped over as well, whatever their color. Flipping a corner also flips the 3 adjoining counters. Flipping the centers of the sides also flips the two nearby corners. Flipping the center also flips the centers of the four sides. The object of the game is to make the center white and all the others black. If you make all the counters white, you are stuck and must press back arrow to exit.

The white or black status of the 9 counters is stored as a 9-bit number in M[X], and the bits are tested and toggled using the Boolean operators. The toggle mask for each move is stored together with the corresponding keycode in the table at the beginning of the program. The table address is stored in N[6:3] and the number of moves is in N[2:0]

The start of the program generates a random nine bit number and drops into the main program loop at FE30. The display routine FE30-47 tests the ninth bit in A[2:0] (the 3 higher bits are zero) and displays it as letter O or boxed star depending on whether it is clear or set, corresponding to white or black. A colon is added to every third one by setting flag 7 when the character code is in the flag register. FE48-58 displays the number of moves with leading spaces.

The keycode test loop is FE69-79; both the toggle mask and keycode are read from ROM, so the loop contains two FETCH S&X instructions. At the same time a test mask is generated in A.M, FE81-6 gets the board from M to A.X and brings the test mask to C.X and tests it with CAND A to see if the keypress corresponds to a black or white counter. FE87-D uses OR, AND, NOT AND to make the XOR operator (8) to flip the appropriate counters. It then loops back to the display routine.



Header AC35	098	"X"	
Header AC36	010	"P"	Flip characters
Header AC37	009	"I"	
Header AC38	00C	"L"	DataFile V11N2p23
Header AC39	006	"F"	G. Vallance
<b>FLIPX</b>	<b>AC3A</b>	<b>0A3</b>	<b>JNC +20d</b>
AC3B	01B		
AC3C	086	"3"	
AC3D	007		
AC3E	076	"2"	
AC3F	036		
AC40	036	"1"	
AC41	049		
AC42	085	"6"	
AC43	0BA		keycodes table
AC44	075	"5"	
AC45	124		
AC46	035	"4"	
AC47	0DB		
AC48	084	"9"	
AC49	1C0		
AC4A	074	"8"	
AC4B	1B0		
AC4C	034	"7"	
AC4D	000		end of table
LB_AC4B	046	C=0 S&X	
AC4F	070	N=C ALL	
LB_AC4D	391	PORT DEP:	Random number
AC51	08C	XQ	sets DEC mode
AC52	104	->A904	[RANL]
AC53	10E	A=C ALL	
AC54	04E	C=0 ALL	
AC55	39C	PT= 0	
AC56	090	LD@PT- 2	
AC57	010	LD@PT- 0	C = 511
AC58	150	LD@PT- 5	
AC59	050	LD@PT- 1	
AC5A	050	LD@PT- 1	
AC5B	135	?NC XQ	
AC5C	060	->184D [	MP2-10]
AC5D	260	SETHX	
AC5E	38D	?NC XQ	
AC5F	008	->02E3	[BCDBIN]
AC60	226	C=C+1 S&X	
AC61	158	M=C ALL	
AC62	106	A=C S&X	
AC63	130	LDI S&X	1EF
AC64	1EF	CON:	
AC65	366	?A#C S&X	
AC66	353	JNC -22	[LB_AC4D]
LB_AC64	3D9	?NC XQ	
AC68	01C	->07F6	[ENLCD]

	AC69	39C	PT= 0	
LB_AC67		130	LDI S&X	00F
	AC6B	00F	CON:	"O"
	AC6C	356	?A#0 XS	
	AC6D	01B	JNC +03	[LB_AC6D]
	AC6E	130	LDI S&X	03A
	AC6F	03A	"*"	boxed star
LB_AC6D		214	?PT= 2	
	AC71	01F	JC +03	[LB_AC71]
	AC72	094	?PT= 5	
	AC73	023	JNC +04	[LB_AC74]
LB_AC71		3D8	C<>ST XP	
	AC75	288	SETF 7	
	AC76	3D8	C<>ST XP	
LB_AC747		3E8	WRIT 15(e)	
	AC78	114	?PT= 8	
	AC79	037	JC +06	[LB_AC7C]
	AC7A	3DC	PT=PT+1	
	AC7B	016	A=0 XS	
	AC7C	086	B=A S&X	
	AC7D	126	A=A+B S&X	
	AC7E	363	JNC -20d	[LB_AC67]
LB_AC7C		0B0	C=N ALL	
	AC80	03C	RCR 3	
	AC81	130	LDI S&X	002
	AC82	002	CON:	
LB_AC80		2FE	?C#0 MS	
	AC84	023	JNC +04	[LB_AC85]
	AC85	3D8	C<>ST XP	
	AC86	388	SETF 0	
	AC87	3D8	C<>ST XP	
LB_AC85		2FC	RCR 13	
	AC89	3E8	WRIT 15(e)	
	AC8A	3C6	RSHFC S&X	
	AC8B	3DC	PT=PT+1	
	AC8C	194	?PT= 11	
	AC8D	3B3	JNC -10d	[LB_AC80]
	AC8E	149	?NC XQ	
	AC8F	024	->0952	[ENCP00]
	AC90	130	LDI S&X	059
	AC91	059	CON:	
	AC92	358	ST=C XP	
	AC93	379	?NC XQ	
	AC94	058	->16DE	[XTONE]
	AC95	198	C=M ALL	
	AC96	106	A=C S&X	
	AC97	130	LDI S&X	1EF
	AC98	1EF	CON:	
	AC99	366	?A#C S&X	
	AC9A	037	JC +06	[LB_AC9D]
	AC9B	2ED	?NC XQ	
	AC9C	040	->10BB	[BEEP]

	AC9D	1F9	?NC GO	
	AC9E	00E	->037E	[STMSGF]
	AC9F	243	JNC -56d	[LB_AC64]
LB_AC9D		391	PORT DEP:	
	ACA1	08C	XQ	
	ACA2	124	->A924	[WAITKY]
	ACA3	01A	A=0 M	
	ACA4	17A	A=A+1 M	
	ACA5	0B0	C=N ALL	
LB_ACA3		23A	C=C+1 M	
	ACA7	330	FETCH S&X	
	ACA8	2E6	?C#0 S&X	
	ACA9	3BB	JNC -09	[LB_AC9D]
	ACAA	0E6	B<>C S&X	
	ACAB	23A	C=C+1 M	
	ACAC	330	FETCH S&X	
	ACAD	366	?A#C S&X	
	ACAE	023	JNC +04	[LB_ACAF]
	ACAF	09A	B=A M	
	ACB0	13A	A=A+B M	
	ACB1	3AB	JNC -11d	[LB_ACA3]
LB_ACAF		0B0	C=N ALL	
	ACB3	2A0	SETDEC	
	ACB4	226	C=C+1 S&X	
	ACB5	289	?C GO	
	ACB6	003	->00A2	[ERROF]
	ACB7	260	SETHX	
	ACB8	070	N=C ALL	
	ACB9	198	C=M ALL	
	ACBA	0AE	A<>C ALL	
	ACBB	03C	RCR 3	
	ACBC	3B0	C=C AND A	
	ACBD	2E6	?C#0 S&X	
	ACBE	30B	JNC -31d	[LB_AC9C] -> [AC64]
	ACBF	0C6	C=B S&X	
	ACC0	370	C=C OR A	
	ACC1	0E6	B<>C S&X	
	ACC2	3B0	C=C AND A	
	ACC3	2A6	C=-C-1 S&X	
	ACC4	066	A<>B S&X	
	ACC5	3B0	C=C AND A	
	ACC6	158	M=C ALL	
	ACC7	106	A=C S&X	
	ACC8	2BB	JNC -41d	[LB_AC9C] -> [AC64]

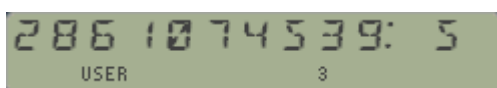
## REVERSE. (Reference [#4])

This is a one-line game which starts by displaying the ten digits 0 to 9 in random order. Pressing a digit key N reverses the display between the Nth. and last digits. The game is solved by getting the digits in order from 0 to 9. It is not difficult to solve; the challenge is to do it in the smallest number of moves.

This is a rather long program for such a simple game, most of it taken up with juggling the display during setup. The type of random selector used, in which selected items are excluded from further selections may be used in other games.

FEA2-E stores the solution in Reg "a" for later testing and also in N for the random selector. This works by first selecting one of the ten digits, then selecting one of the nine left, and so on. FEB3-C6 multiplies the random seed by ten by adding one to the exponent on the first time round but on subsequent rounds does a floating point multiplication by 9, 8, etc. The random digit is left in C[5], FEC7-D0 sets the pointer to the random digit, puts the digit at the pointer in G and closes the gap. FED1-2 decrements the multiplier, and stores it and the shortened source string in N. FED3-B right justifies the display. FEDC-E6 adds the newly selected character to the display, left justifies it and loops back if there is more than one digit left. The new random string is not stored in a register, only in the display, so FEE7-F6 extracts the string from the display, adds the last digit from N and drops into the main program loop.

FEF7-FF26 does the display, beeps and tests for end of game. FF27-34 tests the keycode and at the same time sets the pointer to the relevant digit and creates a counter in A[S]. FF35-A increments the number of goes, maximum 99. FF3B-43 uses Richard Kendon's idea (9) to reverse the number to the right of the pointer and loops back to display the new result.



Header AB7A	085	"E"	
Header AB7B	013	"S"	
Header AB7C	012	"R"	Reverse Digits
Header AB7D	005	"E"	
Header AB7E	016	"V"	
Header AB7F	005	"E"	DataFile V11N2p23
Header AB80	012	"R"	G. Vallance
<b>REVERSE</b>	<b>05B</b>	<b>JNC +11d</b>	
AB82	084	"9"	
AB83	074	"8"	
AB84	034	"7"	
AB85	085	"6"	
AB86	075	"5"	keycodes table
AB87	035	"4"	



	AB88	086	"3"	
	AB89	076	"2"	
	AB8A	036	"1"	
	AB8B	000		end of table
LB_AB89		268	WRIT 9(Q)	
	AB8D	0F8	READ 3(X)	
	AB8E	128	WRIT 4(L)	
	AB8F	04E	C=0 ALL	
	AB90	19C	PT= 11	
	AB91	050	LD@PT- 1	
	AB92	090	LD@PT- 2	
	AB93	0D0	LD@PT- 3	
	AB94	110	LD@PT- 4	C = 0123456789
	AB95	150	LD@PT- 5	
	AB96	190	LD@PT- 6	
	AB97	1D0	LD@PT- 7	
	AB98	210	LD@PT- 8	
	AB99	250	LD@PT- 9	
	AB9A	2E8	WRIT 11(a)	
	AB9B	070	N=C ALL	
	AB9C	3C1	?NC XQ	
	AB9D	0B0	->2CF0	[CLLCDE]
LB_AB9B		260	SETHX	
	AB9F	149	?NC XQ	
	ABA0	024	->0952	[ENCP00]
	ABA1	379	PORT DEP:	Random number
	ABA2	03C	XQ	sets DEC mode
	ABA3	106	->A906	[RAN]
	ABA4	10E	A=C ALL	
	ABA5	0B0	C=N ALL	
	ABA6	2E6	?C#0 S&X	
	ABA7	03F	JC +07	[LB_ABAB]
	ABA8	166	A=A+1 S&X	
	ABA9	130	LDI S&X	010
	ABAA	010	<decimal>	
	ABAB	070	N=C ALL	
	ABAC	0AE	A<>C ALL	
	ABAD	02B	JNC +05	[LB_ABAF]
LB_ABAB		05A	C=0 M	
	ABAF	23C	RCR 2	
	ABB0	135	?NC XQ	
	ABB1	060	->184D	[MP2-10]
LB_ABAF		2E6	?C#0 S&X	
	ABB3	013	JNC +02	[LB_ABB2]
	ABB4	04E	C=0 ALL	
LB_ABB2		2FC	RCR 13	
	ABB6	2DC	PT= 13	
LB_ABB4		3D4	PT=PT-1	
	ABB8	27E	C=C-1 MS	
	ABB9	3F3	JNC -02	[LB_ABB4]
	ABBA	0B0	C=N ALL	
	ABBB	058	G=C @PT,+	

	ABBC	11A	A=C M	
	ABBD	006	A=0 S&X	
	ABBE	3EA	LSHFA R<	
	ABBF	0BA	A<>C M	
	ABC0	266	C=C-1 S&X	
	ABC1	070	N=C ALL	
	ABC2	3D9	?NC XQ	
	ABC3	01C	->07F6	[ENLCD]
	ABC4	130	LDI S&X	020
	ABC5	020	" "	space char
	ABC6	106	A=C S&X	
LB_ABC4		3F8	READ 15(e)	
	ABC8	366	?A#C S&X	
	ABC9	3F7	JC -02	[LB_ABC4]
	ABCA	3B8	READ 14(d)	
	ABCB	39C	PT= 0	
	ABCC	098	C=G @PT,+	
	ABCD	31C	PT= 1	
	ABCE	0D0	LD@PT- 3	
	ABCF	3E8	WRIT 15(e)	
	ABD0	3DD	?NC XQ	
	ABD1	0AC	->2BF7	[LEFTJ]
	ABD2	0B0	C=N ALL	
	ABD3	266	C=C-1 S&X	
	ABD4	2E6	?C#0 S&X	
	ABD5	24F	JC -55d	[LB_AB9B]
	ABD6	2DC	PT= 13	
	ABD7	210	LD@PT- 8	
	ABD8	10E	A=C ALL	
LB_ABD6		3F8	READ 15(e)	
	ABDA	0BE	A<>C MS	
	ABDB	3EE	LSHFA ALL	
	ABDC	39C	PT= 0	
	ABDD	102	A=C @PT	
	ABDE	11E	A=C MS	
	ABDF	1BE	A=A-1 MS	
	ABE0	3CB	JNC -07	[LB_ABD6]
	ABE1	0B0	C=N ALL	
	ABE2	2FC	RCR 13	
	ABE3	11E	A=C MS	
	ABE4	0AE	A<>C ALL	
	ABE5	0FC	RCR 10	
LB_ABE3		070	N=C ALL	
	ABE7	3D9	?NC XQ	
	ABE8	01C	->07F6	[ENLCD]
	ABE9	2DC	PT= 13	
	ABEA	0D0	LD@PT-- 3	
	ABEB	046	C=0 S&X	
	ABEC	37C	RCR 12	
LB_ABEA		3E8	WRIT 15(e)	
	ABEE	3C6	RSHFC S&X	
	ABEF	2FC	RCR 13	

ABF0	3D4	PT=PT-1	
ABF1	014	?PT= 3	
ABF2	3DB	JNC -05	[LB_ABEA]
ABF3	3D8	C<>ST XP	
ABF4	288	SETF 7	
ABF5	3D8	C<>ST XP	
ABF6	3E8	WRIT 15(e)	
ABF7	0B0	C=N ALL	
ABF8	23C	RCR 2	
ABF9	130	LDI S&X	002
ABFA	002	CON:	
ABFB	358	ST=C XP	
LB_ABF9	2FE	?C#0 MS	
ABFD	013	JNC +02	[LB_ABFC]
ABFE	388	SETF 0	
LB_ABFC	398	C=ST XP	
AC00	2FC	RCR 13	
AC01	3E8	WRIT 15(e)	
AC02	3D4	PT=PT-1	
AC03	214	?PT= 2	
AC04	3C7	JC -08	[LB_ABF9]
AC05	149	?NC XQ	
AC06	024	->0952	[ENCP00]
AC07	130	LDI S&X	059
AC08	059	CON:	
AC09	358	ST=C XP	
AC0A	379	?NC XQ	
AC0B	058	->16DE	[XTONE]
AC0C	0B0	C=N ALL	
AC0D	11A	A=C M	
AC0E	2F8	READ 11(a)	
AC0F	37A	?A#C M	
AC10	037	JC +06	[LB_AC13]
AC11	2ED	?NC XQ	
AC12	040	->10BB	[BEEP]
AC13	1F9	?NC GO	
AC14	00E	->037E	[STMSGF]
AC15	28B	JNC -47d	[LB_ABE3]
LB_AC13	260	SETHX	
AC17	391	PORT DEP:	
AC18	08C	XQ	
AC19	124	->A924	[WAITKY]
AC1A	01C	PT= 3	
AC1B	278	READ 9(Q)	
AC1C	01E	A=0 MS	
LB_AC1A	23A	C=C+1 M	
AC1E	330	FETCH S&X	
AC1F	2E6	?C#0 S&X	
AC20	3B3	JNC -10d	
AC21	3DC	PT=PT+1	
AC22	17E	A=A+1 MS	
AC23	366	?A#C S&X	

AC24	3CF	JC -07	[LB_AC1A]
AC25	0B0	C=N ALL	
AC26	2A0	SETDEC	
AC27	226	C=C+1 S&X	
AC28	2F6	?C#0 XS	
AC29	289	?C GO	
AC2A	003	->00A2	[ERROF]
AC2B	11A	A=C M	
AC2C	0E6	B<>C S&X	
LB_AC2A	3CA	RSHFC R<	
AC2E	0A2	A<>C @PT	
AC2F	3FA	LSHFA M	
AC30	1BE	A=A-1 MS	
AC31	3E3	JNC -04	[LB_AC2A]
AC32	0C6	C=B S&X	
AC33	313	JNC -30d	[LB_AC12] -> [LB_ABE3]

## Routine Library:

<b>WAITKY</b>	<b>A924</b>	<b>3C8</b>	<b>CLRKEY</b>
LB_A922	160	?LOWBAT	
A926	06F	JC +13	[LB_A930]
A927	3CC	?KEY	
A928	3EB	JNC -03	[LB_A922]
A929	220	C=KEY	
LB_A927	3C8	CLRKEY	
A92B	3CC	?KEY	
A92C	3F7	JC -02	[LB_A927]
A92D	03C	RCR 3	
A92E	056	C=0 XS	
A92F	106	A=C S&X	
A930	130	LDI S&X	0C3
A931	0C3	CON:	<back arrow>
A932	366	?A#C S&X	
LB_A930	3C1	?NC GO	
A934	002	->00FO	[NFRPU]
LB_A932	130	LDI S&X	018
A936	018	CON:	<ON keycode>
A937	366	?A#C S&X	
A938	360	?C RTN	
A939	260	SETHX	
A93A	0A0	SLCTP	
A93B	149	?NC XQ	
A93C	024	->0952	[ENCP00]
A93D	3B8	READ 14(d)	
A93E	358	ST=C XP	
A93F	321	?NC GO	
A940	046	->11C8	[OFF]
<b>MARK</b>	<b>A941</b>	<b>04E</b>	<b>C=0 ALL</b>
A942	130	LDI S&X	008

A943	008	CON:	loop counter
A944	1BC	RCR 11	
A945	130	LDI S&X	0A0
A946	0A0	CON:	
A947	354	?PT= 12	
A948	0FF	JC +31d	[LB_A964]
A949	130	LDI S&X	150
A94A	150	CON:	
A94B	194	?PT= 11	
A94C	0DF	JC +27d	[LB_A964]
A94D	130	LDI S&X	088
A94E	088	CON:	
A94F	0D4	?PT= 10	
A950	0BF	JC +23d	[LB_A964]
A951	130	LDI S&X	114
A952	114	CON:	
A953	254	?PT= 9	
A954	09F	JC +19d	[LB_A964]
A955	130	LDI S&X	0AA
A956	0AA	CON:	
A957	114	?PT= 8	
A958	07F	JC +15d	[LB_A964]
A959	130	LDI S&X	051
A95A	051	CON:	
A95B	294	?PT= 7	
A95C	05F	JC +11d	[LB_A964]
A95D	130	LDI S&X	022
A95E	022	CON:	
A95F	154	?PT= 6	
A960	03F	JC +07	[LB_A964]
A961	130	LDI S& X	015
A962	015	CON:	
A963	094	?PT= 5	
A964	01F	JC +03	[LB_A963]
A965	130	LDI S&X	00A
A966	00A	CON:	
LB_A963	2FC	RCR 13	
A968	3C6	RSHFC S&X	
A969	1E6	C=C+C S&X	
A96A	2FE	?C#0 MS	
A96B	3E3	JNC -04	[LB_A963]
A96C	3E0	RTN	
<b>STODSPA96D</b>	<b>0E0</b>	<b>SLCT Q</b>	
A96E	3C1	?NC XQ	Enable & Clear LCD
A96F	0B0	->2CF0	[CLLCDE]
A970	0B0	C=N ALL	
A971	0BA	A<>C M	
A972	070	N=C ALL	
A973	11A	A=C M	
A974	2DC	PT= 13	
A975	210	LD@PT- 8	
A976	11E	A=C MS	

	A977	2FC	RCR 13	
LB_A974		130	LDI S&X	002
	A979	002	CON:	
	A97A	2FE	?C#0 MS	
	A97B	03B	JNC +07	[LB_A97E]
	A97C	046	C=0 S&X	
	A97D	31E	?A<C MS	
	A97E	023	JNC +04	[LB_A97E]
	A97F	130	LDI S&X	010
	A980	010	CON:	
	A981	29E	C=0-C MS	
LB_A97E		2FC	RCR 13	
	A983	0D4	?PT= 10	
	A984	02F	JC +05	[LB_A985]
	A985	294	?PT= 7	
	A986	01F	JC +03	[LB_A985]
	A987	054	?PT= 4	
	A988	023	JNC +04	[LB_A988]
[LB_A985]		3D8	C<>ST XP	
	A98A	288	SETF 7	
	A98B	3D8	C<>ST XP	
[LB_A988]		3E8	WRIT 15(e)	
	A98D	3D4	PT=PT-1	
	A98E	014	?PT= 3	
	A98F	34B	JNC -23d	[LB_A974]
	A990	3F8	READ 15(e)	
	A991	3F8	READ 15(e)	
	A992	198	C=M ALL	
	A993	21C	PT= 2	
	A994	010	LD@PT- 0	
	A995	0D0	LD@PT- 3	
	A996	2E2	?C#0 @PT	
	A997	01F	JC +03	[LB_A996]
	A998	130	LDI S&X	020
	A999	020	" "	<space char>
[LB_A996]		3E8	WRIT 15(e)	
	A99B	149	?NC XQ	
	A99C	024	->0952	[ENCP00]
	A99D	0A0	SLCTP	
	A99E	0B0	C=N ALL	
	A99F	130	LDI S&X	059
	A9A0	059	CON:	
	A9A1	3D8	C<>ST XP	
	A9A2	070	N=C ALL	
	A9A3	379	?NC XQ	
	A9A4	058	->16DE	[XTONE]
	A9A5	0B0	C=N ALL	
	A9A6	11A	A=C M	
	A9A7	358	ST=C XP	
	A9A8	3E0	RTN	
	<b>TSTEXP A9A9</b>	<b>278</b>	<b>READ 9(Q)</b>	
	A9AA	05E	C=0 MS	



A9AB	268	WRIT 9(Q)	
A9AC	204	CLRF 2	
[LB_A9A9]	304	CLRF 1	
A9AE	35C	PT= 12	
[LB_A9AB]	198	C=M ALL	
A9B0	082	B=A @PT	
A9B1	122	A=A+B @PT	
A9B2	037	JC +06	[LB_A9B4]
A9B3	062	A<>B @PT	
A9B4	302	?A<C @PT	
A9B5	157	JC +42d	[LB_A9D8]
A9B6	1C2	A=A-C @PT	
A9B7	043	JNC +08	[LB_A9BB]
[LB_A9B4]	062	A<>B @PT	
A9B9	282	C=0-C @PT	
A9BA	222	C=C+1 @PT	
A9BB	302	?A<C @PT	
A9BC	11B	JNC +35d	[LB_A9D8]
A9BD	198	C=M ALL	
A9BE	142	A=A+C @PT	
[LB_A9BB]	308	SETF 1	
A9C0	379	PORT DEP:	
A9C1	03C	XQ	
A9C2	141	->A941	[MARK]
[LB_A9BE]	3D4	PT=PT-1	
A9C4	014	?PT= 3	
A9C5	013	JNC +02	[LB_A9C2]
A9C6	35C	PT= 12	
[LB_A9C2]	2E2	?C#0	
A9C8	05B	JNC +11d	
A9C9	0A2	A<>C @PT	
A9CA	102	A=C @PT	
A9CB	142	A=A+C @PT	
A9CC	013	JNC +02	[LB_A9C9]
A9CD	282	C=0-C @PT	
[LB_A9C9]	222	C=C+1 @PT	
A9CF	38C	?FSET 0	
A9D0	013	JNC +02	[LB_A9CD]
A9D1	282	C=0-C @PT	
[LB_A9CD]	102	A=C @PT	
A9D3	27E	C=C-1 MS	
A9D4	37B	JNC -17d	[LB_A9BE]
A9D5	00C	?FSET 3	
A9D6	027	JC +04	
A9D7	379	PORT DEP:	
A9D8	03C	XQ	
A9D9	16D	->A96D	[STODSP]
A9DA	278	READ 9(Q)	
A9DB	23E	C=C+1 MS	
A9DC	013	JNC +02	[LB_A9D7]
A9DD	208	SETF 2	
[LB_A9D7]	268	WRIT 9(Q)	

[LB_A9D8]	3D4	PT=PT-1	
A9E0	014	?PT= 3	
A9E1	273	JNC -50d	[LB_A9AB]
A9E2	20C	?FSET 2	
A9E3	360	?C RTN	
A9E4	30C	?FSET 1	
A9E5	247	JC -56d	[LB_A9A9]
A9E6	3E0	RTN	
<b>EVAL</b>	<b>A9E7</b>	<b>0E0</b>	<b>SLCTQ</b>
A9E8	35C	PT= 12	
A9E9	0B0	C=N ALL	
A9EA	130	LDI S&X	3FF
A9EB	3FF	CON:	
A9EC	070	N=C ALL	
LB_A9E9	10E	A=C ALL	
A9EE	2E2	?C#0 @PT	
A9EF	01B	JNC +03	LB_A9EE
A9F0	1E2	C=C+C @PT	
A9F1	12B	JNC +37d	LB_AA11
LB_A9EE	006	A=0 S&X	
A9F3	176	A=A+1 XS	
A9F4	114	?PT= 8	
A9F5	023	JNC +04	LB_A9F5
A9F6	342	?A#0 @PT	
A9F7	017	JC +02	LB_A9F5
A9F8	1A6	A=A-1 S&X	
LB_A9F5	1A2	A=A-1 @PT	
A9FA	0A0	SLCTP	
A9FB	008	SETF 3	
A9FC	379	PORT DEP:	
A9FD	03C	XQ	
A9FE	1A9	->A9A9	[TSTEXP]
A9FF	20C	?FSET 2	
AA00	02B	JNC +05	LB_AA00
LB_A9FC	3DC	PT=PT+1	
AA02	120	?P=Q	
AA03	3F3	JNC -02	LB_A9FC
AA04	3E0	RTN	
LB_AA00	2DC	PT= 13	
AA06	210	LD@PT- 8	
AA07	11E	A=C MS	
AA08	04E	C=0 ALL	
AA09	0BA	A<>C M	
AA0A	11A	A=C M	
AA0B	2FC	RCR 13	
LB_AA07	31E	?A<C MS	
AA0D	01B	JNC +03	LB_AA0B
AA0E	130	LDI S&X	OFF
AA0F	OFF	CON:	
LB_AA0B	2FC	RCR 13	
AA11	146	A=A+C S&X	
AA12	046	C=0 S&X	

AA13	2EE	?C#0 ALL	
AA14	3C7	JC -08	LB_AA07
AA15	01B	JNC +03	LB_AA13
AA16	1BB	JNC +55d	LB_AA46
AA17	2B3	JNC -42d	LB_A9E9
LB_AA13	198	C=M ALL	
AA19	262	C=C-1 @PT	
AA1A	282	C=0-C @PT	
AA1B	362	?A#C @PT	
AA1C	0C7	JC +24d	LB_AA2D
AA1D	1A6	A=A-1 S&X	
AA1E	1A6	A=A-1 S&X	
AA1F	027	JC +04	
AA20	379	PORT DEP:	
AA21	03C	XQ	
AA22	141	->A941	[MARK]
LB_AA1C	3D4	PT=PT-1	
AA24	014	?PT= 3	
AA25	013	JNC +02	LB_AA20
AA26	35C	PT= 12	
LB_AA20	2E2	?C#0 @PT	
AA28	053	JNC +10d	LB_AA2B
AA29	0EE	B<>C ALL	
AA2A	198	C=M ALL	
AA2B	262	C=C-1 @PT	
AA2C	362	?A#C @PT	
AA2D	027	JC +04	LB_AA2A
AA2E	130	LDI S&X	00A
AA2F	00A	CON:	
AA30	146	A=A+C S&X	
LB_AA2A	0CE	C=B ALL	
LB_AA2B	27E	C=C-1 MS	
AA33	383	JNC -16d	LB_AA1C
LB_AA2D	3D4	PT=PT-1	
AA35	014	?PT= 3	
AA36	313	JNC -30d	LB_AA13
AA37	0B0	C=N ALL	
AA38	306	?A<C S&X	
AA39	067	JC +12d	LB_AA3E
AA3A	366	?A#C S&X	
AA3B	08F	JC +17d	LB_AA45
AA3C	0EE	B<>C ALL	
AA3D	278	READ ( 9)Q	
AA3E	1E6	C=C+C S&X	
AA3F	01F	JC +03	LB_AA3B
AA40	268	WRIT 9(Q)	
AA41	05B	JNC +11d	LB_AA45
LB_AA3B	226	C=C+1 S&X	
AA43	268	WRIT ( 9)Q	
AA44	0CE	C=B ALL	
LB_AA3E	0A6	A<>C S&X	
AA46	05E	C=0 MS	

LB_AA40	27E	C=C-1 MS	
AA48	3DC	PT=PT+1	
AA49	120	?P=Q	
AA4A	3EB	JNC -03	LB_AA40
AA4B	070	N=C ALL	
LB_AA45	0E0	SLCTQ	
LB_AA46	3D4	PT=PT-1	
AA4E	0B0	C=N ALL	
AA4F	014	?PT= 3	
AA50	23B	JNC -57d	LB_AA12
AA51	0A0	SLCTP	
AA52	01C	PT= 3	
LB_AA4C	3DC	PT=PT+1	
AA54	23E	C=C+1 MS	
AA55	3F3	JNC -02	LB_AA4C
AA56	3E0	RTN	
Header A901	0A3	"#"	
Header A902	00E	"N"	DataFile V11N2p23
Header A903	012	"R"	G. Vallance
<b>RANL A904 0F8</b>		<b>READ 3(X)</b>	
A905	128	WRIT 4(L)	
<b>RAN A906 0F8</b>		<b>READ 3(X)</b>	
A907	00E	A=0 ALL	
A908	355	?NC XQ	
A909	050	->14D5	Unlabeled - Sets DEC mode
A90A	35C	PT= 12	
A90B	250	LD@PT- 9	
A90C	210	LD@PT- 8	
A90D	090	LD@PT- 2	C = 982.3
A90E	050	LD@PT- 1	
A90F	130	LDI S&X	
A910	003	CON: 3	
A911	135	?NC XQ	
A912	060	->184D	[MP2-10]
A913	10E	A=C ALL	
A914	04E	C=0 ALL	
A915	266	C=C-1 S&X	
A916	35C	PT= 12	
A917	090	LD@PT- 2	
A918	050	LD@PT- 1	C = 0.211327
A919	050	LD@PT- 1	
A91A	0D0	LD@PT- 3	
A91B	090	LD@PT- 2	
A91C	1D0	LD@PT- 7	
A91D	01D	?NC XQ	
A91E	060	->1807	[AD2-10]
A91F	084	CLRF 5	
A920	0ED	?NC XQ	
A921	064	->193B	[INTFRC]
A922	0E8	WRIT 3(X)	
A923	3E0	RTN	

## Metronome for the HP-41

### Mark Power, DataFile V9N4 p18

Requires: Any HP41, Time Module (or CX)  
MCODE equipment (e.g ZENROM and RSU)

First of all, many thanks to Frank Wales at Zengrange without whose help I wouldn't have been able to convince the Time Module to talk to me. The functions presented here are my first attempts at using the Time Modules Phineas chip. Being a keen musician I have long been on the lookout for a cheap Metronome, well at last it's here - in disguise as an HP41!

Before we start on the program, a few words of warning to anyone with a ZENROM manual using the chapter on Advanced Machine Code Programming:

- In Section 8.2 on display handling, the diagram showing the use of Bits 6 and 7 is incorrect. The titles for the two columns should be reversed. (Bits 6 should be Bits 7).
- Still in Section 8.2, the hexadecimal value for 'DISOFF' should be '2E0' not '230'.
- Also in this section, the comment for 'READAN' should be "Read the annunciators bit pattern to C[X]".
- More importantly for this article, in Section 8.4, The Timer Module, the hexadecimal values for 'TIMER=A' and 'TIMER=B' should be reversed i.e. 3E8 TIMER=A Select timer A 3A8 TIMER=B Select timer B

The Metronome function makes use of a subroutine I have written which checks the existence of the Time Module, and that it has been initialized. If you want to explore with the module, then this routine may be useful:

#### CHECK TIME MODULE ROUTINE

This function checks that the Timer Module is plugged in and that the timer circuitry has been initialized (this is very likely if you use your '41 normally).

Routines used: ENTMR @ 50E2 (from the Timer Module)

ERRSUB @ 22E8

CLLCDE @ 2CF0

MESSL @ 07EF

MSG105 @ 1C80

ERR110 @ 22FB

Input: None.

Output: Returns if Timer present and initialized or displays "TIMER ERROR" and stops.

Assumes: Nothing.

```
x800 04E CHKTMR: C=0 ALL           ; Check Timer exists.
x801 15C PT= 6                     ;
x802 150 LC 5                       ; Access XROM of ROM in
x803 330 RDR0M                     ; page 5 (Timer page).
x804 106 A=C X                     ; Put XROM into A[X].
x805 130 LDI                       ;
x806 01A CON 26                    ; Load Timer XROM.
x807 366 ?A#C X                    ; Compare the two.
x808 05F JC+0B TMRERR              ; If not the same TMRERR
```

```

x809 389 * ; otherwise, enable Timer.
x80A 140 NC XQ ENTMR ; and disable RAM.
x80B 3A8 TIMER=B ; Enable B timers.
x80C 0B8 RALM ; Read warm start constant.
x80D 00E A=0 ALL ; Check that it is
x80E 2A0 SETDEC ; 099999999999000.
x80F 1BA A=A-1 M ;
x810 260 SETHEX ;
x811 36E ?A#C ALL ; If it is then
x812 3A0 NCRTN ; exit
x813 3A1 TMRERR: * ; Otherwise do custom
x814 088 NC XQ ERRSUB ; error
x815 3C1 * ; TIMER ERROR.
x816 0B0 NC XQ CLLCDE ; Clear & enable LCD.
x817 3BD * ;
x818 01C NC XQ MESSL ; Display message.
x819 014 T ; Text starts here.
x81A 009 I ;
x81B 00D M ;
x81C 005 E ;
x81D 012 R ;
x81E 020 ;
x81F 005 E ;
x820 012 R ;
x821 012 R ;
x822 00F O ;
x823 012 R ; 12 characters so we
x824 320 ; don't need to justify.
x825 201 * ; Output to any printers
x826 070 NC XQ MSG105 ; in TRACE mode.
x827 3ED * ; Exit via standard
x828 08A NC GO ERR110 ; error handler.

```



### METRONOME FUNCTION

This function, called METRO, takes a number of beats per minute, in from the X register. It then calculates using the standard division routine and rounding routine, the interval in hundredths of a second between beats. If a Time module is not present, TIM ER ERROR is displayed (as above).

The interval timer (as used by the CLOCK function) is then set to repeatedly cause an alarm service request. When this occurs, the timer status is checked to ensure that it was us who caused the alarm. If it wasn't then the routine exits immediately. This allows Alarms from the Time Module to interrupt operation without causing problems. If the alarm was due to us, then TONE 89 is sounded and the alarm status cleared.

While the machine is waiting for the alarm, it repeatedly polls the key down flag. If any key is pressed, the routine exits. On exit, the interval timer is stopped and the DTZIT status flag cleared (as required by the Phineas specification).

Note that the processor is running continuously (unlike in the CLOCK function) and so you shouldn't leave this program running for hours unless you want flat batteries.



Routines used: DV2-10 @ 1898  
 ROUND @ 0A35  
 ERRDE @ 282D  
 CHKTMR @ x800 (described above)  
 RSTKB @ 0098  
 ENCP00 @ 0952  
 XTONE @ 16DE  
 ENTMR @ 50E2 (from the Timer Module)

Input: X = Number of beats per minute.

Output: Beeps in accordance with rate set above or exits with error shown.

Assumes: Nothing.

```

x900 08F O
x901 012 R
x902 014 T
x903 005 E
x904 00D M
x905 04E XMETRO: C=0 ALL ; METRO entry point.
x906 130 LDI ;
x907 106 CON 262 ; Load the normalized
x908 23C RCR 2 ; number 60 into N2.
x909 10E A=C ALL ;
x90A 0F8 C=REG 3/X ; Load BPM into N1.
x90B 2A0 SETDEC ; Entry condition for:
x90C 261 * ; Floating point
x90D 060 NC XQ DV2-10 ; division (N2/N1).
x90E 016 A=0 XS ; Specify 2 digits to
x90F 176 A=A+1 XS ; be rounded in A[XS].
x910 176 A=A+1 XS ;
x911 104 CF 8 ; Set 'XRND' condition.
x912 288 SF 7 ; Set 'FIX' mode.
x913 0D5 * ; Execute floating
x914 028 NC XQ ROUND ; point RouND function.
x915 0A6 A<>C X ; Move mantissa out
x916 130 LDI ; whilst loading
x917 004 CON 04 ; normalisation value.
x918 0A6 A<>C X ;
x919 2A0 SETDEC ; This normalises the
x91A 226 C=C+1 X ; number so that it is in
x91B 226 C=C+1 X ; BCD form in 1/100ths
x91C 306 ?A<C X ; of a second in the
x91D 0B5 * ; bottom of the C register.
x91E 0A3 LOOP: C GO ERRDE ;
x91F 3DA CSR M ; If the value was too big
x920 226 C=C+1 X ; then DATA ERROR.
x921 306 ?A<C X ;
x922 3EB JNC-03 LOOP ;
x923 2BC RCR 7 ; Move 1/100ths to bottom
x924 158 M=C ; of C register.
x925 260 SETHEX ;
x926 379 * ; Check Timer module
x927 03C * ; exists and timer is

```

```

x928 000 GOSUB CHKTMR           ; running.
      x929 261 *                ; Don't start until
x92A 000 NC XQ RSTKB           ; keyboard has been
x92B 198 C=M                   ; debounced.
x92C 3E8 TIMER=A              ; Ready to get status.
x92D 168 WINST                 ; Start interval timer.
x92E 36C MLOOP: ?ALM          ; Test for alarm.
x92F 083 JNC+10 CHKEY         ; If no alarm check key.
x930 0F8 RSTS                  ; If alarm, read status
x931 3D8 C<>ST                 ; and check that it is
x932 04C ?SET 4               ; our alarm (DTZIT bit
x933 073 JNC+0E MEND          ; should be set if it is).
x934 044 CF 4                  ; If not ours then end.
x935 130 LDI                   ; Else clear bit and
x936 059 CON 89                ; get TONE number ready.
x937 3D8 C<>ST                 ; Write back status with
x938 0E8 WSTS                  ; DTZIT bit clear.
x939 149 *                     ; Disable Timer and
x93A 024 NC XQ ENCP00         ; enable chip 00.
x93B 379 *                     ; Execute TONE with
x93C 058 NC XQ XTONE          ; number in ST.
x93D 389 *                     ; Enable timer again.
x93E 140 NC XQ ENTMR          ;
x93F 3CC CHKEY: ?KEY          ; Check for key press.
x940 373 JNC-12 MLOOP         ; No key, go round again.
x941 1E8 MEND: STPINT         ; Exit, stop timer.
x942 0F8 RSTS                  ; Read status.
x943 3D8 C<>ST                 ;
x944 044 CF 4                  ; Clear DTZIT bit.
x945 3D8 C<>ST                 ;
x946 0E8 WSTS                  ; Write status back
x947 261 *                     ; Exit after debouncing
x948 002 NC GO RSTKB          ; the keyboard.

```

The DTZIT bit (Decrement Through Zero Interval Timer), indicates that the alarm was caused by the interval timer counter reaching zero at the end of the time period specified in the WINST instruction. Other bits in the status register indicate other alarm conditions (e.g label and stopwatch alarms).

The WINST (Write Interval Timer and Start) takes a BCD value in hundredths of seconds specified in C[4:0]. Hence the interval timer may be used to cause an alarm repeatedly every 0.01 to 999.99 seconds.

The CLOCK function in the Time Module uses the interval timer with an interval of 1 second or 1 minute depending on the clock mode (CLKT/CLKTD). Even using the built in XTONE function, the metronome seems to be quite capable of producing 1000 beats per minute down to one beat every 15 minutes – whether you can play at those speeds is another thing!

For us lesser mortals, it is quite capable of beating at 60 to 208 per minute, which seems to cover all of my music which has a tempo indicated on it. Perhaps the next step is to make it indicate 'Up beats' and be able to play in all of the common time signatures! Any takers?

.END.

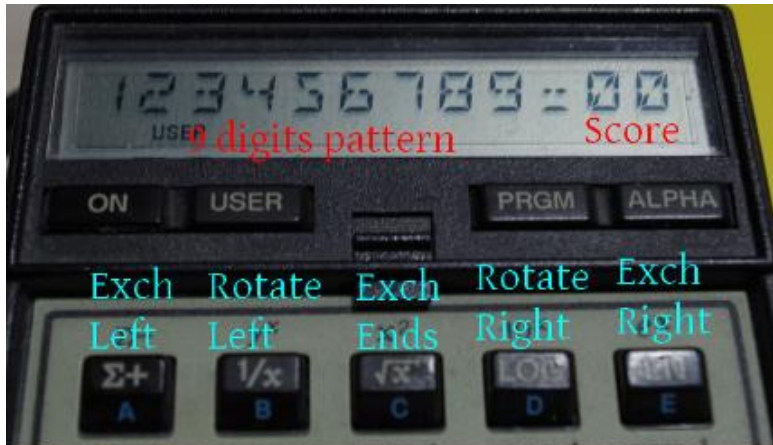
## Pattern Twister(TWIX)

Guillaume Tello- [http://gtello.pagesperso-orange.fr/hp41\\_e.htm](http://gtello.pagesperso-orange.fr/hp41_e.htm)

**Source:**I found this game in a version for the HP67/97 "Super Twix" written by David V. Smith.

**The Objective:**The goal is to transform a pattern of 9 digits into another with limited movements. Traditionally, one tries to transform the pattern "123456789" into "987654321".

**How to play:**You write the starting pattern of 9 digits, then XEQ TWIX, and you can see the board game as show on the right:



```
;          13 12 11 10  9  8  7  6  5  4  3  2  1  0
; A=      x  x  N  NNNNNNNN  D  S  S
; NNN... the 9 digits
; D = character for separation
; SS how many steps in game
; B used to save A
; C used everywhere
; no other register, no flag.
```

```
.NAME      "TWIX"
F006 098      #098          ;"X"
F007 009      #009          ;"I"
F008 017      #017          ;"W"
F009 014      #014          ;"T"
; *****
; initialization
; *****
F00A 0F8      [TWIX]      READ      3          ; copy X into C
F00B 11A      A=C         M          ; the mantissa into A
F00C 39A      RSHFA      M          ; 9 digits in positions 11-4
F00D 006      A=0        S&X        ; clear exp. =no. of steps, score
F00E 2A0      SETDEC     ; everything in decimal.
; *****
; DISPLAY routine
; *****
F00F 130010 [DISP]      LDIS&X  10
F011 270      RAMSLCT          ; dummy ram zone
F012 1300FD      LDIS&X  FD
F014 3F0      PRPHSLCT          ; select screen
```

```

F015 0AE10E      C=A      ALL      ; get the 9 digits + score
F017 21C         R=        2
F018 350         LD@R     D          ; code for separator in XS
F019 1BC         RCR      11         ; the first digit in correct
position
F01A 0A0         SLCTP
F01B 35C         R=        12         ; for the loop
F01C 0E0         [L1]      SLCTQ
F01D 21C         R=        2
F01E 010         LD@R     0
F01F 0D0         LD@R     3          ; prepare code 03N into C
F020 3E8         WRIT     15         ; copy to screen
F021 2FC         RCR      13         ; next digit
F022 0A0         SLCTP
F023 3D4         R=R-1
F024 394         ?R=      0          ; until zero
F025 3BB         JNC      [L1]      ; -9 F01C
F026 130000      LDIS&X  0          ; quit screen
F028 3F0         PRPHSLCT
F029 3C8         [WAIT]    CLRKEY      ; wait for a key
F02A 3CC         ?KEY
F02B 3F3         JNC      [WAIT]      ; -2 F029
F02C 220         C=KEY
F02D 33C         RCR      1          ; key into 3-2
F02E 166         A=A+1    S&X      ; one more step in score
F02F 08E         B=A      ALL
F030 0AE         A<>C    ALL      ; key in A 3-2
F031 21C         R=        2          ; point to XS
F032 0D0         LD@R     3          ; XS = 3
F033 376         ?A#C    XS        ; last nibble of key is 3 ?
F034 3A0         ?NCRTN
F035 356         ?A#0    XS        ; last nibble of key is zero?
F036 39F         JC       [WAIT]      ; -13 F029 ; no, so invalid key !
F037 0F53C0      ?NCXQ   [ACTION]    ; F03D ; subprogram for actions
F039 3C8         [KOFF]    CLRKEY      ; wait for the key be released
F03A 3CC         ?KEY
F03B 3F7         JC       [KOFF]      ; -2 F039
F03C 29B         JNC      [DISP]      ; -45 F00F ; display new pattern
; *****
; execute the action according to
; one of the 5 keys of first line
; *****
F03D 38E         [ACTION]    RSHFA    ALL      ; high nibble of KEY in XS
F03E 15C         R=        6
F03F 050         LD@R     1
F040 0D0         LD@R     3
F041 1D0         LD@R     7
F042 210         LD@R     8
F043 310         LD@R     C          ; codes for the 5 keys
F044 376         ?A#C    XS        ; equal to C (key C0=LN)
F045 0B3         JNC      [XRIGHT]    ; +22 F05B ; yes, exchange right
F046 33C         RCR      1
F047 376         ?A#C    XS        ; equal to 8 (key 80=LOG)
F048 113         JNC      [RRIGHT]    ; +34 F06A ; yes, rotate right
F049 33C         RCR      1
F04A 376         ?A#C    XS        ; equal to 7 (key 70=SQR)
F04B 093         JNC      [XENDS]    ; +18 F05D ; yes, exchange ends
F04C 33C         RCR      1
F04D 376         ?A#C    XS        ; equal to 3 (key 30=1/X)
F04E 0BB         JNC      [RLEFT]     ; +23 F065 ; yes, rotate left
F04F 33C         RCR      1

```

```

F050 376          ?A#C   XS          ; equal to 1 (key 10=S+)
F051 360          ?CRTN          ; no! bad key (impossible...)
; *****
; (S+) exchange left
; *****
F052 0DC          R=      10
F053 0EE [XCOMM]  C<>B   ALL        ; old A value in C
F054 10E          A=C     ALL        ; and back in A
F055 33C          RCR     1
F056 102          A=C     @R        ;old a11 into a10 (or a4 into a3)
F057 37C          RCR     12
F058 3DC          R=R+1
F059 102 [GETR]   A=C     @R        ; old a10 into a11 (or a3 into a4)
F05A 3E0          RTN
; *****
; (LN) exchange right
; *****
F05B 01C [XRIGHT] R=      3
F05C 3BB          JNC     [XCOMM] ; -9   F053
; *****
; (SQR) exchange ends
; *****
F05D 0EE [XENDS]  C<>B   ALL        ; old A value in C
F05E 10E          A=C     ALL        ; and back in A
F05F 17C          RCR     6
F060 19C          R=      11
F061 102          A=C     @R        ; old a3 into a11
F062 23C          RCR     2
F063 01C [GET3]   R=      3
F064 3AB          JNC     [GETR] ; -11  F059 ; old a11 into a4
; *****
; (1/X) rotate left
; *****
F065 0EE [RLEFT]  C<>B   ALL        ; old A value in C
F066 10E          A=C     ALL        ; and back in A
F067 3FA          LSHFA   M         ; shift mantissa, but one digit is
lost!
F068 13C          RCR     8         ; put the old first digit in 3
F069 3D3          JNC     [GET3] ; -6   F063 ; and get the lost
digit
; *****
; (LOG) rotate right
; *****
F06A 0EE [RRIGHT] C<>B   ALL        ; old A value in C
F06B 10E          A=C     ALL        ; and back in A
F06C 39A          RSHFA   M         ; shift mantissa, but one digit
lost!
F06D 17C          RCR     6         ; old last digit in 11
F06E 19C          R=      11
F06F 353          JNC     [GETR] ; -22  F059 ; and get lost digit

```

## “Le Compte est Bon” for the HP-41

Jean-Marc Baillard-<http://hp41programs.yolasite.com/lceb.php>

### Overview

"Le compte est bon" is a popular French TV game: a set of 6 numbers (n1 , n2 , n3 , n4 , n5 , n6) is chosen at random amongst the following 24 integers:

1 - 1 - 2 - 2 - 3 - 3  
 4 - 4 - 5 - 5 - 6 - 6  
 7 - 7 - 8 - 8 - 9 - 9  
 10 - 10 - 25 - 50 - 75 - 100

for instance the set including numbers: 1 - 9 - 8 - 4 - 3 - 100

Then you have to obtain another random number N between 100 and 999 inclusive, say N = 629, using the 6 numbers above (or fewer) and using the 4 basic operations: addition, subtraction, multiplication, division.

For example, one solution of the above problem is:

100 - 1 = 99  
 99 / 3 = 33  
 33 + 4 = 37  
 9 + 8 = 17  
 37 \* 17 = 629

A solution has 5 operations at most, and the same number from the set cannot be used multiple times. More exactly, you can use "ni" twice or less if there are 2 "ni"s, and you can only use "ni" once (or less) if there is just one (or none) "ni".

Example1: 3 - 7 - 7 - 50 - 1 - 8 // 947

3, STO 01, 7, STO 02, STO 03, 50, STO 04, 1, STO 05, 8, STO 06, 947, STO 07

XEQ "LCEB" >>>>	"50+8=58"	Execution time = 15 s
R/S	"58*7=406"	
R/S	"406*7=2842"	
R/S	"2842-1=2841"	
R/S	"2841/3=947"	

Example2: 4 - 7 - 50 - 1 - 4 - 100 // 903

4 STO 01 ..... 903 STO 07

XEQ "LCEB" >>>	"100/4=25"	Execution time = 70s
R/S	"25-7=18"	
R/S	"50*18=900"	
R/S	"900-1=899"	
R/S	"899+4=903"	

Example3: 2 - 3 - 4 - 5 - 6 - 7 // 5041

1 STO 01 ..... 5041 STO 07

XEQ "LCEB" >>>> "Nb NOT FOUND" Execution time = 15m27s

Here, there is no solution at all since we cannot exceed 5040.

The following program can help you to find one or several solution(s) to this problem. Note the use of a MCODE routine to perform most of the auxiliary calculations – optimized for this purpose.

Data Registers: • R01 = n1 • R02 = n2 ..... • R06 = n6 • R07 = N

These 7 registers are to be initialized before executing "LCEB"

R00 & R08 thru R17: temp

Flags: F05-F06-F07

Subroutines: LCEB MCODE function.

Notes:

Since the M-Code routine uses synthetic register a, "LCEB" cannot be used as more than a first-level subroutine.

- If the solution involves 1 operation, flag F01 is set. Press XEQ 01 if you want to display it again.
- If the solution involves 2 operations, flag F02 is set. Press XEQ 02 to display it again.
- If the solution involves 5 operations, flag F05 is set. Press XEQ 05 to display it again.
- Flag F07 is set if the solution has been found in the second part of the search.

It may be wise to add the following lines at the beginning of the program:

```
7          LBL 14      ABS      LN      STO IND Z
E4         CLX        INT      X>Y?   DSE Z
ENTER^    RCL IND Z  X=0?    ACOS    GTO 14
```

to check that all the data in registers R01 thru R07 are valid.

### Program Listing.

Note that the name of the FOCAL program is "TCIG", i.e. 'The Count is Good'

Note also that the program will prompt for the number entry automatically.

<b>01*LBL "TCIG"</b>	31 STO 03	63 RCL 04	95 X<>Y
02 1.007	32 RDN	64 RCL 00	96 STO T
03*LBL 10	33 STO 02	65 XEQ 00	97 X<>Y
04 "N"	34 X<>Y	66*LBL 03	98 " "
05 AIP	35 STO 01	67 RCL 09	99 ARCL Y
06 "`=?"	36 LASTX	68 RCL 03	100 +
07 PROMPT	37 STO 05	69 RCL 10	101 X=Y?
08 STO IND Y	38 RCL [	70 XEQ 00	102 "`+"
09 RDN	39 STO 06	71*LBL 02	103 X<> T
10 ISG X	40 RCL ]	72 FS? 07	104 LASTX
11 GTO 10	41 STO 08	73 GTO 07	105 -
12 CLX	42 RCL ^	74 RCL 08	106 X=Y?
13 X<>F	43 STO 09	75 RCL 02	107 "`-"
14 RCL 07	44 RCL _	76 RCL 09	108 X<> T
15 STO \	45 STO 10	77 XEQ 00	109 LASTX
16 RCL 06	46 RCL a	78*LBL 01	110 ST/ T
17 STO [	47 STO 00	79 RCL 07	111 *
18 RCL 05	48 FS? 01	80 RCL 01	112 X=Y?
19 SIGN	49 GTO 01	81 RCL 08	113 "*" "
20 RCL 01	50 FS? 02	82 GTO 00	114 X<> T
21 RCL 02	51 GTO 02	83*LBL 07	115 X=Y?
22 RCL 03	52 FS? 03	84 RCL 08	116 "`/"
23 RCL 04	53 GTO 03	85 RCL 01	117 ARCL L
<b>24 LCeB</b>	54 FS? 04	86 RCL 02	118 "`="
25 X=0?	55 GTO 04	87 XEQ 00	119 ARCL Y
26 "Nb NOT FOUND"	56*LBL 05	88 RCL 07	120 FIX 4
27 X=0?	57 RCL 00	89 RCL 08	121 SF 29
28 PROMPT	58 RCL 05	90 RCL 09	122 PROMPT
29 STO 04	59 RCL 06	91*LBL 00	123 END
30 RDN	60 XEQ 00	92 FIX 0	
	61*LBL 04	93 CF 29	
	62 RCL 10	94 X<Y?	

Header A817 082 "B"  
Header A818 065 "e"  
Header A819 003 "C"  
Header A81A 00C "L"

Le Comptest Bon

<b>LCeB A81B 2A0 SETDEC</b>		1st. word at addr F08C
A81C 130	LDI S&X	
A81D 006	006	
A81E 070	N=C ALL	
A81F 270	RAMSLCT	
A820 375	PORT DEP:	to shift the digits in the right
A821 03C	XQ	format - part of the mantissa field
A822 284	->AA84	[ADR_F292]
A823 2F0	WRTRDATA	
A824 0B0	C=N ALL	
A825 266	C=C-1 S&X	
A826 3C3	JNC -08	
A827 04E	C=0 ALL	
A828 01C	PT=3	



	A829	090	LD@PT- 3	
	A82A	070	N=C ALL	
	A82B	2DC	PT=13	
	A82C	150	LD@PT- 5	
	A82D	110	LD@PT- 4	
	A82E	0D0	LD@PT- 3	
	A82F	130	LDI S&X	
	A830	006	006	
	A831	158	M=C ALL	
FOA3	A832	1B8	READ 6(N)	LBL01
	A833	10E	A=C ALL	
	A834	0B0	C=N ALL	
	A835	0EE	B<>C ALL	
	A836	04E	C=0 ALL	
	A837	270	RAMSLCT	
	A838	038	READATA	
	A839	304	CLRF 1	
	A83A	204	CLRF 2	
	A83B	004	CLRF 3	
	A83C	31A	?A<C M	
	A83D	017	JC +02	
	A83E	0BA	A<>C M	
	A83F	33A	?A<B M	
	A840	09F	JC +19d	
	A841	0EE	C<>B ALL	parameter passing
	A842	375	PORT DEP:	Quotient-remainder sub
	A843	03C	XQ	
	A844	2B3	->AAB3	[ADR_F2BB]
	A845	35A	?A#0 M	
	A846	1D3	JNC +58d	[LBL_02]
	A847	308	SETF 1	
	A848	04E	C=0 ALL	
	A849	270	RAMSLCT	
	A84A	038	READATA	
	A84B	10E	A=C ALL	
	A84C	1B8	READ 6(N)	
	A84D	0EE	C<>B ALL	parameter passing
	A84E	375	PORT DEP:	Multiplication Subroutine
	A84F	03C	XQ	
	A850	29E	->AA9E	[ADR_F2A8]
	A851	17B	JNC +47d	[LBL_02]
	A852	303	JNC -32d	
	A853	208	SETF 2	
	A854	04E	C=0 ALL	
	A855	270	RAMSLCT	
	A856	038	READATA	
	A857	10E	A=C ALL	
	A858	1B8	READ 6(N)	
	A859	21A	C=A+C M	
	A85A	133	JNC +38d	[LBL_02]
	A85B	008	SETF 3	
	A85C	04E	C=0 ALL	

A85D	270	RAMSLCT	
A85E	038	READATA	
A85F	10E	A=C ALL	
A860	1B8	READ 6(N)	
A861	31A	?A<C M	
A862	013	JNC +02	
A863	0BA	A<>C M	
A864	25A	C=A-C M	
A865	0DB	JNC +27d	[LBL_02]
A866	04E	C=0 ALL	
A867	270	RAMSLCT	
A868	038	READATA	
A869	10E	A=C ALL	
A86A	1D8	C<>M ALL	
A86B	266	C=C-1 S&X	
A86C	270	RAMSLCT	
A86D	1D8	C<>M ALL	
A86E	038	READATA	
A86F	0AE	A<>C ALL	
A870	2F0	WRTDATA	
A871	0AE	A<>C ALL	
A872	028	WRIT 0(T)	
A873	198	C=M ALL	
A874	2E6	?C#0 S&X	
A875	2EF	JC -35d	[LBL_01]
A876	365	PORT DEP:	GTO 2nd part of the routine
A877	03C	GO	if no solution has been found yet
A878	1C5	->A9C5	[ADR_F1F3]
A879	00C	?FSET 3	
A87A	367	JC -20d	
A87B	20C	?FSET 2	
A87C	2FF	JC -33d	
A87D	30C	?FSET 1	
A87E	2AF	JC -43d	
A87F	243	JNC -56d	
LBL 02	A880	1E8 WRIT 7(O)	Since we can only use 4 subroutine-levels,
<b>REDO7</b>	<b>A881</b>	<b>1F8 READ 7(O)</b>	CPU-flags are used to avoid ?NCXQ LBL 02
	A882	10E A=C ALL	
	A883	0B0 C=N ALL	
	A884	0EE B<>C ALL	
	A885	078 READ 1(Z)	
	A886	37A ?A#C M	
	A887	037 JC +06	
	A888	365 PORT DEP:	GTO end1
	A889	03C GO	
	A88A	194 ->A994	[ADR_F1C9]
	A88B	3B3 JNC -10d	
	A88C	36B JNC -19d	
	A88D	31A ?A<C M	
	A88E	017 JC + 02	
	A88F	0BA A<>C M	
	A890	33A ?A<B M	

A891	05F	JC +11d	
A892	0EE	C<>B ALL	parameter passing
A893	375	PORT DEP:	Quotient-remainder sub
A894	03C	XQ	
A895	2B3	->AAB3	[ADR_F2BB]
A896	35A	?A#0 M	
A897	02F	JC +05	
A898	228	WRIT 8(P)	parameter passing
A899	375	PORT DEP:	LBL 03
A89A	03C	XQ	
A89B	0C9	->A8C9	[ADR_F125]
A89C	1F8	READ 7(O)	
A89D	10E	A=C ALL	
A89E	078	READ 1(Z)	
A89F	21A	C=A+C M	
A8A0	02F	JC +05	
A8A1	228	WRIT 8(P)	parameter passing
A8A2	375	PORT DEP:	LBL 03
A8A3	03C	XQ	
A8A4	0C9	->A8C9	[ADR_F125]
A8A5	1F8	READ 7(O)	
A8A6	10E	A=C ALL	
A8A7	078	READ 1(Z)	
A8A8	31A	?A<C M	
A8A9	013	JNC +02	
A8AA	0BA	A<>C M	
A8AB	25A	C=C-A M	
A8AC	02F	JC +05	
A8AD	228	WRIT 8(P)	parameter passing
A8AE	375	PORT DEP:	LBL 03
A8AF	03C	XQ	
A8B0	0C9	->A8C9	[ADR_F125]
A8B1	078	READ 1(Z)	
A8B2	10E	A=C ALL	
A8B3	0B8	READ 2(Y)	
A8B4	0AE	A<>C ALL	
A8B5	0A8	WRIT 2(Y)	
A8B6	0F8	READ 3(X)	
A8B7	0AE	A<>C ALL	
A8B8	0E8	WRIT 3(X)	
A8B9	138	READ 4(L)	
A8BA	0AE	A<>C ALL	
A8BB	128	WRIT 4(L)	
A8BC	178	READ 5(M)	
A8BD	068	WRIT 1(Z)	
A8BE	0AE	A<>C ALL	
A8BF	168	WRIT 5(M)	
A8C0	198	C=M ALL	
A8C1	27E	C=C-1 MS	
A8C2	158	M=C ALL	
A8C3	2FE	?C#0 MS	
A8C4	23F	JC -57d	

	A8C5	2DC	PT=13	
	A8C6	150	LD@PT- 5	
	A8C7	158	M=C ALL	
	A8C8	223	JNC -60d	
F125	A8C9	2A0	SETDEC	LBL 03
<b>REDO8</b>	<b>A8CA</b>	<b>238</b>	<b>READ 8(P)</b>	get parameter
	A8CB	10E	A=C ALL	
	A8CC	0B0	C=N ALL	
	A8CD	0EE	B<>C ALL	
	A8CE	0B8	READ 2(Y)	
	A8CF	37A	?A#C M	
	A8D0	027	JC +04	
	A8D1	365	PORT DEP:	END 02
	A8D2	03C	GO	
	A8D3	199	->A999	[ADR_F1CD]
	A8D4	31A	?A<C M	
	A8D5	017	JC +02	
	A8D6	0BA	A<>C M	
	A8D7	33A	?A<B M	
	A8D8	05F	JC +11d	
	A8D9	0EE	C<>B ALL	parameter passing
	A8DA	375	PORT DEP:	Quotient-remainder sub
	A8DB	03C	XQ	
	A8DC	2B3	->AAB3	[ADR_F2BB]
	A8DD	35A	?A#0 M	
	A8DE	02F	JC +05	
	A8DF	268	WRIT 9(Q)	parameter passing
	A8E0	375	PORT DEP:	LBL 04
	A8E1	03C	XQ	
	A8E2	110	->A910	[ADR_F15C]
	A8E3	238	READ 8(P)	
	A8E4	10E	A=C ALL	
	A8E5	0B8	READ 2(Y)	
	A8E6	21A	C=A+C M	
	A8E7	02F	JC +05	
	A8E8	268	WRIT 9(Q)	parameter passing
	A8E9	375	PORT DEP:	LBL 04
	A8EA	03C	XQ	
	A8EB	110	->A910	[ADR_F15C]
	A8EC	238	READ 8(P)	
	A8ED	10E	A=C ALL	
	A8EE	0B8	READ 2(Y)	
	A8EF	31A	?A<C M	
	A8F0	013	JNC +02	
	A8F1	0BA	A<>C M	
	A8F2	25A	C=A-C M	
	A8F3	02F	JC +05	
	A8F4	268	WRIT 9(Q)	parameter passing
	A8F5	375	PORT DEP:	LBL 04
	A8F6	03C	XQ	
	A8F7	110	->A910	[ADR_F15C]
	A8F8	0B8	READ 2(Y)	

A8F9	10E	A=C ALL	
A8FA	0F8	READ 3(X)	
A8FB	0AE	A<>C ALL	
A8FC	0E8	WRIT 3(X)	
A8FD	138	READ 4(L)	
A8FE	0AE	A<>C ALL	
A8FF	128	WRIT 4(L)	
A900	178	READ 5(M)	
A901	0A8	WRIT 2(Y)	
A902	0AE	A<>C ALL	
A903	168	WRIT 5(M)	
A904	198	C=M ALL	
A905	35C	PT=12	
A906	262	C=C-1@PT	
A907	158	M=C ALL	
A908	2E2	?C#0@PT	
A909	20F	JC- 63d	
A90A	110	LD@PT- 4	
A90B	158	M=C ALL	
A90C	3CC	?KEY	
A90D	3A0	?NC RTN	
A90E	020	XQ>GO	
A90F	3E0	RTN	
F15C	A910	2A0 SETDEC	LBL04
<b>REDO9</b>	<b>A911</b>	<b>278 READ 9(Q)</b>	get parameter
A912	10E	A=C ALL	
A913	0B0	C=N ALL	
A914	0EE	B<>C ALL	
A915	0F8	READ 3(X)	
A916	37A	?A#C M	
A917	02F	JC +05	
A918	365	PORT DEP:	end 03
A919	03C	GO	
A91A	19E	->A99E	[ADR_F1D1]
A91B	3B3	JNC -10d	
A91C	31A	?A<C M	
A91D	017	JC +02	
A91E	0BA	A<>C M	
A91F	33A	?A<B M	
A920	0B7	JC +22d	
A921	0EE	C<>B ALL	parameter passing
A922	375	PORT DEP:	Quotient-remainder sub
A923	03C	XQ	
A924	2B3	->AAB3	[ADR_F2BB]
A925	35A	?A#0 M	
A926	02F	JC +05	
A927	2E8	WRIT 11(a)	parameter passing
A928	375	PORT DEP:	LBL 05
A929	03C	XQ	
A92A	15D	->A95D	[ADR_F197]
A92B	278	READ 9(Q)	
A92C	10E	A=C ALL	

A92D	0F8	READ 3(X)	
A92E	0EE	C<>B ALL	parameter passing
A92F	375	PORT DEP:	Multiplication Subroutine
A930	03C	XQ	
A931	29E	->AA9E	[ADR_F2A8]
A932	2E8	WRIT 11(a)	parameter passing
A933	375	PORT DEP:	LBL 05
A934	03C	XQ	
A935	15D	->A95D	[ADR_F197]
A936	278	READ 9(Q)	
A937	10E	A=C ALL	
A938	0F8	READ 3(X)	
A939	21A	C=A+C M	
A93A	02F	JC +05	
A93B	2E8	WRIT 11(a)	parameter passing
A93C	375	PORT DEP:	LBL 05
A93D	03C	XQ	
A93E	15D	->A95D	[ADR_F197]
A93F	278	READ 9(Q)	
A940	10E	A=C ALL	
A941	0F8	READ 3(X)	
A942	31A	?A<C M	
A943	013	JNC +02	
A944	0BA	A<>C M	
A945	25A	C=A-C M	
A946	02F	JC +05	
A947	2E8	WRIT 11(a)	parameter passing
A948	375	PORT DEP:	LBL 05
A949	03C	XQ	
A94A	15D	->A95D [	ADR_F197]
A94B	0F8	READ 3(X)	
A94C	10E	A=C ALL	
A94D	138	READ 4(L)	
A94E	0AE	A<>C ALL	
A94F	128	WRIT 4(L)	
A950	178	READ 5(M)	
A951	0E8	WRIT 3(X)	
A952	0AE	A<>C ALL	
A953	168	WRIT 5(M)	
A954	198	C=M ALL	
A955	19C	PT=11	
A956	262	C=C-1@PT	
A957	158	M=C ALL	
A958	2E2	?C#0@PT	
A959	217	JC -62d	
A95A	0D0	LD@PT- 3	
A95B	158	M=C ALL	
A95C	3E0	RTN	
F197	A95D	2A0 SETDEC	LBL 05
	A95E	2F8 READ 11(a)	get parameter
	A95F	10E A=C ALL	
	A960	178 READ 5(M)	

A961	37A	?A#C M	
A962	01B	JNC +03	[end4]
A963	138	READ 4(L)	
A964	37A	?A#C M	
A965	1EB	JNC +61d	[end4]
A966	31A	?A<C M	
A967	013	JNC +02	
A968	0BA	A<>C M	
A969	1DA	A=A-C M	
A96A	178	READ 5(M)	
A96B	37A	?A#C M	
A96C	1D3	JNC +58d	[end5]
A96D	2F8	READ 11(a)	
A96E	10E	A=C ALL	
A96F	138	READ 4(L)	
A970	15A	A=A+C M	
A971	178	READ 5(M)	
A972	37A	?A#C M	
A973	19B	JNC +51d	[end5]
A974	2F8	READ 11(a)	
A975	10E	A=C ALL	
A976	0B0	C=N ALL	
A977	0EE	B<>C ALL	
A978	138	READ 4(L)	
A979	31A	?A<C M	
A97A	017	JC +02	
A97B	0BA	A<>C M	
A97C	33A	?A<B M	
A97D	05F	JC +11d	
A97E	0EE	C<>B ALL	parameter passing
A97F	375	PORT DEP:	Quotient-remainder sub
A980	03C	XQ	
A981	2B3	->AAB3	[ADR_F2BB]
A982	35A	?A#0 M	
A983	02F	JC +05	
A984	10E	A=C ALL	
A985	178	READ 5(M)	
A986	37A	?A#C M	
A987	0FB	JNC +31d	[end5]
A988	2F8	READ 11(a)	
A989	10E	A=C ALL	
A98A	138	READ 4(L)	
A98B	0EE	C<>B ALL	parameter passing
A98C	375	PORT DEP:	Multiplication Subroutine
A98D	03C	XQ	
A98E	29E	->AA9E	[ADR_F2A8]
A98F	10E	A=C ALL	
A990	178	READ 5(M)	
A991	37A	?A#C M	
A992	360	?C RTN	
A993	09B	JNC +19d	[end5]
F1C9	A994	2A0	SETDEC end1

	A995	04E	C=0 ALL	
	A996	2DC	PT=13	
	A997	110	LD@PT- 4	
	A998	0A3	JNC +20d	[OK]
F1CD	A999	2A0	SETDEC	end2
	A99A	04E	C=0 ALL	
	A99B	2DC	PT=13	
	A99C	090	LD@PT- 2	
	A99D	073	JNC +14d	[XQGO1]
F1D1	A99E	2A0	SETDEC	end3
	A99F	04E	C=0 ALL	
	A9A0	23E	C=C+1 MS	
	A9A1	04B	JNC +09	[XQGO2]
F1D4	A9A2	04E	C=0 ALL	end4 -
	A9A3	35C	PT=12	
	A9A4	210	LD@PT- 8	
	A9A5	023	JNC +04	[XQGO3]
F1D8	A9A6	04E	C=0 ALL	end5
	A9A7	35C	PT=12	
	A9A8	110	LD@PT- 4	
	A9A9	020	XQ->GO	
	A9AA	020	XQ->GO	
	A9AB	020	XQ->GO	
	A9AC	10E	A=C ALL	free at last...
	A9AD	3B8	READ 14(d)	
	A9AE	20E	C=A+C ALL	
	A9AF	3A8	WRIT 14(d)	
	A9B0	130	LDI S&X	
	A9B1	009	009	
	A9B2	070	N=C ALL	
	A9B3	270	RAMSLCT	
	A9B4	038	READATA	
	A9B5	10E	A=C ALL	parameter passing
	A9B6	375	PORT DEP:	reformat
	A9B7	03C	XQ	
	A9B8	28F	->AA8F	[ADR_F29B]
	A9B9	2F0	WRTDATA	
	A9BA	0B0	C=N ALL	
	A9BB	266	C=C-1 S&X	
	A9BC	3B3	JNC -10d	
	A9BD	2F8	READ 14(d)	
	A9BE	10E	A=C ALL	parameter passing
	A9BF	375	PORT DEP:	reformat
	A9C0	03C	XQ	
	A9C1	28F	->AA8F	[ADR_F29B]
	A9C2	2E8	WRIT 14(d)	
	A9C3	3C1	?NC GO	4 subroutine-levels are used,
	A9C4	002	->00F0	[NFRPU]
F1F3	A9C5	2A0	SETDEC	2nd part of the search
	A9C6	198	C=M ALL	
	A9C7	01C	PT=3	
	A9C8	190	LD@PT- 6	



	A9C9	130	LDI S&X	
	A9CA	006	006	
	A9CB	158	M=C ALL	
	A9CC	04E	C=0 ALL	
	A9CD	35C	PT=12	
	A9CE	050	LD@PT- 1	
	A9CF	10E	A=C ALL	
	A9D0	3B8	READ 14(d)	
	A9D1	20E	C=A+C ALL	
	A9D2	3A8	WRIT 14(d)	
F200	A9D3	2A0	SETDEC	LBL 12
	A9D4	04E	C=0 ALL	
	A9D5	270	RAMSLCT	
	A9D6	038	READATA	
	A9D7	10E	A=C ALL	
	A9D8	078	READ 1(Z)	
	A9D9	21A	C=A+C M	
	A9DA	1E8	WRIT 7(O)	
	A9DB	10E	A=C ALL	
	A9DC	1B8	READ 6(N)	
	A9DD	31A	?A<C M	
	A9DE	017	JC +02	
	A9DF	0BA	A<>C M	
	A9E0	0EE	C<>B ALL	parameter passing
	A9E1	375	PORT DEP:	Quotient-remainder sub
	A9E2	03C	XQ	
	A9E3	2B3	->AAB3	[ADR_F2BB]
	A9E4	35A	?A#0 M	
	A9E5	02F	JC +05	
	A9E6	228	WRIT 8(P)	parameter passing
	A9E7	375	PORT DEP:	LBL 03
	A9E8	03C	XQ	
	A9E9	0C9	->A8C9	[ADR_F125]
	A9EA	04E	C=0 ALL	
	A9EB	270	RAMSLCT	
	A9EC	038	READATA	
	A9ED	10E	A=C ALL	
	A9EE	078	READ 1(Z)	
	A9EF	37A	?A#C M	
	A9F0	0CB	JNC +25d	
	A9F1	31A	?A<C M	
	A9F2	013	JNC +02	
	A9F3	0BA	A<>C M	
	A9F4	25A	C=A-C M	
	A9F5	1E8	WRIT 7(O)	
	A9F6	10E	A=C ALL	
	A9F7	0B0	C=N ALL	
	A9F8	0EE	B<>C ALL	
	A9F9	1B8	READ 6(N)	
	A9FA	31A	?A<C M	
	A9FB	017	JC +02	
	A9FC	0BA	A<>C M	

A9FD	33A	?A<B M	
A9FE	05F	JC +11d	
A9FF	0EE	C<>B ALL	parameter passing
AA00	375	PORT DEP:	Quotient-remainder sub
AA01	03C	XQ	
AA02	2B3	->AAB3	[ADR_F2BB]
AA03	35A	?A#0 M	
AA04	02F	JC +05	
AA05	228	WRIT 8(P)	parameter passing
AA06	375	PORT DEP:	LBL 03
AA07	03C	XQ	
AA08	0C9	->A8C9	[ADR_F125]
AA09	04E	C=0 ALL	
AA0A	270	RAMSLCT	
AA0B	038	READATA	
AA0C	10E	A=C ALL	
AA0D	0B0	C=N ALL	
AA0E	0EE	B<>C ALL	
AA0F	078	READ 1(Z)	
AA10	31A	?A<C M	
AA11	017	JC +02	
AA12	0BA	A<>C M	
AA13	33A	?A<B M	
AA14	1EF	JC +61d	
AA15	0EE	C<>B ALL	parameter passing
AA16	375	PORT DEP:	Multiplication Subroutine
AA17	03C	XQ	
AA18	29E	->AA9E	[ADR_F2A8]
AA19	1E8	WRIT 7(O)	
AA1A	10E	A=C ALL	
AA1B	1B8	READ 6(N)	
AA1C	31A	?A<C M	
AA1D	013	JNC +02	
AA1E	0BA	A<>C M	
AA1F	25A	C=A-C M	
AA20	02F	JC +05	
AA21	228	WRIT 8(P)	parameter passing
AA22	375	PORT DEP:	LBL 03
AA23	03C	XQ	
AA24	0C9	->A8C9	[ADR_F125]
AA25	1F8	READ 7(O)	
AA26	10E	A=C ALL	
AA27	1B8	READ 6(N)	
AA28	21A	C=A+C M	
AA29	02F	JC +05	
AA2A	228	WRIT 8(P)	parameter passing
AA2B	375	PORT DEP:	LBL 03
AA2C	03C	XQ	
AA2D	0C9	->A8C9	[ADR_F125]
AA2E	04E	C=0 ALL	
AA2F	270	RAMSLCT	
AA30	038	READATA	

AA31	10E	A=C ALL	
AA32	078	READ 1(Z)	
AA33	31A	?A<C M	
AA34	017	JC +02	
AA35	0BA	A<>C M	
AA36	0EE	C<>B ALL	parameter passing
AA37	375	PORT DEP:	Quotient-remainder sub
AA38	03C	XQ	
AA39	2B3	->AAB3	[ADR_F2BB]
AA3A	35A	?A#0 M	
AA3B	0B7	JC +22d	
AA3C	1E8	WRIT 7(O)	
AA3D	10E	A=C ALL	
AA3E	1B8	READ 6(N)	
AA3F	21A	C=A+C M	
AA40	02F	JC +05	
AA41	228	WRIT 8(P)	parameter passing
AA42	375	PORT DEP:	LBL 03
AA43	03C	XQ	
AA44	0C9	->A8C9	[ADR_F125]
AA45	1B8	READ 6(N)	
AA46	10E	A=C ALL	
AA47	1F8	READ 7(O)	
AA48	31A	?A<C M	
AA49	013	JNC +02	
AA4A	0BA	A<>C M	
AA4B	25A	C=A-C M	
AA4C	02F	JC +05	
AA4D	228	WRIT 8(P)	parameter passing
AA4E	375	PORT DEP:	LBL 03
AA4F	03C	XQ	
AA50	0C9	->A8C9	[ADR_F125]
AA51	078	READ 1(Z)	
AA52	10E	A=C ALL	
AA53	198	C=M ALL	
AA54	01C	PT=3	
AA55	262	C=C-1@PT	
AA56	158	M=C ALL	
AA57	03C	RCR 3	
AA58	270	RAMSLCT	
AA59	0EE	B<>C ALL	
AA5A	038	READATA	
AA5B	0AE	A<>C ALL	
AA5C	2F0	WRTDATA	
AA5D	0AE	A<>C ALL	
AA5E	068	WRIT 1(Z)	
AA5F	0EE	B<>C ALL	
AA60	266	C=C-1 S&X	
AA61	2E6	?C#0 S&X	
AA62	023	JNC +04	
AA63	365	PORT DEP:	LBL 12
AA64	03C	GO	

	AA65	1D3	->A9D3	[ADR_F200]
	AA66	04E	C=0 ALL	
	AA67	270	RAMSLCT	
	AA68	038	READATA	
	AA69	10E	A=C ALL	
	AA6A	198	C=M ALL	
	AA6B	266	C=C-1 S&X	
	AA6C	158	M=C ALL	
	AA6D	270	RAMSLCT	
	AA6E	038	READATA	
	AA6F	0AE	A<>C ALL	
	AA70	2F0	WRTDATA	
	AA71	0AE	A<>C ALL	
	AA72	028	WRIT 0(T)	
	AA73	198	C=M ALL	
	AA74	10E	A=C ALL	
	AA75	1BC	RCR 11	
	AA76	01C	PT=3	
	AA77	102	A=C@PT	
	AA78	0AE	A<>C ALL	
	AA79	158	M=C ALL	
	AA7A	266	C=C-1 S&X	
	AA7B	2E6	?C#0 S&X	
	AA7C	023	JNC +04	
	AA7D	365	PORT DEP:	lbl 12
	AA7E	03C	GO	
	AA7F	1D3	->A9D3	[ADR_F200]
	AA80	04E	C=0 ALL	
	AA81	0E8	WRIT 3(X)	
	AA82	3C1	?NC GO	4 subroutine-levels are used,
	AA83	002	->00F0	[NFRPU]
F292	AA84	2A0	SETDEC	To shift the integers in the right part of the mantissa field
	AA85	038	READATA	get parameter
	AA86	006	A=0 S&X	
	AA87	39C	PT=0	
	AA88	1A2	A=A-1@PT	
	AA89	366	?A#C S&X	
	AA8A	01B	JNC +03	
	AA8B	3DA	RSHFC M	
	AA8C	3E3	JNC -04	
	AA8D	046	C=0 S&X	
	AA8E	3E0	RTN	
F29B	AA8F	2A0	SETDEC	The reverse operation ( reformatting )
	AA90	0AE	A<>C ALL	get parameter
	AA91	2FA	?C#0 M	
	AA92	3A0	?NC RTN	
	AA93	130	LDI S&X	
	AA94	009	009	
	AA95	35C	PT=12	
	AA96	11A	A=C M	
	AA97	342	?A#0@PT	
	AA98	027	JC +04	

AA99 266 C=C-1 S&X  
 AA9A 3FA LSHFA M  
 AA9B 3E3 JNC -04  
 AA9C 0BA A<>C M  
 AA9D 3E0 RTN  
 F2A8 AA9E 2A0 SETDEC  
 AA9F 0CE C=B ALL  
 AAA0 31A ?A<C M  
 AAA1 013 JNC +02  
 AAA2 0AE A<>C ALL  
 AAA3 08E B=A ALL  
 AAA4 00E A=0 ALL  
 AAA5 01C PT=3  
 AAA6 2E2 ?C#0@PT  
 AAA7 023 JNC +04  
 AAA8 13A A=A+B M  
 AAA9 262 C=C-1@PT  
 AAAA 3E3 JNC -04  
 AAAB 3DC PT=PT+1  
 AAAC 07A A<>B M  
 AAAD 3FA LSHFA M  
 AAAE 07A A<>B M  
 AAAF 154 ?PT=6  
 AAB0 3B3 JNC -10d  
 AAB1 0AE A<>C ALL  
 AAB2 3E0 RTN  
 F2BB AAB3 2A0 SETDEC  
 AAB4 0CE C=B ALL  
 AAB5 23A C=C+1 M  
 AAB6 0EE B<>C ALL  
 AAB7 04E C=0 ALL  
 AAB8 21C PT=2  
 AAB9 3FA LSHFA M  
 AABA 3DC PT=PT+1  
 AABB 33A ?A<B M  
 AABC 3EF JC -03  
 AABD 39A RSHFA M  
 AABE 07A A<>B M  
 AABF 1BA A=A-1 M  
 AAC0 222 C=C+1@PT  
 AAC1 19A A=A-B M  
 AAC2 33A ?A<B M  
 AAC3 3EB JNC -03  
 AAC4 3BA RSHFB M  
 AAC5 3D4 PT=PT-1  
 AAC6 214 ?PT=2  
 AAC7 3DB JNC -05  
 F2CE AAC8 3E0 RTN

Multiplication routine  
get parameter

Quotient-Remainder Routine  
get parameter

Le compte est bon		Le compte est bon		Le compte est bon	
1	14	2	18	3	22
○○○○		○○○○		○○○○	
○○○○		○○○○		○○○○	
Le compte est bon		Le compte est bon		Le compte est bon	
4	331	5	82	6	802
○○○○		○○○○		○○○○	
○○○○		○○○○		○○○○	

## Master Mind.(MCODE version)

JM Baillard – <http://hp41programs.yolasite.com/mastermind.php>

This version allows codes of different length, from 1 to 10 numbers. You determine the length when you enter the first guess, which length is taken as the reference. The program displays "M-N" after your guess to mean that M digits are in right positions and N digits are in wrong positions. (using "\*" & "/" would be difficult to read for more than 6-digit numbers).

The hidden number is computed after your first guess but - of course - independently, except for the number of its digits. After your first guess, all your guesses must have the same number of digits: this program does not check!

The MCODE routine MN? Speeds-up the execution and facilities the gameplay.

**Example:** 1 STO 00 , and suppose you want to play with 7-digit numbers

```
1111111 XEQ "MMD2" >>>> "1111111: 1-0"      there is one "1"

1222222 R/S                "1222222: 0-1"
the "1" is in wrong position and there is no "2"

3133333 R/S                "3133333: 2-1"
2 digits are in right positions, 1 digit is in a wrong position. There are two "3"s

3413444 R/S                "3413444: 2-2"
2 digits are in right positions, 2 digits are in a wrong position there is one "4"

3451355 R/S                "3451355: 1-3"
1 digit is in right position, 3 digits are in a wrong position there is no "5"

3314666 R/S                "3314666: 1-4"
1 digit is in right position, 4 digits are in a wrong position there is one "6"

4613377 R/S                "4613377: 0-5"
4 6 1 3 3 are in a wrong location. There is no "7"

3866431 R/S                "3866431: 2-3"
2 digits are in a right position, 3 in a wrong position. There is no "8"

3936149 R/S                "3936149: 4-3"
4 digits are in a right position, 3 in a wrong position. There are two "9"s

3936914 R/S                "3936914 // 10"
you've found the hidden number in 10 guesses.
```

**FOCAL Program :**

<b>01*LBL "MMIND"</b>	17 *	33 GTO 02
02 4	18 INT	34 "": "
03 XROM "INIT"	19 +	35 ARCL Y
04 RCL Z	20 DSE 02	36 "-"
05 STO 00	21 GTO 00	37 ARCL Z
06 CLA	22 STO 01	38 PROMPT
07 ARCL X	23 RCL 00	39 GTO 01
08 ALENG	24*LBL 01	40*LBL 02
09 STO 02	25 CLA	41 "://"
10 STO 03	26 ARCL X	42 ARCL 02
11 CLX	27 ISG 02	43 FIX 4
12*LBL 00	28 CLX	44 SF 29
13 E1	29 RCL 01	45 AVIEW
14 *	30 MN?	46 END
15 RNG	31 RCL 03	
16 E1	32 X=Y?	

**MCODE Code:**

Header AF7F	0BF	"?"	Mastermind	
Header AF80	00E	"N"	auxiliary fnc	
Header AF81	00D	"M"		
<b>MN?</b>	<b>AF82</b>	<b>2A0</b>	<b>SETDEC</b>	JM Baillard
AF83	0F8	READ 3(X)		
AF84	046	C=0 S&X		
AF85	10E	A=C ALL		
AF86	0B8	READ 2(Y)		
AF87	046	C=0 S&X		
AF88	02E	B=0 ALL		
AF89	35C	PT=12		
AF8A	362	?A#C @PT		
AF8B	057	JC +10d		
AF8C	310	LD@PT- C		
AF8D	3DC	PT=PT+1		
AF8E	0BA	A<>C M		
AF8F	350	LD@PT- D		
AF90	3DC	PT=PT+1		
AF91	0BA	A<>C M		
AF92	0FA	C<>B M		
AF93	23A	C=C+1 M		
AF94	0FA	C<>B M		
AF95	3D4	PT=PT-1		
AF96	2E2	?C#0 @PT		
AF97	39F	JC -13d		
AF98	0EE	C<>B ALL		
AF99	0BC	RCR 5		
AF9A	2DC	PT=13		
AF9B	2E2	?C#0 @PT		
AF9C	01B	JNC +03		

AF9D	33C	RCR 1	
AF9E	226	C=C+1 S&X	
AF9F	0E8	WRIT 3(X)	#of well-placed digits
AFA0	04E	C=0 ALL	
AFA1	0EE	C<>B ALL	
AFA2	130	LDI S&X	
AFA3	013	CON: 19	
AFA4	3D4	PT=PT-1	
AFA5	2E2	?C#0 @PT	
AFA6	04F	JC +09	
AFA7	0EE	C<>B ALL	
AFA8	0BC	RCR 5	
AFA9	2FE	?C#0? MS	
AFAA	01B	JNC +03	
AFAB	33C	RCR 1	
AFAC	226	C=C+1 S&X	
AFAD	0A8	WRIT 2(Y)	#digits in wrong position
AFAE	3E0	RTN	
AFAF	362	?A#C @PT	
AFB0	057	JC +0A	
AFB1	310	LD@PT- C	
AFB2	3DC	PT=PT+1	
AFB3	0BA	A<>C M	
AFB4	350	LD@PT- D	
AFB5	3DC	PT=PT+1	
AFB6	0BA	A<>C M	
AFB7	0EE	C<>B ALL	
AFB8	23A	C=C+1 M	
AFB9	0EE	C<>B ALL	
AFBA	0AE	A<>C ALL	
AFBB	2FC	RCR 13	
AFBC	0AE	A<>C ALL	
AFBD	266	C=C-1 S&X	
AFBE	327	JC -1C	
AFBF	383	JNC -10	



**Random Number Generator Using the Time Module:**

Header AFC0	087	"G"	Random Number
Header AFC1	00E	"N"	with Timer
Header AFC2	012	"R"	
<b>RNG AFC3</b>	<b>18C</b>	<b>?FSET 11</b>	Jean-Marc Baillard
AFC4	3B5	?C XQ	Stack lift
AFC5	051	->14ED [	R_SUB]
AFC6	130	LDI S&X	
AFC7	010	CON:	User memory must be deselected
AFC8	270	RAM SLCT	by selecting non-existent RAM
AFC9	130	LDI S&X	( at 010h )
AFCA	0FB	CON :	then the Timer is selected
AFCB	3F0	PRPH SLCT	
AFCC	3E8	WRIT 15(e)	Set the A/B pointer to A
AFCD	038	READATA	
AFCE	1BC	RCR 11	Rotates C-register 11 digits right
AFCF	046	C=0 S&X	Chp0 is selected again
AFD0	270	RAM SLCT	
AFD1	05E	C=0 MS	
AFD2	130	LDI S&X	
AFD3	041	041	
AFD4	2A0	SETDEC	
AFD5	10E	A=C ALL	
AFD6	04E	C=0 ALL	
AFD7	35C	PT=12	
AFD8	1D0	LD@PT- 7	
AFD9	210	LD@PT- 8	
AFDA	110	LD@PT- 4	
AFDB	050	LD@PT- 1	
AFDC	090	LD@PT- 2	7,841298787
AFDD	250	LD@PT- 9	
AFDE	190	LD@PT- 6	
AFDF	1D0	LD@PT- 7	
AFE0	210	LD@PT- 8	
AFE1	1D0	LD@PT- 7	
AFE2	044	CLRF 4	
AFE3	070	N=C ALL	
AFE4	171	?NC XQ	C = A mod C
AFE5	064	195C	
AFE6	10E	A=C ALL	
AFE7	0B0	C=N ALL	
AFE8	261	?NCXQ	C = A/C
AFE9	060	1898	
AFEA	0E8	WRIT 3(X)	
AFEB	3E0	RTN	

## High Rollers. (MCODE version)

Ross Cooling – PPCCJ V14N2 p21 ; (February 1987)

After working hard at MCODE it's good to take a break . I liked playing the user language HP-41 version of High Rollers ( PPCJ V6N7P47, that's right - way back in October 1979 ) to help me relax It's based on an old TV game where you start off with the digits 1 to 9. You roll a pair of dice and try to remove a digit(s ) adding up to your dice throw total. Anytime you roll a double you get an insurance marker to use when you can't remove totaling your throw (by entering a zero). The objective is to remove all 9 digits and you win !

The problem was that it took a few seconds between moves, which detracted from the enjoyment. Like most user language programs it also disturbed the stack, and some user registers. When I got into MCODE I decided I'd write an MCODE version. I started about a year ago and got most of it done in about a week, except for three things. Getting some of the displays to work properly required some extra reading and experimentation. Trying to find enough internal registers to retail all I needed to keep was the second problem. This just needed some slight rearranging and changes to my algorithm, and use of some of the status registers. The toughest part was getting the digits to remove reply without disturbing the previous display until necessary. For this I used some code in Ken Emery's HP-41 MCODE for Beginners in a routine called HXENTRY. Even with it I still couldn't get it to work properly, until just a short while ago when I got back into looking at it. It's still not perfect, but it's quite acceptable to me!

The program follows the user language version for the most part as far as the displays and logic goes, correcting some shortcomings. Of course it doesn't touch the stack or user registers, and it's about 20 times faster! It does use the Q register, P(13:8), and the "a" register. The routine is non-programmable so it's safe to clobber the subroutine stack in a, and I also zero out most of user register "b" so any values in "a" won't be used accidentally.

The program contains some comments, but what follows is the high-level version.

Display "HIGH ROLLERS" and pause  
#WON = 0, #LOST = 0

[TOP]

#INS = 0  
Set all digits "on"

[NEXT]

Get pseudo-random #1 and store it  
Get pseudo-random #2 and store it  
If die#1 = die#1  
#INS = #INS + 1

[DISPLAY]

Send die#1 to display and a comma  
Send die#2 to display and a comma  
If #INS = 0  
Send a space to display  
Else



Send #INS to display and a colon

[DIGITS]

For each digit 1 to 9

If digit "on"

Send digit to display

Else

Send a space to display

Show the display and wait for keypress

Show a zero and an underscore in display

Accept 0 to 9, ←, R/S, or ON

If ON

Quit and turn off

If 0

If #INS = 0

Go to [LOSE]

Else

#INS = #INS - 1

Go to [NEXT]

Else

If numbers not = die#1 + die#2

Go to [DISPLAY]

Else

Set toper digits "off"

If any digits already "off"

Go to [DISPLAY]

If all digits "off"

Go to [WIN]

Else

Go to [NEXT]

[WIN]

#WON = #WON + 1

Display "YOU WON" and pause

[SCORE]

Display "YOU ARE", #WON, ":", #LOST

pause

display "CONTINUE? 1:NO"

if keypress = 1

exit

else

go to [TOP]

[LOSE]

#LOST = #LOST + 1

Display "YOU LOSE" and pause

Go to [SCORE]

ROLLERS was written using an HP-41DX and the only possible dependency is the call to 5AB6 to get the time for use as a pseudo-random number. It has been published elsewhere, but I'll just show enough code necessary for ROLLERS (feel free to substitute your own if desired).

Registers used:

User P(13:8) – last pseudo-random number

User Q – answer of digit(s):

N (and User "a" temporarily for a copy)

N[0:2] nine bits for digits 1 to 9

N{3} four bits for #INS

N[4] three bits for die#1

N[5] three bits for die#2

N[6] four bits for #LOST

N[7] four bits for #WON

Due to display size once either #WON or #LOST would become 10, it will reset the values so they'll always be one decimal digit.

Due to the large size of this program, and the fact that it's MCODE, if anyone wants a ROM>REG image on cards, ad a fuller annotated listing, send \$1.00 (Canadian in Canada, US elsewhere) and 4 (four) cards to me. I hope you enjoy this program, and if you have any comments or improvements (especially on receiving the answer without throwing away the first keypress) please write!

Ross Cooling (#12433)  
R.R. #1  
Kimberly, Ontario  
Canada N0C 1G0

RNG	ADEA	2A0	SETDEC
ADEB	04E	C=0 ALL	
ADEC	35C	PT= 12	
ADED	250	LD@PT- 9	
ADEE	210	LD@PT- 8	
ADEF	090	LD@PT- 2	982,1
ADF0	050	LD@PT- 1	
ADF1	39C	PT= 0	
ADF2	0D0	LD@PT- 3	
ADF3	135	?NC XQ	
ADF4	060	->184D	[MP2_10]
ADF5	10E	A=C ALL	
ADF6	04E	C=0 ALL	
ADF7	266	C=C-1 S&X	
ADF8	35C	PT= 12	
ADF9	090	LD@PT- 2	
ADFA	050	LD@PT- 1	0,211327
ADFB	050	LD@PT- 1	
ADFC	0D0	LD@PT- 3	
ADFD	090	LD@PT- 2	
ADFE	1D0	LD@PT- 7	
ADFF	01D	?NC XQ	

	AE00	060	->1807	[AD2_10]
	AE01	3C4	ST=0	
	AE02	0ED	?NC XQ	
	AE03	064	->193B [	INTFRC]
	AE04	10E	A=C ALL	
	AE05	3E0	RTN	
Header	AE06	093	"S"	
Header	AE07	012	"R"	
Header	AE08	005	"E"	High Rollers
Header	AE09	00C	"L"	
Header	AE0A	00C	"L"	
Header	AE0B	00F	"O"	PPCCJ V14N2P21
Header	AE0C	012	"R"	Ross Cooling
	<b>ROLLERS</b>	<b>000</b>	<b>NOP</b>	not programmable
	AE0E	338	READ 12(b)	
	AE0F	33C	RCR 1	
	AE10	05A	C=0 M	Clear return stack
	AE11	2FC	RCR 13	to use register a
	AE12	328	WRIT 12(b)	
	AE13	3D9	?NC XQ	Enable Display
	AE14	01C	->07F6	[ENLCD]
	AE15	3BD	?NC XQ	Display Message
	AE16	01C	->07EF	[MESSL]
	AE17	008	"H"	
	AE18	009	"I"	
	AE19	007	"G"	
	AE1A	008	"H"	
	AE1B	020	" "	
	AE1C	012	"R"	"HIGH ROLLERS"
	AE1D	00F	"O"	
	AE1E	00C	"L"	
	AE1F	00C	"L"	
	AE20	005	"E"	
	AE21	012	"R"	
	AE22	213	"S"	
	AE23	149	?NC XQ	Enable RAM
	AE24	024	->0952	[ENCP00]
	AE25	1F9	?NC XQ	View Display
	AE26	00C	->037E [	STMSGF]
	AE27	046	C=0 S&X	
	AE28	266	C=C-1 S&X	pause before showing
	AE29	2E6	?C#0 S&X	next display msg
	AE2A	3F7	JC -02	
	AE2B	2D9	?NC XQ	Calls
	AE2C	168	->5ABC	[TIME]
	AE2D	149	?NC XQ	Enable RAM
	AE2E	024	->0952 [	ENCP00]
	AE2F	158	M=C ALL	
	AE30	04E	C=0 ALL	#won=0, #lost=0
	AE31	070	N=C ALL	
	AE32	198	C=M ALL	
	AE33	2A0	SETDEC	

AE34	13C	RCR 8	
AE35	23E	C=C+1 MS	
AE36	33C	RCR 1	
AE37	05E	C=0 MS	
AE38	046	C=0 S&X	
AE39	266	C=C-1 S&X	
AE3A	10E	A=C ALL	
AE3B	3EE	LSHFA ALL	
AE3C	0E0	SLCT Q	
AE3D	2DC	PT= 13	
AE3E	0A0	SLCT P	
AE3F	11C	PT= 8	
AE40	238	READ 8(P)	
AE41	0B2	A<>C P-Q	
AE42	228	WRIT 8(P)	save random number
<b>NEWGAME</b>	<b>0B0</b>	<b>C=N ALL</b>	get saved value
AE44	01C	PT= 3	
AE45	042	C=0 @PT	#ins.=0
AE46	130	LDI S&X	
AE47	1FF	CON: 511	set on digit bits
AE48	070	N=C ALL	save values
<b>RETHROW</b>	<b>238</b>	<b>READ 8(P)</b>	recall random number
AE4A	3CE	RSHFC ALL	
AE4B	2A0	SETDEC	
AE4C	046	C=0 S&X	
AE4D	266	C=C-1 S&X	
AE4E	108	SETF 8	
AE4F	10E	A=C ALL	
AE50	260	SETHX	
AE51	379	PORT DEP:	Call RNG, 0<=C<1
AE52	03C	XQ	(Returns in DEC)
AE53	1EA	->ADEA	[RNG]
AE54	0AE	A<>C ALL	
AE55	158	M=C ALL	save result in M
AE56	10E	A=C ALL	
AE57	04E	C=0 ALL	
AE58	35C	PT= 12	C = 6
AE59	190	LD@PT- 6	
AE5A	135	?NC XQ	Multiply by 6
AE5B	060	->184D	[MP2_10]
AE5C	088	SETF 5	do integer portion
AE5D	0ED	?NC XQ	Take Integer
AE5E	064	->193B	[INTFRC]
AE5F	35C	PT= 12	
AE60	222	C=C+1 @PT	add 1 to C, 1<=C<=6
AE61	13C	RCR 8	rotate to proper spot
AE62	05C	PT= 4	
AE63	102	A=C @PT	
AE64	0B0	C=N ALL	
AE65	0A2	A<>C @PT	save dice@1 in C[4]
AE66	070	N=C ALL	resave values
AE67	198	C=M ALL	recall random number

AE68	108	SETF 8	
AE69	10E	A=C ALL	
AE6A	260	SETHX	
AE6B	379	PORT DEP:	Call RNG, 0<=C<1
AE6C	03C	XQ	(Returns in DEC)
AE6D	1EA	->ADEA	[RNG]
AE6E	0AE	A<>C ALL	
AE6F	158	M=C ALL	save result in M
AE70	0AE	A<>C ALL	
AE71	3EE	LSHFA ALL	
AE72	0E0	SLCT Q	
AE73	2DC	PT= 13	
AE74	0A0	SLECT P	
AE75	11C	PT= 8	
AE76	238	READ 8(P)	
AE77	0B2	A<>C P-Q	
AE78	228	WRIT 8(P)	
AE79	198	C=M ALL	
AE7A	10E	A=C ALL	
AE7B	04E	C=0 ALL	
AE7C	35C	PT= 12	
AE7D	190	LD@PT- 6	
AE7E	135	?NC XQ	Multiply by 6
AE7F	060	->184D	[MP2_10]
AE80	088	SETF 5	do integer portion
AE81	0ED	?NC XQ	Take Integer
AE82	064	->193B	[INTFRC]
AE83	35C	PT= 12	
AE84	222	C=C+1 @PT	add 1 to C, 1<=C<=6
AE85	2BC	RCR 7	rotate to proper spot
AE86	09C	PT= 5	
AE87	102	A=C @PT	
AE88	0B0	C=N ALL	
AE89	0A2	A<>C @PT	save dice #2 in C[5]
AE8A	102	A=C @PT	
AE8B	38E	RSHFA ALL	shift dice#2 value
AE8C	05C	PT= 4	
AE8D	362	?A#C @PT	compare dice#1 to dice#2
AE8E	01F	JC +03	
AE8F	01C	PT= 3	
AE90	222	C=C+1 @PT	if = increment #ins.
AE91	070	N=C ALL	resave values
AE92	2E8	WRIT 11(a)	also save in user a
<b>TURN</b>	<b>AE93</b>	<b>3D9</b>	<b>?NC XQ</b>
AE94	01C	->07F6	[ENLCD]
AE95	0B0	C=N ALL	retrieve values
AE96	00E	A=0 ALL	
AE97	05C	PT= 4	
AE98	102	A=C @PT	get dice#1
AE99	0AE	A<>C ALL	
AE9A	07C	RCR 4	
AE9B	31C	PT= 1	convert to ASCII

AE9C	3D0	LD@PT- F	and display comma
AE9D	3E8	WRABC1R	display them
AE9E	09C	PT= 5	
AE9F	0A2	A<>C @PT	get dice#2
AEA0	102	A=C @PT	
AEA1	0BC	RCR 5	
AEA2	31C	PT= 1	convert to ASCII
AEA3	2D0	LD@PT- B	and display colon
AEA4	3E8	WRABC1R	display them
AEA5	130	LDI S&X	presume @ins.=0
AEA6	020	CON: 32	so prepare a space
AEA7	01C	PT= 3	
AEA8	342	?A#0 @PT	test #ins.
AEA9	033	JNC +06	jump if zero
AEAA	0A2	A<>C @PT	get #ins.
AEAB	102	A=C @PT	
AEAC	03C	RCR 3	
AEAD	31C	PT= 1	convert to ASCII
AEAE	2D0	LD@PT- B	and display colon
AEAF	3E8	WRABC1R	display #ins.
AEB0	086	B=A S&X	store digit bits
AEB1	2A0	SETDEC	countdown of
AEB2	1BE	A=A-1 MS	number of digits
AEB3	260	SETHX	to check
AEB4	05E	C=0 MS	
AEB5	23E	C=C+1 MS	digit to test
AEB6	1BE	A=A-1 MS	
AEB7	0C6	C=B S&X	only bit in C[2]
AEB8	2F6	?C#0 XS	is digit or blank?
AEB9	02F	JC +05 j	ump to show a digit
AEBA	130	LDI S&X	
AEBB	020	CON: 32	
AEBC	3E8	WRABC1R	display a blank
AEBD	03B	JNC +07	
AEBE	2FC	RCR 13	rotate C[13] to C[0]
AEBF	31C	PT= 1	
AEC0	0D0	LD@PT- 3	
AEC1	056	C=0 XS	change to ASCII
AEC2	3E8	WRABC1R	display digit
AEC3	33C	RCR 1	rotate back to C[13]
AEC4	0C6	C=B S&X	get last digit
AEC5	056	C=0 XS	drop bit for digit
AEC6	1E6	C=C+C S	&X get next bit
AEC7	0E6	C<>B S&X	store digit bits
AEC8	0C6	C=B S&X	
AEC9	35E	?A#0 MS	end of digit checks?
AECA	35F	JC -15	jump to top of loop
AECB	149	?NC XQ	Enable RAM
AECC	024	->0952	[ENCP00]
AECD	3C8	CLRKEY	
AECE	3CC	?KEY	wait for a keypress
AECF	3FB	JNC -01	



AED0	04E	C=0 ALL	
AED1	268	WRIT 9(Q)	save answer in Q
AED2	3C1	?NC XQ	set-up disp.
AED3	0B0	->2CF0	[CLLCDE]
AED4	31C	PT= 1	force first key
AED5	046	C=0 S&X	to show as a zero
AED6	09B	JNC +13	
<b>PRMPT AED7</b>	115	?NC XQ	Partial Data Entry
AED8	038	->0E45	[NEXT1]
CANCEL023	JNC +04		jump if backarrow
ANYKEY	04C	?FSET 4	top two rows (A-J)?
AEDB	3E7	JC -04	ignore if so
AEDC	04B	JNC +09	go to further tests
AEDD	3B8	RDABC1R	del. Last from display
AEDE	149	?NC XQ	Enable RAM
AEDF	024	->0952	[ENCP00]
AEEO	278	READ 9(Q)	get answer so far
AEE1	04A	C=0 PT<-	blank out last digit
AEE2	23C	RCR 2	
AEE3	268	WRIT 9(Q)	resave answer
AEE4	083	JNC +10	go to next prompt
AEE5	00C	?FSET 3	was key a digit?
AEE6	08B	JNC +11	jump to next test
AEE7	0BE	A<>C MS	
AEE8	2FC	RCR 13	
AEE9	0D0	LD@PT- 3	
AEEA	368	WRAB1R	write digit to disp.
AEEB	058	G=C @PT,+	
AEEC	149	?NC XQ	Enable RAM
AEED	024	->0952	[ENCP00]
AEEE	278	READ 9(Q)	get answer so far
AEEF	37C	RCR 12	
AEF0	098	C=G @PT,+	append latest resp.
AEF1	268	WRIT 9(Q)	resave answer
AEF2	042	C=0 @PT	
AEF3	058	G=C @PT,+	
AEF4	3D9	?NC XQ	
AEF5	01C	->07F6	[ENLCD]
AEF6	30B	JNC -31d	go for next prompt
AEF7	28C	?FSET 7	test if dec. point
AEF8	2FF	JC -21	ignore and re-prompt
AEF9	130	LDI S&X	
AEFA	370	CON: 880	test if R/S pressed
AEFB	106	A=C S&X	
AEFC	0B0	C=N ALL	get keycode
AEFD	366	?A#C S&X	ignore other keys
AEFE	2CF	JC -27	and re-prompt
AEFF	3D9	?NC XQ	Clear Display
AF00	0B0	->2CF6	[CLRLCD]
AF01	261	?NC XQ	Reset Keys
AF02	000	->0098	[RSTKB]
AF03	149	?NC XQ	Enable RAM

AF04	024	->0952	[ENCP00]
AF05	215	?NC XQ	Reset Flags
AF06	00C	->0358	[RSTSQ]
AF07	260	SETHX	
AF08	130	LDI S&X	set max. answers
AF09	007	CON: 7	counter in A[13]
AF0A	33C	RCR1	
AF0B	11E	A=C MS	initialize answer
AF0C	006	A=0 S&X	total in A[1:0]
AF0D	278	READ 9(Q)	get answers
AF0E	0A0	SLCT P	
AF0F	39C	PT= 0	
AF10	0E0	SLCT Q	
AF11	31C	PT= 1	convert to ASCII
AF12	042	C=0 @PT	value to binary
AF13	152	A=A+C P-Q	add digit
AF14	23C	RCR 2	rotate to next digit
AF15	1BE	A=A-1 MS	decrement counter
AF16	35E	?A#0 MS	looped 7 times?
AF17	3DF	JC -05	if not go back
AF18	2F8	READ 11(a)	get values
AF19	070	N=C ALL	save for redisplay
AF1A	342	?A#0 @PT	test if total>15
AF1B	07F	JC +15d	jump to redisplay
AF1C	0BC	RCR 5	
AF1D	11E	A=C MS	A[13]= dice#1 value
AF1E	33C	RCR 1	
AF1F	15E	A=A+C MS	add dice#2 value
AF20	0BE	A<>C MS	
AF21	2FC	RCR 13	rotate dice total
AF22	39C	PT= 0	to C[0]
AF23	342	?A#0 @PT	test if answer is 0
AF24	04B	JNC +09	
AF25	362	?A#C @PT	dice total =
AF26	027	JC +04	answer total?
AF27	369	PORT DEP:	
AF28	03C	GO	
AF29	39D	->AF9D	[LAST]
AF2A	369	PORT DEP:	redisplay values
AF2B	03C	GO	if totals don't match
AF2C	293	->AE93	[TURN]
AF2D	0B0	C=N ALL	get values back
AF2E	01C	PT= 3	point to #ins.
AF2F	2E2	?C#0 @PT	test if zero
AF30	033	JNC +06	jump to YOULOSE rtn
AF31	262	C=C-1 @PT	else decrement #ins.
AF32	070	N=C ALL	resave values
AF33	369	PORT DEP:	Back for next turn
AF34	03C	GO	
AF35	249	->AE49	[RETHROW]
AF36	15C	PT= 6	
AF37	2A0	SETDEC	

AF38	222	C=C+1 @PT	
AF39	023	JNC +04	
AF3A	222	C=C+1 @PT	increase #lost
AF3B	29C	PT= 7	
AF3C	042	C=0 @PT	
AF3D	260	SETHEX	
AF3E	070	N=C ALL	
AF3F	3D9	?NC XQ	Enable RAM
AF40	01C	->07F6	[ENLCD]
AF41	3BD	?NC XQ	Display msg.
AF42	01C	->07EF	[MESSL]
AF43	019	"Y"	
AF44	00F	"O"	
AF45	015	"U"	
AF46	020	" "	"YOU LOSE"
AF47	00C	"L"	
AF48	00F	"O"	
AF49	013	"S"	
AF4A	205	"E"	
<b>YLOSE AF4B 3DD</b>		?NC XQ	Left-justify Display
AF4C	0AC	->2BF7	[LEFTJ]
AF4D	046	C=0 S&X	
AF4E	266	C=C-1 S&X	pause before
AF4F	2E6	?C#0 S&X	showing next
AF50	3F7	JC -02	
AF51	3D9	?NC XQ	Enable RAM
AF52	01C	->07F6	[ENLCD]
AF53	3BD	?NC XQ	Display msg.
AF54	01C	->07EF	[MESSL]
AF55	019	"Y"	
AF56	00F	"O"	
AF57	015	"U"	
AF58	020	" "	"YOU ARE "
AF59	001	"A"	
AF5A	012	"R"	
AF5B	005	"E"	
AF5C	220	" "	
AF5D	0B0	C=N ALL	
AF5E	2BC	RCR 7	put #won in C[0]
AF5F	31C	PT= 1	convert to ASCII
AF60	2D0	LD@PT- B	with colon
AF61	3E8	WRABC1R	write #won to disp.
AF62	046	C=0 S&X	
AF63	0D0	LD@PT- 3	put #lost in C[0]
AF64	2FC	RCR 13	convert to ASCII
AF65	3E8	WRABC1R	write #lost to disp.
AF66	130	LDI S&X	
AF67	020	CON: 32	make an ASCII space
AF68	3E8	WRABC1R	write to display 2x
AF69	3E8	WRABC1R	to fill display
AF6A	149	?NC XQ	Enable RAM
AF6B	024	->0952	[ENCP00]

AF6C	1F9	?NC XQ	View Display
AF6D	00C	->037E	[STMSGF]
AF6E	046	C=0 S&X	
AF6F	266	C=C-1 S&X	pause before
AF70	2E6	?C#0 S&X	showing next msg.
AF71	3F7	JC -02	
AF72	3D9	?NC XQ	Enable RAM
AF73	01C	->07F6	[ENLCD]
AF74	3BD	?NC XQ	Display msg.
AF75	01C	->07EF	[MESSL]
AF76	003	"C"	
AF77	00F	"O"	
AF78	00E	"N"	
AF79	014	"T"	
AF7A	009	"I"	
AF7B	00E	"N"	"CONTINUE?1:NO"
AF7C	015	"U"	
AF7D	005	"E"	
AF7E	03F	"?"	
AF7F	0B1	"1:"	
AF80	00E	"N"	
AF81	20F	"O"	
AF82	149	?NC XQ	Enable RAM
AF83	024	->0952	[ENCP00]
AF84	1F9	?NC XQ	View Display
AF85	00C	->037E	[STMSGF]
AF86	3C8	CLRKEY	
AF87	3CC	?KEY	wait for a keypress
AF88	3FB	JNC -01	
AF89	220	C=KEY KY	get key pressed
AF8A	130	LDI S&X	
AF8B	036	CON: 54	check for a 1
AF8C	0A6	A<>C S&X	save in A[2:0]
AF8D	03C	RCR 3	make key to C[2:0]
AF8E	056	C=0 XS	
AF8F	366	?A#C S&X	test if key=1
AF90	023	JNC +04	jump if not
AF91	369	PORT DEP:	Play another game
AF92	03C	GO	
AF93	243	->AE43	[NEWGAME]
AF94	3D9	?NC XQ	Clear Display
AF95	0B0	->2CF6	[CLRLCD]
AF96	261	?NC XQ	Reset Keys
AF97	000	->0098	[RSTKB]
AF98	149	?NC XQ	Enable RAM
AF99	024	->0952	[ENCP00]
AF9A	215	?NC XQ	Reset Flags
AF9B	00C	->0385	[RSTSQ]
AF9C	3E0	RTN	end of the game!
<b>LAST</b>	<b>AF9D</b>	<b>278</b>	<b>READ 9(Q)</b> get answer
AF9E	39C	PT= 0	
AF9F	0EE	C<>B ALL	save in B

AFA0	0B0	C=N ALL	get value
AFA1	058	G=C @PT,+	
AFA2	106	A=C S&X	save digits in A
AFA3	2CE	?B#0 ALL	
AFA4	18B	JNC +31	no more answers?
AFA5	2C2	?B#0 @PT	
AFA6	027	JNC +04	non-zero answers?
AFA7	3AE	RSHFB ALL	move next answer
AFA8	3AE	RSHFB ALL	into position
AFA9	3D3	JNC -06	
AFAA	0A2	A<>C @PT	
AFAB	250	LD@PT- 9	
AFAC	39C	PT= 0	reverse mask
AFAD	0A2	A<>C @PT	
AFAE	182	A=A-B @PT	
AFAF	130	LDI S&X	
AFB0	001	CON: 1	
AFB1	342	?A#0 @PT	
AFB2	023	JNC +04	
AFB3	1A2	A=A-1 @PT	
AFB4	1E6	C=C+C S&X	
AFB5	3E3	JNC -04	
AFB6	0A6	A<>C S&X	
AFB7	098	C=G @PT,+	
AFB8	0A6	A<>C S&X	if digit set
AFB9	3B0	C=C AND A	C=1, else 0
AFBA	2E6	?C#0 S&X	
AFBB	027	JC +04	
AFBC	369	PORT DEP:	redisplay values
AFBD	03C	GO	
AFBE	293	->AE93	[TURN]
AFBF	0B0	C=N ALL	
AFC0	0A6	A<>C S&X	
AFC1	106	A=C S&X	
AFC2	250	LD@PT- 9	reverse mask
AFC3	39C	PT= 0	
AFC4	0A2	A<>C @PT	
AFC5	182	A=A-B @PT	
AFC6	130	LDI S&X	
AFC7	001	CON: 1	
AFC8	342	?A#0 @PT	
AFC9	023	JNC +04	
AFCA	1A2	A=A-1 @PT	
AFCB	1E6	C=C+C S&X	
AFCC	3E3	JNC -04	
AFCD	2A6	C=-C-1 S&X	
AFCE	0A6	A<>C S&X	
AFCF	098	C=G @PT,+	
AFD0	0A6	A<>C S&X	
AFD1	3B0	C=C AND A	clear digit
AFD2	106	A=C S&X	
AFD3	058	C=G @PT,+	

AFD4	29B	JNC -2D	process next answer
AFD5	0B0	C=N ALL	
AFD6	0A6	A<>C S&X	
AFD7	2E6	?C#0 S&X	all digits gone?
AFD8	02B	JNC +05	go to YOUWIN
AFD9	070	N=C ALL	else
AFDA	369	PORT DEP:	Rethrow dice
AFDB	03C	GO	
AFDC	249	->AE49	[RETHROW]
AFDD	29C	PT= 7	
AFDE	2A0	SETDEC	
AFDF	222	C=C+1 @PT	add 1 to #won
AFE0	023	JNC +04	jump if not?9
AFE1	222	C=C+1 @PT	else, set #won to 1
AFE2	15C	PT= 6	
AFE3	042	C=0 @PT	and #lost to 0
AFE4	260	SETHX	
AFE5	070	N=C ALL	
AFE6	3D9	?NC XQ	Enable RAM
AFE7	01C	->07F6	[ENLCD]
AFE8	3BD	?NC XQ	Display msg.
AFE9	01C	->07EF	[MESSL]
AFEA	019	"Y"	
AFEB	00F	"O"	
AFEC	015	"U"	"YOU WIN"
AFED	020	" "	
AFEE	017	"W"	
AFEF	009	"I"	
AFF0	20E	"N"	
AFF1	369	PORT DEP:	
AFF2	03C	GO	
AFF3	34B	->AF4B	[YLOSE]

*Ed's note:*

By now you have no doubt realize that MCODE gaming is *\*hard\** - just comparing the number of steps in the FOCAL version of High Rollers (see FOCAL sections) is enough to make you understand why so few exist and so far and in between these jewels are...

On the other hand, some games don't need to be written on MCODE to be real masterpieces, including the execution speed and user interface options.

## Poker for the HP-41 (MCODE version)

Jean-Marc Baillard - <http://hp41programs.yolasite.com/poker.php>

This version combines three different cases:

1. DRAW: Play against your HP-41
2. BANDIT: Play with your HP-41 (One-armed Bandit)
3. TEXAS: Texas Hold'em Power

The FOCAL driver program below sets the appropriate flags, resets the SIZE and prompt for the total money amount for betting. It then transfers the execution to the main program "POKER"

<u>01*LBL "TEXAS"</u>	09 CF 07	17 STO 00
02 SF 08	10 CF 08	18 "CA\$H=?"
03 GTO 00	11*LBL 00	19 PROMPT
<u>04*LBL "BANDIT"</u>	12 SIZE?	20 STO 41
05 SF 07	13 42	21 XROM "POKER"
06 CF 08	14 X>Y?	22 END
07 GTO 00	15 PSIZE	
<u>08*LBL "DRAW"</u>	16 RNG	

Several MCODE routines facilitate the program flow and accelerate the execution of the FOCAL driver programs, so this is a hybrid combination of both. The MCODE routines check that the registers exist, and when appropriate, that they don't hold ALPHA data.

1. The first one "**TRI**" sorts the 5 cards in R01 thru R05, evaluate the poker hand - the value is returned in X -, compares this result with the number already in R48 and stores the maximum of these 2 numbers into R48.
2. The second M-Code routine "**MV**" is equivalent to 17.001005 REGMOVE  
Checking for alpha data would be unuseful in this case.

Furthermore, these two routines have been combined into a single one called "**SHFL**", which performs the desired action depending on the contents of the X register on input. This saves one entry in the FAT and runs slightly faster:

```

If X = 0 "#$#" is equivalent to TRI
If X < 0 "#$#" ----- MV
If X > 0 "#$#" ----- TRI + MV

```

3. And the 3rd routine "**CARD**" takes a random number in R00 to calculate another random number which is stored in R00 and returns a "card", i-e an integer between 28 & 40 in register X

### Instructions:

*You should refer to the POKER Game in the FOCAL Games Section first.*

The instructions are almost identical to the FOCAL program case, except that the program does not stop before your next cash is displayed. After displaying your cards, GETKEY is used to place your bet ( draw poker or Texas hold'em ):

[S+] does a bet of about 1,668 \$      - [LN] does a bet of about 3,103 \$  
 [X<>Y] does bet of about 6,081 \$      - [TAN] does a bet of about 8,619 \$  
 [SHIFT] does a bet of about 13,253 \$ - [SST] does a bet of about 16,894 \$  
 .....  
 [ / ] does a bet of about 90,480 \$      - [R/S] does a bet of about 97,307 \$

More exactly, the formula is:  $Bet = [ ( \text{keycode} ) \text{Ln}(41) ]^2$

There are, however, 2 exceptions: press ENTER^ to accept the HP41's raise and press the back-arrow key if you want to fold.

Likewise with CF 08 - when you want to discard some cards, say the cards n° 1 3 5 ( seen from the left ), simply press 1 3 5 ENTER^ (not too quickly) - the program doesn't stop.

With CF 07 & CF 08 (one-arm bandit), your bet is randomly evaluated.

### Notes:

- The HP-41 does not bluff when playing "Texas Hold'em Poker" i-e SF 08
- If you want this option, add for instance RCL 49 \* after line 190  
     add RCL 00 R-D FRC STO 00 .12 X>Y? ST/ 49 after line 110  
     and add 1 STO 49 after line 33

With respect to execution time, "RIVER. . ." is now displayed for 7 or 8 seconds instead of 37s . It's surely possible to save bytes if you create loops inside LBL 17& LBL 18, but this would increase execution time...

Header A03B	0A3	"#"	
Header A03C	024	"\$"	Shuffle
Header A03D	023	"#"	JM Baillard
<b>SHFL</b>	<b>A03E</b>	<b>378</b>	<b>READ 13(c)</b>
A03F	03C	RCR 3	
A040	106	A=C S&X	
A041	130	LDI S&X	
A042	1D0	1D0h	
A043	306	?A<C S&X	
A044	381	?NC GO	if R48 doesn't exist
A045	00A	->02E0	[ERRNE]
A046	0F8	READ 3(X)	
A047	2FE	?C#0 MS	
A048	023	JNC +04	
A049	369	PORT DEP:	GOTO the 2nd part
A04A	03C	GO	of the routine if X < 0
A04B	1EE	->A1EE	[LB_FED8]



A04C	104	CLRF 8	A079	0AE	A<>C ALL
A04E	2EE	>C#0	A07A	2F0	WRTDATA
A04E	017	JC +02	A07B	198	C=M ALL
A04F	108	SETF 8	A07C	270	RAMSLCT
A050	130	LDI S&X	A07D	0AE	A<>C ALL
A051	005	005	A07E	2F0	WRTDATA
A052	146	A=A+C S&X	A07F	0B0	C=N ALL
A053	266	C=C-1 S&X	A080	2E6	?C#0 S&X
A054	0E6	B<>C S&X	A081	32F	JC -27d
A055	0A6	A<>C S&X	A082	04E	C=0 ALL
A056	270	RAMSLCT	A083	270	RAMSLCT
A057	0A6	A<>C S&X	A084	168	WRIT 5(M)
A058	038	READATA	A085	35C	PT=12
A059	0E6	B<>C S&X	A086	110	LD@PT- 4
A05A	270	RAMSLCT	A087	1A8	WRIT 6(N)
A05B	0E6	B<>C S&X	A088	0B8	READ 2(Y)
A05C	2F0	WRTDATA	A089	268	WRIT 9(Q)
A05D	1A6	A=A-1 S&X	A08A	0F8	READ 3(X)
A05E	0E6	B<>C S&X	A08B	228	WRIT 8(P)
A05F	266	C=C-1 S&X	A08C	138	READ 4(L)
A060	01F	JC +03	A08D	1E8	WRIT 7(O)
A061	0E6	B<>C S&X	A08E	046	C=0 S&X
A062	39B	JNC -13d	A08F	270	RAMSLCT
A063	130	LDI S&X	A090	038	READATA
A064	005	005	A091	158	M=C ALL
A065	070	N=C ALL	A092	10E	A=C ALL
A066	0EE	B<>C ALL	A093	078	READ 1(Z)
A067	00E	A=0 ALL	A094	070	N=C ALL
A068	0CE	C=B ALL	A095	36E	?A#C ALL
A069	266	C=C-1 S&X	A096	18B	JNC +49d
A06A	057	JC +10d	A097	10E	A=C ALL
A06B	270	RAMSLCT	A098	0B8	READ 2(Y)
A06C	0EE	B<>C ALL	A099	36E	?A#C ALL
A06D	038	READATA	A09A	15B	JNC +43d
A06E	31A	A<C ALL	A09B	10E	A=C ALL
A06F	3CB	JNC -07	A09C	0F8	READ 3(X)
A070	0AE	A<>C ALL	A09D	36E	?A#C ALL
A071	0CE	C=B ALL	A09E	143	JNC +40d
A072	158	M=C ALL	A09F	10E	A=C ALL
A073	3AB	JNC -11d	A0A0	138	READ 4(L)
A074	0B0	C=N ALL	A0A1	36E	?A#C ALL
A075	266	C=C-1 S&X	A0A2	1F3	JNC +62d
A076	070	N=C ALL	A0A3	2A0	SETDEC
A077	270	RAMSLCT	A0A4	10E	A=C ALL
A078	038	READATA	A0A5	198	C=M ALL

A0A6	1CE	A=A-C ALL	A0D3	04E	C=0 ALL
A0A7	08E	B=A ALL	A0D4	35C	PT= 12
A0A8	04E	C=0 ALL	A0D5	050	LD@PT- 1
A0A9	35C	PT= 12	A0D6	090	LD@PT- 2
A0AA	150	LD@PT- 5	A0D7	0D0	LD@PT- 3
A0AB	1A8	WRIT 6(N)	A0D8	110	LD@PT- 4
A0AC	3CE	RSHFC	A0D9	130	LDI S&X
A0AD	30E	?A<C ALL	A0DA	003	003
A0AE	187	JC +48d	A0DB	168	M=C ALL
A0AF	0F8	READ 3(X)	A0DC	04E	C=0 ALL
A0B0	10E	A=C ALL	A0DD	1A8	WRIT 6(N)
A0B1	198	C=M ALL	A0DE	163	JNC +44d
A0B2	1CE	A=A-C ALL	A0DF	1AB	JNC +53d
A0B3	04E	C=0 ALL	A0E0	04E	C=0 ALL
A0B4	19C	PT= 11	A0E1	35C	PT= 12
A0B5	0D0	LD@PT- 3	A0E2	050	LD@PT- 1
A0B6	158	M=C ALL	A0E3	090	LD@PT- 2
A0B7	36E	?A#C ALL	A0E4	0D0	LD@PT- 3
A0B8	087	JC +16d	A0E5	130	LDI S&X
A0B9	06E	A<>B ALL	A0E6	002	002
A0BA	3EE	LSHFA	A0E7	0C3	JNC+24d
A0BB	04E	C=0 ALL	A0E8	10E	A=C ALL
A0BC	35C	PT= 12	A0E9	0F8	READ 3(X)
A0BD	1D0	LD@PT- 7	A0EA	36E	?A#C ALL
A0BE	1CE	A=A-C ALL	A0EB	1CB	JNC +57d
A0BF	1B8	READ 6(N)	A0EC	10E	A=C ALL
A0C0	36E	?A#C ALL	A0ED	138	READ 4(L)
A0C1	0D7	JC +26d	A0EE	36E	?A#C ALL
A0C2	078	READ 1(Z)	A0EF	02F	JC +05
A0C3	1E8	WRIT 7(O)	A0F0	04E	C=0 ALL
A0C4	0D3	JNC +26d	A0F1	35C	PT= 12
A0C5	11B	JNC +35d	A0F2	050	LD@PT- 1
A0C6	1D3	JNC +58d	A0F3	183	JNC +48d
A0C7	0C3	JNC +24d	A0F4	228	WRIT 8(P)
A0C8	138	READ 4(L)	A0F5	078	READ 1(Z)
A0C9	10E	A=C ALL	A0F6	0AE	A<>C ALL
A0CA	078	READ 1(Z)	A0F7	268	WRIT 9(Q)
A0CB	1CE	A=A-C ALL	A0F8	04E	C=0 ALL
A0CC	198	C=M ALL	A0F9	35C	PT= 12
A0CD	36E	?A#C ALL	A0FA	050	LD@PT- 1
A0CE	02F	JC +05	A0FB	110	LD@PT- 4
A0CF	04E	C=0 ALL	A0FC	150	LD@PT- 5
A0D0	35C	PT= 12	A0FD	130	LDI S&X
A0D1	050	LD@PT- 1	A0FE	002	002
A0D2	04B	JNC +09	A0FF	1A3	JNC +52d

A100	10E	A=C ALL	A12D	35C	PT= 12
A101	138	READ 4(L)	A12E	0D0	LD@PT- 3
A102	36E	?A#C ALL	A12F	110	LD@PT- 4
A103	047	JC +08	A130	150	LD@PT- 5
A104	04E	C=0 ALL	A131	130	LDI S&X
A105	35C	PT= 12	A132	002	002
A106	050	LD@PT- 1	A133	168	M=C ALL
A107	090	LD@PT- 2	A134	0AE	A<>C ALL
A108	226	C=C+1 S&X	A135	1E8	WRIT 7(O)
A109	193	JNC +50d	A136	04E	C=0 ALL
A10A	183	JNC +48d	A137	35C	PT= 12
A10B	228	WRIT 8(P)	A138	050	LD@PT- 1
A10C	04E	C=0 ALL	A139	1A8	WRIT 6(N)
A10D	35C	PT= 12	A13A	153	JNC +42d
A10E	050	LD@PT- 1	A13B	123	JNC +36d
A10F	090	LD@PT- 2	A13C	1E8	WRIT 7(O)
A110	150	LD@PT- 5	A13D	10E	A=C ALL
A111	130	LDI S&X	A13E	138	READ 4(L)
A112	002	002	A13F	36E	?A#C ALL
A113	103	JNC +32d	A140	02F	JC +05
A114	10E	A=C ALL	A141	04E	C=0 ALL
A115	0B8	READ 2(Y)	A142	35C	PT= 12
A116	36E	?A#C ALL	A143	050	LD@PT- 1
A117	1BB	JNC +55d	A144	193	JNC +50d
A118	10E	A=C ALL	A145	228	WRIT 8(P)
A119	0F8	READ 3(X)	A146	04E	C=0 ALL
A11A	36E	?A#C ALL	A147	35C	PT= 12
A11B	193	JNC +50d	A148	050	LD@PT- 1
A11C	10E	A=C ALL	A149	150	LD@PT- 5
A11D	138	READ 4(L)	A14A	226	C=C+1 S&X
A11E	36E	?A#C ALL	A14B	0A3	JNC +20d
A11F	037	JC +06	A14C	113	JNC +34d
A120	04E	C=0 ALL	A14D	0C3	JNC +24d
A121	35C	PT= 12	A14E	1E8	WRIT 7(O)
A122	0D0	LD@PT- 3	A14F	10E	A=C ALL
A123	14B	JNC +41d	A150	0F8	READ 3(X)
A124	0C3	JNC +24d	A151	36E	?A#C ALL
A125	228	WRIT 8(P)	A152	12B	JNC +37d
A126	0AE	A<>C ALL	A153	10E	A=C ALL
A127	268	WRIT 9(Q)	A154	138	READ 4(L)
A128	0B8	READ 2(Y)	A155	228	WRIT 8(P)
A129	070	N=C ALL	A156	36E	?A#C ALL
A12A	078	READ 1(Z)	A157	183	JNC +48d
A12B	10E	A=C ALL	A158	0AE	A<>C ALL
A12C	04E	C=0 ALL	A159	268	WRIT 9(Q)

A15A	04E	C=0 ALL
A15B	35C	PT= 12
A15C	110	LD@PT- 4
A15D	150	LD@PT- 5
A15E	226	C=C+1 S&X
A15F	168	WRIT 5(M)
A160	04E	C=0 ALL
A161	35C	PT= 12
A162	0D0	LD@PT- 3
A163	1A8	WRIT 6(N)
A164	11B	JNC +35d
A165	1E8	WRIT 7(O)
A166	10E	A=C ALL
A167	138	READ 4(L)
A168	36E	?A#C ALL
A169	0F3	JNC +30d
A16A	268	WRIT 9(Q)
A16B	04E	C=0 ALL
A16C	35C	PT= 12
A16D	150	LD@PT- 5
A16E	168	WRIT 5(M)
A16F	078	READ 1(Z)
A170	228	WRIT 8(P)
A171	04E	C=0 ALL
A172	35C	PT= 12
A173	090	LD@PT- 2
A174	1A8	WRIT 6(N)
A175	093	JNC +18d
A176	063	JNC +12d
A177	04E	C=0 ALL
A178	35C	PT= 12
A179	1D0	LD@PT- 7
A17A	1A8	WRIT 6(N)
A17B	138	READ 4(L)
A17C	36E	?A#C ALL
A17D	053	JNC +10d
A17E	228	WRIT 8(P)
A17F	04E	C=0 ALL
A180	35C	PT= 12
A181	150	LD@PT- 5
A182	168	WRIT 5(M)
A183	04E	C=0 ALL
A184	35C	PT= 12
A185	190	LD@PT-6
A186	1A8	WRIT 6(N)

A187	260	SETHEX	
A188	378	READ 13(c)	
A189	03C	RCR 3	
A18A	106	A=C S&X	
A18B	130	LDI S&X	
A18C	00B	011d	
A18D	206	C=A+C S&X	
A18E	270	RAMSLCT	
A18F	10E	A=C ALL	
A190	0B0	C=N ALL	
A191	2F0	WRTDATA	
A192	130	LDI S&X	
A193	009	009	
A194	158	M=C ALL	
A195	270	RAMSLCT	
A196	038	READATA	
A197	0AE	A<>C ALL	
A198	266	C=C-1 S&X	
A199	270	RAMSLCT	
A19A	0AE	A<>C ALL	
A19B	2F0	WRTDATA	
A19C	198	C=M ALL	
A19D	266	C=C-1 S&X	
A19E	3B3	JNC -10d	
A19F	04E	C=0 ALL	
A1A0	270	RAMSLCT	
A1A1	35C	PT=12	37 is used to evaluate the poker hands.
A1A2	0D0	LD@PT- 3	Replace these 2 lines if you prefer
A1A3	1D0	LD@PT- 7	another number, but choose at least 16.
A1A4	226	C=C+1 S&X	
A1A5	2A0	SETDEC	
A1A6	128	WRIT 4(L)	
A1A7	10E	A=C ALL	
A1A8	1B8	READ 6(N)	
A1A9	135	?NC XQ	C = A*C
A1AA	060	184D	[MP2_10]
A1AB	000		
A1AC	1F8	READ 7(O)	
A1AD	025	?NC XQ	C = AB+C
A1AE	060	->1809	[AD1_10]
A1AF	000		
A1B0	138	READ 4(L)	
A1B1	13D	?NC XQ	C = AB*C
A1B2	060	->184F	[MP1_10]
A1B3	000		

A1B4	238	READ 8(P)	
A1B5	025	?NC XQ	C = AB+C
A1B6	060	->1809	[AD1_10]
A1B7	000		
A1B8	138	READ 4(L)	
A1B9	13D	?NC XQ	C= AB*C
A1BA	060	->184F	[MP1_10]
A1BB	000		
A1BC	278	READ 9(Q)	
A1BD	025	?NC XQ	C = AB+C
A1BE	060	->1809	[AD1_10]
A1BF	000		
A1C0	138	READ 4(L)	
A1C1	13D	?NC XQ	C= AB*C
A1C2	060	->184F	[MP1_10]
A1C3	000		
A1C4	0B0	C=N ALL	
A1C5	025	?NC XQ	C = AB+C
A1C6	060	->1809	[AD1_10]
A1C7	000		
A1C8	138	READ 4(L)	
A1C9	13D	?NC XQ	C= AB*C
A1CA	060	->184F	[MP1_10]
A1CB	000		
A1CC	046	C=0 S&X	
A1CD	270	RAMSLCT	
A1CE	038	READATA	
A1CF	025	?NC XQ	C = AB+C
A1D0	060	->1809	[AD1_10]
A1D1	0E8	WRIT 3(X)	Here the combination is evaluated.
A1D2	070	N=C ALL	Then its value is compared to R48
A1D3	260	SETHX	and the maximum is stored in R48
A1D4	378	READ 13(c)	
A1D5	03C	RCR 3	
A1D6	106	A=C S&X	
A1D7	130	LDI S&X	
A1D8	030	CON=48	
A1D9	206	C=A+C S&X	
A1DA	270	RAMSLCT	
A1DB	0E6	B<>C S&X	
A1DC	038	READATA	
A1DD	10E	A=C ALL	
A1DE	0B0	C=N ALL	
A1DF	306	?A<C S&X	
A1E0	037	JC +06	

A1E1	0AE	A<>C ALL
A1E2	306	?A<C S&X
A1E3	03F	JC +07
A1E4	31A	?A<C M
A1E5	02F	JC +05
A1E6	0E6	B<>C S&X
A1E7	270	RAMSLCT
A1E8	0B0	C=N ALL
A1E9	2F0	WRTDATA
A1EA	04E	C=0 ALL
A1EB	270	RAMSLCT
A1EC	10C	?FSET 8
A1ED	360	?C RTN
LB_FED8	378	READ 13(c)
A1EF	03C	RCR 3
A1F0	106	A=C S&X
A1F1	130	LDI S&X
A1F2	010	016d
A1F3	206	C=A+C S&X
A1F4	0E6	B<>C S&X
A1F5	130	LDI S&X
A1F6	004	004
A1F7	070	N=C ALL
A1F8	0E6	B<>C S&X
A1F9	226	C=C+1 S&X
A1FA	270	RAMSLCT
A1FB	0E6	B<>C S&X
A1FC	038	READATA
A1FD	0AE	A<>C ALL
A1FE	226	C=C+1 S&X
A1FF	270	RAMSLCT
A200	0AE	A<>C ALL
A201	2F0	WRTDATA
A202	0B0	C=N ALL
A203	266	C=C-1 S&X
A204	39B	JNC-13d
A205	3E0	RTN

Header A207	084	"D"	
Header A208	012	"R"	
Header A209	001	"A"	
Header A20A	003	"C"	JM Baillard
<b>CARD A20B 0F8</b>		<b>READ 3(X)</b>	
A20C	0A8	WRIT 2(Y)	
A20D	378	READ 13(c)	
A20E	03C	WCR 3	
A20F	106	A=C S&X	
A210	130	LDI S&X	check that R00 exist.
A211	1EB	1EBh	
A212	306	?A<C S&X	
A213	381	?NC GO	
A214	00A	->02E0	[ERRNE]
A215	0A6	A<>C S&X	
A216	270	RAMSLCT	
A217	038	READATA	
A218	2A0	SETDEC	
A219	231	?NC XQ	xeq R-D
A21A	064	->198C	[TODEC]
A21B	084	CLRF 5	fractional part
A21C	0ED	?NC XQ	
A21D	064	->193B	[INTFRC]
A21E	2F0	WRITDATA	
A21F	10E	A=C ALL	
A220	04E	C=0 ALL	
A221	270	RAMSLCT	
A222	35C	PT=12	
A223	050	LD@PT- 1	
A224	0D0	LD@PT- 3	
A225	226	C=C+1 S&X	
A226	135	?NC XQ	C=A*C
A227	060	->184D	[MP2_10]
A228	088	SETF 5	Integer part
A229	0ED	?NC XQ	
A22A	064	->193B	[INTFRC]
A22B	10E	A=C ALL	
A22C	04E	C=0 ALL	
A22D	35C	PT=12	
A22E	090	LD@PT- 2	
A22F	210	LD@PT- 8	
A230	226	C=C+1 S&X	
A231	01D	?NC XQ	C=A+C
A232	060	->1807	[AD2_10]
A233	0E8	WRIT 3(X)	
A234	3E0	RTN	



**FOCAL listing:****01\*LBL "POKER"**

02 SIZE?	51 STO 07	101 STO 13	151 +
03 51	52 27	102 RCL 02	152 7
04 X>Y?	53 STO 08	103 STO 14	153 X>Y?
05 PSIZE	54 E3	104 CLA	154 ST* 10
06 10	55 STO 12	105 ARCL 21	155 CLX
07 STO 42	56 CHS	106 ASTO 44	156*LBL 03
08 27	57 STO 13	107 <b>CARD</b>	157 AVIEW
09 "A"	58 " "	108 STO 17	158*LBL 14
10 ASTO 40	59 ASTO 21	109 <b>CARD</b>	159 GETKEY
11 35	60*LBL 02	110 STO 18	160 44
12 "K"	61 CLA	111 <b>CARD</b>	161 X=Y?
13 ASTO 39	62 ARCL 21	112 STO 19	162 GTO 06
14 1.015008	63 <b>CARD</b>	113 "/"	163 X<> Z
15 STO 43	64 STO IND 07	114 ARCL IND 17	164 ST+ 12
16 "Q"	65 ARCL IND X	115 ARCL IND 18	165 X<>Y
17 ASTO 38	66 ASTO 21	116 ARCL IND 19	166 41
18 9	67 FS? 07	117 ASTO 45	167 X=Y?
19 "J"	68 GTO 02	118 " FLOP. . ."	168 GTO 10
20 ASTO 37	69 <b>CARD</b>	119 AVIEW	169 LN
21 "T"	70 STO IND 06	120 XEQ 16	170 *
22 ASTO 36	71*LBL 02	121 STO 47	171 X^2
23 "/"	72 FS? 08	122 XEQ 16	172 RCL 12
24*LBL 00	73 ARCL IND 08	123 STO 46	173 +
25 STO IND Z	74 AVIEW	124 STO 08	174 RCL 10
26 R^	75 DSE 08	125 CLA	175 X<>Y
27 XTOA	76 ISG 07	126 ARCL 44	176 X>Y?
28 ASTO IND X	77 CLX	127 ARCL 45	177 GTO 07
29 RDN	78 DSE 06	128 ARCL 26	178 STO 12
30 DSE T	79 GTO 02	129 E6	179 X<>Y
31 DSE Z	80 FS? 07	130 /	180 3
32 DSE X	81 GTO 11	131 X^2	181 /
33 X>Y?	82 FC? 08	132 RCL 50	182 X<Y?
34 GTO 00	83 GTO 01	133 +	183 GTO 10
35 FS? 08	84 SF 09	134 STO 10	184 *
36 CF 07	85 RCL 00	135 SIGN	185 SQRT
37 CF 09	86 FRC	136 RCL 00	186 RCL 00
38*LBL 01	87 4 E3	137 R-D	187 R-D
39 CLX	88 STO 08	138 FRC	188 FRC
40 STO 50	89 ST* Y	139 STO 00	189 STO 00
41 SIGN	90 ST+ X	140 X^2	190 *
42 STO 49	91 +	141 -	191 " +"
43 FIX 0	92 STO 10	142 ST/ 49	192 ARCL X
44 CF 29	93 STO 50	143 CLX	193 GTO 03
45 5	94 CLX	144 GTO 03	194*LBL 10
46 STO 48	95 GTO 14	145*LBL 01	195 FC? 08
47 FS? 08	96*LBL 20	146 XEQ 09	196 GTO 10
48 2	97 SF 10	147 RCL 07	197 FS?C 09
49 STO 06	98 3	148 X#0?	198 GTO 20
50 15	99 STO 24	149 9	199 DSE 24
	100 RCL 01	150 RCL 06	200 FS? 30

201 GTO 08	253 ATOX	305 STOP	357 PI
202 <b>CARD</b>	254 AVIEW	306 GTO 01	358 RCL 13
203 FS? 10	255 RCL 13	307*LBL 10	359 -
204 STO 20	256 X<> 15	308 ARCL 23	360 X<0?
205 STO 21	257 STO 13	309 AVIEW	361 +
206 " RIVER. . ."	258 RCL 14	310*LBL 11	362 RCL 07
207 FS? 10	259 X<> 16	311 CLA	363 X<Y?
208 " TURN. . ."	260 STO 14	312*LBL 05	364 STO 10
209 AVIEW	261 RCL 47	313 34	365 CLX
210 RCL 46	262 XEQ 18	314 GETKEYX	366 GTO 03
211 FC? 10	263 STO 47	315 X=0?	367*LBL 08
212 GTO 04	264 RCL 46	316 GTO 10	368 FS? 07
213 RCL 47	265 -	317 XTOA	369 GTO 06
214 XEQ 17	266 GTO 07	318 LASTX	370 RCL 43
215 STO 47	267*LBL 10	319 -	371 REGSWAP
216 RCL 46	268 CLA	320 CLRGX	372 XEQ 13
217 XEQ 17	269 ARCL 21	321 GTO 05	373*LBL 06
218*LBL 04	270 RCL 13	322*LBL 10	374 RCL 43
219 FC? 10	271 X<0?	323 ANUM	375 REGSWAP
220 XEQ 18	272 GTO 10	324 STO 20	376*LBL 09
221 STO 46	273 6	325 ALENG	377 XEQ 13
222 STO 08	274 ""/"	326 ST+ X	378 FS? 07
223 E6	275*LBL 15	327 STO 13	379 RTN
224 /	276 E	328 20	380 CLA
225 X^2	277*LBL 19	329 " "	381 ARCL IND 15
226 RCL 50	278 RCL IND X	330 15	382 ARCL IND 16
227 +	279 ARCL IND X	331*LBL 12	383 ARCL IND 17
228 RCL 49	280 AVIEW	332 RCL IND X	384 ARCL IND 18
229 *	281 COS	333 X#0?	385 ARCL IND 19
230 STO 10	282 SIGN	334 ARCL IND X	386 ASTO 21
231 CLA	283 +	335 SIGN	387 ARCL 23
232 ARCL 45	284 X#Y?	336 +	388 E6
233 FS? 10	285 GTO 19	337 X#Y?	389 /
234 ARCL IND 20	286 RCL 22	338 GTO 12	390 X^2
235 FC?C 10	287*LBL 06	339 RCL 06	391 STO 10
236 ARCL IND 21	288 RCL 08	340 DSE X	392 RTN
237 ASTO 45	289 -	341 LOG	393*LBL 07
238 CLA	290*LBL 07	342 24	394 5
239 ARCL 44	291 X#0?	343 +	395 STO 22
240 ARCL 45	292 SIGN	344 FC? 07	396 CHS
241 ARCL 26	293 RCL 12	345 ARCL IND X	397 STO 12
242 CLX	294 *	346 AVIEW	398 CLX
243 GTO 03	295 ST+ 41	347 XEQ 08	399 X<> 08
244*LBL 08	296 RCL 41	348 FS? 07	400 RCL 07
245 ""/"	297 " "	349 GTO 07	401 X=0?
246 ARCL IND 13	298 X>0?	350 SIGN	402 GTO 07
247 ARCL IND 14	299 ""+"	351 RCL 00	403 .47
248 ASTO 24	300 ARCL 41	352 X^2	404 *
249 CLA	301 "" \$"	353 -	405 E^X
250 ARCL 44	302 FIX 4	354 SQRT	406 .75
251 ARCL 45	303 SF 29	355 ST/ 10	407 *
252 ARCL 24	304 AVIEW	356 PI	408 RND

409 STO 12	447 CLX	485 CHS	523 RCL 13
410 E	448 ##	486 ##	524 STO 03
411 X#Y?	449 RCL 48	487 RCL 13	525 ##
412 GTO 07	450 RTN	488 STO 01	526 RCL 13
413 37	451*LBL 17	489 RCL 14	527 STO 04
414 R^	452 STO 48	490 STO 02	528 ##
415 X<Y?	453 CHS	491 ##	529 RCL 14
416 DSE 12	454 ##	492 RCL 13	530 STO 01
417 INT	455 RCL 13	493 STO 01	531 ##
418 33	456 X<> 15	494 RCL 14	532 RCL 14
419 X>Y?	457 STO 13	495 STO 03	533 STO 02
420 DSE 12	458 STO 01	496 ##	534 ##
421*LBL 07	459 RCL 14	497 RCL 13	535 RCL 14
422 RCL 00	460 X<> 16	498 STO 01	536 STO 03
423 FRC	461 STO 14	499 RCL 14	537 ##
424 4	462 STO 05	500 STO 04	538 RCL 14
425 /	463 ##	501 ##	539 STO 04
426 8	464 RCL 14	502 RCL 13	540 ##
427 +	465 STO 05	503 STO 02	541 CLX
428 E^X	466 RCL 13	504 RCL 14	542 ##
429 ST* 12	467 STO 02	505 STO 03	543 RCL 48
430 6	468 ##	506 ##	544 RTN
431 " "	469 RCL 14	507 RCL 13	545*LBL 13
432 GTO 15	470 STO 05	508 STO 02	546 RCL 06
433*LBL 16	471 RCL 13	509 RCL 14	547 INT
434 CLX	472 STO 03	510 STO 04	548 RCL 42
435 STO 48	473 ##	511 ##	549 ST/ 06
436 SIGN	474 RCL 14	512 RCL 13	550 MOD
437 CHS	475 STO 05	513 STO 03	551 X#0?
438 ##	476 ##	514 RCL 14	552 CARD
439 RCL 13	477 RCL 13	515 STO 04	553 X#0?
440 X<> 15	478 STO 05	516 ##	554 STO IND Y
441 STO 13	479 CLX	517 RCL 13	555 X#0?
442 STO 04	480 ##	518 STO 01	556 GTO 13
443 RCL 14	481 RCL 48	519 ##	557 ##
444 X<> 16	482 RTN	520 RCL 13	558 FC? 07
445 STO 14	483*LBL 18	521 STO 02	559 STO 08
446 STO 05	484 STO 48	522 ##	560 END

## Sudoku Solver

Martin/Baillard – <http://hp41programs.yolasite.com/sudoku.php>

A sudoku is a 81-cell grid with 9 rows, 9 columns and 9 regions of 3 x 3 cells. Each row, each column and each region must contain all the integers from 1 to 9 exactly once, for instance:

```

4 6 1 | 3 7 2 | 8 5 9
9 3 5 | 4 8 6 | 1 7 2
2 7 8 | 9 1 5 | 3 6 4
-----
7 9 4 | 2 3 1 | 5 8 6
5 1 2 | 8 6 9 | 4 3 7
6 8 3 | 5 4 7 | 9 2 1
-----
8 4 6 | 7 9 3 | 2 1 5
1 2 9 | 6 5 8 | 7 4 3
3 5 7 | 1 2 4 | 6 9 8

```

Given a partially empty grid, the puzzle consists in finding the missing numbers. The sudoku above is the solution of the problem below, where the empty cells are replaced by zeros:

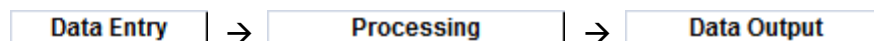
```

0 0 0 | 3 0 0 | 0 5 0
0 0 5 | 4 0 6 | 0 0 2
2 7 0 | 0 1 0 | 3 6 0
-----
7 0 4 | 2 3 0 | 0 0 0
5 1 0 | 0 0 0 | 0 3 7
0 0 0 | 0 4 7 | 9 0 1
-----
0 4 6 | 0 9 0 | 0 1 5
1 0 0 | 6 0 8 | 7 0 0
0 5 0 | 0 0 4 | 0 0 0

```

This M-Code routine **SDK** uses a similar scheme to solve a Sudoku seen in the FOCAL routine, but it is almost 200 times as fast as the focal program. Therefore, many more puzzles can be solved in a "reasonable" time. "SDK" solves a sudoku by backtracking. Though it could theoretically solve any - solvable - sudoku, only a subset can be practically solved, due to very long execution times involved in some instances.

A FOCAL driver program "**SUDOKU**" acts as master of ceremonies, arranging the data input, processing by SDK, and output of the results - represented by the block diagram below:



Data entry in turn also uses its own FOCAL driver, "SUD" – managing function ^SROW.

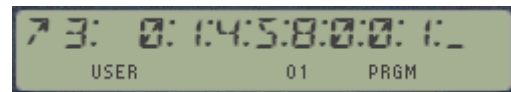
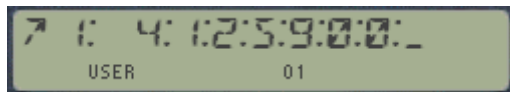
## Grid data entry

One of the nice aspects of the implementation is the fast and convenient way to enter the Sudoku grid data. The traditional approach would require prompting for each individual element, for a total of 81 – clearly a tiring and inefficient system.

Considering that the Sudoku elements are single digits (0 to 9), there must be a better way to go about the data entry process – and there is: meet the **^SROW** function, which uses Alpha as vehicle for the digit input *in blocks of multiple elements at once*, (therefore arranged in rows) all with a special keyboard enabled for the task, as follows:

- *Numeric keypad, for 0-9 as element value*
- *Back arrow to delete previous entry or cancel out*
- *R/S, to terminate the entry sequence*
- *ON turns the calculator off*

The figures below show entry of the elements in rows #1 and #3 mid-way. Note how digits are separated by a colon, delimiting each individual digit. The left of the display has an input arrow plus a number, signaling the “row” being edited.



The editing process may be terminated at any time. Elements are being stored in Alpha, from where they'll be retrieved by the data input routine in a loop using function **ANUMDL** – converting the alpha string into a valid number and storing it in the appropriate data register.

## Solving the puzzle with SDK.

At the core of this Sudoku implementation is the number-crunching engine, SDK, the real heart which actually resolve the puzzles and write the results for the data output routines to pick-up. SDK was written by Jean-Marc Baillard; see the web pages on the URL above.

**Execution times can be very long**, therefore it's most recommended to use **TURBO50** on the 41CL – or a PC-emulator (like V41 running in turbo mode). Note also that some grids won't have a valid solution, which will be displayed as **"DATA ERROR"** or **"NO SOLUTION"** – both really meaning: “sorry, no cigar”.

The best way to test the functionality is by solving the canned examples **"SDK0"**, **"SDK1"**, and **"SDK2"**. The first one is a trivial all-zero (blank) grid – not a valid Sudoku but interesting nonetheless. The second is a simple easy case, which is resolved quickly. The third however will require a much longer execution time – and all the three of them can be used to check that the programs are working fine. Refer to JM's pages for the grid definition and solutions of these examples.

**Example1:** Solve the following sudoku:

```
0 0 0 | 3 0 0 | 0 5 0
0 0 5 | 4 0 6 | 0 0 2
2 7 0 | 0 1 0 | 3 6 0
```

```
-----
7 0 4 | 2 3 0 | 0 0 0
5 1 0 | 0 0 0 | 0 3 7
0 0 0 | 0 4 7 | 9 0 1
```

```
-----
0 4 6 | 0 9 0 | 0 1 5
1 0 0 | 6 0 8 | 7 0 0
0 5 0 | 0 0 4 | 0 0 0
```

**A more difficult example:** With the grid:

```
0 9 0 | 0 4 2 | 0 1 0
0 0 5 | 0 0 0 | 0 0 0
3 0 0 | 0 0 0 | 9 0 4
```

```
-----
0 0 0 | 0 0 0 | 1 9 3
5 2 0 | 7 0 0 | 0 0 6
0 0 0 | 0 0 1 | 0 0 0
```

```
-----
9 0 0 | 0 5 0 | 0 6 0
0 0 0 | 2 0 4 | 0 0 7
0 0 0 | 0 1 6 | 8 0 0
```

### Outputting the results.

Data output is presented in the same compact mode way in all cases – regardless of the program used. There is one prompt per row, repeated until the complete grid is shown. This facilitates reading the solution as it avoids multiple prompts to see the elements of a single row. Below are some examples taken from the solution of the "SDK1" puzzle.

The complete solution to the difficult example is provided below:

```
7 9 8 | 6 4 2 | 3 1 5
4 1 5 | 9 7 3 | 6 2 8
3 6 2 | 1 8 5 | 9 7 4
```

```
-----
6 7 4 | 5 2 8 | 1 9 3
5 2 1 | 7 3 9 | 4 8 6
8 3 9 | 4 6 1 | 7 5 2
```

```
-----
9 4 3 | 8 5 7 | 2 6 1
1 8 6 | 2 9 4 | 5 3 7
2 5 7 | 3 1 6 | 8 4 9
```

Data Registers used:

- R00: scratch
- R01 & R09 hold the row values.
- R10 to R18 hold the elements of the sudoku grid, in column order (or row order) with 0 in the empty cells.

**FOCAL listing:**

First the main driver programs, followed by the example loaders:

<u>01*LBL "SUDOKU"</u>	25 INT	<u>49*LBL "SUDVIEW"</u>
02 SIZE?	26 10^X	50 9
03 20	27 /	51 E3/E+
04 X>Y?	28 ST+ IND Z	52*LBL 03
05 PSIZE	29*LBL 05	53 "R"
06 9	30 RDN	54 AINT
07 E3/E+	31 ISG X	55 "": "
08 CLRGX	32 GTO 00	56 9
09*LBL 01	33*LBL 02	57 E3/E+
10 E	34 RDN	58 RCL IND Y
11 STO IND Y	35 ISG X	59*LBL 04
12 CLX	36 GTO 01	60 FRC
13 9	<u>37*LBL "SUD"</u>	61 E1
14 E3/E+	38 9	62 *
15 ^SROW	39 E3/E+	63 AINT
16 CF 22	40 LASTX	64 "": "
17 -SUDOKU 1C	41 +	65 ISG Y
18*LBL 00	42 E3/E+	66 GTO 04
19 ANUMDL	43 REGMOVE	67 AVIEW
20 FC?C 22	<b>44 SDK</b>	68 RDN
21 GTO 02	45 RASP	69 RDN
22 X=0?	46 " ** DONE **"	70 ISG X
23 GTO 05	47 AVIEW	71 GTO 03
24 RCL Y	48 PSE	72 END
<u>01*LBL "SDK1"</u>	13 1,000047901	25 9
02 -SUDOKU 1C	14 STO 06	26 E3/E+
03 1,00030005	15 1,046090015	27 ENTER^
04 STO 01	16 STO 07	28 E
05 1,005406002	17 1,1006087	29*LBL 05
06 STO 02	18 STO 08	30 STO IND Y
07 1,27001036	19 1,050004	31 ISG Y
08 STO 03	20 STO 09	32 GTO 05
09 1,70423	21 XROM "SUD"	33 XROM "SUD"
10 STO 04	22 RTN	34 END
11 1,510000037	<u>23*LBL "SDK0"</u>	
12 STO 05	24 -SUDOKU 1C	

**MCODE listing:**

Here's the code for the Data Entry routine, ^SROW:

Header A164	097	"W"	
Header A165	00F	"O"	
Header A166	012	"R"	prompts SDK Row
Header A167	013	"S"	
Header A168	01E	"^"	Ángel Martin
<b>^SROW A169</b>	<b>0B8</b>	<b>READ 2(Y)</b>	
A16A	070	N=C ALL	
A16B	345	?NC XQ	Clears Alpha
A16C	040	->10D1	[CLA]
A16D	3C1	?NCXQ	Enable & Clear LCD
A16E	0B0	->2CF0	[CLLCDE]
A16F	3BD	?NC XQ	
A170	01C	->07EF	[MESSL]
A171	21E	"^"	
A172	0B0	C=N ALL	row #
A173	37C	RCR 12	
A174	21C	PT= 2	
A175	010	LD@PT- 0	
A176	2D0	LD@PT- B	
A177	3E8	WRIT 15(e)	write it in display (9-bit)
A178	130	LDI S&X	
A179	020	" "	
A17A	3E8	WRIT 15(e)	
DECNTRY	115	?NC XQ	Partial Data Entry!
A17C	038	->0E45	[NEXT1]
BCKARW	19B	JNC +51d	
A17E	00C	?FSET 3	numeric input?
A17F	0E3	JNC +28d	NO, KEEP LOOKING
A180	0BE	A<>C MS	recall LS digit from A[13]
A181	2FC	RCR 13	move it over digit 0
A182	21C	PT= 2	ensures no lower-case bits!
A183	010	LD@PT- 0	from angle chr = 10D
A184	0D0	LD@PT- 3	add a 3 on digit 1
A185	070	N=C ALL	
A186	3D8	C<>ST	add a colon as separator
A187	288	SETF 7	
A188	3D8	C<>ST	
A189	3E8	WRIT 15(e)	write it in display (9-bit)
A18A	149	?NC XQ	Disable PER, enable RAM
A18B	024	->0952	[ENCP00]



A18C	0B0	C=N ALL	
A18D	39C	PT= 0	
A18E	058	G=C @PT,+	reset PTEMP bits
A18F	051	?NC XQ	
A190	0B4	->2D14	[APNDNW]
A191	130	LDI S&X	
A192	03A	":"	appends ":" [03A]
A193	3D5	?NC XQ	
A194	07C	->1FF5	[APND10]
A195	39C	PT= 0	
A196	042	C=0 @PT	
A197	058	G=C @PT,+	reset PTEMP bits
A198	3D9	?NC XQ	Enable Display (not cleared)
A199	01C	->07F6	[ENLCD]
A19A	30B	JNC -31d	ONE PROMPT
A19B	0B0	C=N ALL	PRESSED KEY CODE
A19C	106	A=C S&X	
A19D	130	LDI S&X	
A19E	370	CON:	R/S keycode [370]
A19F	366	?A#C S&X	terminate digit entry
A1A0	033	JNC +06	
A1A1	265	?NC XQ	Blink Display - pass #1
A1A2	020	->0899	[BLINK1]
A1A3	265	?NC XQ	Blink Display - pass #2
A1A4	020	->0899	[BLINK1]
A1A5	3AB	JNC -11d	ONE PROMPT
WAYOUT	3D9	?NC XQ	Clear LCD
A1A7	0B0	->2CF6	[CLRLCD]
A1A8	261	?NC XQ	Reset keyboard
A1A9	000	->0098	[RSTKB]
A1AA	149	?NC XQ	Disable PER, enable RAM
A1AB	024	->0952	[ENCP00]
A1AC	215	?NC XQ	Reset BIT sequence
A1AD	00C	->0385	[RSTSQ]
A1AE	3C1	?NC GO	
A1AF	002	->00F0	[NFRPU]
DELCHAR	3B8	READ 14(d)	to delete rightmost chr
A1B1	149	?NC XQ	Disable PER, enable RAM
A1B2	024	->0952	[ENCP00]
A1B3	178	READ 5(M)	
A1B4	2EE	?C#0 ALL	anything in Alpha?
A1B5	037	JC +06	yes, go on
A1B6	104	CLRF 8	no, abort if empty
A1B7	1B1	?NC XQ	Mainframe Message
A1B8	070	->1C6C	[MSGA]

A1B9	03C	"NULL"	from table
A1BA	363	JNC -20d	
A1BB	36D	PORT DEP:	remove Alpha's last chr
A1BC	08C	XQ	Ken Emery's book
A1BD	07D	->A47D	[ADEL]
A1BE	3D9	?NC XQ	Enable Display (not cleared)
A1BF	01C	->07F6	[ENLCD]
A1C0	369	PORT DEP:	display ONE prompt
A1C1	03C	GO	(too far for jumps)
A1C2	17B	->A17B	[DECNTY]

The code for the SDK Solver is shown below:

Header A078	08B	"K"	
Header A079	004	"D"	MCODE Solver
Header A07A	013	"S"	Expects data in R10-R19
<b>SDK A07B 379</b>	<b>PORT DEP:</b>		give feedback
A07C	03C	XQ	
A07D	064	->A064	[SHOWUP]
A07E	3C8	CLRKEY	
A07F	378	READ 13( c)	
A080	03C	RCR 3	
A081	106	A=C S&X	
A082	130	LDI S&X	
A083	012	CON:	needs 11 registers
A084	146	A=A+C S&X	
A085	130	LDI S&X	
A086	200	CON: 512	
A087	306	?A<C S&X	
A088	381	?NC GO	Displays"NonExistent"
A089	00A	->02E0	[ERRNE]
A08A	378	READ 13(c)	
A08B	0A6	A<>C S&X	save it in A
A08C	106	A=C S&X	keep it in C
A08D	1BC	RCR 11	
A08E	130	LDI S&X	
A08F	009	CON:	
A090	246	C=A-C S&X	
A091	106	A=C S&X	
A092	27C	RCR 9	
A093	0A6	A<>C S&X	
A094	0BC	RCR 5	
A095	070	N=C ALL	
A096	04E	C=0 ALL	
A097	19C	PT=11	

	A098	050	LD@PT- 1	C = 111111111
	A099	214	?PT= 2	in nybbles 11 to 3
	A09A	3F3	JNC -02	
	A09B	000		
	A09C	000		
	A09D	000		
	A09E	000		
	A09F	000		
	A0A0	000		
	A0A1	268	WRIT 9(Q)	
	A0A2	10E	A=C ALL	
	A0A3	1EE	C=C+C ALL	
	A0A4	1EE	C=C+C ALL	
	A0A5	1EE	C=C+C ALL	
	A0A6	20E	C=A+C ALL	
	A0A7	228	WRIT 8(P)	P = 999999999 in nybbles 11 to 3
	A0A8	0A0	SLCT P	
LOOP0	A0A9	21C	PT=2	
LOOP1	A0AA	3DC	PT=PT+1	
	A0AB	0B0	C= N ALL	
	A0AC	354	?PT=12	
	A0AD	063	JNC +12d	
	A0AE	266	C=C-1 S&X	
	A0AF	27A	C=C-1 M	
	A0B0	070	N=C ALL	
	A0B1	106	A=C S&X	
	A0B2	17C	RCR 6	
	A0B3	366	?A#C S&X	
	A0B4	3AF	JC -11d	[LOOP0]
	A0B5	04E	C=0 ALL	
	A0B6	270	RAMSLCT	
	A0B7	228	WRIT 8(P)	
	A0B8	3E0	RTN	the routine stops here if a solution is found
	A0B9	270	RAMSLCT	
	A0BA	038	READATA	
	A0BB	2E2	?C#0 @PT	
	A0BC	377	JC -18d	[LOOP1]
	A0BD	10E	A=C ALL	
	A0BE	046	C=0 S&X	
	A0BF	270	RAMSLCT	
	A0C0	238	READ 8(P)	
LOOP2	A0C1	158	M=C ALL	
	A0C2	0E0	SLCT Q	
	A0C3	01C	PT=3	
	A0C4	362	?A#C@PT	we test if the digit is different from all the other digits in

the same column

A0C5 07B JNC +15d  
 A0C6 3DC PT=PT+1  
 A0C7 354 ?PT=12  
 A0C8 3E3 JNC -04  
 A0C9 0A0 SLCT P  
 A0CA 130 LDI S&X  
 A0CB 008 CON:  
 A0CC 10E A=C ALL  
 A0CD 0B0 C=N ALL  
 A0CE 17C RCR 6  
 A0CF 226 C=C+1 S&X  
 A0D0 270 RAMSLCT  
 A0D1 0E6 B<>C S&X  
 A0D2 038 READATA  
 A0D3 362 ?A#C@PT

we test if the digit is different from all the other digits in the same row

A0D4 1C3 JNC +56d  
 A0D5 0E6 B<>C S&X  
 A0D6 1A6 A=A-1 S&X  
 A0D7 3C3 JNC -08  
 A0D8 0B0 C=N ALL  
 A0D9 10E A=C ALL  
 A0DA 1A6 A=A-1 S&X  
 A0DB 17C RCR 6  
 A0DC 226 C=C+1 S&X  
 A0DD 226 C=C+1 S&X  
 A0DE 226 C=C+1 S&X  
 A0DF 306 ?A<C S&X  
 A0E0 03F JC +07  
 A0E1 226 C=C+1 S&X  
 A0E2 226 C=C+1 S&X  
 A0E3 226 C=C+1 S&X  
 A0E4 306 ?A<C S&X  
 A0E5 017 JC +02  
 A0E6 03C RCR 3  
 A0E7 0E6 B<>C S&X

B.X contains the address of the register in the right lower corner of the 3x3 region (R09 or R06 or R03)

A0E8 013 JNC +02  
 A0E9 20B JNC -63d  
 A0EA 0E0 SLCT Q  
 A0EB 01C PT=3  
 A0EC 0A0 SLCT P  
 A0ED 014 ?PT=3  
 A0EE 087 JC +16d

[LOOP1]

A0EF	054	?PT=4	
A0F0	077	JC +14d	
A0F1	094	?PT=5	
A0F2	067	JC +12d	
A0F3	0E0	SLCT Q	
A0F4	15C	PT=6	
A0F5	0A0	SLCT P	
A0F6	154	?PT=6	
A0F7	03F	JC +07	
A0F8	294	?PT=7	
A0F9	02F	JC +05	
A0FA	114	?PT=8	
A0FB	01F	JC +03	
A0FC	0E0	SLCT Q	
A0FD	25C	PT= 9	
A0FE	0E0	SLCT Q	now, pointer Q points to the right lower corner of the 3x3 region ( 9 or 6 or 3 )
A0FF	013	JNC +02	
A100	20B	JNC -63d	[LOOP2]
A101	198	C=M ALL	
A102	130	LDI S&X	the 4 instructions on the left prepare 2 loops that may be executed 3 times
A103	002	CON:	
A104	23E	C=C+1 MS	
A105	23E	C=C+1 MS	
A106	10E	A=C ALL	
A107	0E6	B<>C S&X	
A108	270	RAMSLCT	
A109	0E6	B<>C S&X	
A10A	038	READATA	
A10B	362	?A#C @PT	Test if the digit is different from all the other digits In the same 3x3 region
A10C	0B3	JNC +22d	
A10D	3DC	PT=PT+1	
A10E	1A6	A=A-1 S&X	
A10F	3E3	JNC -04	
A110	166	A=A+1 S&X	
A111	3D4	PT=PT-1	
A112	166	A=A+1 S&X	
A113	3D4	PT=PT-1	
A114	166	A=A+1 S&X	
A115	3D4	PT=PT-1	
A116	0E6	B<>C S&X	
A117	266	C=C-1 S&X	
A118	0E6	B<>C S&X	

A119	1BE	A=A-1 MS	
A11A	36B	JNC -19	
A11B	0A0	SELCT P	
A11C	0B0	C=N ALL	if the candidate number has successfully passed all the tests, it replaces the zero in the empty cell
A11D	270	RAMSLCT	
A11E	038	READATA	
A11F	0A2	A<>C@PT	
A120	2F0	WRITDATA	
A121	243	JNC -56d	[LOOP1]
A122	0A0	SLCT P	the execution jumps here if the digit has been rejected
A123	198	C=M ALL	
A124	10E	A=C ALL	
A125	046	C=0 S&X	
A126	270	RAMSLCT	
A127	278	C=Q	
A128	1CE	A=A-C ALL	we started with 999999999 in M, then we try 888888888 in M , ...etc...
A129	0B0	C=N ALL	
A12A	270	RAMSLCT	
A12B	038	READATA	
A12C	0AE	A<>C ALL	
A12D	2EE	?C#0 ALL	... until we arrive at 000000000
A12E	013	JNC +02	
A12F	28B	JNC -47d	[LOOP2]
A130	3D4	PT=PT-1	if all the digits are rejected for this cell, we go to the previous cell ( backtracking )
A131	214	?PT=12	
A132	107	JC +32d	we must also move to the previous register if we were at the left of a register
A133	0B0	C=N ALL	
A134	03C	RCR 3	
A135	270	RAMSLCT	otherwise, we check in a register between R10 and R18 if the cell was empty or not
A136	038	READATA	
A137	2E2	?C#0 @PT	
A138	3C7	JC -08	If the cell was not empty, go to the previous cell again
A139	0B0	C=N ALL	
A13A	270	RAMSLCT	
A13B	038	READATA	
A13C	10E	A=C ALL	
A13D	04E	C=0 ALL	
A13E	0A2	A<>C @PT	
A13F	0AE	A<>C ALL	
A140	2F0	WRITDATA	the number in the non-fixed cell is replaced by 0

A141	1A2	A=A-1 @PT	
A142	342	?A#0 @PT	
A143	36B	JNC -19d	if the last tested digit was 1, we again go to the previous non-fixed cell
A144	046	C=0 S&X	
A145	270	RAMSLCT	
A146	278	READ 9(Q)	
A147	08E	B=A ALL	
A148	10E	A=C ALL	
A149	322	?A<B @PT	
A14A	01B	JNC +03	if the digit in the previous cell was, say 8, we must recreate 777777777 which will be stored in CPU register M ( loop2 )
A14B	14E	A=A+C ALL	
A14C	3EB	JNC -03	
A14D	0B0	C=N ALL	
A14E	270	RAMSLCT	
A14F	038	READATA	
A150	0AE	A<>C ALL	
A151	2F3	JNC -34d	[LOOP2]
A152	265	?NCXQ	
A153	020	->0899	[BLINK1]
A154	35C	PT=12	we get here if we must go to the "previous" register (R01->R02->R03 ... )
A155	0B0	C=N ALL	
A156	226	C=C+1 S&X	
A157	23A	C=C+1 M	
A158	070	N=C ALL	
A159	27C	RCR 9	
A15A	106	A=C S&X	
A15B	0BC	RCR 5	
A15C	160	?LOWBAT	the 4 instructions written in yellow on the left
A15D	360	?C RTN	will stop the routine if the batteries are low
A15E	3CC	?KEY	or if you press any key
A15F	360	?CRTN	They may be deleted if you have a "newest" HP-41: simply press ENTER^ ON to stop the program
A160	306	?A<C S&X	
A161	27B	JNC -49d	Replace this line by 2A3 JNC-44d if you don't key in the yellow instructions
A162	0B5	?NC GO	shows "DataError" msg
A163	0A2	->282D	[ERRDE]

## Roman Numerals

### Martin-Baillard,

This implementation uses Jean-Marc's FOCAL routines as the basis of the Arabic<>Roman conversions "N>R" and "R>N" - but adds a nice twist by providing a MCODE data entry for Roman numbers. (**PMTR**, "prompt Roman").

The data entry is therefore simplified, as only the valid roman letters (I, V, X, C, L, D, and M) are allowed by the function. You can delete entries using the back arrow, or proceed using R/S – which will leave the roman string in ALPHA and transfer the execution to the Roman to Arabic conversion program (R>A).

In fact this auxiliary function is very similar to the sudoku row value entering described in the previous section – both use the same techniques and share many characteristics.

#### MCODE listing:

BCKARW	369	PORT DEP	DEL char from LCD & Alpha
A505	03C	GO	
A506	162	->A562	[DELCHAR]
Header A507	092	"R	
Header A508	014	"T"	Roman Prompt
Header A509	00D	"M"	
Header A50A	010	"P"	Ángel Martin
<b>PMTR</b>	<b>A50B</b>	<b>345</b>	<b>?NC XQ</b>
A50C	040	->10D1	[CLA]
A50D	3C1	?NC XQ	Enable & Clear Disp
A50E	0B0	->2CF0	[CLLCDE]
A50F	3BD	?NC XQ	Message Line
A510	01C	->07EF	[MESSL]
A511	092	"R:"	"R:" roman clue
A512	220	" "	
RENTRY	115	?NC XQ	Partial Data Entry!
A514	038	->0E45	[NEXT1]
A515	37B	JNC -17d	back arrow pressed
A516	0B0	C=N ALL	PRESSED KEY CODE
A517	106	A=C S&X	
A518	130	LDI S&X	
A519	370	CON:	R/S keycode
A51A	366	?A#C S&X	
A51B	02B	JNC +05	terminate digit entry
A51C	130	LDI S&X	
A51D	030	CON:	ENTER^ keycode
A51E	366	?A#C S&X	



A51F	027	JC +04	terminate digit entry
A520	369	PORT DEP:	
A521	03C	GO	
A522	186	->A586 [	EXIT]
A523	21C	PT= 2	
A524	130	LDI S&X	
A525	200	CON:	SQRT keycode [200]
A526	366	?A#C S&X	
A527	027	JC +04	NO, KEEP LOOKING
A528	130	LDI S&X	
A529	003	"C"	100
A52A	033	JNC +06	way out
A52B	222	C=C+1 @PT	LOG keycode [300]
A52C	366	?A#C S&X	
A52D	027	JC +04	NO, KEEP LOOKING
A52E	130	LDI S&X	
A52F	004	"D"	500
A530	03B	JNC +07	way out
A531	130	LDI S&X	
A532	220	CON:	STO keycode [220]
A533	366	?A#C S&X	
A534	027	JC +04	NO, KEEP LOOKING
A535	130	LDI S&X	
A536	00C	"L"	50
A537	033	JNC +06	way out
A538	222	C=C+1 @PT	RCL keycode [320]
A539	366	?A#C S&X	
A53A	027	JC +04	NO, KEEP LOOKING
A53B	130	LDI S&X	
A53C	00D	"M"	1000
A53D	03B	JNC +07	way out
A53E	130	LDI S&X	
A53F	310	CON:	COS keycode [310]
A540	366	?A#C S&X	
A541	027	JC +04	NO, KEEP LOOKING
A542	130	LDI S&X	
A543	009	"I"	1
A544	03B	JNC +07	way out
A545	130	LDI S&X	
A546	350	CON:	"6" keycode [310]
A547	366	?A#C S&X	
A548	027	JC +04	NO, KEEP LOOKING
A549	130	LDI S&X	
A54A	018	"X"	10
A54B	03B	JNC +07	way out

A54C	130	LDI S&X	
A54D	150	CON:	"4" keycode [310]
A54E	366	?A#C S&X	
A54F	097	JC +18	One Prompt
A550	130	LDI S&X	
A551	016	"V"	5
A552	3E8	WRIT 15(e)	write CHR in display
A553	106	A=C S&X	
A554	130	LDI S&X	
A555	040	CON:	
A556	206	C=C+A S&X	add 40 for Alpha code
A557	39C	PT= 0	
A558	058	G=C @PT,+	
A559	149	?NC XQ	Disable PER, enable RAM
A55A	024	->0952	[ENCP00]
A55B	051	?NC XQ	
A55C	0B4	->2D14	[APNDNW]
A55D	042	C=0 @PT	
A55E	058	G=C @PT,+	reset PTEMP bits
A55F	3D9	?NC XQ	Enable Display (not cleared)
A560	01C	->07F6	[ENLCD]
A561	113	JNC +34	One Prompt
<b>DELCHAR</b>	<b>3B8</b>	<b>READ 14(d)</b>	to delete rightmost chr
A563	149	?NC XQ	Disable PER, enable RAM
A564	024	->0952	[ENCP00]
A565	178	READ 5(M)	
A566	2EE	?C#0 ALL	abort if empty
A567	037	JC +06	
A568	104	CLRF 8	
A569	1B1	?NCXQ	Mainframe Message
A56A	070	->1C6C	[MSGA]
A56B	03C	"NULL"	from table
A56C	0D3	JNC +26d	
DCHR	A56D	238	READ 8(P)
A56E	10E	A=C ALL	
A56F	1F8	READ 7(O)	
A570	0AA	A<>C PT<-	
A571	23C	RCR 2	
A572	2F0	WRIT DATA	
A573	1B8	READ 6(N)	
A574	0AA	A<>C PT<-	
A575	23C	RCR 2	Delete Alpha last chr
A576	2F0	WRIT DATA	(from Ken Emery's book)
A577	178	READ 5(M)	
A578	04A	C=0 PT<-	

A579	0AA	A<>C PT<-		
A57A	23C	RCR 2		
A57B	2F0	WRIT DATA		
A57C	0AE	A<>C ALL		
A57D	23C	RCR 2		
A57E	228	WRIT 8(P)		
A57F	042	C=0 @PT		
A580	058	G=C @PT,+	reset PTEMP bits	
A581	3D9	?NC XQ	Enable Display (not cleared)	
A582	01C	->07F6	[ENLCD]	
A583	369	PORT DEP:	display ONE prompt	
A584	03C	GO		
A585	113	->A513	[REENTRY]	
<b>EXIT</b>	<b>A586</b>	<b>3D9</b>	<b>?NC XQ</b>	Clear LCD
A587	0B0	->2CF6	[CLRLCD]	
A588	149	?NC XQ	Disable PER, enable RAM	
A589	024	->0952	[ENCP00]	
A58A	261	?NC XQ	Reset keyboard	
A58B	000	->0098	[RSTKB]	
A58C	215	?NC XQ	Reset BIT sequence	
A58D	00C	->0385	[RSTSQ]	
A58E	178	READ 5(M)		
A58F	2EE	?C#0 ALL		
A590	3A0	?NC RTN	abort if empty	
A591	191	?NC XQ		
A592	00C	->0364	[XAVIEW]	
A593	35D	?NC XQ	get this address	
A594	000	->00D7	[PCTOC]	
A595	03C	RCR 3	page number to C<3>	
A596	10E	A=C ALL		
A597	130	LDI S&X		
A598	010	CON: 17		
A599	146	A=A+C S&X	A[(3-0) = addr of first user code word	
A59A	2F9	?NC GO		
A59B	0BE	->2FBE	[XRM20]	

Note the direct transfer from the PMTR to the FOCAL routine, using a call to the OS routine [XRM20], assuming that the first FOCAL instruction on "R>N" is located 17 bytes below the point of call.

### FOCAL drivers:

**00 LBL "R>N"**

01 CLST  
 02\*LBL 01  
 03 ATOX  
 04 X=0?  
 05 GTO 00  
 06 RDN  
 07 XEQ IND T  
 08 X<>Y  
 09 X<Y?  
 10 CHS  
 11 ST+ Z  
 12 RDN  
 13 GTO 01  
 14\*LBL 77  
 15 RTN  
 16\*LBL 68  
 17 5 E2  
 18 RTN  
 19\*LBL 67  
 20 RTN  
 21\*LBL 76  
 22 50  
 23 RTN  
 24\*LBL 88  
 25 RTN  
 26\*LBL 86  
 27 5  
 28 RTN  
 29\*LBL 73  
 30 RTN  
 31\*LBL 00  
 32 +  
 33 +  
 34 CLD  
 35 END

**00 LBL "N>R"**

01 CLA  
 02\*LBL 01  
 03 E3  
 04 X>Y?  
 05 GTO 00

06 -  
 07 ""M"  
 08 GTO 01  
 09\*LBL 00  
 10 CLX  
 11 9 E2  
 12 X<=Y?  
 13 ""CM"  
 14 X>Y?  
 15 CLX  
 16 -  
 17 5 E2  
 18 X<=Y?  
 19 ""D"  
 20 X>Y?  
 21 CLX  
 22 -  
 23 4 E2  
 24 X<=Y?  
 25 ""CD"  
 26 X>Y?  
 27 CLX  
 28 -  
 29\*LBL 02  
 30 E2  
 31 X>Y?  
 32 GTO 00  
 33 -  
 34 ""C"  
 35 GTO 02  
 36\*LBL 00  
 37 CLX  
 38 90  
 39 X<=Y?  
 40 ""XC"  
 41 X>Y?  
 42 CLX  
 43 -  
 44 50  
 45 X<=Y?  
 46 ""L"  
 47 X>Y?  
 48 CLX

49 -  
 50 40  
 51 X<=Y?  
 52 ""XL"  
 53 X>Y?  
 54 CLX  
 55 -  
 56\*LBL 03  
 57 E1  
 58 X>Y?  
 59 GTO 00  
 60 -  
 61 ""X"  
 62 GTO 03  
 63\*LBL 00  
 64 DSE X  
 65 X<=Y?  
 66 ""IX"  
 67 X>Y?  
 68 CLX  
 69 -  
 70 5  
 71 X<=Y?  
 72 ""V"  
 73 X>Y?  
 74 CLX  
 75 -  
 76 4  
 77 X<=Y?  
 78 ""IV"  
 79 X>Y?  
 80 CLX  
 81 -  
 82\*LBL 04  
 83 E  
 84 X>Y?  
 85 GTO 00  
 86 -  
 87 ""I"  
 88 GTO 04  
 89\*LBL 00  
 90 AVIEW  
 91 END

## Splash Screen (Scrolling Message)

Nelson F. Crowle

Those of you with MCODE experience know how idiosyncratic the LCD display is, and if you don't have the personal experience you can take my word for it. It's impressive to imagine how Nelson came up with this jewel of a routine, but we're glad he wrote it! *Note that V41 emulator has a bug that prevents it from running properly*, and it will hung up the program.

The listing below uses the text "LIBRARY#4" for the scrolling message – you can replace it with your own text as appropriate. Also note that provision is made for the TURBO settings on the CL to allow to the necessary time for the displaying action to show.

Header 4705	<b>093</b>	<b>"S"</b>	
Header 4706	<b>001</b>	<b>"A"</b>	SPLASH
Header 4707	<b>00C</b>	<b>"L"</b>	
Header 4708	<b>010</b>	<b>"P"</b>	
Header 4709	<b>093</b>	<b>"S"</b>	
Header 470A	<b>02D</b>	<b>"_"</b>	Nelson C. Crowle
<b>SPLASH 4710</b>	<b>063</b>	<b>JNC +12d</b>	
MSGTXT	1E2	"**"	1C7
470D	005	"E"	"M"
470E	00E	"N"	" "
470F	009	"I"	"V"
4710	00C	"L"	"E"
4711	02D	"_"	"R"
4712	00E	"N"	"_"
4713	00F	"O"	"4"
4714	022	""	"#"
4715	034	"4"	"B"
4716	023	"#"	"L"
SPLASH 4717	35D	?NC XQ	
4718	000	->00D7	[PCTOC]
4719	27A	C=C-1 M	backup two bytes
471A	27A	C=C-1 M	to beginning of string
471B	0EE	B<>C ALL	save it in B[M]
SPLSH4 471C	37D	?NC XQ	Cancels TURBO mode
471D	13C	->4FDF	[TURBOO]
471E	3C1	?NC XQ	Enable and Clear Display
471F	0B0	->2CF0 [	CLLCDE]
4720	130	LDI S&X	
4721	107	"**"	little "T"
4722	31C	PT= 1	
4723	3A8	WRIT 14(d)	Places it (Long) into the LEFT
4724	3B8	READ 14(d)	Fetch Right Long: moves right to left
LB_A09D	0CE	C=B ALL	
4726	27A	C=C-1 M	previous address
4727	0EE	B<>C ALL	update repository
4728	0CE	C=B ALL	keep it in C

	4729	330	FETCH S&X	read byte value
	472A	358	ST=C XP	place it in ST
	472B	28C	?FSET 7	is it > "OFF"?
	472C	023	JNC +04	no, skip to A0A8
	472D	284	CLRF 7	clear it
	472E	398	C=ST XP	restore byte to C[S&X]
	472F	288	SETF 7	flag this fact
[LB_A0A8		3E8	WRIT 15(e)	write it to RIGHT end
	4731	28C	?FSET 7	was it last one?
	4732	10F	JC +33	yes, we're done -> A0CB
LB_A0AB		130	LDI S&X	
	4734	100	CON:	
	4735	266	C=C-1 S&X	count until 100(hex)
	4736	3FB	JNC -01	[A0AD]
	4737	130	LDI S&X	
	4738	020	" "	blank space
	4739	106	A=C S&X	store in A[S&X]
LB_A0B2		3B8	READ 14(d)	read right chr and move it to left
	473B	366	?A#C S&X	is it a space?
	473C	3F3	JNC -02	yes, read next chr. -> A0B2
	473D	158	M=C ALL	no, save chr1 in M
	473E	3B8	READ 14(d)	read chr2 and move to left
	473F	366	?A#C S&X	is it a space?
	4740	05F	JC +11	yes, jump over to A0C3
	4741	198	C=M ALL	no, recall chr1
	4742	3E8	WRIT 15(e)	write it to RIGHT end
	4743	130	LDI S&X	
	4744	020	" "	blank space
	4745	3E8	WRIT 15(e)	write it to RIGHT end
LB_A0BE		3F8	READ 15(e)	Fetch Left Long
	4747	2E2	?C#0 @PT	
	4748	3F7	JC -02	[A0BE]
	4749	3A8	WRIT 14(d)	
	474A	34B	JNC -23 d	[A0AB]
LB_A0C3		3E8	WRIT 15(e)	write it to RIGHT end
	474C	198	C=M ALL	recall chr1 for real
	474D	3E8	WRIT 15(e)	write it to RIGHT end
LB_A0C6		3F8	READ 15(e)	Fetch Left Long
	474F	366	?A#C S&X	
	4750	3F3	JNC -02	[A0C6]
	4751	3A8	WRIT 14(d)	
	4752	29B	JNC -45d	next character, A09D
LB_A0CB		38D	?NC XQ	Sets TURBO mode
	4754	13C	->4FE3	[TURB50]
STMSG4		149	?NC XQ	Enables Chip0
	4756	024	->0952	[ENCP00]
	4757	1F9	?NC GO	Set MSG Flag
	4758	00E	->-037E	[STMSGF]



#	XROM	FNC#	Module	Program	Description	Field	Author	Source	Type	CL id#
1	8	0	CHESS	-CH 136	Section Header	Admin	Ángel Martín	This Module	MCODE	CHESS
2	8	1	CHESS	"CHESS"	Starts New Game	Checked	Ángel Martín	This Module	FOCAL	CHESS
3	8	2	CHESS	"AUX"	Auxiliary for CHESS	Checked	Claude Roeltgen	Author's Collection	FOCAL	CHESS
4	8	3	CHESS	"P1002"	routine for CHESS	Checked	Claude Roeltgen	Author's Collection	FOCAL	CHESS
5	8	4	CHESS	"P1018"	routine for CHESS	Checked	Claude Roeltgen	Author's Collection	FOCAL	CHESS
6	8	5	CHESS	"P1125"	routine for CHESS	Checked	Claude Roeltgen	Author's Collection	FOCAL	CHESS
7	8	6	CHESS	"P1288"	routine for CHESS	Checked	Claude Roeltgen	Author's Collection	FOCAL	CHESS
8	8	7	CHESS	"P1338"	routine for CHESS	Checked	Claude Roeltgen	Author's Collection	FOCAL	CHESS
9	8	8	CHESS	"P2049"	routine for CHESS	Checked	Claude Roeltgen	Author's Collection	FOCAL	CHESS
10	8	9	CHESS	"P2287"	routine for CHESS	Checked	Claude Roeltgen	Author's Collection	FOCAL	CHESS
11	8	10	CHESS	"BOARD"	Sets up board game	Checked	Claude Roeltgen	Author's Collection	FOCAL	CHESS
12	8	11	CHESS	"LEVEL"	Selects level 1-4	Checked	Ángel Martín	This Module	FOCAL	CHESS
13	8	12	CHESS	"COLOR"	Chooses color B/W	Checked	Ángel Martín	This Module	FOCAL	CHESS
14	8	13	CHESS	"HPWHITE"	Start config 1	Checked	Claude Roeltgen	Author's Collection	FOCAL	CHESS
15	8	14	CHESS	"HPBLACK"	start config 2	Checked	Claude Roeltgen	Author's Collection	FOCAL	CHESS
16	8	15	CHESS	"B^W"	routine for CHESS	Checked	Ángel Martín	This Module	FOCAL	CHESS
17	8	16	CHESS	"AMOVE_	Inputs Move	Checked	Ángel Martín	This Module	MCODE	CHESS
18	8	17	CHESS	-BRD GAMES	Section Header	Admin	Ángel Martín	This Module	MCODE	CHESS
19	8	18	CHESS	"8Q"	8 Queens	Checked	Kiyoshi Akima	PPCCJ V8N6	FOCAL	CHESS
20	8	19	CHESS	"8QP"	Routine for 8Q	Checked	Kiyoshi Akima	PPCCJ V8N6	FOCAL	CHESS
21	8	20	CHESS	"8QS"	Eight Queens	Checked	Stefan Fegert	Programmsammlung Book	FOCAL	CHESS
22	8	21	CHESS	"CHCLK"	Chess Clock	Checked	F. Fisher	Prisma Magazine	FOCAL	CHESS
23	8	22	CHESS	"CDAT"	Chess Load Data	Checked	Ángel Martín	This Module	FOCAL	CHESS
24	8	23	CHESS	"CHKERS"	Checkers	Checked	Valentin Albillo	PPCCJ V8N1	FOCAL	CHESS
25	8	24	CHESS	"CPRT"	Chess Print Board	Checked	Valentin Albillo	PPCCJ V8N6	FOCAL	CHESS
26	8	25	CHESS	"KNIGHT"	Knight Moves	Checked	JM Baillard	hp41.programs	FOCAL	CHESS
27	8	26	CHESS	"KPRT"	Checkers Print board	Checked	Valentin Albillo	PPCCJ V8N1	FOCAL	CHESS
28	8	27	CHESS	"MINCHS"	Mini-Chess 5x5	Checked	Valentin Albillo	PPCCJ V8N6	FOCAL	CHESS
29	8	28	CHESS	"SCHES"	Simplified Chess	Checked	Paul Dale	HP Museum Forums	FOCAL	CHESS
30	8	29	CHESS	-AUX FNS	Section Header	Admin	Ángel Martín	This Module	MCODE	CHESS
31	8	30	CHESS	E3/E+	builds pointer	Utils	Ángel Martín	SandMath	MCODE	CHESS
32	8	31	CHESS	LEFT	Moves LCD left	utils	Nelson F. Crowle	NFC ROM	MCODE	CHESS
33	8	32	CHESS	GOOSE	Puts left-goose char in LCD	Utils	Nelson F. Crowle	NFC ROM	MCODE	CHESS
34	8	33	CHESS	CLAXON	sounds claxon	Sounds	Mark Power	Data File	MCODE	CHESS
35	8	34	CHESS	RASP	sounds rasping sound	Sounds	Mark Power	Data File	MCODE	CHESS
36	8	35	CHESS	RNG	Random Number w/Timer	Utils	JM Baillard	hp41.programs	MCODE	CHESS
37	8	36	CHESS	"JMIT"	Initialization	Utils	Ángel Martín	This Module	FOCAL	CHESS
38	8	37	CHESS	"VREG"	View Registers	Utils	Ángel Martín	This Module	FOCAL	CHESS
39	8	38	CHESS	Y/N?	Yes/No Answer	Utils	Ángel Martín	CL Utils	MCODE	CHESS
40	8	39	CHESS	"pp"	Pawn Promotion	Checked	Ángel Martín	This Module	FOCAL	CHESS



41	8	40	CHES	"7CHS"	hp-67 End Game	Checked	Valentin Albillo	PPCIV	FOCAL	CHES
42	8	0	RUBIKS	-RUBIKS CB	Section header	Admin	Ángel Martin	This Module	MCODE	RUBK
43	8	1	RUBIKS	"CUBE1"	Example1	Puzzles	Ángel Martin	This Module	FOCAL	RUBK
44	8	2	RUBIKS	"CUBE2"	Example2	Puzzles	Ángel Martin	This Module	FOCAL	RUBK
45	8	3	RUBIKS	"RUBIK"	Main Data Entry	Puzzles	Martin-Perry	This Module	FOCAL	RUBK
46	8	4	RUBIKS	"R1"	Stage -1	Puzzles	Julian Perry	PPCJ V9N2p23 -	FOCAL	RUBK
47	8	5	RUBIKS	"R2"	Stage-2	Puzzles	Julian Perry	PPCJ V9N2p23 -	FOCAL	RUBK
48	8	6	RUBIKS	"S"	Display/Print	Puzzles	HJ Gessler	PPCJ V9N2p23 -	FOCAL	RUBK
49	8	7	RUBIKS	"T"	Calculate Next Move	Puzzles	Julian Perry	PPCJ V9N2p23 -	FOCAL	RUBK
50	8	8	RUBIKS	"X"	Initialize X Axis - R01-R04	Puzzles	Julian Perry	PPCJ V9N2p23 -	FOCAL	RUBK
51	8	9	RUBIKS	"Y"	Initialize Y Axis - R01-R04	Puzzles	Julian Perry	PPCJ V9N2p23 -	FOCAL	RUBK
52	8	10	RUBIKS	"Z"	Initialize Z Axis - R01-R04	Puzzles	Julian Perry	PPCJ V9N2p23 -	FOCAL	RUBK
53	8	11	RUBIKS	^CROW ---	Enter Color ROW	utils	Ángel Martin	This Module	MCODE	RUBK
54	8	12	RUBIKS	AINT	ARCL Integer	utils	Frits Ferwerda	ML ROM	MCODE	RUBK
55	8	13	RUBIKS	CLAXON	Claxon Sound	Sounds	Mark Power	Data File	MCODE	RUBK
56	8	14	RUBIKS	E3/3+	Builds pointer	utils	Ángel Martin	SandMath	MCODE	RUBK
57	8	15	RUBIKS	GOOSE	Puts Left Goose in Display	utils	Nelson F. Crowle	NFC ROM	MCODE	RUBK
58	8	16	RUBIKS	LEFT	Moves LCD to the left	utils	Nelson F. Crowle	NFC ROM	MCODE	RUBK
59	8	17	RUBIKS	RASP	Rasping sound	Sounds	Mark Power	Data File	MCODE	RUBK
60	8	18	RUBIKS	RNG	Random Number w/ Timer	Utils	JM Baillard	hp41. programs	MCODE	RUBK
61	8	19	RUBIKS	"VREG"	View Registers	utils	Ángel Martin	This Module	FOCAL	RUBK
62	9	0	QUEENS NN	-QUEENS NxN	Section Header	Admin	Ángel Martin	This Module	MCODE	QUEN
63	9	1	QUEENS NN	"2DNQ"	2D N-Queens Problem	Checked	JM Baillard	hp41. programs	FOCAL	QUEN
64	9	2	QUEENS NN	"2DNQ+"	Driver for 2DNQ	Checked	Ángel Martin	hp41. programs	FOCAL	QUEN
65	9	3	QUEENS NN	"3DNQ"	3D N-Queens Problem	Checked	JM Baillard	hp41. programs	FOCAL	QUEN
66	9	4	QUEENS NN	"3DNQ+"	Driver for 3DNQ	Checked	Ángel Martin	hp41. programs	FOCAL	QUEN
67	9	5	QUEENS NN	"3DNQ?"	Checks for valid arrangement	Checked	JM Baillard	hp41. programs	FOCAL	QUEN
68	9	6	QUEENS NN	"N?"		Checked	JM Baillard	hp41. programs	FOCAL	QUEN
69	9	7	QUEENS NN	"N?+"		Checked	JM Baillard	hp41. programs	FOCAL	QUEN
70	9	8	QUEENS NN	"QUEEN"		Checked	JM Baillard	hp41. programs	FOCAL	QUEN
71	9	9	QUEENS NN	"QUEEN2"		Checked	JM Baillard	hp41. programs	FOCAL	QUEN
72	9	10	QUEENS NN	"8Q"	Eight Queens	Checked	Kiyoshi Akima	PPCJ V8N6	FOCAL	QUEN
73	9	11	QUEENS NN	"8QP"	Routine for 8Q	Checked	Kiyoshi Akima	PPCJ V8N6	FOCAL	QUEN
74	9	12	QUEENS NN	"8QS"	8-Queens (2)	Checked	Stefan Fegert	Programmsammlung Book	FOCAL	QUEN
75	9	13	QUEENS NN	-SQUARES 1B	Section Header	Admin	Ángel Martin	This Module	MCODE	QUEN
76	9	14	QUEENS NN	"6PAWN"	Hexapawn	Board Games	HP Co.	Games Solutions Books	FOCAL	QUEN
77	9	15	QUEENS NN	"MAZE"	Builds Maze	Mazes	JM Baillard	hp41. programs	FOCAL	QUEN
78	9	16	QUEENS NN	"MAZE+"	Driver for MAZE	Mazes	Ángel Martin	hp41. programs	FOCAL	QUEN
79	9	17	QUEENS NN	"MAG*XM"	Magic Multiplication - X-Mem	Squares	Martin-Baillard	hp41. programs	FOCAL	QUEN
80	9	18	QUEENS NN	"MAG*RG"	Magic Multiplication - Registers	Squares	Martin-Baillard	hp41. programs	FOCAL	QUEN
81	9	19	QUEENS NN	"MAGIC"	Magic Squares	Squares	JM Baillard	hp41. programs	FOCAL	QUEN

82	9	20	QUEENS NN	"PMAGIC"	Pan-magic Squares	Squares	JM Baillard	hp41.programs	FOCAL	QUEN
83	9	21	QUEENS NN	E3/E+	builds pointer	Utils	Ángel Martin	SandMath	MCODE	QUEN
84	9	22	QUEENS NN	LEFT	Moves display to left	Utils	Nelson F. Crowle	NFC ROM	MCODE	QUEN
85	9	23	QUEENS NN	GOOSE	Puts Left Goose in Display	Utils	Nelson F. Crowle	NFC ROM	MCODE	QUEN
86	9	24	QUEENS NN	"VREG"	View Data Registers	Utils	Ángel Martin	This Module	FOCAL	QUEN
87	9	25	QUEENS NN	"VXMF"	View Data File regs	Utils	JM Baillard	hp41.programs	FOCAL	QUEN
88	9	26	QUEENS NN	RNG	Random Number w/ Timer	Utils	JM Baillard	hp41.programs	MCODE	QUEN
89	9	27	QUEENS NN	"XMAZE"	X-Fns. Maze	Mazes	Erik Christensen	PPCC1V10N5_p31	FOCAL	QUEN
90	10	0	GAMES 1A	GAMES 1A	Section Header	Admin	HP Co.	Games Pac	MCODE	GAME
91	10	1	GAMES 1A	"BAGELS"	Bagels		HP Co.	Games Pac	FOCAL	GAME
92	10	2	GAMES 1A	"BIOR"	Biorythms		HP Co.	Games Pac	FOCAL	GAME
93	10	3	GAMES 1A	"BIOF"	routine for BIOR		HP Co.	Games Pac	FOCAL	GAME
94	10	4	GAMES 1A	"CRAPS"	A Dice game		HP Co.	Games Pac	FOCAL	GAME
95	10	5	GAMES 1A	"HANG"	Hangman Game	Puzzles	HP Co.	Games Pac	FOCAL	GAME
96	10	6	GAMES 1A	"PINBALL"	Pinball		HP Co.	Games Pac	FOCAL	GAME
97	10	7	GAMES 1A	"SWAR"	Star Wars		HP Co.	Games Pac	FOCAL	GAME
98	10	8	GAMES 1A	"SUBHUNT"	Submarine Hunt		HP Co.	Games Pac	FOCAL	GAME
99	10	9	GAMES 1A	"BOOM"	SFX routine		HP Co.	Games Pac	FOCAL	GAME
100	10	10	GAMES 1A	"INI"	Initialization routine		HP Co.	Games Pac	FOCAL	GAME
101	10	11	GAMES 1A	"P"			HP Co.	Games Pac	FOCAL	GAME
102	10	12	GAMES 1A	"SIZE?"	Size setting routine		HP Co.	Games Pac	FOCAL	GAME
103	10	13	GAMES 1A	"RNDM"	Pseudo-random number		HP Co.	Games Pac	FOCAL	GAME
104	10	14	GAMES 1A	"RNDMW"			HP Co.	Games Pac	FOCAL	GAME
105	16	0	GAMES_SB1	-GAMES_SB1	Section Header	Admin	Ángel Martin	This Module	MCODE	GSB2
106	16	1	GAMES_SB1	"3DITT"	3D Tic-Tac-Toe	Puzzles	Read Predmore	PPC V7N9p29	FOCAL	GSB2
107	16	2	GAMES_SB1	"DTXR/W"	Card Reader Storage	Utils	HP Co.	Games Solutions Books	FOCAL	GSB2
108	16	3	GAMES_SB1	"FLIP"	Nuber digit reversal	Puzzles	HP Co.	Games Solutions Books	FOCAL	GSB2
109	16	4	GAMES_SB1	"FLIPO"	Variant of Othello (GO)	Board Games	Robert Swanson	Games Solutions Books	FOCAL	GSB2
110	16	5	GAMES_SB1	"GRAVITY"	Planet Lander	Action	HP Co.	Games Solutions Books	FOCAL	GSB2
111	16	6	GAMES_SB1	"HUMAN"	Hexapawn - w/ Player	Action	HP Co.	Games Solutions Books	FOCAL	GSB2
112	16	7	GAMES_SB1	"MACHINE"	Hexapawn - w/ Calculator	Checked	HP Co.	Games Solutions Books	FOCAL	GSB2
113	16	8	GAMES_SB1	"MIM"	Code-Cracker (Master Mind)	Checked	Edward Hume	Games Solutions Books	FOCAL	GSB2
114	16	9	GAMES_SB1	"ORBIT"	Orbit Lander	Puzzles	HP Co.	Games Solutions Books	FOCAL	GSB2
115	16	10	GAMES_SB1	"RT"	Robot Trap	Action	HP Co.	Games Solutions Books	FOCAL	GSB2
116	16	11	GAMES_SB1	"SCATTER"	Scatter	Mazes	HP Co.	Games Solutions Books	FOCAL	GSB2
117	16	12	GAMES_SB1	"SIMON"	Simon	Action	HP Co.	Games Solutions Books	FOCAL	GSB2
118	16	13	GAMES_SB1	"R"	auxiliary for SIMON	Puzzles	HP Co.	Games Solutions Books	FOCAL	GSB2
119	16	14	GAMES_SB1	"TONES"	auxiliary for SIMON	Puzzles	HP Co.	Games Solutions Books	FOCAL	GSB2
120	16	15	GAMES_SB1	"?"	auxiliary for SIMON	Puzzles	HP Co.	Games Solutions Books	FOCAL	GSB2
121	16	16	GAMES_SB1	"WARI"	Mancala - Wari version	Utils	HP Co.	Games Solutions Books	FOCAL	GSB2
122	16	17	GAMES_SB1	"WIZARD"	Pin-ball Machine	Board Games	Craig Pearce	Games Solutions Books	FOCAL	GSB2

123	16	GAMES_SB1	"PDAT"	Pinball Data load	Action	Ángel Martin	Games Solutions Books	FOCAL	GSB2
124	16	GAMES_SB1	"WUMPIUS"	Hunt the Wumpus	Adventure	HP Co.	Games Solutions Books	FOCAL	GSB2
125	10	<b>FUNSTUFF</b>	<b>-PLAYGROUND</b>	<b>Section Header</b>	<b>Admin</b>	<b>Ángel Martin</b>	<b>This Module</b>	<b>MCODE</b>	<b>FUNS</b>
126	10	FUNSTUFF	"3DTT1"	3D Tic-Tac-Toe	Puzzles	HP Co.	Games Solutions Books	FOCAL	FUNS
127	10	FUNSTUFF	<b>AINT</b>	<b>Alpha Integer Part</b>	Utils	<i>Frits Ferwerda</i>	ML ROM	<b>MCODE</b>	FUNS
128	10	FUNSTUFF	"ALIENS"	Alien Invaders	Action	<i>Ramón Cererols</i>	Boletín PontReyes	FOCAL	FUNS
129	10	FUNSTUFF	"BSHIP"	True Battleship	Strategy	<i>Luis Gasperini</i>	UPL #00796C	FOCAL	FUNS
130	10	FUNSTUFF	"FLIP"	Nuber digit reversal	Puzzles	HP Co.	Games Solutions Books	FOCAL	FUNS
131	10	FUNSTUFF	"HANGM"	Hangman w/ X-Functions	Puzzles	<i>Michael Harwood</i>	hpMuseum	FOCAL	FUNS
132	10	FUNSTUFF	"HANOI"	Hanoi Towers	Puzzles	<i>Stefan Fegert</i>	Programmsammlung Book	FOCAL	FUNS
133	10	FUNSTUFF	"MMIND"	Master Mind	Puzzles	<i>JM Baillard</i>	hp41.programs	FOCAL	FUNS
134	10	FUNSTUFF	<b>MIN?</b>	<b>MM Utility</b>	Puzzles	<i>JM Baillard</i>	hp41.programs	<b>MCODE</b>	FUNS
135	10	FUNSTUFF	"ROBOTS"	Robot Trap	Mazes	HP Co.	Games Solutions Books	FOCAL	FUNS
136	10	FUNSTUFF	<b>ROLLERS</b>	<b>High Rollers</b>	Puzzles	<i>Ross Cooling</i>	PPCCJ14N2p21	<b>MCODE</b>	FUNS
137	10	FUNSTUFF	"WARI"	Game of Wari	Board Games	HP Co.	Games Solutions Books	FOCAL	FUNS
138	10	FUNSTUFF	<b>RNG</b>	<b>Random Number w/ Timer</b>	Utils	<i>JM Baillard</i>	hp41.programs	<b>MCODE</b>	FUNS
139	10	FUNSTUFF	<b>7P&lt;-&gt;S</b>	<b>Register Swap</b>	Utils	HP Co.	Card Reader ROM	<b>MCODE</b>	FUNS
140	10	FUNSTUFF	<b>-ALBILLOS</b>	<b>Section Header</b>	<b>Admin</b>	<b>Ángel Martin</b>	<b>This Module</b>	<b>MCODE</b>	FUNS
141	10	FUNSTUFF	"CHDTA"	Chess Data Load	Checked	<i>Ángel Martin</i>	This Module	FOCAL	FUNS
142	10	FUNSTUFF	"CHKERS"	Checkers	Checked	<i>Valentín Albillo</i>	PPCCJ1V8N1	FOCAL	FUNS
143	10	FUNSTUFF	"CPRT"	Chess Print board	Checked	<i>Valentín Albillo</i>	PPCCJ1V8N6	FOCAL	FUNS
144	10	FUNSTUFF	"KPRT"	Checkers Print board	Checked	<i>Valentín Albillo</i>	PPCCJ1V8N1	FOCAL	FUNS
145	10	FUNSTUFF	"MINCHSS"	Mini-Chess 5x5	Checked	<i>Valentín Albillo</i>	PPCCJ1V8N6	FOCAL	FUNS
146	10	FUNSTUFF	"REVERSI"	Reversi (Othello)	Board Games	<i>Valentín Albillo</i>	PPCCJ1V8N3	FOCAL	FUNS
147	10	FUNSTUFF	<b>-ROMANS</b>	<b>Section Header</b>	<b>Admin</b>	<b>Ángel Martin</b>	<b>This Module</b>	<b>MCODE</b>	FUNS
148	10	FUNSTUFF	"N>R"	Arab number to Roman	Funnies	<i>JM Baillard</i>	hp41.programs	FOCAL	FUNS
149	10	FUNSTUFF	<b>PMTR_</b>	<b>Roman Prompt</b>	<b>Funnies</b>	<b>Ángel Martin</b>	<b>This Module</b>	<b>MCODE</b>	FUNS
150	10	FUNSTUFF	"R>N"	Roman to Arab number	Funnies	<i>JM Baillard</i>	hp41.programs	FOCAL	FUNS
151	10	FUNSTUFF	<b>-ACTION</b>	<b>Section Header</b>	<b>Admin</b>	<b>Ángel Martin</b>	<b>This Module</b>	<b>MCODE</b>	FUNS
152	10	FUNSTUFF	"GOUF"	Catch the Goufers	Funnies	<i>Thomas Fange</i>	PPCCJ1V8NGP44	FOCAL	FUNS
153	10	FUNSTUFF	"LANDER"	Planet Lander	Action	HP Co.	Games Solutions Books	FOCAL	FUNS
154	10	FUNSTUFF	"SCATTER"	Scatter	Action	HP Co.	Games Solutions Books	FOCAL	FUNS
155	10	FUNSTUFF	"SIMON"	Simon	Puzzles	HP Co.	Games Solutions Books	FOCAL	FUNS
156	10	FUNSTUFF	"TARG"	Target: War Game	Board Games	<i>Mark Gessner</i>	PPCCJ1V11N9	FOCAL	FUNS
157	10	FUNSTUFF	"ZCAR"	Zcar - An Action Game	Action	<i>Cary E. Reinstein</i>	PPCCJ1V11N1	FOCAL	FUNS
158	10	FUNSTUFF	<b>-CASINO</b>	<b>Section Header</b>	<b>Admin</b>	<b>Ángel Martin</b>	<b>This Module</b>	<b>MCODE</b>	FUNS
159	10	FUNSTUFF	"BJ"	Black Jack	Card Games	<i>Unknown</i>	Swap Disk	FOCAL	FUNS
160	10	FUNSTUFF	"DOMINO"	Dominoes - requires real game	Table Games	<i>Unknown</i>	Swap Disk	FOCAL	FUNS
161	10	FUNSTUFF	"GAMBL"	Gambler	Card Games	<i>Neil G. Jarman</i>	DataFile V2N3 p20	FOCAL	FUNS
162	10	FUNSTUFF	"PETALS"	Petals around the Rose	Funnies	<i>Edward M. Keefe</i>	UPL #00479C	FOCAL	FUNS
163	10	FUNSTUFF	"POKER"	Poker	Card Games	<i>JM Baillard</i>	hp41.programs	FOCAL	FUNS

164	10	FUNSTUFF	"RLTTE"	American Roulette	Table Games	JM Baillard	hp41.programs	FOCAL	FUNS
165	10	FUNSTUFF	"SKK"	Skunkers	Funnies	Gene Wright	Web Site	FOCAL	FUNS
166	10	FUNSTUFF	"SNLD"	Snakes & Ladders	Puzzles	JM Baillard	hp41.programs	FOCAL	FUNS
167	10	FUNSTUFF	"YATZ"	Yathzee	Puzzles	Gene Wright	PPCCJ1V2N5 p39	FOCAL	FUNS
168	10	FUNSTUFF	<b>-PINBALL</b>	<b>Section Header</b>	Admin	Ángel Martin	This Module	<b>MCODE</b>	FUNS
169	10	FUNSTUFF	"WIZARD"	Pinball Wizard	Action	Craig Pearce	hpmuseum	FOCAL	FUNS
170	10	FUNSTUFF	"PDAT"	Pinball Data load	Action	Ángel Martin	This Module	FOCAL	FUNS
171	10	FUNSTUFF	"INIT"	Initialization	Utils	Ángel Martin	This Module	FOCAL	FUNS
172	10	FUNSTUFF	<b>-SLOT MCH</b>	<b>Section Header</b>	Admin	Ángel Martin	This Module	<b>MCODE</b>	FUNS
173	10	FUNSTUFF	"FRUIT"	Fruit Machine	Action	Brian Ward	hpmuseum	FOCAL	FUNS
174	10	FUNSTUFF	"HOLD"	Hold Reels	Action	Brian Ward	hpmuseum	FOCAL	FUNS
175	10	FUNSTUFF	"NUDGE"	Nudge Reels	Action	Brian Ward	hpmuseum	FOCAL	FUNS
176	10	FUNSTUFF	"SPIN"	Spin Reels	Action	Brian Ward	hpmuseum	FOCAL	FUNS
177	10	FUNSTUFF	"WIN"	Winners	Action	Brian Ward	hpmuseum	FOCAL	FUNS
178	10	FUNSTUFF	<b>-SOUND FX</b>	<b>Section Header</b>	Admin	Ángel Martin	This Module	<b>MCODE</b>	FUNS
179	10	FUNSTUFF	<b>BZZON</b>	Buzz mode on	Sounds	Andreas Meyer	Prisma Magazine	<b>MCODE</b>	FUNS
180	10	FUNSTUFF	<b>CLAXON</b>	Claxon Sound	Sounds	Mark Power	Data File	<b>MCODE</b>	FUNS
181	10	FUNSTUFF	<b>NOBZZ</b>	Buzz mode Off	Sounds	Ángel Martin	This Module	<b>MCODE</b>	FUNS
182	10	FUNSTUFF	"MORSE"	Morse Code	Sounds	JM Baillard	hp41.programs	FOCAL	FUNS
183	10	FUNSTUFF	<b>RASP</b>	Rasping sound	Sounds	Mark Power	Data File	<b>MCODE</b>	FUNS
184	10	FUNSTUFF	<b>TONEXY</b>	Configurable TONE	Sounds	Nelson F. Crowle	NFC ROM	<b>MCODE</b>	FUNS
185	10	FUNSTUFF	<b>LEFT</b>	Moves display to left	Utils	Nelson F. Crowle	PPCCJ1V	<b>MCODE</b>	FUNS
186	10	FUNSTUFF	<b>GOOSE</b>	Puts Left Goose in Display	Utils	Nelson F. Crowle	PPCCJ1V	<b>MCODE</b>	FUNS
187	10	FUNSTUFF	<b>E3/E+</b>	x=1,00x	Utils	Ángel Martin	SandMath	<b>MCODE</b>	FUNS
188	10	<b>POKER</b>	<b>-POKER 41X</b>	<b>Section Header</b>	Admin	Ángel Martin	This Module	<b>MCODE</b>	<b>PPOK</b>
189	10	POKER	<b>#\$#</b>	Sorts cards in R00-R0 and evaluates ha	Card Games	JM Baillard	hp41.programs	<b>MCODE</b>	<b>PPOK</b>
190	10	POKER	<b>CARD</b>	Returns card from rng stored in R00	Card Games	JM Baillard	hp41.programs	<b>MCODE</b>	<b>PPOK</b>
191	10	POKER	<b>RNG</b>	Random Number w/ Timer	Utils	JM Baillard	hp41.programs	<b>MCODE</b>	<b>PPOK</b>
192	10	POKER	<b>SRT</b>	Sorts poker7 cards	Card Games	JM Baillard	hp41.programs	<b>MCODE</b>	<b>PPOK</b>
193	10	POKER	<b>VAL</b>	Evaluates Poker7 hand	Card Games	JM Baillard	hp41.programs	<b>MCODE</b>	<b>PPOK</b>
194	10	POKER	"TEXAS"	Driver for POKER	Card Games	Ángel Martin	hp41.programs	FOCAL	<b>PPOK</b>
195	10	POKER	"BANDIT"	Driver for POKER	Card Games	Ángel Martin	hp41.programs	FOCAL	<b>PPOK</b>
196	10	POKER	"DRAW"	Driver for POKER	Card Games	Ángel Martin	hp41.programs	FOCAL	<b>PPOK</b>
197	10	POKER	"POKER"	Main Program	Card Games	JM Baillard	hp41.programs	FOCAL	<b>PPOK</b>
198	10	POKER	"POKN"	Poker with N cards	Card Games	JM Baillard	hp41.programs	FOCAL	<b>PPOK</b>
199	10	POKER	<b>-BLK JACK</b>	<b>Section Header</b>	Admin	Ángel Martin	This Module	<b>MCODE</b>	<b>PPOK</b>
200	10	POKER	"BJ"	Black Jack	Card Games	Unknown	Swap Disks	FOCAL	<b>PPOK</b>
201	10	POKER	"BLJ"	Black Jack	Card Games	JM Baillard	hp41.programs	FOCAL	<b>PPOK</b>
202	10	POKER	"GAMBLR"	Gambler	Card Games	Neil G. Jarman	DataFile V2N3 p20	FOCAL	<b>PPOK</b>
203	10	POKER	"SCHES"	Simplified Chess	Checked	Paul Dale	HP Museum Forums	FOCAL	<b>PPOK</b>
204	16	<b>SUDOKU</b>	<b>-SUDOKU</b>	<b>Section Header</b>	Admin	Ángel Martin	This module	<b>MCODE</b>	<b>SUD1</b>

205	16	1	SUDOKU	^SROW_	Input Row	Puzzles	Ángel Martin	This module	MCODE	SUD1
206	16	2	SUDOKU	ADEL	Alpha Back Space	Utils	Ken Emery	MCODE for Beginners	MCODE	SUD1
207	16	3	SUDOKU	AINT	Append Integer	Utils	Frits Ferwerda	ML ROM	MCODE	SUD1
208	16	4	SUDOKU	ANUMDL	ANUM w/ Deletion	Utils	HP Co.	Extended I/O	MCODE	SUD1
209	16	5	SUDOKU	E3/E+	1,00x	Utils	Ángel Martin	SandMath	MCODE	SUD1
210	16	6	SUDOKU	"GRID"	Creates Sudoku GRID	Puzzles	JM Baillard	hp41. programs	FOCAL	SUD1
211	16	7	SUDOKU	SDK	Fast Solver	Puzzles	JM Baillard	hp41. programs	MCODE	SUD1
212	16	8	SUDOKU	"SDK0"	Example #0	Puzzles	Ángel Martin	hp41. programs	FOCAL	SUD1
213	16	9	SUDOKU	"SDK1"	Example #1	Puzzles	Ángel Martin	hp41. programs	FOCAL	SUD1
214	16	10	SUDOKU	"SDK2"	Example #2	Puzzles	Ángel Martin	hp41. programs	FOCAL	SUD1
215	16	11	SUDOKU	"SDKIN"	Data input	Puzzles	Ángel Martin	This Module	FOCAL	SUD1
216	16	12	SUDOKU	"SDKOUT"	Data Output	Puzzles	Ángel Martin	This Module	FOCAL	SUD1
217	16	13	SUDOKU	"SLSDK"	Slow Solver	Puzzles	JM Baillard	hp41. programs	FOCAL	SUD1
218	16	14	SUDOKU	"SUD"	Housekeeping	Puzzles	Ángel Martin	hp41. programs	FOCAL	SUD1
219	16	15	SUDOKU	"SUDOKU"	Main Program	Puzzles	Ángel Martin	hp41. programs	FOCAL	SUD1
220	16	16	SUDOKU	"SUDRPN"	RPN Program	Puzzles	Ángel Martin	hp41. programs	FOCAL	SUD1
221	16	17	SUDOKU	"SUDVIEW"	View Sudoku	Puzzles	Ángel Martin	hp41. programs	FOCAL	SUD1
222	16	18	SUDOKU	ΣDIG	Digit Sum	Utils	Ángel Martin	SandMath	MCODE	SUD1
223	16	19	SUDOKU	-SOUND FX	Section Header	Admin	Ángel Martin	This module	MCODE	SUD1
224	16	20	SUDOKU	BUZZON	Set Buzzer Mode	Sounds	Andreas Meyer	Data File	MCODE	SUD1
225	16	21	SUDOKU	CLAXON	Claxon sound	Sounds	Mark Power	Data File	MCODE	SUD1
226	16	22	SUDOKU	NOBUZZ	Clear Buzzer Mode	Sounds	Ángel Martin	This Module	MCODE	SUD1
227	16	23	SUDOKU	RASP	Rasping sound	Sounds	Mark Power	Data File	MCODE	SUD1
228	16	24	SUDOKU	RNG	Random Number w/ Timer	Utils	JM Baillard	hp41. programs	MCODE	SUD1
229	16	25	SUDOKU	TONEXY	Configurable TONE	Sounds	JM Baillard	hp41. programs	MCODE	SUD1
230	16	26	SUDOKU	"C-N"	Chords to Notes	Sounds	JM Baillard	hp41. programs	FOCAL	SUD1
231	16	27	SUDOKU	"N-C"	Notes to Chords	Sounds	JM Baillard	hp41. programs	FOCAL	SUD1
232	16	28	SUDOKU	"MORSE"	Morse Code	Sounds	JM Baillard	hp41. programs	FOCAL	SUD1
233	16	29	SUDOKU	"SSB"	Star-spangled Banner	Sounds	JM Baillard	hp41. programs	FOCAL	SUD1
234	16	30	SUDOKU	LEFT	Shifts display left	Utils	Nelson F. Crowle	NFC ROM	MCODE	SUD1
235	16	31	SUDOKU	GOOSE	Puts Left Goose in Display	Utils	Nelson F. Crowle	NFC ROM	MCODE	SUD1
236	16	32	SUDOKU	"MAGIC"	Magic Squares	Squares	JM Baillard	hp41. programs	FOCAL	SUD1
237	16	33	SUDOKU	"PANMG"	Pan-Magic Squares	Squares	JM Baillard	hp41. programs	FOCAL	SUD1
238	9	0	SWPGMS	-NFL & MORE	Section Header	Admin	Ángel Martin	This Module	MCODE	GSWP
239	9	2	SWPGMS	"CHECKUP"	Medical Check - French	Health	Unknown	Swap Disks	FOCAL	GSWP
240	9	3	SWPGMS	"LABY"	Laberynth - French	Mazes	Unknown	Swap Disks	FOCAL	GSWP
241	9	4	SWPGMS	"AR"	routine for LABY	Mazes	Unknown	Swap Disks	FOCAL	GSWP
242	9	5	SWPGMS	"DE"	routine for LABY	Mazes	Unknown	Swap Disks	FOCAL	GSWP
243	9	6	SWPGMS	"DR"	routine for LABY	Mazes	Unknown	Swap Disks	FOCAL	GSWP
244	9	7	SWPGMS	"GA"	routine for LABY	Mazes	Unknown	Swap Disks	FOCAL	GSWP
245	9	8	SWPGMS	"NFL"	Americal Football	Action	Unknown	Swap Disks	FOCAL	GSWP

246	9	9	SWPGMS	"NIM"	Game of Nim	Puzzles	Bob Laughton	Australian Tech Notes V6p11	FOCAL	GSWP
247	9	10	SWPGMS	"PILOTE"	Pilot Driving - French	Action	Unknown	Swap Disks	FOCAL	GSWP
248	9	11	SWPGMS	RNG	Random Number w/ Timer	Utils	JM Baillard	hp41.programs	MCODE	GSWP
249	9	12	SWPGMS	"SWAP"	Swap Digits	Puzzles	Mark Power	DataFile V8N5P15, July/August 1988	FOCAL	GSWP
250	9	13	SWPGMS	"XOR"	XOR Game	Puzzles	Dejan Ristanovic	DataFile V2N1 p18	FOCAL	GSWP
251	9	14	SWPGMS	-PPC GAMES	Section Header	Admin	Ángel Martín	This Module	MCODE	GSWP
252	9	15	SWPGMS	"SUB"	Submarine Hunt	Action	James R. Merrill	PPCJ1 V8N2p18	FOCAL	GSWP
253	9	16	SWPGMS	"HEX"	Cybernetic Hexapawn	Board Games	Marco de Vries	PPCJ1 V8N1P26	FOCAL	GSWP
254	9	17	SWPGMS	"Y"	routine for HEX	Utils	Marco de Vries	PPCJ1 V8N1P26	FOCAL	GSWP
255	9	18	SWPGMS	"M"	routine for HEX	Utils	Marco de Vries	PPCJ1 V8N1p26	FOCAL	GSWP
256	9	19	SWPGMS	"3WAYS"	3-Ways Game	Mazes	L. Stein	PPC V9N4p58	FOCAL	GSWP
257	9	20	SWPGMS	"BT"	Blank Tile	Mazes	L. Stein	PPC V9N4p58	FOCAL	GSWP
258	9	21	SWPGMS	"EM"	routine for 3WAYS	Mazes	L. Stein	PPC V9N4p58	FOCAL	GSWP
259	9	22	SWPGMS	"FM"	routine for 3WAYS	Mazes	L. Stein	PPC V9N4p58	FOCAL	GSWP
260	9	23	SWPGMS	"IM"	routine for 3WAYS	Mazes	L. Stein	PPC V9N4p58	FOCAL	GSWP
261	9	24	SWPGMS	"MP"	routine for 3WAYS	Mazes	L. Stein	PPC V9N4p58	FOCAL	GSWP
262	9	25	SWPGMS	"OB"	routine for 3WAYS	Mazes	L. Stein	PPC V9N4p58	FOCAL	GSWP
263	9	26	SWPGMS	"OC"	Occupied Tile	Mazes	L. Stein	PPC V9N4p58	FOCAL	GSWP
264	9	27	SWPGMS	"R"	routine for 3WAYS	Mazes	L. Stein	PPC V9N4p58	FOCAL	GSWP
265	9	28	SWPGMS	"R7"	routine for 3WAYS	Mazes	L. Stein	PPC V9N4p58	FOCAL	GSWP
266	9	29	SWPGMS	"R8"	routine for 3WAYS	Mazes	L. Stein	PPC V9N4p58	FOCAL	GSWP
267	9	30	SWPGMS	"RN"	Random Number	Mazes	L. Stein	PPC V9N4p58	FOCAL	GSWP
268	9	31	SWPGMS	"S"	routine for 3WAYS	Mazes	L. Stein	PPC V9N4p58	FOCAL	GSWP
269	9	32	SWPGMS	"SM"	routine for 3WAYS	Mazes	L. Stein	PPC V9N4p58	FOCAL	GSWP
270	9	33	SWPGMS	"SW"	routine for 3WAYS	Mazes	L. Stein	PPC V9N4p58	FOCAL	GSWP
271	9	34	SWPGMS	"WG"	routine for 3WAYS	Mazes	L. Stein	PPC V9N4p58	FOCAL	GSWP
272	9	35	SWPGMS	"WN"	Show Wins	Mazes	L. Stein	PPC V9N4p58	FOCAL	GSWP
273	18	0	RAWGMS	-MC GAMES	Section Header	Admin	Ángel Martín	This Module	MCODE	RGME
274	18	1	RAWGMS	FLIPX	Flip characters	Puzzles	C. Vallance	Datafile V11N2p23	MCODE	RGME
275	18	2	RAWGMS	XPLODE	Explosion Game	Puzzles	C. Vallance	Datafile V11N2p23	MCODE	RGME
276	18	3	RAWGMS	TOFIVE	One to Five	Puzzles	C. Vallance	Datafile V11N2p23	MCODE	RGME
277	18	4	RAWGMS	REVERSE	Reverse Digits to Order	Puzzles	C. Vallance	Datafile V11N2p23	MCODE	RGME
278	18	5	RAWGMS	ROLLERS	High Rollers	Puzzles	Ross Cooling	PPCJ1 V14N2p21	MCODE	RGME
279	18	6	RAWGMS	RN#	Random Number	Utils	C. Vallance	Datafile V11N2p23	MCODE	RGME
280	18	7	RAWGMS	TWIX	Pattern Twist	Puzzles	Guillaume Tello	Web Site	MCODE	RGME
281	18	8	RAWGMS	-XTRA GAME	Section Header	Admin	Ángel Martín	This Module	MCODE	RGME
282	18	9	RAWGMS	"ALIENS"	Alien Invaders	Action	Ramón Cerecros	Boletín PontReyes	FOCAL	RGME
283	18	10	RAWGMS	"BIO"	Biorythms	Health	Philippe Beaugrand	UPLE #10411	FOCAL	RGME
284	18	11	RAWGMS	"GALAXIS"	Galaxis Game	Space Games	Heymchr	HP Museum Forums	FOCAL	RGME
285	18	12	RAWGMS	"HANGX"	Hangman w/ X-Functions	Puzzles	Julian Perry	DataFile V1N1 P5	FOCAL	RGME
286	18	13	RAWGMS	"HM"	Original Hangman	Puzzles	W. C. Wickes	PPCJ1 V7N4p31	FOCAL	RGME

287		"LEM"	RAWGMS	14	RAWGMS	"MANCA"	Lander - An orbital Mechanics Game	Action	Alan Stein	UPL #02921D	FOCAL	RGME
288		"MANCA"	RAWGMS	15	RAWGMS	Mancala - Khala version	Table Games	Table Games	Dan K. Briand	HP Museum Forums	FOCAL	RGME
289		"MAZE1"	RAWGMS	16	RAWGMS	Creates Example Maze	Mazes	Mazes	Ángel Martín	This Module	FOCAL	RGME
290		"XMAZE"	RAWGMS	17	RAWGMS	X-Mem Mazes	Mazes	Mazes	Erik Christensen	PPCCJ V10N5 p31	FOCAL	RGME
291		"SN"	RAWGMS	18	RAWGMS	Secret Number	Puzzles	Puzzles	Tom Cadwallader	CHHU N2N1p17, Jan/Feb 1985	FOCAL	RGME
292		"SWAP"	RAWGMS	19	RAWGMS	Swap Digits	Puzzles	Puzzles	Mark Power	DataFile V8N5P15, July/August 1988	FOCAL	RGME
293		"TIR"	RAWGMS	20	RAWGMS	Alien Invasion ?	Funnies	Funnies	Unknown	Swap Disks	FOCAL	RGME
294		"T"	RAWGMS	21	RAWGMS	Routine for TIR	Funnies	Funnies	Unknown	Swap Disks	FOCAL	RGME
295		<b>-SEA BATTLE</b>	RAWGMS	22	RAWGMS	<b>Section Header</b>	Admin	Admin	Ángel Martín	This Module	<b>MCODE</b>	RGME
296		"SBP"	RAWGMS	23	RAWGMS	Sea Battle w/ Printer	Action	Action	Mark Cracknell	Datafile V8N6 p9	FOCAL	RGME
297		"A"	RAWGMS	24	RAWGMS	routine for SBP	Action	Action	Mark Cracknell	Datafile V8N6 p9	FOCAL	RGME
298		"B"	RAWGMS	25	RAWGMS	routine for SBP	Action	Action	Mark Cracknell	Datafile V8N6 p9	FOCAL	RGME
299		"C"	RAWGMS	26	RAWGMS	routine for SBP	Action	Action	Mark Cracknell	Datafile V8N6 p9	FOCAL	RGME
300		"D"	RAWGMS	27	RAWGMS	routine for SBP	Action	Action	Mark Cracknell	Datafile V8N6 p9	FOCAL	RGME
301		"E"	RAWGMS	28	RAWGMS	routine for SBP	Action	Action	Mark Cracknell	Datafile V8N6 p9	FOCAL	RGME
302		<b>-TUNE UP</b>	RAWGMS	29	RAWGMS	<b>Section Header</b>	Admin	Admin	Ángel Martín	This Module	<b>MCODE</b>	RGME
303		BUZZON	RAWGMS	30	RAWGMS	Sets Buzzer Mode	Sounds	Sounds	Andreas Meyer	Data File	<b>MCODE</b>	RGME
304		CLAXON	RAWGMS	31	RAWGMS	Claxon Sound	Sounds	Sounds	Mark Power	Data File	<b>MCODE</b>	RGME
305		NOBUZZ	RAWGMS	32	RAWGMS	Ends Buzzer mode	Sounds	Sounds	Ángel Martín	This Module	<b>MCODE</b>	RGME
306		TFX	RAWGMS	33	RAWGMS	Toggle Flag	Utils	Utils	Ken Emery	MCODE for Beginners	<b>MCODE</b>	RGME
307		RASP	RAWGMS	34	RAWGMS	Rasping sound	Sounds	Sounds	Mark Power	Data File	<b>MCODE</b>	RGME
308		"JBELLS"	RAWGMS	35	RAWGMS	Jingle Bells tune	Sounds	Sounds	Tom Beach	PPCCJ V9N8p7 - Dec 1982	FOCAL	RGME
309		<b>-ACTION GMS</b>	<b>ACTION</b>	0	<b>ACTION</b>	<b>Section Header</b>	Admin	Admin	Ángel Martín	This Module	<b>MCODE</b>	AGAM
310		"BIRD"	ACTION	1	ACTION	Flying Goose	Funnies	Funnies	Donald R. Houchins	PPCCJ V6N6p3	FOCAL	AGAM
311		"C-A"	ACTION	2	ACTION	Constant Arithmetic??	Action	Action	Unknown	Swap Disks	FOCAL	AGAM
312		"CATCHME"	ACTION	3	ACTION	Detective Game	Action	Action	Tom Langland	PPCCJ V12N8p2	FOCAL	AGAM
313		"FIGHT"	ACTION	4	ACTION	Fighters Gun Duel	Action	Action	Ruys Dirk	Dartafile V1N3 P26	FOCAL	AGAM
314		"HR"	ACTION	5	ACTION	High Rollers	Puzzles	Puzzles	Randal C. Gibson	PPCCJ V6N7p47	FOCAL	AGAM
315		"JCG"	ACTION	6	ACTION	Integer Choice Game	Puzzles	Puzzles	Jakub Tartarkiewicz	PPC V7N9P24	FOCAL	AGAM
316		"LL"	ACTION	7	ACTION	routine for LOONER	Action	Action	Mark Gessner	PPCCJ V12N3p26	FOCAL	AGAM
317		"LOONER"	ACTION	8	ACTION	Lunar Lander	Action	Action	Mark Gessner	PPCCJ V12N3p26	FOCAL	AGAM
318		"RACE"	ACTION	9	ACTION	Car Race	Action	Action	Martin Meyer	PRISMA 3/89, p29	FOCAL	AGAM
319		"STAR"	ACTION	10	ACTION	Star Raiders	Space Games	Space Games	George Ruppert	PPCJ V11N1p14 - Jan/Feb 1984	FOCAL	AGAM
320		"SUB"	ACTION	11	ACTION	Submarine Hunt	Action	Action	Brian Steel	DataFile V2N3 p7	FOCAL	AGAM
321		"SWAP"	ACTION	12	ACTION	Swap Digits	Puzzles	Puzzles	Mark Power	DataFile V8N5P15, Jul/Aug 1988	FOCAL	AGAM
322		"TRUCK"	ACTION	13	ACTION	Truck Routes	Action	Action	Kenneth Sharp	Games Solutions Books	FOCAL	AGAM
323		"XOR"	ACTION	14	ACTION	XOR Game	Puzzles	Puzzles	Dejan Ristanovic	DataFile V2N1 p18	FOCAL	AGAM
324		RNG	ACTION	15	ACTION	Random Number w/ Timer	Utils	Utils	JM Baillard	hp41 programs	<b>MCODE</b>	AGAM
325		TFX	ACTION	16	ACTION	Toggle Flag	Utils	Utils	Ken Emery	MCODE for Beginners	<b>MCODE</b>	AGAM
326		<b>-MAH JONG</b>	<b>MAHJONG</b>	0	<b>MAHJONG</b>	<b>Section Header</b>	Admin	Admin	Ángel Martín	This Module	<b>MCODE</b>	<b>MAHJ</b>
327		"THRJONG"	MAHJONG	1	MAHJONG	Three-players Mah-John Scoring	Table Games	Table Games	Colin Leggate	Datafile V6N4 p41	FOCAL	MAHJ

328	10	2	MAHJONG	"FORJONG"	Four Players Mah-Jong Scoring	Table Games	Colin Leggate	Datafile V6N4 p41	FOCAL	MAHJ
329	10	3	MAHJONG	"2X?"	routine for JONG	Table Games	Ángel Martín	This Module	FOCAL	MAHJ
330	10	4	MAHJONG	"EW?"	routine for JONG	Table Games	Ángel Martín	This Module	FOCAL	MAHJ
331	10	5	MAHJONG	"GT?"	routine for JONG	Table Games	Ángel Martín	This Module	FOCAL	MAHJ
332	10	6	MAHJONG	"MJ?"	routine for JONG	Table Games	Ángel Martín	This Module	FOCAL	MAHJ
333	10	7	MAHJONG	"PLYRS?"	routine for JONG	Table Games	Ángel Martín	This Module	FOCAL	MAHJ
334	10	8	MAHJONG	"RB"	routine for JONG	Table Games	Ángel Martín	This Module	FOCAL	MAHJ
335	10	9	MAHJONG	"RCALL"	routine for JONG	Table Games	Ángel Martín	This Module	FOCAL	MAHJ
336	10	10	MAHJONG	"SB"	routine for JONG	Table Games	Ángel Martín	This Module	FOCAL	MAHJ
337	10	11	MAHJONG	"SKP"	routine for JONG	Table Games	Ángel Martín	This Module	FOCAL	MAHJ
338	10	12	MAHJONG	"TR"	routine for JONG	Table Games	Ángel Martín	This Module	FOCAL	MAHJ
339	10	13	MAHJONG	<b>-PRINTER FUN</b>	<a href="#">Section Header</a>	Admin	Ángel Martín	This Module	<b>MCODE</b>	MAHJ
340	10	14	MAHJONG	"BANCHI"	Data set for BANNER	Utils	Jackie Woltering	PPCCJ V9N1p12 - Jan/Feb 1982	FOCAL	MAHJ
341	10	15	MAHJONG	"BANNER"	Banner	Funnies	Charles R. Ammerman	PPCCJ V7N9p32 - Nov 1980	FOCAL	MAHJ
342	10	16	MAHJONG	"BIO-D"	Biorythms - German	Health	Unknown	BBS ROM#3	FOCAL	MAHJ
343	10	17	MAHJONG	"CAL-F"	FAST Calendar printer	Funnies	Roger Hill	PPCCJ V7N8p18 - Oct 1980	FOCAL	MAHJ
344	10	18	MAHJONG	"CRAPS"	A Dice game	Casino	Samuel J. Hartmen	PPCCJ V14N2 p16	FOCAL	MAHJ
345	10	19	MAHJONG	"GORK"	Foxhole Grenades	Action	Philip T. Frohne	PPCCJ V12N8p11 - August 1985	FOCAL	MAHJ
346	10	20	MAHJONG	"RN"	Random Number	Admin	PPC Members	PPC ROM	FOCAL	MAHJ
347	10	21	MAHJONG	"VA"	View Alpha	Admin	PPC Members	PPC ROM	FOCAL	MAHJ
348	10	22	MAHJONG	"MP"		Admin	PPC Members	PPC ROM	FOCAL	MAHJ
349	10	20	MAHJONG	"VAL"	Valentine Printout	Funnies	Roger Hill	PPCCJ V10N1p58 - Jan/Feb 1983	FOCAL	MAHJ
350	11	0	TREKKIES	<b>-STAR TREK</b>	<a href="#">Section Header</a>	Admin	Ángel Martín	This Module	<b>MCODE</b>	TREK
351	11	1	TREKKIES	"SPST"	Advanced Trek	Space Games	John Rausch	PPCJ V7N2P43 - Feb/Mar 1980	FOCAL	TREK
352	11	2	TREKKIES	"EL"	Federation Logic	Space Games	John Rausch	PPCJ V7N2P43 - Feb/Mar 1980	FOCAL	TREK
353	11	3	TREKKIES	"KL"	Klingon Logic	Space Games	John Rausch	PPCJ V7N2P43 - Feb/Mar 1980	FOCAL	TREK
354	11	4	TREKKIES	<b>-4 TREK</b>	<a href="#">Section Header</a>	Admin	Ángel Martín	This Module	<b>MCODE</b>	TREK
355	11	5	TREKKIES	"4TREK"	Star Trek Program	Space Games	Cary E. Reinstejn	PPCCJ V8N2p28 - Mar/Apr 1981	FOCAL	TREK
356	11	6	TREKKIES	"ATTACK"	Game Command	Space Games	Cary E. Reinstejn	PPCCJ V8N2p28 - Mar/Apr 1981	FOCAL	TREK
357	11	7	TREKKIES	"COMPUTE"	Game Command	Space Games	Cary E. Reinstejn	PPCCJ V8N2p28 - Mar/Apr 1981	FOCAL	TREK
358	11	8	TREKKIES	"DIS'NG"	Game Command	Space Games	Cary E. Reinstejn	PPCCJ V8N2p28 - Mar/Apr 1981	FOCAL	TREK
359	11	9	TREKKIES	"PHASER"	Game Command	Space Games	Cary E. Reinstejn	PPCCJ V8N2p28 - Mar/Apr 1981	FOCAL	TREK
360	11	10	TREKKIES	"PHOTON"	Game Command	Space Games	Cary E. Reinstejn	PPCCJ V8N2p28 - Mar/Apr 1981	FOCAL	TREK
361	11	11	TREKKIES	"REPAIR"	Game Command	Space Games	Cary E. Reinstejn	PPCCJ V8N2p28 - Mar/Apr 1981	FOCAL	TREK
362	11	12	TREKKIES	"SCAN"	Game Command	Space Games	Cary E. Reinstejn	PPCCJ V8N2p28 - Mar/Apr 1981	FOCAL	TREK
363	11	13	TREKKIES	"SHIELD"	Game Command	Space Games	Cary E. Reinstejn	PPCCJ V8N2p28 - Mar/Apr 1981	FOCAL	TREK
364	11	14	TREKKIES	"WARP"	Game Command	Space Games	Cary E. Reinstejn	PPCCJ V8N2p28 - Mar/Apr 1981	FOCAL	TREK
365	11	15	TREKKIES	"#INIT"	aux routine	Space Games	Cary E. Reinstejn	PPCCJ V8N2p28 - Mar/Apr 1981	FOCAL	TREK
366	11	16	TREKKIES	<b>-WARP 2</b>	<a href="#">Section Header</a>	Admin	Ángel Martín	This Module	<b>MCODE</b>	TREK
367	11	17	TREKKIES	"DAYS"		Space Games	Rick Garner	hp41.org	FOCAL	TREK
368	11	18	TREKKIES	"DIST"		Space Games	Rick Garner	hp41.org	FOCAL	TREK



369	11	19	TREKKIES	"HRS"	Space Games	Rick Garner	hp41.org	FOCAL	TREK
370	11	20	TREKKIES	"MIN"	Space Games	Rick Garner	hp41.org	FOCAL	TREK
371	11	21	TREKKIES	"MNTH"	Space Games	Rick Garner	hp41.org	FOCAL	TREK
372	11	22	TREKKIES	"SEC"	Space Games	Rick Garner	hp41.org	FOCAL	TREK
373	11	23	TREKKIES	"SLIP"	Space Games	Rick Garner	hp41.org	FOCAL	TREK
374	11	24	TREKKIES	"STAND"	Space Games	Rick Garner	hp41.org	FOCAL	TREK
375	11	25	TREKKIES	"TME2"	Space Games	Rick Garner	hp41.org	FOCAL	TREK
376	11	26	TREKKIES	"TRANS"	Space Games	Rick Garner	hp41.org	FOCAL	TREK
377	11	27	TREKKIES	"ULTRA"	Space Games	Rick Garner	hp41.org	FOCAL	TREK
378	11	28	TREKKIES	"VELC"	Space Games	Rick Garner	hp41.org	FOCAL	TREK
379	11	29	TREKKIES	"WRP2"	Space Games	Rick Garner	hp41.org	FOCAL	TREK
380	11	30	TREKKIES	"YRS"	Space Games	Rick Garner	hp41.org	FOCAL	TREK
381	11	31	TREKKIES	<b>-ULTRA TRK</b>	Admin	Angel Martin	This Module	<b>MICODE</b>	TREK
382	11	32	TREKKIES	"UTREK"	Space Games	Ted W. Beers		FOCAL	TREK
383	11	33	TREKKIES	"UL "	Fed. Logic	Ted W. Beers		FOCAL	TREK
384	11	34	TREKKIES	"RL"	Romulan Logic	Ted W. Beers		FOCAL	TREK
385	11	35	TREKKIES	"ENTRPRZ"	Draws Enterprise	Tom Hooper	PPCCJ.V4N7p18 - May 1980	FOCAL	TREK
386	10	0	<b>SAMLUNG</b>	<b>-SPIEL SAM</b>	Section Header	Angel Martin	This Module	<b>MICODE</b>	<b>GRMIK</b>
387	10	1	SAMLUNG	"BUG"	Kaefer Jagt	Burkhard Oerttel	Programmsammlung Book	FOCAL	GRMIK
388	10	2	SAMLUNG	"CAMEL"	Kamel Abenteuer - German	Stefan Fegert	Programmsammlung Book	FOCAL	GRMIK
389	10	3	SAMLUNG	"HAMU"	Hamurabi Game - German	Unknown	BBS ROM#3	FOCAL	GRMIK
390	10	4	SAMLUNG	"97"	routine for HAMU	Unknown	BBS ROM#3	FOCAL	GRMIK
391	10	5	SAMLUNG	"HOE"	Holz Stocks	Michael Schilli	Programmsammlung Book	FOCAL	GRMIK
392	10	6	SAMLUNG	"JACK"	Black Jack - German	Burkhard Oerttel	Programmsammlung Book	FOCAL	GRMIK
393	10	7	SAMLUNG	"LUNA"	Moon Lander	Frank Leissler	Programmsammlung Book	FOCAL	GRMIK
394	10	8	SAMLUNG	"CL"	routine for LUNA	Frank Leissler	Programmsammlung Book	FOCAL	GRMIK
395	10	9	SAMLUNG	"PR"	routine for LUNA	Frank Leissler	Programmsammlung Book	FOCAL	GRMIK
396	10	10	SAMLUNG	"MBUG"	Moon Buggy - German	Stefan Fegert	Programmsammlung Book	FOCAL	GRMIK
397	10	11	SAMLUNG	"NEU"	routine for MBUG	Stefan Fegert	Programmsammlung Book	FOCAL	GRMIK
398	10	12	SAMLUNG	"L"	routine for MBUG	Stefan Fegert	Programmsammlung Book	FOCAL	GRMIK
399	10	13	SAMLUNG	"R"	routine for MBUG	Stefan Fegert	Programmsammlung Book	FOCAL	GRMIK
400	10	14	SAMLUNG	"S"	routine for MBUG	Stefan Fegert	Programmsammlung Book	FOCAL	GRMIK
401	10	15	SAMLUNG	"FE"	routine for MBUG	Stefan Fegert	Programmsammlung Book	FOCAL	GRMIK
402	10	16	SAMLUNG	"REV"	Zahlenumkehrung	Stefan Fegert	Programmsammlung Book	FOCAL	GRMIK
403	10	17	SAMLUNG	"ROU"	Roulette - German	Michael Schilli	Programmsammlung Book	FOCAL	GRMIK
404	10	18	SAMLUNG	"SLOTII"	Slot Machine - German	Stefan Fegert	Programmsammlung Book	FOCAL	GRMIK
405	10	19	SAMLUNG	<b>-NAVY WAR</b>	Section Header	Angel Martin	This Module	<b>MICODE</b>	GRMIK
406	10	20	SAMLUNG	"CNAV"	Create Data File	Angel Martin	This Module	FOCAL	GRMIK
407	10	21	SAMLUNG	"NAVY"	Main Program - German	Burkhard Oerttel	Programmsammlung Book	FOCAL	GRMIK
408	10	22	SAMLUNG	"NAV2"	Main Program - Part 2	Burkhard Oerttel	Programmsammlung Book	FOCAL	GRMIK
409	10	23	SAMLUNG	"J"	routine for NAVY	Burkhard Oerttel	Programmsammlung Book	FOCAL	GRMIK

410	10	24	SAMLUNG	"K"	routine for NAVY	Utils	Burkhard Oerttel	Programmsammlung Book	FOCAL	GRMIK
411	10	25	SAMLUNG	"M"	routine for NAVY	Utils	Burkhard Oerttel	Programmsammlung Book	FOCAL	GRMIK
412	10	26	SAMLUNG	"N"	routine for NAVY	Utils	Burkhard Oerttel	Programmsammlung Book	FOCAL	GRMIK
413	10	27	SAMLUNG	"P"	routine for NAVY	Utils	Burkhard Oerttel	Programmsammlung Book	FOCAL	GRMIK
414	10	28	SAMLUNG	"V"	routine for NAVY	Utils	Burkhard Oerttel	Programmsammlung Book	FOCAL	GRMIK
415	10	29	SAMLUNG	"W"	routine for NAVY	Utils	Burkhard Oerttel	Programmsammlung Book	FOCAL	GRMIK
416	10	30	SAMLUNG	"#"	routine for NAVY	Utils	Burkhard Oerttel	Programmsammlung Book	FOCAL	GRMIK
417	10	31	SAMLUNG	"?"	routine for NAVY	Utils	Burkhard Oerttel	Programmsammlung Book	FOCAL	GRMIK
418	12	0	DUNGEON	-DUNGEONS	Section Header	Admin	Ángel Martín	This Module	MCODE	ADV1
419	12	1	DUNGEON	"DUNGEON"	Dungeon Game	Adventure	Antonio Lagana	Author's Collection	FOCAL	ADV1
420	12	2	DUNGEON	"D1"	Dungeon 1	Adventure	Antonio Lagana	Author's Collection	FOCAL	ADV1
421	12	3	DUNGEON	"D2"	Dungeon 2	Adventure	Antonio Lagana	Author's Collection	FOCAL	ADV1
422	12	4	DUNGEON	"D3"	Dungeon 3	Adventure	Antonio Lagana	Author's Collection	FOCAL	ADV1
423	12	5	DUNGEON	"D4"	Dungeon 4	Adventure	Antonio Lagana	Author's Collection	FOCAL	ADV1
424	12	6	DUNGEON	"D5"	Dungeon 5	Adventure	Antonio Lagana	Author's Collection	FOCAL	ADV1
425	12	7	DUNGEON	"D6"	Dungeon 6	Adventure	Antonio Lagana	Author's Collection	FOCAL	ADV1
426	12	8	DUNGEON	"D7"	Dungeon 7	Adventure	Antonio Lagana	Author's Collection	FOCAL	ADV1
427	12	9	DUNGEON	"DX"	Generic Dungeon	Adventure	Antonio Lagana	Author's Collection	FOCAL	ADV1
428	12	10	DUNGEON	AJP	Alpha Integer Part	Utils	Frits Ferwerda	ML ROM	MCODE	ADV1
429	12	11	DUNGEON	RNG	Random Number w/ Timer	Utils	JM Baillard	hp41.programs	MCODE	ADV1
430	12	12	DUNGEON	-ADVENTURE	Section Header	Admin	Ángel Martín	This Module	MCODE	ADV1
431	12	13	DUNGEON	"AD"	Adventure Game	Adventure	Bruce Bailey	PPCCJ.V10N8 p40	FOCAL	ADV1
432	12	14	DUNGEON	"SAD1"	Data Set 1	Adventure	Bruce Bailey	PPCCJ.V10N8 p40	FOCAL	ADV1
433	12	15	DUNGEON	"SAD2"	Data Set 2	Adventure	Bruce Bailey	PPCCJ.V10N8 p40	FOCAL	ADV1
434	12	16	DUNGEON	"FFFF"	Game Command	Adventure	Bruce Bailey	PPCCJ.V10N8 p40	FOCAL	ADV1
435	12	17	DUNGEON	"FILL"	Game Command	Adventure	Bruce Bailey	PPCCJ.V10N8 p40	FOCAL	ADV1
436	12	18	DUNGEON	"GET"	Game Command	Adventure	Bruce Bailey	PPCCJ.V10N8 p40	FOCAL	ADV1
437	12	19	DUNGEON	"INVEN"	Game Command	Adventure	Bruce Bailey	PPCCJ.V10N8 p40	FOCAL	ADV1
438	12	20	DUNGEON	"KILL"	Game Command	Adventure	Bruce Bailey	PPCCJ.V10N8 p40	FOCAL	ADV1
439	12	21	DUNGEON	"LOOK"	Game Command	Adventure	Bruce Bailey	PPCCJ.V10N8 p40	FOCAL	ADV1
440	12	22	DUNGEON	"NOSEE"	Game Command	Adventure	Bruce Bailey	PPCCJ.V10N8 p40	FOCAL	ADV1
441	12	23	DUNGEON	"OPEN"	Game Command	Adventure	Bruce Bailey	PPCCJ.V10N8 p40	FOCAL	ADV1
442	12	24	DUNGEON	"PLOVE"	Game Command	Adventure	Bruce Bailey	PPCCJ.V10N8 p40	FOCAL	ADV1
443	12	25	DUNGEON	"PLUGH"	Game Command	Adventure	Bruce Bailey	PPCCJ.V10N8 p40	FOCAL	ADV1
444	12	26	DUNGEON	"POUR"	Game Command	Adventure	Bruce Bailey	PPCCJ.V10N8 p40	FOCAL	ADV1
445	12	27	DUNGEON	"SCORE"	Game Command	Adventure	Bruce Bailey	PPCCJ.V10N8 p40	FOCAL	ADV1
446	12	28	DUNGEON	"THROW"	Game Command	Adventure	Bruce Bailey	PPCCJ.V10N8 p40	FOCAL	ADV1
447	12	29	DUNGEON	"WAVE"	Game Command	Adventure	Bruce Bailey	PPCCJ.V10N8 p40	FOCAL	ADV1
448	12	30	DUNGEON	"XYZZY"	Game Command	Adventure	Bruce Bailey	PPCCJ.V10N8 p40	FOCAL	ADV1
449	12	31	DUNGEON	"N "	Game Command	Adventure	Bruce Bailey	PPCCJ.V10N8 p40	FOCAL	ADV1
450	12	32	DUNGEON	"S"	Game Command	Adventure	Bruce Bailey	PPCCJ.V10N8 p40	FOCAL	ADV1

451	12	DUNGEON	"E"	Game Command	Adventure	Bruce Bailey	PPCCJ\V10N8 p40	FOCAL	ADV1
452	12	DUNGEON	"W"	Game Command	Adventure	Bruce Bailey	PPCCJ\V10N8 p40	FOCAL	ADV1
453	12	DUNGEON	"U"	Game Command	Adventure	Bruce Bailey	PPCCJ\V10N8 p40	FOCAL	ADV1
454	12	DUNGEON	"D"	Game Command	Adventure	Bruce Bailey	PPCCJ\V10N8 p40	FOCAL	ADV1
455	12	DUNGEON	<b>-CAVES &amp; CO.</b>	<b>Section Header</b>	Admin	Ángel Martín	This Module	<b>MICODE</b>	ADV1
456	12	DUNGEON	"ARENA"	Arena of Death	Adventure	Tom Longland	PPCCJ\V12N5p9	FOCAL	ADV1
457	12	DUNGEON	"CASTLE"	Black Castle	Adventure	Legros & Maryssael	HP Museum Forums	FOCAL	ADV1
458	12	DUNGEON	"CAVES"	Caves	Adventure	James Surber	UPL #00900C	FOCAL	ADV1
459	12	DUNGEON	"A"	routine for CAVES	Adventure	James Surber	UPL #00900C	FOCAL	ADV1
460	12	DUNGEON	"HOBBIT"	Hobbit Adventure	Adventure	Dejan Ristanovic	PPCCJ\V10N6p26	FOCAL	ADV1
461	12	DUNGEON	"WOMPIN"	Routine for WOMPUS	Adventure	David Hayden	UPL #00866C - March 1980	FOCAL	ADV1
462	12	DUNGEON	"WOMPUS"	Catch the Wompus	Adventure	David Hayden	UPL #00866C - March 1980	FOCAL	ADV1
463	12	DUNGEON	"WUMPUS"	Hunt the Wumpus	Adventure	HP Co.	Games Solutions Books	FOCAL	ADV1
464	13	<b>ADVENTURE</b>	<b>-TERR DAMM</b>	<b>Section Header</b>	Admin	Ángel Martín	This Module	<b>MICODE</b>	<b>ADV2</b>
465	13	ADVENTURE	"ADVT"	Terror Damm	Adventure	Layne K. Johnson	Games Solutions Books	FOCAL	ADV2
466	13	ADVENTURE	"AD1"	Data Set 1	Adventure	Ángel Martín	This Module	FOCAL	ADV2
467	13	ADVENTURE	"AD2"	Data Set 2	Adventure	Ángel Martín	This Module	FOCAL	ADV2
468	13	ADVENTURE	"AD3"	Data Set 3	Adventure	Ángel Martín	This Module	FOCAL	ADV2
469	13	ADVENTURE	"AT"	Game Command	Adventure	Layne K. Johnson		FOCAL	ADV2
470	13	ADVENTURE	"CL"	Game Command	Adventure	Layne K. Johnson		FOCAL	ADV2
471	13	ADVENTURE	"DR"	Game Command	Adventure	Layne K. Johnson		FOCAL	ADV2
472	13	ADVENTURE	"EX"	Game Command	Adventure	Layne K. Johnson		FOCAL	ADV2
473	13	ADVENTURE	"FE"	Game Command	Adventure	Layne K. Johnson		FOCAL	ADV2
474	13	ADVENTURE	"GE"	Game Command	Adventure	Layne K. Johnson		FOCAL	ADV2
475	13	ADVENTURE	"GI"	Game Command	Adventure	Layne K. Johnson		FOCAL	ADV2
476	13	ADVENTURE	"GO"	Game Command	Adventure	Layne K. Johnson		FOCAL	ADV2
477	13	ADVENTURE	"IN"	Game Command	Adventure	Layne K. Johnson		FOCAL	ADV2
478	13	ADVENTURE	"L"	Game Command	Adventure	Layne K. Johnson		FOCAL	ADV2
479	13	ADVENTURE	"LI"	Game Command	Adventure	Layne K. Johnson		FOCAL	ADV2
480	13	ADVENTURE	"MO"	Game Command	Adventure	Layne K. Johnson		FOCAL	ADV2
481	13	ADVENTURE	"OP"	Game Command	Adventure	Layne K. Johnson		FOCAL	ADV2
482	13	ADVENTURE	"PR"	Game Command	Adventure	Layne K. Johnson		FOCAL	ADV2
483	13	ADVENTURE	"PU"	Game Command	Adventure	Layne K. Johnson		FOCAL	ADV2
484	13	ADVENTURE	"RE"	Game Command	Adventure	Layne K. Johnson		FOCAL	ADV2
485	13	ADVENTURE	"TH"	Game Command	Adventure	Layne K. Johnson		FOCAL	ADV2
486	13	ADVENTURE	"WO "	Game Command	Adventure	Layne K. Johnson		FOCAL	ADV2
487	13	ADVENTURE	"*N"	Game Command	Adventure	Layne K. Johnson		FOCAL	ADV2
488	13	ADVENTURE	"*S"	Game Command	Adventure	Layne K. Johnson		FOCAL	ADV2
489	13	ADVENTURE	"*E"	Game Command	Adventure	Layne K. Johnson		FOCAL	ADV2
490	13	ADVENTURE	"*W"	Game Command	Adventure	Layne K. Johnson		FOCAL	ADV2
491	13	ADVENTURE	"?"	Game Command	Adventure	Layne K. Johnson		FOCAL	ADV2

492	13	ADVENTURE	-DNG DEATH	Section Header	Admin	Angel Martin	This Module	MCODE	ADV2
493	13	ADVENTURE	"DD"	Dungeon of Death	Adventure	Frank Wales	Datafile V8N7p3	FOCAL	ADV2
494	13	ADVENTURE	"A"	routine for DD	Utils	Frank Wales	Datafile V8N7p3	FOCAL	ADV2
495	13	ADVENTURE	"B"	routine for DD	Utils	Frank Wales	Datafile V8N7p3	FOCAL	ADV2
496	13	ADVENTURE	"C"	routine for DD	Utils	Frank Wales	Datafile V8N7p3	FOCAL	ADV2
497	13	ADVENTURE	"D"	routine for DD	Utils	Frank Wales	Datafile V8N7p3	FOCAL	ADV2
498	13	ADVENTURE	"F"	routine for DD	Utils	Frank Wales	Datafile V8N7p3	FOCAL	ADV2
499	13	ADVENTURE	-BLK TOWER	Section Header	Admin	Angel Martin	This Module	MCODE	ADV2
500	13	ADVENTURE	"BTWR"	Black Tower	Adventure	Doug Hansen	PPCCJ1V11N9p12	FOCAL	ADV2
501	13	ADVENTURE	"BTDAT"	Data Loading	Utils	Angel Martin	This Module	FOCAL	ADV2
502	13	ADVENTURE	"P1"	Player 1	Adventure	Angel Martin	PPCCJ1V11N9p12	FOCAL	ADV2
503	13	ADVENTURE	"P2"	Player 2	Adventure	Angel Martin	PPCCJ1V11N9p12	FOCAL	ADV2
504	13	ADVENTURE	"P3"	Player 3	Adventure	Angel Martin	PPCCJ1V11N9p12	FOCAL	ADV2
505	13	ADVENTURE	"P4"	Player 4	Adventure	Angel Martin	PPCCJ1V11N9p12	FOCAL	ADV2
506	13	ADVENTURE	"RF"		Utils	PPC Members	PPC ROM	FOCAL	ADV2
507	13	ADVENTURE	"SE"	Selection wout Replacement	Utils	PPC Members	PPC ROM	FOCAL	ADV2
508	13	ADVENTURE	-SORCERY	Section Header	Admin	Angel Martin	This Module	MCODE	ADV2
509	13	ADVENTURE	"SNS"	Swords & Sorcery	Adventure	Peter O. Johnson		FOCAL	ADV2
510	13	ADVENTURE	RNG	Random Number w/ Timer	Utils	JM Baillard	hp41. programs	MCODE	ADV2
511	13	ADVENTURE	-TWR SKELOS	Section Header	Admin	Angel Martin	This Module	MCODE	ADV2
512	13	ADVENTURE	"TWR"	Tower of Skelos	Adventure	Michael Heinz	PPCCJ1V11N4p14	FOCAL	ADV2
513	13	ADVENTURE	"SKDTA"	Data Loading	Utils	Michael Heinz	PPCCJ1V11N4p14	FOCAL	ADV2
514	13	ADVENTURE	"PK-N"	Data compression	Urils	Michael Heinz	PPCCJ1V11N4p14	FOCAL	ADV2
515	13	ADVENTURE	"JP-N"	Data decompression	Utils	Michael Heinz	PPCCJ1V11N4p14	FOCAL	ADV2
516	13	ADVENTURE	-EF/M CAVES	Section Header	Admin	Angel Martin	This Module	MCODE	ADV2
517	13	ADVENTURE	"CVS"	EFM/Caves	Adventure	Philip T. Frohne	PPCCJ1V14N5p11 - May 1987	FOCAL	ADV2
518	13	ADVENTURE	"MKCVA"	Make Caves File	Utils	Philip T. Frohne	PPCCJ1V14N5p11 - May 1988	FOCAL	ADV2
519	12	ADVENTURE	-GHOST TOWN	Section Header	Admin	Angel Martin	This Module	MCODE	ADV2
520	12	ADVENTURE	"D1"	Ghost Town Adventure	Adventure	Doug Hansen	PPCCJ1V10N3 p33	FOCAL	ADV2
521	12	ADVENTURE	"DIG"	Game Command	Adventure	Doug Hansen	PPCCJ1V10N3 p33	FOCAL	ADV2
522	12	ADVENTURE	"DO"	Game Command	Adventure	Doug Hansen	PPCCJ1V10N3 p33	FOCAL	ADV2
523	12	ADVENTURE	"DR"	Game Command	Adventure	Doug Hansen	PPCCJ1V10N3 p33	FOCAL	ADV2
524	12	ADVENTURE	"DROP"	Game Command	Adventure	Doug Hansen	PPCCJ1V10N3 p33	FOCAL	ADV2
525	12	ADVENTURE	"EA"	Game Command	Adventure	Doug Hansen	PPCCJ1V10N3 p33	FOCAL	ADV2
526	12	ADVENTURE	"FILL"	Game Command	Adventure	Doug Hansen	PPCCJ1V10N3 p33	FOCAL	ADV2
527	12	ADVENTURE	"GET"	Game Command	Adventure	Doug Hansen	PPCCJ1V10N3 p33	FOCAL	ADV2
528	12	ADVENTURE	"HE"	Game Command	Adventure	Doug Hansen	PPCCJ1V10N3 p33	FOCAL	ADV2
529	12	ADVENTURE	"I2"	Game Command	Adventure	Doug Hansen	PPCCJ1V10N3 p33	FOCAL	ADV2
530	12	ADVENTURE	"IL"	Game Command	Adventure	Doug Hansen	PPCCJ1V10N3 p33	FOCAL	ADV2
531	12	ADVENTURE	"INVEN"	Game Command	Adventure	Doug Hansen	PPCCJ1V10N3 p33	FOCAL	ADV2
532	12	ADVENTURE	"LIGHT"	Game Command	Adventure	Doug Hansen	PPCCJ1V10N3 p33	FOCAL	ADV2

533	12	14	ADVENTURE	"LOOK"	Game Command	Adventure	Doug Hansen	PPCCJ V10N3 p33	FOCAL	ADV2
534	12	15	ADVENTURE	"MA"	Game Command	Adventure	Doug Hansen	PPCCJ V10N3 p33	FOCAL	ADV2
535	12	16	ADVENTURE	"MO"	Game Command	Adventure	Doug Hansen	PPCCJ V10N3 p33	FOCAL	ADV2
536	12	17	ADVENTURE	"N"	Game Command	Adventure	Doug Hansen	PPCCJ V10N3 p33	FOCAL	ADV2
537	12	18	ADVENTURE	"OB"	Game Command	Adventure	Doug Hansen	PPCCJ V10N3 p33	FOCAL	ADV2
538	12	19	ADVENTURE	"OP"	Game Command	Adventure	Doug Hansen	PPCCJ V10N3 p33	FOCAL	ADV2
539	12	20	ADVENTURE	"OPEN"	Game Command	Adventure	Doug Hansen	PPCCJ V10N3 p33	FOCAL	ADV2
540	12	21	ADVENTURE	"PUT"	Game Command	Adventure	Doug Hansen	PPCCJ V10N3 p33	FOCAL	ADV2
541	12	22	ADVENTURE	"S"	Game Command	Adventure	Doug Hansen	PPCCJ V10N3 p33	FOCAL	ADV2
542	12	23	ADVENTURE	"SO"	Game Command	Adventure	Doug Hansen	PPCCJ V10N3 p33	FOCAL	ADV2
543	12	24	ADVENTURE	"SR"	Game Command	Adventure	Doug Hansen	PPCCJ V10N3 p33	FOCAL	ADV2
544	12	25	ADVENTURE	"TAKE"	Game Command	Adventure	Doug Hansen	PPCCJ V10N3 p33	FOCAL	ADV2
545	12	26	ADVENTURE	"U"	Game Command	Adventure	Doug Hansen	PPCCJ V10N3 p33	FOCAL	ADV2
546	12	27	ADVENTURE	"VE"	Game Command	Adventure	Doug Hansen	PPCCJ V10N3 p33	FOCAL	ADV2
547	12	28	ADVENTURE	"W"	Game Command	Adventure	Doug Hansen	PPCCJ V10N3 p33	FOCAL	ADV2
548	12	29	ADVENTURE	"WI"	Game Command	Adventure	Doug Hansen	PPCCJ V10N3 p33	FOCAL	ADV2
549	14	0	EXPLORE	-EXPLORE 1A	Section Header	Admin	Angel Martin	This Module	MCODE	GAMX
550	14	1	EXPLORE	"EXPLORE"	Explore Game	Adventure	Bruce Bailey	PPCCJ v12n3P4 - March 1985	FOCAL	GAMX
551	14	2	EXPLORE	"MT"	Make files	Adventure	Bruce Bailey	PPCCJ v12n3P4 - March 1985	FOCAL	GAMX
552	14	3	EXPLORE	"NOSEE"	Game Command	Adventure	Bruce Bailey	PPCCJ v12n3P4 - March 1985	FOCAL	GAMX
553	14	4	EXPLORE	"??"	Game Command	Adventure	Bruce Bailey	PPCCJ v12n3P4 - March 1985	FOCAL	GAMX
554	14	5	EXPLORE	"GET"	Game Command	Adventure	Bruce Bailey	PPCCJ v12n3P4 - March 1985	FOCAL	GAMX
555	14	6	EXPLORE	"GIVE"	Game Command	Adventure	Bruce Bailey	PPCCJ v12n3P4 - March 1985	FOCAL	GAMX
556	14	7	EXPLORE	"DROP"	Game Command	Adventure	Bruce Bailey	PPCCJ v12n3P4 - March 1985	FOCAL	GAMX
557	14	8	EXPLORE	"KILL"	Game Command	Adventure	Bruce Bailey	PPCCJ v12n3P4 - March 1985	FOCAL	GAMX
558	14	9	EXPLORE	"SIT"	Game Command	Adventure	Bruce Bailey	PPCCJ v12n3P4 - March 1985	FOCAL	GAMX
559	14	10	EXPLORE	"SCORE"	Game Command	Adventure	Bruce Bailey	PPCCJ v12n3P4 - March 1985	FOCAL	GAMX
560	14	11	EXPLORE	"EAT"	Game Command	Adventure	Bruce Bailey	PPCCJ v12n3P4 - March 1985	FOCAL	GAMX
561	14	12	EXPLORE	"DRINK"	Game Command	Adventure	Bruce Bailey	PPCCJ v12n3P4 - March 1985	FOCAL	GAMX
562	14	13	EXPLORE	"OPEN"	Game Command	Adventure	Bruce Bailey	PPCCJ v12n3P4 - March 1985	FOCAL	GAMX
563	14	14	EXPLORE	"PLAY"	Game Command	Adventure	Bruce Bailey	PPCCJ v12n3P4 - March 1985	FOCAL	GAMX
564	14	15	EXPLORE	"RUN"	Game Command	Adventure	Bruce Bailey	PPCCJ v12n3P4 - March 1985	FOCAL	GAMX
565	14	16	EXPLORE	"SWIM"	Game Command	Adventure	Bruce Bailey	PPCCJ v12n3P4 - March 1985	FOCAL	GAMX
566	14	17	EXPLORE	"JUMP"	Game Command	Adventure	Bruce Bailey	PPCCJ v12n3P4 - March 1985	FOCAL	GAMX
567	14	18	EXPLORE	"SAVE"	Game Command	Adventure	Bruce Bailey	PPCCJ v12n3P4 - March 1985	FOCAL	GAMX
568	14	19	EXPLORE	"MIKM"	Make file M	Adventure	Bruce Bailey	PPCCJ v12n3P4 - March 1985	FOCAL	GAMX
569	14	20	EXPLORE	"MIKN"	Make file N	Adventure	Bruce Bailey	PPCCJ v12n3P4 - March 1985	FOCAL	GAMX
570	14	21	EXPLORE	"MKS"	Make file S	Adventure	Bruce Bailey	PPCCJ v12n3P4 - March 1985	FOCAL	GAMX
571	14	22	EXPLORE	"MKT"	Make file T	Adventure	Bruce Bailey	PPCCJ v12n3P4 - March 1985	FOCAL	GAMX
572	14	23	EXPLORE	"MK4"	Make file Alpha	Adventure	Bruce Bailey	PPCCJ v12n3P4 - March 1985	FOCAL	GAMX
573	14	24	EXPLORE	"MK5"	Make file Beta	Adventure	Bruce Bailey	PPCCJ v12n3P4 - March 1985	FOCAL	GAMX

574	14	25	EXPLORE	"MK6"	Make file Gamma	Adventure	Bruce Bailey	PPCJ\12n3P4 - March 1985	FOCAL	GAMX
575	14	26	EXPLORE	-TRSR QUEST	Section Header	Admin	Angel Martin	This Module	FOCAL	GAMX
576	14	27	EXPLORE	"TQST"	Treasure Quest	Adventure	Gerard Maynard	HPMuseum DVD	FOCAL	GAMX
577	14	28	EXPLORE	"WG"	Word Game	Puzzles	Mark Gessner	PPCJ\12N11 p20	FOCAL	GAMX
578	14	29	EXPLORE	"YAMS"	A Yatzee Game	Action	JM Baillard	Web Site	FOCAL	GAMX
579	16	0	MONOPLY	-MONOPLY	Section Header	Admin	Angel Martin	This Module	MCODE	MONO
580	16	1	MONOPLY	"AB"			Thomas Rodke		FOCAL	MONO
581	16	2	MONOPLY	"BH"			Thomas Rodke		FOCAL	MONO
582	16	3	MONOPLY	"7I"			Thomas Rodke		FOCAL	MONO
583	16	4	MONOPLY	"CDG"			Thomas Rodke		FOCAL	MONO
584	16	5	MONOPLY	"dc"			Thomas Rodke		FOCAL	MONO
585	16	6	MONOPLY	"FDFB"			Thomas Rodke		FOCAL	MONO
586	16	7	MONOPLY	"FT"			Thomas Rodke		FOCAL	MONO
587	16	8	MONOPLY	"JAIL"			Thomas Rodke		FOCAL	MONO
588	16	9	MONOPLY	"L"			Thomas Rodke		FOCAL	MONO
589	16	10	MONOPLY	"MINPLY"			Thomas Rodke		FOCAL	MONO
590	16	11	MONOPLY	"O"			Thomas Rodke		FOCAL	MONO
591	16	12	MONOPLY	"P"			Thomas Rodke		FOCAL	MONO
592	16	13	MONOPLY	"Q"			Thomas Rodke		FOCAL	MONO
593	16	14	MONOPLY	"R"			Thomas Rodke		FOCAL	MONO
594	16	15	MONOPLY	"RE"			Thomas Rodke		FOCAL	MONO
595	16	16	MONOPLY	"PAY"			Thomas Rodke		FOCAL	MONO
596	16	17	MONOPLY	"SETUP"			Thomas Rodke		FOCAL	MONO
597	16	18	MONOPLY	"SL"			Thomas Rodke		FOCAL	MONO
598	16	19	MONOPLY	"T"			Thomas Rodke		FOCAL	MONO
599	16	20	MONOPLY	"X"			Thomas Rodke		FOCAL	MONO
600	16	21	MONOPLY	"S"			Thomas Rodke		FOCAL	MONO
601	16	22	MONOPLY	-CHEST CRDS	Section Header	Admin	Angel Martin	This Module	MCODE	MONO
602	16	23	MONOPLY	"24"			Thomas Rodke		FOCAL	MONO
603	16	24	MONOPLY	"4"			Thomas Rodke		FOCAL	MONO
604	16	25	MONOPLY	"5"			Thomas Rodke		FOCAL	MONO
605	16	26	MONOPLY	"48"			Thomas Rodke		FOCAL	MONO
606	16	27	MONOPLY	"52"			Thomas Rodke		FOCAL	MONO
607	16	28	MONOPLY	"53"			Thomas Rodke		FOCAL	MONO
608	16	29	MONOPLY	"57"			Thomas Rodke		FOCAL	MONO
609	16	30	MONOPLY	"60"			Thomas Rodke		FOCAL	MONO
610	16	31	MONOPLY	"61"			Thomas Rodke		FOCAL	MONO
611	16	32	MONOPLY	"62"			Thomas Rodke		FOCAL	MONO
612	16	33	MONOPLY	"63"			Thomas Rodke		FOCAL	MONO
613	16	34	MONOPLY	"64"			Thomas Rodke		FOCAL	MONO
614	16	35	MONOPLY	"65"			Thomas Rodke		FOCAL	MONO

615	16	16	36	MONOPLY	"66"	Thomas Rodke	MONO	FOCAL	MONO
616	16	16	37	MONOPLY	"67"	Thomas Rodke	MONO	FOCAL	MONO
617	16	16	38	MONOPLY	"68"	Thomas Rodke	MONO	FOCAL	MONO
618	16	16	39	MONOPLY	"69"	Thomas Rodke	MONO	FOCAL	MONO
619	16	16	40	MONOPLY	"70"	Thomas Rodke	MONO	FOCAL	MONO
620	16	16	41	MONOPLY	"71"	Thomas Rodke	MONO	FOCAL	MONO
621	16	16	42	MONOPLY	"72"	Thomas Rodke	MONO	FOCAL	MONO
622	16	16	43	MONOPLY	"73"	Thomas Rodke	MONO	FOCAL	MONO
623	16	16	44	MONOPLY	"74"	Thomas Rodke	MONO	FOCAL	MONO
624	16	16	45	MONOPLY	"75"	Thomas Rodke	MONO	FOCAL	MONO
625	16	16	46	MONOPLY	"76"	Thomas Rodke	MONO	FOCAL	MONO
626	16	16	47	MONOPLY	"77"	Thomas Rodke	MONO	FOCAL	MONO
627	16	16	48	MONOPLY	"78"	Thomas Rodke	MONO	FOCAL	MONO
628	16	16	49	MONOPLY	"79"	Thomas Rodke	MONO	FOCAL	MONO
629	16	16	50	MONOPLY	"80"	Thomas Rodke	MONO	FOCAL	MONO
630	16	16	51	MONOPLY	"81"	Thomas Rodke	MONO	FOCAL	MONO
631	16	16	52	MONOPLY	"82"	Thomas Rodke	MONO	FOCAL	MONO
632	16	16	53	MONOPLY	"83"	Thomas Rodke	MONO	FOCAL	MONO
633	16	16	54	MONOPLY	"84"	Thomas Rodke	MONO	FOCAL	MONO
634	16	16	55	MONOPLY	"85"	Thomas Rodke	MONO	FOCAL	MONO
635	16	16	56	MONOPLY	<b>-BONUS PGMS</b>	<b>Section Header</b>	<b>Admin</b>	<b>Admin</b>	<b>MONO</b>
636	16	16	57	MONOPLY	"3JONG"	Mah-Jong Scoring 3 Players	Table Games	Table Games	MONO
637	16	16	58	MONOPLY	"4JONG"	Mah-Jong Scoring 4 Players	Table Games	Table Games	MONO
638	16	16	59	MONOPLY	"HERBIE"	Herbie the Hippo			MONO
639	16	16	60	MONOPLY	"C-N"	Chords to Notes			MONO
640	16	16	61	MONOPLY	"N-C"	Notes to Chords			MONO
641	16	16	62	MONOPLY	"NIM"	Game of Nim	Puzzles	Puzzles	MONO
642	10	10	0	4WINS	<b>-FOUR WINS</b>	<b>Section Header</b>	<b>Admin</b>	<b>Admin</b>	<b>MONO</b>
643	10	10	1	4WINS	"4WINS"	Connect Four Game			4WIN
644	10	10	2	4WINS	"XM"	Aux routine			4WIN
645	10	10	3	4WINS	"K"	Aux routine			4WIN
646	10	10	4	4WINS	"L"	Aux routine			4WIN
647	10	10	5	4WINS	"M"	Aux routine			4WIN
648	10	10	6	4WINS	"N"	Aux routine			4WIN
649	10	10	7	4WINS	"O"	Aux routine			4WIN
650	10	10	8	4WINS	<b>-MATRIX GAME</b>	<b>Section Header</b>	<b>Admin</b>	<b>Admin</b>	<b>MONO</b>
651	10	10	9	4WINS	"MATRXG"	Marix Game Main			4WIN
652	10	10	10	4WINS	"EP"	Aux routine			4WIN
653	10	10	11	4WINS	"GO"	Aux routine			4WIN
654	10	10	12	4WINS	"HP"	Aux routine			4WIN
655	10	10	13	4WINS	"SRT1"	Aux routine			4WIN

656	10	14	4WINS	"BOE"	Boerse Spiel - German	Stefan Fegert	Programmsammlung Book	FOCAL	4WIN
657	10	0	GMZONE	-GAME ZONE	Section Header	Ángel Martin	This Module	MCODE	ZONE
658	10	1	GMZONE	"BRAIN"	Color Guessing	Stefan Fegert	hp-41in der Praxis	FOCAL	ZONE
659	10	2	GMZONE	"CHASE"	Cosmic Chase	Stefan Fegert	hp-41in der Praxis	FOCAL	ZONE
660	10	3	GMZONE	"BR"	Aux routine	Stefan Fegert	hp-41in der Praxis	FOCAL	ZONE
661	10	4	GMZONE	"GA"	Aux routine	Stefan Fegert	hp-41in der Praxis	FOCAL	ZONE
662	10	5	GMZONE	"L"	Aux routine	Stefan Fegert	hp-41in der Praxis	FOCAL	ZONE
663	10	6	GMZONE	"P"	Aux routine	Stefan Fegert	hp-41in der Praxis	FOCAL	ZONE
664	10	7	GMZONE	"R"	Aux routine	Stefan Fegert	hp-41in der Praxis	FOCAL	ZONE
665	10	8	GMZONE	"U"	Aux routine	Stefan Fegert	hp-41in der Praxis	FOCAL	ZONE
666	10	9	GMZONE	"W"	Aux routine	Stefan Fegert	hp-41in der Praxis	FOCAL	ZONE
667	10	10	GMZONE	"INVDRS"	Space Invaders	Stefan Fegert	hp-41in der Praxis	FOCAL	ZONE
668	10	11	GMZONE	"MINES"	Mine Field	Stefan Fegert	hp-41in der Praxis	FOCAL	ZONE
669	10	12	GMZONE	"MFLD"	Mine Field Data	Stefan Fegert	hp-41in der Praxis	FOCAL	ZONE
670	10	13	GMZONE	"MONSTER"	Monster Island	Stefan Fegert	hp-41in der Praxis	FOCAL	ZONE
671	10	14	GMZONE	"TENNIS"	Tennis Set	Stefan Fegert	hp-41in der Praxis	FOCAL	ZONE
672	10	15	GMZONE	"TTT"	Tic-Tac-Toe	Stefan Fegert	hp-41in der Praxis	FOCAL	ZONE
673	16	0	MAZES	-AMAZING 41	Section Header	Ángel Martin	This Module	MCODE	MAZZ
674	16	1	MAZES	AIP	Alpha Integer Part	HP Co.		MCODE	MAZZ
675	16	2	MAZES	RNG	Random Number w/ Timer	JM Baillard		MCODE	MAZZ
676	16	3	MAZES	"MAZE1"	Creates Example Maze	Ángel Martin	This Module	FOCAL	MAZZ
677	16	4	MAZES	"XMAZE"	X-Mem Mazes	Erik Christensen	PPCCJ10N5 p31	FOCAL	MAZZ
678	16	5	MAZES	"MAZE"	Builds Maze	JM Baillard	hp41. programs	FOCAL	MAZZ
679	16	6	MAZES	"MAZE+"	Driver for MAZE	Ángel Martin	hp41. programs	FOCAL	MAZZ
680	16	7	MAZES	"MAZES"	Maze Construction & Play	Layne K. Johnson	UPL #00663C	FOCAL	MAZZ
681	16	8	MAZES	-TSP FNS	Section Header	Ángel Martin	This Module	MCODE	MAZZ
682	16	9	MAZES	"TSP2"	2D Travelling Salesman Problem	JM Baillard		FOCAL	MAZZ
683	16	10	MAZES	"TSP2+"	Driver for TSP2	Ángel Martin		FOCAL	MAZZ
684	16	11	MAZES	"TSP3"	3d Travelling Salesman Problem	JM Baillard		FOCAL	MAZZ
685	16	12	MAZES	"TSP3+"	Driver for TSP3	Ángel Martin		FOCAL	MAZZ
686	16	13	MAZES	"TSPS"	Spherical 3D TSP	JM Baillard		FOCAL	MAZZ
687	16	14	MAZES	"TSPS+"	Driver for TSPS	Ángel Martin		FOCAL	MAZZ
688	16	15	MAZES	"*TS"	Auxiliary Subroutine	JM Baillard		FOCAL	MAZZ
689	16	16	MAZES	-PUZZLES	Section Header	Ángel Martin	This Module	MCODE	MAZZ
690	16	17	MAZES	"TCIG"	The Count is Good	Ángel Martin	hp41. programs	FOCAL	MAZZ
691	16	18	MAZES	LCeB	Le Compte est Bon	Martin-Baillard	hp41. programs	MCODE	MAZZ
692	16	19	MAZES	"MAGIC"	Magic Squares	JM Baillard	hp41. programs	FOCAL	MAZZ
693	16	20	MAZES	"PMAGIC"	Pan-magic Squares	JM Baillard	hp41. programs	FOCAL	MAZZ
694	16	21	MAZES	"STEPS"	Step Game	George G. Sandoval	UPL #00363C	FOCAL	MAZZ
695	16	22	MAZES	"STPS"	Data Loadig for STEPS	Ángel Martin	This Module	FOCAL	MAZZ
696	10	0	PRISFUN	-PRISMA FUN	Section Header	Ángel Martin	This Module	MCODE	FUNP



697	10	1	PRISFUN	"BIGBEN"	BigBen Chime	Sounds	JM Baillard	Web Site	FOCAL	FUNP
698	10	2	PRISFUN	"LIFE"	Life Game	Simulation	Eric van der Wateren	PPCJ.V1N3.p10	FOCAL	FUNP
699	10	3	PRISFUN	"RACE"	Car Race	Action	Martin Meyer	PRISMA 3/89.p29	FOCAL	FUNP
700	10	4	PRISFUN	"RUMMY"	Rummy Game	Puzzles	Wolfgang Pawlowsky	PRISMA 84.4.12	FOCAL	FUNP
701	10	5	PRISFUN	"RN"	Random Number	Utils	PPC Members	PPC ROM	FOCAL	FUNP
702	10	6	PRISFUN	"VA"	View ALPHA	Utils	PPC Members	PPC ROM	FOCAL	FUNP
703	10	7	PRISFUN	"3X3"	Matrix Game	Puzzles	S. Zeidler	Prisma B5N3.p35	FOCAL	FUNP
704	10	8	PRISFUN	"1"	Aux routine	Puzzles	S. Zeidler	Prisma B5N3.p36	FOCAL	FUNP
705	10	9	PRISFUN	"2"	Aux routine	Puzzles	S. Zeidler	Prisma B5N3.p37	FOCAL	FUNP
706	10	10	PRISFUN	"3"	Aux routine	Puzzles	S. Zeidler	Prisma B5N3.p38	FOCAL	FUNP
707	10	11	PRISFUN	"4"	Aux routine	Puzzles	S. Zeidler	Prisma B5N3.p39	FOCAL	FUNP
708	10	12	PRISFUN	"5"	Aux routine	Puzzles	S. Zeidler	Prisma B5N3.p40	FOCAL	FUNP
709	10	13	PRISFUN	"6"	Aux routine	Puzzles	S. Zeidler	Prisma B5N3.p41	FOCAL	FUNP
710	10	14	PRISFUN	"7"	Aux routine	Puzzles	S. Zeidler	Prisma B5N3.p42	FOCAL	FUNP
711	10	15	PRISFUN	"8"	Aux routine	Puzzles	S. Zeidler	Prisma B5N3.p43	FOCAL	FUNP
712	10	16	PRISFUN	"9"	Aux routine	Puzzles	S. Zeidler	Prisma B5N3.p44	FOCAL	FUNP
713	10	17	PRISFUN	"67HP"	Eighth Passanger (HP-67)	Action	Web Site	MySilicium Forum	FOCAL	FUNP
714	10	18	PRISFUN	"BIOR"	Biorythms	Funnies	Neil G. Jarman	DataFile V2N3.p5	FOCAL	FUNP
715	10	19	PRISFUN	"CRAPS"	A Dice game (updated)	Casino	Samuel J. Hartmen	PPCJ.V1N3.p27	FOCAL	FUNP
716	10	20	PRISFUN	"FOURS"	Virtually Connected	Puzzles	David Kipling	DataFile V2N4.p5	FOCAL	FUNP
717	10	21	PRISFUN	"HI-LO"	Game for Children	Puzzles	Philip K. Frohme	PPCJ.V12N6p26	FOCAL	FUNP
718	10	22	PRISFUN	"KIBUR"	Letter Sorting	Puzzles	Robert Pulluad	Lórdinateur de Poche	FOCAL	FUNP
719	10	23	PRISFUN	"KIBIN"	Letter Sorting	Puzzles	Robert Pulluad	Lórdinateur de Poche	FOCAL	FUNP
720	10	24	PRISFUN	"KIBIS"	Letter Sorting	Puzzles	Robert Pulluad	Lórdinateur de Poche	FOCAL	FUNP
721	10	25	PRISFUN	"PX"	Pendú (French Hangman)	Puzzles	Unknown	Swap Disks	FOCAL	FUNP
722	10	26	PRISFUN	"SPACE"	Space War - Interactive	Action	Roger M. Stenerson	UPL #00655C	FOCAL	FUNP
723	10	27	PRISFUN	"WG"	A Word Game	Puzzles	Mark Gessner	PPCJ.V12N11.p20	FOCAL	FUNP
724	12	1	FRENCH	-FRENCH ADV	Section Header	Admin	Ángel Martín	This Module	MCODE	FADV
725	12	2	FRENCH	"DON"	Dungeon Game (French)	Adventure	Unknown	Swap Disks	FOCAL	FADV
726	12	3	FRENCH	"DONSOTI"	Dungeon Game (French)	Adventure	Unknown	Swap Disks	FOCAL	FADV
727	12	4	FRENCH	"A"	Advance	Adventure	Unknown	Swap Disks	FOCAL	FADV
728	12	5	FRENCH	"S"	Sud	Adventure	Unknown	Swap Disks	FOCAL	FADV
729	12	6	FRENCH	"O"	Ouest	Adventure	Unknown	Swap Disks	FOCAL	FADV
730	12	7	FRENCH	"N"	Nord	Adventure	Unknown	Swap Disks	FOCAL	FADV
731	12	8	FRENCH	"WG"	Word Game	Adventure	Unknown	Swap Disks	FOCAL	FADV
732	12	9	FRENCH	"FADV"	French Adventure	Puzzles	Mark Gessner	PPCJ.V12N11.p20	FOCAL	FADV
733	12	10	FRENCH	"ALLUME"	Aux routine	Adventure	Unknown	Swap Disks	FOCAL	FADV
734	12	11	FRENCH	"CREUSE"	French Adventure	Adventure	Unknown	Swap Disks	FOCAL	FADV
735	12	12	FRENCH	"DEPOSE"	French Adventure	Adventure	Unknown	Swap Disks	FOCAL	FADV
736	12	13	FRENCH	"EMPLIR"	French Adventure	Adventure	Unknown	Swap Disks	FOCAL	FADV
737	12	14	FRENCH	"LISTER"	French Adventure	Adventure	Unknown	Swap Disks	FOCAL	FADV

738	12	15	FRENCH	"OUVRIR"	French Adventure	Adventure	Unknown	Swap Disks	FOCAL	FADV
739	12	16	FRENCH	"PRENDR"	French Adventure	Adventure	Unknown	Swap Disks	FOCAL	FADV
740	12	17	FRENCH	"RANGER"	French Adventure	Adventure	Unknown	Swap Disks	FOCAL	FADV
741	12	18	FRENCH	"SORTIR"	French Adventure	Adventure	Unknown	Swap Disks	FOCAL	FADV
742	12	19	FRENCH	"VOIR"	French Adventure	Adventure	Unknown	Swap Disks	FOCAL	FADV
743	12	20	FRENCH	"A"	French Adventure	Adventure	Unknown	Swap Disks	FOCAL	FADV
744	12	21	FRENCH	"AE"	French Adventure	Adventure	Unknown	Swap Disks	FOCAL	FADV
745	12	22	FRENCH	"AS"	French Adventure	Adventure	Unknown	Swap Disks	FOCAL	FADV
746	12	23	FRENCH	"AO"	French Adventure	Adventure	Unknown	Swap Disks	FOCAL	FADV
747	12	24	FRENCH	"AH"	French Adventure	Adventure	Unknown	Swap Disks	FOCAL	FADV
748	12	25	FRENCH	"AB"	French Adventure	Adventure	Unknown	Swap Disks	FOCAL	FADV
749	16	0	<b>SEATBTL</b>	<b>-ARMADA INV</b>	<b>Section Header</b>	<b>Admin</b>	<b>Ángel Martín</b>	<b>This Module</b>	<b>MCODE</b>	<b>BSEA</b>
750	16	1	SEATBTL	"BATNAV"	Batalla Naval	Action	<i>F.J. Chamorro Pagani</i>	Electro-1 V2N2 p106	FOCAL	BSEA
751	16	2	SEATBTL	"BSHIP"	True Battleship	Strategy	<i>Luis Gasperini</i>	UPL #00796C	FOCAL	BSEA
752	16	3	SEATBTL	7P<->S	Register Swap	Utils	<i>HP Co.</i>	Card Reader ROM	MCODE	BSEA
753	16	4	SEATBTL	E3/E+	builds pointer	Utils	<i>Ángel Martín</i>	SandMath	MCODE	BSEA
754	16	5	SEATBTL	LEFT	Moves LCD left	utils	<i>Nelson F. Crowle</i>	NFC ROM	MCODE	BSEA
755	16	6	SEATBTL	GOOSE	Puts left-goose char in LCD	Utils	<i>Nelson F. Crowle</i>	NFC ROM	MCODE	BSEA
756	16	7	SEATBTL	RNG	Random Number w/ Timer	Utils	<i>JM Baillard</i>	hp41. programs	MCODE	BSEA
757	16	8	SEATBTL	TFX	Toggle Flag	Utils	<i>Ken Emery</i>	MCODE for Beginners	MCODE	BSEA
758	16	9	<b>SEATBTL</b>	<b>-SEA BATTLE</b>	<b>Section Header</b>	<b>Admin</b>	<b>Ángel Martín</b>	<b>This Module</b>	<b>MCODE</b>	<b>BSEA</b>
759	16	10	SEATBTL	"SBP"	Sea Battle w/ Printer	Action	<i>Mark Cracknell</i>	Datafile V8N6 p9	FOCAL	BSEA
760	16	11	SEATBTL	"A"	routine for SBP	Action	<i>Mark Cracknell</i>	Datafile V8N6 p9	FOCAL	BSEA
761	16	12	SEATBTL	"B"	routine for SBP	Action	<i>Mark Cracknell</i>	Datafile V8N6 p9	FOCAL	BSEA
762	16	13	SEATBTL	"C"	routine for SBP	Action	<i>Mark Cracknell</i>	Datafile V8N6 p9	FOCAL	BSEA
763	16	14	SEATBTL	"D"	routine for SBP	Action	<i>Mark Cracknell</i>	Datafile V8N6 p9	FOCAL	BSEA
764	16	15	SEATBTL	"E"	routine for SBP	Action	<i>Mark Cracknell</i>	Datafile V8N6 p9	FOCAL	BSEA
765	16	16	SEATBTL	CLAXON	sounds claxon	Sounds	<i>Mark Power</i>	Data File	MCODE	BSEA
766	16	17	SEATBTL	RASP	sounds rasping sound	Sounds	<i>Mark Power</i>	Data File	MCODE	BSEA
767	16	18	<b>SEATBTL</b>	<b>-NAVY WAR</b>	<b>Section Header</b>	<b>Admin</b>	<b>Ángel Martín</b>	<b>This Module</b>	<b>MCODE</b>	<b>BSEA</b>
768	16	19	SEATBTL	"CNAV"	Create Data File	Utils	<i>Ángel Martín</i>	This Module	FOCAL	BSEA
769	16	20	SEATBTL	"NAVY"	Main Program - German	Action	<i>Burkhard Oerttel</i>	Programmsammlung Book	FOCAL	BSEA
770	16	21	SEATBTL	"NAV2"	Main Program - Part 2	Utils	<i>Burkhard Oerttel</i>	Programmsammlung Book	FOCAL	BSEA
771	16	22	SEATBTL	"J"	routine for NAVY	Utils	<i>Burkhard Oerttel</i>	Programmsammlung Book	FOCAL	BSEA
772	16	23	SEATBTL	"K"	routine for NAVY	Utils	<i>Burkhard Oerttel</i>	Programmsammlung Book	FOCAL	BSEA
773	16	24	SEATBTL	"M"	routine for NAVY	Utils	<i>Burkhard Oerttel</i>	Programmsammlung Book	FOCAL	BSEA
774	16	25	SEATBTL	"N"	routine for NAVY	Utils	<i>Burkhard Oerttel</i>	Programmsammlung Book	FOCAL	BSEA
775	16	26	SEATBTL	"P"	routine for NAVY	Utils	<i>Burkhard Oerttel</i>	Programmsammlung Book	FOCAL	BSEA
776	16	27	SEATBTL	"Y"	routine for NAVY	Utils	<i>Burkhard Oerttel</i>	Programmsammlung Book	FOCAL	BSEA
777	16	28	SEATBTL	"W"	routine for NAVY	Utils	<i>Burkhard Oerttel</i>	Programmsammlung Book	FOCAL	BSEA
778	16	29	SEATBTL	"#"	routine for NAVY	Utils	<i>Burkhard Oerttel</i>	Programmsammlung Book	FOCAL	BSEA

779	16	SEATBTL	"?"	routine for NAVY	Utils	Burkhard Oerttel	Programmsammlung Book	FOCAL	BSEA
780	16	<b>SEATBTL</b>	<b>-SUBMARINES</b>	<b>Section Header</b>	<b>Admin</b>	<b>Angel Martin</b>	<b>This Module</b>	<b>MCODE</b>	BSEA
781	16	SEATBTL	"SH"	Sub Hunt Program	Action	Charles Campbell	PPC V7N4 p15	FOCAL	BSEA
782	16	SEATBTL	"SUB"	Submarine Hunt	Action	Brian Steel	DataFile V2N3 p7	FOCAL	BSEA
783	16	SEATBTL	"SUBHT"	Submarine Hunt	Action	James R. Merrill	PPCCJ\V8N2p18	FOCAL	BSEA
784	16	SEATBTL	"SUBHUNT"	Submarine Hunt	Action	HP Co.	Games Pac	FOCAL	BSEA
785	16	SEATBTL	"BOOM"	Aux routine	Utils	HP Co.	Games Pac	FOCAL	BSEA
786	16	SEATBTL	"RNDM"	Aux routine	Utils	HP Co.	Games Pac	FOCAL	BSEA
787	16	SEATBTL	"SEED"	Aux routine	Utils	HP Co.	Games Pac	FOCAL	BSEA
788	16	SEATBTL	"7BPS"	hp-67 Sea Battle	Action	???	hp-67 Games	FOCAL	BSEA