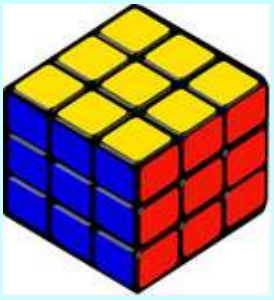
HP-41 Rubik's Cube Solver Module



Overview

This module includes the programs written by Julian Perry and published in PPCCJ to solve the Rubik's Cube. The ROM format allows for a simpler program execution, not restricted by the main memory space limitations. New routines for data entry are included to define the cube's initial configuration, and Extended Memory support is also implemented to store different cubes as ASCII files.

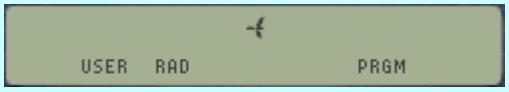
A new MCODE function has been added for easier and more reliable U/I : the row colors entered as a three-digit prompt, which only allowed values correspond to the individual color codes, i.e. "R,G,B,Y,W,O"

The table below shows the function names in pseudo-alphabetical order with a brief description. The Authors and sources are listed below for a complete program description and user instructions in case you're interested. You're encouraged to read the original articles by Julian Perry for a better appreciation of the difficulty of the task, and the ingenious solution he came up with.

XROM	Function	Description	Author	Source
8,00	-RUBIKS CB	*Section header*	*Ángel Martin*	*This project*
8,01	"CUBE1"	*Example1*	*Ángel Martin*	*This Project – see below*
8,02	"CUBE2"	*Example2*	*Ángel Martin*	*This Project – see below*
8,03	"RUBIK"	*Main Data Entry*	*Martin- Perry*	*PPCCJ V9N1P30*
8,04	"R1"	*Stage -1*	*Julian Perry*	*PPCCJ V9N2P23*
8,05	"R2"	*Stage-2*	*Julian Perry*	*PPCCJ V9N2P23*
8,06	"S"	*Display/Print*	*HJ Gessler*	*PPCCJ V10N4P3*
8,07	"T"	*Calculate Next Move*	*Julian Perry*	*PPCCJ V9N2P23*
8,08	"X"	*Initialize R01-R04 X Axis*	*Julian Perry*	*PPCCJ V9N2P23*
8,09	"Y"	*Initialize R01-R04 Y Axis*	*Julian Perry*	*PPCCJ V9N2P23*
8,10	"Z"	*Initialize R01-R04 Z Axis*	*Julian Perry*	*PPCCJ V9N2P23*
8,11	^CROW	*Enter Color ROW*	*Ángel Martin*	*This project – see below*
8,12	AINT	*ARCL Integer*	*Frits Ferwerda*	*MLROM project*
8,13	CLAXON	*Claxon Sound*	*Mark Power*	*DataFile V7N7 p12*
8,14	E3/3+	*Builds pointer*	*Ángel Martin*	*SandMath project*
8,15	GOOSE	*Shows left goose*	*Nelson C. Crowle*	*NFCROM Project*
8,16	LEFT	*Moves LCD to the left*	*Nelson C. Crowle*	*NFCROM Project*
8,17	RASP	*Rasping sound*	*Mark Power*	*DataFile V7N7 p12*
8,18	RNG	*Random number*	*JM Baillard*	*A few M-CODE Routines*
8,19	"VREG"	*View Registers*	*Ángel Martin*	*RAMPAGE Project*

As you can see the modification proposed by HJ Gessler has been added to the original program as well. This is especially well suited for the execution on V41 and other emulators lacking printer support – whilst taking full advantage of their turbo speeds. Thanks go also to Jackie Woldering for his program transcription and barcodes provided in the PPC Barcode project.

Custom sounds (courtesy of Mark Power) replace the traditional BEEP and TONEs for a more diverse experience. Also as a fun note, during the main program execution the left goose is shown moving leftwards – so there's no doubt a left-handed person is working on the cube ;-) We have Nelson F. Crowle to thank for this light and joyous touch.

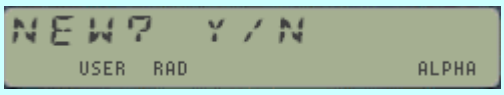


Thanks also to Warren Furlow for hosting the referenced material and the actual ROM images on his excellent web site, www.hp41.org.

Below is a brief description for the most important functions of the module for your convenience –

"RUBIK": Main program w/ Data Entry

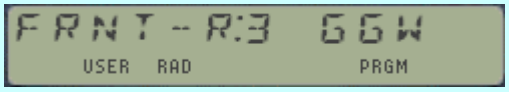
This is the main data entry and cube solving program. The cube can be configured from the scratch (that's part of the fun!), or you can provide an ASCII file name to use a cube configuration saved previously – like the examples provided in the module. If a new cube configuration is entered it will be saved in X-Memory as well, with a File name of your choice.



Answer Y/N to determine the case.

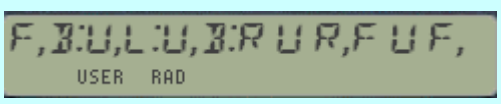
For a new Cube, first the File Name is to be entered and then there will be three prompts for each of the cube faces, FRONT, LEFT, BACK, RIGHT, LOW, and UP. Each one of the 18 prompts will input the three colors for the row until the full configuration is entered.

For an existing cube stored in an X-Mem file, the program will show each row in the display as it's being read from the file.



Shows the colors "Green-Green-White" for the 3rd. row in the "FRONT" Face

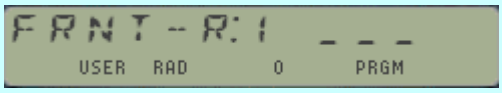
After the calculation of each move, the solution is presented in the display following the Singmaster's convention (check original articles provided above) in "chunks" of 12 rotations; like it's shown below - with the exception being the transition from "stage1"to stage2". This was needed for the RAM-based version of the programs and even if not needed in the ROM module has been left in here only to maintain the original arrangements.



Once you have performed the group of 12 rotations you need to press R/S to continue solving the cube until completion.

^CROW _ _ _ : Enter Color Row

This function provides the main U/I aid to enter the three colors within a given row. The function presents a three-digit prompt, and each digit must conform to the allowed characters that represent the possible colors: “R,G,B,Y,W,O”. No other keys will be accepted.



Pressing Back-Arrow cancels the entry and stops the program execution. At that point you can resume with R/S and the prompt will be presented again.

Note that even if the colors within a row cannot be mistyped, the data entry will not check for valid combinations taking into account the previous rows. This however will be checked later on during the program execution, and an error message will be shown if the configuration is bad.

“CUBE1” and “CUBE2” Examples

These two programs will create and populate ASCII files in X-Memory with two examples to test the program. You’ll need to execute the corresponding “CUBE” program first to load the data; and then the main “RUBIK” program as before. The ASCII files will remain in extended memory after execution and until they are purged by the user.

The solutions to these 2 examples are provided below:

Example1: Solution to the CUBE1 case in 111 MOVES

Move	Turns	Move	Turns
01	F:RDFD,D:FD:U:R:RU:	06	B,U,R,U,RF,LFL,U,L,U
02	R,FU:F,U:LUL,BU:B,U,	07	LUDFD,F,DFD,F,U,D
03	BUB, - stage2	08	FD,F,DFD,F,U,DFD,F,
04	F,U,FURUR,L,U,LUF	09	DFD,F,U:F,U:FRBLU:
05	UF,UB,U,BULUL,BU	10	L,B,R,F:ULR,F:RL,UF:

Example 2: Solution to the CUBE2 case in 99 MOVES

Move	Turns	Move	Turns
01	F,B:U,L:U,B:RUR,FUF,	06	UF,LFL,U,L,ULUDF
02	B,UB – stage2	07	D,F,DFD,F,UFDF,D,F
03	LU,LF:UFU,F:L,U,L,B	08	DF,D,U,L,B,R,URBLF
04	L:ULU,L:B,LU:L,ULU:	09	U,F,L,R:U,FB,R:BF,U,R:
05	L,UB,U,BL,B:U,B,UB:L	-	