INFO48 program description

The HP48

The HP48 uses a proprietary language called **RPL** for **reverse polish lisp.** This is a FORTH like language in that it is stack based, and every "word" gets translated into a token to be run by an inner interpreter. Several extensions have been made including the following:

1. Stack is a tagged data structure. Possible data types include flags, binary integers, real numbers, complex numbers, arrays of real or complex numbers, lists of any data type, and secondaries (program objects).

2. A system of local variables has been implemented. Local variables are defined by taking a group of items on the stack, and executing the word BIND to associate these stack items with a list of LAMs (local variable names).

3. secondaries (programs objects) can be manipulated just as easily as lists.

The HP48 is programable at three levels. **User RPL** is the language directly enterable/editable/debugable via the calculator keyboard. **User RPL** is actually a subset of **system rpl**. Though **system rpl** is the language used for the HP48's system programs and is run by the same inner interpreter as **user rpl**, it is not normally possible to create programs written in **system rpl** on the calculator itself. HP company has supplied several tools for programming **system rpl** offline on a PC, including a compiler, an assembler and a linking loader to convert entry points into binary tokens. Third party products, filling in the gaps left by HP, include **system rpl** debugger, and more thoughough documentation of supported and unsupported entry points.

INFO48 program organization

INFO48 is divided into two modules. **Select_obj** is the basic browser engine which allows the user to move around the data structure, and **info** is the main program.

Select_obj is based on the HP48's parameterized outer loop, which defines a standard means of implementing a user interface. Parameters passed to it define a display update secondary, keyboard definitions, menu definitions, and various miscellaneous flags such as wether or not user and default keyboard definitions are to be executed for undefined keys.

One parameter for the parameterized outer loop is the list of menu key definitions. For each menu key, there is a list containing the key's label, and secondaries for the menu key's unshifted, left-shifted, and right-shifted actions. Since these secondaries are programs in their own right, they can refer to **select_obj**'s local variables and even call upon **select_obj** recursively when it is necessary for the user to specify a second object for the menu key command to act upon. When **select_obj** is

called recursively, it is called with menu keys simply defining an exit condition, and a flag preventing the user from doing anything but selecting an object.

To minimize memory used by local variable identifiers, an include file "INFODEFN.S" is used to translate the long meaningfull names into single or double character identifiers. For example "cursor_pos" gets translated to "c"; "cursor_line" gets translated to "cl". When debugging the program, it is these single and double character variables names that get displayed.

INFO48 eventually is translated into an HP48 library object. The <-LIB-> translator, resident on the HP48, looks for the following objects which define its operation:

1. $ROMID — a unique number used to diferentiate library objects. The programer must be sure not to use an id already in use by another library. Several FTP sites maintain lists of previously used ROMID's. INFO48 uses 1702, a currently unused library ROMID.
2. $VISIBLE — a list of routines visible to the user. **All** other routines are translated into **romptr**'s which are a pair of numbers, as opposed to a named identifier. **INFO** is the only visible identifier in INFO48.
3. $TITLE — The name under which the library is displayed in the library menu. ("INFO48")
4. $CONFIG — A secondary to be executed when the calculator is warm-booted. (occurs whenever a change in memory configuration is detected, or the calculator is attempting to recover from a crash.)

All identifiers for program objects are downloaded into the HP48 prior to library creation. Since we are dealing with an MSDOS file for each program object; "INFODEFN.S" converts each global program object identifier into a name suitable for an MSDOS filename. For example, **select_obj** gets translated to **SELOBJ**; **reorder_vars** gets translated to **reordvar**. Each of these MSDOS files is stored with a **.s** extension, and a batch file was created to compile them and put them into a directory of executables. A kermit batch download was then used to transfer all these program objects to the HP48, which was put into kermit server mode.

Brief description of INFO48 routines:

select_obj module:

	routine name	DOS filename	description
1.	check_display	CHKDISP	check display; perform scroll or complete updated as required
2.	disp_def_msg	DISPDEF	display default message; clear previous message from top of display

INFO48 program description, page 3

3.	disp_msg	DISPMSG	display message; convert string to graphics object, display at top
4.	draw_obj	DRAWOBJ	draw object; a single line on display, two columns
5.	disp_search	DISPSEA	display search; display current search string at top of display
6.	enter_cmd	ENTER	enter command; command executed when user hit enter key: either go to subdirectory/link, or call up standard object editor
7.	first_dir	FIRSTDIR	first directory; go to directory that INFO48 was started from
8.	home_dir	HOMEDIR	home directory; go to root directory
9.	id_descr	IDDESC	id description; given id, produce string used for first column
10.	init_ vars	INIVARS	init vars; read variables in current directory, store them and count
11.	pop_dir	POPDIR	pop_dir; return to prior directory
12.	print_screen	PRINTSCR	print screen; how do you think calculator pictures in documentation were created?
13.	save_path	SAVEPATH	save path; save current directory, and cursor position in path_history
14.	search_cmd	SEARCH	search command; alphanumeric search
15.	select_obj	SELECT	select_obj; main module program
16.	select_keys	SELKEYS	select keys; keyboard definitions
17.	up_dir	UPDR	up directory; go to parrent of current directory

info module:

	routine name	DOS filename	description
1.	copy_obj	COPYOBJ	copy object; select destination/source, copy to/from
2.	create_link	CRLINK	create link; select object, create pointer to that object

INFO48 program description, page 4

3.	edit_link	EDLINK	edit link; call select to allow user to edit where link points
4.	get_obj	GETOBJ	get object; select another object move above/below current
5.	get_uniq_id	GETUNIQ	get unique id; prompt to name, make sure it is valid, and does not already exist
6.	info	INFO	INFO48 main program
7.	move_obj	MOVOBJ	move object from one place to another; params for source/destination directory/location
8.	put_obj	PUTOBJ	put object in new location
9.	rename_obj	RENOBJ	just like it sounds
10.	reorder_vars	REORDVAR	reorder variables; call system routine and let user know info48 hasn't crashed
11.	sort_vars	SORTVAR	use Joe Horn's quicksort; sort is 10 times faster then reordering done immediately afterwards

INFO48 program description, page 5