INFO48, project description

The HP48SX is a scientific expandable calculator manufactured by Hewlett-Packard company. It is based on HP's Saturn processor which is a four bit microprocessor with twenty bit integer registers and sixty-four bit floating point registers, and is optimized for low current consumption. The HP48 includes HP's Reverse Polish Lisp (RPL) operating system, a FORTH like system with extensions for handling symbolic math. The data stack is tagged to allow complex data types to be stored including integers, reals, arrays, lists, and even programs. The HP48SX has two plugin slots which allow the system to be extended with additional ROM software, or user programmable RAM up to a possible total of 256K of additional RAM/ROM. Communications facilities are also included, based on RS-232 voltage levels and the KERMIT protocol.

These communications facilities have fostered a large body of software to be made available for the HP48. Unfortunately, the available information organizers have in my opinion been deficient in several areas. They are usually based on a flat file system, with a large array of records, each containing a fixed number of fields. The purpose of this project has been to create a new type of information organizer, based on a more natural organization modeled after the tree structured directory systems of modern operating systems and including links to allow multiple pointers from different areas to point to a single block of information.

During the design phase of this project, several data structures were considered. Most of the existing information organizers are based on the RPL list data type which allows lists of any other data type, including nested lists. However, information organized in this way is very difficult to work on, except while using the application that created the data structure. I decided to use the HP48's built-in directory structure. This can be manipulated outside the application and other project oriented data to be intermixed. It also handles links to other places easily, simply by storing a path and identifier name.

Within any project running on relatively slow hardware, there are design tradeoffs that must be considered. A decision was made early on to make memory conservation the lowest priority. The highest priority was assigned to ease of use and compatibility with the existing directory scheme. Speed was given the next highest priority, with optimizations for speed frequently producing code spaces four times larger than if memory conservation had the highest priority. A high priority was also given to idiot-proof (user-proof?) error handling. Just about every conceivable type of user error has been accounted for.

The HP48 can be programmed at three levels. The usual way of programming is called user-RPL and is programmable directly on the calculator. An intermediate level between user-RPL and machine language programming is system-RPL. User-RPL is actually a subset of system-RPL. They are both executed by the same interpreter and can call programs written in the other. The main difference is that user-RPL commands include parameter type and range checking and operate on a limited set of data types. System-RPL can be developed off-line on a PC, gaining the software development features that a PC affords, such as a better keyboard and screen, faster processor speed, and larger secondary storage.

INFO48, project description, page 2

The speed optimization of system-RPL is realized by the elimination of parameter type and range checking and makes programming in it much more difficult. A seemingly trivial bug can easily cause unrecoverable corruption to the system's data structures. System crashes are a frequent result with all memory being lost. However, it is possible to store libraries of application software within a plug in memory card, and set it up as not being merged in to the main body of system RAM. This allows software stored in the plug-in card to be protected from system crashes by setting the card to read-only operation.

Without third party software tools, this project would have been impossible to complete. System-RPL is almost impossible to debug using the tools made available by Hewlett-Packard company. Special thanks go to Detlef Mueller & Raymond Hellstern for their wonderful set of system-RPL tools which include a calculator resident compiler and decompiler, a system-RPL debugger and a calculator resident library maker. Special thanks also go to Mika Heiskanen for his very complete documentation of system-RPL entry points.

The basic design of INFO48 revolved around a browser engine which gives the user a display of variables in the current directory. Entering a subdirectory is as simply as pointing the cursor at the subdirectory's identifier, and hitting enter. Editing an object is done similarly. Menu key definitions frequently have to recursively call the browser engine, and this required that it be passed several control parameters. The main program module simply calls the browser engine with startup values for the parameters, and a list of menu key definitions.

The browser engine gives the user a two column display; on the left side are names of objects, and on the right side are the values of the objects. Since an object may be a subdirectory, some way was necessary to allow some data be displayed for that object, such as a phone number. Simply displaying the first object of the directory was not effective since it often produced undesirable results, such as showing the first person in a group of people, where that first person was not a summary for the group. I ended up using a sentinel -- that is, it the first identifier of a subdirectory ends with a period, then it is used for the display on the right hand column.

INFO48 was designed for general use by the HP48 using population. It will be posted to Usenet, and uploaded to an FTP site. Packaged in it's .ZIP file will be the library, the user's manual in postscript form, a simple readme file in standard ascii form, and the informator's source code so that others may learn from the techniques I used, and perhaps inform me of better techniques that I'm not aware of yet.