Kitt Peak Multi-Tasking FORTH-11

Thomas E. McGuire

Health Systems International
100 Broadway
New Haven, CT 06517

Abstract

An overview of the Kitt Peak National Observatory (KPNO) real-time environment is presented. The eleven optical telescopes at Kitt Peak employ a variety of experimental instruments, including Fourier Transform Spectrometers and large array digital cameras, that impose a diverse set of real-time requirements. Presented next will be a historical perspective of FORTH and KPNO over the last decade outlining the evolution of the embryonic version of FORTH on an 8K mainframe (CDC), mini (Digital VAX-11 and PDP-11), and micro (Digital LSI-11) machines. The real-time attributes of FORTH in general and KPNO multi-tasking FORTH-11 in particular are discussed. Some details of the implementation of FORTH-11 are presented to reveal the power of this real-time programming environment. Finally, the KPNO CCD camera system will be used to illustrate a complex real-time application of multi-tasking FORTH-11.

Overview

KPNO is a part of the National Optical Astronomical Observatories (NOAO). NOAO also includes Sacramento Peak Solar Observatory in New Mexico and Cerro Tololo Inter-american Observatory in Chile (CTIO). KPNO is located on Kitt Peak, a 7000 ft. mountain in southwest Arizona, about 55 miles from Tucson, Arizona. The first major telescope was dedicated in 1959 with the completion of the McMath solar telescope, still the largest solar telescope in the world. Today the mountain supports four observatories utilizing 18 large telescopes including a 36 ft. radio telescope, the Mayall 4-meter telescope, and five other telescopes having primary mirrors greater than four feet in diameter. KPNO provides an advanced research environment where astronomers from all over the world utilize the research techniques of photometry, spectroscopy, and imagery to define the frontiers of observational astronomy.

Computers have always been an integral part of the observatory. Today over 20 mini-computers and a dozen micro-computers are used for telescope and instrument control and data acquisition. Almost every telescope system contains a computer for controlling its environment: the dome, the telescope, and instrumentation. These computers are often dedicated to the complex problems of locating and tracking the optical source. For example, at the Mayall telescope coudè focus the image plane is the result of five reflections. The telescope control program must transform the given celestial coordinates of a source into position on the sky while accounting for errors due to refraction and telescope flexure. The program must then position the telescope and dome such that the image falls on a fixed position two floors below in the coudè room. After the image is acquired, the position must be maintained and controlled for the analysis instrumentation. Complex motor and encoder assemblies impose high data rates and constitute a servo system that is closed by the control computer. The telescope control program at the Mayall telescope is able to position the telescope to within several arc seconds anywhere on the sky.

A variety of instruments ranging from room sized grating and Fourier spectrographs to tiny Helium cooled photometric detectors impose a diverse range of speed and space requirements on the data acquisition and control computers. A single observation may consist of a million point Fourier spectrogram or a million pixel digital image acquired in a matter of minutes. These applications use interrupt routines on the order of a hundred micro-seconds in duration to record a single data point or initiate a DMA transfer of ten kilobytes. The most severe requirements for the near future are imposed by multi-anode array detectors with their ability to resolve single photon events on a million point grid with time resolution of less than a micro-second.

The Evolution of KPNO FORTH

The FORTH language was developed in the early seventies at the National Radio Astronomical Observatory (NRAO) and KPNO. The initial development resulted in a standalone FORTH operating system and programming language on an 8K byte Varian minicomputer in 1970.

FORTH language use and development at KPNO began during 1972 when the standalone FORTH environment was used for instrument development [1]. The first major application of FORTH on Kitt Peak was a photometry program used as part of a project that generated a high resolution spectrographic atlas of the Sun in 1975. This application used a simple and elegant foreground/background scheme that allowed dynamic operator control of a continuous data acquisition process. Since that time almost all the telescope control and data acquisition computers on Kitt Peak have been programmed in FORTH. In 1977 the language development was paced by the development of the Interactive Picture Processing System (IPPS). The IPPS lab utilized a minicomputer to interface image processing devices to a Control Data Corporation 6400 mainframe. The control system on the CDC machine for the lab applications was written in FORTH running under the SCOPE operating system. FORTH in that instance was used primarily as a command language that allowed linking FORTRAN routines and FORTH routines to create a powerful and flexible image processing system.

During the late seventies the widespread use of FORTH at KPNO and the advancement of computer hardware led to the next phase of FORTH development at KPNO. Beginning in 1979 a synthesis based on several versions of FORTH from both the observatory and external sources yielded the current KPNO FORTH model. That model has been used to build implementations running under RSX-11M, RT-11, VAX/VMS, Version 7 UNIX, VAX UNIX, and standalone real-time FORTH-11. The use and development of FORTH at KPNO continues; all the real-time applications are performed using FORTH and a significant amount of data reduction and analysis as well.

Attributes of FORTH for a Real-Time Environment

FORTH provides an interactive system that allows quick resolution of the often uncertain details of a complex hardware and software environment. Typically FORTH functions are constructed in a hierarchical and modular fashion that allows the functions to be executed easily in a discrete and singular manner. This separability of functions aids in the isolation of software or hardware difficulties. In addition, the minimal transformation of source code into compiled code and the accessibility of the total environment source code simplifies the task of locating problems in a complex application.

The complex real-time environment is composed of numerous devices that are often unreliable and may interact in a complicated fashion. The interactive, hierarchical, and modular attributes of FORTH assist greatly in the development and maintenance of a complex software/hardware system. Most FORTH systems provide a macro assembler for directly executable code definitions. The assembler routines may also be required for interfacing hardware. Embedding a coded routine in the critical path of a complex program is a method that allows a high level solution without sacrificing efficiency.

Interpreted FORTH code that is constructed in the typical hierarchical fashion provides maximum functionality while using very little memory. The redundancy of code is minimized automatically. The virtual I/O mechanisms for terminal and disk also aid in rapid prototyping.

Specific attributes of multi-tasking FORTH-11 for a real-time environment (discussed below) include an event driven scheduling algorithm, prioritized tasks, shared memory process I/O, and a hierarchical file system utilizing contiguous disk files. In addition, the task context switch is very fast and the uninterruptible code sequences are few in number and of minimal length.

Model for Multi-Tasking FORTH-11

The real-time model for FORTH-11 was constrained primarily by the nature of the PDP-11 architecture. The PDP-11 architecture provides sixteen bits of direct memory addressability and options of memory management allowing twenty-two bits of physical memory space and separate instruction and data spaces.

The multi-tasking model consists of two or more tasks where each task is an identical interactive virtual FORTH computer. Specifically, each task emulates a 16-bit PDP-11 computer that is limited to 64K bytes and appears to be operating standalone FORTH. A virtual FORTH computer may be provided with direct I/O capability and thus interface a peripheral device. Each task executes routines that are part of the FORTH primitive dictionary (the interpreter, for example) and routines that are defined specifically for that task (the application code). The primitive dictionary routines are contained within a section of memory that is shared by all tasks. The shared memory segment is re-entrant and read-only. The data structures required for the common dictionary are contained in the tasks's specific memory and are executable only by the particular task.

Each task performs a unique and independent function; for example, interfacing an I/O device, providing a user interface, or performing numeric calculations. Tasks are programmed individually and, after the communication interfaces and protocols are satisfied, linked together into the multi-tasking system. The independence of tasks allows modular programming at a particularly high level. Communication between the tasks is through well defined protocols and the multiple virtual computers can be regarded as a virtual network (Figure 1). The communication protocol and the independence of tasks also allows a particular task to be removed from the virtual network so as to be installed in a separate processor in the physical network.

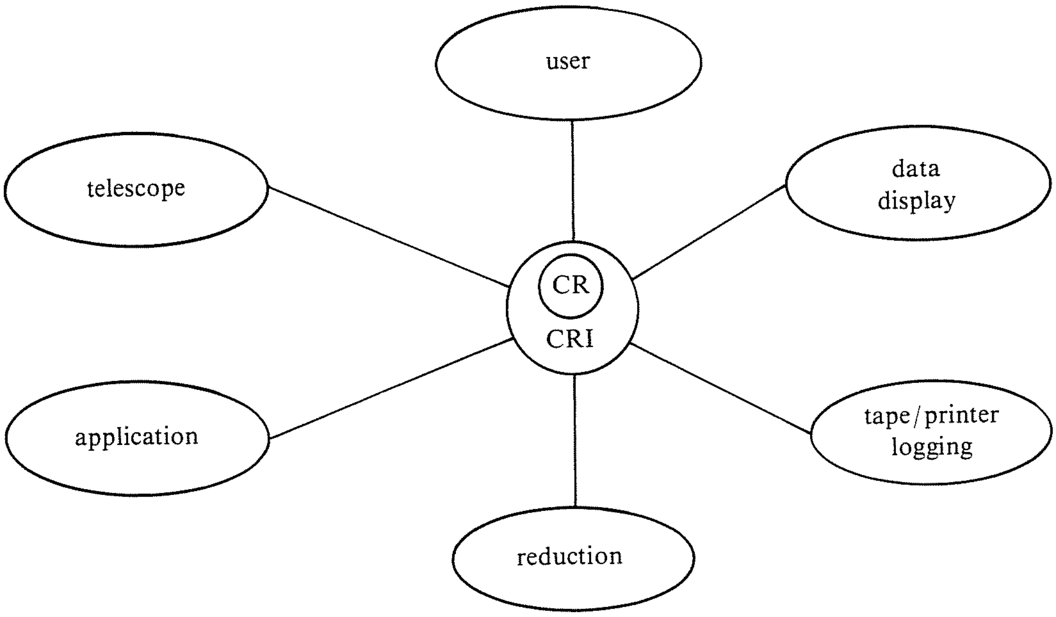

Figure 1.

Within the virtual network all access to other tasks or common resources (CR) are regulated by the Common Resource Interface (CRI).* The CRI consists of assembled service routines for software interrupts and process scheduling. The software interrupts are generated by processor traps and are invoked with *trap* instructions. The allocation of the CPU to a virtual computer (scheduling) is checked whenever an activation event occurs. When an *event* occurs that activates a task that is higher priority than the currently scheduled task, rescheduling will occur. Activation events are primarily related to device interrupts but can also correspond to software synchronization, as in intertask communication. In this model the common resources are the CPU, disk, terminal, and non re-entrant common routines.

The CPU is always allocated to the highest priority task that is "ready" (activated).

Every FORTH-11 task is logically identical and consists of three or four segments that begin at constant boundaries and, with the exception of the Instruction/Data (I/D) segment, are of fixed sizes (Figure 3 **). The segments consist of the *specific D, common I, I/D, and I/O.*

Memory allocation for a task is normally performed at the time the task is created and occurs in quanta of 8K byte *pages*. A *page* is an artifact of the PDP-11 memory management scheme and represents a memory segment of variable size (32–8192 bytes) with certain attributes. Eight pages represent the maximum available memory for the first three segments (when the I/O page is not needed).

The first two segments, the common instruction (common I) segment and the task specific data segment (specific D), are required for all tasks and literally define a virtual FORTH computer. They contain the *read-only* primitive FORTH dictionary (common I) and associated data structures (specific D) required for that dictionary.*** Virtual pages 0–2 are always allocated for these segments (Figure 4).

The I/D segment contains the task-specific code and data, namely the code that uniquely defines a task. The space available ranges from 8K bytes minimum to 40K bytes maximum. When the memory management option of separate instruction and data spaces is utilized, an additional data only segment of up to 64K bytes may be used.

The Input/Output (I/O) segment is required only for tasks that perform direct device input or output. If this is not needed the indicated space is available for the I/D segment.

Page 0

The elements in page 0 that are special to FORTH-11 are the spooling buffer, the system overlays, and the specific data structures.

Task logical structure	
segment	function
I	task specific data area
II	common dictionary area
III	task specific memory area
IV	memory mapped I/O area

Figure 2.

Task page structure**		
segment	page#	function
I	0	specific D
II	1	common I
II	2	common I
III	3	I/D
III	4	[I/D]
III	5	[I/D]
III	6	[I/D]
IV	7	[I/O or I/D]

Figure 3.

The spooling buffer is typically used as a disk I/O buffer for the virtual terminal. For example, the standard output of a task can be redirected to a file for communiction to another task or spooled to a printer driver. The interrupt and trap vectors occupy the beginning of page 0 in the kernel task only.

The block buffers usually number two but they are simply linked and the number can be extended dynamically to include as many as space allows.

*When writable memory is shared between multiple tasks, a locking mechanism outside the CRI must be provided.

**Functions within brackets are optional.

***References [2-4] are the formal definition of KPNO FORTH and FORTH-11, including glossaries. Reference [5] provides an introduction to KPNO FORTH.

The system overlay area is a memory segment that is multiplexed for precompiled routines. A significant number of tools and utilities are available in this area which forms a virtual extension of the *common I* segment. For example, a complete assembler for the PDP-11 instruction set is one overlay. Currently another 8K bytes of common routines are multiplexed into this 2K byte overlay.

The task specific data structures are the variable storage area for *common I* and system overlay algorithms. All the shared code is read-only and re-entrant.

Page 0	Page 1–2
return stack	assembler
parameter stack	utilities
spooling buffer	file system
system overlay area	overlay driver
FORTH block buffers	vocabulary routines
task specific data structures	compiler conditionals
[interrupt and trap vectors]	formatted I/O routines
	task communication interface
	high level structures
	error routines
	string operators
	integer, long integer, mixed precision
	disk routines
	stack operators
	compiler primitives and macros
	interpreter

Figure 4.

Pages 1-2

FORTH-11 contains numerous extensions in the common dictionary (pages 1–2) and system overlay area. Some features, such as redirectable input/output (virtual terminal, redirectable error routine, and switchable stack and redefinition checking, are fundamental additions that are commensurate with the current state of system design.

The important FORTH-11 extensions are the following:

Overlay driver.
File system routines.
Redirectable standard input/output.
Redirectable error routine.
Dynamically switchable stack checking.
Dynamically switchable redefinition checking.
Binary loader.
Intertask communication routines.
Automatic interrupt routine (“ready” event).
Unexpected interrupt routine (all unassigned vectors initialized).

Pages 3-7

The remaining available pages in a task are used for the task application code and data, unless the task requires direct device I/O. A task that performs direct I/O must have page 7 mapped into the physical I/O page where all memory mapped I/O is performed.

A variation on the model of all tasks having separate I/D segments is the case where two or more tasks share one or more I/D pages. The memory characteristics available to the separate tasks may be read-only or read-write and an intertask protocol must be utilized to sychronize accesses to such shared memory. For tightly coupled tasks such an arrangement allows the fastest exchange of data.

Creation and Operation

Tasks may be created dynamically but in practice the task environment is configured unique to the application. The priority of tasks is determined by the ordering of a simple linked list of structures that are referred to as *task tables*. The task table for a particular task contains all the task specific information required by the CRI; namely, the complete *state* of the task as required by the scheduling, common device interrupt, and trap service routines. In the KPNO real-time environment the multi-tasking structure is rigorously defined by the hardware capabilities and the observing protocols. The priority order of the tasks is fixed and equivalent to the creation order.

Throughout the FORTH-11 system, critical uninterruptible code has been limited. This attribute along with an event driven and interruptible scheduling routine and contiguous data files meets the demands of a real-time environment. In addition, the architecture of a task allows a simple and fast context switch and independent task modules. Each task is an independent entity that is interfaced to other tasks by a well defined protocol. The nature of the interface and attributes of memory management prevent a catastrophic task failure from directly affecting any other task. Tasks may be loaded with binary images to initially start a task, recover a crashed task, or to overlay a working task.

Communication and synchronization

Since each task is a virtual FORTH computer that executes commands input from its virtual terminal, a very simple communication mechanism is provided by virtual terminal I/O. For example, one task can request an action from another task by sending the destination task an ASCII command string through its virtual terminal (i.e., typing on the virtual terminal). Such a link can be a uni-directional command link or a bi-directional pipe. When the pipe is established the tasks can communicate in a full duplex mode. The virtual terminal link is very flexible and lends itself to dynamic alterations of the virtual or physical network. In cases where the virtual terminal link is too slow or laborious, several other communication mechanisms are provided such as shared memory or disk files, or memory packet communication. The more useful methods for task communication and synchronization follow.

Input or output via the virtual terminal.
    (Redirection of standard I/O, command or parameter strings.)
Bi-directional memory pipe.
    (Parameter or small blocks of data, packets.)
Sychronized shared memory.
    (Synchronization flags or large data blocks.)
Software *events* for activating a waiting task.
Extended memory block transfers. (unmapped memory)
Command or data files from disk.

The CCD Camera Application

The KPNO CCD camera system is a mobile instrument that uses a charge coupled device (CCD) microchip for the detection of images. The system is capable of acquiring data using a CCD detector while simultaneously displaying, processing, and recording prior images. An astronomer using the system is able to fully reduce the acquired data and extract astronomical results in real-time. The system can be used with any CCD chip format currently available. There are six such systems currently in use; five at Kitt Peak and one at the Cerro Tololo Interamerican Observatory (CTIO).

An LSI 11/23 microcomputer is the host of a distributed network consisting of local camera and display processors and a remote telescope computer. The basic hardware configuration of the system is shown in Figure 5. The LSI 11/23 peripherals consist of graphic terminal, disk, tape, and plotting printer. High speed interfaces for the camera processor, display processor, disk, and tape, allow the real-time multi-tasking operating system to fully utilize the hardware while providing a responsive interactive user interface and background reduction processing.

The important FORTH-11 extensions are the following:

Overlay driver.
File system routines.
Redirectable standard input/output.
Redirectable error routine.
Dynamically switchable stack checking.
Dynamically switchable redefinition checking.
Binary loader.
Intertask communication routines.
Automatic interrupt routine ("ready" event).
Unexpected interrupt routine (all unassigned vectors initialized).

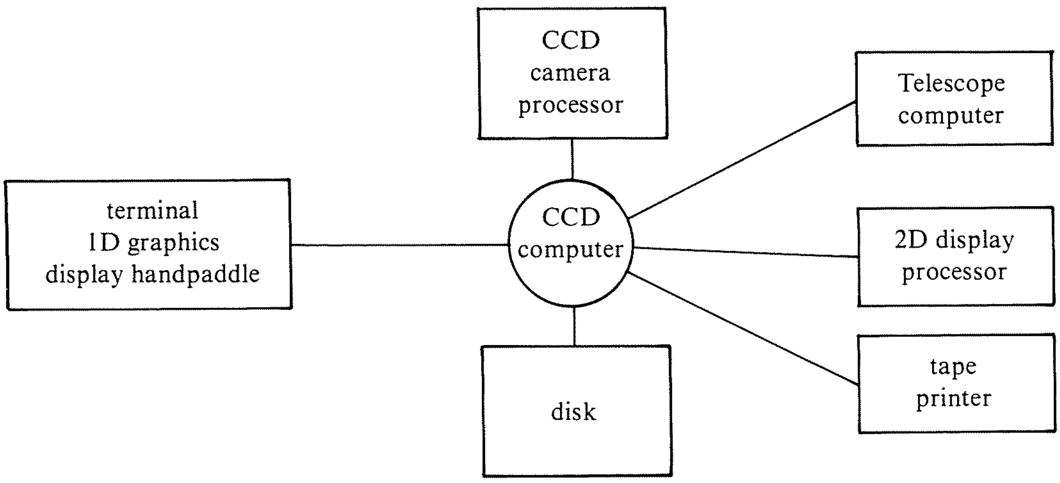

Figure 5.

The LSI 11/23 used for controlling the CCD system contains 256K bytes of RAM memory, hardware memory management, and micro-coded floating point instruction set. The mass storage device is an 80M byte Winchester technology disk that is used for both program and data storage. Serial interfaces are used for terminal I/O and the telescope link whereas DMA interfaces are used for the camera controller, 2D display processor, disk, and tape.

The CCD program consists of five tasks with each corresponding closely to one of the hardware elements listed above.

1. Camera control/telescope interface	
2. User interface	(terminal, 1D display)
3. Display interface	(2D display)
4. Data logging	(tape/printer)
5. Data reduction	(cpu/disk)

The software network is portrayed in Figure 6.

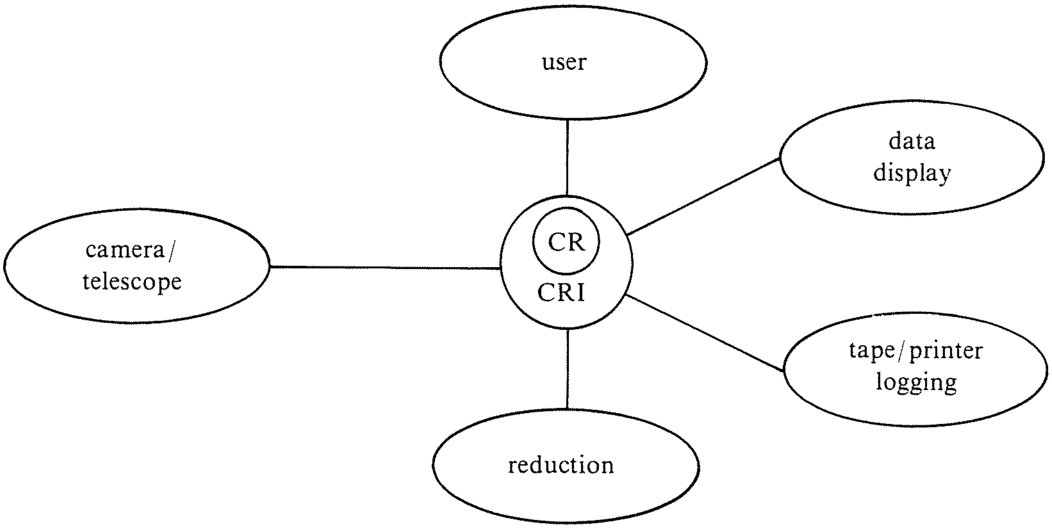

Figure 6.

In the CCD application the CPU is always allocated to the CCD control task first. This is to insure the uninterrupted time critical transfer of the camera data. Specifically, when a CCD integration is finished the camera controller interrupts the LSI 11/23 which must then initiate a DMA data transfer. At the completion of a transfer the camera/telescope task is made "ready" and will be scheduled so that the newly transferred data may be copied to the disk database. The time critical transfer of data into memory is controlled by an assembler interrupt service routine, whereas the disk transfer routines are high level FORTH routines.

Communication between tasks is facilitated using a variety of mechanisms. The most frequent method is for two tasks to hold a private conversation by typing on each other's virtual terminal. In this method character strings are passed back and forth that represent commands and parameters using redirection of standard input and output. Any commands that a user can type to a FORTH machine can be programmed to be sent from one FORTH task to another. In the CCD application there are numerous interactive routines that are executed in different tasks but which are setup through a conversation link between the user terminal and the executing task's virtual terminal. For the routines that do not need interactive attention, the complete command string with parameters may be sent as input to the destination task's virtual terminal; a method analogous to parcel post. The command may in fact be queued so that it will be executed by the busy task when it arrives at the top of the queue.

Each task manipulates digital images that are usually too large to be stored in physical memory. The image data is stored in the files on disk and is managed by database management routines referred to as the Picture Disk Manager (PDM). The PDM allows operations on virtual pictures at the levels of sub arrays, rows, columns, or single pixels. Dynamic format of two dimensional data is also provided.
A list of the capabilities of each task follows.

- Kernel task: CRI (common resource interface)
  - Trap and interrupt service routines
  - Task tables
  - Scheduler
  - Disk and terminal drivers
- Camera task: Camera control/Telescope interface
  - Local area network interface.
  - Dynamic camera control
  - Format, on-chip summing, pause-resume, abort-end
  - Real-time response
  - Database access
- User task: Operator interface
  - Network coordinator
  - Interactive graphics interface (one or two dimensions)
  - Operator observing interface
  - Database status
  - Data "quick-look" or interactive reductions and analysis
- Display task:
  - Interface for the two dimensional graphics system
  - Database access
- Tape task: Data logging
  - Drivers for tape and printer
  - Tape, disk database I/O
  - Printer spooler
- Reduction task:
  - Data reduction and analysis

Some of the capabilities of the reduction task that are available in real-time follow.

- Miscellaneous preparation for specific applications:
  - Extended precision picture summing and averaging.
  - Interlacing multiframe detectors.
  - Creation of special bias or flat field images.
- Removal of detector flaws. (bad pixels)
- Removal of spurious events. (cosmic rays)
- DC offset subraction and overscan trimming. (bias and trim)

Compensation for nonuniform quantum efficiency. (flat field)
Removal of monochromatic background features. (fringe removal)

The KPNO CCD Camera Systems have been in routine use since August 1981. The system has been a notable success in astronomy due to the functionality of the hardware, software, and overall ability to provide a complete tool for the astronomer at the experimental site in real-time. More detail on the implementation is given in reference [6].

References

[1] C. H. Moore, "FORTH: A New Way to Program a Minicomputer", *Astrn. & Astrphy. Supp.* 15, 1974, pp. 497-511.

[2] Computer Support Group, *Real-Time FORTH-11 Reference Manual*, Kitt Peak Natl. Obs., Nov. 1982.

[3] W. R. Stevens, "The Forth Programming System", *Proceedings of the IEEE Arizona Technical Symposium*, IEEE Press, 1980, pp. 81-84.

[4] Computer Support Group, *Forth Systems Reference Manual*, Kitt Peak Natl. Obs., Nov. 1982.

[5] W. R. Stevens, *A Forth Primer*, Kitt Peak Natl. Obs., (also published by Mountain View Press, Mountain View, CA), Oct. 1979.

[6] T. E. McGuire, "The Kitt Peak CCD Camera System", *Publications of the Astronomical Society of the Pacific*, 95:919-924, November 1983.

Tom McGuire was employed by Kitt Peak National Observatory (KPNO) from 1977-1983 as a research associate and senior scientific programmer and worked on telescope and instrumentation systems. Since then he has worked for Health Systems International in New Haven, CT. He received B.Sc. and M.Sc. degrees in physics in 1975 and 1977 and is currently enrolled in the Ph.D. program of Operations Research at Yale University. His interests include expert systems, information and optimization theory, and a wide spectrum of outdoor sports.