An Experiment in Remote Monitoring of Mu-Ranging Operation at Mariner Mars 1971 Superior Conjunction

D. E. Erickson and J. W. Layland
Communications Systems Research Section

This article describes the computer configurations and software used at JPL in an experimental remote monitoring and verification of the operation of the Sequential Component Ranging System during the Superior Conjunction of the Mariner Mars 1971 spacecraft. At the time of the spacecraft's closest approach to the Sun, the ranging operation was subjected to both an extremely low signal-to-noise ratio and perturbations from solar plasma. The Sigma 5 computer at JPL was programmed to perform a Maximum-Likelihood range measurement, using the range-code correlation values supplied (in real time) from the ranging system at the Mars Deep Space Station (DSS 14). The Maximum-Likelihood decision process provided about a 1.5-dB improvement in ranging error probability considering additive noise alone. The process was, however, relatively impractical to implement in the 920 computer which controlled the ranging operation at DSS 14 and performed range measurement via a sequential decision process.

I. Introduction

This article describes the computer configurations and software used at JPL in an experimental remote monitoring and verification of the operation of the Sequential Component Ranging System (Refs. 1 and 2) during the Superior Conjunction of the Mariner Mars 1971 (MM'71) spacecraft. At the time of the spacecraft's closest approach to the Sun, the ranging operation was subjected to both an extremely low signal-to-noise ratio and perturbations from solar plasma. The Sigma 5 computer at JPL was programmed to perform a Maximum-Likelihood range measurement, using the range-code correlation values supplied (in real time) from the ranging system at DSS 14. The Maximum-Likelihood decision process provided about a 1.5-dB improvement in ranging error probability considering additive noise alone (Ref. 1, p. 78). The process was, however, relatively impractical to implement in the 920 computer which controlled the ranging operation at DSS 14 and performed range measurement via a sequential decision process.

Fig. 1 shows a block diagram of the equipment involved. The ranging system proper was operated from DSS 14 during the period of interest. The normal data path for the ranging output data was via the Ground Communications Facility (GCF) teletype (TTY) lines to the Ranging Advisor position at Mission Control. For this experiment, the ranging data were also directed to the JGUS terminal in the Telecommunication Building. The SDS-910 computer which is attached to JGUS, in turn, transferred the ranging output to a time-shared program in the Sigma 5, which recalculated the range via the Maximum-Likelihood technique. At the completion of each range acquisition, the Sigma 5 program had either agreed with the sequential decision process or had calculated a correction term (in microseconds of round-trip light time) to be added to the range calculated by the ranging system at DSS 14. The appropriate result was relayed to the 910 computer and then, via the GCF TTY circuitry, to Mission Control. The likelihood function itself, the *a posteriori* probability of possible range values, was subsequently plotted (non-real time) and used as a final validity check on the range measurements. During the experiment, the Sigma 5 continued to be used in the background mode for other DSN development activities.

II. Software Structure in the Sigma 5

The relevant software operated in the Sigma 5 consisted of two distinct segments: the ranging monitoring program itself, written in FORTRAN, and a JPL-modified version of the Sigma Batch Timesharing Monitor. Both parts will be described in the following section.

Fig. 2 is an example of the data output from the ranging system at DSS 14 onto the GCF TTY lines. These data are received by the 910 in this form and delivered intact to the ranging monitoring program in the Sigma 5. The lines are scanned by the Sigma program to select relevant data from the ranging data format and to eliminate irrelevant "garbage" records. The "component number" field is checked to ensure that all of the needed correlation values are received. The "X correlation" and "Y correlation" fields represent the values obtained in the ranging receiver by correlation of the received range code against the local reference range code and a quadrature, or one-quarter-cycle delayed copy of the local reference range code. For square-wave range codes, these two correlation values are sufficient to estimate the phase and amplitude of the received range code and, from this, to reconstruct the correlation function between the local reference range code and all possible delays of the received range code. Assuming that the received code is disturbed by additive white gaussian noise, this latter correlation function is, in fact, the likelihood function, or the *a posteriori* probability of all possible range (delay) values.

Each ranging component received is an independent measurement of the range, with a resolution and ambiguity which depend upon the period of that square-wave component. The likelihood functions from independent measurements of the same parameter may be added directly, provided that proper account is taken of the code ambiguities and of the delays induced in the local reference code by the ranging system software at DSS 14. This accumulation of the likelihood function is done within the Sigma range monitoring software to a resolution of one-quarter cycle of the highest frequency code component. At the end of a range acquisition sequence, the location of the largest value is the most probable range value, or the Maximum-Likelihood estimate of the range value. The behavior of the sequential decision process with its attendant shifts of the local reference code has been concurrently reconstructed from the "X correlation" values. At the end of the acquisition, if the Maximum-Likelihood range estimate and the sequential decision process agree, that fact is transmitted to the 910 and then to appropriate destinations on the GCF TTY network. If they do not agree, a correction term is transmitted.

The Sigma Batch Timesharing Monitor (BTM) provides a limited multiprogramming operation, with a single background batch-stream and multiple time-shared terminal-oriented jobs. The batch stream operates from one area of the main memory, and the time-shared terminal jobs are swapped in and out of another area of memory. The time-sharing sub-executive, a collection of monitor routines occupying approximately 3.5K words, mediates between these two operations. Since both memory areas must contain a minimum of 12K words and the basic Sigma monitor occupies at least 7K words, operation via BTM with our 32K-word Sigma 5 was clearly impossible; yet, for ease of operation and flexibility in developing an experimental operation such as was performed with the MM'71 ranging, BTM was equally clearly desirable.

With some modifications to the time-sharing sub-executive, the batch-stream jobs could be caused to be swapped in and out of the same area of memory from which the terminal tasks were executed. Such a change would idle the entire system while the swap took place, but the alternatives were to abandon this monitor or to acquire an additional 16K words of main memory from which terminal-connected tasks could be executed. Given some reasonable assumptions as to terminal behavior, the swapping operation will idle less resources than the additional memory area, which is idle whenever all terminal tasks are waiting for data. For a general-purpose time-sharing system, the needed assumption is eight or fewer "typical" users. For our application where the terminals are, in fact, "minicomputers" with real-time data, the terminal behavior can be controlled, (if control is needed) to limit the extent of swapping and thus make the swapping of batch-stream jobs the preferable operation.

The needed modifications were performed primarily during the spring and summer of 1971. The time-sharing sub-executive was segmented into three parts. A minimal scheduler and the interrupt-connected portions of the terminal device handling software are always resident and, together with terminal-oriented buffers, occupy about 0.7K words. The available batch-stream area is 22.5K words, into which the appropriate programs are swapped. When terminal-connected tasks are active, another 2K-word segment of the time-sharing sub-executive occupies the lower end of this area. This second segment provides the remainder of the scheduling operations and analyzes the requests for monitor service (file I/O, etc.) by terminal-connected tasks. The third and last segment of the sub-executive interprets command-character strings from the terminal and is swapped in only when initializing a terminal-connected task.

After operation of the modified BTM was satisfactorily verified in this form, device-handling software to control the intercomputer link to the 910 was added and interfaced to BTM on a character-oriented level. The JPL BTM system was used in this form for the experiments with MM'71 and exists in that form now. A change to a message-oriented interface for the intercomputer links will be performed when higher-speed data transfers are needed. After the JPL BTM system was operational for some time, Xerox announced an essentially identical Batch Swapping BTM system for the small Sigma systems which were delivered in late 1972. Changeover to the manufacturer-supported BS:BTM will eventually be done, but the intercomputer link-handling interface must be added by JPL personnel.

III. Software Structure in the 910/JGUS Terminal

To properly handle real-time monitoring of the ranging system, the 910 had to be supplied with a program which would route data among various devices (Table 1). The I/O operations which it needed to perform were: input or output on the JGUS teletype (TTI or TTO), input from or output to the Sigma 5 computer via the intercomputer link (SI or SO), input from the keyboard or output to the typewriter (KB or TTY), or input from the photo-reader or output to the paper-tape punch (PR or PP). These eight I/O functions may be considered to be separate devices except for the typewriter and keyboard TTY and KB, which must be coordinated because output messages should not be interrupted for characters of input.

Only the Sigma link devices SI and SO are capable of any great speed; since data transmitted on this link would, for the most part, be transmitted or received on one of the slower devices as well, only character-oriented I/O was needed. This character orientation resulted in the simplest interface to the time-sharing monitor on the Sigma computer.

Some of the functions performed by the 910 had to be relatively independent. For instance, if the Sigma 5 link failed, a paper tape of the ranging data would still be punched. To aid in the control of independent functions on the 910, a multitasking monitor was developed. Once the monitor was completed, independence of the tasks was easy to implement and the application program Mu 910 was very easy to debug.

Mu 910 is composed of many independent tasks which transfer characters from one device handler to another under the control of a set of software switches, as well as the hardware sense switches (SSs) of the 910. Other tasks control the settings of the software switches. Some switches in Table 2 are set by a task which reads and interprets operator keyins from the 910's keyboard (KB). This task is called into action when hardware sense switch 1 (SS1) is set. Others (Table 3) are altered by a task which scans data being received from the Sigma computer when the switch RNG is set to ON. SS2 and SS3 are used as on–off switches for PP and PR, respectively. Fig. 3 shows the possible paths which the data can take depending on the switch settings, 3a showing which switch setting makes each path active, and 3b showing the real-time monitoring configuration.

To operate Mu 910, the program must be started and, using keyins, the switches properly set. Using the keyboard, the operator logs onto BTM on the Sigma and loads the previously compiled program from the Sigma random-access disk (RAD). This program then completes the connection by sending the appropriate characters to the 910 to switch it to teletype rather than keyboard input.

The operator at the 910 may use the typewriter for three keyboard control features. He may alter the keyin controlled switches (see Table 2), check the status of these switches, or send a double escape to BTM. To activate these features, the operator sets SS1 on the 910 console and then resets it. The keyboard control program will prompt with a square right bracket and will accept an operator keyin. The legal keyins are shown in Table 4. An illegal keyin causes the program to type KEYERR and repeat the prompt.

The keyin-controlled switch RNG is a master for the Sigma-controlled switches. When RNG is off, RANGE, RGO, and TTYON are also off. When RNG is on, the task SIEAR (Sigma Input Ear) scans each data line, beginning with the first character which is not a line feed or carriage return, for the sequence <RANGE+>! (! representing a carriage return). If it finds this sequence, it turns RANGE on and scans subsequent lines for "Δ" or "<RANGE−>!" or "<RGO+>!". The first two of these turn RANGE off; the latter turns RGO on. The appearance of a "Δ" in the character stream from the Sigma 5 indicates that the ranging program on the Sigma has exited to the monitor, probably because of a program error, and that cleanup/termination of ranging operations must be performed. If RGO is on, the scanner looks for "Δ", "<RANGE−>!", "<RGO−>!", or "<HEAD>". The last of these is a signal to send a teletype header with the data on the remainder of the line and turn TTYON on. The full header line is:

"<HEAD>aaaa dd/tttt!"

where aaaa is the four-character teletype destination address, dd is the day of the month, and tttt is the time in GMT.

Once TTYON is on, data received from the Sigma are transmitted on the teletype until "Δ", "<RANGE−>!", "<RGO−>!", or "<EOM>!" is received. The last of these causes an End-of-Message sequence to be transmitted and TTYON to be shut off. Note that, when RNG is shut off, it automatically shuts off RANGE, which in turn shuts off RGO, which kills TTYON so that no switches are left dangling.

The Sigma ranging program therefore outputs "<RANGE+>" to connect itself with the ranging input device. It outputs "<RGO+>!" to type a message on the 910 typewriter and follows it with "<RGO−>!". To transmit to the JPAS teletype, it sends "<RGO+>!" followed by "<HEAD> JPAS dd/tttt!", the message, "<EOM>!", and "<RGO−>!".

IV. 910 Multitasking Monitor

The 910 Multitasking Monitor (MULTI) is a set of subroutines and 910 POPSs (Programmed Operators: subroutines with a special calling sequence). These are listed in Table 5. When using MULTI, a program may be considered to be composed of interrupt routines and tasks. An interrupt routine is a section of code which is given control due to a hardware interrupt, performs a specific action such as reading a character from an I/O buffer and transferring it to a program buffer, and then relinquishes control to the code which was being executed before the interrupt occurred. A task under MULTI is a conceptual unit of the program which performs a set of actions in a specific sequence. It may be temporarily interrupted by a hardware interrupt, but not by another task. A task relinquishes control voluntarily to allow a higher-priority task to gain control or to await an action by another task or interrupt routine.

Each task is identified with a Task Control Block (TCB). The TCB is a data area which MULTI uses for bookkeeping for the task. It is an eight-word block (Fig. 4a) which contains the priority, starting address, and status of the task; a four-word area to save the registers and return address when the task relinquishes control; and a one-word chain field used by MULTI for stringing the tasks together into queues. By using queues of TCBs, MULTI is not restricted as to the number of tasks which it can control. The priority of a task is a small non-negative integer. The first task used from a queue is the one with the highest priority.

The first task is created and the multitasking monitor initialized by a subroutine call to INIMON, passing as a parameter, the TCB address for the first task. With this call, the CPU queue is cleared, a timing task is added to the CPU queue, control is returned to the point following the call, and the named TCB is made the current TCB. The current TCB or current task is the one which has control of the CPU. It executes until it relinquishes control, and at that time a new current task is taken from the CPU queue. New tasks may be created by using the "FORK" POP, passing to MULTI the address of the TCB of the task to be created. The new task will be added to the CPU queue.

The current task may relinquish control in one of four ways: DIE, BLOCK, WAIT, or CHECK. Each of these is a POP. DIE has no parameters and specifies that the current task relinquishes control and its TCB is not to be put into any queue. It is forgotten, but may be restarted by a

FORK from another task. BLOCK and WAIT are similar POPs. With BLOCK and WAIT, the parameters are a Resource Control Block (RCB) address and a State Report Block (SRB) address, respectively. RCBs and SRBs are identical in structure and differ only in function. (They are described in the following paragraphs.) CHECK is a BLOCK on the CPU queue.

The RCBs and SRBs are two-word data blocks used for communication among tasks. The first word is a count, and the second is the head of a queue. The queue is empty when the count is not positive. When the count is positive, it represents the number of TCBs on the queue. The head field is a pointer to the first TCB, while the chain field of each TCB points to the next TCB on the queue. The TCBs are arranged on the queue by descending priority. Among TCBs of the same priority, first-in, first-out order is observed. When a new TCB is put onto a queue, it is placed immediately before the first TCB of lower priority. To facilitate end conditions in this search down the queue, a dummy TCB is attached after the end of every queue whose priority is -2 and whose chain field points to itself. This was also an aid in debugging MULTI.

An RCB is used to control the use of a limited resource, such as an I/O device, which can be used by only one task at a time. The RCB controlling CPU usage is CPUQUE. If the current task needs a limited resource, it blocks on its RCB with the BLOCK POP. If the count field of the RCB is non-negative, the resource is not available. In this case, the current TCB is added to the queue on that RCB, and the count is increased by one. If the count is negative, it is bumped by one to signal unavailability to other tasks, and the current task is given control of the resource. In blocking, however, the task has lost its right to the CPU, so MULTI blocks it on CPUQUE (whose count is never negative). Now the CPU is available, and MULTI unblocks it by removing the first task on CPUQUE, making it the new current task and reducing the count of CPUQUE by one.

When a resource becomes available, its RCB is unblocked by either a task or an interrupt routine using the UNBLOCK POP. The count field in the RCB is decremented by one, and, if the count is now non-negative, the queue is not empty and the first TCB is removed and blocked on the CPUQUE. No routine except MULTI should unblock the CPUQUE. After an UNBLOCK, control is returned to the routine issuing it and there is no change of current task. The typical sequence for a task is to block on a resource, use it, and then unblock it.

There are other uses for an RCB. To wait for another task to fill a buffer, either character or message, and then signal the other task that it is available again, a task may block on a message RCB, read the message, and then unblock a buffer RCB. The task which sends the message blocks on the buffer RCB, writes the message in the buffer, and then unblocks the message RCB. An RCB may be unblocked even though the count is negative, and the count will still be decremented. In this way, unblocks may be accumulated.

The above feature is employed in the use of circular character buffers. Two RCBs control each circular buffer; they are called Number of Characters (NC) and Number of Characters Left (NCL). There are also two pointers, IN and OUT. To put a character into the buffer, a task blocks on NCL, puts the character into the buffer in position IN, updates IN to point to the next character circularly, and then unblocks NC. To get a character from the buffer, a task blocks on NC, takes a character from position OUT in the buffer, updates OUT, and unblocks NCL. Thus, when the NCL count is negative, its absolute value is the number of character positions left to fill the buffer. Similarly, when the NC count is negative, its absolute value is the number of characters in the buffer.

When a task performs a WAIT on a SRB, if the count is non-negative, the action is the same as BLOCK. The TCB is added to the queue and the count is bumped. If, however, the count is negative, it is not bumped, but the TCB is still blocked on CPUQUE. The count of an SRB is reset to zero from a negative value by a SHUT POP. If the count is non-negative, SHUT has no effect. SIGNAL removes all TCBs from an SRB queue, blocks them on CPUQUE, and resets the count to a negative value.

A task waits for an SRB in situations similar to test loops in non-multitasking programming. MULTI provides SRBs which it signals and shuts, depending on the states of the hardware sense switches. BPnON, n = 1, 2, 3, or 4, is SIGNALed when the sense switch is set and SHUT when it is reset; BPnOFF operation is the opposite. Thus, a task may WAIT for BP1ON if it wishes to be reactivated when SS1 is set. These SRBs are updated each time the current task relinquishes control by DIE, BLOCK, WAIT, or CHECK. Also updated by MULTI at this time are the RCBs of WBLK and BREAK. WBLK controls the use of the W buffer which four of the I/O devices share. The BREAK RCB is unblocked once each time the hardware interrupt button is pressed. The subroutine call to INIMON which initializes MULTI also initializes these RCBs and SRBs.

A timing task FORKed by MULTI at initialization time runs when all other tasks are blocked on RCBs or waiting for SRBs. It uses an SRB named TIC. The sequence of operations performed by this task is as follows: SIGNAL TIC, SHUT TIC, loop through 128 CHECKs, and branch back to the start. The CHECK POPs allow any task of higher priority to take control. If no higher-priority tasks want the CPU, the timing task continues. When not interrupted, the entire loop takes about one-half second. A task wishing to wait for approximately n seconds may loop through 2n WAITs for TIC. This obviously is not very precise, but it was perfectly satisfactory for five-second time-outs on typewriter and paper-tape input.

Several I/O routines are also provided with the multitasking monitor. These consist of device interrupt handlers and device tasks, as well as subroutines which are to be shared by user tasks. The interrupt routines effect the actual data transfer between the physical device and a circular buffer in core. The device tasks control the activating of the interrupt routines, the sharing of the W buffer through the WBLK RCB, and the timing-out of I/O operations which use the W buffer. The shared subroutines are the interfaces between the user and the device handlers. To use a given I/O device, a user task blocks on the RCB associated with that device. The devices and RCBs are listed in Table 1 along with the names of the entry points to the non-translating subroutines associated with that device.

To perform typewriter output, a task blocks on TY1W and then makes subroutine calls to TYPPUT, with the characters to be typed right-aligned in the A register. When completed with its message, the task unblocks TY1W, making it available to other tasks. TYPPUT puts the character in a circular buffer, and, if it was previously empty, it unblocks the device task, which in turn acquires the W buffer and starts the interrupt routine.

For input, the situation is a bit more complicated. We would like to start filling up the buffer when input is requested, and not as each character is removed from it. To read from the keyboard, a task blocks on KB1W and calls KBSETP, which starts the buffer-filling operation. Then, to get a character from the buffer into the A register, the task calls KBPULL. When it is done with its use of the keyboard, the task calls KBREL to halt the buffer-filling operation and then unblocks KB1W to allow its use by other tasks. Another entry KBCANC is provided for other tasks to cancel the current keyboard input operation. This is used by the keyboard time-out task, for example. Usually, before a task begins a keyboard input operation, it blocks on TY1W as well as KB1W to eliminate interference from typing tasks.

Other entries are available which do appropriate character translation and call xxxPUT or xxxPULL. These are named xxxOUT and xxxGET, respectively. Thus, TTGET calls TTPULL and translates the input character from TTY code to BCD. The translation routines TYPOUT and KEYIN translate between the typewriter's BCD and a modified ASCII for use with the BTM interface.

V. Conclusion

The system described herein was operated for several weeks surrounding the Superior Conjunction of the MM'71 spacecraft; it provided considerable assistance to the Celestial Mechanics Experiment team in clarifying and evaluating ranging data obtained during that critical period. As can be seen from the descriptions presented in this article, the software investment is relatively high, but much of it may be recoverable for use in other activities. The modified BTM system has already proven valuable in software development for a number of projects. MULTI, the minimal multiprogramming monitor for the 910, was written with the expectation that the needed data transfer and conversion operations could be programmed significantly easier with such a monitor. This expectation seems to have been fulfilled. The total 910 range-monitor software package seems to be flexible enough that most experiments involving JGUS and the Sigma 5 could be performed with it in its present form. For applications which require a transfer path or data conversion not available in the present package, MULTI provides a flexible base upon which the needed software can be easily built.

References

1. Timor, U., "Sequential Ranging With the Viterbi Algorithm," in *The Deep Space Network Progress Report for January and February 1971*, Technical Report 32-1526, Vol. II, pp. 75–79. Jet Propulsion Laboratory, Pasadena, Calif., Apr. 15, 1971.

2. Goldstein, R. M., "Ranging With Sequential Components," in *The Deep Space Network for the Period May 1 to June 30, 1968*, Space Programs Summary 37-52, Vol. II, pp. 46–49. Jet Propulsion Laboratory, Pasadena, Calif., July 31, 1968.

Table 1. 910 I/O devices

Device description	Abbreviation	RCB name	Subroutine entries
Typewriter	TY	TY1W	TYPPUT
Keyboard	KB	KB1W	KBSETP, KBPULL, KBREL, KBCANC
Paper-tape punch	PP	PP1W	PPPUT
Photo-reader	PR	PR1W	PRSETP, PRPULL, PRREL, PRCANC
TTY output	TTO	TTOBLK	TTOPUT
TTY input	TTI	TTIBLK	TISETP, TIPULL, TIREL, TICANC
Output to Sigma	SO	SOBLK	SOPUT
Input from Sigma	SI	SIBLK	SISETP, SIPULL, SIREL, SICANC

Table 2. Keyin-controlled switches

Switch	Value	Meaning
TT	0	Ignore TTY input
TT	1	TTY input to PP only
TT	2	TTY input to PP and Sigma
RNG	ON	Activate ranging (see Table 3)
RNG	OFF	Deactivate ranging
RO	1	Ranging output to TY only
RO	2	Ranging output to TY and TTO
RI	KB	Ranging input from keyboard
RI	PR	Ranging input from photo-reader
RI	TT	Ranging input from TTY
NUM	nnndd	nnn = channel number, dd = day of last teletype message sent

Table 3. Sigma-controlled switches

Switch	Value	Set by	Meaning
RANGE	ON	RNG ON and <RANGE+> from Sigma	Get Sigma output from ranging input device KB, PR, or TT (see Table 2)
RANGE	OFF	RNG OFF or Δ or <RANGE−> from Sigma	Get Sigma output from KB
RGO	ON	RANGE ON and <RGO+> from Sigma	Send Sigma input to TY
RGO	OFF	RANGE OFF or <RGO−> from Sigma	If RANGE ON, ignore Sigma input If RANGE OFF, Sigma input to TY
TTYON	ON	RGO ON, RO 2, and header from Sigma	Send Sigma input to TTO as well as TY
TTYON	OFF	RGO OFF, RO < 2, or <EOM> from Sigma	Do not send Sigma input to TTO

Table 4. Operator keyins

Keyin[a]	Action
bb	Send a double escape to BTM
!	Cancel request for keyin
anything + + +	Ignore this line, repeat prompt
switch value!	Where switch is a keyin-controlled switch TT, RNG, RO, RI, or NUM and value is a legal value for it, set switch to given value (e.g., RI TT!).
S!	Type values of switches
anything else!	Type KEYERR, repeat prompt

[a]! represents a carriage return, and + + + represents a delete.

Table 5. MULTI subroutines and POPs

Name	Parameter	Function	Subroutine (S) or POP (P)
INIMON	TCB address	Initialize MULTI and create first task	S
FORK	TCB address	Create new task	P
DIE	—	Delete current task	P
BLOCK	RCB address	Block on resource	P
UNBLOCK	RCB address	Mark resource available	P
WAIT	SRB address	Wait for a condition true	P
SIGNAL	SRB address	Signal condition true	P
SHUT	SRB address	Signal condition false	P

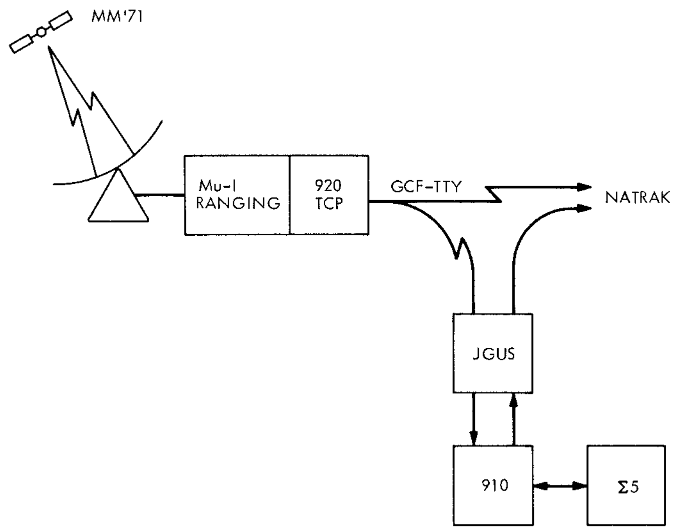
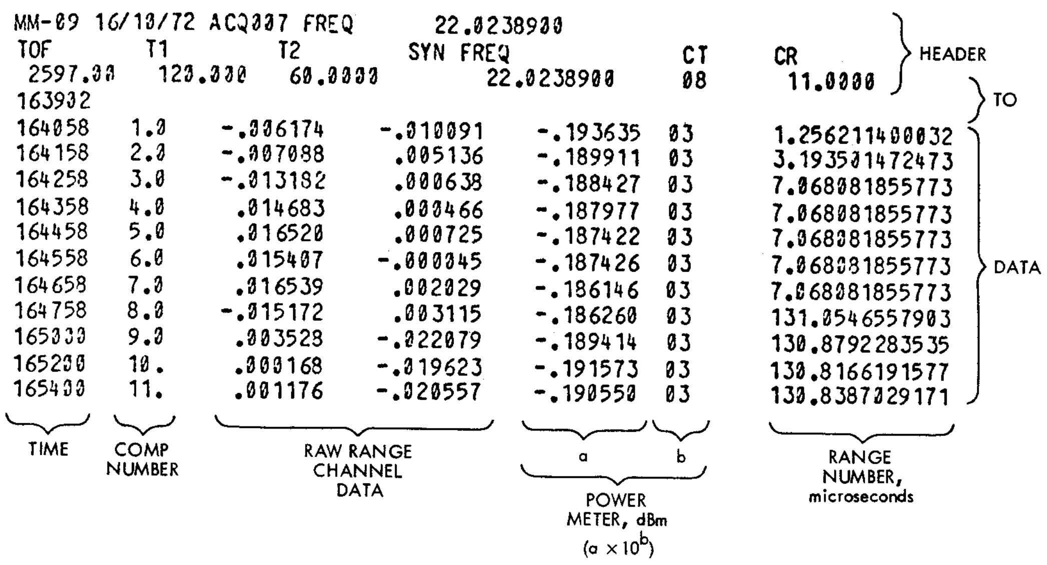
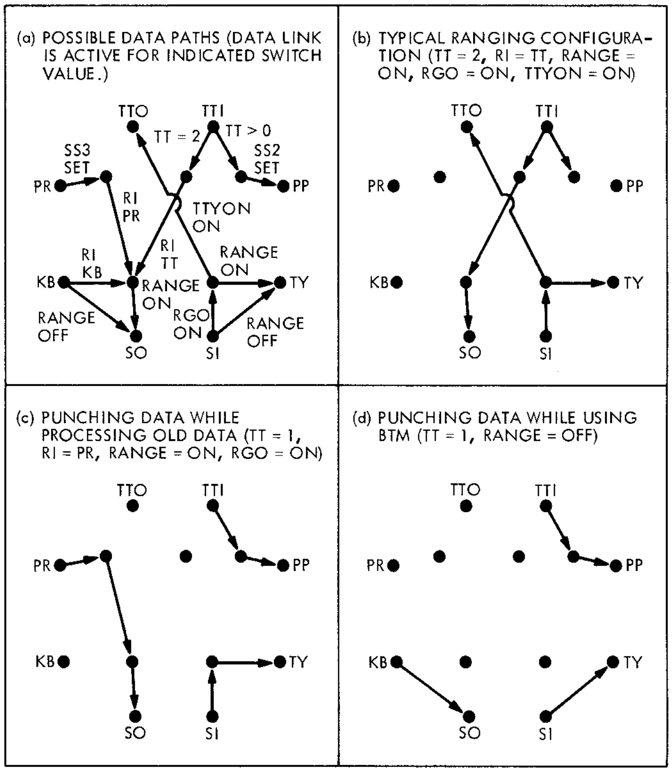


Fig. 1. Experiment configuration



```
MM-09 16/10/72 ACQ007 FREQ        22.0238900
 TOF          T1          T2          SYN FREQ              CT       CR
  2597.00    120.000    60.0000          22.0238900        08        11.0000
 163932
 164058    1.0     -.006174     -.010091     -.193635  03      1.256211400032
 164158    2.0     -.007088      .005136     -.189911  03      3.193501472473
 164258    3.0     -.013182      .000638     -.188427  03      7.068081855773
 164358    4.0      .014683      .000466     -.187977  03      7.068081855773
 164458    5.0      .016520      .000725     -.187422  03      7.068081855773
 164558    6.0      .015407     -.000045     -.187426  03      7.068081855773
 164658    7.0      .016539      .002029     -.186146  03      7.068081855773
 164758    8.0     -.015172      .003115     -.186260  03      131.0546557903
 165000    9.0      .003528     -.022079     -.189414  03      130.8792283535
 165200    10.      .000168     -.019623     -.191573  03      130.8166191577
 165400    11.      .001176     -.020557     -.190550  03      130.8387029171
```

Fig. 2. Super Mu printout



(a) POSSIBLE DATA PATHS (DATA LINK IS ACTIVE FOR INDICATED SWITCH VALUE.)
TTO
TTI
TT = 2
TT > 0
SS3 SET
SS2 SET
PR
PP
RI PR
TTYON ON
RI TT
RANGE ON
RI KB
KB
RANGE ON
TY
RANGE OFF
RGO ON
RANGE OFF
SO
SI
(b) TYPICAL RANGING CONFIGURATION (TT = 2, RI = TT, RANGE = ON, RGO = ON, TTYON = ON)
TTO
TTI
PR
PP
KB
TY
SO
SI
(c) PUNCHING DATA WHILE PROCESSING OLD DATA (TT = 1, RI = PR, RANGE = ON, RGO = ON)
TTO
TTI
PR
PP
KB
TY
SO
SI
(d) PUNCHING DATA WHILE USING BTM (TT = 1, RANGE = OFF)
TTO
TTI
PR
PP
KB
TY
SO
SI

Fig. 3. Mu 910 data routing configurations

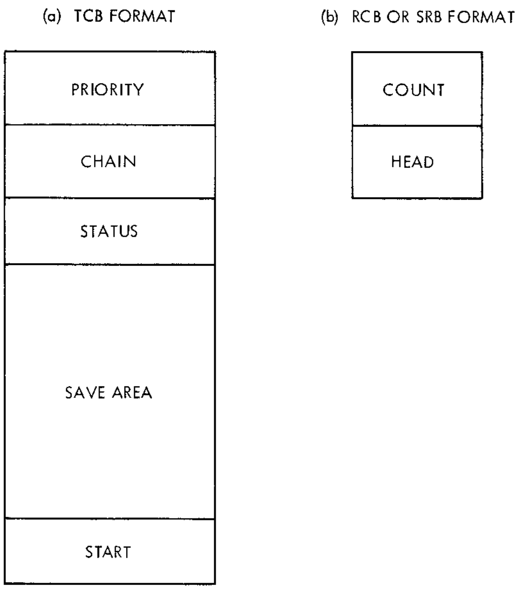


Fig. 4. TCB format and RCB or SRB format