MIDISS: A UNIQUE MULTI-PROCESSOR TELEMETRY GROUND STATION[1]

DONALD L. FEINBERG
Spacetac, Inc.
Bedford, Massachusetts

Summary An optimum architecture for real-time telemetry data acquisition and display has been developed. It provides a high degree of flexibility and throughput while using a minimum of hardware: as applied to the MIDISS system, this approach enables users to specify data processing for each sensor on several satellites independently of downlink formats. Each user can specify and modify CRT displays without interrupting the data processing performance. Processing may include algebraic manipulation, logical branching, logic products, time averaging, etc. This flexibility is provided by SPACETALK, a FORTRAN-like language that enables the user to deal with each sensor by name. Users can also format and transmit uplink commands.

Hardware minimization is evidenced by MIDISS's compactness. It consists of a 16-bit minicomputer and a SPACEPIPE (microcomputer-based, front-end processor) that decommutates multiple downlinks, provides rate-buffering, and performs data compression and alarm-checking. The minicomputer is thus free to perform extensive data processing for those downlink channels that are changing at significant rates,

Introduction In many mini-computer or midi-computer based ground stations, a requirement to change the downlink format or to modify data processing instructions can give rise to a major systems reprogramming effort. Further, a significant portion of the processor's capability is usually devoted to repetitive front-end communication tasks. The addition of the front-end pre-processing modules tailored to perform high-speed communication tasks, data compression, and alarm checking, would dramatically free main frame capability for more flexible, more complex processing.

At any given time, only some downlink channels from a satellite are changing at a significant rate. Therefore an optimal system should be able to decide which downlink sources actually require attention from the mainframe.

[1] This work was performed under contract to the Office of Naval Research, Naval Research Laboratory.

From the user's viewpoint, the optimum system should enable him to specify processing for each channel (data point), and to display intermediate or final results wherever desired on his CRT monitor, independently of that data point's location in the downlink format. It would also allow him to change the display during downlink operations. Further, he wants the ability to interact with the system himself via an English-language program, rather than use system programming specialists to implement his requirements. He also would like to calculate and display functions of several variables, and functions that involve data sampled at successive times, with the same ease as simpler functions.

This paper describes hardware and software concepts that provide the above capabilities inexpensively. The concepts have been successfully applied in Spacetac's general purpose telemetry station, MIDISS, which is fairly compact - it has been used at a launch site for launch operations, and then re-located to a ground station for orbital operations

System Description MIDISS provides a unique data acquisition and display oriented language compiler which allows user specification of all data conversion and display activities within the system. Each independently remoteable console (Figure 1) can selectively display converted and computed data from any spacecraft sensor, while at the same time sending commands (where applicable) to any of the data sources.

The system uses a microprocessor controlled, data pre-processor called a SPACEPIPE. The SPACEPIPE decommutates and compresses data from three asynchronous downlinks. This data is passed to a PDP 11/40 host computer through a shared memory. The SPACEPIPE's primary function is to maintain the most current value for each channel in this memory at all times.

To do this, it uses two Schottky bi-polar microcomputers in a pipelined configuration. The first microcomputer in the pipeline interfaces the downlink frame synchronizers and performs preliminary data verification. Selected data may be extracted for buffering directly to the host computer, while other data are pushed to the second pipelined microcomputer via a hardware FIFO. On the pull end of this FIFO, the second microcomputer accesses the rate-decoupled data and completes the decommutation. process. The now identified data points are subjected to floating aperture data compression and alarm checking before being passed to the shared memory. When significant data changes occur, the name of the associated spacecraft sensor is queued to the host computer.

The host computer executes the data processing routine (analysis, conversion and display) corresponding to each queued sensor channel, These routines are coded in advance by the user in an English-like language called SPACETALK, which is "compiled" into packages of context-free intermediate code and stored on a fixed-head disk. The package, or set of

packages, associated with each queued data source is “executed” at object time asynchronously, as needed.

SPACETALK provides the user with several capabilities, First, the arithmetic power of a FORTRAN-like compiler language (including a set of library functions) is available to the user. Secondly, logical and conditional branching allows real time (self) modifiable displays and alternative processing. Thirdly, SPACETALK data processing is specified symbolically: sensor names are defined without reference to their telemetry frame (line, word, subcom) position. A decom program correlates names with telemetry positions, giving a decommutation scheme for a particular format. Several decom programs can be stored, allowing a set of SPACETALK data processing instructions to run with a variety of telemetry formats, changable in real time. Finally, a sophisticated and extremely flexible display scheme is provided, controllable both by compiler instructions and by the human operator. Any value can be displayed, along with engineering units and messages.

All of this capability is available to a user who is familiar with FORTRAN-type programming, but knows nothing further about computer systems or telemetry systems. Examples of user-generated processing routines are given later in this paper.

The system also provides extensive up-link command assembly and formatting capability.

Hardware Description Most of the major hardware elements shown in Figure 1 are standard units from various manufacturers, and thus will not be described in detail. The mainframe occupies two 69"-high, 19" racks, and includes the host processor - a Digital Equipment Corporation PDP-11/40. There are two dual-port memories on the PDP-11 Unibus -that are shared with the SPACEPIPE. The fixed-head disk is a high-speed unit (250kHz word transfer rate with 8.5 ms average access time). The SPACEPIPE and the Control and Display Consoles, on the other hand, embody innovations that add greatly to MIDISS’s flexibility and power, and are described below.

Using one SPACEPIPE increases the host processor’s downlink rate capability by an order of magnitude, yet increases hardware costs by only a small amount. SPACEPIPE occupies a 9" high, 19-inch shelf. The primary tasks assigned to it include: 1) electrical interfacing with three downlink synchronizers; 2) handshaking for data transfer; 3) validating incoming data (several frames of data must be held before being passed to the rest of the system; 4) passing certain data directly to the mainframe; and then 5) decommutating and sub-decommutating, in several levels, the remaining data; 6) compressing data (both zero-order and floating aperature); 7) checking all changed data against individually predefined alarm limits; 8) storing the most current data into a shared memory located on the PDP-11 Unibus; and 9) presenting to the host processor an over-run protected queue (software FIFO) of significantly changing channel addresses.

Schottky microcomputers were selected for these tasks because of their inherent high-speed (complete instruction execution in 300 ns). Each contains a PROM, for storing fixed generalized algorithmic control instructions (firmware), and a small, high-speed RAM. Since the firmware is not modifiable, control tables are stored in the RAM and can be modified by the host computer for overall control, and by input data conditions for internal algorithmic control.

Interconnection among the microcomputers and the host processor is shown in Figure 2. The microcomputer ports shown are tri-state, and each provides both input and output. The primary communication path between the SPACEPIPE and the host computer is implemented using two dual port, PDP-11 Unibus compatible memories, one dedicated to each microcomputer. Data are deposited by each microcomputer into its assigned dual port memory, where it can be accessed asynchronously by the 11/40. Low speed control information is also communicated between the microcomputers and the 11/40 using the shared memories. A secondary communication path between the SPACEPIPE and the host computer is a high speed, 16-bit duplex interface. It directly connects each microcontroller to the 11/40 Unibus and enables interrupt type handshaking.

Microcomputer No. 1 performs and/or controls tasks 1 through 4 listed above. It receives data from the frame synchronizers via optical isolators. Hardware latches store service requests from the frame synchronizers, and MSI priority encoders indicate the highest priority request pending. Data from the corresponding latch are passed, with only one microcomputer instruction,[2] directly to shared memory 1 in the mainframe, and stored there temporarily while microcomputer 1 performs data verification. This shared memory also provides a direct communication channel to the PDP-11/40 for those data that do not require further SPACEPIPE processing, and it provides some of the buffering required for data that must be passed to microcomputer 2.

Electrical communication between microcomputer 1 and microcomputer 2 is accomplished with a 61-byte (8 bits) hardware FIFO. The hardware FIFO allows a rate decoupling between the processing going on in microcomputer 1 and the continuation of that processing in microcomputer 2. The hardware FIFO allows coherent processing at the average processing period of either machine rather than at the sum of the periods of processing in both machines. Throughput enhancement is a function of the buffering depth of the hardware FIFO as well as the difference between the shortest processing period and the longest processing period in the two processors.

Microcomputer No. 2 performs functions 5 through 9 listed above, and also inserts a time-of-year code. As described in the software section of this paper, it presents processed data to shared memory 2 in the mainframe, as well as two queues - a 25-element alarm queue

[2] For implemetation of this scheme and other details on the SPACEPIPE, see reference (1).

that contains the addresses of data sources that are out of limits, and a 1,000 element normal processing queue.

Both microcomputers in the SPACEPIPE can interrupt the host processor when immediate attention is required.

The second hardware element to be described is the operator's control and display console. Each console fits one rack, and includes three CRT monitors, a keyboard, control and power units, and a signal interface. One monitor (called the common display) tells each operator the current status of MIDISS's telemetry and command systems, describes the status of the other consoles, and provides an independent alarm message area for each telemetry link. The other two monitors are dedicated to the operator's use for offline (preparation time) processing, display, and command instructions, and for on-line (run-time) TLM link selection, display specification and/or modification, and uplink command execution. (The system can be set up to either allow several operators to execute uplink commands, or to limit command capability to one operator). Several types of displays are available on the monitor, as described in the software description in this paper.

A key feature of the console hardware is its signal interface, which allows remoting each console up to 1,000 feet from the mainframe via coax cables. Alternately, via 9600 baud modems and telephone lines, consoles can be remoted as needed.

Any display can be saved, either in hard copy or in one of the electronic storage media available.

Software Description - DATA BASE ORGANIZATION For each telemetry link (Figure 3), one can consider that five storage locations are significant to MIDISS's operation. The key to the system is that the decommutated data from the SPACEPIPE is placed directly into a shared core memory segment. There is no data communication on-line handshaking. All pre-processing in the SPACEPIPE is transparent to the host processor, which accesses data and request queues at its convenience on its side of the shared memory.

The PDV (Process Data Vector) shown in the shared segment is the table of values kept current by the SPACEPIPE. Each location in this table is associated with a unique data source on the satellite. This approach allows "relative to base" addressing.

The set of PRV's (Position Resolution Vectors) shown is essentially a map - each PRV correlates the location of a word in the telemetry format (e.g. supercom; maincom, sub-com, and sub-subcom values) with a location in the PDV. The host processor generates a new set of PRV's during preparation time for each new format anticipated. Because all

decommutation is performed by the SPACEPIPE, the host processor has no need to access the PRV's during run-time, so the appropriate PRV's are loaded into the microcomputer's private RAM each time the format changes.

The third data interface between the microprocessor and the host processor contains two FIFO's which enable asynchronous "queue coupling" between the processors. These FIFO's are over-run protected: if the SPACEPIPE receives a new data value for a sensor whose PDV address is already in the FIFO (awaiting processing), it updates the data value in the PDV, but does nothing to the FIFO. Thus the FIFO capacity is used efficiently, and the processor capacity is not spent processing "old" data. (Data that changes faster than MIDISS processes it, is changing faster than a human being can follow on a CRT).

The remaining two data base "locations" shown in Figure 3 contain user-specified processing and display instructions called process control blocks (PCB's). One or more PCB's may be generated for each data source, and any PCB may process any number of data sources. Normally, most PCB's are stored on the disk, with a "most recently-used-cache" algorithm being used to store some in core. If a PCB requires the reading of data at various times, temporary storage is available for intermediate values. A PCB may include any of the instructions available in SPACETALK.

Software Description - PROGRAMS The PCB's and other software modules are related to each other, and to the data base, as shown in Figure 4.

During off-line operations, the SPACETALK compiler generates the PCB's based on user-coded instructions. The PCB's reference downlink sensors by name. The PRV generator combines sensor information from the compiler with telemetry format information (not necessarily from the same user) to produce the PRV's. Also, the user has the option to specify display parameters during off-line preparation time and save them for use during run time.

Run-time operation uses an executive and an interpreter in the mainframe, and microcomputer firmware in the SPACEPIPE. The interpreter performs the specified processing under executive control. Also, users frequently take advantage of the display software capability to change display parameters during real-time runs without having to "take the system down".

Each of the software modules shown in Figure 4 has clearly defined logical interfaces, and can be considered to operate independently of the others. Thus, they are described separately in turn below.

Spacetalk Compiler SPACETALK is a Fortran-like data processing language that was developed by Spacetac specifically for telemetry applications. Included are a complete arithmetic and logical branching capability, plus the capability to compute logical products (e.g. exclusive OR), and extensive output formatting and display capability. Complicated or often-used functions may be established in a library, and called by name as needed in any user program. The user writes Fortran-like programs in SPACETALK (Figure 5) dealing with his spacecraft sensors only by names (which he invents). He specifies all the data processing and display that is to take place for each sensor. The processing can vary, dependent on external or internal information which can be tested.

If the name of a PCB is the same as a sensor name, it will he executed during run-time whenever the SPACEPIPE queues that sensor to the mainframe, If the name is not a sensor name, the PCB will be executed only when called by another PCB (or by the MIDISS system, for example, when the system requires time-of-year code conversion). The user may make any intermediate value in a PCB available to other PCB's, merely by declaring it to be a GLOBAL variable. One PCB may have any number of sensor inputs. (Any PCB may read any location in the PDV.) If desired, data and interim results can be stored, so the PCB's requiring multiple sensor readings over a period of time can be accomodated. There is capability for specifying DISPLAY and SAVE (e.g. on disk) formats for any value appearing in a PCB, as well as for messages. For each sensor, the SPACETALK user also indicates the type of data compression desired, and specifies processing limits (value changes that should trigger normal priority or ALARM priority processing of the PCB). If he does not enter such limits for a given sensor, MIDISS can enter default limits.

After the program is coded and edited in source language by the user, the SPACETALK compiler generates two outputs. The first is a context-free, intermediate-code representation of each block of processing information (the PCB's). These will be later "executed" by the run time interpreter during system use to perform the actual data processing. The second output, derived by implication from the SPACETALK source language program, is a catalog that lists the sensors that the compiler needs to deal with, and specifies where it expects to find their values in the PDV. That is, the PCB's have been encoded with the assumption of where in the PDV the sensor information is kept. These compiler outputs are independent of telemetry formats and can be used with a variety of them. The PRV generator program which correlates the PDV with any given downlink format is described next,

PRV Generator PRV's are blocks of information which the microprocessors can use to decom the incoming data on one downlink. The PRV generator is a program which has as its input:

1. The catalog of sensor names and the assumed PDV positions that was output by SPACETALK.

2. Frame/word/sub-com position of sensors, by name, as specified for each downlink format.

The program then generates a set of PRV's for each format. At runtime, the set corresponding to the current format is loaded by the runtime executive into the microprocessor's private memory, providing a link form the un-decommutated data to the interpreter. If the downlink format changes during runtime, the PRV's corresponding to the new format are loaded, independently of the data processing. In this manner, MIDISS permits a number of downlink schemes, while the user need specify the data processing only once.

SPACEPIPE Firmware The basic job of these routines is to decommutate frame synchronizer outputs, load them into the PDV within shared memory No. 2, land queue PDV addresses that require processing. The firmware does this by using control tables loaded by the host processor, for example, the PRV's and the processing limits. Firmware tasks are organized into relatively small processing modules, each of which is executed under executive control and returns to the executive within a minimum period, so that the executive can then service the highest priority task currently outstanding. (A full range of suspended background tasks can be maintained.) This and other concepts described in Reference 1 contribute to an optimized SPACEPIPE throughput rate.

Runtime Executive This program principally performs system resource management. It is the only program visible to the user at runtime (unless he wants to change the display); it services all his runtime communication needs with MIDISS.

Typical downlink functions are to process requests from the user to load a new set of PRV's or to begin or halt runtime execution; or to pass display change information to the runtime interpreter (without taking the system off-line or otherwise interrupting real-time operation). Typical uplink functions enable a user to formulate, edit, store, and execute command strings to the spacecraft. (This latter could be through a variety of transmission equipment).

Runtime Interpreter The runtime interpreter is the program which performs the actual data processing in the MIDISS. Data processing procedures are kept in the PCB's, and the incoming data is kept in the PDV. The runtime interpreter acts upon these instructions and information. It also performs all displays, which may be user-specified entirely (or partially) at compile time and then modified (or completed) during runtime. The display specification process is described next.

Screen Painting and Display Setup Software The user can construct displays by simply naming the parameter desired and the location where he would like it to appear. Further examples of display flexibility include: the same parameter may be called to more than one location; the same parameter may appear in more than one form (decimal, octal, binary, floating point, scientific notation); ASCII messages and units can be preassigned or can appear based on data content; alarm, commands, and data can be assigned to separate areas of the display; a large number of parameters may be called up with one blanket name; etc.

The user may specify display information either completely or partially. A complete specification made during preparation time directs MIDISS to always display the indicated data at the specified place, regardless of other system conditions - for example, alarm conditions. A partial specification allows the display to be either turned on or off at runtime, and in addition, positioned on a CRT screen at runtime. Since the number of displayed variables may be high, software exists to organize the effort.

Before actual data is run, but with the MIDISS runtime system loaded, the user can exercise his display options. “Null” displays appear on the CRT’s showing the user what he has done. (Modification is possible). Finally, the screen paintings are saved and may be used at runtime - in fact, there may be several such paintings which can be invoked and changed at runtime. (This is not to say that the user loses runtime control over individual displays - he does not; any parameter can be called up or re-located during runtime, without affecting system data processing performance. This software only makes it convenient to make large numbers of specifications in a way that they can be invoked with a simple command at runtime.)

Naturally, permanent logging of specified parameters (such as alarms and commands) is available.

Conclusion The architecture described allows the desired flexibility. For a system with 1,000 downlink data sources, changes in data processing instructions, for any given source, have been implemented rapidly, with no impact on the overall system. Changes to the downlink format have been successfully accomodated on-line, with no interruption to the system performance. This flexiblilty has proven to be extremely useful during launch preparation and during orbital operation.

The hardware approach chosen - using LSI bi-polar microprocessors as pre-processors - has provided high reliability as well as a large increase in throughput for a small cost increase. This type front-end provides a good cost-to-processing-rate ratio for mini or midi computer systems.

Reference (1): S.K. Brede, R.L. Gonyea, and M.L. Harvey "High Speed Data Processing System Using Bipolar Microcomputers," Proceedings, Industrial Electronics and Control Instrumentation Group, 1976 Annual Conference, IEEE, New York, N.Y., pp 62-67; 1976

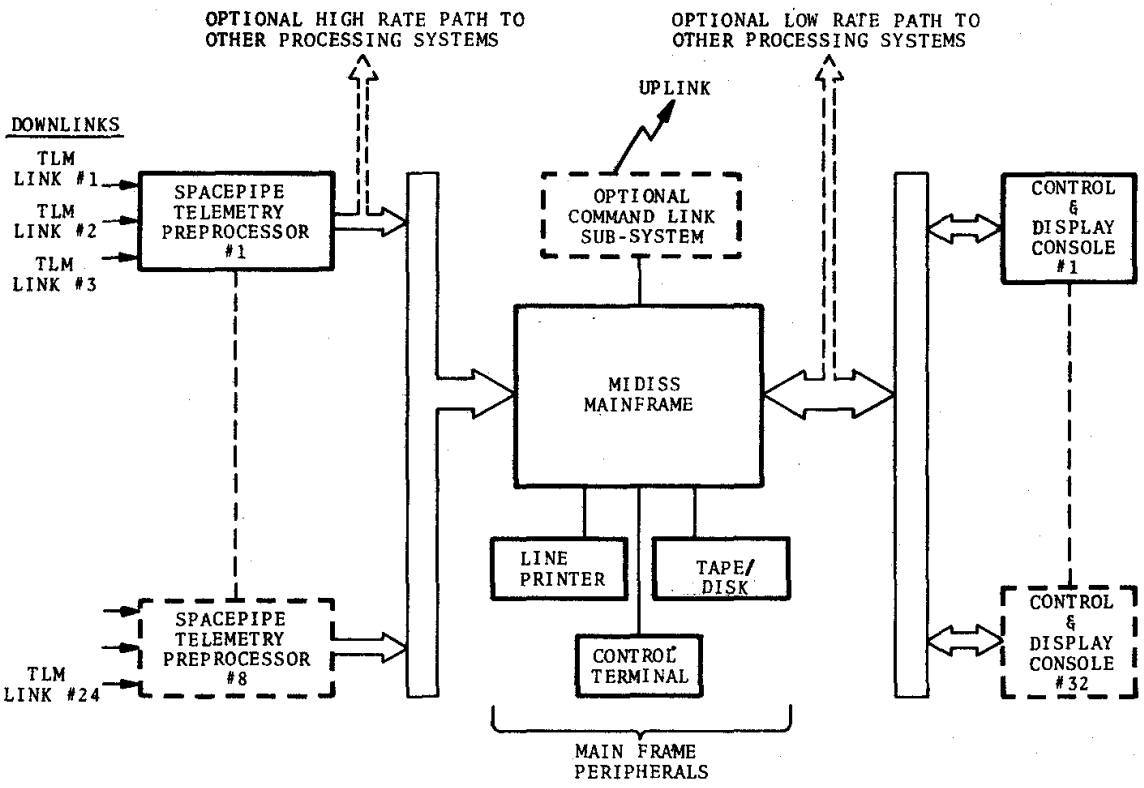

Figure 1. MIDISS Architecture. One SPACEPIPE interfaces three telemetry links that have a total data rate of 1 Mbps.

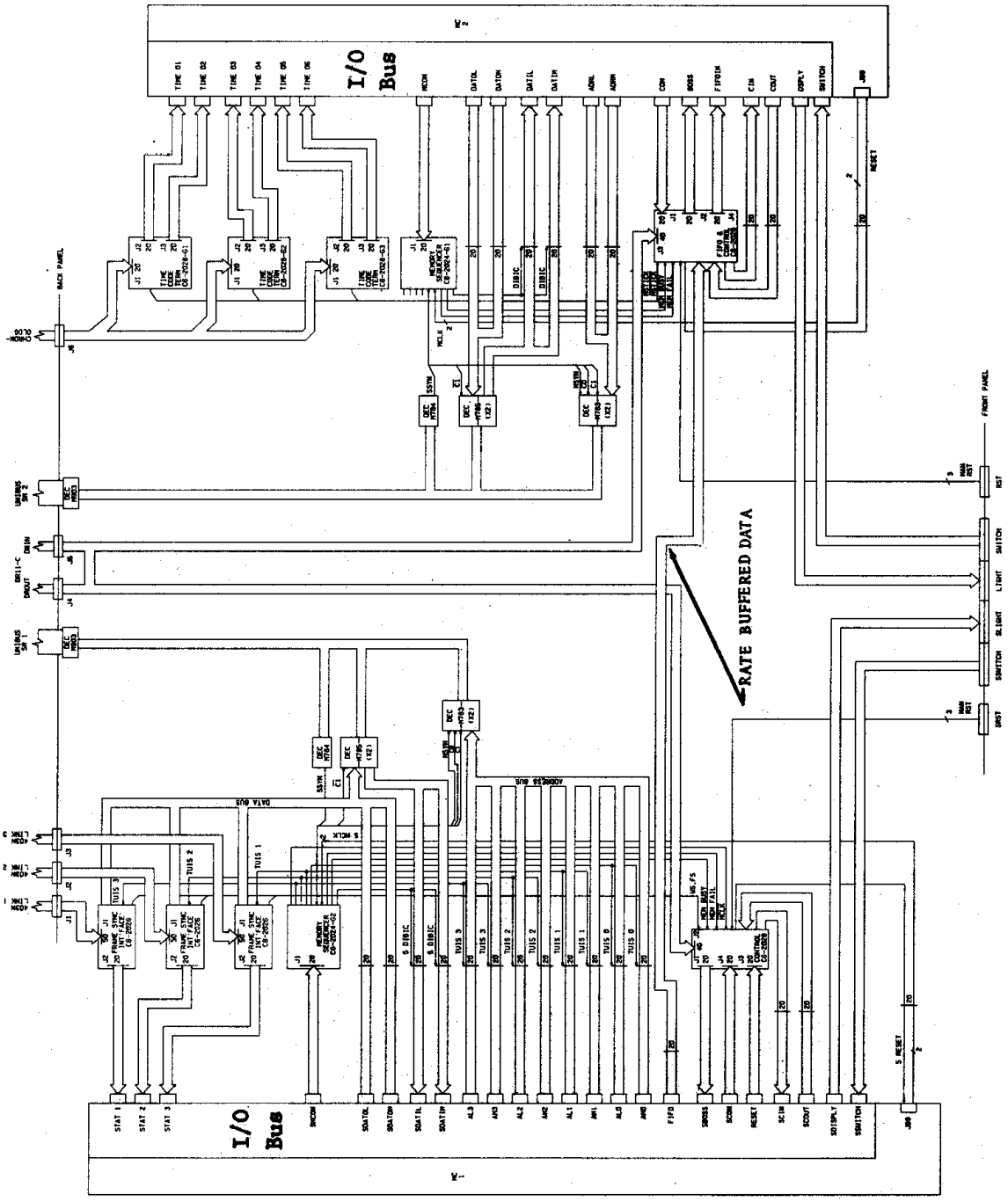

Figure 2. SPACEPIPE Interconnections

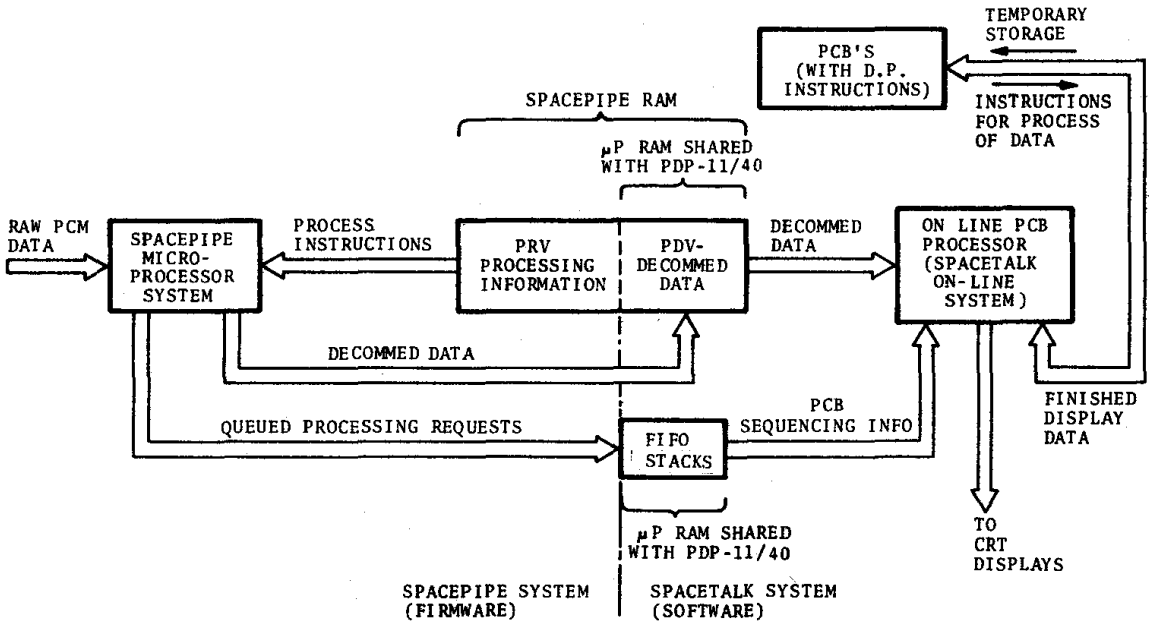

Figure 3. Data Base Structure for One Downlink

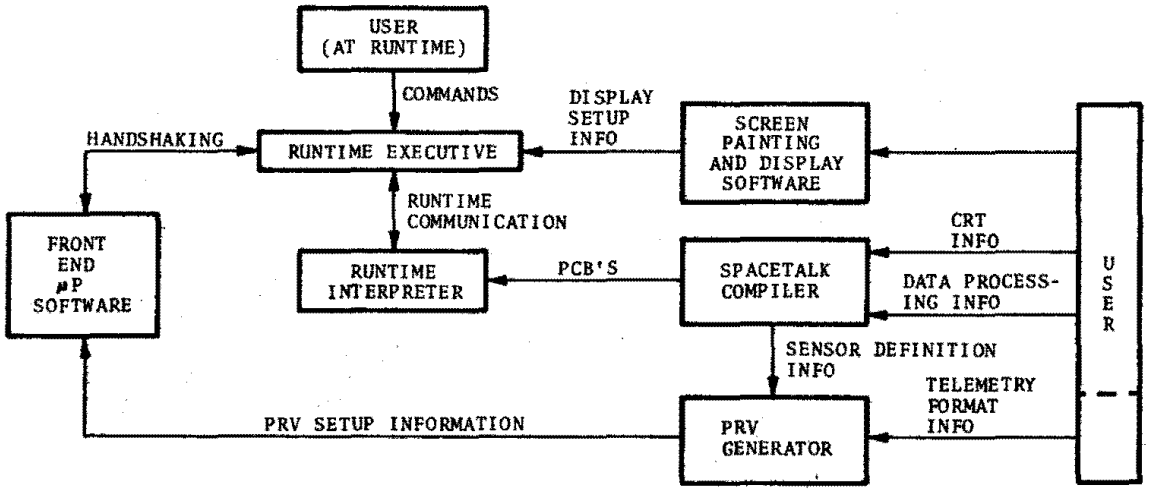

Figure 4. MIDISS Software Structure

```
//PCB CELL1VOLTAGE

        IN THIS PCB, THE AVERAGE VOLTAGE
        OF SIX CELLS IS CONVERTED FROM
        ANALOG TO ENGINEERING UNITS,
        THEN AVERAGED


DEFINE PDV CELL1VOLTAGE,CELL2VOLTAGE
DEFINE PDV CELL3VOLTAGE,CELL4VOLTAGE
DEFINE PDV CELL5VOLTAGE,CELL6VOLTAGE

C0 = 0.1234
C1 = 0.567


CONVERT PDV VARIABLES TO FLOAT

        C1V = FLOAT(CELL1VOLTAGE)
        C2V = FLOAT(CELL2VOLTAGE)
        C3V = FLOAT(CELL3VOLTAGE)
        C4V = FLOAT(CELL4VOLTAGE)
        C5V = FLOAT(CELL5VOLTAGE)
        C6V = FLOAT(CELL6VOLTAGE)

DO ENGINEERING CONVERSIONS

        C1V = POLYN1(C0,C1,C1V)
        C2V = POLYN1(C0,C1,C2V)
        C3V = POLYN1(C0,C1,C3V)
        C4V = POLYN1(C0,C1,C4V)
        C5V = POLYN1(C0,C1,C5V)
        C6V = POLYN1(C0,C1,C6V)

COMPUTE AVERAGE

        VAVG = (C1V + C2V + C3V + C4V + C5V + C6V) /6


        DEFINE STRING CAVOLT

        CONV/CAVOLT/'AVG CELL VOLTAGE =',VAVG(F6.3) 'V'
        DISPLAY CAVOLT

        STOP
```

Declare Spaceborne Sensors as PCB Variables

Initialize Constants

Fixed-to-Floating Conversion

Conversion to Engineering Units Using First-order Polynominal Expansion Function POLYN1 = $\sum_{j=o}^{i} C_j x^j$

The Resulting Display
Might be AVG CELL VOLTAGE = 1.125 V

Figure 5. Typical SPACETALK Processing Program