# FORTH AS THE BASIS FOR AN INTEGRATED OPERATIONS ENVIRONMENT FOR A SPACE SHUTTLE SCIENTIFIC EXPERIMENT

**Henry M. Harris**
*Mission Design and Operations Manager SIR-B*
*Jet Propulsion Laboratory/California Institute of Technology*
*4800 Oak Grove Drive 156/220*
*Pasadena, CA 91103*

## Abstract

Over a period of three years, a FORTH-based system was developed by JPL for the operations of a major scientific instrument onboard the Space Shuttle. The software had to meet a very demanding operations environment where the the interactiveness of the software was not merely desirable but essential to the success of the mission. Forth was chosen for its capability of integrating divergent software needs into an interactive package. The mission flown in October 1984 was beset with numerous hardware failures and challanged the capability of the system to its fullest.

## Introduction

In November of 1981, the Space Shuttle was first used as a platform for scientific instruments. Orbiting above the Earth at an altitude of 259 km, the Shuttle's 60 foot long cargo bay contained a number of scientific instruments for studying the Earth from the Shuttle's unique perspective. Principal among these was the Shuttle Imaging Radar-A (SIR-A) JPL built managed and performed the ground-based operations. As a sucessor to this high successful experiment a more complex radar experiment was flown in October of 1984. See [ELA82]. Both of these experiments used ground-based software to facilitate command and control of the instruments. (This phase of the experiment is usually referred to as "mission operations".) The software, especially in the the second experiment, proved indispensable in reacting to real-time problems that develop in any space mission where successful data gathering may depend upon a timely solution to unplanned events.

This paper will survey software concepts developed for this project and discuss how the software (called the Shuttle Mission Design and Operations System or SMDOS [HAR84]) was used for the SIR-B mission and the results obtained.

## Instrumentation

Mission operations was conducted for both flights at Johnson Space Center

(JSC). Although some computer support was provided by JSC, most SIR specific software had to be generated by JPL on computers that were to be transported to Houston form JPL's Laboratories in California. Rather than centralizing the software in a minicomputer, a distributed intelligence concept was evolved that connected six microcomputers, each performing a parallel task for the mission planning cycle.
The system used on the SIR-B mission were six IBM XTs linked together by an ethernet network. Each computer was equipped with an IBM monchrome and an RGB moniter. The RGB was driven by a standard IBM graphics card. One of the computers was used as a file server for the network. Figure 1. shows the configuration of four of the computers and identifies their function.

## Forth System

Forth was selected as the language for all of these tasks for a number of reasons:

1) Forth provides accessibility to the machine in a way high-level language actively discourages.

2) The interactive environment of Forth allows rapid program development.

3) Forth is extensible in a way that allows diverse programming tasks operating in an integrated environment.

4) Forth is compact which in this case was essential considering the large amount of tasks that had to be resident in memory at any given time.

The Forth system used was PC/Forth+ by Laboratory Microsystems Inc., a 32 bit implementation with extensions for the 8087 math coproccesor.

## Scope

The software that was needed for the space mission was diverse in applications. Modules that were created for the 1984 mission had to accomplish the following tasks:

1) High-speed orbit propagation calculations that had double-precision accuracy.
2) interactive computer graphics that created area-filled aniamated color pictures of the Earth as would be seen from the shuttle bay.

3)A hueristic simulation of the performance of the radar under a wide variety of conditions.

4) Monitor and decommutate the real-time telemetry stream as it was transmitted to Earth from the Orbitera via the TDRS satellite. The software had to simultaneously decommutate, select, store and record the raw data in real-time from two seperate data sources as provided by JSC.

5) Edit a data-taking plan containing hundreds of separate events while providing constraints in the type and range of the variables.

## Approach

Because the amount of code produced (almost one megabyte) was so large, a considerable amount of effort was given to adopting proceures and programming practices which encourage readability and a uniform approach to programming. Three areas in particular were targeted for development:
1) data structures
2) array equations
3) graphics.

These specific areas will serve to illustrate the approach taken in creating SMDOS .

### 1) Data structures

The data structure word created for SMDOS is TYPE. TYPE compiles a structure into memory while creating a named instance of TYPE. The word created by type can then be used inside new TYPE definitions that inherit the structure. The TYPE instances are used in SMODS to create records, files and arrays. A simple example of a TYPE definition is the following:

```
TYPE COMPLEX
   real        : REAL
    imaginary :  REAL
END.
```

This definition results in the word COMPLEX appearing in the dictionary. COMPLEX can then be used for array or file . Execution of COMPLEX leaves the number of bytes in the record on the stack and creates the words "real" and "imaginary" in the dictionary. Execution of "real" or "imaginary" leaves the offset to the "COMPLEX" record on the stack. For example let is define the complex file Z.

```
10 COMPLEX FILE Z
```

The number preceeding COMPLEX is the number of buffers equal in length to one record of COMPLEX created. To store the real part of the complex mumber Z we have

```
45.23 E o real Z F!
```

Note that the address of the real part of Z is defined by the selector word "real." The command

```
' Z PUT
```

moves and internal pointer to the next storage area . If the FILE has been associated with a disk file name using

```
" Complex.DAT" ' Z OPEN
```

then the buffer area will be written to disk when when the pointer reaches the end of the allocated storage.

This is very brief look at the approach to data structures that was taken in the implementation of SMDOS. For more information on Forth data structures see [BAS83].

## 2) Arrays

The principle advantage to be gained by using the array concept can be realized by the implementation of the array operator. Simply providing a scheme to provide storage for and a method of indexing arrays is not much of an improvement over using scalars. It is important that these operators are object oriented; that is, they operate on a class of object called an array which can be defined at any time for the particular application. (To extend the object concept even further the fields are accessed by selectors which are analagous to the method of the object.) Array operators implemented include:

```
A+ ( <array1> <array2> -> <virtual array> )     - additton
A- ( <array1> <array2> -> <virtual array> )     - subtraction

A* ( <array1> <array2> -> <virtual array> )     - multiplication
x  ( <array1> <array2> -> <virtual array> )     - cross product
```

Arrays are created by a defining word with the following syntax:

```
ARRAY X 1 3 OF REAL
```

which creates an array of rank 1 ( a vector ) of floating point numbers. Arrays of any rank or dimensionality may be created but an error condition may result if an inappropriate operator is applied. Another example of array creation is:

```
ARRAY Y 3 2 2 3 OF COMPLEX
```

which defines an array of rank 3 with dimensions 2 by 2 by 3. COMPLEX was defined above under data structures. Addressing the real component of the [1,2,1] element of the array Y is accomplished by the form:

```
real Y { 1 2 1 }
```

The result will be to leave the address of that particular component on Forth stack. It is probably worth repeating here that, in general, arrays are ultimately only a useful concept when most of the work is done by operators. Access to individual elements should be kept at a minimum.

## 3) Graphics

The graphics standard selected is a subset of several well-known graphics standards such as the ACM Core and the GKS standard. This particular subset was chosen because it is well described in a popular book on graphics. Our graphics module is called the Simple Graphics Package (SGP) which is discussed in the book Fundamental of Interactive Computer Graphics by Foley and Van Dam. See [FOL82]. This package includes words such as:

```
MOVE_ABS_2 ( <X> <Y> -> - )
MOVE_REL_2 ( <dX> <dY> -> - )
POINT_ABS_2 ( <X> <Y> -> - )
LINE_ABS_2 ( <X> <Y> -> - )
LINE_REL_2 ( <dX> <dY> -> - )
POLYGON ( <X_array> <Y_array> n -> - )
TEXT ( <string> -> - )
```

where the comment in parenthesis give the stack effect of these Forth words. <X> refers to a floating point number representing an X coordinate. <X_array> and <string> are address' of an array and a string repsectively.
An inportant part of the SGP was the concept of the window and the

viewport constructed by the words:

```
WINDOW ( <x_min> <x_max> <y_min> <y_max> -> - )
VIEWPORT ( <x_min> <x_max> <y_min> <y_max> -> - ).
```

The WINDOW word specifies the window into world coordinates, that is the actual coordinates used by the application, which will be displayed on the device.

VIEWPORT specifies the area of the device viewing area that will be used in Normal Device Coordinates (NDC).

All of the words in the SGP are device dependent. In this case this means that all graphics produced by SMDOS could be sent to a pen plotter, a CRT screen or a printer. SGP worries about the translation into each device's particular language. A further enhancement that is being currently developed is to add the capability to create a standard intermediate graphics file (using the NAPLPS standard for example.)

The fact that once a structure is defined its characteristics can be inherited by other new structures is a very powerful concept for organizing data. The use of selector words for accessing fields with these structures lends itself to code that is more standardized and improves readability. These concepts are important on any large programming project.

The simularity to the PASCAL TYPE (and more remotely to smalltalk inheritance and message sending concepts) should be apparent and is another example of the adherence to known or often used structures to create Forth extensions.

## Programming Style

The management of such a large programming task in Forth was helped considerably by the adoption of constraints on programming style. The objective was consistent readable code independent of the particular programmer involved. Two elements of this style that were emphasized were indentation and delimiter rules. An example can serve to illustrate both ideas.

```
  360 0
 DO
    ?WITHIN_WINDOW
   IF
     PLOT_GROUND_TRACK
   THEN
 LOOP
```

Note that the delimiters DO, LOOP and IF, THEN are always on the same column.
Also the logical elements contained by the delimeters are always indented. This is, of course, wastefull of disk space considering the way Forth stores its screens but the waste is more than made up for by the clear, readable source code that results. Disk space is not considered a problem by us in these days of multi-megabye disk drives.

## Mission Planning

The object of mission planning is to develop a plan to achieve all the scientific objectives within the contraints of the mission. Since these objectives depend upon very dynamic geometrical relationships, it is important to have tools which can predict these relationships from known initial conditions and integrate these with the specific scientific objectives.

One of the scientific objectives was to image specific targets on the Earth from different angles from Earth orbit. A file was created using the TYPE word explained above which contained information for each of the investigators's sites. This file was then automatically called by SMDOS when constructing windows onto the Earth 's surface and polygons were drawn to represent the sites for the investagators.

The known initial conditions were state vectors provided by Johnson Space Center which represented a known orbit and position of the space shuttle. These vectors were propagated forward in time in one minute increments to produce an ephemeris file resident on the IBM hard disk.

Mission planning could begin by entering SMDOS, defining a view window of some area of the Earth of interest, selecting a science file, some ephemeris which contained the planned orbit and simply instructing SMDOS to create a window that would have all these elements overlayed in one projection. This image could be sent to a screen, plotter or printer in any of number of projections as selected from a menu. Standard mecator, cylindrical and polar projections are available.

The shuttle orbit was represented by a projection of the orbit track on the ground through the center of the Earth. In addition the trace on the surface of the Earth created by the viewing cone of the radar ( called the footprint) could be shown on the device selected. The radar is capable of tilting in the azimith direction and this angle can be simulated by the software with the appropriate change in the radar footprint reflected in the display.

On-off times for the radar and tilt angle can be adjusted interactively until the operator had satisfied himself that the required targets are to be imaged under conditions dictated by the needs of the scientist and the constraints of the mission.

The process described above can be repeated as many times as necessary as the precise orbit and science opportunities were refined. Once the final plan had been decided upon the command generator portion of SMDOS could take over. Photo 1. (taken during the SIR-B mission) shows a science work station using a planning computer. The screen in the photo shows various sites of scientific interest oulined by polygons superimposed over a region in the Middle East The footprints of the radar are the two stripes that travel from top to bottom of the screen.

The command generator works from the mission plan file to translate the plan into a form understood by the shuttle computer called the command file. Once the translation process has been completed and verified, the command file is transmitted to PAYCOM where it is integrated with all other payload commands and orbiter events. Once PAYCOM has verified that the command list violates no known constraints the command list can be transmitted to the shuttle at the times with which each command is flagged.

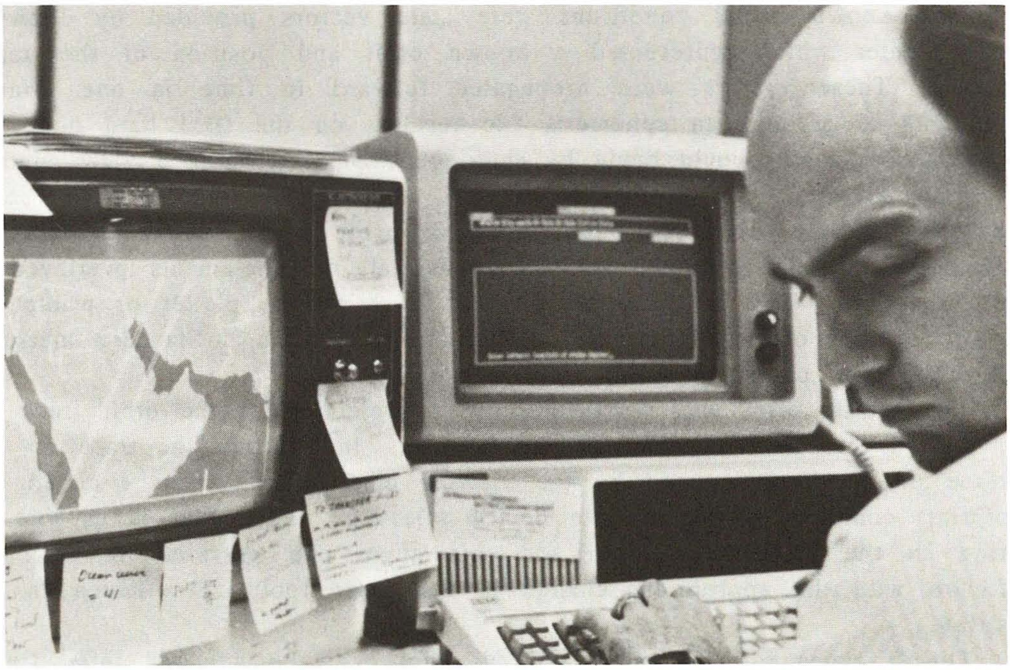


Photo 1. Science Planning Workstation

## Mission Operations

Two weeks before the launch all computers were shipped to JSC and installed in the Mission Operations Control Room (MOCR.) Telemetry was provided by two interfaces, a high data-rate link (KU band) and a low data-rate link (S band.) Also a command link was established to JSC's Payload Conmmander (PAYCOM) over ordinary telephone lines via a 1200 baud modem for transmission of the commands generated by SMDOS.

Liftoff occured without incident on the morning of October 4,1984. After opening the huge cargo bay doors the SIR-B antenna was deployed and we made ready to take our first data.

The computers monitored the orbit. SMDOS displayed ground tracks over selected sites on the Earth. The orbits matched our planning; we could proceed with the planned scenario of data taking.

It was in the first day of data taking that we noticed the first problem. The reconstructed echo display provided by one of the SMDOS computers, that normally would show the bell-shaped curve of the radation pattern that was being returned from the Earth, went flat. Normal engineering telemetry was insufficient to allow the radar engineers to diagnose the problem.

The engineers decided on a course of action. Since the returned echo was negligible as received by the SIR-B antenna it was decided to increase the gain of the receiver. The problem was in not understanding what had happened to cause the failure it was not immediately apparent what the appropriate gain should be. The reconstructed echo display could be used interactively to judge the effect of a gain increase but it had not been designed for this purpose and was not calibrated properly.

No problem. With the advice of the radar engineers, the Forth module that was used to generate the display was quickly modified to produce a calibrated display. The gain of the receiver was increased until a perfect bell-shaped pattern again appeared on the display.

We discovered later that a cable feeding the signal to the transmitting antenna had developed a short that antennuated the signal by 10 db. Instead of irradiating the Earth with 1000 watt bursts we were transmitting about 100 watts, the power of an ordinary light bulb.

This was only the start of our problems. A satellite on board failed to deploy properly. The shuttle had to remain in high orbit until the problem was resolved before it could fire its engines to descend to the orbit that had been planned for the SIR-B data taking. This meant we had to replan for a possible new orbit. Fortunately we were prepared for this task and soon the SMDOS computers were displaying new radar footprints on the Earth.

A crucial link in the task of acquiring the data is the Tracking Data Relay System (TDRS) satellite, Figure 2. show how the shuttle orbiter transmitts the SIR-B data to the ground where it is recorded. The satellite is in geosynchronous orbit which means it is at an altitude such that its orbital speed exactly matches that of the Earth. Since the orbiter is in a lower orbit the satellite is constantly moving with respect to the shuttle and high bandwidth communication must be accomplished by tracking the satellite with an antenna (called the KU band antenna) mounted in the shuttle's cargo bay. Well into the mission this antenna failed, losing its ability to track the TDRS satellite.

A bolt had sheared in the antenna's pointing mechenism and the KU band antenna was trashing around, threatening to destroy itself. It was necessary for an astronaut to exit the shuttle (EVA) in a spacesuit to pin the antenna down.

A solution was found for this problem also. Onboard were tape recorders capable of recording short portions of our data. We could use these to store segments of our data, if a way could be found to transmitt the contents of the tapes to the ground. The KU band antenna could not track but it could transmitt. It was decided to use the shuttle itself , firing its small attitude thrusters to point the antenna at the moving target of the TDRS satellite. Of course in this mode, since we were not pointing at the Earth, we could not take data. We developed a scenario that eventually proved successful: Take data and record on tape. When the tape was full move the shuttle to track the TDRS and transmitt to ground. When the tape is empty move the shuttle so the SIR-B radar again points at the Earth and take more data.

Of course this meant an entirely new data-taking strategy. Again the SMDOS computers were put to work displaying new plans for the stringent new conditions.

The shuttle returned to Earth on October 12. We had managed to take about 20 percent of the data orginally planned.

## Conclusions

SMDOS represents a pioneering effort by NASA/JPL to extend the possibilites inherent in using a manned space platform for scientific experimentation. The field is still very new and ripe for methods and ideas that will fully exploit the existence of the space shuttle. Command and control of an instrument like SIR-B requires a new intergrated approach that combines, as SMDOS has done, predictive capabilities as well as the means to use telemetry to provide interactive visualzations of experiment status.

We have found in the implementation of SMDOS that the use of Forth has several advantages over other languages. The development time was generally halved in comparision to the normal accepted times for code development using FORTRAN. The extensible nature of the language allowed us to create, within the same run-time system, diverse, interacting modules. The nature of Forth allowed us to create formalized commands modeled on standards in graphics, array manipulation etc while still retaining the free access to the machine that Forth provides.

That is not to say Forth is not without its problems. Though by implementing inheritable data structures, and array manipulations we avoided a a lot of common Forth programming problems we still found heavy number crunching tasks difficult to program because of the the RPN notation. This was true even of programmers with a lot of experience with Forth. We believe that Forth was the correct choice for this project because we were willing to forgo the numerical programming ease of a number crunching language such as FORTRAN for the wide range of non-numerical tasks that were easily implemented in Forth. We found the speed of doing numerical work like orbit propogation in Forth quite acceptable due mainly to Forth extensions for the 8087 numerical co-processor provided by LMI. See [MAC84] for a good discussion of number crunching using the 8087.

The best example of the power of Forth was discussed above. When the antenna feed failed and we realized that the software had to *adapt* to that failure, it was relatively easy given the interactive Forth enviroment to change the required module to meet the new specifications. This is clearly beyond the capabilites of most languages. Indeed, most shuttle software is required to remain fixed not only during missions but for intervals typically six months before the launch as a matter of policy. Real-time programming would be unthinkable give the realities of most current software design. The success that we have seen in using SMDOS to handle adaptive space mission design stand as a testimony to the unique and powerful capabilities of Forth.

The work described in this paper was carried out by the Jet Propulsion Laboratory, California Institute of Technology under contract with the National Aeronautics and Space Administration.

## References

[FOL82] J. Foley and A. Van Dam, *Fundamentals of Interactive Computer Graphics*, Addison-Wesley, 1982.

[HAR84] H. Harris, "SMDOS: SIR-B Mission Design and Operations Software," Jet Propulsion Laboratory Document D-1081, 1984.

[BAS83] J. Basile, "Implementing Data structures in FORTH," Journal of Forth Applications & Research Vol. 1 No. 2, 1983.

[ELA82] C. Elachi, "Radar Images of the Earth from Space," Scientific American, December 1982.

[MAC84] F. MacIntyre. "Number Crunching with 8087 FQUANs: The Mie Equations," Journal of Forth Applications & Research Vol 2, Number 3,1984.

**Figure 1. Hardware configuration**

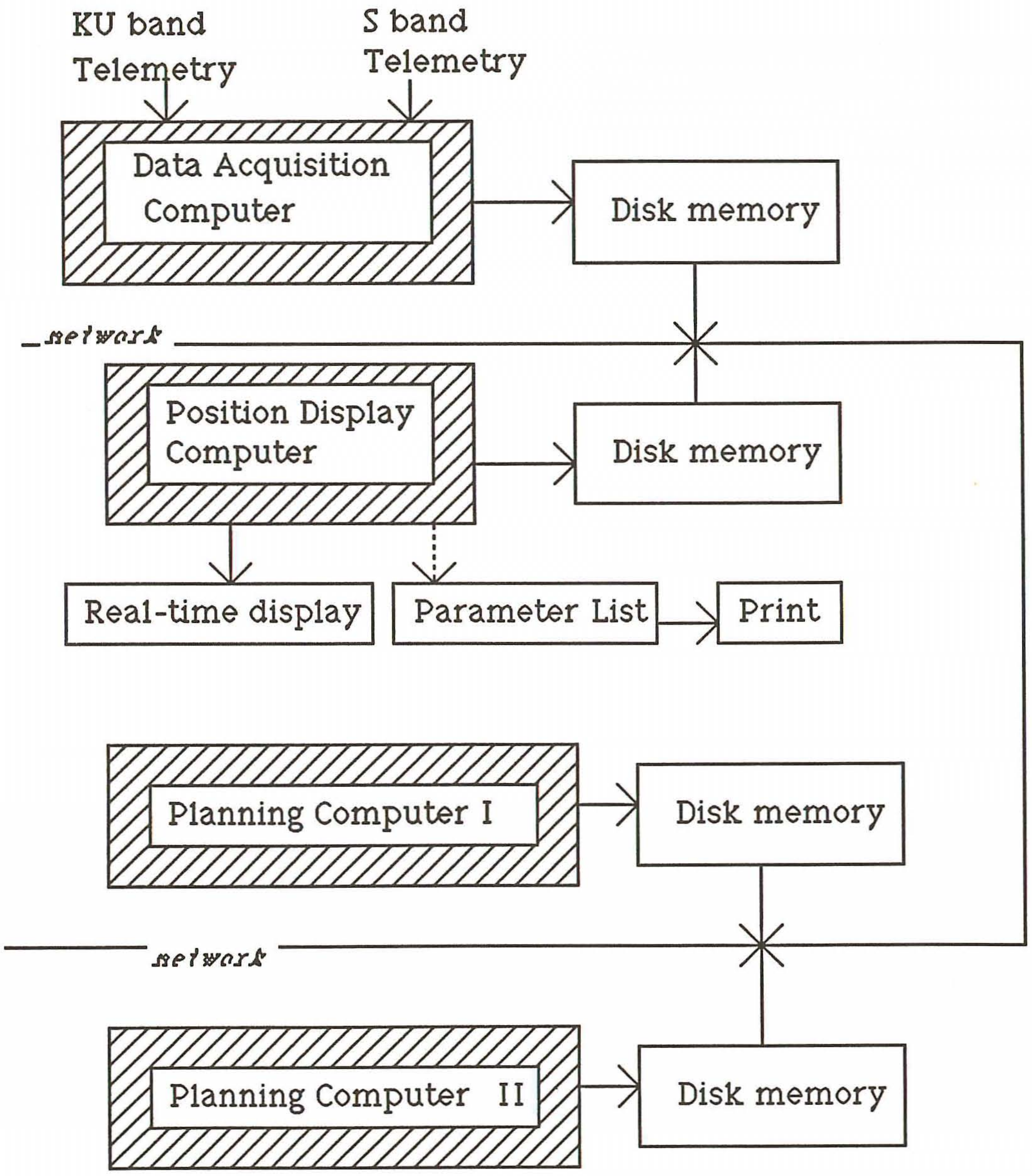
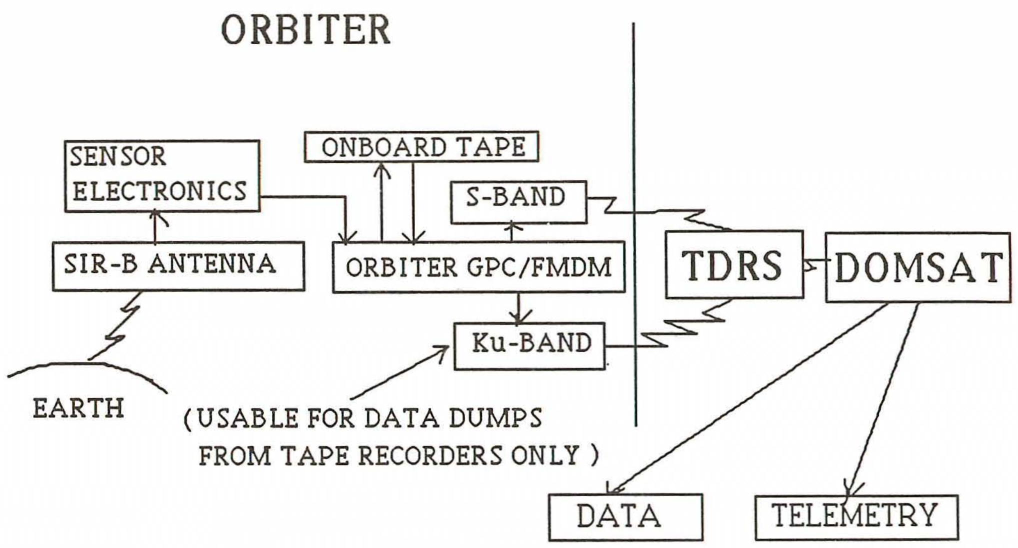


**Figure 2. SIR-B Data Flow**


ORBITER
SENSOR ELECTRONICS
ONBOARD TAPE
S-BAND
SIR-B ANTENNA
ORBITER GPC/FMDM
TDRS
DOMSAT
Ku-BAND
EARTH
(USABLE FOR DATA DUMPS FROM TAPE RECORDERS ONLY)
DATA
TELEMETRY