CHAPTER 20

SOFTWARE SYSTEM DEVELOPMENT

20.1 Safeguards Appropriate for Mission Support Software
20.2 Use of Graphic Support Systems
20.3 Utility Subroutines
   Vector and Matrix Algebra Routines, Time-Conversion Routines, Ephemeris Routines, Plotting Routines

In practice, much of the time devoted to preparation for mission support is spent in the development of computer software systems. Although some progress has been made in the standardization of software, the variations in attitude determination and control hardware, mission requirements, and processing sophistication have meant that most spacecraft series have required largely new attitude determination and control software systems. Therefore, questions of software structure and performance are central to the practical problems of mission support. This chapter describes the general principles for the development of attitude software and the use of executive support systems and utility subroutines.

20.1 Safeguards Appropriate for Mission Support Software

Myron A. Shear

Attitude determination requirements may be divided into three categories: real time, near real time, and definitive. A real-time requirement implies that attitude must be determined within seconds of the receipt of data and is usually associated with monitoring an attitude maneuver or attitude acquisition sequence. A near-real-time requirement implies that attitude must be determined within minutes or hours of the receipt of data, usually to compute control commands to achieve or maintain a desired attitude. A definitive requirement implies that an accurate attitude history is to be generated, perhaps weeks or months after the fact, generally for use in analysis of experimental results. The most critical demands on mission support software generally arise in real-time or near-real-time support, when results must be obtained shortly after the receipt of data. Failure to obtain accurate results within the prescribed time may jeopardize the success of the mission. Therefore, software intended for use in real-time or near-real-time support must be designed to meet particularly high standards of reliability, flexibility, and ease of operation. In some missions, even a minor software error could lead to total mission failure; furthermore, software must be capable of handling contingencies as well as nominal mission conditions. For example, if one attitude sensor fails, the software should still be capable of supporting the mission to the extent that the remaining attitude sensors permit. In real-time and near-real-time
support, there is no time to make software modifications either to correct errors or to add new capabilities. Even a minor modification to a large software system may require hours or days to implement and the system reliability would be in doubt until extensive testing had been performed. For these reasons, specific safeguards should be considered in the design, implementation, and testing of mission support software. This section describes some of the safeguards used in mission software developed for the Attitude Determination and Control Section of NASA's Goddard Space Flight Center.

The software environment for mission support programs at Goddard Space Flight Center is typically a multiprogrammed, large-scale computer with interactive graphics terminals, card readers, printers, and other peripheral devices. Mission support programs are assigned relatively high priority, and nonmission support programs are run only as resources permit. Most mission support software is designed for interactive graphics operation, primarily because of the greater flexibility provided by allowing an analyst to examine the input data and program results and change the processing options accordingly. Graphics operation, described further in Section 20.2, also allows rapid correction of user input errors. Nongraphic systems utilizing card input and printed output are normally limited to utility programs which do not directly process telemetry data and, therefore, require fewer processing options. The safeguards discussed in this section can be applied to either nongraphic or graphic systems.

**Error Checking.** If a program terminates unexpectedly and must be restarted, a period of 15 minutes or more may be required to resubmit the job, schedule the required resources, and initialize the program. A delay of this magnitude is unacceptable in real-time support, and very inconvenient in near-real-time support. For this reason, mission support software must be fully protected against failures due to user errors or unexpected telemetry data. An interactive graphics program must not be allowed to terminate abnormally except for the most severe error conditions; most common errors can be corrected by the user if the program provides appropriate error messages. Thus, mission support software must check for all foreseeable error conditions, provide standard corrective actions whenever possible, or provide an error message which is clear enough to allow the user to diagnose and correct the problem promptly. If further diagnosis is required, the user must be able to request intermediate displays to obtain additional information.

User input errors are almost inevitable; these may be simple typographical errors, or logical errors resulting from specifying an inconsistent set of input parameters. The program should check user input parameters for validity, especially in cases in which a user input error could lead to abnormal program termination. For example, a user error which leads to overfilling an array or infinite looping may result in a program termination which is difficult to diagnose and relate to the original error.

Potential mathematical singularities should also be checked to avoid errors such as division by zero, square root of a negative argument, or inverse trigonometric functions of invalid arguments. These singularities may result from user input errors, invalid telemetry data, or spacecraft hardware noise. Most operating systems provide features to intercept such errors, apply standard fixups, print warning messages, and/or terminate the program. However, these operating system features (such as the FORTRAN monitor) are generally inadequate for an interac-