AD NUMBER

ADB020488

LIMITATION CHANGES

TO:

Approved for public release; distribution is unlimited.

FROM:

Distribution authorized to U.S. Gov't. agencies only; Proprietary Information; NOV 1976. Other requests shall be referred to Air Force Armament Laboratory, ATTN: DLMM, Eglin AFB, FL 32542.

AUTHORITY

ADTC ltr dtd 23 Oct 1979

(2)

AFATL-TR-76-132, VOLUME II

DIGITAL GUIDED WEAPONS TECHNOLOGY
VOLUME II: SYSTEM SIMULATIONS

MISSILE SYSTEMS DIVISION
HUGHES AIRCRAFT COMPANY
CANOGA PARK, CALIFORNIA 91304

NOVEMBER 1976

FINAL REPORT: AUGUST 1974-NOVEMBER 1976

AIR FORCE ARMAMENT LABORATORY
AIR FORCE SYSTEMS COMMAND • UNITED STATES AIR FORCE
EGLIN AIR FORCE BASE, FLORIDA

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

REPORT DOCUMENTATION PAGE	READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER: AFATL-TR-76-132 Vol II-2	2. GOVT ACCESSION NO. / 3. RECIPIENT'S CATALOG NUMBER
4. TITLE *(and Subtitle)*: DIGITAL GUIDED WEAPONS TECHNOLOGY. VOLUME II. SYSTEM SIMULATIONS.	5. TYPE OF REPORT & PERIOD COVERED: Final Report, August 1974 to November 1976
	6. PERFORMING ORG. REPORT NUMBER: HAC-Reference No. D3003
7. AUTHOR(s): R. Woolley, D. Sobek, R. Hoolko, F. Watson, J. Mouton	8. CONTRACT OR GRANT NUMBER(s): F08635-75-C-0014
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Hughes Aircraft Company, Missile Division, Canoga Park, California 91304	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: Project No. 670B, Task No. 01, Work Unit No. 014
11. CONTROLLING OFFICE NAME AND ADDRESS: Air Force Armament Laboratory, Armament Development and Test Center, Eglin Air Force Base, Florida 32542	12. REPORT DATE: November 1976
	13. NUMBER OF PAGES: 156
14. MONITORING AGENCY NAME & ADDRESS *(if different from Controlling Office)*: 162p.	15. SECURITY CLASS. *(of this report)*: Unclassified
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT *(of this Report)*

Distribution limited to U.S. Government agencies only; this report documents test and evaluation; distribution limitation applied November 1976. Other requests for this document must be referred to the Air Force Armament Laboratory (DLMM), Eglin Air Force Base, Florida 32542.

17. DISTRIBUTION STATEMENT *(of the abstract entered in Block 20, if different from Report)*

16 670B

17 01

18. SUPPLEMENTARY NOTES

Available in DDC.

19. KEY WORDS *(Continue on reverse side if necessary and identify by block number)*

Digital Processors	Aided Inertial Midcourse Guidance
System Component Integration	Long-Range Air-to-Surface Weapon
Advanced Modular Weapon System	Hybrid Computer Simulation
Performance Evaluation	LORAN Sensor
Navigation Accuracy	Software Documentation

20. ABSTRACT *(Continue on reverse side if necessary and identify by block number)*

This report describes simulations and field tests performed with the two digital processing systems, DP1 and DP2, which were built on the Digital Guided Weapons Technology Program. The DP1 system was integrated with a GFE Inertial Measurement Unit (IMU); and, after testing at Hughes Aircraft Company, it was integrated with the Completely Integrated Reference Instrumentation System (CIRIS) at the Central Inertial Guidance Test Facility (CIGTF), Holloman Air Force Base, New Mexico. Perfor-

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

403 848

20. ABSTRACT (CONCLUDED)

mance tests were carried out by the Air Force in the laboratory, in a van, and in an aircraft. Performance in navigation accuracy relative to CIRIS was determined. The DP2 system was integrated into a semiphysical hybrid computer simulation at the contractor facility. The simulation performed was that of a long-range air-to-surface weapon which uses an aided inertial guidance system for midcourse guidance. A simulated LORAN sensor was used to update the inertial navigation subsystem. The DP2 software included Management and Executive Software and the following computational elements: LORAN processing, alignment filter midcourse guidance law, navigation, and flight control.

PREFACE

This final report was prepared by the Missile Division, Hughes Aircraft Company, Canoga Park, California 91304, under Contract No. F08635-75-C-0014 with the Air Force Armament Laboratory, Armament Development and Test Center, Eglin Air Force Base, Florida. Major Robert L. Haney (DLMM) monitored the program for the Armament Laboratory. This effort was conducted during the period from August 1974 to November 1976.

This report consists of three volumes. Volume I contains Digital Processor System Studies. Volume II is concerned with System Simulations. Volume III deals with a Programmable Digital Autopilot. This is Volume II.

This technical report has been reviewed and is approved for publication.

FOR THE COMMANDER:

CLIFFORD H. ALLEN, JR., Colonel, USAF
Chief, Guided Weapons Division

(The reverse of this page is blank.)

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF FIGURES (Continued)

LIST OF FIGURES (Concluded)

LIST OF TABLES

(The reverse of this page is blank)

SECTION I

INTRODUCTION

The Digital Guided Weapon Technology (DGWT) program was initiated with the general intent of determining the role of digital processing techniques in guided weapons. Recognizing that too broad a scope can be self-defeating for this kind of program, the Air Force set two specific goals to be accomplished. The first was a near term application of digital processing to an existing weapon family. Specifically, a digital autopilot for the GBU-15 weapon was to be designed and evaluated. The evaluation required fabrication of a brassboard digital processor for the autopilot and verification of its performance with a hybrid simulation. That part of the program was completed in 1975. A separate program was initiated to bring the GBU-15 digital autopilot into engineering development based on the results. The Programmable Digital Autopilot (PDAP) work is reported in Volume III of this final report.

The second goal was to determine how digital processing techniques could be used to assist in integrating the components of an advanced modular weapon system. Earlier studies sponsored by the Air Force had investigated the characteristics of a modular weapon. The results of one of these studies, as expressed in AFATL-TR-72-202, were used as a starting point to define the weapon system to be studied in the DGWT program. Specifically, the program was to accomplish the following objectives:

1. Determine which functions could be done digitally in the weapon system.
2. Determine what the digital processing system should do to assist in integrating the weapon components.
3. Determine which other functions should be done in the digital processor.
4. Define the interfaces of the digital processing subsystem within the weapon system.
5. Determine the requirements of the digital processing system.
6. Produce a preliminary design of the digital processing system.
7. Build two breadboard processing systems.
8. Evaluate the breadboard systems in hybrid simulation and other tests.

All of these objectives were accomplished. The work performed and results achieved relative to the first six objectives are reported in Volume I, Digital Processor System Studies. That volume describes the digital processing system recommended to perform the integration function in an advanced modular weapon. The work done relative to the seventh and eighth objectives is reported here in Volume II.

To fulfill the seventh objective, two breadboarded digital processing systems, DP1 and DP2, were constructed. The first of these, the DP1 system was integrated with an inertial measurement unit (IMU) furnished by the Air Force, and after tests at Hughes Aircraft Company, was shipped to the Central Inertial Guidance Test Facility (CIGTF) at Holloman Air Force Base in New Mexico. At CIGTF, the DP1 system was integrated with CIRIS (Completely Integrated Reference Instrumentation System) and then underwent performance tests under the direction of the Air Force. In addition to laboratory tests, the system was tested in a van, on surface roads, and also in an aircraft.

The DP2 system was integrated into a six-degree-of-freedom (6DOF) hybrid computer simulation of the GBU-15 cruciform wing weapon in the Hughes Hybrid Computer Laboratory at Canoga Park. The simulation was performed to investigate the use of aided inertial guidance for midcourse.

Detailed information about these two breadboarded systems, DP1 and DP2, and about the related software developed for the simulations and tests performed is contained in the following documents and reports:

1. Operating Manual and System Description for Digital Processor Number 1, Report No. DGWT 0210-1, Revision A.

2. Operating Manual and System Description for Digital Processor Number 2, Report No. DGWT 0210-2.

3. Programmer's Manual for PDAP, DP1 and DP2, Report No. DGWT 0170-2.

4. Digital Processor System Specification, Report No. DGWT 0240-1.

5. Digital Processor Software Development Report, Report No. DGWT 0165-1.

This Volume II provides a brief description of the system hardware, the documentation for the system execitive software, and a detailed description of the actual tests and simulations involving the two systems, DP1 and DP2.

SECTION II

DIGITAL PROCESSING SYSTEM

The basic digital processing (DP) system is composed of a digital processor, a weapon bus and the executive software. In an operating system, there will be other subsystems which must be interfaced with the DP. The interface is accomplished by means of a Bus Interface Unit (BIU), which presents a common interface to the bus, and the executive software. In practice, most existing subsystems are not directly compatible with the common BIU interface, therefore, additional interfacing equipment is required. This was true for all the subsystems used in the CIGTF operation and the simulations.

The executive software is common to all configurations. It acts as the software interface between the external world and the DP system. In addition, it interfaces between the various applications software modules which the system requires. When applications software modules (e.g., inertial navigation) are added to the system, it is also necessary to add special interfacing software modules to allow the executive to do its job. These special modules are part of the System Management function.

DIGITAL PROCESSING SYSTEM HARDWARE

The basic DP hardware is shown in Figure 1. Figure 1 illustrates how external subsystems can be integrated into the system. The Control Panel, used in system development, allows monitoring and control of system functions. It would not be present in a tactical weapon installation. The digital processor (DP) is a rack-mounted breadboard unit. The BIU's are separate units which terminate sections of the weapon bus. The processor in the DP2 system differs from the DP1 processor in having an enlarged instruction set and in having the Master Bus Control in the DP rather than in the BIU box as in DP1. Further details of the hardware are contained in Sections III and IV. For a complete description, the reader is referred to the DP1 and DP2 System Description Manuals, Reports DGWT 0210-1 and DGWT 0210-2, respectively.

EXECUTIVE

The executive is an organized collection of software routines (shown in Figure 2) designed to manage the processor resources. It is responsible for interfacing with all interrupt hardware and input/output equipment and for supervising the execution of all software modules. The executive acts as a buffer between software modules and the hardware by performing all input/output operations, thereby making software modules independent of the mechanization of input/output hardware. It also acts as a buffer between software modules, providing a common way of interfacing one module to another. The isolation provided by the executive reduces the likelihood that changes made in one software module will propagate into other modules. It also makes it a relatively easy matter to add more functions to the system.

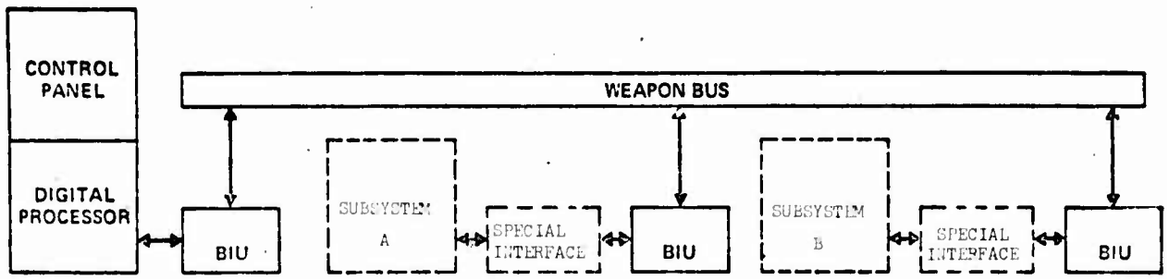


Figure 1. DP System Block Diagram

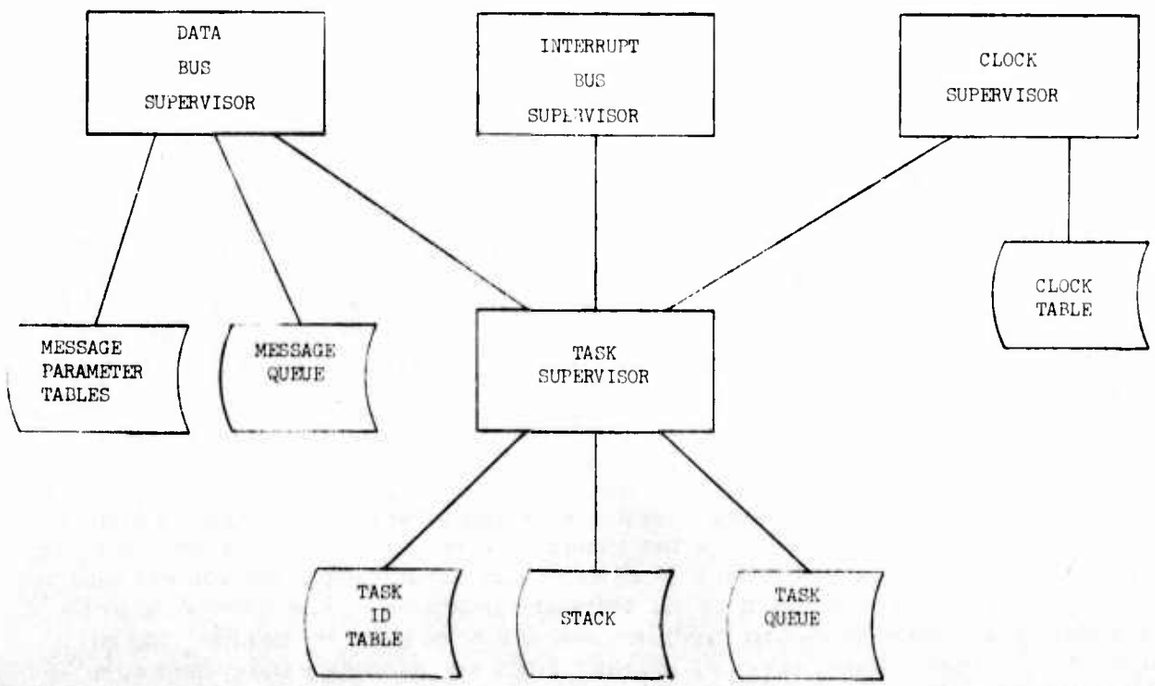


Figure 2. Executive Routines and Data Bases

The executive is a permanent part of the system that is common to all configurations.

The processor functions are partitioned into units called tasks. Each task is a unit of work that is to be performed as a result of some external event or time event or command from another task. Tasks can range in size from a few instructions to a few hundred instructions, and in execution time from microseconds to seconds. Some tasks may only be executed once while others may be executed from 100 to 400 times per second. They are characterized by a unique identification number, a starting address, and a priority. Tasks are called by each other or associated with external events only by their ID number. Only the executive needs to know the location and priority of a task, and since this information is contained in a table, it can vary from one configuration to another without the tasks having to be changed.

In determining how to partition a processor function into tasks, some general rules should be followed:

1. Each task should consist of a single operation at some level of abstraction in the description of the function. At a lower level of abstraction the task might include many operations but there should be some level at which the programmer can consider the task as s single operation. Otherwise, it should be broken down into smaller tasks.

2. A task should not have delay loops longer than 30 microseconds in it that wait for some external event to be completed. If a task must initiate an external event, such as the transmission of a block of data from a weapon subsystem to the DP, and has no other operations to perform until the data block has been received by the DP, then it should be broken into two tasks. The first task would initiate the external event and then end by returning to the executive. The second task would execute when the data transfer is complete. During the time in between, the executive can start executing some other task so the time is not wasted.

3. A task should not have a small subset of operations that have a much higher priority than the bulk of the task. Instead it should be partitioned into two tasks which have different priority levels. One task containing the bulk of the operations would run at a low priority, and the remaining few operations would be performed in a separate task at a higher priority.

All these tasks must be tied together and executed in a sequence appropriate to the weapon configuration. This will be accomplished by two levels of software. One of these levels is the executive which causes software modules to be executed in response to signals from external devices or in response to commands from other software modules. The executive does this without any knowledge of the weapon configuration or what each module is supposed to do. This knowledge is contained in the system management

module which is the second level of software that controls the sequence of execution of all tasks. The system management function is described in software subsections of Sections III and IV.

The remainder of the EXECUTIVE subsection discusses the various components of the executive.

Task Supervisor

The purpose of the task supervisor is to supervise the execution of all tasks in the DP assuring that high priority tasks take precedence over low priority tasks. The task supervisor consists of the task queueing routine TASKQU, the task dispatching routine DISPAT, the queue searching routine SEARCH, the ID table IDTAB, and the task queue TQ. The hierarchy of these routines is shown in Figure 3.

TASKQU. The purpose of the task queueing routine shown in Figure 4 is to place the address and priority of a task in the TASK QUEUE.

It can be called by interrupt routines or by any task. When called, TASKQU looks up the address and priority of the task in the TASK ID TABLE and places them in the TASK QUEUE which is an unordered list. It also compares the priority of the task being placed in the queue with the CURRENT TASK PRIORITY. If the priority of the new task is higher it sets a flag called HIGH PRIORITY TASK WAITING. It then returns to the program that called. The calling program may look at the HIGH PRIORITY TASK WAITING flag to decide whether to go to the task DISPATCHER immediately or not. The flowchart for TASKQU is shown in Figure 5.

DISPAT. The purpose of the task dispatching routine shown in Figure 6 is to search the TASK QUEUE for the highest priority task, and if this task has a higher priority than the CURRENT TASK PRIORITY, then the task should be removed from the queue and executed.

DISPAT includes the routine SEARCH.

DISPAT can be called by any program that queues up a task although it is not necessary. All tasks are called by the dispatcher so all tasks return to the dispatcher by executing an RTN when they are finished.

When it is called DISPAT first clears the HIGH PRIORITY TASK WAITING flag if it is set and saves processor status. It then searches the TASK QUEUE for the highest priority task. If the highest priority task in the queue is higher in priority than the CURRENT TASK PRIORITY, then the dispatcher removes that task from the queue and calls it. When the task is finished, it executes a return instruction returning it to the dispatcher. The dispatcher then searches the TASK QUEUE again for the highest priority task. When the highest priority task in the queue has lower priority than the CURRENT TASK PRIORITY, the dispatcher restores the processor status and executes a return instruction. The flowchart for DISPAT is shown in Figure 7.

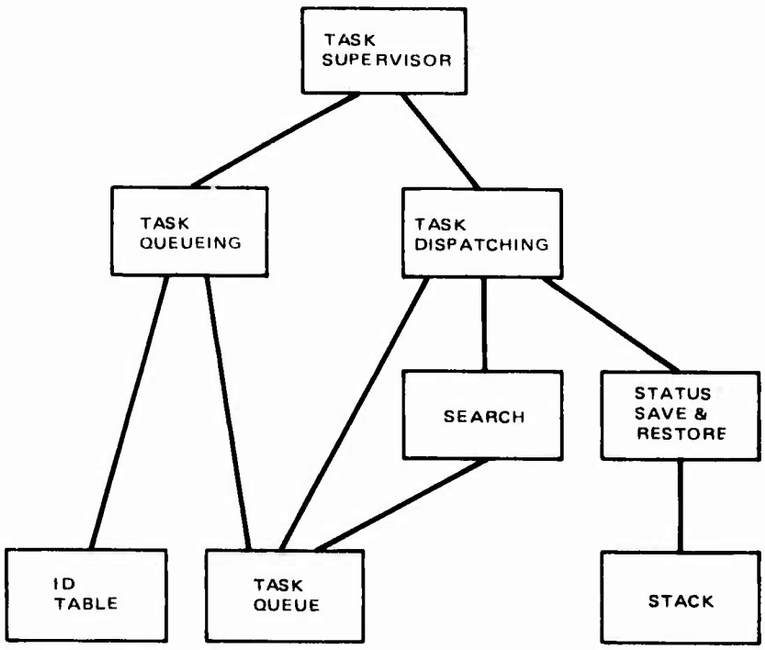


Figure 3. Task Supervisor Resource Hierarchy

```
/          TASK QUEUEING ROUTINE
/          ENTER WITH TASK ID IN L2
TASKQU     NOP
           LD,1,TP101          ;GET TEST POINT COUNTER
           AD1,1,K1            ;INCREMENT COUNT
           WR1,,TP101          ;WRITE BACK IN MEMORY
           DAT1,64              ;ID TABLE SIZE
           JIFFS,L1<=L2,TASKEN;IF ID NOT IN TABLE EXIT
           DAT1,0              ;LOCATION 0 IN TABLE NOT USED
           JIFFS,L1>=L2,TASKEN;IF ID NOT IN TABLE EXIT
           LD,3,K15            ;MASK ALL INTERRUPTS
           STL2,27             ;MOVE TASK ID INTO I27
           LD2I,27,IDTAB       ;LOAD TASK ADDRESS,PRIORITY INTO L2
           STM,27,TQ           ;MOVE POINTER TO END OF TQ INTO I27
           DCI,27,-1           ;INCREMENT POINTER
           WR2I,27,TQ          ;STORE ADDRESS,PRIORITY IN QUEUE
           RIRM,27,TQ          ;STORE POINTER
           LD,3,INTPRI         ;UNMASK INTERRUPTS
           AND2I,30,H000F      ;EXTRACT PRIORITY
           LD1I,29,0           ;COMPARE TO CURRENT PRIORITY
           JIFFS,L1>=L2,TASKEN;JUMP IF CURRENT PRIORITY>=NEW PRIORITY
           WRF,45,1            ;SET FLAG INDICATING A HIGHER PRIORITY
/                              ;TASK IS IN THE QUEUE
TASKEN     RTN                 ;END OF TASK QUEUER
```

Figure 4. Task Queueing Routine (TASKQU)

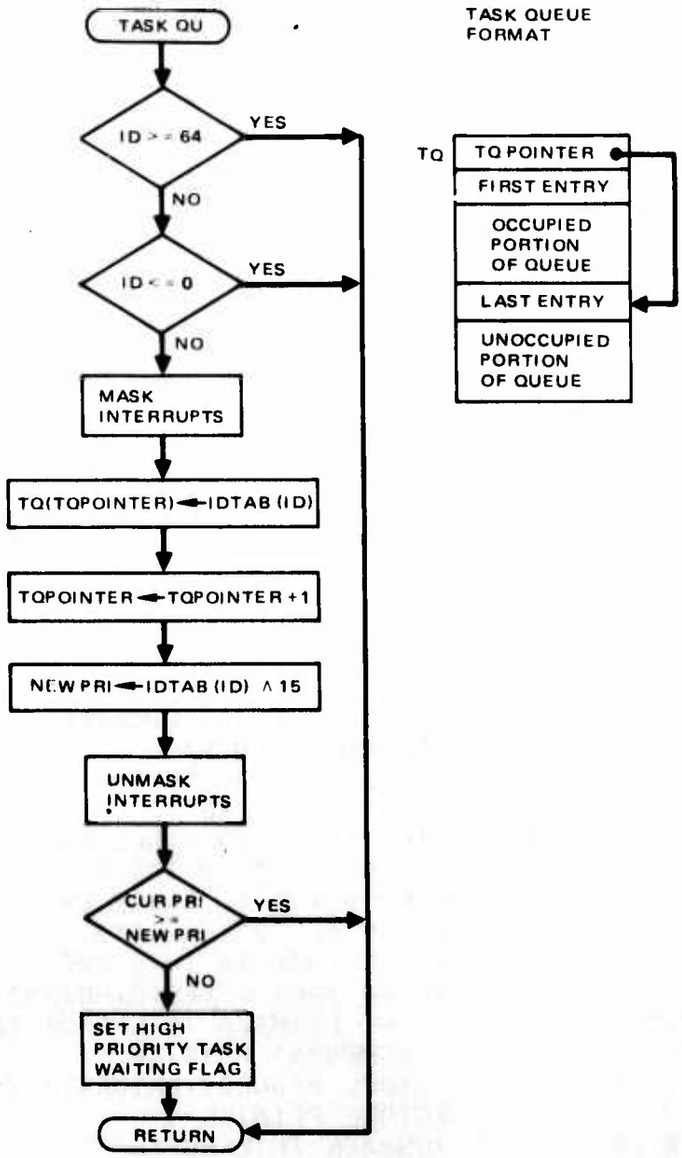


Figure 5. TASKQU Flowchart

```
/
/           TASK DISPATCHING ROUTINE
DISPAT      WRF,45,0              ;CLEAR 'HIGH PRIORITY TASK WAITING' FLAG
            LD,3,K15              ;MASK INTERRUPTS
            GSB,,SAVE             ;SAVE PROCESSOR STATUS
DISSEA      GSB,,SEARCH           ;SEARCH TQ FOR HIGHEST PRIORITY TASK
            LD,1,TP102           ;GET TEST POINT COUNTER
            AD1,1,K1             ;INCREMENT COUNT
            WR1,,TP102           ;WRITE BACK IN MEMORY
            LD1I,29,0             ;LOAD CURRENT PTIORITY
            JIFFS,L1<L2,DISTSK    ;COMPARE CURRENT PRIORITY WITH PRIORITY
/                                 ;OF TASK IN QUEUE
            GSB,,RESTOR           ;RESTORE PROCESSOR STATUS
            LD,3,INTPRI           ;ALL TASKS IN QUEUE HAVE LOWER PRIORITY
            RTN                   ;UNMASK INTERRUPTS AND RETURN
            NOP
DISTSK      DCI,29,1              ;PUT PRIORITY OF NEW TASK ON PRI STACK
            WR2I,29,0             ;PUSHING DOWN THE PREVIOUS PRIORITY
            LD2I,1,TQ             ;GET ADDRESS OF TASK
            SHR2,,4               ;SHIFT OFF THE PRIORITY
            WR2,2,JMPLNK          ;STORE IN JMPLNK
            STM,0,TQ              ;GET POINTER TO LAST ENTRY IN QUEUE
            LD2I,0,TQ             ;GET LAST ENTRY
            WR2I,1,TQ             ;STORE IN SPOT JUST VACATED
            DCI,0,1               ;DECRIMENT POINTER
            RIRM,0,TQ             ;AND STORE
            LD,3,INTPRI           ;ENABLE INTERRUPTS
            GSBOM,,JMPLNK         ;CALL TASK
            LD,3,K15              ;DISABLE INTERRUPTS
            DCI,29,-1             ;POP PRIORITY STACK
            JT,,DISSEA
/
```

Figure 6. Task Dispatching Routine (DISPAT)

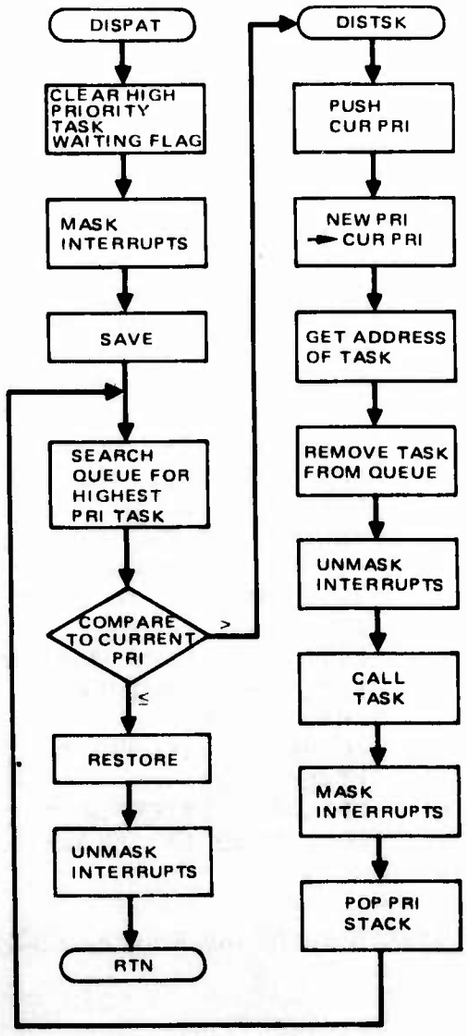


Figure 7. DISPAT Flowchart

SEARCH. The purpose of the SEARCH routine shown in Figure 8 is to search the task queue for the highest priority task.

SEARCH is only called by DISPAT. When called, it searches the task queue for the highest priority task. It returns the highest priority in the queue in latch 2 and the location in the queue of the highest priority task in index register 1. If the queue was empty, it returns a zero in latch 2. The flowchart for SEARCH is in Figure 9.

SAVE and RESTORE. The purpose of the routines SAVE and RESTORE (see Figures 10 and 11, respectively) is to save the status of the processor when a task is going to be interrupted by another task and to restore the status of the processor when the interrupted task is to be resumed.

SAVE pushes the contents of the two latches, the carry bit, the shift counter, memory location JMPLNK, and index registers 0 through 5 onto the stack. RSAVE is used when only the latches and carry bit need to be saved.

RESTORE restores the contents of the two latches, the carry bit, the shift counter, memory location JMPLNK, and index registers 0 through 5 from the stack. RREST restores only the two latches and the carry bit.

Task ID Table. The TASK ID TABLE shown in Figure 12 stores the starting address and priority of each of the tasks in the DP. The routine IDTABL, also shown in Figure 12, puts entries in the table.

THE STACK. The STACK is a push down (last in first out) stack implemented in operand memory with index register 31 used as the stack pointer. The stack expands downward in memory toward location 0. The stack pointer normally points to the last occupied position in the stack. To push an item onto the stack the stack pointer should be decremented by one and then the item should be stored in the stack at the location indicated by the stack pointer. When popping an item off the stack the item should first be read from the stack, then the stack pointer should be incremented.

Data Bus Supervisor

The purpose of the data bus supervisor is to maintain a queue of messages to be transmitted on the data bus, and to schedule the messages according to priority.

The data bus supervisor is called by tasks when data is to be sent on the data bus. If the bus is available, the message is sent immediately. Otherwise, the message is queued up to be transmitted when the data bus becomes available. The data bus supervisor includes a routine to put messages in a queue (MQINSE), a routine to remove messages from the queue (MQREMO), a driver for the data bus (DBDRV), an interrupt routine to handle the normal completion of a message (DBCOMP), and an interrupt routine to handle error conditions (DBERR). Each block of data to be transmitted on the data bus is called a message and must have a Message Parameter Table associated with it that contains descriptions of the source and

```
/
/            SEARCH TQ FOR HIGHEST PRIORITY TASK
SEARCH     NOP
           STM,0,TQ              #PUT POINTER TO END OF QUEUE IN I0
           JIFIZ,0,SEAEX         #IF QUEUE IS EMPTY EXIT
           LD2I,0,TQ             #LOAD ENTRY FROM QUEUE INTO L2
           AND2I,30,H000F        #EXTRACT PRIORITY AND PLACE IN L1
           WR2,1,0
           JT,,SEASAV            #JUMP
SEALOO     LD2I,0,TQ             #LOAD ENTRY FROM QUEUE INTO L2
           AND2I,30,H000F        #EXTRACT PRIORITY
           JIFFS,L1>L2,SEALOW    #JUMP IF PRIORITY L1 > PRIORITY L2
           WR2,1,0               #MOVE PRIORITY INTO L1
SEASAV     RIR2,0                #MOVE POINTER INTO L2
           STL2,1                #THEN INTO I1
SEALOW     IND,0,SEALOO          #DECRIMENT POINTER AND LOOP IF NOT 0
           WR1,2,0               #MOVE PRIORITY TO L2
           RTN
SEAEX      DAT2,0                #SAY PRIORITY IS 0
           RTN
/          EXIT WITH PRIORITY IN L2 AND LOCATION OF TASK IN I1
/
```

Figure 8. Search TQ for Highest Priority Task Routine (SEARCH)

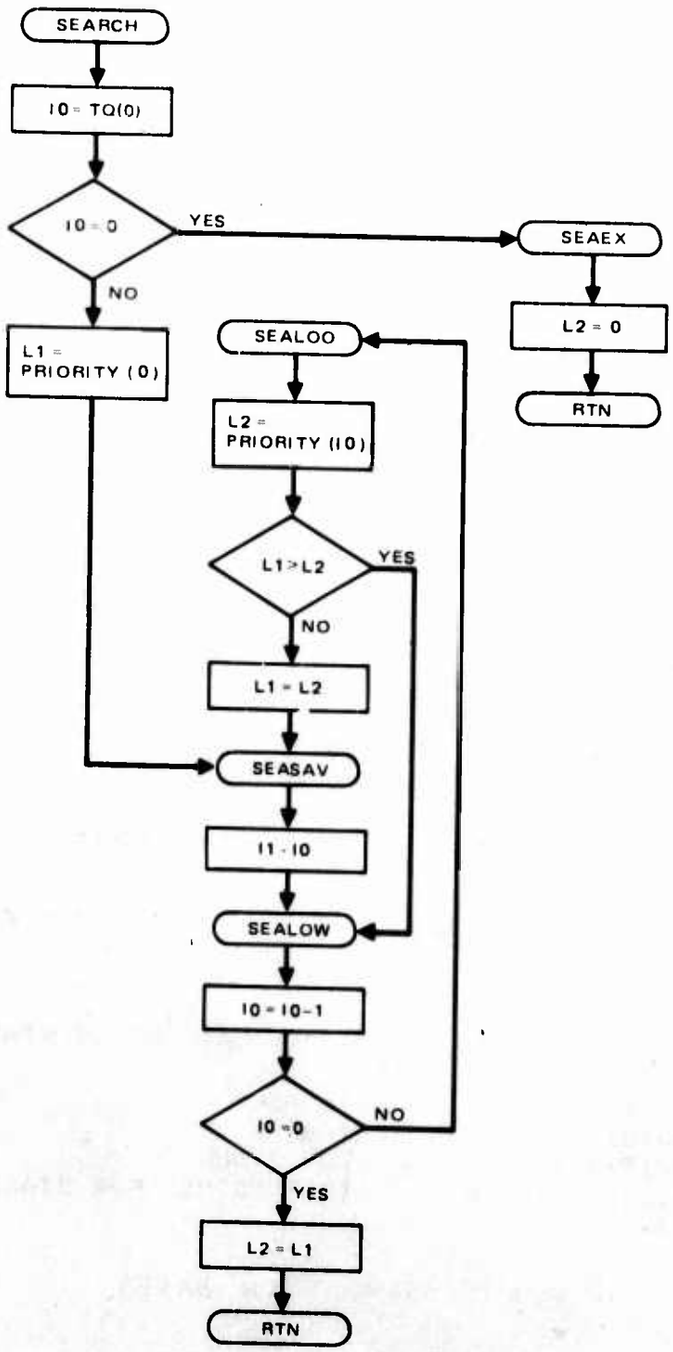


Figure 9. SEARCH Flowchart

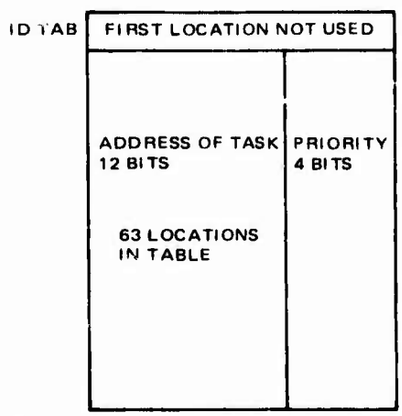
```
/           SAVE PROCESSOR STATUS
SAVE        DCI,31,12           ;MAKE ROOM ON STACK
            WR1I,31,0           ;L1
            WR2I,31,1           ;L2
            DAT1,0              ;CARRY
            AD1C,1,K0
            WR1I,31,2
            RDSC,2,0            ;SHIFT COUNTER
            WR2I,31,3
            LD,2,JMPLNK         ;JUMP LINK
            WR2I,31,5
            RIR2,0              ;I0
            WR2I,31,6
            RIR2,1              ;I1
            WR2I,31,7
            RIR2,2              ;I2
            WR2I,31,8
            RIR2,3              ;I3
            WR2I,31,9
            RIR2,4              ;I4
            WR2I,31,10
            RIR2,5              ;I5
            WR2I,31,11
            NOP
            RIRM,31,TPSTK       ;TEST POINT  FOR STACK POINTER
            RTN


/
RSAVE       DCI,31,3
            WR1I,31,0           ;SAVE LATCHES ON STACK
            WR2I,31,1
            DAT1,0              ;SAVE
            AD1C,1,K0           ;CARRY
            WR1I,31,2           ;ON STACK
            RIRM,31,TPSTK       ;TEST POINT  FOR STACK POINTER
            RTN
```

Figure 10. Save Routine (SAVE)

```
/
/           RESTORE PROCESSOR STATUS
RESTOR      LD2I,31,11          ;I5
            STL2,5
            LD2I,31,10          ;I4
            STL2,4
            LD2I,31,9           ;I3
            STL2,3
            LD2I,31,8           ;I2
            STL2,2
            LD2I,31,7           ;I1
            STL2,1
            LD2I,31,6           ;I0
            STL2,0
            LD2I,31,5           ;JUMP LINK
            WR2,,JMPLNK
            DAT1,-1             ;SHIFT COUNTER
            DAT2,-1
            SHLDMI,31,3
            SCALE,,31
            LD2I,31,2           ;CARRY
            AD2,2,M1
            LD2I,31,1           ;L2
            LD1I,31,0           ;L1
            DCI,31,-12          ;CLEAR STACK
            NOP
            RTN
/
RREST       LD2I,31,2           ;RESTORE CARRY
            AD2,2,M1            ;FROM STACK
            LD2I,31,1
            LD1I,31,0           ;RESTOR  LATCHES FROM STACK
            DCI,31,-3
            RTN
```

Figure 11. Restore Routine (RESTOR)




```
/          MAKE AN ENTRY IN THE ID TABLE
/          ENTER WITH ID IN I1, ADDRESS IN L1, PRIORITY IN L2
IDTABL     SHL1M,,K4
           ADDI,1,IDTAB
           RTN
```

Figure 12. Task ID Table (IDTABL)

destination of the message, the number of words in the message, the priority of the message, and the ID number of a task to be queued up when the message has been transmitted. The data bus supervisor is shown in Figure 13.

DBSUPR. To send a block of data on the weapon bus, a task puts the address of a message parameter table in index register 0 and calls the data bus supervisor (DBSUPR). DBSUPR (shown in Figure 14) checks to see if the bus is busy. If the bus is not busy, DBSUPR calls the data bus driver (DBDRV) to send the data. If the bus is busy, DBSUPR calls the message queue insert routine (MQINSE) to put the message in the message queue for later transmission. The flowchart of DBSUPR is shown in Figure 15.

DBDRV. The data bus driver is the interface between the rest of the software and the weapon bus. Figure 16 shows the data bus driver routine. It takes parameters from the message parameter table, transforms them if necessary, and writes them in the I/O registers for the weapon bus. After the parameter describing the source, destination, and size of the data block have been given to the bus master, DBDRV sends the Start Data Transfer signal and returns to the calling routine. While the weapon bus is transmitting the block of data, the DP is free to resume executing tasks. The flowchart of DBDRV is shown in Figure 17.

MQINSE. The message queue insert routine shown in Figure 18 places the address index register 0 in the message queue. The address in index register 0 is assumed to be the address of a message parameter table. The flowchart for MQINSE is also shown in Figure 18.

MQREMO. The message queue remove routine shown in Figure 19 searches the queue looking for the highest priority message in the queue. It does this by looking at the priority word in each message parameter table listed in the queue. If there are several messages at the same priority level, the one closest to the bottom of the queue (oldest message) will be chosen. The selected message is removed from the queue and the address of the message parameter table is placed in index register 23. If there are no messages in the queue, a 0 is placed in index register 23. The flowchart for MQREMO is shown in Figure 20.

DBCOMP. The data bus complete routine shown in Figure 21 is an interrupt service routine that is executed when the bus master determines that the message has been transmitted correctly. DBCOMP clears a status bit in the message parameter table of the message just completed. This feature is useful only if the message parameter table is in read/write memory. DBCOMP also checks the status word of the message parameter table to determine which task, if any, is to be executed as a result of the completion of the message transmission. If the task ID in the status word is non zero, that task is queued up by calling the task queueing routine (TASKQU). DBCOMP then calls the message queue, remove routine (MQREMO) which returns the address of the highest priority message in the message queue. If there was any message in the queue the data bus driver (DBDRV) is called to transmit the message. DBCOMP checks the "HIGH PRIORITY TASK WAITING" flag to determine whether to jump to the dispatcher

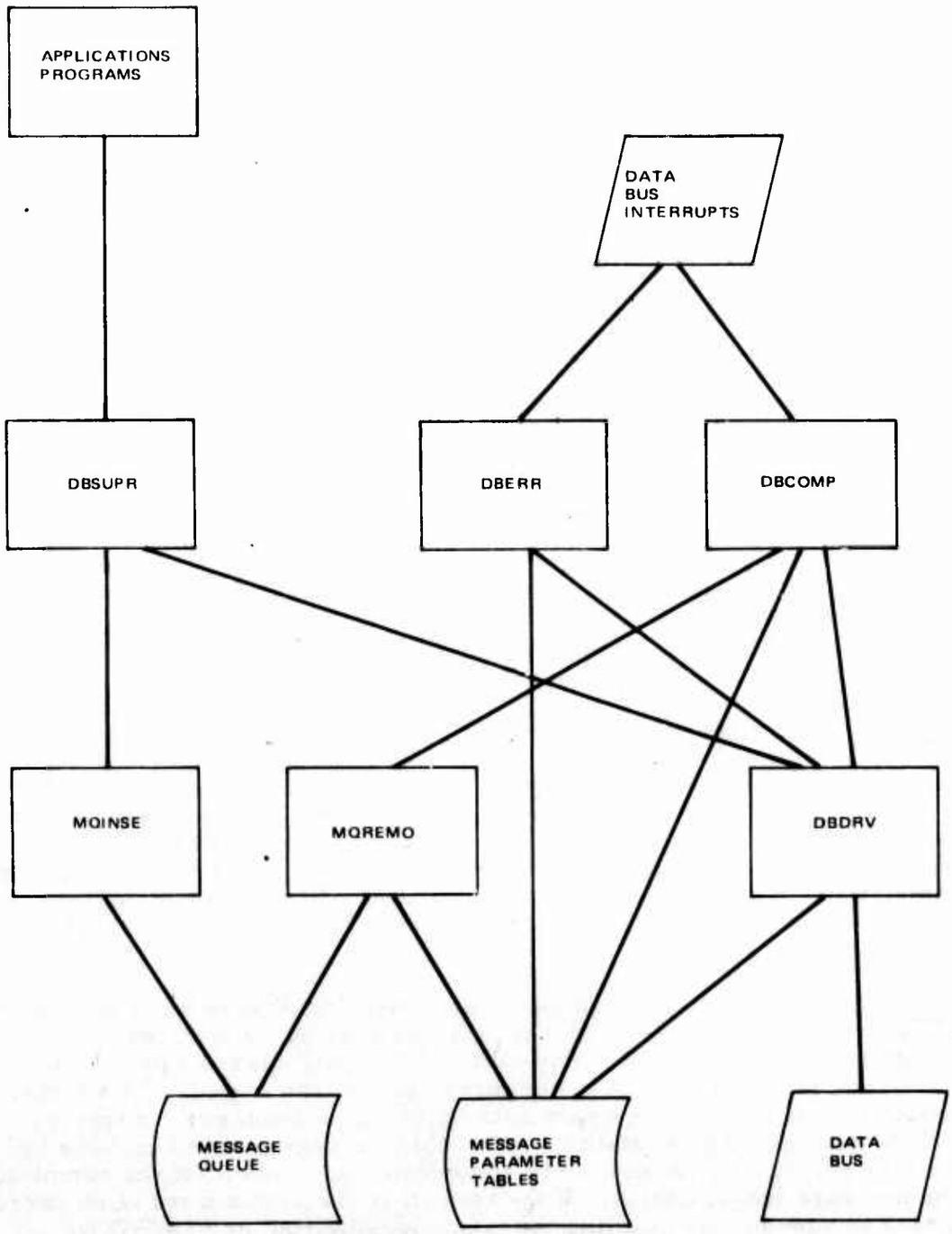


Figure 13. Data Base Supervisor Resources

```
/
/          DATA BUS SUPERVISOR
/          ENTER WITH ADDRESS OF MPT IN IO
/          I23 IS THE ADDRESS OF THE ACTIVE MPT
DBSUPR     LD,3,K15              ;MASK INTERRUPTS WHILE IN DBSUPR
           LD,1,TP103            ;GET TEST POINT COUNTER
           AD1,1,K1              ;INCREMENT COUNT
           WR1,,TP103            ;WRITE BACK IN MEMORY
           LD1I,0 ,MSTAT         ;SET STATUS OF MPT
           AD1,1,MBUSY           ;TO INDICATE THAT MESSAGE
           WR1I,0 ,MSTAT         ;HAS NOT BEEN SENT YET
           JIFIZ,23,DBNBUS       ;IF NO ACTIVE MESSAGE JUMP TO NOT BUSY
           LD,1,MPRILI           ;IF BUSY CHECK PRIORITY LIMIT
           LD2I,0 ,MPRI          ;AGAINST PRIORITY OF NEW MESSAGE
           JIFFS,L1<L2,DBBUSY    ;IF NEW MESS NOT IMPORTANT ENOUGH JUMP
           LD1I,23,MDWCNT        ;MESS IS IMPORTANT SO GET NUMBER OF WORDS
           SUB1,1,DBRCNT         ;IN ACTIVE MESSAGE AND SUBTRACT NUMBER
           JIFFS,L1<L2,DBBUSY    ;ALREADY SENT. THEN COMPARE TO NEW MESS PRI
           WRF,10,1              ;IF NEW MESS IS MORE IMPORTANT STOP ACTIVE
           RIRM,0,DBTEM1         ;MESSAGE AND MAKE NEW MESSAGE ACTIVE
           RIRM,23,DBTEM2        ;BY SWAPPING OLD MESSAGE
           STM,23,DBTEM1         ;AND NEW MESSAGE
           STM,0,DBTEM2          ;IE. SWAP IO,I23
           GSB,,MQINSE           ;INSERT OLD MESSAGE IN MQUE
           GSB,,DBDRV            ;CALL DRIVER
           JT,,DBDONE            ;DONE
DBNBUS     RIRM,0,DBTEM1         ;MAKE NEW MESSAGE ACTIVE BY
           STM,23,DBTEM1         ;PUTTING ADDRESS OF MPT IN I23
           GSB,,DBDRV            ;CALL DRIVER
           JT,,DBDONE            ;DONE
DBBUSY     GSB,,MQINSE           ;PUT NEW MESSAGE IN MQUE
DBDONE     LD,3,INTPRI           ;ALLOW INTERRUPTS
           NOP
           RTN
```

Figure 14. Data Base Supervisor Routine (DBSUPR)

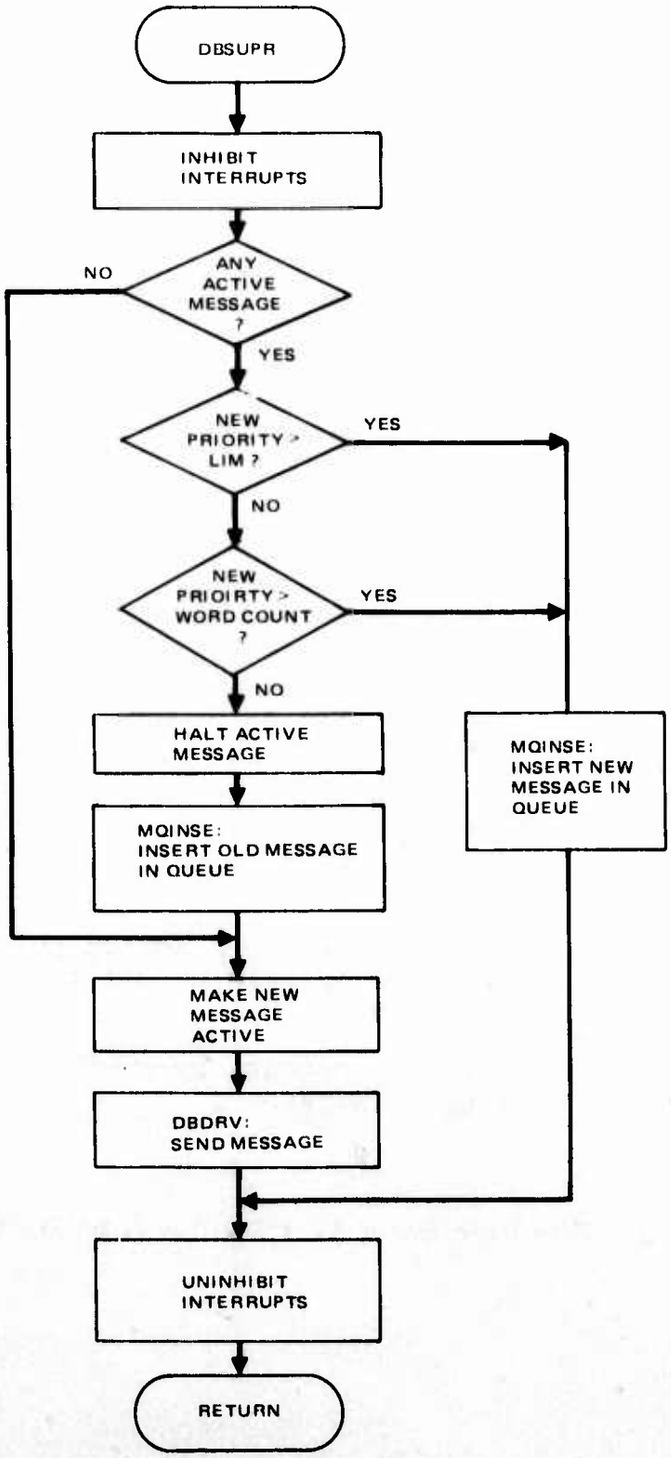


Figure 15. DBSUPR Flowchart

```
/           DATA BUS DRIVER
/           ENTER WITH ADDRESS OF MPT IN I23
DBDRV       DAT1,18              ;COMMAND WORD SIZE
            WR1,1,DBCSIZ         ;SEND TO BIU
            LD1I,23,MDSTID       ;GET DEST ID AND OPCODE
            WR1,1,DBID           ;SEND TO BIU
            WRF,8,1              ;CLEAR BIU DONE FLAG
            LD1I,23,MDSTAD       ;GET DEST ADDRESS
            WR1,1,DBADR          ;SEND TO BIU
            LD1I,23,MSRCID       ;GET SOURCE ID AND OPCODE
            LD2I,23,MSRCAD       ;GET SOURCE ADDRESS
            ST,24,50             ;LIMIT THE AMOUNT OF TIME WE WILL WAIT
DBDWA1      JIFFS,8,DBDON1       ;IS DATA BUS DONE
            IND,24,DBDWA1        ;NO.  WAIT AWHILE
            JT,,DBDRV            ;TOO LONG - ERROR
DBDON1      ST,24,31             ;WAIT FOR THE SIGNAL THAT SETS
DBDWAF      IND,24,DBDWAF        ;THE FLAG TO GO AWAY
            WRF,8,1              ;CLEAR THE FLAG
            WR1,1,DBID           ;SEND ID TO BIU
            WR2,2,DBADR          ;SEND ADDRESS TO BIU
            LD2I,23,MDWCNT       ;GET WORD COUNT
            SHL2,,5              ;SHIFT WORD COUNT INTO PROPPER FIELD
            WR2,1,DBDCNT         ;SEND TO BIU
            DAT1,19              ;DATA WORD SIZE
           WR1,1,DBDSIZ          ;SEND TO BIU
            ST,24,50             ;LIMIT THE AMOUNT OF TIME WE WILL WAIT
DBDWA2      JIFFS,8,DBDON2       ;IS DATA BUS DONE
            IND,24,DBDWA2        ;NO WAIT AWHILE
            JT,,DBDRV            ;TOO LONG - ERROR
DBDON2      ST,24,31             ;DELAY 1 MICROSEC
DBDWA3      IND,24,DBDWA3        ;WAIT
            WRF,9,1              ;START MESSAGE
            LD,1,TP105          ;GET TEST POINT COUNTER
            AD1,1,K1            ;INCREMENT COUNT
            WR1,,TP105          ;WRITE BACK IN MEMORY
            NOP
            RTN
```

Figure 16. Data Bus Driver Routine (DBDRV)

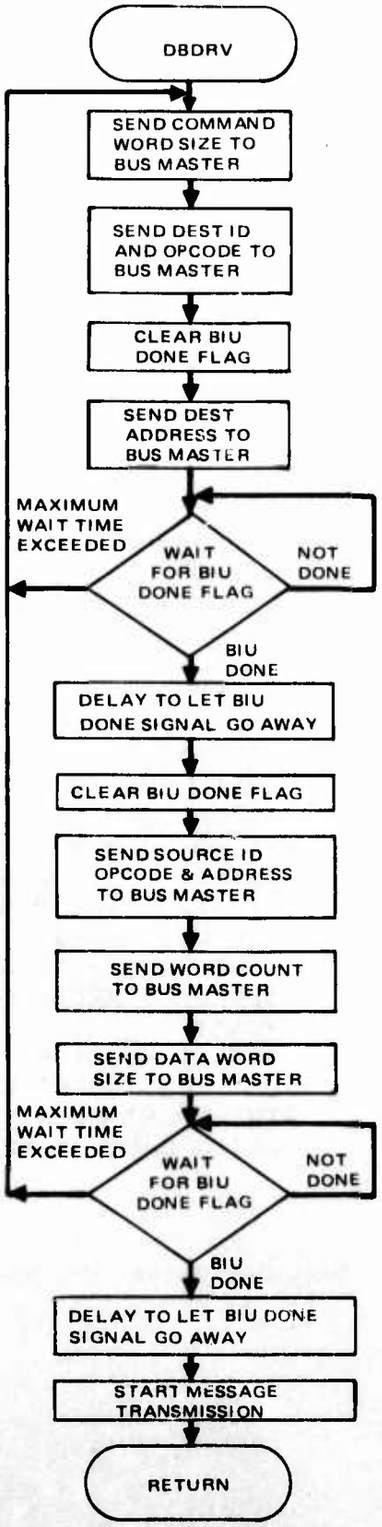


Figure 17. DBDRV Flowchart

```
/          MESSAGE QUEUE INSERT ROUTINE
/          ENTER WITH ADDRESS OF MPT IN I0
MQINSE     STM,21,MQUE          ;PUT MQUE  BOUNDRY IN I21
           DCI,21,-1            ;INCREMENT MQUE BOUNDRY
           RIR2,0               ;PUT MESS PARM TABLE ADDR IN L2
           WR2I,21,MQUE         ;NOW PUT IT IN MQUE
           RIRM,21,MQUE         ;AND SAVE BOUNDRY
           NOP
           RTN
```

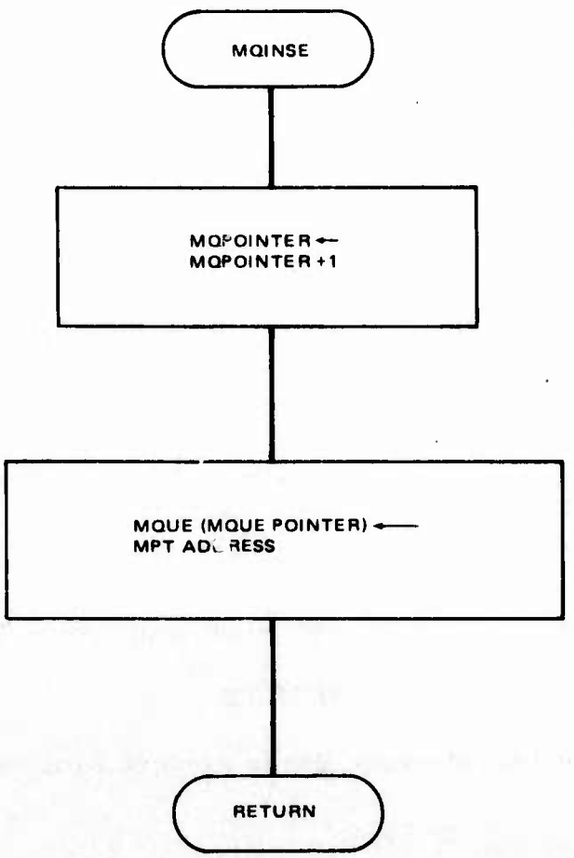


Figure 18. Message Queue Insert Routine and Flowchart (MQINSE)

```
/          MESSAGE QUEUE REMOVE ROUTINE
/          EXIT WITH ADDRESS OF MFT IN I23
MQREMO     STM,24,MQUE          ;GET POINTER TO END OF MQUE
           JIFIZ,24,MQREX2      ;IF MQUE EMPTY EXIT
           RIR2,24              ;COPY POINTER TO L2
           LD1I,24,MQUE         ;GET ADDRESS FROM END OF MQUE
           WR1,1,IT             ;AND TRANSFER IT TO
           STM,23,IT            ;INDEX REG 23
           DCI,24,1             ;DECRIMENT POINTER
           RIRM,24,MQUE         ;STORE POINTER INDICATING THAT ONE ENTRY
           JIFIZ,24,MQREX       ;IF ONLY ONE ENTRY, THIS MUST BE IT.
           DCI,31,2             ;OTHERWISE, MAKE ROOM ON STACK
           WR2I,31,0            ;SAVE POINTER IN TEMP0
           WR1I,31,1            ;SAVE ADDRESS IN TEMP1
           LD1I,23,MPRI         ;GET MESS PRIORITY
MQRLOO     LD2I,24,MQUE         ;GET NEXT ADDRESS
           STL2,23              ;PUT ADDRESS IN INDEX REG 23
           LD2I,23,MPRI         ;GET PRIORITY
           JIFFS,L1<L2,MQRLOW   ;COMPARE PRIORITIES LOOKING FOR HIGHEST
           WR2,1,0              ;PRIORITY(I24) IS HIGHER MOVE L2 TO L1
           RIR2,24              ;SAVE POINTER TO HIGHEST PRIORITY
           WR2I,31,0            ;IN TEMP0
MQRLOW     IND,24,MQRLOO        ;DECRIMENT POINTER&LOOP TILL END OF MQUE
MQREX1     LD1I,31,1            ;GET ADDRESS IN TEMP1, WAS AT END OF MQUE
           LD2I,31,0            ;GET POINTER IN TEMP0 TO HIGHEST PRIOR
           STL2,24              ;PUT POINTER IN INDEX REG 24
           LD2I,24,MQUE         ;GET ADDRESS TO BE REMOVED
           STL2,23              ;PUT ADDRESS IN INDEX REG 23
           WR1I,24,MQUE         ;PUT SAVED ADDRESS IN SLOT JUST VACATED
           DCI,31,-2            ;CLEAR STACK
           NOP
MQREX      RTN
MQREX2     ST,23,0              ;IF MQUE WAS EMPTY PUT 0 IN I23
           NOP
           RTN                  ;RETURN
```

Figure 19. Message Queue Remove Routine (MQREMO)

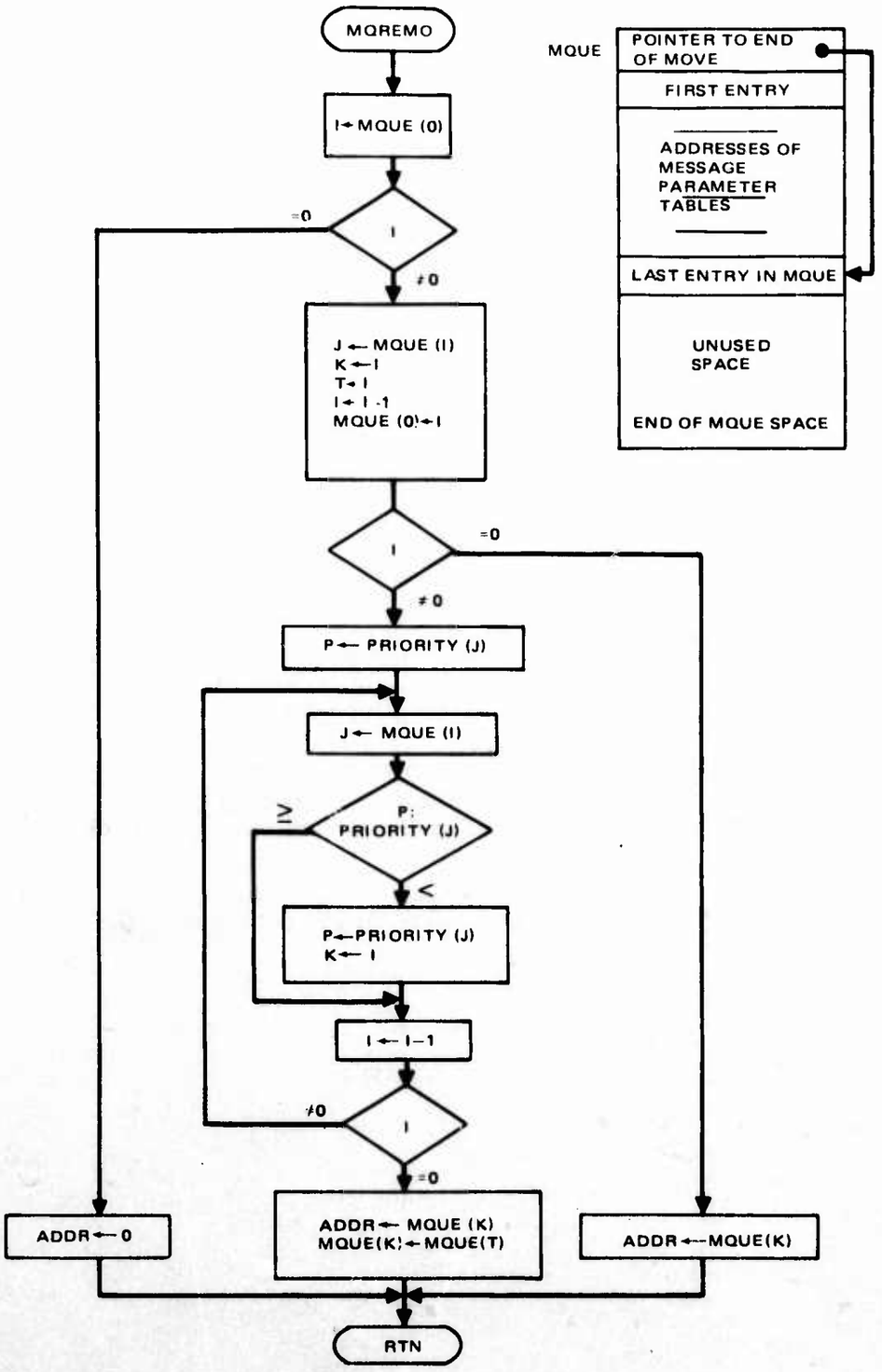


Figure 20. MQREMO Flowchart

```
/
/          DATA BUS COMPLETE INTERRUPT ROUTINE
           NOP
DBCOMP     GSB,,RSAVE           ;SAVE CONTENTS OF LATCHES
           LD,1,TP104          ;GET TEST POINT COUNTER
           AD1,1,K1            ;INCREMENT COUNT
           WR1,,TP104          ;WRITE BACK IN MEMORY
           LD2I,23,MSTAT        ;SET STATUS OF ACTIVE MESSAGE
           AND2I,30,H7FFF       ;TO SHOW MESSAGE COMPLETE
           WR2I,23,MSTAT        ;THEN LOOK FOR COMPLETION TASK
           DAT1,0               ;LOAD 0 FOR COMPARISON
           JIFFS,L1=L2,DBCNTA  ;IF NONE JUMP
           GSB,,TASKQU          ;IF YES QUEUE THE TASK
DBCNTA     GSB,,MQREMO          ;ANY MESSAGES IN QUEUE
           JIFIZ,23,DBCMTY     ;IF MQUE WAS EMPTY DO NOT CALL DRIVER
           NOP
           GSB,,DBDRV           ;CALL DRIVER
DBCMTY     NOP
           GSB,,RREST           ;RESTOR  LATCHES
           GSB,,RSTINT           ;RESET INTERRUPT
           JIFFS,45,DISPAT     ;IF ANY HIGHER TASK THEN DO IT
           RTN
```

Figure 21. Data Bus Complete Interrupt Routine (DBCOMP)

(DISPAT) or to just return to the interrupted program. The flowchart for DBCOMP is shown in Figure 22.

DBERR. The data bus error routine shown in Figure 23 is an interrupt service routine. The interrupt occurs when the bus master detects that an error has occurred in data being transmitted on the bus. DBERR transfers data from the active message parameter table to an empty message parameter table. The source and destination addresses and the word count are adjusted to account for the number of words already sent, and the data bus driver (DBDRV) is called to transmit the remainder of the message. Whether the word in which the error occurred is retransmitted is determined by the sign bit of the word count. If the sign bit is one, the word in which the error occurred is retransmitted. The flowchart for DBERR is shown in Figure 24.

MQUE. The message queue shown in Figure 25 is a data structure which is an unordered list of message parameter table addresses. The pointer contains the number of entries in the queue and therefore points to the last entry in the queue.

Message Parameter Table. The message parameter tables are tables (one for each message as shown in Figure 26) describing the block of data to be transmitted on the bus. Each one describes the source, destination, size, and priority of a message, and also indicates which task is to be executed when the message has been transmitted.

Interrupt Bus Supervisor

The purpose of the interrupt bus supervisor shown in Figure 27 is to supervise the transmission and reception of interrupt messages on the interrupt bus. The interrupt bus supervisor includes the interrupt bus interrupt service routine (IBUSIN) shown in Figure 28.

To send an interrupt message from the DP to an external device, the interrupt bus supervisor (IBUSSU) is called with the interrupt message in Latch 2. IBUSSU will send the message on the interrupt bus immediately if the bus is not busy. If the bus is busy, IBUSSU will wait long enough for the bus to transmit one message and then try again. The flowchart for IBUSSU is shown in Figure 29.

Interrupt Bus Service Routine. The interrupt bus service routine (IBUSIN) shown in Figure 28 is executed each time an interrupt message is received by the DP. IBUSIN gets the interrupt from the bus interface unit and converts the message into a task ID. It then calls the task queueing routine which queues up the appropriate task. IBUSIN then checks the HIGH PRIORITY TASK WAITING flag, and if it is set, jumps to the dispatcher which will execute the task just queued up. If the flag is not set, IBUSIN returns to the task that was interrupted. The flowchart for IBUSIN is shown in Figure 30.

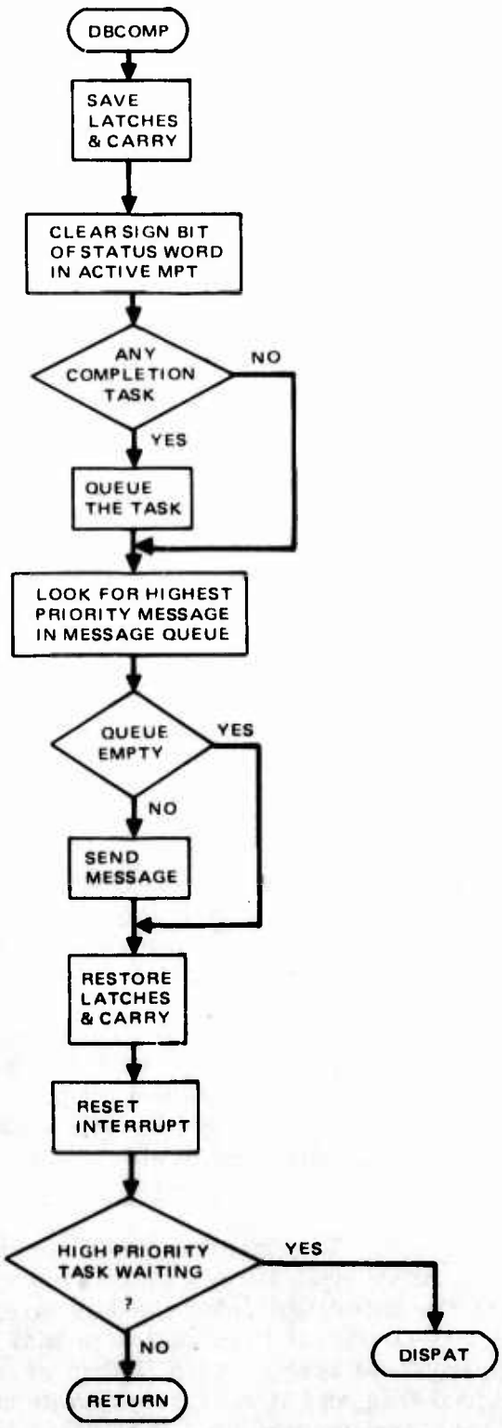


Figure 22. DBCOMP Flowchart

```
/
DBERR     NOP                 ;DATA BUS ERROR ROUTINE
          GSB,,RSAVE          ;SAVE LATCHES
          ST,24,DBECNT        ;COUNT DATA BUS ERRORS
          GSB,,INCM           ;INCREMENT COUNTER
          ST,24,MPTE          ;GET ADDERSS OF ERROR MPT
          LD1I,23,MDSTID      ;TRANSFER
          WR1I,24,MDSTID      ;DESTINATION ID
          LD1I,23,MSRCID
          WR1I,24,MSRCID      ;SOURCE ID
          LD1I,23,MPRI
          WR1I,24,MPRI        ;PRIORITY
          LD1I,23,MSTAT
          WR1I,24,MSTAT       ;STATUS
          LD,2,DBRCNT         ;GET NUMBER OF WORDS ALREADY SENT
          AD2,2,M1            ;ADD -1 BECAUSE LAST WORD SENT WAS BAD
          LD1I,23,MDWCNT      ;NUMBER OF WORDS TO BE SENT
          JIFFS,L1<0,DBERX    ;IF SIGN BIT IS ZERO
          AD2,2,K1            ;THEN ADD 1 TO THE WORDS SENT
DBERX     SUBI,24,MDWCNT      ;SUB NO SENT FROM NO TO BE SENT, STORE
          LD1I,23,MDSTAD      ;GET DEST ADDRESS
           ADDI,24,MDSTAD      ;ADD NO SENT AND STORE
          LD1I,23,MSRCAD      ;GET SOURCE ADDRESS
          ADDI,24,MSRCAD      ;ADD NO SENT AND STORE
          ST,23,MPTE          ;GET ADDRESS OF ERROR MPT
          GSB,,DBDRV          ;AND CALL DATA BUS SUPERVISOR
          GSB,,RREST          ;RESTORE LATCHES
          RTNINT              ;END DATA BUS ERROR ROUTINE


/
INCM      LD1I,24,0            ;GET WORD FROM MEM
          AD1,1,K1            ;ADD ONE
          WR1I,24,0            ;PUT WORD BACK IN MEM
          RTN
```

Figure 23. Data Bus Error Routine (DBERR)

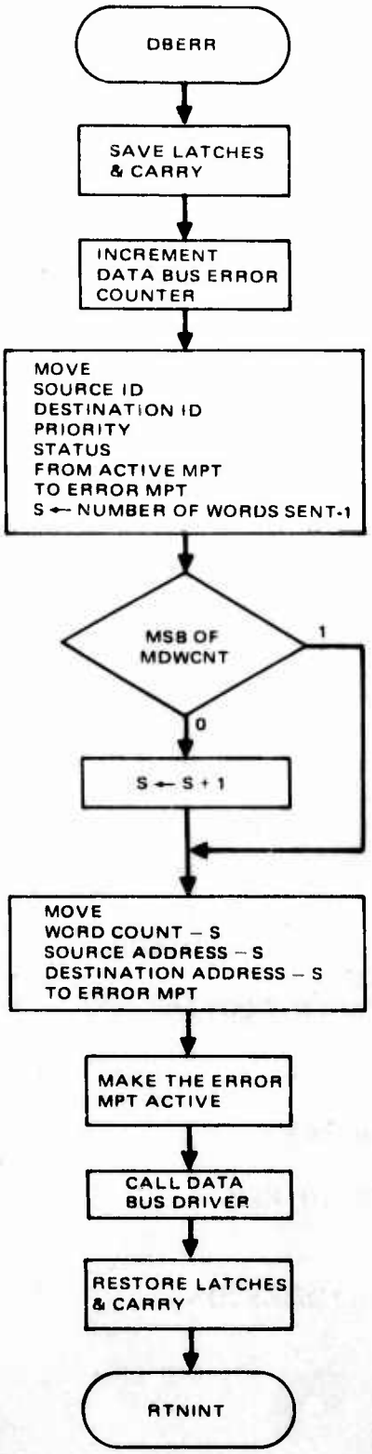


Figure 24. DBERR Flowchart

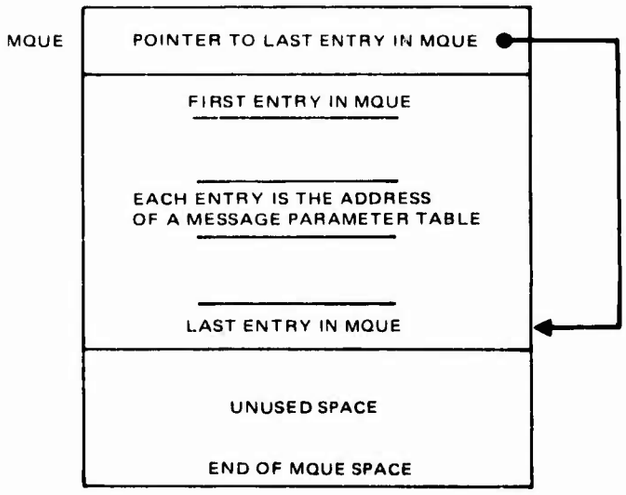


Figure 25. Message Queue Structure

MDSTID		DESTINATION ID – OPCODE
MSRCID		SOURCE ID – OPCODE
MPSTAD		DESTINATION ADDRESS
MSRCAD		SOURCE ADDRESS
MDWCNT	*	DATA WORD COUNT
MPRI		MESSAGE PRIORITY
MSTAT	#	COMPLETION TASK ID

*MSB CONTROLS ERROR RECOVERY ROUTINE. IF BIT IS SET, RETRANSMIT WORD THAT WAS IN ERROR. IF BIT NOT SET IGNORE BAD WORD AND JUST RETRANSMIT REST OF MESSAGE.

#STATUS BIT – INDICATES THAT MESSAGE IS NOT COMPLETE YET. CAN ONLY BE USED WHEN MPT IS IN READ/WRITE MEMORY.

Figure 26. Message Parameter Table

```
/
/          INTERRUPT BUS SUPERVISOR
/          ENTER WITH INTERRUPT ID IN L2
IBUSSU     NOP
           LD,1,TP106          ;GET TEST POINT COUNTER
           AD1,1,K1            ;INCREMENT COUNT
           WR1,,TP106          ;WRITE BACK IN MEMORY
           JIFFS,9,IBUSNF       ;IS BUFFER FULL
           DAT1,-32             ;YES  WAIT
IBUSWA     AD1,1,K1             ;DELAY
           JIFFS,L1<0,IBUSWA    ;LOOP
           JIFFS,9,IBUSNF       ;IS BUFFER FULL
           NOP                  ;YES  SOMETHING IS WRONG
           NOP
           RTN
IBUSNF     WR2,2,IBUSOU         ;BUFFER NOT FULL. OUTPUT INTERRUPT ID
           RTN
```

Figure 27. Interrupt Bus Supervisor (IBUSSU)

```
/
/          INTERRUPT BUS INTERRUPT ROUTINE
IBUSIN     LD,3,K15             ;MASK INTERRUPTS
           JIFFS,31,IBURTN      ;IF FLAG 31 RETURN WITH INTERRUPTS MASKED
           GSB,,RSAVE           ;SAVE LATCHES
           LD,1,TP107          ;GET TEST POINT COUNTER
           AD1,1,K1            ;INCREMENT COUNT
           WR1,,TP107          ;WRITE BACK IN MEMORY
IBUSID     LD,2,INTID           ;GET INTERRUPT ID FROM BIU BUFFER
           AND2I,30,IDMSK       ;MASK OFF GARBAGE BITS
           WR2,0 ,0             ;DISPLAY ID
           GSB,,RSTINT          ;RESET INTERRUPT
           GSB,,TASKQU          ;CALL TASK QUEUER
           GSB,,RREST           ;RESTOR  LATCHES
           NOP
           JIFFS,45,DISPAT      ;IF ANY HIGHER TASK THEN DO IT
           RTN
IBURTN     RTNINT               ;RESET INTERRUPT, RETURN
```

Figure 28. Interrupt Bus Interrupt Routine IBUSIN)

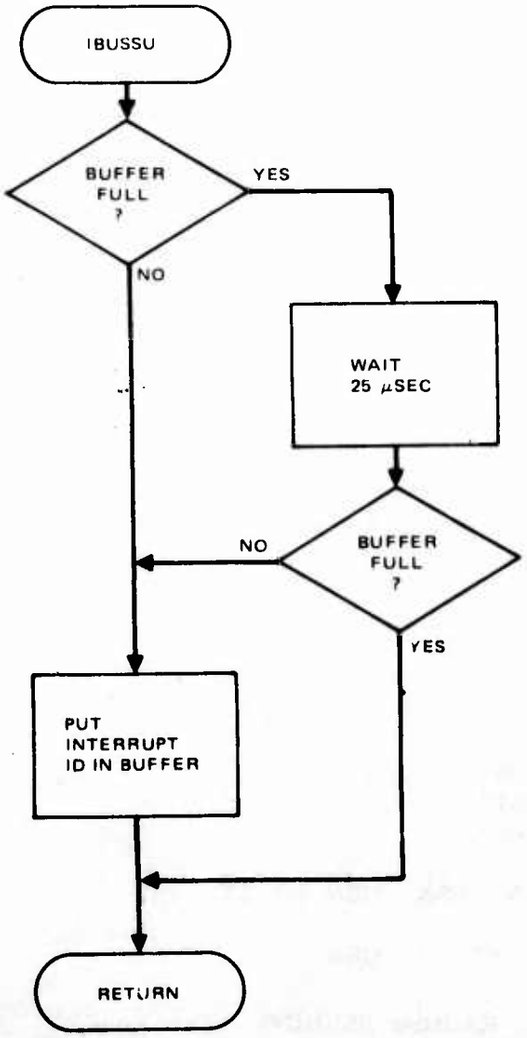


Figure 29. IBUSSU Flowchart

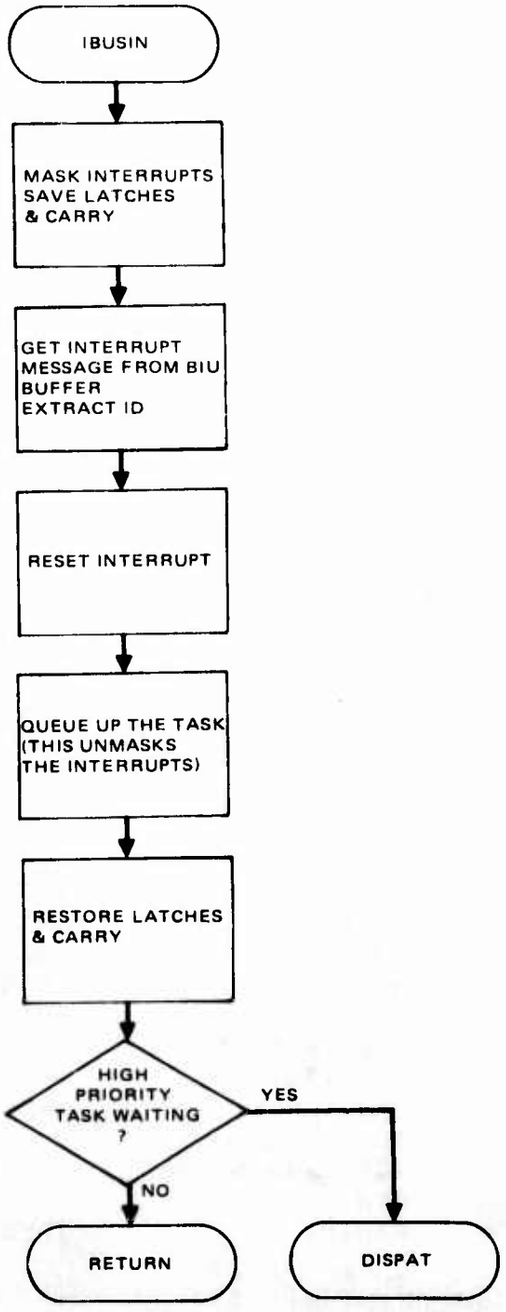


Figure 30. IBUSIN Flowchart

Clock Supervisor

The purpose of the clock supervisor is to process the 400 Hz clock interrupts and count the 400 Hz down to a lower iteration rate needed by tasks in the clock table.

The clock supervisor includes the clock interrupt service routine (CLKINT) shown in Figure 31, a routine to place tasks in the clock table (CLKTAB) as shown in Figure 32, and a routine to remove tasks from the clock table (CLKREM) as shown in Figure 33. The clock table contains task ID's, the period of execution of each task, and a counter for each task. CLKTAB is called with a task ID in latch 1 and the iteration period in units of 2.5 ms in latch 2. CLKTAB places the task ID and its period in the clock table and sets the counter. The flowchart for CLKTAB is shown in Figure 32. CLKINT is executed each time the 400 Hz clock signal is received. For each entry in the table the counter is decremented by one. If the counter has reached zero, it is reset to the period and the task is queued up by calling TASKQU. The flowchart for CLKINT is shown in Figure 35. CLKREM is called with a task ID in latch 2. It searches the clock table and removes all occurrences of the task ID in the table. The flowchart for CLKREM is shown in Figure 34.

Test Interrupt

The purpose of the test interrupt service routine shown in Figure 36 is to execute a test task each time the test interrupt button is pushed.

The test interrupt routine requires that the test tasks be numbered sequentially and has a counter that keeps track of how many of the tests have been executed so far. Each time the test interrupt button is pushed, TSTINT increments its counter and queues up the next task in the sequence by calling TASKQU. It then jumps to DISPAT which will execute the task. The flowchart for TESINT is shown in Figure 37.

Interrupt Priority Level

Latch 3 is the interrupt priority mask register. Only interrupts that have a priority greater than the contents of latch 3 can interrupt the DP. The contents of latch 3 can be temporarily changed without losing the original contents through the use of PRICHG and PRIRES shown in Figure 38. PRICHG saves the old contents of latch 3 and sets a new level. PRIRES restores the old contents. The memory word INTPRI normally contains the same value as latch 3.

RSTINT, also shown in Figure 38, is used by interrupt routines to clear an interrupt request that is being processed without returning to the interrupted program yet.

Linkage Tables

The executive contains linkage tables to all the executive subroutines and interrupt routines that could be entered from outside the executive (see Figure 39). This allows changes to be made in the executive which move routines around without having to modify any external routines that call them.

```
/
/          CLOCK INTERRUPT ROUTINE
CLKINT     LD,3,K15            ;MASK INTERRUPTS
           JIFFS,31,CLKRTN     ;IF FLAG 31 THEN RETURN
           GSB,,RSAVE          ;SAVE L1,L2
           LD,1,TP108          ;GET TEST POINT COUNTER
           AD1,1,K1            ;INCREMENT COUNT
           WR1,,TP108          ;WRITE BACK IN MEMORY
           DCI,29,1            ;PUSH PRIORITY DOWN ON STACK
           DAT1,11             ;MAKE NEW CURRENT PRIORITY
           WR1I,29,0           ;IN CASE THIS ROUTINE GETS INTERRUPTED
           LD,3,INTPRI         ;NOW UNMASK INTERRUPTS
           STM,26,CNUM         ;SET UP LOOP FOR CLOCKS
           JIFIZ,26,CLKMTY     ;CHECK FOR 0
CLKLP      LD1I,26,CCOUNT      ;GET COUNTER
           SUB1,1,K1           ;COUNT
           WR1I,26,CCOUNT      ;STORE NEW COUNT
           DAT2,0              ;COMPARE COUNT TO 0
           JIFFS,L1>L2,CLKOK   ;IF GREATER CLOCK IS OK JUMP
           LD1I,26,CMAX        ;ALARM HAS GONE OFF RESET CLOCK
           WR1I,26,CCOUNT      ;WITH MAXIMUM COUNT
           LD2I,26,CID         ;PUT ID IN L2
           GSB,,TASKQU         ;CALL TASK QUEUER
           WRF,45,1            ;MAKE SURE WE WILL GO TO DISPAT
CLKOK      IND,26,CLKLP        ;LOOP UNTIL ALL CLOCKS ARE CHECKED
CLKMTY     STM,26,CTIC         ;GET ELAPSED TIME PRECOUNTER
           IND,26,CLKNT        ;COUNT AND CHECK FOR 0 ,IF NOT 0 THEN
           STM,26,CRATE        ;=0 RESET COUNTER
           LD,1,TIME           ;AND GET ELAPSED TIMER
           AD1,1,K1            ;UPDATE TIME
           WR1,1,TIME          ;SAVE NEW TIME
CLKNT      RIRM,26,CTIC        ;SAVE PRECOUNTER
           GSB,,RREST          ;L1,L2
           GSB,,RSTINT         ;RESET INTERRUPT
           DCI,29,-1           ;POP PRIORITY STACK
           JIFFS,45,DISPAT     ;IF ANY HIGHER TASK THEN DO IT
           RTN
CLKRTN     RTNINT              ;RESET INTERRUPT, RETURN
```

Figure 31. Clock Interrupt Routine (CLKINT)

```
/          MAKE AN ENTRY IN THE CLOCK TABLE
/          ENTER WITH ID IN L1, COUNT IN L2
CLKTAB     STM,0,CNUM          ;GET NUMBER OF ENTRIES
           DCI,0,-1            ;INCREMENT NUMBER OF ENTRIES
           WR1I,0,CID          ;STORE ID IN TABLE
           WR2I,0,CMAX         ;STORE MAX COUNT IN TABLE
           RIRM,0,CNUM         ;STORE IN CNUM
           RTN
```

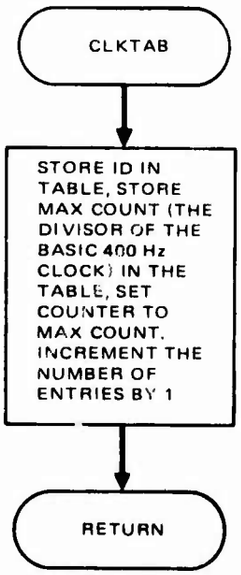


Figure 32. Clock Table – Entry Routine and Flowchart (CLKTAB)

```
/          REMOVE AN ENTRY FROM THE CLOCK TABLE
/          ENTER WITH ID OF TASK TO BE REMOVED IN L1
/          USES L1,L2,I0,I1
CLKREM     NOP
           ST,0,CNUM           ;GET NUMBER OF CLOCKS
           JIFIZ,0,CLKREX      ;IF EMPTY, EXIT
CLKRLP     LD2I,0,CID          ;GET ID FROM TABLE
           JIFFS,L1=L2,CLKRMV  ;COMPARE, IF EQUAL REMOVE FROM TABLE
           JT,,CLKROK          ;OTHERWISE IT'S OK
CLKRMV     STM,1,CNUM          ;TO REMOVE: GET ADDRESS OF LAST ENTRY
           LD2I,1,CID          ;GET LAST ID
           WR2I,0,CID          ;MOVE TO EMPTY SPACE
           LD2I,1,CMAX         ;GET MAX COUNT
           WR2I,0,CMAX         ;MOVE TO EMPTY SPACE
           LD2I,1,CCOUNT       ;GET COUNTER
           WR2I,0,CCOUNT       ;MOVE TO EMPTY SPACE
           DCI,1,1             ;DECRIMENT NUMBER OF CLOCKS
           RIRM,1,CNUM         ;AND STORE
CLKROK     IND,0,CLKRLP        ;LOOP THROUGH TABLE
CLKREX     RTN
```

Figure 33. Clock Table – Remove an Entry (CLKREM)

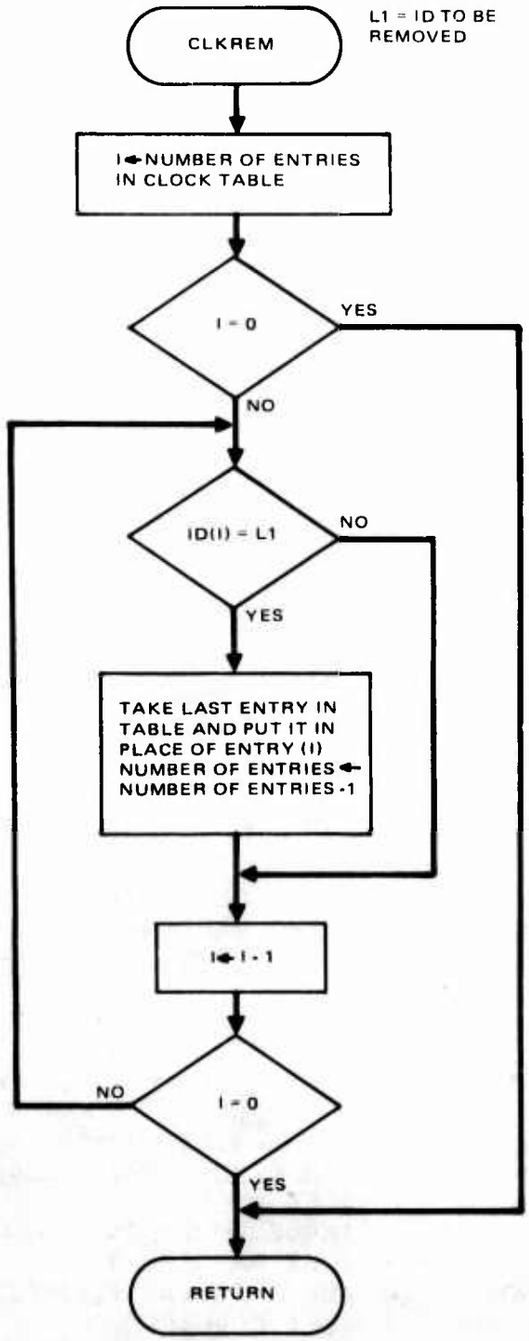


Figure 34. CLKREM Flowchart

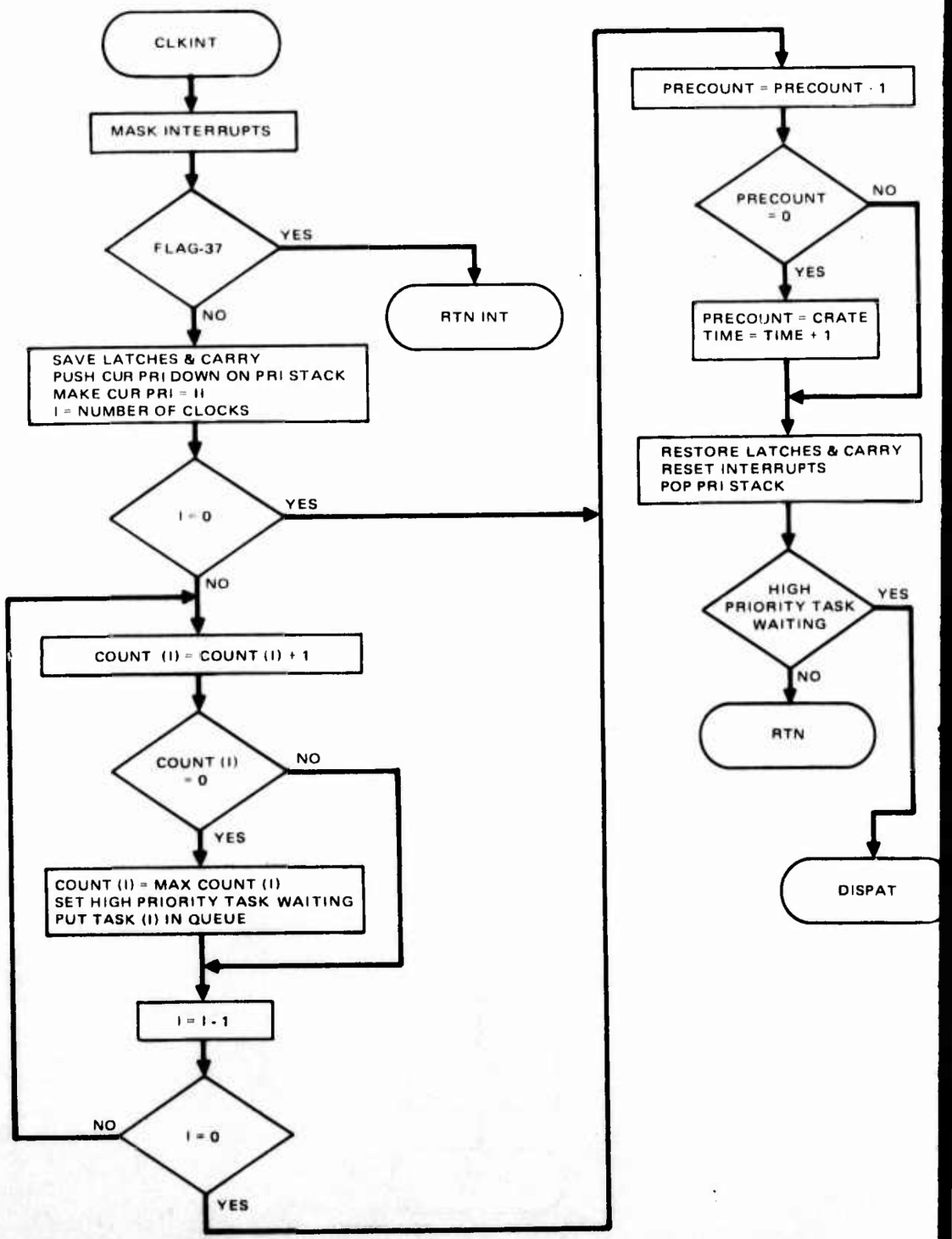


Figure 35. CLKINT Flowchart

```
/          TEST INTERRUPT ROUTINE
TESINT     GSB,,RSAVE          ;SAVE REGISTERS
           DAT2,-1             ;BLANK OUT
           WR2,5,0             ;DISPLAY
           LD,2,TESTNO         ;GET TEST NUMBER
           AD2,2,K1            ;INCREMENT
           WR2,,TESTNO         ;STORE
           AD2,2,TESTID        ;ADD ID NUMBER OF FIRST TEST
           GSB,,TASKQU         ;QUEUE UP TEST
           GSB,,RREST          ;RESTORE REGISTERS
           GSB,,RSTINT         ;RESET INTERRUPT
           JT,,DISPAT          ;GO TO DISPATCHER
/
```

Figure 36. Test Interrupt Routine (TESINT)

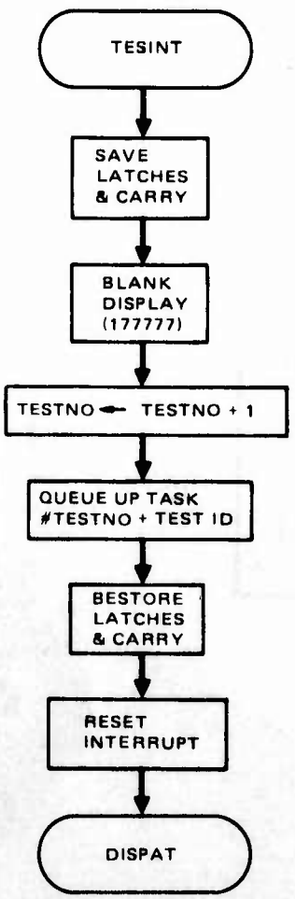


Figure 37. TESINT Flowchart

```
/ TEMPORARILY CHANGE INTERRUPT PRIORITY MASK
/ ENTER WITH NEW LEVEL IN L2
/ BOTH L1 AND L2 ARE USED
PRICHG    NOP
          DCI,31,1            ;MAKE ROOM ON STACK
          LD,1,INTPRI         ;GET INTERRUPT PRIORITY LEVEL
          WR1I,31,0           ;SAVE ON STACK
          WR2,3,INTPRI        ;SET NEW LEVEL
          RTN
/
/ RESTORE INTERRUPT PRIORITY LEVEL
/ LATCH 1 IS USED
PRIRES    LD1I,31,0           ;GET OLD LEVEL FROM STACK
          WR1,3,INTPRI        ;RESTORE IT TO INTPRI AND MASK
          DCI,31,-1           ;CLEAR STACK
          RTN


/         RESET INTERRUPT REQUEST AND RETURN TO CALLING PROGRAM
          WITHOUT RETURNING TO INTERRUPTED PROGRAM
RSTINT    RTNINT              ;RESET INTERRUPT REQUEST
/
/-------------------------------------------------------------
```

Figure 38. PRICHG, PRIRES and RSTINT Routines

```
*SET      3072
/
/         LINK TABLE TO JUMP THROUGH WHEN CALLING EXECUTIVE
          JT,,INITI
          JT,,TASKQU
          JT,,DISPAT
          JT,,DBSUPR
          JT,,IBUSSU
          JT,,IDTABL
          JT,,CLKTAB
          JT,,CLKREM
          JT,,PRICHG
          JT,,PRIRES

/         INTERRUPT VECTORS
*SET      3578
          JT,,TESINT
          JT,,CLKINT
          JT,,IBUSIN
          JT,,IBUSIN
          JT,,DBERR
          JT,,DBCOMP
END       NOP
```

Figure 39. Linkage Tables

(The reverse of this page is blank)

SECTION III

CENTRAL INERTIAL GUIDANCE TEST FACILITY OPERATION

This section defines the digital processing system and the associated software which is to be evaluated at the CIGTF, Holloman Air Force Base. The elements of the digital processing system are shown in Figure 40. The Hughes digital processor is a breadboard unit designated DP1 by the Air Force Armament Laboratory (AFATL/DLMM), Eglin Air Force Base, Florida. The IMU is a Hamilton Standard 3030 breadboard strapdown inertial measurement unit (GFE). The interconnection of these elements is performed by the weapon bus, the bus interface units (BIU's), and appropriate interface equipment developed by the contractor.

This equipment mechanizes a strapdown inertial navigation system in which the DP1 performs the required computational and communication functions under software control. The performance of the system is to be evaluated on the basis of navigation accuracy relative to the Completely Integrated Reference Instrumentation System (CIRIS). CIRIS performs as a master navigator and records DP system parameters for these tests.

SYSTEM DESCRIPTION

This section defines the system functions to be performed and the allocation of these functions among the system elements. The system hardware is described, and performance parameters are summarized in the subsequent HARDWARE DESCRIPTION subsection. The interfacing of the system software with the system hardware is performed by the Executive software described in Section II. The system software functions are contained in a number of software tasks. The functions performed in each task and the interrelationships of the tasks are presented in the SOFTWARE DESCRIPTION subsection. The detailed software documentation is presented in the Digital Processor Software Development Report, Report No. DGWT 0165-1.

Functional Description

The elements of the system configuration to be evaluated at CIGTF are shown in Figure 40. The functions of the CIRIS are to (1) supply reference data to the DP1 and (2) request and format DP1 data for the instrumentation system. The system-level timing of both of these data transfers is controlled by the HP2100 computer in CIRIS. The strapdown IMU supplies incremental velocity and rotation angle data to DP1 in a coordinate system determined by the IMU mounting position. The timing of IMU data transfers is controlled by DP1. The data from the altimeter (GFE) is used for stabilization of the vertical channel of a strapdown inertial reference system. The timing of altimeter data transfer to DP1 is controlled by DP1.

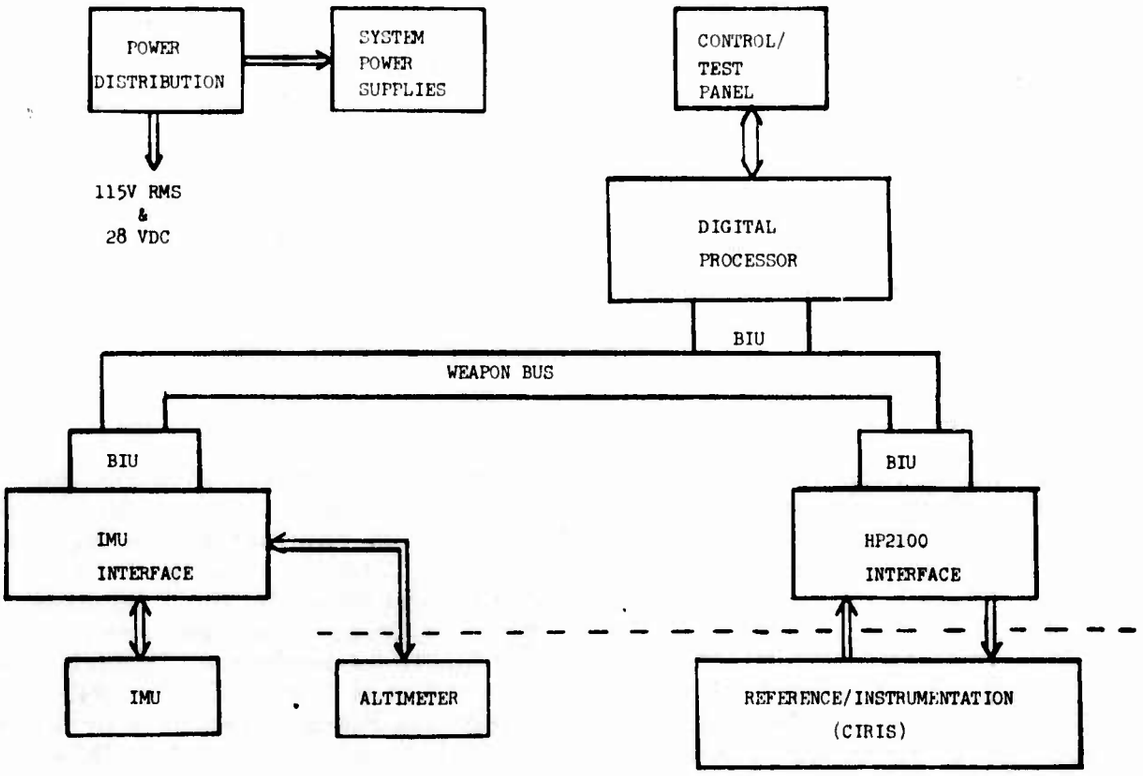


Figure 40. Digital Processing System (CIGTF) Block Diagram

The weapon bus and BIU elements are used for all data transfers within the system. The synchronization of the system operations is accomplished by interrupt messages which are transmitted on the interrupt bus portion of the weapon bus.

All DP1 software operations are initiated either by operator action using the control panel or by interrupts. The control panel provides means for the operator to start and stop DP1, control system modes, and monitor system operation in real time. When the START button is pressed, the state of the DP1 is initialized in preparation for receiving reference data from the HP2100 interface. Only self-test software execution can be performed after this time if the HP2100 is not connected (or simulated) to the system.

When the HP2100 is supplying reference data, the available system modes are set by the control panel switches for either free navigation or alignment. The first set of reference data received after navigation is enabled will cause the initialization of the strapdown inertial navigation software to the state corresponding to the reference data. Data transfers from the IMU at 100 Hz and from the altimeter at 10 Hz are also initiated under DP1 software control in response to internal DP1 time strobe interrupts. The appropriate strapdown navigation software routines will be performed at the corresponding iteration rates until the RESET button on the control panel is pressed.

The alignment filter software is executed once for each set of reference data received after alignment is enabled. The total number of alignment filter iterations is 900 and is under software control. After this number of iterations, the system reverts to free navigation.

Instrumentation data is transferred from the DP1 to the HP2100 computer when requested by the HP2100. The first 13 of the 64 data words are a repetition of the last set of reference data received from HP2100 by the DP1. Whenever the DP1 is running and the HP2100 is operating, the reference data may be compared to the appropriate instrumentation data block to confirm interface integrity. The remaining 51 instrumentation data words allow evaluation of the strapdown inertial reference performance.

Hardware Description

Two categories of hardware elements are included in the CIGTF Operation: Government Furnished Equipment (GFE) and Hughes equipment developed on this contract. The GFE consists of the Hamilton Standard 3030 breadboard inertial measurement unit (IMU), the altimeter, and the CIRIS. The Hughes-developed equipment includes a breadboard digital processor (DP1), a control panel for the processor, the weapon bus, equipment to interface the GPE with the weapon bus, and the power distribution and power supplies. The Hughes-developed equipment and the IMU are installed in bays 3 and 4 of a palletized rack, as shown in Figure 41. The HP2100 computer (part of CIRIS) is located in the same rack. The remainder of the CIRIS is installed on a different pallet. The altimeter is also remotely located. The following paragraphs describe the hardware elements and show their performance characteristics.

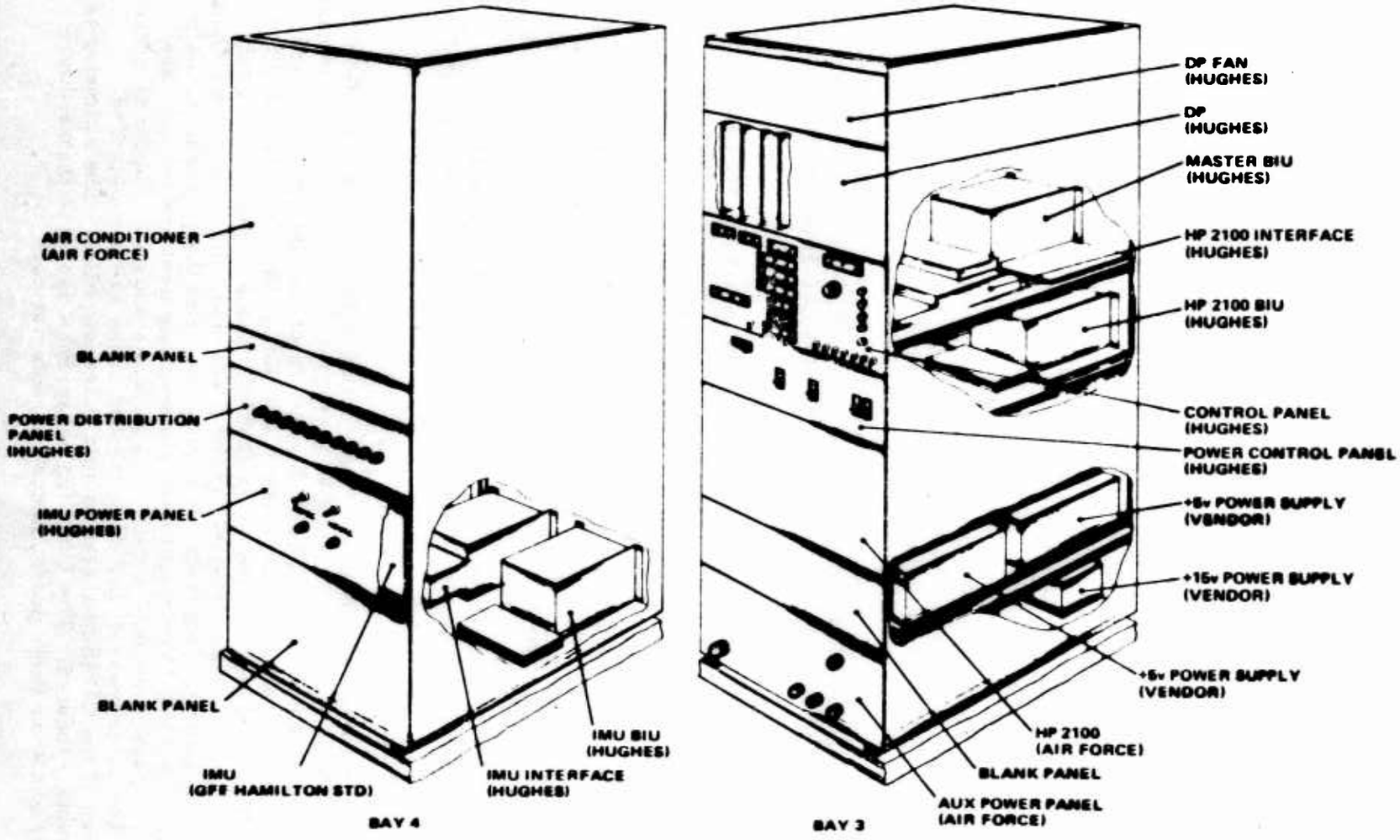


Figure 41. Rack Layout

HS-3030 Inertial Measurement Unit. The HS-3030 strapdown inertial measurement unit was developed for aircraft and missile applications requiring moderate inertial quality navigation performance in the range of 4 to 10 nmi/hour CEP.

The HS-3030 IMU consists of an orthogonal triad of three pulse rebalanced Mini-Rig 30 strapdown rate integrating gyros, three pulse rebalanced Systron-Donner 484 accelerometers, modular supporting Digital Control Electronics (i.e., countdown, waveform generation, and input/output interface circuits), and a sensor block temperature controller and power supply.

The three HS-3030 gyro loops are scaled to provide a full-scale rate capability of ±200 degrees/second independently about each of the three vehicle primary axes. This full-scale capability is sufficient to accommodate the most severe measuring range requirements currently anticipated for the system. In order to provide both high rate and precision readout capabilities while at the same time conserving system operating power, a dual gyro measuring range capability is provided for each of the three gyro rebalance channels. Full-scale rate capability in the low rate mode is ±25 degrees/second. Mode switching is accomplished automatically within each servo rebalance electronics channel. Each of the three accelerometer channels is scaled for ±40 g full-scale operation.

Inertial performance characteristics are presented in Table 1.

TABLE 1. HS-3030 INERTIAL MEASUREMENT SYSTEM CHARACTERISTICS

GYROS:	HAMILTON STANDARD MINI-RIG 30 (3)
ACCELEROMETERS%	SYSTRON-DONNER 484 (3)
INPUT RATE CAPABILITY:	HIGH RATE MODE ±200 DEGREES/SECOND LOW RATE MODE ±25 DEGREES/SECOND
GYRO SCALE FACTOR:	HIGH RATE MODE 5.6 ARC SEC/PULSE LOW RATE MODE 0.7 ARC SEC/PULSE
INPUT ACCELERATION CAPABILITY:	±40 G
ACCELEROMETER SCALE FACTOR:	0.04 FPS/PULSE

CIRIS Description. The Completely Integrated Reference Instrumentation System (CIRIS) shown in Figure 42 is capable of providing highly accurate position, velocity and attitude reference over long flight paths for real-time use in testing guidance and navigation systems. The CIRIS is palletized and normally occupies pallet station No. 1 in the C-141. This airborne automated system is operationally independent due to integration of all its reference measurement sources by minicomputers.

CIRIS generates the reference data by using four measurement devices that are controlled and time-coordinated by a minicomputer to provide inputs to a 15-state Kalman filter. The real-time filtered reference data which is generated in a second minicomputer is distributed to the test data acquisition computer and recorded with the raw measurement data on the magnetic tape. Further processing (backward filtering and smoothing) can be done postflight as required.

The measurement hardware includes an inertial navigation system stabilized by barometric altitude from an Air Data Computer, a Doppler Radar, and a precision radio range/range-rate system. The inertial system data is used in the filter as a continuous reference for data propagation and reference for the filter error states. The error states are updated by incorporation of barometric altitude, doppler velocities, and precision range and range-rates to precisely surveyed ground sites. The CIRIS accuracies are directly dependent on the measurements obtained from the range/range-rate system which includes an airborne interrogator that is used to selectively interrogate one ground-base transponder every two seconds. A set of four transponders nearest the current aircraft location is used to provide one redundant measurement in a time-phased triangulation scheme. The transponders and associated omnidirectional antenna are portable and are designed for remote operation. They are deployed in a triangular pattern separated by approximately 150 miles in a line along the flight path. CIRIS degradation can occur when flight paths leave areas of radio range coverage which extends to 200 nautical miles line-of-sight. Incorporation of doppler radar data will minimize degradation until coverage is resumed.

CIRIS data has the following specifications:

1. Position accuracy to 13 feet (1 sigma) in three axes.
2. Velocity accuracy to 0.10 ft/sec (1 sigma) in three axes.
3. Attitude accuracy to 3 arc-minutes (1 sigma).
4. Real-time reference points every 5 seconds.
5. Postflight reference points every 2-4 seconds.

Altimeter Description. The altimeter is an HLT Industries, Inc. No. 502000-39 Indicated Altitude Pressure Transducer System. The altimeter output is provided by a potentiometer whose resistance ratio is a function of altitude. In the system, the potentiometer is supplied with a reference

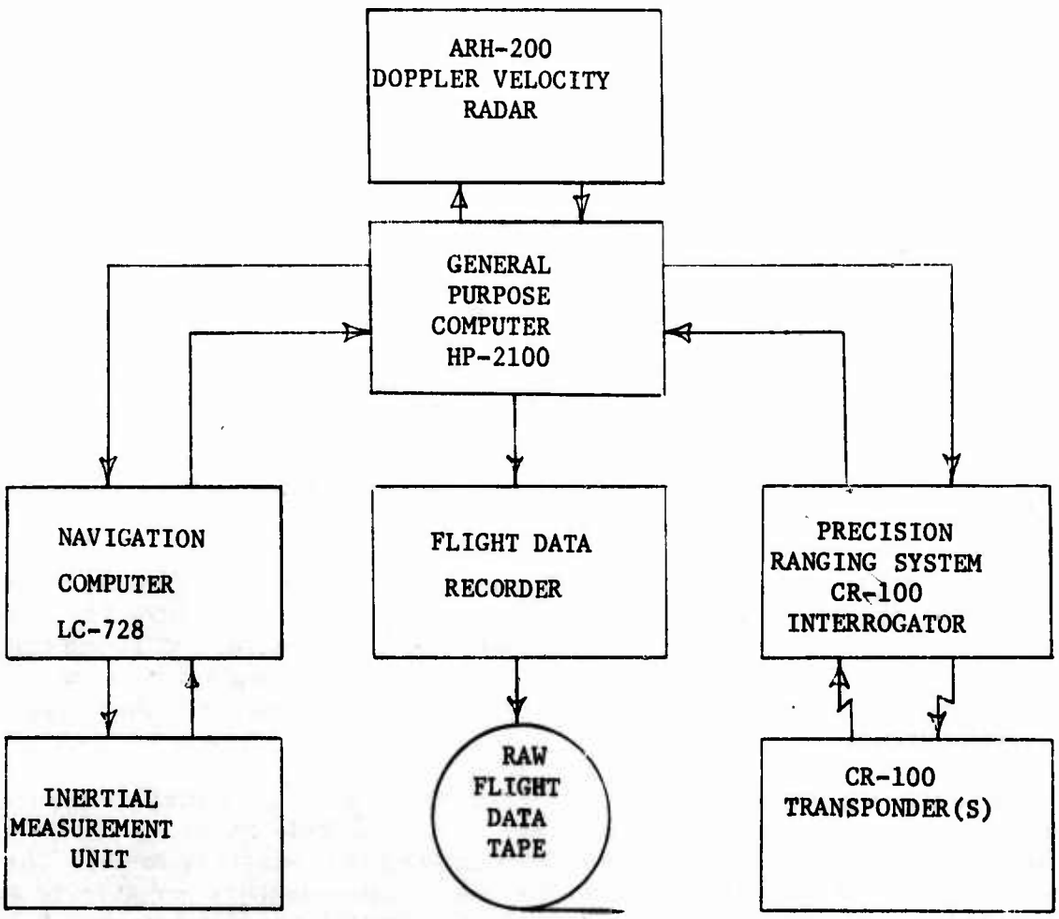


Figure 42. Completely Integrated Reference Instrumentation System (CIRIS)

voltage from the IMU interface, and the pickoff voltage relative to the reference voltage provides a measure of altitude to the IMU interface. The performance characteristics of the altimeter are:

Altitude Range:	0 - 80000 feet
Output Linearity:	±1% of full range
Nominal Altitude Resolution:	20 feet
Repeatability Error:	±60 ±130 feet (altitude dependent)

Hughes Developed Equipment. A detailed description of the Hughes developed equipment is presented in Operating Manual and System Description for Digital Processor Number 1 and Installation at Central Inertial Guidance Test Facility, DGWT 0210-1. This section provides only a summary of the equipment characteristics.

Digital Processor. The DP1 is a breadboard implementation of a subset of the digital processor architecture presented in Volume I of this report. The design is an outgrowth of the PDAP design and incorporates some instruction set modifications to facilitate execution of digital processor software functions. The DP1 instruction set is presented in the Programmers Manual, DGWT 0170-2.

The DP1 breadboard processor consists of a number of circuit boards each of which contains an autonomous function as shown in Figure 43. The circuit boards are interconnected by a common bus structure allowing the relative position of the boards in the rack to be changed with no wiring changes. Additional boards; e.g., program memory, may also be plugged into the bus structure.

The program memory (PM) contains the machine language instructions. Three PM cards were fabricated: one read-only-memory (ROM) card for flight test, and two read/write (RAM) cards for laboratory tests. The operand memory (OM) card contains the read/write memory for storing system computational variables. A block of ROM for storage of system constants were installed on the ROM PM card.

The control unit (CU) card decodes the instructions from the PM, generates timing and control signals for the processor, and generates addresses for both the OM and PM cards. The arithmetic unit (AU) card performs arithmetic and logic operations on operands. The interrupt card (INT) processes interrupts from the BIU I/O card to synchronize the processor operations with external events. The BIU I/O interfaces the processor with the weapon bus and provides both the processor clock and real time clock interrupts (rate set under software control).

The DP1 parameters are shown in Table 2.

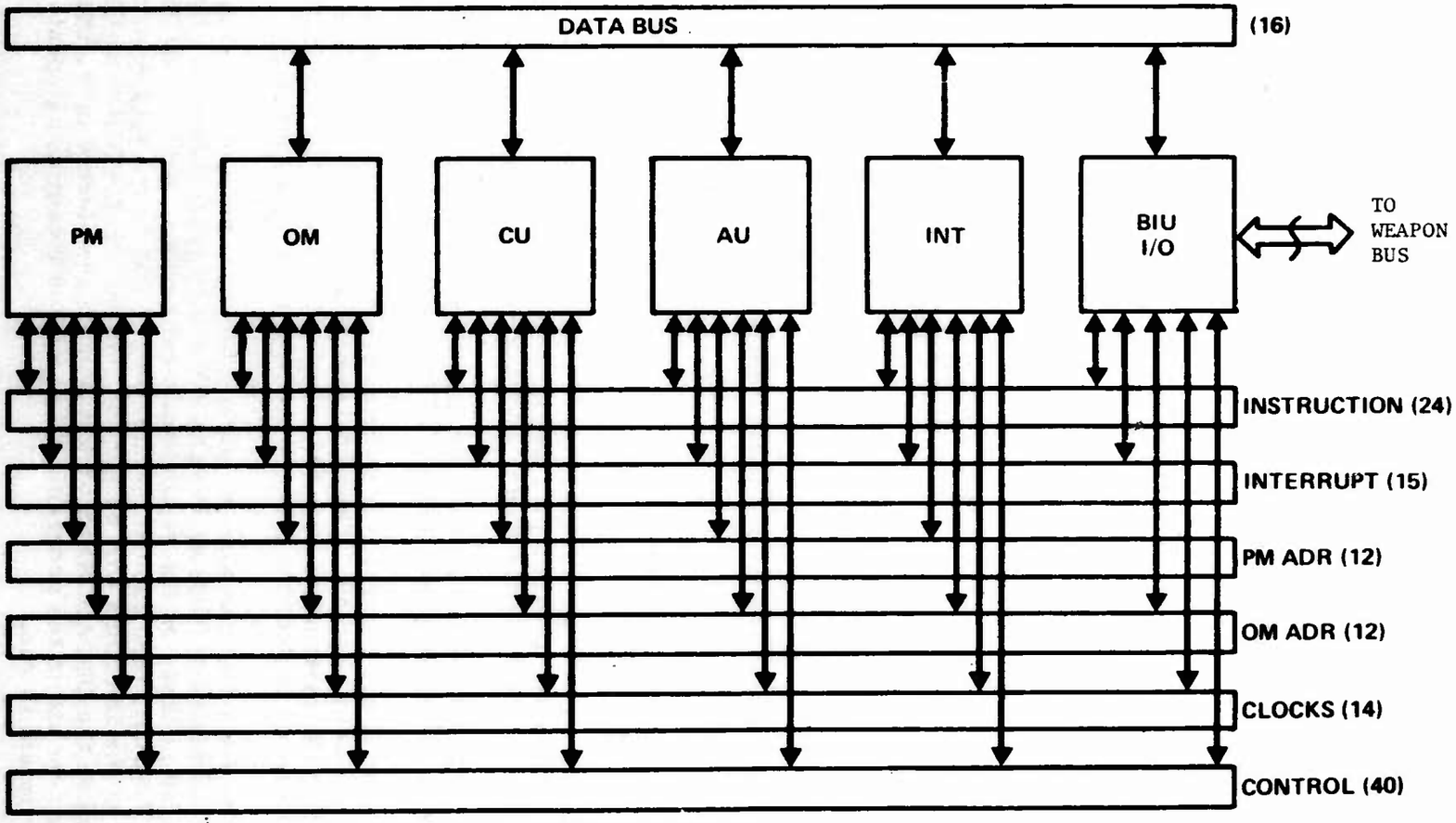


Figure 43. DPI Block Diagram

TABLE 2. DP1 PARAMETERS

NUMBER OF INSTRUCTIONS:		64
COMPUTATION WORD LENGTH:		16 BITS
PROCESSOR THROUGHPUT (SPECIFICATION MIX):		1.85 MOPS
PROGRAM MEMORY		4 K WORDS
PROGRAM WORD SIZE		
TECHNOLOGY:	ROM OR RAM – SELECTABLE IN 1 K WORD BLOCKS VIA SWITCHES ON PM CARDS.	24 BITS
OPERAND MEMORY:	RAM	2 K WORDS
	ROM	512 WORDS
TOTAL ADDRESS SPACE (INCLUDING I/O)		4 K WORDS
VECTORED PRIORITY INTERRUPTS:		15 LEVELS
SUBROUTINE STACK:		32 LEVELS
INDEX REGISTERS:		2
ARITHMETIC REGISTERS:		2
FLAGS:		64

Control Panel. The Control Panel (Figure 44) is used for both monitoring and controlling the DP1.

Information from Program Memory, the 24-bit instruction word, is displayed on the top octal display and is broken down into the OP code, the Tag Field, and the Address Field, left to right, respectively.

The 16-bit Data Bus is displayed in the next octal display and also can be selected for decimal display by the rotary select switch.

By software control, the information on the Data Bus can be latched into the next octal display. The programmer has this feature at his control by inserting the instruction to output data to I/O Latch 5. The information can be displayed in decimal through the decimal display switch.

The Data Bus can also be latched for octal and decimal display, using the Operand Memory (OM) or Program Memory (PM) Address. By selection of a desired address on the octal thumbwheel switch labeled OM/PM ADDR FOR DB LATCH, then choosing either OM or PM on the toggle switch, the contents of the Data Bus will be latched into the DB LATCH (PM/OM ADDR) display when the OM ADDRESS or PM ADDRESS selected in the thumbwheel switch is reached in the program. This will be a continuing update as long as the program is run.

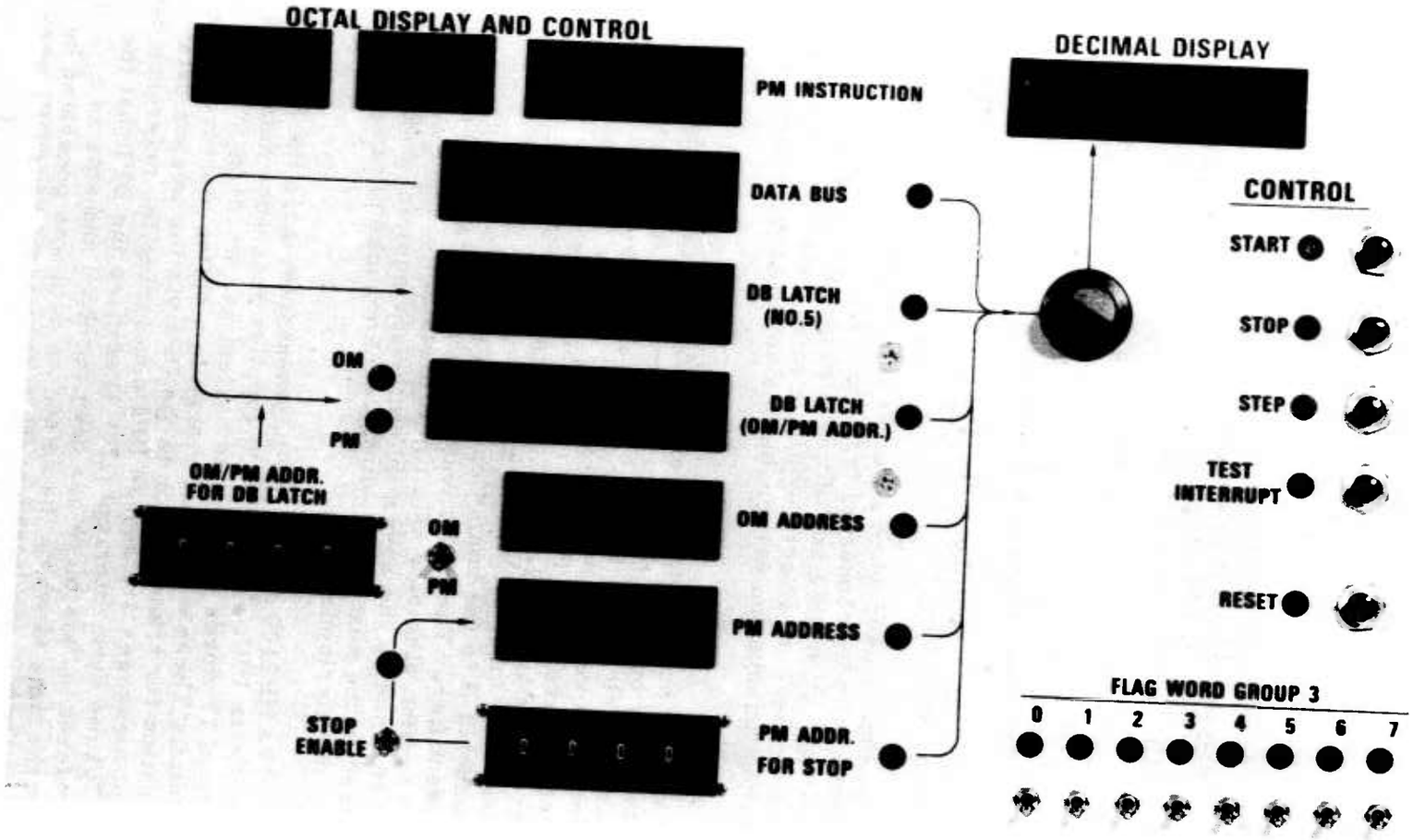


Figure 44. Control Panel

The program can be stopped by several means. The simplest is to depress the STOP pushbutton which will halt the DP1 at its present address. Pressing the START button will continue the program from that address. If RESET is pressed, the program will stop and the program counter will be set to zero where the program will continue from when restarted. Pressing STEP will also stop the program, similar to STOP, and continual depressing will continue the program from that point, one instruction at a time.

The PM ADDRESS FOR STOP octal thumbwheel switch can be used for stopping the DP1 when the STOP toggle switch is set to enable. When the program reaches the thumbwheel PM address, it will stop. Pressing START, the program will continue on from that point.

Both the OM and PM address are displayed in octal continuously and in addition can be selected for display as a decimal number. The PM ADDR FOR STOP OCTAL thumbwheel can be selected for decimal display to assist perhaps in any octal to decimal conversion calculation.

The Flag Word Group 3 switches are used under software recognition, usually as conditions for branching.

The Test-Interrupt is used when all other interrupts (such as the B. I. U. -I/O) are electrically disconnected from the DP1 and is purely for test and troubleshooting. To enable any of the 15 priority interrupts, the 15 rocker DIP switches on the Test Panel Card have to be switched accordingly. Then the Test-Interrupt pushbutton will enable the selected interrupt to the DP1.

Weapon Bus. The weapon bus provides a communication link for the transfer of data and control signals between the system elements. All communication over the weapon bus is in bit serial format. Separate communication paths are provided for data (weapon data bus) and control signals (weapon interrupt bus). All data transfers are under the control of the DP1 software via the master bus interface unit connected to DP1. Control signal transfers may be initiated by any system element by sending the appropriate commands to the slave BIU which connects it to the bus. Control signals are transferred as an 8 bit interrupt identifier. The BIU's only provide an interface between the weapon bus and the system elements -- all interpretation of command messages or data is performed by the respective interface equipment (IMU or HP2100) as discussed in the next section. The performance parameters of the weapon bus are summarized in Table 3.

HP2100 Interface. The HP2100 Interface equipment interfaces the HP2100 computer (CIRIS) with its BIU. The HP2100 supplies reference data to the interface as a block of 13 words (16 bit data content per word) in CAROUSEL IV format. These words are stored in a buffer memory in the interface in preparation for transmission to the DP1 via the weapon data bus. Two interrupt messages are sent to DP1 concerning the reference data by the interface. A time Sync interrupt (ID = 1) notifies the DP1 that the first word of the reference data block has been received and acts as a marker to indicate the time at which the reference data was generated by CIRIS. A Reference Data Ready interrupt (ID = 2) is sent to the DP1 when all 13 words have been stored in the buffer.

When the HP2100 desires instrumentation data from DP1, an Instrumentation Data Request discrete is sent to the interface which sends the corresponding interrupt (ID = 5) to DP1. As the instrumentation data is received from DP1 via the weapon data bus, it is transferred in word serial format to the HP2100 at the data bus word transfer rate. A data valid strobe is also output to the HP2100 by the interface for each word.

TABLE 3. WEAPON BUS PARAMETERS

WEAPON DATA BUS	
DATA CONTENT/WORD	16 BITS
MAXIMUM DATA BLOCK SIZE	256 WORDS
TRANSMISSION RATE CAPACITY	100K WORDS/SECOND
WEAPON INTERRUPT BUS	
INTERRUPT IDENTIFIER	8 BITS
SINGLE WORD TRANSMISSION	
TRANSMISSION RATE CAPACITY	120K WORDS/SECOND

IMU Interface. The IMU interface connects both the IMU and the altimeter to the BIU. Data is requested of the IMU by the interface when it receives either the Data Request interrupt (ID = 64) from the DP1 via the weapon interrupt bus. Either of these interrupts will cause the IMU REQ signal to be sent to the IMU which responds with the IMU DR signal when the first of the six IMU parameters is ready at its output. The interface sends the Data Ready (IMU) interrupt (ID = 8) to DP1 at the time. The DP1 then accesses the IMU data via the weapon data bus. The six IMU parameters are transferred from the IMU through the interface to BIU in word serial format at the data bus transmission rate. The receipt of the Test Data Interrupt also causes a Test discrete to be output by the interface, but the test function is not available in the breadboard IMU.

When the Start Conversion interrupt (ID = 32) is received by the interface from DP1, the altimeter output analog signal is converted to digital format and stored in a buffer. When the digital word is stored, the interface outputs a Conversion Complete interrupt (ID = 10) to DP1 which may then initiate the transfer of the altimeter data via the weapon data bus.

Software Description

The DP1 software consists of 25 modules (tasks) which perform five system functions: System Management, Navigation, Alignment, Instrumentation, and Self-Test. Any discussion of these software functions must also include the operations occurring in the system hardware elements and the operations of the executive software which interfaces the functional software with the hardware. The function of the system is to implement a strapdown inertial navigator. Within this system function, the software performs the required communication control and computational functions.

The operating sequences in the system and the individual software tasks are described herein. Detailed documentation of these software modules is presented in the Digital Processor Software Development Report, No. DGWT 0165-1.

System Operating Sequences. All operating sequences are initiated by hardware interrupts. The selection of computational processes is performed by the appropriate System Management software module (task) under control by the operator via the control panel switches. The system operating procedure is presented in DGWT 0220-1 and will not be repeated in this section. The following paragraphs are concerned with the processing of system interrupts and system data after the appropriate system power-on and initialization procedure.

Reference Data Interrupt Sequences. Two system interrupts are generated by the HP2100 interface in response to reference data inputs by the HP2100 computer. This TIME SYNC interrupt (ID = 1) (see Figure 45) is generated and sent to DP1 when the first reference data word is received. The executive module, IBUSIN, is called when the interrupt is received by DP1 and queues up Task 1 (TSINT) for execution. This System Management task enables the processing of the Reference Data Ready interrupt, sets an

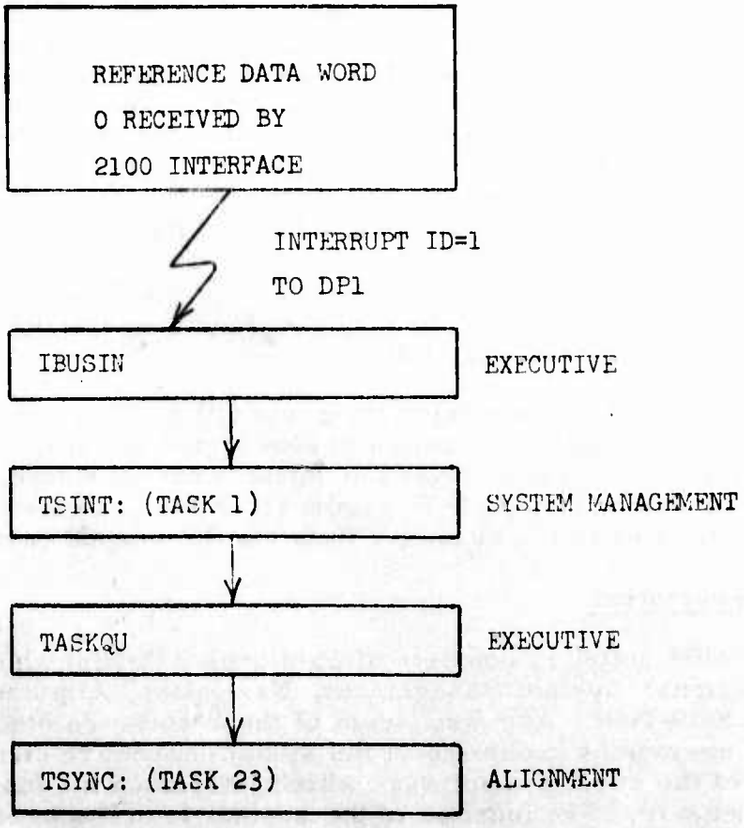


Figure 45. Time Sync Interrupt Sequence

ALIGN flag according to the position of the control panel ALIGN switch, and calls the TASKQU (Executive module) to queue up Task 23 (TSYNC). The alignment module, TSYNC, transfers DP1 navigation data for use by the alignment filter.

The Reference Data Ready interrupt (ID = 2) processing is shown in Figure 46. The receipt of the interrupt by DP1 causes the IBUSIN module to queue the REFRED (System Management task) for execution. REFRED calls the DBSUPR executive module with the appropriate data bus message parameters. DBSUPR initiates the transfer of the reference data from the HP2100 interface to DP1 via the weapon data bus. Completion of the transfer generates the XFER COMPLETE interrupt which calls the DBCOMP executive module. DBCOMP queues the RDXFRD system management task which initiates processing of the reference data. The first operation of RDXFRD is to queue the RINIT system management task via a call to the executive TASKQU module. RINIT transfers the reference data to the DP1 operand memory and performs format conversions required for compatibility with the DP1 computational modules. Further processing of the reference data is controlled by the RDXFRD module.

Navigation Initialization and Initiation (Figure 47). The navigation computation state of DP1 is initialized to correspond to the first set of reference data received after the control panel navigation switch is on. The initialization operation is controlled by RDXFRD which queues up IMUINT via a call to the executive TASKQU module. RDXFRD also inserts the I100-Hz task in the clock table to be called by the executive at a 100-Hz rate.

IMU Data Formatting and Data Transfer (Figure 48). The executive CLKINT module is called at a 400-Hz rate and, in turn, queues up the I100-Hz task for execution at 100 Hz. I100 Hz calls the executive IBUSSU module which sends the IMU DATA REQUEST interrupt (ID = 128) to the IMU interface via the weapon interrupt bus. When the IMU data is formatted and ready for transfer, the IMU interface sends the DATA READY interrupt (ID = 8) to DP1. Receipt of this interrupt calls the IBUSIN executive module which queues the DATRED system management task. DATRED calls the DBSUPR executive module with the appropriate data message parameters. DBSUPR then initiates the transfer of IMU data to the DP1 via the weapon data bus. The completion of the data transfer generates the XFER COMPLETE interrupt which calls the DBCOMP executive module. DBCOMP queues the IMUCOM system management task which checks the validity of the IMU data and transfers it to the DP1 operand memory.

Navigation Computations (Figure 49). The navigation computations are initiated by the IMUCOM task which queues up the IMU100 task via a call to the executive TASKQU module. The IMU100 updates the navigator state using the IMU data at a 100-Hz rate and also performs high priority computations at a 100-Hz rate. Lower priority computations at 10-Hz and 1-Hz rates are initiated by this module. The altimeter data sequence (see Figure 50) is performed at a 10-Hz rate. The 1-Hz computations are initiated via a call to the executive TASKQU module which queues up the IMU1 task.

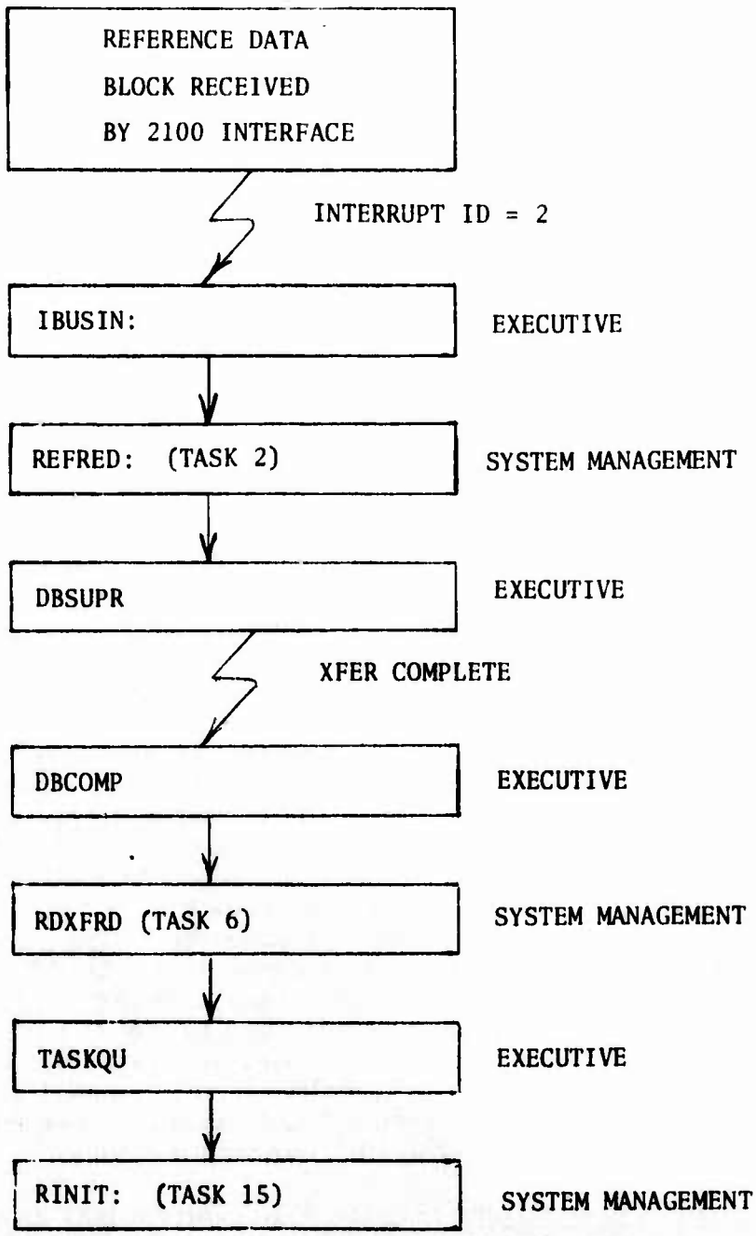


Figure 46. Reference Data Ready Interrupt Processing

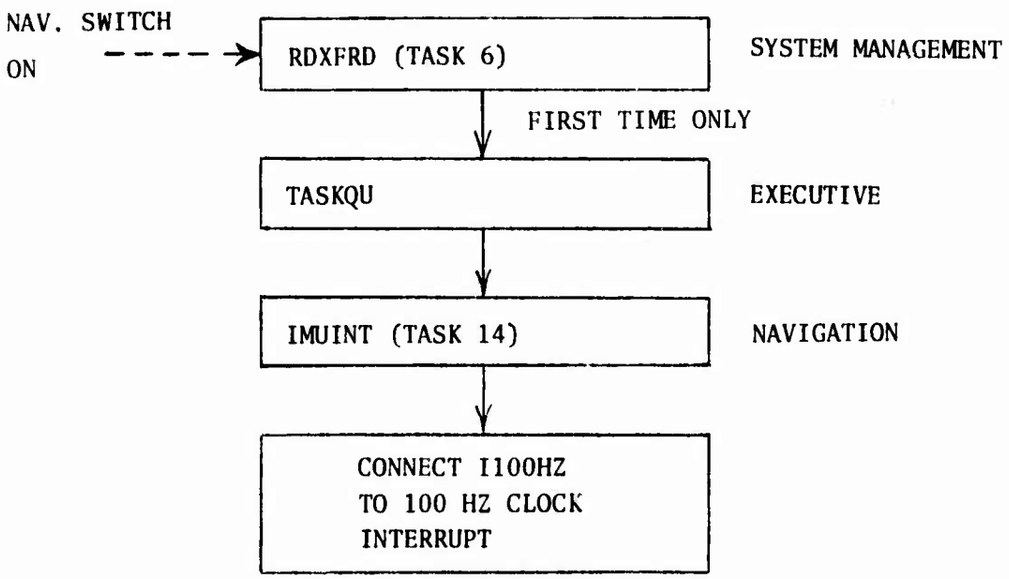


Figure 47. Navigation Initialization and Initiation

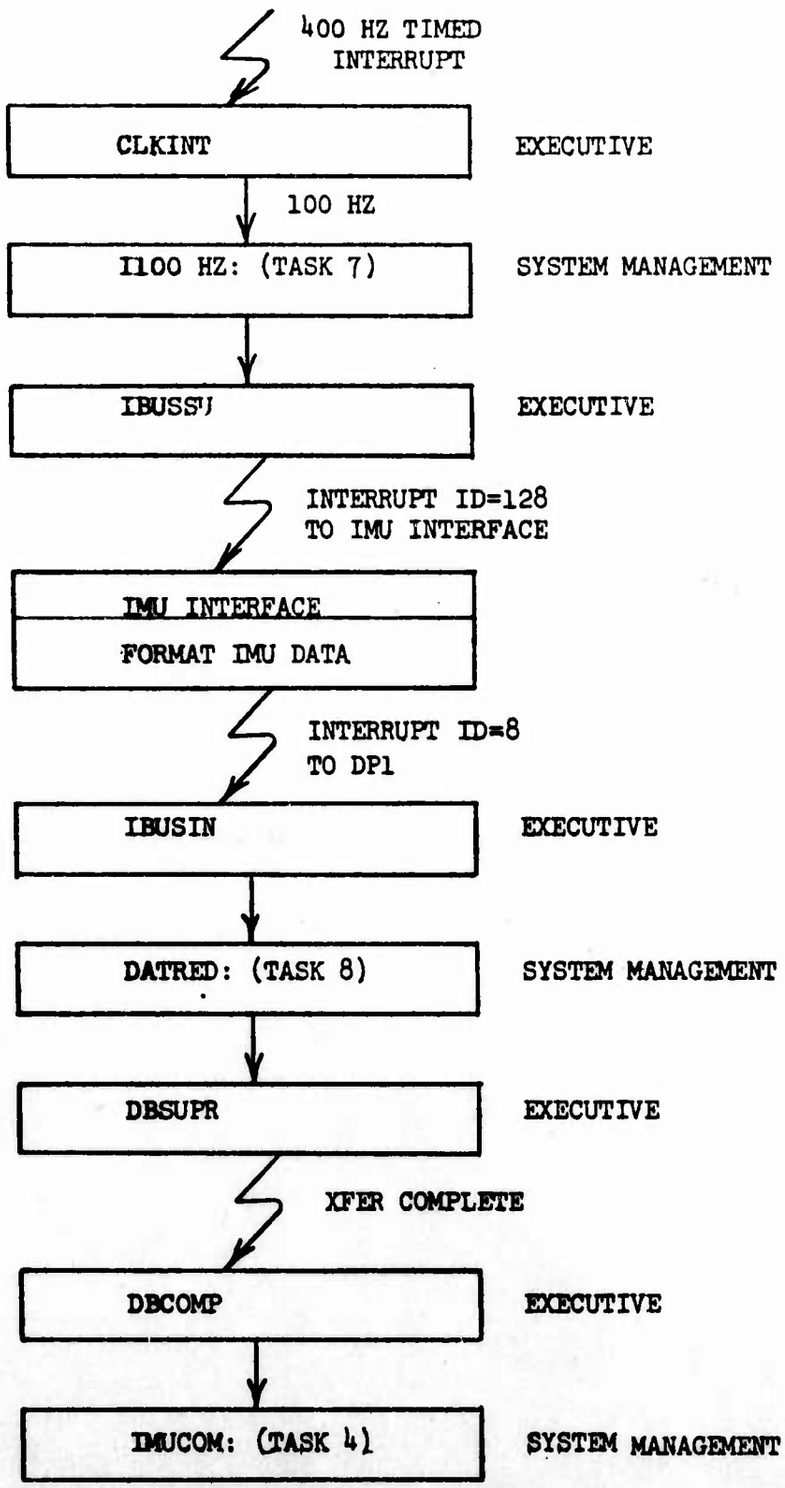


Figure 48. IMU Data Formatting and Transfer

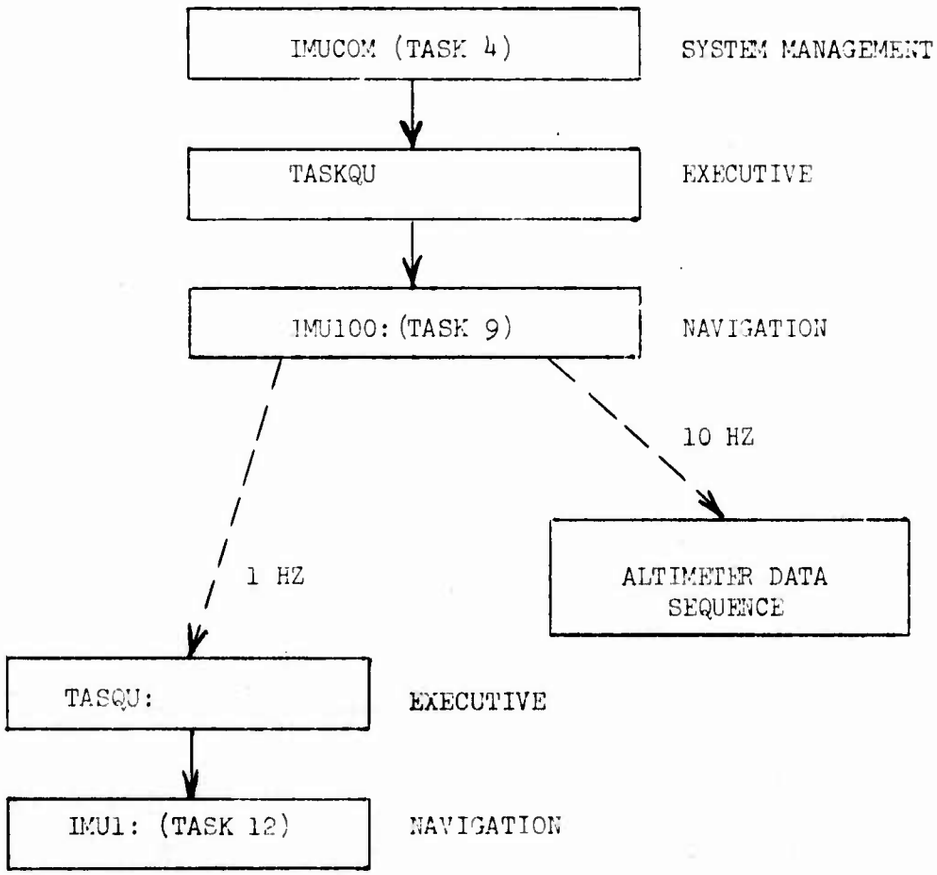


Figure 49. Navigation Computations

Altimeter Data Sequence (Figure 50). The altimeter data sequence is initiated by the IMU100 which calls the executive IBUSSU module at a 10-Hz rate. The IBUSSU sends the START CONVERSION interrupt (ID = 32) to the IMU interface. The IMU interface converts the altimeter signal to digital format and sends the CONVERSION COMPLETE interrupt (ID = 10) to DP1. This interrupt calls the executive IBUSIN module which queues the ALTRED module. ALTRED calls the DBSUPR module with the altimeter message transfer parameters and DBSUPR initiates the transfer. The XFER COMPLETE interrupt calls the executive DBCOMP module which queues the navigation IMU10 task.

Alignment Initialization and Computations (Figure 51). A sequencer associated with the alignment function is controlled by the RDXFRD system management module. The first execution of this module (see Figure 46) after the ALIGN flag is ON (set by TSINT module) will initialize the alignment filter. Initialization is performed by the FINI module which is queued via a call to the executive TASKQU module. As a part of the alignment initialization, the 100-Hz alignment task (CENTSK) is connected to the clock interrupt. Task 6 also queues the alignment filter computation task (DRITSK) 900 times via a call to the executive TASKQU module (CONTRL) each time it is executed. When the alignment filter is operating, Task 6 also queues the gyro bias computation module (GYRO) after each 300 iterations of the alignment filter.

IMU Data Accumulation (Figure 52). The IMU data accumulation module (COMPT) is connected to the clock interrupt as part of the DP1 initialization function.

Instrumentation Data Transfer Sequence (Figure 53). The instrumentation data transfer sequence is initiated by the HP2100 interface when it receives the instrumentation request from the 2100. The interface sends the INSTRUMENTATION DATA REQUEST interrupt (ID = 5) to DP1. This interrupt calls the IBUSIN executive module which queues the INSREQ task. This task formats the instrumentation data in preparation for transfer and then calls the DBSUPR executive module with the data message parameters. The DBSUPR initiates the data transfer. No software task is queued by the DBCOMP module when it is called by the XFER COMPLETE interrupt.

Self-Test (Figure 54). Three self-test software tasks are sequentially called by pressing the control panel test interrupt as shown in the figure. The sequencing of these tasks is performed by the executive TEST INTERRUPT handling routine. TESTDP and BIUTES perform tests on the DP1 and weapon bus hardware elements, respectively. CHKSU is a sequencing module which queues a test data generation module, SIMULT, at a simulated 100-Hz rate and controls the sequencing of the navigation and alignment computational functions at the corresponding rates. This sequencing module essentially replaces the normal system interrupts associated with IMU and reference data generation and transfers by queueing the appropriate modules under software control.

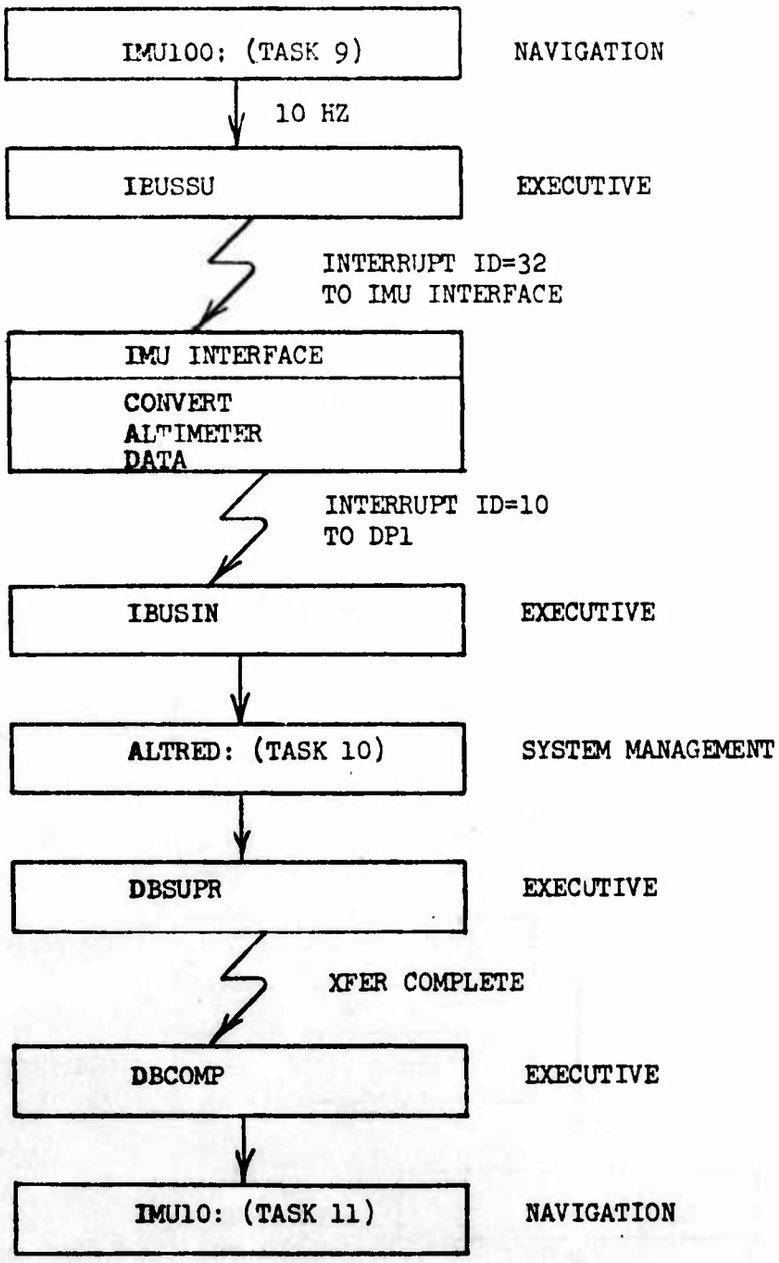


Figure 50. Altimeter Data Sequence

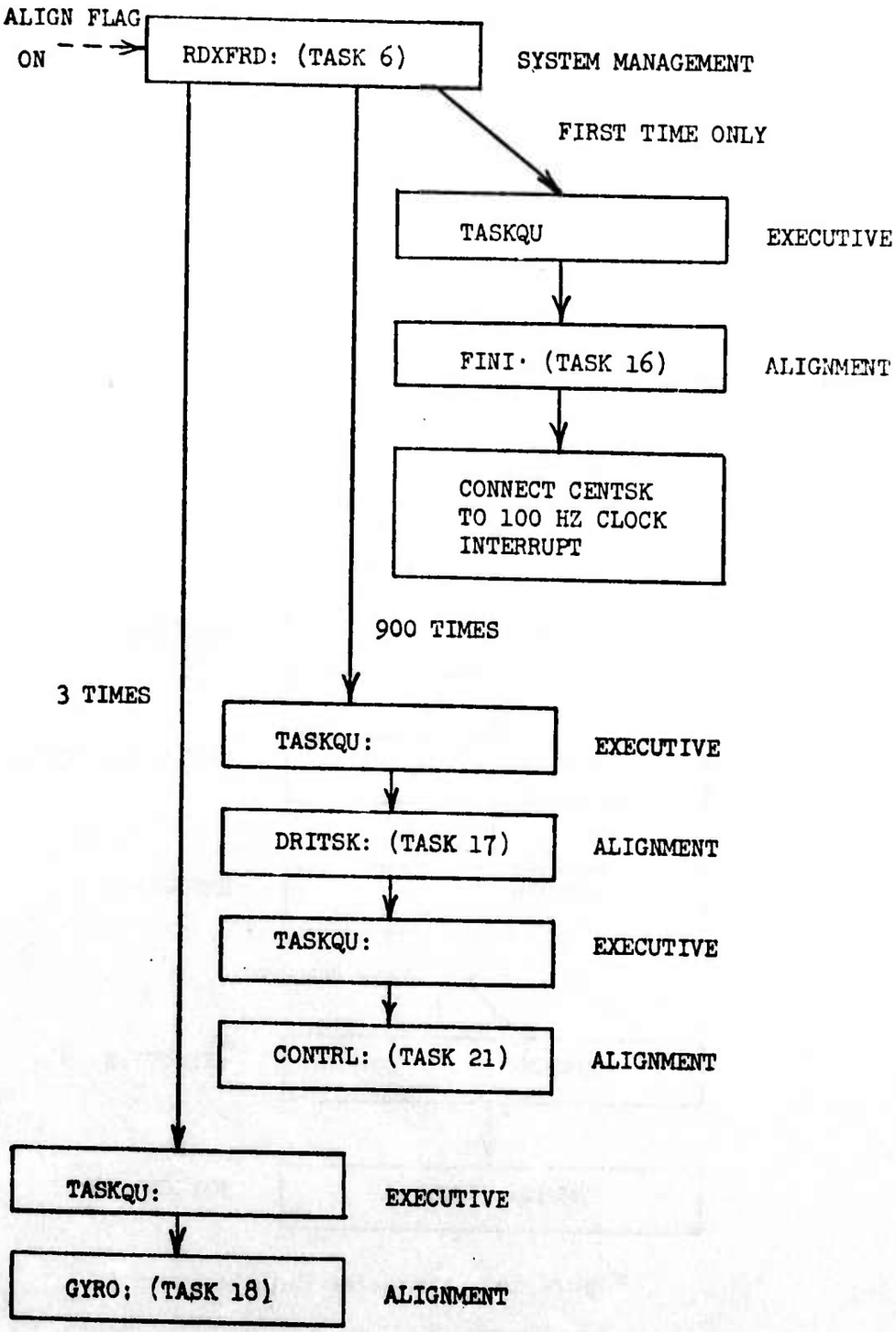


Figure 51. Alignment Initialization and Computation

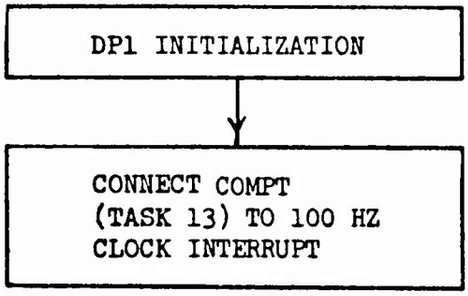


Figure 52. IMU Data Accumulation

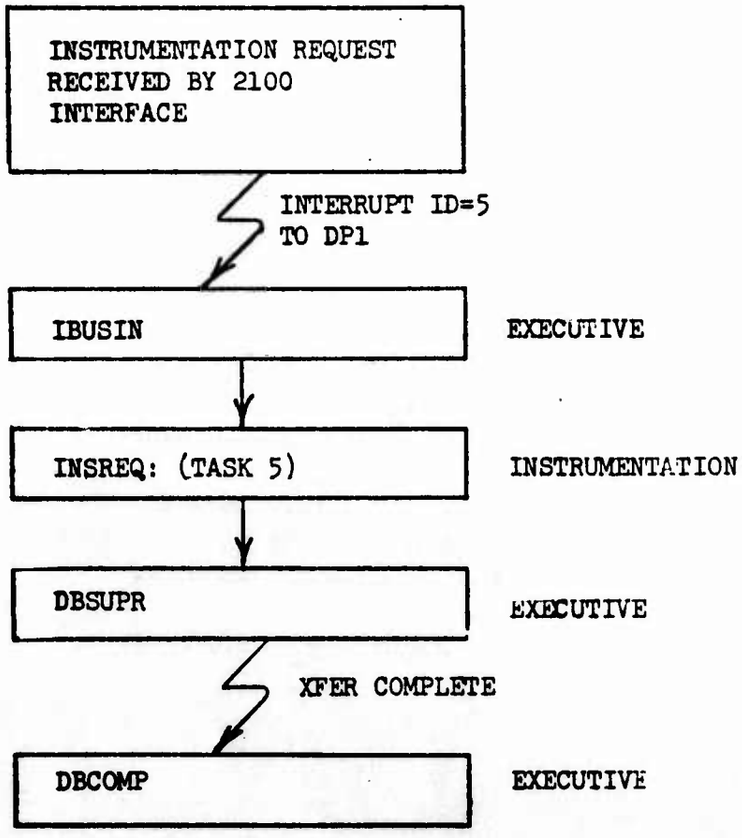


Figure 53. Instrumentation Data Transfer Sequence

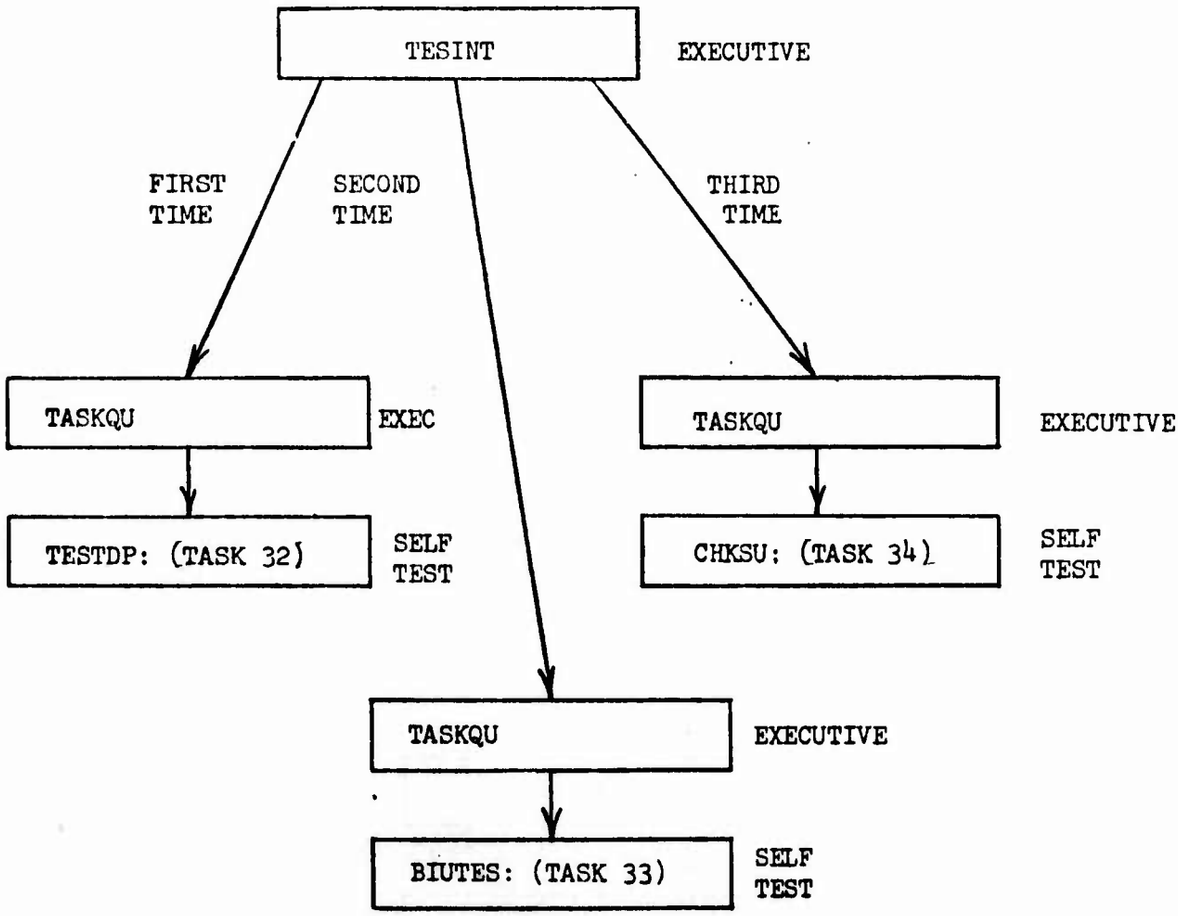


Figure 54. Self Test

Software Module Description. This subsection contains a brief description of each of the software modules applicable to the DP1 system. The executive software is discussed in subsection 2.2 and is not repeated in this subsection. A list of the software modules by task ID is shown in Table 4. In the following paragraphs, the software module descriptions are organized by the system functions.

System Management. Task 1 (TSINT) is queued up when the time sync interrupt is received from the HP2100. This occurs at a 1-Hz rate. This task stores the 100-Hz counter in the instrumentation buffer and sets a flag (REF-DATA-XFER) to indicate that a block of reference data is being received from the 2100. It also examines the ALIGN switch on the control panel and sets the ALIGN flag if the switch is up. This task queues up Task 23.

Task 2 (REFRED) is queued up when the reference data ready interrupt is received from the HP2100. This occurs at a 1-Hz rate approximately 50 ms after the time sync interrupt is received. If the REF-DATA-XFER flag is true this task starts the transfer of the reference data from the 2100 interface unit to the DP via the data bus. The completion of this transfer will queue up Task 6.

Task 4 (IMUCOM) is queued up by the completion of the IMU data transfer to DP1. This task checks the validity of the IMU data just received by comparison with maximum expected data magnitudes. Valid data is transferred to the appropriate operand memory locations, and invalid data is discarded (previous valid data is used). This task then queues Task 9 (IMU100).

Task 6 (RDXFRD) is queued up when the reference data from the HP2100 has been transferred to the DP. This transfer is started by Task 2 and occurs at a 1-Hz rate. This task queues Task 15 to transfer the reference data to the appropriate operand memory locations. When the navigate switch on the control panel is raised, this task starts the navigation routines running by queueing up Task 14 once and connecting Task 7 to the clock for execution at a 100-Hz rate. When the align switch on the control panel is raised, this task provides the sequencing for the alignment tasks by queueing up Task 16 once, connecting Task 22 to the clock for execution at a 100-Hz rate, queueing up Task 17, 900 times at a 1-Hz rate, and queueing up Task 18, three times at 5-minute intervals.

Task 7 (I100 Hz) is queued up by the clock at a 100-Hz rate. It requests data from the IMU. When the IMU responds, Task 8 will be queued up.

Task 8 (DATRED) is queued up when the IMU data ready interrupt is received from the IMU interface at a 100-Hz rate. This task starts the transfer of data from the IMU to the DP via the data bus. When this transfer is complete, Task 4 will be queued up.

Task 10 (ALTRED) is queued up when the altimeter data ready interrupt is received from the IMU interface at a 10-Hz rate. This task starts

TABLE 4. SOFTWARE MODULES

ID	TASK	PRIORITY	NAME
1	PROCESS TIME SYNC FROM 2100	8	TSINT
2	TRANSFER REFERENCE DATA TO DP	5	REFRED
3			
4	VALIDATE IMU DATA	8	IMUCOM
5	PROCESS INSTRUMENTATION DATA REQUEST	3	INSREQ
6	PROCESS REFERENCE DATA	5	RDXFRD
7	REQUEST IMU SENSOR DATA AT 100 HZ	8	I100HZ
8	TRANSFER IMU DATA TO DP	8	DATRED
9	PROCESS IMU DATA	8	IMU100
10	TRANSFER ALTIMETER DATA TO DP	6	ALTRED
11	PROCESS ALTIMETER DATA	6	IMU10
12	UPDATE SLOW NAV PARAMETERS AT 1 HZ	8	IMU1
13	ACCUMULATE IMU DATA	8	COMPT
14	INITIALIZE NAVIGATION PARAMETERS	6	IMUINT
15	SCALE REFERENCE DATA	6	RINIT
16	INITIALIZE ALIGNMENT PARAMETERS	6	FINI
17	PERFORM ALIGNMENT ALGORITHM	6	DRITSK
18	PERFORM GYRO BIAS CALCULATIONS	6	GYRO
19			
20			
21	UPDATE ALIGNMENT CONTROL FEEDBACK	8	CONTRL
22	PERFORM ALIGNMENT FILTER PROCESSING AT 100 HZ	8	CENTSK
23	STORE ALIGNMENT AND NAV. PARAMETERS AT TIME STROBE	8	TSYNC
24	INITIALIZE SELF TEST PARAMETERS	6	SIMINT
25	PERFORM 100 HZ SELF TEST CALCULATIONS	8	SIMULT
26			
27			
28			
29			
30			
31			
32	TEST DP FUNCTIONAL UNITS	5	TESTDP
33	TEST BIU'S	5	BIUTES
34	TEST NAVIGATION AND ALIGNMENT EQUATIONS	5	CHKSU

the transfer at altimeter data from the IMU interface to the DP via the data bus. When this transfer is complete, Task 11 will be queued up.

Task 15 (RINIT) is queued by Task 6 to transfer the reference data to the appropriate instrumentation data operand memory locations. This task also formats, rescales, and stores the reference data in operand memory for use by the navigation and alignment software modules.

Navigation. Task 9 (IMU100) is queued up by Task 4 at 100 Hz to process the IMU data. This task performs the IMU data compensation calculations, the quaternion (attitude) update calculations, and integrates the accelerometer data in the navigation frame to determine velocity and position at a 100-Hz rate. Navigation frame rotations are performed at a 10-Hz rate. The altimeter data sequence is initiated at a 10-Hz rate. This task also queues Task 12 at a 1-Hz rate.

Task 11 (IMU10) is queued when the altimeter data has been transferred to DP1 (10-Hz rate). This task performs gravity and coriolis compensation computations and uses the altimeter data for vertical channel damping. Latitude and longitude parameters are updated by this task.

Task 12 (IMU1) is queued by Task 9 at a 1-Hz rate. This task computes slowly varying navigation parameters which are used in Tasks 9 and 11.

Task 14 (IMUINT) is queued once by Task 6. This task initializes the attitude, velocity, and position navigation parameters to correspond to the reference data. Navigation software parameters, both computational and control, are also initialized to the appropriate state by this task.

Alignment. Task 16 (FINI) is queued once by Task 6 at the first occurrence of the REFERENCE DATA READY interrupt after the align flag is set. (This flag is raised by the system management software when one of the DP1 front panel flag switches, Switch No. 1, is set. Thereafter, the position of the switch has no effect.) FINI initializes the Kalman filter software parameters. FINI also serves one additional function: prefilter leveling, which is designed to reduce the initial tilt errors.

Task 22 (CENTSK) is queued by the clock service routine and contains all of the 100-Hz filter calculations. The sole purpose of CENTSK is to accumulate north, west, and vertical velocity increments with the scaling FS = 128 ft/sec. These, after approximately one second's accumulation, make up the rapidly-varying transition matrix elements. Because this module uses a vector variable which is updated at 100 Hz (by a navigation routine with priority 8), CENTSK must have priority 8.

Task 23 (TSYNC) is initiated by Task 1 as a response to the TIME SYNC interrupt, which signals the start of the reference data transfer. This is the earliest available time marker for the reference velocity and position values, and in order to make a meaningful comparison, the corresponding DP navigation parameters should be sampled at this time and put into safe storage. Since DP navigation software calls for latitude/longitude updates at

only a 10-Hz rate, a call to the update module LTNG is made at this time to ensure that the most current values are used. (No correction is made for the staleness of the reference data, that is, for the brief passage of time between the generation of the data and the start of the transfer to the DP.) TSYNC also performs two other functions: (1) transfers the accumulated velocity increments into the correct transition matrix elements, resetting the accumulators to zero, and (2) calculates the elapsed time since the last reference data transfer, so that a rather large variation in the nominal 1-Hz transfer rate can be accommodated.

Task 17 (DRITSK) is queued by Task 6 when the reference data set has been entirely transferred (all information needed in the Kalman filter update has been assembled). In order of occurrence, DRITSK does the following things: (1) Updates the slowly varying transition matrix elements, bypassing some of these when velocity-match only is selected. (Note: all of the rapidly varying elements are updated in the module TSYNC. Also, we actually use the matrix "D," that is, the transition matrix minus the 7 by 7 identity matrix.) (2) Updates the covariance matrix P to reflect the passage of time ΔT since the last measurement. (3) Symmetrizes the P matrix and adds 1 LSB process noise, Q, to the diagonal elements. (4) Tests for scaling headroom in the matrix P and the measurement noise covariance matrix R. If these can be shifted up, this is done, recording the cumulative shift count in the variable SCLCNT. (5) Calculates position and velocity errors, scaling them up to form single precision measurement vector elements. (6) Performs the measurement updates. (7) Rescales the single-precision state estimates to form the appropriate corrections to the navigation parameters. (8) Places CONTRL in the task queue to perform the corrections at the necessary higher priority. (9) Resets the state estimates to zero and returns control to the executive.

Task 18 (GYRO) is queued a total of three times: at the 300^{th}, the 600^{th}, and 900^{th} occurrence of the REFERENCE DATA READY interrupt after the ALIGN mode is entered (that is, after 5 minutes, 10 minutes, and 15 minutes have elapsed) by Task 6. In each case, the cumulative misalignment estimates are reset to zero, the Kalman filter is reinitialized, and DRITSK is called as a subroutine. On the second and the third entries, however, GYRO uses the accumulated misalignment estimates to estimate gyro biases and compensate for them. The general idea behind this calculation is that the misalignment which builds up during these periods is due to gyro drift; i.e.,

$$\beta_i = \delta_i \div (5 \text{ minutes})$$

where

$i = 1, 2, 3.$

The first 5-minute period is disqualified as a large initial misalignment may be present. GYRO also has the relatively low priority of 6.

Task 21 (CONTRL) is queued by Task 17 at high priority (level 8) to correct the navigation state. The first correction is to the navigation

quaternion q, in whose four elements all attitude information resides. Essentially what is done is to multiply by a correction quaternion, which is close to the identify (0, 0, 0, 1); actually, it is equal to (δ_1, δ_2, δ_3, 1 - $1/2\delta_1^2$ - $1/2\ \delta_2^2$ - $1/2\delta_3^2$) to second order in the misalignment angles δ_1, δ_2, δ_3. More precisely, we use the fact that the correction quaternion has the form $I + \Delta$; first $\Delta^* q$ is calculated, and the result added (with overflow safeguards) to q to form the corrected attitude quaternion. The next part of the control task is to correct the latitude, longitude, north velocity, and west velocity variables by subtracting from these (double-precision) the best estimates of their errors.

Instrumentation. Task 5 (INSREQ) is queued up when an instrumentation data request is received from the HP2100. This occurs at a 1-Hz rate. This task calculates the current position and velocity errors and transfers them along with the rest of the instrumentation data into the output buffer. Then it starts the transfer of the instrumentation data from the output buffer to the HP2100 via the data bus. No tasks are queued up at the completion of this transfer.

Task 13 (COMPT) is queued at 100 Hz by the executive clock service routine. When navigation is enabled, this task accumulates raw and compensated IMU data for instrumentation. This task also provides gyro bias compensation updating by matching compensated gyro rates to earth rate if enabled by the control panel switches.

Self-Test. Task 24 (SIMINT) is a self-test initialization module, providing initial values for position, velocity, and other kinematic parameters for the 100 Hz self-test driving function SIMULT. SIMINT is iterated once only at the start of the self-test processing. This task is queued by Task 34.

Task 25 (SIMULT) is the self-test driver, whose purpose is to compute simulated (100 Hz) IMU data and simulated (1 Hz) reference data. Since the self-test processing takes place much faster than real time, SIMULT is iterated at a rate much greater than 100 Hz (as are the 100-Hz navigation and alignment tasks also). SIMULT simulates (unrealistic) high-dynamic flight conditions so that the navigation and alignment modules are extensively exercised. Since only a checksum test is performed, the fact that these conditions are unrealistic is of no importance. This task is queued by Task 34.

Task 32 (TESTDP) is queued up the first time the test interrupt button on the control panel is pushed after a reset. It determines that the memory, arithmetic, and control boards are present in the DP.

Task 33 (BIUTES) is queued up the second time the test interrupt button on the control panel is pushed after a reset. It tests the BIU's to insure that they respond to command words.

Task 34 is queued up the third time the test interrupt button on the control panel is pushed after a reset. It tests the DP hardware and software

by executing the navigation and alignment tasks for a simulated 60 seconds using known inputs. Then it computes a checksum over part of operand memory and compares it to a precomputed checksum. This task queues up Tasks 11, 14, 15, 16, 17, 18, 22, 23, 24, and 25.

SYSTEM TESTS

This subsection reports the results of tests which were performed on various combinations of hardware and software elements of the DP1 system. These tests range from detailed testing of individual software functions in a laboratory environment through complete system testing in a flight environment.

Expected Navigation Performance

The required level of navigation performance was not specified for the DP1 system. However, a design goal of 1600 feet (1σ) in each axis at the end of a 10-minute free navigation period following system alignment was used in the development of system software. The principal error sources to be considered are IMU parameter deviations from nominal values, tilt misalignment, and software computation error (algorithm truncation, round off, limited precision, etc.). In lieu of a detailed error allocation to the elements of the system software, the design philosophy was to make the computation error contributions to total navigation error small. The measured computation errors are presented in the following section.

For several reasons, the performance of the transfer alignment software could not be quantitatively predicted. First, the superficial structure of the filter (number of states, measurement model, etc.) was derived, developed, and verified by covariance analysis, a main-frame computer program which measures the performance of the filter relative to a more elaborate truth model. No truth model of manageable size, however, can be complete, and so the performance measure is probably optimistic to some slight but indeterminate degree.

Worse yet, the fixed-point implementation in DP assembly language forced some unforeseen structural deformations in the filter, the impact of which could not be assessed—by covariance analysis, or otherwise—in the short time scale allowed for software development. The process noise, for example, which optimized the covariance analysis filter model could not be faithfully reproduced in the fixed-point version; this is discussed in some detail in the DP1 alignment software documentation. The filter iteration rate used in the covariance analysis was 6 Hz; in the finished software, however, a more promising rate of 1 Hz was adopted. Moreover, the gyro bias estimation feature was added as an afterthought and though it assumed a very reasonable and convincing form in the finished software, no verification of its performance (or the theory of operation, for that matter) existed beforehand, either in the literature or in the form of a simulation.

Using the covariance analysis as a guide, and some rough calculations of filter sensitivity to gyro bias, a set of residual estimation errors

were promulgated which, if actually approached by the DP1 alignment software, would constitute success, These are:

Tilt misalignments	1 arc-minute
Azimuth misalignment	3 arc-minutes
Gyro bias	0.2 degree/hour

. Since the uncompensated gyro biases expected in the Hamilton-Standard 3030 IMU are fully an order of magnitude greater than the last figure (2 degrees/hour as opposed to 0.2 degree/hour), the untested gyro bias estimation module appeared to be a good gamble.

System navigation performance is shown as a function of tilt misalignment and gyro bias in Figure 55. The effect of the azimuth misalignment error is dependent on the azimuth maneuvers during free navigation. If the average azimuth maneuver during free navigation were 1g then the navigation error due to this source would be equivalent to that shown for tilt misalignment. In the absence of an azimuth maneuver, azimuth misalignment error only produces a second order effect on navigation error through its coupling of earth rate into the attitude parameters.

Laboratory Tests

A series of laboratory tests were performed both at Hughes and at CIGTF. The emphasis in the laboratory tests at Hughes was on the verification of the software modules and the integration of the software with the Hughes-developed hardware and the Hamilton Standard IMU. The CIRIS interface with this equipment was simulated during this phase of testing. Tests involving the IMU at Hughes were primarily static although limited dynamic conditions were used in some tests.

The laboratory tests at CIGTF were concerned with integration of the HP2100 computer (CIRIS simulation) with the other system elements, and system level tests with the IMU both in a static environment and using coning motion. Subsequent paragraphs provide the detailed results of the laboratory tests.

Software Module Tests. The executive software modules were extensively tested prior to their integration with the other system software. Executive software testing during this phase involved the use of test drivers which simulated the system software modules interfacing with the executive. These test drivers exercised both the executive software and the system hardware elements in a high stress environment. The test environment was much more stringent than the actual system operating environment and, thus, ensured adequate capability to accomplish the system testing.

The following navigation and alignment software module tests also involved the use of test drivers to simulate the operations of the other system software and hardware elements. These tests, therefore, did not use the weapon bus, interface hardware, or the executive software.

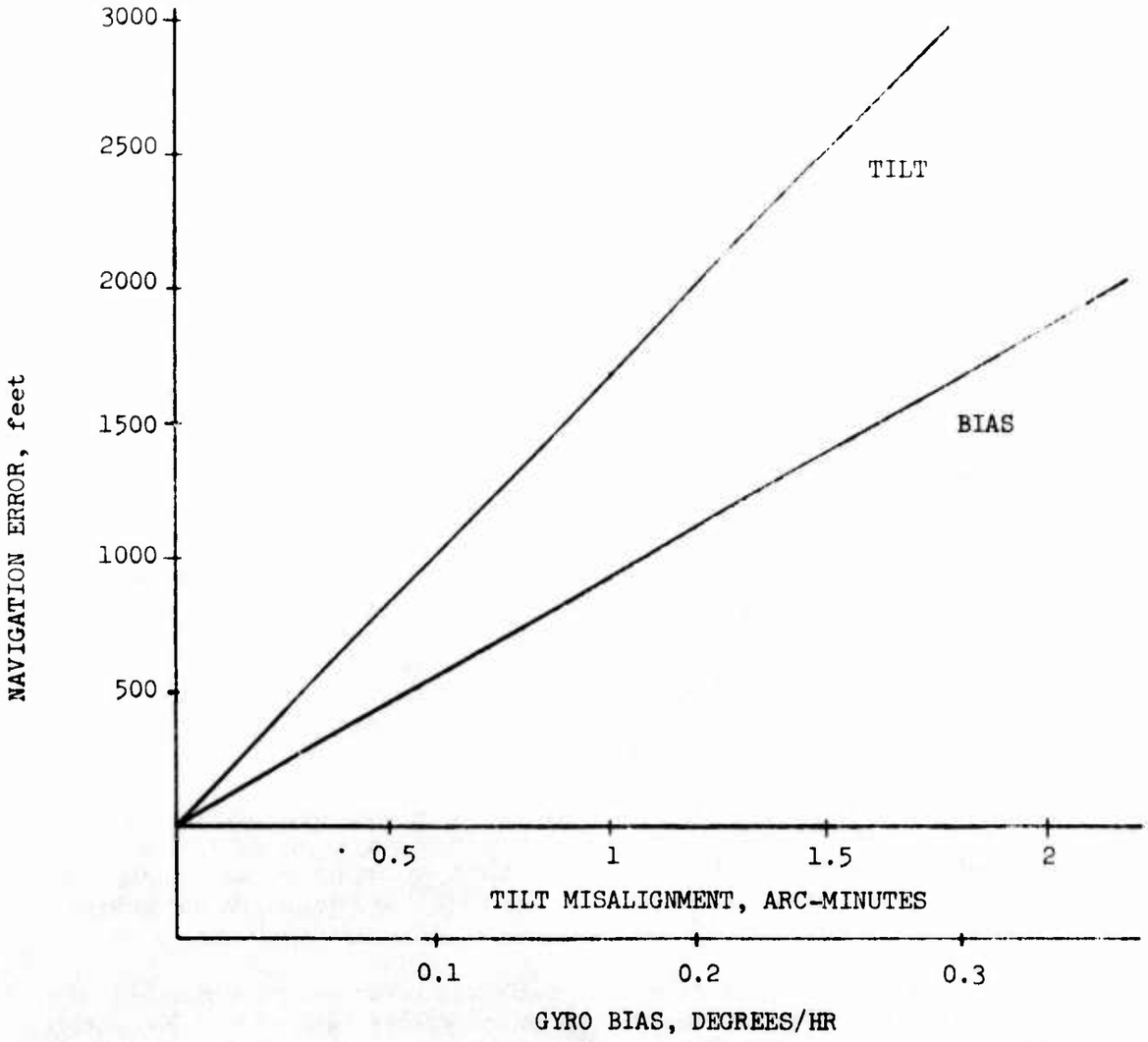


Figure 55. Navigation Error After 10 Minutes

Quaternion Update/Direction Cosine Matrix (Portion of Task 9). The attitude determination portion of the navigation software was driven using simulated coning motion as well as high constant angular rates about one axis. The constant rate input about one axis produced no error and merely checked scaling and gross code errors. Coning motion exercised the quaternion algorithm so that a good accuracy assessment could be made. Test results showed that the adaptive (dynamic scaling) algorithm is very accurate. Figure 56 shows the Z axis error in the direction cosine matrix versus time during coning motion,

$$\omega_x = A \sin(\omega t)$$

$$\omega_y = A \cos(\omega t)$$

$$\omega_z = 0$$

with A = 0.4575695 rad/sec, ω = 20.0951483 rad/sec. This corresponds to a coning half angle of 1.3 degrees at a coning frequency of 3.2 Hz. Drift rate for this test case was 0.17 degrees/hour. A 36-bit floating point machine showed a 0.07 degree/hour drift using a direction cosine algorithm with the same forcing function. Less violent motions than the above will have considerably less computation induced drift than 0.17 degree/hour.

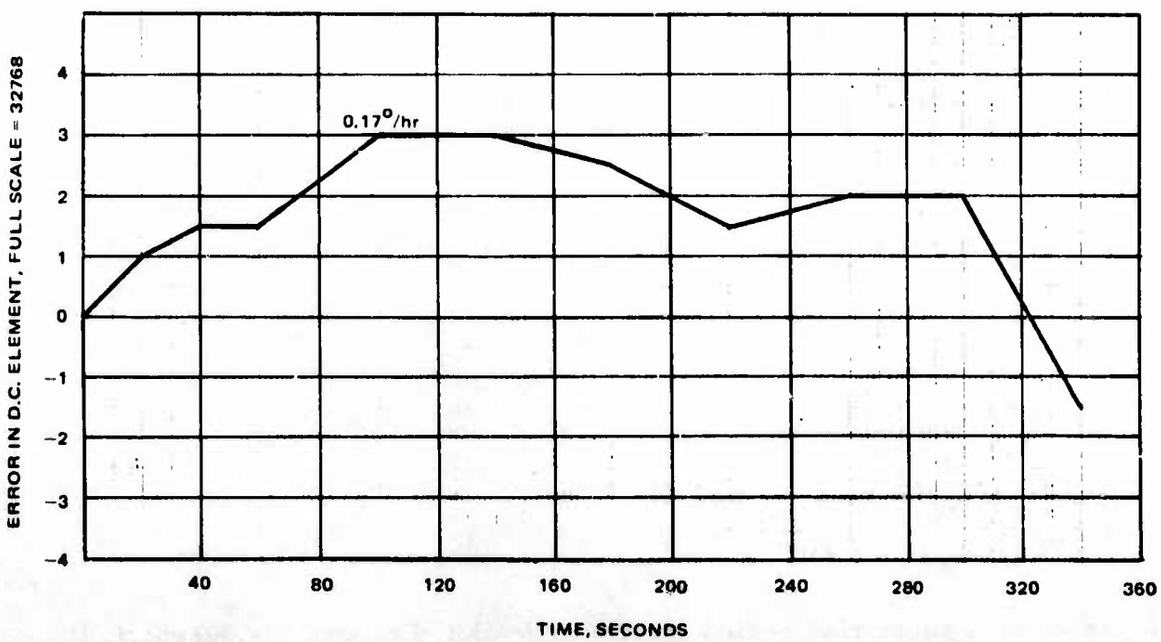


Figure 56. Coning Motion Z Axis, $\frac{C_{21}+C_{12}}{2}$, Error Relative to Analytic Solution

Velocity and Position, Gravity and Coriolis Corrections, Navigation Frame Rotations (Portions of Tasks 9 and 11). These blocks of code were checked against analytic results. The differential equations solved for checkout were contrived (that is, simplified) in the case of the Gravity and Coriolis Corrections and the Navigation Frame Rotations. Upper bounds on the calculation biases include 0.01 degree/hour in the Navigation Frame Rotations, 1.5×10^{-4} ft/sec^2 in Coriolis Corrections, while calculation scale factor errors cause 0.008 percent error in latitude movement and ~0.005 percent error in longitude movement. Small statistical variations occur in the latitude and longitude calculations with a simplified analysis showing an error with a standard deviation of 26.6 feet in 5 minutes. Some of the calculation error sources could be fixed but were not fixed since IMU randomness easily swamps all of them.

As an example of the checkout analysis performed, the following test of the coriolis and latitude/longitude code was made. With the body accelerations and vertical velocity forced equal to zero, the navigation equations simplify to

$$\dot{V}_N = V_W(\dot{L} + 2\Omega) \sin\lambda$$

$$\dot{V}_W = - V_N(\dot{L} + 2\Omega) \sin\lambda$$

$$\dot{\lambda} = \frac{V_N}{\rho_L}$$

$$\dot{L} = \frac{-V_W}{\rho_\lambda} \cos\lambda$$

where

V_N = north velocity

V_W = west velocity

λ = latitude

ρ_L = longitude

$\rho_L = R_e(1 - e, 2 - 3 \sin^2\lambda)$

$\rho_\lambda = R_e (1 - e \sin^2\lambda)$

R_e is earth's equatorial radius (20925695 feet) and e, earth's eccentricity (1/298.3). The differential equations for V_N and V_W are still too complicated for solving. By NOP'ing certain lines in the G&C code, V_W is forced to zero and the equations become

$$\dot{V}_N = V_W \left(\frac{-V_W}{\rho_\lambda \cos\lambda} + 2\Omega\right) \sin\lambda$$

$$\dot{V}_W = 0$$

Letting λ and L be initially zero and then noting that λ will not change appreciably over a few minutes,

$$\dot{V}_N \approx V_W\left(\frac{-V_W}{\rho_\lambda(0)} + 2\Omega\right)\lambda \approx V_W\left(\frac{-V_W}{\rho_\lambda(0)} + 2\Omega\right)\frac{\int V_N}{\rho_L(0)}$$

Letting

$$\omega^2 = -V_W\left(\frac{-V_W}{\rho_\lambda(0)} + 2\Omega\right)\frac{1}{\rho_L(0)}$$

the differential equation satisfied by V_N becomes

$$V_N + \omega^2 V_N = 0$$

The solution to this equation is

$$\ddot{V}_N = V_{N_o} \cos t \qquad V_{N_o} = V_N(t = 0)$$

S $\qquad\qquad \dot{V}_N(t = 0) = 0$ since $\lambda = 0$ at $t = 0$

and then

$$\lambda = \frac{V_{N_o}\int_o^t \cos\omega t}{\rho_L} = \frac{V_{N_o}\sin\omega t}{\omega\rho_L}$$

$$L = \frac{-V_W t}{\rho_\lambda(\cos\lambda)_{average}} .$$

With the initial conditions V_W =-2048 ft/sec, V_N = 2048 ft/sec, L - λ = 0 degrees (large velocities were desired to see the small coriolis effects), measured and analytic results are shown in Table 5. The velocity error is very small and is easily accounted for by the worst case 1.5×10^{-4} ft/sec^2 known error ($1.5 \times 10^{-4} \times 300 = 0.045$ ft/sec). However, the small V_N error

and no V_W error does not account for the error in latitude and longitude. A detailed analysis reveals the error sources to be calculation limitations in forming ρ_L^{-1} and $-\rho_\lambda^{-1} \cos(\lambda)^{-1}$. At $\lambda = 0$ degree, ρ_L^{-1} is 0.008 percent too small and $\rho_\lambda^{-1} \cos(\lambda)^{-1}$ is ~ 0.0045 percent too small. The error in ρ_L^{-1} explains the latitude error but the error in $\rho_\lambda^{-1} \cos \lambda^{-1}$ is smaller and opposite in sign to the measured error. A detailed examination of the code with the specific west velocity of -2048 ft/sec revealed a rounding error peculiar to the longitude channel of -0.0142 percent. A code statement was subsequently added, but no further verification was warranted.

TABLE 5. EXPERIMENTAL VERSUS ANALYTIC RESULTS AFTER 5 MINUTES (299.9 SECONDS)

VARIABLE	MEASURED	ANALYTIC	ANALYTIC – MEASURED	NORMALIZED ERROR (%)
V_N	2045.7803 FT/SEC	2045.7888 FT/SEC	0.0085 FT/SEC	4.15×10^{-4}%
λ	0.02953617 RAD.	0.02953873 RAD.	$2.56\ 10^{-6}$ RAD. OR 53.5 FT	0.0087%
L	0.02935975	0.029355514	-4.24×10^{6} RAD. -88.6 FT	-0.0144%

Alignment Filter (Tasks 16, 17, 22). An abbreviated driving routine was written in DP1 assembly language for the specific purpose of testing the alignment software package (1) within the processor itself (i.e., an on-line test), and (2) in the absence of the executive and navigation software functions, which were themselves undergoing stand-alone tests. Initial values for tilt misalignments (δ_N, δ_W) about the North and West axes were postulated in the driver, and navigation errors were calculated by the deterministic equations:

$$\text{North velocity error} = -gt \sin(\delta_W) \cong -gt\, \delta_W$$

$$\text{West velocity error} = +gt \sin(\delta_N) \cong gt\, \delta_N$$

$$\text{Latitude error} = -\frac{1}{2} gt^2 \sin(\delta_W)/Re \cong -\frac{1}{2} gt^2\, \delta_W/Re$$

$$\text{Longitude error} = -\frac{1}{2} gt^2 \sin(\delta_N)/Re \cos(\lambda), \cong -\frac{1}{2} gt^2\, \delta_N/Re \cos(\lambda)$$

where

g = gravity = 32.2 ft/sec^2

Re = Earth equatorial radius

λ = latitude ;

the driver then called the alignment modules to estimate misalignments, reducing the initial misalignments by the estimated values. This process was iterated many times, calculating navigation errors from the residual misalignments and calling the alignment filter.

Assuming initial tilts of 3 degrees to 5 degrees, it was found that the prefilter leveling routine reduced these errors to several arc-minutes, and fewer than 20 iterations of the Kalman filter further reduced them to a stable limit of less than 1 arc-second. (Azimuth misalignment estimates were consistently nil, which was the correct answer for these tests.) Of course, these impressive results were not indicative of the ultimate performance of the alignment function, when operating in a noisy environment and with many other navigation disturbances. All that was convincingly demonstrated by these tests was that there was no gross logical error present in some of the filter modules, and that the package was in fact endowed with a capacity for misalignment estimation. The next step was to test the alignment function in the integrated software package.

Gyro and Accelerometer Compensation (Portion of Task 9). The compensation software was checked by static testing with the IMU. The test set-up is shown in Figure 57. Special test software was used to drive the executive (for data formatting and bus transfers) and to accumulate IMU data. By accumulating raw and compensated gyro and accelerometer data for a fixed time interval in the positions X axis up and down, Y axis up and down, Z axis up and down, the compensation for accelerometer and gyro bias, gyro input and spin axis unbalance, and accelerometer scale factor could be verified.

A typical Z-axis-up laboratory data sheet is shown as Figure 58. The IMU is warmed up on a fairly level table with Z-axis-up and compensated and uncompensated gyro and accelerometer outputs are accumulated double precision for 100 seconds and then recorded. A rotation of 180 degrees about the vertical (Z in this case) axis is made, and the 100-second accumulation is repeated. Averaging of the 0 degree and 180 degrees azimuth data removes the Earth rotation rate azimuth uncertainty as well as table tilt uncertainties. With the Hamilton Standard sensor axes defined as in Figure 59 and Z^M pointing up, the accelerometer and gyro averages measure the following forces and rates over a 100-second period:

A_X = X accelerometer bias + g (tilt about Y axis of IMU body with respect to table)

A_Y = Y accelerometer bias + g (tilt about X axis of IMU body with respect to table)

A_Z = g + Z accelerometer bias

ω_x = X gyro bias

ω_y = Y gyro bias

ω_z = Z gyro bias -g· (Z gyro Spin Axis Unbalance) + Vertical component of Earth Rate

Performing the accumulations in all the various positions and then averaging and rescaling, a table, such as that shown as Table 6 here, can be generated.* In Table 6 raw and compensated gyro azimuth averaged accumulations are listed in degrees/hour. In the third numerical column, the measured raw data is differenced from the appropriate Hamilton Standard compensation, e.g., the first row entry is X gyro raw data (-27.85 degrees/hour) - HS calibrated X gyro bias, the second is Y gyro raw data (37.69 degrees/hour) - HS calibrated Y gyro bias and the third is Z gyro raw data (13.18 degrees/hour) - HS calibrated Z bias + g (HS calibrated spin axis unbalance). That the compensated data closely agrees with the Raw-HS column means that this part of the compensation code is correct and the HS calibration constants have been correctly converted. The Gyro Error column gives the system error. With the correctly calibrated compensation, the IMU axes should measure 0 degree/hour or else 8.16 degrees/hour, the vertical component of Earth rate.

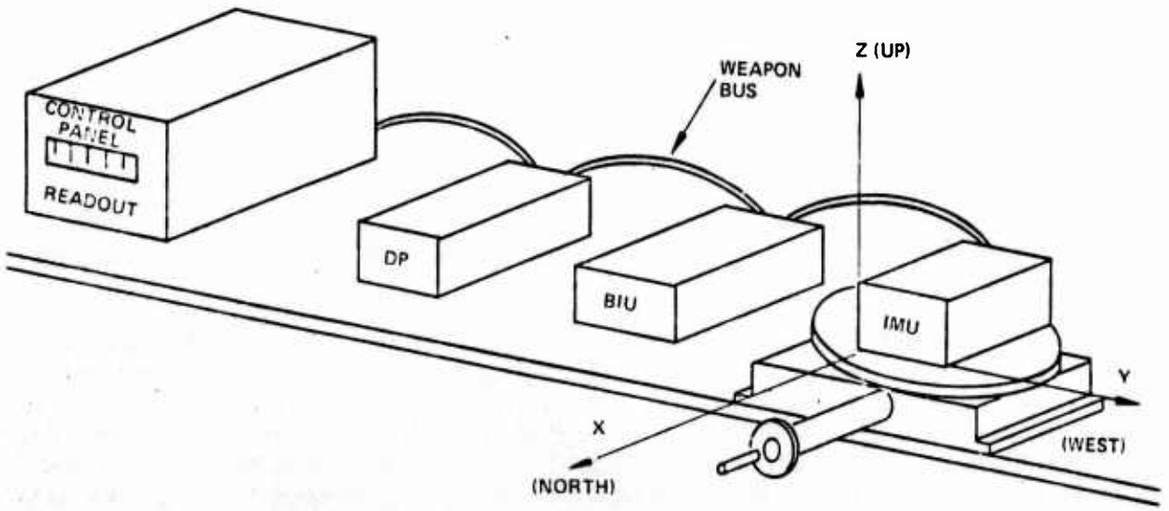


Figure 57. Compensation Software Test Setup

*Although the data used in this table was taken at Holloman, numerous tests identical to this were made at Hughes. These tests were required every time a new HS calibration was made in order to check out the new software constants. In addition, many tests were made just to monitor IMU performance and make sure the HS calibration was reasonable.

Accelerometer bias and scale factor (gravity constant is known) can also be verified with these tests. The gyro scale factor can be verified by making an accurate 180 degrees rotation during a 100-second accumulation period. The compensation not checkable by the fairly crude Hughes test setup included gyro and accelerometer sensor to body axis misalignments as well as the pseudo coning constant. In these cases, it had to be sufficient to visually check the code against the desired equations a number of times.

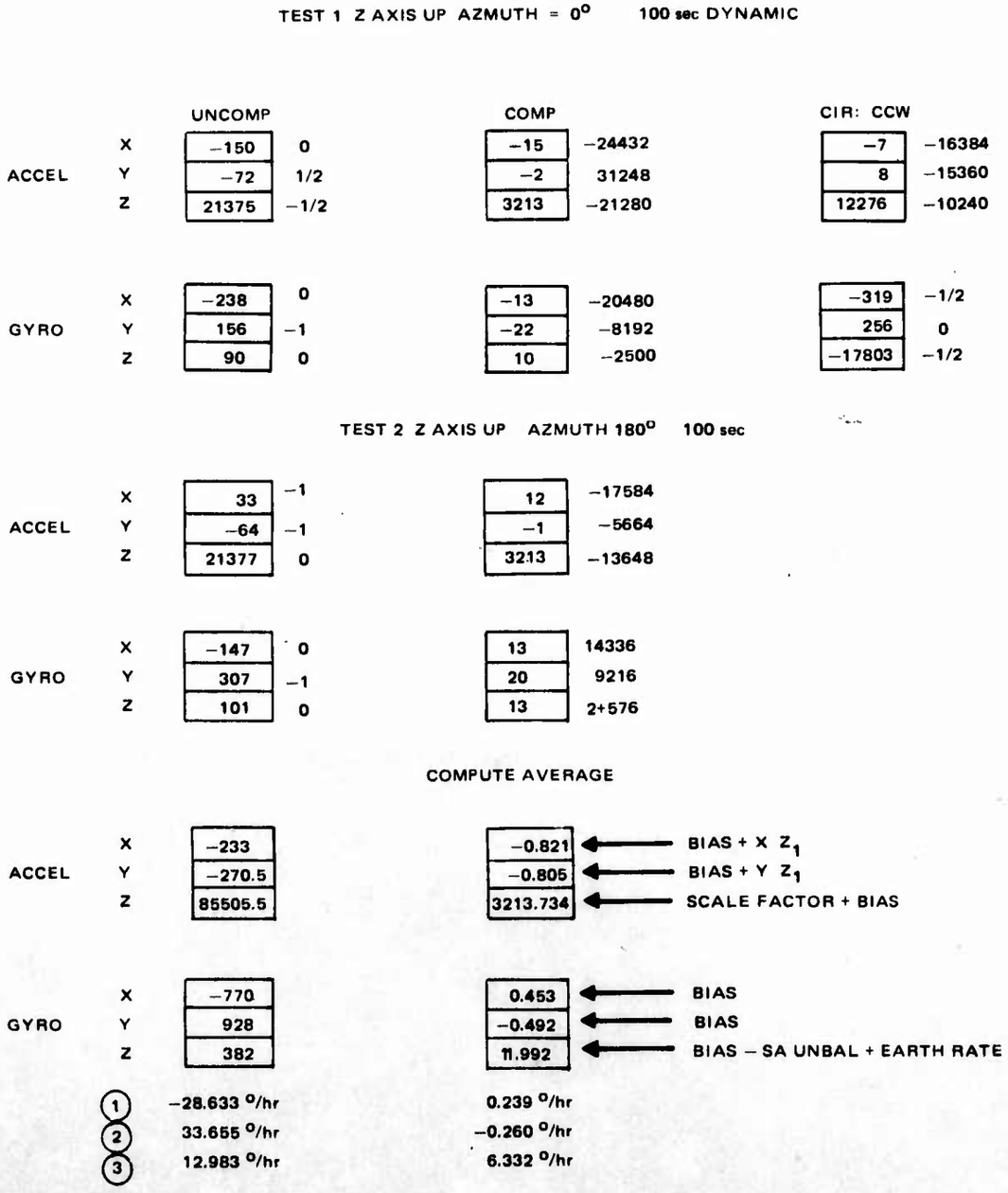
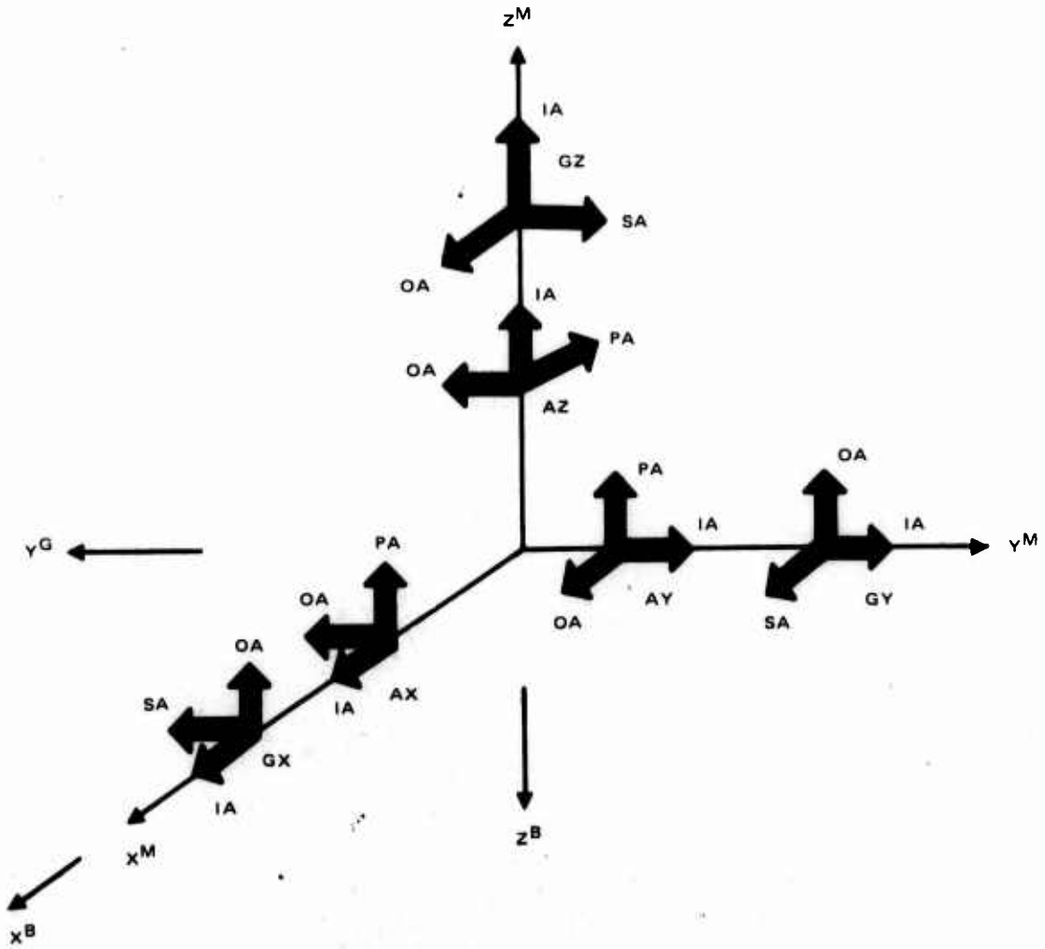


Figure 58. DGWT Laboratory Tests



BODY AXES ARE DEFINED AS NORMAL TO CUBE FACES

Figure 59. HS 3030 Sensor Axes

TABLE 6. GYRO COMPENSATION ANALYSIS

MEASUREMENT POSITION		QUANTITY MEASURED	RAW DATA VALUE (°/hr)	COMP. DATA VALUE (°/hr)	RAW – H. S. (SHOULD EQUAL COMP.)	GYRO ERROR °/hr
	X	B_X	−27.85	−1.24	−1.29	−1.24
Z↑	Y	B_Y	37.69	0.21	0.21	0.21
	Z	B_Z−SA+ER	13.18	11.46	11.47	3.31
	X	B_X	−27.47	−0.875	−0.91	−0.875
Z↓	Y	B_Y	37.35	−0.12	−0.13	−0.13
	Z	B_Z+SA−ER	−9.98	−12.39	−12.39	−4.23
	X	B_X−IA	−25.11	−0.19	−0.26	−0.19
Y↑	Y	B_Y −SA+ER	28.43	0.825	0.853	−7.335
	Z	B_X+IA	7.48	−0.536	−0.54	−0.536
	X	B_X+IA	−33.52	−5.18	−5.25	−5.18
Y↓	Y	B_Y+SA−ER	38.855	−8.515	−8.53	−0.355
	Z	B_Z−IA	−4.23	−0.33	−0.33	−0.33
	X	B_X−SA+ER	−17.81	2.266	2.17	−5.894
X↑	Y	B_Y+IA	48.95	−5.285	−5.28	−5.285
	Z	B_Z	2.04	−0.025	−0.02	−0.025
	X	B_X+SA−ER	−40.30	−7.13	−7.16	1.03
X↓	Y	B_Y−IA	22.975	2.26	2.245	2.26
	Z	B_Z	0.54	−1.53	−1.52	−1.53

NOMENCLATURE: B – BIAS (°/hr), SA – SPIN AXIS UNBALANCE (°/hr/g)
IA – INPUT AXIS UNBALANCE (°/hr/g), ER – EARTH RATE

System Integration Tests (Hughes). This portion of the laboratory tests at Hughes involved the integration of all system software functions – executive, system management, navigation, alignment, instrumentation, and self test – with the system hardware elements. Since the HF2100 computer of the CIRIS was not available, its functions were simulated either by hardware or software depending on the type of test being performed.

In integrating the executive, navigation, and alignment software functions, there ensued a brief period of struggle with incorrect calling sequences, improper relative priorities, and, in general, the inevitable mismatches which result when programs written by more than one person are merged. The resolution of these problems was straightforward, and once the integrated software appeared to be functioning in some fashion, a new period of testing began, with the object being to determine whether the software was functioning properly, and if so, how well. This period was considerably protracted, in part because the processor itself had some residual hidden faults and developed more along the way and also, in part, because of a lack of instrumentation for measuring performance quantitatively.

The integration involved two fundamentally different types of experiment. One required the development of an assembly-language (on-line) simulation program, which was to reside within the processor and generate simulated IMU outputs and simulated reference data. Initial values for misalignment and gyro bias were selectable, and the program could simulate a static test (equivalent to sitting still on the Earth's surface), level flight, or flight test complete with three 3-minute S-turn maneuvers. The other experiment – true static alignment – required using the actual outputs of the IMU, with constant reference data being transmitted to DP1 from a test box (HP2100 simulation).

Static alignment tests of the second type were sporadically performed throughout the integration period; the results of these tests were ambiguous for a considerable period. The alignment filter uses three periods of 5 minutes each to remove initial tilt misalignments (first period) and to compensate for estimated gyro bias (next two periods). At the end of each gyro bias estimation period, the alignment filter control feedback corrects the gyro bias parameters in the navigation compensation software. The expected result of executing the control feedback is a monotonic decrease in residual gyro bias. Initial static alignment tests showed an apparent discrepancy in the control feedback gyro bias gain between the X and Y gyros. The results of the second bias estimates were always less than the magnitude of the first estimates, indicating convergence of the estimator. This type of behavior was apparent in all static alignment tests and detailed examination of the software involved did not result in any changes.

At this point, it was decided to perform tests using the simulation software routine which simulates a perfect IMU (except for preset misalignment and bias errors) and the reference system. As a first step, the simulation set-up was for zero velocity to duplicate the static alignment with the IMU. The static alignment tests using the simulation showed proper operation of both the navigation and alignment filter and the residual bias was reduced to 1, 5 percent of the initial value. Correct results were also obtained with the simulation operating at a constant velocity of 600 fps. However, these results were obtained only after disabling the position matching in the alignment filter (only velocity matching was used). The symtomatic behavior of the alignment filter with both position and velocity matching was the generation of a large azimuth misalignment estimate, which should not occur in this type of testing since no lateral acceleration was being applied. The cause of this problem was found to be a persistent position error rate when the corresponding velocity error was essentially zero. A position error rate with no velocity error can be due to (1) inadequate accuracy in the scale factor which converts a distance increment to a Lat/Long increment and/or 2) truncation error in determining the distance increment. Attempts to null the position error rate by changing the scale factor in the simulation program were not successful. The most probable cause was lack of precision in determining the position increment in the simulation since this routine is performed at 100 Hz. Subsequent testing of the alignment filter proceeded using only velocity matching in order to minimize the number of potential error sources.

As previously mentioned, static alignment tests do not allow verification of the azimuth misalignment estimate since the only driving function in these tests is the inaccurate compensation of sidereal rate due to coordinate system rotation. A lateral maneuver is required to obtain proper performance of the azimuth channel. The simulation software allows a maneuver to be simulated. With the maneuver, a number of problems arose, all involving matching the simulation results with the corresponding navigation outputs under error-free conditions. One source of these problems was found to be a defect in the navigation quaternion update algorithm under large dynamic conditions, which was corrected.

Since static alignment is proper using the simulation program, investigation of the IMU static alignment problem was initiated by attempting to determine what is different between the simulated IMU and the actual IMU environment. One possibility involved the noisy nature of the IMU outputs rather than the well-behaved deterministic simulation. A second difference is that the simulation tests bypass the navigation IMU compensation software. These possibilities were investigated by first adding a noise sequence to the simulation output and then testing the compensation software with a noise sequence. Proper performance was found in both cases. Next, the possibility that the gyro biases were not constant with time were investigated.

All IMU gyro bias measurements were made after an adequate stabilization period. Initial tests showed variations of 1^{o}/hr peak-to-peak using a series of 100-second averaging intervals spaced at approximately 5-minute intervals. Neither the navigator nor the alignment software were operating during these tests. The next series of tests used a series of consecutive 5-minute averaging periods within a 30-minute period. Both the navigator and alignment filter were operating during these tests, but free navigation was not performed. By running the alignment software and additional test software, the performance of the alignment filter in estimating gyro bias and the response of the compensation software to the bias estimates were independently verified. The longer averaging period reduced the peak-to-peak bias variation to 0.3 - 0.4^{o}/hr. In general, the results of these tests indicated that the alignment filter estimate of gyro bias becomes more accurate when the actual gyro bias is stable with time. In fact, the alignment filter estimate error was greater than the peak-to-peak variation in actual bias, indicating a probable dependence of filter performance on the higher frequency components of the gyro bias. This dependence could not be verified in our tests due to inadequate instrumentation facilities.

To aid in diagnosing the effect of bias variations on alignment filter performance, the instrumentation software for CIGTF testing was modified to allow post-testing analysis. Rather than only instrumenting the last IMU samples prior to the instrumentation request, all samples of both raw and compensated IMU data are accumulated during the period between instrumentation requests and the results will be output. This data may then be analyzed to determine variations in IMU performance over varying time intervals and the response of the alignment filter to these variations. It should be noted that the accumulation period is matched (approximately) to the iteration period of the alignment filter.

During the course of subsequent testing, a number of software and hardware faults, as well as procedural errors, were discovered and corrected; these are enumerated below. Which of these were present and detracting from earlier performance is not precisely known.

(1) Extraneous clock interrupts were found to be occurring, due to a noisy inverter in the clock divide chain. Some of the software tasks were therefore being queued up too often and at odd times.

(2) A tri-state multiplex chip in the IMU interface circuitry was found to be bad; although it was discovered in a stable failure mode, it may be that intermittent failure was occurring earlier.

(3) During simulated alignment maneuvers the filter convariance matrix sometimes becomes increasingly nonsymmetric and the misalignment estimates diverged catastrophically. This type of nonsymmetry injection was not foreseen, and was immune to the symmetry-protection features of the software. A brute-force symmetrization routine was added to prevent any recurrence.

(4) A source of inaccuracy in the gyro-bias estimation was also found and purged with a resulting small improvement in performance. The original design accumulated misalignment estimates at a high (8x) scaling for use in bias estimation. However, the incremental estimates were shifted down – as a fail-safe hedge against overflow problems – before being used to correct the navigation attitude quaternion, with a consequent precision loss affecting the accumulated estimate. After ascertaining that overflow was not occurring, the shift was permanently removed.

(5) In order to imitate the temporal spacing of the TIME SYNC and DATA TRANSFER COMPLETE interrupts from the HP2100, the DP1 test box used INTERRUPT 0 (a null task) transmissions as spacers. The unexpected result of each such transmission was that, when execution of an interrupted task was resumed, further interrupts were masked off by the system management until execution was completed. A number of other timing schemes were later used which did not have this defect.

(6) It was not recognized at first that the HP2100 data transfer interrupt spacing was inappropriate for the simulation tests. This is because the simulation software updates the reference data at a 100-Hz rate, requiring that the reference and navigation states be sampled with no more than 100-msec intervening delay. Some improvement in performance was evident after this was ensured.

At the end of this integration period, static alignment with the IMU appeared to be satisfactory; that is, tilt misalignment and horizontal (X- and Y- axis) gyro bias estimates were reproducible, and successive bias estimates appeared to converge rapidly. Without some independent means of determining misalignments or gyro biases, the static alignment accuracy could not be

quantitatively measured. When simulating the conditions of static alignment, it was found that residual tilt misalignments are typically less than one arc-minute and residual gyro biases less than 0.1°/hr. When flight conditions with alignment maneuvers are simulated, the typical performance may be summarized:

0.6 arc-minute tilt misalignment

5 arc-minutes azimuth misalignment

0.3°/hr bias, Y- and Z-axis gyros

0.9°/hr bias, X-axis gyro.

The azimuth alignment and X-axis bias compensation accuracies indicated are considerably worse than expected; however, it is quite likely that this inaccuracy was, in large measure, an artifact of precision losses in the simulation software. The large residual X-axis (roll axis) bias, in particular, appeared to be due to the rather large roll rates encountered in simulated alignment maneuvers, which tended to aggravate the simulation precision loss problem. Since the system software was performing substantially as expected and a major design effort would be required to improve the simulation software precision, the system was shipped to CIGTF for further integration tests.

System Integration Tests (CIGTF). The initial integration period after system delivery to CIGTF was involved with installation of the equipment in the racks and interfacing with the HP2100 computer. All interface problems concerning the HP2100 computer involved hardware incompatibilities or HP2100 software. Two problems were discovered and corrected in the DP1 instrumentation software during this period (one parameter omission, one data formatting).

In the laboratory tests which followed, the HP2100 computer was programmed to output reference data (CIRIS simulation). The altimeter data was also simulated. System navigation tests were performed with the IMU in both a static environment and a rotational environment.

These tests normally start with the IMU physically aligned with X axis North, Y axis West, and Z axis up. The alignment Kalman filter is run for fine alignment and residual tilts and gyro drifts after the alignment period are determined by monitoring North and West velocity. Various movements of the IMU box after the alignment is finished (free navigation) included 180° yaw turns, 15° roll and pitch turns (the table limits these to 15°), and Scorsby tests. These laboratory tests are easy to monitor and error sources are therefore easily isolated. It will be apparent in the test discussion to follow that the IMU error sources swamp all other possible sources of error.

The earliest laboratory test runs were made in late September and results are shown here as Table 7. Unless specifically stated otherwise, all tests were made with X axis North, Y axis West, and pitch, roll, yaw reference angles of 0,0,0. Also, the gyro bias is adjusted prior to alignment so that the compensated gyro outputs correctly match expected Earth rate. Since no lateral maneuvers can be made in the laboratory, no information about the vertical axis gyro drift is available to the alignment filter. Taking out Z-axis turn-on to turn-on drift is essential for analyzing performance results.

For the tests in Table 7 the standard deviation of accumulated gyro pulses (164 seconds summation period) is shown as well as the mean difference between the gyro output during and after alignment. In the static tests, the mean difference should appear directly in the free navigation drift. Test number 10 shows a large difference in mean drift of 0.677°/hr in the Y gyro. The result is that a drift of 0.56°/hr shows up on free navigation along the West axis and North position error is large. The performance results in Table 7 are reasonable considering the raw gyro variations, and distance errors here should be typical of the minimum possible.

In October 1976, tests were made involving a 180° yaw turn after free navigation and 30 to 60 minutes of tilt appeared. The problem was isolated to a bad X accelerometer as well as an incorrect program statement. Sending the IMU back to Connecticut, Hamilton Standard replaced the X accelerometer and recalibrated the IMU.

Table 8 lists performance data on the tests of 26 through 28 October 1976. Using velocity error versus time to fit tilts and gyro drifts, the distance error after 10 minutes of flight was computed and is shown in the right-hand columns. Typical velocity profiles for some of the tabulated tests are shown in Figures 60 and 61.

The static tests of 26 through 28 October 1976 are good and compare with the September 1976 tests. Scorsby tests show larger distance errors, but the frequency of rotation was 0.2 Hz and 0.3 Hz as opposed to 0.1 Hz. Judging from the quality of the IMU, it is likely that the poorer performance at higher coning frequencies is due to the IMU and not the software. Coning motion tests show reasonably similar performance to other tests not involving continuous motion.

An analysis of the 180° yaw turn experiments will illustrate how useful the controlled laboratory is in determining the origin of error sources. The large drift of -1 to -1.6°/hr will be traced to an X gyro anomoly. The larger tilts for the 180° yaw tests over the static tests are reasonably attributed to gyro and accelerometer misalignment errors.

A simple analysis shows that gyro and accelerometer misalignments as well as accelerometer bias errors result in tilt errors appearing after a 180° turn is made in free navigation. Quantitatively, twice the accelerometer misalignment/bias error and 3.1416 times the gyro misalignment error will appear following at 180° turn. Table 9 shows the misalignment matrices provided by Hamilton Standard for use in software compensation. Variations or inaccuracies of 0.5 minute in the third column entries of TAB and TGB would account for most of the increased tilt in the 180° yaw tests. Compensation tests (accumulation of raw data with Z up X North and South, Z down X North and South, Y up ...) show that accelerometer bias should account for 0.1 to 0.2 minute of tilt. On top of these error sources is IMU randomness. A few arc-minutes of residual tilt is therefore not unreasonable.

The North drift in the 180° yaw tests particularly documents the IMU anomolies with which this navigation system has to content. The following analysis shows that the residual North drift following a 180° yaw turn should be small or at least smaller than West drift. Remember that a 180° yaw

TABLE 7. STATIC

TEST NO.	TEST DESCRIPTION ALL TESTS PERFORMED WITH IMU X-AXIS POINTING NORTH, 15 minute ALIGNMENT	TEST DATE	RESIDUAL TILTS		RESIDUAL DRIFT		NORTH NAV ERROR DISTANCE (ft)	WEST ERR DIST (ft
			δN ($\widehat{min}$)	δW ($\widehat{min}$)	b_N (^{o}/hr)	b_W (^{o}/hr)		
1	STATIC	09/28	0.26	−0.43	0.014	−0.047	2066	5
2	STATIC	09/29	2.33	0.99	0.40^o	0.02^o	2407	15
3	STATIC	09/30	VERY SMALL	−0.89	VERY SMALL	−0.109	4294	−2
4	STATIC	09/29	1.35	0.43	−0.128	0.383	−2830	−36
5	STATIC WITH POSITION MATCH	09/29	0.36	1.34	0.087	−0.120	−1483	−14
6	STATIC NO EARTH RATE MATCH	09/28	−2.38	−0.34	0.108	−0.84	—	—
7	STATIC	09/28	1.99	VERY SMALL	0.096	VERY SMALL	−230	19
8	CONING MOTION 3^o HALF ANGLE, 0.1 Hz RATE	09/30					−461	−42
9	CONING MOTION 3^o HALF ANGLE, 0.1 Hz RATE	09/30					−2877	−32
10	STATIC	09/29	VERY SMALL	−0.174	VERY SMALL	0.57	−7262	−26

*SIGN IS SUCH THAT THE X MEAN DRIFT SHOULD APPEAR AS b_N AND Y AS b_W.

(The rev

TABLE 7. STATIC TEST SUMMARY OF 28-30 SEPTEMBER 1976

WITH RTH,	TEST DATE	RESIDUAL TILTS		RESIDUAL DRIFT		NORTH NAV ERROR DISTANCE (ft)	WEST NAV ERROR DISTANCE (ft)	TIME OF POSITION ERROR (min)	RAW GYRO ST. DEV. DURING ALIGN/NAV. 164 sec SUM PERIOD			RAW GYRO DIFFERENCE IN MEAN VALUE IN ALIGN/NAV MODE*		
		δN ($\widehat{min}$)	δW ($\widehat{min}$)	b_N (^{o}/hr)	b_W (^{o}/hr)				X (^{o}/hr)	Y (^{o}/hr)	Z (^{o}/hr)	X (^{o}/hr)	Y (^{o}/hr)	Z (^{o}/hr)
	09/28	0.26	−0.43	0.014	−0.047	2066	543	11	0.073	0.066	0.059	0.08	−0.049	−0.068
	09/29	2.33	0.99	0.40^o	0.02^o	2407	1506	11.8	0.223	0.137	0.180	−0.011	0.273	−0.0125
	09/30	VERY SMALL	−0.89	VERY SMALL	−0.109	4294	−282	15	0.226	0.601	0.029	0.188	−0.062	−0.025
	09/29	1.35	0.43	−0.128	0.383	−2830	−3614	21	0.064	0.104	0.029	−0.082	0.161	−0.024
ATCH	09/29	0.36	1.34	0.087	−0.120	−1483	−1471	12	0.064	0.104	0.029	0.082	0.161	−0.024
	09/28	−2.38	−0.34	0.108	−0.84	–	–	–	0.047	0.076	0.035	0.069	0.022	0.047
	09/28	1.99	VERY SMALL	0.096	VERY SMALL	−230	1925	8	0.036	0.071	0.043	−0.044	−0.073	−0.069
	09/30					−461	−4289	16.3	0.069	0.077	0.0815	–	–	–
	09/30					−2877	−3209	12	0.177	0.922	0.728	–	–	–
	09/29	VERY SMALL	−0.174	VERY SMALL	0.57	−7262	−260	13.6	0.136	0.426	0.152	0.082	0.677	−0.14

DRIFT SHOULD APPEAR AS b_N AND Y AS b_W.

(The reverse of this page is blank)

2

TEST NO.	TEST DESCRIPTION	TEST DATE	TILTS		DRIFT		DISTANCE ERROR		
			δ_N (min)	δ_W (min)	b_N (°/hr)	b_W (°/hr)	NORTH (ft)	WEST (ft)	(m
1	STATIC, 5 minute ALIGNMENT	10/26	VERY SMALL	−0.632	VERY SMALL	−0.07	941	−364	
2	180° YAW, 5 minute ALIGN 2 minute FREE NAV. 6 min. AFTER TURN	10/26	−3.62	2.41	−1.27	0.42	−1575	−2614	
3	15° PITCH DOWN, 5 minute ALIGN. 2-1/2 FREE NAV. 7-1/2 minutes PITCH	10/26	1.29	–	0.405	–	−1021	2614	
4	15° ROLL UP, 15 minute ALIGN. 2 min. FREE NAV. 7 minutes ROLL.	10/26	2.25	−0.68	−2.29	1.35	−1719	−1535	
5	180° YAW 15 minute ALIGN 6 min. FREE NAV. 6-1/2 min. ROTATE	10/27	−1.75	–	−1.61	–	−2632	−2109	
6	180° YAW, 15 minute ALIGN 1 min. FREE NAV. 6-1/2 min. ROTATE	10/27	−2.88	−3.78	−1.03	−0.22	−2646	−4559	
7	180° YAW, 15 minute ALIGN 1 min. FREE NAV. 6-1/4 min. ROTATE	10/27	−0.75	−2.56	−0.54	−0.813	3224	−1141	
8	STATIC, 5 minute ALIGNMENT	10/28	−0.523	−0.175	−0.139	−0.222	−1188	1903	1
9	180° YAW, 5 minute ALIGN YAW TURN RIGHT AFTER FREE NAV.	10/28	−0.477	0.308	−0.962	−0.363	796	−3663	
10	STATIC, 5 minute ALIGN +5° ERROR IN YAW EULER ANGLE	10/28	−0.475	−0.21	−0.286	0.336	−695	−1264	
11	180° YAW, 5 minute ALIGN +5° ERROR IN YAW EULER ANGLE	10/28	−1.15	0.886	−1.11	3.00	−8027	−3623	
12	SCORSBY, 3° HALF ANGLE 0.2 Hz, 5 min. ALIGN 1/2 min. FREE NAV. 7-3/4 min. SCORSBY	10/28	0.25	−0.349	0.774	1.03	−1965	2274	
13	SCORSBY, 3° HALF ANGLE 0.3 Hz, 3 minutes AFTER ABOVE TEST	10/28	0.25	−0.349	1.00	3.00	−6362	5954	3 A A T

TABLE 8. STATIC TEST SUMMARY OF 26-28 OCTOBER 1976

DRIFT		DISTANCE ERROR		TIME	CALCULATED DISTANCE ERROR USING GYRO DRIFT (10 minutes)		TILT ERROR IN 10 minutes		TOTAL CALCULATED ERROR IN 10 minutes		
^{o}/hr)	b_W (^{o}/hr)	NORTH (ft)	WEST (ft)	(minutes)	NORTH (ft)	WEST (ft)	NORTH (ft)	WEST (ft)	NORTH (ft)	WEST (ft)	TEST TYPE
RY ALL	−0.07	941	−364	8	392	–	1063	–	1455	–	STATIC
27	0.42	−1575	−2614	6	−2355	−7120	−4054	−6089	−6409	−13209	180° YAW TURN
405	–	−1021	2614	7-1/2	–	2270	–	2170	–	4440	15° PITCH
29	1.35	−1719	−1535	7	−7568	−12838	1144	3785	−6424	−9053	15° ROLL
1	–	−2632	−2109	6-1/4	–	−9026	–	−2944	–	−11970	180° YAW
3	−0.22	−2646	−4559	6-1/2	1233	−5774	−6358	−4844	−5125	−10618	180° YAW
4	−0.813	3224	−1141	6-1/4	4558	−3027	4306	−1262	8864	−4289	180° YAW
39	−0.222	−1188	1903	10-1/2	−1245	779	294	880	−951	1659	STATIC
62	−0.363	796	−3663	9	2035	−5392	−518	−802	1517	−6194	180° YAW
86	0.336	−695	−1264	8	−1884	−1603	353	−799	1531	−2402	STATIC 5° YAW ERROR
1	3.00	−8027	−3623	7-3/4	−16818	−6223	−1490	−1934	−18308	−8157	180° YAW 5° YAW
74	1.03	−1965	2274	7-3/4	−5774	4339	587	420	−5187	4759	CONING 3° HALF ANGLE AT 0.2 Hz
0	3.00	−6362	5954	3 min. AFTER ABOVE TEST	−16818	5606	587	420	−16231	5186	CONING 3° HALF ANGLE AT 0.3 Hz

(The reverse of this page is blank)

turn keeps the same gravity field as was present during alignment (hence, there are no spin axis or input axis unbalance effects) and that gyro and accelerometer misalignments and accelerometer bias errors result in residual tilts.

The only remaining explanation for the large residual drift following 180° yaw turns might be an incorrect gyro bias Earth-rate-match due to an incorrect azimuth heading. Assume we are not correctly aligned to North by an azimuth misalignment of $-\delta$. Gyro biases are adjusted so that

$$\omega_x = \Omega \cos \lambda, \ \omega_y = 0, \ \omega_z = \Omega \sin \lambda.$$

Referring to Figure 62, the actual matching should be

$$\omega_x = \Omega \cos \lambda \cos \delta \qquad \omega_z = \Omega \sin \lambda$$

$$\omega_y = -\Omega \cos \lambda \sin \delta$$

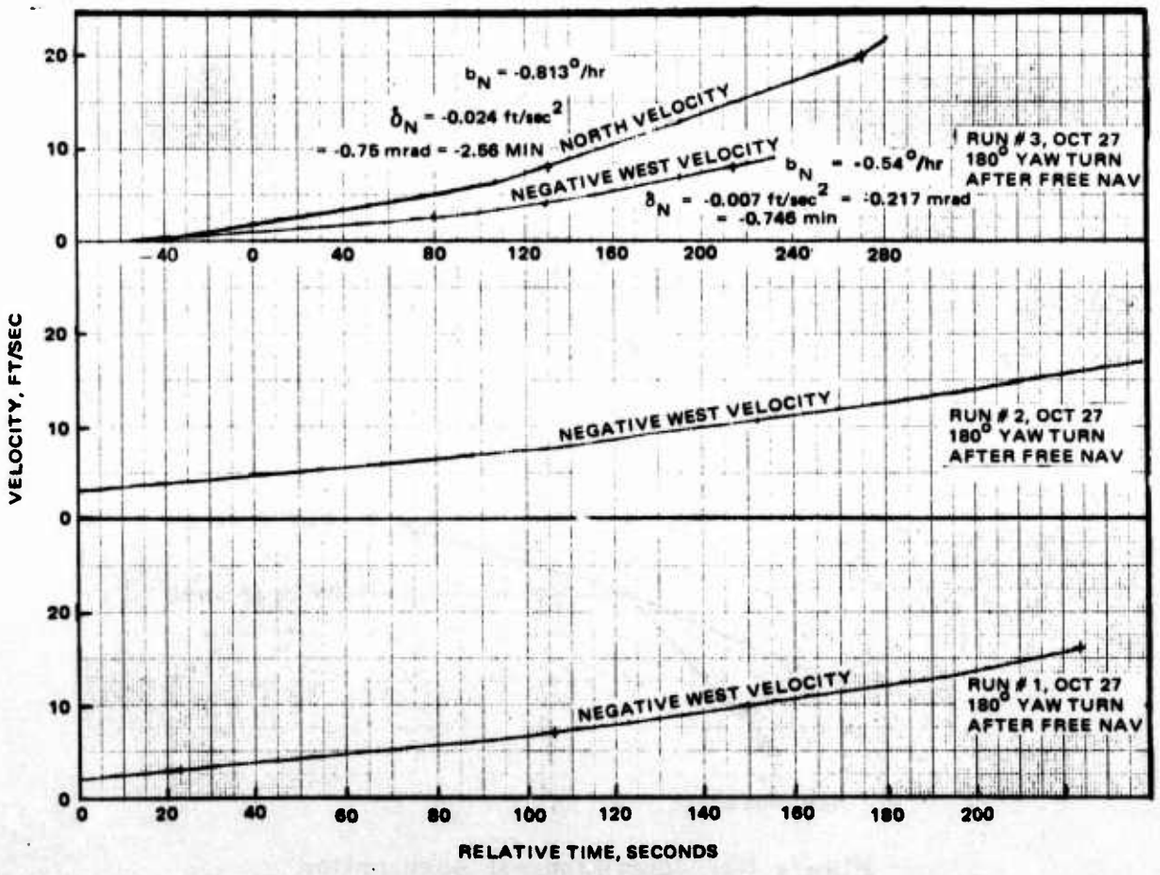


Figure 60. Typical Velocity Profiles, Runs 1 through 3

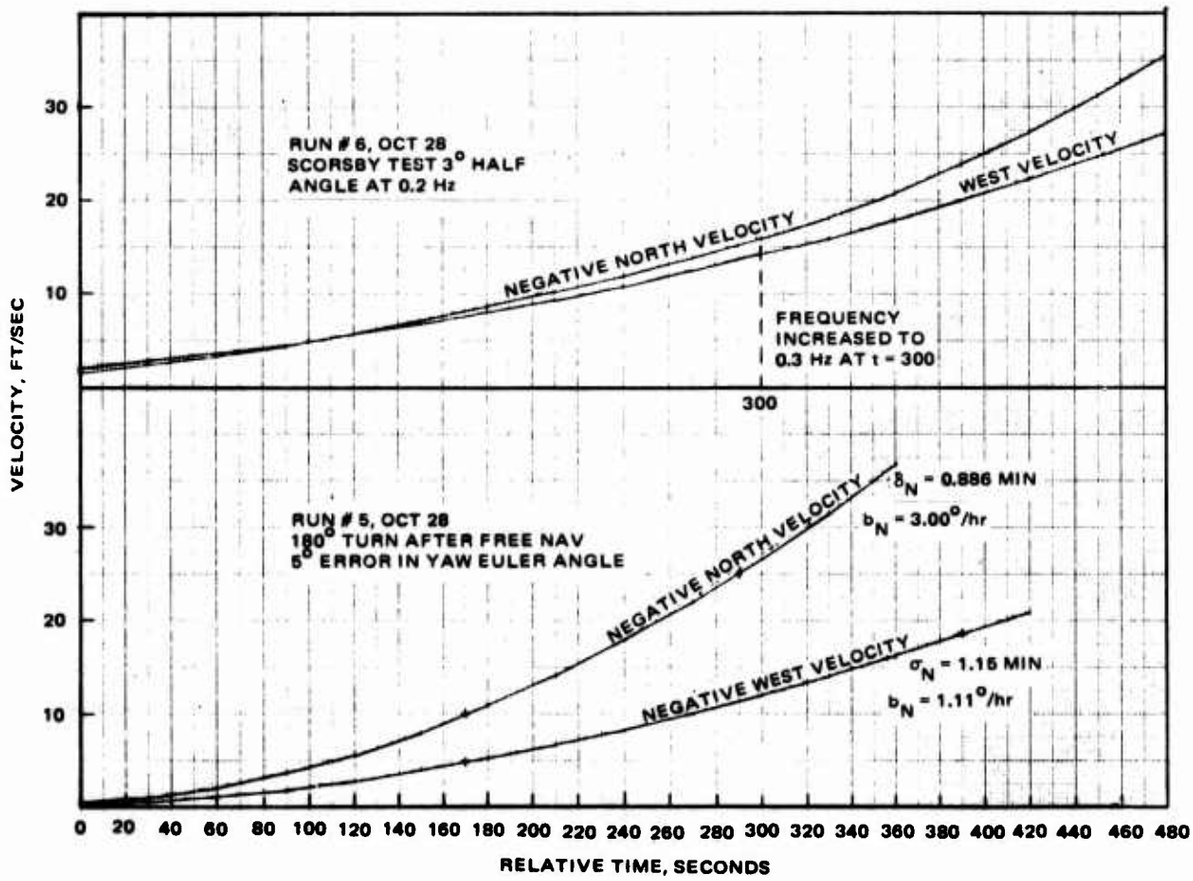


Figure 61. Typical Velocity Profiles, Runs 5 and 6

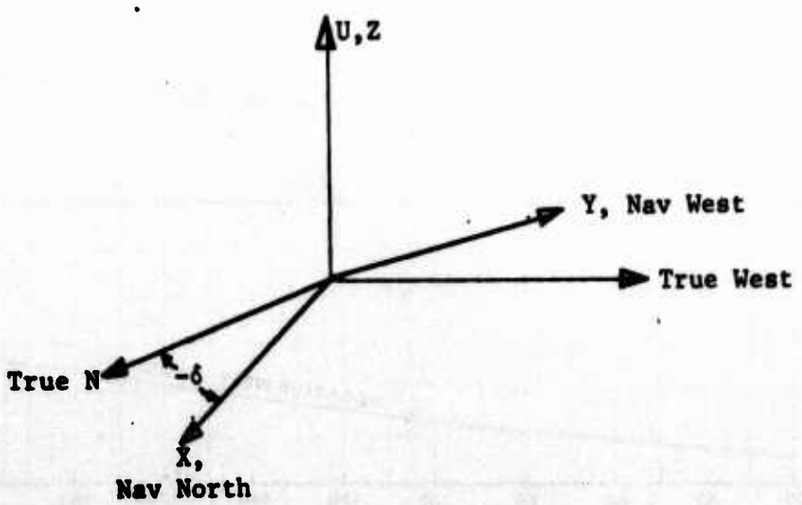


Figure 62. Misalignment Assumption

CALIBRATION ANALYSIS					CALIBRA
VARIABLE		CONSTANT OF 6/22	CONSTANT OF 10/22	10/22 – 6/22	
ACCELEROMETER SCALE FACTOR	X	0.036927 ft/sec/pulse	0.038561 ft/sec/pulse	–	
	Y	0.038787 ft/sec/pulse	0.038788 ft/sec/pulse	$1 \cdot 10^{-6}$ ft/sec/pulse	TAB = [1
	Z	0.037637 ft/sec/pulse	0.037636 ft/sec/pulse	$-1 \cdot 10^{-6}$ ft/sec/pulse	4.473 mra −0.573 mra
ACCELEROMETER BIAS	X	-0.03176 ft/sec^2	0.01967 ft/sec^2	–	[1
	Y	-0.02211 ft/sec^2	-0.026182 ft/sec^2	-0.00407 ft/sec^2	= 15.377 min
	Z	0.04705 ft/sec^2	0.04385 ft/sec^2	-0.0032 ft/sec^2	−1.970 min
GYRO SCALE FACTOR	X	0.0624166 min/pulse	0.0624232 min/pulse	$6.6 \cdot 10^{-6}$ min/pulse	
	Y	0.0606418 min/pulse	0.0606351 min/pulse	$6.7 \cdot 10^{-6}$ min/pulse	[1
	Z	0.056670 min/pulse	0.0566762 min/pulse	$6.2 \cdot 10^{-6}$ min/pulse	TGB = −0.9925 mra
GYRO BIAS	X	-27.81°/hr	-26.56°/hr	1.25°/hr	−0.1791 mra
	Y	33.66°/hr	37.48°/hr	3.82°/hr	[1
	Z	1.77°/hr	2.06°/hr	0.29°/hr	= −3.41 min
SPIN AXIS UNBALANCE	X	-5.13°/hr/g	-6.58°/hr/g	-1.44°/hr/g	−0.616 min
	Y	1.85°/hr/g	9.903°/hr/g	8.05°/hr/g	
	Z	-7.324°/hr/g	0.349°/hr/g	7.67°/hr/g	
INPUT AXIS UNBALANCE	X	-1.78°/hr/g	-1.71°/hr/g	0.071°/hr/g	1 $\widehat{\text{min}}$ OF ERROR IN THI
	Y	14.70°/hr/g	16.75°/hr/g	2.051°/hr/g	1 $\widehat{\text{min}}$ OF ERROR IN THI
	Z	5.314°/hr/g	5.96°/hr/g	0.646°/hr/g	

TABLE 9. CALIBRATION ANALYSIS SUMMARY

	10/22 – 6/22
	–
	$1 \cdot 10^{-6}$ ft/sec/pulse
	$-1 \cdot 10^{-6}$ ft/sec/pulse
	–
	-0.00407 ft/sec^2
	-0.0032 ft/sec^2
	$6.6 \cdot 10^{-6}$ min/pulse
	$6.7 \cdot 10^{-6}$ min/pulse
	$6.2 \cdot 10^{-6}$ min/pulse
	1.25°/hr
	3.82°/hr
	0.29°/hr
	−1.44°/hr/g
	5.05°/hr/g
	7.67°/hr/g
	0.071°/hr/g
	2.051°/hr/g
	0.646°/hr/g

CALIBRATION ANALYSIS 10/22

$$TAB = \begin{bmatrix} 1 & 4.42 \text{ mrad} & 0.948 \text{ mrad} \\ 4.473 \text{ mrad} & 1 & 2.88 \text{ mrad} \\ -0.573 \text{ mrad} & 0.015 \text{ mrad} & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 15.195 \text{ min} & 3.259 \text{ min} \\ 15.377 \text{ min} & 1 & 9.90 \text{ min} \\ -1.970 \text{ min} & 0.0516 \text{ min} & 1 \end{bmatrix}$$

$$TGB = \begin{bmatrix} 1 & 2.49 \text{ mrad} & 0.2473 \text{ mrad} \\ -0.9925 \text{ mrad} & 1 & 0.644 \text{ mrad} \\ -0.1791 \text{ mrad} & -0.6525 \text{ mrad} & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 8.56 \text{ min} & 0.850 \text{ min} \\ -3.41 \text{ min} & 1 & 2.214 \text{ min} \\ -0.616 \text{ min} & -2.243 \text{ min} & 1 \end{bmatrix}$$

CALIBRATION ANALYSIS 6/22

$$TAB = \begin{bmatrix} 1 & -0.0453 \text{ mrad} & 1.377 \text{ mrad} \\ 5.777 \text{ mrad} & 1 & 2.257 \text{ mrad} \\ -2.63 \text{ mrad} & 0.48 \text{ mrad} & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & -0.156 \text{ min} & 4.6 \text{ min} \\ 19.86 \text{ min} & 1 & 7.76 \text{ min} \\ -9.04 \text{ min} & 1.65 \text{ min} & 1 \end{bmatrix}$$

$$TGB = \begin{bmatrix} 1 & 0.884 \text{ mrad} & 2.342 \text{ mrad} \\ 0.36 \text{ mrad} & 1 & 0.221 \text{ mrad} \\ 2.507 \text{ mrad} & 0.181 \text{ mrad} & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 3.04 \text{ min} & 8.05 \text{ min} \\ 1.28 \text{ min} & 1 & 0.761 \text{ min} \\ -8.62 \text{ min} & -0.621 \text{ min} & 1 \end{bmatrix}$$

1 $\widehat{\text{min}}$. OF ERROR IN THE THIRD COLUMN OF TGB RESULTS IN A TILE OF 3.42 $\widehat{\text{min}}$ AFTER A 180° TURN.

1 $\widehat{\text{min}}$ OF ERROR IN THIRD COLUMN OF TAB RESULTS IN A TILE OF 2 $\widehat{\text{min}}$. AFTER A 180° TURN.

(The reverse of this page is blank)

Erroneous biases of $\Omega\cos(\lambda)\cos(\delta)-\Omega\cos(\lambda) = -\Omega\cos(\lambda)\,\delta^2/2$ and $-\Omega\cos(\lambda)\sin(\delta)-0 = -\Omega\cos(\lambda)\delta$ are put into the North (X) and West (Y) gyros. Azimuth misalignment is not corrected during alignment since the accelerometers give very little information about δ. Misalignment exists but is statically balanced quite well as test number 10 shows. Here a 5-degree error in Yaw Euler angle induces a known error relative to the other static tests. No noticeable degradation from other static tests appears.

Due to Earth rate matching, the static balance equations read as follows:

$$\omega_x = \omega_{\text{NAV NORTH}} = \Omega\cos(\lambda)\cos(\delta)+b_x = \Omega\cos(\lambda)$$

$$\omega_y = \omega_{\text{NAV WEST}} = -\Omega\cos(\lambda)\sin(\delta)+b_y = 0$$

With a 180° turn, the X and Y compensated gyro outputs measure

$$\omega_x = -\Omega\cos(\lambda)\cos(\delta)+b_x$$

$$\omega_y = \Omega\cos(\lambda)\sin(\delta)+b_y$$

Since the Z (or vertical) gyro measured a 180° turn, the navigation frame requires $\omega_{\text{NAV NORTH}}$ and $\omega_{\text{NAV WEST}}$ to be

$$\omega_{\text{NAV NORTH}} = -\cos(\lambda)$$

$$\omega_{\text{NAV WEST}} = 0$$

or else a residual drift will arise.

The errors are then

$$b_N = \omega_x-\omega_{\text{NAV NORTH}} = (-\Omega\cos(\lambda)\cos(\delta)+b_x)-(-\Omega\cos(\lambda))$$

$$= 2(\Omega\cos(\lambda)-\Omega\cos(\lambda)\cos(\delta))$$

$$= \delta^2\Omega\cos(\lambda)$$

and

$$b_W = \omega_y - \omega_{NAV\ WEST} = (\Omega\cos(\lambda)\sin(\delta)+b_y)-0$$

$$= 2\Omega\cos(\lambda)\sin(\delta) = 2\delta\Omega\cos(\lambda)$$

This analysis says that there should appear a West gyro drift (North velocity related) but no North drift. A 5-degree azimuth misalignment should reveal a West drift of 2.2^{o}/hr and a North drift of 0.05^{o}/hr after a 180^{o} turn. Test number 11 bears out the effect of azimuth misalignment on West drift. However, all of the 180^{o} yaw tests show an unaccountable North drift ranging from -0.54^{o}/hr to -1.61^{o}/hr.

Two raw data compensation tests revealed that the problem is due to the X gyro. It appears that the X gyro bias is different depending on whether the axis is pointing North or South. The compensation test results were

-1.77^{o}/hr additional bias South, uncertainty 0.68^{o}/hr
-2.0^{o}/hr additional bias South, uncertainty 1^{o}/hr

The uncertainties arise from table levelness and are upper bounded using alignment test results, physically estimating table leveiness, and compensated Z gyro North and South accumulations. A tilt of 3-1/2 degrees about the Y axis results in an X axis gyro North/South difference of $2(8.16^{o}/hr)\sin(3.5^{o}) = 1^{o}/hr$. In one of the above compensation tests, the Z gyro data indicated a maximum of 3-1/2-degree tilt. These tests conclusively point to an X gyro anomoly which is unaccounted for in the compensation equation formulations.

Finally, the roll and pitch movements of 15 degrees are again reasonably good. Table 9 shows that spin and input axis unbalances are large and since the sine of 15 degrees is 0.26, uncertainties in these quantities can cause large residual drifts. The compensation test results in Table 10 even more succinctly show how the navigation performance is limited by the IMU. Raw and compensated gyro outputs are accumulated for 100 seconds and displayed in units of degrees per hour in the first two numerical columns. That the Raw data minus Hamilton Standard compensation constants is very close to the compensated data shows that the compensation software is correct. Gyro errors in the last column are due to improper HS compensation constants and/or not a complete enough compensation equation formulation. That is, the Y gyro behavior for Y up and Y down implies the spin axis unbalance requires a gravity squared (g^2) compensation. Hamilton Standard did not provide g^2 compensation coefficients, probably because these terms vary wildly from turn-on to turn-on.

TABLE 10. GYRO COMPENSATION ANALYSIS

MEASUREMENT POSITION		QUANTITY MEASURED	RAW DATA VALUE (^{o}/hr)	COMP. DATA VALUE (^{o}/hr)	RAW − H.S. (SHOULD EQUAL COMP.)	GYRO ERROR ^{o}/hr
	X	B_X	−27.85	−1.24	−1.29	−1.24
Z↑	Y	B_Y	37.69	37.69	.21	.21
	Z	B_Z − SA+ER	13.18	13.18	11.46	3.31
	X	B_X	−27.47	.875	−.91	−.875
Z↓	Y	B_Y	37.35	−.12	−.13	−.13
	Z	BA+SA − ER	−9.98	−12.39	12.39	−4.23
	X	B_X − IA	−25.11	−.19	.26	−.19
Y↑	Y	B_Y − SA+ER	28.43	.825	.853	−7.335
	Z	B_X + IA	7.48	−.536	−.54	−.536
	X	B_X + IA	−33.52	−5.18	−5.25	5.18
Y↓	Y	B_Y + SA − ER	38.855	−8.515	−8.53	−.355
	Z	B_Z − IA	4.23	−.33	−.33	−.33
	X	B_X − SA+ER	−17.81	2.266	2.17	5.894
X↑	Y	B_Y + IA	48.95	−5.285	−5.28	5.285
	Z	B_Z	2.04	−0.25	−.02	−.025
	X	B_X + SA − ER	−40.30	−7.13	−7.16	1.03
X↓	Y	B_Y − IA	22.975	2.26	2.245	2.26
	Z	B_Z	.54	−1.53	−1.52	1.53

NOMENCLATURE: B − BIAS (^{o}/hr), SA − SPIN AXIS UNBALANCE (^{o}/hr/g)
IA − INPUT AXIS UNBALANCE (^{o}/hr/g), ER − EARTH RATE

In conclusion, the laboratory tests give some idea of the navigation performance to expect in the van or aircraft. The IMU limitations are clearly presented and will have to be considered in all tests.

Dynamic Navigation Tests

Following the system integration tests, the DP1 system was installed in a van for dynamic navigation tests. CIRIS was also installed in the van for these tests. Altimeter data was simulated (constant altitude) by test hardware.

A preliminary analysis of various navigation tests in the van is shown in Table 11. This analysis shows that the navigation performance is essentially as expected based on laboratory test results. Further testing of the system will consist of aircraft navigation tests. CIGTF personnel are responsible both for performing all dynamic tests and for the detailed analysis of test results. Consequently, the complete analysis of system performance is not contained in this report.

TABLE 11. VAN TEST SUMMARY OF 9 NOVEMBER – 11 NOVEMBER

	TILTS		DRIFT		DISTANCE ERROR		TIME
	δ_N MINUTE	δ_W MINUTE	b_N (°/HR)	b_W (°/HR)	NORTH (FT)	WEST (FT)	(MINUTES)
NO EARTH RATE MATCH; STRAIGHT ROADS	0.74	-1.9	0.277	0.142	——*	-1650	12¼
NO ERM. SMOOTHLY CURVED ROADS	1.134	0.33	-0.113	0.147	——*	1628	11
NO ERM. SHARP 90° TURNS	3.76	-3.95	0.636	0.789	1250	2165	5
NO ERM. SMOOTH ROADS WITH ACCELERATIONS AND DECELERATIONS	-0.153	0.663	-0.277	-0.288	1345	-492	10
SAME AS ABOVE	-0.342	-0.634	0.322	0.103	-795	99	10
NO ERM. SHARP TURNS, ROCKING, AND ACCELERATIONS	-0.355	-3.37	0.04	-0.107	2964	-219	7

*CIRIS HAD A LATITUDE PROBLEM ON THIS DAY.

Summary

The tests made with the DP1 system ranged from detailed testing of individual software elements in a laboratory environment to complete system testing in a flight environment.

Laboratory tests were performed both at Hughes and at CIGTF. The Hughes Aircraft Company laboratory tests were directed at verification of the software modules and integration of the software with the DP hardware and IMU. The CIGTF laboratory tests were aimed at integration of the HP2100 computer (CIRIS simulation) with other system elements and system level tests with the IMU both in a static environment using coning motion.

In the software module tests at Hughes Aircraft Company, the executive software was verified by dynamic testing under conditions exceeding the expected requirements for CIGTF operation. The results achieved in the navigation and alignment software module tests are summarized below:

(1) The navigation software performance using a simulated IMU (DP software test driver) matched well with results predicted by analytic simulation.

(2) The alignment software quickly converged to accurate estimates of misalignments and gyro biases which were input from a software test driver simulating the IMU and navigation software.

The system integration tests at Hughes Aircraft Company involved the integration of all the system software functions with the system hardware elements. At the end of the integration, static alignment with the IMU appeared to be satisfactory; tilt misalignment and horizontal gyro bias estimates were reproducible and successive bias estimates appeared to converge rapidly.

A software test driver capable of simulating IMU data for both static alignment and alignment maneuvers was used for system level verification of the DP software. For simulated static alignment conditions, it was found that residual tilt misalignments were typically less than one arc-minute and residual gyro biases under 0.1°/hr. For simulated alignment maneuvers, the typical performance was as follows:

		Expected Error
Tilt misalignment	0.6 arc-minute	1 arc-minute
Azimuth misalignment	5 arc-minutes	3 arc-minutes
Bias, Y and Z-axis gyros	0.3°/hour	0.2°/hour
Bias, X-axis gyro	0.9°/hour	0.2°/hour

The azimuth misalignment and X-axis bias were found to be considerably worse than expected, as indicated, probably due to precision losses in the simulation software.

At CIGTF, system navigation laboratory tests were performed with the IMU in both a static environment and a rotational environment. The following is a summary of the results obtained.

CIGTF Laboratory Tests

(1) Distance error performance was reasonable for the expected IMU error sources. The tests verified the DP software.

(2) Residual tilt misalignment and gyro biases calculated from velocity error histories were reasonable for the raw IMU gyro variations.

The laboratory test indicated the expected navigation performance for the subsequent dynamic tests in the van and aircraft.

Following these tests, the DP1 system was installed in a van (with CIRIS) for dynamic navigation tests. The test results showed that the navigation performance was, in general, the same as that expected based on laboratory test results.

SECTION IV

HYBRID SIMULATION

The CIGTF operation demonstrated the CORE electronics concept in a real environment, but did not include all of the functions which the CORE electronics will be required to perform in a tactical system. The hybrid simulation was designed to include all of the CORE functions except system test.

The concept of the system simulated is shown in Figure 63. Basically, it is a long-range air-to-surface weapon which uses an aided inertial guidance system for midcourse guidance. The simulated weapon was based on the GBU-15 system with appropriate guidance elements added.

The simulation was performed in the Hughes Hybrid Simulation facility at Canoga Park, California. The simulation of the GBU-15 was originally developed for the PDAP simulations in 1975. Some modifications to the simulation were necessary and are described in the following paragraphs. The entire system was simulated on the hybrid computer except for the digital processor in the CORE electronics. The DP2 breadboard processor was used to emulate the CORE digital processor.

SYSTEM DESCRIPTION

The system which was simulated is a modular, long range, air-to-surface weapon. It has a midcourse guidance subsystem which can fly a pre-set course from launch point to the target area. The trajectory is not, in general, a great circle route or a straight line. Rather, it more often will be a segmented path to allow flying over intermediate check points or to allow avoidance of some point. The basic guidance is strapdown inertial which is implemented with an inertial measurement unit (IMU) and a digital processor to perform the strapdown inertial calculations. The inertial guidance system is not accurate enough, by itself, to achieve the required CEP at the target area; therefore, it must be periodically corrected during the flight. The correction is made with the aid of a position sensor such as a LORAN receiver or a GPS receiver. The data from the position sensor is smoothed by a Kalman filter and then is used to update the inertial system's estimate of position.

The weapon may fly all the way to the target via the inertial guidance or it may switch to a terminal guidance system. The terminal sensor could be, for example, a readiometric seeker or an electro-optical seeker. In the latter case, lockon for terminal guidance would be achieved by an operator working through a data link.

The combination of the digital processor and the IMU is thought of as the core electronics for the weapon. The other subsystems (airframe, warhead, position sensor, terminal sensor) may be modularly changed to suit the operational requirements, but the core electronics remains the

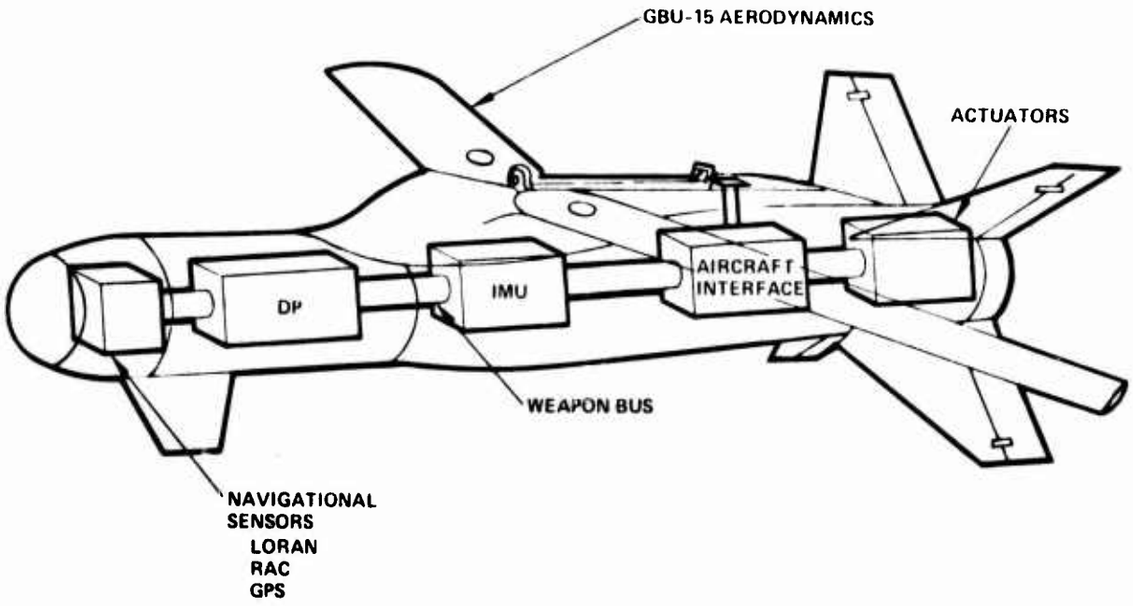


Figure 63. Weapon to be Simulated

same in all configurations. The digital processor not only performs the strapdown inertial calculations but also does the autopilot calculations and the system integration function.

The simulation performed was a semi-physical simulation; that is, a portion of the system was simulated by a hybrid computer and the remainder was simulated by real hardware. In this case the real hardware was the digital processor and the digital weapon bus.

Airframe

The airframes simulated were those of the GBU-15. The GBU-15 is a modular weapon with two major aerodynamic configurations: a weapon with an unfolding planar wing and a cruciform tail which is referred to as the planar-winged-weapon (PWW) and a weapon with a set of cruciform wings aligned with the cruciform tail which is referred to as the cruciform-winged-weapon (CWW). The control surface orientation remains the same for either aerodynamic configuration. Each airframe requires its own autopilot. The autopilot configuration is selected by discrete signals, the values of which are determined by the airframe configuration and other subsystems which may be on board.

Core Electronics

The core electronics consists of the digital processor, the weapon bus, the IMU, and an altimeter. The altimeter is required to keep the inertial guidance system stable in the vertical channel. Functionally, the core electronics is identical with the corresponding components used in the CIGTF operation and previously described in Section III.

In the simulation, the IMU and altimeter functions were simulated. The functions of the digital processor and the weapon bus were performed with real hardware: the DP2 system which is described in the following section.

Position Sensor

The function of the position sensor is shown in Figure 64. As shown in the figure, the basic guidance is strapdown inertial. At periodic times, the position sensor makes an accurate determination of the weapon position and the position data are used to update the inertial estimate of position. Actually, the position update may not be periodic for some of the sensors. The radiometric area correlation (RAC) sensor makes update measurements at preselected check points. These check points, in general, will not be on a straight line; therefore, the guidance law may have to compute a segmented trajectory so as to fly over the check points.

As indicated in the figure, several possible position sensors have been suggested for use in an advanced weapon. The operation of any of them in the system is the same except for two factors:

(1) Some of the sensors require a shaped trajectory either for passing over intermediate check points or for terminal guidance (if they are used for terminal).

(2) Each sensor requires a different transformation applied to its output data before it can be used to update the inertial system.

Only the LORAN sensor was used in the simulation; however, the guidance law did allow segmented trajectories to be flown as would be required for sensors such as RAC and TERCOM.

Terminal Guidance

The terminal guidance subsystem simulated was an EO seeker. In real operation the seeker would be locked on to the target at initiation of terminal guidance by an operator working through a data link. In the simulation, the operation of the operator and data link was not simulated in detail. Rather, the function was simulated by pointing the simulated seeker look angle at the target.

SIMULATION IMPLEMENTATION

The system just described was simulated using a semi-physical simulation. A semi-physical simulation is a simulation in which a portion of the actual physical system is incorporated into the simulation. The physical hardware performs its expected function while the remainder of the system functions are modeled using analogous mathematical functions usually on computers. In this simulation the actual physical system is the DP and weapon bus. The remainder of the system was modeled on a hybrid computer consisting of a Sigma 8 digital computer, two EAI 781 model analog computers, and a Beckman 2200 model analog computer.

Functional Description

The major functions of the weapon are shown in Figure 65. During the launch sequence, initialization information is received across an interface with the aircraft stores management system. The system management software supervises the receipt of this data, placing it in its proper operand memory locations and placing initialization tasks in the queue as the proper data is received from the aircraft fire control system.

At detection of separation of the weapon from the aircraft, the system management software starts an internal clock which is used to control periodic tasks and time-dependent selection of processing alternatives. The flight control system (autopilot) provides weapon stabilization and steering based upon inputs from the IMU sensors and guidance calculations. The autopilot sends commands to the flipper actuators. The weapon stabilization loop is closed through the IMU sensors which measure the vehicle dynamic response to the flipper actuator commands. The motion of the flipper on the weapon causes the aerodynamic forces to change which induces body motion of the weapon. The changes in body motion are detected by the IMU sensors which send the information to the autopilot and the strapdown inertial navigation calculations. The dynamic motion of the weapon is used to determine weapon position, velocity and attitude data in both the strapdown inertial navigation processing and geometry functions. The location and velocity of the weapon calculated by the strapdown inertial navigation processing are used by the guidance law to generate steering signals which are sent to the autopilot for yaw plane steering in the midcourse phase of weapon flight. The body motion changes the geometry of the weapon with respect to the target and LORAN stations, and the geometry calculation of weapon position is combined with the LORAN station data to generate LORAN receiver outputs during midcourse. The LORAN receiver outputs are processed to determine weapon position data which are used to correct the strapdown inertial navigator via an alignment (Kalman) filter.

The geometry calculations of weapon position, velocity, and attitude are also combined with target position data to generate steering signals from the EO seeker which is used for terminal guidance. Thus, the guidance (steering) loop is closed through the updating of the geometry function which, in turn, affects the guidance sensor outputs. The equipment on which each of the system functions is implemented in the simulation is designated in each box in the figure.

Digital Processor Hardware

The basic digital processing system consists of a digital processor (DP) and a serial, multiplexed digital bus which is called the Weapon Bus. The DP can communicate with other subsystems via the weapon bus. The interface with the weapon bus is by means of a bus interface unit (BIU), which formats data for transmission over the bus.

In the simulation, the DP2 system was required to interface with a digital and an analog computer. Since neither of these computers has an interface compatible with the Weapon Bus, it was necessary to fabricate

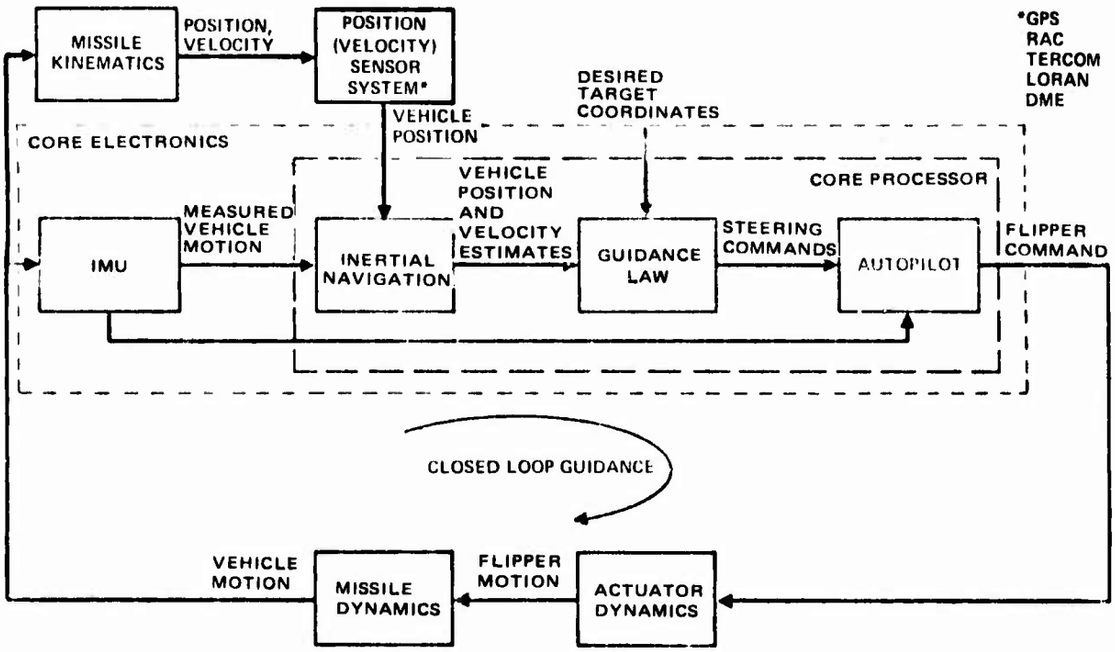


Figure 64. Inertial Navigation Update Guidance System

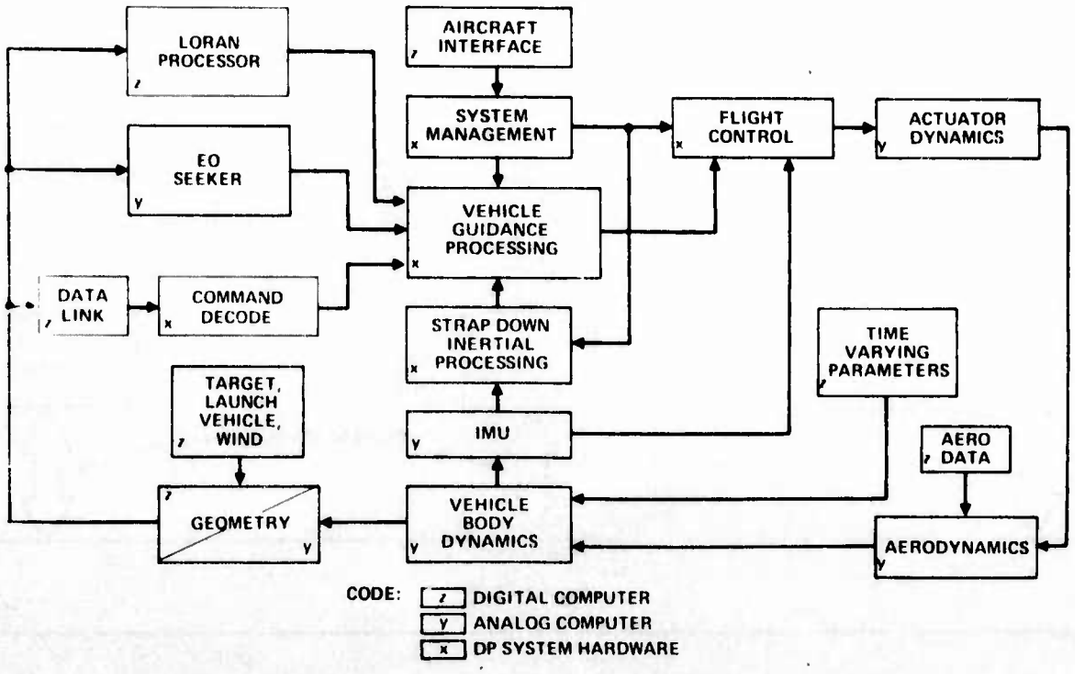


Figure 65. DP Hybrid Simulation

two additional interface units. The first of these, called the Signal Conversion Interface Unit, provides an interface with the analog computer. It provides an analog-to-digital and digital-to-analog conversion function as well as a BIU-compatible interface. The second unit, called the Sigma 8 Interface Unit, provides an interface with the digital computer.

A block diagram of the DP2 system, as it was used in the Hybrid Simulation Laboratory, is shown in Figure 66. A control panel, not shown in the diagram, allows the operator to monitor and control the system.

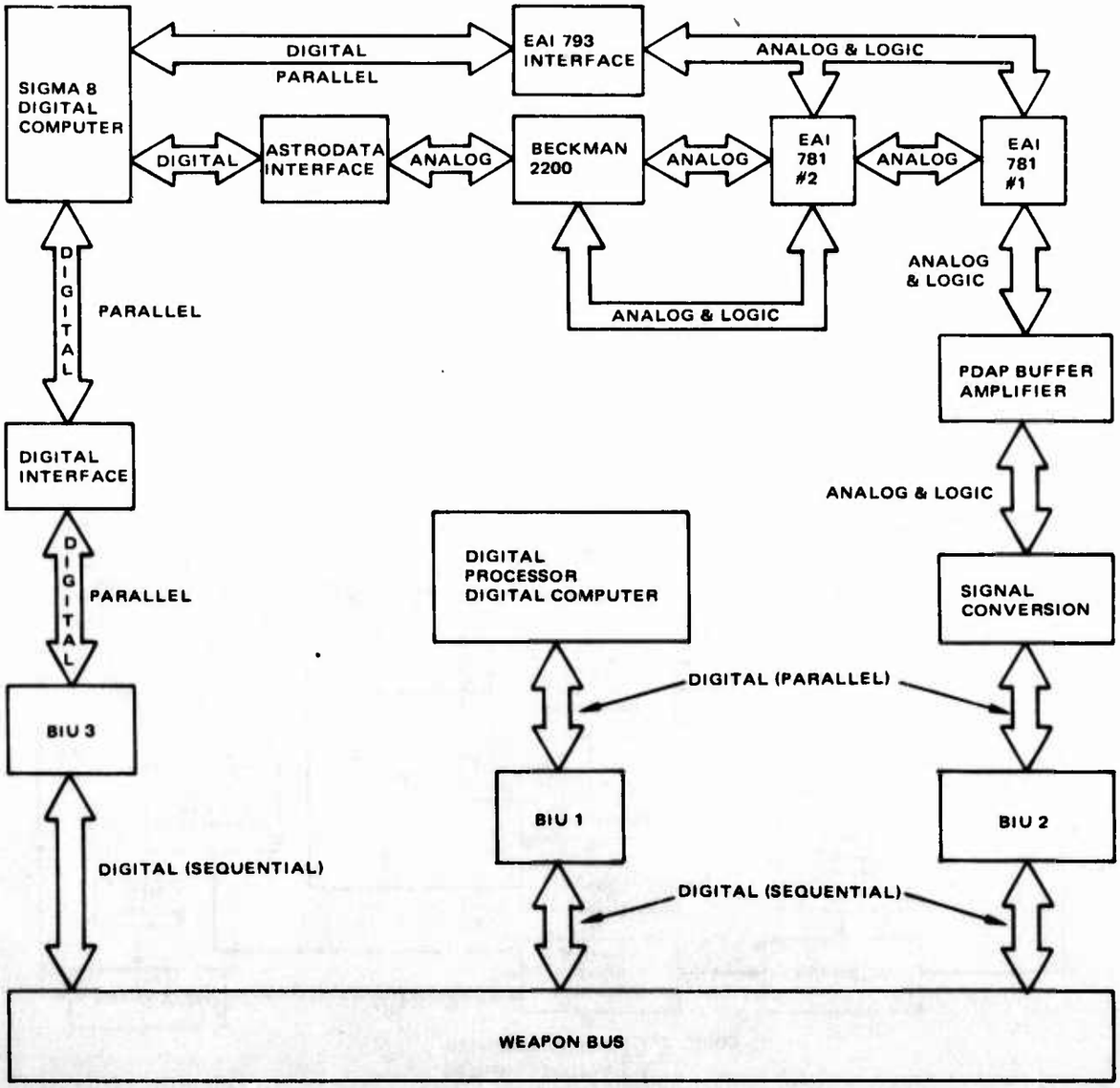


Figure 66. Block Diagram of Hardware Interfaces Used in Simulation

The DP2 system differs from the DP1 (described in Section III) in two ways. First, the DP2 processor has an extended instruction set. Second, in the DP2 system, the Master Bus Control Circuits are contained in the DP cabinet. In the DP1 system, these control circuits were contained in one of the BIU boxes which was called the Master BIU. This latter change simplifies the interface between the DP and the weapon bus.

A detailed description of the DP2 system is given in the report DGWT 0210-2, Operating Manual and System Description for Digital Processor Number 2. The instruction set for the processor is described in DGWT 0170-2, Programmer's manual for PDAP, DP1 and DP2.

Hybrid Computer Simulation

A hybrid simulation of the GBU-15 weapon system was originally developed for validating PDAP performance and has since been modified to support the WCU development program for GBU-15. A functional block diagram of the baseline GBU-15 simulation is shown in Figure 67. Several modifications were required to the baseline configuration to incorporate the additional weapon system functions for the DP2 simulation effort. The new weapon functions were: strapdown inertial navigation, LORAN data processing and the inertial midcourse guidance law. The simulation was required to not only furnish the appropriate real time data for these functions, but also initialization data via a simulated avionics interface for prelaunch weapon preparation.

The only interface of the hybrid computer with an external digital processor was via analog signal lines in the baseline configuration. However, a digital interface was desired in the DP2 simulation not only to simulate the digital avionics interface but also to provide adequate resolution on the functional interface signals (initialization and real time). Therefore, the required digital interface hardware was installed, and an assembly language program was written to allow the SIGMA 8 to both read data from and write data to the digital interface unit.

The following paragraphs address the simulation modifications to incorporate the new weapon functions.

Strapdown Inertial Navigation. The simulation must provide angular rate (or incremental angle) and acceleration (or incremental velocity) data in the three missile body axes to DP2. The required transfer rate (100 Hz) of this data would have produced loading problems in the SIGMA 8 if the digital interface were used for this transfer. Therefore, the simulated IMU data was transferred to DP2 via analog signals. Angular rate and acceleration data were output by the analog computer for minimum simulation interface complexity. This required that the baseline simulation be modified to output weapon roll rate and longitudinal acceleration signals in addition to the other two rate and acceleration signals already existing.

The data required to initialize the DP2 navigation software parameters were output by the SIGMA 8 via the digital interface.

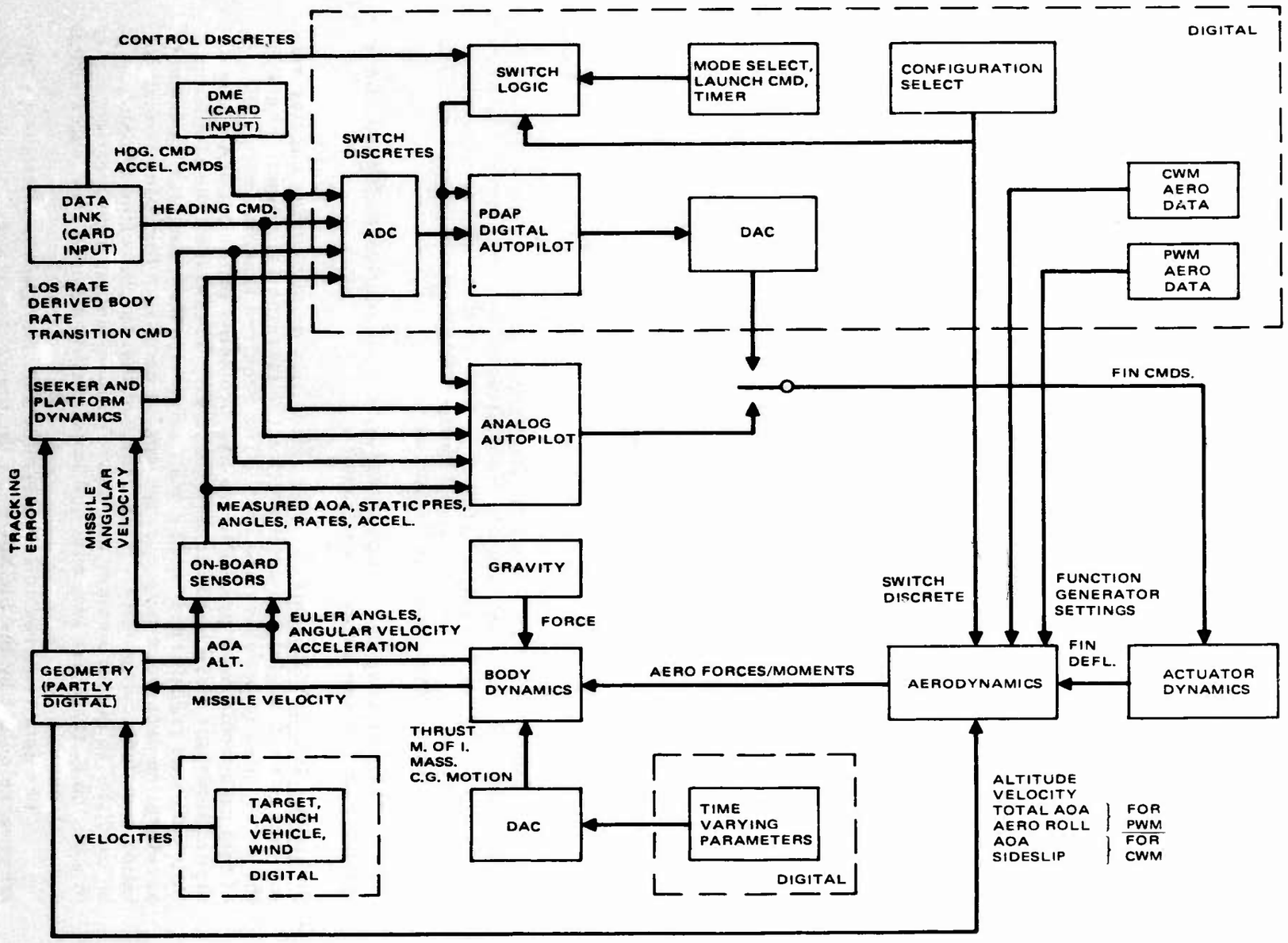


Figure 67. Baseline Simulation Functional Block Diagram

LORAN Receiver Function. The LORAN receiver function was simulated by a FORTRAN program in the SIGMA 8 computer. A block diagram of the LORAN simulation is shown in Figure 68. This program determines the master-slave distance differences corresponding to the simulation weapon position for each of the two slave stations. Distance difference data was output to DP2 via the digital interface rather than the more customary time

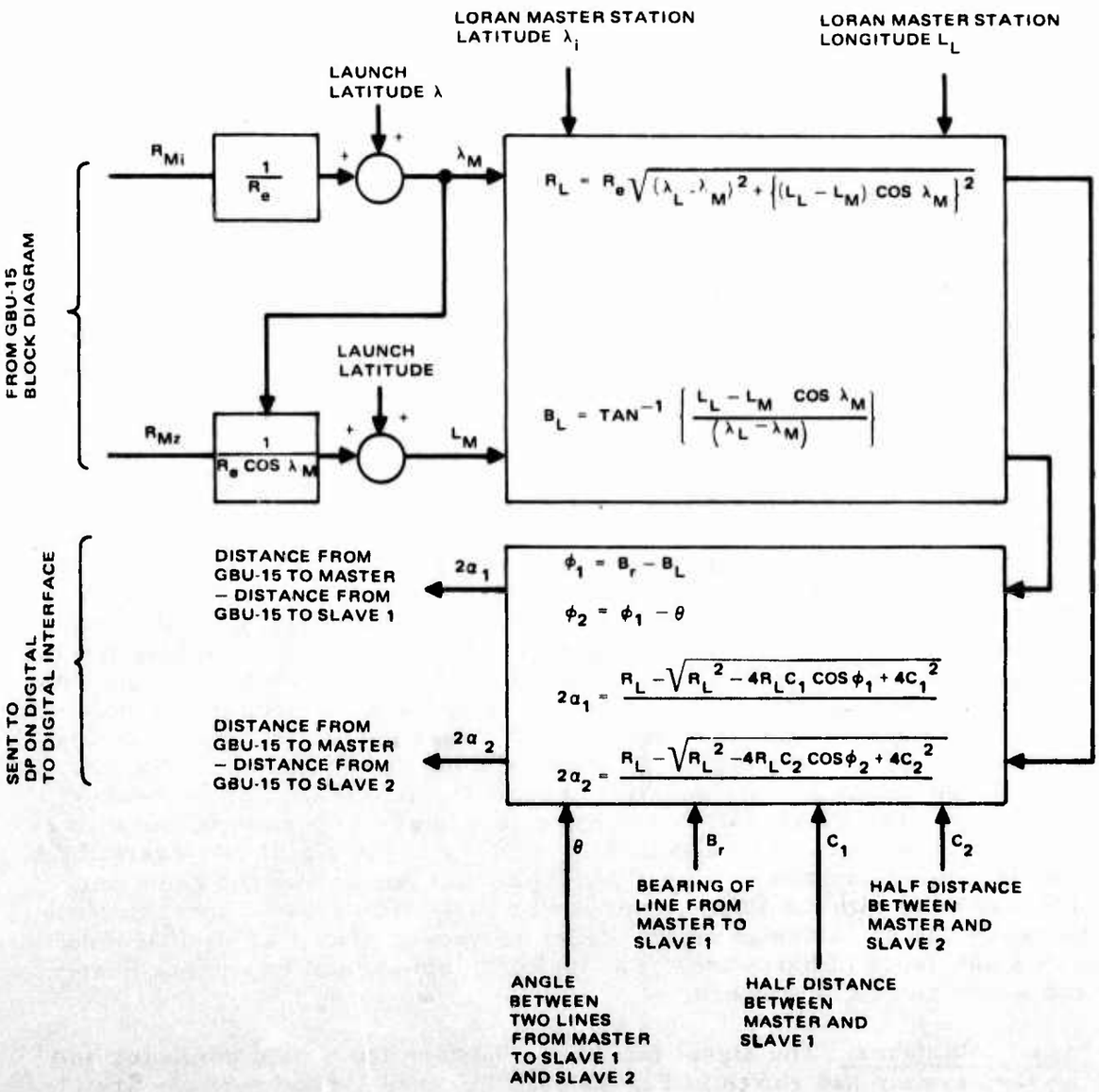


Figure 68. Block Diagram of Additiona to GBU-15 Simulation for LORAN Receiver Type Information

difference data output of a LORAN receiver. However, this data format difference only corresponds to a scale factor discrepancy, namely, the speed of light. The scale factor was compensated in LORAN network parameters sent by the SIGMA 8 to DP2 via the digital interface as part of system initialization.

Inertial Midcourse Guidance Law. The simulation modification required to support the DP2 midcourse guidance law calculations consists of outputting target and checkpoint position data via the digital interface as part of system initialization. In addition, the midcourse guidance law was also implemented by a FORTRAN routine on the SIGMA 8. The SIGMA 8 calculation of midcourse steering command was output to DP2 at 50 Hz via the digital interface. Selection of either the SIGMA 8 or DP2 implementation of the guidance law was made by modification of the appropriate DP2 memory reference instruction.

The next subsection details both the analog and digital parameters which are transferred between the hybrid computer and DP2.

Simulation Interfaces

A block diagram of the hardware interfaces is shown in Figure 69. The signal format for each interface is shown also. There are four types of signals: analog, discrete logic, parallel digital, and bit-sequential digital. The interface between the DP2 equipment and the SIGMA 8 is parallel digital.

As with most semi-physical simulations, special purpose interface hardware is required to create a compatible interface between the system hardware and the modeling hardware. For this simulation the special pieces of hardware were the digital interface, the signal conversion unit, and the PDAP buffer amplifier. The digital interface contained a 32-word buffer storage area and two latches for storing control words being transferred in either direction. The signal conversion unit converted 16 analog signals to digital words and stored them in latches. At the end of the A/D conversion, it latched the status of 16 discrete logic lines into another digital word. The signal conversion unit also contained three D/A converters. The signal conversion unit and the digital interface are described in Report No. DGWT 0210-2, Operating Manual and System Description for Digital Processor Number 2. The PDAP buffer amplifier contained 19 operational amplifiers to buffer between the A/D and D/A converters in the signal conversion unit and the analog computer. The PDAP buffer amplifier was the same unit that was used with the PDAP simulation effort. The signal conversion was basically the same design as the signal conversion unit used by PDAP but was a new piece of hardware. The digital interface unit was a new design and a new piece of hardware.

Signal Interfaces. The signal interfaces between the hybrid computer and the DP2 system are shown in Figure 69. The three signal formats are discussed separately below.

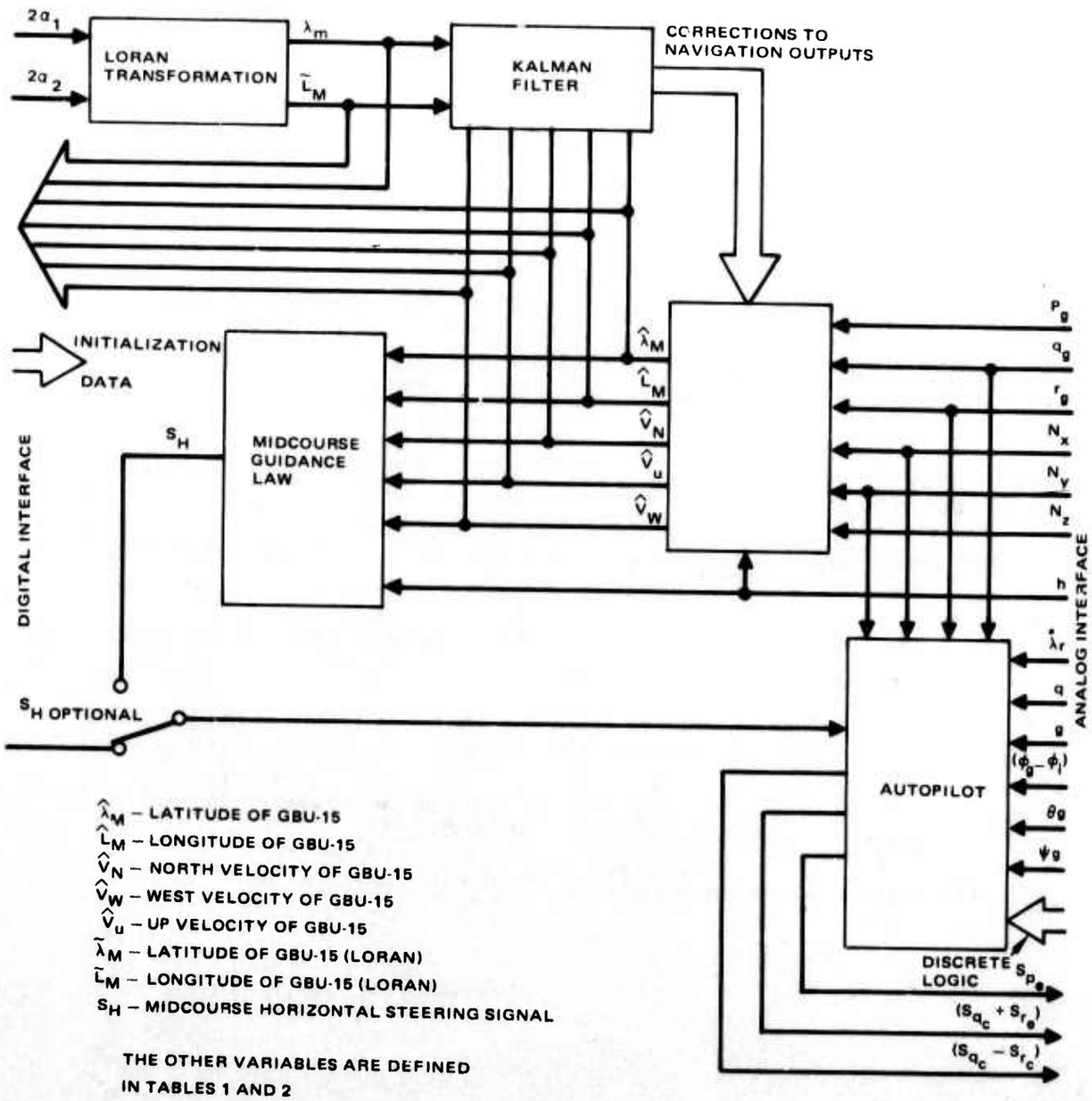


Figure 69. Block Diagram Showing Functions and Data Flow in DP

Analog Signal Interfaces. Of the 16 available analog signals from the analog computer, only 13 were required for the DP2 simulation. Of these, six were used exclusively by the autopilot, four were used both in the autopilot and in the inertial navigation calculations, two were used exclusively for inertial navigation, and one was used both in the inertial navigation and the guidance law computations. The analog interface variables are listed in Table 12.

Three analog signals were generated by the autopilot and sent to the analog computer as flipper commands.

Discrete Logic Signal Interface. Sixteen discrete logic signal lines are output by the analog computer and their states are stored as a 16-bit logic word in the signal conversion unit. Only 12 of the discrete lines must be energized by the analog computer to accomplish the PDAP functions. The locations of the active signal bits in the DP2 logic word, DLVW, and their function in the PDAP are shown in Table 13. The usage of these bits in the DP2 simulation effort is also indicated in the table. This logic word is decoded in the flight control software to select signal sources and guidance modes for weapon flight.

Digital Signal Interface. As indicated in Figure 69, the digital interface accommodated four functions.

(1) Initialization data, which was input to the SIGMA 8 on cards, was sent to DP2.

(2) The distance differences for the simulated LORAN receiver was transmitted to the DP2.

(3) The navigation results from the DP2 were sent to the SIGMA 8 for conversion to analog and printing on a strip chart recorder.

(4) The optional calculation of the midcourse guidance law (computed in the SIGMA 8) was sent to the DP2.

The digital interface signals are listed in Table 14.

TABLE 12. INTERFACE SIGNAL
AND DIGITAL PROCESSOR
(DIGITAL GUIDED WEAP

SIGNAL DESCRIPTION	HYBRID COMPUTER			ANALOG INTERFACE BOX			SIG A/D	
	DIAGRAM SYMBOL	DEFINITION AND SIGNAL	SCALE FACTOR	PIN NO.	AMPLIFIER TYPE	GAIN (V/V)	±10v = RANGE	RESULTI LSB
PITCH - YAW FLIPPER COMMAND	$\delta_{qc} - \delta_{rc}$	+PITCH = NOSE UP	0.666v/o	J4	NON-INVERT	1.0	±15°	0.0293°
PITCH + YAW FLIPPER COMMAND	$\delta_{qc} + \delta_{rc}$	+YAW = NOSE RIGHT	0.666v/o	J5	NON-INVERT	1.0	±15°	0.0293°
ROLL FLIPPER COMMAND	δ_p	ROLL CLOCKWISE (FROM REAR)	0.666v/o	J6	NON-INVERT	1.0	±15°	0.0293°
PITCH ANGLE OF SEEKER	θ_g	SEEKER UP	−0.444v/o	J7	INVERT	0.6	±37.5	0.0183°
ANGLE OF ATTACK	$\alpha_M - \alpha_c$	NOT USED	1v/o	J8	NON-INVERT	0.4		
ROLL RATE GYRO	P_G	ROLL CLOCKWISE	0.5v/o/sec	J9	NON-INVERT	0.2	±100°/sec	0.0488°/
LONGITUDINAL ACCELERATION	N_X	FORWARD	6.66v/g	J10	INVERT	0.2	±7.5/g	0.0037g
PITCH (VERTICAL ACCELEROMETER	N_Z	UP	−3.33v/g	J11	NON-INVERT	0.4	±7.5/g	0.0037g
YAW ACCELERATION	N_Y	RIGHT	6.66v/g	J12	INVERT	0.2	±7.5/g	0.0037g
PITCH SEEKER DERIVED RATE	$\dot{\theta}_g - \dot{\lambda}_q$	NOT USED	−0.555v/o/sec	J13	INVERT	0.4		
YAW SEEKER DERIVED RATE	$\dot{\psi}_g - \dot{\lambda}_r$	NOT USED	0.555v/o/sec	J14	NON-INVERT	0.4		
EO STEERING SIGNAL (PITCH)	$\dot{\lambda}_q$		8.33v/o/sec	J15	NON-INVERT	0.1	±12°/sec	0.0059°/s
EO STEERING SIGNAL (YAW)	$\dot{\lambda}_r$		−8.33v/o/sec	J16	INVERT	0.1	±12°/sec	0.0059°/s
PITCH ALTITUDE	θ_G	NOSE UP	−0.277v/o	J17	INVERT	0.4	±90.25°	0.0441°/s
YAW ALTITUDE	ψ_G	NOSE RIGHT	0.277v/o	J18	NON-INVERT	0.4	±90.25°	0.0441°/s
PITCH RATE GYRO	q_G	NOSE UP	1.11v/o/sec	J19	NON-INVERT	0.2	±45°/sec	0.022°/se
YAW RATE GYRO	r_G	NOSE RIGHT (FROM REAR)	−1.11v/o/sec	J20	NON-INVERT	0.2	±45°/sec	0.022°/se
ROLL GYRO	$\phi_G - \phi_c$	ROLL CLOCKWISE (FROM REAR)	−4.25v/o	J21	NON-INVERT	0.2	±117.5°	0.0574°
ALTITUDE	h	ALTITUDE ABOVE SPHERE OF EARTH	0.002v/ft	J22	INVERT	0.1	50,000	24.41 ft
							DISCRETE LOGIC LIN	

(The reverse

TABLE 12. INTERFACE SIGNALS BETWEEN THE ANALOG COMPUTER AND DIGITAL PROCESSOR (DP) DURING HYBRID SIMULATION (DIGITAL GUIDED WEAPONS TECHNOLOGY PROGRAM)

HYBRID COMPUTER		ANALOG INTERFACE BOX			SIGNAL CONVERSION UNIT (SCU) A/D = 12 BITS D/A = 10 BITS					DIGITAL PROCESSOR 16-BIT DATA WORD
DEFINITION AND SIGNAL	SCALE FACTOR	PIN NO.	AMPLIFIER TYPE	GAIN (V/V)	±10v = RANGE	RESULTING LSB	PHASE OF OUTPUT	BIU ADDRESS (D)	TYPE CONVERSION	SCALE FACTOR MSB
PITCH = NOSE UP	0.666v/o	J4	NON-INVERT	1.0	$\pm 15^{o}$	0.0293^{o}	$\dot{\delta}_{qc} - \delta_{rc}$	192	D/A	7.5^{o}
YAW = NOSE RIGHT	0.666v/o	J5	NON-INVERT	1.0	$\pm 15^{o}$	0.0293^{o}	$\delta_{qc} + \delta_{rc}$	193	D/A	7.5^{o}
OLL CLOCKWISE FROM REAR)	0.666v/o	J6	NON-INVERT	1.0	$\pm 15^{o}$	0.0293^{o}	δ_{p}	194	D/A	7.5^{o}
EEKER UP	−0.444v/o	J7	INVERT	0.6	±37.5	0.0183^{o}	$-\sigma_{g}$	64	A/D	18.75^{o}
OT USED	1v/o	J8	NON-INVERT	0.4			$+\alpha$	65	A/D	
OLL CLOCKWISE	0.5v/o/sec	J9	NON-INVERT	0.2	$\pm 100^{o}$/sec	0.0488^{o}/sec	$+P_{G}$	66	A/D	50^{o}/sec
ORWARD	6.66v/g	J10	INVERT	0.2	±7.5/g	0.0037g	$-N_{X}$	67	A/D	3.75g
	−3.33v/g	J11	NON-INVERT	0.4	±7.5/g	0.0037g	$-N_{Z}$	68	A/D	3.75g
IGHT	6.66v/g	J12	INVERT	0.2	±7.5/g	0.0037g	$-N_{Y}$	69	A/D	3.75g
OT USED	−0.555v/o/sec	J13	INVERT	0.4				70	A/D	
OT USED	0.555v/o/sec	J14	NON-INVERT	0.4				71	A/D	
	8.33v/o/sec	J15	NON-INVERT	0.1	$\pm 12^{o}$/sec	0.0059^{o}/sec	$+\dot{\lambda}_{q}$	72	A/D	6^{o}/sec
	−8.33v/o/sec	J16	INVERT	0.1	$\pm 12^{o}$/sec	0.0059^{o}/sec	$+\lambda_{r}$	73	A/D	6^{o}/sec
OSE UP	−0.277v/o	J17	INVERT	0.4	$\pm 90.25^{o}$	0.0441^{o}/sec	$+\theta_{G}$	74	A/D	45.125^{o}
OSE RIGHT	0.277v/o	J18	NON-INVERT	0.4	$\pm 90.25^{o}$	0.0441^{o}/sec	$+\psi_{G}$	75	A/D	45.125^{o}
OSE UP	1.11v/o/sec	J19	NON-INVERT	0.2	$\pm 45^{o}$/sec	0.022^{o}/sec	$+q_{G}$	76	A/D	22.5^{o}/sec
OSE RIGHT ROM REAR)	−1.11v/o/sec	J20	NON-INVERT	0.2	$\pm 45^{o}$/sec	0.022^{o}/sec	$-r_{G}$	77	A/D	22.5^{o}/sec
OLL CLOCKWISE ROM REAR)	−4.25v/o	J21	NON-INVERT	0.2	$\pm 117.5^{o}$	0.0574^{o}	$-p_{G}$	78	A/D	58.78^{o}
TITUDE ABOVE HERE OF RTH	0.002v/ft	J22	INVERT	0.1	50,000	24.41 ft	$-h$	79	A/D	25,000 ft
					DISCRETE LOGIC LINES			80		

(The reverse of this page is blank)

2

TABLE 13. DISCRETE LOGIC VARIABLES

DLVW BIT	NAME	PDAP FUNCTION	DP2 SIMULATION USAGE
SIGN	WEAPON SEPARATION	ENABLE FLIGHT CONTROL	(NOT USED FUNCTION BY SIGMA 8 INTERRUPT)
14	EOT COMMAND	COMMANDS EO TERMINAL GUIDANCE (IF = 0)	SAME AS PDAP
13	LOBL	SELECT EO TERMINAL GUIDANCE THROUGHOUT FLIGHT (IF = 0)	NOT USED (SET = 1)
12	TRANSITION ENABLE	ENABLE TRANSITION FROM MID-COURSE TO EO TERMINAL GUIDANCE	SAME AS PDAP
11	CWM	SELECT CWW FLIGHT CONTROL	SELECT CWW FLIGHT CONTROL
10	PWM	SELECT PWW FLIGHT CONTROL	NOT USED (ALWAYS = 0)
9	EO TAKEOVER	ENABLE EO/DL MIDCOURSE GUIDANCE	SET = 1 TO ENABLE INERTIAL MIDCOURSE GUIDANCE
8	DMEI	INDICATES DME MODULE INSTALLED	SELECT RATE GYRO INPUTS FOR STABILIZATIONS
7	EO INHIBIT	ENABLE SWITCH FROM EO TO DME GUIDANCE	NOT USED (SET = 0)
6	DME PITCHOVER	ENABLE DME TERMINAL GUIDANCE	NOT USED (SET = 0)
5			
4			
3			
2			
1	EO/DL STEER MINUS	DECREMENT YAW CMD BY 1 degree	NOT USED
0	EO/DL STEER PLUS	INCREMENT YAW CMD BY 1 degree	NOT USED

TABLE 14. VARIABLES PASSED OVER DIGITAL INTERFACE

DESCRIPTION OF VARIABLE INCOMING VARIABLES (SIGMA 8 → DP2)	SYMBOL	FULL SCALE VALUE	FREQUENCY
1. DISTANCE FROM GBU-15 TO MASTER LORAN SECTION MINUS DISTANCE FROM GBU-15 TO SLAVE 1	$2a_1$	607,600 ft	50 Hz
2. DISTANCE FROM GBU-15 TO MASTER LORAN STATION MINUS DISTANCE FROM GBU-15 TO SLAVE 2	$2a_2$	607,600 ft	50 Hz
3. MIDCOURSE STEERING SIGNAL	S_H	30 units	50 Hz
4. LAUNCH VELOCITY (NORTH)	V_N	32,767 ft/sec	I.C.
5. LAUNCH VELOCITY (WEST)	V_W	32,767 ft/sec	I.C.
6. LAUNCH VELOCITY (UP)	V_U	32,767 ft/sec	I.C.
7. LAUNCH LATITUDE (+ NORTH)	λ_L	π radians	I.C.
8. LAUNCH LONGITUDE (+ EAST)	L_L	π radians	I.C.
9. LAUNCH ALTITUDE	h_L	50,000 ft	I.C.
10. LAUNCH PITCH ANGLE (ORDER p, y, r)	θ_o	π radians	I.C.
11. LAUNCH YAW ANGLE	ψ_o	π radians	I.C.
12. LAUNCH ROLL ANGLE	ϕ_o	π radians	I.C.
13. TARGET LATITUDE	λ_T	π radians	I.C.
14. TARGET LONGITUDE	L_T	π radians	I.C.
15. DESIRED BEARING ANGLE TO TARGET	σ_{DT}	π radians	I.C.
16. CHECKPOINT LATITUDE	$\lambda_{CP(3)}$	π radians	I.C.
17. CHECKPOINT LONGITUDE	$L_{CP(3)}$	π radians	I.C.
18. CHECKPOINT ALTITUDE	$h_{CP(3)}$	50,000 ft	I.C.
19. DESIRED BEARING ANGLE TO CHECKPOINT	$\sigma_{DCP(3)}$	π radians	I.C.
20. CHECKPOINT LATITUDE	$\lambda_{CP(2)}$	π radians	I.C.
21. CHECKPOINT LONGITUDE	$L_{CP(2)}$	π radians	I.C.
22. CHECKPOINT ALTITUDE	$h_{CP(2)}$	50,000 ft	I.C.
23. DESIRED BEARING ANGLE TO CHECKPOINT	$\sigma_{DCP(2)}$	π radians	I.C.
24. CHECKPOINT LATITUDE	$\lambda_{CP(1)}$	π radians	I.C.
25. CHECKPOINT LONGITUDE	$L_{CP(1)}$	π radians	I.C.
26. CHECKPOINT ALTITUDE	$h_{CP(1)}$	50,000 ft	I.C.
27. DESIRED BEARING ANGLE TO CHECKPOINT	$\sigma_{DCP(1)}$	π radians	I.C.
28. DISTANCE FROM TARGET TO MASTER LORAN STATION MINUS DISTANCE FROM TARGET TO SLAVE 1	ΔD_1	607,600 ft	I.C.
29. DISTANCE FROM TARGET TO MASTER LORAN STATION MINUS DISTANCE FROM TARGET TO SLAVE 2	ΔD_2	607,600 ft	I.C.
30. ELEMENT OF MATRIX FOR LORAN TRANSFORMATION	b_{11}	4 ft/ft	I.C.
31. ELEMENT OF MATRIX FOR LORAN TRANSFORMATION	b_{12}	4 ft/ft	I.C.
32. ELEMENT OF MATRIX FOR LORAN TRANSFORMATION	b_{21}	4 ft/ft	I.C.
33. ELEMENT OF MATRIX FOR LORAN TRANSFORMATION	b_{22}	4 ft/ft	I.C.

Table 14 (Concluded)

DESCRIPTION OF VARIABLE OUTGOING VARIABLES (DP2 SIGMA 8)	SYMBOL	FULL SCALE VALUE	FREQUENCY
1. TIME	t	3276.7 sec	50 Hz
2. LATITUDE OF GBU-15 ESTIMATED BY NAVIGATOR	$\hat{\lambda}_M$	π radians	50 Hz
3. LONGITUDE OF GBU-15 ESTIMATED BY NAVIGATOR	$\hat{L}_M$	π radians	50 Hz
4. NORTH VELOCITY OF GBU-15 ESTIMATED BY NAVIGATOR	$\hat{V}_N$	32,767 ft/sec	50 Hz
5. WEST VELOCITY OF GBU-15 ESTIMATED BY NAVIGATOR	$\hat{V}_W$	32,767 ft/sec	50 Hz
6. UP VELOCITY OF GBU-15 ESTIMATED BY NAVIGATOR	$\hat{V}_U$	32,767 ft/sec	50 Hz
7. DIFFERENCE IN LATITUDE	$\hat{\lambda}_M - \lambda_T$	1.4063 deg	50 Hz
8. DIFFERENCE IN LONGITUDE	$\hat{L}_M - L_Y$	1.4063 deg	50 Hz
9. ERROR IN LATITUDE CALCULATED IN KALMAN FILTER			50 Hz
10. ERROR IN LONGITUDE CALCULATED IN KALMAN FILTER			50 Hz
11. MIDCOURSE STEERING SIGNAL	S_H	30 units	50 Hz

DP2 Software

This subsection describes the software for DP2 which was used in the simulation. The system executive is functionally identical to the executive used in DP1. The Kalman filter, used for smoothing the LORAN data, and the inertial navigation program were modified versions of the DP1 software. These programs had to be translated from the DP1 language to the DP2 language. The flight control software was adapted from the PDAP flight control software. It is functionally the same but was rewritten to put in a modular form. The other program modules were written especially for the simulation.

The complete documentation of the software is contained in the Digital Processor Software Development Report, Report No. DGWT 0165-1.

Functional Descriptions. This subsection describes the functions performed by the DP2 software during the simulations. A functional block diagram of the DP2 software computational elements is shown in Figure 70. In addition to these computational elements, the System Management and Executive software, in combination, control the execution of the DP2 software in accordance with the simulation requirements (interpretation of hardware interrupts and control of the weapon bus communications with the hybrid computer). The following paragraphs describe the software elements including the relationships between weapon functions and software tasks.

Flight Control. The flight control software is contained in three tasks: 27, 28, and 29. Two additional tasks (30 and 10) are associated with the flight control function and are concerned with the formatting of sensor data and transfer of the data over the weapon bus.

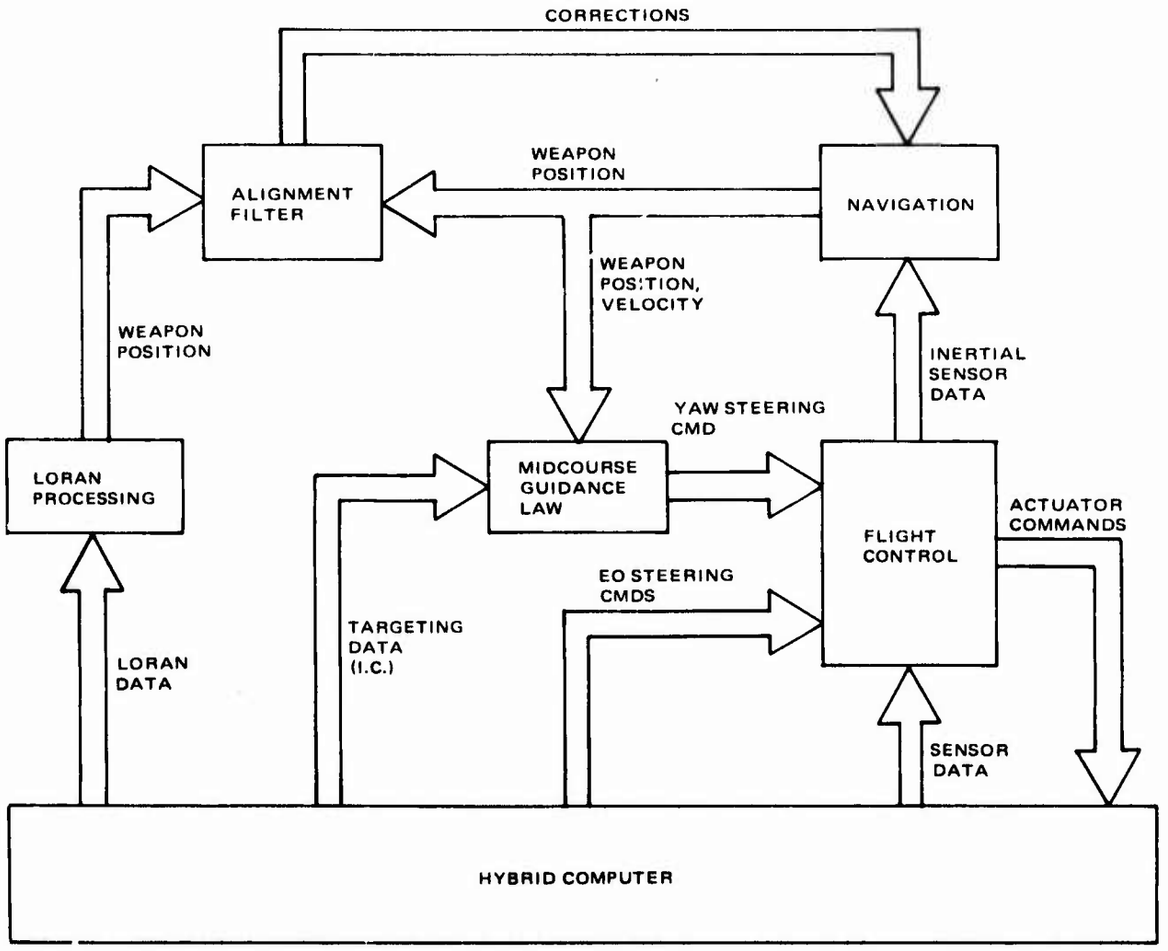


Figure 70. DP2 Software Functional Block Diagram

Task 29 (FCADX) performs rate damping (stabilization) loop computations for either the PWW or CWW airframes as designated by the flight control logic. These computations are performed using one of three types of sensor inputs with the selection controlled by the flight control logic. Actuator commands are output to the signal conversion unit via the weapon bus to control the weapon. Only the CWW airframe was used, and rate gyro signals were used for airframe stabilization calculations. This task queues the steering calculation task (27 or 28) at a 50-Hz rate. Task 29 also contains a module to format rate gyro and accelerometer data for use by the navigation software and queue the navigation software at 100 Hz.

Task 27 (COUTER) and Task 28 (COUTER) perform steering computations for the CWW and PWW airframes, respectively. Both of these tasks embody a number of alternative guidance modes which may be selected as a function of flight time, guidance sensors, and airframe dynamics. The selection is performed by a logic module (FCLOG) which is called by both tasks. The steering modes and mode selection logic are identical to the functions described in DGWT 0160-1, PDAP Software Development, with the following exceptions. The yaw heading hold mode for CWW (only

COUTER was used in the simulation) was modified to use yaw steering commands from an inertial midcourse guidance law module rather than the alternative DME or EO data link commands of the PDAP flight control software. Terminal guidance was performed using steering commands from a simulated EO seeker.

Top level flow charts of FCADX and COUTER are shown in Figures 71 and 72. The flight control mode and sensor selection is initialized by calling FCLOG on the first iteration of FCADX after simulated weapon separation. The selection is based on discrete logic variables received from the analog computer via the weapon bus.

Navigation. The navigation software calculates weapon position, velocity, and altitude parameters which are used in the midcourse guidance law. Several modifications were made in the navigation software used in DP1 tests at CIGTF for compatibility with the simulation. The entire gyro and accelerometer compensation software routine of IMU100 (Task 9) was deleted since (1) this routine was specifically written for the Hamilton Standard IMU and (2) the gyro and accelerometer outputs from the simulation could be treated as perfect (the identical sensor data is processed in the simulation to generate reference data). However, a requirement exists for gyro bias compensation, due to A/D conversion errors, and the required software was inserted in the IMU data accumulation routine in FCADX.

The major functional difference between the simulated IMU data and a true IMU is that the hybrid simulation assumes a flat, non-rotating earth. Therefore, the navigation frame rotation routine was deleted from IMU100, and the coriolis correction routine was deleted from IMU10 (task 11). A top level flowchart of the navigation software tasks is shown in Figure 73. The changes, with respect to DP1 software, are indicated by the strikeout lines.

The navigation software parameters are initialized by IMUINT (Task 14) in accordance with digital reference data from the SIGMA 8 computer. This task is called as a subroutine by System Management Task 45. During simulated flight, the navigation parameters are updated in accordance with the rate gyro and accelerometer data from the analog computer. Corrections are made to the navigation parameters in flight by the alignment filter on the basis of position data from a LORAN subsystem.

Alignment. The DP2 alignment software is virtually a duplicate of the DP1 alignment. However, the system operating procedures and simulation techniques have caused some functional differences. The major procedural difference is that the filter operates throughout the midcourse phase of weapon flight rather than only during a prelaunch alignment phase.

In the DP2 system, the reference data is generated by processing LORAN signals and only position data is available. Therefore, the filter was required to operate in a position-match-only mode. The DP2 filter outputs were identical to the outputs of the DP1 filter: estimates of

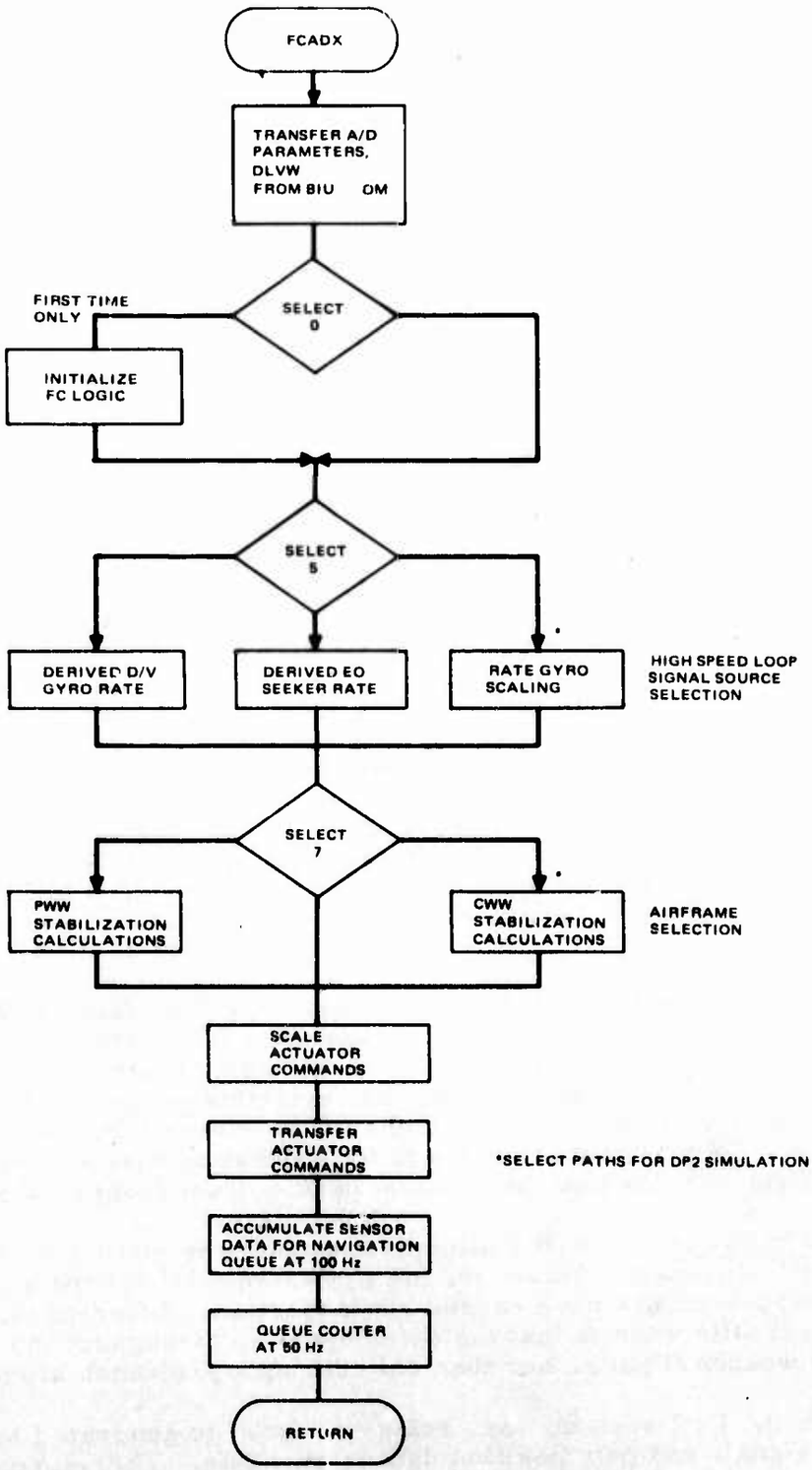


Figure 71. FCADX Flowchart

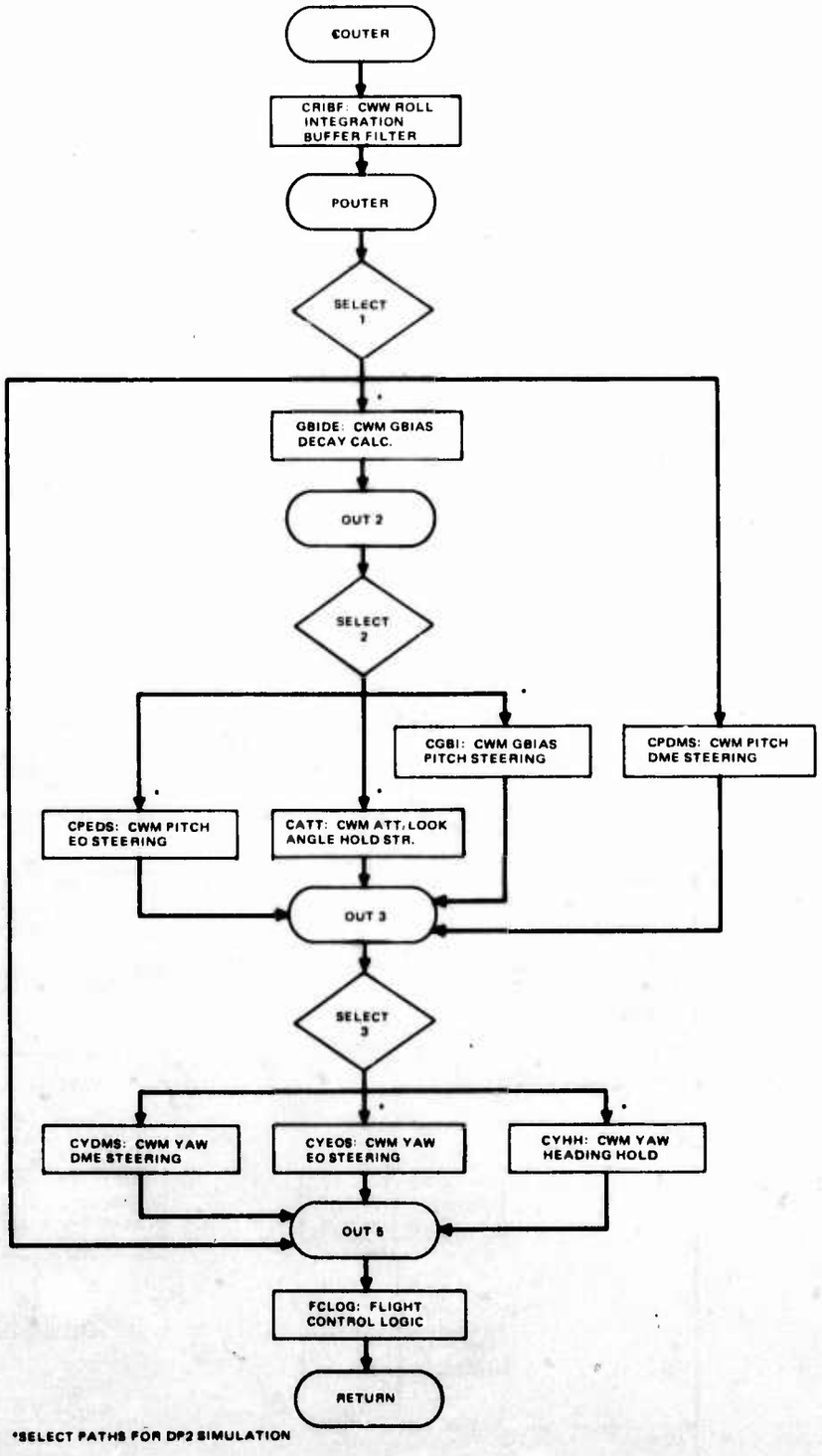


Figure 72. Couter Flowchart

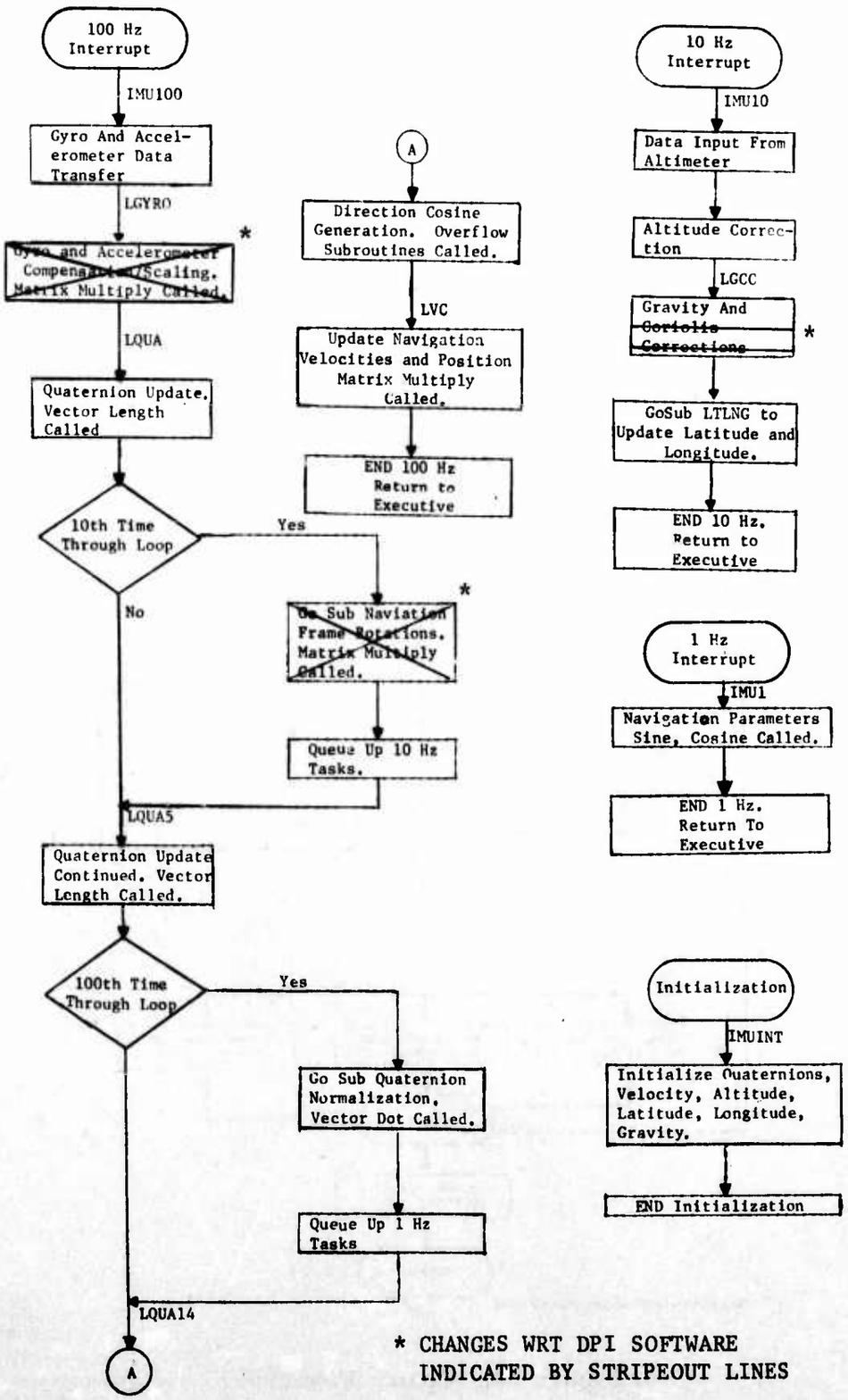


Figure 73. Navigation Software Task Flowchart

misalignment, position errors, velocity errors, and gyro biases. The update interval for gyro bias estimation was reduced from 5 minutes to 12 seconds. This change was caused by the expected value of the offset in the analog-to-digital conversion of the gyro data which was on the order of 40°/hour. Similar offsets occur in the accelerometer data, but accelerometer biases are effectively equivalent to misalignments which are estimated at each iteration of the filter.

The alignment software is contained in Tasks 16, 17, 18, 21, 22, and 23. The filter is initialized by FINI, Task 16, which is queued by System Management Task 48. The 100-Hz alignment routine, CENTSK (Task 22), is queued by flight control Task 29 (FCADX). Tasks 23 (TYSNC) and 17 (DRITSK) are queued by the LORAN transformation routine (Task 25) when it generates position data (nominal 3.3-Hz rate). The filter control feedback routine (CONTROL, Task 21) is queued by DRITSK on each iteration to correct the navigator altitude, velocity, and position parameters. The gyro bias correction routine (GYRO, Task 18) is called as a subroutine by CONTROL at 12-second intervals.

Midcourse Guidance Law. The guidance law calculations are used to control the heading of the missile during the midcourse phase of flight. In the simulation, only yaw steering commands were generated by the guidance law software. Midcourse steering in the pitch plane came from the existing pitch steering modes of the GBU-15 flight control subsystem. The guidance law, as implemented, flies the simulation from one checkpoint target to another until it determines that it is on the last segment. This means that the next target is the actual target, and terminal guidance is enabled. As many as three intermediate checkpoint targets can be used to shape the missile flight path to the target so that the trajectory over the ground could have up to four segments with different headings.

The equation for midcourse guidance which was implemented is of the form:

$$\text{Steering Command} = K_1 \dot{\sigma}_H + K_2(\sigma_H - \sigma_{HD})$$

where, σ_H is the missile heading angle measured from north, σ_{HD} is the desired heading to the target, and $\dot{\sigma}_H$ is the time rate of change of the missile heading. The constants K_1 and K_2 are gain terms which control the relative importance of the heading angle steering versus the angle rate steering terms.

If σ values are in radians and $\dot{\sigma}$ values are in radians/second, the values of K_1 and K_2 that were used are 62.8 and 1.57, respectively. These values were selected so that the GBU-15 turning rate was approximately equal to $4\dot{\sigma}_H + 0.1(\sigma_H - \sigma_{HD})$. In the calculation of $\dot{\sigma}$, the range to the checkpoint target is used as a divisor. To avoid division by zero, the next checkpoint target was selected when the range to a checkpoint target was less than a thousand feet. At long ranges from the next checkpoint target, the $\dot{\sigma}_H$ term is reduced in effectivity compared to the bearing angle term and the missile tends to fly toward the correct heading angle. When this angle is approached and as range to the next checkpoint target is diminished, the $\dot{\sigma}_H$ term gains prominence and the steering approximates proportional navigation, which gives a small miss distance with respect to the checkpoint target. This guidance law seemed to work very well except for the

case where the initial heading differed from the desired heading by a large amount. This caused the missile to turn away from the target and fly to the correct heading before turning toward the target. To correct this problem, the $(\sigma_H - \sigma_{HD})$ term was limited to one degree. This limit was used in the simulation with satisfactory results. Without the one-degree limit on the $(\sigma_H - \sigma_{HD})$ term, the simulation has the missile making a hard turn toward the desired heading, as shown in Figure 74. This figure shows typical yaw plane trajectories corresponding to $\sigma_{HD} = 30^o$, $\sigma_H\ (t = 0) = 0^o$ with and without limiting of the $(\sigma_H - \sigma_{HD})$ turn in the guidance law calculations. The limiting of the $(\sigma_H - \sigma_{HD})$ term causes the GBU-15 to fly a low valued constant acceleration turn until the desired heading is reached.

This midcourse guidance law was implemented both in the SIGMA 8 software and in the DP2 software (GDLW, Task 60). Initialization of the DP2 guidance law software parameters in accordance with data furnished by the SIGMA 8 computer is performed by the subroutine, GDLWI, which is called by System Management Task 46, TGTINT. Alternatively, the SIGMA 8 calculation of yaw midcourse steering commands may be used. These commands are made available to the flight control through System Management Task 50, GUID.

Top level flow charts of GDLWI and GDLW are shown in Figures 75 and 76.

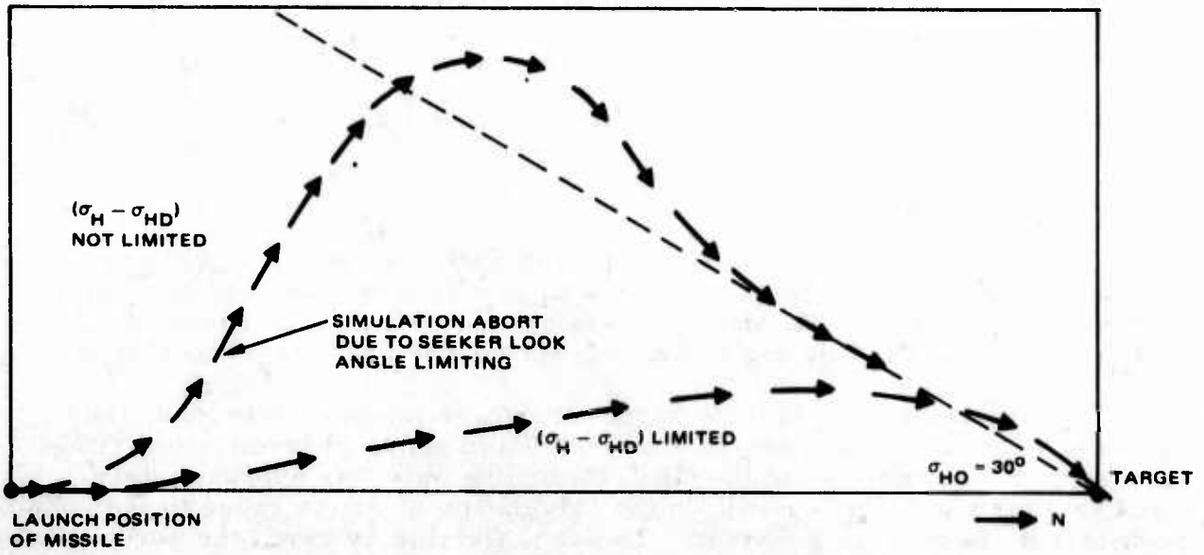


Figure 74. Illustration of Trajectories Resulting from Guidance Law With and Without Limiting of $(\sigma_H - \sigma_{HO})$ Term in Calculations

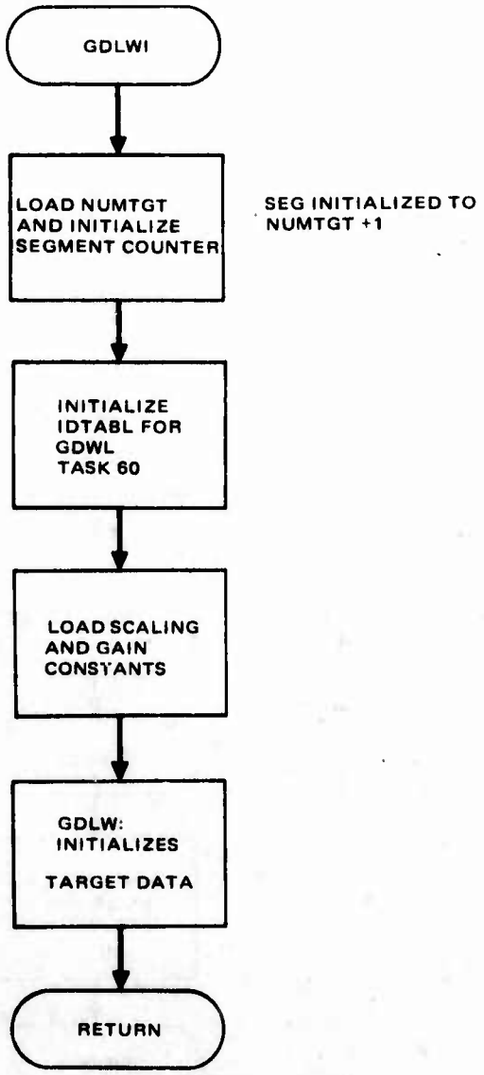


Figure 75. Guidance Law Software Initialization

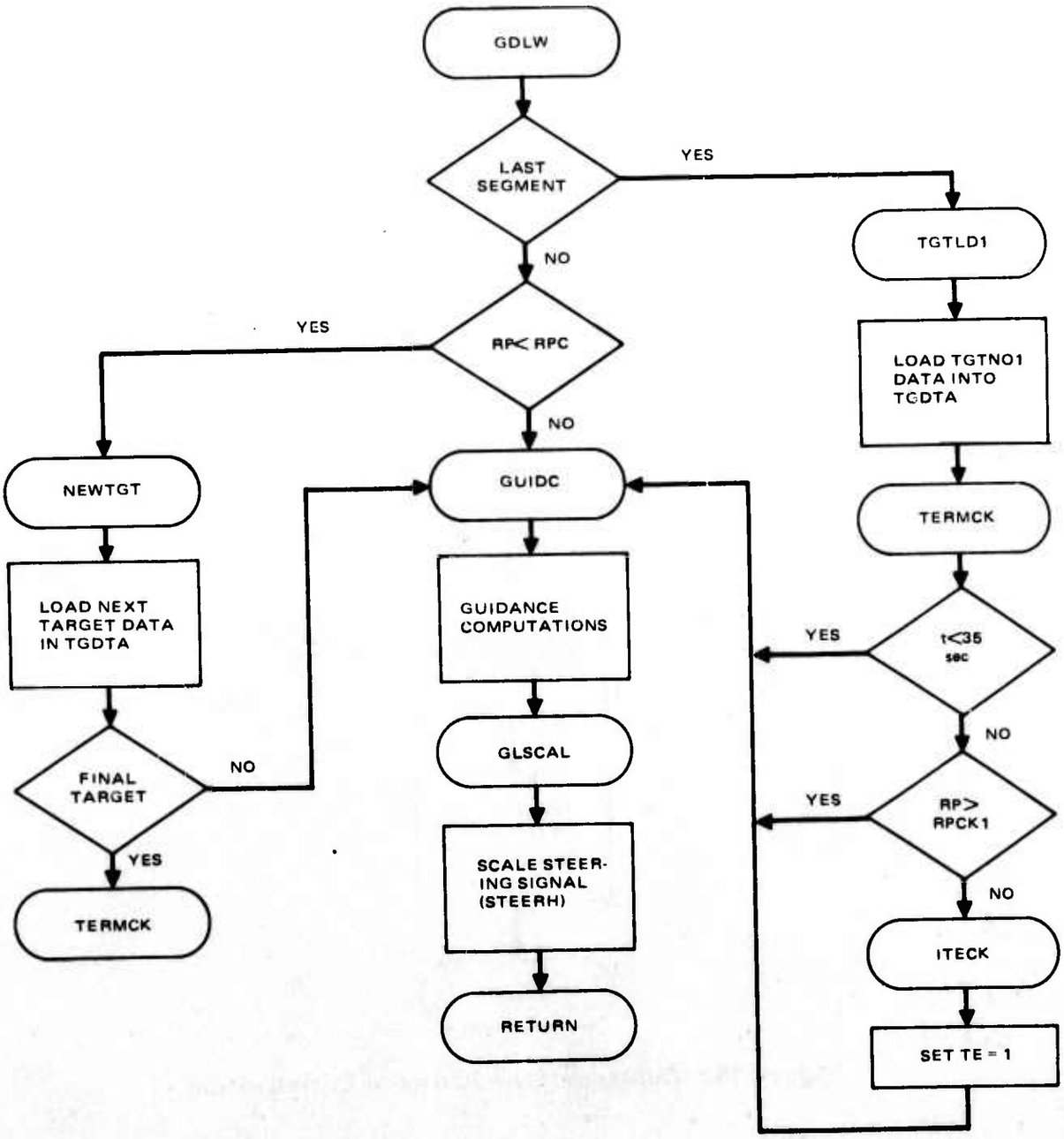


Figure 76. Guidance Law Flow Diagram (TASK 60)

LORAN. The LORAN software (LRNX, Task 25) calculates weapon position based on digital data from the SIGMA 8 computer. The SIGMA 8 simulates the LORAN receiver to produce path length differences for the two slave stations in the simulated LORAN network. These two difference parameters are calculated and sent to DP2 at a 50-Hz rate. LRNX is queued by System Management Task 48 at the 50-Hz rate, but the computation of weapon position is performed at a 3.3 Hz under the control of a counter within LRNX. Weapon position (latitude, longitude) is calculated using a linear transformation from slave station path differences to weapon position differences relative to the target coordinates. This linearization of the hyperbolic LORAN grid is relative to the target position such that the algorithm error approaches zero as the weapon approaches the target.

Initialization of the LRNX software parameters is performed by LORINT (Task 47) which is queued by Task 54, both of which are in the System Management section. This routine sets the target distance difference and transformation matrix parameters to values input from the SIGMA 8 computer.

System Management Software. The system management software, in general, is responsible for interpreting the system interrupts (clock and SIGMA 8), and controlling the weapon bus transmissions and the sequencing of the other software modules. In addition, the System Management contains the routines which initialize the DP2 software parameters. DP2 initialization is performed in two parts. The first part of initialization is performed automatically when the DP2 is started by pushing the RUN button on the control panel. This routine initializes the DP2 hardware facilities (memory, index registers, and flags) to the appropriate states and sets up the Task ID and message parameter tables. Following this initialization, the DP2 enters an idle loop which is continually executed except for responding to interrupts. The second part of initialization sets mission-dependent parameters for the various computational functions to values input from the SIGMA 8 computer (simulated avionics interface). This part of initialization is performed by a number of routines in response to SIGMA 8 interface interrupts.

The idle loop has two functions: maintaining a measure of idle time to facilitate time loading calculations, and allowing monitoring of any location in operand memory via the control panel display. Two conditional branches are included in the idle loop: The first branch establishes an alternative idle loop path in which the weapon separation discrete from the analog computer is accessed and monitored. When weapon separation is detected, the periodic tasks are connected to the clock interrupts, and the alternative idle loop is disabled. This branch is initially controlled by a control panel switch and is enabled only if the SIGMA 8 is not connected. If the SIGMA 8 is connected, System Management Task 55 performs the weapon separate function. The second branch provides an idle loop exit to the start of the initialization routine and is used to allow successive simulation runs without resetting DP2. This branch is controlled by a flag which is set in System Management Task 51 in response to a SIGMA 8 interrupt.

The other System Management tasks which interpret interrupts and control weapon bus transmissions are discussed in the System Operating Sequences.

System Operating Sequences. All DP2 operating sequences are initiated by hardware interrupts. The selection of computational sequences is performed by the appropriate System Management software task after receipt of the interrupt from the SIGMA 8 computer or the internal clock. The following paragraphs describe the processing associated with each of the system interrupts. These sequences include System Management, Executive, and applications software tasks as appropriate.

Sigma 8 Interrupts. The following sequences pertain to the SIGMA 8 interrupts and the corresponding SIGMA 8 data and shown in Figure 77 and Table 15. In the normal method of operation all flags on the front panel are set low, then the reset button is depressed, followed by the run button. This places the DP in the idle loop. A sequence of interrupts from the SIGMA 8, which is shown in Figure 78, is then sent to the DP. Table 15 describes the variables sent to the DP and the reaction of the DP to the SIGMA 8 interrupts. Interrupts 51 through 54 cause the initialization of the system. Interrupt 51 which sets flag 48 high to cause the DP to go back and reinitialize itself was included for multiple runs,all controlled from the SIGMA 8. Interrupt 55 is sent after initialization at the beginning of the run. This interrupt replaces the weapon separate detection used with the analog interface only. Once the run begins, the SIGMA 8 continues to send a periodic sequence of interrupts, namely, 56 followed by 57. These interrupts are not synchronous with those initiated by the clock within the DP.

System Initialization Interrupt Sequence. When the SIGMA 8 computer writes 51 into the control word latch of the D/D interface, an interrupt with ID = 51 is sent to the DP2 (see Figure 78); the executive module, IBUSIN, is called when the interrupt is received by DP2 and queues up Task 51 (INITSY) for execution. This System Management task sets a flag (49). When the task queue is emptied and execution of the IDLE loop is resumed, the flag (49) is detected and subroutine INIT is called to initialize the operand memory, flags, and index registers. Upon completion of initialization, execution returns to the IDLE loop until another interrupt occurs.

Pseudo-Alignment Interrupt Sequence. The initial position, orientation, and velocity are written into the D/D interface buffer memory, and then 52 is written into the control word latch by the SIGMA 8 computer. This causes interrupt 52 to be sent to DP2 (see Figure 79). The executive module, IBUSIN, is called when the interrupt is received and queues up Task 52 (POSITX). POSITX calls the executive module DBSUPR with a pointer to the appropriate data message parameters. DBSUPR initiates the transfer of the initial position, orientation, and velocity of the weapon via the weapon data bus to DP2. The completion of the data transfer generates a transfer complete interrupt which calls the executive module, DBCOMP, which then queues Task 45 (INITPS). INITPS moves

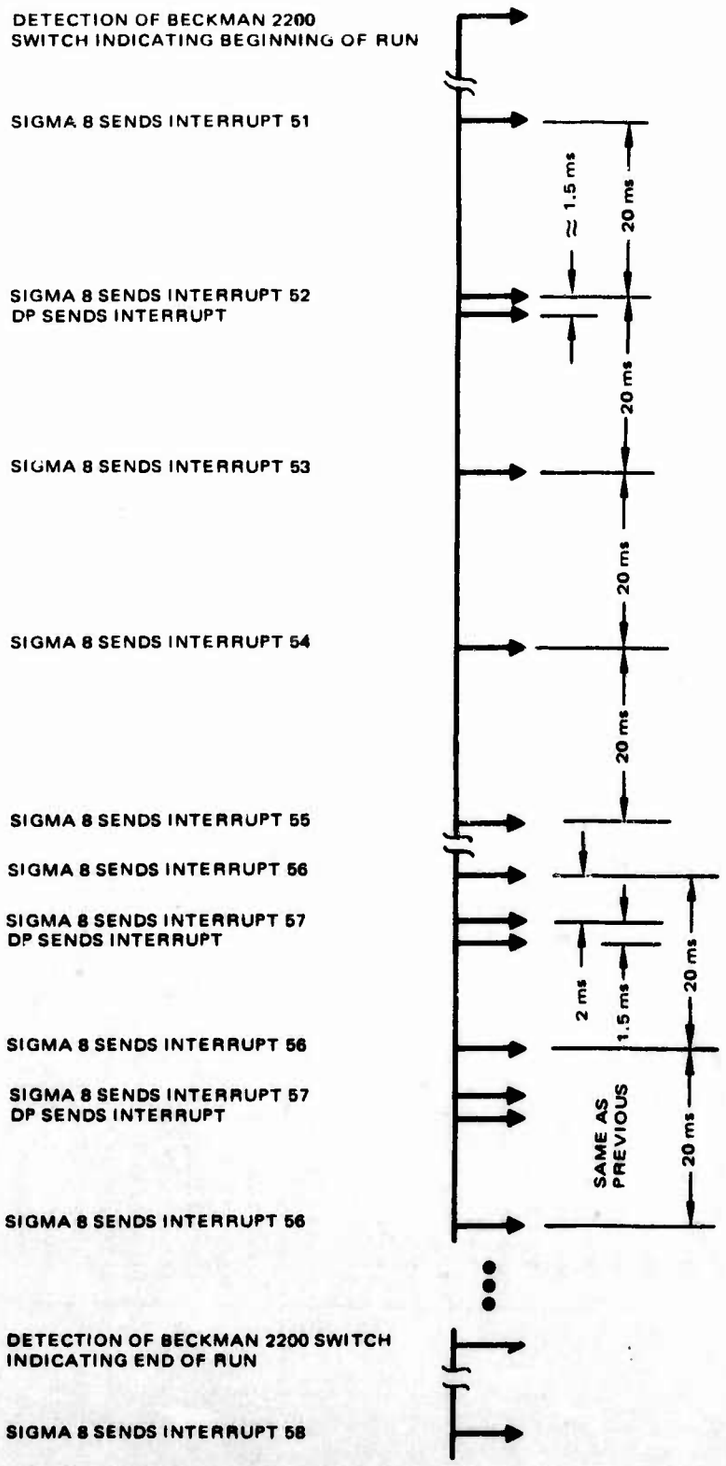


Figure 77. Sequence of Interrupts from SIGMA 8 to DP

TABLE 15. SIGMA 8 INTERRUPTS AND RESPONSES OF THE DIGITAL PROCESSOR (DP)

SIGMA 8 INTERRUPT NO.	DATA FROM SIGMA 8		SCALE FACTOR MSB	BIU #3 ADDRESS	RESPONSE OF THE DP TO INTERRUPT
51	NO DATE IS TRANSFERRED		---	-	SET A FLAG WHICH CAUSES THE DP TO REINITIALIZE WHEN CONTROL TO RETURNED TO THE IDLE LOOP.
52	GBU–15 NORTH VELOCITY	(LSW)		0	TRANSFER DATA OVER WEAPON BUS TO THE INPUT BUFFER AND PLACE TASK 45 IN THE QUEUE. TASK 45 MOVES DATA TO PROPER MEMORY LOCATIONS, CALLS THE NAVIGATOR INITIALIZATION SUBROUTINE, AND INITIATES THE TRANSFER OF OUTPUT DATA TO BIU 3. WHEN DATA TRANSFER IS COMPLETE TASK 49 IS PLACED IN THE QUEUE.
	GBU–15 NORTH VELOCITY (	(MSW)	16,384 ft/sec	1	
	GBU–15 WEST VELOCITY	(LSW)		2	
	GBU–15 WEST VELOCITY	(MSW)	16,384 ft/sec	3	
	GBU–15 UP VELOCITY		16,384 ft/sec	4	
	GBU–15 LATITUDE	(LSW)		5	
	GBU–15 LATITUDE	(MSW)	π/2 rad	6	
	GBU–15 LONGITUDE	(LSW)		7	
	GBU–15 LONGITUDE	(MSW)	π/2 rad	8	
	GBU–15 ALTITUDE		25,000 ft	9	
	GBU–15 EULER ANGLE PITCH		π/2 rad	10	
	GBU–15 EULER ANGLE YAW		π/2 rad	11	
	GBU–15 EULER ANGLE ROLL		π/2 rad	12	
53	NUMBER OF CHECKPOINTS		16,384	0	TRANSFER DATA OVER WEAPON BUS TO THE INPUT BUFFER: THEN PERFORM TASK 46 WHICH MOVES THE DATA INTO THE PROPER MEMORY LOCATION.
	TARGET LATITUDE	(LSW)		1	
	TARGET LATITUDE	(MSW)	π/2 rad	2	
	TARGET LONGITUDE	(LSW)		3	
	TARGET LONGITUDE	(MSW)	π/2 rad	4	
	ALTITUDE OF TARGET		25,000 ft	5	
	DESIRED ANGLE OF APPROACH		π/2 rad	6	
	LATITUDE OF CHECKPOINT	(LSW)		7	
	NEAREST TARGET	(MSW)	π/2 rad	8	
	LONGITUDE OF CHECKPOINT	(LSW)		9	
	NEAREST TARGET	(MSW)	π/2 rad	10	
	ALTITUDE OF CHECKPOINT		25,000 ft	11	
	DESIRED ANGLE OF APPROACH		π/2 rad	12	
	LATITUDE OF CHECKPOINT	(LSW)		13	
	SECOND IN PROXIMITY	(MSW)	π/2 rad	14	
	LONGITUDE OF CHECKPOINT	(LSW)		15	
	SECOND IN PROXIMITY	(MSW)	π/2 rad	16	
	ALTITUDE OF CHECKPOINT		25,000 ft	17	
	DESIRED ANGLE OF APPROACH		π/2 rad	18	

TABLE 15. SIGMA 8 INTERRUPTS AND RESPONSES OF THE DIGITAL PROCESSOR (DP) (CONCLUDED)

SIGMA 8 INTERRUPT NO.	DATA FROM SIGMA 8		SCALE FACTOR MSB	BIU #3 ADDRESS	RESPONSE OF THE DP TO INTERRUPT
53	LATITUDE OF CHECKPOINT	(LSW)		19	
	THIRD IN PROXIMITY	(MSW)	$\pi/2$ rad	20	
	LONGITUDE OF CHECKPOINT	(LSW)		21	
	THIRD IN PROXIMITY	(MSW)	$\pi/2$ rad	22	
	ALTITUDE OF CHECKPOINT		25,000 ft	23	
	DESIRED ANGLE OF APPROACH		$\pi/2$ rad	24	
54	TARGET DISTANCE FROM LORAN MASTER STATION	(LSW)		0	TRANSFER DATA OVER WEAPON BUS TO THE INPUT BUFFER AND PLACE TASK 47 IN THE TASK QUEUE. THIS TASK MOVES THE DATA INTO THE PROPER MEMORY LOCATIONS AND CALLS GUIDANCE INITIALIZATION SUBROUTINE.
	MINUS DISTANCE FROM SLAVE 1	(MSW)	303,800	1	
	TARGET DISTANCE FROM LORAN MASTER STATION	(LSW)		1	
	MINUS DISTANCE FROM SLAVE 2	(MSW)	303,800	3	
	MATRIX ELEMENTS	B11	2 ft/ft	4	
	FOR LORAN	B12	2 ft/ft	5	
	TRANSFORMATION	B21	2 ft/ft	6	
		B22	2 ft/ft	7	
55	NO DATA		---	---	STARTS THE INTERRUPT CLOCK.
56	GBU–15 DISTANCE FROM LORAN MASTER STATION	(LSW)		0	TRANSFER DATA OVER WEAPON BUS TO THE INPUT BUFFER; THEN PLACE TASK 48 INTO THE QUEUE. THIS TASK MOVES DATA INTO MEMORY AND PLACES TASK 25 INTO THE TASK QUEUE.
	MINUS DISTANCE FROM SLAVE 1	(MSW)	303,800 ft	1	
	GBU–15 DISTANCE FROM LORAN MASTER STATION	(LSW)		2	
	MINUS DISTANCE FROM SLAVE 2	(MSW)	303,800 ft	3	
57	MIDCOURSE GUIDANCE STEERING SIGNAL		15 UNITS	0	TRANSFER DATA OVER WEAPON BUS TO THE INPUT BUFFER; THEN PLACE TASK 50 INTO TASK QUEUE. TASK 50 MOVES THE STEERING SIGNAL INTO MEMORY, THE OUTPUT SIGNAL INTO THE OUTPUT BUFFER, AND INITIATES OUTPUT TRANSFER TO BIU 3. WHEN DATA TRANSFER IS COMPLETE, TASK 49 IS PLACED IN THE QUEUE. TASK 49 SENDS A CONTROL WORD (INTERRUPT) CONTAINING THE NUMBER OF DATA WORDS TRANSFERRED TO BIU 3.
58	NO DATA		---	---	STOP THE INTERRUPT CLOCK.

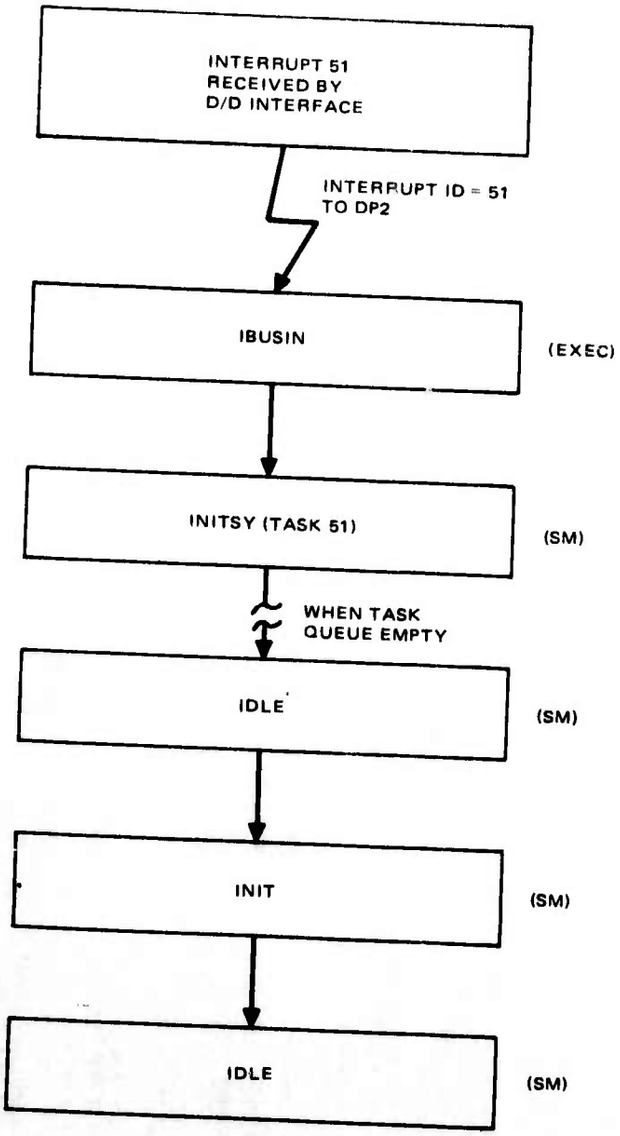


Figure 78. System Initialization Interrupt Sequence

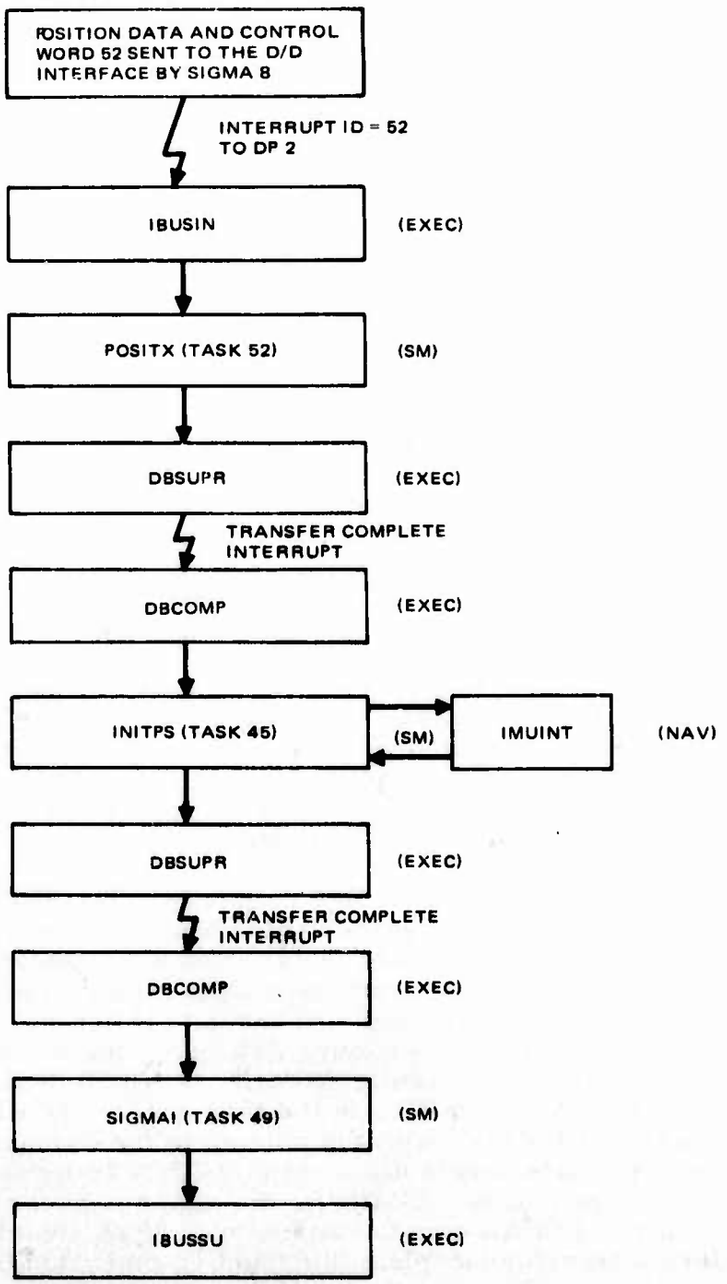


Figure 79. Pseudo-Alignment Interrupt Sequence

the data into operand memory locations, and the output buffer then calls IMUINT. IMUINT initializes the strapdown inertial navigation calculations. This replaces the alignment sequence which would be required to accomplish the same results in an actual weapon launch. When execution returns from IMUINT to INITPS, then the executive module DBSUPR is called with the pointer for the appropriate data message parameters to send the initial position and velocity back to the D/D interface for plotting on the hybrid computer equipment. DBSUPR initiates the transfer of this output data via the weapon data bus to the D/D interface. When the data transfer is completed, a transfer complete interrupt is generated. This calls the executive module DBCOMP, which places Task 49 (SIGMAI) into the task queue. SIGMAI calls the executive module, IBUSSU, with the parameters to send an interrupt to BIU3 (D/D interface). The ID of the interrupt sent to the D/D interface is 17, the number of words of data to be read by the SIGMA 8 computer. IBUSSU sends the interrupt or control word.

Target Information Interrupt Sequence. The SIGMA 8 computer (Σ8) sends the latitude, longitude, altitude, and desired angle of approach of the actual target and checkpoints on the way to the D/D interface. Control word 53 is then sent to the latch. This causes an interrupt with ID = 53 to be sent to the DP2. Upon receipt of the interrupt executive module, IBUSIN, is called (see Figure 80). IBUSIN places Task 53 (TGTDTX) into the task queue. TGTDTX calls the executive module DBSUPR with a pointer to the location where the parameters which define the data transfer across the weapon data bus are stored. The module, DBSUPR, initiates the data transfer. When the data transfer is completed, a transfer complete interrupt is generated. The executive module, DBCOMP, is called by the interrupt which then places Task 46, TGTINT, in the task queue. TGTINT move the target data from the input buffer into operand memory and calls subroutine, GUIDI, which performs the calculations necessary to begin the midcourse steering guidance law.

LORAN Initialization Interrupt Sequence. The SIGMA 8 computer sends the LORAN coordinates of the target (two distance differences) and the four elements of a matrix used to describe the transformation from change in distance differences from the target to North-South distance difference and East-West distance difference. This data is followed by a control word which is 54 (see Figure 81). An interrupt with the ID = 54 is sent to DP2. Executive module, IBUSIN, is called by the interrupt. IBUSIN places Task 54, LORIDX, in the task queue. LORIDX calls the executive module, DBSUPR, with the pointer to the operand memory location where the parameters describing the data transfer over the weapon data bus are stored. DBSUPR, the data bus supervisor, causes the transfer of the LORAN data to commence. Upon completion of the data transfer, a transfer complete interrupt is generated by the hardware. This interrupt causes the executive module, DBCOMP, to execute. DBCOMP places Task 47, LORINT, in the task queue. LORINT moves the LORAN initialization data from the input buffer to the appropriate operand memory locations.

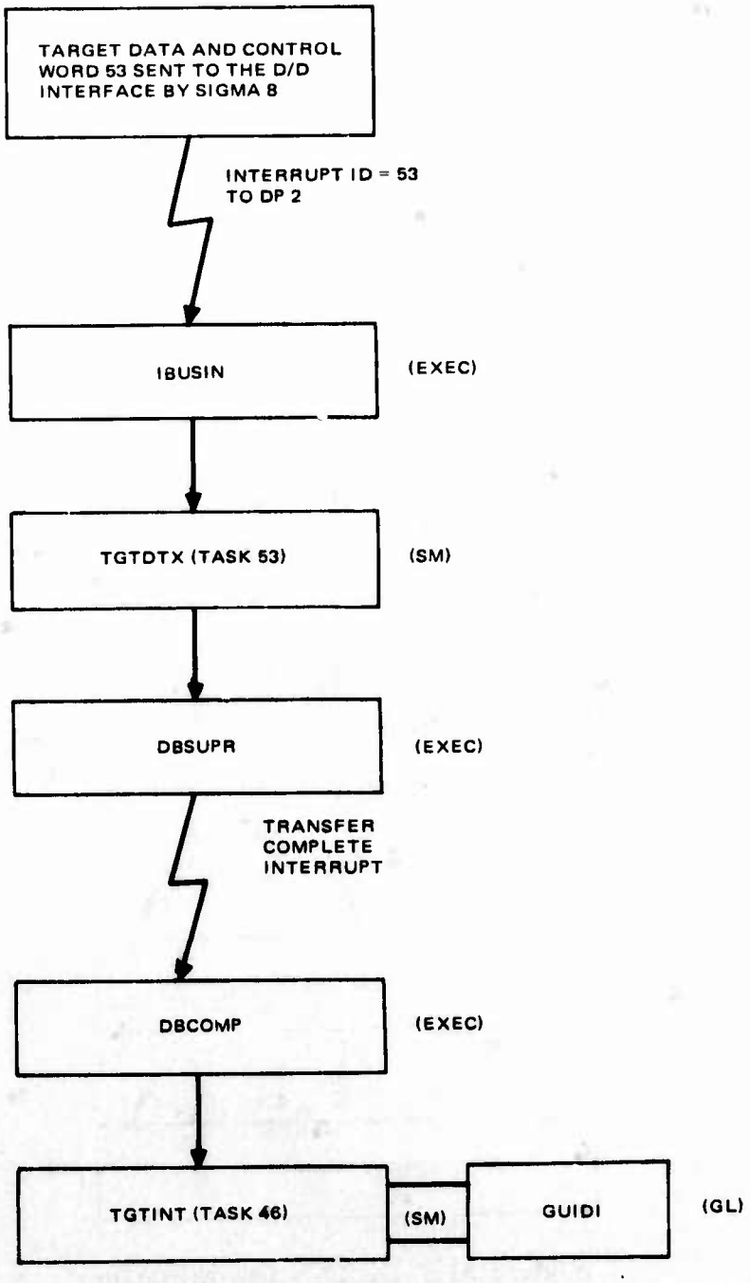


Figure 80. Target Information Interrupt Sequence

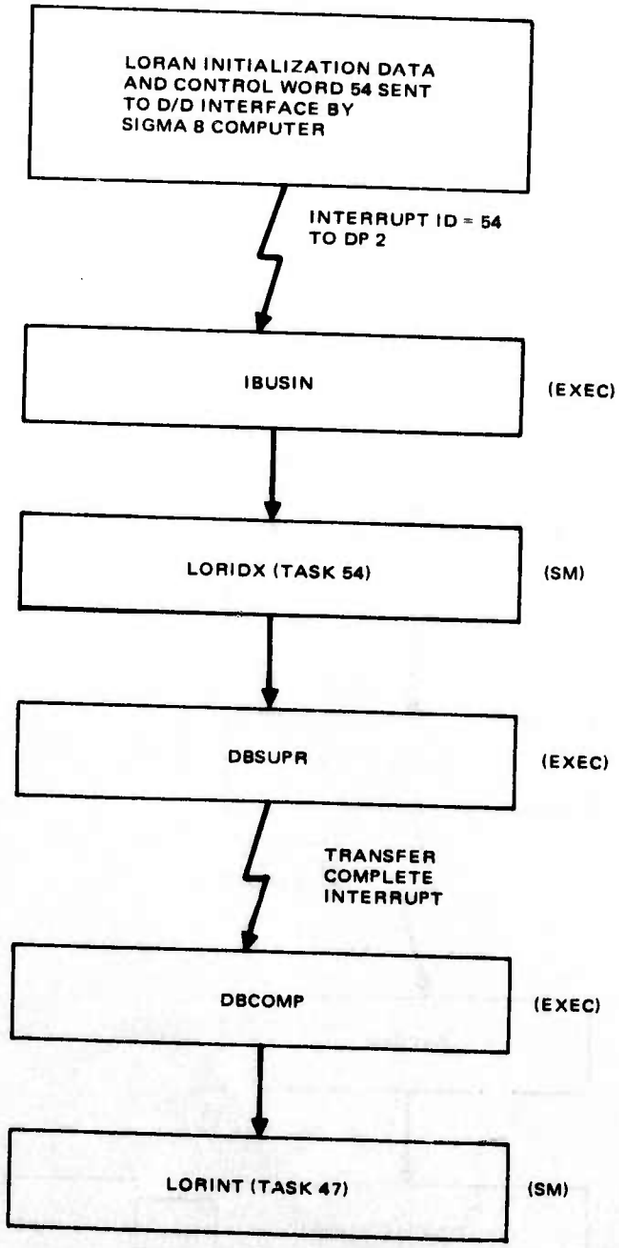


Figure 81. LORAN Initialization Interrupt Sequence

Weapon Separate Interrupt Sequence. After the D/D interface was integrated into the simulation, the SIGMA 8 computer communicated the beginning of a run by sending control word 55 to the D/D interface. This generated an interrupt in DP2 with ID = 55 (see Figure 82). The incoming interrupt caused the executive module, IBUSIN, to be called. IBUSIN placed Task 55, WEPSEP, in the task queue. WEPSEP called Systems Management subroutine INITCL which initialized the memory location required to start the DP2 clocked interrupts.

LORAN Receiver Interrupt Sequence. To simulate the LORAN receiver, the SIGMA 8 computer calculated two differences in horizontal distance from the weapon: (1) distance from weapon to LORAN master station minus distance from weapon to LORAN above station number 1, (2) distance from weapon to LORAN master station minus distance from weapon to LORAN slave station number 2. These two distance differences were sent to the D/D interface followed by control word 56. BIU No. 3 then sent an interrupt to DP2 with ID = 56 (see Figure 84). This called the executive module, IBUSIN, which placed Task 56, LORDTX, in the task queue. LORDTX calls the data bus supervisor, DBSUPR, with a pointer to the operand memory locations where the parameters describing the desired data transfer are stored. Based upon this information, DBSUPR starts the transfer of data across the weapon data bus. When the data has been transferred over the weapon data bus and placed in the

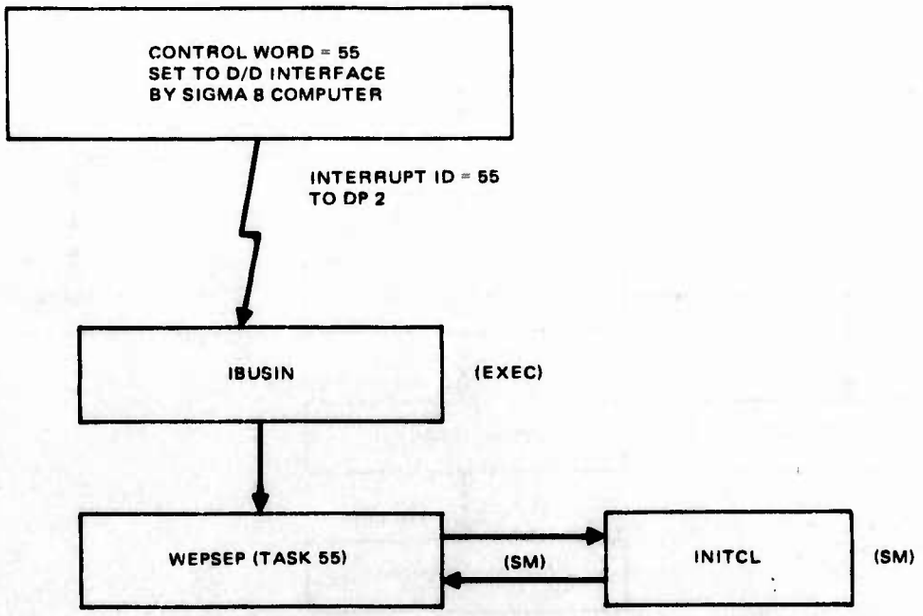


Figure 82. Weapon Separate Interrupt Sequence

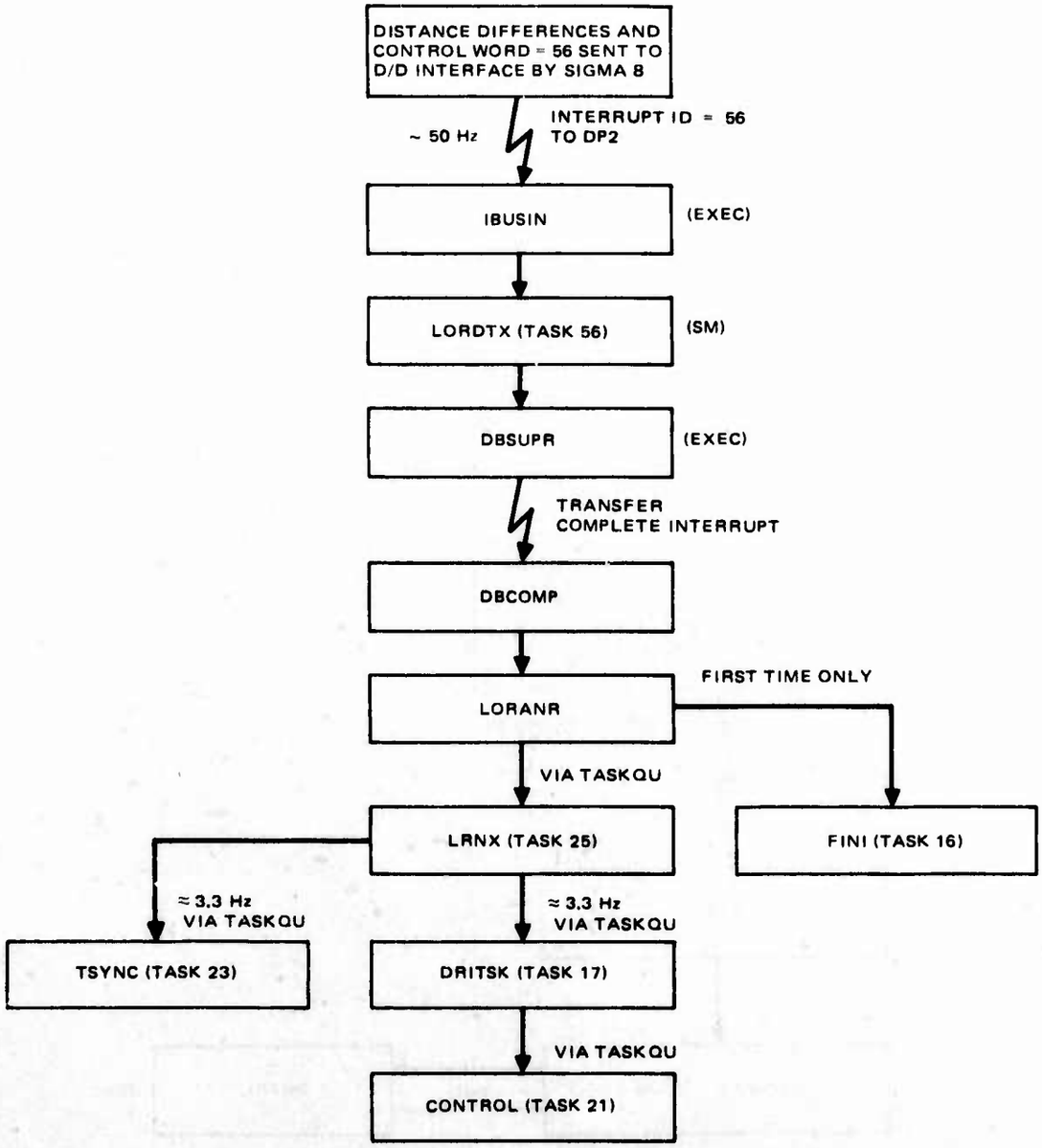


Figure 83. LORAN Receiver Interrupt Sequence

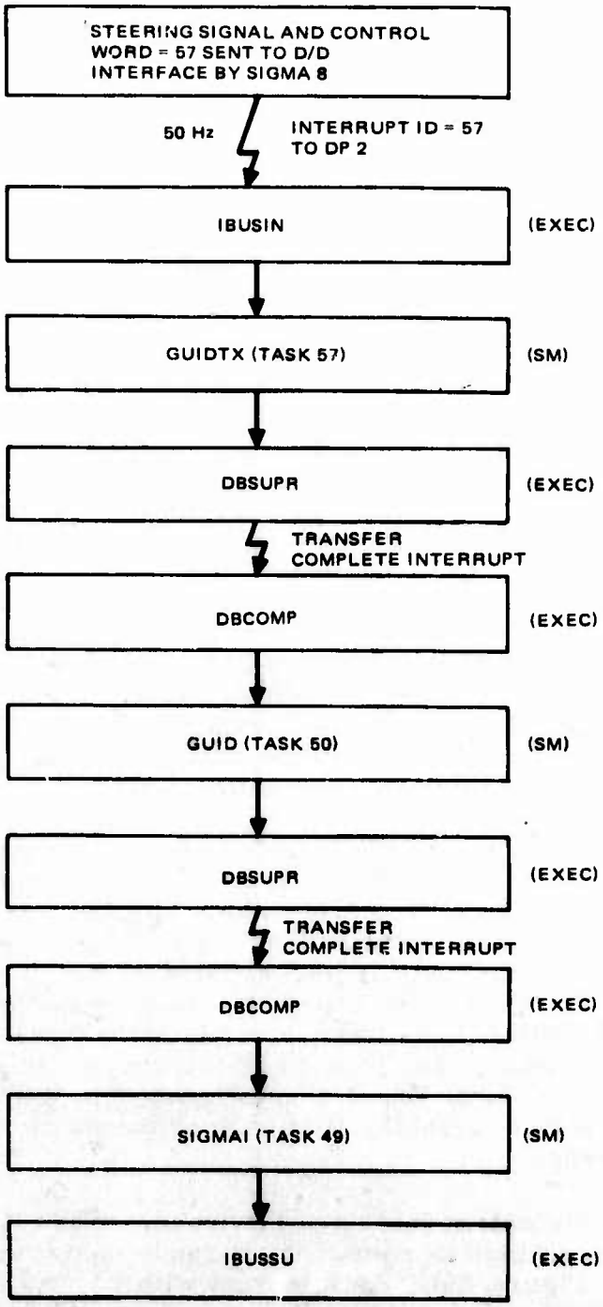


Figure 84. Midcourse Guidance Interrupt Sequence

input buffer a transfer complete interrupt is generated. The transfer complete interrupt vectors the program execution to executive module, DBCOMP. DBCOMP places Task 48, LORANR, in the queue. LORANR moves the distance differences from the input buffer to the proper operand memory locations and calls the executive module TASKQU to place Task 25, LRNX, in the task queue. Approximately every 0.03 second LRNX transforms the distance differences into a latitude and longitude position of the weapon for the Kalman filter and calls the executive module twice to place Task 23, TSYNC, and then Task 17, DRITSK, in the queue. TSYNC transfers the contemporaneous latitude and longitude calculated by the navigator to another location in operand memory and stores transition matrix before resetting the accumulators. DRITSK calculates the position alignment errors and calls the executive module TASKQU to place Task 21, CONTRL, which corrects the quaternion, latitude, and longitude of the navigation into the task queue.

Midcourse Guidance Interrupt Sequence. The SIGMA 8 computer calculates the midcourse steering signal and sends the value to the D/D interface followed by control word 57. An interrupt with ID = 57 (see Figure 84) is generated and sent to DP2. Executive module, IBUSIN, is called by the interrupt. IBUSIN places Task 57, GUIDTX, in the task queue. GUIDTX calls executive module DBSUPR with a parameter which points to the memory locations where the parameters describing the data transfer are stored. DBSUPR initiates the transfer of the data transfer over the weapon data bus. Upon completion of the data transfer to the input buffer, a transfer complete interrupt is generated. This calls the executive module DBCOMP which places Task 50, GUID, in the queue. GUID moves the steering signal into operand memory and writes time, weapon, latitude, longitude, altitude, velocity, North, West, and up, and other output variables into the output buffer. The data bus supervisor DBSUPR is called with a pointer to the memory locations where the parameters describing the transfer of these variables to BIU No. 3 is stored. DBSUPR initiates the transfer of these variables over the weapon data bus to the D/D interface from where they will be read by the SIGMA 8 computer just before the next LORAN receiver interrupt sequence. A transfer complete interrupt is generated when all of the data has been transferred. This interrupt calls the executive module DBCOMP which places Task 49, SIGMAI, in the queue. SIGMAI calls the executive module, IBUSSU with the parameter to send an interrupt with ID = 17 to BIU No. 3 which latches the 17 into the D/D interface. The 17 tells the SIGMA 8 computer the number of variables it is to read on the next LORAN receiver interrupt sequence.

Run Termination Interrupt Sequence. When the SIGMA 8 computer detects a run termination condition, it sends control word 58 to the D/D interface (see Figure 85). An interrupt with ID = 58 is received by the DP2. The incoming interrupt calls executive module IBUSIN which then places Task 58, STOPCK, in the task queue. STOPCK stops the clocked interrupts by zeroing the memory location which controls the hardware clocked interrupt (CLKCNT = 2181).

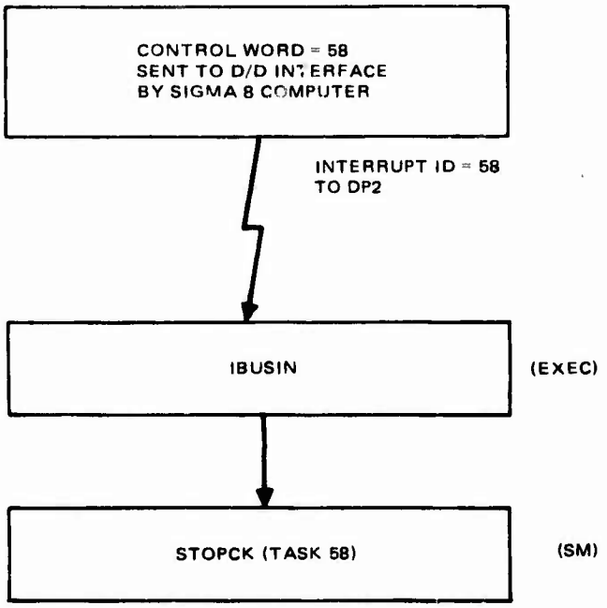


Figure 85. Run Termination Interrupt Sequence

Clock Interrupts. The following sequences are associated with clock interrupts which occur at 400-Hz and 10-Hz rates. The clock interrupts are enabled when weapon separation is detected. As a part of the 400-Hz sequence, periodic tasks at iteration rates which are subharmonics of 400 Hz, and which operate on common data with the 400-Hz routines, are queued by the 400-Hz routine, FCADX.

400-Hz Interrupt Sequence. Every 2.5 milliseconds (400 Hz) a clocked interrupt is generated in the DP2. This interrupt calls the executive module CLKINT which places Task 30, FCADI, in the queue for execution (see Figure 86). When FCADI is executed, the interrupt bus supervisor, IBUSSU, is called with the proper parameter to send an interrupt to BIU No. 2 to signal the signal conversion unit to begin converting the analog signals. Upon completion of the A/D conversion, an interrupt with ID = 10 is sent to the DP2. This calls executive module IBUSIN which places Task 10, ADPX, in the task queue. ADPX calls the data bus supervisor, DBSUPR, with a pointer to the operand memory locations where the parameters for transfer of the A/D data are stored. DBSUPR initiates the data transfer over the weapon data bus. Completion of the data transfer causes an interrupt to be generated which calls executive module DBCOMP which places Task 29, FCADX, in the task queue. FCADX moves the A/D variables along with the DLVW into operand memory and performs the 400-Hz stability loop calculations. The flipper commands are written to the output buffer, and the data bus supervisor, DBSUPR, is again called with a pointer to the location in operand memory where the parameter describing the transfer of the flipper commands to the D/A converters are stored. FCADX also places the outer loop calculations for the

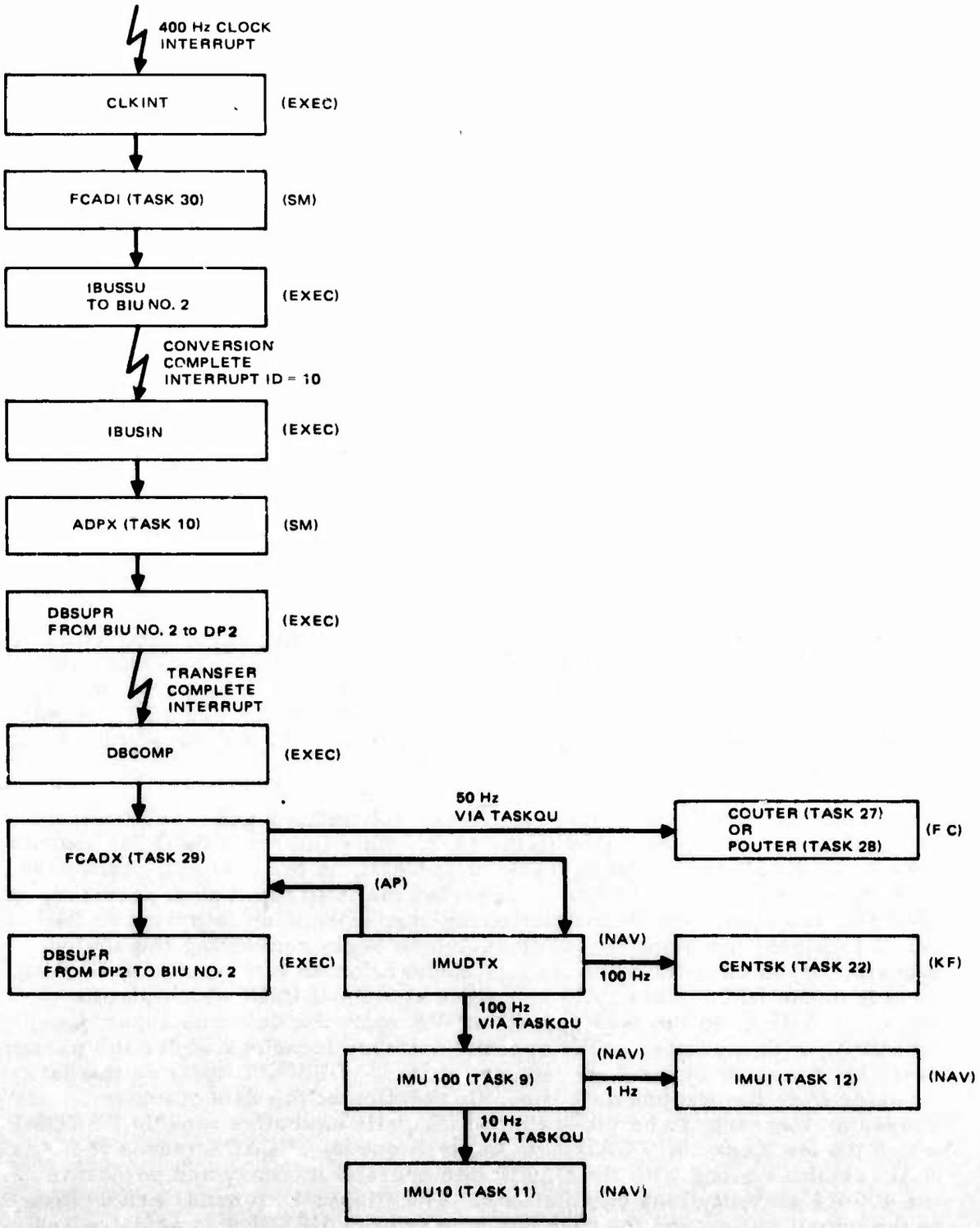


Figure 86. 400-Hz Interrupt Sequence

flight control in the task queue every eighth iteration by calling the executive module TASQU with the task number. After calling DBSUPR, FCADX call subroutine IMUDTX which accumulates the accelerometer and rate gyro data and calls the executive module TASKQU every fourth iteration to place Task 9, IMUIOO, and Task 22, CENTSK, in the task queue. IMUIOO performs the 100-Hz calculation for the strapdown inertial calculations and places the 10-Hz and 1-Hz strapdown inertial calculations in the task queue at the correct interval. CENTSK performs the 100 Hz Kalman filter accumulations.

10-Hz Interrupt Sequence. The 10-Hz clock interrupt which is synchronized with the 400-Hz clock interrupt calls CLKINT which places Task 60, GDLW, into the task queue to be executed (see Figure 87). GDLW performs the midcourse guidance law calculations.

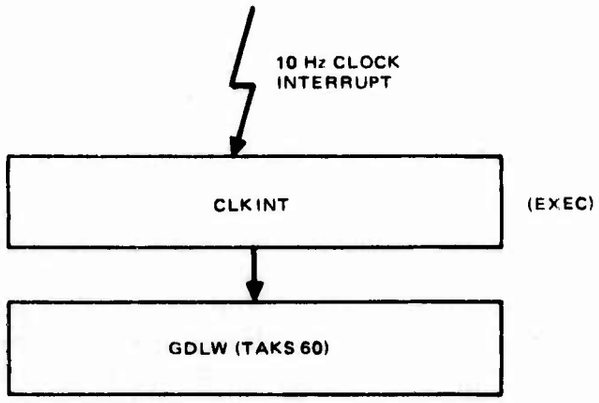


Figure 87. 10-Hz Interrupt Sequence

SYSTEM INTEGRATION AND RUNNING SIMULATION

In this subsection, the integration of the system components is described, and some results of running the simulation are given.

System Integration

The integration procedure was to first get the hybrid simulation of the GBU-15 weapon working with the analog version of the cruciform wing autopilot, and then to add the other components, one by one, until the entire system was operating. The order of integration of the other components was:

1. DP-2, weapon bus, analog interface, and the executive software.
2. The flight control software.

3. The digital interface between the weapon bus and the SIGMA 8 computer and the associated system management software.

4. The strapdown inertial navigation software.

5. The LORAN update subsystem. This included the simulation of the LORAN receiver in the SIGMA 8, the LORAN coordinate transformation software in the DP, and the update data filter software in the DP.

6. The midcourse guidance law software.

A brief description of the major integration events is given below. In the course of the integration, some problems occurred which have a bearing on future development efforts for a weapon system which uses the core electronics. These problems are discussed below.

Digital Autopilot Integration. The digital autopilot had already been validated in the earlier PDAP work; however, there were two changes in going from the PDAP to the DP2 situation. First, the flight control software had been rewritten, as previously described, to fit into the DP modularity concept. Second, the weapon bus and the executive software had been added to the system. This last change increases the transport lag on the data passing through the DP system. The results of the integration essentially duplicates the results obtained earlier in the PDAP simulation; thus, the two changes mentioned above have no deleterious effect on autopilot performance. Since the DP simulation adds no new knowledge, relative to autopilot performance, from what was already learned in the PDAP simulation, the detailed results are not presented here.

The throughput requirements for the DP system, derived in Volume I of the final report, are based on the peak loads occasioned by the allowed transport lag for autopilot data. Thus, the results obtained in the simulation verify the adequacy of the requirements.

While the DP2 flight control software contains autopilot implementations for both the planar wing weapon and the cruciform wing weapon, only the latter was used in the simulation.

Strapdown Inertial Navigation. The software for the inertial navigation functions was derived from the DP1 software by eliminating some unnecessary parts as described in the DP2 Software subsection. No particular problems were encountered in the integration. It is to be noted, however, that the performance of the software cannot be checked very accurately on this type of simulation. This is because the inertial instruments are simulated on the analog computer. The drifts in the analog computer and the analog input/output devices are large enough to swamp any inaccuracies in the navigation software. In this simulation, the drifts were large enough that errors of as much as 20 kilometers would be accumulated during a flight of 75 kilometers. Thus, it is necessary to have an update system in order to get adequate performance in the simulation.

LORAN Update Subsystem. The LORAN receiver was simulated in the SIGMA 8 computer. The LORAN coordinates, in terms of distance differences (time differences and speed of light), were passed to the DP over the digital interface. These time difference coordinates were transformed in the DP to the corresponding weapon position data which were then passed through a Kalman filter. This filter was a slightly modified version of the alignment filter used in the DP1 system. An interface problem became apparent when it was attempted to integrate these components. The problem was due to an incompatibility between the accuracy of the LORAN position algorithm and the input scaling for the alignment filter.

The LORAN position, calculated in the SIGMA 8, was determined by a linearization of the basic LORAN equations. The target position was chosen as the point about which the linearization was made. With this choice of expansion point, the error in the algorithm goes to zero when the target is approached, but the error can be substantial at long ranges. The effect is to give a rather large initial offset from the true position, with the offset error going to a small value quite rapidly as the flight progresses.

The alignment filter was designed to handle a large initial offset; however after the initial error is corrected, the filter changes scaling so that the maximum input it can accommodate is consistent with the expected correction signals it will obtain in the alignment process aboard an aircraft. This scaling did not have enough dynamic range to accommodate the rather large error changes accumulated between updates by the inaccuracies in the LORAN position algorithm. This measurement error overflow caused the filter to be ineffective in correcting the navigation software parameters.

The problem could have been eliminated either by rescaling the filter or by designing a more accurate LORAN position algorithm. However, since neither of these efforts fit into the schedule, another solution was found. The LORAN position algorithm was bypassed, and true position was sent to the DP where it was filtered by the Kalman filter.

It should be pointed out that the inaccuracy in the LORAN position algorithm might not be a problem in many system applications, since the error goes to zero when the target is approached. Where a problem might occur is when it is desired to shape the trajectory to closely follow some desired path or to pass over some intermediate point. For this application, one or more additional expansion points would be required, or a more accurate algorithm would have to be found.

It should also be noted that, if it is desired to use the same Kalman filter for the alignment and any of several update subsystems, the characteristics of the update system must be known in advance so that the filter scaling can accommodate their error idiosyncrasies.

As noted previously, the drift in the simulated inertial instruments was rather large. In order to hold the effect of these drifts within the filter dynamic range, the iteration rate of the update was increased to 3.33 Hz. The design point for the filter was one Hertz, but it operated well at the increased rate.

Guidance Law. The form of the guidance law used in the simulation was described in the DP2 Software subsection. It was first implemented on the SIGMA 8 in order to check its performance before coding it for DP2. In operating with the guidance law in the SIGMA 8, the weapon position, computed by the inertial navigation software in DP2, was transmitted to the SIGMA 8 over the digital interface. The steering signals were then computed with the guidance law software in the SIGMA 8 and sent back to the DP2 over the digital interface.

As it turned out, all of the performance data was taken with the SIGMA 8 version of the guidance law. When it was attempted to integrate the version of the law coded for DP2, it was found that the system became unstable. This was due to several logic and scaling faults in the code. By the time these problems were diagnosed and corrected, there was not time to repeat the data runs.

Other System/Software Problems. Two other problems occurred during the integration: one of them was in the hybrid simulator, and the other was in the DP2 software.

In the GBU-15 simulation, the E-O terminal guidance function is implemented so that, if the seeker look angle exceeds 30 degrees, the amplifiers will saturate, and the run must be terminated. The look angle is computed continuously, even during midcourse, although E-O guidance is initiated only when the terminal guidance command is given. Thus, if the look angle ever exceeds 30 degrees during the flight, the run is terminated. This characteristic put some limitations on the kind of trajectories that could be used in the midcourse guidance, but the limitation was not severe.

A software problem was discovered during the integration of the digital interface circuitry. It was observed that occasionally an interrupt signal sent by the SIGMA 8 was not responded to by the DP. The problem was traced to the action of the Idle Loop in the system management software. As previously described, the Idie Loop sequentially assesses all operand memory locations whenever nothing else needs to be done by the DP. The purpose of this action is to allow monitoring of any memory location. For most operand memory locations there is no problem with this procedure. In the case of the interrupt signal, however, a problem can occur. When an interrupt signal comes in from the weapon bus, the bus hardware stores the signal in a latch, which is addressed as operand memory, and then initiates a hardware interrupt in the DP to service the incoming signal. Subsequent reading of the latch by the interrupt routine zeros it. The problem was due to the fact that occasionally the Idle Loop would access that location between the time that the interrupt word was stored there and before the interrupt routine could be initiated. It was solved by limiting the range of operand memory addresses which the Idle Loop can access. The problem is mentioned here as an example of some of the rather subtle things that can happen in the interaction of the software and hardware.

Running Simulation

The following procedural sequence was used for the simulation runs.

1. The hybrid simulation was tested by running it with the analog autopilot.

2. The system constants were loaded into the DP2 operand memory.

3. The program was loaded into the DP2 program memory.

4. The DP2 start button was pushed which put the DP2 in the Idle Loop.

5. The hybrid computer was started.

The first step in the hybrid computer operation was for the SIGMA 8 to send initialization data to the DP2 across the weapon bus. When this was completed, the weapon separate signal was sent which initiates the flight sequence.

For recording data from the DP, the desired quantities were first stored in operand memory and then read out to the SIGMA 8 over the weapon bus. The SIGMA 8 transmitted these quantities to the analog computer where they were converted to analog signals and then recorded on a strip chart.

Simulation Run Test Conditions. The run conditions were chosen primarily to evaluate the performance of the inertial navigation with update and the guidance law, since the autopilot operation had been previously evaluated in the PDAP simulations. All runs were made with the following launch condition:

- Altitude = 12.2Km (40,000 feet)
- Range from target = 73.2Km (40 nautical miles)
- Mach number = 1.5

All trajectories were flown in a generally North direction with the target located at 52° North latitude and 12° East longitude.

To test the performance of the inertial navigation and the guidance law various values of approach angles to intermediate check points and the target were chosen. These are listed in Table 16. In all of the runs the guidance was switched to E-O terminal at about four kilometers from the target.

Performance of Midcourse Guidance

Representative trajectories for the conditions given in Table 16 are shown in Figures 88 through 92. Steering in the vertical plane was accomplished by the modes already in the GBU-15 cruciform wing flight control system. Steering in the horizontal plane was done by the midcourse guidance functions until the range to the target decreased to about 4 kilometers at which time the steering was switched to E-O terminal. The results for the five cases are discussed below.

Case 1. This case is illustrated in Figure 88. The vertical trajectory is normal for the cruciform wing weapon and was essentially the same for all runs. It will be noted that the trajectory in the horizontal plane drifted off from the direct course during the flight and then returned to the desired path as the target was approached. This deviation is due to the fact that the guidance law gain is not very high at large distances from the target. As the range to the target decreases, the gain increases and pulls the trajectory back to the desired flight path.

At about 4 kilometers from the target, E-O terminal is switched in and the altitude decreases rapidly. The miss distances for all cases were within acceptable CWW GBU-15 boundaries.

Case 2. In this case, illustrated in Figure 89, the desired approach angle to the target is 10 degrees off from the North-South line. The trajectory veers to the West in order to get on the desired approach course and then comes into the target at the correct angle.

The glitch in the vertical trajectory is due to the simulation mechanization and occurs when transition to E-O terminal guidance is made. During midcourse, before transition to terminal, the velocity increments (Δv's) are accumulated in body coordinates, transformed through the body Euler angles, and then integrated to obtain the range vector in inertial coordinates. It is this range vector which appears on the data record during midcourse. When E-O terminal guidance is initiated, the simulation configuration is changed so that the Δv's are transformed through the gimbal angles and then integrated to obtain the range vector in seeker coordinates. This change is made to decrease drifts in the seeker loop during terminal guidance. During this portion of the flight, the range vector for the data record is obtained by transforming from seeker coordinates to inertial coordinates. To obtain an initial condition for the integration in seeker coordinates, the midcourse version of the range vector is transformed from inertial coordinates to seeker coordinates. Thus, immediately prior to transition, the value of the range vector which appears in the data record is obtained directly from the midcourse integration. Immediately afterwards it is the same value, but it has gone through several transformations: first from inertial coordinates to seeker coordinates and then back again to inertial coordinates. In principle, a discontinuity is not expected; however, because of small drifts in the analog system, the transformation matrices are not perfectly orthogonal. Therefore, there is sometimes a noticeable glitch in the data record at transition. The glitch has no significance with respect to performance of the guidance law.

Case 3. This case, illustrated in Figure 90, has one intermediate check point which the weapon is to fly over. The check point is placed on the North-South line at about 41 kilometers from the target. It is interesting to compare this case with Case 1 which had the same desired path but no intermediate check point. In Case 3 the deviation from the desired path is less. This is because the course is initially directed to the check point and hence the guidance law gain is higher during the first part of the flight.

TABLE 16. TEST CONDITIONS FOR SIMULATION RUNS

CASE	NUMBER OF INTERMEDIATE CHECK POINTS	CHECK POINT PARAMETERS									APPROACH ANGLE TO TARGET
		CHECK POINT NUMBER									
		1			2			3			
		R	D	A	R	D	A	R	D	A	
1	0	•	•	•	•	•	•	•	•	•	0
2	0	•	•	•	•	•	•	•	•	•	10
3	1	42	0	0	•	•	•	•	•	•	0
4	1	37	2	−4.76	•	•	•	•	•	•	4.76
5	3	56	2	−9.46	38	−2	9.46	19	2	−9.46	9.46

R = NORTH RANGE FROM TARGET, KILOMETERS

D = EAST/WEST OFFSET DISTANCE FROM NORTH/SOUTH LINE

A = APPROACH ANGLE, DEGREES FROM NORTH

• MEANS DOES NOT APPLY

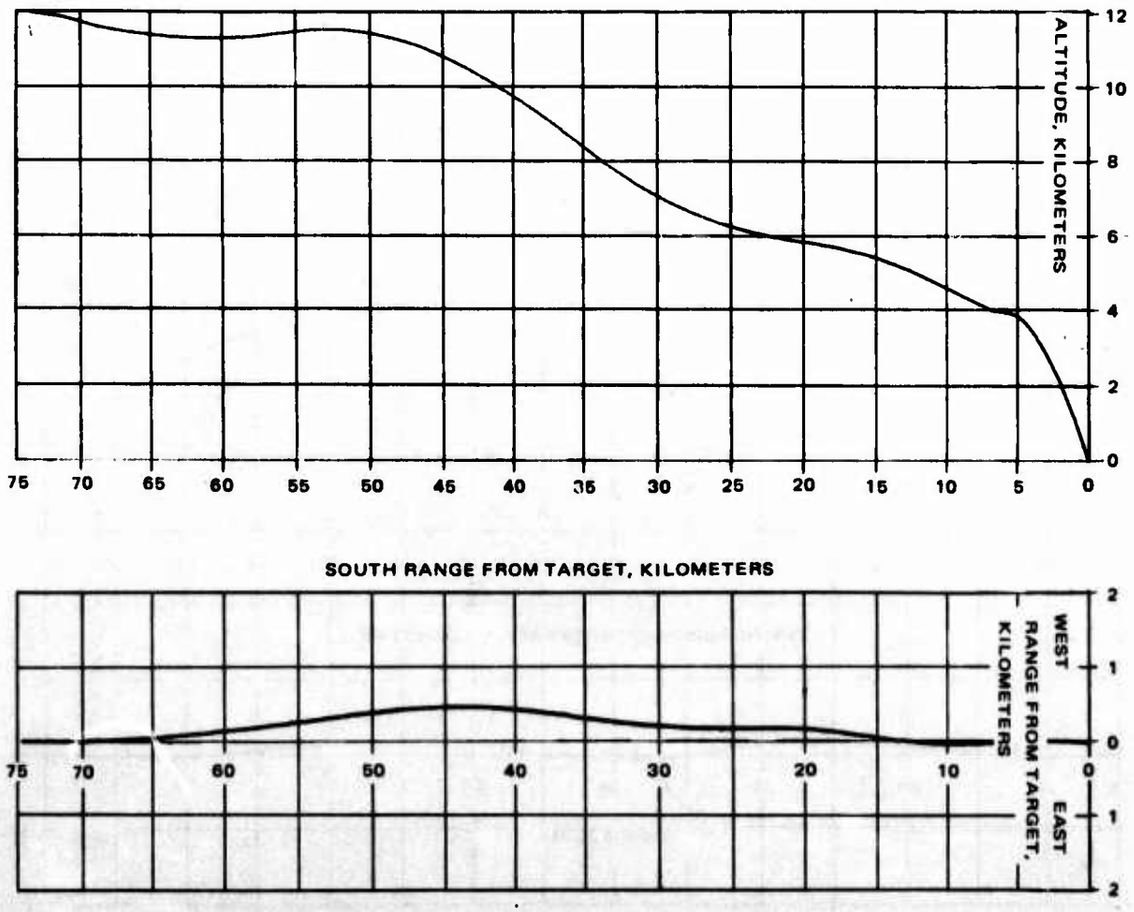


Figure 88. Case 1 Trajectories

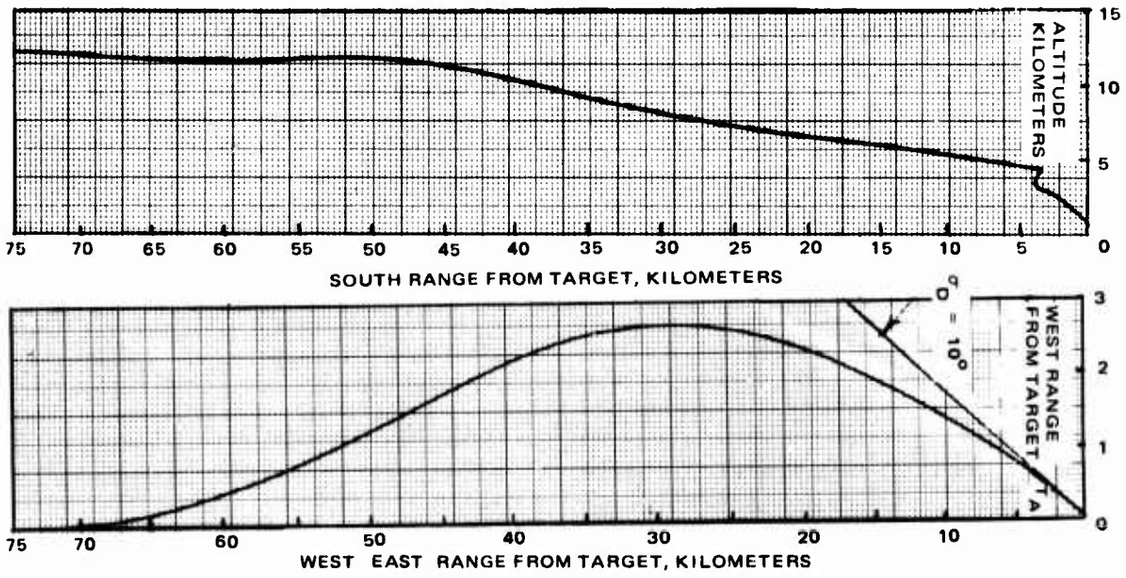


Figure 89. Case 2 Trajectories

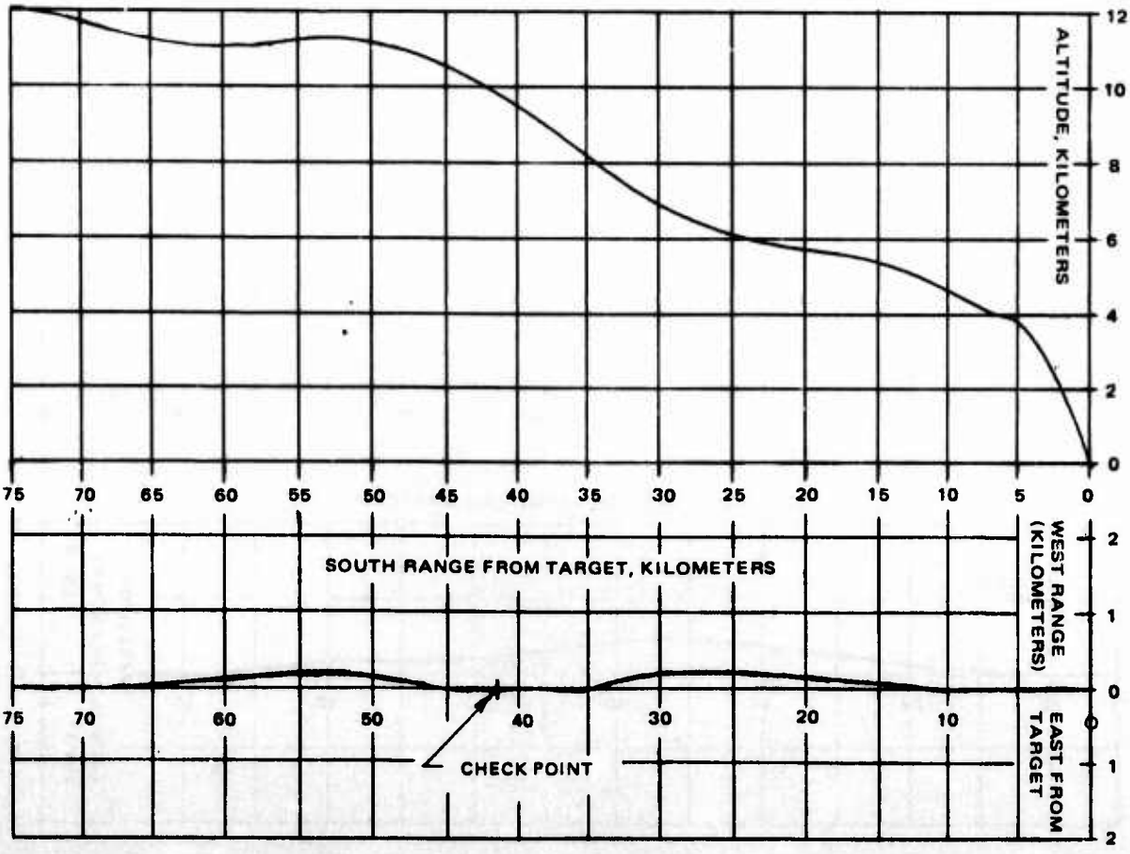


Figure 90. Case 3 Trajectories

Case 4. This case, illustrated in Figure 91, has an intermediate check point offset 2 kilometers to the West from the North-South line. The desired angle of approach to the check point is -4.76 degrees and the desired angle of approach to the target is +4.76 degrees. The trajectory follows the desired course very well.

Case 5. As illustrated in Figure 92, this case has three intermediate check points. It will be noted that the weapon has some difficulty getting onto the desired approach paths for the second and third check points. At the third check point the trajectory overshoots and does not pass over the point. This case overstresses the system; there is not enough gain to successfully accomplish the required turns. If these kind of maneuvers are required, the guidance law gain would have to be increased.

Conclusions

The results of the simulation go far toward validating the concept of the Core electronics. Specifically, the following items were demonstrated:

1. The Core electronics operating in a simulated functional environment (including interfaces) which would be typical of a real application.

2. Satisfactory performance of an integrated guidance system consisting of:

   (a) Midcourse inertial guidance with periodic position updates (in Core electronics).

   (b) E-O terminal guidance (sensor signal processing not in Core electronics).

It can be concluded that the simulation verified the adequacy of the requirements for the digital processing system which were derived in Volume I of the final report.

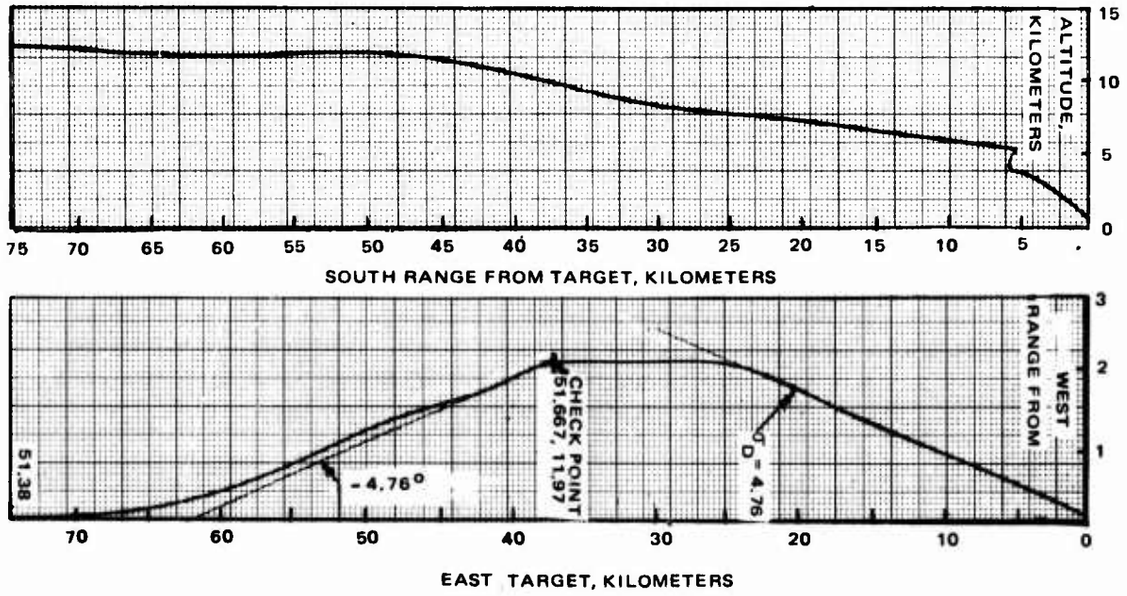


Figure 91. Case 4 Trajectories

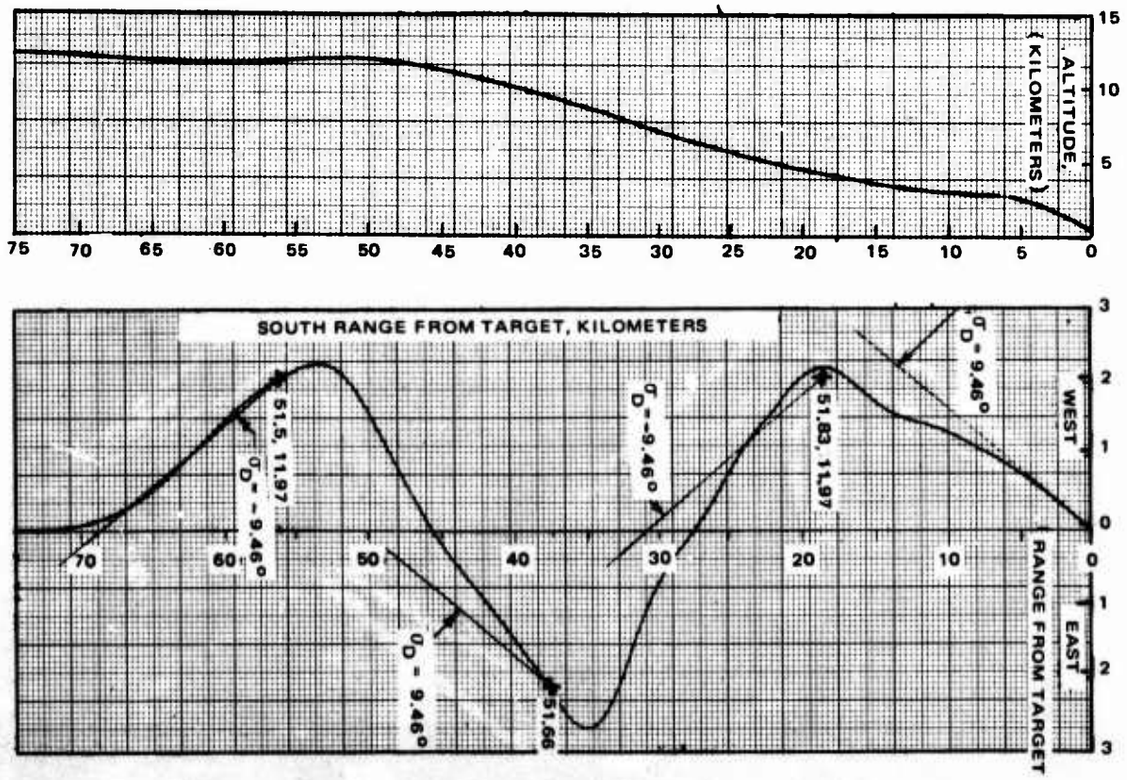


Figure 92. Case 5 Trajectories

INITIAL DISTRIBUTION

Hq USAF/RDPA	1	CG USAMICOM/AMCPM-CT-E	1
Hq USAF/RDQRME	1	DDC	2
Hq USAF/XOOFA	1	AFATL/DLJA	1
AFSC/INA	1	AFATL/DLJF	1
AFSC/SDA	1	USAF Academy	1
AFSC/DLCAW	1	ASD/SD-65	1
ASD/YFEI	1	AUL (AUL-LSE-70-239)	1
ASD/YFEA	1	ASD/ENFEA	1
ASD/YPEX	1	Hq USAF/SAMI	1
ASD/SD	1	Ogden ALC/MMWM	2
AFFDL/FEI	1	AFIS/INTA	1
AFAL/RW	2	Hq TAC/DRA	1
ASD/ENA	3	Hq USAFE/DOQ	1
TAC/DRAI	2	Hq PACAF/DOO	1
TAC/XPSY	1	Redstone Ars/DRDMI-TGG	1
AFAL/AAI	1		
ASD/YHEV	1		
ASD/XRG	2		
NAVAIR SYS COMD/AIR 360E	1		
NWC/Code 143	4		
NWC/Code 533	1		
AFFDL/FGL	1		
ATC (XPQS)	1		
NAVAIR SYS COMD/AIR-5323	1		
NAVAIR SYS COMD/AIR-5324	1		
OSD, ODDR&E/TST&E	1		
DARPA/TIO	1		
ADTC/PP	1		
ADTC/ADE	1		
TAWC/DT	1		
TAWC/TEFA	1		
TAWC/FTS	1		
ADTC/XR	2		
TRADDC/ADTC/DO	1		
AFATL/DL	1		
AFATL/DLB	1		
AFATL/DLA	1		
AFATL/DLMA	1		
ADTC/SD15	1		
AFATL/DLMT	1		
AFATL/DLMM	15		
AFATL/DLY	2		
ADTC/SD-7	1		
TAWC/TRADOCLO	1		
AFATL/DLOSL	9		
Redstone Sci Info Center	2		
Off Naval Research/Code 211	1		

(The reverse of this page is blank)