UNCLASSIFIED

| AD NUMBER |
| --- |
| **AD915422** |
| LIMITATION CHANGES |
| TO:<br>**Approved for public release; distribution is unlimited.** |
| FROM:<br>**Distribution authorized to U.S. Gov't. agencies only; Test and Evaluation; 21 AUG 1972. Other requests shall be referred to Army Advanced Ballistic Missile Defense Agency, Huntsville, AL 35807.** |
| AUTHORITY |
| **USASSC ltr 1 May 1974** |

THIS PAGE IS UNCLASSIFIED

AD 915422

RCA

DATA PROCESSING/HAPDAR FIELD TEST FACILITY

# Final Technical Report

For Period 1 July 1973 to 30 September 1973

Contract No.: DAHC60-73-C-0035
CDRL Sequence No.: B002
September 1973

DDC
RECEIVED
DEC 19 1973
B

Prepared for:

US Army SAFEGUARD Systems Command
Contracts Office, SSC-C
P.O. Box 1500
Huntsville, Alabama 35807

Sponsored by:

The U.S. Army Advanced Ballistic Missile Defense Agency

Contractor:

RCA | Government and Commercial Systems
Missile and Surface Radar Division | Moorestown, N.J.

September 1973

# ABMDA

# DATA PROCESSING

# FIELD TEST FACILITY

CONTRACT NO.: DAHC60-73-C-0035

CDRL SEQUENCE NO.: B002

FINAL TECHNIAL REPORT FOR

PERIOD 1 JULY 1973 TO 30 SEPTEMBER 1973

The findings of this report are not to be construed as an official Department of the Army position.

Prepared for:

U.S. Army SAFEGUARD Systems Command
Contracts Office, SSC-C
P.O. Box 1500
Huntsville, Alabama 35807

Sponsored by:

The U.S. Army Advanced Ballistic Missile Defense Agency

Submitted by:

RCA|Government and Commercial Systems
Missile and Surface Radar Division| Moorestown, N.J.

Distribution limited to U.S. Government Agencies Only; Test and Evaluation; 21 Aug. 72. Request by other agencies should be addressed to the Director, U.S. Army Advanced Ballistic Missile Defense Agency, Huntsville Office, ATTN: RDMH-P, P.O. Box 1500, Huntsville, Alabama 35807

## FOREWORD

The Data Processing/HAPDAR Field Test Facility was created by adding an IBM 360/65 Computer and specially developed software to the existing Sperry Rand HAPDAR (Hardpoint Demonstration Array Radar) at White Sands Missile Range, New Mexico. The integrated combination of computer, software, and phased array radar, resulted in a system capable of controlled search and track operations against missile and aircraft targets at WSMR. The initial creation of the facility occurred under Contract DAHC60-72-C-0032 which ended 18 December 1972. A description of the Facility appears in the Final Technical Report prepared by RCA, and issued under that contract as CDRL Sequence No. A006.

Debugging and initial operation of the Data Processing/HAPDAR Field Test Facility occurred under the present contract (DAHC60-73-C-0035). This contract was originally written to cover the period 18 December 1972 to 30 June 1973, and a final report was issued as Contract Data Item A002. In the meantime, however, the contract was extended to 30 September, 1973, mainly to permit completion of a Passive Jammer Locator Field Test prior to deactivation of the entire Facility. The present report describes RCA activity at the Facility for the three-month period 30 June to 30 September 1973.

## TABLE OF CONTENTS


| Section | | Page |
|---|---|---|
| 1. | Introduction | 1-1 |
| 2. | Technical Effort During Report Period | 2-1 |
| | 2.1 Support of PJL Field Experiments | 2-1 |
| | 2.2 Data Processing Support to SAFSEA and ADA | 2-2 |
| | 2.3 AFAR Turnover | 2-2 |
| | 2.4 Summary of Computer Usage During Report Period | 2-3 |
| 3. | Conclusions and Recommendations | 3-1 |
| | 3.1 Technical Conclusions | 3-1 |
| | 3.2 Recommendations | 3-2 |
| 4. | Summary of Data Items Delivered | 4-1 |
| 5. | References | 5-1 |
| 6. | Glossary of Terms | 6-1 |
| Appendix 1 | Final HAPDAR Program Versions | A1-1 |
| Appendix 2 | Timing Macros | A2-1 |
| Appendix 3 | RCA Support for PJL Experiment | A3-1 |
| Appendix 4 | Distribution List for This Report | A4-1 |
| DD Form 1473 | | |

## Section 1

## INTRODUCTION

Contract DAHC60-73-C-0035 was originally scheduled to end on 30 June 1973, and a Final Report was issued as Contract Data Item A002. In the meantime however, the contract was extended for three additional months, and the present Final Report is being issued to cover the three-month extension. Issued as Contract Data Item B002, the present report describes activity at the Data Processing/HAPDAR Field Test Facility during the period 1 July 1973 to 30 September 1973. The activity involved the HAPDAR and the IBM 360/65 computer, both owned by ABMDA and operated at White Sands Missile Range, New Mexico. The computer, associated hardware, and related software were originally installed and developed under a previous contract, DAHC60-73-C-0032. Several radar and tactical data processing experiments were performed using the DP/HAPDAR facility.

The activities performed during July through September, 1973, were as follows:

(1) Support the Passive Jammer Locator Field Test efforts by Syracuse University Research Corp. under their Contract F30602-72-0075.

(2) Provide data-processing services to support using agencies, U.S. Army Safeguard System Evaluation Agency (SAFSEA) and U.S. Army Combat Development Command Air Defense Agency (USACDCADA).

(3) Provide support to insure an efficient turnover of equipment and software to the ABMDA Field Array Radar (AFAR) project.

This report reviews the above-listed activities. A summary of the DP/HAPDAR program is presented, with technical conclusions and recommendations. A list of primary items of documentation delivered is provided.

## SECTION 2

## TECHNICAL EFFORT DURING THE REPORT PERIOD

### 2.1 SUPPORT OF PASSIVE JAMMER LOCATOR (PJL) EXPERIMENTS BY SYRACUSE UNIVERSITY RESEARCH CORPORATION (SURC)

A principal use of the DP/HAPDAR facility during the report period was to support the SURC PJL tests. The PJL tests involved positioning the HAPDAR and RONDO receive beams at particular locations in real time. Additionally, radar returns with special PJL returns were recorded on the 360/65 tape drives.

To accomplish the above functions, the experiment team of RCA, SURC, and Sperry utilized the DP software-package real-time operating system and its associated input/output functions. The normal HAPDAR Tactical software package was replaced with the SURC PJL experiment software. The PJL software communicated with the Real-Time Execuitve (BMDOS) using the same commands as originally designed in the Execuitve. Data recording in real time was accomplished by using the HAPDAR real-time tape output (WRITDL) program to record the desired parameters.

During the report period, data reduction and program development was accomplished by SURC personnel utilizing the IBM 360/65 machine.

During this period RCA supported SURC by supplying operations personnel as required and by providing assistance in the use of the BMDOS and related software. One major activity in this area was the conversion of data tapes to be compatible with the SURC Sigma-5 computer during the terminal phases of the contract.

All SURC software has been preserved, utilizing the Archive program, so that subsequent data-reduction may be performed on the 360/65 at RCA Moorestown.

Details of the support requirements are found in the SURC document cited in Reference 3. The RCA support effort is presented in Appendix 3.

## 2.2 DATA PROCESSING SUPPORT TO SAFSEA AND USACDCADA

Operations and technical support was provided to the using agencies, SAFSEA and USACDCADA, during the report period. No unusual problems or events occurred during the period of this report. A summary of computer usage is presented in Subsection 2.4. The operation of the computer was completed on 30 September 1973. Teardown of the system for shipment started on 1 October for shipment to the AFAR project.

## 2.3 AFAR TURNOVER

In connection with the AFAR Program, ABMDA had decided to utilize the IBM 360/65 Computer at the HAPDAR Site as the Radar Control Computer for the AFAR development. This decision precipitated activity at the site, to prepare for the movement of hardware and software from the site to the RCA plant in Moorestown, N.J.

Specific requirements were that a final version of the HAPDAR software be defined and prepared for transfer to AFAR software functions. Additionally, it was felt that timing information of HAPDAR software functions was needed on the selected versions to aid in AFAR software design decisions. To accomplish this, two tasks were initiated:

(a) Develop an 'Archive' program to aid in establishing the HAPDAR final program versions and AFAR baseline versions. This program was used to provide an AFAR Baseline.

(b) Develop timing macros that would measure execution times of selected HAPDAR program modules and execute these with HAPDAR real-time software.

(c) Develop a move plan for the hardware and software.

The Archive program was developed and used to produce final program versions for HAPDAR.

The HAPDAR final versions are shown in Appendix 1 with Archive output. The Archive system will be used on the AFAR project as a major configuration management tool.

A timing study of software functions on HAPDAR was conducted during the HAPDAR/AFAR Turnover to aid in AFAR software design decisions. A previous HAPDAR Timing Model (Reference 1) indicated the need for additional data to substantiate the model assumptions. Since the hardware monitoring equipment (SUM) was not available to perform this function, it was decided to generate software macros that were the equivalent of SUM equipment. The design intent of the macros was to minimize the execution overhead. These macros have been designed, tested and are operational. They are described in Appendix 3.

A move plan was developed and executed during this period to insure that a successful transition of hardware and software to AFAR. All elements of this plan were completed by September 30, 1973 as scheduled.

## 2.4 SUMMARY OF COMPUTER USAGE DURING REPORT PERIOD

Table 2-1 presents a summary of jobs and hours of utilization by various users of the DP/HAPDAR Facility. This accounting does not include time that the system was operated in real-time with the radar; the table shows only those jobs and hours in a standard data processing mode under OS-360.

TABLE 2-1. SUMMARY OF 360/65 COMPUTER USAGE

| | July | | August | | September | |
|---|---|---|---|---|---|---|
| User | Jobs | Hours | Jobs | Hours | Jobs | Hours |
| ABMDA | 1226 | 106.3 | 998 | 68.1 | 878 | 93.0 |
| USACDCADA | 213 | 5.8 | 294 | 16.4 | 320 | 24.1 |
| USASAFSEA | 352 | 31.7 | 386 | 43.0 | 227 | 24.3 |
| SURC | 513 | 61.1 | 329 | 34.9 | 494 | 60.3 |
| *USAMISO | -- | | -- | | 201 | 30.9 |
| TOTALS | 2316 | 204.9 | 2010 | 161.5 | 2121 | 232.4 |

Grand Total - 6447 Jobs - 598.8 hours

* USAMISO did not run in July and August.

## SECTION 3

## CONCLUSIONS AND RECOMMENDATIONS

This section briefly discusses the DP/HAPDAR contract effort and presents overall conclusions and recommendations which have arisen based on experience with the program list of all reports delivered by RCA is included.

### 3.1 TECHNICAL CONCLUSIONS

(1) Experience shows the practicality and economy of using a software simulation of an actual radar (in this case, HAPDAR) in order to test real-time control and tactical software. Simulation of the actual HAPDAR was accomplished using an External Environment Simulator (core-resident in the IBM 360/65) and also using the SACS-H Simulation (located in Huntsville, Ala., and connected to the DP/HAPDAR Site via data link). Extensive use of the EES was made in testing the changes made in the tactical software for conduct of various experiments.

(2) Execution of programs in both real-time and interrupted real-time was made utilizing the same software configuration, with a simple software switch to control the mode. This was accomplished at HAPDAR with EES and the real-time system.

(3) The support software is highly important in a multiple-user site environment. Ease in making and testing changes to baseline software demands a systematic means for controlling, recording, and documenting the changes. The Library Management Facility developed under PHSD, and associated software that was developed at DP/HAPDAR is a valuable asset for control of changes. Fortunately, it is directly transferrable to AFAR.

(4) The Ballistic Missile Defense Operating System (BMDOS) from PHSD was modified to operate in real-time or interrupted real-time and was used successfully in several experiments. It is another resource that will be transferred to AFAR and used basically intact. The principal changes that were made were real-time input/output functions, and these will be extended for use in AFAR.

(5) The design of computer-controlled radar systems must consider hardware aids to detect error conditions in execution of radar commands which cannot be detected by software means within a given command frame. This feedback will prevent the occurrence of system 'Hangs' that were experienced on HAPDAR where no feedback of error conditions to 360 existed.* Software recovery at HAPDAR from Hangs consisted of using timer interrupts to break the deadlock and to ignore any data collected in the frame concerned. This is not the way to recover. Extensive data has been documented on this problem in other reports.

(6) The Kalman filter developed for PHSC and used at DP/HAPDAR was not designed for powered (accelerating) targets. Attempts to track powered vehicles in HAPDAR experiments required software logic to test acceleration in range before calling the Kalman filter. The target is kept in polynomial track until the acceleration term becomes less than the value tolerated by the Kalman.

(7) The development of real-time software for HAPDAR resulted in a framework and software transition for the AFAR project for the Radar Control Computer. This is possibly the most important output of the DP/HAPDAR contract. The Real-Time Operating System (BMDOS) is a seasoned software package. The LMF system for program development is a highly useful tool and is being improved for AFAR. Experience in a multiple-user environment is highly useful in the transition to the AFAR environment at KMR.

## 3.2 RECOMMENDATIONS

(1) Future radar designs should be reviewed in light of DP/HAPDAR experience for the following items:

*The problem of "system hangs" is described in the Final Report covering activity through June, 1973, and issued as Data Item A002 under this contract.

(a) Methods of preventing possible deadlock conditions (Hangs) in both hardware and software should be the subject of design reviews. Quick non-disruptive recovery is required.

(b) A unique identification on each type of radar return should be a design requirement. Additionally, a method of framing the returns from a single dwell (look) should be employed allowing blocking of several dwells in a command buffer without complicating software to pack and unpack radar orders and resulting returns.

(c) Separate I/O channels for radar orders and returns are a desirable feature.

(d) Buffering in the radar control unit is a desirable feature. The RCI at HAPDAR contained no buffering capability, thus forcing the 360/65 to have dual buffers for both orders and returns and to operate in a '3-ring circus' rather than a '2-ring circus'. The number of rings in the circus sets the maximum block size with a given filter update rate.*

(e) A byte-oriented command word structure is desirable to aid the packing and unpacking of orders and commands from and to computational forms used in software.

2. The Kalman filter design and criteria for accelerating targets should be studied. The effect of MIRV's on any filters and algorithms for search and acquisition should be studied. HAPDAR data is available to test filters in a realistic accelerating-target environment.

*The "3-ring circus" concept is described in the Final Report issued as Data Item A002 under this contract and covering activity throughJune 1973.

## SECTION 4

## SUMMARY OF DATA ITEMS DELIVERED

This section summarizes the data items prepared during this and preceding DP/HAPDAR contracts.

Table 4-1 lists the documentation furnished under the present contract.

Table 4-2 lists documentation submitted by RCA Corporation to SDC Corporation under SDC Subcontract 73-323. In this effort the prime contract is DAHC-73-C-0055.

Table 4-3 lists documentation delivered by RCA under Contract DAHC60-72-C-0032, the first contract for creation of the Data Processing/HAPDAR Field Test Facility.

TABLE4-1. DATA DELIVERABLE TO ABMDA
under
CONTRACT DAHC60-73-C-0035

| Date Item | Title | No. of Reports |
|---|---|---|
| A001 | Program Plan | Once |
| B001 | Program Plan | Once |
| A002 | Final Technical Report for period ending 6/30/73 | Once |
| A003 | Source Programs & Data | Once |
| B002 | Final Technical Report | Once |

A items cover from 12/18/72 to 6/30/73

B items cover 7/1/73 to 9/30/73.

**TABLE 4-2**

RCA Subcontract 73-323

Data Items Delivered

to SDC under

Contract DAHC-73-C-0055

1. Letter Report to SDC as Data Item 1.0, under RCA Subcontract No. 73-323, entitled "HAPDAR-SACS-H Simulation Experiments," 29-30 June 1973.

2. Data Requirements Plan to SDC as Data Item 2.0, Under RCA Subcontract No. 73-323. Report covers the 26 April DP-3 Athena Experiment, 30 April DP-1 Sphere Drop Experiment, 25 May DP-1 and DP-2 Sphere Drop Experiments.

3. Letter Report to SDC as Data Item 3.0, under RCA Subcontract No. 73-323, entitled Sphere Drop Mission of 25 May 1973.

4. Operations Procedure Plan - Athena Mission of 26 April 1973, to SDC as Data Item 4.0, under RCA Subcontract No. 73-323.

5. Operations Procedure Plan - DP-1 Sphere Drops of 30 April 1973, to SDC as Data Item 4.0, under RCA Subcontract No. 73-323.

6. Operations Procedure Plan - Sphere Drop Mission of 25 May 1973, to SDC as Data Item 4.0, under RCA Subcontract No. 73-323.

7. Operations Procedure Plan - Sphere Drop Mission of 25 May 1973, Supplement No. 1, to SDC as Data Item 4.0, under RCA Subcontract No. 73-323.

8. Monthly Activity Reports, to SDC as Data Item 9.0, under RCA Subcontract No. 73-323, to cover the periods through 31 March 1973, 30 April 1973 and 8 June 1973.

9. Test Evaluation Reports, Data Item 5.0

10. 'Summary of Recorded Data on Targets of Opportunity', Data Item 7.0

TABLE 4-3

Data Deliverables to ABMDA

Under

Contract DAHC60-73-C-0032

| Data Item | Title | No. of Reports |
|---|---|---|
| A001 | Cost Planning and Appraisal Chart | Monthly |
| A002 | Contract Funds Status Report DD-1586 | Periodic |
| A003 | Program Plan | Nov. 1971 |
| A004 | Letter Progress Reports | Monthly (13 total) |
| A005 | Quarterly Technical Reports | Quarterly (4 total) |
| A006 | Final Report | Issued January 1973 |
| A007 | Special Technical Report:<br>Software Capability Description (SCD),<br>Part 1 of 2 parts | Issued May 1972 |
| A008 | Software Test & Evaluation Plan | Issued Oct. 1972 |
| A009 | Special Technical Reports:<br>Software Design Specs. (SDS)<br>Qty: 5 (Covering 5 subsystems) | Issued Sept.-Nov. 1972 |
| A010 | Field Test & Evaluation Plan | Preliminary issue,<br>Aug. 1972 |
| A011 | HAPDAR Interface Specification<br>Radar Computer Interface (RCI)<br>HAPDAR Computer Interface<br>Design Disclosure | Issued Feb. 1972 |
| A012 | HAPDAR Computer Interface RCI<br>Document | Issued March 1972 |

## SECTION 5

## REFERENCES


1. 'AFAR Software Timing Analysis', Dr. J. W. Haake, SDC 4 Oct. 1973, AFAR Program Memorandum 101

2. HAPDAR Field Test Facility Test and Evaluation Reports 41 - 850001-30 thru 34. Texas Instruments, R. O'Bryant, June 1973

3. 'PJL Software Programming Support Requirements' SURC TD 73-047, 16 February 1973, Contract F30602-72-0075

4. 'HAPDAR BMD Software System Software Design Specification, BMDOS Modifications and HAPDAR Real Time Input/Output', RCA October 1972 Contract DAHC60-72-C-0032, Data Item A009-2

5. Software Design Specification for HAPDAR BMD Tactical Process, RCA November 1972, Contract DAHC60-72-C-0032, Data Item A009-5

6. 'Final Technical Report', RCA Contract DAHC60-73-C-0035, Data Item A002, July 1973.

7. 'HAPDAR BMD Software System, Control Console Processing Program', RCA, July 1972 Contract DAHC60-72-C-0032, Data Item A009-3

8. Software Design Specification for External Environment Simulator, RCA August 1972 Contract DAHC-72-C-0032, Data Item A009-1

9. Software Design Specification for Performance Measurement Program, RCA October 1972 Contract DAHC-72-C-0032, Data Item A009-4

## SECTION 6

## GLOSSARY OF TERMS

| | | |
|---|---|---|
| Archive | - | A program developed to list all versions of a load module and preserve it on tape. |
| AFAR | - | ABMDA Field Array Radar - A new solid-state Radar system. |
| AL/I | - | Analyst Language I. A high-level language for BMD applications. |
| BMD | - | Ballistic Missile Defense |
| BMDOS | - | Ballistic Missile Defense Operating System. |
| BSU | - | HAPDAR Beam Steering Unit. |
| EES | - | External Environment Simulator. |
| HAPDAR | - | Hard Point Defense Array Radar. |
| HANG(s) | - | A term denoting a deadlock situation in a real-time system. |
| KMR | - | Kwajalein Missile Range |
| LMF | - | Library Management Facility |
| Master | - | HAPDAR intialization program |
| MIRU | - | Multiple Independent Reentry Vehicle. |
| OS-360 | - | Operating System 360. The Executive Program for 360/65 Precision Acquisition System. |
| PHSD | - | Preliminary Hardsite Demonstration. |
| PJL | - | Passive Jammer Location. |
| PJLRT | - | Passive Jammer Location Real Time Program. |
| PJLSIM | - | Passive Jammer Location Simulator |
| RCI | - | Radar Computer Interface. Interface for 360 to HAPDAR. |
| RONDOA | - | Receive Only Radar Antenna used in PJL experiments. |
| SUM | - | System Utilization Monitor, used to measure computer and software performance. |
| SURC | - | Syracuse University Research Corporation |

TIME ON and TIME OFF - Macros developed to measure program execution times.

VDP - HAPDAR Video Data Processor.

**WRITDL** - A real time recording program.

# APPENDIX 1

## Final HAPDAR Program Versions

This Appendix details release information and program versions for the HAPDAR Real-Time Software system at shutdown on September 28, 1973.

DATE: October 2, 1973

SUBJECT: System Release Notice

FROM: WGS/LC

TO: HAPDAR-AFAR Distribution

System Purpose: This software system operates the HAPDAR radar in real-time and the EES simulator in interrupted real time. It also contains the special template software the the DP experiments.

Special Features: This release contains a standard CTDS for full volume coverage. A single load module operates both Real-time and EES. This is the final HAPDAR version.

Documentation: Partial Documentation from the Archive system showing Versions is attached.

Release Tape Description:

Date Created: 26 Sept. 73

Tape Label: SEP2ØT

| Load Module(s): | Data Set | Member Name | Entry Name |
|---|---|---|---|
| | $DP3_1$ Load | SEP 20 | INIT ## |

```
//SEP22 JOB (ABML,0000),HAPDAR,MSGLEVEL=(1,1),REGION=400K
//STEP0 EXEC PGM=IEHINITT
//SYSPRINT DD SYSOUT=A
//TAPE1 DD UNIT=(2400,,DEFER),DCB=DEN=3
//SYSIN DD *
TAPE1 INITT SER=SEP20T
/*
//STEP1 EXEC PGM=IEBGENER
//SYSPRINT DD SYSOUT=A
//SYSIN DD DUMMY
//SYSUT2 DD DSN=SEP22X,UNIT=2400,LABEL=(1,SL),
//    DCB=(RECFM=FB,LRECL=80,BLKSIZE=9600),
//    DISP=(NEW,CATLG),VOL=(PRIVATE,RETAIN,SER=SEP20T)
//SYSUT1 DD *
**** ARCHIVE INDEX   MODULES:  SEP22   LIBRARY:  DP3.LOAD
****                          USER:  HAPDAR
****
****  ARCHIVED MACRO LIBRARIES:
**
**      DEFT.ASM            VOL=SER=000981
**      DEFH.ASM            VOL=SER=000916
**      DEF.ASM             VOL=SER=000981
**
****                    CREATED:  73.264     00:03:14.9
****
**** PROG    VRS  LIB        LANGUAGE                    OPTIONS
****
      BLKSTS  SG  BMDOS     ASMH
      BMDIPL  BI  BMDOS     ASMH
      BMDISP  SG  BMDOS     ASMH
      BMDRST  SG  BMDOS     ASMH
      BMDOSB  XG  BMDOS     ASMH
      BMDSWT  SG  BMDOS     ASMH
      BMTERT  SG  BMDOS     ASMH
      COIPXM  SG  BMDOS     ASMH
      CHANGE  SG  BMDOS     ASMH
      CLEAMW  SG  BMDOS     ASMH
      COMPAR  BI  BMDOS     ASMH
      DMBCOP  TC  BMDOS     ASMH
      FORTAN  SG  BMDOS     ASMH
      FIACCE  SG  BMDOS     ASMH
      FIAPIP  SG  BMDOS     ASMH
      FIGETS  SG  BMDOS     ASMH
      FIEPCK  SG  BMDOS     ASMH
      FINISD  ZZ  BMDOS     ASMH
      FIPUTS  SG  BMDOS     ASMH
      FIRELE  SG  BMDOS     ASMH
      FLIPIN  XG  BMDOS     ASMH
      FLPSVE  SG  BMDOS     ASMH
      GETSPC  SG  BMDOS     ASMH
      GETSTC  SE  BMDOS     ASMH
      HASHTL  SG  BMDOS     ASMH
      HEMCOP  SG  BMDOS     ASMH
      HEXTRN  SG  BMDOS     ASMH
      INMCOD  SH  BMDOS     ASMH
      INACCE  SG  BMDOS     ASMH
      INTSCD  SG  BMDOS     ASMH
      IOHDT#  BZ  BMDOS     ASMH
      IPGMSS  SG  BMDOS     ASMH
      IPIGNB  SG  BMDOS     ASMH
```

| | | | | |
|---|---|---|---|---|
| LOCK22 | SG | BMDOS | ASMH | 00000100 |
| MCHELM | SG | BMDOS | ASMH | 00000200 |
| PCELTM | SG | BMDOS | ASMH | 00000300 |
| RCELTM | SG | BMDOS | ASMH | 00000400 |
| RDTOOT | SG | BMDOS | ASMH | 00000500 |
| RECTIM | SG | BMDOS | ASMH | 00000600 |
| REPEAT | SG | BMDOS | ASMH | 00000700 |
| REQUST | SG | BMDOS | ASMH | 00000800 |
| RESERV | SG | BMDOS | ASMH | 00000900 |
| RESUME | ZZ | BMDOS | ASMH | 00001000 |
| RESUME | XX | BMDOS | ASMH | 00001100 |
| RESUME | YY | BMDOS | ASMH | 00001200 |
| RTPACK | XG | BMDOS | ASMH | 00001300 |
| RTUNPK | SG | BMDOS | ASMH | 00001400 |
| SCLOCK | SG | BMDOS | ASMH | 00001500 |
| STOPTO | SG | BMDOS | ASMH | 00001600 |
| STOPTM | ZZ | BMDOS | ASMH | 00001700 |
| TSIMED | SG | BMDOS | ASMH | 00001800 |
| UNLOCK | SG | BMDOS | ASMH | 00001900 |
| WRITOK | SH | BMDOS | ASMH | 00002000 |
| WRITOL | ZZ | BMDOS | ASMH | 00002100 |
| WRITOL | XX | BMDOS | ASMH | 00002200 |
| WRITOL | YY | BMDOS | ASMH | 00002300 |
| CTDSTALS | | | ASMH | 00002400 |
| HCDST | | DEFH | ASMH | 00002500 |
| PSDS | | DEFH | ASMH | 00002600 |
| TICDS | | DEFH | ASMH | 00002700 |
| DPDS | | DEFT | ASMH | 00002800 |
| SCHEXP | | DEFT | ASMH | 00002900 |
| BEGINZ | ZZ | EXPR | ASMH | 00003000 |
| CMAZEL | TS | EXPR | FORTRAN | 00003100 |
| COOKED | ZZ | EXPR | FORTRAN | 00003200 |
| CPRMAT | ZZ | EXPR | FORTRAN | 00003300 |
| CRATAN | TS | EXPR | FORTRAN | 00003400 |
| CRPVEC | ZZ | EXPR | FORTRAN | 00003500 |
| CUVEED | SI | EXPR | FORTRAN | 00003600 |
| FLOTDS | SS | EXPR | FORTRAN | 00003700 |
| GETBKD | ZZ | EXPR | FORTRAN | 00003800 |
| PUTAMD | ZZ | EXPR | FORTRAN | 00003900 |
| PUTBKD | ZZ | EXPR | FORTRAN | 00004000 |
| SCALED | DP | EXPR | FORTRAN | 00004100 |
| SETUPD | ZZ | EXPR | FORTRAN | 00004200 |
| SETBLD | SG | EXPR | ASMH | 00004300 |
| START | DP | EXPR | FORTRAN | 00004400 |
| TRANSF | DD | EXPR | FORTRAN | 00004500 |
| UPDATE | DP | EXPR | FORTRAN | 00004600 |
| VPASSD | XX | EXPR | ASMH | 00004700 |
| ZTRACE | ZZ | EXPR | ASMH | 00004800 |
| ADDRED | HI | SUPPORT | ASMH | 00004900 |
| DEMOD | HI | SUPPORT | FORTRAN | 00005000 |
| FILETD | HI | SUPPORT | ASMH | 00005100 |
| INSTAT | HI | SUPPORT | ASMH | 00005200 |
| LOCMVD | HI | SUPPORT | ASMH | 00005300 |
| RECORD | HI | SUPPORT | ASMH | 00005400 |
| WSTFIL | HI | SUPPORT | ASMH | 00005500 |
| ALIPOS | SS | TACTICAL | ASMH | 00005600 |
| ARRAY | HC | TACTICAL | ASMH | 00005700 |
| ASSESS | TS | TACTICAL | ASMH | 00005800 |
| BOUNDS | DF | TACTICAL | ASMH | 00005900 |
| BSP000 | SG | TACTICAL | ASMH | 00006000 |

```
DSPMOD   SG  TACTICAL  ASMH                                    00012200
CHMOPD   ZZ  TACTICAL  ASMH                                    00012300
CC       XG  TACTICAL  FORTRAN                                 00012400
CCTOPD   H1  TACTICAL  FORTRAN                                 00012500
CCPACK   SG  TACTICAL  ASMH                                    00012600
CCRESC   SG  TACTICAL  ASMH                                    00012700
CCUPCK   SG  TACTICAL  ASMH                                    00012800
CHMSOL   ZZ  TACTICAL  ASMH                                    00012900
CCMOD    SG  TACTICAL  ASMH                                    00013000
CCMOD    SG  TACTICAL  ASMH                                    00013100
CTLPAS   SG  TACTICAL  ASMH                                    00013200
DLCORE   K1  TACTICAL  ASMH                                    00013300
DLPUMP   K1  TACTICAL  ASMH                                    00013400
DLFFMT   K3  TACTICAL  ASMH                                    00013500
DLCOFF   K1  TACTICAL  ASMH                                    00013600
DLVERM   PS  TACTICAL  ASMH                                    00013700
DTTAPO   SG  TACTICAL  ASMH                                    00013800
DTTPOD   XK  TACTICAL  ASMH                                    00013900
EXPRMD   IA  TACTICAL  ASMH                                    00014000
HMDVRP   H1  TACTICAL  FORTRAN                                 00014100
IOCMRT   SG  TACTICAL  ASMH                                    00014200
KMSGD    XG  TACTICAL  ASMH                                    00014300
MASTER   ZZ  TACTICAL  ASMH                                    00014400
MKPOOD   SG  TACTICAL  ASMH                                    00014500
MYBOTT   K1  TACTICAL  ASMH                                    00014600
MPFALL   SH  TACTICAL  ASMH                                    00014700
MYPURG   SG  TACTICAL  ASMH                                    00014800
NOTODE   SG  TACTICAL  ASMH                                    00014900
NSPOTT   SG  TACTICAL  ASMH                                    00015000
ONLINE   ED  TACTICAL  ASMH                                    00015100
PRINIT   H1  TACTICAL  ASMH                                    00015200
PRNTFT   LE  TACTICAL  ASMH                                    00015300
QASUAS   K1  TACTICAL  ASMH                                    00015400
RTNCOP   H1  TACTICAL  ASMH                                    00015500
RUXXXZ   RF  TACTICAL  FORTRAN                                 00015600
SCHALL   SG  TACTICAL  ASMH                                    00015700
TRBEAP   H1  TACTICAL  FORTRAN                                 00015800
TIMED    H1  TACTICAL  FORTRAN                                 00015900
VIDEO    SG  TACTICAL  ASMH                                    00016000
WRLASW   H1  TACTICAL  ASMH                                    00016100
WRLCRT   H1  TACTICAL  ASMH                                    00016200
WRLFOT   H1  TACTICAL  ASMH                                    00016300
WRLINW   H1  TACTICAL  ASMH                                    00016400
WRLIRS   K3  TACTICAL  ASMH                                    00016500
WRLKOT   K1  TACTICAL  ASMH                                    00016600
WRLTRW   K1  TACTICAL  ASMH                                    00016700
WLCWRP   BT  TACTICAL  ASMH                                    00016800
/*                                                             00016900
//STEP2 EXEC LKED,PARM.LKED='MAP,LET,LIST,CALL,SIZE=(360K,64K)' 00017000
//LKED.SYSLMOD DD DSN=A.SHRT(SLP20),VOL=SER=WORK01,DISP=(NEW,PASS) 00017100
//LKED.SYSLIB DD DSN=TACTICAL.OBJECT,DISP=SHR                  00017200
//          DD DISP=SHR,DSN=AMDOS.OBJECT                       00017210
//          DD DISP=SHR,DSN=DFT.OBJECT                         00017220
//          DD DISP=SHR,DSN=DFM.OBJECT                         00017230
//          DD DISP=SHR,DSN=DFL.OBJECT                         00017240
//          DD DISP=SHR,DSN=SUPPORT.OBJECT                     00017250
//          DD DISP=SHR,DSN=FORTPLI.OBJECT                     00017260
//INCLUDE DD DISP=SHR,DSN=SUPPORT.LOAD                         00017270
//AMDOS DD DSN=AMDOS.OBJECT,DISP=SHR                           00017280
//CROSTALS DD DSN=CROSTALS,DISP=SHR                            00017300
```

# APPENDIX 2

## Timing Macros

This Appendix contains details concerning special macros that were developed to aid the timing of HAPDAR software functions as a replacement for Hardware monitoring (SUM) equipment that was used for earlier timing experiments by Texas Instruments in Reference 2.

Timing Macros

TIME ON - TIME OFF

## 1. General

In order to obtain software timing statistics approximating the output of the hardware SUM equipment used with the HAPDAR Software, it is necessary to insert the timing macros-TIMEON and TIMEOFF into each module or section of code to be timed.

In order to envoke the TIMEON and TIMEOFF macros at compile time, a global parameter &TIMER must be defined as follows:

```
        GBLC  &TIMER
&TIMER  SETC  'ON'
```

If the above code is not included, generation of timing macros will be suppressed. This modification allows the user to suppress the expansion of the macros without removing the actual statements from the code.

Each TIMEON-TIMEOFF Macro pair is given (by the Configuration Manager) a unique identifier (1-99) which corresponds to an entry in the AFAR Timing Information Table (ATIT) residing within the code for MASTER (BMDOS Initialization). Statistics are accrued in the table until the end of the BMDOS run (EES or Real-Time), at which time the termination routine (FINIS) is entered. A modification to FINIS will calculate from the timing statistics available the average time for each execution as well as the Standard Deviation. This data will be printed out at the end of the run.

The current execution time (timing overhead) for each TIMEON-TIMEOFF pair is approximately:

| | | |
|---|---|---|
| TIMEON | 21.85 | μ sec. |
| TIMEOFF | 34.81 | μ sec. |
| TOTAL | 56.66 | μ sec. |

## II. Timing Activation Parameters

The timing code while it is resident in each module to be timed, will not be executed unless the timing table (ATIT) entry for that module or section of code has been turned on via the following Run Time input parameter added to the "OPTIN" BMDOS Data Set:

| Column 1 | 10 ... 80 |
|---|---|
| TIMING | $n, n_1-n_2, n_3, n_4, \ldots$etc. |

The parameter "TIMING" must appear in the data card starting in column 1, followed by the numbers of the entries to be turned on starting in column 10. Entries must be separated by a comma (,), and a range of entries may be specified by the first and last numbers in the range separated by a dash (-). A blank (Ø) terminates the parameter field.

Example:

```
TIMING   1,3-8,11,21-26,5Ø-99
```

An additional Data Definition (DD) card (detailed below) must be added to the run deck to allow printing of the run-statistics.

```
//TIMING  DD  SYSOUT=A
```

## 1.0 TIMEON Statement

The TIMEON Statement initiates timing within a module by creating a period START time (START time = Six Hour Pseudo Clock Time - CPU Clock Time) and updating the corresponding field in the AFAR Timing Information Table (ATIT) Entry identified with the TIMEON Statement. Also an event counter is incremented by 1 every time a TIMEON is processed.

| | | |
|---|---|---|
| label | TIMEON | entry-identifier [,COMREG={YES / NO}] [,SSM={YES / NO}] [,REGS={YES / NO}] |

a. Entry-identifier

Number (1-99) corresponding to the relative position of the entry within the AFAR Timing Information Table.

b. COMREG={YES / NO}

This parameter is optional. If COMREG=YES is specified, the generated code will assume that the base register for the COMREG table has been established somewhere else in the code. If COMREG=NO, or the operand is omitted the macro generation will establish Register 14 as a base register (and DROP 14 at the end of the generation).

c. SSM={YES / NO}

This parameter is optional. If SSM=NO is specified, no code will be generated to disable and enable interrupts before and after timing calculations are made. Otherwise, interrupts will be turned off (SSM *+1) before and turned on (SSM VEXITPSW) after the calculation is made.

d. REGS={YES / NO}

This parameter is optional. If REGS=YES is specified, Registers 14, 15, 0, 1 will be saved into and restored from a save area within the macro generation. Otherwise, the contents of those registers will not be guaranteed.

Note: Two TIMEON statements with the same entry-identifier cannot be issued in a row.

## 2.0 TIMEOFF Statement

The TIMEOFF Statement calculates the time for the end of a period (END Time-Six Hour Pseudo Clock Time-CPU Clock Time) and subtracts that time from the START time (calculated by the TIMEON statement) to obtain the duration of the period. Then the fields within the AFAR Timing Information Table corresponding to the sum of the periods calculated ($\Sigma\Delta T$) and the sum of the squares of the differences between the calculated period and a representative period ($\Sigma(T_2-T_n)^2$) are updated.

| | | |
|---|---|---|
| label | TIMEOFF | entry-identifier [,COMREG={YES / NO}] [,SSM={YES / NO}] [,REGS={YES / NO}] |

a. Entry-identifier

Number (1-99) corresponding to the relative position of the entry within the AFAR Timing Information Table.

b. COMREG={YES / NO}

This parameter is optional. If COMREG=YES is specified, the generated code will assume that the base register for the COMREG table has been established somewhere else in the code. If COMREG=NO, or the operand is omitted the macro generation will establish Register 14 as a base register (and DROP 14 at the end of the generation).

c. SSM={YES / NO}

This parameter is optional. If SSM=NO is specified, no code will be generated to disable and enable interrupts before and after timing calculations are made. Otherwise, interrupts will be turned off (SSM *+1) before and turned on (SSM VEXITPSW) after the calculation is made.

d. REGS={YES / NO}

This parameter is optional. If REGS=YES is specified, Registers 14, 15, Ø, 1 will be saved into and restored from a save area within the macro generation. Otherwise, the contents of those registers will not be guaranteed.

Note: Two TIMEOFF statements with the same entry-identifier can not be issued in a row. Also a TIMEOFF may not be executed without a previous TIMEON statement with the same entry-identifier.

## 3.0 AFAR Timing Information Table (ATIT)

The Timing Information Table contains 99 entries, each consisting of 6 full word data fields (detailed below). The table is resident in MASTER and is generated by adding an inline macro instruction (also named ATIT with no parameters). The table is initialized by modifications in the subroutine VLXWTOR located in MASTER and entries are turned on via run-time parameters.

Timing Information Table Entry

| BYTE Ø | 4 | 8 | 9 | 12 | 16 | 20 23 |
|---|---|---|---|---|---|---|
| ATISTART | ATIACCUM | FLAG | ATIEVENT | ATIT2 | ATLTI | ATISTD |

| Field | Length | Description |
|---|---|---|
| ATISTART | 4 Bytes (Binary) | Period start time stored by TIMEON and used by TIMEOFF to calculate duration START TIME= Six Hour Pseudo Clock Time - System Timer |
| ATIACCUM | 4 Bytes (Binary) | The sum of the times recorded for all accesses of the module (or TIMEON-TIMEOFF pair) $S_i = \Sigma_{n=1}^{N} T_n$ |
| FLAG | 1 Byte (Binary) | Status indicator:<br>x'ØØ' - Entry Inactive<br>x'8Ø' - Entry Active<br>x'40' - TIMEON has been Issued (x'CØ')<br>x'20' - TIMEON twice in a row (x'EØ')<br>x'1Ø' - TIMEON without TIMEON (x'9Ø) |
| ATIT2 | 4 Bytes (Binary) | The sum of the squares of the differences between the calculated time interval and the second time interval calculated (the first calculation is skipped because initialization time throws the standard deviation calculation off. $S_{i2} = \Sigma_{n=1}^{N} (T_i - T_N)^2$ |
| ATITI | 4 Bytes (Binary) | The initial time interval calculated ($T_1$) (see description for ATIT2). |

| | | |
|---|---|---|
| ATISTD | 4 Bytes (Floating Pt) | This field contains the Standard Deviation (updated by FINIS only at the completion of the mission-during the mission it is used as a work area by the TIMEOFF Macro). $\sigma = \frac{1}{N-1} (\Sigma_{n=1}^{N} (T_n - \overline{T})^2) =$ Std. Deviation |

# APPENDIX 3

RCA

SUPPORT

FOR

PJL EXPERIMENT

## TABLE OF CONTENTS

PART I - A general description of the necessary software development and software support required of the RCA Field Test Facility to accomplish the integration of PJL software into the HAPDAR system.

PART II - A detailed documentation of the PJL/BMDOS software integration routines.

## PART I GENERAL PJL SUPPORT

Three distinct modes of operation of the PJL experiment are supported by the FTF. They are:

1. Execution of the PJLRT/PJLSIM software in a non-real-time environment running under the control of OS.

2. Execution of the PJLRT/PJLSIM software in an interrupted real-time environment where PJLRT runs under the control of BMDOS and PJLSIM runs under the control of OS.

3. Execution of the PJLRT software in a real-time environment where PJLRT runs under the control of BMDOS and interacts with a real-world as sensed through reports from HAPDAR sub-units, PJL equipment, and RONDO A remote radar.

Each of these three modes of operation required the development of various interface software and support software. This document is an attempt to show the scope of this software development by RCA and to serve as a source of information for the duration of the PJL experiment.

A. RCA Support and Development for Mode 1 (PJLRT/PJLSIM Under OS)

RCA's primary task was that of systems support to help SURC programmers clear up any incompatabilities between Sigma IV FORTRAN and IBM 360/65 FORTRAN EXTENDED, and to aid in solving various JCL problems.

B. RCA Support and Development for Mode 2 (PJLRT Under BMDOS, PJLSIM Under OS)

Since PJLRT and PJLSIM were designed to communicate through only one common area, they lent themselves to execution in an IRT environment such as was developed for POST PHSD. To accomplish this it was necessary to replace the PJL Driver with WLC and have MASTER call various SURC supplied initialization routines, namely, PJLINT, PJLCON, PJLSII, and PJLPRT. Also, it was necessary to replace the SURC I/O software with FTF I/O software routines, TAPESV and PJTAPE.

It was at this stage of the development that PJLRT was run in a real-time environment under the control of BMDOS for the first time. The problems that resulted were primarily due to differences in supporting philosophies between OS and BMDOS.

C. RCA Support and Development for Mode 3

Since modes 1 and 2 verified the PJL real-time software, mode three was mostly verification of RCA's I/O interface software.

The executive for I/O interfacing is PJLIO. This executive along with all the support routines pictured in Figure 1 comprise the bulk of the RCA support software for the PJL experiment.

A preliminary PJLIO real-time process was established using a dummy PJLRT. In this manner the FTF real-time structure was quickly debugged.

With the real-time I/O structure fairly well established, it was a simple matter to execute the real PJLRT software.

RCA FTF SOFTWARE DEVELOPMENT
FOR
PJL REAL-TIME PROCESS

| | | | |
|---|---|---|---|
| PJLIO | (S)* | TTYOUT | (S) |
| RTUNPK | (M) | MASTER | (M) |
| PJLRTN | (S) | CLOCK | (S) |
| PJLRRA | (S) | TAPESV | (M) |
| CONIN | (M) | RTPACK | (M) |
| CCUNPK | (M) | PJLORD | (S) |
| PJLPAS | (S) | PJLROA | (S) |
| PJIRIG | (M) | CONOT | (M) |
| PJLR | (S) | CCPACK | (M) |
| PJLO | (S) | | |

For a description of the precise role each of these modules plays in the real-time system, consult part II of this document.

D. MISSION TIME SUPPORT

All PJL users get BMD mission time support via a call to the RCA subroutine, CLOCK.

See CLOCK documentation for further details.

E. DATA LINK SUPPORT

PJL users have real-time data recording capabilities available through the use of the RCA supplied TAPESV subroutine. See TAPESV documentation for further details.

F. INITIALIZATION SUPPORT

Successful initialization of the PJL process is accomplished in the following manner.

1. Files PJLO and PJLR are zeroed out by the BMD loader

2. MASTER for PJL processing calls various SURC supplies initialization routines (PJLINT, PJLPMI).

* (M = modified from basic FTF software; S = created from scratch)

## G. POST MISSION SUPPORT

Through use of the RCA supplied PJTAPE routine, PJL post mission analysis routines have access to the PJL data recorded by various calls to TAPESV.

## H. 1218 PRINTER SUPPORT

It is desirable during real-time for PJL analysts to be able to interrogate and print various critical run parameters. This capability is supported using a mixture of RCA/SURC/SPERRY software.

Basic communication to on-line SURC provided display programs is handled via switches on the HAPDAR control consol. These programs interface with RCA supplied I/O programs which handle the actual formatting of desired messages and transmition to the RCI. A SPERRY provided character scan program resides in the 1218 which will formulate line messages and print as necessary.

- HAPDAR/PJL INTERFACE SOFTWARE
- HAPDAR/PJL DATA FLOW

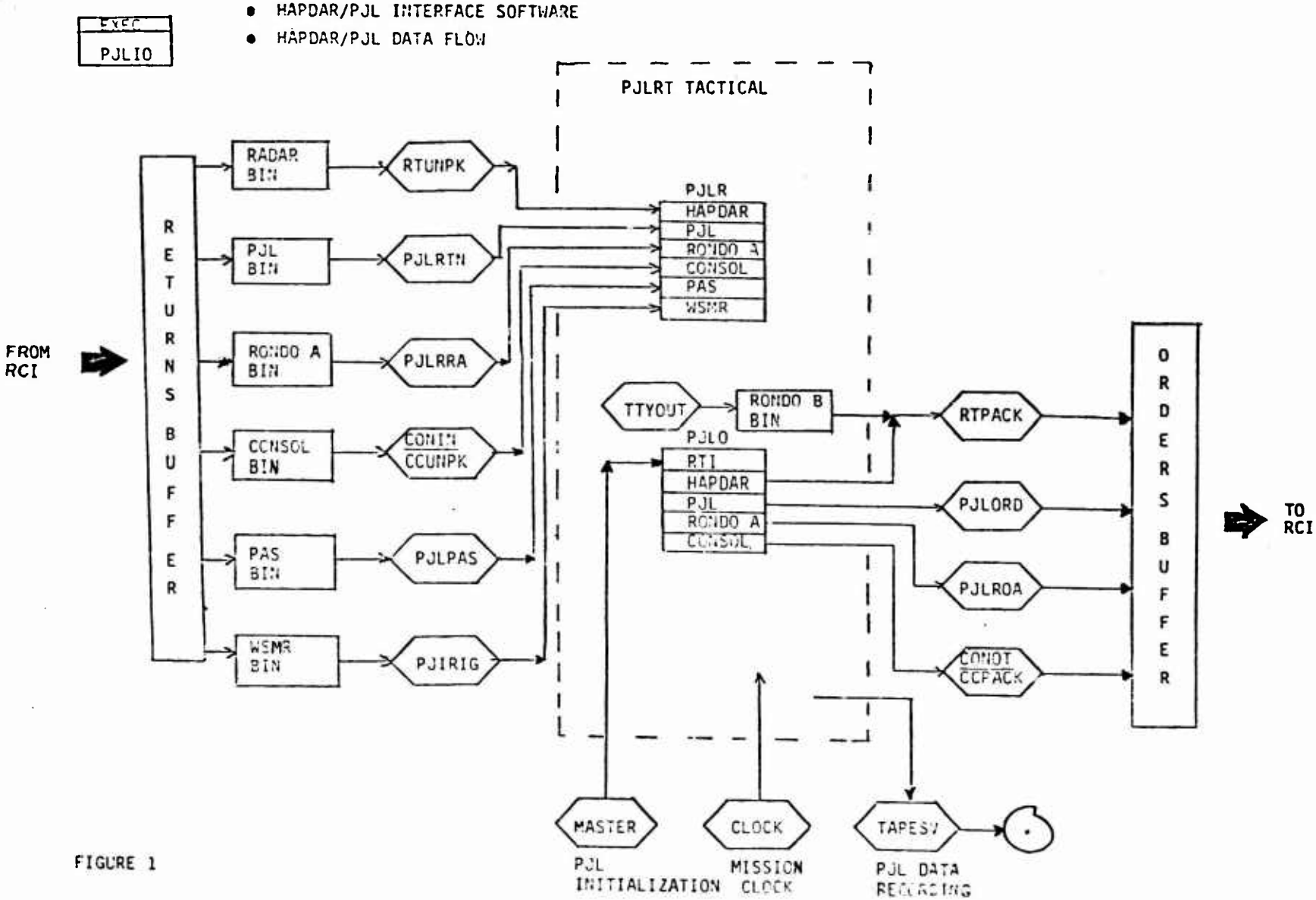



FIGURE 1

PART II DETAILED DOCUMENTATION

A. IDENTIFICATION PJLIO RT CONTROL EXECUTIVE

TITLE: PJLI/O EXECUTIVE

ID: PJLIO

PROGRAMMER: T. Royer

B. FUNCTIONAL DESCRIPTION

1. General: All interfacing with PJL software is done by way of two common blocks called:

PJLO and PJLR

For our purposes at HAPDAR PJLO and PJLR are viewed as static files.

PJLO ⟶ orders for the radar.

PJLR ⟶ returns from the radar.

Basic integration EXECUTIVE software (PJLIO) must accomplish the following tasks.

a. Support a standard 3 ring circus cycle

b. Unpack RADAR/PJL/RONDO/CONSOLE/RDT/PAS

c. Execute PJLRT

d. Pack RADAR/RONDO/PJL/CONSOLE orders

e. Schedule PJLIO for next cycle

f. Provide for termination and execution PMA subroutine before total control is returned to OS.

a. Support standard three ring circus

PJL THREE RING CIRCUS

| | |
|---|---|
| Cycle n=0 | MASTER |
| Cycle n=1 | PJLRT<br>PACK HAPDAR/RONDO/PJL/CC<br>SCHEDULE HRRTRN FOR N+1<br>WAIT |
| Cycle n=2 | XMIT ORDERS MADE IN CYCLE n-1<br>PJLRT<br>PACK HAPDAR/RONDO/PJL/CC<br>SCHEDULE HRRTRN FOR n+1<br>WAIT |
| Cycle n=3 | XMIT ORDERS MADE IN CYCLE n-1<br>UNPACK RADAR/RONDO/PJL/CC/PAS/RDT FROM CYCLE n-2<br>PJLRT<br>PACK HAPDAR/RONDO/PJL/CC<br>SCHEDULE<br>WAIT |
| Cycle n=4 | etc. |

b. UNPACK RADAR/PJL/RONDO/CONSOLE/RDT/PAS

Use the following support routines:

RTUNPK (unpack radar)

no calling sequence
Input is the standard RADARBIN

PJLRTN (unpack PJL)

Set up PJLBIN (480 bytes)

Call PJLRTN (Input → PJLBIN, Output → PJLIN #1#1)

PJLRRA (unpack RONDO A returns)

Make sure RONDO is properly set up 16 bytes

Call PJLRRA ( Input → RONDOBIN, Output → RONDOI #1#1)

CONIN

Check first GID on GIDQ
(NOTE: GIDS will be replaced by mode words which will distinquish between track and search orders. GIDQ (MODES) will be built in ORDRMAKE. If console input from the RCI was requested the high order bit of the first mode word in the GIDQ will be set.

No calling sequence required, just call CONIN.

RDT IRIG Time

Call PJIRIG

with no calling sequence

This routine will be called when valid IRIG message time is known to exist in the message buffer. Unpacked integer data will be stored into

WISEC
WIMS } in PJLR
WITIME

PJLPAS (unpack PAS data)

Standard calling sequence

Call PJLPAS (PASBIN, PIRNGE)

PIRNGE is the FWA of the integer PJL PAS data located in PJLR file.

c. Execute PJLRT

PJLRT software is set up to receive 'live' returns on cycle three as dictated by the three ring circus cyclical processing.

4. PACK RADAR/PJL/RONDO/CONSOLE

Use the following support routines:

RTPACK (pack RADAR)

No calling sequence (# orders in VDPOUT #1#1)

PJLROA (pack RONDO A orders)

Check RONDOADD for =-1
If -1, no RONDO scheduled this cycle
If +1, use stored contents as output address

(NOTE: RTPACK will have left room for both RONDO and PJL in front of the first order if necessary)
RTPACK will maintain RONDOADD, PJLADD

Calling sequence

Call PJLROA (RONDOT#2#1, RONDOADD)

PJLORD (Pack PJL equipment orders)

Check contents of PJLADD.

If =-1, no PJL this cycle
If = +, use contents as FWA of packed output from PJLORD.

Calling sequence

Call PJLORD (PJLOUT#2#1,PJLADD)

CONOT

Check CECCOP in PJLO
If set pack consol by calling CONOUT and chain to end of order buffer
If not set forget packing consol

Calling sequence

None

HRORDR was not used but was incorporated into PJLIO as an internal subroutine. This subroutine is called ORDRMAKE and is called immediately after PJLRT.

e. Schedule PJLIO for Next Cycle

In BMDCP(2), BMDCP(3) is a floating point mission time to reschedule PJLIO.

PJLIO returns in BMDCP (4), (5) a floating point time:

if time = 0.0, everything is OK;

if time > 0, then PJLIO indicates to PJLRT that a hang condition was encountered.

f. Termination

If the field XEOM is set greater than zero, then PJLIO will terminate the PJLRT process, call PMA subroutine (quick look post mission analysis routine) and return control to OS.

2. Key integration details:

PJL software interfacing has been designed to have a minimum of interaction between modules for simplicity sake. Some interaction was necessary as briefly highlighted below:

1. Consol Pack/Unpack interaction

The CC external function consol word will be valid in only the last order to HAPDAR. Previous CC-EF words are set to zero conditions.

If CECCOP is set then PSL/O knows to pack and chain consol. In addition the CECCIP bit will be packed into the highest bit of the first mode/wordin the current GIDQ.

2. PJL, RONDO schedule

When either PJLOUT #1#1 or
RONDOT #1#1 or both
are set > 0, then RTPACK must leave room for this data respectively. Also, PJLADD and RONDOADD will be maintained by RTPACK to tell subsequent PJL and RONDO Pack routines where to store their output.

3. XEOM - end of mission flag

4. BMDCP(2)(3) HRRTRN reschedule

5. BMDCP(4)(5) Hang time indications

6. ORDRMAKE/PJLIO interaction.

ORDRMAKE makes GIDQ's whose contents are mode words. A zero in GIDQ instance indicates no more orders.

| | | |
|---|---|---|
| Modes | 1,3,5,7 | search |
| Modes | 8*,2,4,6 | track |

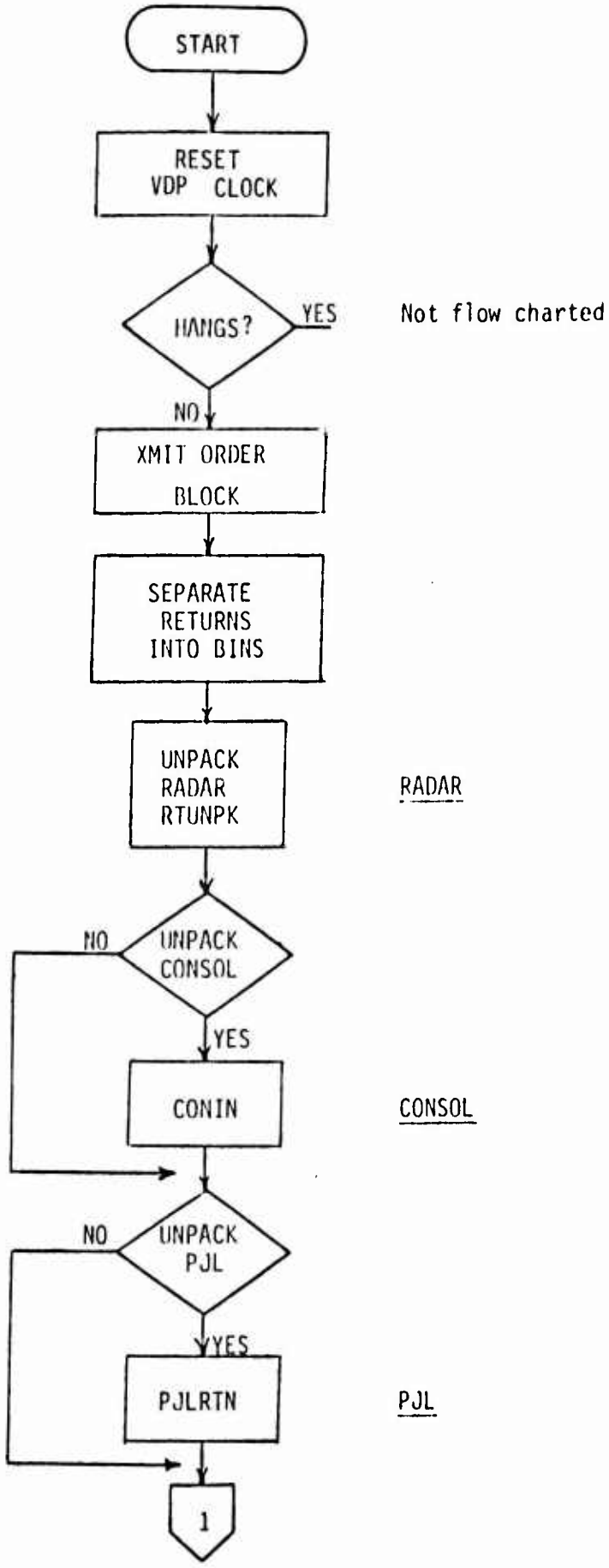
*Note: for track split gate 000 mode is made 1000 so an invalid end of order indication will not be indicated.

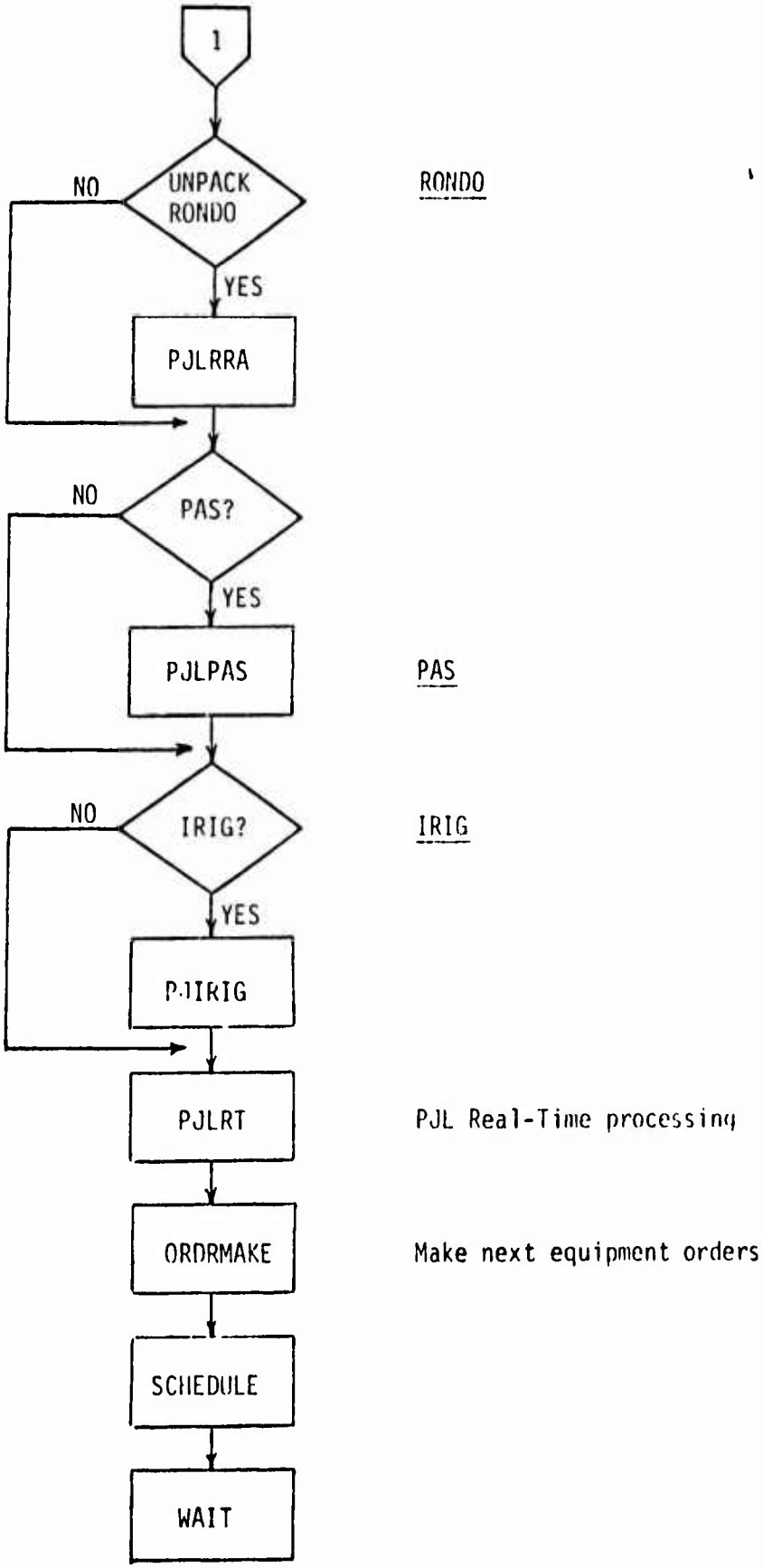
PJLIO will check GIDQ for any scheduled returns.

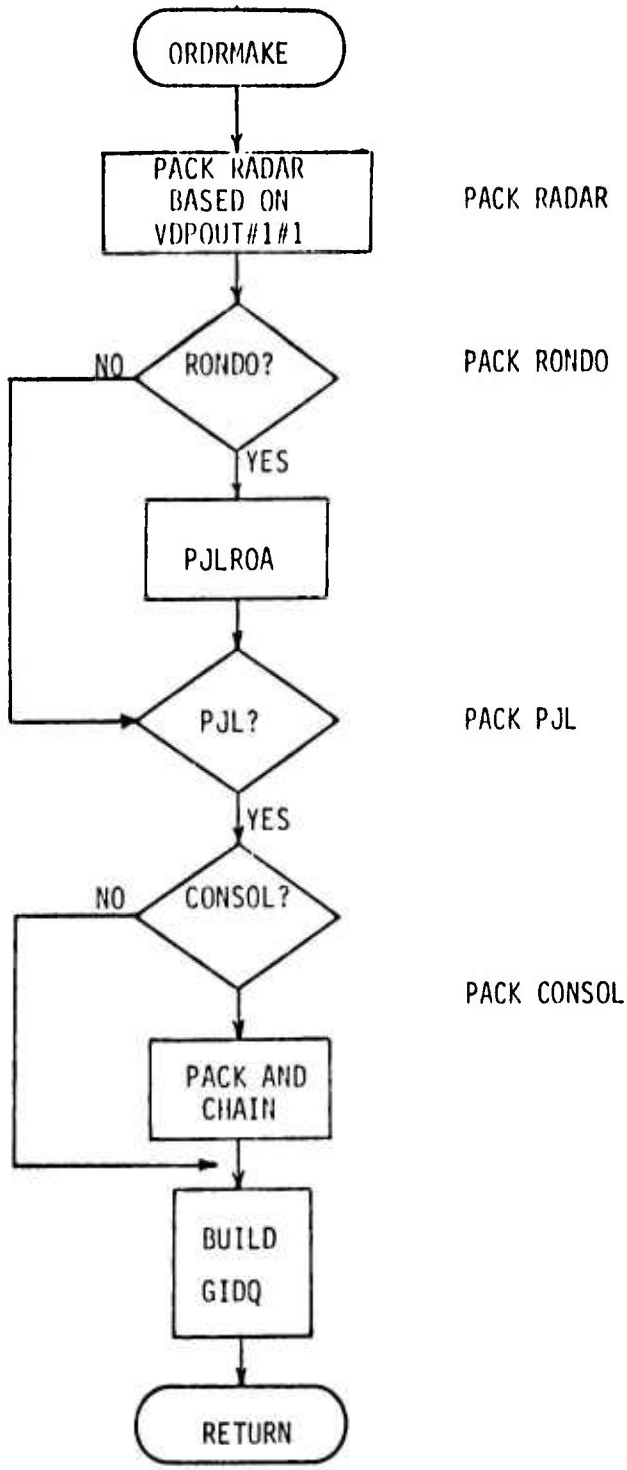
RTUNPK will check modes for appropriate track or search decode.

7. PJLADD, RONDOADD will be maintained for PJLIO by RTPACK so PJLORD and PJLROA will know were to put the packed RONDO AND PJL data.

3. General PJLIO flow chart




1
UNPACK RONDO
NO
YES
RONDO
PJLRRA
PAS?
NO
YES
PJLPAS
PAS
IRIG?
NO
YES
IRIG
PJIRIG
PJLRT
PJL Real-Time processing
ORDRMAKE
Make next equipment orders
SCHEDULE
WAIT


ORDRMAKE
PACK RADAR BASED ON VDPOUT#1#1
PACK RADAR
RONDO?
NO
YES
PACK RONDO
PJLROA
PJL?
PACK PJL
YES
CONSOL?
NO
PACK CONSOL
PACK AND CHAIN
BUILD GIDQ
RETURN

A. IDENTIFICATION EXECUTIVE SUBROUTINE

TITLE: PJL CONTROL CONSOLE INTERFACE

ID: CC

PROGRAMMER: J. J. OGAN

B. FUNCTIONAL DESCRIPTION

This routine is the executive interface routine for consol communication to the PJL experiment. Packed input/output is via the common areas CCIN and CCOUT. PJLIO uses channel command words to dump consol input data into CCIN and transmit consol output data from CCOUT.

CC has two basic entry points, CONIN and CONOT, used to accomplish the basic consol input and output respectively.

C. INPUT

CCIN - 20 bytes of input data are dumped into this area whenever PJLIO has received a full report from the consol. The data is left justified in the common area with the function and ID bits stripped off (total of six bits) by PJLIO.

CONIN will process each word. It compares each of the five input words with their previous values. When a change is sensed, the appropriate unpack entry point will be involked. These entry points (IN1 - IN5) are all contained in the subroutines CCUNPK (for PJL).

The routine ANDH is used to turn those bits to zero in the INPUT data that are return to zero consol parameters. The 'modified' consol input words are then stored into CCOI1 - CCOI5 for comparison with the next group of 5 consol report words.

D. OUTPUT

CCOUT - 80 bytes of output data are dumped into this area to be transmitted by PJLIO to the consol display.

The data is properly positioned for RCI transmition with appropriate consol identity and function codes (six bits) masked into the high six bits of each word.

CONOT accesses the consol output data in PJLR starting at C6SGNL and ending with CECCOP.

Proper conversion of hexadecimal count ranges, azimuths and elevations to 4 bit BCD, as specified in consol output formats, are made using RTBCD, and ANGBCD. CONOT uses the following entry points in CCPACK to make the appropriate consol output words:

| | |
|---|---|
| OUT1P2 | PACKS words 1 and 2 according to format |
| C6SGNL | signal strength |
| NRL(3) | three 4-bit BCD range digits (lsb = 1 meter), one character right justified in 3 half words |
| NRG(4) | four 4-bit BCD range digits (lsb = 1K meter), one character right justified in 4 half words |
| C2DATA | channel access - bits 17, 16, 15 in word 2 of consol output. This field was not on the original Sperry consol format specification. |
| OUT3 | PACKS word 3 according to format |
| IDUM | Diode fault not used by PJL |
| C7BUP | Beta limit (up) |
| C7BDWN | Beta limit (down) |
| NANGL | 4-bit BCD elevation all four characters in 4 bytes. |
| OUT4 | PACKS word 4 according to format |
| C8ART | Alpha limit right |
| C8ALFT | Alpha limit left |
| NANGL | 4 bit BCD azimuth all 4 characters in 4 bytes. |
| OUT5 | PACKS word 5 according to format |
| C9TKCH | active channel indicators |
| C9GAIN | manual gain operative |
| IRBL | Range coast, manual range track, auto range track functionally OR'ed together. |
| ICBL | coast, manual track, auto track functionally OR'ed together |
| C9WAIT | standby |

| | |
|---|---|
| C9AAQ | auto-acquisition |
| C9DESG | Designate |
| C9LOCK | Range lock on |
| C9BRKT | Break track |
| C9WRIT | Record data |
| <u>OUT615</u> | PACKS words 6-15, alpha and beta errors |
| (INPUT) ICANT | not presently assigned |
| (INPUT) ICABER | beta error (10) for scope |
| (INPUT) ICAA | alpha error (10) for scope |
| (OUTPUT) IERRSP | location to start storage of ten packed words for A scope. |
| <u>OUT1617</u> | PACK words 16, 17 |
| NRL(3) | three 4-bit BCD range digits for A SCOPE (lsb = 1 meter), one character right justified in 3 half words. |
| CMULSY | multi-A synch bit |
| NRG(4) | four 4-bit BCD range digits (lsb = 1K meters), one character right justified in four half words. |
| <u>OUT18</u> | PACK word 18 |
| NANGE | 4-bit BCD elevation all four characters in 4 bytes. |
| <u>OUT 19</u> | PACK word 19 |
| NANGE | 4-bit BCD aximuth for multi A scope. |
| <u>OUT20</u> | PACK word 20 |
| CATRK | track status coast status not used in PJL |

E. SUBROUTINES USED

| | |
|---|---|
| ANDH | see user memo #21 |
| RTBCD | range to BCD (4-bit) |
| ANGBCD | azimuth or elevation to BCD (4-bit) |
| IN1→IN5 | (CCUNPK) |
| OUT1→OUT5 | (CCPACK) |
| OUT615 | (CCPACK) |
| OUT1617 | (CCPACK) |
| OUT18→OUT20 | (CCPACK) |

F. CONSOL READ/WRITE CONTROL

The PJLRT process controls reading and writing from the consol via communication through the words CECCIP and CECCOP located in PJLO static file.

CEECIP - RESET CC for input from the 360

CECCOP - RESET CC for output to the 360

Various routines key on these words. One is considered condition on. Two is off. RTPACK (PJL version) will pass the CECCIP bit condition through the GIDQ for later use by PJLIO. PJLIO will determine if it should have received consol input from the RCI.

PJLIO will use CEECOP to schedule chained control words which will in turn actually transmit consol orders to the RCI.

Presently, the PJL process is alternating read write cycles. Timing problems might develop when the PJL processing is exercised in normal mode with 35 ms frame times. This will be a matter of experimentation. However, as long as enough dead time is available after the last TOT, say 1 milisec, there should be no problem.

| | |
|---|---|
| OUT1P2 | PACKS words 1 and 2 according to format |
| C6SGNL | signal strength |
| NRL(3) | three 4-bit BCD range digits (lsb = 1 meter), one character right justified in 3 half words |
| NRG(4) | four 4-bit BCD range digits (lsb = 1K meter), one character right justified in 4 half words |
| C2DATA | channel access - bits 17, 16, 15 in word 2 of consol output. This field was not on the original Sperry consol format specification. |
| OUT3 | PACKS word 3 according to format |
| IDUM | Diode fault not used by PJL |
| C7BUP | Beta limit (up) |
| C7BDWN | Beta limit (down) |
| NANGL | 4-bit BCD elevation all four characters in 4 bytes. |
| OUT4 | PACKS word 4 according to format |
| C8ART | Alpha limit right |
| C8ALFT | Alpha limit left |
| NANGL | 4 bit BCD azimuth all 4 characters in 4 bytes. |
| OUT5 | PACKS word 5 according to format |
| C9TKCH | active channel indicators |
| C9GAIN | manual gain operative |
| IRBL | Range coast, manual range track, auto range track functionally OR'ed together. |
| ICBL | coast, manual track, auto track functionally OR'ed together |
| C9WAIT | standby |

A. IDENTIFICATION SUBROUTINE

TITLE: PJL CONTROL CONSOL PACK INTERFACE

ID: CCPACK

PROGRAMMER: J. J. OGAN

B. FUNCTIONAL DESCRIPTION

This routine has ten entry points (OUT1, OUT2, OUT3, OUT4, OUT5, OUT615, OUT1617, OUT18, OUT19, OUT20) each of which will pack the number of consol words indicated by the entry name into the consol output common area CCOUT. (i.e., OUT615 packs words 6 through 15).

These packed words are complete with consol identity code (hex '8') and function code (binary '00').

C. INPUT

All inputs are via calling sequences to each of the individual entry points.

| | |
|---|---|
| C9AAQ | auto-acquisition |
| C9DESG | Designate |
| C9LOCK | Range lock on |
| C9BRKT | Break track |
| C9WRIT | Record data |
| <u>OUT615</u> | PACKS words 6-15, alpha and beta errors |
| (INPUT) ICANT | not presently assigned |
| (INPUT) ICABER | beta error (10) for scope |
| (INPUT) ICAA | alpha error (10) for scope |
| (OUTPUT) IERRSP | location to start storage of ten packed words for A scope. |
| <u>OUT1617</u> | PACK words 16, 17 |
| NRL(3) | three 4-bit BCD range digits for A SCOPE (lsb = 1 meter), one character right justified in 3 half words. |
| CMULSY | multi-A synch bit |
| NRG(4) | four 4-bit BCD range digits (lsb = 1K meters), one character right justified in four half words. |
| <u>OUT18</u> | PACK word 18 |
| NANGE | 4-bit BCD elevation all four characters in 4 bytes. |
| <u>OUT 19</u> | PACK word 19 |
| NANGE | 4-bit BCD aximuth for multi A scope. |
| <u>OUT20</u> | PACK word 20 |
| CATRK | track status coast status not used in PJL |

D. CALLING SEQUENCE

```
CALL OUT1P2 (C6RNGE, NRL, NRG)
CALL OUT3 (IDUM, C7BUP, C7BDWN, NANGL)
CALL OUT4 (C8ART, C8ALFT, NANGL)
CALL OUT5 (C9TKCH, C9GAIN, IRBL, ICBL, C9WAIT, C9AAQ, C9DESG,
           C9LOCK, C9BRKR, C9WRIT)
CALL OUT615 (ICANI(J), ICABER(J), ICAA(J), IERRSP(J))
CALL OUT1617 (NRL, CMULS9, NRG)
CALL OUT18(NANGL)
CALL OUT19 (NANGL)
CALL OUT20 (CATRK)
```

A. IDENTIFICATION SUBROUTINE

TITLE: PJL CONTROL CONSOL UNPACK

ID: CCUNPK

PROGRAMMER: J. J. OGAN

B. FUNCTIONAL DESCRIPTION

This routine has five entry points (IN1, IN2, IN3, IN4, and IN5) each of which will unpack one of the five 4-byte consol input words respectively. In addition, they access the consol input region in the PJL static file starting at C1FAGL and ending with C5DESG, and store consol parameters into the file as necessary.

It is on this level that device pecular parameters are checked for reasonable values. For example, the handwheel fields are return to zero perameters and only real data is available when the fields are non-zero.

It is easy to find these parameters by looking at the code. If a check is made for zero before storage of data into PJLR, then the parameter is a return to zero type.

A word of caution: C3TKCH is a break-before-make switch.

C. INPUT

CCIN 20 byte common control console input area

D. OUTPUT

C1CAGL ⟶ C5DESG, control console input section of PJLR

E. CALLING SEQUENCES

CALL IN1

CALL IN2

CALL IN3

CALL IN4

CALL IN5

A. IDENTIFICATION SUBROUTINE

TITLE: PJL MISSION CLOCK SERVICES

ID: CLOCK

PROGRAMMER: J. J. OGAN

B. FUNCTIONAL DESCRIPTION

Written in FORTRAN, this routine allows PJL users to get the current mission time in floating point seconds. For each call to the routine, CLOCK will access the BMDOS mission time register via the read floating point time macro (RDFLTM).

C. INPUT

No formal parameter inputs

D. OUTPUT

Eight bytes (double precision) floating point mission time are stored in user supplied address area.

E. CALLING SEQUENCE

CALL CLOCK (SYSTEM)

Where SYSTEM is pointer to address to store mission time.

F. MISCELLANEOUS

Timer resolution is 13.6 μ sec. The constraint being levied by the OS-360 13.6 μ sec interval timer.

A. IDENTIFICATION PROCESS INITIALIZER

TITLE: MASTER PJL CONTROL

ID: MASTER

PROGRAMMER: C. BLACKWELL

B. FUNCTIONAL DESCRIPTION

The basic version of master has been drastically modified for use in initiating the PJLIO/PJLRT process.

This new master 1) schedules the initial mission start time

2) calls PJLINT to initialize the RTI array in PJLO

3) calls PJLPMI to initialize the PJLRT post-mission array

4) starts the mission clocks

5) schedules recording by setting C5WRIT = 1 in PJLR

6) schedules PJLIO for real-time processing

7) schedules WLC for interrupted real-time processing.

A. IDENTIFICATION SUBROUTINE

TITLE: PJL IRIG INTERFACE (WSMR TIME)

ID: PJIRIG

PROGRAMMER: T. R. COTTLER

B. FUNCTIONAL DESCRIPTION

This routine accesses the IRIGBIN in PJLIO which contains the WSMR time message (8 bytes), unpacks the message, accesses the WSMR time area in PJLR, and stores the raw unpacked WSMR time.

Cells affected in PJLR are

WISEC - raw White Sands second

WIMS - raw White Sands milliseconds

WITME - raw White Sands tenths of seconds

C. INPUT

IRIGBIN maintained by PJLIO

D. OUTPUT

WISEC, WIMS, WITIME in PJLR.

E. CALLING SEQUENCE

CALL PJIRIG

A. IDENTIFICATION STATIC FILE

TITLE: PJL ORDERS FILE (STATIC)

ID: PJLO

PROGRAMMER: J. J. OGAN

B. FUNCTIONAL DESCRIPTION

PJLO is the machine implementation of the PJLRT ORDERS file as described in the SURC interface document with minor additions.

Briefly, PJLO is a description of a static load-time fixed area in core to be used as the primary interface for transfer of order information from the PJL process to BMDOS and ultimately the radar subsystems, PJL equipment, RONDO A, and consol.

With the exception of TIMOTR (which is REAL*8) all data in PJLO is INTEGER*4.

At load time, all information in PJLO is set to 0.

C. FILE DESCRIPTION

RTI this is a baisc real-time initialization array used by the PJLRT process to establish basic variable run-time parameters. During the initialization of the PJL process, MASTER will call PJLINT which will read run-time parameters from cards and store them into the RTI array.

XEOM This single 4-byte word is used by PJLRT to communicate to PJLIO the termination of the run.

= 0 continue with run
= 1 end run

BMDCP this array called BMD control parameter array is used to pass schedule and hang information

BMDCP(1) = not used

BMDCP(2), BMDCP(3) = floating point mission time in seconds for PJLIO process to reschedule itself for execution

BMDCP(4), BMDCP(5) = length of hang duration is floating point seconds

BMDCP(6), BMDCP(7), BMDCP(8), BMDCP(9) = spares

TIMOTR(1) = floating point (REAL*4), # of orders

TIMORT(2)
↓ = Up to 16 time-of-transmition (TOT's) in
TIMOTR(17) floating point mission time

VDPOUT (1,1) INTEGER*4, # of orders this cell is maintained by PJLRT and is actually used by RTPACK (PJL version)

VDPOUT = Up to 16 valid orders for the VDP

BSUOUT = Up to 16 valid orders to the BSU

PJLOUT (1,1) PJL equipment schedule information

= 0 no PJL
= 1 PJL data this cycle

this information is used by RTPACK and PJLIO

PJLOUT(2,1)
↓ = One PJL equipment order. Double subscripting
PJLOUT(15,1) has been used to allow for later expansion to multiple PJL orders during one frame.

RONDOT(1,1) = RONDO A equipment schedule information

= 0 no RONDO
= 1 RONDO data this cycle

this information is used by RTPACK and PJLIO

RONDOT(2,1)
↓ = One RONDO equipment order. Double subscripting
RONDOT(13,1) has been used to allow for later expansion to multiple PJL orders during one frame.

C6SGNL
↓ = One consol order
CMULSY

CEASCP
CESCER
CECCIP
CECCOP } = external function schedule words for the CC-EF external function consol word. CECCIP and CECCOP are of particular importance. CECCIP controls reset control consol for input from the 360/65. CECCOP controls reset control consol for output to the 360/65.
PJLRT makes his consol scheduling wished known to PJLIO through these words. =1 value for the external functions selects the option. =0 designates no selection.

A. IDENTIFICATION STATIC FILE

TITLE: PJL RETURNS FILE (STATIC)

ID: PJLR

PROGRAMMER: J. J. OGAN

B. FUNCTIONAL DESCRIPTION

PJLR is the machine implementation of the PJLRT returns file as described in the SURC interface document.

PJLR is a description of a static, load-time fixed area in core to be used as the primary interface for transfer of return information from the radar sub-systems, the PJL Equipment, RONDO A and the consol to the PJLRT process.

All information in PJLR is INTEGER*4 format.

At load time, all information in PJLR is set to 0.

C. FILE DESCRIPTION

VDPIN = Space for up to sixteen returns from the VDP (unpacked)

BSUIN = Space for up to sixteen returns from the BSU (unpacked)

PJLIN = Space for one report of 480 bytes from the PJL equipment (unpacked)

RONDOI = Space for one report of 16 bytes from the RONDO A remote radar (unpacked)

C1FA6L ↓ C5DES6 = Space for one complete unpacked consol return

PIRN6E ↓ PIID = Space for one unpacked PAS data return

WISER ↓ WITIME = Space for one range time return (unpacked)

NULLE = An extra 'sneak' word slipped in by PJLRT for no known reason.

DVCERR = An array to pass information concerning the various returns from the various subsystems and equipment. The following code convertion is followed:

= -1 devise error for some unspecified reason

= 0 no return this cycle from this equipment

= 1 valid data has been received, unpacked and stored in the appropriate regions in PJLR.

Device code slots have been assigned in the following manner:

DCCERR (1) = VDP
(2) = BSU
(3) = CONSOL
(4) = RONDO A
(5) = RONDO B (not used)
(6) = PJL Equipment
(7) = RDT (PAS) report
(8) = WSMR TIME
(9) - (13) SPARE

NULL3 - another sneaky word

A. IDENTIFICATION SUBROUTINE

TITLE: PJL EQUIPMENT ORDERS INTERFACE

ID: PJLORD

PROGRAMMER: J.J. Ogan

B. FUNCTIONAL DESCRIPTION

When called with proper address pointers, this routine will commence PJL parameter packing from the PJLO static file [starting address - PJLOUT (2,1)] and return as output four packed PJL equipment order words as specified in the SURC interface document.

C. INPUT/OUTPUT

Calling Sequence for PJLORD is

CALL PJLORD PJLOUT (2,1), IPACK

(INPUT) PJLOUT = F.W.A. of unpacked PJL equipment orders in PLJO file.

(OUTPUT) IPACK = F.W.A. of location to begin storage of 16 bytes packed PJL equipment orders.

D. MISCELLANEOUS comments

Bits 0-12 in each of the 4 order words to the PJL equipment are left justified in the data portion of the RCI commands instead of right justified as one might suppose from SURC's interface document.

EXAMPLE:

| | 31——26 | 25——8 | 7——0 |
|---|---|---|---|
| RCI WORD | ID/FUNCT. | DATA PORTION | 0—0 |
| | | PJL DATA LEFT JUSTIFIED IN THIS FIELD | |

Make sure starting input address is PJLOUT (2,1)

PJLOUT (1,1) is a key call to instruct PJLIO that PJL equipment orders are scheduled (=1) or not scheduled (=o).

A. IDENTIFICATION SUBROUTINE

TITLE: PJL PASSED DATA INTERFACE

ID: PJLPAS

PROGRAMMER: J. J. Ogan

B. FUNCTIONAL DESCRIPTION

When called with proper address pointers, this routine will unpack PAS parameters and store them into the PAS return area in PJLR. This area starts with the variable PIRNGE and ends with PIID.

C. INPUT/OUTPUT

Calling sequence for PJLPAS is

CALL PJLPAS (PASBIN, PIRNGE)

(INPUT) PASBIN = address pointer to packed PAS return as sorted by PJLIO.

(OUTPUT) PIRNGE - address pointer to starting location for storage of unpacked PAS parameters of range, azimuth, elevation, time tag, status and id.

D. MISCELLANEOUS

None

A. IDENTIFICATION SUBROUTINE

TITLE: PJL RONDO "A" Orders INTERFACE

ID: PJLROA

PROGRAMMER: J. J. Ogan

B. FUNCTIONAL DESCRIPTION

When called with proper address pointers, this routine will commence RONDO parameter packing from the PJLO static file [starting address RONDOT (2,1)] and return as output four packed RONDO A order words (16 bytes) as specified in the SURC interface document.

C. INPUT/OUTPUT

Calling sequence for PJLROA is

CALL PJLROA RONDOT (2,1), IPACK

(INPUT) RONDOT (2,1) = address pointer to location in which to start packing RONDO orders.

(OUTPUT) IPACK = location to store 16 bytes of packed RONDO order data.

D MISCELLANEOUS

Be careful that the first input address to start order access is RONDOT (2,1). RONDOT (1,1) is a key call to instruct PJLIO that RONDO is scheduled (=1) or not scheduled (=o).

A. IDENTIFICATION

TITLE: PJL RONDO "A" RETURNS INTERFACE

ID: PJLRRA

PROGRAMMER: J.J. Ogan

B. FUNCTIONAL DESCRIPTION

This routine is used to unpack a RONDO A return stored in the RONDOBIN in PJLIO and transfer its parameters to the RONDO A return area [starting at RONDOI (1,1)] in PJLR.

C. INPUT/OUTPUT

Calling sequence for PJLRRA is

CALL PJLRRA RONDOBIN, RONDOI (1,1)

(INPUT) RONDOBIN = address pointer to packed return (16 bytes) as sorted by PJLIO.

(OUTPUT) RONDOI = address pointer to starting location for parameter storage into the PJLR RONDO return area as specified in the SURC interface document.

D. MISCELLANEOUS

All unpacking is based on normal distribution in the RCI word with 6 high order bits of ID/function, 18 bits of raw data, and 8 bits of low order zero.

No attempt is made to store 1 or 0 in RONDOT (1,1) (3,1), (5,1), (10,1) as shown in the SURC interface document.

This has not affected the operation of PJLRT in anyway.

A. IDENTIFICATION SUBROUTINE

TITLE: PJL EQUIPMENT RETURNS INTERFACE

ID: PJLRTN

PROGRAMMER: J. J. Ogan

B. FUNCTIONAL DESCRIPTION

This routine will unpack a PJL equipment rturn stored in the PJLBIN array in PJLIO and transfer its parameter to the PJL return area [starting at PJLIN (1,1)] in PJLR.

C. INPUT/OUTPUT

Calling sequence for PJLRN is

CALL PJRTN (PJLBIN, PJLIN)

(INPUT) PJLBIN = address pointer to packed return (480 bytes) as sorted by PJLIO.

(OUTPUT) PJLIN = address pointer to array in which store to unpacked returns as specified in the SURC interface document.

D. MISCELLANEOUS

All 32 bits are utilized with PJL equipment returns.

| | 31——28 | 27——0 |
|---|---|---|
| RCI/PJL | ID = "F" | PJL DATA |

This is different from other returns where the high 6 bits and the low eight bits are thrown away.

A. IDENTIFICATION SUBROUTINE

TITLE: PJL TAPE READ ROUTINE

ID: PJTAPE

PROGRAMMER: J. J. Ogan

B. FUNCTIONAL DESCRIPTION

This routine was created for two reasons:

a) to relieve the PJL applications programmer burden on a VB tape using FORTRAN I/O access methods.

b) to functionally provide one PJL logical record of information for each individual call to the routine. This was necessary because after any one real-time run, the data link has a combination of PJL data and BMDOS performance monitor records.

PJTAPE uses READDK to effect its physical I/O. READDK returns one logical record for each call. This logical record still has the record descriptor word (RDW).

PJTAPE will look for the PJL identifier would "PJLR" in bytes 4-8 of the logical record. If found, PJTAPE will strip the RDW and pass the remaining part of the logical record to the caller.

If the record is not PJL the routine will loop until one is found or end-of-file is sensed.

PJTAPE also supports a rewind capability.

C. INPUT/OUTPUT

Calling sequence for PJTAPE is

CALL PJTAPE (LUN, MODE, IADD, IND, NW, BS)

(INPUT) LUN = logical unit # (not used)

(INPUT) MODE = mode to read tape (not used)

(OUTPUT) IADD = location to store stripped logical PJL record

(OUTPUT) IND = indication of results of read operation
=1 incomplete read (not used)
=2 read OK no errors
=3 end-of-file on tape
=4 read complete, error occurred, data in doubt

C. INPUT/OUTPUT CONT'D.

(OUTPUT) NW = number of bytes in this PJL logical record

(INPUT) BS = rewind requester
=0 no rewind
=1 rewind

D. MISCELLANEOUS

The first call to PJTAPE will result in a call to OPENDK which will open the DCB for the data link tape. OPENDK is an entry point along with READDK in the subroutine DLINKR.

A. IDENTIFICATION SUBROUTINE

TITLE: REAL TIME HAPDAR ORDER PACKER

ID: RTPACK

PROGRAMMER: J. J. OGAN

B. FUNCTIONAL DESCRIPTION

RTPACK was modified from the BASIC real-time HAPDAR software to accommodate PJL software needs. The new PJL RTPACK accesses the PJLO (radar portions) to get up to 16 basic raw, unpacked commands for HAPDAR.

Portions of interest are

TIMOTR - times of transmition, scheduled VDP trigger times

VDPOUT - scheduled commands for the VDP

BSUOUT - scheduled commands for the beam stearing unit

PJLRT stores into VDPOUT(1,1)aninteger # from 1 to 16 which is used by RTPACK to determine number of orders scheduled for any one frame.

For any one order to the sub-units of HAPDAR,the RTPACK routine will select one TOT, one set of VDP commands and one set of BSU commands as described in the SURC interface document. Casual inspection reveals this set of commands is a subset of all possible command parameters. As a rule, any parameters not used in the PJL processing will in the final packed orders be zeroes.

A major accommodation made in RTPACK for PJL processing is the capability to leave room in the orders buffer for RONDO and PJL orders if need.

RTPACK has scheduling information concerning RONDO/PJL in RONDOT(1,1) and PJLOUT (1,1) RTPACK scheduling MATRIX

| | CONDITIONS | | | |
|---|---|---|---|---|
| RONDOT(1,1) | 0 | 1 | 0 | 1 |
| PJLOUT(1,1) | 0 | 0 | 1 | 1 |
| Bytes reserved by RTPACK | 0 | 16 | 16 | 32 |

Depending upon which matrix condition is valid for any one cycle, RTPACK will leave room for none, either one, or both. On any one cycle a maximum of one order per RONDO/PJL device is valid. These orders are always located at the beginning of the order block before any orders for HAPDAR subunits are packed.

The space left for RONDO and PJL be filled by appropriate calls to PJLROA and PJLORD in the IO executive PJLIO. This PJL version of RTPACK will send no BSU diode test words.

The PJL version of RTPACK has been modified to support transmition of XS-3 character line messages to be transferred to the RONDO B port on the RCI (actually connected to the printer).

Three characters per cycle of PJLIO/RTPACK are sent to the RCI. For further details on the 1218 printer support see documentation on the routine TTYOUT and the general discussion of 1218 on-line printer capabilities in the general discussion of RCA/PJL interface software. For consol scheduling purposed RTPACK places read/write information in the CC-EF word inforporated in the last radar output order. Extensive experimentation, on both RCA and SURC's part, has shown that SURC should at least alternate scheduling of consol reads and writes.

The last major difference in the PJL RTPACK routine is that all accessed raw data is already in integer count form and need not to be run through floating to fix scaling conversions.

The basic philosophy of sequence of order words to the VDP and BSU has been kept consistent with that employed in the basic real-time HAPDAR software.

A. IDENTIFICATION

TITLE: REAL TIME HAPDAR UNPACK

ID: RTUNPK

PROGRAMMER: J. J. OGAN

B. FUNCTIONAL DESCRIPTION

RTUNPK has been modified for PJL interface support in two major ways:

One, all data is stored into the radar return areas of PJLR namely VDPIN. This is a significant departure from the regular process of storing unpack data into dynamic SROQ's and TROQ's.

The second major difference is that all unpack data is stored into PJLR in a raw binary count form leaving the fix to float conversion to the PJL tactical software. (The reader will recall that integer count data was also passed to RTPACK). Integer interfaces are maintained in this manner to afford a more realistic dividing line between RCA interface support software and PJL tactical real-time software.

Track return or search return information is determined through access of the GIDQ in the same manner as the BASIC software only in this case the actual mode code associated with this order/return pair is stored in the GIDQ slots instead of using GID pointer addresses.

This special GIDQ is made by the ORDRMAKE logic in PJLIO.

A. IDENTIFICATION SUBROUTINE

TITLE: PJL TAPE SAVE DATA

ID: TAPESV

PROGRAMMER: C. W. ZIEGLER, JR.

B. FUNCTIONAL DESCRIPTION

To support PJL data link recording needs, the BASIC real-time WRITDL tape routine was modified in the following manner:

An additional entry point for PJL was added to WRITDL called TAPESV. This new entry supports a slightly different calling sequence.

Any call to this entry point will result in an overhead of 8 bytes of identification tagged on to each logical record. This will enable proper handling of PJL data by PJPM1, the reduction program.

These eight bytes of overhead are as follows:

| 1 | 2 | 3 | 4 |
|---|---|---|---|
| 'P | J | L | R' |
| ← ID → | | 0 | 0 |

'PJLR' will be in core as hexadecimal

$D7D1D3D9_{16}$

ID will be a binary number, 0-50, passed through the TAPESV calling sequence.

C. INPUT/OUTPUT

Calling Sequence for TAPESV

CALL TAPESV(IADD,NBYTE,IRC,ID)

(INPUT) IADD = address to start recording starting in a full word boundary

(INPUT) NBYTE = number of bytes to be recorded

(OUTPUT) IREC = indicator of recording status

IRC = 0 recording is OK

IRC = 1 buffer is full (no action taken, waiting for tape)

IRC = 2 record

IRC = 3 call made to TAPESV with record button off

(INPUT) ID = binary ID, 0-50, for purposes of PJL record distinction.

D. MISCELLANEOUS

Calls to TAPESV with the C5WRIT parameter in PJLR turned off will result in data transfer to recording buffers, however, no physical recording of data will be executed. Maximum record size supported is determined by the BUFMAX field in the operational version of IOBUF@. The BUFMAX is currently 16,304 bytes or 4,096 words.

Two different versions of TAPESV exist. One for interrupted real-time operation and one for real-time operation.

The interrupted real-time version stops the mission time and transfers control to O/S for the actual physical tape operations.

All block discriptor and record descriptor conversions follow those used with IBM VB logical records.

A. IDENTIFICATION SUBROUTINE

TITLE: TTY OUTPUT

ID: TTYOUT

PROGRAMMER: J. J. OGAN

B. FUNCTIONAL DESCRIPTION

It is desireable to be able to print during real-time messages on the 1218 printer. Making use of the fact that RONDO B port for the RCI has been connected to a portion of the 1218 memory, it is functionally very simple to get to the 1218 printer.

This routine makes up a standard XS-3 coded line message complete with proper RCI ID/function code for sending information to RONDO B port.

C. INPUT/OUTPUT

The calling sequence to TTYOUT is:

```
CALL TTYOUT(IMES,ILEN)
```

(INPUT) IMES = the address of the first word of the desired message to be displayed on the 1218 printer (4 characters/word; standard S/360 8-bit code)

(INPUT) ILEN = the number of characters in the message (max allowed is 255)

Output is to an array in core addressible from outside of TTYOUT. The data in the core array is a grouping of three characters (in XS-3 code) per 4 bytes of core with appropriate RONDO B ID and function codes.

For example: call TRYOUT with the message 'PJL' (in hex D7D1D3) and the routine would convert these characters to XS-3 code and eventually make the word D2A92600

Broken down it looks like this

| D | 2 | A | 9 | 2 | 6 | 0 | 0 |
|---|---|---|---|---|---|---|---|
| 1011 | 00 10 | 1010 | 1001 | 00 10 | 0110 | 0000 | 0000 |
| RONDO B ID | FUNC | P | J | | L → in XS-3 code | | |

TTYOUT makes the assumption that any one call to it constitutes one line of data for the 1218 printer. By agreement between Sperry and RCA, a $$$ in XS-3 code is to be sent to flag end-of-line. Therefore, TTYOUT will tag $$$ into the end of every message. Also for further separation a /// is added to the beginning of every message.

D. SYSTEM INTERACTION

RTPACK is responsible for taking one full word out of the TTY output array (i.e., transmition of 3 characters/cycle) . Since the TTYBUF is initially filled with timeouts('20000000'), RTPACK will pack time outs. When and if data is dumped into TTYBUF by TTYOUT, RTPACK immediatly begins to pick up valid RONDO B transmition words. Once a word is fetched from TTYBUF, RTPACK stores a time out back into the cell. PJLIO, meanwhiles, has soft-wired CCW's that will pick up the one word RTPACK has managed to service with the most recent information for the 1218.

E. DATA CAPACITIES/RATES

TTYOUT can support a 1500 character burst backlog before it will loop back on itself and destroy the oldest data not transferred.

Presently, the PJL process is set up to transfer 3 characters/cycle so a 1500 burst backlog would take 500 cycles to transfer over to the 1218 computer.

One additional constraint to keep in mind is that the 1218 can probably only support 2 lines/sec maximum before data messages start getting scrambled. This implies that with a 100 millisecond frame (for example) the message lengths should be at least 15 characters (remember 6 characters of overhead; /// and $$$).

For basic HAPDAR software with 20 millisecond frame times, 75 character messages would be advised.

The whole point of this being -- watch the burst data rate (give the TTYOUT /RTPACK team time to catch up) and avoid message scramble with appropriate length messages.

F. MISCELLANEOUS

TTYOUT will support all length messages from 1 to 255 characters. When # characters is not module 3, the routine will pad to the right blanks to make a module 3 count. A message of > 255 characters will have only the first 255 sent.

A. IDENTIFICATION IRT CONTROL EXECUTIVE

TITLE: PJL CONTROL EXECUTIVE FOR IRT

ID: WLC (PJL VERSION)

PROGRAMMER: J. J. OGAN

B. FUNCTIONAL DESCRIPTION

The basic WLC has been modified significantly to control the PJL processing in IRT. The SURC software was ideally set up for IRT operation with PJLSIM and PJLRT working as independent units sharing the basic common areas known as PJLO and PJLR. WLC is a very dumb executive that

(1) Calls PJLRT process

(2) Stops clocks

(3) Calls PJLSIM

(4) Calls PJLCON which simulates consol input via cards

(5) Calls PJLPRT to enable display of results from previous real-time cycle

(6) Starts up clock

(7) Reschedules itself based upon the mission time passed through BMDCP (2), (3).

The interrupted real-time processing allows SURC to check out tactical modifications to PJLRT without wasting valuable computer time and equipment time and have the I/O facilities of O/S available for extensive printer display.

In reality, this IRT process has been run very little; the primary benefit being derived during the initial attempts to run PJLRT under BMDOS.

## APPENDIX 4

## DISTRIBUTION LIST FOR THIS REPORT

| ADDRESSEE | NO. OF COPIES |
|---|---|
| Director<br>U.S. Army Advanced Ballistic Missile Defense Agency,<br>Huntsville Office<br>P.O. Box 1500,<br>Huntsville, AL 35807 | |
| Attn: RDMH-P | 10 |
| Attn: RDMH-S | 1 |
| Director<br>U.S. Army Advanced Ballistic Missile Defense Agency<br>Commonwealth Building<br>1300 Wilson Boulevard<br>Arlington, VA 22209 | |
| Attn: RDMD-AO | 1 |
| Attn: RDMD-TD | 1 |
| Commander<br>U.S. Army Safeguard System Command<br>P.O. Box 1500<br>Huntsville, AL 35807 | |
| Attn: SSC-CS | 1 |
| Defense Documentation Center<br>Cameron Station<br>Alexandria, VA 22314 | 2 |
| Commanding General<br>U.S. Army Safeguard System Office<br>Commonwealth Building<br>Attn: Dr. R. Merwin<br>1320 Wilson Boulevard<br>Arlington, VA 22209 | 1 |
| The MITRE Corporation<br>Attn: Mr. M. Barov<br>3222 S. Memorial Parkway<br>Holiday Office Center, Suite 115<br>Huntsville, AL 35801 | 1 |
| IBM Corporation<br>Attn: Mr. J. Gilbert<br>18100 Frederick Pike<br>Gaithersburg, Maryland 20706 | 1 |

| ADDRESSEE | NO. OF COPIES |
|---|---|
| System Development Corporation<br>Attn: Mr. W. Arnold<br>4810 Bradford Boulevard NW<br>Huntsville, AL 35807 | 2 |
| Sperry Rand Corporation<br>Sperry Gyroscope Division<br>Attn: Mr. P. Liebman<br>Great Neck, NY 11020 | 1 |
| Syracuse University Research Corp<br>Special Projects Laboratory<br>Attn: Mr. E. Nordell<br>8888 Dyer Street, Suite 314<br>El Paso, Texas 79904 | 2 |
| Syracuse University Research Corp<br>Attn: Mr. Z. Zenon<br>Merrill Lane, University Heights<br>Syracuse, NY 13210 | 1 |
| Stanford Research Institute<br>Attn: Mr. T. Brandon<br>333 Ravenswood Avenue<br>Menlo Park, CA 94025 | 1 |
| General Research Corporation<br>Attn: Dr. C. Perkins<br>P.O. Box 3587<br>Santa Barbara, Calif. 93105 | 1 |

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

# REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER
CDRL No. B002

2. GOVT ACCESSION NO.
-

3. RECIPIENT'S CATALOG NUMBER
-

4. TITLE (and Subtitle)
Data Processing/HAPDAR Field Test Facility
Final Technical Report for Period
June - September, 1973

5. TYPE OF REPORT & PERIOD COVERED
Final, 6/73 to 9/73

6. PERFORMING ORG. REPORT NUMBER
-

7. AUTHOR(s)
L. J. Campbell
T. R. Royer
J. E. Copestakes

8. CONTRACT OR GRANT NUMBER(s)
DAHC60-73-0035

9. PERFORMING ORGANIZATION NAME AND ADDRESS
RCA
Missile & Surface Radar Division
Moorestown, N.J. 08057

10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS

11. CONTROLLING OFFICE NAME AND ADDRESS
U.S. Army Safeguard Systems Command
Code SSC-C, P.O. Box 1500
Huntsville, Alabama 35807

12. REPORT DATE
September, 1973

13. NUMBER OF PAGES
80

14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)
-

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; Test and Evaluation; 21 Aug. '72. Requests by other agencies should be addressed to the Director, U.S. Army Advanced Ballistic Missile Defense Agency, ATTN: RDMH-P, Box 1500, Huntsville, AL 35807

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

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)
Phased Array Radars
Ballistic Missile Defense
ABM

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

This report describes use of the Data Processing/HAPDAR Field Test Facility as an adjunct to Passive Jammer Locator experiments by Syracuse University Research Corporation (SURC) at White Sands Missile Range, N.M. The Data Processing/HAPDAR Facility consists of a specially programmed IBM 360/65 Computer operating with the Sperry Rand HAPDAR (Hardpoint Demonstration Array Radar) at WSMR.

(Cont'd)

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

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

20. Abstract (Cont'd)

The Passive Jammer Locator experiments were performed by an ABMDA team with representatives from SURC, Sperry, and RCA. The Data Processing/ HAPDAR Facility was equipped with special programming and was linked to other WSMR sensors. The HAPDAR antenna and a remote RONDO facility were used to seek test jammers and to record jamming signals.