U N C L A S S I F I E D

Personnel Resource Information Management

(PRIM)

<u>System Integration Test Plan</u>
(SITP-C20-5)

by

PRIM Project Team

ODP/A/SDD
OP/ID/ADRB

23 September 1983

U N C L A S S I F I E D

CONTENTS

CHAPTER PAGE

1. INTRODUCTION 1

1.1 PURPOSE 1
1.2 SCOPE 1
1.3 REFERENCES 2

2. PLAN 3

2.1 FUNCTIONS 3
2.2 METHODOLOGY 3
2.2.1 TEST TEAM 4
2.2.2 RESOURCE REQUIREMENTS 5
2.2.3 SYSTEM STRESS 9
2.2.3.1 RELIABILITY 9
2.2.3.2 SIZING 10
2.2.3.3 ACCURACY 12
2.2.3.4 TIMING 13
2.2.3.5 AVAILABILITY 14
2.2.3.6 BACKUP 14
2.2.3.7 RECOVERY 14
2.2.4 CONTROLS 15
2.2.4.1 SOFTWARE (DEVELOPMENT AND PRODUCTION) . . 15
2.2.4.2 SYSTEM SOFTWARE 15
2.2.4.3 HARDWARE 15
2.2.4.4 SECURITY CONTROLS 16
2.2.5 TEST DATA 17
2.2.6 CONSTRAINTS 19
2.2.7 REGRESSION TESTING 19
2.2.8 DATA RECORDING 19
2.2.9 EVALUATION 19
2.2.10 TEST CASE MAINTENANCE 20

LIST OF TABLES

TABLE   PAGE

1. Documents, Publications and Reference Material 2

Chapter 1

INTRODUCTION

1.1 PURPOSE

The purpose of the System Integration Test Plan is to demonstrate through system testing that the system satisfies the requirements mapped in the Requirements Traceability Matrix (RTM) to the Detailed System Requirements Document (DSRD) and accurately reflects the design specified in the Detailed System Design Specifications (DSDS).

1.2 SCOPE

This document presents the PRIM System Integration Test Plan/tasks which will subject the programmers' products to a thorough set of tests neither designed nor executed by the programmers and run in as nearly live environment as possible with a minimum of simulation.

The intent of this test plan is to identify responsibilities; state the overall test environment, identify objectives, responsibilities, procedures, and tools to be used for each of the various System Integration Test phases of the test plan. The PRIM System is being designed in a phased approach; therefore, this System Integration Test Plan will be written to test Release 1 as well as the overall structure of the PRIM System. Thereafter, the System Integration Test Plan will be updated for each successive release. However, the testing philosophy established in Release 1 will continue to be utilized in each successive release.

Due to resource limitations, ODP/QAD will not be providing QA support or test plans in Release 1. Releases 2 through 5 are (TBR) regarding QAD participation.

The test plan consists of 2 Chapters:

Chapter 1 - "Introduction" presents the purpose and scope of the PRIM System Integration Test Plan and references applicable to the contents of the System Integration Test Plan.

Chapter 2 - "Plan" presents the major testing methodologies and defines testing criteria to be applied against the functional, performance, security, hardware, human engineering and interface requirements.

1.3 REFERENCES

The PRIM Team is utilizing a number of documents, publications and other reference material in writing the PRIM System Integration Test Plan. They are listed in Table 1.

TABLE 1

Documents, Publications and Reference Material

1. Data Requirements Document
2. Data Specifications Document
3. Detailed System Design Specifications
4. Detailed System Requirements Document
5. Interface Control Document
6. ODP Applications Documentation Standards
7. Requirements Traceability Matrix
8. System Development Plan

Chapter 2

PLAN

2.1 FUNCTIONS

PRIM is a centralized data base for use by the Personnel Officer, Career Management Officer, Office Director or Training Officer of a component[1] in direct support of the component's day-to-day personnel management activities. The overall objective of the test plan is to ensure that the system meets all of the requirements identified in the Detailed System Requirements Document and Detailed System Design Specifications as mapped in the Requirements Traceability Matrix and listed below.

a) Centralizing Official Data for Component Access (Release 1, 3 & 5)

b) Data Transfers Between Components (Release 1)

c) Component Data Manipulation (Releases 2 & 4)

d) Queries and Reports (Release 1)

e) Controlled Component Data Access (Release 1)

f) Other Requirements: Performance, Security, Hardware, Human Engineering, Interface, Legal

2.2 METHODOLOGY

The System Integration Test Plan is being written for Release 1 of PRIM; however, the testing philosophies established in Release 1 will be utilized in successive releases of PRIM. The PRIM System Integration Test Plan will ensure that program/system specifications were interpreted correctly; coding standards and conventions were followed; subject procedures/functions have been sufficiently tested by the programmers involved; previously prepared integration test packets

[1] The term component in the context of this document is defined as a separate entity in the Agency's organizational structure be it a directorate level, an office, a staff, a division, a service or a center.

covering all appropriate areas were successfully tested, user production documentation, where required for a procedure/function is complete and accurate; program/system documentation is complete and accurate for procedures/functions; and that the procedures/functions will perform as specified by the user prior to "official pass" to the PRIM Library for acceptance testing by the user.

2.2.1 TEST TEAM

The System Integration Test Team will consist of the two ODP and two OP analysts. Skills required will be systems analysis, system design, data base design, and project management. A detailed state-of-the-art knowledge of computer systems architecture and software engineering are mandatory. Skills will be required in interpreting the previously prepared integration test packets and in certifying that the manuals are complete and accurate. The analysts will be required to service "Requests for Change (RFC)" and to estimate impact on schedule, etc. All system analytical and design work is to be completed prior to System Integration testing. The PRIM Project Leader will keep management apprised on the status of System Integration testing.

The analysts are responsible for:

a) Re-examining the internal workings of the program structure - - testing of procedures, testing system functions, the HRS2 Interface coming in, security testing and data transfer between components. In-depth testing of system security is considered critical. System test cases are designed by analyzing the objectives and then formulated by analyzing the user documentation along with the program/system documentation.

b) Ensuring that procedures are set up to control the System Integration Test environment of the various test packets and the status of the test plan and tests at any given point in time.

c) Re-certifying that there is sufficient disk space to contain source code, ODP load modules, OP load modules, ODP test data, if applicable, and OP test data (which after acceptance will be used to test future PRIM releases of software).

d) Reviewing the established procedures and the required JCL for ODP to load their source code and test load modules back to the PERTEST test libraries, if applicable.

e) JCL executes system without errors or unnecessary operator intervention and file labels are correct. JCL - defined file capacities adequate, assuming moderate expansion.

f) Execution sequence (system flow) accurate and workable.

g) Re-testing the established procedures and the required JCL to transfer tested load modules to the PRIMTEST library, if applicable. During the System Integration Test phase the PRIMTEST library may contain only functions that have been "officially passed", i.e., certified by ODP; whereas the ODP PERTEST library may contain functional areas that have to be retested.

h) Creating and generating test data for System Integration tests. The test data will be established in such a manner that any test or group of tests may be executed at any given time.

NOTE: Only testing generalities have been addressed above; more in-depth testing criteria is covered beginning with Section 2.2.2. If a more in-depth knowledge is required of any area, it will be necessary to refer to the supporting documentation, i.e., Program Manual, Product Manual, User Documentation, etc.

2.2.2 RESOURCE REQUIREMENTS

a) Human

1. Production Division/ODP - Data Base Management support and DAC support during the System Integration testing phase.

2. Operations Division/ODP - Operating the computer equipment that the PRIM development system will utilize during System Integration Testing.

3. Engineering Division/ODP - System performance measurement and monitoring.

4. Information and Analysis Branch/OP - HRS2 RAMIS Report definition support.

5. Automated Data Resources Branch/OP - PRIM DBM and ADRB analysts to prepare selected HRS transactions to complete a PRIM test case, i.e. Interface Transaction for a "ZZ" TOA.

6. PRIM Users Group - Directorate Referents and Directorate ADP Control Officers represent various components during development. PRIM Users Group will formalize changes to current requirements if the need arises. Request for Change (RFC) will be controlled via Configuration Management Procedures.

7. Office of Communications

i) Domestic Networks Division (DND) within OC handles all PRIM domestic communications. DND is responsible for the installation and/or procurement of communications lines, modems, and cryptographic equipment in support of all PRIM peripheral equipment (terminals, remote printers, and remote job entry stations). DND interfaces directly with Engineering Division (ED/ODP) in response to communications requirements or problems encountered in the PRIM Project.

ii) Communications Security Division (CSD) within OC is responsible for TEMPEST testing/approval of all terminals and printers accessing PRIM. CSD interfaces directly with ED/ODP in response to communications requirements or problems encountered in the PRIM Project.

8. Office of the Inspector General (OIG) - The Information Systems Audit Division of the Audit Staff may examine PRIM procedures, records, reports, and test cases during the System Integration Test phase.

9. Office of Security - The Information Systems Security Group (ISSG) will play a consulting role in the evaluation of the PRIM System security feature during System Integration Test and Acceptance Testing.

b) Equipment

1. Hardware

The PRIM System will be accessed by the individual components through existing equipment. Peripherals in common use include video terminals (Delta Data 5000 or 7260 series) printers (TI Silent 700, Design 100). The components will be responsible for acquiring any additional equipment needed to access the PRIM System. Any equipment, specifically terminals or printers, must comply with standard Agency computer security regulations.

All System Integration Testing, Acceptance Testing and Production activity performed outside of the Headquarters building must utilize only equipment approved for classified use. (Refer to Resource Requirements, paragraph a) Human, above for further details regarding point of contact.)

2. Software	In House
GIMS	Yes
DBM System Software related to controlling read and/or write to PRIM and to data in PRIM	Yes
VM	Yes

c) Materials

Applicable Documents and Forms

Form 2968 - File Description

The index or table of contents noting the acronym, relative sequence, test description, etc. of each data element in the project file.

Form 3692 - Data Element Description

Used to define each data element within a given aplication of the system and its associated edit specifications. There is a copy of this form for each data element listed in the file description mentioned above.

Form 3692A - Input Requirements

Defines only those data elements that may be input (required or optional) to accomplish a particular task or transaction specified in the system design. An error message number is produced if a required data element is missing from the input.

Form 3692B - Program Specification (PRIM has added this spec to the front of the POL in lieu of using the form.)

Provides additional specifications necessary for the program to be written that are not previously recorded on the data element description form or the input requirements form.

Form 3692C - Message Descriptions

Used to document all messages generated by the project transactions in message number sequence, their program source number, and corrective action to be taken upon display of the message.

Form 3719 - Menu Format

A graphic layout of each menu to be used with a project transaction which is input on a CRT terminal. (All menus must be tested on both the Delta Data 5000 and 7260T series.)

Form 2278 - Program Narrative (PRIM has added this narrative to the front of the POL in lieu of using the Form.)

Provides a general description of the program relating to any project function.

Form 3715 - Verification Procedures

Document verification procedures that must be performed to determine successful program execution and the identification of any messages that pertain to the program.

Form 3417 - Report Specifications

Form 3417A - Report Data Elements

Used to supplement Form 3417 (above) in the event of insufficient space.

Form 3984 - Problem Report (Discrepancy Report) (To be used for Release 1 only)

The IOC Testers will record problems on this form. The OP PRIM Analysts will analyze and if appropriate forward copy to ODP PRIM Analysts. This documents and controls the correcting and retesting of the problem(s). When the problem has been corrected, the ODP PRIM Analysts will forward it (Orig plus 1) back to the OP PRIM Analysts who in turn will ask the user testers to retest the transaction.

PANVALET - Current list of PANVALET Specifications and Source - to include program/module, PANVALET numbers, date last updated and level numbers.

COMVAD

DATADOC/PROGRAMDOC

PRIM Users Manual

PRIM Program Manual

PRIM Production Manual

Applications Manual

Report-Writer Software

RAMIS Graphics

PDL

PRIM DBM Manual

Audit Trail - Containing sufficient information to permit a regular security review of system activity by Office of Security.

Contingency Plan - Including backup procedures in the event the main system is damaged or destroyed.

GIMS Procedures, Dictionaries, M and M/Dict, SYSMAN2 (except for passwords) - For Office of Personnel Analysts only.

NOTE: Other documentation which sets forth any special conditions and/or limitations impacting the system requirements will be provided by the PRIM Analysts.

Examples: Integration Test Review
System Test Packets

2.2.3 SYSTEM STRESS

System Stress tests will be measured in terms of reliability, sizing, accuracy, timing, availability, backup and recovery. In Release 1 the System will provide components terminal access and retrieval of official HRS2 data along with the capability for components to transfer access to employee data between components and to generate their own on-line queries or reports, offline reports and graphs utilizing "official data" only.

2.2.3.1 RELIABILITY

2.2.3.1.1 Consistent Results

Tests will be conducted to insure that the system provides the same results for the same successive requests for data providing neither the data nor the software has been changed by an update.

2.2.3.1.2 Consistent Output Response

Tests will be conducted to demonstrate that the requested response level with a data reporting activity of up to 200 users can be accommodated. Release 1 will have approximately 40-50 users; therefore, the full test with 200 users will be certified in a future release.

2.2.3.1.3 Consistent Update Response Level for Component Data

Component Data Files will not be available in Release 1; testing of the update response level will be certified beginning with Release 2.

2.2.3.2 SIZING

2.2.3.2.1 Users

The system must support up to 200 concurrent users actively querying, reporting or updating the Data Base. Release 1 will only certify approximately 40-50 concurrent users, therefore, the full test with 200 concurrent users will be certified in future releases.

2.2.3.2.2 Data Files

The system must support queries and reports requesting data from up to 30 different user data files utilizing a minimum of 70 edit/validation dictionaries. In release 1 the System Integration Test Analysts will certify 10 data files and a minimum of 65 dictionaries exclusive of data files required to satisfy security. (Security is addressed in Section 2.2.4)

	PRIM FILES	ACCESS
INDEX FILES	- A group of files will be provided for both Active data and Separated data indexing. The files will be used indirectly via XBRIDGING from the PRIM PERSIGN and PRIMSEP files. A file will be provided for each Selection element used when building a Component's Active and Separated data links. The Index files will be established once and updated through XBRIDGES. However, the Index files can be re-established with the PRINDX process.	DB Manager
STATEMENT FILE	- This file will permit the PRIM DBMGR to format high volume query statements which components will utilize via the LISTSTMT Procedure.	DB Manager
INTERFACE	- This file contains all current INTERFACE data passed to PRIM from HRS2. The file will be accessed through SEGACCESS segments spanning to an INTERINDX which spans to INTERFACE. An INPURGE date for each INTERFACE record will be stored in the INTERINDX file and INTERFACE records will be deleted when that date is reached. Data will be loaded into the PRIM INTERINDX file	Components via SEGACCESS

	via a BULK-LOAD statement using the disk created by the PRCAUPDT procedure creating an INPURGE date for each record. Data will be loaded into the PRIM INTERFACE file via an ITEM-DUMP statement using the same disk.	
INTERINDX	- This file is the link between the SEGACCESS files and the INTERFACE file. It contains a purge date (INPURGE) for each INTERFACE record. This date will be used when purging data from the INTERFACE file. The INTERINDX link (INTERLINK) will also be deleted.	Components
OFFICIAL DATA FILES	- Files containing official personnel data. These files are moved from HRS2 and include all Official data required from HRS2 by PRIM.	Components
PRIM PERSIGN	- This is considered one of the Official Data files in so far as how it is linked. The entire PERSIGN file will be moved only once (not daily) into PRIM and will be updated daily by the PRCHGLOAD procedure with data received from the HRS2 INTERFACE file via the PRCAUPDT procedure. The PERSIGN data is accessed by a component through the SEGACCESS file (SYSMAN2+A) containing that component's Active data links.	Components
PRIFN	- PRIM INTERFACE File Name on HRS2 and PRIM. This file will insure dumps and loads are done in the same order. This file will contain all files to be dumped with a flag on those files to be dumped only on weekends.	DB Manager
PRIMSEP	- This file will contain the PRIM PERSIGN record for a separated employee plus that employee's reason for separation and date of separation. This data is accessed by a component through the SEGACCESS file (SYSMAN2+S) containing that component's Separated data links.	Components
SEGACCESS	- Set of segmented files. One set of segments will be established for each level of component access requested.	DB Manager

	The file name will be the SYSMAN2 Signon concatentated to either "A" (Active), "S" (Separated) or "C" (Component). All access of Official data (Active and Separated) will be through these segmented files.	
SELECTION	- Set of segmented files. One segment will be established for each Signon ORG. This file will contain the Criteria used to select records authorized for the Signon ORG, Signon ORG text, ORGCODE links (SLORGLINK) and POSNR links (SLPOSLINK). The ORGCODE and POSNR data is accessed by a component through the SELECTION File.	Component-Individual Selection File only
SYSER	- In addition to its normal function, this file will contain the Delete-Data and Invert statements needed to create new Index files for PRIM PERSIGN and PRIMSEP. This file is used by the PRINDX procedure.	DB Manager
COMVAD	- Files containing edit and validation data used in HRS2. These files are moved from HRS2 to support the Official data files and in Release 2 will support the Component data files.	Components

2.2.3.3 ACCURACY

2.2.3.3.1 Transferred Data

Data transferred from HRS2 into PRIM must reflect the exact same data values in the PRIM System as was present in HRS2 at the time the data was extracted (i.e., COMVAD) or processed via the HRS2 INTERFACE Data List to PRIM Active and PRIMSEP.

2.2.3.3.2 Edits and Validations for Component Data

Component Data Files will not be available in Release 1, therefore, the edit and validations for these files will be certified in Releases 2 and 4.

2.2.3.4 TIMING

2.2.3.4.1 Direct Query

The System Integration Test Analysts will attempt to conduct stress testing to ensure that the system can complete 95 per cent of the direct queries in 4-7 seconds under normal operating conditions. However, a more realistic stress test will be conducted during the Acceptance Testing where the OP analysts will utilize all 40+ IOC component representatives to stress test at a given point in time. The direct query will utilize the DBMS Query Language to retrieve data from selected data lists using the DLID as the selection criteria for retrieving the information. (Normal operating conditions are as defined in the Detailed System Requirements Document, Section 3.2.4.1).

2.2.3.4.2 Complex Query

The System Integration Test Analysts will conduct stress testing to ensure that the system can complete a complex query (end-to-end search) at the rate of 2,000 records per minute under normal operating conditions. However, a more realistic stress test will be conducted during the Acceptance Testing where the OP analysts will utilize all 40+ IOC component representatives to stress test at a given point in time. In addition, the System Integration Test Analysts will review the daily Data Base Exception Report which identifies query statements executing longer than 3 minutes.

2.2.3.4.3 Hardcopy Reports

The System Integration Test Analysts will conduct stress testing to ensure that the system can print reports in 2 hours under normal operating conditions with a maximum overnight turnaround. Subsequent releases will include "scheduled" batch production reports.

2.2.3.4.4 Scheduling Updates to PRIM from HRS2

The extract of HRS2 data for updating the PRIM System must occur after the complete nightly update of the HRS2 Data Base. The System Integration Test Analysts serving as "PRIM Data Base Manager" will review the Data Base Stats and output from any nightly updates (Route *A) to ensure the updates to PRIM were made. In addition, the "Data Base Manager" compares selected HRS2 Data Lists and number of data items to the PRIM Data Lists and number of data items.

2.2.3.4.5 Updates to Component Data

Component Data Files will not be available in Release 1. Stress Testing of the update timings will be certified beginning with Release 2.

2.2.3.5 AVAILABILITY

The PRIM System must be available during regular working hours (0800-1800 hours), thus the System Integration Test analysts will conduct system availability tests on the data base during these hours. In addition, the system cannot operate in a degraded mode for more than one day and the degraded mode must only affect response time with no affect on the system's functional capabilities.

2.2.3.6 BACKUP

Tests will be conducted to insure that the PRIM System is backed up daily with audit trails of all updates to data and PRIM software. The daily backup must occur without user activation or intervention. The daily backup must be retained for a period of one-work week, a weekly backup must be kept until the next monthly backup; and each monthly backup is replaced by the next monthly backup.

2.2.3.7 RECOVERY

The PRIM System must virtually self recover in the event of hardware failure or the inadvertent destruction of data and/or files. The recovery process to restore the PRIM System or restart an interrupted activity must minimize the need for the user to re-enter data.

2.2.3.7.1 Restore

The capability must exist to restore the PRIM System as of the close of business the previous day, and also reprocess activity for the current day to minimize the need for the user to re-enter data. The restore and reprocess activity must take less than two hours. The System Integration Test Analysts serving as "PRIM Data Base Manager" will notify and coordinate with Production Division/ODP when the Data Base must be restored for either data or software problems.

2.2.4 CONTROLS

2.2.4.1 SOFTWARE (DEVELOPMENT AND PRODUCTION)

All PRIM Software received by the PRIM Team from the programmers will be thoroughly tested on the PRIMTEST Data Base and before it is moved to the Production System. The first phase of the System Integration Test Plan for Release 1 will consist of a System Test by the System Integration analysts who will perform an integrated test of all procedures/programs in a semi-live environment. This will verify that the system still performs in the same manner with all its functional areas completed as it did when only specific areas were complete. If an error is detected either on PRIMTEST or PRIM, a problem report and/or change control (request for change) will be used to officially report the error. PRIM software which requires change must be conducted first on ODP's PERTEST Data Base. After the error has been corrected and tested by the programmer, they will certify to the PRIM Project Leader that it is ready for retest on PRIMTEST. PRIM test data is established utilizing accepted numbering and dating conventions to ensure the capability of re-executing or referencing any given test or group of tests. After testing on PRIMTEST, the PRIM Project Leader will notify the System Integration Test Analyst that the revised version is ready for acceptance testing and certify such via Integration Test Review. Where appropriate, PANVALET specifications and source levels are required to be updated during all System Integration testing.

2.2.4.2 SYSTEM SOFTWARE

All of the Data Base Management System software related to controlling read and/or write access to the PRIM System and to data in the PRIM System is highly sensitive; therefore, tests will be conducted to ensure access and data is limited to only the individuals needing the information. Security locks and keys will be placed on selected GIMS Procedures and Dictionaries (i.e., M, M/Dict, SYSMAN2).

2.2.4.3 HARDWARE

The Agency's regulation on Computer Security, HR [redacted] controls the hardware security for the PRIM System. All System Integration Testing, Acceptance Testing and Production activity performed outside of Headquarters Building must utilize only equipment approved for classified use. System Integration Test Analysts will monitor that access to the system will not be conducted on equipment not approved for classified use, i.e., graphics printer.

STATINTL

2.2.4.4 SECURITY CONTROLS

2.2.4.4.1 Component Record Level Access

Tests will be conducted to ensure that a component is restricted to

a) access only the records for individuals assigned to that component or have a Career Service Designation of that component

b) access data passed electronically between components for individuals with an upcoming/proposed assignment to that component

c) enter, update and retrieve component data

d) query and report on organizational, position and employee data that applies to their assigned job requirements.

e) Directorate-level requires read access to all of the official organizational and position data as well as data for employees assigned to every office within the Directorate, or who have a grandfather Career Service Designation associated with that Directorate. Directorate level cannot have access to a component's input data.

2.2.4.4.2 Limited Data Access

Tests will be conducted to ensure limited read access to an employee's race code. Only designated personnel such as those responsible for preparing applicant and promotion data for Uniform Selection Review Reports can have read access to this code.

2.2.4.4.3 HRS2 Transferred Data

Tests will be conducted to ensure that no one can update the data transferred into the PRIM Data Base from the HRS2 Data Base. The only way the data can be changed is by the next transfer of data from the HRS2 Data Base. The Security Matrix prevents anyone from entering new data, deleting data or changing data values of the HRS2 data stored in PRIM.

2.2.4.4.4 PRIM Data Base Reporting Security

The System Integration Test Analysts will review the daily Security Violations Report against the PRIM Data Base which identifies violations of established read and/or write access control and the standard ACF2 violations report. Directorate Referents may be notified if abuses are detected.

2.2.5 TEST DATA

Release 1 will provide components terminal access and retrieval of "official" HRS2 data which is transferred to the PRIM Data Base each night. The data will be tested by the System Integration Test Analysts in a semi-live environment and will consist of re-executing the 13 PRIM procedures, querying and reporting from:

a) 65+ COMVAD Dictionaries

b) the HRS2 Official data files and

c) the new PRIM data files.

PRIM PROCEDURES/PROCESSES		Test Responsibility*
PRDDUMP	- Create ITEM-DUMP statements.	DBCC
PRDLOAD	- Create ITEM-LOAD statements.	DBCC
PRCAESTB	- Establish a component's access criteria in that component's SELECTION file.	DB Manager
PRCAUPDT	- Extract data from the HRS2 INTERFACE file which pertains to PRIM to be used by the PRCHGLOAD procedure. This is a BATCHGIM II program.	DBCC
PRCHGLOAD	- Uses disk from PRCAUPDT to make changes to PRIM PERSIGN and related Index files, PRIMSEP and related Index files, INTERINDX and load INTERFACE record into the PRIM INTERFACE file.	DBCC
PRCATRSF	- Allows a component to create an access to an employee's record for another component or remove an access to a record created for one component by another component.	Components

PRNTEPRG	- Review component access segmented file and purge any records with PRGDTE LE the System date.	DBCC
PRINTPRG	- Review the PRIM INTERINDX file for INTERLINKS with INPURGE LE the System date. Delete INTERLINK as well as the INTERFACE record related to it.	DBCC
PRACCUPDT	- Allows the user the freedom to change a PRGDTE and add SSNORs to their component data file.	Components
PRCALINK	- Uses SELECTION Criteria data from the SELECTION segment for Signon ORG to trigger the appropriate Index file(s) to build the daily link for that ORG. Done once a day - at the time of the first Signon for a particular ORG.	Components
PRINDX	- This process extracts data from PRIM PERSIGN and/or PRIMSEP to re-establish the Index files.	DBCC DB Manager
PRCRTLINK	- Uses SLORGLINK in the SELECTION segments to build new links to the ORGCODE file for each segment and SLPOSLINK in the SELECTION segments to build new links to the POSNR file for each segment. This procedure is executed on weekends by DBCC.	DBCC
SEARCH	- This procedure will be used by the components to query the NAME and HPOSNR files.	Components
LISTSTMT	- Allows a component to access per-formatted statements which contain selected full file data (QUAL, LREQID, ORGCODE, etc.) as well as high volume query statements, thus eliminating the need for a component to key the various data elements to execute their request.	DB Manager

*Indicates test responsibility for Acceptance Testing.

The testing will be conducted daily by the System Integration Test Team. In addition, the System Integration Test Team will pass access to an employee's official data from one component to another for individuals pending an upcoming assignment to a specific component.

2.2.6 CONSTRAINTS

The initial timing requirement for transferring data to PRIM in Release 1 is geared to the update process of data in the HRS2 Data Base. The extract of HRS2 data for updating the PRIM System nightly must occur after the complete nightly update of the HRS2 Data Base. On occasion, due to a timing problem with the HRS2 job stream PRIM may not get updated; however, this will not seriously impact the users. A message appears at Signon time of the last date of update to the Data Base.

2.2.7 REGRESSION TESTING

All PRIM test cases will be documented using preplanned test formats for each and every test with accepted numbering and dating standards to ensure the capability of re-executing any given test or group of tests. The System Integration Test Analysts test top down, where applicable; therefore, if an error is detected and fixed, the testers ensure that the change made will not adversely affect previously tested code.

2.2.8 DATA RECORDING

There is a test packet for every test case which is made from specifications. The System Integration Test Analysts will review the results of all tests executed against these test packets. If an error is detected and corrected, the originator of the Problem Report must certify to the System Integration Test Analysts that he/she has retested the applicable procedure and along with the retest date. Data Base Statistics between HRS2 and PRIM Data Lists will be analyzed daily and maintained for at least 30 days after PRIM goes into Production.

2.2.9 EVALUATION

Release 1 of PRIM is basically transferring "official" data from the HRS2 Data Base to PRIM via extracts; therefore, testing of range of data values used, combination of input types used, etc. are not applicable to PRIM in Release 1. In addition, since PRIM is due to be updated nightly, the System Integration Test Analysts will have to create test data for manipulation on a very "limited basis". Testing evaluation is basically one of evaluating security matrices, verifying the HRS2 INTERFACE coming in is applied properly and passing access to an employee's official data from one component to another.

2.2.10 TEST CASE MAINTENANCE

The System Integration Test Analysts will maintain and file the results of all major or necessary tests during the System Integration Test phase or until Audit Staff has completed their audit of the system.

U N C L A S S I F I E D

Personnel Resource Information Management

(PRIM)

<u>Acceptance</u> <u>Test</u> <u>Plan</u>
(ATP-C20-5)

by

PRIM Project Team

ODP/A/SDD
OP/ID/ADRB

23 September 1983

U N C L A S S I F I E D

CONTENTS

CHAPTER PAGE

1. INTRODUCTION 1

1.1 PURPOSE 1
1.2 SCOPE 1
1.3 REFERENCES 2

2. PLAN 3

2.1 FUNCTIONS 3
2.2 METHODOLOGY 3
2.2.1 TEST TEAM 4
2.2.2 RESOURCE REQUIREMENTS 6
2.2.3 SYSTEM STRESS 10
2.2.3.1 RELIABILITY 10
2.2.3.2 SIZING 11
2.2.3.3 ACCURACY 13
2.2.3.4 TIMING 14
2.2.3.5 AVAILABILITY 15
2.2.3.6 BACKUP 15
2.2.3.7 RECOVERY 15
2.2.4 CONTROLS 16
2.2.4.1 SOFTWARE (DEVELOPMENT AND PRODUCTION) . . 16
2.2.4.2 SYSTEM SOFTWARE 16
2.2.4.3 HARDWARE 16
2.2.4.4 SECURITY CONTROLS 17
2.2.5 TEST DATA 18
2.2.6 CONSTRAINTS 20
2.2.7 REGRESSION TESTING 20
2.2.8 DATA RECORDING 20
2.2.9 EVALUATION 20
2.2.10 TEST CASE MAINTENANCE 21

LIST OF TABLES

Chapter 1

INTRODUCTION

1.1 PURPOSE

The purpose of the Acceptance Test Plan is to demonstrate that the system satisfies the requirements as traced in the Requirements Traceability Matrix to the functional requirements. In addition, this document will address Quality Assurance test plans and procedures which will be conducted by Office of Personnel/PRIM Analysts from Automated Data Resources Branch. ADRB is officially charged with QA support for all OP/ADP applications and will be conducting an independent audit for verification and validation of the PRIM System prior to recommending acceptance of the system. Due to the complexity of the PRIM System, it is being designed and tested in a phased approach. This document will further provide for a mutual understanding by all parties concerned of the Acceptance Test criteria to be conducted during the development of the PRIM System.

1.2 SCOPE

This document presents the PRIM System Acceptance Test Plan/tasks which will control the PRIM test environment during development.

The PRIM System is being designed in a phased approach, therefore, the initial test plan will be written for Release 1 of PRIM. Thereafter the test plan will be updated for each successive release. However, the testing philosophy established in Release 1 will continue to be utilized in each successive release.

Due to resource limitations, ODP/QAD will not be providing QA support or test plans in Release 1. Releases 2 through 5 are (TBR) regarding QAD participation.

The test plan consists of 2 Chapters:

Chapter 1 - 'Introduction' presents the purpose and scope of the PRIM Acceptance Test Plan and references applicable to the contents of the Acceptance Test Plan.

Chapter 2 - 'Plan' presents the major testing methodologies and defines testing criteria to be applied against the functional, performance, security, hardware, human engineering and interface requirements.

TABLE PAGE

1. Documents, Publications and Reference Material 2

1.3 REFERENCES

The OP Analysts from the PRIM Team are utilizing a number of documents, publications and other reference material in writing the PRIM Acceptance Test Plan. They are listed in Table 1.

TABLE 1

Documents, Publications and Reference Material

1. Applications Manual
2. Data Requirements Document
3. Data Specifications Document
4. Detailed System Design Specifications
5. Detailed System Requirements Document
6. Interface Control Document
7. ODP Applications Documentation Standards
8. Production Manual
9. Program Manual
10. Requirements Traceability Matrix
11. System Development Plan
12. System Integration Test Plan
13. User Manual

Chapter 2

PLAN

2.1 FUNCTIONS

PRIM is a centralized data base for use by the Personnel Officer, Career Management Officer, Office Director or Training Officer of a component in direct support of the component's day-to-day personnel management activities. The overall objective of the test plan is to ensure that the system meets all of the requirements identified in the Detailed System Requirements Document and Detailed System Design Specifications as mapped in the Requirements Traceability Matrix and listed below.

a) Centralizing Official Data for Component Access (Release 1, 3 & 5)

b) Data Transfers Between Components (Release 1)

c) Component Data Manipulation (Releases 2 & 4)

d) Queries and Reports (Release 1)

e) Controlled Component Data Access (Release 1)

f) Other Requirements: Performance, Security, Hardware, Human Engineering, Interface, Legal

2.2 METHODOLOGY

The initial Acceptance Test Plan is being written for Release 1 of PRIM; however, the testing philosophies established in Release 1 will be utilized in successive releases of PRIM.

The test plan has been divided into 2 distinct areas - System and Acceptance. This is to aid in the detection of errors or deficiencies at the earliest point in time. Test packets are to be used; they are

----

The term component in the context of this document is defined as a separate entity in the Agency's organizational structure be it a directorate level, an office, a staff, a division, a service, or a center.

specifically designed to control the testing. There is a packet for every test case which is made from specifications received from the designers. The test packets are identified by a cover sheet with a unique test case number and include procedures which guide the testers in performing their tests. The OP/PRIM analysts will review the results of all tests executed against the system test packets. The PRIM test library will only contain functions that have passed all of ODP's tests-certified by official pass. In addition, ODP is responsible for providing OP the tools which they may use in establishing their system test data base. PRIM's test data will be established in such a manner that any test or group of tests may be executed at any given time. However, to achieve continuity and effective working relationships with the ODP analysts, the testers will, to the fullest extent possible, execute project testing in the same developmental sequence followed by the ODP analysts. The first phase of the Acceptance Test Plan for Release 1 will consist of a system test by the OP/PRIM Analysts who will perform an integrated test of all procedures/programs in a semi-live environment. This will verify that the system still performs in the same manner with all its functional areas completed as it did when only specific areas were complete. An additional benefit from this phase will be the training and familiarization of the system to the PRIM Data Base Manager and the IOC users.

Upon successful completion of the first phase, the second phase of acceptance test for Release 1 will begin. This will consist of successful testing of data base security (security matrices, integrity, access, etc.) site testing, data transfers, performance, hardware, human engineering, reporting, and massaging interface data from the HRS2 Data Base. Testing will consist of processing transactions/data until test specifications are met and all documentation required for production and maintenance of the system are complete and accurate.

The ODP PRIM Analysts will be notified of any system error or deficiency to ensure that appropriate action is taken. The OP/PRIM analysts will maintain and file the results of all major or necessary tests during the acceptance phase or until Audit Staff has completed their audit of the system.

2.2.1 TEST TEAM

The Acceptance Test Team consists of the two OP/PRIM Analysts and employees from 10 offices who have been selected to participate in the Initial Operating Capability (IOC) of PRIM. The senior OP PRIM Analyst will provide the communications link between ODP and the IOC participants, Directorate Referents, the PRIM User Group and the Office of Personnel. She will also keep management apprised on the status of the testing effort via weekly reports or official briefings.

The OP PRIM Analysts are responsible for:

a) Assistance and guidance in writing system test specifications.

b) Assistance and guidance in writing acceptance test specifications based on the re-execution of selected segments of all of the system test(s).

c) Assistance and guidance in creating test data.

d) Assistance and guidance in scheduling test resources.

e) Assistance and guidance in sequencing and executing tests.

f) Assistance and guidance in analyzing test results.

g) Assistance and guidance in documenting all test results with accepted numbering and dating standards to ensure the capability of re-executing or referencing any given test or group of tests.

h) Communicating both positive and negative test results to the ODP PRIM Analysts and following up on remedial action necessary to correct deficiencies or errors with subsequent retesting.

i) Ensuring that user documentation is complete and accurate as tests are executed.

j) Ensuring that all Audit Staff's written requirements are met.

k) Ensuring that all test objectives are met for the various data transfers. (Example: HRS2 Interface coming in and Data Transfers between components.)

l) Security controls (production, reports, users, etc).

m) Privacy Act compliances.

n) Ensuring official pass of all documentation/access to all files/data lists/complete testing capabilities, etc. (See Section)

<u>The IOC Participants are responsible for</u>:

a) Writing acceptance test specifications based on the re-execution of selected segments or all of the system test(s).

b) Creating test data in terms of transaction (menu) processing.

c) Scheduling test resources.

d) Sequencing and executing tests.

e) Analyzing test results.

f) Documenting all test results using preplanned test formats for each and every test with accepted numbering and dating standards to ensure the capability of re-executing or referencing any given test or group of tests.

g) Communicating both positive and negative test results to the OP/PRIM Analysts and following up on remedial action necessary to correct deficiencies or errors with subsequent retesting.

h) Ensuring that user documentation is complete and accurate as tests are executed. Provide monthly status report to OP/PRIM Analysts.

i) Complying with all Audit Staff's written requirements.

j) Ensuring that all test objectives are met.

Skills Required - The PRIM testers (IOC Components) must be knowledgable in GIMS, VM, RAMIS and the batch process, to input, update and retrieve information from the PRIM System. In addition, the IOC components are committed to assisting in the training of other careerists in their Directorate who will be participating in future releases of PRIM.

2.2.2 RESOURCE REQUIREMENTS

a) Human

1. Production Division/ODP - Data Base Management support and DAC support during the development testing phase

2. Operations Division/ODP - Operating the computer equipment that the PRIM development system will utilize during Acceptance Testing.

3. Engineering Division/ODP - System performance measurement and monitoring.

4. Information and Analysis Branch/OP - HRS2 RAMIS Report definition support.

5. Automated Data Resources Branch/OP - PRIM DBM

6. PRIM Users Group - Directorate Referents and Directorate ADP Control Officers represent various components during development. PRIM Users Group will formalize changes to current requirements if the need arises. Request For Change (RFC) will be controlled via Configuration Management Procedures.

7. Office of Communications

i) Domestic Networks Division (DND) within OC handles all PRIM domestic communications. DND is responsible for the installation and/or procurement of communications lines, modems, and cryptographic equipment in support of all PRIM peripheral equipment (terminals, remote printers, and remote job entry stations). DND interfaces directly with Engineering Division (ED/ODP) in response to communications requirements or problems encountered in the PRIM Project.

ii) Communications Security Division (CSD) within OC is responsible for TEMPEST testing/approval of all terminals and printers accessing PRIM. CSD interfaces directly with ED/ODP in response to communications requirements or problems encountered in the PRIM Project.

8. Office of the Inspector General (OIG) - The Information Systems Audit Division of the Audit Staff will examine PRIM procedures, records and reports and certify that their tests are complete prior to the PRIM analysts recommending that the system be implemented.

9. Office of Security - The Information Systems Security Group (ISSG) will play a consulting role in the evaluation of the PRIM System security feature during Acceptance Testing.

b) Equipment

1. Hardware

The PRIM System will be accessed by the individual components through existing equipment. Peripherals in common use include video terminals (Delta Data 5000 or 7260 series) printers (TI Silent 700, Design 100). The components will be responsible for acquiring any additional equipment needed to access the PRIM System. Any equipment, specifically terminals or printers, currently installed or newly acquired by the components, must comply with standard Agency computer security regulations.

All Acceptance Testing and Production activity performed outside of the Headquarters building must utilize only equipment approved for classified use. (Refer to Resource Requirements, paragraph a) Human, above for further details regarding point of contact.)

2.

Software	In House
GIMS	Yes
DBM System Software related to controlling read and/or write to PRIM and to data in PRIM	Yes
VM	Yes

c) Materials

Applicable Documents and Forms

Form 2968 - File Description

The index or table of contents noting the acronym, relative sequence, test description, etc. of each data element in the project file.

Form 3692 - Data Element Description

Used to define each data element within a given aplication of the system and its associated edit specifications. There is a copy of this form for each data element listed in the file description mentioned above.

Form 3692A - Input Requirements

Defines only those data elements that may be input (required or optional) to accomplish a particular task or transaction specified in the system design. An error message number is produced if a required data element is missing from the input.

Form 3692B - Program Specification (PRIM has added this spec to the front of the POL in lieu of using the form.)

Provides additional specifications necessary for the program to be written that are not previously recorded on the data element description form or the input requirements form.

Form 3692C - Message Descriptions

Used to document all messages generated by the project transactions in message number sequence, their program source number, and corrective action to be taken upon display of the message.

Form 3719 - Menu Format

A graphic layout of each menu to be used with a project transaction which is input on a CRT terminal. (All menus must be tested on both the Delta Data 5000 and 7260T series.)

Form 2278 - Program Narrative (PRIM has added this narrative to the front of the POL in lieu of using the Form.)

Provides a general description of the program relating to any project function.

Form 3715 - Verification Procedures

Document verification procedures that must be performed to determine successful program execution and the identification of any messages that pertain to the program.

Form 3417 - Report Specifications

Form 3417A - Report Data Elements

Used to supplement Form 3417 (above) in the event of insufficient space.

Form 3984 - Problem Report (Discrepancy Report) (This Form to be used for Release 1 only.)

The IOC Testers will record problems on this form. The OP PRIM Analysts will analyze and if appropriate forward copy to ODP PRIM Analysts. This documents and controls the correcting and retesting of the problem(s). When the problem has been corrected, the ODP PRIM Analysts will forward it (Orig plus 1) back to the OP PRIM Analysts who in turn will ask the user testers to retest the transaction.

PANVALET - Current list of PANVALET Specifications and Source - to include program/module, PANVALET numbers, date last updated and level numbers.

COMVAD

DATADOC/PROGRAMDOC

PRIM Users Manual

PRIM Program Manual

PRIM Production Manual

Applications Manual

Report-Writer Software

RAMIS Graphics

PDL

PRIM DBM Manual

Audit Trail - Containing sufficient information to permit a regular security review of system activity by Office of Security.

Contingency Plan - Including backup procedures in the event the main system is damaged or destroyed.

GIMS Procedures, Dictionaries, M and M/Dict, SYSMAN2 (except for passwords) - For Office of Personnel Analysts only.

NOTE: Other documentation which sets forth any special conditions and/or limitations impacting the system requirements will be provided by the OP PRIM Analysts. Copies (some with examples and comments) of additional forms to be used in system testing will be provided with the Acceptance Test Procedures Package.

Examples: Integration Test Review
System Test Packets

2.2.3 SYSTEM STRESS

System Stress tests will be measured in terms of reliability, sizing, accuracy, timing, availability, backup and recovery. In Release 1 the System will provide components terminal access and retrieval of official HRS2 data along with the capability for components to transfer access to employee data between components and to generate their own on-line queries or reports, offline reports and graphs utilizing "official data" only.

2.2.3.1 RELIABILITY

2.2.3.1.1 Consistent Results

Tests will be conducted to insure that the system provides the same results for the same successive requests for data providing neither the data nor the software has been changed by an update.

2.2.3.1.2 Consistent Output Response

Tests will be conducted to demonstrate that the requested response level with a data reporting activity of up to 200 users can be accommodated. Release 1 will have approximately 40-50 users; therefore, the full test with 200 users will be certified in a future release.

2.2.3.1.3 Consistent Update Response Level for Component Data

Component Data Files will not be available in Release 1; testing of the update response level will be certified beginning with Release 2.

2.2.3.2 SIZING

2.2.3.2.1 Users

The system must support up to 200 concurrent users actively querying, reporting or updating the Data Base. Release 1 will only certify approximately 40-50 concurrent users, therefore, the full test with 200 concurrent users will be certified in future releases.

2.2.3.2.2 Data Files

The system must support queries and reports requesting data from up to 30 different user data files utilizing a minimum of 70 edit/validation dictionaries. Release 1 will certify 10 data files and a minimum of 65 dictionaries exclusive of data files required to satisfy security. (Security is addressed in Section 2.2.4)

	PRIM FILES	ACCESS
INDEX FILES	- A group of files will be provided for both Active data and Separated data indexing. The files will be used indirectly via XBRIDGING from the PRIM PERSIGN and PRIMSEP files. A file will be provided for each Selection element used when building a Component's Active and Separated data links. The Index files will be established once and updated through XBRIDGES. However, the Index files can be re-established with the PRINDX procedure.	DB Manager
STATEMENT FILE	- This file will permit the PRIM DBMGR to format high volume query statements which components will utilize via the INDXSTMT Procedure.	DB Manager
INTERFACE	- This file contains all current INTERFACE data passed to PRIM from HRS2. The file will be accessed through SEGACCESS segments spanning to an INTERINDX which spans to INTERFACE. An INPURGE date for each INTERFACE record will be stored in the INTERINDX file and INTERFACE records will be deleted when that date is reached. Data will be loaded into the PRIM INTERINDX file via a BULK-LOAD statement using the	Components via SEGACCESS

	disk created by the PRCAUPDT procedure creating an INPURGE date for each record. Data will be loaded into the PRIM INTERFACE file via an ITEM-DUMP statement using the same disk.	
INTERINDX	- This file is the link between the SEGACCESS files and the INTERFACE file. It contains a purge date (INPURGE) for each INTERFACE record. This date will be used when purging data from the INTERFACE file. The INTERINDX link (INTERLINK) will also be deleted.	Components
OFFICIAL DATA FILES	- Files containing official personnel data. These files are moved from HRS2 and include all Official data required from HRS2 by PRIM.	Components
PRIM PERSIGN	- This is considered one of the Official Data Files in so far as how it is linked. The entire PERSIGN file will be moved only once (not daily) into PRIM and will be updated daily by the PRCHGLOAD procedure with data received from the HRS2 INTERFACE file via the PRCAUPDT procedure. The PERSIGN data is accessed by a component through the SEGACCESS file (SYSMAN2+A) containing that component's Active data links.	Components
PRIFN	- PRIM INTERFACE File Name on HRS2 and PRIM. This file will insure dumps and loads are done in the same order. This file will contain all files to be dumped with a flag on those files to be dumped only on weekends.	DB Manager
PRIMSEP	- This file will contain the PRIM PERSIGN record for a separated employee plus that employee's reason for separation and date of separation. This data is accessed by a component through the SEGACCESS file (SYSMAN2+S) containing that component's Separated data links.	Components
SEGACCESS	- Set of segmented files. One set of segments will be established for each level of component access requested. The file name will be the SYSMAN2 Signon concatentated to either "A"	DB Manager

	(Active), "S" (Separated) or "C" (Component). All access of Official data (Active and Separated) will be through these segmented files.	
SELECTION	- Set of segmented files. One segment will be established for each Signon ORG. This file will contain the Criteria used to select records authorized for the Signon ORG, Signon ORG text, ORGCODE links (SLORGLINK) and POSNR links (SLPOSLINK). The ORGCODE and POSNR data is accessed by a component through the SELECTION File.	Component-Individual Selection File only
SYSER	- In addition to its normal function, this file will contain the Delete-Data and Invert statements needed to create new Index files for PRIM PERSIGN and PRIMSEP. This file is used by the PRINDX procedure.	DB Manager
COMVAD	- Files containing edit and validation data used in HRS2. These files are moved from HRS2 to support the Official data files and in Release 2 will support the Component data files.	Components

2.2.3.3 ACCURACY

2.2.3.3.1 Transferred Data

Data transferred from HRS2 into PRIM must reflect the exact same data values in the PRIM System as was present in HRS2 at the time the data was extracted (i.e., COMVAD) or processed via the HRS2 INTERFACE Data List to PRIM Active and PRIMSEP.

2.2.3.3.2 Edits and Validations for Component Data

Component Data Files will not be available in Release 1, therefore, the edit and validations for these files will be certified in Releases 2 and 4.

2.2.3.4 TIMING

2.2.3.4.1 Direct Query

The OP PRIM Analysts will conduct stress testing utilizing IOC component representatives to ensure that the system can complete 95 per cent of the direct queries in 4-7 seconds under normal operating conditions. The direct query will utilize the DBMS Query Language to retrieve data from selected data lists using the DLID as the selection criteria for retrieving the information. (Normal operating conditions are as defined in the Detailed System Requirements Document, Section 3.2.4.1)

2.2.3.4.2 Complex Query

The OP PRIM Analysts will conduct stress testing utilizing IOC component representatives to ensure that the system can complete a complex query (end-to-end search) at the rate of 2,000 records per minute under normal operating conditions. In addition, they will review the daily Data Base Exception Report which identifies query statements executing longer than 3 minutes.

2.2.3.4.3 Hardcopy Reports

The OP PRIM Analysts will conduct stress testing utilizing IOC component representatives to ensure that the system can print reports in 2 hours under normal operating conditions with a maximum overnight turnaround. Subsequent releases will include "scheduled" batch production reports.

2.2.3.4.4 Scheduling Updates to PRIM from HRS2

The extract of HRS2 data for updating the PRIM System must occur after the complete nightly update of the HRS2 Data Base. The PRIM Data Base Manager will review the Data Base Statistics and the output from any nightly updates (ROUTE *A) to ensure that the updates to PRIM were made. In addition, the Data Base Manager compares selected HRS2 Data Lists and number of data items to the PRIM Data Lists and number of data items.

2.2.3.4.5 Updates to Component Data

Component Data Files will not be available in Release 1. Stress Testing of the update timings will be certified beginning with Release 2.

2.2.3.5 AVAILABILITY

The PRIM system must be available during regular working hours (0800-1800 hours), thus the analysts and customers will conduct system availability tests on both the development and production data base prior to recommending acceptance of the system. In addition, the system cannot operate in a degraded mode for more than one day and the degraded mode must only affect response time with no affect on the system's functional capabilities.

2.2.3.6 BACKUP

Tests will be conducted to insure that the PRIM System is backed up daily with audit trails of all updates to data and PRIM software. The daily backup must occur without user activation or intervention. The daily backup must be retained for a period of one-work week, a weekly backup must be kept until the next monthly backup; and each monthly backup is replaced by the next monthly backup.

2.2.3.7 RECOVERY

The PRIM System must virtually self recover in the event of hardware failure or the inadvertent destruction of data and/or files. The recovery process to restore the PRIM System or restart an interrupted activity must minimize the need for the user to re-enter data.

2.2.3.7.1 Restore

The capability must exist to restore the PRIM System as of the close of business the previous day, and also reprocess activity for the current day to minimize the need for the user to re-enter data. The restore and reprocess activity must take less than two hours. The PRIM Data Base Manager will notify and coordinate with Production Division/ODP when the Data Base must be restored for either data or software problems.

2.2.4 CONTROLS

2.2.4.1 SOFTWARE (DEVELOPMENT AND PRODUCTION)

All PRIM Software received by "official pass" from ODP will be thoroughly tested on the PRIMTEST Data Base and accepted by the user before it is moved to the Production System for semi-live environment tests. The first phase of the Acceptance Test Plan for Release 1 will consist of a System Test by the OP/PRIM Analysts who will perform an integrated test of all procedures/programs in a semi-live environment. This will verify that the system still performs in the same manner with all its functional areas completed as it did when only specific areas were complete. Once the data base has been loaded with the live production version, all procedures, dictionaries, etc. will again be certified prior to the users having access to the system. If an error is detected either on PRIMTEST or PRIM, a problem report and/or change control (request for change) will be used to officially report the error. PRIM software which requires change must be conducted first on ODP's PERTEST Data Base. After the error has been corrected and tested, ODP will certify via "official pass" that it is ready for retest on PRIMTEST. PRIM test data is established utilizing accepted numbering and dating conventions to ensure the capability of re-executing or referencing any given test or group of tests. After user testing on PRIMTEST, the Senior OP Analyst will request that it be moved to the Production System via a DAC PIR. A copy of the DAC PIR is to be forwarded to ADRB certifying that the software was moved. Where appropriate, PANVALET specifications and source levels are required as part of the "official pass" during all acceptance testing.

2.2.4.2 SYSTEM SOFTWARE

All of the Data Base Management System software related to controlling read and/or write access to the PRIM System and to data in the PRIM System is highly sensitive; therefore, tests will be conducted to ensure access and data is limited to only the individuals needing the information. Security locks and keys will be placed on selected GIMS Procedures and Dictionaries (i.e., M, M/Dict, SYSMAN2).

2.2.4.3 HARDWARE

The Agency's regulation on Computer Security, HR [redacted] controls the hardware security for the PRIM System. All acceptance testing and Production activity performed outside of Headquarters Building must utilize only equipment approved for classified use. IOC participants will be briefed on the use of Tempest Approved equipment. IOC participants will be referred to Office of Security and Office of Communications/Domestic Networks Division regarding Hardware Security questions.

STATINTL

2.2.4.4 SECURITY CONTROLS

2.2.4.4.1 Component Record Level Access

Tests will be conducted to ensure that a component is restricted to

a) access only the records for individuals assigned to that component or have a Career Service Designation of that component

b) access data passed electronically between components for individuals with an upcoming/proposed assignment to that component

c) enter, update and retrieve component data

d) query and report on organizational, position and employee data that applies to their assigned job requirements.

e) Directorate-level requires read access to all of the official organizational and position data as well as data for employees assigned to every office within the Directorate, or who have a grandfather Career Service Designation associated with that Directorate. Directorate level cannot have access to a component's input data.

2.2.4.4.2 Limited Data Access

Tests will be conducted to ensure limited read access to an employee's race code. Only designated personnel such as those responsible for preparing applicant and promotion data for Uniform Selection Review Reports can have read access to this code.

2.2.4.4.3 HRS2 Transferred Data

Tests will be conducted to ensure that no one can update the data transferred into the PRIM Data Base from the HRS2 Data Base. The only way the data can be changed is by the next transfer of data from the HRS2 Data Base. The Security Matrix prevents anyone from entering new data, deleting data or changing data values of the HRS2 data stored in PRIM.

2.2.4.4.4 PRIM Data Base Reporting Security

The OP Analysts will review the daily Security Violations Report against the PRIM Data Base which identifies violations of established read and/or write access control and the standard ACF2 violations report. Directorate Referents may be notified if abuses are detected.

2.2.5 TEST DATA

Release 1 will provide components terminal access and retrieval of "official" HRS2 data which is transferred to the PRIM Data Base each night. The data will be tested in a semi-live environment and will consist of re-executing the 13 PRIM procedures, querying and reporting from

a) 65+ COMVAD Dictionaries

b) the HRS2 Official data files and

c) the new PRIM data files.

PRIM PROCEDURES/PROCESSES		Test Responsibility
PRDDUMP	- Create ITEM-DUMP statements.	DBCC
PRDLOAD	- Create ITEM-LOAD statements.	DBCC
PRCAESTB	- Establish a component's access criteria in that component's SELECTION file.	DB Manager
PRCAUPDT	- Extract data from the HRS2 INTERFACE file which pertains to PRIM to be used by the PRCHGLOAD procedure. This is a BATCHGIM II program.	DBCC
PRCHGLOAD	- Uses disk from PRCAUPDT to make changes to PRIM PERSIGN and related Index files, PRIMSEP and related Index files, INTERINDX and load INTERFACE record into the PRIM INTERFACE file.	DBCC
PRCATRSF	- Allows a component to create an access to an employee's record for another component or remove an access to a record created for one component by another component.	Components

PRNTEPRG	- Review component access segmented file and purge any records with PRGDTE LE the System date.	DBCC
PRINTPRG	- Review the PRIM INTERINDX file for INTERLINKS with INPURGE LE the System date. Delete INTERLINK as well as the INTERFACE record related to it.	DBCC
PRACCUPDT	- Allows the user the freedom to change a PRGDTE and add SSNORs to their component data file.	Components
PRCALINK	- Uses SELECTION Criteria data from the SELECTION segment for Signon ORG to trigger the appropriate Index file(s) to build the daily link for that ORG. Done once a day - at the time of the first Signon for a particular ORG.	Components
PRINDX	- This process extracts data from PRIM PERSIGN and/or PRIMSEP to re-establish the Index files.	DBCC DB Manager
PRCRTLINK	- Uses SLORGLINK in the SELECTION segments to build new links to the ORGCODE file for each segment and SLPOSLINK in the SELECTION segments to build new links to the POSNR file for each segment. This procedure is executed on weekends by DBCC.	DBCC
SEARCH	- This procedure will be used by the components to query the NAME and HPOSNR files.	Components
LISTSTMT	- Allows a component to access per-formatted statements which contain selected full file data (QUAL, LREQID, ORGCODE, etc.) as well as high volume query statements, thus eliminating the need for a component to key the various data elements to execute their request.	DB Manager

The testing will be conducted daily by 10 offices and the volume of data to be used consists of 12,000 tracks. In addition, components will also be able to pass access to an employee's official data from one component to another for individuals pending an upcoming assignment to a specific component. Security is the major focus for Release 1 testing.

2.2.6 CONSTRAINTS

The initial timing requirement for transferring data to PRIM in Release 1 is geared to the update process of data in the HRS2 Data Base. The extract of HRS2 data for updating the PRIM System nightly must occur after the complete nightly update of the HRS2 Data Base. On occasion, due to a timing problem with the HRS2 job stream PRIM may not get updated; however, this will not seriously impact the users. A message appears at Signon time of the last date of update to the Data Base.

2.2.7 REGRESSION TESTING

All PRIM test cases will be documented using preplanned test formats for each and every test with accepted numbering and dating standards to ensure the capability of re-executing any given test or group of tests. The OP Analysts and IOC Test Officers test top down; therefore, if an error is detected and fixed, the testers ensure that the change made will not adversely affect previously tested code.

2.2.8 DATA RECORDING

There is a test packet for every test case which is made from specifications received from the designers. The OP PRIM Analysts will review the results of all tests executed against these test packets. If an error is detected and corrected, the originator of the Problem Report must certify to the OP PRIM Analysts that he/she has retested the applicable procedure and along with the retest date. Data Base Statistics between HRS2 and PRIM Data Lists will be analyzed daily and maintained for at least 30 days after PRIM goes into Production.

2.2.9 EVALUATION

Release 1 of PRIM is basically transferring "Official" data from the HRS2 Data Base to PRIM via extracts; therefore, testing of range of data values used, combination of input types used, etc are not applicable to PRIM in Release 1. In addition since PRIM is due to be updated nightly, the OP PRIM Analysts will not have to create test data for manipulation. Testing evaluation is basically one of evaluating security matrices, verifying the HRS2 INTERFACE coming in is applied properly and passing access to an employee's official data from one component to another.

2.2.10 TEST CASE MAINTENANCE

The OP PRIM Analysts will maintain and file the results of all major or necessary tests during the acceptance phase or until Audit Staff has completed their audit of the system.

PROJECT LIFE CYCLE PHA   OBJECTIVE AND ACTION PLAN

STATINTL

OBJECTIVE NO.	OFFICE	RESPONSIBLE OFFICER
PRIM	DCI/OP	[redacted]
		SDD/A/ODP

OBJECTIVE
(Personnel Resource Information Management)
To provide a computer system satisfying the requirements of Agency components for personal data using the HRS2 Data Base as the prime source of data with the capability to input and update component data, and retrieve all data stored.

FY	RESOURCE ESTIMATE WKYR	DOLLARS
83	4.5	
84	4.1	
85	4.1	
86	1.2	

STATUS period	+	=	-
OCT-DEC		=	
JAN-MAR		=	
APR-JUN		=	
JUL-SEP		=	

+ EXCEEDING PLA
= MEETING PLAN
- BEHIND PLAN

COMPLETION MONTH: SCHEDULED O; ACTUAL X

ACTION PLAN (Milestones) FY: 84	OCT	NOV	DEC	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP
3.2 Design Phase/Detailed Design Development												
3.2.1 Develop Detailed System Design Specifications				⊗		O	⊗					
3.2.2 Develop System Integration Test Plan				⊗		O	⊗					
3.2.3 Develop Acceptance Test Plan					⊗	O	⊗					
3.2.4 Develop Draft of Users Manual					⊗	⊗						
3.2.5 Update System Development Plan				⊗		⊗						
3.2.6 Update Data Specifications Document				⊗		⊗						
3.2.7 Update Requirements Traceability Matrix						⊗ O			⊗			
3.2.8 Update Interface Control Document				⊗		⊗						
3.2.9 Conduct Critical Design Review						O		O		O		⊗
Peak Personnel				4	4	5	4	4	4	4	4	4

STATINTL

PROJECT LIFE CYCLE PHASE OBJECTIVE AND ACTION PLAN

OBJECTIVE NO. PRIM	OFFICE DCI/OP	RESPONSIBLE OFFICER [redacted] SDD/A/ODP

OBJECTIVE
(Personnel Resource Information Management)
To provide a computer system satisfying the requirements of Agency components for personal data using the HRS2 Data Base as the prime source of data with the capability to input and update component data, and retrieve all data stored.

RESOURCE ESTIMATE

FY	WKYR	DOLLARS
83	4.5	
84	4.1	
85	4.1	
86	1.2	

STATUS

period	+	=	-
OCT-DEC		=	
JAN-MAR		=	
APR-JUN		=	
JUL-SEP		=	

+ EXCEEDING PLAN
= MEETING PLAN
- BEHIND PLAN

ACTION PLAN (Milestones) FY: 84

COMPLETION MONTH: SCHEDULED O; ACTUAL X

4.0 System Implementation/Integration Phase

4.1 Develop Program Manual

4.2 Develop Application Manual

4.3 Develop System Integration Test Procedures

4.4 Develop Acceptance Test Procedures

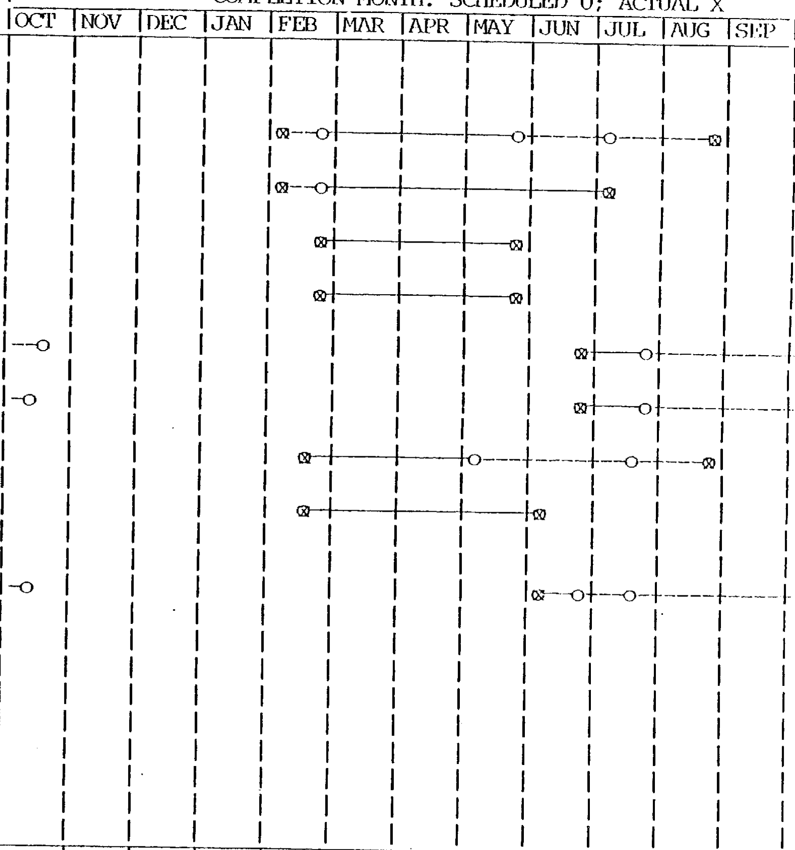
4.5 Write System Integration Test Report

4.6 Write Acceptance Test Report

4.7 Develop Production Manual

4.8 Develop Users Manual

4.9 Conduct Development Test and Evaluation Review



	OCT	NOV	DEC	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP
Peak Personnel					5	5	9	9	9	4	4	4

U N C L A S S I F I E D

Personnel Resource Information Management

(PRIM)

<u>Requirements Traceability Matrix</u>
(RTM-C20-5)

by

PRIM Project Team

ODP/A/SDD
OP/ID/ADRB

23 September 1983

U N C L A S S I F I E D

CONTENTS

CHAPTER PAGE

1. INTRODUCTION 1

 1.1 Purpose 1
 1.2 Scope 1

2. REQUIREMENTS TRACEABILITY MATRIX 2

Chapter 1

INTRODUCTION

1.1 PURPOSE

The purpose of the Requirements Traceability Matrix (RTM) is to identify and trace requirements from the System Initiation Phase through the System Design Phase.

1.2 SCOPE

The Requirements Traceability Matrix is established in the Definition Phase with the identification of requirements specified in the System Requirements Document. The documentation for the PRIM System was already in motion when the ODP Documentation Standards became effective, therefore the PRIM Team did not produce a System Requirements Document. The requirements listed under the SRD in the matrix were actually restated from the Detailed System Requirements Document. A further refinement of the requirements comes in the Detailed System Requirements Document. These detailed requirements are added and their subsequent allocation is identified in the System Definition Document. As the system life cycle progresses through the System Design Phase, the RTM will be expanded to include the Preliminary System Design Specifications, Data Specification Document and the Detailed System Design Specifications.

Chapter 2

REQUIREMENTS TRACEABILITY MATRIX

Listed on the following pages are the traceable requirements for the Detailed System Requirements Document, System Definition Document, Preliminary System Design Specifications, Data Specification Document, and the Detailed System Design Specifications. The traceable requirements for the System Requirements Document are documented in the third level of the Detailed System Requirements Document.

REQUIREMENTS	DOCUMENTS					
	SRD	DSRD	SDD	PSDS	DSD	DSDS
FUNCTIONAL						
Centralizing Official Data for Component Access	3.1.1		3.1.2 3.1.7	4.1.1 4.1.2 4.1.3 4.3.3 4.3.4 4.3.7 4.3.9 4.3.10 4.3.11 4.4.2	2.1 2.2	
Data Transfers from the HRS2 Data Base		3.1.1.1	3.1.2 3.1.7	4.1.2.1 4.1.2.2 4.1.2.3 4.1.2.4 4.1.2.5 4.1.3.1 4.1.3.2 4.1.3.3 4.1.3.4 4.1.3.5 4.3.3.1 4.3.3.2 4.3.3.3 4.3.3.4 4.3.3.5 4.3.4.1 4.3.4.2 4.3.4.3 4.3.4.4 4.3.4.5 4.3.7.1 4.3.7.2 4.3.7.3 4.3.7.4 4.3.7.5 4.3.9.1 4.3.9.2 4.3.9.3 4.3.9.4 4.3.9.5 4.3.10.1 4.3.10.2 4.3.10.3 4.3.10.4 4.3.10.5	2.1.7 2.2.3 2.2.7	4.1.2.6 4.1.2.7 4.1.3.6 4.1.3.7 4.3.3.6 4.3.3.7 4.3.4.6 4.3.4.7 4.3.7.6 4.3.7.7 4.3.9.6 4.3.9.7 4.3.11.6 4.3.11.7 4.4.2.2.6 4.4.2.2.7

REQUIREMENTS	DOCUMENTS					
	SRD	DSRD	SDD	PSDS	DSD	DSDS
				4.3.11.1 4.3.11.2 4.3.11.3 4.3.11.4 4.3.11.5 4.4.2.2		
Data Transfer of COMVAD Dictionaries		3.1.1.2	3.1.7	4.1.2.1 4.1.2.2 4.1.2.3 4.1.2.4 4.1.2.5 4.1.3.1 4.1.3.2 4.1.3.3 4.1.3.4 4.1.3.5	2.1.7 2.2.3 2.2.7	4.1.2.6 4.1.2.7 4.1.3.6 4.1.3.7
Data Transfer of Remaining HRS2 Data		3.1.1.3	3.1.7	4.1.2.1. 4.1.2.2 4.1.2.3 4.1.2.4 4.1.2.5 4.1.3.1 4.1.3.2 4.1.3.3 4.1.3.4 4.1.3.5	2.1.7 2.2.3 2.2.7	4.1.2.6 4.1.2.7 4.1.3.6 4.1.3.7
Data Transfer of Planned/ New HRS2 Data		3.1.1.4	3.1.7	4.1.2.1 4.1.2.2 4.1.2.3 4.1.2.4 4.1.2.5 4.1.3.1 4.1.3.2 4.1.3.3 4.1.3.4 4.1.3.5	2.1.7 2.2.3 2.2.7	4.1.2.6 4.1.2.7 4.1.3.6 4.1.3.7
Non-HRS Data		3.1.1.5	-----	-----	-----	-----
Data Transfer Between Components	3.1.2		3.2.2 3.3.2	4.3.5 4.3.6 4.3.8	2.2.5	
Potential Reassignments		3.1.2.1	3.2.2	4.3.5.1	2.2.5	4.3.5.6

REQUIREMENTS	SRD	DSRD	SDD	PSDS	DSD	DSDS
			3.3.2	4.3.5.2		4.3.5.7
				4.3.5.3		4.3.6.6
				4.3.5.4		4.3.6.7
				4.3.5.5		4.3.8.6
				4.3.6.1		4.3.8.7
				4.3.6.2		
				4.3.6.3		
				4.3.6.4		
				4.3.6.5		
				4.3.8.1		
				4.3.8.2		
				4.3.8.3		
				4.3.8.4		
				4.3.8.5		
Component Data Manipulation	3.1.3		3.2.2	4.2.1	2.1.6	4.3.8.6
			3.2.4	4.3.8	2.2.6	4.3.8.7
			3.3.2			
Queries and Reports	3.1.4		3.1.4	4.4.1		
			3.3.4	4.4.2		
			3.3.6	4.4.3		
			3.4.2	4.4.4		
			3.4.3	4.4.5		
			3.4.4	4.4.6		
			3.4.6	4.4.7		
				4.4.8		
				4.4.9		
Component Generated Online Queries		3.1.4.1	3.4.2	4.4.2.1		4.4.2.2.6
				4.4.2.2		4.4.2.2.7
				4.4.2.3		4.4.2.3.1
						4.4.2.3.2
						4.4.2.3.3
						4.4.2.3.4
						4.4.2.3.5
						4.4.2.3.6
						4.4.2.3.7
Component Generated Offline Reports		3.1.4.2	3.4.2	4.4.3.1		
				4.4.3.2		
				4.4.3.3		
				4.4.3.4		
				4.4.3.5		
				4.4.7.1		
				4.4.7.2		
				4.4.7.3		
				4.4.7.4		
				4.4.7.5		

REQUIREMENTS	SRD	DSRD	SDD	PSDS	DSD	DSDS
				4.4.9.1 4.4.9.2 4.4.9.3 4.4.9.4 4.4.9.5		
Component Generated Basic Graphs		3.1.4.3	3.4.2			
Career Management Reports		3.1.4.4	3.4.2			
Data Base Statistics		3.1.4.5	3.4.3	4.4.8.1 4.4.8.2 4.4.8.3		
Data Base Exception Report		3.1.4.6	3.3.4 3.3.6 3.4.6	4.4.5.1 4.4.5.2 4.4.5.3 4.4.5.4 4.4.5.5		
Data Base Security Reporting		3.1.4.7	3.1.4 3.4.4	4.4.4.1 4.4.4.2 4.4.4.3 4.4.4.4 4.4.4.5 4.4.6.1	2.2.5	
Data Dictionary Reporting		3.1.4.8				
PERFORMANCE						
Reliability	3.2.1		3.1.3 3.2.3			
Consistent Results		3.2.1.1	3.1.3 3.2.3			
Consistent Output Response		3.2.1.2				
Consistent Update Response Level for Component Data		3.2.1.3	3.2.3			
Sizing	3.2.2		3.1.3 3.2.3 3.4.3			
Users		3.2.2.1	3.1.3 3.2.3			

REQUIREMENTS	DOCUMENTS					
	SRD	DSRD	SDD	PSDS	DSD	DSDS
			3.4.3			
Data Files		3.2.2.2	3.1.3 3.2.3 3.4.3			
Accuracy	3.2.3		3.1.3 3.2.3 3.3.3			
Transferred Data		3.2.3.1	3.1.3 3.3.3			
Edits and Validations for Component Data		3.2.3.2	3.2.3			
Timing	3.2.4		3.1.3 3.2.3 3.4.3	4.4.2 4.4.3 4.4.4 4.4.5 4.4.6 4.4.7 4.4.8 4.4.9		
Direct Query		3.2.4.1	3.1.3 3.2.3 3.4.3	4.4.2.1 4.4.2.2		4.4.2.2.6 4.4.2.2.7
Complex Query		3.2.4.2	3.1.3 3.2.3 3.4.3	4.4.2.1 4.4.2.2		4.4.2.2.6 4.4.2.2.7
Hardcopy Reports		3.2.4.3	3.4.3	4.4.3.1 4.4.3.2 4.4.3.3 4.4.3.4 4.4.3.5 4.4.4.1 4.4.4.2 4.4.4.3 4.4.4.4 4.4.4.5 4.4.5.1 4.4.5.2 4.4.5.3 4.4.5.4 4.4.5.5 4.4.6.1 4.4.7.1 4.4.7.2		

REQUIREMENTS	SRD	DSRD	SDD	PSDS	DSD	DSDS
				4.4.7.3		
				4.4.7.4		
				4.4.7.5		
				4.4.8.1		
				4.4.8.2		
				4.4.8.3		
				4.4.9.1		
				4.4.9.2		
				4.4.9.3		
				4.4.9.4		
				4.4.9.5		
Scheduling Updates to PRIM from External Systems		3.2.4.4	3.1.3	4.1.2.1		4.1.2.6
			3.4.3	4.1.2.2		4.1.2.7
				4.1.2.3		4.1.3.6
				4.1.2.4		4.1.3.7
				4.1.2.5		
				4.1.3.1		
				4.1.3.2		
				4.1.3.3		
				4.1.3.4		
				4.1.3.5		
Updates to Component Data		3.2.4.5	3.2.3	4.2.1		
Flexibility	3.2.5		3.4.2			
			3.4.3			
Methods		3.2.5.1	3.4.3			
Media		3.2.5.2	3.4.3			
Component Data		3.2.5.3	3.4.2			
Availability	3.2.6		3.1.3			
			3.2.3			
Normal Hours		3.2.6.1	3.1.3			
			3.2.3			
Other Hours		3.2.6.2	3.1.3			
			3.2.3			
Degraded Mode		3.2.6.3	3.1.3			
			3.2.3			
Maintainability	3.2.7		3.1.3			
			3.4.3			

REQUIREMENTS	DOCUMENTS					
	SRD	DSRD	SDD	PSDS	DSD	DSDS
Software Errors		3.2.7.1	3.1.3 3.4.3			
Software Expandability		3.2.7.2	3.1.3 3.4.3			
Data Base Integrity	3.2.8		3.1.4 3.3.3 3.4.3			
Data Base Currency		3.2.8.1				
Documentation Currency		3.2.8.2				
Stability of User Access		3.2.8.3	3.3.3			
Stability of Data Base Organization		3.2.8.4	3.3.3			
Stability of Data Values		3.2.8.5	3.1.4 3.3.3			
Data Base Statistics		3.2.8.6	3.4.3			
Backup	3.2.9		3.2.3		2.1.4.1 2.2.4.1	
Frequency		3.2.9.1	3.2.3		2.1.4.1 2.2.4.1	
Storage		3.2.9.2	3.2.3		2.1.4.1 2.2.4.1 2.3.4.1	
Recovery	3.2.10		3.2.3			
Restore		3.2.10.1	3.2.3			
Restart		3.2.10.2	3.2.3			
<u>SECURITY</u>						
PRIM Data Base Access Security	3.3.1		3.1.4 3.2.4 3.3.2 3.3.3 3.3.4 3.4.4	4.3.1 4.3.2 4.3.9 4.3.10 4.3.11	2.1.4.2 2.2.4.2 2.3.4.2	

REQUIREMENTS	DOCUMENTS					
	SRD	DSRD	SDD	PSDS	DSD	DSDS
User Data Base Access		3.3.1.1	3.2.4	4.3.2.1	2.1.4.2	4.3.2.6
			3.3.3	4.3.2.2	2.2.4.2	4.3.2.7
			3.3.4	4.3.2.3	2.3.4.2	4.3.9.6
				4.3.2.4		4.3.9.7
				4.3.2.5		4.3.11.6
				4.3.9.1		4.3.11.7
				4.3.9.2		
				4.3.9.3		
				4.3.9.4		
				4.3.9.5		
				4.3.10.1		
				4.3.10.2		
				4.3.10.3		
				4.3.10.4		
				4.3.10.5		
				4.3.11.1		
				4.3.11.2		
				4.3.11.3		
				4.3.11.4		
				4.3.11.5		
Component Data Base Access		3.3.1.2	3.1.4	4.3.2.1	2.2.4.2	4.3.2.6
			3.2.4	4.3.2.2	2.2.5	4.3.2.7
			3.3.2	4.3.2.3		4.3.9.6
			3.4.4	4.3.2.4		4.3.9.7
				4.3.2.5		4.3.11.6
				4.3.9.1		4.3.11.7
				4.3.9.2		
				4.3.9.3		
				4.3.9.4		
				4.3.9.5		
				4.3.10.1		
				4.3.10.2		
				4.3.10.3		
				4.3.10.4		
				4.3.10.5		
				4.3.11.1		
				4.3.11.2		
				4.3.11.3		
				4.3.11.4		
				4.3.11.5		
Non-Agency Data Base Access		3.3.1.3	3.2.4	4.3.2.1	2.2.4.2	4.3.2.6
			3.3.2	4.3.2.2	2.2.5	4.3.2.7
			3.4.4	4.3.2.3		4.3.9.6
				4.3.2.4		4.3.9.7
				4.3.2.5		4.3.11.6
				4.3.9.1		4.3.11.7

REQUIREMENTS	DOCUMENTS					
	SRD	DSRD	SDD	PSDS	DSD	DSDS
				4.3.9.2		
				4.3.9.3		
				4.3.9.4		
				4.3.9.5		
				4.3.10.1		
				4.3.10.2		
				4.3.10.3		
				4.3.10.4		
				4.3.10.5		
				4.3.11.1		
				4.3.11.2		
				4.3.11.3		
				4.3.11.4		
				4.3.11.5		
PRIM Data Access Security	3.3.2		3.1.4	4.3.1	2.1.5	
			3.3.2	4.3.2	2.2.5	
			3.4.4	4.3.3	2.3.5	
				4.3.4		
				4.3.9		
				4.3.10		
				4.3.11		
Component Record Level Access		3.3.2.1	3.1.4	4.3.2.1		4.3.2.6
			3.3.2	4.3.2.2		4.3.2.7
			3.4.4	4.3.2.3		4.3.3.6
				4.3.2.4		4.3.3.7
				4.3.2.5		4.3.4.6
				4.3.3.1		4.3.4.7
				4.3.3.2		4.3.9.6
				4.3.3.3		4.3.9.7
				4.3.3.4		4.3.11.6
				4.3.3.5		4.3.11.7
				4.3.4.1		
				4.3.4.2		
				4.3.4.3		
				4.3.4.4		
				4.3.4.5		
				4.3.9.1		
				4.3.9.2		
				4.3.9.3		
				4.3.9.4		
				4.3.9.5		
				4.3.10.1		
				4.3.10.2		
				4.3.10.3		
				4.3.10.4		
				4.3.10.5		
				4.3.11.1		
				4.3.11.2		

REQUIREMENTS	DOCUMENTS					
	SRD	DSRD	SDD	PSDS	DSD	DSDS
				4.3.11.3		
				4.3.11.4		
				4.3.11.5		
Non-Agency Record Level Access		3.3.2.2	3.3.2	4.3.2.1		4.3.2.6
			3.4.4	4.3.2.2		4.3.2.7
				4.3.2.3		4.3.3.6
				4.3.2.4		4.3.3.7
				4.3.2.5		4.3.4.6
				4.3.3.1		4.3.4.7
				4.3.3.2		4.3.9.6
				4.3.3.3		4.3.9.7
				4.3.3.4		4.3.11.6
				4.3.3.5		4.3.11.7
				4.3.4.1		
				4.3.4.2		
				4.3.4.3		
				4.3.4.4		
				4.3.4.5		
				4.3.9.1		
				4.3.9.2		
				4.3.9.3		
				4.3.9.4		
				4.3.9.5		
				4.3.10.1		
				4.3.10.2		
				4.3.10.3		
				4.3.10.4		
				4.3.10.5		
				4.3.11.1		
				4.3.11.2		
				4.3.11.3		
				4.3.11.4		
				4.3.11.5		
Limited Data Element Access		3.3.2.3	3.1.4	4.3.2.1		4.3.2.6
			3.3.2	4.3.2.2		4.3.2.7
			3.4.4	4.3.2.3		4.3.3.6
				4.3.2.4		4.3.3.7
				4.3.2.5		4.3.4.6
				4.3.3.1		4.3.4.7
				4.3.3.2		4.3.9.6
				4.3.3.3		4.3.9.7
				4.3.3.4		4.3.11.6
				4.3.3.5		4.3.11.7
				4.3.4.1		
				4.3.4.2		
				4.3.4.3		
				4.3.4.4		

REQUIREMENTS	SRD	DSRD	SDD	PSDS	DSD	DSDS
				4.4.7.4		
				4.4.7.5		
				4.4.9.1		
				4.4.9.2		
				4.4.9.3		
				4.4.9.4		
				4.4.9.5		
Online Data Base Reporting		3.3.3.3	3.4.4	4.4.2.1		4.4.2.2.6
				4.4.2.2		4.4.2.2.7
PRIM Data Base Update Security	3.3.4		3.1.4	4.1.1		
			3.2.4	4.1.2		
			3.3.3	4.1.3		
			3.4.4	4.2.1		
HRS2 Transferred Data		3.3.4.1	3.1.4	4.1.2.1		4.1.2.6
			3.3.3	4.1.2.2		4.1.2.7
				4.1.2.3		4.1.3.6
				4.1.2.4		4.1.3.7
				4.1.2.5		
				4.1.3.1		
				4.1.3.2		
				4.1.3.3		
				4.1.3.4		
				4.1.3.5		
Component Data		3.3.4.2	3.2.4	4.2.1		
			3.3.3			
			3.4.4			
Software	3.3.5		3.1.4			
			3.1.6			
			3.2.4			
			3.2.6			
			3.3.4			
Production Software		3.3.5.1	3.1.4		2.2.3	
			3.2.6			
			3.3.4			
Development/Maintenance Software		3.3.5.2	3.1.6		2.3.3	
			3.2.4			
			3.3.4			
System Software		3.3.5.3	3.2.4			
			3.3.4			
Hardware	3.3.6		3.1.4			

REQUIREMENTS	DOCUMENTS					
	SRD	DSRD	SDD	PSDS	DSD	DSDS
				4.3.4.5		
				4.3.9.1		
				4.3.9.2		
				4.3.9.3		
				4.3.9.4		
				4.3.9.5		
				4.3.10.1		
				4.3.10.2		
				4.3.10.3		
				4.3.10.4		
				4.3.10.5		
				4.3.11.1		
				4.3.11.2		
				4.3.11.3		
				4.3.11.4		
				4.3.11.5		
Access to Classified Data		3.3.2.4	3.3.2			
			3.4.4			
PRIM Data Base Reporting Security	3.3.3		3.1.4	4.4.2	2.2.5	
			3.3.4	4.4.3		
			3.4.4	4.4.4		
				4.4.5		
				4.4.6		
				4.4.7		
				4.4.9		
Data Base Security Reporting		3.3.3.1	3.1.4	4.4.4.1	2.2.5	
			3.3.4	4.4.4.2		
			3.4.4	4.4.4.3		
				4.4.4.4		
				4.4.4.5		
				4.4.5.1		
				4.4.5.2		
				4.4.5.3		
				4.4.5.4		
				4.4.5.5		
				4.4.6.1		
Offline Data Base Reporting		3.3.3.2	3.4.4	4.4.3.1		
				4.4.3.2		
				4.4.3.3		
				4.4.3.4		
				4.4.3.5		
				4.4.7.1		
				4.4.7.2		
				4.4.7.3		

REQUIREMENTS	SRD	DSRD	SDD	PSDS	DSD	DSDS
			3.1.5			
			3.2.4			
			3.2.5			
			3.3.5			
			3.4.5			
Existing Equipment		3.3.6.1	3.1.5			
			3.2.5			
			3.3.5			
			3.4.5			
New Equipment		3.3.6.2	3.1.5			
			3.2.5			
			3.3.5			
			3.4.5			
Tempest		3.3.6.3	3.1.4			
			3.2.4			
			3.3.5			
<u>HARDWARE</u>						
Current Equipment	3.4.1		3.1.5			
			3.2.5			
			3.3.5			
			3.4.5			
Type of Equipment		3.4.1.1	3.1.5			
			3.2.5			
			3.3.5			
			3.4.5			
Future Equipment	3.4.2					
Type of Equipment		3.4.2.1				
<u>HUMAN ENGINEERING</u>						
Simplicity	3.5.1	3.5.1	3.4.6			
System Access		3.5.1.1	3.3.6			
Query and Report		3.5.1.2	3.4.6			
Job Initiation		3.5.1.3	3.4.6			
User Friendliness	3.5.2		3.1.6			
			3.2.6			
			3.3.6			
			3.4.6			

REQUIREMENTS	DOCUMENTS					
	SRD	DSRD	SDD	PSDS	DSD	DSDS
Guidance		3.5.2.1	3.2.6 3.3.6			
Warning/Error Messages		3.5.2.2	3.1.6 3.2.6 3.3.6 3.4.6			
<u>INTERFACE</u>						
Data Transfers from the HRS2 Data Base	3.6.1		3.1.7 3.2.7	4.1.1 4.1.2 4.1.3 4.3.3 4.3.4 4.3.7	2.1.7 2.2.3 2.2.7 2.3.7	
Employee Data		3.6.1.1	3.1.7	4.1.2.1 4.1.2.2 4.1.2.3 4.1.2.4 4.1.2.5 4.1.3.1 4.1.3.2 4.1.3.3 4.1.3.4 4.1.3.5 4.3.3.1 4.3.3.2 4.3.3.3 4.3.3.4 4.3.3.5 4.3.4.1 4.3.4.2 4.3.4.3 4.3.4.4 4.3.4.5 4.3.7.1 4.3.7.2 4.3.7.3 4.3.7.4 4.3.7.5	2.1.7 2.2.3 2.2.7 2.3.7	4.1.2.6 4.1.2.7 4.1.3.6 4.1.3.7 4.3.3.6 4.3.3.7 4.3.4.6 4.3.4.7 4.3.7.6 4.3.7.7
Organization and Position Data		3.6.1.2	3.1.7	4.1.2.1 4.1.2.2 4.1.2.3 4.1.2.4 4.1.2.5	2.1.7 2.2.3 2.2.7 2.3.7	4.1.2.6 4.1.2.7 4.1.3.6 4.1.3.7 4.3.3.6

REQUIREMENTS	DOCUMENTS					
	SRD	DSRD	SDD	PSDS	DSD	DSDS
				4.1.3.1 4.1.3.2 4.1.3.3 4.1.3.4 4.1.3.5 4.3.3.1 4.3.3.2 4.3.3.3 4.3.3.4 4.3.3.5 4.3.4.1 4.3.4.2 4.3.4.3 4.3.4.4 4.3.4.5 4.3.7.1 4.3.7.2 4.3.7.3 4.3.7.4 4.3.7.5		4.3.3.7 4.3.4.6 4.3.4.7 4.3.7.6 4.3.7.7
Edit/Validation Dictionaries		3.6.1.3	3.1.7 3.2.7	4.1.2.1 4.1.2.2 4.1.2.3 4.1.2.4 4.1.2.5 4.1.3.1 4.1.3.2 4.1.3.3 4.1.3.4 4.1.3.5	2.1.7 2.2.3 2.2.7 2.3.7	4.1.2.6 4.1.2.7 4.1.3.6 4.1.3.7
Data Transfer of COMVAD Dictionaries	3.6.2		3.1.7 3.2.7	4.1.1 4.1.2 4.1.3	2.1.7 2.2.3 2.2.7 2.3.7	
Data Transfer of Remaining HRS2 Data	3.6.3		3.1.7	4.1.1 4.1.2 4.1.3		
Data Transfer of Planned/New HRS2 Data	3.6.4		3.1.7	4.1.1 4.1.2 4.1.3		
<u>CONVERSION</u>	-----	-----	-----	-----	-----	-----

REQUIREMENTS	DOCUMENTS					
	SRD	DSRD	SDD	PSDS	DSD	DSDS
LEGAL						
Privacy Act	3.8.1					
Significant Changes		3.8.1.1				

ADMINISTRATIVE-INTERNAL USE ONLY

PERSONNEL RESOURCE INFORMATION MANAGEMENT (PRIM)

Detailed System Design Specifications

by

PRIM Project Team

ODP/A/SDD
OP/ID/ADRB

23 September 1983

ADMINISTRATIVE-INTERNAL USE ONLY

CONTENTS

CHAPTER PAGE

1. INTRODUCTION 1

1.1 PURPOSE 1
1.2 SCOPE 1
1.3 REFERENCES 2

2. DESIGN OVERVIEW 3

3. HARDWARE/OPERATING ENVIRONMENT 4

3.1 SYSTEM ARCHITECTURE OVERVIEW 4
3.1.1 LEX AND SYNTAX ANALYSIS 4
3.1.2 SECURITY PROCESSOR 5
3.1.3 DICTIONARY DEFINITION AND STRUCTURE 5
3.1.4 DEBUG PROCESSOR 5
3.1.5 LOADER PROCESSOR 5
3.1.6 DATA BASE MANAGEMENT PROCESSOR (DBM) 6
3.1.7 PARSE AND PROCESS CONTROLLER (PCON) 6
3.1.8 UPDATE PROCESSOR 7
3.1.9 RETRIEVAL PROCESSOR 7
3.1.10 FILE MANAGEMENT SYSTEM PROCESSOR 7
3.1.11 PROCEDURE LIST PROCESSOR 7
3.1.12 MISCELLANEOUS SUPPORT PROCESSOR (MSSC) 7
3.2 HARDWARE DEVICES 8
3.3 SYSTEM SOFTWARE 8

4. FUNCTIONAL SOFTWARE DESIGN 10

4.1 CENTRALIZING OFFICIAL DATA FOR COMPONENT ACCESS 10
4.1.1 DEFINITION 10
4.1.2 PRDDUMP - HRS2/PRIM INTERFACE 10
4.1.3 PRDLOAD - LOAD PRIM 13
4.2 COMPONENT DATA FILES FOR MANIPULATION 16
4.2.1 DEFINITION 16
4.3 CONTROLLED COMPONENT DATA ACCESS 16
4.3.1 DEFINITION 16
4.3.2 PRCAESTB - ESTABLISH COMPONENT ACCESS 17
4.3.3 PRCAUPDT - COMPONENT ACCESS UPDATE 26
4.3.4 PRCHGLOAD - COMPONENT ACCESS CHANGES 29
4.3.5 PRCATRSF - TRANSFER COMPONENT ACCESS 84
4.3.6 PRNTEPRG - NOT-TO-EXCEED PURGE 96
4.3.7 PRINTPRG - INTERFACE PURGE 100
4.3.8 PRACCUPDT - ACCESS UPDATE 103

4.3.9 PRCALINK - PRIM LINKS CREATED DAILY BY COMPONENT 114
4.3.10 PRINDX - PRIM INDEXING OF DATA 122
4.3.11 PRCRTLINK - PRIM LINKS CREATED WEEKENDS BY DBCC 124
4.4 COMPONENT RETRIEVAL 127
4.4.1 DEFINITION 127
4.4.2 ONLINE QUERIES 127
4.4.3 OFFLINE REPORTS 133
4.4.4 DATA BASE ACCESS VIOLATION REPORT 134
4.4.5 DATA BASE EXCEPTION REPORT 135
4.4.6 ACF2 VIOLATION REPORT 136
4.4.7 PRNTEPRG REPORT 137
4.4.8 DBSTATS REPORT 138
4.4.9 PRCHANGE REPORT - REMOVE, SEPARATE, AND SSN CHANGES 138

5. CONVERSION - NOT APPLICABLE TO PRIM 140

6. APPENDIX 141

LIST OF TABLES

TABLE PAGE

1. Documents, Publications and Reference Material 2

Chapter 1

INTRODUCTION

1.1 PURPOSE

The Detailed System Design Specifications (DSDS) document is an expansion of the Preliminary System Design Specifications (PSDS) document for the PRIM System to describe the detailed approach by functional area. This document is used for coding and permits a controlled initiation of the Implementation and Integration Phase.

1.2 SCOPE

The PSDS presents the design overview, hardware/operating environment, and description of system software by functional area. The DSDS expands the PSDS to include system design at a level of detail which allows the software to be coded. Once approved by the Critical Design Review, this document is used for coding, as well as, a control point for the Implementation and Integration Phase.

Chapter 1 - 'INTRODUCTION' presents the purpose and scope of the PSDS and all references applicable to the contents of this document.

Chapter 2 - 'DESIGN OVERVIEW' presents the scope of the design effort, identification of the functional areas comprising the development effort, and the functional flow of the system.

Chapter 3 - 'HARDWARE/OPERATING ENVIRONMENT' presents a discussion of the design for the PRIM operating system, including the following subjects: Identification of the hardware and software packages to be utilized, an overview of the system architecture, description of hardware devices involved, and a description of each piece of system software in the design.

Chapter 4 - 'FUNCTIONAL SOFTWARE DESIGN' presents the design of the PRIM System software by functional area and a detailed design of each function.

Chapter 5 - 'SYSTEM CONVERSION SPECIFICATIONS'. N.A.

1.3 REFERENCES

The PRIM Project Team is utilizing a number of documents, publications and other reference material listed in Table 1 below.

TABLE 1

Documents, Publications and Reference Material

1. ODP Applications Documentation Standards
2. PRIM System Development Plan
3. PRIM Detailed System Requirements Document
4. PRIM Data Requirements Document
5. System Definition Document
6. Pleminary System Design Specifications

Chapter 2

DESIGN OVERVIEW

The purpose of this effort is to develop a centralized data base for use by the Personnel Officer, Career Management Officer, Office Director, or Training Officer of a component in direct support of the component's day-to-day personnel management activities. The PRIM Data Base will contain organizational, position, and employee data currently resident on the HRS2 Data Base, and will provide components with the capability to access the data for queries and reports as well as the capability to enter, update, and retrieve their own component data. Data will be received from HRS2 on a nightly basis. For ease in the interface and because HRS2 is a GIM-II data base, PRIM will also be designed as a GIM-II data base. An intricate security matrix will control all read/write access to the PRIM Data Base as appropriate.

The PRIM development effort is comprised of the following four functional areas:

- Centralizing Official Data For Component Access.

- Component Data Files For Manipulation

- Controlled Component Data Access.

- Data Retrieval by Components.

Figure 1 of the Appendix depicts the functional flow of the PRIM System.

The PRIM System is a new application and conversion from an existing system is unnecessary. It will, however, interface with the HRS2 Data Base for all of it's data in the first Release.

Chapter 3

HARDWARE/OPERATING ENVIRONMENT

Figure 2 of the Appendix illustrates the Hardware Configuration for the PRIM System.

The schematic in Figure 2.1 of the Appendix illustrates a simplified version of the overall design of the Data Base Management System (DMS) services of GIM-II. Each of these blocks represents a basic function which is performed by the GIM-II system. The GIM-II functions are evoked through language statements or transactions. Each statement is analyzed via the Lex and Syntax Analysis Processor and, depending upon the keywords identified within that transaction, control is passed to either an appropriate processor or a specified processing string is built. Each of the processors as they pertain to PRIM will be described below in System Architecture.

3.1 <u>SYSTEM ARCHITECTURE OVERVIEW</u>

3.1.1 <u>LEX AND SYNTAX ANALYSIS</u>

The Lex & Syntax Analysis Processor translates the user input statement into an interpretive processing string.

1. Performs all Master Dictionary and User Dictionary Retrievals, and S/EDITS,

2. Checks statements for conformance to syntactical rules,

3. Initializes all processing tables,

4. Adds all appropriate elements to the interpretive string as directed by edits and/or correlatives, and

5. Forms special strings and exits to special processors for types 1 and 2 verbs.

3.1.2 SECURITY PROCESSOR

The Security Processor provides for data base protection via the SIGNON/SIGNOFF verbs and LOCK/KEY processing. The Security Processor processes the SIGNON and SIGNOFF functions and provides support services for data list, device and program protection. The LOCK/KEY processor will limit access to specific fields (i.e. race code), specific files, and verbs.

3.1.3 DICTIONARY DEFINITION AND STRUCTURE

The Dictionary Definition and File Structure Processor provides for the structural control of the data base and performs the following functions:

1. Audits user's structural definitions and dictionary formats,
2. Allocates new data list areas and releases data list areas,
3. Compiles dictionaries into an interrelated set of interpreted definitions,
4. Provides for the management of the compile area,
5. Provides support to other processors interfacing with the dictionary or name tables, and
6. Prepares appropriate user messages.

3.1.4 DEBUG PROCESSOR

The Debug Processor is a collection of functions whose requirements are derived from the necessity to assist both program maintenance and initial data base construction and maintenance activities. These functions are not used in a production or operational environment but in a test environment. These multiple functions allow the analyst/programmers to check and debug the performance of the PRIM application.

3.1.5 LOADER PROCESSOR

The Loader Processor provides the capability to bulk load data via the DLOAD/SYSLOAD/BULK-LOAD functions. The Loader Processor performs the following functions:

1. Bootstraps the system,

2. Forms disk images for loading data into the data base from fixed format records which are externally definable, and

3. Forms disk images for loading data into the data base from records in GIM-II record format.

3.1.6 DATA BASE MANAGEMENT PROCESSOR (DBM)

The Data Base Management Processor provides the basic checkpoint and recovery capabilities via functions such as DDUMP, RESTORE, REPROCESS, REALLOCATE, etc. The DBM Processor:

1. Determines the services required and dispatches the proper function,

2. Performs user and data list integrity checks,

3. Provides a PHYSICAL and/or LOGICAL dump tape, and a PHYSICAL/LOGICAL disk restore capability,

4. Provides history tape analysis and reprocessing, and

5. Prepares appropriate user messages.

3.1.7 PARSE AND PROCESS CONTROLLER (PCON)

The function of the Parse and Process Controller is to form the executable string and select the appropriate logical item (record) for processing. Processors return to PCON for each new item to be processed. The Parse and Process Controller:

1. Constructs executable strings in reverse polish notation where appropriate,

2. Performs analysis of logical and arithmetical phrases for consistency of definition,

3. Selects all logical items (records) to be processed,

4. Supports all logical and arithmetical calculations, including Editing and Selection, and

5. Dispatches the selected logical item (record) to the appropriate processor.

3.1.8 UPDATE PROCESSOR

The Update Processor controls the basic functions of ADD, CHANGE, and DELETE. All new record images from these functions are created via the Update Processor. The Update Processor:

1. Validates the input values against the defined structural requirements,

2. Evokes any secondary data lists to be updated,

3. Creates a new image of the primary and secondary data list records, and

4. Prepares user notification of results.

3.1.9 RETRIEVAL PROCESSOR

The Retrieval Processor consists of the basic retrieval functions, e.g. LIST, COUNT, TOTAL, and EXTRACT, and their derivative functions, such as LISTV, LISTSA, LISTSD, etc. All general user retrievals from the data base are processed via this function.

3.1.10 FILE MANAGEMENT SYSTEM PROCESSOR

3.2 HARDWARE DEVICES

The PRIM Data Base will be physically located in the Ruffing Computer Center (GC03), and will reside on one or more direct access storage devices such as 3350 disk packs. The data base management system for PRIM (GIM-II) will operate on a central processing unit (CPU) in GC03 such as the AMDAHL. Online access to the PRIM Data Base will be via video terminals such as the Delta Data 5000 or 7260, controlled by switching units such as the COMTEN, which connect the terminals to the main computer.

Offline access to PRIM, for hardcopy reports, will be through printers in the component offices, such as the Design 100 and Texas Instrument Silent 700, or through high speed printers located in the area Data Base Control Centers and the Ruffing Center.

Changes to the hardware configuration of the CPU or to any peripheral equipment such as terminals, printers, switching units or disk drives must be compatible with the PRIM System.

3.3 SYSTEM SOFTWARE

The software system utilized by PRIM will be the Generalized Information Management (GIM-II) System. The purpose in using GIM-II is to simplify the definition, creation, maintenance, and interrogation of the PRIM Data Base. The GIM-II software functions provide for such features as:

- File Definition
- Initial Data Base Construction,
- Update, Selection, and Retrieval of Items,
- Checkpoint and Restoration of the Data Base,
- Security,
- Data Base Validation and Statistics, and
- Recording of Transactions on History Tapes.

Designed as a generalized data base management system, GIM-II has as objectives the following:

- Flexible data structures suitable to each application (hierarchy-network),
- Variety of access/search methods,
- Centralized control of the physical organization of data,
- Storage of data in relation to access frequency and response requirements,
- Data independence of programs and devices,
- Integrity of the data base against destruction and/or security breach,
- Recycle and restart techniques in the event of hardware/software failures,
- User interaction with the data base via inquiry

system or language, and

- Multiple update/retrieval of information from the data base.

The GIM-II data base and program libraries reside on direct-access devices to permit rapid response to online queries. In addition, the GIM-II System will provide a limited number of PRIM users with the capability to extract data to tape from the data base and to use these extract tapes to produce offline reports in a variety of formats.

The GIM-II software system will satisfy PRIM's requirements to provide simultaneous, controlled access to the PRIM Data Base for multiple components, and allow for both online and offline reporting of current data from Official Personnel and Component Data files.

The GIM-II system operates under all the normal constraints of the host operating system (MVS) as any other task within the job stream. The PRIM System will reside on an online computer in the Ruffing Computer Center.

Chapter 4

FUNCTIONAL SOFTWARE DESIGN

A matrix of PRIM requirements is provided in Figure 3 of the Appendix. Data flow diagrams are also provided for each of the programs listed. Figure 4 in the Appendix is a list of all the procedures and a brief description of their function. Figure 5 in the Appendix is a list of files that are mentioned in the flow diagrams and their basic function.

4.1 CENTRALIZING OFFICIAL DATA FOR COMPONENT ACCESS

4.1.1 DEFINITION

THE PRIM System will provide a centralized data base for Agency Components to retrieve official data (organizational, position, and employee data) currently resident in the Human Resources System (HRS2 Data Base). Specific files will be centralized on the PRIM Data Base through a series of ITEM-DUMPCX statements on HRS2 and then a series of DELETE-DATA statements followed by an ITEM-LOADX statement on PRIM. A flow diagram of the two procedures to perform this function is provided in Figure 6 of the Appendix.

4.1.2 PRDDUMP - HRS2/PRIM INTERFACE

4.1.2.1 PURPOSE

The movement of all specified data files from HRS2 to PRIM will be done via this procedure. The data file names are listed and maintained in the PRIFN file by the HRS2 Data Base Manager. See flow diagram in Figure 6 of the Appendix.

4.1.2.2 GENERAL INFORMATION

Proc Name	: PRDDUMP
Language Used:	GIM POL
Initiated By	: Production Division - DBCC
Frequency Of Execution	: After all HRS2 updates are completed; will be done nightly as part of the HRS2/PRIM interface.
Input	: File names stored in PRIFN file on the HRS2 Data Base. PRIFN (DL/ID) PRWKENDFLG

Output	: The PRIFN file will be dumped to a reserved disk and all data from specified files will be dumped to a tape.
	DSN= G3P.A3176600.PRIM.PRIFN
	DSN= G31.A31C2000.PRIM.UPDATES

4.1.2.3 FUNCTIONAL DESCRIPTION

The PRIFN file is updated by the HRS2 Data Base Manager whenever additional files are included in the HRS2/PRIM interface. The PRDDUMP procedure will acquire interface file names from the PRIFN file and then 'ITEM-DUMPX' the PRIFN file to a disk. A PRIFN record (WKEND) will be updated to contain a 'Y' in the PRWKENDFLG field when all files are to be dumped. If an "N" appears in this field, only the files listed in the PRIFN file that do not contain a value in PRWKENDFLG field will be moved. Some COMVAD translate files are to be moved only on weekends (Friday, Saturday or Sunday). These files will be flagged with an 'W' in the PRWKENDFLG field. The PRDDUMP will build an ITEM-DUMPCX statement for each of the files to be moved. After all statements have been built, the procedure will execute each statement. Data for all files will be dumped to a tape which will then be passed to the PRDLOAD procedure along with the disk containing the PRIFN file.

4.1.2.4 INPUT

This procedure will not utilize a menu; 'E PRDDUMP' will begin procedure execution. The procedure will create a dump statement for each file.

```
- $DSN=(disk),$VOL=(CATLOG). ITEM-DUMPX 'PRIFN'

- $DSN=(tape),$VOL=(CATLOG) . ITEM-DUMPCX '1ST FILE NAME'

- $DSN=(tape),$VOL=(CATLOG) . ITEM-DUMPCX '2nd FILE NAME'
```

- etc.

4.1.2.5 OUTPUT

One disk dataset will be created containing the PRIFN file and one tape will be created containing all selected data files to be loaded to the PRIM Data Base by the PRDLOAD procedure.

4.1.2.6 GENERAL PROGRAM CONSIDERATIONS

The names of the files to be moved from HRS2 to PRIM are stored in the PRIFN file. The files that are to be moved only on weekends are not critical files and contain a 'W' in the PRWKENDFLG field.

An algorithm will be performed in the PRDDUMP procedure to determine the day of the week. On the weekend (Friday, Saturday, or Sunday), all files named in the PRIFN will be loaded. On other days, only those files with a blank in the PRWKENDFLG field will be loaded. The WKEND record in the PRIFN file will be updated with a Y or N value dependent on the result of the algorithm performed.

A disk and a tape will be used for the PRDDUMP and PRDLOAD operations. The disk will contain only the PRIFN file, the tape will contain the data of files dumped from HRS2.

4.1.2.7 DETAILED PROGRAM SPECIFICATIONS

```
DEC PRDDUMP
  DISK        disk containing PRIFN data
  TAPE        tape containing multiple file data
  PRIFN       file containing files to be moved
    WKEND     record in PRIFN showing type of DUMP to do
    WKENDFLG  Flags those files for weekend move only
enddec prddump

PROC PRDDUMP
  DO algorithm to determine day of week
    YYDDD = YY = Year
           DDD = Julian day of year
    XY = 365.25 * YY
    XY = DDD + XY
    IF decimal portion of XY = 0 ($SBF(XY,1,1'.')
    THEN  XY = XY - 1.00
    endif;
    WHOLEXY = WHOLE PORTION OF XY / 7
              ($SBF(XY,0,1,'.') / 7)
    REMAINDER = Decimal Portion of WHOLEXY
```

```
               ($SBF(WHOLEXY,1,1,'.')
   enddo;
   IF REMAINDER EQ 0 or REMAINDER EQ 7 or
      REMAINDER EQ 8
   THEN ADD 'Y' to PRIFN WKEND record
        (FOR PRIFN 'WKEND' ADD PRWKENDFLG 'Y')
   ELSE ADD 'N' to PRIFN WKEND record
        (FOR PRIFN 'WKEND' ADD PRWKENDFLG 'N')
   endif;
   DUMP PRIFN file to DISK
     . $DISK,$DSN=G3P.A3176600.PRIM.PRIFN,$VOL=LOCATE .
     FOR PRIFN ITEM-DUMPX
   IF REMAINDER EQ 0 or REMAINDER EQ 7 OR
      REMAINDER EQ 8
   THEN ACQUIRE all PRIFN records with PRWKENDFLG EQ
        'W' or with null PRWKENDFLG
   ELSE ACQUIRE all PRIFN records with null PRWKENDFLG
   endif;
   DO FOR ALL RECORDS ACQUIRED
      BUILD ITEM-DUMPCX statement
      EXECUTE statement
   enddo;
   RELEASE tape
 endproc PRDDUMP;
```

4.1.3 <u>PRDLOAD</u> <u>-</u> <u>LOAD</u> <u>PRIM</u>

4.1.3.1 PURPOSE

This procedure will create and execute statements to load HRS2 data to PRIM from a disk and a tape produced in the PRDDUMP procedure. A second step to the PRDLOAD process will roll the STRENGTH data up to the Directorate level. See flow diagram in Figure 6 of the Appendix.

4.1.3.2 GENERAL INFORMATION

Proc Name :	PRDLOAD
	DAC-STMTS
Language Used:	GIM POL
Initiated By :	Production Division - DBCC
Frequency Of Execution :	This procedure must be executed in PRIM after the PRDDUMP procedure is executed in HRS2.
Input :	PRIFN file data on disk produced by the PRDDUMP procedure in HRS2.
	PRIFN(DL/ID) PRWKENDFLG
	Selected files per PRIFN PRWKENDFLG on

		tape produced by the PRDDUMP procedure procedure in HRS2. Second Step: STRENGTH data (loaded daily)
Output	:	PRIM files will be in sync with HRS2. STRENGTH data will be current as well as rolled up to Directorate level.

4.1.3.3 FUNCTIONAL DESCRIPTION

The PRDLOAD procedure will execute an ITEM-LOADX of the PRIFN file resident on the disk. The PRIFN file data will then be used to build DELETE-DATA statements for the additional files on the tape produced by the PRDDUMP procedure. An ITEM-LOADX statement will then be executed to load all the files to PRIM from the tape. Files to be loaded only on weekends (Friday, Saturday or Sunday) will be flagged with an 'W' in the PRWKENDFLG field.

A second step will be executed after successful completion of the file loads. This step will utilize a permanent disk (100 tracks) to extract (ETF) data from the STRENGTH file and BULK-CHANGE the data back to the STRENGTH file at the Directorate level (first position of ORGCODE).

4.1.3.4 INPUT

No menus will be required for this procedure. The procedure will be initiated with the following statement:

```
     E PRDLOAD    (PRIM LOAD)

E DAC-STMTS STRENGTH   (STRENGTH rollup)
```

The procedure will execute an ITEM-LOAD statement for the PRIFN file. The procedure will then create a DELETE-DATA statement for each file named in the PRIFN file that is to be moved followed by another ITEM-LOAD statement to load new data into the files.

Another step will use a second disk to rollup STRENGTH data to the Directorate level using the DAC-STMTS procedures and statements stored in the STRENGTH record in SYSER.

```
 -  $DSN=(disk1),$VOL=(LOCATE) . ITEM-LOADX

 -  $DSN=(file),$VOL=(LOCATE) . ITEM-LOADX

 -  $DSN=(disk2),$VOL=(LOCATE) . FOR STRENGTH ETF . . .

 -  $DSN=(disk2),$vol=(LOCATE) . BULK-CHANGE STRENGTH . . .
```

4.1.3.5 OUTPUT

The system will generate an information line when each file load is completed and a final statement when the entire load procedure is completed.

The second step (rollup of STRENGTH) will produce an completion message in the STRENGTHMSG record of SYSER.

4.1.3.6 GENERAL PROGRAM CONSIDERATIONS

The names of the files to be moved from HRS2 to PRIM are stored in the PRIFN file. The files that are to be moved only on weekends are not critical files and contain a 'W' in the PRWKENDFLG field.

An algorithm is performed in the PRDDUMP procedure to determine the day of the week. The result of the algorithm is stored in the WKEND record of the PRIFN file. The PRIFN file will always be loaded first and the WKEND record will be checked to determine if a weekend or daily load is to be done. The old file data is deleted from the appropriate files before the ITEM-LOAD is executed.

A disk and a tape will be used for the PRDDUMP and PRDLOAD operations. The disk will contain only the PRIFN file, the tape will contain the files dumped from HRS2.

A second step in the LOAD process will use a second disk to extract data from the STRENGTH file and BULK-CHANGE the data back to the STRENGTH file at the Directorate level. This process will use the DAC-STMTS procedure and two statements stored in the SYSER STRENGTH record.

4.1.3.7 DETAILED PROGRAM SPECIFICATIONS

```
DEC PRDLOAD
  DISK1          disk containing PRIFN data
  TAPE           tape containing multiple file data
  PRIFN          file containing files to be loaded
    WKEND        record to notify which kind of load to do
    PRWKENDFLG   flags files to be moved on weekends only
enddec PRDLOAD

PROC PRDLOAD
  LOAD the disk        (contains PRIFN data)

  ACQUIRE WKEND record PRWKENDFLG

  IF WKEND record PRWKENDFLG EQ 'Y' (weekend)
  THEN ACQUIRE all PRIFN records except WKEND
       (FOR PRIFN WITH PRIFN NE 'WKEND' ACQUIRE)
```

```
ELSE ACQUIRE all PRIFN records with null PRWKENDFLG
     (FOR PRIFN WITH NULL PRWKENDFLG ACQUIRE)
endif;

DO for all file names acquired
  DELETE-DATA filename
enddo;

LOAD data from the tape
 (. $DSN=G31.A31c2000.PRIM.UPDATES(0),$VOL=LOCATE . ITEM-LOADX)

EXIT PRDLOAD

endproc PRDLOAD

SECOND PART OF LOAD PROCESS:

  E DAC-STMTS STRENGTH
```

4.2 COMPONENT DATA FILES FOR MANIPULATION

4.2.1 DEFINITION

The Component Data files will give the components the ability to add, change, and delete data in files reserved for each component's use. The actual data field requirements for these files have not been gathered. Decisions will be made at meetings with the referents. The actual design of this requirement will be done in Release 2.

4.3 CONTROLLED COMPONENT DATA ACCESS

4.3.1 DEFINITION

The PRIM System must provide a means of restricting components to only that data which pertains to that component. This will be done through a security matrix utilizing segmented files. Access segments will be established in the SEGACCESS file for each component on the PRIM System consisting of a separate access segment for each level of access requested by each component. The PRIM Data Base Manager, in conjunction with the component, will be responsible for ordering the establishment of new segments and for adding the selection criteria needed to establish the access data in the SEGACCESS segments used to access Active (PRIM PERSIGN) and Separated (PRIMSEP) data. Further changes to this SELECTION data will be done by the data base manager. Flow diagrams of the seven procedures required to perform this function are provided in Figures 7 thru 13 of the Appendix.

4.3.2 PRCAESTB - ESTABLISH COMPONENT ACCESS

4.3.2.1 PURPOSE

To establish a component's access in the PRIM System, the Data Base Manager will request that new segments be created in the SEGACCESS file (one segment for Active links and one segment for Separated links). A new segment will be created in the SELECTION file to contain the Selection Criteria (SLCRITERIA), SLORGLINKS, and SLPOSLINKS. The Data Base Manager will request an addition to SYSMAN2, and then use the PRCAESTB procedure to add Criteria to the SELECTION file which is used to establish the access links to the PRIM Official Data files and to the Separated Data file (PRIMSEP) for specified SEGACCESS file segments. See flow diagram in Figure 7 of the Appendix.

NOTE: The first character of the SEGACCESS file name cannot be a 'Z' or a numeric value because of SYSMAN2 limitations.

4.3.2.2 GENERAL INFORMATION

Proc Name :	PRCAESTB
Signon ID :	The PRIM ORG identifier will be established by the Data Base Manager and will be unique for each level of access.
Language Used:	GIM POL
Frequency Of Execution :	Executed whenever a new segment is to be created, changed, or deleted.
Input :	SSNOR Career Service Designation ORGCODE Schedule/Grade Sub-Category Code Occupational Code
Output :	An entry will be made into the new SELECTION segment used by the PRCALINK Procedure to input the DL/ID (SSNOR) of records that can be accessed by the newly established segment.

4.3.2.3 FUNCTIONAL DESCRIPTION

This procedure will establish the access links to the PRIM Official Data Files based on ORGCODE, SD, OCCE, SCH, GR, or SCCE. The PRIM Data Base Manager will input the selection parameters via a menu. This data will be used by the PRCALINK procedure to retrieve and store all SSNOR's to a specific SEGACCESS segment at the time of the first daily access. The PRCAESTB procedure can also be used to change the access level by entering new access level selection parameters for a given SEGACCESS segment.

4.3.2.4 INPUT

This procedure will utilize the following menu which will be stored in the MENU-FORMATS file: (Component Access Establish)

```
E PRCAESTB    * * * DATA BASE MANAGER USE ONLY * * *

PROCEDURE ACTION: __     ADD(A)  CHANGE(C)  RETRIEVE(R)  EXIT(X)
                         DELETE(D)  ADD NEW ORG ONLY(AS)

ACCESS SEGMENT: ______            TEXT: ____________

   SD      OCCE       SCH    GR thru GR     SCCE         ORGCODE

  ___    _______     ____   __       __      _     ___________________
  ___    _______     ____   __       __      _     ___________________
  ___    _______     ____   __       __      _     ___________________
  ___    _______     ____   __       __      _     ___________________
  ___    _______     ____   __       __      _     ___________________
  ___    _______     ____   __       __      _     ___________________
  ___    _______     ____   __       __      _     ___________________
  ___    _______     ____   __       __      _     ___________________
  ___    _______     ____   __       __      _     ___________________
  ___    _______     ____   __       __      _     ___________________
  ___    _______     ____   __       __      _     ___________________
  ___    _______     ____   __       __      _     ___________________
  ___    _______     ____   __       __      _     ___________________
  ___    _______     ____   __       __      _     ___________________
  ___    _______     ____   __       __      _     ___________________
```

<u>MENU ITEM DESCRIPTION</u>

(R) PROCEDURE ACTION:	A,C,D,R,X,AS	(LA02)
(R) ACCESS SEGMENT:	Name of SEGACCESS file (Same as SYSMAN2 Signon ORG)	(LA06)
(R) TEXT:	Access Segment Text	(LA12)

(At least one of the following must be present for A or C)

(O) SD:	Career Service Designation	(LA03)
(O) OCCE:	Occupational Series Code	(LA07)
(O) SCH:	Schedule Of Grade	(LA04)
(O) GR:	Individual Grade (One GR or from/thru range)	(LA02)
(O) SCCE:	Sub-Category Code	(LA01)
(O) ORGCODE:	Organization Code	(LA19)

4.3.2.5 OUTPUT

```
If Action is absent and/or Access Segment is absent
then error message (action and segment file name required).
endif;
```

NOTE: An edit in SYSMAN2 prohibits a SYSMAN2 ORG with a first character of numeric or "Z".

```
If SD, OCCE, SCH, GR, SCCE and ORGCODE EQ BLANK
   and Action NE 'R' or 'D'
then   - error message (one of the selection fields
         must be present)
endif;

If Action = 'A' and null Text
then  - error message (Text required for 'A' action)
endif;

If Action = 'A' and present Text
then  - review remaining fields for values (at least one must
        be present)
      - add Criteria to SELECTION file for the specified
        Access Segment.  Each line should be added as a single
        selection with each field separated by an asterisk(*).

If Action = 'C' and present SD, OCCE, SCH, GR, SCCE, ORGCODE or
   present TEXT (will replace all SELECTION Criteria with the
   new Criteria on the menu).
then  - review remaining menu lines for values.
      - Access Segment on menu = SELECTION file to be changed.
      - delete Criteria present in SELECTION file
      - add all Criteria entered on the menu to SELECTION
        file (each line as a separate value).

endif;
```

```
If Action = 'D'
then  - delete all Criteria from the SELECTION file for the
        Access Segment (SYSMAN2) specified on the menu.
endif;

If Action = 'AS'
then  - add new ORG selections only on menu to the SELECTION
        segment (do not delete or replace existing Criteria).
endif;

If Action = 'R'
then  - retrieve all Criteria from the SELECTION file for the
        Access Segment listed on the menu.  Print below the
        menu.
      - retrieve TEXT from the SELECTION file - print on the
        menu.
endif;

If Action = 'X'
then exit the procedure.
endif;
```

4.3.2.6 GENERAL PROGRAM CONSIDERATIONS

The PRCAESTB procedure will update a segmented file (SELECTION). The segment to be updated is specified in the Access Segment field of the menu. The segment must have been previously established and the segment name will be the same as the SYSMAN2 Signon Org the Component will use at signon time.

This procedure is for the exclusive use of the PRIM Data Base Manager to establish Criteria to be used in creating Access links for a Component.

4.3.2.7 DETAILED PROGRAM SPECIFICATIONS

```
DEC PRCAESTB
  BLANK                value ' '
  SELECTION            value 'NO'
  SARRAY               value 5
  $A                   single dimensional arrays
   $A(2)               array containing Action value
   $A(3)               array containing Access Segment
   $A(4)               array containing Text value
   $A(5) thru $A(109)  arrays contain Criteria data
enddec PRCAESTB

PROC PRCAESTB  (establish selection criteria)

  READ menu values into $A single demensional arrays
```

```
IF Signon Org ne DBMGRP
THEN - PRINT message stating for data base
       manager use only
 endif;

IF $A(2) EQ BLANK or $A(2) NE 'A' or $A(2) NE 'C' or
   $A(2) NE 'D' or $A(2) NE 'R' or $A(2) NE 'X'
   or $A(2) NE 'AS'
THEN  - PRINT message stating valid Action required
      - RUN EXIT-FOR-RESTART
ELSE
  IF $A(3) EQ BLANK
  THEN  - PRINT message stating Access Segment
          required
        - RUN EXIT-FOR-RESTART
  endif;
endif;

DO UNTIL SELECTION = 'YES' or SARRAY = 109
  IF $A(SARRAY) NE BLANK
  THEN  - SELECTION = 'YES'
  ELSE  - SARRAY = SARRAY + 1
  endif;
enddo;

IF SELECTION = 'NO' and $A(2) NE 'R' or 'D'
THEN  - PRINT error message requesting one selection
        field to be filled in
      - RUN EXIT-FOR-RESTART
endif;

IF $A(2) EQ 'A' and null $A(3)
THEN  - PRINT error message requesting Text for
        Action 'A'
      - RUN EXIT-FOR-RESTART
endif;

IF $A(2) EQ 'A' and present $A(3)
   (Add new to the SELECTION segment specified by
    Access Segment field on menu)
THEN  - RUN ACASE
      - RUN ADDCASE
      - RUN EXIT-FOR-MORE
endif;

IF $A(2) EQ 'AS'
   (Add additional ORG and/or TEXT to SELECTION segment
    specified by Access Segment field on menu)
THEN  - RUN ASCASE
      - RUN ADDCASE
      - RUN EXIT-FOR-MORE
endif;
```

```
IF $A(2) EQ 'C' (Change entire Criteria)
THEN  - RUN CCASE
      - RUN ADDCASE
      - RUN EXIT-FOR-MORE
endif;

IF $A(2) EQ 'D' (Delete all Criteria)
THEN  - RUN DCASE
      - RUN EXIT-FOR-MORE
endif;

IF $A(2) EQ 'R' (Retrieve current Criteria)
THEN  - RUN RCASE
      - RUN EXIT-FOR-MORE
endif;

IF $A(2) EQ 'X' (Exit the Menu)
THEN  - RUN EXIT-FOR-NOMORE
endif;

DEC ACASE
  STMT    temporary buffer area
  $A(3)   array containing Access Segment
  $A(4)   array containing Text
enddec ACASE

PROC ACASE
  (Build first part of an add statement using ADD NEW)
   (STMT = 'ADD NEW '//$A(3)//$QM//$A(3)//$QM//SLTEXT//
    $QM//$A(4)//$QM)
endproc ACASE;

DEC ASCASE
  BLANK    value ' '
  STMT     temporary buffer area
  $A(3)    array containing Access Segment
  $A(4)    array containing Text
enddec ASCASE

PROC ASCASE
  (Build first part of statement using ADD verb)
   (STMT = 'ADD '//$A(3)//$QM//$A(3)//$QM)

  IF $A(4) NE BLANK
  THEN  - continue add statement to include Text
          (STMT = STMT//' SLTEXT '//$QM//$A(4)//$QM)
  endif;
endproc ASCASE

DEC ADDCASE
  BLANK              value ' '
  STMT2              temporary buffer area
  A                  value 5
  B                  value 6
```

```
    C                    value 7
    D                    value 8
    E                    value 9
    F                    value 10
    G                    value 11
    $A(5 through 11)     1st line of Criteria data
    $A(12 through 18)    2nd line of Criteria data
    $A(19 through 25)    3rd line of Criteria data
    $A(26 through 32)    4th line of Criteria data
    $A(33 through 39)    5th line of Criteria data
    $A(40 through 46)    6th line of Criteria data
    $A(47 through 53)    7th line of Criteria data
    $A(54 through 60)    8th line of Criteria data
    $A(61 through 67)    9th line of Criteria data
    $A(68 through 74)    10th line of Criteria data
    $A(75 through 81)    11th line of Criteria data
    $A(82 through 88)    12th line of Criteria data
    $A(89 through 95)    13th line of Criteria data
    $A(96 through 102) 14th line of Criteria data
    $A(103 through 109) 15th line of Criteria data
  enddec ADDCASE

  PROC ADDCASE
    (Start at first array of each Criteria line and build
     a Criteria value.  Separate each element with an
     asterisk and enclose each Criteria value in quotes)

    INITIATE STMT2
          (STMT2 = ' SLCRITERIA ')

    DO for each Criteria line
      IF $A(A) NE BLANK or $A(B) NE BLANK or
         $A(C) NE BLANK or $A(D) NE BLANK or
         $A(E) NE BLANK or $A(F) NE BLANK or
         $A(G) NE BLANK
      THEN  - include in STMT2 (remove imbedded blanks)
             (STMT2 = STMT2//$QM//$A(A)//'*'//$A(B)//'*'
              $A(C)//'*'//$A(D)//'*'//$A(E)//'*'//$A(F)//
              '*'//$A(G)//$QM)
             (STMT2 = $EAB(STMT2)
      endif;

      INCREASE array values for next line
             A = A + 7, B = B + 7, C = C + 7, D = D + 7,
             E = E + 7, F = F + 7, G = G + 7
    enddo;

    EXECUTE complete statement for Action A, AS, or C
            (using statements formed in ACASE, ASCASE, or
             CCASE with criteria value formed in ADDCASE)

            (&STMT &STMT2)
 endproc ADDCASE
```

```
DEC CCASE
  BLANK                value ' '
  STMT                 temporary buffer area
  $A(3)                (array containing Access Segment)
  $A(4)                (array containing Text)
enddec CCASE

PROC CCASE

  START with Access Segment and build a change statement
        (STMT = 'CHANGE '//$A(3)//$QM//$A(3)//$QM)

  IF $A(4) NE BLANK
  THEN  - (continue statement to include Text)
          (STMT = STMT//' SLTEXT TO '//$QM//$A(4)//$QM)
  endif;

  EXTEND STMT to contain beginning of Criteria
          (STMT = STMT//' SLCRITERIA TO ')
endproc CCASE

DEC DCASE
  BLANK                value ' '
  STMT                 temporary buffer area
  $A(3)                array containing access segment
enddec DCASE

PROC DCASE

  START with Access Segment and build a delete statement
        (STMT = 'DELETE '//$A(3)//$QM//$A(3)//$QM
         //' SLCRITERIA ')

  EXECUTE &STMT
endproc DCASE

DEC RCASE
  Area       value 5
  $A(3)      array containing Access Segment
  TEXT       temporary area for SLTEXT value
  CRITERIA   temporary area for SLCRITERIA value
  SD         temporary area for SD value
  OCCE       temporary area for OCCE value
  SCH        temporary area for SCH value
  GRFM       temporary area for GRFM value
  GRTO       temporary area for GRTO value
  SCCE       temporary area for SCCE value
  ORGCODE    temporary area for ORGCODE value
enddec RCASE

PROC RCASE

  (Execute ACQUIRE statement for Text and Criteria
   for Access Segment on menu)
   (FOR &($A(3) '&($A(3)' ACQUIRE)

  GFAV SLTEXT
  TEXT = $W
  VOPRINT (*I4,TEXT)

  DO for all Criteria in Access Segment SELECTION file
    GET  SLCRITERIA   (using GFAV and GNAV)
    CRITERIA = $W
    SD  = $SBF(CRITERIA,0,1,'*')
    OCCE = $SBF(CRITERIA,1,1,'*')
    SCH =  $SBF(CRITERIA,2,1,'*')
    GRFM = $SBF(CRITERIA,3,1,'*')
    GRTO = $SBF(CRITERIA,4,1,'*')
    SCCE = $SBF(CRITERIA,5,1,'*')
    ORGCODE = $SBF(CRITERIA,6,1,'*')

    VOPRINT ('*I//$ALF(area)//','//SD)
    area = area + 1
    VOPRINT ('*I//$ALF(area)//','//OCCE)
    area = area + 1
    VOPRINT ('*I//$ALF(area)//','//SCH)
    area = area + 1
    VOPRINT ('*I//$ALF(area)//','//GRFM)
    area = area + 1
    VOPRINT ('*I//$ALF(area)//','//GRTO)
    area = area + 1
    VOPRINT ('*I//$ALF(area)//','//SCCE)
    area = area + 1
    VOPRINT ('*I//$ALF(area)//','//ORGCODE)
    area = area + 1
  enddo;
endproc RCASE

DEC EXIT-FOR-RESTART
  $A(2)             array containing Action
  $A(3)             array containing Access Segment
  $A(4)             array containing Text
  $A(5-109)         arrays containing Criteria data
  BLANK             value ' '
enddec EXIT-FOR-RESTART

PROC EXIT-FOR-RESTART

  IF $A(2) EQ BLANK or $A(2) NE 'A' or $A(2) NE 'C' or
     $A(2) NE 'D' or $A(2) NE 'R' or $A(2) NE 'X'
     or $A(2) NE 'AS'
  THEN  - RELOCATE cursor to Action (*RL2)
  ELSE
    IF $A(3) EQ BLANK
    THEN  - RELOCATE cursor to Access Segment (*RL3)
    ELSE
      IF SELECTION = 'NO'
      THEN  - RELOCATE cursor to first Criteria (*RL5)
```

```
        ELSE
          IF $A(2) EQ 'A' and $A(4) EQ BLANK
          THEN  - RELOCATE cursor to Text (*RL4)
          endif;
        endif;
      endif;
    endif;
 endproc EXIT-FOR-RESTART

 PROC EXIT-FOR-MORE
   VOPRINT '*C'                clears menu
   VOPRINT '*I1,E PRCAESTB'    establishes value in $A(1)
   VOPRINT '*+B'               big buffer
   VOPRINT '*RL2'              relocate cursor to $A(2)
 endproc EXIT-FOR-MORE

 PROC EXIT-FOR-NOMORE
   VOPRINT  '*-F'              exit format mode
   VOPRINT  '*C'               clear entire screen
 endproc EXIT-FOR-NOMORE

 endproc PRCAESTB
```

4.3.3 PRCAUPDT - COMPONENT ACCESS UPDATE

4.3.3.1 PURPOSE

This BATCHGIM II program is designed to gather all INTERFACE changes that will affect PRIM data access. HRS2 INTERFACE changes that would affect PRIM would be changes to SSNOR, SD, ORGCODE, SCHEDULE, GRADE, SUBCAT CODE, and/or OCCUPATIONAL SERIES CODE as well as any other data that may be required. See flow diagram in Figure 8 of the Appendix.

4.3.3.2 GENERAL INFORMATION

Proc Name	: PRCAUPDT
Language Used	: PLI (BATCHGIM II)
Initiated By	: Production Division - DBCC
Frequency Of Execution	: Will be done nightly as part of the HRS2/PRIM Interface.
Input	: ACCESSCTL file (DLID) HRS2 INTERFACE file (all data)
Output	: Extract disk file with the required HRS2 INTERFACE data.

DSN= ______________________________

4.3.3.3 FUNCTIONAL DESCRIPTION

This program will extract changes made to PERSIGN data in the HRS2 Data Base that also apply to PRIM. As updates are made to the HRS2 PERSIGN data, entries are made in two other files (ACCESSCTL and INTERFACE). The ACCESSCTL file is merely an access file and is used by all external data bases receiving PERSIGN data from HRS2. The INTERFACE file contains the actual from/to change that was made during the nightly update.

The PRCAUPDT program will scan the ACCESSCTL file in the HRS2 Data Base for those records with a value in ACCPRIM. The DL/ID of these records will be retained by the program. After a complete scan of the ACCESSCTL file, the retained DL/ID's will be used to retrieve data from the HRS2 INTERFACE file. This data will be written to disk and will be used on the PRIM Data Base by the PRCHGLOAD procedure to update the PRIM PERSIGN or PRIMSEP file and the PRIM INTERFACE file. The final step of the PRCAUPDT program will delete the ACCPRIM values from the ACCESSCTL file to avoid redundant data moves.

4.3.3.4 INPUT

Input to this program will be all data from the HRS2 INTERFACE Data file.

4.3.3.5 OUTPUT

The data extracted by this program will be placed on a disk. The disk will contain the DL/ID of the HRS2 INTERFACE record and all of the INTERFACE elements.

4.3.3.6 GENERAL PROGRAM CONSIDERATIONS

A BATCHGIM II program used to extract the INTERFACE data from HRS2 to a disk. BATCHGIM II is being used in order to be consistent with other systems using HRS2 INTERFACE data.

When an external system wants to access HRS2 INTERFACE data it must first access the ACCESSCTL file to know which INTERFACE record have not already been seen. The ACCESSCTL file in HRS2 uses the same DL/ID values as INTERFACE file and contains flags for each system Interfacing with HRS2 requiring INTERFACE data. PRIM flags will be in the ACCPRIM field. Once PRIM accesses an INTERFACE record, the access flag must be removed from the ACCESSCTL file to show that INTERFACE record has been reviewed and is no longer needed by PRIM.

4.3.3.7 DETAILED PROGRAM SPECIFICATIONS

```
 DEC PRCAUPDT
   ACCESSCTL
     ACCPRIM            access flag
   $A                   single dimensional array
   INTERFACE            data file
   LVL                  value 1
   STMT                 temporary buffer area
 enddec PRCAUPDT

 PROC PRCAUPDT

   ACCESS HRS2 Data Base

   ORDER ACCESSCTL

   DO for all ACCESSCTL records
     (Execute ACQUIRE statement for ACCESSCTL
     records with a value in ACCPRIM)
     (FOR ACCESSCTL WITH PRESENT ACCPRIM ACQUIRE)

     $A = ACCESSCTL ($W)    (load into next available array)
   enddo;

   IF no records found with present ACCPRIM
   THEN  - print message that no records were found to
           move to PRIM
   ELSE
     (build extract statement to include all filled arrays)
     DO for all filled arrays
       IF $A(LVL) NE $AM
       THEN  - INCLUDE in Extract statement)
               (STMT = STMT//$QM//$A(LVL)//$QM)
       endif;
       LVL = LVL + 1
     enddo;
   IF STMT NE BLANK
   THEN - (execute statement to place data on disk)
          . $DISK,$DSN=G3P.A3176600.....,$VOL=LOCATE .
          FOR INTERFACE &STMT EXTRACT
        - (delete ACCPRIM flags from ACCESSCTL file for
           all filled arrays)
          FOR ACCESSCTL &STMT DELETE ACCPRIM
        - PRINT message notifying of successful completion
          and files are ready for load to PRIM
   endif;

endproc PRCAUPDT
```

4.3.4 PRCHGLOAD - COMPONENT ACCESS CHANGES

4.3.4.1 PURPOSE

This procedure is designed to use the output from the PRCAUPDT program to make updates to the PRIM PERSIGN or PRIMSEP file, update the Access INDEX files through XBRIDGES, and make an entry in the PRIM INTERINDX and INTERFACE files. See flow diagram in Figure 8 of the Appendix.

4.3.4.2 GENERAL INFORMATION

Proc Name	: PRCHGLOAD
Language Used	: GIM POL
Initiated By	: Production Division - DBCC
Frequency Of Execution	: Will be done nightly as part of the HRS2/PRIM Interface.
Input	: Disk produced by PRCAUPDT. DSN= ________________________.
Output	: Updates will be made to the PRIM PERSIGN, PRIMSEP, INDEX files (SDINDX,ORGINDX,SCCEINDX, OCCEINDX,SSDINDX,SORGINDX,SSCCEINDX, SOCCEINDX), INTERINDX, and PRIM INTERFACE.

4.3.4.3 FUNCTIONAL DESCRIPTION

The PRCHGLOAD procedure uses the disk file created by the PRCAUPDT procedure. The disk contains updates made to PERSIGN data in HRS2 which pertain to PRIM, as well as, data which will be required for the procedure to accurately analyze the updates. These records are used to build an update statement against the PRIM PERSIGN or PRIMSEP files, make an entry in the PRIM INTERINDX file and load the INTERFACE records into the PRIM INTERFACE file. When updates are made to the PRIM PERSIGN file XBRIDGES are evoked which cause updates to INDEX files. These INDEX files are used by the PRCRTLINK procedure to establish daily links for Components at Signon time.

4.3.4.4 INPUT

No menu will be required for this procedure. The following statement will initiate the procedure: E PRCHGLOAD

4.3.4.5 OUTPUT

The output from the PRCHGLOAD procedures will be determined by the following specifications:

```
 DOWHILE the INTERFACE data is read from the disk:

   - Perform unique processing for the ZZ actions. (TBR)

 If the record contains an SSNOR change (both segments of
    IFCSIGN are different and the first segment of IFCSIGN
    NE $$$$$$$$$)
 then  - change PERSIGN record
       - change SSNOR in SEGACCESS file (SYSMAN2+C) (TBR Release 2)
 endif;
 If record contains PRIM PERSIGN updates
 then  - update PERSIGN record (which evokes XBRIDGING and updates
         INDEX files where applicable.
       - add record to PRIM INTERINDX with INPURGE (RUNDATE + 30).
 endif;

 If the record contains CEILING 'X' (separated employee)
    (IFCCEIL newside EQ 'X' oldside NE 'X')
 then  - copy PRIM PERSIGN record to PRIMSEP
       - add SD, OCCE, SCCE, ORG separately to PRIMSEP
         (to evoke XBRIDGING and update INDEX files where
         applicable).
       - add reason for separation from INTERFACE (IFISEPCODE)
         to PRIMSEPCODE, separation date (IFCDOA(newside)) to
         PRIMSEPDTE.
       - add record to PRIM INTERINDX with INPURGE (RUNDATE + 30).
 endif;

 If the record contains CEILING 'X' or 'G' (updates to separated
    record)
 then  - update PRIMSEP RECORD
       - add record to PRIM INTERINDX with INPURGE (RUNDATE + 30).
 endif;

 If the record contains an old CEILING 'X' and new CEILING
    NE 'X' or 'G'  (cancellation of an immediate separation)
 then  - add record to PRIM PERSIGN
       - add record to PRIM INTERINDX with INPURGE (RUNDATE + 30).
 endif;

enddowhile;

 If all updates are completed
 then  - load all INTERFACE records into the PRIM
         INTERFACE file.
 endif;
```

4.3.4.6 GENERAL PROGRAM CONSIDERATIONS

Using INTERFACE data placed on a disk by the PRCAUPDT procedure, this procedure must analyze each INTERFACE field, pertinent to PERSIGN, and make the appropriate changes to the PERSIGN or PRIMSEP file on PRIM.

The process order will be guaranteed by doing an ORDER of the ACCESSCTL file before acquiring any records from it. The ACCESSCTL file is a file on HRS2 used to show which records have been accessed or have not been accessed by a system interfacing with HRS2.

Changes to some PRIM PERSIGN fields will envoke XBRIDGES and update INDEX files as noted below:

PERSD	SDINDX
PERORGCODE	ORGINDX
PEROCCE	OCCEINDX
PERSCCE	SCCEINDX

Changes to some PRIMSEP fields will envoke XBRIDGES and update INDEX files as noted below:

PERSD	SSDINDX
PERORGCODE	SORGINDX
PEROCCE	SOCCEINDX
PERSCCE	SSCCEINDX

The HRS2 INTERFACE record will be added to the PRIM INTERFACE file, an entry with an INPURGE date will be added to INTERINDX, and an update date will be added to SYSER to notify components of the most recent HRS2/PRIM update.

4.3.4.7 DETAILED PROGRAM SPECICATIONS

```
 DEC PRCHGLOAD
 CSIGN    Input value from disk, shows SSNOR change
 CCEIL    Input value from disk,
            Shows record moving from Active to Separated
 NEWREC     value received from BLDSTMT (CCEIL)
enddec PRCHGLOAD


PROC PRCHGLOAD

  RUN SETLVL

  DO FOR ALL INTERFACE records on Disk

    RUN READREC

     CSIGN NE BLANK    (SSNOR CHANGE)
    THEN   RUN SSNCHG
    endif;

     $SBF(CCEIL,1,1,'*') EQ 'X'  (new side)
    THEN   RUN NEWSEP
    endif;

    RUN BLDSTMT

     NEWREC = 'YES'       (NEW EOD)
    THEN   RUN ADDNEW
    endif;

    RUN ADDREC
  enddo;

PROC SETLVL

  FILE   RECORD
  LVL01  INTER,CS
  LVL02  CACTREQ,C,5
  LVL02  CAFF,C,7
  LVL02  CAFFTXTL,C,49
  LVL02  CANF,C,5
  LVL02  CANO,C,7
  LVL02  CANTYST,C,3
  LVL02  CAORG,C,9
  LVL02  CAPNTE,C,13
  LVL02  CASALARY,C,15
  LVL02  CCEIL,C,3
  LVL02  CCIT,C,3
  LVL02  CCOVDPT,C,7
  LVL02  CCOVDPTXT,C,17
  LVL02  CCOVDT,C,13
  LVL02  CCOVORG,C,5
  LVL02  CCOVORGTXT,C,25
  LVL02  CCSEOD,C,13
  LVL02  CDETC,C,5
  LVL02  CDETONTE,C,13
  LVL02  CDETTXTS,C,25
  LVL02  CDEVC,C,5
  LVL02  CDEVDI,C,13
  LVL02  CDEVLAST,C,13
  LVL02  CDEVNTE,C,13
  LVL02  CDOA,C,13
  LVL02  CDOB,C,13
  LVL02  CDOG,C,13
  LVL02  CETP,C,13
  LVL02  CFLSA,C,3
  LVL02  CFLSADT,C,13
  LVL02  CGR,C,5
  LVL02  CGRPRT1,C,11
  LVL02  CGSDTXTS,C,11
  LVL02  CHQTXTS,C,9
  LVL02  CHRS,C,7
```

```
LVL02   CLCD,C,13
LVL02   CLEI,C,13
LVL02   CNAMEOR,C,49
LVL02   CNAMEOR3,C,7
LVL02   CNOA,C,5
LVL02   CNSCA,C,3
LVL02   CNSCADT,C,13
LVL02   CNTY,C,3
LVL02   COCCE,C,15
LVL02   COCCETXT,C,41
LVL02   COCCFAM,C,5
LVL02   COCCSUF,C,11
LVL02   CONAMEOR3,C,7
LVL02   CORGCODE,C,39
LVL02   CORGDTIN,C,13
LVL02   COVERLAP,C,19
LVL02   COVLAPING,C,19
LVL02   COVNTE,C,13
LVL02   CPAYB,C,5
LVL02   CPAYDT,C,13
LVL02   CPAYDTC,C,3
LVL02   CPDATE,C,13
LVL02   CPGVT,C,3
LVL02   CPOSNO,C,11
LVL02   CPRA,C,3
LVL02   CPRADI,C,13
LVL02   CPRANTE,C,13
LVL02   CPRMNTE,C,13
LVL02   CPROJNO,C,13
LVL02   CPWGI,C,13
LVL02   CRACE,C,3
LVL02   CRESV1,C,11
LVL02   CRESV2,C,11
LVL02   CRESV3,C,11
LVL02   CRESV4,C,21
LVL02   CRESV5,C,21
LVL02   CRETDI,C,13
LVL02   CRETNTE,C,13
LVL02   CRTMT,C,3
LVL02   CRTMTTXT,C,13
LVL02   CSALARY,C,15
LVL02   CSCCE,C,3
LVL02   CSCCESK,C,3
LVL02   CSCCETXTS,C,9
LVL02   CSCD,C,13
LVL02   CSCH,C,9
LVL02   CSCHGRSK1,C,5
LVL02   CSCHGRSK2,C,5
LVL02   CSCHPRT1,C,9
LVL02   CSD,C,7
LVL02   CSECCL,C,3
LVL02   CSEQ,C,7
LVL02   CSERIAL,C,15
LVL02   CSEX,C,3
```

```
LVL02  CSFN,C,15
LVL02  CSIGN,C,19
LVL02  CSREF,C,5
LVL02  CSSNOTH,C,19
LVL02  CSTAN,C,13
LVL02  CSTANAREA,C,5
LVL02  CSTANTXT,C,51
LVL02  CSTEP2,C,5
LVL02  CSTIND,C,3
LVL02  CSTRCTR,C,5
LVL02  CTOA,C,5
LVL02  CTOUR,C,3
LVL02  CVET,C,3
LVL02  CWGIE,C,3
LVL02  GCUTOFFDT,C,6
LVL02  GPROJUPDTD,C,1
LVL02  IACTREQ,C,2
LVL02  IAFF,C,3
LVL02  IAL,C,2
LVL02  IANO,C,3
LVL02  IANPRPAY,C,2
LVL02  IANTYSAL,C,5
LVL02  IANTYST,C,1
LVL02  IAORG,C,4
LVL02  IAPNTE,C,6
LVL02  ICCANO,C,3
LVL02  ICCNOA,C,2
LVL02  ICCTOA,C,12
LVL02  ICIT,C,1
LVL02  ICSEOD,C,6
LVL02  ICTPCLASS,C,3
LVL02  IDETC,C,2
LVL02  IDETONTE,C,6
LVL02  IDEVC,C,2
LVL02  IDEVDI,C,6
LVL02  IDEVLAST,C,6
LVL02  IDEVNTE,C,6
LVL02  IDOA,C,6
LVL02  IDOB,C,6
LVL02  IDOG,C,6
LVL02  IETP,C,6
LVL02  IFLSA,C,1
LVL02  IFLSADT,C,6
LVL02  IGR,C,2
LVL02  IHQ,C,1
LVL02  IHRS,C,3
LVL02  IINDT,C,6
LVL02  IINORIGDT,C,6
LVL02  ILCD,C,6
LVL02  ILEI,C,6
LVL02  ILWOPSTRT,C,4
LVL02  INAMEOR,C,24
LVL02  INAMEOR3,C,3
LVL02  INCPP,C,2
```

```
LVL02   INEWIDN,C,9
LVL02   INSCA,C,1
LVL02   INSCADT,C,6
LVL02   INTY,C,1
LVL02   IOCCE,C,7
LVL02   IOCCSUF,C,5
LVL02   IOPERATOR,C,10
LVL02   IORGCODE,C,09
LVL02   IORGDTIN,C,6
LVL02   IOUTSK,C,3
LVL02   IOVERLAP,C,9
LVL02   IOVNTE,C,6
LVL02   IOVRD,C,2
LVL02   IPAYB,C,2
LVL02   IPGVT,C,1
LVL02   IPOSNO,C,5
LVL02   IPRA,C,1
LVL02   IPRADI,C,6
LVL02   IPRANTE,C,6
LVL02   IPRMNTE,C,6
LVL02   IPROJNO,C,6
LVL02   IPWGI,C,6
LVL02   IRACE,C,1
LVL02   IREMC,C,2
LVL02   IREMNS,C,69
LVL02   IRESV1,C,5
LVL02   IRESV2,C,5
LVL02   IRESV3,C,5
LVL02   IRESV4,C,10
LVL02   IRESV5,C,10
LVL02   IRETDI,C,6
LVL02   IRETNTE,C,6
LVL02   IRTMT,C,1
LVL02   ISALARY,C,7
LVL02   ISCCE,C,1
LVL02   ISCD,C,6
LVL02   ISCH,C,4
LVL02   ISD,C,3
LVL02   ISECCL,C,1
LVL02   ISEPCODE,C,10
LVL02   ISEPCOMP,C,8
LVL02   ISERIAL,C,7
LVL02   ISEX,C,1
LVL02   ISFN,C,7
LVL02   ISIGN,C,9
LVL02   ISREF,C,2
LVL02   ISSNOTH,C,9
LVL02   ISTAN,C,6
LVL02   ISTEP2,C,2
LVL02   ISUSPEN,C,2
LVL02   ITIDN,C,9
LVL02   ITNAME,C,24
LVL02   ITOA,C,12
LVL02   ITOASK,C,3
```

```
  LVL02  ITOUR,C,1
  LVL02  ITRANS,C,10
  LVL02  IVET,C,1
  LVL02  IWGIE,C,1
  LVL02  SAFF,C,3
  LVL02  SANF,C,2
  LVL02  SANO,C,3
  LVL02  SAPNTE,C,6
  LVL02  SCEIL,C,1
  LVL02  SDEVC,C,2
  LVL02  SDOB,C,6
  LVL02  SDOG,C,6
  LVL02  SGR,C,2
  LVL02  SLEI,C,6
  LVL02  SNAMEOR,C,24
  LVL02  SPAYB,C,2
  LVL02  SPGVT,C,1
  LVL02  SPROJNO,C,6
  LVL02  SRTMT,C,1
  LVL02  SSALARY,C,7
  LVL02  SSCD,C,6
  LVL02  SSCH,C,4
  LVL02  SSERIAL,C,7
  LVL02  SSFN,C,7
  LVL02  SSTEP2,C,2
  LVL02  STOUR,C,1
  LVL02  INTERFACE,C,22
endproc SETLVL

 DEC READREC
 $DSN       value = permanent data set name
 $VOL       value = locate
enddec READREC

PROC READREC

OPEN     RECORD,INPUT,$DSN=DSN,$VOL='LOCATE',$DISK='DISK'

  DO for INPUT record
    GET Corresponding PERSIGN record
      FOR PERSIGN '&ISIGN' ACQUIRE PERSIGN

     IF record found in PERSIGN
    THEN   FILE = 'PERSIGN'
    endif;

    GET Corresponding PRIMSEP record
      FOR PRIMSEP '&ISIGN' ACQUIRE PRIMSEP

     IF record found in PRIMSEP
    THEN   FILE = 'PRIMSEP'
    ELSE   FILE = 'PERSIGN'
    endif;
  enddo;
```

```
endproc READREC

DEC BLDSTMT
 BLANK       value ' '
 $A          array area containing fields to be added
 $C          array area containing fields to be changed
 $D          array area containing fields to be deleted
 $E          array area containing add to INTERFACE
 STMT        value ' '
             area contains SEPCODE and SEPDTE for PRIMSEP ONLY
 OLD         old value of IFC item
 NEW         new value of IFC item
 NEWREC      value 'NO' - flags new EOD - see CCEIL
enddec BLDSTMT

PROC BLDSTMT

 use WHEN operator for this PROC

  IF CACTREQ NE BLANK
  THEN  OLD=$SBF(CACTREQ,0,1,'*')
        NEW=$SBF(CACTREQ,1,1,'*')
        $E=' CACTREQ '//$QM//CACTREQ//$QM

     $SBF(OLD,0,1) NE '$' and $SBF(NEW,0,1) NE '$'
    THEN $C=' PERACTREQ '//$QM//OLD//$QM//' TO '
         //$QM//NEW//$QM
    endif;

    IF $SBF(OLD,0,1) EQ '$'
    THEN $A=' PERACTREQ '//$QM//NEW//$QM
    endif;

    IF $SBF(NEW,0,1) EQ '$'
    THEN $D=' PERACTREQ '//$QM//OLD//$QM
    endif;
  endif;

  IF CAFF NE BLANK
  THEN
    OLD=$SBF(CAFF,0,1,'*')
    NEW=$SBF(CAFF,1,1,'*')
    $E=' CAFF '//$QM//CAFF//$QM

    IF $SBF(OLD,0,1) NE '$' and $SBF(NEW,0,1) NE '$'
    THEN $C=' PERAFF TO '//$QM//NEW//$QM
    endif;

    IF $SBF(OLD,0,1) EQ '$'
    THEN $A=' PERAFF '//$QM//NEW//$QM
    endif;

    IF $SBF(NEW,0,1) EQ '$' (only occurs if ITOA = 'ZZ')
    THEN $D=' PERAFF '//$QM//OLD//$QM
  endif;
endif;

IF CAFFTXTL NE BLANK
THEN
  OLD=$SBF(CAFFTXTL,0,1,'*')
  NEW=$SBF(CAFFTXTL,1,1,'*')
  $E=' CAFFTXTL '//$QM//CAFFTXTL//$QM

  IF $SBF(OLD,0,1) NE '$' and $SBF(NEW,0,1) NE '$'
  THEN $C=' PERAFFTXTL TO '//$QM//NEW//$QM
  endif;

  IF $SBF(OLD,0,1) EQ '$'
  THEN $A=' PERAFFTXTL '//$QM//NEW//$QM
  endif;

  IF $SBF(NEW,0,1) EQ '$' (only occurs if ITOA = 'ZZ')
  THEN $D=' PERAFFTXTL '//$QM//OLD//$QM
  endif;
endif;

IF CANF NE BLANK
THEN
  OLD=$SBF(CANF,0,1,'*')
  NEW=$SBF(CANF,1,1,'*')
  $E=' CANF '//$QM//CANF//$QM

  IF $SBF(OLD,0,1) NE '$' and $SBF(NEW,0,1) NE '$'
  THEN $C=' PERANF TO '//$QM//NEW//$QM
  endif;

  IF $SBF(OLD,0,1) EQ '$'
  THEN $A=' PERANF '//$QM//NEW//$QM
  endif;

  IF $SBF(NEW,0,1) EQ '$'
  THEN $D=' PERANF '//$QM//OLD//$QM
  endif;
endif;

IF CANO NE BLANK
THEN
  OLD=$SBF(CANO,0,1,'*')
  NEW=$SBF(CANO,1,1,'*')
  $E=' CANO '//$QM//CANO//$QM

  IF $SBF(OLD,0,1) NE '$' and $SBF(NEW,0,1) NE '$'
  THEN $C=' PERANO TO '//$QM//NEW//$QM
  endif;

  IF $SBF(OLD,0,1) EQ '$'
  THEN $A=' PERANO '//$QM//NEW//$QM
  endif;
  IF $SBF(NEW,0,1) EQ '$' (only occurs if ITOA = 'ZZ')
  THEN $D=' PERANO '//$QM//OLD//$QM
  endif;
endif;

IF CANTYST NE BLANK
THEN
  OLD=$SBF(CANTYST,0,1,'*')
  NEW=$SBF(CANTYST,1,1,'*')
  $E=' CANTYST '//$QM//CANTYST//$QM

  IF $SBF(OLD,0,1) NE '$' and $SBF(NEW,0,1) NE '$'
  THEN $C=' PERANTYST TO '//$QM//NEW//$QM
  endif;

  IF $SBF(OLD,0,1) EQ '$'
  THEN $A=' PERANTYST '//$QM//NEW//$QM
  endif;

  IF $SBF(NEW,0,1) EQ '$'
  THEN $D=' PERANTYST '//$QM//OLD//$QM
  endif;
endif;

IF CAORG NE BLANK
THEN
  OLD=$SBF(CAORG,0,1,'*')
  NEW=$SBF(CAORG,1,1,'*')
  $E=' CAORG '//$QM//CAORG//$QM

  IF $SBF(OLD,0,1) NE '$' and $SBF(NEW,0,1) NE '$'
  THEN $C=' PERAORG TO '//$QM//NEW//$QM
  endif;

  IF $SBF(OLD,0,1) EQ '$'
  THEN $A=' PERAORG '//$QM//NEW//$QM
  endif;

  IF $SBF(NEW,0,1) EQ '$' (only occurs if ITOA = 'ZZ')
  THEN $D=' PERAORG '//$QM//OLD//$QM
  endif;
endif;

IF CAPNTE NE BLANK
THEN
  OLD=$SBF(CAPNTE,0,1,'*')
  NEW=$SBF(CAPNTE,1,1,'*')
  $E=' CAPNTE '//$QM//CAPNTE//$QM

  IF $SBF(OLD,0,1) NE '$' and $SBF(NEW,0,1) NE '$'
  THEN $C=' PERAPNTE TO '//$QM//NEW//$QM
  endif;
  IF $SBF(OLD,0,1) EQ '$'
  THEN $A=' PERAPNTE '//$QM//NEW//$QM
  endif;

  IF $SBF(NEW,0,1) EQ '$'
  THEN $D=' PERAPNTE '//$QM//OLD//$QM
  endif;
endif;

IF CASALARY NE BLANK
THEN
  OLD=$SBF(CASALARY,0,1,'*')
  NEW=$SBF(CASALARY,1,1,'*')
  $E=' CASALARY '//$QM//CASALARY//$QM

  IF $SBF(OLD,0,1) NE '$' and $SBF(NEW,0,1) NE '$'
  THEN $C=' PERASALARY TO '//$QM//NEW//$QM
  endif;

  IF $SBF(OLD,0,1) EQ '$'
  THEN $A=' PERASALARY '//$QM//NEW//$QM
  endif;

  IF $SBF(NEW,0,1) EQ '$'
  THEN $D=' PERASALARY '//$QM//OLD//$QM
  endif;
endif;

IF CCEIL NE BLANK
THEN
  OLD=$SBF(CCEIL,0,1,'*')
  NEW=$SBF(CCEIL,1,1,'*')
  $E=' CCEIL '//$QM//CCEIL//$QM

  IF $SBF(OLD,0,1) NE '$' and $SBF(NEW,0,1) NE '$'
     and NEW NE 'X' and NEW NE 'G'
  THEN $C=' PERCEIL TO '//$QM//NEW//$QM
  endif;

  IF $SBF(OLD,0,1) EQ '$'
  THEN $A=' PERCEIL '//$QM//NEW//$QM
       NEWREC = 'YES'        (new EOD)
  endif;

  IF $SBF(NEW,0,1) EQ '$' (only occurs if ITOA = 'ZZ')
  THEN $D=' PERCEIL '//$QM//OLD//$QM
  endif;
endif;

IF CCIT NE BLANK
THEN
  OLD=$SBF(CCIT,0,1,'*')
  NEW=$SBF(CCIT,1,1,'*')
  $E=' CCIT '//$QM//CCIT//$QM
```

```
    IF $SBF(OLD,0,1) NE '$' and $SBF(NEW,0,1) NE '$'
    THEN $C=' PERCIT TO '//$QM//NEW//$QM
    endif;

    IF $SBF(OLD,0,1) EQ '$'
    THEN $A=' PERCIT '//$QM//NEW//$QM
    endif;

    IF $SBF(NEW,0,1) EQ '$' (only occurs if ITOA = 'ZZ')
    THEN $D=' PERCIT '//$QM//OLD//$QM
    endif;
  endif;

  IF CCOVDPT NE BLANK
  THEN
    OLD=$SBF(CCOVDPT,0,1,'*')
    NEW=$SBF(CCOVDPT,1,1,'*')
    $E=' CCOVDPT '//$QM//CCOVDPT//$QM

    IF $SBF(OLD,0,1) NE '$' and $SBF(NEW,0,1) NE '$'
    THEN $C=' PERCOVDPT TO '//$QM//NEW//$QM
    endif;

    IF $SBF(OLD,0,1) EQ '$'
    THEN $A=' PERCOVDPT '//$QM//NEW//$QM
    endif;

    IF $SBF(NEW,0,1) EQ '$'
    THEN $D=' PERCOVDPT '//$QM//OLD//$QM
    endif;
  endif;

  IF CCOVDPTXT NE BLANK
  THEN
    OLD=$SBF(CCOVDPTXT,0,1,'*')
    NEW=$SBF(CCOVDPTXT,1,1,'*')
    $E=' CCOVDPTXT '//$QM//CCOVDPTXT//$QM

    IF $SBF(OLD,0,1) NE '$' and $SBF(NEW,0,1) NE '$'
    THEN $C=' PERCOVDPTXT TO '//$QM//NEW//$QM
    endif;

    IF $SBF(OLD,0,1) EQ '$'
    THEN $A=' PERCOVDPTXT '//$QM//NEW//$QM
    endif;

    IF $SBF(NEW,0,1) EQ '$'
    THEN $D=' PERCOVDPTXT '//$QM//OLD//$QM
    endif;
  endif;

  IF CCOVDT NE BLANK
  THEN
  OLD=$SBF(CCOVDT,0,1,'*')
  NEW=$SBF(CCOVDT,1,1,'*')
  $E=' CCOVDT '//$QM//CCOVDT//$QM

  IF $SBF(OLD,0,1) NE '$' and $SBF(NEW,0,1) NE '$'
  THEN $C=' PERCOVDT TO '//$QM//NEW//$QM
  endif;

  IF $SBF(OLD,0,1) EQ '$'
  THEN $A=' PERCOVDT '//$QM//NEW//$QM
  endif;

  IF $SBF(NEW,0,1) EQ '$'
  THEN $D=' PERCOVDT '//$QM//OLD//$QM
  endif;
endif;

IF CCOVORG NE BLANK
THEN
  OLD=$SBF(CCOVORG,0,1,'*')
  NEW=$SBF(CCOVORG,1,1,'*')
  $E=' CCOVORG '//$QM//CCOVORG//$QM

  IF $SBF(OLD,0,1) NE '$' and $SBF(NEW,0,1) NE '$'
  THEN $C=' PERCOVORG TO '//$QM//NEW//$QM
  endif;

  IF $SBF(OLD,0,1) EQ '$'
  THEN $A=' PERCOVORG '//$QM//NEW//$QM
  endif;

  IF $SBF(NEW,0,1) EQ '$'
  THEN $D=' PERCOVORG '//$QM//OLD//$QM
  endif;
endif;

IF CCOVORGTXT NE BLANK
THEN
  OLD=$SBF(CCOVORGTXT,0,1,'*')
  NEW=$SBF(CCOVORGTXT,1,1,'*')
  $E=' CCOVORGTXT '//$QM//CCOVORGTXT//$QM

  IF $SBF(OLD,0,1) NE '$' and $SBF(NEW,0,1) NE '$'
  THEN $C=' PERCOVORGTXT TO '//$QM//NEW//$QM
  endif;

  IF $SBF(OLD,0,1) EQ '$'
  THEN $A=' PERCOVORGTXT '//$QM//NEW//$QM
  endif;

  IF $SBF(NEW,0,1) EQ '$'
  THEN $D=' PERCOVORGTXT '//$QM//OLD//$QM
  endif;
endif;
```

```
IF CCSEOD NE BLANK
THEN
  OLD=$SBF(CCSEOD,0,1,'*')
  NEW=$SBF(CCSEOD,1,1,'*')
  $E=' CCSEOD '//$QM//CCSEOD//$QM

  IF $SBF(OLD,0,1) NE '$' and $SBF(NEW,0,1) NE '$'
  THEN $C=' PERCSEOD TO '//$QM//NEW//$QM
  endif;

  IF $SBF(OLD,0,1) EQ '$'
  THEN $A=' PERCSEOD '//$QM//NEW//$QM
  endif;

  IF $SBF(NEW,0,1) EQ '$' (only occurs if ITOA = 'ZZ')
  THEN $D=' PERCSEOD '//$QM//OLD//$QM
  endif;
endif;

IF CDETC NE BLANK
THEN
  OLD=$SBF(CDETC,0,1,'*')
  NEW=$SBF(CDETC,1,1,'*')
  $E=' CDETC '//$QM//CDETC//$QM

  IF $SBF(OLD,0,1) NE '$' and $SBF(NEW,0,1) NE '$'
  THEN $C=' PERDETC TO '//$QM//NEW//$QM
  endif;

  IF $SBF(OLD,0,1) EQ '$'
  THEN $A=' PERDETC '//$QM//NEW//$QM
  endif;

  IF $SBF(NEW,0,1) EQ '$'
  THEN $D=' PERDETC '//$QM//OLD//$QM
  endif;
endif;

IF CDETONTE NE BLANK
THEN
  OLD=$SBF(CDETONTE,0,1,'*')
  NEW=$SBF(CDETONTE,1,1,'*')
  $E=' CDETONTE '//$QM//CDETONTE//$QM

  IF $SBF(OLD,0,1) NE '$' and $SBF(NEW,0,1) NE '$'
  THEN $C=' PERDETONTE TO '//$QM//NEW//$QM
  endif;

  IF $SBF(OLD,0,1) EQ '$'
  THEN $A=' PERDETONTE '//$QM//NEW//$QM
  endif;

  IF $SBF(NEW,0,1) EQ '$'
```

```
    THEN $D=' PERDETONTE '//$QM//OLD//$QM
    endif;
  endif;

  IF CDETTXTS NE BLANK
  THEN
    OLD=$SBF(CDETTXTS,0,1,'*')
    NEW=$SBF(CDETTXTS,1,1,'*')
    $E=' CDETTXTS '//$QM//CDETTXTS//$QM

    IF $SBF(OLD,0,1) NE '$' and $SBF(NEW,0,1) NE '$'
    THEN $C=' PERDETTXTS TO '//$QM//NEW//$QM
    endif;

    IF $SBF(OLD,0,1) EQ '$'
    THEN $A=' PERDETTXTS '//$QM//NEW//$QM
    endif;

    IF $SBF(NEW,0,1) EQ '$'
    THEN $D=' PERDETTXTS '//$QM//OLD//$QM
    endif;
  endif;

  IF CDEVC NE BLANK
  THEN
    OLD=$SBF(CDEVC,0,1,'*')
    NEW=$SBF(CDEVC,1,1,'*')
    $E=' CDEVC '//$QM//CDEVC//$QM

    IF $SBF(OLD,0,1) NE '$' and $SBF(NEW,0,1) NE '$'
    THEN $C=' PERDEVC TO '//$QM//NEW//$QM
    endif;

    IF $SBF(OLD,0,1) EQ '$'
    THEN $A=' PERDEVC '//$QM//NEW//$QM
    endif;

    IF $SBF(NEW,0,1) EQ '$'
    THEN $D=' PERDEVC '//$QM//OLD//$QM
    endif;
  endif;

  IF CDEVDI NE BLANK
  THEN
    OLD=$SBF(CDEVDI,0,1,'*')
    NEW=$SBF(CDEVDI,1,1,'*')
    $E=' CDEVDI '//$QM//CDEVDI//$QM

    IF $SBF(OLD,0,1) NE '$' and $SBF(NEW,0,1) NE '$'
    THEN $C=' PERDEVDI TO '//$QM//NEW//$QM
    endif;

    IF $SBF(OLD,0,1) EQ '$'
    THEN $A=' PERDEVDI '//$QM//NEW//$QM
  endif;

  IF $SBF(NEW,0,1) EQ '$'
  THEN $D=' PERDEVDI '//$QM//OLD//$QM
  endif;
endif;

IF CDEVLAST NE BLANK
THEN
  OLD=$SBF(CDEVLAST,0,1,'*')
  NEW=$SBF(CDEVLAST,1,1,'*')
  $E=' CDEVLAST '//$QM//CDEVLAST//$QM

  IF $SBF(OLD,0,1) NE '$' and $SBF(NEW,0,1) NE '$'
  THEN $C=' PERDEVLAST TO '//$QM//NEW//$QM
  endif;

  IF $SBF(OLD,0,1) EQ '$'
  THEN $A=' PERDEVLAST '//$QM//NEW//$QM
  endif;

  IF $SBF(NEW,0,1) EQ '$'
  THEN $D=' PERDEVLAST '//$QM//OLD//$QM
  endif;
endif;

IF CDEVNTE NE BLANK
THEN
  OLD=$SBF(CDEVNTE,0,1,'*')
  NEW=$SBF(CDEVNTE,1,1,'*')
  $E=' CDEVNTE '//$QM//CDEVNTE//$QM

  IF $SBF(OLD,0,1) NE '$' and $SBF(NEW,0,1) NE '$'
  THEN $C=' PERDEVNTE TO '//$QM//NEW//$QM
  endif;

  IF $SBF(OLD,0,1) EQ '$'
  THEN $A=' PERDEVNTE '//$QM//NEW//$QM
  endif;

  IF $SBF(NEW,0,1) EQ '$'
  THEN $D=' PERDEVNTE '//$QM//OLD//$QM
  endif;
endif;

IF CDOA NE BLANK
THEN
  OLD=$SBF(CDOA,0,1,'*')
  NEW=$SBF(CDOA,1,1,'*')
  $E=' CDOA '//$QM//CDOA//$QM

  IF $SBF(OLD,0,1) NE '$' and $SBF(NEW,0,1) NE '$'
  THEN $C=' PERDOA TO '//$QM//NEW//$QM
  endif;
  IF $SBF(OLD,0,1) EQ '$'
  THEN $A=' PERDOA '//$QM//NEW//$QM
  endif;

  IF $SBF(NEW,0,1) EQ '$' (only occurs if ITOA = 'ZZ')
  THEN $D=' PERDOA '//$QM//OLD//$QM
  endif;
endif;

IF CDOB NE BLANK
THEN
  OLD=$SBF(CDOB,0,1,'*')
  NEW=$SBF(CDOB,1,1,'*')
  $E=' CDOB '//$QM//CDOB//$QM

  IF $SBF(OLD,0,1) NE '$' and $SBF(NEW,0,1) NE '$'
  THEN $C=' PERDOB TO '//$QM//NEW//$QM
  endif;

  IF $SBF(OLD,0,1) EQ '$'
  THEN $A=' PERDOB '//$QM//NEW//$QM
  endif;

  IF $SBF(NEW,0,1) EQ '$' (only occurs if ITOA = 'ZZ')
  THEN $D=' PERDOB '//$QM//OLD//$QM
  endif;
endif;

IF CDOG NE BLANK
THEN
  OLD=$SBF(CDOG,0,1,'*')
  NEW=$SBF(CDOG,1,1,'*')
  $E=' CDOG '//$QM//CDOG//$QM

  IF $SBF(OLD,0,1) NE '$' and $SBF(NEW,0,1) NE '$'
  THEN $C=' PERDOG TO '//$QM//NEW//$QM
  endif;

  IF $SBF(OLD,0,1) EQ '$'
  THEN $A=' PERDOG '//$QM//NEW//$QM
  endif;

  IF $SBF(NEW,0,1) EQ '$'
  THEN $D=' PERDOG '//$QM//OLD//$QM
  endif;
endif;

IF CETP NE BLANK
THEN
  OLD=$SBF(CETP,0,1,'*')
  NEW=$SBF(CETP,1,1,'*')
  $E=' CETP '//$QM//CETP//$QM
  IF $SBF(OLD,0,1) NE '$' and $SBF(NEW,0,1) NE '$'
  THEN $C=' PERETP TO '//$QM//NEW//$QM
  endif;

  IF $SBF(OLD,0,1) EQ '$'
  THEN $A=' PERETP '//$QM//NEW//$QM
  endif;

  IF $SBF(NEW,0,1) EQ '$'
  THEN $D=' PERETP '//$QM//OLD//$QM
  endif;
endif;

IF CFLSA NE BLANK
THEN
  OLD=$SBF(CFLSA,0,1,'*')
  NEW=$SBF(CFLSA,1,1,'*')
  $E=' CFLSA '//$QM//CFLSA//$QM

  IF $SBF(OLD,0,1) NE '$' and $SBF(NEW,0,1) NE '$'
  THEN $C=' PERFLSA TO '//$QM//NEW//$QM
  endif;

  IF $SBF(OLD,0,1) EQ '$'
  THEN $A=' PERFLSA '//$QM//NEW//$QM
  endif;

  IF $SBF(NEW,0,1) EQ '$'
  THEN $D=' PERFLSA '//$QM//OLD//$QM
  endif;
endif;

IF CFLSADT NE BLANK
THEN
  OLD=$SBF(CFLSADT,0,1,'*')
  NEW=$SBF(CFLSADT,1,1,'*')
  $E=' CFLSADT '//$QM//CFLSADT//$QM

  IF $SBF(OLD,0,1) NE '$' and $SBF(NEW,0,1) NE '$'
  THEN $C=' PERFLSADT TO '//$QM//NEW//$QM
  endif;

  IF $SBF(OLD,0,1) EQ '$'
  THEN $A=' PERFLSADT '//$QM//NEW//$QM
  endif;

  IF $SBF(NEW,0,1) EQ '$'
  THEN $D=' PERFLSADT '//$QM//OLD//$QM
  endif;
endif;

IF CGR NE BLANK
THEN
  OLD=$SBF(CGR,0,1,'*')
  NEW=$SBF(CGR,1,1,'*')
  $E=' CGR '//$QM//CGR//$QM

  IF $SBF(OLD,0,1) NE '$' and $SBF(NEW,0,1) NE '$'
  THEN $C=' PERGR TO '//$QM//NEW//$QM
  endif;

  IF $SBF(OLD,0,1) EQ '$'
  THEN $A=' PERGR '//$QM//NEW//$QM
  endif;

  IF $SBF(NEW,0,1) EQ '$' (only occurs if ITOA = 'ZZ')
  THEN $D=' PERGR '//$QM//OLD//$QM
  endif;
endif;

IF CGRPRT1 NE BLANK
THEN
  OLD=$SBF(CGRPRT1,0,1,'*')
  NEW=$SBF(CGRPRT1,1,1,'*')
  $E=' CGRPRT1 '//$QM//CGRPRT1//$QM

  IF $SBF(OLD,0,1) NE '$' and $SBF(NEW,0,1) NE '$'
  THEN $C=' PERGRPRT1 TO '//$QM//NEW//$QM
  endif;

  IF $SBF(OLD,0,1) EQ '$'
  THEN $A=' PERGRPRT1 '//$QM//NEW//$QM
  endif;

  IF $SBF(NEW,0,1) EQ '$'
  THEN $D=' PERGRPRT1 '//$QM//OLD//$QM
  endif;
endif;

IF CGSDTXTS NE BLANK
THEN
  OLD=$SBF(CGSDTXTS,0,1,'*')
  NEW=$SBF(CGSDTXTS,1,1,'*')
  $E=' CGSDTXTS '//$QM//CGSDTXTS//$QM

  IF $SBF(OLD,0,1) NE '$' and $SBF(NEW,0,1) NE '$'
  THEN $C=' PERGSDTXTS TO '//$QM//NEW//$QM
  endif;

  IF $SBF(OLD,0,1) EQ '$'
  THEN $A=' PERGSDTXTS '//$QM//NEW//$QM
  endif;

  IF $SBF(NEW,0,1) EQ '$' (only occurs if ITOA = 'ZZ')
  THEN $D=' PERGSDTXTS '//$QM//OLD//$QM
  endif;
endif;
```

```
IF CHQTXTS NE BLANK
THEN
  OLD=$SBF(CHQTXTS,0,1,'*')
  NEW=$SBF(CHQTXTS,1,1,'*')
  $E=' CHQTXTS '//$QM//CHQTXTS//$QM

  IF $SBF(OLD,0,1) NE '$' and $SBF(NEW,0,1) NE '$'
  THEN $C=' PERHQTXTS TO '//$QM//NEW//$QM
  endif;

  IF $SBF(OLD,0,1) EQ '$'
  THEN $A=' PERHQTXTS '//$QM//NEW//$QM
  endif;

  IF $SBF(NEW,0,1) EQ '$' (only occurs if ITOA = 'ZZ')
  THEN $D=' PERHQTXTS '//$QM//OLD//$QM
  endif;
endif;

IF CHRS NE BLANK
THEN
  OLD=$SBF(CHRS,0,1,'*')
  NEW=$SBF(CHRS,1,1,'*')
  $E=' CHRS '//$QM//CHRS//$QM

  IF $SBF(OLD,0,1) NE '$' and $SBF(NEW,0,1) NE '$'
  THEN $C=' PERHRS TO '//$QM//NEW//$QM
  endif;

  IF $SBF(OLD,0,1) EQ '$'
  THEN $A=' PERHRS '//$QM//NEW//$QM
  endif;

  IF $SBF(NEW,0,1) EQ '$'
  THEN $D=' PERHRS '//$QM//OLD//$QM
  endif;
endif;

IF CLCD NE BLANK
THEN
  OLD=$SBF(CLCD,0,1,'*')
  NEW=$SBF(CLCD,1,1,'*')
  $E=' CLCD '//$QM//CLCD//$QM

  IF $SBF(OLD,0,1) NE '$' and $SBF(NEW,0,1) NE '$'
  THEN $C=' PERLCD TO '//$QM//NEW//$QM
  endif;

  IF $SBF(OLD,0,1) EQ '$'
  THEN $A=' PERLCD '//$QM//NEW//$QM
  endif;

  IF $SBF(NEW,0,1) EQ '$'
  THEN $D=' PERLCD '//$QM//OLD//$QM
```

```
    endif;
  endif;

  IF CLEI NE BLANK
  THEN
    OLD=$SBF(CLEI,0,1,'*')
    NEW=$SBF(CLEI,1,1,'*')
    $E=' CLEI '//$QM//CLEI//$QM

    IF $SBF(OLD,0,1) NE '$' and $SBF(NEW,0,1) NE '$'
    THEN $C=' PERLEI TO '//$QM//NEW//$QM
    endif;

    IF $SBF(OLD,0,1) EQ '$'
    THEN $A=' PERLEI '//$QM//NEW//$QM
    endif;

    IF $SBF(NEW,0,1) EQ '$'
    THEN $D=' PERLEI '//$QM//OLD//$QM
    endif;
  endif;

  IF CNAMEOR NE BLANK
  THEN
    OLD=$SBF(CNAMEOR,0,1,'*')
    NEW=$SBF(CNAMEOR,1,1,'*')
    $E=' CNAMEOR '//$QM//CNAMEOR//$QM

    IF $SBF(OLD,0,1) NE '$' and $SBF(NEW,0,1) NE '$'
    THEN $C=' PERNAMEOR TO '//$QM//NEW//$QM
    endif;

    IF $SBF(OLD,0,1) EQ '$'
    THEN $A=' PERNAMEOR '//$QM//NEW//$QM
    endif;

    IF $SBF(NEW,0,1) EQ '$' (only occurs if ITOA = 'ZZ')
    THEN $D=' PERNAMEOR '//$QM//OLD//$QM
    endif;
  endif;

  IF CONAMEO3 NE BLANK
  THEN
    OLD=$SBF(CONAMEO3,0,1,'*')
    NEW=$SBF(CONAMEO3,1,1,'*')
    $E=' IFCONAMEOR3 '//$QM//CONAMEO3//$QM

    IF $SBF(OLD,0,1) NE '$' and $SBF(NEW,0,1) NE '$'
    THEN $C=' PERONAMEOR3 TO '//$QM//NEW//$QM
    endif;

    IF $SBF(OLD,0,1) EQ '$'
    THEN $A=' PERONAMEOR3 '//$QM//NEW//$QM
    endif;
  IF $SBF(NEW,0,1) EQ '$' (only occurs if ITOA = 'ZZ')
  THEN $D=' PERONAMEOR3 '//$QM//OLD//$QM
  endif;
endif;

IF CNOA NE BLANK
THEN
  OLD=$SBF(CNOA,0,1,'*')
  NEW=$SBF(CNOA,1,1,'*')
  $E=' CNOA '//$QM//CNOA//$QM

  IF $SBF(OLD,0,1) NE '$' and $SBF(NEW,0,1) NE '$'
  THEN $C=' PERNOA '//$QM//OLD//$QM//'TO'//$QM//NEW//$QM
  endif;

  IF $SBF(OLD,0,1) EQ '$'
  THEN $A=' PERNOA '//$QM//NEW//$QM
  endif;

  IF $SBF(NEW,0,1) EQ '$' (only occurs if ITOA = 'ZZ')
  THEN $D=' PERNOA '//$QM//OLD//$QM
  endif;
endif;

IF CNSCA NE BLANK
THEN
  OLD=$SBF(CNSCA,0,1,'*')
  NEW=$SBF(CNSCA,1,1,'*')
  $E=' CNSCA '//$QM//CNSCA//$QM

  IF $SBF(OLD,0,1) NE '$' and $SBF(NEW,0,1) NE '$'
  THEN $C=' PERNSCA TO '//$QM//NEW//$QM
  endif;

  IF $SBF(OLD,0,1) EQ '$'
  THEN $A=' PERNSCA '//$QM//NEW//$QM
  endif;

  IF $SBF(NEW,0,1) EQ '$'
  THEN $D=' PERNSCA '//$QM//OLD//$QM
  endif;
endif;

IF CNSCADT NE BLANK
THEN
  OLD=$SBF(CNSCADT,0,1,'*')
  NEW=$SBF(CNSCADT,1,1,'*')
  $E=' CNSCADT '//$QM//CNSCADT//$QM

  IF $SBF(OLD,0,1) NE '$' and $SBF(NEW,0,1) NE '$'
  THEN $C=' PERNSCADT TO '//$QM//NEW//$QM
  endif;
  IF $SBF(OLD,0,1) EQ '$'
  THEN $A=' PERNSCADT '//$QM//NEW//$QM
  endif;

  IF $SBF(NEW,0,1) EQ '$'
  THEN $D=' PERNSCADT '//$QM//OLD//$QM
  endif;
endif;

IF CNTY NE BLANK
THEN
  OLD=$SBF(CNTY,0,1,'*')
  NEW=$SBF(CNTY,1,1,'*')
  $E=' CNTY '//$QM//CNTY//$QM

  IF $SBF(OLD,0,1) NE '$' and $SBF(NEW,0,1) NE '$'
  THEN $C=' PERNTY TO '//$QM//NEW//$QM
  endif;

  IF $SBF(OLD,0,1) EQ '$'
  THEN $A=' PERNTY '//$QM//NEW//$QM
  endif;

  IF $SBF(NEW,0,1) EQ '$' (only occurs if ITOA = 'ZZ')
  THEN $D=' PERNTY '//$QM//OLD//$QM
  endif;
endif;

IF COCCE NE BLANK
THEN
  OLD=$SBF(COCCE,0,1,'*')
  NEW=$SBF(COCCE,1,1,'*')
  $E=' COCCE '//$QM//COCCE//$QM

  IF $SBF(OLD,0,1) NE '$' and $SBF(NEW,0,1) NE '$'
  THEN $C=' PEROCCE TO '//$QM//NEW//$QM
  endif;

  IF $SBF(OLD,0,1) EQ '$'
  THEN $A=' PEROCCE '//$QM//NEW//$QM
  endif;

  IF $SBF(NEW,0,1) EQ '$' (only occurs if ITOA = 'ZZ')
  THEN $D=' PEROCCE '//$QM//OLD//$QM
  endif;
endif;

IF COCCETXT NE BLANK
THEN
  OLD=$SBF(COCCETXT,0,1,'*')
  NEW=$SBF(COCCETXT,1,1,'*')
  $E=' COCCETXT '//$QM//COCCETXT//$QM

  IF $SBF(OLD,0,1) NE '$' and $SBF(NEW,0,1) NE '$'
  THEN $C=' PEROCCETXT TO '//$QM//NEW//$QM
  endif;

  IF $SBF(OLD,0,1) EQ '$'
  THEN $A=' PEROCCETXT '//$QM//NEW//$QM
  endif;

  IF $SBF(NEW,0,1) EQ '$' (only occurs if ITOA = 'ZZ')
  THEN $D=' PEROCCETXT '//$QM//OLD//$QM
  endif;
endif;

IF COCCFAM NE BLANK
THEN
  OLD=$SBF(COCCFAM,0,1,'*')
  NEW=$SBF(COCCFAM,1,1,'*')
  $E=' COCCFAM '//$QM//COCCFAM//$QM

  IF $SBF(OLD,0,1) NE '$' and $SBF(NEW,0,1) NE '$'
  THEN $C=' PEROCCFAM TO '//$QM//NEW//$QM
  endif;

  IF $SBF(OLD,0,1) EQ '$'
  THEN $A=' PEROCCFAM '//$QM//NEW//$QM
  endif;

  IF $SBF(NEW,0,1) EQ '$' (only occurs if ITOA = 'ZZ')
  THEN $D=' PEROCCFAM '//$QM//OLD//$QM
  endif;
endif;

IF COCCSUF NE BLANK
THEN
  OLD=$SBF(COCCSUF,0,1,'*')
  NEW=$SBF(COCCSUF,1,1,'*')
  $E=' COCCSUF '//$QM//COCCSUF//$QM

  IF $SBF(OLD,0,1) NE '$' and $SBF(NEW,0,1) NE '$'
  THEN $C=' PEROCCSUF TO '//$QM//NEW//$QM
  endif;

  IF $SBF(OLD,0,1) EQ '$'
  THEN $A=' PEROCCSUF '//$QM//NEW//$QM
  endif;

  IF $SBF(NEW,0,1) EQ '$'
  THEN $D=' PEROCCSUF '//$QM//OLD//$QM
  endif;
endif;

IF CNAMEOR3 NE BLANK
THEN
  OLD=$SBF(CNAMEOR3,0,1,'*')
  NEW=$SBF(CNAMEOR3,1,1,'*')
```

```
    $E=' CNAMEOR3 '//$QM//CNAMEOR3//$QM

    IF $SBF(OLD,0,1) NE '$' and $SBF(NEW,0,1) NE '$'
    THEN $C=' PERNAMEOR3 TO '//$QM//NEW//$QM
    endif;

    IF $SBF(OLD,0,1) EQ '$'
    THEN $A=' PERNAMEOR3 '//$QM//NEW//$QM
    endif;

    IF $SBF(NEW,0,1) EQ '$' (only occurs if ITOA = 'ZZ')
    THEN $D=' PERNAMEOR3 '//$QM//OLD//$QM
    endif;
  endif;

  IF CORGCODE NE BLANK
  THEN
    OLD=$SBF(CORGCODE,0,1,'*')
    NEW=$SBF(CORGCODE,1,1,'*')
    $E=' CORGCODE '//$QM//CORGCODE//$QM

    IF $SBF(OLD,0,1) NE '$' and $SBF(NEW,0,1) NE '$'
    THEN $C=' PERORGCODE TO '//$QM//NEW//$QM
    endif;

    IF $SBF(OLD,0,1) EQ '$'
    THEN $A=' PERORGCODE '//$QM//NEW//$QM
    endif;

    IF $SBF(NEW,0,1) EQ '$' (only occurs if ITOA = 'ZZ')
    THEN $D=' PERORGCODE '//$QM//OLD//$QM
    endif;
  endif;

  IF CORGDTIN NE BLANK
  THEN
    OLD=$SBF(CORGDTIN,0,1,'*')
    NEW=$SBF(CORGDTIN,1,1,'*')
    $E=' CORGDTIN '//$QM//CORGDTIN//$QM

    IF $SBF(OLD,0,1) NE '$' and $SBF(NEW,0,1) NE '$'
    THEN $C=' PERORGDTIN TO '//$QM//NEW//$QM
    endif;

    IF $SBF(OLD,0,1) EQ '$'
    THEN $A=' PERORGDTIN '//$QM//NEW//$QM
    endif;

    IF $SBF(NEW,0,1) EQ '$' (only occurs if ITOA = 'ZZ')
    THEN $D=' PERORGDTIN '//$QM//OLD//$QM
    endif;
  endif;

  IF COVERLAP NE BLANK
```

```
THEN
  OLD=$SBF(COVERLAP,0,1,'*')
  NEW=$SBF(COVERLAP,1,1,'*')
  $E=' COVERLAP '//$QM//COVERLAP//$QM

  IF $SBF(OLD,0,1) NE '$' and $SBF(NEW,0,1) NE '$'
  THEN $C=' PEROVERLAP TO '//$QM//NEW//$QM
  endif;

  IF $SBF(OLD,0,1) EQ '$'
  THEN $A=' PEROVERLAP '//$QM//NEW//$QM
  endif;

  IF $SBF(NEW,0,1) EQ '$'
  THEN $D=' PEROVERLAP '//$QM//OLD//$QM
  endif;
endif;

IF COVLAPING NE BLANK
THEN
  OLD=$SBF(COVLAPING,0,1,'*')
  NEW=$SBF(COVLAPING,1,1,'*')
  $E=' COVLAPING '//$QM//COVLAPING//$QM

  IF $SBF(OLD,0,1) NE '$' and $SBF(NEW,0,1) NE '$'
  THEN $C=' PEROVLAPING TO '//$QM//NEW//$QM
  endif;

  IF $SBF(OLD,0,1) EQ '$'
  THEN $A=' PEROVLAPING '//$QM//NEW//$QM
  endif;

  IF $SBF(NEW,0,1) EQ '$'
  THEN $D=' PEROVLAPING '//$QM//OLD//$QM
  endif;
endif;

IF COVNTE NE BLANK
THEN
  OLD=$SBF(COVNTE,0,1,'*')
  NEW=$SBF(COVNTE,1,1,'*')
  $E=' COVNTE '//$QM//COVNTE//$QM

  IF $SBF(OLD,0,1) NE '$' and $SBF(NEW,0,1) NE '$'
  THEN $C=' PEROVNTE TO '//$QM//NEW//$QM
  endif;

  IF $SBF(OLD,0,1) EQ '$'
  THEN $A=' PEROVNTE '//$QM//NEW//$QM
  endif;

  IF $SBF(NEW,0,1) EQ '$'
  THEN $D=' PEROVNTE '//$QM//OLD//$QM
  endif;
```

```
endif;

IF CPAYB NE BLANK
THEN
  OLD=$SBF(CPAYB,0,1,'*')
  NEW=$SBF(CPAYB,1,1,'*')
  $E=' CPAYB '//$QM//CPAYB//$QM

  IF $SBF(OLD,0,1) NE '$' and $SBF(NEW,0,1) NE '$'
  THEN $C=' PERPAYB TO '//$QM//NEW//$QM
  endif;

  IF $SBF(OLD,0,1) EQ '$'
  THEN $A=' PERPAYB '//$QM//NEW//$QM
  endif;

  IF $SBF(NEW,0,1) EQ '$'
  THEN $D=' PERPAYB '//$QM//OLD//$QM
  endif;
endif;

IF CPAYDT NE BLANK
THEN
  OLD=$SBF(CPAYDT,0,1,'*')
  NEW=$SBF(CPAYDT,1,1,'*')
  $E=' CPAYDT '//$QM//CPAYDT//$QM

  IF $SBF(OLD,0,1) NE '$' and $SBF(NEW,0,1) NE '$'
  THEN $C=' PERPAYDT TO '//$QM//NEW//$QM
  endif;

  IF $SBF(OLD,0,1) EQ '$'
  THEN $A=' PERPAYDT '//$QM//NEW//$QM
  endif;

  IF $SBF(NEW,0,1) EQ '$'
  THEN $D=' PERPAYDT '//$QM//OLD//$QM
  endif;
endif;

IF CPAYDTC NE BLANK
THEN
  OLD=$SBF(CPAYDTC,0,1,'*')
  NEW=$SBF(CPAYDTC,1,1,'*')
  $E=' CPAYDTC '//$QM//CPAYDTC//$QM

  IF $SBF(OLD,0,1) NE '$' and $SBF(NEW,0,1) NE '$'
  THEN $C=' PERPAYDTC TO '//$QM//NEW//$QM
  endif;

  IF $SBF(OLD,0,1) EQ '$'
  THEN $A=' PERPAYDTC '//$QM//NEW//$QM
  endif;
  IF $SBF(NEW,0,1) EQ '$'
  THEN $D=' PERPAYDTC '//$QM//OLD//$QM
  endif;
endif;

IF CPDATE NE BLANK
THEN
  OLD=$SBF(CPDATE,0,1,'*')
  NEW=$SBF(CPDATE,1,1,'*')
  $E=' CPDATE '//$QM//CPDATE//$QM

  IF $SBF(OLD,0,1) NE '$' and $SBF(NEW,0,1) NE '$'
  THEN $C=' PERPDATE TO '//$QM//NEW//$QM
  endif;

  IF $SBF(OLD,0,1) EQ '$'
  THEN $A=' PERPDATE '//$QM//NEW//$QM
  endif;

  IF $SBF(NEW,0,1) EQ '$' (only occurs if ITOA = 'ZZ')
  THEN $D=' PERPDATE '//$QM//OLD//$QM
  endif;
endif;

IF CPGVT NE BLANK
THEN
  OLD=$SBF(CPGVT,0,1,'*')
  NEW=$SBF(CPGVT,1,1,'*')
  $E=' CPGVT '//$QM//CPGVT//$QM

  IF $SBF(OLD,0,1) NE '$' and $SBF(NEW,0,1) NE '$'
  THEN $C=' PERPGVT TO '//$QM//NEW//$QM
  endif;

  IF $SBF(OLD,0,1) EQ '$'
  THEN $A=' PERPGVT '//$QM//NEW//$QM
  endif;

  IF $SBF(NEW,0,1) EQ '$'
  THEN $D=' PERPGVT '//$QM//OLD//$QM
  endif;
endif;

IF CPOSNO NE BLANK
THEN
  OLD=$SBF(CPOSNO,0,1,'*')
  NEW=$SBF(CPOSNO,1,1,'*')
  $E=' CPOSNO '//$QM//CPOSNO//$QM

  IF $SBF(OLD,0,1) NE '$' and $SBF(NEW,0,1) NE '$'
  THEN $C=' PERPOSNO TO '//$QM//NEW//$QM
  endif;

  IF $SBF(OLD,0,1) EQ '$'
```

```
    THEN $A=' PERPOSNO '//$QM//NEW//$QM
    endif;

    IF $SBF(NEW,0,1) EQ '$' (only occurs if ITOA = 'ZZ')
    THEN $D=' PERPOSNO '//$QM//OLD//$QM
    endif;
  endif;

  IF CPRA NE BLANK
  THEN
    OLD=$SBF(CPRA,0,1,'*')
    NEW=$SBF(CPRA,1,1,'*')
    $E=' CPRA '//$QM//CPRA//$QM

    IF $SBF(OLD,0,1) NE '$' and $SBF(NEW,0,1) NE '$'
    THEN $C=' PERPRA TO '//$QM//NEW//$QM
    endif;

    IF $SBF(OLD,0,1) EQ '$'
    THEN $A=' PERPRA '//$QM//NEW//$QM
    endif;

    IF $SBF(NEW,0,1) EQ '$'
    THEN $D=' PERPRA '//$QM//OLD//$QM
    endif;
  endif;

  IF CPRADI NE BLANK
  THEN
    OLD=$SBF(CPRADI,0,1,'*')
    NEW=$SBF(CPRADI,1,1,'*')
    $E=' CPRADI '//$QM//CPRADI//$QM

    IF $SBF(OLD,0,1) NE '$' and $SBF(NEW,0,1) NE '$'
    THEN $C=' PERPRADI TO '//$QM//NEW//$QM
    endif;

    IF $SBF(OLD,0,1) EQ '$'
    THEN $A=' PERPRADI '//$QM//NEW//$QM
    endif;

    IF $SBF(NEW,0,1) EQ '$'
    THEN $D=' PERPRADI '//$QM//OLD//$QM
    endif;
  endif;

  IF CPRANTE NE BLANK
  THEN
    OLD=$SBF(CPRANTE,0,1,'*')
    NEW=$SBF(CPRANTE,1,1,'*')
    $E=' CPRANTE '//$QM//CPRANTE//$QM

    IF $SBF(OLD,0,1) NE '$' and $SBF(NEW,0,1) NE '$'
    THEN $C=' PERPRANTE TO '//$QM//NEW//$QM
  endif;

  IF $SBF(OLD,0,1) EQ '$'
  THEN $A=' PERPRANTE '//$QM//NEW//$QM
  endif;

  IF $SBF(NEW,0,1) EQ '$'
  THEN $D=' PERPRANTE '//$QM//OLD//$QM
  endif;
endif;

IF CPRMNTE NE BLANK
THEN
  OLD=$SBF(CPRMNTE,0,1,'*')
  NEW=$SBF(CPRMNTE,1,1,'*')
  $E=' CPRMNTE '//$QM//CPRMNTE//$QM

  IF $SBF(OLD,0,1) NE '$' and $SBF(NEW,0,1) NE '$'
  THEN $C=' PERPRMNTE TO '//$QM//NEW//$QM
  endif;

  IF $SBF(OLD,0,1) EQ '$'
  THEN $A=' PERPRMNTE '//$QM//NEW//$QM
  endif;

  IF $SBF(NEW,0,1) EQ '$'
  THEN $D=' PERPRMNTE '//$QM//OLD//$QM
  endif;
endif;

IF CPROJNO NE BLANK
THEN
  OLD=$SBF(CPROJNO,0,1,'*')
  NEW=$SBF(CPROJNO,1,1,'*')
  $E=' CPROJNO '//$QM//CPROJNO//$QM

  IF $SBF(OLD,0,1) NE '$' and $SBF(NEW,0,1) NE '$'
  THEN $C=' PERPROJNO TO '//$QM//NEW//$QM
  endif;

  IF $SBF(OLD,0,1) EQ '$'
  THEN $A=' PERPROJNO '//$QM//NEW//$QM
  endif;

  IF $SBF(NEW,0,1) EQ '$' (only occurs if ITOA = 'ZZ')
  THEN $D=' PERPROJNO '//$QM//OLD//$QM
  endif;
endif;

IF CPWGI NE BLANK
THEN
  OLD=$SBF(CPWGI,0,1,'*')
  NEW=$SBF(CPWGI,1,1,'*')
  $E=' CPWGI '//$QM//CPWGI//$QM
  IF $SBF(OLD,0,1) NE '$' and $SBF(NEW,0,1) NE '$'
  THEN $C=' PERPWGI TO '//$QM//NEW//$QM
  endif;

  IF $SBF(OLD,0,1) EQ '$'
  THEN $A=' PERPWGI '//$QM//NEW//$QM
  endif;

  IF $SBF(NEW,0,1) EQ '$'
  THEN $D=' PERPWGI '//$QM//OLD//$QM
  endif;
endif;

IF CRACE NE BLANK
THEN
  OLD=$SBF(CRACE,0,1,'*')
  NEW=$SBF(CRACE,1,1,'*')
  $E=' CRACE '//$QM//CRACE//$QM

  IF $SBF(OLD,0,1) NE '$' and $SBF(NEW,0,1) NE '$'
  THEN $C=' PERRACE TO '//$QM//NEW//$QM
  endif;

  IF $SBF(OLD,0,1) EQ '$'
  THEN $A=' PERRACE '//$QM//NEW//$QM
  endif;

  IF $SBF(NEW,0,1) EQ '$' (only occurs if ITOA = 'ZZ')
  THEN $D=' PERRACE '//$QM//OLD//$QM
  endif;
endif;

IF CRESV1 NE BLANK
THEN
  OLD=$SBF(CRESV1,0,1,'*')
  NEW=$SBF(CRESV1,1,1,'*')
  $E=' CRESV1 '//$QM//CRESV1//$QM

  IF $SBF(OLD,0,1) NE '$' and $SBF(NEW,0,1) NE '$'
  THEN $C=' PERRESV1 TO '//$QM//NEW//$QM
  endif;

  IF $SBF(OLD,0,1) EQ '$'
  THEN $A=' PERRESV1 '//$QM//NEW//$QM
  endif;

  IF $SBF(NEW,0,1) EQ '$'
  THEN $D=' PERRESV1 '//$QM//OLD//$QM
  endif;
endif;

IF CRESV2 NE BLANK
THEN
```

```
  OLD=$SBF(CRESV2,0,1,'*')
  NEW=$SBF(CRESV2,1,1,'*')
  $E=' CRESV2 '//$QM//CRESV2//$QM

  IF $SBF(OLD,0,1) NE '$' and $SBF(NEW,0,1) NE '$'
  THEN $C=' PERRESV2 TO '//$QM//NEW//$QM
  endif;

  IF $SBF(OLD,0,1) EQ '$'
  THEN $A=' PERRESV2 '//$QM//NEW//$QM
  endif;

  IF $SBF(NEW,0,1) EQ '$'
  THEN $D=' PERRESV2 '//$QM//OLD//$QM
  endif;
endif;

IF CRESV3 NE BLANK
THEN
  OLD=$SBF(CRESV3,0,1,'*')
  NEW=$SBF(CRESV3,1,1,'*')
  $E=' CRESV3 '//$QM//CRESV3//$QM

  IF $SBF(OLD,0,1) NE '$' and $SBF(NEW,0,1) NE '$'
  THEN $C=' PERRESV3 TO '//$QM//NEW//$QM
  endif;

  IF $SBF(OLD,0,1) EQ '$'
  THEN $A=' PERRESV3 '//$QM//NEW//$QM
  endif;

  IF $SBF(NEW,0,1) EQ '$'
  THEN $D=' PERRESV3 '//$QM//OLD//$QM
  endif;
endif;

IF CRESV4 NE BLANK
THEN
  OLD=$SBF(CRESV4,0,1,'*')
  NEW=$SBF(CRESV4,1,1,'*')
  $E=' CRESV4 '//$QM//CRESV4//$QM

  IF $SBF(OLD,0,1) NE '$' and $SBF(NEW,0,1) NE '$'
  THEN $C=' PERRESV4 TO '//$QM//NEW//$QM
  endif;

  IF $SBF(OLD,0,1) EQ '$'
  THEN $A=' PERRESV4 '//$QM//NEW//$QM
  endif;

  IF $SBF(NEW,0,1) EQ '$'
  THEN $D=' PERRESV4 '//$QM//OLD//$QM
  endif;
endif;
```

```
IF CRESV5 NE BLANK
THEN
  OLD=$SBF(CRESV5,0,1,'*')
  NEW=$SBF(CRESV5,1,1,'*')
  $E=' CRESV5 '//$QM//CRESV5//$QM

  IF $SBF(OLD,0,1) NE '$' and $SBF(NEW,0,1) NE '$'
  THEN $C=' PERRESV5 TO '//$QM//NEW//$QM
  endif;

  IF $SBF(OLD,0,1) EQ '$'
  THEN $A=' PERRESV5 '//$QM//NEW//$QM
  endif;

  IF $SBF(NEW,0,1) EQ '$'
  THEN $D=' PERRESV5 '//$QM//OLD//$QM
  endif;
endif;

IF CRETDI NE BLANK
THEN
  OLD=$SBF(CRETDI,0,1,'*')
  NEW=$SBF(CRETDI,1,1,'*')
  $E=' CRETDI '//$QM//CRETDI//$QM

  IF $SBF(OLD,0,1) NE '$' and $SBF(NEW,0,1) NE '$'
  THEN $C=' PERRETDI TO '//$QM//NEW//$QM
  endif;

  IF $SBF(OLD,0,1) EQ '$'
  THEN $A=' PERRETDI '//$QM//NEW//$QM
  endif;

  IF $SBF(NEW,0,1) EQ '$'
  THEN $D=' PERRETDI '//$QM//OLD//$QM
  endif;
endif;

IF CRETNTE NE BLANK
THEN
  OLD=$SBF(CRETNTE,0,1,'*')
  NEW=$SBF(CRETNTE,1,1,'*')
  $E=' CRETNTE '//$QM//CRETNTE//$QM

  IF $SBF(OLD,0,1) NE '$' and $SBF(NEW,0,1) NE '$'
  THEN $C=' PERRETNTE TO '//$QM//NEW//$QM
  endif;

  IF $SBF(OLD,0,1) EQ '$'
  THEN $A=' PERRETNTE '//$QM//NEW//$QM
  endif;

  IF $SBF(NEW,0,1) EQ '$'
  THEN $D=' PERRETNTE '//$QM//OLD//$QM
  endif;
endif;

IF CRTMT NE BLANK
THEN
  OLD=$SBF(CRTMT,0,1,'*')
  NEW=$SBF(CRTMT,1,1,'*')
  $E=' CRTMT '//$QM//CRTMT//$QM

  IF $SBF(OLD,0,1) NE '$' and $SBF(NEW,0,1) NE '$'
  THEN $C=' PERRTMT TO '//$QM//NEW//$QM
  endif;

  IF $SBF(OLD,0,1) EQ '$'
  THEN $A=' PERRTMT '//$QM//NEW//$QM
  endif;

  IF $SBF(NEW,0,1) EQ '$'
  THEN $D=' PERRTMT '//$QM//OLD//$QM
  endif;
endif;

IF CRTMTTXT NE BLANK
THEN
  OLD=$SBF(CRTMTTXT,0,1,'*')
  NEW=$SBF(CRTMTTXT,1,1,'*')
  $E=' CRTMTTXT '//$QM//CRTMTTXT//$QM

  IF $SBF(OLD,0,1) NE '$' and $SBF(NEW,0,1) NE '$'
  THEN $C=' PERRTMTTXT TO '//$QM//NEW//$QM
  endif;

  IF $SBF(OLD,0,1) EQ '$'
  THEN $A=' PERRTMTTXT '//$QM//NEW//$QM
  endif;

  IF $SBF(NEW,0,1) EQ '$'
  THEN $D=' PERRTMTTXT '//$QM//OLD//$QM
  endif;
endif;

IF CSALARY NE BLANK
THEN
  OLD=$SBF(CSALARY,0,1,'*')
  NEW=$SBF(CSALARY,1,1,'*')
  $E=' CSALARY '//$QM//CSALARY//$QM

  IF $SBF(OLD,0,1) NE '$' and $SBF(NEW,0,1) NE '$'
  THEN $C=' PERSALARY TO '//$QM//NEW//$QM
  endif;

  IF $SBF(OLD,0,1) EQ '$'
  THEN $A=' PERSALARY '//$QM//NEW//$QM
```

```
   endif;

   IF $SBF(NEW,0,1) EQ '$'
   THEN $D=' PERSALARY '//$QM//OLD//$QM
   endif;
endif;

IF CSCCE NE BLANK
THEN
   OLD=$SBF(CSCCE,0,1,'*')
   NEW=$SBF(CSCCE,1,1,'*')
   $E=' CSCCE '//$QM//CSCCE//$QM

   IF $SBF(OLD,0,1) NE '$' and $SBF(NEW,0,1) NE '$'
   THEN $C=' PERSCCE TO '//$QM//NEW//$QM
   endif;

   IF $SBF(OLD,0,1) EQ '$'
   THEN $A=' PERSCCE '//$QM//NEW//$QM
   endif;

   IF $SBF(NEW,0,1) EQ '$' (only occurs if ITOA = 'ZZ')
   THEN $D=' PERSCCE '//$QM//OLD//$QM
   endif;
endif;

IF CSCCESK NE BLANK
THEN
   OLD=$SBF(CSCCESK,0,1,'*')
   NEW=$SBF(CSCCESK,1,1,'*')
   $E=' CSCCESK '//$QM//CSCCESK//$QM

   IF $SBF(OLD,0,1) NE '$' and $SBF(NEW,0,1) NE '$'
   THEN $C=' PERSCCESK TO '//$QM//NEW//$QM
   endif;

   IF $SBF(OLD,0,1) EQ '$'
   THEN $A=' PERSCCESK '//$QM//NEW//$QM
   endif;

   IF $SBF(NEW,0,1) EQ '$' (only occurs if ITOA = 'ZZ')
   THEN $D=' PERSCCESK '//$QM//OLD//$QM
   endif;
endif;

IF CSCCETXTS NE BLANK
THEN
   OLD=$SBF(CSCCETXTS,0,1,'*')
   NEW=$SBF(CSCCETXTS,1,1,'*')
   $E=' CSCCETXTS '//$QM//CSCCETXTS//$QM

   IF $SBF(OLD,0,1) NE '$' and $SBF(NEW,0,1) NE '$'
   THEN $C=' PERSCCETXTS TO '//$QM//NEW//$QM
   endif;
```

```
  IF $SBF(OLD,0,1) EQ '$'
  THEN $A=' PERSCCETXTS '//$QM//NEW//$QM
  endif;

  IF $SBF(NEW,0,1) EQ '$' (only occurs if ITOA = 'ZZ')
  THEN $D=' PERSCCETXTS '//$QM//OLD//$QM
  endif;
endif;

IF CSCD NE BLANK
THEN
  OLD=$SBF(CSCD,0,1,'*')
  NEW=$SBF(CSCD,1,1,'*')
  $E=' CSCD '//$QM//CSCD//$QM

  IF $SBF(OLD,0,1) NE '$' and $SBF(NEW,0,1) NE '$'
  THEN $C=' PERSCD TO '//$QM//NEW//$QM
  endif;

  IF $SBF(OLD,0,1) EQ '$'
  THEN $A=' PERSCD '//$QM//NEW//$QM
  endif;

  IF $SBF(NEW,0,1) EQ '$'
  THEN $D=' PERSCD '//$QM//OLD//$QM
  endif;
endif;

IF CSCH NE BLANK
THEN
  OLD=$SBF(CSCH,0,1,'*')
  NEW=$SBF(CSCH,1,1,'*')
  $E=' CSCH '//$QM//CSCH//$QM

  IF $SBF(OLD,0,1) NE '$' and $SBF(NEW,0,1) NE '$'
  THEN $C=' PERSCH TO '//$QM//NEW//$QM
  endif;

  IF $SBF(OLD,0,1) EQ '$'
  THEN $A=' PERSCH '//$QM//NEW//$QM
  endif;

  IF $SBF(NEW,0,1) EQ '$' (only occurs if ITOA = 'ZZ')
  THEN $D=' PERSCH '//$QM//OLD//$QM
  endif;
endif;

IF CSCHGRSK1 NE BLANK
THEN
  OLD=$SBF(CSCHGRSK1,0,1,'*')
  NEW=$SBF(CSCHGRSK1,1,1,'*')
  $E=' CSCHGRSK1 '//$QM//CSCHGRSK1//$QM
  IF $SBF(OLD,0,1) NE '$' and $SBF(NEW,0,1) NE '$'
  THEN $C=' PERSCHGRSK1 TO '//$QM//NEW//$QM
  endif;

  IF $SBF(OLD,0,1) EQ '$'
  THEN $A=' PERSCHGRSK1 '//$QM//NEW//$QM
  endif;

  IF $SBF(NEW,0,1) EQ '$'
  THEN $D=' PERSCHGRSK1 '//$QM//OLD//$QM
  endif;
endif;

IF CSCHGRSK2 NE BLANK
THEN
  OLD=$SBF(CSCHGRSK2,0,1,'*')
  NEW=$SBF(CSCHGRSK2,1,1,'*')
  $E=' CSCHGRSK2 '//$QM//CSCHGRSK2//$QM

  IF $SBF(OLD,0,1) NE '$' and $SBF(NEW,0,1) NE '$'
  THEN $C=' PERSCHGRSK2 TO '//$QM//NEW//$QM
  endif;

  IF $SBF(OLD,0,1) EQ '$'
  THEN $A=' PERSCHGRSK2 '//$QM//NEW//$QM
  endif;

  IF $SBF(NEW,0,1) EQ '$'
  THEN $D=' PERSCHGRSK2 '//$QM//OLD//$QM
  endif;
endif;

IF CSCHPRT1 NE BLANK
THEN
  OLD=$SBF(CSCHPRT1,0,1,'*')
  NEW=$SBF(CSCHPRT1,1,1,'*')
  $E=' CSCHPRT1 '//$QM//CSCHPRT1//$QM

  IF $SBF(OLD,0,1) NE '$' and $SBF(NEW,0,1) NE '$'
  THEN $C=' PERSCHPRT1 TO '//$QM//NEW//$QM
  endif;

  IF $SBF(OLD,0,1) EQ '$'
  THEN $A=' PERSCHPRT1 '//$QM//NEW//$QM
  endif;

  IF $SBF(NEW,0,1) EQ '$'
  THEN $D=' PERSCHPRT1 '//$QM//OLD//$QM
  endif;
endif;

IF CSD NE BLANK
THEN
  OLD=$SBF(CSD,0,1,'*')
```

```
    NEW=$SBF(CSD,1,1,'*')
    $E=' CSD '//$QM//CSD//$QM

    IF $SBF(OLD,0,1) NE '$' and $SBF(NEW,0,1) NE '$'
    THEN $C=' PERSD TO '//$QM//NEW//$QM
    endif;

    IF $SBF(OLD,0,1) EQ '$'
    THEN $A=' PERSD '//$QM//NEW//$QM
    endif;

    IF $SBF(NEW,0,1) EQ '$' (only occurs if ITOA = 'ZZ')
    THEN $D=' PERSD '//$QM//OLD//$QM
    endif;
  endif;

  IF CSECCL NE BLANK
  THEN
    OLD=$SBF(CSECCL,0,1,'*')
    NEW=$SBF(CSECCL,1,1,'*')
    $E=' CSECCL '//$QM//CSECCL//$QM

    IF $SBF(OLD,0,1) NE '$' and $SBF(NEW,0,1) NE '$'
    THEN $C=' PERSECCL TO '//$QM//NEW//$QM
    endif;

    IF $SBF(OLD,0,1) EQ '$'
    THEN $A=' PERSECCL '//$QM//NEW//$QM
    endif;

    IF $SBF(NEW,0,1) EQ '$'
    THEN $D=' PERSECCL '//$QM//OLD//$QM
    endif;
  endif;

  IF CSEQ NE BLANK
  THEN
    OLD=$SBF(CSEQ,0,1,'*')
    NEW=$SBF(CSEQ,1,1,'*')
    $E=' CSEQ '//$QM//CSEQ//$QM

    IF $SBF(OLD,0,1) NE '$' and $SBF(NEW,0,1) NE '$'
    THEN $C=' PERSEQ TO '//$QM//NEW//$QM
    endif;

    IF $SBF(OLD,0,1) EQ '$'
    THEN $A=' PERSEQ '//$QM//NEW//$QM
    endif;

    IF $SBF(NEW,0,1) EQ '$'
    THEN $D=' PERSEQ '//$QM//OLD//$QM
    endif;
  endif;
```

```
IF CSERIAL NE BLANK
THEN
  OLD=$SBF(CSERIAL,0,1,'*')
  NEW=$SBF(CSERIAL,1,1,'*')
  $E=' CSERIAL '//$QM//CSERIAL//$QM

  IF $SBF(OLD,0,1) NE '$' and $SBF(NEW,0,1) NE '$'
  THEN $C=' PERSERIAL TO '//$QM//NEW//$QM
  endif;

  IF $SBF(OLD,0,1) EQ '$'
  THEN $A=' PERSERIAL '//$QM//NEW//$QM
  endif;

  IF $SBF(NEW,0,1) EQ '$'
  THEN $D=' PERSERIAL '//$QM//OLD//$QM
  endif;
endif;

IF CSEX NE BLANK
THEN
  OLD=$SBF(CSEX,0,1,'*')
  NEW=$SBF(CSEX,1,1,'*')
  $E=' CSEX '//$QM//CSEX//$QM

  IF $SBF(OLD,0,1) NE '$' and $SBF(NEW,0,1) NE '$'
  THEN $C=' PERSEX TO '//$QM//NEW//$QM
  endif;

  IF $SBF(OLD,0,1) EQ '$'
  THEN $A=' PERSEX '//$QM//NEW//$QM
  endif;

  IF $SBF(NEW,0,1) EQ '$' (only occurs if ITOA = 'ZZ')
  THEN $D=' PERSEX '//$QM//OLD//$QM
  endif;
endif;

IF CSFN NE BLANK
THEN
  OLD=$SBF(CSFN,0,1,'*')
  NEW=$SBF(CSFN,1,1,'*')
  $E=' CSFN '//$QM//CSFN//$QM

  IF $SBF(OLD,0,1) NE '$' and $SBF(NEW,0,1) NE '$'
  THEN $C=' PERSFN TO '//$QM//NEW//$QM
  endif;

  IF $SBF(OLD,0,1) EQ '$'
  THEN $A=' PERSFN '//$QM//NEW//$QM
  endif;

  IF $SBF(NEW,0,1) EQ '$' (only occurs if ITOA = 'ZZ')
  THEN $D=' PERSFN '//$QM//OLD//$QM
  endif;
endif;

IF CSREF NE BLANK
THEN
  OLD=$SBF(CSREF,0,1,'*')
  NEW=$SBF(CSREF,1,1,'*')
  $E=' CSREF '//$QM//CSREF//$QM

  IF $SBF(OLD,0,1) NE '$' and $SBF(NEW,0,1) NE '$'
  THEN $C=' PERSREF '//$QM//OLD//$QM//'TO'//$QM//NEW//$QM
  endif;

  IF $SBF(OLD,0,1) EQ '$'
  THEN $A=' PERSREF '//$QM//NEW//$QM
  endif;

  IF $SBF(NEW,0,1) EQ '$'
  THEN $D=' PERSREF '//$QM//OLD//$QM
  endif;
endif;

IF CSSNOTH NE BLANK
THEN
  OLD=$SBF(CSSNOTH,0,1,'*')
  NEW=$SBF(CSSNOTH,1,1,'*')
  $E=' CSSNOTH '//$QM//CSSNOTH//$QM

  IF $SBF(OLD,0,1) NE '$' and $SBF(NEW,0,1) NE '$'
  THEN $C=' PERSSNOTH TO '//$QM//NEW//$QM
  endif;

  IF $SBF(OLD,0,1) EQ '$'
  THEN $A=' PERSSNOTH '//$QM//NEW//$QM
  endif;

  IF $SBF(NEW,0,1) EQ '$'
  THEN $D=' PERSSNOTH '//$QM//OLD//$QM
  endif;
endif;

IF CSTAN NE BLANK
THEN
  OLD=$SBF(CSTAN,0,1,'*')
  NEW=$SBF(CSTAN,1,1,'*')
  $E=' CSTAN '//$QM//CSTAN//$QM

  IF $SBF(OLD,0,1) NE '$' and $SBF(NEW,0,1) NE '$'
  THEN $C=' PERSTAN TO '//$QM//NEW//$QM
  endif;

  IF $SBF(OLD,0,1) EQ '$'
  THEN $A=' PERSTAN '//$QM//NEW//$QM
  endif;
  IF $SBF(NEW,0,1) EQ '$' (only occurs if ITOA = 'ZZ')
  THEN $D=' PERSTAN '//$QM//OLD//$QM
  endif;
endif;

IF CSTANAREA NE BLANK
THEN
  OLD=$SBF(CSTANAREA,0,1,'*')
  NEW=$SBF(CSTANAREA,1,1,'*')
  $E=' CSTANAREA '//$QM//CSTANAREA//$QM

  IF $SBF(OLD,0,1) NE '$' and $SBF(NEW,0,1) NE '$'
  THEN $C=' PERSTANAREA TO '//$QM//NEW//$QM
  endif;

  IF $SBF(OLD,0,1) EQ '$'
  THEN $A=' PERSTANAREA '//$QM//NEW//$QM
  endif;

  IF $SBF(NEW,0,1) EQ '$' (only occurs if ITOA = 'ZZ')
  THEN $D=' PERSTANAREA '//$QM//OLD//$QM
  endif;
endif;

IF CSTANTXT NE BLANK
THEN
  OLD=$SBF(CSTANTXT,0,1,'*')
  NEW=$SBF(CSTANTXT,1,1,'*')
  $E=' CSTANTXT '//$QM//CSTANTXT//$QM

  IF $SBF(OLD,0,1) NE '$' and $SBF(NEW,0,1) NE '$'
  THEN $C=' PERSTANTXT TO '//$QM//NEW//$QM
  endif;

  IF $SBF(OLD,0,1) EQ '$'
  THEN $A=' PERSTANTXT '//$QM//NEW//$QM
  endif;

  IF $SBF(NEW,0,1) EQ '$' (only occurs if ITOA = 'ZZ')
  THEN $D=' PERSTANTXT '//$QM//OLD//$QM
  endif;
endif;

IF CSTEP2 NE BLANK
THEN
  OLD=$SBF(CSTEP2,0,1,'*')
  NEW=$SBF(CSTEP2,1,1,'*')
  $E=' CSTEP2 '//$QM//CSTEP2//$QM

  IF $SBF(OLD,0,1) NE '$' and $SBF(NEW,0,1) NE '$'
  THEN $C=' PERSTEP2 TO '//$QM//NEW//$QM
  endif;
```

```
    IF $SBF(OLD,0,1) EQ '$'
    THEN $A=' PERSTEP2 '//$QM//NEW//$QM
    endif;

    IF $SBF(NEW,0,1) EQ '$'
    THEN $D=' PERSTEP2 '//$QM//OLD//$QM
    endif;
  endif;

  IF CSTIND NE BLANK
  THEN
    OLD=$SBF(CSTIND,0,1,'*')
    NEW=$SBF(CSTIND,1,1,'*')
    $E=' CSTIND '//$QM//CSTIND//$QM

    IF $SBF(OLD,0,1) NE '$' and $SBF(NEW,0,1) NE '$'
    THEN $C=' PERSTIND TO '//$QM//NEW//$QM
    endif;

    IF $SBF(OLD,0,1) EQ '$'
    THEN $A=' PERSTIND '//$QM//NEW//$QM
    endif;

    IF $SBF(NEW,0,1) EQ '$'
    THEN $D=' PERSTIND '//$QM//OLD//$QM
    endif;
  endif;

  IF CSTRCTR NE BLANK
  THEN
    OLD=$SBF(CSTRCTR,0,1,'*')
    NEW=$SBF(CSTRCTR,1,1,'*')
    $E=' CSTRCTR '//$QM//CSTRCTR//$QM

    IF $SBF(OLD,0,1) NE '$' and $SBF(NEW,0,1) NE '$'
    THEN $C=' PERSTRCTR '//$QM//OLD//$QM//' TO '
            //$QM//NEW//$QM
    endif;

    IF $SBF(OLD,0,1) EQ '$'
    THEN $A=' PERSTRCTR '//$QM//NEW//$QM
    endif;

    IF $SBF(NEW,0,1) EQ '$'
    THEN $D=' PERSTRCTR '//$QM//OLD//$QM
    endif;
  endif;

  IF CTOA NE BLANK
  THEN
    OLD=$SBF(CTOA,0,1,'*')
    NEW=$SBF(CTOA,1,1,'*')
    $E=' CTOA '//$QM//CTOA//$QM
    IF $SBF(OLD,0,1) NE '$' and $SBF(NEW,0,1) NE '$'
    THEN $C=' PERTOA '//$QM//OLD//$QM//'TO'//$QM//NEW//$QM
    endif;

    IF $SBF(OLD,0,1) EQ '$'
    THEN $A=' PERTOA '//$QM//NEW//$QM
    endif;

    IF $SBF(NEW,0,1) EQ '$' (only occurs if ITOA = 'ZZ')
    THEN $D=' PERTOA '//$QM//OLD//$QM
    endif;
  endif;

  IF CTOUR NE BLANK
  THEN
    OLD=$SBF(CTOUR,0,1,'*')
    NEW=$SBF(CTOUR,1,1,'*')
    $E=' CTOUR '//$QM//CTOUR//$QM

    IF $SBF(OLD,0,1) NE '$' and $SBF(NEW,0,1) NE '$'
    THEN $C=' PERTOUR TO '//$QM//NEW//$QM
    endif;

    IF $SBF(OLD,0,1) EQ '$'
    THEN $A=' PERTOUR '//$QM//NEW//$QM
    endif;

    IF $SBF(NEW,0,1) EQ '$'
    THEN $D=' PERTOUR '//$QM//OLD//$QM
    endif;
  endif;

  IF CVET NE BLANK
  THEN
    OLD=$SBF(CVET,0,1,'*')
    NEW=$SBF(CVET,1,1,'*')
    $E=' CVET '//$QM//CVET//$QM

    IF $SBF(OLD,0,1) NE '$' and $SBF(NEW,0,1) NE '$'
    THEN $C=' PERVET TO '//$QM//NEW//$QM
    endif;

    IF $SBF(OLD,0,1) EQ '$'
    THEN $A=' PERVET '//$QM//NEW//$QM
    endif;

    IF $SBF(NEW,0,1) EQ '$'
    THEN $D=' PERVET '//$QM//OLD//$QM
    endif;
  endif;

  IF CWGIE NE BLANK
  THEN
    OLD=' ',OLD=$SBF(CWGIE,0,1,'*')
    NEW=$SBF(CWGIE,1,1,'*')
    $E=' CWGIE '//$QM//CWGIE//$QM

    IF $SBF(OLD,0,1) NE '$' and $SBF(NEW,0,1) NE '$'
    THEN $C=' PERWGIE TO '//$QM//NEW//$QM
    endif;

    IF $SBF(OLD,0,1) EQ '$'
    THEN $A=' PERWGIE '//$QM//NEW//$QM
    endif;

    IF $SBF(NEW,0,1) EQ '$' (only occurs if ITOA = 'ZZ')
    THEN $D=' PERWGIE '//$QM//OLD//$QM
    endif;
  endif;

  IF GCUTOFFDT NE BLANK
  THEN  $E=' GCUTOFFDT '//$QM//GCUTOFFDT//$QM
  endif;

  IF GPROJUPDTD NE BLANK
  THEN  $E=' GPROJUPDTD '//$QM//GPROJUPDTD//$QM
  endif;

  IF IACTREQ NE BLANK
  THEN  $E=' IACTREQ '//$QM//IACTREQ//$QM
  endif;

  IF IAFF NE BLANK
  THEN  $E=' IAFF '//$QM//IAFF//$QM
  endif;

  IF IAL NE BLANK
  THEN  $E=' IAL '//$QM//IAL//$QM
  endif;

  IF IANO NE BLANK
  THEN  $E=' IANO '//$QM//IANO//$QM
  endif;

  IF IANPRPAY NE BLANK
  THEN  $E=' IANPRPAY '//$QM//IANPRPAY//$QM
  endif;

  IF IANTYSAL NE BLANK
  THEN  $E=' IANTYSAL '//$QM//IANTYSAL//$QM
  endif;

  IF IANTYST NE BLANK
  THEN  $E=' IANTYST '//$QM//IANTYST//$QM
  endif;

  IF IAORG NE BLANK
  THEN  $E=' IAORG '//$QM//IAORG//$QM
```

```
endif;

IF IAPNTE NE BLANK
THEN  $E=' IAPNTE '//$QM//IAPNTE//$QM
endif;

IF ICCANO NE BLANK
THEN  $E=' ICCANO '//$QM//ICCANO//$QM
endif;

IF ICCNOA NE BLANK
THEN  $E=' ICCNOA '//$QM//ICCNOA//$QM
endif;

IF ICCTOA NE BLANK
THEN  $E=' ICCTOA '//$QM//ICCTOA//$QM
endif;

IF ICIT NE BLANK
THEN  $E=' ICIT '//$QM//ICIT//$QM
endif;

IF ICSEOD NE BLANK
THEN  $E=' ICSEOD '//$QM//ICSEOD//$QM
endif;

IF ICTPCLASS NE BLANK
THEN  $E=' ICTPCLASS '//$QM//ICTPCLASS//$QM
endif;

IF IDETC NE BLANK
THEN  $E=' IDETC '//$QM//IDETC//$QM
endif;

IF IDETONTE NE BLANK
THEN  $E=' IDETONTE '//$QM//IDETONTE//$QM
endif;

IF IDEVC NE BLANK
THEN  $E=' IDEVC '//$QM//IDEVC//$QM
endif;

IF IDEVDI NE BLANK
THEN  $E=' IDEVDI '//$QM//IDEVDI//$QM
endif;

IF IDEVLAST NE BLANK
THEN  $E=' IDEVLAST '//$QM//IDEVLAST//$QM
endif;

IF IDEVNTE NE BLANK
THEN  $E=' IDEVNTE '//$QM//IDEVNTE//$QM
endif;
```

```
IF IDOA NE BLANK
THEN  $E=' IDOA '//$QM//IDOA//$QM
endif;

IF IDOB NE BLANK
THEN  $E=' IDOB '//$QM//IDOB//$QM
endif;

IF IDOG NE BLANK
THEN  $E=' IDOG '//$QM//IDOG//$QM
endif;

IF IETP NE BLANK
THEN  $E=' IETP '//$QM//IETP//$QM
endif;

IF IFLSA NE BLANK
THEN  $E=' IFLSA '//$QM//IFLSA//$QM
endif;

IF IFLSADT NE BLANK
THEN  $E=' IFLSADT '//$QM//IFLSADT//$QM
endif;

IF IGR NE BLANK
THEN  $E=' IGR '//$QM//IGR//$QM
endif;

IF IHQ NE BLANK
THEN  $E=' IHQ '//$QM//IHQ//$QM
endif;

IF IHRS NE BLANK
THEN  $E=' IHRS '//$QM//IHRS//$QM
endif;

IF IINDT NE BLANK
THEN  $E=' IINDT '//$QM//IINDT//$QM
endif;

IF IINORIGDT NE BLANK
THEN  $E=' IINORIGDT '//$QM//IINORIGDT//$QM
endif;

IF ILCD NE BLANK
THEN  $E=' ILCD '//$QM//ILCD//$QM
endif;

IF ILEI NE BLANK
THEN  $E=' ILEI '//$QM//ILEI//$QM
endif;

IF ILWOPSTRT NE BLANK
THEN  $E=' ILWOPSTRT '//$QM//ILWCPSTRT//$QM
```

```
IF IOVERLAP NE BLANK
THEN  $E=' IOVERLAP '//$QM//IOVERLAP//$QM
endif;

IF IOVNTE NE BLANK
THEN  $E=' IOVNTE '//$QM//IOVNTE//$QM
endif;

IF IOVRD NE BLANK
THEN  $E=' IOVRD '//$QM//IOVRD//$QM
endif;

IF IPAYB NE BLANK
THEN  $E=' IPAYB '//$QM//IPAYB//$QM
endif;

IF IPGVT NE BLANK
THEN  $E=' IPGVT '//$QM//IPGVT//$QM
endif;

IF IPOSNO NE BLANK
THEN  $E=' IPOSNO '//$QM//IPOSNO//$QM
endif;

IF IPRA NE BLANK
THEN  $E=' IPRA '//$QM//IPRA//$QM
endif;

IF IPRADI NE BLANK
THEN  $E=' IPRADI '//$QM//IPRADI//$QM
endif;

IF IPRANTE NE BLANK
THEN  $E=' IPRANTE '//$QM//IPRANTE//$QM
endif;

IF IPRMNTE NE BLANK
THEN  $E=' IPRMNTE '//$QM//IPRMNTE//$QM
endif;

IF IPROJNO NE BLANK
THEN  $E=' IPROJNO '//$QM//IPROJNO//$QM
endif;

IF IPWGI NE BLANK
THEN  $E=' IPWGI '//$QM//IPWGI//$QM
endif;

IF IRACE NE BLANK
THEN  $E=' IRACE '//$QM//IRACE//$QM
endif;

IF IREMC NE BLANK
THEN  $E=' IREMC '//$QM//IREMC//$QM
```

```
endif;

IF INAMEOR NE BLANK
THEN  $E=' INAMEOR '//$QM//INAMEOR//$QM
endif;

IF INAMEOR3 NE BLANK
THEN  $E=' INAMEOR3 '//$QM//INAMEOR3//$QM
endif;

IF INCPP NE BLANK
THEN  $E=' INCPP '//$QM//INCPP//$QM
endif;

IF INEWIDN NE BLANK
THEN  $E=' INEWIDN '//$QM//INEWIDN//$QM
endif;

IF INSCA NE BLANK
THEN  $E=' INSCA '//$QM//INSCA//$QM
endif;

IF INSCADT NE BLANK
THEN  $E=' INSCADT '//$QM//INSCADT//$QM
endif;

IF INTY NE BLANK
THEN  $E=' INTY '//$QM//INTY//$QM
endif;

IF IOCCE NE BLANK
THEN  $E=' IOCCE '//$QM//IOCCE//$QM
endif;

IF IOCCSUF NE BLANK
THEN  $E=' IOCCSUF '//$QM//IOCCSUF//$QM
endif;

IF IOPERATOR NE BLANK
THEN  $E=' IOPERATOR '//$QM//IOPERATOR//$QM
endif;

IF IORGCODE NE BLANK
THEN  $E=' IORGCODE '//$QM//IORGCODE//$QM
endif;

IF IORGDTIN NE BLANK
THEN  $E=' IORGDTIN '//$QM//IORGDTIN//$QM
endif;

IF IOUTSK NE BLANK
THEN  $E=' IOUTSK '//$QM//IOUTSK//$QM
endif;
```

```
endif;

IF IREMNS NE BLANK
THEN  $E=' IREMNS '//$QM//IREMNS//$QM
endif;

IF IRESV1 NE BLANK
THEN  $E=' IRESV1 '//$QM//IRESV1//$QM
endif;

IF IRESV2 NE BLANK
THEN  $E=' IRESV2 '//$QM//IRESV2//$QM
endif;

IF IRESV3 NE BLANK
THEN  $E=' IRESV3 '//$QM//IRESV3//$QM
endif;

IF IRESV4 NE BLANK
THEN  $E=' IRESV4 '//$QM//IRESV4//$QM
endif;

IF IRESV5 NE BLANK
THEN  $E=' IRESV5 '//$QM//IRESV5//$QM
endif;

IF IRETDI NE BLANK
THEN  $E=' IRETDI '//$QM//IRETDI//$QM
endif;

IF IRETNTE NE BLANK
THEN  $E=' IRETNTE '//$QM//IRETNTE//$QM
endif;

IF IRTMT NE BLANK
THEN  $E=' IRTMT '//$QM//IRTMT//$QM
endif;

IF ISALARY NE BLANK
THEN  $E=' ISALARY '//$QM//ISALARY//$QM
endif;

IF ISCCE NE BLANK
THEN  $E=' ISCCE '//$QM//ISCCE//$QM
endif;

IF ISCD NE BLANK
THEN  $E=' ISCD '//$QM//ISCD//$QM
endif;

IF ISCH NE BLANK
THEN  $E=' ISCH '//$QM//ISCH//$QM
endif;
```

```
IF ISD NE BLANK
THEN  $E=' ISD '//$QM//ISD//$QM
endif;

IF ISECCL NE BLANK
THEN  $E=' ISECCL '//$QM//ISECCL//$QM
endif;

IF ISEPCODE NE BLANK
THEN  $E=' ISEPCODE '//$QM//ISEPCODE//$QM
      STMT=' PRIMSEPCODE '//$QM//ISEPCODE//$QM//
           //' PRIMSEPDTE '//$QM//IDOA//$QM
endif;

IF ISEPCOMP NE BLANK
THEN  $E=' ISEPCOMP '//$QM//ISEPCOMP//$QM
endif;

IF ISERIAL NE BLANK
THEN  $E=' ISERIAL '//$QM//ISERIAL//$QM
endif;

IF ISEX NE BLANK
THEN  $E=' ISEX '//$QM//ISEX//$QM
endif;

IF ISFN NE BLANK
THEN  $E=' ISFN '//$QM//ISFN//$QM
endif;

IF ISIGN NE BLANK
THEN  $E=' ISIGN '//$QM//ISIGN//$QM
endif;

IF ISREF NE BLANK
THEN  $E=' ISREF '//$QM//ISREF//$QM
endif;

IF ISSNOTH NE BLANK
THEN  $E=' ISSNOTH '//$QM//ISSNOTH//$QM
endif;

IF ISTAN NE BLANK
THEN  $E=' ISTAN '//$QM//ISTAN//$QM
endif;

IF ISTEP2 NE BLANK
THEN  $E=' ISTEP2 '//$QM//ISTEP2//$QM
endif;

IF ISUSPEN NE BLANK
THEN  $E=' ISUSPEN '//$QM//ISUSPEN//$QM
endif;
```

```
IF ITIDN NE BLANK
THEN  $E=' ITIDN '//$QM//ITIDN//$QM
endif;

IF ITNAME NE BLANK
THEN  $E=' ITNAME '//$QM//ITNAME//$QM
endif;

IF ITOA NE BLANK
THEN  $E=' ITOA '//$QM//ITOA//$QM
endif;

IF ITOASK NE BLANK
THEN  $E=' ITOASK '//$QM//ITOASK//$QM
endif;

IF ITOUR NE BLANK
THEN  $E=' ITOUR '//$QM//ITOUR//$QM
endif;

IF ITRANS NE BLANK
THEN  $E=' ITRANS '//$QM//ITRANS//$QM
endif;

IF IVET NE BLANK
THEN  $E=' IVET '//$QM//IVET//$QM
endif;

IF IWGIE NE BLANK
THEN  $E=' IWGIE '//$QM//IWGIE//$QM
endif;

IF SAFF NE BLANK
THEN  $E=' SAFF '//$QM//SAFF//$QM
endif;

IF SANF NE BLANK
THEN  $E=' SANF '//$QM//SANF//$QM
endif;

IF SANO NE BLANK
THEN  $E=' SANO '//$QM//SANO//$QM
endif;

IF SAPNTE NE BLANK
THEN  $E=' SAPNTE '//$QM//SAPNTE//$QM
endif;

IF SCEIL NE BLANK
THEN  $E=' SCEIL '//$QM//SCEIL//$QM
endif;

IF SDEVC NE BLANK
THEN  $E=' SDEVC '//$QM//SDEVC//$QM
```

```
endif;

IF SDOB NE BLANK
THEN  $E=' SDOB '//$QM//SDOB//$QM
endif;

IF SDOG NE BLANK
THEN  $E=' SDOG '//$QM//SDOG//$QM
endif;

IF SGR NE BLANK
THEN  $E=' SGR '//$QM//SGR//$QM
endif;

IF SLEI NE BLANK
THEN  $E=' SLEI '//$QM//SLEI//$QM
endif;

IF SNAMEOR NE BLANK
THEN  $E=' SNAMEOR '//$QM//SNAMEOR//$QM
endif;

IF SPAYB NE BLANK
THEN  $E=' SPAYB '//$QM//SPAYB//$QM
endif;

IF SPGVT NE BLANK
THEN  $E=' SPGVT '//$QM//SPGVT//$QM
endif;

IF SPROJNO NE BLANK
THEN  $E=' SPROJNO '//$QM//SPROJNO//$QM
endif;

IF SRTMT NE BLANK
THEN  $E=' SRTMT '//$QM//SRTMT//$QM
endif;

IF SSALARY NE BLANK
THEN  $E=' SSALARY '//$QM//SSALARY//$QM
endif;

IF SSCD NE BLANK
THEN  $E=' SSCD '//$QM//SSCD//$QM
endif;

IF SSCH NE BLANK
THEN  $E=' SSCH '//$QM//SSCH//$QM
endif;

IF SSERIAL NE BLANK
THEN  $E=' SSERIAL '//$QM//SSERIAL//$QM
endif;
```

```
  IF SSFN NE BLANK
  THEN  $E=' SSFN '//$QM//SSFN//$QM
  endif;

  IF SSTEP2 NE BLANK
  THEN  $E=' SSTEP2 '//$QM//SSTEP2//$QM
  endif;

  IF STOUR NE BLANK
  THEN  $E=' STOUR '//$QM//STOUR//$QM
  endif;

endproc BLDSTMT

DEC SSNCHG
 CSIGN    Input value from disk to check for SSN Change
 OLD        old side of CSIGN
 NEW        new side of CSIGN
 $C         statement to change DL/ID (SSNOR)
 $E         part of statement for INTERFACE record

enddec SSNCHG

PROC SSNCHG
  IF CSIGN NE BLANK
  THEN
    OLD=$SBF(CSIGN,0,1,'*')
    NEW=$SBF(CSIGN,1,1,'*')
    $E=' CSIGN '//$QM//CSIGN//$QM

    IF $SBF(OLD,0,1) NE '$' and $SBF(NEW,0,1) NE '$'
    THEN $C=FILE//$QM//OLD//$QM//' TO '//
            $QM//NEW//$QM
    endif;

    CHANGE &($C)

    $C = BLANK
  endif;
endproc SSNCHG

DEC ADDREC
 $A         all values to be added
 $C         all values to be changed
 $D         all values to be deleted
 LVL        current array level
 $E         all values to be added to INTERFACE
 FILE       file changes are to be applied to
 STMT       SEPCODE and SEPDTE for PRIMSEP add only
 STMTS      concatenation of array values for execution

enddec ADDREC

PROC ADDREC
```

```
IF $C NE BLANK
THEN
  LVL = 1
  STMTS = 'FOR '//FILE//' CHANGE '
  DO UNTIL LVL GT $CM
    STMTS = STMTS//$C(LVL)
    LVL = LVL + 1
  enddo;

  &STMTS     (execute change statememt)

  $C=BLANK, STMTS=BLANK
endif;

IF $A NE BLANK or STMT NE BLANK
THEN
  LVL = 1
  STMTS = 'FOR '//FILE//' ADD '
  DO UNTIL LVL GT $AM
    STMTS = STMTS//$A(LVL)
    LVL = LVL + 1
  enddo;

  &STMTS &STMT      (execute add statement)

  $A=BLANK, STMTS=BLANK
endif;

IF $D NE BLANK AND ITOA NE 'ZZ'
THEN
  LVL = 1
  STMTS = 'FOR '//FILE//' DELETE '
  DO UNTIL LVL GT $DM
    STMTS = STMTS//$D(LVL)
    LVL = LVL + 1
  enddo;

  &STMTS      (execute delete statement)

  $D=BLANK, STMTS=BLANK, STMT=BLANK
endif;

IF $E NE BLANK
THEN
  LVL = 1
  STMTS = 'ADD INTERFACE '//INTERFACE
  DO UNTIL LVL GT $EM
    STMTS = STMTS//$E(LVL)
    LVL = LVL + 1
  enddo;
  &STMTS          (update INTERFACE file)

  $E=BLANK, STMTS=BLANK
endif;
```

```
DEC ADDNEW
 $A          all values to be added
 NEWREC      flag to designate new record
 ISIGN     INPUT value from disk containing record DL/ID

enddec ADDNEW

PROC ADDNEW

  IF $A NE BLANK
  THEN
    LVL = 1
    STMTS = 'ADD NEW PERSIGN '//$QM//ISIGN//$QM
    DO UNTIL LVL GT $AM
      STMTS = STMTS//$A(LVL)
      LVL = LVL + 1
    enddo;
    &STMTS          (add new PERSIGN record)

    $A=BLANK, STMTS=BLANK
  endif;

  NEWREC = 'NO'

endproc ADDNEW

DEC NEWSEP
  FILE     value received in READREC 'PERSIGN'
  ISIGN  value received on input
enddec NEWSEP

PROC NEWSEP

  FOR &FILE '&ISIGN' MOVE REPLACE TO PRIMSEP

  DELETE &FILE '&ISIGN'

  FILE = 'PRIMSEP'
endproc NEWSEP
endproc PRCHGLOAD
```

4.3.5 <u>PRCATRSF - TRANSFER COMPONENT ACCESS</u>

4.3.5.1 PURPOSE

When an employee has been nominated for reassignment consideration to another component, the owning component can electronically provide the requesting component with the employee's biographic data. If a Component wants to extend a Purge Date (PRGDTE) to avoid losing a record from it's access, that Component must use the PRACCUPDT (Access Update) menu to change the PRGDTE. See flow diagram in Figure 9 of the Appendix.

4.3.5.2 GENERAL INFORMATION

Proc Name	: PRCATRSF
Language Used:	GIM POL
Initiated By	: Component owning the employee
Input	: Menu parameters SSNOR of transferring employee SYSMAN2 Signon ORG PRGDTE (Purge Date)
Output	: The SSNOR will be copied from one SEGACCESS file to another.

4.3.5.3 FUNCTIONAL DESCRIPTION

The PRCATRSF procedure is used by one component to pass an employee's SSNOR to another component, thus creating a link to the employee's biographic data. This will be done when that employee is being nominated for an assignment to another component or transferred temporarily to the other component. The sending component should enter a Purge Date. If the employee is actually assigned to the receiving office, the Purge Date will be removed by an entry in the HRS2 INTERFACE file via the PRCAUPDT procedure. If the employee is not reassigned, the entire entry made for the employee in the receiving office's file will be removed by the PRNTEPRG procedure. The SEGACCESS file with the use of Security update keys will allow the component owning the record to delete an SSNOR passed in error to another component. Access to this procedure will be controlled through a key in the SYSMAN2 file.

NOTE: If Access is required to an official record and no ownership has been established in PRIM for that record, the component must request the PRIM Data Base Manager to transfer the record.

4.3.5.4 INPUT

This procedure will utilize the following menu which will be stored in the MENU-FORMATS file:

```
E PRCATRSF   (Component Access Transfer)

     ACTION: _ (A) ADD (D) DELETE (X) EXIT
```

SSNOR/STATUS: _________ - _________ - _________ - _________ -
STATUS: A - Active S- Separated

SYSMAN2	PURGE DATE	SYSMAN2	PURGE DATE
______	______	______	______
______	______	______	______
______	______	______	______
______	______	______	______
______	______	______	______
______	______	______	______
______	______	______	______
______	______	______	______

<u>MENU</u> <u>ITEM</u> <u>DESCRIPTION</u>

(R) ACTION:	Action (A,D,X)	(LA01)
(R) SSNOR:	Social Security Number	(LN09)
(R) STATUS:	Active or Separated Employee	(LA01)
(R) SYSMAN2:	ORG Receiving Employee Info	(LA06)
(O) PURGE DATE:	Purge Date (YYMMDD)	(LN06)

4.3.5.5 OUTPUT

```
If Action = 'X'
then  - exit the procedure.
endif;

If Purge Date  = blank
then  - create PRGDTE (current date + 90)
endif;

If Action = 'A' or 'D' and null SSNOR, STATUS, and/or null SYSMAN2
then  - issue error message (Required fields must be present)
endif;

If Action = 'A' and present SSNOR, STATUS and SYSMAN2
then  - use the SYSMAN2 Signon ORG to verify ownership (service
        designation on record EQ SD field of SELECTION Criteria).
```

```
    If any menu STATUS = 'A'
    then  - Build add statement for each active SYSMAN2 entry
          - Add SSNOR, and PRGDTE to each SEGACCESS segment as
            designated by SYSMAN2 entry(ies) on menu.  Active
            SSNOR(s) will be added to SYSMAN2+A

             EX: Add (SYSMAN2+A) '_______' PRGDTE '______'
    endif;

    If any menu STATUS = 'S'
    then  - Build add statement for each separated SYSMAN2 entry
          - Add SSNOR, and PRGDTE to each SEGACCESS segment as
            designated by SYSMAN2 entry(ies) on menu.  Separated
            SSNOR(s) will be added to SYSMAN2+S

             EX: Add (SYSMAN2+S) '_______' PRGDTE '______'
    endif;
 endif;

 If Action = 'D' and present SYSMAN2 on menu NE SYSMAN2 Signon ORG
 then  - Use the SYSMAN2 Signon ORG to verify Ownership (SD on
         record EQ SD field of SELECTION Criteria).

    If any menu STATUS = 'A'
    then  - Build delete statements for each SYSMAN2 entry
            on the menu.
          - Delete SSNOR with null SDTE from each active SEGACCESS
            segment as designated by the SYSMAN2 entry(ies)
            on the menu.  Active records will be deleted from
            SYSMAN2+A.

            EX: Delete (SYSMAN2+A) '_______' when null SDTE.
    endif;

    If any menu STATUS = 'S'
    then  - Build delete statements for each SYSMAN2 entry
            on the menu.
          - Delete SSNOR with null SSDTE from each separated
            SEGACCESS segment as designated by the SYSMAN2
            entry(ies) on the menu. Separated records will be
            deleted from SYSMAN2+S.

            EX: Delete (SYSMAN2+S) '_______' when null SSDTE.
    endif;
 endif;
```

4.3.5.6 GENERAL PROGRAM SPECIFICATIONS

A component can transfer an employee's record if the component owns that record. Ownership is valid if the SD of the record equals the SD in the components SELECTION Criteria (SD is first segment of Criteria). The initiating component is determined by the Signon ORG ($UNAME1).

4.3.5.7 DETAILED PROGRAM SPECIFICATIONS

```
DEC PRCATRSF
  BLANK      value ' '
  ERRSW      value ' '
  SLVL       value 11 (SYSMAN2 level)
  SWCH       value 'OFF'
  $A         single dimensional array
    $A(2)    area containing Action
    $A(3)    1st SSNOR
    $A(4)    Status for $A(3)
    $A(5)    2nd SSNOR
    $A(6)    Status for $A(5)
    $A(7)    3rd SSNOR
    $A(8)    Status for $A(7)
    $A(9)    4th SSNOR
    $A(10)   Status for $A(9)
    $A(11)   1st SYSMAN2
    $A(12)   Purge date for 1st SYSMAN2
    $A(13)   2nd SYSMAN2
    $A(14)   Purge date for 2nd SYSMAN2
    $A(15)   3rd SYSMAN2
    $A(16)   Purge date for 3rd SYSMAN2
    $A(17)   4th SYSMAN2
    $A(18)   Purge date for 4th SYSMAN2
    $A(19)   5th SYSMAN2
    $A(20)   Purge date for 5th SYSMAN2
    $A(21)   6th SYSMAN2
    $A(22)   Purge date for 6th SYSMAN2
    $A(23)   7th SYSMAN2
    $A(24)   Purge date for 7th SYSMAN2
    $A(25)   8th SYSMAN2
    $A(26)   Purge date for 8th SYSMAN2
    $A(27)   9th SYSMAN2
    $A(28)   Purge date for 9th SYSMAN2
    $A(29)   10th SYSMAN2
    $A(30)   Purge date for 10th SYSMAN2
    $A(31)   11th SYSMAN2
    $A(32)   Purge date for 11th SYSMAN2
    $A(33)   12th SYSMAN2
    $A(34)   Purge date for 12th SYSMAN2
    $A(35)   13th SYSMAN2
    $A(36)   Purge date for 13th SYSMAN2
    $A(37)   14th SYSMAN2
    $A(38)   Purge date for 14th SYSMAN2
    $A(39)   15th SYSMAN2
    $A(40)   Purge date for 15th SYSMAN2
    $A(41)   16th SYSMAN2
    $A(42)   Purge date for 16th SYSMAN2 enddec PRCATRSF
```

```
PROC PRCATRSF

  READ  menu into $A arrays

  IF $A(2) EQ 'X'
  THEN  - RUN EXIT-FOR-NOMORE
  endif;

  IF $A(2) EQ 'A' or 'D' and
      (($A(3) NE BLANK and $A(4) EQ BLANK) or
       ($A(3) EQ BLANK and $A(4) NE BLANK) or
       ($A(5) NE BLANK and $A(6) EQ BLANK) or
       ($A(5) EQ BLANK and $A(6) NE BLANK) or
       ($A(7) NE BLANK and $A(8) EQ BLANK) or
       ($A(7) EQ BLANK and $A(8) NE BLANK) or
       ($A(9) NE BLANK and $A(10) EQ BLANK) or
       ($A(9) EQ BLANK and $A(10) NE BLANK))
  THEN  - PRINT message that both SSNOR and STATUS
          must be entered
        - RUN EXIT-FOR-RESTART
  endif;

  DO UNTIL SWCH EQ 'ON' or SLVL EQ 43
    IF $A(SLVL) EQ BLANK
    THEN  - SLVL = SLVL + 2
    ELSE  - SWCH EQ 'ON'
    endif;
  enddo;

  IF SWCH EQ 'OFF'
  THEN  - PRINT message stating at least one
          SYSMAN2 value must be entered
        - RUN EXIT-FOR-RESTART
  endif;


  IF $A(2) EQ 'A'
  THEN  - RUN VERIFYCASE
    IF ERRSW EQ BLANK
    THEN  - RUN SSNCASE
          - RUN SDCASE
    endif;
    IF ERRSW EQ BLANK
    THEN  - RUN ADDCASE
    endif;
    RUN EXIT-FOR-MORE
  endif;

  IF $A(2) EQ 'D'
  THEN  - RUN VERIFYCASE
    IF ERRSW EQ BLANK
    THEN  - RUN SSNCASE
          - RUN SDCASE
    endif;
    IF ERRSW EQ BLANK
    THEN  - RUN DELETECASE
    endif;
    RUN EXIT-FOR-MORE
  endif;

DEC VERIFYCASE
  SEGMENT        value $UNAME1
  CRITERIA       temporary buffer area
  $B             single dimensional arrays
  $BM            count of $B arrays
  $W             current value
enddec VERIFYCASE

PROC VERIFYCASE

  (execute statement to acquire SD values from the
   SELECTION segment for the initiating ORG)
  FOR &SEGMENT ACQUIRE SLCRITERIA

  DO for all Criteria lines in the file
    GET SLCRITERIA (use GFAV and GNAV)
    CRITERIA = $W
    $B = $SBF(CRITERIA,0,1,'*')
  enddo;

  IF $BM EQ 0  (no SD value found)
  THEN  - PRINT message that initiator does not
          have ownership of record
        - ERRSW = '1'
  endif;
endproc VERIFYCASE;

DEC SSNCASE
  $A         single dimensional array
    $A(2)    area containing Action
    $A(3)    1st SSNOR
    $A(4)    Status for $A(3)
    $A(5)    2nd SSNOR
    $A(6)    Status for $A(5)
    $A(7)    3rd SSNOR
    $A(8)    Status for $A(7)
    $A(9)    4th SSNOR
    $A(10)   Status for $A(9)
  $B         single dimensional arrays
  $C      value ' '
             buffer area for active SSNORs
  $D      value ' '
             buffer area for separated data
enddec SSNCASE

PROC SSNCASE
```

```
   IF $A(4) EQ 'A'
   THEN  - add $A(3) to active SSNOR statement
         - $C = $C//$QM//$A(3)//$QM
   endif;

   IF $A(4) EQ 'S'
   THEN  - add $A(3) to separated SSNOR statement
         - $D = $D//$QM//$A(3)//$QM
   endif;

   IF $A(6) EQ 'A'
   THEN  - add $A(5) to active SSNOR statement
         - $C = $C//$QM//$A(5)//$QM
   endif;

   IF $A(6) EQ 'S'
   THEN  - add $A(5) to separated SSNOR statement
         - $D = $D//$QM//$A(5)//$QM
   endif;

   IF $A(8) EQ 'A'
   THEN  - add $A(7) to active SSNOR statement
         - $C = $C//$QM//$A(7)//$QM
   endif;

   IF $A(8) EQ 'S'
   THEN  - add $A(7) to separated query statement
         - $D = $D//$QM//$A(7)//$QM
   endif;

   IF $A(10) EQ 'A'
   THEN  - add $A(9) to active query statement
         - $C = $C//$QM//$A(9)//$QM
   endif;

   IF $A(10) EQ 'S'
   THEN  - add $A(9) to separated SSNOR statement
         - $D = $D//$QM//$A(9)//$QM
   endif;
endproc SSNCASE

DEC SDCASE
  BLANK      value ' '
  BLVL       value 1
  ERRSW      value ' '
  $B         single dimensional arrays
  $BM        count of used $B arrays
  $C     buffer area containing active SSNORs
  $D     buffer area containing separated SSNORs
  SDSCAN     value ' '
             temporary buffer area for SD value
enddec sdcase

PROC SDCASE
```

```
  DO UNTIL BLVL EQ $BM
     (build SD scan portion of statement)
    SDSCAN = SDSCAN//' WITH $SCAN(PERSD, '
      //$QM//$B(BLVL)//$QM
    IF BLVL NE $BM
    THEN  - BLVL = BLVL + 1
          - SDSCAN = SDSCAN//' OR '
    endif;
  enddo;

  IF $C NE BLANK
  THEN  - (scan for SD value in PERSIGN)
          FOR PERSIGN &($C) &SDSCAN ACQUIRE PERSD
    IF NO VALUE ACQUIRED (SD does not match)
    THEN  - PRINT message stating initiator does not
            have ownership of record
          - VOPRINT &($C)

    endif;
  endif;

  IF $D NE BLANK
  THEN  - (scan for SD value in PRIMSEP)
          FOR PRIMSEP &($D) &SDSCAN ACQUIRE PERSD
    IF NO VALUE ACQUIRED (SD does not match)
    THEN  - PRINT message stating initiator does not
            have ownership of record
          - VOPRINT $D
    endif;
  endif;
endproc SDCASE

DEC ADDCASE
  BLANK       value ' '
  $C       buffer area containing active SSNORs
  $D       buffer area containing separated SSNORs
  $A(11)      1st SYSMAN2
  $A(12)      Purge date for 1st SYSMAN2
  $A(13)      2nd SYSMAN2
  $A(14)      Purge date for 2nd SYSMAN2
  $A(15)      3rd SYSMAN2
  $A(16)      Purge date for 3rd SYSMAN2
  $A(17)      4th SYSMAN2
  $A(18)      Purge date for 4th SYSMAN2
  $A(19)      5th SYSMAN2
  $A(20)      Purge date for 5th SYSMAN2
  $A(21)      6th SYSMAN2
  $A(22)      Purge date for 6th SYSMAN2
  $A(23)      7th SYSMAN2
  $A(24)      Purge date for 7th SYSMAN2
  $A(25)      8nd SYSMAN2
  $A(26)      Purge date for 8th SYSMAN2
  $A(27)      9rd SYSMAN2
  $A(28)     Purge date for 9th SYSMAN2
  $A(29)     10th SYSMAN2
  $A(30)     Purge date for 10th SYSMAN2
  $A(31)     11th SYSMAN2
  $A(32)     Purge date for 11th SYSMAN2
  $A(33)     12th SYSMAN2
  $A(34)     Purge date for 12th SYSMAN2
  $A(35)     13th SYSMAN2
  $A(36)     Purge date for 13th SYSMAN2
  $A(37)     14th SYSMAN2
  $A(38)     Purge date for 14th SYSMAN2
  $A(39)     15th SYSMAN2
  $A(40)     Purge date for 15th SYSMAN2
  $A(41)     16th SYSMAN2
  $A(42)     Purge date for 16th SYSMAN2
  SLVL       value 11 (SYSMAN2 array level)
  PURGE      buffer area for current purge date
  TPRGDTE    $ODATE  (GIM current date)
  TSTMT      buffer containing verb and file name
enddec ADDCASE

PROC ADDCASE

  IF $C NE BLANK
  THEN  - (build and execute add statement for each
           SYSMAN2+A)
    DO UNTIL SLVL EQ 43
      IF $A(SLVL) NE BLANK
      THEN  - (SYSMAN2+A is file for add statement)
        IF $A(SLVL+1) NE BLANK
        THEN  - PURGE = ' PRGDTE '//$QM//$A(SLVL+1)//$QM
        ELSE  - PURGE = ' PRGDTE '//$QM//TPRGDTE//$QM
        endif;

        TSTMT = 'ADD '//$A(SLVL)//'A'

        DO execute created statement
          &TSTMT &($C) &PURGE
        enddo;
      endif;
      SLVL = SLVL + 2
    enddo;
  endif;

  SLVL = 11

  IF $D EQ BLANK
  THEN  - (build and execute add statement for all
           SYSMAN2+S)
    DO UNTIL SLVL EQ 43
      IF $A(SLVL) NE BLANK
      THEN  - (SYSMAN2+S is file for add statement)
        IF $A(SLVL+1) NE BLANK
        THEN  - PURGE = ' SPRGDTE '//$QM//$A(SLVL+1)//$QM
        ELSE  - PURGE = ' SPRGDTE '//$QM//TPRGDTE//$QM
        endif;

        TSTMT = 'ADD '//$A(SLVL)//'S'

        DO execute created statement
          &TSTMT &($D) &PURGE
        enddo;
      endif;

      SLVL = SLVL + 2
    enddo;
  endif;
endproc ADDCASE

DEC DELETECASE
  BLANK        value ' '
  $C        buffer area containing active SSNORs
  $D        buffer area containing separated SSNORs
  $A(11)       1st SYSMAN2
  $A(12)       Purge date for 1st SYSMAN2
  $A(13)       2nd SYSMAN2
  $A(14)       Purge date for 2nd SYSMAN2
  $A(15)       3rd SYSMAN2
  $A(16)       Purge date for 3rd SYSMAN2
  $A(17)       4th SYSMAN2
  $A(18)       Purge date for 4th SYSMAN2
  $A(19)       5th SYSMAN2
  $A(20)       Purge date for 5th SYSMAN2
  $A(21)       6th SYSMAN2
  $A(22)       Purge date for 6th SYSMAN2
  $A(23)       7th SYSMAN2
  $A(24)       Purge date for 7th SYSMAN2
  $A(25)       8nd SYSMAN2
  $A(26)       Purge date for 8th SYSMAN2
  $A(27)       9rd SYSMAN2
  $A(28)       Purge date for 9th SYSMAN2
  $A(29)       10th SYSMAN2
  $A(30)       Purge date for 10th SYSMAN2
  $A(31)       11th SYSMAN2
  $A(32)       Purge date for 11th SYSMAN2
  $A(33)       12th SYSMAN2
  $A(34)       Purge date for 12th SYSMAN2
  $A(35)       13th SYSMAN2
  $A(36)       Purge date for 13th SYSMAN2
  $A(37)       14th SYSMAN2
  $A(38)       Purge date for 14th SYSMAN2
  $A(39)       15th SYSMAN2
  $A(40)       Purge date for 15th SYSMAN2
  $A(41)       16th SYSMAN2
  $A(42)       Purge date for 16th SYSMAN2
  SLVL         value 11 (SYSMAN2 array level)
  TSTMT        buffer containing verb and file name
enddec DELETECASE
```

```
PROC DELETECASE

  IF $C NE BLANK
  THEN  (build and execute delete statement for each
         SYSMAN2+A)
    DO UNTIL SLVL EQ 43
      IF $A(SLVL) NE BLANK
      THEN (SYSMAN2+A is file for delete statement)
        TSTMT = 'DELETE '//$A(SLVL)//'A'
        DO execute created statement to remove SSNORs
           from file
          &TSTMT &($C)
        enddo;
      endif;

      SLVL = SLVL + 2
    enddo;
  endif;
  SLVL = 11
  IF $D NE BLANK
  THEN  (build and execute delete statement for each
         SYSMAN2+S)
    DO UNTIL SLVL EQ 43
      IF $A(SLVL) NE BLANK
      THEN (SYSMAN2+S is file for delete statement)
        TSTMT = 'DELETE '//$A(SLVL)//'S'
        DO execute created statement to remove SSNORs
           from file
          &TSTMT &($D)
        enddo;
      endif;

      SLVL = SLVL + 2
    enddo;
  endif;
endproc DELETECASE
 PROC EXIT-FOR-RESTART

   IF $A(2) EQ BLANK
   THEN  - RELOCATE cursor to Action (*RL2)
   endif;

   IF $A(3) EQ BLANK and $A(4) NE BLANK
   THEN  - RELOCATE cursor to $A(3) (*RL3)
   endif;

   IF $A(4) EQ BLANK and $A(3) NE BLANK
   THEN  - RELOCATE cursor to $A(4) (*RL4)
   endif;

   IF $A(5) EQ BLANK and $A(6) NE BLANK
   THEN  - RELOCATE cursor to $A(5) (*RL5)
   endif;
   IF $A(6) EQ BLANK and $A(5) NE BLANK
   THEN  - RELOCATE cursor to $A(6) (*RL6)
   endif;

   IF $A(7) EQ BLANK and $A(8) NE BLANK
   THEN  - RELOCATE cursor to $A(7) (*RL7)
   endif;

   IF $A(8) EQ BLANK and $A(7) NE BLANK
   THEN  - RELOCATE cursor to $A(8) (*RL8)
   endif;

   IF $A(9) EQ BLANK and $A(10) NE BLANK
   THEN  - RELOCATE cursor to $A(9) (*RL9)
   endif;

   IF $A(10) EQ BLANK and $A(9) NE BLANK
   THEN  - RELOCATE cursor to $A(10) (*RL10)
   endif;

   IF SWCH EQ 'OFF'
   THEN  - RELOCATE cursor to first SYSMAN2 (*RL11)
   endif;
endproc EXIT-FOR-RESTART


PROC EXIT-FOR-MORE
   VOPRINT '*C'                clears menu
   VOPRINT '*I1,E PRCATRSF'    establishes value in $A(1)
   VOPRINT '*+B'               big buffer
   VOPRINT '*RL2'              relocate cursor to $A(2)
endproc EXIT-FOR-MORE


PROC EXIT-FOR-NOMORE
   VOPRINT  '*-F'              exit format mode
   VOPRINT  '*C'               clear entire screen
endproc EXIT-FOR-NOMORE
endproc PRCATRSF
```

4.3.6 PRNTEPRG - NOT-TO-EXCEED PURGE

4.3.6.1 PURPOSE

When an employee's reassignment is anticipated or planned, the receiving organization can request that employee's record from the owning organization. The owning organization will pass this data via the PRCATRSF procedure (Component Access Transfer). A Purge Date must also be passed with the data. If the employee actually transfers, an INTERFACE transaction will officially move the individual from one organization to another. However, if the employee does not move, the PRNTEPRG procedure will be used to purge the record from the Security Matrix (SEGACCESS) file of the receiving organization when the Purge Date is reached. See flow diagram in Figure 9 of the Appendix.

4.3.6.2 GENERAL INFORMATION

Proc Name	: PRNTEPRG
Language Used	: GIM POL
Initiated By	: Monthly process by DBCC
Frequency of Execution	: Monthly
Input	: System Date
Output	: List of SYSMAN2, SSNOR, PRGDTE, purged data, (PRNTEPRG Report)

4.3.6.3 FUNCTIONAL DESCRIPTION

The PRNTEPRG procedure will be used to purge records from the SEGACCESS files. The various components can purge records using the PRCATRSF (limited use) or PRACCUPDT procedures. However, if this is not done the PRNTEPRG procedure will purge all records with a Purge Date LE to the System Date. When a SEGACCESS segment has a Purge Date LE to the System Date a purge statement will be produced and executed for that segment.

EX: For (SEGACCESS Active segment) with PRGDTE LE 'System Date' delete.

For (SEGACCESS Separated segment) with PRGDTE LE 'System Date' with present PRIMSEPDTE delete.

4.3.6.4 INPUT

No menu will be required for this procedure. The procedure will be initiated by issuing the following command:

```
E PRNTEPRG
```

4.3.6.5 OUTPUT

This procedure will use the following statement to purge all SEGACCESS Segments of the SSNOR with a Purge Date LE System Date.

EX: For (SEGACCESS Active segment) with PRGDTE LE '(System Date)' delete.

For (SEGACCESS Separated segment) with PRGDTE LE '(System Date)' with present PRIMSEPDTE delete.

This procedure will also produce the PRNTEPRG Report which lists all data purged. The report will be produced for the PRIM Data Base Manager via REPORTW (See Report Section). xxx <u>PRNTEPRG</u>

4.3.6.6 GENERAL PROGRAM CONSIDERATIONS

This procedure will use the REPORTW verb against each SEGACCESS active and separated segment (for all Component Signon ORGs) to produce a 2-part report for each Component. The report will list all data to be purged from the Component's SEGACCESS file (SYSMAN2+A (active), SYSMAN2+S (separated)). After the report has been produced, a delete statement will be issued against the active and separated segments to purge the data.

The file name and the Text field of the segments will be acquired from the SELECTION file which also has a segment for each Component SYSMAN2.

4.3.6.7 DETAILED PROGRAM SPECIFICATIONS

DEC PRNTEPRG

$A	single dimensional array to hold SELECTION IDs
$AM	count of filled $A arrays
ALVL	level of $A array
SDATE	value current date YYMMDD
$B	single dimensional array to hold SELECTION TEXT
$BM	count of filled $B arrays
IDENT	used to store 'type of segment' for report header
INSRZ1	used to hold organizational text for report header
RPTCNT	count of reports produced (2 per Signon ORG)
STMT1A	used to build main body of the format
STMT1B	statement before specifics pertaining to the
STMT1C	organization is added.
STMT1D	
STMT2	used to store finished format statement.

```
  STMT3      used to hold deletion statement.
enddec PRNTEPRG PROC PRNTEPRG

  GET all SELECTION DLIDs through master
   FOR SELECTALL ACQUIRE

  DO for all SELECTION records
    GET DLID      (use NEXT)
     $A = $W
    GET SLTEXT    (use GFAV)
     $B = $W
    IF $BM NE $AM
    THEN  - $B = $A(ALVL)//' NO TEXT AVAILABLE'
    endif;
  enddo;

  Build main body of format statements. Leave places for
  inserting organizational name, and report type.

  DO UNTIL ALVL EQ $AM
    RPTCNT = 1
    DO WHILE RPTCNT LE 2

      IDENT = Type of report being processed.
       First pass - Active segment
       Second Pass - Separated segment
      INSRZ1 = Filename (Value from $A concatinated with
        'A' in first pass, and with 'S' in second pass)
      Build format statement
        FORMAT P'60,131' H'$61,SECRET'
        'Report of data requiring purging
        from the //IDENT// for //$B(ALVL)'
        F '$61,SECRET'; FOR //INSRZ1//
        WITH SAPRGDTE LE //SDATE//
        REPORTW SSNOR = //INSRZ1// : 'L9,,A1'
        NAME = PERNANEOR : 'L20'
        SEGMENT-ACCESS-DATE = SADTE : 'L19'
        PURGE-DATE = SAPRGDTE : 'L10'

      FOR //INSRZ1// WITH SAPRGDTE LE '&SDATE'
      DELETE

      RPTCNT = RPTCNT + 1
    enddo;
    ALVL = ALVL + 1
  enddo;

endproc PRNTEPRG
```

4.3.7 PRINTPRG - INTERFACE PURGE

4.3.7.1 PURPOSE

This procedure will be used to purge interface data from the PRIM INTERFACE file. When INTERFACE data is received from HRS2, the PRCHGLOAD procedure is used to analyze the data, make necessary updates to the PRIM PERSIGN or PRIMSEP file, create an INTERINDX record with a purge date (INPURGE = RUNDATE + 30), and add a record to the PRIM INTERFACE file. This procedure will scan the PRIM INTERINDX file and delete the INTERINDX and INTERFACE records with a purge date (INPURGE) LE the System Date. See flow diagram in Figure 10 of the Appendix.

4.3.7.2 GENERAL INFORMATION

Proc Name	: PRINTPRG
Language Used	: GIM POL
Frequency Of Execution	: Monthly
Initiated By	: Monthly process by DBCC

Input	: System Date
	INTERINDX
	  INTERLINK (D1)
	   INPURGE (D2)

Output	: All items associated with the purge date (INPURGE) will be removed from the INTERINDX and PRIM INTERFACE files.

4.3.7.3 FUNCTIONAL REQUIREMENTS

When data is added to the PRIM INTERFACE file, a link record will also be created in the INTERINDX file. The INTERINDX file will use the SSNOR as the DL/ID and any INTERFACE actions associated with that SSNOR will be entered by INTERFACE DL/ID (RUNDATE*TIME*STMT) into the INTERLINK field. The Purge Date (INPURGE) will be entered as D2s under INTERLINK.

4.3.7.4 INPUT

No menu will be required for this procedure. The following statement will initiate the procedure: E PRINTPRG

4.3.7.5 OUTPUT

The PRINTPRG procedure will purge the PRIM INTERFACE file data with the following statement:

```
   FOR INTERFACE WITH INPURGE NE 'system date' DELETE INTERFACE#
```

The INTERINDX file data will be analyzed and purged using the following decision formula:

```
 If INTERINDX SSNOR has only one D1 and INPURGE LE System Date
     or has many D1s all with INPURGE LE System Date
 then  - delete INTERINDX record
 endif;

 If INTERINDX SSNOR has more than one D1 and some INPURGE LE System
Date
 then  - delete INTERLINK when INPURGE LE System Date
 endif;
```

4.3.7.6 GENERAL PROGRAM CONSIDERATIONS

The PRINTPRG procedure will delete INTERFACE records that are 30 days old. A span from INTERFACE to the INTERINDX file will be used to delete the INTERFACE record when the INTERINDX INPURGE date has been reached or passed. After the INTERFACE records are deleted, the associated INTERINDX record will also be deleted.

4.3.7.7 DETAILED PROGRAM SPECIFICATIONS

```
DEC PRINTPRG
  BLANK     value ' '
  DATE      value current date YYMMDD
  GDATE     value current date - GIM format NNNN
                 (N denoting a numeric character)
  INTERINDX
    INPURGE used via Span from INTERFACE
  DCNT      value 0 (count of items to delete)
  DTECNT    value 0 (count of dates checked)
  ID        buffer for current INTERINDX DLID
  INDTE     INPURGE date being checked
  LNK       INTERLINK of INPURGE being checked
  $A        buffer area to contain all delete
            statements to be executed
  ALVL      level of $A array
  $AM       count of used $A arrays
  DLNK      INTERLINKs to delete
enddec PRINTPRG

PROC PRINTPRG
```

```
  SCAN INTERINDX file for INPURGE LE DATE

    FOR INTERINDX WITH INPURGE LE '&DATE'
    ACQUIRE

    If no data found
    THEN print message (no data found for purge)
         EXIT PRINTPRG

  DO for all records found in SCAN
    GET record DLID
      ID = $QM//$W//$QM
    DO for all INTERLINK values for DLID
      GET INTERLINK    (use GFAV or GNAV)
        LNK = $W
      GET INPURGE      (use GCAV)
        INDTE = $W
      IF INDTE LE DATE
      THEN
        DO (INTERLINK and INTERFACE should be deleted)
          DLNK = DLNK//$QM//LNK//$QM
          DCNT = DCNT + 1
        enddo;
      endif;
      DTECNT = DTECNT + 1
    enddo;

    IF DTECNT EQ DCNT
    THEN      (entire INTERINDX record can be deleted)
      $A = 'DELETE INTERINDX '//ID
    ELSE   (only INTERLINK in DLNK can be deleted)
      $A = 'DELETE INTERINDX '//ID//' INTERLINK '//DLNK
    endif;

    IF DLNK NE ' '
    THEN   (INTERFACE record may be deleted)
      $A = 'DELETE INTERFACE '//DLNK
    ELSE;
    endif;

    ID = BLANK
    LNK = BLANK
    INDTE = BLANK
    DCNT = 0
    DTECNT = 0
  enddo;

  DO UNTIL ALVL EQ $A  (execute statement in $A array)
    DO &($A(ALVL))
    enddo;
    ALVL = ALVL + 1
  enddo;
endproc PRINTPRG
```

4.3.8 <u>PRACCUPDT - ACCESS UPDATE</u>

4.3.8.1 PURPOSE

This procedure will be used by the various components to make changes to their SEGACCESS segments. The components will be able to change PRGDTE, add SSNORs needed to link to their component file (Release 2), and delete SSNORs which were passed from another component but are no longer needed. See flow diagram in Figure 10 for the Appendix.

4.3.8.2 GENERAL INFORMATION

Proc Name	: PRACCUPDT
Language Used	: GIM POL
Frequency Of Execution	: Random per component need
Initiated by	: Individual Components
Input	: SSNOR STATUS Purge Date OWNCODE (Release 2)
Output	: Deletes or adds PRGDTE and/or SSNOR to file

4.3.8.3 FUNCTIONAL REQUIREMENTS

This procedure will be used by a component to update it's personal SEGACCESS files. Because the SEGACCESS file ID is the same as the Signon SYSMAN2 ID, this procedure will use the Signon SYSMAN2 to determine the SEGACCESS file to be updated (SYSMAN2+A for Active, SYSMAN2+S for Separated). If a component has received an employee's record from another component and wants to change the Purge Date (PRGDTE) or delete the record entirely, this procedure will be used. If the SSNOR had not been previously entered into the SEGACCESS file via the PRCALINK or the PRCATRSF procedure, the component can use PRACCUPDT to enter the SSNOR in the file (SYSMAN2+C) used to link to their component file. This will allow a component to enter an SSNOR on an employee or applicant in their component file but will not allow that component access to the Official Data for that SSNOR.

4.3.8.4 INPUT

This procedure will utilize the following menu which will be stored in the MENU-FORMATS file.

E PRACCUPDT (Access Update)

```
Action: _

(A) Add (C) Change Purge Date (D) Delete (R) Retrieve (X) Exit

    SSNOR        STATUS     PURGE DATE      OWNERSHIP
                 (A/S/C)     (YYMMDD)

  _________        _         _______           _

  _________        _         _______           _

  _________        _         _______           _

  _________        _         _______           _

  _________        _         _______           _

  _________        _         _______           _
```

<u>MENU ITEM DESCRIPTION</u>

(R)	ACTION	Action to be done (A,C,D,R,X)	(LA01)
(R)	SSNOR	Social Security Number	(LA09)
(R/O)	STATUS	Active,Separated,Component	(LA01)
(O)	Purge Date	Removal from System (YYMMDD)	(LN06)
(O)	Ownership	Ownership of Record	(LA01)

4.3.8.5 OUTPUT

```
If Action = 'X'
then  - exit the procedure.
endif;

If Action = 'R' and present SSNOR(s)
then  - leave data on menu
      - retrieve data
endif;


If Action = 'A', 'C'  or 'D' and null SSNOR or null STATUS
then  - error message (SSNOR and STATUS must be entered)
else  - SEGACCESS = SYSMAN2+A for Active record
        SEGACCESS = SYSMAN2+S for Separated record
        SEGACCESS = SYSMAN2+C for Component record
endif;
```

```
If Action = 'A' and present SSNOR and STATUS = 'C'
then  - CSDTE = Current Date (generated by procedure)
      - build add statement to add SSNOR(s) to (SYSMAN2+C)
        with SDTE, PRGDTE, and OWNCODE
endif;

If Action = 'C' and present SSNOR, STATUS and present Purge Date
then  - build statement to change Purge Date in SEGACCESS file
        SYSMAN2+A PRGDTE (Active) for SSNOR(s) with STATUS = 'A'
      - build statement to change Purge Date in SEGACCESS file
        SYSMAN2+S PRGDTE (Separated) for SSNOR(s) with
        STATUS = 'S'
      - build statement to change Purge Date in SEGACCESS file
        SYSMAN2+C PRGDTE (Component) for SSNOR(s) with
        STATUS = 'C'. If Ownership field present on menu with
        STATUS = 'C' add ownership value to OWNCODE.
endif;

If Action = 'D' and present SSNOR and present STATUS
then  - build statement to delete SSNOR(s) with STATUS = 'A' from
        (SYSMAN2+A) when SDTE NE current date ($ODATEG)
      - build statement to delete SSNOR(s) with STATUS = 'S' from
        (SYSMAN2+S) when SDTE NE current date ($ODATEG)
      - build statement to delete SSNOR(s) with STATUS = 'C' from
        (SYSMAN2+C)
endif;
```

4.3.8.6 GENERAL PROGRAM CONSIDERATIONS

The PRACCUPDT procedure will be used by the Component to extend the purge date of SSNORs passed from another Component, extend the purge date of an employee's SSNOR who has left that Component, remove the SSNOR entirely or add the SSNOR to the Component link segment. Each record must be verified before it can be deleted.

In Release 2, Component files will be added to the PRIM data base. Before an SSNOR can be added to a Component's file, it must be present in one of the Component's SEGACCESS official data link files (SYSMAN2+A or SYSMAN2+S). If the SSNOR is not present in either of those files, it must be added to the Component data link file (SYSMAN2+C) via the PRACCUPDT procedure. Adds can only be done against Component files. SSNOR cannot begin with a '6', '8', '90', '91', '92', '95', '96', or '97'.

4.3.8.7 DETAILED PROGRAM SPECIFICATIONS

```
DEC PRACCUPDT
   BLANK      value ' '
   $A         single dimensional arrays
    $A(2)     Action value
    $A(3)      1st SSNOR
    $A(4)      1st SSNOR Status
    $A(5)      1st SSNOR Purge Date (SAPRGDTE)
    $A(6)      1st SSNOR Ownership Code (SAOWNCODE)
    $A(7)      2nd SSNOR
    $A(8)      2nd SSNOR Status
    $A(9)      2nd SSNOR Purge Date (SAPRGDTE)
    $A(10)     2nd SSNOR Ownership Code (SAOWNCODE)
    $A(11)     3rd SSNOR
    $A(12)     3rd SSNOR Status
    $A(13)     3rd SSNOR Purge Date (SAPRGDTE)
    $A(14)     3rd SSNOR Ownership Code (SAOWNCODE)
    $A(15)     4th SSNOR
    $A(16)     4th SSNOR Status
    $A(17)     4th SSNOR Purge Date (SAPRGDTE)
    $A(18)     4th SSNOR Ownership Code (SAOWNCODE)
    $A(19)     5th SSNOR
    $A(20)     5th SSNOR Status
    $A(21)     5th SSNOR Purge Date (SAPRGDTE)
    $A(22)     5th SSNOR Ownership Code (SAOWNCODE)
    $A(23)     6th SSNOR
    $A(24)     6th SSNOR Status
    $A(25)     6th SSNOR Purge Date (SAPRGDTE)
    $A(26)     6th SSNOR Ownership Code (SAOWNCODE)
   STATSW      value established in ADDCASE
   LOCATE      value established in CKCASE
enddec PRACCUPDT

PROC PRACCUPDT (User Update of Access Date)

  READ menu values into $A single demensional arrays

  IF $A(2) EQ 'X' (Exit the Menu)
  THEN  - RUN EXIT-FOR-NOMORE
  endif;

  IF $A(2) EQ BLANK or ($A(2) NE 'A' and $A(2) NE 'C' and
     $A(2) NE 'D' and $A(2) NE 'R')
  THEN  - PRINT message stating valid Action required
        - RUN EXIT-FOR-RESTART
  ELSE
    IF $A(3) EQ BLANK and $A(7) EQ BLANK
       and $A(11) EQ BLANK and $A(15) EQ BLANK
       and $A(19) EQ BLANK and $A(23) EQ BLANK)
    THEN  - PRINT message stating SSNOR ia a required field
          - RUN EXIT-FOR-RESTART
    endif;
  endif;

  IF $A(2) EQ 'R' (Retrieve current Access Links)
  THEN  - RUN CKCASE
    IF LOCATE NE 0
    THEN - RUN EXIT-FOR-RESTART
    endif;
    - RUN RCASE
    - RUN EXIT-FOR-MORE
  endif;

  IF $A(2) EQ 'A' and ($A(4) EQ 'C' or $A(8) EQ 'C' or
     $A(12) EQ 'C' or $A(16) EQ 'C' or $A(20) EQ 'C' or
     $A(24) EQ 'C')
  THEN  - RUN CKCASE
    IF LOCATE NE 0
    THEN RUN EXIT-FOR-RESTART
    ELSE - RUN ADDCASE
      IF STATSW EQ 'ON'
      THEN - RUN EXIT-FOR-RESTART
      ELSE - RUN EXIT-FOR-MORE
      endif;
    endif;
  endif;

  IF $A(2) EQ 'C' (Change Purge Date)
  THEN  - RUN CKCASE
    IF LOCATE NE 0
    THEN - RUN EXIT-FOR-RESTART
    endif;
    - RUN CCASE
    - RUN EXCASE
    - RUN EXIT-FOR-MORE
  endif;

  IF $A(2) EQ 'D' (Delete SSNOR if SADTE not Current)
  THEN  - RUN CKCASE
    IF LOCATE NE 0
    THEN - RUN EXIT-FOR-RESTART
    endif;
        - RUN DCASE
        - RUN EXCASE
        - RUN EXIT-FOR-MORE
  endif;

DEC ADDCASE
  STATSW     value 'OFF'
             value 'ON' if STATUS NE 'C'
  STMT       buffer area for add statement
  STMT1      buffer area for 1st part of statement
  SEGMENT    $UNAME1
  TDATE      buffer area for date
  ALVL       value 3
  $A(3)      1st SSNOR
  $A(4)      1st SSNOR Status
  $A(5)      1st SSNOR Purge Date (SAPRGDTE)
  $A(6)      1st SSNOR Ownership Code (SAOWNCODE)
  $A(7)      2nd SSNOR
  $A(8)      2nd SSNOR Status
  $A(9)      2nd SSNOR Purge Date (SAPRGDTE)
  $A(10)     2nd SSNOR Ownership Code (SAOWNCODE)
  $A(11)     3rd SSNOR
  $A(12)     3rd SSNOR Status
  $A(13)     3rd SSNOR Purge Date (SAPRGDTE)
  $A(14)     3rd SSNOR Ownership Code (SAOWNCODE)
  $A(15)     4th SSNOR
  $A(16)     4th SSNOR Status
  $A(17)     4th SSNOR Purge Date (SAPRGDTE)
  $A(18)     4th SSNOR Ownership Code (SAOWNCODE)
  $A(19)     5th SSNOR
  $A(20)     5th SSNOR Status
  $A(21)     5th SSNOR Purge Date (SAPRGDTE)
  $A(22)     5th SSNOR Ownership Code (SAOWNCODE)
  $A(23)     6th SSNOR
  $A(24)     6th SSNOR Status
  $A(25)     6th SSNOR Purge Date (SAPRGDTE)
  $A(26)     6th SSNOR Ownership Code (SAOWNCODE)
enddec ADDCASE

PROC ADDCASE

  STMT1 = 'ADD '//SEGMENT//'C' (1st part of add statement)

  DO for all SSNORs on menu
    IF $A(ALVL+1) EQ 'C'
    THEN - TDATE = $QM//$ODATEA($DATEG)//$QM
         - STMT=STMT//$QM//$A(ALVL)//$QM//' SADTE '//TDATE
      IF $A(ALVL+2) NE BLANK
      THEN  - TDATE = $QM//$A(ALVL+2)//$QM
            - STMT=STMT//' SAPRGDTE '//TDATE
      ELSE  - TDATE = $QM//$ODATEA($DATEG + 30)//$QM
            - STMT=STMT//' SAPRGDTE '//TDATE
      endif;
      IF $A(ALVL+3) NE BLANK
      THEN STMT=STMT//' SAOWNCODE '//$QM//$A(ALVL+3)//$QM
      endif;
    ELSE  STATSW = 'ON'
    endif;

    IF &STMT NE BLANK
    THEN
      DO execute created statement
        &STMT1 &STMT
      enddo;
    endif;
    ALVL = ALVL + 4
  enddo;

  IF STATSW EQ 'ON'
  THEN PRINT message stating only Status 'C'
       items have been added
  endif;
endproc ADDCASE

DEC CCASE
```

```
   BLANK      value ' '
   SEGMENT    $UNAME1
   TSEGMENT   current segment value
   TDATE      buffer area for date
   $B         single dimensional array to
              hold change statements
   $BM        count of used $B arrays
   $BC        current $B array
   ALVL       value 3
   $A         single dimensional array
    $A(3)     1st SSNOR
    $A(4)     1st SSNOR Status
    $A(5)     1st SSNOR Purge Date (SAPRGDTE)
    $A(6)     1st SSNOR Ownership Code (SAOWNCODE)
    $A(7)     2nd SSNOR
    $A(8)     2nd SSNOR Status
    $A(9)     2nd SSNOR Purge Date (SAPRGDTE)
    $A(10)    2nd SSNOR Ownership Code (SAOWNCODE)
    $A(11)    3rd SSNOR
    $A(12)    3rd SSNOR Status
    $A(13)    3rd SSNOR Purge Date (SAPRGDTE)
    $A(14)    3rd SSNOR Ownership Code (SAOWNCODE)
    $A(15)    4th SSNOR
    $A(16)    4th SSNOR Status
    $A(17)    4th SSNOR Purge Date (SAPRGDTE)
    $A(18)    4th SSNOR Ownership Code (SAOWNCODE)
    $A(19)    5th SSNOR
    $A(20)    5th SSNOR Status
    $A(21)    5th SSNOR Purge Date (SAPRGDTE)
    $A(22)    5th SSNOR Ownership Code (SAOWNCODE)
    $A(23)    6th SSNOR
    $A(24)    6th SSNOR Status
    $A(25)    6th SSNOR Purge Date (SAPRGDTE)
    $A(26)    6th SSNOR Ownership Code (SAOWNCODE)
enddec CCASE

PROC CCASE

 DO UNTIL ALVL EQ 23
   IF $A(ALVL) NE BLANK AND $A(ALVL+1) EQ 'A'
   THEN - TSEGMENT = SEGMENT//'A'//$qm//$A(ALVL)
   //$qm
   endif;

   IF $A(ALVL) NE BLANK AND $A(ALVL+1) EQ 'S'
   THEN - TSEGMENT = SEGMENT//'S'//$qm//$A(ALVL)
   //$qm
   endif;

   IF $A(ALVL) NE BLANK AND $A(ALVL+1) EQ 'C'
   THEN - TSEGMENT = SEGMENT//'C'//$qm//$A(ALVL)
   //$qm
   endif;
```

```
    IF $A(ALVL+1) EQ 'A' OR $A(ALVL+1) EQ 'S'
    THEN - TDATE = $QM//$ODATEA($DATEG)//$QM
         - $B='FOR '//TSEGMENT//' WITH SADTE NE '//
           TDATE//' OR WITH NULL SADTE CHANGE
           SAPRGDTE TO '//$QM//$A(ALVL+2)//$QM
    endif;

    IF $A(ALVL+1) EQ 'C' AND $A(ALVL+2)
       NE BLANK
    THEN - $B = 'FOR '//TSEGMENT//' CHANGE
           SAPRGDTE TO '//$QM//$A(ALVL+2)//$QM
      IF $A(ALVL+3) NE BLANK
      THEN - $B($BC)=$B($BC)//' SAOWNCODE TO '//
        $QM//$A(ALVL+3)//$QM
      endif;
    endif;
    ALVL = ALVL + 4
  enddo;
endproc CCASE

DEC DCASE
  SEGMENT   $UNAME1
  TSEGMENT  current segment value
  $B        single dimensional array to
            hold change statements
  ALVL      value 3
  $A        single dimensional array
   $A(3)    1st SSNOR
   $A(4)    1st SSNOR Status
   $A(7)    2nd SSNOR
   $A(8)    2nd SSNOR Status
   $A(11)   3rd SSNOR
   $A(12)   3rd SSNOR Status
   $A(15)   4th SSNOR
   $A(16)   4th SSNOR Status
   $A(19)   5th SSNOR
   $A(20)   5th SSNOR Status
   $A(23)   6th SSNOR
   $A(24)   6th SSNOR Status
enddec DCASE

PROC DCASE

 DO UNTIL ALVL EQ 23
    IF $A(ALVL) NE BLANK AND $A(ALVL+1) EQ 'A'
    THEN - TSEGMENT = SEGMENT//'A'//$qm//$A(ALVL)
    //$qm
    endif;

    IF $A(ALVL) NE BLANK AND $A(ALVL+1) EQ 'S'
    THEN - TSEGMENT = SEGMENT//'S'//$qm//$A(ALVL)
    //$qm
    endif;
    IF $A(ALVL) NE BLANK AND $A(ALVL+1) EQ 'C'
    THEN - TSEGMENT = SEGMENT//'C'//$qm//$A(ALVL)
    //$qm
    endif;

    IF $A(ALVL+1) EQ 'A' OR $A(ALVL+1) EQ 'S'
       OR $A(ALVL+1) EQ 'C'
    THEN - $B = 'DELETE '//TSEGMENT//$QM//$A(ALVL)//$QM
    endif;
    ALVL = ALVL + 4
  enddo;
endproc DCASE

DEC CKCASE
  BLANK     value ' '
  ALVL      value 3
  LOCATE    value 0
  $A        single dimensional array
   $A(3)    1st SSNOR
   $A(4)    1st SSNOR Status
   $A(5)    1st SSNOR Purge Date (SAPRGDTE)
   $A(6)    1st SSNOR Ownership Code (SAOWNCODE)
   $A(7)    2nd SSNOR
   $A(8)    2nd SSNOR Status
   $A(9)    2nd SSNOR Purge Date (SAPRGDTE)
   $A(10)   2nd SSNOR Ownership Code (SAOWNCODE)
   $A(11)   3rd SSNOR
   $A(12)   3rd SSNOR Status
   $A(13)   3rd SSNOR Purge Date (SAPRGDTE)
   $A(14)   3rd SSNOR Ownership Code (SAOWNCODE)
   $A(15)   4th SSNOR
   $A(16)   4th SSNOR Status
   $A(17)   4th SSNOR Purge Date (SAPRGDTE)
   $A(18)   4th SSNOR Ownership Code (SAOWNCODE)
   $A(19)   5th SSNOR
   $A(20)   5th SSNOR Status
   $A(21)   5th SSNOR Purge Date (SAPRGDTE)
   $A(22)   5th SSNOR Ownership Code (SAOWNCODE)
   $A(23)   6th SSNOR
   $A(24)   6th SSNOR Status
   $A(25)   6th SSNOR Purge Date (SAPRGDTE)
   $A(26)   6th SSNOR Ownership Code (SAOWNCODE)
enddec CKCASE

PROC CKCASE

 DO UNTIL ALVL EQ 23 or LOCATE NE 0
    IF $A(ALVL) NE BLANK AND $A(ALVL+1) EQ BLANK
    THEN - LOCATE = ALVL+1
         - PRINT msg (status is required)
    endif;

    ALVL = ALVL + 4
 enddo;
```

```
endproc CKCASE

DEC EXCASE
  BLVL       value 1
  $B         single dimensional array
             containing created statements
  $BM        count of used $B arrays
enddec EXCASE

PROC EXCASE

 DO UNTIL BLVL EQ $BM
   &($B(BLVL))       (Execute all statement)
   BLVL = BLVL + 1
 enddo;
endproc EXCASE

DEC RCASE
  ALVL       value 3
  SEGMENT    $UNAME1
  TSEGMENT   current segment value
  HOLD       temporary hold area for SAPRGDTE
  $A         single dimensional array
enddec RCASE

PROC RCASE

  DO UNTIL ALVL EQ 23
    IF $A(ALVL) NE BLANK
    THEN - TSEGMENT = SEGMENT//$A(ALVL+1)//$QM
           //$A(ALVL)//$QM
         - FOR &TSEGMENT ACQUIRE SAPRGDTE SAOWNCODE
         - get SAPRGDTE         (use GFAV)
         - HOLD = $W
      VOPRINT '*I'//$ALF(ALVL+2)//','//HOLD
      IF $A(ALVL+1) EQ 'C'
      THEN  - get SAOWNCODE       (use GFAV)
            - VOPRINT '*I'//$ALF(ALVL+3)//','//$W
      endif;
    endif;
    ALVL = ALVL + 4
  enddo;
endproc RCASE

DEC EXIT-FOR-RESTART
  $A          single dimensional arrays
   $A(2)      Action value
   $A(3)      1st SSNOR
   $A(4)      1st SSNOR Status
   $A(5)      1st SSNOR Purge Date (SAPRGDTE)
   $A(6)      1st SSNOR Ownership Code (SAOWNCODE)
   $A(7)      2nd SSNOR
   $A(8)      2nd SSNOR Status
   $A(9)      2nd SSNOR Purge Date (SAPRGDTE)
   $A(10)    2nd SSNOR Ownership Code (SAOWNCODE)
   $A(11)    3rd SSNOR
   $A(12)    3rd SSNOR Status
   $A(13)    3rd SSNOR Purge Date (SAPRGDTE)
   $A(14)    3rd SSNOR Ownership Code (SAOWNCODE)
   $A(15)    4th SSNOR
   $A(16)    4th SSNOR Status
   $A(17)    4th SSNOR Purge Date (SAPRGDTE)
   $A(18)    4th SSNOR Ownership Code (SAOWNCODE)
   $A(19)    5th SSNOR
   $A(20)    5th SSNOR Status
   $A(21)    5th SSNOR Purge Date (SAPRGDTE)
   $A(22)    5th SSNOR Ownership Code (SAOWNCODE)
   $A(23)    6th SSNOR
   $A(24)    6th SSNOR Status
   $A(25)    6th SSNOR Purge Date (SAPRGDTE)
   $A(26)    6th SSNOR Ownership Code (SAOWNCODE)
  BLANK     value ' '
enddec EXIT-FOR-RESTART

PROC EXIT-FOR-RESTART

  IF $A(2) EQ BLANK or ($A(2) NE 'A' and $A(2) NE 'C' and
     $A(2) NE 'D' and $A(2) NE 'R' and $A(2) NE 'X')
  THEN  - PRINT MESSAGE VALID ACTION REQUIRED
       - RELOCATE cursor to Action (*RL2)
  ELSE
    IF $A(3) EQ BLANK AND $A(7) EQ BLANK AND
       $A(11) EQ BLANK AND $A(15) EQ BLANK AND
       $A(19) EQ BLANK AND $A(23) EQ BLANK
    THEN  - RELOCATE cursor to Access Segment (*RL3)
    endif;
    IF LOCATE NE 0
    THEN - PRINT MESSAGE STATING SSNOR AND STATUS
           ARE REQUIRED FIELDS AND PURGE DATE IS A
           REQUIRED FIELD FOR 'C'
         - RELOCATE CURSOR TO $A(LOCATE)
             '*RL'//$A(LOCATE)
         - LOCATE = 0
    endif;
  endif;
endproc EXIT-FOR-RESTART

PROC EXIT-FOR-MORE
  VOPRINT '*C'                  clears menu
  VOPRINT '*I1,E PRACCUPDT'     establishes value in $A(1)
  VOPRINT '*+B'                 big buffer
  VOPRINT '*RL2'                relocate cursor to $A(2)
endproc EXIT-FOR-MORE

PROC EXIT-FOR-NOMORE
  VOPRINT  '*-F'                exit format mode
  VOPRINT  '*C'                 clear entire screen
endproc EXIT-FOR-NOMORE
```

endproc PRACCUPDT

4.3.9 PRCALINK - PRIM LINKS CREATED DAILY BY COMPONENT

4.3.9.1 PURPOSE

This procedure establishes the links for the initiating Signon Org to the Official Data files, ORGCODE file, POSNR file, and the Separated Data file (PRIMSEP). The procedure is activated at the time of Signon but will only perform the link-building function when the first Signon is done for an ORG. All SSNORs to be made available to the ORG initiating the Signon will be added to the ORGCODE and POSNR files if the user requests it. These links are stored in the ORG's SELECTION segment file where the Selection Criteria also is stored. The ORGCODE and POSNR links will be created on the weekend but the user can request a new link creation at Signon time with this procedure. See flow diagram in Figure 11 of the Appendix.

4.3.9.2 GENERAL INFORMATION

```
Proc Name     : PRCALINK
Language Used: GIM POL
Frequency Of
Execution     : Daily, the first time a Signon
                is initiated against an ORG.
Input         : Signon ORG's SELECTION file
                    Selection Criteria
                        SLOLNKDTE
                        SLLNKDTE
                      PERSIGN (XBRIDGE will trigger appropriate
                      index)
                        PERSD
                        PERAORG
                        PEROCCE
                        PERSCCE
                      PRIMSEP (XBRIDGE will trigger appropriate
                      index)
                        PERSD
                        PERAORG
                        PEROCCE
                        PERSCCE
                      INDEX files (For active data)
                        SDINDX      SDSSN
                        ORGINDX     ORGSSN
                        OCCEINDX    OCCESSN
                        SCCEINDX    SCCESSN
                       INDEX files (For separated data)
```

	SSDINDX	SSDSSN
	SORGINDX	SORGSSN
	SOCCEINDX	SOCCESSN
	SSCCEINDX	SSCCESSN
Output	: Current links created for the initiating ORG to Official, Separated and INTERFACE data	

4.3.9.3 FUNCTIONAL DESCRIPTION

This procedure is triggered at Signon time by the EXECPROC element in SYSMAN2. If the initiator is the first to sign on to that ORG that day, the procedure will acquire the ORGs Selection Criteria from the correct SELECT segment and with the use of INDEX files will select the SSNORs which can be accessed by that ORG (Both Active and Separated files). This procedure will also give the initiator the option to establish the most recent updated version of SLORGLINKs and SLPOSLINKs.

4.3.9.4 INPUT

No menu will be used for this procedure. The procedure will be initiated automatically at Signon time. If the link has not been built by a previous Signon (SLOLNKDTE IN ORG'S Selection file NE current date), the procedure will acquire Selection Criteria data from the Signon ORG's SELECTION segment and build a copy statement against the PRIM PERSIGN and PRIMSEP files to copy the appropriate SSNORs to the ORG's SEGACCESS files for active data and separated data links. The copy statement will trigger INDEX files through XBRIDGES which will eliminate an end-to-end search of PRIM PERSIGN or PRIMSEP. Upon request via prompts, the procedure will build an access link at first Signon to ORGCODE and POSNR files if the user requests it. This link data will be stored in the ORG's SELECTION segment and will be accessed through that file.

Procedure Prompt: Do You Want Your ORG/POS Link Updated?
User Response :

YES	- create links for ORGCODE and POSNR file
	- create links for Active data and Separated data
NO	- create links for Active data and Separated data

4.3.9.5 OUTPUT

The output from the PRCALINK will be the links to the PRIM PERSIGN and PRIMSEP files. The initiator will be asked if updated links are requested for ORGCODE and POSNR data.

If return from VOPREADI EQ 'NO'

```
then  - Acquire SLOLNKDTE (Official Link date) in SELECTION
        file (Signon ORG)
endif;

If SLOLNKDTE LT System Date (Link not established)
then  - acquire Selection Criteria for Signon ORG from
        SELECTION segment.
      - Build statement against PRIM PERSIGN file using
        Selection Criteria (triggers INDEX files that
        apply to Selection Criteria for Active data).
      - use copy verb to move SSNORs into SEGACCESS
        segment for Active data link (SYSMAN2+A)
        with SDTE and PRGDTE (SDTE + 60).
      - build statement against PRIMSEP file using
        Selection Criteria (triggers INDEX files that
        apply to Selection Criteria for Separated
        data).
      - use COPY verb to move SSNORs into SEGACCESS
        segment for Separated data link (SYSMAN2+S)
        with a PRGDTE (current date + 60).
      - add current date to SLOLNKDTE in SELECTION file
        for Signon ORG.
endif;

If return from VOPREADI EQ 'YES'
then  - acquire SLLNKDTE (POS/ORG link date) in SELECTION file
  If SLLNKDTE NE current date
  then  - acquire ORGCODE from Criteria line in the SELECTION file.
        - build statement against ORGCODE file using ORGCODE from
          Criteria lines.
        - add ORGCODE to the SELECTION file (SLORGLINK).
        - acquire SD from Criteria lines in the SELECTION file
        - build statement against POSNR file using ORGCODE from
          Criteria line and SD from Criteria line (with $scan POSCSD
          EQ 'SD value' or with $scan POSORG EQ 'ORGCODE value')
        - add POSNR to the SELECTION file (SLPOSLINK)
        - add SLLNKDTE "Current Date"
  endif;
endif;
```

4.3.9.6 GENERAL PROGRAM CONSIDERATIONS

This program establishes daily links for the Signon ORGs. This program is initiated by any Signon ORG at signon time if the program name has been entered in Sysman2 EXECPROC for the Signon ORG. Only one signon per day per ORG will generate the daily links. The program will check the SLOLNKDTE in the SELECTION file to verify whether the daily link has been done.

The PRCALINK procedure will also give the Signon ORG the option of creating new links to the POSNR and ORGLINK files. If the initiator requests new links and the SLLNKDTE is not equal to the current date then new links will be created. Otherwise, new POSNR and ORGCODE links are only created on weekends.

4.3.9.7 DETAILED PROGRAM SPECIFICATIONS

```
DEC PRCALINK
  ODATE      value 0, Official link date
  LDATE      value 0, POSNR/ORGCODE link date
  RESPONSE   value 'NO'
             value returned from VOPREADI
  SEGMENT    value $UNAME1
  STMT1      holds message to be displayed in
             the VOPREADI statement
  TDATE      $DATEG (GIM date format)
  $A         single dimensional array
             used to hold criteria data
enddec PRCALINK


PROC PRCALINK

  DO acquire data from SELECTION segment
    FOR &SEGMENT '&SEGMENT' ACQUIRE
  GET SLOLNKDTE          (use GFAV)
  ODATE = $W
  GET SLLNKDTE           (use GFAV)
  LDATE = $W

    DO for all SLCRITERIA values
      GET SLCRITERIA           (use GFAV & GNAV)
      $A = $W
    enddo;
  enddo;

  IF ODATE NE TDATE
  THEN - RUN OFFLNK
  endif;

  IF LDATE NE TDATE
    THEN
      VOPREADI 'Should new POSNR/ORGCODE links be
               Created (Yes or No)'//RESPONSE

      RESPONSE = response from VOPREADI

      IF RESPONSE EQ 'YES'
        THEN - RUN OTHLNK
      endif;
  endif;
  EXIT PRCALINK DEC OFFLNK
  ALVL       value 1
  BLANK      value ' '
  $A         single dimension array used to
             hold criteria data
  $AM        count of filled $A arrays
  SD         SD value from SLCRITERIA
  OCCE       OCCE value from SLCRITERIA
  SCHGR      SCH and GRADE values from SLCRITERIA
  SCCE       SCCE value from SLCRITERIA
  ORGCODE    ORGCODE value from SLCRITERIA
  SEGMENT    $UNAME1
  TSEGMENT   current segment being updated
  TDATE      $DATEG
  PDATE      $DATEG + 60
  STMT       area for 'WITH' portion of statement
  STMTA      1st part of Active statement(PERSIGN)
  STMTS      1st part of Separated statement (PRIMSEP)
enddec OFFLNK

PROC OFFLNK

  DO UNTIL ALVL EQ $AM
    IF $SBF($A(ALVL),0,1,'*') NE BLANK
    THEN SD=$QM//$SBF($A(ALVL),0,1,'*')//$QM
    endif;

    IF $SBF($A(ALVL),1,1,'*') NE BLANK
    THEN OCCE=$QM//$SBF($A(ALVL),1,1,'*')
    endif;
    IF $SBF($A(ALVL),2,1,'*') NE BLANK
    THEN SCHGR=$QM//$SBF($A(ALVL),2,1,'*')
      IF $SBF($A(ALVL),3,1,'*') NE BLANK
      THEN SCHGR=SCHGR//$SBF($A(ALVL),3,1,'*')//$QM
      endif;

      IF $SBF($A(ALVL),4,1,'*') NE BLANK
      THEN SCHGR=' GE '//SCHGR//' AND WITH SCHGR LE '
           //$QM//$SBF($A(ALVL),2,1,'*')//
           $SBF($A(ALVL),4,1,'*')//$QM
      endif;
    endif;

    IF $SBF($A(ALVL),5,1,'*') NE BLANK
    THEN SCCE=$QM//$SBF($A(ALVL),5,1,'*')//$QM
    endif;

    IF $SBF($A(ALVL),6,1,'*') NE BLANK
    THEN ORGCODE=$QM//$SBF($A(ALVL),6,1,'*')//$QM
    endif;
```

```
(BUILD COPY STATEMENT)

IF SD NE BLANK
THEN STMT=STMT//' WITH $SCANX(PERSD,'//SD//')'
endif;

IF OCCE NE BLANK
  IF STMT NE BLANK
  THEN STMT=STMT//' AND'
  endif;
THEN STMT=STMT//' WITH $SCANX(PEROCCE,'//OCCE//')'
endif;

IF SCHGR NE BLANK
  IF STMT NE BLANK
  THEN STMT=STMT//' AND'
  endif;
THEN STMT=STMT//' WITH PERSCHGR '//SCHGR
endif;

IF SCCE NE BLANK
  IF STMT NE BLANK
  THEN STMT=STMT//' AND'
  endif;
THEN STMT=STMT//' WITH PERSCCE '//SCCE
endif;

IF ORGCODE NE BLANK
  IF STMT NE BLANK
  THEN STMT=STMT//' AND'
  endif;
THEN STMT=STMT//' WITH $SCANX(PERORGCODE,'//ORGCODE//')'
endif;
```

```
  (EXECUTE BUILD STATEMENT AND DATE UPDATE STATEMENT)

  STMTA='FOR PERSIGN '
  STMTS='FOR PRIMSEP '

  TSEGMENT = SEGMENT//'A'

  &STMTA &STMT COPY REPLACE TO &TSEGMENT, PERTD TO SADTE,
  PERTD TO SAPRGDTE

  FOR &TSEGMENT WITH NULL SADTE AND WITH NULL SAPRGDTE
  ADD SADTE '&TDATE' SAPRGDTE '&PDATE'

  TSEGMENT = SEGMENT//'S'

  &STMTS &STMT COPY REPLACE TO &TSEGMENT, PERTD TO SADTE,
  PERTD TO SAPRGDTE

  FOR &TSEGMENT WITH NULL SADTE AND WITH NULL SAPRGDTE
  ADD SADTE '&TDATE' SAPRGDTE '&PDATE'
  enddo;

  FOR &SEGMENT '&SEGMENT' ADD SLOLNKDTE '&TDATE'

endproc OFFLNK DEC OTHLNK
  ALVL       current $A array being addressed
  BLANK      value ' '
  LDATE      value 0, POSNR/ORGCODE link date
  ORGCODE    ORGCODE value from SLCRITERIA
  SEGMENT    $UNAME1
  STMT       use to build 'WITH' statement for ORGCODE
  STMT1      use to build 'WITH' statement for POSNR
  STMT2      use to build 'ADD' statements to add
             dates to the office file
  STMTO      contains first part of ORGCODE statement
  STMTP      contains first part of POSNR statement
  TDATE      $DATEG (GIM date format)
  $A         single dimensional array
             containing all SLCRITERIA
  $AM        count of filled $A arrays
  $B         single dimensional array
             contains ORGCODE DLID for link
  BLVL       count of $B elements stored in current
             $C element
  $C         single dimension array used to store
             groups of $B elements
  CLVL       current $C element in use
  $CM        last $C element used
  $W         current value being addressed
 enddec OTHLNK

 PROC OTHLNK
   DO UNTIL ALVL EQ $AM
     IF $SBF($A(ALVL),6,1,'*') NE BLANK
     THEN ORGCODE=$QM//$SBF($A(ALVL),6,1,'*')//$QM
          STMT=STMT//' WITH $SCANX(ORGCODE,'//ORGCODE//')'
          STMT1=STMT1//' WITH $SCANX(POSORG,'//ORGCODE//')'
     endif;

     ALVL=ALVL + 1
     ORGCODE = BLANK
   enddo;

     IF STMT EQ BLANK (No ORGCODES found)
     THEN PRINT MESSAGE 'No Org Value found in Selection
            Criteria - See Data Base Administrator'
          EXIT PRCALINK
     endif;
   STMTO = 'FOR ORGCODE '
  &STMTO &STMT ACQUIRE

   DO for all ORGCODE
     GET DLID                     (use NEXT)
     $B = $QM//$W//$QM
     IF BLVL EQ 50
       THEN BLVL=0,CLVL=CLVL+1
     endif;
     $C(CLVL)=$C(CLVL)//$B($BC)
     BLVL=BLVL+1
   enddo;

   (Delete previous links)
   FOR &SEGMENT '&SEGMENT' DELETE SLORGLINK SLPOSLINK

   DO for all $C elements used
     FOR &SEGMENT '&SEGMENT' ADD SLORGLINK $C
   enddo;

   $B = BLANK,$C=BLANK
   STMTP = 'FOR POSNR '
   &STMTP &STMT1 ACQUIRE

   DO for all POSNR
     GET DLID                     (use NEXT)
     $B = $QM//$W//$QM
     IF BLVL EQ 50
       THEN BLVL=0,CLVL=CLVL+1
     endif;
     $C(CLVL)=$C(CLVL)//$B($BC)
   enddo;

   DO for all $C elements used
    FOR &SEGMENT '&SEGMENT' ADD SLPOSLINK $C
   enddo;

    FOR &SEGMENT '&SEGMENT' ADD SLLNKDTE '&TDATE'
endproc PRCALINK
```

4.3.10 PRINDX - PRIM INDEXING OF DATA

4.3.10.1 PURPOSE

The PRINDX procedure is used to establish the INDEX files for Active and Separated data. These files will normally be established once only and updated via XBRIDGING from the PRIM PERSIGN (Active) and PRIMSEP (Separated) files. If, however, a problem occurs with these INDEX files, they can be re-established with this procedure. See flow diagram in Figure 12 of the Appendix.

4.3.10.2 GENERAL INFORMATION

```
Program Name : PRINDX
Language Used: GIM Statements
               in SYSER via DAC-STMTS POL
Frequency of
Execution    : When data base is established,
               re-establish INDEX files, or
               incorporate a new INDEX file.
Input        : SYSER
                PRINDX1
                PRINDX2
                PRINDX3
                PRINDX4
                PRINDX5
                PRINDX6
                PRINDX7
                PRINDX8
               PERSIGN
                 PERSD
                 PEROCCE
                 PERSCCE
                 PERORGCODE
               PRIMSEP
                 PERSD
                 PEROCCE
                 PERSCCE
                 PERORGCODE
Output       : SDINDX
                 SDSSN
               ORGINDX
                 ORGSSN
               OCCEINDX
                 OCCESSN
               SCCEINDX
                 SCCESSN
               SSDINDX
                 SSDSSN
               SORGINDX
```

```
  SORGSSN
SOCCEINDX
  SOCCESSN
SSCCEINDX
  SSCCESSN
```

4.3.10.3 FUNCTIONAL REQUIREMENTS

This procedure will be used by the PRIM Data Base Manager or DBCC on instructions from the Data Base Manager to re-establish the data in the INDEX files for both Active and Separated data. This procedure will be used to originally establish the INDEX files, re-establish the files any time there is a question about the connections of INDEX data, or when a new INDEX file is added to the system. This will be done with DELETE-DATA statements to remove old INDEX values and INVERT statements to establish new INDEX values. The statements will be stored in SYSER records starting with PRINDX. These statements will be stored in the order of normal execution, however, any one could be executed alone.

4.3.10.4 INPUT

This procedure will not require a menu. The following statements which are stored in SYSER will be executed using the DAC-STMTS procedure in the following format:

```
E DAC-STMTS PRINDX1  (for SDINDX)
E DAC-STMTS PRINDX2  (for ORGINDX)
E DAC-STMTS PRINDX3  (for SCCEINDX)
E DAC-STMTS PRINDX4  (for OCCEINDX)
E DAC-STMTS PRINDX5  (for SSDINDX)
E DAC-STMTS PRINDX6  (for SORGINDX)
E DAC-STMTS PRINDX7  (for SSCCEINDX)
E DAC-STMTS PRINDX8  (for SOCCEINDX)
```

4.3.10.5 OUTPUT

Execution of this procedure will load into designated INDEX files all SSNORs associated to that file by associated element (ex: SDINDX file will contain all SSNORs by SD). This process will perform the following steps by using statements stored in SYSER file:

- delete data from Active INDEX file.
- load each Active INDEX file via INVERT verb.
- delete data from separated INDEX file.
- load each separated INDEX via INVERT verb.
- create a completion message for each PRINDX record used.
  Ex. PRINDX1 will have a completion message in PRINDX1MSG

4.3.11 PRCRTLINK - PRIM LINKS CREATED WEEKENDS BY DBCC

4.3.11.1 PURPOSE

This procedure will be used on weekends to create new component links to the ORGCODE and POSNR files. See flow diagram in Figure 13 of the Appendix.

4.3.11.2 GENERAL INFORMATION

Proc Name	: PRCRTLINK
Language Used	: GIM POL
Initiated by	: DBCC
Frequency of Execution	: Will be done each weekend.
Input	: SELECTION file (all segments) SEGNAME ORGCODE data SLLNKDTE ORGCODE POSNR
Output	: SELECTION file (all segments) SLORGLINK SLPOSLINK SLLNKDTE

4.3.11.3 FUNCTIONAL REQUIREMENTS

This procedure will use the ORGCODE portion of the SELECT field in each SELECTION segment to build an acquire statement against the ORGCODE and POSNR files. The ORGCODEs received will be added to the segment (SLORGLINK) and used by the component to query the ORGCODE file. The POSNRs received will be added to the segment (SLPOSLINK) and used by the component to query the POSNR file. A date will be stored in the SLLNKDTE field to show when the last SLORGLINK/SLPOSLINK link was created on PRIM.

4.3.11.4 INPUT

This procedure will require no menu. It will be initiated with the following statement:

```
E PRCRTLINK
```

This procedure will acquire through the SELECTION master, SEGNAME (to know what segment is being updated), ORGCODE (to know what ORGCODE to scan for in the ORGCODE and POSNR files), and SLLNKDTE (to know if new links are required for that segment).

4.3.11.5 OUTPUT

Do for each SELECTION segment:

```
 If the SLLNKDTE is LT current date
 then  - delete old SLORGLINK
       - Scan ORGCODE file for ORGCODE record containing
         ORGCODE from SELECTION Criteria.
       - add ORGCODEs acquired to current segment SLORGLINK
       - delete old SLPOSLINK
       - scan POSNR file for POSNR records containing ORGCODE
         from SELECTION Criteria
       - add POSNRs acquired to current segment SLPOSLINK
       - add current date to SLLNKDTE
 endif;
```

4.3.11.6 GENERAL PROGRAM CONSIDERATIONS

This program establishes ORGCODE and POSNR links for all PRIM Signon ORGs. This program is initiated each weekend by DBCC.

4.3.11.7 DETAILED PROGRAM SPECIFICATIONS

DEC PRCRTLINK

DATE	$odatea($dateg) (GIM date format)
I	current $A being used
$A	array containing file name
$W	current value being addressed
STMT	statement containing 'WITH $SCANX' statements
STMT1	statement containing 'WITH $SCANX' statements
STMT2	statement containing 'WITH $SCANX' statements
VAR	value containing the SLCRITERIA
SDVAR	value containing the Service Designations
OGVAR	value containing the ORGCODES
WLVL	count of CTEMP or DTEMP elements stored in current $C or $D array
CLVL	current $C array element in use
$C	array used to store groups of CTEMP elements
CTEMP	variable used to store values acquired from ORGCODE
DTEMP	variable used to store values acquired from POSNR
DLVL	current $D array element in use
$D	array used to store groups of DTEMP elements
J	current array being updated

```
enddec PRCRTLINK

PROC PRCRTLINK

  DO for all SELECTION segments
    FOR SELECTALL WITH PRESENT SLCRITERIA ACQUIRE

   GET file name          (use NEXT)

   $A = $W

    FOR filename ACQUIRE SLCRITERIA
    GET SLCRITERIA        (use GFAV's and GNAV's)

  DO for all SLCRITERIA values
    VAR = $W
    SDVAR = $SBF(VAR,0,1,'*')
    ORGVAR = $SBF(VAR,6,1,'*')
    IF ORGVAR and SDVAR NE blank
      THEN
        STMT = STMT//' WITH $SCANX(ORGCODE,'//$QM//ORGVAR//$QM')',
        STMT1 = STMT1//' WITH $SCANX(POSORG,'//$QM//ORGVAR//$QM')',
    and STMT2 = STMT2//' WITH $SCANX(POSCSD,'//$QM//SDVAR//$QM')',
    endif;
  enddo;

  DO for twenty values at a time
    FOR ORGCODE &STMT ACQUIRE
    GET value             (use NEXT)
    CTEMP = $W
    $C(CLVL) = $C(CLVL)//$QM//CTEMP//$QM
  enddo;

  DO for twenty values at a time
    FOR POSNR &STMT1 OF &STMT2 ACQUIRE
    GET value             (use NEXT)
    DTEMP = $W
    $D(DLVL) = $D(DLVL)//$QM//DTEMP//$QM
  enddo;

  (Delete previous links)
  FOR &($A(I)) '&($A(I)' DELETE SLORGLINK SLPOSLINK

   FOR &($A(I)) '&($A(I))' ADD SLORGLINK &($C(J))
      SLPOSLINK &($D(J)) SLLNKDTE '&DATE'
 enddo;
endproc PRCRTLINK
```

4.4 COMPONENT RETRIEVAL

4.4.1 DEFINITION

Component Retrieval in Release 1 will be through adhoc queries. All queries will be through the SEGACCESS file. The SEGACCESS file is a segmented file, segmented by ORG Access Levels which correspond to the SYSMAN2 Signon ORG. Flow diagrams are provided in Figures 14 and 15 of the appendix to graphically depict how a query will be controlled in PRIM.

4.4.2 ONLINE QUERIES

4.4.2.1 BASIC QUERIES - VIA SEGACCESS and SELECTION

4.4.2.1.1 PURPOSE

PRIM will provide components with online access to Official Personnel data. A great deal of the component's access will be done through online queries. All queries will be done through the SELECTION and SEGACCESS files. A query made through these files can be direct or complex. A complex query, however, will only search the SSNORs listed in the appropriate SEGACCESS file. This will eliminate the need for using reports for simple queries. See flow diagram in Figure 14 of the Appendix.

4.4.2.1.2 GENERAL INFORMATION

Language Used:	GIM statements
Initiated By :	Individual in a component authorized to use PRIM.

4.4.2.1.3 FUNCTIONAL DESCRIPTION

Each person accessing PRIM must have pre-authorization to use one of the SEGACCESS segments established on PRIM. All queries will be done through the SEGACCESS segment and data retrieved will be for only those records authorized for that segment.

4.4.2.1.4 INPUT

The component online query requirements from PRIM are varied and will include such items as:

1. List the Position, Schedule, and Grade of an employee to insure the proposed assignment conforms to Office of Personnel mandated assignment controls.

2. Count Positions of a selected Occupational Series.

3. Counts of LWOP Cases and N-T-E Dates.

4. List date of last change to a position.

5. Count of vacant positions.

6. List the date a position was officially deleted.

7. List daily strength for:

   - part-time
   - full-time
   - Dev Comp
   - LWOP
   - Details in/out
   - Sick Leave (approved for disability retirement)

8. List the Service Designation of a position versus the incumbent.

9. List the Sub-Category Code of a position.

10. List selected Cover Items.

11. List FLSA Designation of Employee.

12. List projected WGI to ascertain whether to delay a promotion until the WGI is granted.

4.4.2.1.5 OUTPUT

Output from a GIM statement will follow in the standard GIM output vertical/horizontal print rule. The print will be horizontal if the total record length is LE 80 and vertical if it is GT 80.

4.4.2.2 SEARCH - SEARCH NAME AND HPOSNR FILES

4.4.2.2.1 PURPOSE

This procedure is used to search for a particular record in either the NAME or the HPOSNR file. This procedure is used, rather than a statement, to restrict the user to only one record at a time. See flow diagram in Figure 15 of the Appendix.

4.4.2.2.2 GENERAL INFORMATION

```
Proc Name     : SEARCH
Language Used: GIM POL
Initiated by : User
Frequency of
Execution     : Random
Input         : Parameters
                   File Name (NAME,HPOSNR)
                   Value to search for
```

4.4.2.2.3 FUNCTIONAL REQUIREMENTS

The NAME and HPOSNR files are not linked to any primary PRIM files. At times, a user may know a person's name but not the SSNOR or may wish to look at a deleted position number. Because of security restrictions, the user will be limited to only one record from either file per request.

4.4.2.2.4 INPUT

This procedure does not require a menu. However, a file name and value to be searched for must be passed at execution.

```
Example:     E SEARCH NAME (last*first)
             E SEARCH HPOSNR (position number)
```

4.4.2.2.5 OUTPUT

Using the parameters passed, a query statement will be made against the file specified for the record specified. The NAME file responses will list all individuals with the same last and first name specified. The HPOSNR response will only list one position record.

4.4.2.2.6 GENERAL PROGRAM CONSIDERATIONS

All retrieval from the HPOSNR and NAME files will be through the SEARCH procedure, direct queries against these files will not be allowed. The procedure will not use a menu but will expect two (2) parameters via prompts (file name and ID of record to be retrieved).

4.4.2.2.7 DETAILED

```
DEC SEARCH
   $A       array containing file name
   $C       array containing ID of record
   $B       array used to contain file name synonym
 enddec SEARCH


PROC SEARCH

  VOPREADI   'ENTER FILE TO BE ACCESSED (NAME, HPOSNR)',$A

  VOPREADI   'ENTER RECORD ID FOR RETRIEVAL',$C

  BUILD and EXECUTE query statements for NAME or HPOSNR

  FOR &($b(1)) '&($c(1))' LIST

  EXIT PROC SEARCH

 endproc SEARCH
```

4.4.2.3 LISTSTMT - GENERALIZED QUERY

4.4.2.3.1 PURPOSE

The procedure will use generalized statements entered by the Data Base Manager and personalize the statement with pre-determined parameters. Because of the security matrix on PRIM, users will not be allowed to use the LIST verb with a data file name to list all elements or all records. The user will have to go through a unique SEGACCESS or SELECTION file and list each element. Because of the large number of elements, a statement will be provided in the STATEMENT file for this purpose.

4.4.2.3.2 GENERAL INFORMATION

Proc Name	: LISTSTMT
Language Used	: GIM POL
Initiated by	: User
Frequency of Execution	: Random
Input	: Parameters STATEMENT name ORG initiating Any additional parameters pre-established for that statement.

4.4.2.3.3 FUNCTIONAL REQUIREMENTS

Statements must be entered in the STATEMENT file by the Data Base Manager and can use the LIST, COUNT, REPORTW, or EXTRACT verbs. Statements added should be generalized so they can be used by all components.

4.4.2.3.4 INPUT

Each statement used by LISTSTMT will have different parameter requirements, therefore, the user must refer to the sample given for each statement. The first and second parameters will always be STATEMENT name and ORG followed by any additional parameters, if required.

```
Ex.  E LISTSTMT POSNR T56PER
```

4.4.2.3.5 OUTPUT

If the correct parameters are passed, the output will be from the specified file for the specified ORG. One ORG accessing data for another ORG will envoke security restrictions and will not be allowed.

4.4.2.3.6 GENERAL SPECIFICATIONS

The position of each parameter for a STATEMENT record should be shown by a percent sign (%) followed by the parameter number, followed by a space. Ex. (For %1 LIST PERNAMEOR) would execute as (FOR T56PER LIST PERNAMEOR) when executed by E LISTSTMT PERTEST T56PER. NOTE: There are two spaces following %1. Up to 5 parameters are allowed.

4.4.2.3.7 DETAILED SPECIFICATIONS

```
LISTSTMT (Executes Pre-established statements)

DEC LISTSTMT
   $AM    Total $A arrays read
   $A(1)  STATEMENT Name
   $A(2)  Signon ORG of Initiator
   $A(3)  first parameter
   $A(4)  second parameter
   $A(5)  third parameter
   $A(6)  forth parameter
   $A(7)  fifth parameter
enddec LISTSTMT

PROC LISTSTMT

   READ parameters into $A array

  ACQUIRE STATEMENT record per $A(1)

   IF STATEMENT record not found
   THEN Issue error (STMT not found)
        EXIT LISTSTMT
   endif;

   IF $AM LT 2
   THEN Issue error (not enough parameters)
        EXIT LISTSTMT
   endif;

   IF $AM GT 2
   THEN SCAN statement for %1
        replace with $A(3) value
   endif;
```

```
    IF $AM GT 3
    THEN SCAN statement for %2
         replace with $A(4) value
    endif;

    IF $AM GT 4
    THEN SCAN statement for %3
         replace with $A(5) value
    endif;

    IF $AM GT 5
    THEN SCAN statement for %4
         replace with $A(6) value
    endif;

    IF $AM EQ 6
    THEN SCAN statement for %5
         replace with $A(7) value
    endif;

    Execute READF  (GIM System module)
       to change all single quotes to double

    Execute STATEMENT

  endproc LISTSTMT
```

4.4.3 OFFLINE REPORTS

4.4.3.1 PURPOSE

PRIM will provide components with RAMIS report definitions currently used on HRS2, as well as, frequently used REPORTW statements which can be used for offline reporting.

4.4.3.2 GENERAL INFORMATION

Language Used:	RAMIS REPORTW
Initiated by :	Individual in a component authorized to use PRIM

4.4.3.3 FUNCTIONAL DESCRIPTION

Each person accessing PRIM must have pre-authorization to use one of the SEGACCESS segments established on PRIM. All REPORTW or extract statements will be done through the SEGACCESS segment and data retrieved will be for only those records authorized for that segment.

4.4.3.4 INPUT

The initiator of a REPORTW statement or an extract statement will be required to pass the SEGACCESS segment name and the desired report selection to initiate an offline report.

4.4.3.5 OUTPUT

Output for a REPORTW report will follow the format specified in the REPORTW statement. Output will be produced with a ROUTE *A and PRINT command. A successful extract will result in a tape or disk of required data. A job must then be executed on VM to use the tape/disk with a RAMIS report program to produce final output.

4.4.4 <u>DATA BASE ACCESS VIOLATION REPORT</u>

4.4.4.1 PURPOSE

The access violation report is used by the PRIM Data Base Manager to locate those persons having difficulty using the system or violating data base access security restrictions.

4.4.4.2 GENERAL INFORMATION

Program Name :	PRIMSEC
Language Used:	RAMIS
Initiated By:	DBCC
Input :	DBCC Source
Output :	PRIM Security Access Violation Data

4.4.4.3 FUNCTIONAL DESCRIPTION

The PRIM System must provide weekly reports to the PRIM Data Base Manager identifying any violations of established read and/or write access control to the PRIM System. The violations of established access control report should be similar to the HRS2 Data Base report. The report will also identify individuals who have been forced off the PRIM System after 3 security code violations.

4.4.4.4 INPUT

Input will be provided by DBCC.

4.4.4.5 OUTPUT

The report should list at a minimum:

1. Data Base Name
2. User Identification
3. Terminal Identification
4. Date and Time of violation
5. If appropriate:
    - a) Data List Name
    - b) Transaction Number
    - c) Verb Name

4.4.5 DATA BASE EXCEPTION REPORT

4.4.5.1 PURPOSE

This report is used by the Data Base Manager to identify those statements executing longer than 3 minutes.

4.4.5.2 GENERAL INFORMATION

Program Name	: PRIMEXCP
Language Used	: RAMIS
Initiated By	: DBCC
Input	: DBCC Source
Output	: PRIM Exceptions Data

4.4.5.3 FUNCTIONAL DESCRIPTION

A weekly Data Base Exception Report will be produced to identify query statements executing longer than 3 minutes. The report will be used by the Data Base Manager to focus on component areas that may need guidance on creating cost effective queries. This report can use the same program used for the HRS2 Data Base.

4.4.5.4 INPUT

Input to this report will be provided by DBCC.

4.4.5.5 OUTPUT

The report should list at a minimum:

1. Data Base Name
2. User Identification
3. Terminal Identification
4. Date and Time of statement
5. If appropriate:
   a) Data List Name
   b) Transaction Number

4.4.6 <u>ACF2 VIOLATION REPORT</u>

4.4.6.1 PURPOSE

A standard report will be available online or offline to an ACF2 Control Officer showing ACF2 violations by persons attempting to illegally access a dataset.

4.4.7 <u>PRNTEPRG REPORT</u>

4.4.7.1 PURPOSE

The PRNTEPRG Report will give the user a list of SSNORs and associated data which was purged from each segment of the SEGACCESS file.

4.4.7.2 GENERAL INFORMATION

```
Program Name : PRNTEPRG Report
Language Used: GIMS REPORTW
Initiated By:  PRNTEPRG Procedure
Input        : Records with Purge Date (PRGDTE)
               LE System Date
Output       : SYSMAN2 (SEGACCESS Segment Name)
                 SSNOR
                 STATUS
                 NAME
```

4.4.7.3 FUNCTIONAL DESCRIPTION

The PRNTEPRG Report will be produced by the PRNTEPRG Procedure. The procedure will produce the report using a REPORTW statement just prior to executing the delete statement. The report will be printed with a ROUTE *A and a PRINT command which will route the listing to the PRIM Data Base Manager. A comment line will be used to slot the listing to the Data Base Manager.

4.4.7.4 INPUT

The PRNTEPRG Procedure will create and execute a REPORTW statement for each SEGACCESS that has data to purge.

4.4.7.5 OUTPUT

The PRNTEPRG report will be a basic REPORTW output. It will provide the user by SEGACCESS (SYSMAN2) with SSNOR, STATUS, Name for each item purged.

4.4.8 <u>DBSTATS REPORT</u>

4.4.8.1 PURPOSE

The DBSTATS report is a statistical report of the outlay of the data base resources.

4.4.8.2 GENERAL INFORMATION

Program Name	: DBSTATS Report
Initiated By	: DBCC nightly
Input	: DBCC source
Output	: Statistics of PRIM Data Base

4.4.8.3 FUNCTIONAL DESCRIPTION

The DBSTATS report will be produced nightly by DBCC for the PRIM Data Base Manager. This report will be compared with a similar report produced for the HRS2 Data Base for verification that the nightly Interface between the two systems was successful. The report will also be used by the PRIM Program Applications Specialist (PAS) to determine when a reallocation is required.

4.4.9 <u>PRCHANGE REPORT - REMOVE, SEPARATE, AND SSN CHANGES</u>

4.4.9.1 PURPOSE

The PRCHANGE report will be a REPORTW produced report and will provide the user a list of Active SSNORs that are to be deleted (with SDTE NE current date), Separated SSNORs that are to be deleted (SSDTE NE current date) and SSNORs that have changed (IFCSIGN old NE $$$$$$$$$ and NE new).

4.4.9.2 GENERAL INFORMATION

Proc Name	: PRCHANGE Report
Language Used	: GIM REPORTW
Frequency Of Execution	: Random per component need
Initiated by	: Individual Components
Input	: SEGACCESS (SYSMAN2) STATUS (A/S) System Date (SDTE or SSDTE) Purge Date (PRGDTE) INTERFACE (PRIM) IFCSIGN (old/new SSNOR) IFCNAMEOR (old/new name)
Output	: Report of Component link data

4.4.9.3 FUNCTIONAL DESCRIPTION

The PRCHANGE report will be created by a REPORTW statement. By going through the master SEGACCESS segment with a selection statement for SDTE NE current date or SSDTE NE current date and with IFCSIGN 'OLDSSNOR' NE '$$$$$$$$$' and NE 'NEWSSNOR' a report can be produced to advise the generating office of records that will soon be purged and the date of purge (PRGDTE) as well as, SSNORs that have changed and what they were changed to.

4.4.9.4 INPUT

A file will be provided which will contain statements which can be adapted for any segment. This can be done with a temporary change and execution by any individual. If, however, this report is used extensively by all segments, a procedure can be implemented that will accept parameters via a procedure and change the statement and execute it. (TBR)

4.4.9.5 OUTPUT

The output will be produced with a ROUTE *A and a PRINT command for the appropriate printer. It will contain only data for the initiating segment and that data will include:

			PRIM			
SSNOR	STATUS	NAME	IN Date	Purge Date	New SSNOR	NEW NAME

Chapter 5

CONVERSION - NOT APPLICABLE TO PRIM

Chapter 6

APPENDIX

Figure 1 - PRIM FUNCTIONAL FLOW
Figure 2 - PRIM HARDWARE CONFIGURATION
Figure 2.1 - DMS ARCHITECTURE
Figure 3 - PRIM REQUIREMENTS MATRIX
Figure 4 - PRIM PROCEDURES
Figure 5 - PRIM SHORT FILE DESCRIPTIONS
Figure 6 - PRDDUMP & PRDLOAD DATA FLOWS
Figure 7 - PRCAESTB DATA FLOW
Figure 8 - PRCAUPDT & PRCHGLOAD DATA FLOWS
Figure 9 - PRCATRSF & PRNTEPRG DATA FLOWS
Figure 10 - PRINTPRG & PRACCUPDT DATA FLOWS
Figure 11 - PRCALINK DATA FLOW
Figure 12 - PRINDX DATA FLOW
Figure 13 - PRCRTLINK DATA FLOW
Figure 14 - BASIC QUERY DATA FLOW
Figure 15 - SEARCH DATA FLOW