APPENDIX 11
ANALYSIS RETURN, CP PROCESSOR
FINAL SOFTWARE REPORT
DATA NO. A005

INTEGRATED ELECTRONIC WARFARE SYSTEM
ADVANCED DEVELOPMENT MODEL (ADM)

PREPARED FOR:
NAVAL AIR DEVELOPMENT CENTER
WARMINSTER, PENNSYLVANIA

7800987-12

1 OCTOBER 1977

UNCLASSIFIED
APPENDIX 11
CLASSIFICATION PROCESSOR, ANALYSIS RETURN, DESIGN SPECIFICATION
FINAL SOFTWARE REPORT
DATA ITEM A005

INTEGRATED ELECTRONIC WARFARE SYSTEM (JEWS)
ADVANCED DEVELOPMENT MODEL (ADM)

Contract No. N62269-75-C-0070

Prepared for:
Naval Air Development Center
Warminster, Pennsylvania

Prepared by:
RAYTHEON COMPANY
Electromagnetic Systems Division
6380 Hollister Avenue
Goleta, California 93017

1 OCTOBER 1977
# COMPUTER SUBPROGRAM DESIGN DOCUMENT

**ANALYSIS RETURN PROCESSING, JEWS, ADM**

<table>
<thead>
<tr>
<th>FUNCTION</th>
<th>APPROVED</th>
<th>DATE</th>
<th>FUNCTION</th>
<th>APPROVED</th>
<th>DATE</th>
</tr>
</thead>
<tbody>
<tr>
<td>WRITER</td>
<td>John A. Clark</td>
<td>1/15/76</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>George D. Morgan</td>
<td>12/1/76</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

## REVISIONS

<table>
<thead>
<tr>
<th>CHK</th>
<th>DESCRIPTION</th>
<th>REV</th>
<th>CHK</th>
<th>DESCRIPTION</th>
<th>REV</th>
</tr>
</thead>
</table>

## SHEET NO.

<table>
<thead>
<tr>
<th>REVISION</th>
<th>SHEET NO.</th>
<th>REVISION</th>
<th>SHEET NO.</th>
</tr>
</thead>
</table>

---

RAYTHEON COMPANY
LEXINGTON, MASS. 02173

CODE IDENT NO. 49956

SPEC NO. 58959-GT-0761

SHEET 1 OF 103 REV 103
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>SCOPE</td>
<td>3</td>
</tr>
<tr>
<td>1.1</td>
<td>Identification</td>
<td>3</td>
</tr>
<tr>
<td>1.2</td>
<td>Subprogram Tasks</td>
<td>3</td>
</tr>
<tr>
<td>2.0</td>
<td>APPLICABLE DOCUMENTS</td>
<td>6</td>
</tr>
<tr>
<td>2.1</td>
<td>Computer Program Performance Specification</td>
<td>6</td>
</tr>
<tr>
<td>2.2</td>
<td>Computer Program Design Specification</td>
<td>7</td>
</tr>
<tr>
<td>2.3</td>
<td>Data Base Design Document</td>
<td>7</td>
</tr>
<tr>
<td>2.4</td>
<td>Miscellaneous Documents</td>
<td>7</td>
</tr>
<tr>
<td>3.0</td>
<td>REQUIREMENTS</td>
<td>8</td>
</tr>
<tr>
<td>3.1</td>
<td>Subprogram Detailed Description</td>
<td>8</td>
</tr>
<tr>
<td>3.2</td>
<td>Subprogram Flow Diagrams</td>
<td>35</td>
</tr>
<tr>
<td>3.3</td>
<td>Computer Subprogram Environment</td>
<td>68</td>
</tr>
<tr>
<td>3.4</td>
<td>Input/Output</td>
<td>84</td>
</tr>
<tr>
<td>3.5</td>
<td>Required External Subroutines</td>
<td>86</td>
</tr>
<tr>
<td>3.6</td>
<td>Conditions for Initialization</td>
<td>86</td>
</tr>
<tr>
<td>3.7</td>
<td>Subprogram Limitations</td>
<td>87</td>
</tr>
<tr>
<td>3.8</td>
<td>Interface Description</td>
<td>88</td>
</tr>
</tbody>
</table>

Figure 1a  (Scan) Analysis Return Message Format  96
Figure 1b  " " " " " "  97
Figure 2a  Analysis Request Message Format  98
Figure 2b  " " " " " "  99
Figure 3a  Classification Message Format  100
Figure 3b  " " " " " "  101
Figure 4a  Update Message Format  102
Figure 4b  " " " " " "  103
1.0 SCOPE

1.1 IDENTIFICATION

This document describes the implementation of the Analysis Return Functional Group (ARFG) of the SC Operational Software resident in the Classification Processor (CP).

1.2 SUBPROGRAM TASKS

1.2.1 Analysis Return Driver (ANDR)

ANDR shall have the responsibility of decoding Analysis Return messages. These messages originate in the Analysis Processor and are the response to an analysis request. After decoding, ANDR shall call the appropriate Analysis Return processing routine.

1.2.2 New Emitter Processing 2 (ANNE2)

ANNE2 shall process Analysis Return messages which have a return module code of 1. These messages are the result of deinterleaving analysis requests from New Emitter Processing 1 (see Sorter Message Processing Design Document 53959-GT-0755).

1.2.3 New Emitter Processing 3 (ANNE3)

ANNE3 shall process Analysis Return messages which have a return module code of 2. These messages are the result of contemporaneous analysis requests from ANNE2.

1.2.4 NOFA 2 Processing 2 (ANNA2)

ANNA2 shall process Analysis Return messages which have a return module code of 3. These messages are the result of scan analysis requests from NOFA2 Processing 1 (see Sorter Message Processing Design Document, 53959-GT-0755).
1.2.5  NOFA 2 Processing 3 (ANNA3)
ANNA3 shall process Analysis Return messages which have a return module code of 4. These messages are the result of contemporaneous analysis requests from ANNA2.

1.2.6  Emitter of Concern (EOC) Processing 2 (ANOC2)
ANOC2 shall process Analysis Return messages which have a return module code of 5. These messages are the result of scan analysis requests from EOC Processing 1 (see Sorter Message Processing Design Document, 53959-GT-0755).

1.2.7  EOC Processing 3 (ANOC3)
ANOC3 shall process Analysis Return messages which have a return module code of 6. These messages are the result of contemporaneous analysis requests from ANOC2.

1.2.8  EOC Processing 4 (ANOC4)
ANOC4 shall process Analysis Return messages which have a return module code of 7. These messages are also the result of contemporaneous analysis requests from ANOC2.

1.2.9  Emitter Classification 2 (ANEC2)
ANEC2 shall process Analysis Return messages which have a return module code of 8. ANEC2 shall be the principal routine for accomplishing the second level of emitter classification, namely, the elimination of candidates on the basis of scan type and scan period from a list created by Emitter Classification 1 (see Emitter Classification 1 Design Document, 53959-GT-0760).
1.2.10 Emitter Classification 3 (ANEC3)

ANEC3 shall process Analysis Return messages which have a return module code of 9. These messages are the result of contemporaneous analysis requests from ANEC2.
2.0 APPLICABLE DOCUMENTS

The following documents, of the exact issue shown, form a part of this specification to the extent specified herein. In the event of conflict between the documents referenced herein and the contents of this specification, the contents of the Computer Program Design Specification for the Integrated Electronic Warfare System (IEWS) Advanced Development Model (ADM) Program shall be considered superseding requirements.

2.1 COMPUTER PROGRAM PERFORMANCE SPECIFICATION


2.1.1 Applicable CPPS Paragraphs

<table>
<thead>
<tr>
<th>Paragraph</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Analysis Return Driver</td>
<td>Not Specified Explicitly</td>
</tr>
<tr>
<td>New Emitter Processing 2</td>
<td>3.3.2.1.2.1</td>
</tr>
<tr>
<td>New Emitter Processing 3</td>
<td>3.3.2.1.2.1</td>
</tr>
<tr>
<td>NOFA 2 Processing 2</td>
<td>3.3.2.1.2.2</td>
</tr>
<tr>
<td>NOFA 2 Processing 3</td>
<td>3.3.2.1.2.2</td>
</tr>
<tr>
<td>EOC Processing 2</td>
<td>3.3.2.1.2.2</td>
</tr>
<tr>
<td>EOC Processing 3</td>
<td>3.3.2.1.2.2</td>
</tr>
<tr>
<td>EOC Processing 4</td>
<td>3.3.2.1.2.2</td>
</tr>
<tr>
<td>Emitter Classification 2</td>
<td>3.3.3</td>
</tr>
<tr>
<td>Emitter Classification 3</td>
<td>3.3.3</td>
</tr>
</tbody>
</table>
2.2 COMPUTER PROGRAM DESIGN SPECIFICATION


2.3 DATA BASE DESIGN DOCUMENT

The Common Data Base Design Document, System Controller Unit, IEWS, ADM, document No. 53959-GT-0751, shall apply to this subprogram.

2.4 MISCELLANEOUS DOCUMENTS

The following documents shall apply to this subprogram:

<table>
<thead>
<tr>
<th>Document No.</th>
<th>Document Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>53959-GT-0756</td>
<td>Computer Subprogram Design Document, Executive, IEWS, ADM</td>
</tr>
<tr>
<td>53959-GT-0755</td>
<td>Computer Subprogram Design Document, Sorter Message Processing, IEWS, ADM</td>
</tr>
<tr>
<td>53959-GT-0760</td>
<td>Computer Subprogram Design Document, Emitter Classification 1 IEWS, ADM</td>
</tr>
<tr>
<td>53959-GT-0759</td>
<td>Computer Subprogram Design Document, Data Extraction, IEWS, ADM</td>
</tr>
<tr>
<td>WS-8506</td>
<td>Requirements for Digital Computer Program Documentation</td>
</tr>
<tr>
<td>Revision 1,</td>
<td>1 November 1971</td>
</tr>
</tbody>
</table>

3.0 REQUIREMENTS

3.1 SUBPROGRAM DETAILED DESCRIPTION

3.1.1 Analysis Return Driver (ANDR)

ANDR shall be the driver routine of the Analysis Return Functional Group. The EXEC shall pass to ANDR a pointer to the Analysis Return message ANMNO word (see Figure 1), in the X-register. ANDR shall use the pointer to retrieve the return module code from the message. The code shall be verified to be a valid code and then used as an index to the Analysis Return Processing table (ANMPT). The index (equal to the value of ANRMC) shall be added to the base address of the table and this address shall be used indirectly to call one of the Analysis Return processing routines (whose list of symbolic names constitute ANMPT). Each of the Analysis Return processing routines shall receive the same input:

1) The address of the Analysis Return message word 2 in the X-register.

2) The emitter file number from the message in the A-register.

After the Analysis Return processing routine has completed its task, control shall be returned to the driver. If the Analysis Return processing routine has returned via return 1, the X-register shall contain a pointer to an analysis request message buffer and the Executive message function shall be called to output the analysis request (label AND99). Control shall then be returned to the EXEC. If the Analysis Return processing routine has returned via return 2, control shall be returned to the EXEC.
3.1.2 New Emitter Processing 2 (ANNE2)

ANNE2 shall perform the following tasks:

(a) Calculate the address (EFP) of the ETF entry to be processed.
(b) Accept and process the results of deinterleaving in PRI Test 2 (ANPT2).
(c) Assess PW quality and check for presence of long pulse data.
(d) Perform a check for emitters having harmonically related PRI's in Harmonic PRI Test 1 (ANHP1).
(e) Pass an analysis request message for contemporaneous analysis to the analysis return driver (ANDR).

To accomplish this, ANDR shall call ANNE2 with a pointer to word 2 of the Analysis Return message stored in the X-register. The A-register shall contain the emitter file number (EFN) in the least significant byte.

ANNE2 shall immediately call subroutine SOGET which shall compute the address of the emitter track file (ETF) entry and shall return it in the B-register as EFP. ANNE2 shall then call subroutine PRI Test 2 (ANPT2) to process the results of deinterleaving.

The subroutine PW Test (ANPWT) shall be called by ANNE2 to assess the validity of the PW data and to check for long pulse data. ANNE2 shall test the return from ANPWT to determine if long pulse data has been detected. If long pulse data is indicated, the PW validity (EFPWV) bit in the ETF shall be reset to zero and then ANNE2 shall proceed. (At this point a time-out period shall be initiated and a return made when long pulse processing is implemented). If long pulse data is not indicated, the
processing shall continue directly to call Harmonic PRI Test 1 (ANHP1). (ANHP1 will check for the presence of emitters with harmonically related PRI's when contemporaneous analysis is implemented).

ANNE2 shall test the return from ANHP1 to determine if contemporaneous analysis (CA) should be requested. If CA is not required, ANNE2 shall reset the analysis wanted (AW) bit and set the CA Request (CAR) bit in the analysis request message. If CA is required, ANNE2 shall set both the AW and CAR bits. (CA required path shall never be executed until CA is implemented). In either case, ANNE2 shall store the EFN in the analysis request message. ANNE2 shall load the address of the analysis request message into the X-register and shall return to the analysis return driver (ANDR).

3.1.2.1 PRI Test 2

The logic flow for PRI Test 2 (ANPT2) shall be as shown in 3.2.2.1. ANPT2 shall be a direct return to the calling routine. (This is a dummy subroutine which will be enhanced when deinterleaving is implemented).

3.1.2.2 Pulse Width Test

The logic flow for PW Test (ANPWT) shall be as shown in 3.2.2.2. A subroutine call shall be made to ANPWT with the address of the ETF entry, EFP, in the B-register. ANPWT shall extract the PW code, EFPW (EFP), from the ETF and shall test for the value B'1111'. If EFPW (EFP) = B'1111', then the long pulse indication EFLP (EFP) shall be set to 1 and a normal return shall be made to the calling routine indicating the presence of long pulse data. If EFPW (EFP) ≠ B'1111', the PW quality factor, EFQPW (EFP) shall be tested. If EFQPW (EFP) = B'1111', indicating bad quality, the PW validity bit, EFPWV (EFP), shall be reset to zero. If EFQPW (EFP) ≠ B'1111', then EFPWV (EFP) shall be set to 1. For either result
of the EFQPW test, the return address shall be incremented by one to indicate that no long pulse data is present. ANPWT shall then return to the calling routine.

3.1.2.3 Harmonic PRI Test 1

The logic flow for Harmonic PRI Test 1 (ANHP1) shall be as shown in 3.2.2.3. ANHP1 shall increment the return address by one to indicate no contemporaneous analysis required and shall return. (When contemporaneous analysis is implemented, ANHP1 shall be enhanced).

3.1.3 New Emitter Processing 3 (ANNE3)

ANNE3 shall perform the following tasks:

(a) Calculate the address (EFP) of the ETF entry to be processed.

(b) Accept and process the results of contemporaneous analysis (CA) in Harmonic PRI Test 2 (ANHP2).

(c) Assess frequency quality and store result in ETF by calling Frequency Test (ANFQT).

(d) Output a classification message to the Executive and return to the Analysis Return Driver (ANDR).

To accomplish this, ANDR shall call ANNE3 with a pointer to word 2 of the Analysis Return message in the X-register. The A-register shall contain the emitter file number (EFN) in the least significant byte.

ANNE3 shall immediately call subroutine SOGET which shall compute the address of the emitter track file (ETF) entry and shall return it in the B-register as EFP. ANNE3 shall then call Harmonic PRI Test 2 to process the results of contemporaneous analysis (CA).
The subroutine Frequency Test (ANFQT) shall be called by ANNE3 to assess the validity of the frequency data. Upon return from ANFQT, ANNE3 shall output a classification message to the Executive. The X-register shall contain a pointer to the first word in the classification message. ANNE3 shall then return to the analysis return driver (ANDR).

3.1.3.1 Harmonic PRI Test 2

The logic flow for Harmonic PRI Test 2 (ANHP2) shall be as shown in 3.2.3.1. ANHP2 shall be a direct return to the calling routine. (This is a dummy subroutine which will be enhanced when CA is implemented).

3.1.3.2 Frequency Test

The logic flow for the Frequency Test (ANFQT) shall be as shown in 3.2.3.2. A subroutine call shall be made to ANFQT with the address of the ETF entry (EFP) in the B-register. ANFQT shall establish a local data area to store PARAM, M, and QVAL in consecutive locations. ANFQT shall set PARAM equal to the value of EFFREQ (EFP). PARAM shall be tested for the presence of all 1's which shall be the default frequency value if no IFMR output occurs. If the default value is detected, the frequency validity, EFV (EFP), shall be reset to zero to indicate bad frequency data and ANFQT shall return. If the default value is not detected, processing shall proceed by setting M to 15 and QVAL to the value of EFQF (EFP).

ANFQT shall call parameter quality test (SOQUT) with a pointer in the X-register to PARAM. SOQUT shall return with an indication of good quality (GDQ) contained in the A-register. The value of GDQ shall be stored in the frequency validity bit EFV (EFP). ANFQT shall return to the calling routine.
3.1.4 NOFA2 Process 2 (ANNA2)

3.1.4.1 ANNA2

ANNA2 shall be called by the Analysis Return Driver (ANDR), if the return module code of the analysis return data (AR data) is X'03' (See Figure 1). The driver shall pass to ANNA2 the address of AR data word 2 in the X-register and the emitter file number (EFN) in the A-register. SOGET (see Sorter Message Processing CSDD) shall immediately be called to convert EFN to the address of an Emitter Track File entry (EF entry). The address shall be returned by SOGET in the B-register. The scan type, as determined by the scan analysis module, shall be retrieved from the AR data. If the scan type is "sidelobe", processing shall continue at label ANN1Ø. If the scan type indicates a null measurement, the return address on the stack shall be incremented by 1 so that subroutine return No. 2 is performed (label ANN9Ø). Control shall then be returned to the AR driver.

3.1.4.2 Subroutine Returns from ANNA2

Two returns from ANNA2 shall be possible:

1) AR driver shall output an analysis request message to the EXEC. A pointer to the message buffer shall be returned to the AR driver in the X-register.

2) AR driver shall not output any Analysis Request message to the EXEC.

3.1.4.3 ANN1Ø

The scan state indicator (EFSIND) shall be retrieved from the EF entry. If this indicator is Ø, it shall be set equal to 1 (in the EF entry) and control shall be returned to the AR driver (via ANN9Ø). If EFSIND is not Ø, processing shall continue at label ANN2Ø.
3.1.4.4 ANN2Ø

The scan type (EFSTYP) shall be retrieved from the EF entry and from the analysis return data (ANSTY). The two codes shall be compared. If they are not equal, processing shall continue at label ANN3Ø. If they are equal, the scan period (EFSPRD) shall be retrieved from the EF entry and from the analysis data (ANSPR). The absolute value of the difference of the two scan periods shall be computed. If this difference is less than, or equal to, the constant $\Delta_{SPRD}$, control shall be returned to the AR driver (via ANN9Ø). Otherwise, processing shall continue at label ANN3Ø.

3.1.4.5 ANN3Ø

The value of scan type code (EFSTYP) in the EF entry shall be set to that of the AR data (ANSTY). Similarly, for the scan period (EFSPRD). An attempt shall then be made to reclassify this "changed" emitter. To do this, the emitter file number (ANEFN) shall be retrieved from the AR data and saved in the Analysis Request message buffer (ANNCA). ANEFN shall then be passed to the Level 1 search module (ECLV1) in the X-register (see Emitter Classification 1 CSDD). If this routine finds no candidates (1st return from ECLV1), control shall be returned to the AR driver via ANN9Ø. If there are candidates (2nd return from ECLV1), a pointer to the candidate list (in the common data base) shall be returned in the X-register. The Level 2 Search Module (ANLV2) shall receive this pointer to the candidate list as input and shall output a refined candidate list. If there are no candidates as a result of the Level 2 Search, control shall be returned to the AR driver via ANN9Ø. If candidates still exist, the pointer to the refined candidate list shall be saved in the Analysis Request message buffer, ANNCA. (See Figure 2). The New Emitter Link Analysis 1 module (ANEL1) shall be called to determine if contemporaneous analysis is required.
ANEL1 shall be a dummy routine in the priority 1 implementation. If contemporaneous analysis is required (return 2, which is never executed in priority 1), the analysis wanted bit (ANAW) shall be set in the Analysis Request message buffer, ANNCA. If not required (return 1), the ANAW bit shall be cleared. If either case, return 1 shall be performed to return control to ANDR, with the address of ANNCA in the X-register.

3.1.5 NOFA2 Process 3 (ANNA3)

3.1.5.1 ANNA3

This routine shall be called by the Analysis Return driver (ANDR), if the return module code of the analysis return data (AR data) is X'04' (See Figure 1). The driver shall pass to ANNA3 the address of the AR data in the X-register. The New Emitter Link Analysis 2 routine (ANEL2) and the Family Association routine (ANFAM) shall immediately be called. Then the Ambiguity Resolution (ANAMB) shall be called. This routine shall be passed a pointer to the AR data in the X-register. Finally, the return-to-AR-driver address (on the stack) shall be incremented, so that ANNA3 will never cause any analysis request messages to be sent to the EXEC by the AR driver.

3.1.5.2 Subroutine Returns from ANNA3

ANNA3 shall always cause the instruction after the call to ANNA3 to be skipped. The returns from ANNA3 shall be:

1) Null. Never executed.

2) AR driver shall not output any Analysis Request message to the EXEC.
3.1.6 EOC Process 2 (ANOC2)

3.1.6.1 ANOC2

This routine shall be called by the Analysis Return driver (ANDR), if the return module code of the analysis return data is X'BD5' (See Figure 1). The driver shall pass to ANOC2 the address of the AR data in the X-register and the EFN in the A-register. The EFN shall be saved in the update message buffer, ANUPM (See Figure 4) and in the analysis request buffer, ANOCA, (See Figure 2). SOGET shall then be called to convert EFN into an EF entry address, which shall be returned in the B-register.

The Scan Test 2 (ANST2) routine shall then be called. It shall receive as input the pointer to the AR Data in the X-register. Upon return, the pointer to the candidate list (ANPTR) shall be retrieved from the AR data. This shall be passed as input to the Level 2 Search routine (ANLV2) in the X-register. If Level 2 Search finds candidates (2nd return), a pointer to the refined candidate list shall be returned in the X-register and control shall be transferred to label ANC30. Otherwise, processing shall continue at label ANC10.

3.1.6.2 ANC10

The platform link pointer (EFPLNK) shall be retrieved from the EF entry. A test shall be made to see if EFPLNK is equal to the emitter file number. If not equal, the emitter is "platform linked" and the Delete Link Processing (SODLK) routine shall be called. Otherwise, the call to SODLK shall be skipped. The EXEC shall then be called to output an update message, ANUPM (See Figure 4). Finally, the return-to-AR-driver address on the stack shall be incremented and subroutine return No. 2 (do not output any analysis request) shall be performed to return control to the AR driver.
3.1.6.3  ANC3Ø

The pointer to the refined candidate list shall be saved in the contemporaneous analysis (CA) request message buffer (ANOCA). The emitter file number (CLEFN) shall be retrieved from the candidate list and saved in the CA request buffer. The identification code (EFID) shall be retrieved from the EF entry. This code shall then be compared to the identification code of each candidate in the list. If there is no match, processing shall continue at label ANC5Ø. If there is a match, the old EF entry id code is still valid. The return module code (ANRMC) in the CA request buffer (ANOCA) shall be set to X'Ø7', to indicate EOC Process 4 as the analysis return module. The Update Link Analysis 1 routine (ANUL1) shall then be called to determine if contemporaneous analysis is required. If required (return 2), the ANAW bit in the analysis request buffer shall be set (label ANC6Ø). If analysis is not required (return 1), the ANAW bit shall be cleared (label ANC7Ø). In either case, subroutine return No. 1 (output the analysis request) shall be performed to return control back to the AR driver, with the address of the analysis request buffer in the X-register.

3.1.6.4  ANC5Ø

The return module code (ANRMC) in the CA request buffer (ANOCA) shall be set to X'Ø8', to indicate EOC Process 3 as the analysis return module. The New Emitter Link Analysis 1 routine (ANEL1) shall then be called to determine if contemporaneous analysis is required. If required (return 2), control shall be sent to label ANC6Ø (described above). If not required (return 1) label ANC7Ø (also described above).

3.1.6.5  Subroutine Returns from ANOC2

Same as 3.1.4.2
3.1.6.6 Update Link Analysis 1 (ANUL1)

ANUL1 shall be a dummy routine in the priority 1 implementation. ANUL1 shall always perform return 1 to the calling routine. This return shall indicate that no contemporaneous analysis is required.

3.1.7 EOC Process 3 (ANOC3)

3.1.7.1 ANOC3

See 3.1.5

Same as ANNA3, except for the fact that the routine shall be called by the Analysis Return driver (ANDR), if the return module code of the AR data is $X'06$' (see Figure 1).

3.1.8 EOC Process 4 (ANOC4)

3.1.8.1 ANOC4

This routine shall be called by the Analysis Return driver (ANDR), if the return module code of the analysis return data (AR data) is $X'07$' (See Figure 1). The driver shall pass to ANOC4 the address of the AR data in the X-register and the EFN in the A-register. The EFN shall be passed in the A-register to the Update Link Analysis 2 (ANUL2) routine. ANUL2 shall determine if there is any platform linkage change. If there is change (return 2), processing shall continue at label ANK90. If no change is detected (return 1), the emitter file number shall be saved in the update message buffer (ANUPM). This message shall then be sent to the EXEC. Processing shall continue at ANK90.

3.1.8.2 ANK90

The return-to-AR-driver address on the stack shall be incremented so that ANOC4 shall always perform the "no analysis" return to the AR driver. Control shall then be returned to the AR driver.
3.1.8.3 Subroutine Returns from ANOC4

The returns from ANOC4 shall be:

1) Null. Never executed.
2) AR driver shall not output any Analysis Request message to the EXEC.

3.1.8.4 Update Link Analysis 2 (ANUL2)

ANUL2 shall receive the EFN in the A-register. SOGET shall immediately be called to convert EFN to an EF entry address. The function of ANUL2 shall be to determine if there has been any change in the platform linkage of the emitter. This function has not been implemented. The abbreviated priority 1 implementation shall merely set the platform link in the Emitter track file entry for this emitter to the emitter file number (EFN), i.e., no platform links. Control shall then be returned to the calling program.

3.1.8.4.1 Subroutine Returns from ANUL2 - The returns from ANUL2 shall be:

1) No platform linkage changed detected.
2) Platform linkage change detected.

In the abbreviated implementation, return 1 shall always be performed.

3.1.9 ANEC2 Emitter Classification 2

This is the principal subroutine for accomplishing the second level of emitter classification, namely: eliminating candidates from the list created by ECDR and its subroutines, on the basis of scan type (exact match) and scan period (between limits match). As such, it is largely a logical skeleton, most of the aforesaid task being accomplished by its dependent subroutines (described hereafter).
Upon entry X-register points to word 1 of a 3-word block:

```
   15   8  7   0
     - dc -    EFN
      CLAD
     15  12  9  0
       ST    dc   SPD
```

**EFN:** Emitter Track File # to which the candidate list applies.

**CLAD:** Candidate List Address

**ST, SPD:** Scan Type & Scan Period as obtained from a scan analysis request executed on behalf of Emitter Track File # EFN after ECDR was called for EFN and before the present call on ANEC2 for EFN.

The steps executed are as follows. (Note: Steps are keyed to program labels and unlabelled blocks preceded by a numbered comment line, e.g., "; 22").

**ANEC2** - Call ANST2 with X-register as described above. X is unchanged on return.

; 2 - Save A and B-registers on stack. Fetch 2nd word (CLAD) of input block store it in contemp. analysis request block (CRCLAD). Save a copy of CLAD in X-register.

; 3 - Call ANLV2. Most of the winnowing down is done here. If no candidates survive, return to call +1 (To increment return-to-driver address to call +2 and go to step "Done") else to call +2 (Next).

; 4 - X-register still contains CLAD. Fetch (CLAD) = EFN in right byte. Build byte-split word with EFN still in right byte and ANEC2's return module code (RMCEC2) in the left byte. Store result in contemp. analysis request block (CRRMCD).
Set A-register = X'8888'. This will be the contemp. analysis request word if the forthcoming call to ANEL1 indicates that analysis will be wanted. (Bit 15 is "Analysis Wanted" bit, Bit 3 indicates analysis type is Contemp.

Set B-register = X'8888'. This will be used upon analysis-not-wanted return from ANEL1 to wipe out the analysis-wanted bit in A-register.

Call ANEL1. If contemp. analysis is not wanted, return to call +1 (Step 6) else to call +2 (Step 7).

Contemp. Analysis not wanted: Use B-register to wipe out bit 15 of A-register.

Store request word now in A-register in the Contemp.-Analysis request block (CRREQW).

Set X-register = Address of 1st word of Contemp. Analysis request block = CRQ MSG.

Done - Entered from Step 3 (No candidates left) or Step 1. Restore B and A-registers from stack.

Return.

3.1.9.1 ANST2 - Subroutine of ANEC2 Scan Test 2

ANST2 tests the existing ETF scan type (ESTY) against certain standard types and under certain conditions alters both the ETF scan type (ESTY) and scan period (ESPD).

Upon entry X-register is exactly as described for the entry to ANEC2. The steps are as follows:
ANST2 - Save A, B and X-registers on stack.

; 1 - Fetch 1st word of input block = (X)) with EFN in right byte. Mask out left byte and call SOGET. This is a subroutine in Sorter Message Processing (Document No. 53959-GT-0755). That computes

\[ B-\text{Reg} \leftarrow ETF + 16 \times EFN \]

where EFN is in A-Reg.

; 2 - Fetch word containing scan type field ST in input block pttd to by X-reg, mask off extraneous fields of word and compare ST to sidelobe scantype code (SIDLOB). If not equal, go to step LKNMC else next.

; 3 - Move Addr ETF (EFN) = ETF + 16 \times EFN now in B-reg into X-reg and call ECSTC. This is a subroutine shared with ECST1 (Scan Test 1) in Emitter Classification 1 (Driver ECDR - Document No. 53959-GT-0760). Return is always to call +2.

DONE - This step is entered from Steps 3 (above), LKNMC (below), and 4 (below).

Restore X, B and A-registers.

Return.

LKNMC - (Look at Null - Measure Code)

This step is entered from Step 2 if ST ≠ sidelobe scan-type code.

Compare ST to Null-Measure Scan-type code. If equal, go to Step DONE, else next.

; 4 - Pick up 3rd word of input block (containing latest analysis return values for scan type & period) and use it to update the ETF word containing the same two items (ESTYD Rel to B-register).

Go to step DONE.
3.1.9.2  ANLV2 - Subroutine of ANEC2 Emitter Classification
Level 2 Search

This is the work horse subroutine of ANEC2. It's chief function
is to eliminate candidates from an existing candidate list on the basis of
scan type and period comparisons.

It is entered with X-Reg = Address or existing candidate list.
The process is carried out "in place" so that the resulting, reduced candidate
list is stored at the same address (which is also exit value of X-Reg).
If there are no candidates, special actions are taken; these occur in steps
NOCAND through 13.

ANLV2 -

Save A, B, E and X-Reg's on stack. The saved
X-Reg on the stack will be referred to as clad
(Lower case indicating contents of location CLAD
relating to S-Reg when the stack map is computed)

; 1    - A-Reg ← (X) and X ← X + 1
X-Reg which was pointing to the Cand. list header
word now points to the first Cand. list entry. Push
X-Reg twice to the stack for later use as rdpt
(Read point) and stpt (Store point).

; 2    - The Cand. list header word:

<table>
<thead>
<tr>
<th>NCAND</th>
<th>EFN</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>087</td>
</tr>
</tbody>
</table>

Was loaded into A-Reg in Step 1. Now separate the
bytes so that EFN ends up in A-Reg and NCAND
(Right justified) ends up in E-Reg.

Push E-Reg twice to the stack for later use as
ncand (Loop iteration control) and as nleft (Number
; 3  - Call SOGET. This is a subroutine in Sorter Message Processing (Document No. 53959-GT-0755) that computes

\[ B-\text{Reg} \leftarrow ETF + 16 \times EFN \]

where EFN is in A-Reg.

Push returned value of B-Reg to stack for (possible) later use as adef (step NOCAND).

Pick up ETF word containing both scan type (ESTY) and scan period (ESPD) - at ESTYD relative to B-Reg.

; 4  - The word just loaded into A-Reg is:

<table>
<thead>
<tr>
<th>ESTY</th>
<th>ESPD</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>12</td>
</tr>
<tr>
<td>9</td>
<td>0</td>
</tr>
</tbody>
</table>

Unpack so that ESPD is in A-Reg and ESTY (right justified) is in X-Reg (Mask: ESTYM; SHIFT: ESTYS). Push A-Reg to stack for later use as spud.

; 5  - Pick up and push to stack (ODA, ST + ESTY + 1) \( \equiv (B) + (X) + 1 \). This will later be used as higr = the largest group # (index into EL2) that has scan type ESTY.

Pick up and push to stack (ODA, ST + ESTY) \( \equiv (B) + (X) \). This will later be used as logr = the smallest group # that has scan type ESTY.

Winnow  - This step begins the major loop of the subroutine. It is entered once from above (Step 5) and N-1 times from step tally where N = original value stored in ncand in Step 2. All steps from the present, down to and including step tally are within the loop.

Fetch to A-Reg the next cand. list entry = (rdpt).

\[ \text{rdpt} \leftarrow \text{rdpt + 1} \]
Make a copy of A-Reg in X-Reg for use in step keep.

Mask out the ident field in A-Reg so that A = group # of current Cand. List entry.

Compare this group # to logr (see step 5).

If group # < logr go to step CANCEL - else next

; 6 - Compare group # to higr

If group # ≥ higr go to step CANCEL - else next

; 7 - Compute in B-Reg the address of the file in EL2 whose index is group # = EL2 + 11 * (Group # - 1)

(Done by Call E2ADR)

Double load to A and E-Reg's from MXSND relative to B-Reg. This puts the maximum scan period (MXSN) in A-Reg and the minimum scan period (MNSN) in E-Reg. Both fields occupy BITS Ø - 9 and require masking.

; 8 - Mask out extraneous fields leaving A-Reg = MXSN (Mask: ESPDM).

Compare MXSN to spud (see step 4).

If spud > MXSN go to step CANCEL - else next

; 9 - Mask out extraneous fields leaving E-Reg = MNSN.

Compare MNSN to spud (loaded into A-Reg.)

If spud ≥ MNSN go to step keep - else next
Cancel - This step is entered from steps

Winnow - Group # < logr
6 - Group # > higr
8 - spud > MXSN
9 - spud < MNSN

Do: nleft ← nleft - 1

If nleft ≠ ∅ go to step TALLY
else go to step NOCAND

Keep - This step is entered from step 9 under the conditions

logr ≤ Group # < higr - AND-
MNSN ≤ spud ≤ MXSN

Do: A ← X-Reg copy of Cand. List entry made in step winnow.

Store Cand. List entry at (stpt)

stpt ← stpt + 1

Tally - This step is entered from steps

CANCEL - nleft ≠ ∅

KEEP - Unconditionally

Do: ncand ← ncand - 1

If ncand ≠ ∅ go to step winnow - else next

; 10 - (Out-of-Loop here to End)

Store nleft in left byte of Cand. List header word at address (clad)

; 11 - Bump return address to call +2:

rtd ← rtd + 1
Abnorm - Entered from Step 11 (Returning to Call +2)
Or from Step 13 (Returning to Call +1)

Clean up stack, i.e., return to available status
the 8 locations on stack that were appropriated for
temporary local storage in Steps 1 - 5 inclusive.

S-Reg ← S-Reg + 8

Restore X, E, B and A-Reg's from stack

Return

NOCAND - This step is entered from step CANCEL when
nleft = 0, i.e., the Candidate List has been
entirely eliminated.

B-Reg ← adef = ETF + 16 * EFN (Step 3)

Replace ETF identity field (EIDD relative to B-Reg)
by code NOFA2 (Using mask EIDM).

; 12 - Replace ETF display code field (EDISD Rel B) by
code UNKNO (Using mask EDISM)

; 13 - Clear left byte of Cand. List header word (ADDR
(clad)).

Go to step ABNORM

3.1.9.2.1 E2ADR - Subroutine of ANLV2
Compute EL2 Address

The subroutine is also called by TRNSL8 in Emitter Classification 1
(Document No. 53959-GT-0760) and by ANAMB in Analysis Return Sub-
routine ANEC3.

On input the right byte of A-Reg contains an Index (Ground Number/
Emitter Library No.). On output the left byte is cleared and

B-Reg = EL2 + 11 * (Index -1)
E2ADR - Mask out A-Reg left byte

; 1 - Copy A to B

; 2 - Multiply B-Reg by 11. This is done by a sequence of double - B's (B ← B + B) and add's (B ← B + A) that is much faster than an MPY instruction.

; 3 - Add the constant EL2 - 11 to B

Return

3.1.9.3 ANEL1 - Subroutine of ANEC2
New Emitter Link Analysis - 1

This version is a dummy. Its one and only step is a return to Call +1 indicating: No Contemp. analysis wanted.

Any non-dummy version must preserve and restore the A and B-Reg's.

3.1.10 ANEC3
ANEC3 shall be entered with X-Reg pointing to a two word block such that the right byte of word 1 contains the emitter track file # (EFN) and word 2 = the Candidate List Address (CLAD). This value of X-Reg shall remain as the entry value of X-Reg for each of the subroutines called by ANEC3.

ANEC3 shall consist only of the following steps:

- Call New Emitter Link Analysis #2 (ANEL2)
- Call Family Associateion (ANFAM)
- Call Ambiguity Resolution (ANAMB)
- Bump return address to call + 2 to cause a No-Analysis return to Analysis Return Driver.

- Return
3.1.10.1 ANEL2 - Subroutine of ANEC2
New Emitter Link Analysis #2

ANEL2 shall be entered with X-Reg set as upon entry to ANEC3.

The present version of ANEL2 shall be a dummy in that it shall only cause the platform link field of ETF (EFN) to point to itself, i.e., contain the value EFN. The steps shall be as follows:

ANEL2 - Save A and B-Reg's on stack

; 1 - A ← (X) =
    \{ Junk in left byte
    \{ EFN in right byte

Call SOGET:
\{ B ← ETF + 16 * EFN
\{ A ← EFN, left byte cleared.

Save EFN, now in A-Reg, on stack

; 2 - Fetch ETF word containing platform link field.
This shall be at displacement EPLKD relative to B-Reg. This shall put in A-Reg a byte split word whose left byte is to be retained and whose right byte is to be replaced by EFN.

Mask out A-Reg. right byte and OR in EFN from top-of-stack (S ← S + 1).

Store result at location from which fetch was made at beginning of this step.

; 3 - Restore B and A-Reg's from stack

Return

3.1.10.2 ANFAM - Subroutine of ANEC 3
Family Association

The present version of ANFAM shall be a dummy. Its one step shall be an Exit instruction.
3.1.10.3 ANAMB - Subroutine of ANEC3
Ambiguity Resolution

Ambiguity resolution shall be entered with X-Reg set as upon entry to ANEC3. ANAMB shall reduce the designated candidate list to a single entry (The winner, hereafter) by selecting the candidate whose weighting factor is highest, breaking ties, if any, in favor of the lower-numbered candidate.

ANAMB shall set various ETF fields with information taken from the winner's EL2 file as detailed below.

ANAMB shall send a classification-concluded (update) message to the Executive.

The steps followed by ANAMB shall be the following:

ANAMB - Save A, B, E, X-Reg's on stack
; 1 - Set X = Candidate List Address
; 2 - Set A = Candidate List Header word
     \[= (X)\] and \[X \leftarrow X + 1\]

The header word shall consist of NCAND (Candidate List Length) in the left byte and EFN in the right byte.

The bytes shall be separated by calling an internal subroutine Bumpak so that

\[A \leftarrow\text{EFN}\]
\[E \leftarrow\text{NCAND}\]

EFN shall now be stored in the third word of the update message (UPEFN).
NCAND shall be pushed to stack and referred to hence by S-relative instructions with symbolic displacement NCAND. In the following text, we refer to contents of said location as ncand.

; 3 - 

External subroutine SOGET (in Sorter Message Processing - document 53959-GT-0755) shall now be called using A-Reg = EFN as input to do.

\[ B \leftarrow ETF + 16 \times EFN \]

This value shall be pushed to stack and value thus stored shall be referred to by S-relative instructions with symbolic displacement ADEF; contents referred to in following text as adef.

; 4 - 

E-Reg shall be set = -1 as initial value of highest weight to be carried throughout forthcoming loop in said register

Room shall be made on stack for a temporary location for storing the Candidate List entry of candidates as successive maximal weighting factors are discovered in the following loop. (S-Relative symbolic displacement: Winner, contents: winner).

AMLOOP -

This step shall be entered 1st time from step 4 and N-1 thereafter from step tally, where N = original value of ncand as set in step 2.

This step shall do:

Pick up next Candidate List item = \((X)\); \(X \leftarrow X + 1\)

This shall load A-Reg with Candidate identity code (left byte) and group \# (right byte).

Call External Subroutine ELADR (part of ANEC2) which shall mask out A-Reg left byte and set

\[ B \leftarrow EL2 + 11 \times (\text{Group } \# - 1) \]
; 5  - Fetch Candidate's weighting factor word which shall be at displacement MFCTD relative to B and mask out extraneous fields.

; 6  - Candidate weight shall be compared to current maximum (in E-Reg):

Cand. Wt ≤ Current Max: Go to step tally else next

; 7  - Replace current max by Candidate Weight

Fetch Candidate List entry, which shall be found at -1 relative to X and store it as current winner.

Tally - This step shall be entered from either

Step 6 - Candidate weight ≤ Current maximum
Step 7 - Candidate has become new winner.

This step shall test for loop completion by doing:

ncand←ncand - 1

If ncand still > 0 go to step AMLOOP else next

; 8  - Loop is now complete. E-contains maximum Candidate weight (not used) and the winning candidate's Cand. List entry is stored as winner.

This step shall do:

X ← adef = ETF + 16 · EFN

A ← winner = Byte-split word (Ident, Group #)

Internal byte unpacking subroutine BUNPAK shall be called to do:

A ← Group #
E ← Ident
; 9 - Winner Group # shall be saved in B freeing A for use in this and next three steps.

This step shall insert winner Group # in ETF ELN field:

    \( A \leftarrow (ELND + X) \)
    \( A \leftarrow A \wedge \text{Mask}; \) Clear ELN field 
    \( A \leftarrow A \lor B; \) Inserts Group # 
    \( (ELND + X) \leftarrow A \)

; 10 - This step shall insert winner ident in ETF ident field:

    \( A \leftarrow (EIDD + X) \)
    \( A \leftarrow A \wedge \text{Mask}; \) Clears EID field 
    \( A \leftarrow A \lor E; \) Inserts Ident 
    \( (EIDD + X) \leftarrow A \)

; 11 - Winner ident (8-Bit field) shall now be compared to 16 to see whether its particular value will fit into the 4-Bit ETF Display code field.

If no: Clear E-Reg

; 12 - This step shall store the 4 least significant bits of E-Reg in ETF Display code field:

    \( A \leftarrow (EDISD + X) \)
    \( A \leftarrow A \wedge \text{Mask}; \) Clears EDIS field 
    \( E \leftarrow \text{Left Shift} (E); \) Appropriate # bits to align 
    \( A \leftarrow A \lor E \)
    \( (EDISD + X) \leftarrow A \)

; 13 - This step shall put Winner Group # (saved in B) back into A as input to 

Call E2ADR (See Step AMLOOP) so that 

    \( B \leftarrow EL2 + 11 \times (\text{Group} \# - 1) \)
13 -continued- The balance of this step shall use B as just set to fetch and isolate Winner's EL2 platform code and test it against standard "Naval" code.

= : E←1 in ETF ENAV Bit position

≠ : E←∅

14 - This step shall set or reset ETF ENAV bit depending on result of Step 13.

A←(ENAVD + X)
A←A ∨ E
(ENAVD + X)←A

15 - This step shall send an update message to the Executive:

X←Address UPMSG
Call EXMES

16 - This step shall clear up stack by

S←S + 3

Restore X, E, B, A-Reg's from stack

Return
3.2 SUBPROGRAM FLOW DIAGRAMS

The logic flow for all routine comprising this subprogram is shown in the following flow diagrams. The flow diagrams are labeled so as to correspond to paragraph 3.1. That is, flow diagram 3.2.9 is described in paragraph 3.1.9. Data extraction points for instrumentation are shown as comment blocks with the text "DP__".
START

GET RETURN
MODULE CODE
FROM ANAL RET. MSG

1 ≤ I ≤ 9?

Y

ADD I TO
TABLE ADDR

GET ANYFN
FROM ANAL RET. MSG
(INSERTED A-REG)

X ← X + 2
(X NOW PTS TO
ANYFN WORD
OF ANAL RET MSG)

N

EXEC PASSES MTR TO
ANAL RETURN MSG IN
X-REG

AND80

SEND ERRONEOUS
ANAL RET. MSG
TO INSTRUMENTATION

A

B
CALL ITH ANAL RETURN PROCESSING ROUTINE

ANAL REQUIRED Y

CALL EXEC TO OUTPUT ANAL REQUEST MSG

AND99

RETURN TO EXEC

AND99

X-REG PTS TO ANAMC OF ANAL RET MSG

A-REG CONTAINS VALUE IF ANEFN

IF ANAL REQUEST IS TO BE OUTPUT, X-REG WILL CONTAIN PTR TO ANAL REQ MSG

3.2.1 (concluded)

ANALYSIS RETURN DRIVER
TLC 00 SEP 76
SHT 8 OF 2
3.2.2

NE PROC 2

(ANNE2)

9/6/76

DP1: ANAL RTN

START

GET ETF ADDR(EFN)
(SOGET)

PERFORM PRI TST2
(ANPT2)

PERFORM PW TEST
(ANPW3)

ANNE26

LP

RETURN

Y

EFNWN(EFP) ← 0

THIS WILL BE A TIMED-OUT RETURN WHEN
LONG PULSE PROC IS IMPLEMENTED

N

ANNE22

PERFORM HAEM PRI TEST 1
(ANHP1)

ANAL REQ RTN

Y

FORMAT ANAL REQ
WITH AN=1 CAT=1

AW WILL BE ONE
WHEN CONTIP ANAL
IS IMPLEMENTED

N

FORMAT ANAL REQ
WITH AN=0 CAT=1

DP2: ETF ENTRY

Y

RETURN

N

DP2: ETF ENTRY

ANAL RTN

MSG

A-REG: Emitter File Number, EFN

B-REG: ADDR OF ETF ENTRY, EFP

X-REG: PTR TO ANAL RET MSG

1) X-REG: PTR TO ANAL REQ MSG

2) ANALYSIS RETURN ALWAYS EXECUTED

DUMMY SUBROUTINE

DUMMY SUBROUTINE
3.2.2.1

PRI TEST 2
(ANPTZ)
9/6/76

\[ B \text{- REG: ADDR OF ETF} \]
ENTRY, EFP
3.2.2.2

PW TEST

RETURN
START

INCREMENT RTN ADDR BY ONE

RETURN

\[ b_{REG}: \text{ADDR of ETF ENTRY, ECP} \]

\[ \text{ANAL NOT WANTED RETURN} \]

3.2.2.3

HARMONIC PR 7 TEST 1

(AN HP1)

916/76 $f$
START

GET ETF ADDR(EFR) (SOGET)

PERFORM HARMF RTZ (ANHPZ)

PERFORM FREQ TST (ANFQT)

OUTPUT CLASS MSG

INCREMENT RETURN ADDRESS FOR NO-ANALYSIS RETURN

RETURN

X-REG: PTR TO ANAL RETURN MESSAGE
A-REG: EMMITTER FILE NUMBER, EFN

B-REG: ADDR OF ETF ENTRY, EFP

Dummy SUBROUTINE

DP1: ANAL RTN MSG

DP2: CONTENTS OF ETF ENTRY

DP3: CLASS MSG

3.2.3
NE PROC 3
(ANN 3)
9/6/76 - S
3.2.3.1

HARMONIC PRI TEST 2
(AN HP2)

9/6/17 6 병
START

PARAM ← EFREQ(EFP)

IS EFREQ(EFP) ALL 1's

EFU(EFP) ← 0

M ← 15

QUAL ← EFQF(EFP)

PERFORM QUAL TST (SOQOUT)

EFU(EFP) ← GDQ

RETURN

B-REG: ADDR OF ETF ENTRY EFP

3.2.3.2

FREQUENCY TEST

(After)

4/6/76 JG
A.R. DRIVER PASSES PTR TO A.R. DATA IN X-REG AND EFN IN A-REG

START

CONVERT EFN INTO EF ENTRY ADDR (SOGET)

GET ANSTY FROM AR DATA

ANSTY = SDBE?

ANSTY = NULL MEAS.?

GET EFSIND

= φ²?

GET EFSIND ← 1

A

INCREMENT RTN ADDR FOR NO-ANALYSIS RETURN

D

A

ANSTY = EFSTYP?

B

IS X = ANSPR EFSIRD

GET ABS VALUE OF X

IS |X| ≤ ΔSPRD

3.0.4

NOFA 2

PROCESS 2

TLC 27 AUG 76

SHT 1 OF 3
SET EFSTYP = ANSTY

SET EFSRDP = ARNP.

GET EFN FROM A.R.
DATA AND SAVE IN ANAL. REQ. MSG BUFF.

PERFORM LEVEL 1 SEARCH (ECLV1)

NO CANDIDATES RETURN?

Y

LEVEL 1 SEARCH RETURNS PTR TO CAND. LIST IN X-REG

NO CANDIDATES RETURN?

Y

PERFORM LEVEL 2 SEARCH (ANLY2)

N

SAVE PTR TO CAND. LIST IN AR MSG BUFFER

3.2.4 (continued)

NOFA 2
PROCESS 2

TLC 27 Aug 76

SHT 2 OF 3
3.2.4 (concluded)

NOTA 2

PROCESS 2

TLC 30 44 76
5 3 2 3 1 2 3

1. Output "ANAL REQ"
2. RETURN TO AR DRIVER

IF No. 2 X HAS PTR TO ANAL REQ
2. OUTPUT NO ANAL MISC
MSC BUFFER

NOTA PROCESSOR HAS
2. RETURNS TO AR
AIR DRIVER PASSES PPR TO A.R. DATA IN X-REG, AND EFN IN A-REG.

PERFORM NE LINK ANALYSIS 2 (ANEL2)

PERFORM FAMILY ASSOC. (ANFAM) (dummy)

PERFORM AMBIG UITY RESOL. (ANAMB)

INCREMENT RETURN ADDR ANAL (REG MSG)

5-B ROUTINE RETURN

2 RETURNS:
- NULL, NEVER EXECUTED
- NO ANALYSIS

3.2.5 and 3.2.7

NOFAL2 PROCESS 3

OR

EOC PROCESS 3

TLU 07FUG76
3.2.6 (concluded)
ANUL

START

SUBROUTINE
RETURN

DUMMY ROUTINE IN THE PRIORITY 4 SOFTWARE
2 RETURNS:
1) NO CONTEMP ANAL - ALWAYS EXECUTE;
2) CONTEMP ANAL REQUEST - NULL

3.2.6.6

UPDATE LINK ANALYSIS 1
TLC 19 OCT 76
START

PERFORM UPDATE LINK ANALYSIS 2 (ANUL2)

A.R. DRIVER PASSES PTR TO A.R. DATA IN X-REG. AND EFN IN A-REG.

NO PLATFORM LINK CHANGE RETURN?

ANK5Φ

SAVE EFN IN UPDATE MSG BUFFER

CALL EXEC TO OUTPUT UPDATE MESSAGE

ANK5Φ

INCREMENT RETURN ADDRESS FOR AN-ANALYSIS

SUBROUTINE RETURN

ADDITIONAL NON-PRIORITY 1 PROCESSING:

1. PERFORM FAMILY ASSOCIATION FOR ONE MEMBER OF EACH GROUP WITH PLATFORM LINK CHANGE
2. PERFORM AMBIGUITY RESOLUTION FOR EACH GROUP

2 RETURNS:
- NULL, NEVER EXECUTED
- NO ANALYSIS

EXEC PROCESS 4

3.8.8
TLC 30 Aug 76
A-REG WILL CONTAIN EFN

CONV EFN INTO EF ENTRY ADDR (SOGET)

SET EFPLINK(EIFN) ← EFN

CALLING SEQUENCE FOR THIS ROUT.
LDDA (EFN)
JSUB (= ANUL2)
JUMP (NO PLAT LINK CHNG)
    (PLAT LINK CHNG)
IN PRIORITY 1, "NO PLAT LINK CHNG" PATH IS ALWAYS TAKEN.

3.2.8.1
UPDATE LINK ANALYSIS 2
TLC 30 AUG 76
LSTZ2

SAVE A, B & X-REG'S ON STACK

31

AREG <- EFN

SOGET  
B <- AREG  
ETF(EFN)

32

FETCH ETF, SCAN TYPE, ST, FROM INPUT BLOCK

IST1 IS SIDE-LOGE ?

YES

X <- B-REG

ECSTFC

DONE

NO

EST1 IS NULL-HEADER ?

YES

STORE Scan TREE (HEADER OF Input BLOCK IN ETF

RETURN

X <- 3-NORD Block (See ANEC2 Flow Chart)  
3rd word within Box (2 AND 3) BELOW
Winnow

\[ A \leftarrow (rdpt) \]
\[ rdpt \leftarrow rdpt + 1 \]
\[ X \leftarrow A \]

Mask off ID
\[ A \leftarrow \text{GRN} \]

\[ \text{GEN} \leftarrow \text{LOGR} \]

\[ \text{IF} \quad \text{GEN} \leq \text{LOGR} \]
\[ \text{IF} \quad \text{GEN} > \text{LOGR} \]

\[ \text{IF} \quad \text{GRN} \text{within the group # limits that pertain to scan time?} \]
\[ \text{NO} \quad \text{YES} \]

\[ \text{CANCEL} \]

\[ n_{\text{left}} \leftarrow n_{\text{left}} - 1 \]

\[ \text{IF} \quad n_{\text{left}} = 0 \]
\[ \text{NO} \quad \text{YES} \]

\[ \text{TALLY} \]

\[ \text{KEEP} \]

\[ \text{FETCH AGR} \]
\[ \text{ADDR} \leftarrow E2 + 11 \cdot (\text{GRN} - 1) \]

\[ \text{IF} \quad \text{MXSN and HNSN to } (A, B) \]
\[ \text{IF} \quad \text{SPUD vs MXSN} \]
\[ \text{IF} \quad \text{SPUD vs HNSN} \]

\[ \text{IF} \quad \text{SCAN period ESPD (SPUD) within limits for group # GRN?} \]
\[ \text{NO} \quad \text{YES} \]
NOCAND

B-reg-adef = ETF + 16 · EFN

STORE NODF2
CODE IN ETF
IDENT-EID
EZN < c0

"NONE OF ABOVE"

STORE UNKNO
CODE IN ETF
DISPLAY CODE
HEADER-EPIS

STORE c0 IN
LEFT BYTE
OF (clad)

ABNORM

RETURN TO CALL +1
E2APR

$A\text{-REG} = \text{-DC- INDEX}$

CLEAR
A-REG
LEFT BYTE

$\beta \leftarrow A$

$\beta \leftarrow \beta + 11 \times \beta$

$\beta \leftarrow \beta + EL2-11$

RETURN

$A\text{-REG} = \text{INDEX}$

$B\text{-REG} = \text{EL2+11(INDEX-1)}$
3.2.10 ANEC3

ANEC3

-X-REG

ANEL2

ANFAM

ANAMB

BUMP RETURN ADDR TO CALL+2

RETURN

EFN: Emitter Track File #
CLAD: Candidate List Addr.
SAVE A & B REG'S ON STACK

A ← (JUNK, EFN)

X-REG → JUNK EFN

B ← ETF + 16 * EFN

A ← EFN (LEFT BYTE CLEARED)

SAVE EFN ON STACK

A ← ETF
PLATFORM LINK WORD: EPLKD
RUN TO B

MASK OUT RIGHT BYTE

OR IN EFN FROM STACK 
S ← S + 1

STORE AS NEW ETF PLATFORM LINK WORD: EPLKD REL TO B

RETURN
S -> rtad

S -> xsav

X -> CLAD
CAND, LIST
ADDR

A -> CAND, LIST
Header Word
X -> X+1

Bunker
A -> EFN
E -> NCAND

Store EFN in
UPDAT MSG.
NCAND+NCAND
on Stack

S -> ncand

S -> adef

E <- -1

Stack Map

<table>
<thead>
<tr>
<th>Disp</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td>6</td>
</tr>
<tr>
<td>7</td>
</tr>
</tbody>
</table>

Initial Value of
High Weighting Factor

S -> winner

S -> AMLOOP

3.2.10.3

11/01/78
3.3 COMPUTER SUBPROGRAM ENVIRONMENT

3.3.1 Tables

3.3.1.1 Analysis Return Driver Table

Analysis Return Processing Table (ANMPT)

Purpose and Type -

Fixed length table containing the addresses of the subroutines called to process an Analysis Return message.

Size and Indexing Procedure -

Nine entries of one 16-Bit word. All entries shall be referenced by indexed displacement from the start of the table.

Entry Format -

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
<th>Units</th>
<th>LSB</th>
</tr>
</thead>
<tbody>
<tr>
<td>Routine Addr</td>
<td>Address of an analysis return message processing routine</td>
<td>N/A</td>
<td>N/A</td>
</tr>
</tbody>
</table>
3.3.1.2 NE Processing 2 Tables
None.

3.3.1.3 NE Processing 3 Tables
None.

3.3.1.4 NOFA2 Processing 2 Tables
None.

3.3.1.5 NOFA2 Processing 3 Tables
None.

3.3.1.6 EOC Processing 2 Tables
None.

3.3.1.7 EOC Processing 3 Tables
None.

3.3.1.8 EOC Processing 4 Tables
None.

3.3.1.9 Emitter Classification 2 Tables

3.3.1.9.1 Contemporaneous Analysis Request Message

a) The name of this table is CRQMSG. It is local to ANEC2.

b) CRQMSG is used to hold two fixed constants and three variable words filled in and by ANEC2, the totality constituting a message to the Executive stating that contemp. analysis is or is not wanted. The location of the message is made known to ANEC2's caller by returning the address (CRQMSG) of its 1st word in the X-Reg.

c) CRQMSG is of fixed length = 5. It is indexed by use of individual labels attached to the locations requiring access.

d) CRQMSG's structure and Bit layout is shown in the accompanying diagram.
3.3.1.91 Contemporaneous Analysis Request Message
3.3.1.9.2 Scan Type Outer Directory to EL2

a) The name of this table shall be ODA.ST

b) The purpose of ODA.ST is based on the requirement that the 11-word files of EL2 be sorted on ascending scan type. Then each element of ODA.ST, say the $i$-th, $i = 0, 1, \ldots, 15$ contains the lowest file number in EL2 that exhibits scan type = $i$. The 17th entry ($i = 16$) of ODA.ST contains the number $N + 1$ where $N$ = the number of files of EL2.

Thus, in ANLV2 to eliminate candidates on the basis of exact match to the current scan type of the subject ETF file (ESTY) we look up:

\[
\text{logr} = (\text{ODA.ST} + \text{ESTY}) \text{ and } \text{higr} = (\text{ODA.ST} + \text{ESTY} + 1)
\]

and ask if the candidate group # is such that $\text{logr} \leq \text{group #} < \text{higr}$ (yes-keep; no-cancel)

c) ODA.ST shall be of length = 17 based on allocation of 4 bits to scan type. It shall be indexed by anding an index $I = 0, 1, \ldots, 16$, to address ODA.ST.

d) The structure of ODA.ST and its relation to EL2 are shown in the accompanying diagram. Bit layout is not applicable since each element is a whole word item.
3.3.1.7.3 ODA ST Structure & Relation to EL 2
3.3.1.10 Emitter Classification 3 Tables

3.3.1.10.1 Update Message

a) The name of this table shall be UPMSG. It shall be local to subroutine ANAMB.

b) The purpose of UPMSG shall be to inform the executive that classification has been completed on the emitter whose track file # was input to ANAMB, so that the Executive may take and/or schedule those actions which properly emanate from said event.

c) UPMSG shall be three words long and shall be indexed by use of labels attached to those entries which require access.

d) Structure and Bit layout shall be as shown:

<table>
<thead>
<tr>
<th>Label</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>UPMSG:</td>
<td>EMN EC3</td>
</tr>
<tr>
<td></td>
<td>Executive Message # of words to follow</td>
</tr>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td>UPEFN:</td>
<td>EFN</td>
</tr>
<tr>
<td></td>
<td>Stored by ANAMB</td>
</tr>
</tbody>
</table>

3.3.2 Variables

3.3.2.1 Analysis Return Driver Variables

None.

3.3.2.2 New Emitter Processing 2 Variables

ANNE2 variable are defined in Table I.

3.3.2.3 New Emitter Processing 3 Variables

ANNE3 variables are defined in Table II.
# TABLE I

VARIABLE DESCRIPTIONS FOR NEPROC2

<table>
<thead>
<tr>
<th>Descriptive Item</th>
<th>Variable Name</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>EFN</td>
</tr>
<tr>
<td>Purpose</td>
<td>Value of ETF entry</td>
</tr>
<tr>
<td>Type</td>
<td>Fixed point</td>
</tr>
<tr>
<td>Size</td>
<td>8</td>
</tr>
<tr>
<td>Binary Pt.</td>
<td>Bit 0</td>
</tr>
<tr>
<td>Max. Value</td>
<td>127</td>
</tr>
<tr>
<td>Min. Value</td>
<td>-128</td>
</tr>
<tr>
<td>Initial Value</td>
<td>Don't care</td>
</tr>
<tr>
<td>Static/Dynamic</td>
<td>Dynamic</td>
</tr>
</tbody>
</table>

* Memory map assignment will restrict this.
### TABLE II

**VARIABLE DESCRIPTIONS FOR NE PROC 3**

<table>
<thead>
<tr>
<th>Descriptive Item</th>
<th>EFN</th>
<th>EFP</th>
<th>GDQ</th>
<th>M</th>
<th>PARAM</th>
<th>QUAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Purpose</td>
<td>Value of ETF entry</td>
<td>Provides address of 1st word of ETF entry given by EFN</td>
<td>Indicator of data quality</td>
<td>One less than number of significant bits in PARAM.</td>
<td>Parameter which is to be tested for quality with PARAM.</td>
<td>Quality factor associated with PARAM.</td>
</tr>
<tr>
<td>Type</td>
<td>Fixed point</td>
<td>Fixed point</td>
<td>Fixed point</td>
<td>Fixed point</td>
<td>Fixed point</td>
<td>Fixed point</td>
</tr>
<tr>
<td>Size</td>
<td>8</td>
<td>16</td>
<td>1</td>
<td>4</td>
<td>16</td>
<td>4</td>
</tr>
<tr>
<td>Binary Point</td>
<td>Bit 0</td>
<td>Bit 0</td>
<td>N/A</td>
<td>Bit 0</td>
<td>Bit 0</td>
<td>Bit 0</td>
</tr>
<tr>
<td>Max. Value</td>
<td>127</td>
<td>65,536*</td>
<td>1=good quality</td>
<td>15</td>
<td>65,536</td>
<td>15</td>
</tr>
<tr>
<td>Min. Value</td>
<td>-123</td>
<td>0*</td>
<td>0=bad quality</td>
<td>3</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Initial Value</td>
<td>Don't care</td>
<td>Don't care</td>
<td>Don't care</td>
<td>Don't care</td>
<td>Don't care</td>
<td>Don't care</td>
</tr>
<tr>
<td>Static/Dynamic</td>
<td>Dynamic</td>
<td>Dynamic</td>
<td>Dynamic</td>
<td>Dynamic</td>
<td>Dynamic</td>
<td>Dynamic</td>
</tr>
</tbody>
</table>

* Memory map assignment will restrict this.
3.3.2.4 NOFA Process 2 Variables

1) NOFA2 Process 2 Contemporaneous Analysis Request Message (ANNCA)

ANNCA2 may generate a contemporaneous analysis request message (which will then be sent to the EXEC by the AR driver). This message has the format of an Analysis Request Message (see Figure 2), with

- ANNW = 3
- ANRMC = X'94'
- ANCA = 1
- ANAW = ∅ or 1

2) ΔSPRD

ΔSPRD is used by ANNA2 to determine if the scan analysis scan period measurement differs significantly from the scan period stored in the emitter track file.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
<th>Units</th>
<th>LSB</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANDSP</td>
<td>Delta SPRD</td>
<td>Msec</td>
<td>1/4</td>
</tr>
</tbody>
</table>

3.3.2.5 NOFA2 Process 3 Variables

None.
3.3.2.6 EOC Process 2 Variables

3.3.2.6.1 Update Message (ANUPM) - ANOC2 may generate an update message and send it to the EXEC. Format is shown in Figure 4.

3.3.2.6.2 Contemporaneous Analysis Request Message (ANOCA) - ANOC2 may generate a contemporaneous analysis request message. This message has the format of an Analysis Request Message (see Figure 2) with:

\[
\begin{align*}
\text{ANNW} & = 3 \\
\text{ANRMC} & = X'\emptyset'6' \text{ or } X'\emptyset'7'
\end{align*}
\]

\[
\begin{align*}
\text{ANCA} & = 1 \\
\text{ANAW} & = \emptyset \text{ or } 1
\end{align*}
\]

3.3.2.7 EOC Process 3 Variables
None.

3.3.2.8 EOC Process 4 Update Message (ANUPM)
ANOC4 may generate an update message and send it to the EXEC. Format is shown in Figure 4.

3.3.2.9 Emitter Classification 2 Variables
Only subroutine ANLV2 has any local variables, i.e., entities stored and retrieved from memory. These variables are all maintained on the stack during ANLV2's execution and their space is relinquished before exiting.

A stack map is shown in the accompanying figure. It, and the text to follow employ the following convention:

A symbolic displacement (for use in $S$-indexed access instructions) is shown in upper-case. The contents of such a location are denoted by the same symbol written in lower-case.
3.3.2.9.1 Variables of Subroutine ANLV2 - The following items are described in top-of-stack to bottom order.

\[ \text{logr} - \text{Low Group Number} = (\text{ODA.ST} + \text{ESTY}) = \text{smallest group \# that can exhibit scan type = ESTY.} \]

\[ \text{higr} - \text{High Group Number} = (\text{ODA.ST} + \text{ESTY} + 1) = \text{one more than largest group \# that can exhibit scan type = ESTY.} \]

\[ \text{spud} - \text{Temporary repository for ETF scan period (ESPD).} \]

\[ \text{adef} - \text{Temporary repository for ETF (EFN) address. Needed only if all candidates are eliminate and failure codes must be stored back in ETF (EFN).} \]

\[ \text{nleft} - \text{This variable and the next are both originally set = the number of candidates in the input candidate list (left byte of 1st word thereof).} \]

\[ \text{nleft is decremented by 1 each time a candidate is eliminated and immediately thereafter the new value of nleft is tested for equality to } 0. \text{ If equality is attained, the candidate list has been entirely eliminated and the algorithm ends: See steps NOCAND - 13 in 3.1.9.2.} \]

\[ \text{If the algorithm ends with nleft} > 0, \text{ then nleft is size of the reduced (or possibly same size) candidate list. This value is stored in the left byte of the word whence nleft was initialized.} \]

\[ \text{ncand} - \text{Is initialized as described under nleft.} \]

\[ \text{ncand is decremented by 1 after the consideration of each candidate whether kept or cancelled. When ncand goes to 0, the algorithm ends, unless ended prematurely as described under nleft.} \]

\[ \text{sft} - \text{This variable and the next, are both originally set = ADDR of 1st Cand. List entry (2nd word of list, immediately after header word).} \]
continued-

- stpt is incremented by 1 after each store of a kept (retained) candidate back into the list. Note that at all times clad + 1 ≤ stpt ≤ rdpt, where clad and rdpt are described below.

- rdpt - Is initialized as described under stpt.

  rdpt is incremented by 1 for each candidate list entry fetched for consideration. This is done after rdpt has been used as an indirect address to fetch the Cand. List entry = (rdpt).

- clad - Pointer to the header word of the input Cand. List as received upon entry in the X-Reg.

\[
\begin{align*}
\text{esav} & \quad \text{Save input contents of E-Reg} \\
\text{bsav} & \quad \text{Save input contents of B-Reg} \\
\text{asav} & \quad \text{Save input contents of A-Reg}
\end{align*}
\]

Not referenced by name.

- rtad - Return address. Accessed by name on a normal return to call +2:

\[
\text{rtad} \leftarrow \text{rtad} + 1
\]
3.3.2.10 Emitter Classification 3 Variables

No permanent space shall be allocated to non-tabular data. Three temporary local variables shall be used by subroutine ANAMB. Space for them shall be allocated on the stack during initialization and relinquished prior to Exit. The stack map shall be as shown on the first page of the ANAMB flow chart, 3.3.2.10.3.

The variables shown there are:

**Winner** - During execution of the loop which searches for the maximum weighting factor over the input set of candidates, winner shall be set = the Candidate List entry word of the each candidate whose weighting factor exceeds the maximum factor found up to that point. Note that the maximum weight shall be initialized = -1 guaranteeing that the first Candidate List entry, at least, will be stored at winner.

**adef** - Shall be used to hold for later use the emitter track file base address for the EFN-TM file:

\[
ETF (EFN) = ETF + 16 \cdot EFN
\]

**ncand** - Shall be initialized with the Candidate List length as extracted from the left byte of the header word thereof.

Thereafter, ncand shall be used as a loop iteration control; ncand = ncand - 1 and repeat loop if ncand ≠ 0.

3.3.3 Constants

There are no local constants associated with the Analysis Return Functional Group.

3.3.4 Flags

There are no local flags associated with the Analysis Return Functional Group.
3.3.5 Indices

The Emitter File Number (EFN), is an index that is used throughout the Analysis Return Functional Group. It is used to access an entry in the Emitter Track File (EF). EFN assumes the following range of values:

\[ 0 \leq EFN \leq 127 \]

3.3.5.1 Analysis Return Driver Indices

Analysis Return Message Processing Table Index:

a) Index Name. I (Not a symbolic label)

b) Purpose. This index is used to fetch an Analysis Return message processing routine address from table ANMPT. "I" assumes the following range of values:

\[ 1 \leq I \leq 9 \]

3.3.6 Common Data Base References

3.3.6.1 Analysis Return Driver (ANDR) Common Data Base References

None.

3.3.6.2 New Emitter Processing 2 (ANNE2) Common Data Base References

1) Emitter Track File (EF)

3.3.6.3 New Emitter Processing 3 (ANNE3) Common Data Base References

1) Emitter Track File (EF)

3.3.6.4 NOFA2 Process 2 (ANNA2) Common Data Base References

1) Emitter Track File (EF)

3.3.6.5 NOFA2 Process 2 (ANNA3) Common Data Base References

None.

3.3.6.6 EOC Process 2 (ANOC2) Common Data Base References

1) Candidate List (CL)

2) Emitter Track File (EF)
### 3.3.6.7 EOC Process 3 (ANOC3) Common Data Base References

None.

### 3.3.6.8 EOC Process 4 (ANOC4) Common Data Base References

None.

### 3.3.6.8.1 Update Link Analysis 2 Common Data Base References

1) Emitter Track File (EF)

### 3.3.6.9 Emitter Classification 2 Common Data Base References

<table>
<thead>
<tr>
<th>Item</th>
<th>ANEC2</th>
<th>ANST2</th>
<th>ANLV2</th>
<th>ANEL1</th>
</tr>
</thead>
<tbody>
<tr>
<td>ETF</td>
<td>EDIS</td>
<td></td>
<td>S</td>
<td>S</td>
</tr>
<tr>
<td></td>
<td>EID</td>
<td></td>
<td>S</td>
<td>U</td>
</tr>
<tr>
<td></td>
<td>ESPD</td>
<td></td>
<td></td>
<td>U</td>
</tr>
<tr>
<td></td>
<td>ESTY</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EL2</td>
<td>MXSN</td>
<td></td>
<td>U</td>
<td>U</td>
</tr>
<tr>
<td></td>
<td>MNSN</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Parameters:
- ESDLB
- NUL
- ENA2
- EUNK

✓, U: Used
S: Set

### 3.3.6.10 Emitter Classification 3 Common Data Base References

<table>
<thead>
<tr>
<th>Item</th>
<th>ANEC3</th>
<th>ANEL2</th>
<th>ANFAM</th>
<th>ANAMAB</th>
</tr>
</thead>
<tbody>
<tr>
<td>ETF</td>
<td>EPLK</td>
<td></td>
<td>S</td>
<td>S</td>
</tr>
<tr>
<td></td>
<td>EDIS</td>
<td></td>
<td></td>
<td>S</td>
</tr>
<tr>
<td></td>
<td>EID</td>
<td></td>
<td></td>
<td>S</td>
</tr>
<tr>
<td></td>
<td>ELN</td>
<td></td>
<td></td>
<td>S</td>
</tr>
<tr>
<td></td>
<td>ENAV</td>
<td></td>
<td></td>
<td>S</td>
</tr>
<tr>
<td>EL2</td>
<td>MFCT</td>
<td></td>
<td>U</td>
<td>U</td>
</tr>
<tr>
<td></td>
<td>MPLT</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Parameters
- NAVAL

✓
3.4 INPUT/OUTPUT

The format of all input and output messages shall be as specified below:

<table>
<thead>
<tr>
<th>Item</th>
<th>Input or Output</th>
<th>Specification Document</th>
</tr>
</thead>
<tbody>
<tr>
<td>Analysis Return Message</td>
<td>Input</td>
<td>CDBDD, 53959-GT-0751</td>
</tr>
<tr>
<td>Analysis Request Message</td>
<td>Output</td>
<td>&quot;</td>
</tr>
<tr>
<td>Classification Message</td>
<td>Output</td>
<td>&quot;</td>
</tr>
<tr>
<td>Instrumentation Data</td>
<td>Output</td>
<td>Data Extraction CSDD, 53959-GT-0759</td>
</tr>
</tbody>
</table>

3.4.1 Input/Output Formats for ANEC2

Inputs: X-Reg

```
  1  | EFN
  2  | CLAD
  3  | ST
  4  | SPD
  5  | NCAND

Input Candidate List

<table>
<thead>
<tr>
<th>NCAND</th>
<th>EFN</th>
<th>GRN_1</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID_1</td>
<td>GRN_1</td>
<td></td>
</tr>
<tr>
<td>ID_2</td>
<td>GRN_2</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>

Outputs: X-Reg

```

```
  1  | Executive Message No.  
    | # Words to Follow = 3  
    | Return Module Code | 7 | EFN |

Reduced Candidate List

<table>
<thead>
<tr>
<th>NLEFT</th>
<th>EFN</th>
<th>GRN_1</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID_1</td>
<td>GRN_1</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>
```
3.4.1.1 ANST2 Input/Output

Inputs: X-Reg exactly as input to ANEC2.

Outputs: Possible changes to
- ETF Scan Type & Period
- ETF Scan Type & Period
- ETF State Indicator
  (in SCTCOM if called)

3.4.1.2 ANLV2 Input/Output

Inputs: X-Reg→Candidate List as illustrated in inputs for ANEC2.

Outputs: X-Reg (Unchanged) points to reduced Candidate List illustrated in Section 3.4.1.
3.5 REQUIRED EXTERNAL SUBROUTINES

3.5.1 SOGET

SOGET is a subroutine in Sorter Message Processing (Document No. 53959-GT-0755) called by ANST2, ANLV2, ANNA2, ANEC2, and others.

Input: EFN whole word item in A-Reg.

Output: ETF + 16 * EFN in B-Reg.

3.5.2 SCTCOM

SCTCOM (Scan Test Common Logic) is a subroutine in Emitter Classification Processing -1 (Document No. 53959-GT-0760) which, as implied by its title, is shared with ECST1 (Scan Test 1).

SCTCOM requires the address ETF + 16 * EFN to be in X-Reg on entry. It complements the ETF State Indicator (ESIN) and if now on (= 1), sets the ETF Scan Type (ESTY) to circular and the ETF Scan Period (ESPd) to Time-Out.

3.6 CONDITIONS FOR INITIALIZATION

This subprogram shall have unconditional entry and shall require no special initialization procedure.
3.7 SUBPROGRAM LIMITATIONS

The Analysis Return Functional Group shall make the following assumptions and be subject to the following limitations:

1. ANDR retrieves the Return Module Code from the Analysis Return message and verifies that it is a valid code (= 1, 2, 3, ..., 8, or 9). If not valid, an error alert message shall be sent to Instrumentation.

2. Emitter Classification 2 Algorithm Limitations - The algorithms in this subprogram are programmatic sequels to those of Emitter Classification Processing -1 (Document No. 53959-GT-0760), and are part of a single, overall search-and-classification strategy. Hence, the limitations on the algorithms stated in the referenced document carry over to here.

The one local limitation that does stand out is that the length of ODA.ST is imposed by the allocation of 4-bits to scan type. If more than 16 types should be required in the future, ODA.ST would have to be lengthened.

Note also, that the method of matching scan type using ODA.ST imposes a design requirement on EL2, namely

EL2 files must be sorted on ascending scan type. (see Section 3.3.1.9.2).
3.8 INTERFACE DESCRIPTION

The Analysis Return Driver (ANDR) shall be called by the EXEC. ANDR shall then call one of the Analysis Return processing routines (ANNE2, ANNE3, ANNA2, etc.). The routines called by each Analysis Return processing routines are shown in the following interface diagrams. Instrumentation shall be called as required for data extraction and is not shown on the diagrams. Calls to the Executive message function are also not shown.
A.R. DRIVER

NOFA2 PROC 2 (ANNA2)

- CONV EFN TO EF ENTRY ADDR. (SOCET)
- LEVEL 1 SEARCH (ECLV1)
- LEVEL 2 SEARCH (ANLVD)
- NE LINK ANALYSIS 1 (ANEL1)
  (DUMMY)

A.R. DRIVER

NOFA2 PROC 3 (ANNA3)

- NE LINK FINAL 2 (ANEL2)
  (DUMMY)
- FAMILY ASSOC. (ANELA)
  (DUMMY)
- AMBIGUITY RESOL. (ANAMB)

ALSO: EOC PROC 3 (ANOC3)

INTERFACE DESCRIPTION:
- NOFA2 PROC 2
- NOFA2 PROC 3
- EOC PROC 3

T.A. 3 SEP 76
EMITTED CLASSIFICATION

WHO CALLS WHOM / STACK DEPTHS

ANEC2
(3)

ANSL2
(4)

ANLV2
(13)

ANEL11
(1)

SCTCOM
(1)

SOGET
(?)

E2APR
(1)

Dummy

Used also by:
TRANS2 in ECI
ANAME in ANEC3

Shared with ECSTI in ECI

In System Message Processing

(N) = Stack Depth Increase (Maximum) caused by call to routine. This includes the return address, but not additions caused by further calls.

Maximum Stack Depth x 17 (at least 1 for call on SOGET)

Reached when ANEC2 -> ANLV2 -> SOGET.

SYSTEM DESCRIPTION
EXIT CLASS 2
Emitter Classification Processing - 3

Who Calls Whom & Stack Depths

ANEC 3
(1)

ANEL 2
(4)

ANFM
(1)

ANAME
(9)

SOGET
(?)

E2ADR
(1)

AXNES
(?)

INTERFACE DESCRIPTION

EXIT CLASS 3
<table>
<thead>
<tr>
<th>Word</th>
<th>ANMNO</th>
<th>ANNW</th>
<th>ANRMN</th>
<th>ANEFN</th>
<th>ANPTR</th>
<th>ANSTY</th>
<th>NOT USED</th>
<th>ANSPR</th>
<th>NOT USED</th>
<th>NOT USED</th>
</tr>
</thead>
</table>

Figure 1a. (Scan) Analysis Return Message Format
<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
<th>Units</th>
<th>LSB</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANMNO</td>
<td>Executive Message No. (= 4)</td>
<td>N/A</td>
<td>1</td>
</tr>
<tr>
<td>ANNW</td>
<td>No. of Words in Message (= 3)</td>
<td>N/A</td>
<td>1</td>
</tr>
<tr>
<td>ANRMC</td>
<td>Return Module Code</td>
<td>N/A</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>NEPROC2 = 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>NEPROC3 = 2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>NOFA2 PROC2 = 3</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>NOFA2 PROC3 = 4</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>EOC PROC2 = 5</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>EOC PROC3 = 6</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>EOC PROC4 = 7</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>EM CLASS 2 = 8</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>EM CLASS 3 = 9</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ANEFN</td>
<td>Emitter File No. (0 ≤ ANEFN ≤ 127)</td>
<td>N/A</td>
<td>1</td>
</tr>
<tr>
<td>ANPTR</td>
<td>Pointer to Candidate List</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>ANSTY</td>
<td>Scan Type of Emitter (see CDBDD for codes)</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>ANSPR</td>
<td>Scan Period of Emitter</td>
<td>msec</td>
<td>1/4</td>
</tr>
</tbody>
</table>

Figure 1b. (Scan) Analysis Return Message Format
<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
<th>Units</th>
<th>LSB</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANMNO</td>
<td>Executive Message No. (= 1)</td>
<td>N/A</td>
<td>1</td>
</tr>
<tr>
<td>ANNW</td>
<td>No. of Words in Message (= 3)</td>
<td>N/A</td>
<td>1</td>
</tr>
<tr>
<td>ANRMC</td>
<td>Return Module Code</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>ANEFN</td>
<td>Emitter File No. (0 ≤ ANEFN ≤ 127)</td>
<td>N/A</td>
<td>1</td>
</tr>
<tr>
<td>ANPTR</td>
<td>Pointer to Candidate List</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>ANAW</td>
<td>Analysis Wanted Code</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>ANDI</td>
<td>Deinterleaving Analysis Request</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>ANCA</td>
<td>Contemporaneous Analysis Request</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>ANPA</td>
<td>PRI Analysis Request</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>ANFA</td>
<td>Frequency Analysis Request</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>ANSA</td>
<td>Scan Analysis Request</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>AND1</td>
<td>Not Used in Priority 1 Software</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AND 'N'</td>
<td>Not Used in Priority 1 Software</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 2b. Analysis Request Message Format
<table>
<thead>
<tr>
<th>Word 0</th>
<th>MNO</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>NW</td>
</tr>
<tr>
<td>2</td>
<td>NOT USED</td>
</tr>
<tr>
<td>3</td>
<td>NOT USED</td>
</tr>
<tr>
<td>4</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>NOT USED</td>
</tr>
</tbody>
</table>

**Figure 3a. Classification Message Format**
<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
<th>Units</th>
<th>LSB</th>
</tr>
</thead>
<tbody>
<tr>
<td>MNO</td>
<td>Executive Message No. (= 9)</td>
<td>N/A</td>
<td>1</td>
</tr>
<tr>
<td>NW</td>
<td>No. of Words in Message (= 1)</td>
<td>N/A</td>
<td>1</td>
</tr>
<tr>
<td>EFN</td>
<td>Emitter File No. ((0 \leq \text{EFN} \leq 127))</td>
<td>N/A</td>
<td>1</td>
</tr>
</tbody>
</table>

Figure 3b. Classification Message Format
<table>
<thead>
<tr>
<th>Word</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ANMNO</td>
<td>ANNW</td>
<td>D</td>
<td>NOT USED</td>
<td>ANEFN</td>
<td>NOT USED</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 4a. Update Message Format
<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
<th>Units</th>
<th>LSB</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANMNO</td>
<td>Executive Message No. (= 7)</td>
<td>N/A</td>
<td>1</td>
</tr>
<tr>
<td>ANNW</td>
<td>No. of Words in Message (= 1)</td>
<td>N/A</td>
<td>1</td>
</tr>
<tr>
<td>ANEFN</td>
<td>Emitter File No. (0 ≤ AEFN ≤ 127)</td>
<td>N/A</td>
<td>1</td>
</tr>
<tr>
<td>D</td>
<td>Deletion Flag</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td>1 = Emitter AEFN has been made inactive</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Ø = Normal Update Message</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 4b. Update Message Format