CT
ACD
32094
5

COPY

DOT/FAA/CT-ACD32094/5

Configuration Management and Software Quality Assurance Standards for TCAS Logic Systems

J. Stuart Searight

SEP 1

September 16, 1994



Engineering, Research, and Development Service
Concepts Analysis Division
FAA Technical Center
Atlantic City International Airport, N.J. 08405

EXECUTIVE SUMMARY

This document describes the Software Configuration Management (SCM) and Quality Assurance (QA) standards and procedures used for all software maintained by Traffic Alert and Collision Avoidance System (TCAS) Logic Systems personnel at the FAA Technical Center. These standards and procedures will ensure that integrity is maintained throughout the life-cycle of all TCAS software, and provide a well defined course of action to implement changes to the baseline software.

1.0 PURPOSE AND SCOPE OF THIS DOCUMENT

This document describes the Software Configuration Management (SCM) and Quality Assurance (QA) standards and procedures to be used for all software maintained by Traffic Alert and Collision Avoidance System (TCAS) Logic Systems personnel at the FAA Technical Center. It details the steps required to make a change to the baseline software. These steps include the procedures for documenting the need or desire for a change to the baseline software; evaluating the request for change; assigning of the duty of making a change; changing of the baseline software; testing of the software change for accuracy, completeness, and proper integration with the existing baseline software; and admitting the changed software to the baseline.

2.0 BACKGROUND

The FAA Technical Center TCAS Logic Systems personnel have developed software tools to assist in evaluation of proposed/revised TCAS II logic version releases, and recorded flight data from the TCAS Transition Program (TTP). This library of baseline software is currently stored on a Local Area Network (LAN) file server that connects all TCAS personal computers (PCs) on a Novell Netware 3.21 platform. The source code is currently in Borland Turbo Pascal 7.0, which is currently run in the MS DOS 5.0/Windows 3.1 environment. For a detailed description of the Turbo Pascal environment, refer to section 5.1 of this report.

To ensure baseline integrity and effectively monitor software development efforts, SCM and QA standards and procedures must be adhered to. This document describes those standards in a formal and detailed manner.

2.1 SOFTWARE DESCRIPTION

There are two main software systems and support programs in the TCAS baseline. The first is the FAA Technical Center's Fast Time Encounter Generator (FTEG), with its support routines and post-processing programs. This software simulates TCAS units operating in predefined aircraft encounter geometries. It can implement a variety of TCAS Logic versions and can simulate thousands of aircraft encounter geometries in a short period of time. The second system is TCAS Data Reduction and Analysis (DRNA). This software analyzes data recorded from both the FTEG and actual TCAS units. There are currently five manufacturers of TCAS I and TCAS II units. The data from TCAS units is called "live" data. The DRNA processing of live data forms the basis of the FAA Technical Center's evaluation of TCAS performance which is part of the TCAS Transition Program (TTP). While some software is common to all live data, much is specific to the various manufacture's recording formats and cycle timing.

The baseline is categorized into groups in two different ways. The first is a physical grouping into two categories labeled PROGS and UNITS. These labels correspond to two directories on the LAN file server which contain the entire software baseline of the Logic Systems project. PROGS contains all source code files that contain Pascal programs. All support routines are found within the UNITS directory. Each is mapped to a logical drive on the file server. While everyone has read permission to these directories, only the Configuration Manager (CM) and Technical Program Manager (TPM) have write permissions.

The second grouping is into logical groups, with the baseline broken down into a variety of subsystems. These are more detailed divisions within the FTEG and DRNA partitions. These groupings include TCAS logic versions, FTEG and other drivers for the logic, FTEG post-processing, encounter summary printouts, TTP processing, recorded encounter data processing, recorded track data processing, and common support routines shared by various subsystems. These subsystems have been assigned among the FAA staff. The FAA employee is considered the Technical Lead (TL) of his or her assigned subsystems.

Familiarity of source code, code usage, and output evaluation were considerations in the current TL subsystem assignments. The TL will play a vital role in software development and maintenance.

The TLs and the TPM meet weekly as a configuration management control board (CMCB). In these meetings submitted Problem/Trouble Reports (PTRs), assigned work, and completed baseline updates are all discussed. Project short- and mid-range priorities and plans are developed as the different individual subsystem priorities and issues are merged together. The support contractors join the CMCB every other week after the CMCB meeting to provide status updates on assigned work, and to exchange feedback on work loads, priorities, etc..

3.0 CONFIGURATION MANAGEMENT

This section defines the procedures necessary to make an update to the baseline. It covers the entire process including problem identification, design, coding, and testing.

3.1 PROBLEM DETECTION and CHANGE REQUESTS

Any TCAS personnel can submit a Problem/Trouble Report form to the Configuration Manager at any time. (This form is shown in Page 1 of Appendix A.) These forms serve as either Change Requests or Problem Reports. Change Requests are usually used to suggest improvements to the software systems in areas such as user interface, output formats, documentation, and performance efficiency. Problem Reports deal with errors such as faulty output, incorrect internal processing, inadequate algorithms, system crashes, and incorrect error handling. When a Problem Report is submitted, the data set that caused the problem to be noticed must be cited.

PTRs should be written with respect to the requirements of the system that contains the respective problem, error, or deficiency. If a requirements specification document exists for the system, the PTR should reference the requirement being met. If the system does not have a requirements specification written for it, the author of the PTR should identify the quantifiable and testable requirement(s) of the system that are not being met and write the PTR in reference to that requirement.

The Configuration Manager will examine the PTR for completeness and accuracy. Once the PTR is accepted, it will be numbered, and entered into the PTR database. The TL of the identified subsystem will then evaluate the PTR's priority and, at an appropriate time, assign the PTR to an FAA staff member or a support contractor. This person will be referred to as the programmer throughout the software development process. The TL will coordinate with the assigned programmers regarding their work performed to resolve the PTRs assigned from the TL's particular subsystems.

3.2 UPDATING and UPGRADING OF BASELINE SOFTWARE

This section defines the processes necessary, and the individuals responsible for addressing identified deficiencies in the baseline, updating applicable source code, and readmitting that code back into the baseline.

3.2.1 PROCESS OF MAKING UPDATES TO BASELINE

This section lists the steps to create or modify code as dictated by one or more PTRs.

1) The programmer will study the assigned PTR and all applicable code.

2) The programmer will formulate design and implementation considerations for the work needed and write documentation stating these design and implementation plans. These documents will be reviewed by the Technical Lead. The process does not continue until the following are agreed upon by both

programmer and TL: design and implementation documentation is developed that specifies a design that will meet all requirements of the assigned PTR; all documentation that will need revision to reflect the software update are identified; the level of testing to be performed after coding is complete is specified.

3) The programmer will sign out all files from the Configuration Manager that will undergo changes to resolve the PTR. The SIGN OUT list is a Word 2.0 file found on the NEW logical LAN drive. All Logic Systems team members, including contractors. have read/write access to this file. The process of signing out files involves updating this file with the following information: adding the programmer's name, the file(s) being signed out, the date and size of the baseline version of the file(s), the date signed out, and the PTR being addressed. A file should only be signed out by one person at one time to prevent simultaneous updates to the baseline.

4) The programmer will copy the code from the LAN's baseline directories to the proper directory local to the programmer's PC. Each TCAS Logic Systems PC has directories that mirror the baseline, and standard compile directories that correspond to this directory structure. (These are specified in section 5.1.)

5) The programmer will edit or create the code to meet the requirements specified in the PTR and agreed upon in step 2. This includes adhering to all documentation standards described in section 4.

6) The programmer will test the new code to satisfy the TL and as outlined in section 3.3.

7) Any deficiencies found through the testing phase shall be corrected and the code retested until no deficiencies exist.

8) Upon successful completion of the testing procedures, the programmer will complete a CHANGE FORM (CF) that thoroughly describes the change(s) that have been made to new or baseline software (see Page 2 Appendix A).

9) The programmer will copy the unit(s) and/or program(s) into the LAN's NEW directory to await readmittance to the baseline. NEW's subdirectories mirror the baseline directories (PROGS and UNITS). All TCAS personnel have write permission to the NEW subdirectories. The extension of the unit's file is to be the initials of the programmer responsible for the changes. This safeguards against duplicate updates not prevented by the SIGN-OUT sheet.

10) The Technical Lead will review the programmer's submitted code, documentation and test results.

The set of submitted files will include updated or new source code, output files from performed testing, and output files from the file compare DOS utility of updated files vs. the baseline files. The submitted source code will be copied into the appropriate NEW subdirectory (UNITS or PROGS) with the extension being the programmer's initials. The test output files will retain their original extensions and be copied into the TESTING subdirectory of NEW. File compare outputs will have an extension of 'FC', and be copied into the same subdirectory as the corresponding source code.

Documentation will include completed CF(s), updated reference documents, when applicable, and a written summary of performed testing. This summary will included what files were tested, a description of each test's purpose, and its result. The TL will examine all submitted materials and reference the original PTR and the design and implementation documents previously agreed to by the TL and programmer.

11) The TL can request the programmer, or perform themselves, any extra testing deemed necessary.

12) If the TL is satisfied with the update, the CF is filed with the Configuration Manager.

13) The Configuration Manager will review the submitted CF(s) and examine completed review process with TL.

14) The Configuration Manager will check all updated files for CM integrity. This includes checking that all previous updates are still in submitted file, date of file matches date of most recent comments, etc..

15) The Configuration Manager will initiate or perform any extra testing deemed necessary.

16) If all previous steps have been satisfied, the updated or new code is written into the baseline with a PAS extension.

3.2.2 CREATING NEW CODE FOR THE BASELINE

If a new program or support unit has been created by TCAS personnel, it must be put into the proper NEW subdirectory on the LAN. First, a PTR must be written and accepted stating the need for the new code, and describing the new code's requirements. The new code is to be accompanied by a CHANGE FORM that is sent to the Technical Lead. This CHANGE FORM should describe the new code's purpose, its output, and any interface it has with existing baseline software. The new code must go through the review and testing procedures defined in section 3.3 of this document. Upon successful testing it will await admittance to the baseline.

3.3 INDIVIDUAL'S RESPONSIBILITIES

The above sections described the process of identifying deficiencies within the baseline, and updating source code to bring the issue to closure. This section also addresses these activities, but from a different point of view. The sections below are divided by individuals, and their responsibilities in seeing a PTR brought to closure. There are four sections. One each for the programmer, Technical Lead, Configuration Manager, and Technical Program Manager. For all software updates there are three positions whose roles must be performed: Programmer - completes design, actual updates to baseline code, and testing; Technical Lead - reviews design, code changes and testing; Configuration Manager - accepts the change and updates baseline.

It should also be noted that the Technical Leads can function as programmer, the Configuration Manager is also a Technical Lead, and the TPM can function as any of the other three roles. No one person is to act in two different capacities for any one piece of work that leads to an update to the baseline. If, for example, a TL assigns himself/herself as programmer to close out a PTR, they must first solicit another person to act as TL for the work being done. When a change is being done for a subsystem of which the CM is TL, the TPM must act as CM for those efforts.

3.3.1 PROGRAMMER

1) Develops thorough design and implementation documentation addressing the assigned problem.
2) Signs out source code after design is approved.
3) Reviews progress and coordinate with TL during and after each step of software development process.
4) Thoroughly tests work before submitting it to TL for final inspection.

3.3.2 TECHNICAL LEAD

1) Review new PTRs for the following:
    a) is a duplicate, or similar PTR already in the database?
    b) level of urgency
    c) magnitude of the change
    d) impact on other subsystems

e) changes to recorded record structures
f) changes needed to support documentation

2) Assign PTR to a programmer.
3) Review programmer's initial design:
a) Will the design fix the identified problem?
b) Will the design have any effects on other subsystems?
c) Will the design change the format of any recorded record structures?
d) Does the design conform to defined software standards?
4) Review the programmer's implementation:
a) Are all changes commented in source code?
b) Examine all changes (use file compare utility).
c) The cited problem is fixed. (New code appears to address problem.)
5) Review programmer's testing coverage and results.
a) Was all necessary testing completed?
b) Were there test cases that addressed all concerns?
c) Are results as expected?
d) Are there any side-effects to other subsystems? Is this acceptable with the TL(s) of those subsystems?
e) Has all needed documents been updated?
6) Review Change Form(s) submitted with updated code.
a) Are all required fields filled in?
b) Does the CF clearly describe the change(s) made?

3.3.3 CONFIGURATION MANAGER

1) Perform initial review of all submitted PTRs.
a) Are all required fields filled in?
b) Is proper subsystem identified?
c) Is priority appropriate for scope of problem cited?
d) Is problem clearly stated?
2) Number accepted PTR.
3) Forward PTR to appropriate TL.
4) Review submitted code, testing, and documentation after TL has accepted update.
a) Does date at top of file reflect last change?
b) Does file contain all previously accepted changes?
c) Are comment blocks properly updated?
d) Are updated lines of code properly commented?
e) Does CF reference correct PTR?
f) Was level of programmer's testing and TL's review appropriate for the change's magnitude.
g) Were all necessary updates to documents completed?
6) Enter CF into database.
7) Update PTR database to have cited PTR reference CF.
8) Update baseline with accepted source code and document changes.
9) Update SIGN OUT list to reflect files being signed back in.

3.3.4 TECHNICAL PROGRAM MANAGER

1) Monitor overall software development and testing processes.
2) Evaluate if the defined processes are:
a) being adhered to
b) effective means of ensuring baseline code reliability
3) Act as CM or TL when needed. (If a TL has acted as programmer, the TPM might act as TL. If the CM has acted as programmer or TL, the TPM must fill role of CM.)

3.4 BASELINE SUPPORT DOCUMENTATION

This section describes the support documentation for the baseline programs and major subsystems. It also defines the manner in which these documents fall under the configuration management rules and procedures defined in sections 3.2 and 3.3.

3.4.1 DOCUMENTATION DESCRIPTION

All documents will be stored as part of the TCAS Baseline in a directory named DOC. (This is a subdirectory of TCAS just as UNITS and PROGS are.) There will be user documents for all baseline programs, or groups of similar programs. (For the rest of section 3.6, 'programs' will be equal to the phrase 'programs, or groups of similar programs'.) There will be output definition files for all programs that define all fields of the generated output file. There will be design documents for all major subsystems of the TCAS baseline. The three sections below describe these documents in greater detail.

3.4.1.1 USER DOCUMENTS

A user document will be written for all programs. These documents will be named the same as the program it describes, or that part of the programs' name that is common to all programs. (i.e. SUM for a document describing the use of BX_SUM, HW_SUM, and RC_SUM.) The extension will be 'USE'. These documents will describe all user/program interface, available options, default options, and define needed input files - any restrictions on their location (drives, subdirectories), and their format - and the available destinations of output.

3.4.1.2 OUTPUT DEFINITION DOCUMENTS

A definition document will be written for all output files that can be generated by baseline programs. These documents will be named the same as the program's name, or major subsystem from which the file is generated. (i.e. SUMMARY, LISTER, ERD_CNTS) The extension will be the same as the output file which the document defines. These documents will describe each field in the output file, its origins, expected range, units, and display format if text, or storage format if binary.

3.4.1.3 SUBSYSTEM DESIGN DOCUMENTS

A design document will be written for all major subsystems of the baseline. These documents will be named to describe the subsystem as best possible. (i.e. SUMMARY, PRM for pilot response model) The extension will be 'DSN'. These documents will describe subsystem designs and implementations. This will include interfaces to programs and other subsystems, data structures filled in or processed, outputs generated, and algorithms implemented.

3.4.2 CONFIGURATION MANAGEMENT OF DOCUMENTS

This section describes the manner in which the documents described in section 3.6.1 fall under the TCAS CM practices and standards. As with the UNITS and PROGS directories, the DOC directory is a subdirectory of TCAS, and only the CM and TPM have write permission to that directory. There is also a DOC subdirectory of NEW that all programmers have write permission to. It is this directory that updates are written into, and are promoted to the baseline directory by the CM or TPM.

The originator of a PTR, and the assigned programmer should always be examining if the work they are doing dictates that any document(s) will need updating as a result of the pending baseline update. It is the TL's responsibility to ensure that all submitted updates include updates to the corresponding document(s).

This is included in the TL's responsibilities defined in section 3.3.2. The Configuration Manager also must examine if any submitted updates require updates to support documentation (section 3.3.3).

3.5 SOFTWARE TESTING STANDARDS and PROCEDURES

This section describes the routines followed when developing and testing new or updated software. There are two philosophies utilized to assist in the development of correct, robust, and reliable software. These philosophies are used during design and development to assist the programmer, and after the code has been written or updated to test the programmer's work. Preventive measures, used during the design review, are taken during software development. These practices are in place to help ensure that work requirements are thoroughly defined and solutions are logically designed. Corrective measures, used during implementation verification, are taken after code is updated or written. Testing is done in an effort to find faults in design or implementation. These procedures are defined below.

3.5.1 DESIGN REVIEW

1) After a programmer has been assigned a PTR and investigated the problem, he/she will develop a design document as a proposal to bring the PTR to closure. This document will be reviewed by the subsystem Technical Lead.

2) After the design has been agreed upon, the assigned programmer will submit documentation defining proposed implementation plans to the subsystem Technical Lead.

3) Actual coding will not begin until both documents have been reviewed and approved by the Technical Lead.

3.5.2 IMPLEMENTATION VERIFICATION

1) A code inspection is performed. The author of the new or changed code will inspect the code for logical accuracy, clarity, and adherence to format, naming and documentation conventions of the TCAS Project. The Technical Lead will also inspect the completed code after it has been submitted by the programmer.

2) The code is to be compiled with the baseline units on the LAN. This will be in the form of a BUILD to assure the new units compile with all other units and programs in the baseline.

3) The code is to be tested thoroughly to show that the program execution - A) Reflects the changes the PTR requested. This is done by processing the Problem Data Set(s) data cited in the PTR and showing that the faulty output, system crash, or improper error handling has been corrected.
B) Does not cause other changes or side-effects that are not desired. This is done by running all program drivers that implement the unit(s) that have been created or modified against Standard Data Sets (SDS) that were selected by the programmer BEFORE starting work on the source code. The programmer will run all applicable programs against the SDSs before code is changed and compare the results with the output from the updated code.

All test results must be kept for the Technical Lead to inspect. Documentation discussing all testing performed, the purpose of each test, and the results must also be presented to the TL. The Technical Lead also will review the updated code, and the submitted Change Form. If everything is found to be acceptable, the CF is forwarded to the Configuration Manager for his/her final inspection.

As coding changes are made, the TL should ask for, or locate sufficient test cases for the change being made.

3.6 ADMITTANCE TO THE BASELINE

Upon receiving a CHANGE FORM from a programmer, along with the respective test results, the Technical Lead will then inspect the test results and do any additional testing he/she feels is necessary. The Configuration Manager, after his/her own review, will then copy the new file(s) back into the baseline with the proper .PAS extension, or hand it back to the programmer with any problems found in the code. (For more detailed description of the review process, refer to section 3.2 of this document.) If the file(s) are put back into the baseline, they will be removed from the SIGN OUT sheet by the Configuration Manager. The Configuration Manager will then cross-reference the PTR that precipitated the change to the CHANGE FORM. This ensures the traceability of how PTRs were closed out, and why changes to the baseline were made.

3.7 DELETING CODE FROM BASELINE

PTRs can be submitted requesting routines be deleted from the baseline if routines or entire units are identified as dead code, or if it is to be replaced by new code that will perform the same tasks in an improved manner. The PTR must justify the reasons for the routine's or file's deletion. The programmer must also assure the Technical Lead that this deletion will not have any undesired effects on the rest of the baseline. The PTR's closure must go through the same procedures and reviews as other PTRs. (For more detailed description of those procedures, refer to section 3.2 of this document.)

3.8 BASELINE ARCHIVES

The Configuration Manager will archive the entire source code baseline once a month. Archives will also be saved immediately before any major data collection efforts performed by the Logic Systems team. The monthly archive will include all intermediate versions of files that have been updated more than once in the past month. This ensures that any previous state of the baseline can be recreated. The data collection archives will include the entire baseline's source code, the executable code used for collecting the data, all input files used, and all output files generated.

3.9 OFF-SITE CONTRACTORS

For reasons beyond the Logic Systems' control, there exists one major problem with practicing precise configuration management. Due to a shortage of desk space at the Technical Center, the support contractors are forced to have office space off-site. In addition, the Technical Center currently does not have the capability to allow off-site contractors full LAN connectivity. This has forced Logic Systems to have the off-site contractors use a copy of the baseline for their software development. While this admittedly is not conducive to proper configuration management control, measures have been implemented to make the best of the limitations described above. This section will describe the off-site contractor's use of the baseline copy, and define procedures implemented to help ensure that this copy is kept up to date with the baseline maintained at the Technical Center.

3.9.1 OFF-SITE BASELINE

The off-site contractors have a LAN local to their office. They have set up identical directory structures used at the Technical Center. This structure has the same read/write permissions as well. There is a baseline set of directories that are read-permission only and a NEW directory structure that everyone has full read and write permission for. One contractor has been appointed CM for the contractors. He and the contractor's Program Lead (PL) are the only people with write permission to the baseline directories.

3.9.2 OFF-SITE BASELINE UPDATES

While the contractors contact the TLs directly during software development, all communication and file transfers when submitting completed work must go through the contractor's CM. The contractor's CM should examine the work submitted in the same manner that the FAA CM would before admitting updates to the baseline. The updated files, CFs, test results, and any updated documents are then passed from the contractor's CM to the appropriate TL and CM at the Technical Center. There they will go through the formal review process defined in section 3.2. The contractor's baseline is not to be updated with the new files until the FAA CM informs the contractor CM that the update has been accepted into the baseline.

When updates to the baseline are made from work done at the Technical Center, the FAA CM will send a Baseline Update Package (BUP) to the contractor's CM. This BUP will include the following: all updated programs and/or units archived (using the PKWARE utility PKZIP) into files named UNITS.ZIP and/or PROGS.ZIP; a copy of all CFs that document the update, any updated documents archived into a file named DOCS.ZIP; the baseline directories' listings captured into files UNITS.DIR and PROGS.DIR; and a current copy of the SIGN OUT list. The files are archived because the current file transfer software used (CC:Mail) changes the date of files attached to the electronic correspondence. Using PKZIP preserves the original date of the files. The directory listings are sent so that the contractors can compare the two baselines to assure that all files found in the two baselines are the same, and have the same date and file size. (The contractor has developed a utility that automates this process.)

The FAA and contractor CMs will exchange directory listings weekly (Monday mornings). They will both use the directory listing comparison utility to check the other's baseline version against their own. This will ensure that the two baselines have not diverged despite the procedures defined in this section.

4.0 QUALITY ASSURANCE

To ensure Software Quality Assurance, a set of standards has been established that all TCAS Logic System personnel are required to follow. These standards define conventions documentation, file and data labeling, and units of measurement.

4.1 SOFTWARE STANDARDS

4.1.1 DOCUMENTATION

Each unit should contain a comment section identifying the source file, procedures defined, their intended function, author, update history and referenced Global variables and procedures. This code should be at the top of the file, immediately after the TURBO PASCAL UNIT name. This should be accompanied by a date on the top line of the file, and reflects the date of creation, or last date of change. An example of a comment block follows:

```
{$O+,F+}        (* 12/30/93 *)

UNIT M6_INITS;

(*..START UNIT M6_INITS COMMENTS..*)
(* DESCRIPTION:
(*     FILE \TCAS\LOGIC\M6_INITS
(*     Performs TCAS Ver 6.0 task INITIALIZE. Sets all Global data
(*     structures to Null (STRUCTURE ITF, TF, WL, N, G).
(*     Initializes TCAS Ver 6.0 Global Variables (STRUCTURE G).
(*     MOPS Vol II, Sec 3.1
(* INTERFACE DESCRIPTION:
(*     USES TYPE_COM, TYPE_TC;
(*     PROCEDURE M_INIT(P:P_PTR;G:G_PTR);
```

```
(* USED BY UNITS:
(*      INITAL.PAS
(* UPDATED:
(*      12/30/93 : Ed Butterly, CTA - Deleted unused comment block headings.
(*      08/09/93 : E. Butterly, CTA - Filled in comment blocks.
(*      11/05/92 : J. Bottiger, CTA - Deleted all occurrences of G^.CORRECTIVE.
(*      09/17/92 : J.S.Searight - Replaced TPTS with SIM_RTRN in USES clause.
(*              Moved appropriate UNITS out of INTERFACE USES clause,
(*              and into IMPLEMENTATION section.
(*..END UNIT M6_INITS COMMENTS..*)
```

Each subroutine should contain a comment block that lists its name, the data fields input, the data fields output, and any subroutines called. An example of a comment block follows:

```
PROCEDURE M_INIT(P:P_PTR;G:G_PTR);

(*..START PROCEDURE M_INIT COMMENTS..*)
(* DESCRIPTION:
(*      Performs TCAS Ver 6.0 task INITIALIZE.  Sets all Global data
(*      structures to Null (STRUCTURE ITF, TF, WL, N, G).
(*      Initializes TCAS Ver 6.0 Global Variables (STRUCTURE G).
(*      MOPS Vol II, Sec 3-P4
(* INPUTS:
(*      P^.ZLARGE
(*      P^.RADARLOST
(*      P^.IDOWN
(* OUTPUTS:
(*      G:G_PTR
(* CALLED BY UNIT/PROCEDURE OR LOCAL:
(*      INITIAL                         :P:INITAL.PAS
(* EXTERNAL PROCEDURE/FUNCTION CALLS:
(*      REL_G                           :P:RELEASE.PAS
(*      TPT_UPDATE                          :P:SIM_RTRN.PAS
(* UPDATED:
(*      11/05/92 : J. Bottiger, CTA - Deleted all occurrences of G^.CORRECTIVE.
(*..END PROCEDURE M_INIT COMMENTS..*)
```

The BEGIN/END statements of the body should also identify the routine.

```
e.g.  BEGIN (* routine name *)
        ...
        ...
      END;  (* routine name *)
```

Each END statement should identify what it is closing out. e.g.

```
      END; (* CASE *)
      END; (* WHILE *)
      END; (* IF *)
      END; (* ELSE *)
```

4.1.2 STYLE

The lines of source code should be nested to reflect the statement groupings. This nesting will be an indentation of at least two spaces for each statement block. BEGINs and ENDs will also be indented two spaces from the previous line.

e.g.
```
IF (FLAG = TRUE) THEN
   BEGIN
      A := B;
     B := C;
   END (* IF *)
 ELSE
   BEGIN
      A := C;
   END; (* ELSE *)

WHILE (FLAG = TRUE) DO
   BEGIN
      A := A+B;
      IF (A > C) THEN
         FLAG := FALSE;
   END; (* WHILE *)

REPEAT
   A := A+B;
UNTIL (A > C);

FOR I := 1 TO 5 DO
   BEGIN
      A := A+C;
   END; (* FOR *)
```

4.1.3 CONSTANTS

Only constants that are program specific will be declared within a program or unit file. (e.g. loop controls, array limits) Constants should be put into their corresponding CONSTANT UNIT whenever applicable. These CONSTANT UNITS are as follows:

CONST_SU - constants that support the SUMMARY software
CONST_CO - constants for other DRNA software and data, including those for manufacturers live data
CONST_FT - constants that support the FTEG system
CONST_TC - constants used by the MOPS TCAS Logic

4.1.4 DATA INITIALIZATION

All programs will use an initialization procedure to ensure that all data fields, data structures, and memory pointers are properly initialized. The baseline software maintains procedures that initialize all record structures used by TCAS Logic Systems software. These should be used whenever those records are created. When a data structure is modified, its associated initialization should also be modified.

The standard initial value for integers and real numbers is the "BAD_VALUE" constant that corresponds to the field size of the variable being initialized. BAD_VALUE equals -353430, which is the largest negative value that can be stored in a 32 bit integer divided by the largest scale factor the TCAS Logic Systems baseline uses (6076 ft per nmi), and then truncated to a multiple of ten. This value can then be checked as an indication that a field has never been assigned a meaningful value, without overflowing

integer values when performing units scale calculations. There are also BAD_VALUE_1 and BAD_VALE_2, for one and two byte fields, which are used to mark undefined fields recorded in binary record files. Their values are the largest negative values possible for those fields

Arrays are to be used only when there is a definite number of values being collected. Dynamic memory allocation is strongly recommended as a means of linking similar pieces of data.

BIT POSITIONS are such that when writing routines that manipulate bits, the programmer must be aware, and adhere to, the bit numbering conventions that are implemented by the various TCAS manufacturers. It is the programmer's responsibility to identify what data is to be processed, what bit numbering and accessing specifications where used to collect the data, and develop code to adhere to those specifications.

4.1.5 NAMING CONVENTIONS

This section describes the standards that TCAS personal adhere to when naming files, procedures, functions, data types, constants, and variables. There are some standard prefixes for different categories of files, subroutines, data types, etc.

DV_..... : Dalmo Victor support routines
BX_..... : Bendix support routines
HW_.....: Honeywell support routines
RC_..... : Rockwell Collins support routines
BXO.... : Bendix TCAS I support routines
BFG_....: B.F. Goodrich TCAS I support routines
M#_..... : A version of a routine used for TCAS II (Minimum), with Change Number
S#_..... : A version of a routine used for TCAS II Change 7.
E#_..... : A version of a routine used for TCAS III (Enhanced), with Version Number
BXL_....: Bendix TCAS II LIP
SDV_....: Sperry Dalmo Victor TCAS II LIP
BXP_....: Bendix TCAS III Prototype
SIM_.... : Simulation support/interface
OWN_....: Own summary modules
ENC_.... : Encounter summary modules
ERD_.... : Programs that process files that contain records of
ENCOUNTER_RECORDING_DATA
TRD_.... : Programs that process files that contain records of TRACK_RECORDING_DATA
PR_..... : Print routines for special fields
WR_..... : Write to output file routines

* TERMINOLOGY AND LABELING FREQUENTLY USED :

AOA : ANGLE OF ARRIVAL (BEARING MEASUREMENT)
ADOT,
ZDOT,
ZD : ALTITUDE RATE
RDOT,
RD : RANGE RATE
IDOWN : IDENTIFICATION OF OWN AIRCRAFT
IDINT : IDENTIFICATION OF INTRUDER AIRCRAFT

4.1.6 UNITS OF MEASUREMENT

Table 4.1 lists the common fields of measurement processed by the TCAS software systems, the units they are stored and displayed in, their range, and their display format.

Unit	storage format	internal units	display units	range	display format
Time of day	real6	sec	HMS	[0 to 86,4000.0)s	HH:MM:SS(.s)
System time	real6	sec	sec	[0 TO 1024) sec	F7.2 (16 bit)
				[0 to 24 hrs = 86400 s)	F8.2 (32 bit)
TAUs	real4	sec	sec	[0 to 120]s	F6.2
Altitude	real4	ft	ft	[-2000 to +64,000]ft	I5
Alt. rate	real4	ft/sec	ft/min	±10,000fpm	I5
Range	real4	nmi	nmi	[0 to 30]nmi	F6.3
Range rate	real4	nmi/sec	knots	±1200kt	I4
Bearing/Heading	real4	degrees	degrees	[0 to 360]ø	I3
Mode-S ID	int4	24 bits	Hex		Z6
ATCRBS msg	int4	DABC	DABC/ABCD	12 bits	O4
ATCRBS Garble	int4	DABC	DABC/ABCD	12 bits	O4
Power/Amplitude	int4	dB	dB	[-80, 80]	I3
Pitch/Roll	real4	degrees	degrees	[-90, +90]	F6.2

TABLE 4.1

4.2 SOFTWARE STANDARDS

GLOBAL variables will not be used except when absolutely necessary. The programmer should explore every other possibility before implementing the use of a GLOBAL variable. If the programmer decides one must be used he will first discuss the implementation with the Configuration Manager. Avoiding GLOBAL variables will prevent variable name conflicts. Routines that share variables should be part of the same unit.

USES clauses will be carefully checked by the programmer to make sure only those units needed are included. This will keep EXE and TPU file sizes to a minimum.

The INTERFACE section will contain only those subroutines that will be called by other units, and only reference units in the USES clause that are required for those routines. The programmer should make an effort to keep these to a minimum.

5.0 SOFTWARE ENVIRONMENT

All TCAS Personnel use a predefined set of Turbo Pascal default compiler settings to insure that executable code compiled on different PCs will be identical.

5.1 TURBO PASCAL STANDARDS

5.1.1 SYSTEM CONFIGURATION

Since all TCAS Personnel and the supporting contractors have similar PCs, a standard setup of system software, and support directories is expected to be upheld. This will help to ensure software integrity, as everyone will be compiling and linking from identical environments, and will make it easier for TCAS Personnel to work on any given PC. The minimum DOS path must include the following: C:\DOS;C:\ZIP;C:\TP7\BIN;D:\TCAS\TPU.

The Turbo Pascal system should be found on the C: drive in a directory named TP7 (C:\TP7\). All Turbo system subdirectories should be subdirectories of C:\TP7. The D: drive should contain a directory TCAS (D:\TCAS). This directory should contain at least the following subdirectories: PROGS, UNITS, and TPU. The directories PROGS and UNITS are to be used to store software an individual is currently working on. All *.TPP and *.EXE files should be directed to the TPU directory when compiling from either the Turbo environment or the BPC line command.

The user should be aware of the various ways to compile and link TCAS code. If a baseline compilation is to be done, then only units on drive U: should be accessed. If someone is testing code they are developing, their D:\TCAS\UNITS directory should be searched, and then U:. If someone wants to compile software that has been submitted for acceptance to the baseline, but it has not yet been accepted, N:UNITS should be searched before searching U:. This can be easily controlled by having TPC files for all three circumstances present in the C:\TP7 directory, and copying the appropriate one into BPC.CFG before compilation. All TPC files should direct *.TPP, and *.EXE files to D:\TCAS\TPU. The TPC files should be named as follows, respectively: BPC.BAS, BPC.LOC, BPC.LAN.

The BPC files must contain the following compiler options after the source code path specifications:

```
/CP          { compile in protected mode }
/M           { make }
/GD          { generate detailed MAP file }
/$A+         { align words to even addresses }
/$B-         { short circuit Boolean evaluation }
/$L-         { disable local symbol info }
/$N+         { numeric coprocessor on }
/$Q+         { overflow checking on }
/$R+         { range checking on }
/$M32000       { stack size }
```

These options must also be the default options in the Turbo Pascal integrated development environment (IDE).

APPENDIX A

No: _____

CHANGE REQUEST / PROBLEM REPORT

Name: T. Choyce Date:

| | Change Request | | | Problem Report

| PRIORITY (circle one) | I critical | II major | III minor | IV annoyance | V other |

AREA of CONCERN:

	SUMMARY		BENDIX	ADDITIONAL COMMENTS:
	LISTER		HONEYWELL	
	INTERVAL		ROCKWELL	
	LOGIC VER. _____		FTEG	
	OTHER: _____			

DESCRIPTION of PROBLEM:

SUGGESTED RESOLUTION:

Initials & Date: _____

STATUS:

| | COMPLETED | | TESTING | | PLANNING | | ON HOLD

| | REJECTED

Initials & Date: _____

Change No.	Subsystem	Procedure Name	Change Date

Change By: Tom Choyce	Unit Name

Category of Failure (circle one) I critical II major III minor IV annoyance V other

Brief Description: (50 character max)

|_|_|_

ATTACHMENT #1

TECHNICAL LEAD ASSIGNMENTS

PMK : FTEG - FTEGMAIN program and its support routines.
PLAYBACK - PLAYBACK program.
TRD - All routines that are involved in the recording of TRD output files, and the programs that process them. (TRD_EXCL, TRD_LOOP, TRD_PLOT, and TRD_TTP)

KMC : LOGIC - All routines that are called to invoke TCAS logic. This applies to all versions of the Logic, and all TCAS system variables.

JP : TTP Processing - All programs and support routines that are tools used analyze TTP data. Much of this code is manufaeturer specific. This is broken down into the following subdivisions:
- INTERVAL
- LISTER
- LOOP
- SCAN

SUMMARY - All SUMMARY programs, and their major subsystems (ERD_REC, OWN_START_CAS, OWN_END_SURV)

JSS : ERD - All programs and routines that are involved in processing ERD record files. (ERD_1045, ERD_731, ERD_CHK, ERD_CNTS, ERD_CUT, ERD_DEL, ERD_FAIL, ERD_LIST, ERD_LOOP, ERD_MRG, ERD_NAME, ERD_TTP, ERD_VRSN, ERDMATRX, and ERDSORTM)
COMMON - All units that contain routines that are shared by more than one major TCAS system or subsytem.
SUPPORT - All units that contain support routines such as WR_* and PR_*, data field or unit conversion, and file handling routines.

KF : TCAS I - All programs and units that deal solely with TCAS I data processing. (BFG and BXO)