AN INTEGRATED DEVELOPMENT ENVIRONMENT FOR SPACECRAFT CHECKOUT TEST PROCEDURES

U.N.Vasantha Kumari

ISRO SATELLITE CENTRE
BANGALORE, INDIA
unv@isac.gov.in

INTRODUCTION:

Spacecraft Testing during the AIT Phase in an involved activity. Ensuring Automation, repetivity of the tests during various phases of AIT are quite complex in nature. To prepare the satellite test procedures in advance in an English like language with features like automated checking of the related parameters, pre-requisites, linking of test procedures are available in the CHECKOUT COMMAND LANGUAGE (CCL) designed and developed in-house at ISRO Satellite Centre, Spacecraft Checkout Group. The integrated development environment for the preparation, validation, updation and maintenance of the Test Procedures helps in a professional way of managing the satellite test procedures. It maintains a set of libraries, directories based on the sub-system wise, Phase wise etc. which helps in choosing the appropriate test procedure files and creating the Test Schedules. This paper discusses the various features available in the IDE for Spacecraft Test Procedures designed and developed with the idea of helping the test engineer during the testing phase. It provides ease of operation, quick way of generating schedules etc. Simulation feature added to this IDE provides the test engineer in evaluating the test time required, checking of conflicting procedures and also helps in training the new test engineers.

SPACECRAFT CHECKOUT TEST PROCEDURES:

At ISRO Satellite Centre, a well established, proven set of software packages called Automatic Checkout Software System (ACSS), which is developed in-house is being used for testing all classes of spacecrafts. ACSS consists of real-time packages, offline packages and support software packages.

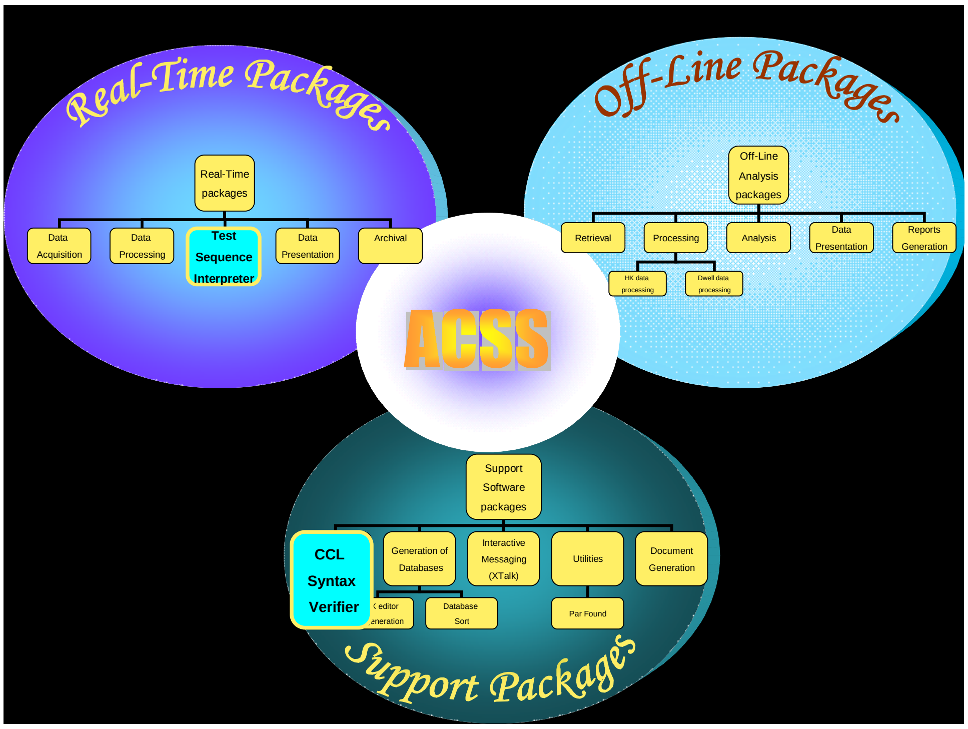


Fig 1 : Over View of ACSS

Before start of any spacecraft checkout phase there will be a preparatory phase where the hardwares, softwares, test plans and procedures including the databases required are made ready. This phase includes the study of on board systems and preparation of test plans for various phases of checkout like Integration Phase, Dis-Assembled IST, Assembled IST, Environmental tests like Thermovac, Acoustic, Vibration, CATF etc. and various special tests like deployment of panels, antenna, illumination, end-to-end tests etc.

After approval of various test plans, test procedures are written in in-house developed English like language called Checkout Command Language (CCL). CCL has all required instructions for writing s satellite test procedure, and the list of some of the instructions available in CCL is given below:

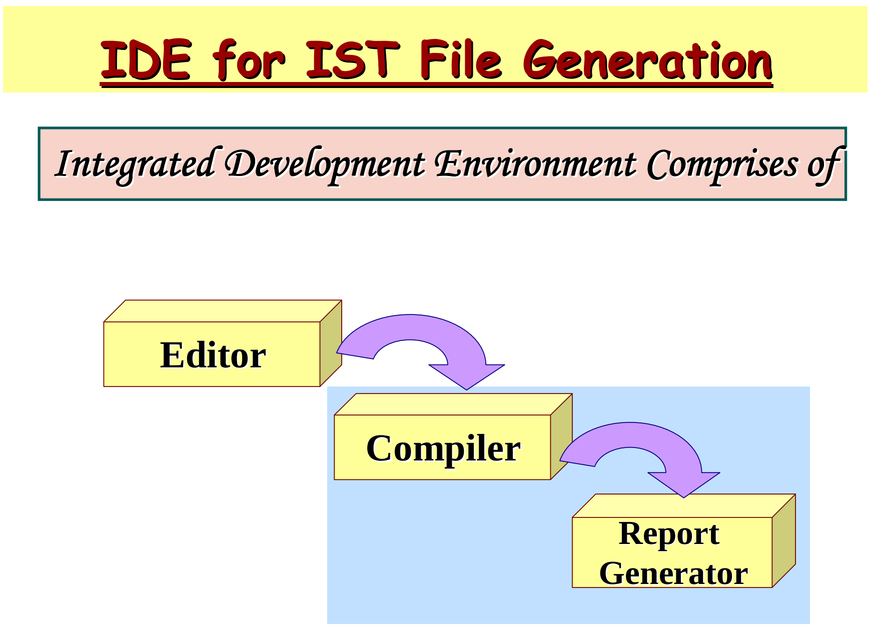
SEND	CHECK	PLOT	PRINT	SINGLESTEP
SENDLIST	EXPECTED	DISPLAY	PRINTPAGE	BREAKPOINT
MACRO	CHECKSCOE	TESTNAME	PRINTPLOT	CALL
SET	READ	INHIBIT	PRINTMACRO	END
REMARK				

Every test procedure will have a header portion containing the ‘testname’, test conditions, pre-conditions then the actual test procedures and before ending the test putting back the original conditions if required.

For some spacecraft checkouts procedures may be inherited from earlier similar satellite checkout. Test procedure preparation, validation for syntax etc. is a voluminous work done by a team of identified test engineers.

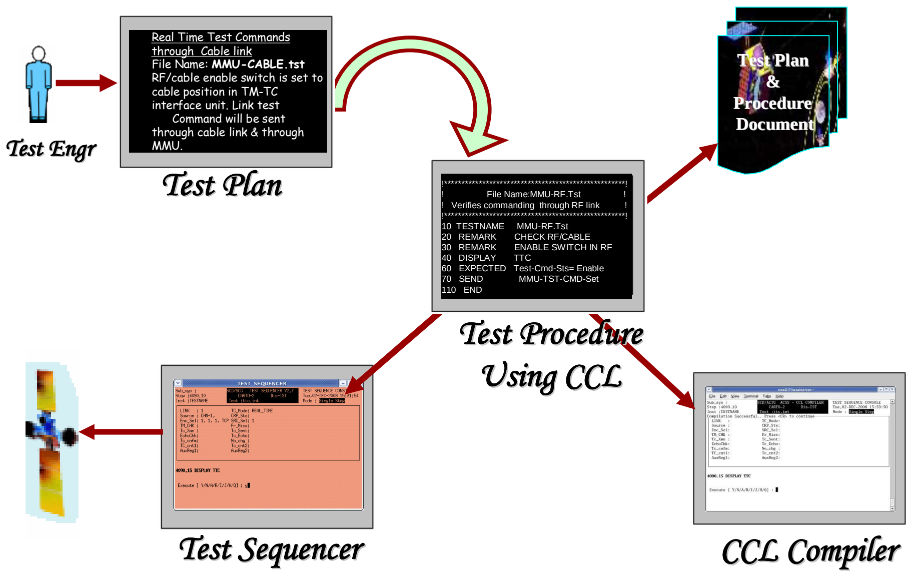
INTEGRATED DEVELOPMENT ENVIRONMENT(IDE) FOR TEST PROCEDURES:

IDE for test procedures of a spacecraft checkout consists of an Editor, Compiler, Report Generator as shown below:


IDE for IST File Generation
Integrated Development Environment Comprises of
Editor
Compiler
Report Generator

Currently, the test engineer prepares the test plan, discusses with the concerned sub-system, obtains approval, generates the document and based on this starts the test procedure preparation using CCL. These procedures are passed through a support software called CCL-Compiler, which verifies the syntax and clears the procedure for further activities.

Present Scenario



Real Time Test Commands through Cable link
File Name: **MMU-CABLE.tst**
RF/cable enable switch is set to cable position in TM-TC interface unit. Link test Command will be sent through cable link & through MMU.

```
!****************************************************!
!             File Name:MMU-RF.Tst                   !
!   Verifies commanding  through RF link             !
!****************************************************!
10  TESTNAME    MMU-RF.Tst
20  REMARK      CHECK RF/CABLE
30  REMARK      ENABLE SWITCH IN RF
40  DISPLAY     TTC
60  EXPECTED    Test-Cmd-Sts= Enable
70  SEND         MMU-TST-CMD-Set
110   END
```

```
Sub_sys :          SCD/SCG   TEST SEQUENCER V2.7     TEST SEQUENCE CONSO
Step :4090.10        CARTO-2       Dis-IST           Tue,02-DEC-2008 15:31:54
Inst :TESTNAME     Test :ttc.int                     Mode : Single Step

 LINK    : 1            TC_Mode: REAL_TIME
 Source  : CHN-1,       CRP_Sts:
 Enc_Sel: 1, 1, 1, TCP  SRC_Sel: 1
 TM_CHK  :              Fr_Miss:
 Tc_Xmn  :              Tc_Sent:
 EchoChk:               Tc_Echo:
 Tc_cnfm:               No_chg :
 TC_cnt1:               Tc_cnt2:
 AuxReg1:               AuxReg2:

4090.15 DISPLAY TTC

Execute [ Y/N/A/R/I/J/H/Q] : y
```

```
Sub_sys :          SCD/AITG  ACSS - CCL COMPILER     TEST SEQUENCE CONSOLE
Step :4090.10        CARTO-2       Dis-IST           Tue,02-DEC-2008 15:33:50
Inst :TESTNAME     Test :ttc.int                     Mode : Single Step
Compilation Successful.. Press <CR> to continue
 LINK    :              TC_Mode:
 Source  :              CRP_Sts:
 Enc_Sel:               SRC_Sel:
 TM_CHK  :              Fr_Miss:
 Tc_Xmn  :              Tc_Sent:
 EchoChk:               Tc_Echo:
 Tc_cnfm:               No_chg :
 TC_cnt1:               Tc_cnt2:
 AuxReg1:               AuxReg2:

4090.15 DISPLAY TTC

Execute [ Y/N/A/R/I/J/H/Q] :
```

Existing System

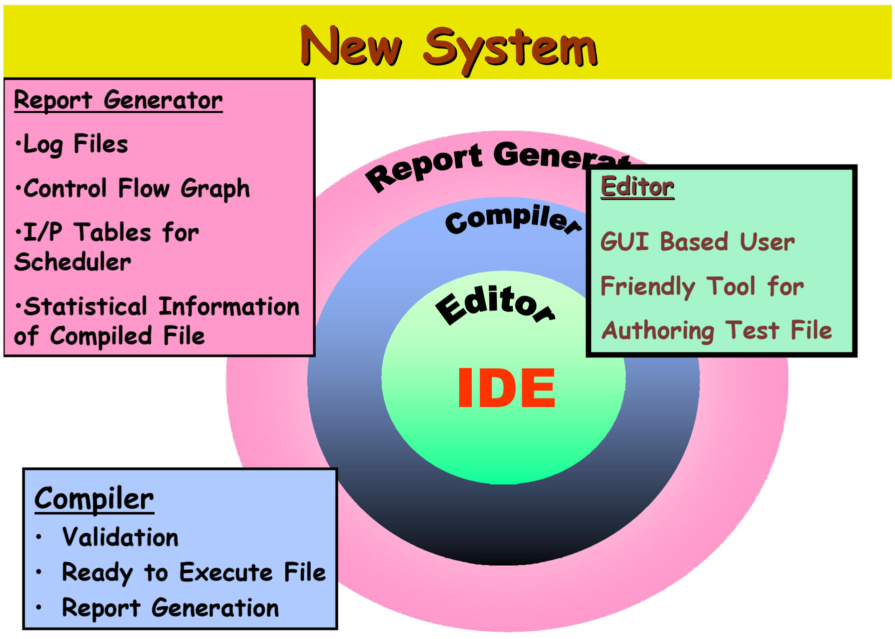
A Typical Test File

```
!*******************************************************************************!
!                        File Name: MMU-Cable.Tst                               !
!            This procedure is to verify commanding through Cable link          !
!*******************************************************************************!

10  TESTNAME    MMU-Cable.Tst

20  REMARK      CHECK RF/CABLE ENABLE SWITCH IN CABLE POSITION TM-TC

30 REMARK       INTERFACE UNIT

40  DISPLAY     TTC

50 EXPECTED     Test-Cmd-Sts= Enable

60  SEND            MMU-TST-CMD-Set

70  EXPECTED    DECODER-TEST = Disable

90   SEND           MMU-TST-CMD-Rst

100   REMARK      BOTH TEST COMMANDS TESTED THROUGH CABLE LINK

110   END
```

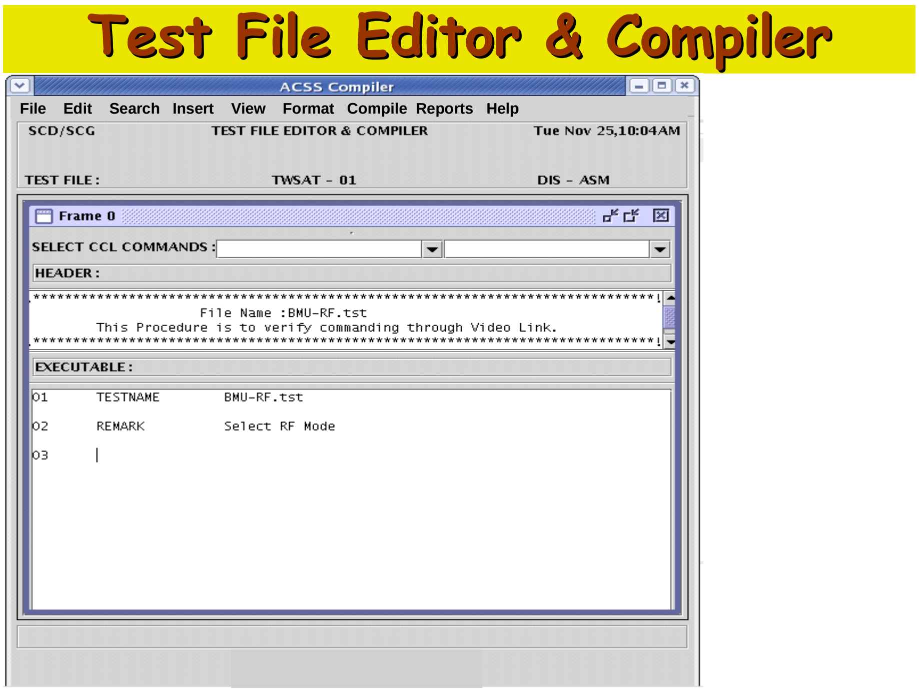
Features

- ASCII Based Test Files
- Written Using CCL
- 10 Lines to 10,000 Lines
- Header and Body
- Generated on any System
- Ported to UNIX
- Compiled using Compiler
- Errors Reported

Current system had many limitations which were reported by the users, some of them are file is text oriented and cumbersome to enter using ‘vi’ editor of UNIX/LINUX or word. CCL compiler provides only the errors list. To overcome some of these issues and to improve upon the existing system in terms of additional features a new system of IDE is worked out and the schematic of the new system is given below:

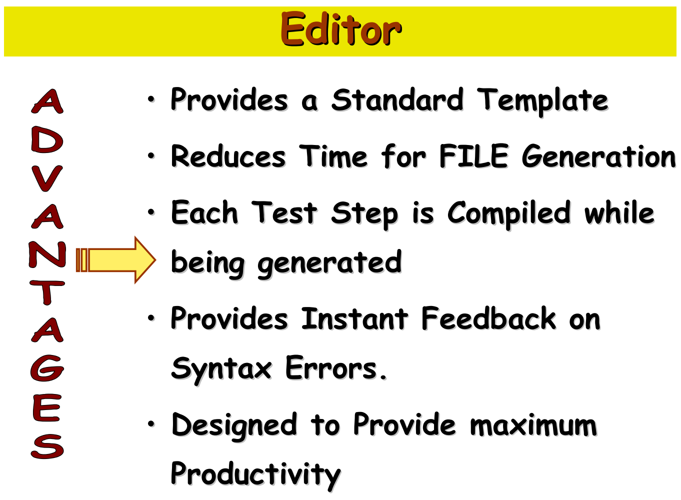


The new test file editor is GUI based and has many drag-drop features, link with database files and provides hints to the user while creating the file itself.


Test File Editor & Compiler
ACSS Compiler
File Edit Search Insert View Format Compile Reports Help
Frame 0

```
SCD/SCG              TEST FILE EDITOR & COMPILER          Tue Nov 25,10:04AM

TEST FILE :                TWSAT - 01                      DIS - ASM

SELECT CCL COMMANDS :
HEADER :
*****************************************************************************
                         File Name :BMU-RF.tst
            This Procedure is to verify commanding through Video Link.
*****************************************************************************
EXECUTABLE :
01      TESTNAME        BMU-RF.tst
02      REMARK          Select RF Mode
03      |
```

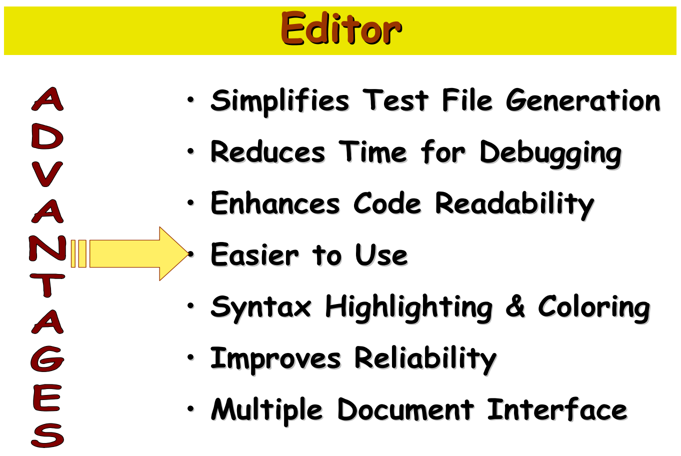
The advantages of the new Editor are listed below:



Editor

A D V A N T A G E S

- Provides a Standard Template
- Reduces Time for FILE Generation
- Each Test Step is Compiled while being generated
- Provides Instant Feedback on Syntax Errors.
- Designed to Provide maximum Productivity

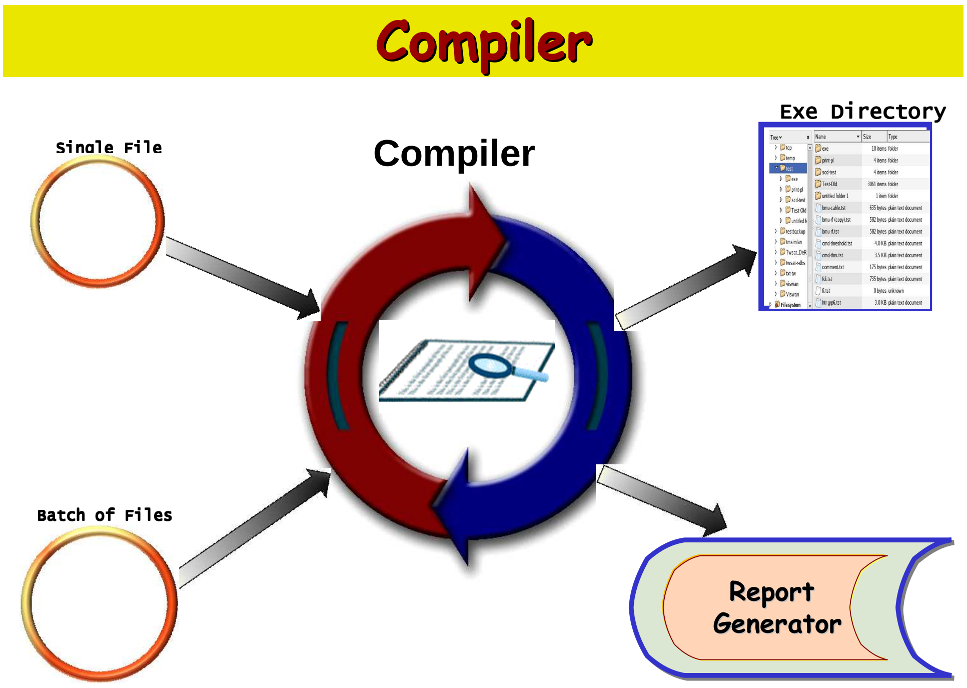


Editor

A D V A N T A G E S

- Simplifies Test File Generation
- Reduces Time for Debugging
- Enhances Code Readability
- Easier to Use
- Syntax Highlighting & Coloring
- Improves Reliability
- Multiple Document Interface

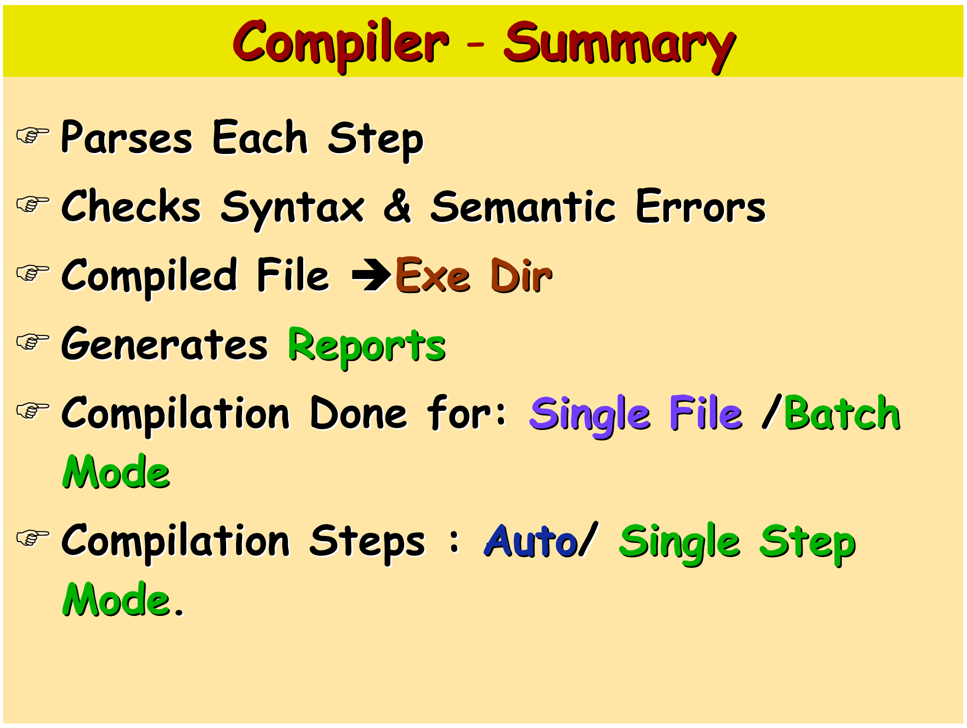
After the editor's job is done, compiler takes over and the block schematic of the compiler is shown below:



Test engineers can compile files imported from previous projects or generated earlier also (these files need not be through the IDE editor). The main functions of the compiler are:

- Parsing every step in the test file
- Checking for sematic and Syntax errors
- Reporting the errors to the user with proper step.no
- Archieving error free files to the executive directory
- Setting of privileges to the test file

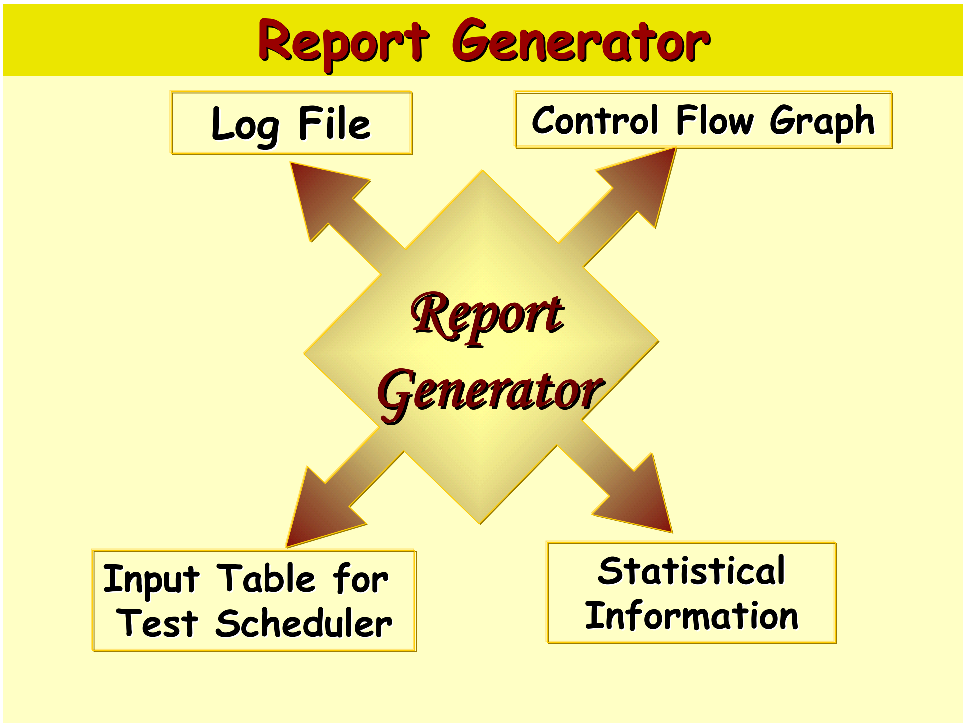
Files can be compiled one by one or as a batch. Compilation can be done is automode or in single step mode. Reports generated also contains the quality of the file, proving various useful information to the test engineer. Summary of the functions of the compiler are given below:



Compiler - Summary

- Parses Each Step
- Checks Syntax & Semantic Errors
- Compiled File ➔Exe Dir
- Generates Reports
- Compilation Done for: Single File /Batch Mode
- Compilation Steps : Auto/ Single Step Mode.

Let us look at the functions of the report generator:



As shown in the above figure, report generator maintains a 'LOG FILE' which contains:

Log File

Contents

- **Syntax and Semantic Errors**
- **Single File/Batch Mode Files**
- **Date and Time Stamping**

Archival

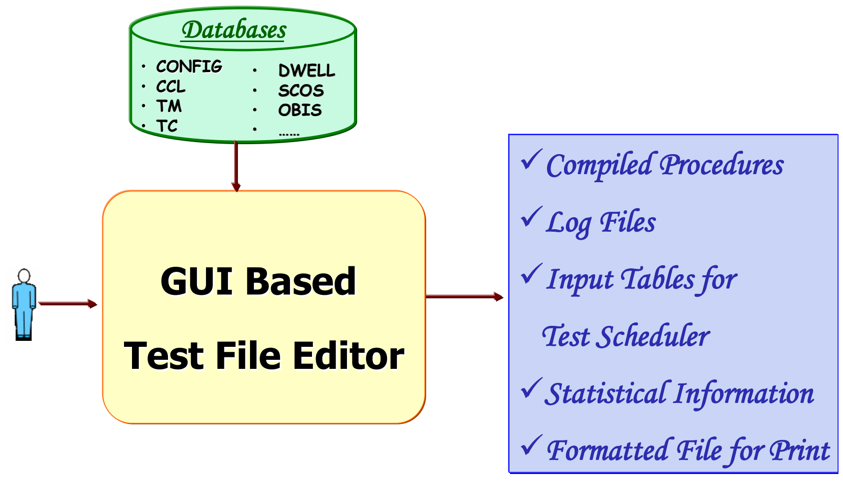
Statistical Information

- **Quality of the File**
- **Time For Execution**
- **Parallel Execution Possible or Not**
- **Optimization Guidelines.**

The arvhieval path and the details are also part of the log file.

CONCLUSION:
IDE is a powerful tool for generation of test procedure files, and it provides the user friendly environment for the generation, maintenance, history of test files used for any satellite checkout. To sum up the system over view of IDE is shown below:

System Overview


Databases
CONFIG
CCL
TM
TC
DWELL
SCOS
OBIS
……
GUI Based Test File Editor
✓ Compiled Procedures
✓ Log Files
✓ Input Tables for Test Scheduler
✓ Statistical Information
✓ Formatted File for Print

The output of IDE also contains information which will form input to the test scheduler, which helps the test engineer in scheduling the tests in an optimum way. Improvements and additional features addition to IDE is being carried out as a continuous process and new versions are released at regular intervals.

The author wishes to thank all her team members for the support in realizing this IDE and for being involved in continuous improvements in this area. My sincere thanks are also to the management and Centre Director, ISRO SATELLITE CENTRE for giving me encouragement and permission to submit this paper for the workshop.

Abbreviations and Acronyms

ACSS	Automatic Checkout Software System
CCL	Checkout Command Language
IDE	Integrated Development Environment