How To Get The Most From Agilent Intelligent Yield Enhancement Test (IYET)

White Paper

Last Modified: 6/21/2005

Version 1.0

Agilent Technologies
Electronics Manufacturing
Automated Test Group

Agilent Technologies

Introduction

This paper describes how to get the most from Agilent Technologies' new Intelligent Yield Enhancement Test (IYET) for Agilent board test systems.

IYET is a dynamic re-test strategy that automatically re-tests *failing* unpowered tests after cycling vacuum. Since only *failing* unpowered tests are re-tested (not *all* unpowered tests), IYET can yield the following benefits:

• Higher throughput - Fewer tests are executed during each re-test sequence.

• Fewer false calls - Passing tests are not re-tested, so they have less of a chance to falsely fail.

• Fewer escapes - Re-tested bad boards are failed correctly with fewer tests, so they have less of a chance to escape.

IYET is implemented entirely in the testplan using BT-Basic. No software update is required. No tests need to be recompiled. No external programs are called.

The IYET testplan executes the unpowered tests just like a standard testplan. When an unpowered test fails, however, the IYET testplan dumps the report buffer to a file, parses it for failing tests, cycles vacuum, and re-tests only the failures. File I/O is done during vacuum cycle for maximum throughput. All other features of a standard testplan (datalogging, panelized boards, multiple board versions, serialization, ART, etc.) are retained.

IYET captures re-test failure data and displays re-test history for corrective action. Over time, this can result in better fixture and tester performance by examining re-test history, false calls due to most fixture contact problems or poorly debugged tests can be identified and corrected before they impact yield.

New IYET-enabled testplans can be created for the 3070UX and 3070WN platforms from testmains found in the zip package in the electronic version of this paper. These testmains are based on the 3070 05.30p standard testmains for single and panelized boards. With minor edits, they can be used with older software revisions. IYET can be also be added to most existing testplans. Finally, IYET will be part of the standard testmains for the i5000 platform and the 3070PC beginning with the next major software release.

Creating a new IYET Testplan

This procedure assumes that you have completed the typical test development process in the target board directory but have not yet begun debug. A new testplan will be created based on one of the IYET testmains.

Download the testmains.zip file and unzip the IYET testmains.

testmains.zip

A directory called "testmains" will be created with the following files:

• 3070ux/testmain

• 3070ux/testmain_panel

• 3070pc/testmain

• 3070pc/testmain_panel

Copy the appropriate testmain to the board directory under the "ipg" directory. If you use FTP to copy the "ux" testmains to a 3070UX system, make sure to use the "ascii" transfer method. For panelized boards, re-name the file "ipg/testmain_panel" to "ipg/testmain".

Go to the target board directory. Examine the "testorder" file and change the line:

generate testplan off

to

generate testplan on

if present and re-save. From BT-Basic in the target board directory, execute the command:

Testplan generation

A new "testplan" will be written using the testmain found in "ipg/testmain". You can now begin your normal testplan debug process.

The IYET testmains can also be copied to your "/Agilent3070/standard" or "/hp3070/standard" directories and named "testmain" and "testmain_panel", respectively. This will cause the testplan writer to create IYET testmains for all board directories developed on that system. Be aware, however, that these files will get overwritten during a software update.

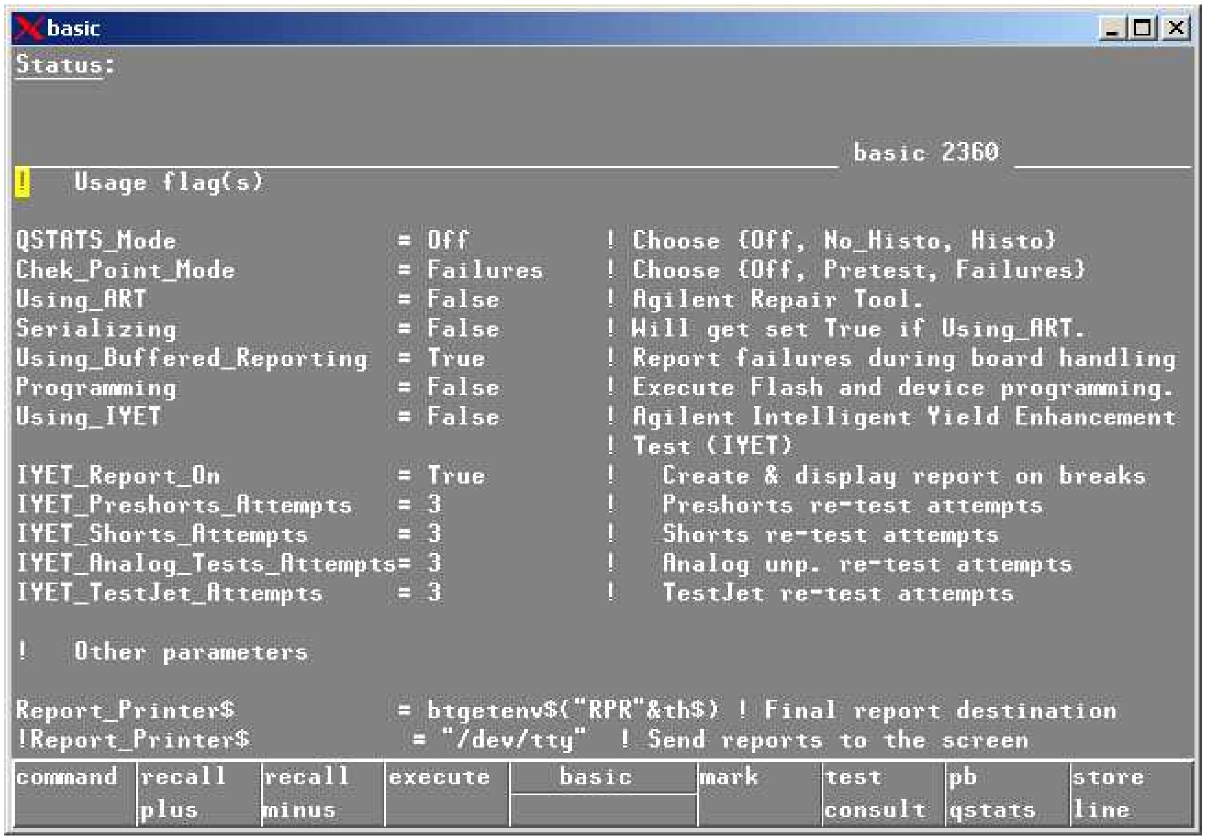
Getting started with an IYET Testplan

The newly created testplan contains additional flags in the subroutine "Set_Custom_Options" that are used to control IYET. By default, IYET is turned off, so that you can do nothing and debug the IYET testplan just like you would a standard testplan.

Edit "Set_Custom_Options"

To begin using the IYET, find the subroutine "Set_Custom_Options".

A screenshot is shown in Figure 1 with the IYET features in the yellow box.



```
!   Usage flag(s)

QSTATS_Mode               = Off        ! Choose {Off, No_Histo, Histo}
Chek_Point_Mode           = Failures   ! Choose {Off, Pretest, Failures}
Using_ART                 = False      ! Agilent Repair Tool.
Serializing               = False      ! Will get set True if Using_ART.
Using_Buffered_Reporting  = True       ! Report failures during board handling
Programming               = False      ! Execute Flash and device programming.
Using_IYET                = False      ! Agilent Intelligent Yield Enhancement
                                       ! Test (IYET)
IYET_Report_On            = True       !   Create & display report on breaks
IYET_Preshorts_Attempts   = 3          !   Preshorts re-test attempts
IYET_Shorts_Attempts      = 3          !   Shorts re-test attempts
IYET_Analog_Tests_Attempts= 3          !   Analog unp. re-test attempts
IYET_TestJet_Attempts     = 3          !   TestJet re-test attempts

!   Other parameters

Report_Printer$           = btgetenv$("RPR"&th$) ! Final report destination
!Report_Printer$            = "/dev/tty"  ! Send reports to the screen
```

Figure 1 BT-Basic screenshot of IYET usage flags and parameters

Set the "Using_IYET" flag to "True". IYET has other usage flags and parameters. They are summarized in Table 1 and can be modified as desired. Other testplan usage flags and parameters are unchanged and can be edited as desired. The exception is "Chek_Point_Mode" which is described in the section "IYET SUBROUTINE DESCRIPTIONS".

Table 1 Description of IYET usage flags and parameters

IYET Usage flag/parameter	Description
Using_IYET	As described above. Turns on IYET for the testplan.
IYET_Report_On	Displays the top 10 re-tests for the current day's production. Data comes from the re-test log file named "retest<date>.txt" in the "iyet' directory in the board directory. This directory is created if not present.
IYET_Preshorts_Attempts	Controls the number of re-test attempts for failing tests in the "Preshorts" subroutine of the testplan. Must be an integer >= 1. Setting equal to 1 will cause the "Preshorts" subroutine to be executed only once, regardless of failures, effectively turning off IYET for this subroutine
IYET_Shorts_Attempts	Controls the number of re-test attempts for failing tests in the "Shorts" subroutine of the testplan. Must be an integer >= 1. Setting equal to 1 will cause the "Shorts" subroutine to be executed only once, regardless of failures, effectively turning off IYET for this subroutine.
IYET_Analog_Tests_Attempts	Controls the number of re-test attempts for failing tests in the "Analog_Tests" subroutine of the testplan. Must be an integer >= 1. Setting equal to 1 will cause the "Analog_Tests" subroutine to be executed only once, regardless of failures, effectively turning off IYET for this subroutine.
IYET_TestJet_Attempts	Controls the number of re-test attempts for failing tests in the "TestJet" subroutine of the testplan. Must be an integer >= 1. Setting equal to 1 will cause the "TestJet" subroutine to be executed only once, regardless of failures, effectively turning off IYET for this subroutine.

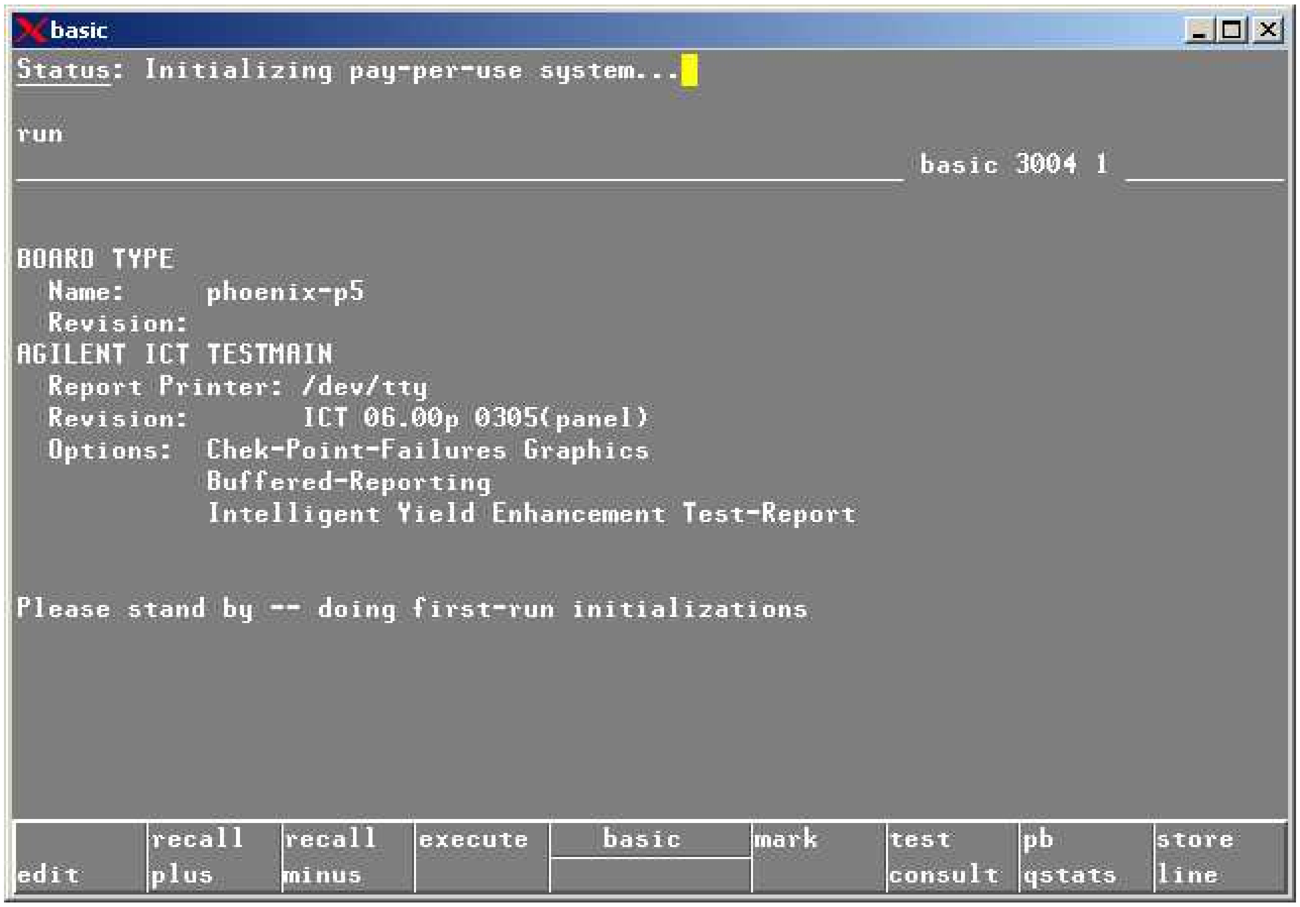
Other Editing

IYET assumes that the fixture can be correctly actuated using the "faon" and "faoff" commands with a 1.5 second delay for each. If not, see the section "IYET SUBROUTINE DESCRIPTIONS" for details on the subroutines, "IYET_Vacuum_On" and "IYET_Vacuum_Off".

IYET executes re-tested tests individually. Be sure that the tests in each test subroutine will pass "on their own". Watch out for gprelay connections, variables, additional "unpowered" statements, or other setups that are in the testplan but not in the test source.

Running the IYET Testplan the First Time

Once enabled, the IYET testplan will differ slightly from the standard testplan flow. First, the startup message will indicate that IYET is enabled. Figure 2 shows the new startup message.

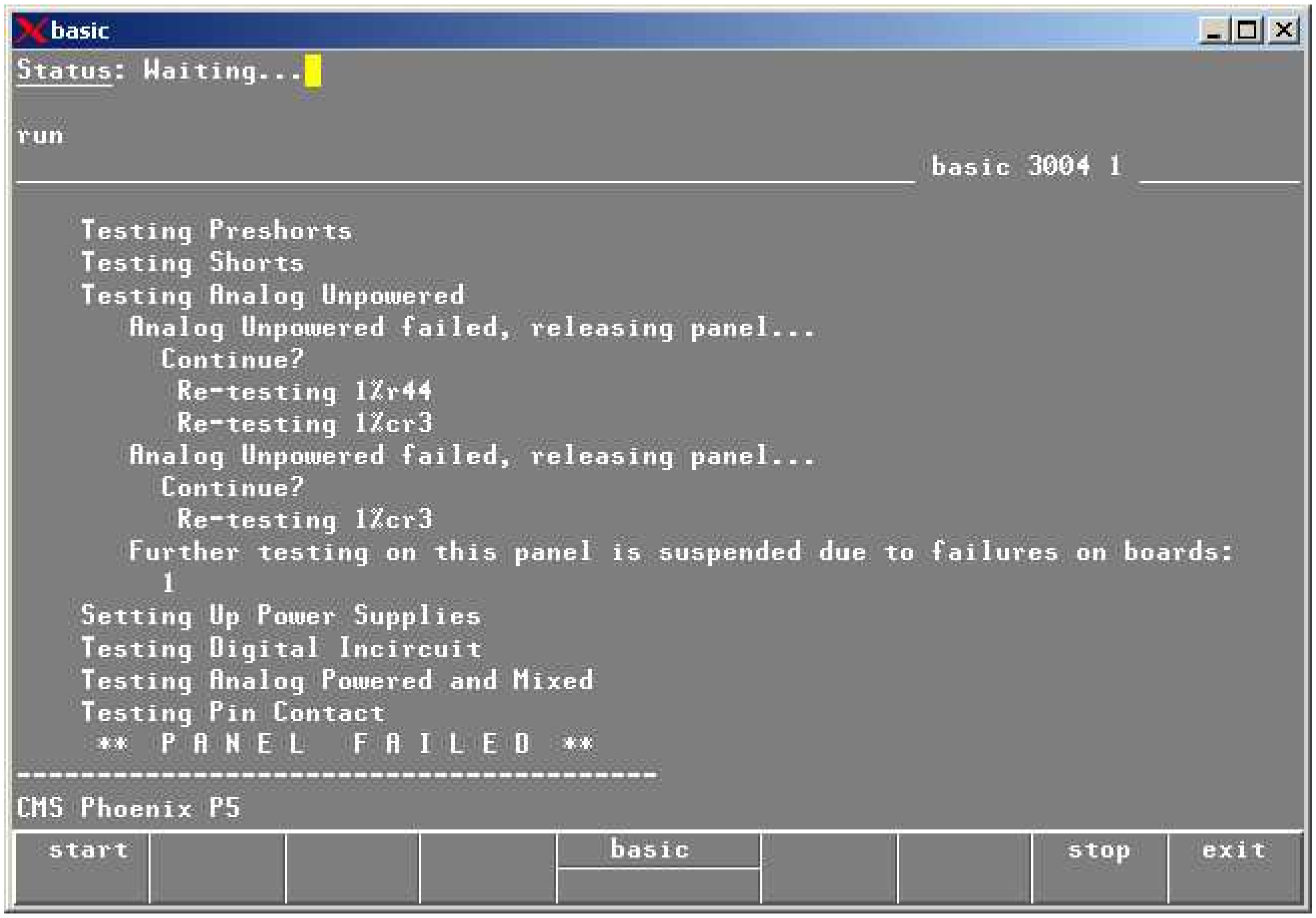


```
Status: Initializing pay-per-use system...

run
______________________________________________________ basic 3004 1 __________


BOARD TYPE
  Name:     phoenix-p5
  Revision:
AGILENT ICT TESTMAIN
  Report Printer: /dev/tty
  Revision:       ICT 06.00p 0305(panel)
  Options:  Chek-Point-Failures Graphics
            Buffered-Reporting
            Intelligent Yield Enhancement Test-Report


Please stand by -- doing first-run initializations
```

Figure 2 BT-Basic screenshot of IYET Startup message

Note the new line for the IYET message highlighted in yellow. If the flag "IYET_Report_On" is false, then the string, "-Report" will be missing.

The testplan creates a directory called "iyet" in the local board directory if one is not present. The temporary failure ticket and re-test log files are kept there. If the "iyet" directory is created (for example, this is the first time the testplan is run), then an unstable test report will not be generated, even if the flag "IYET_Report_On" is true.

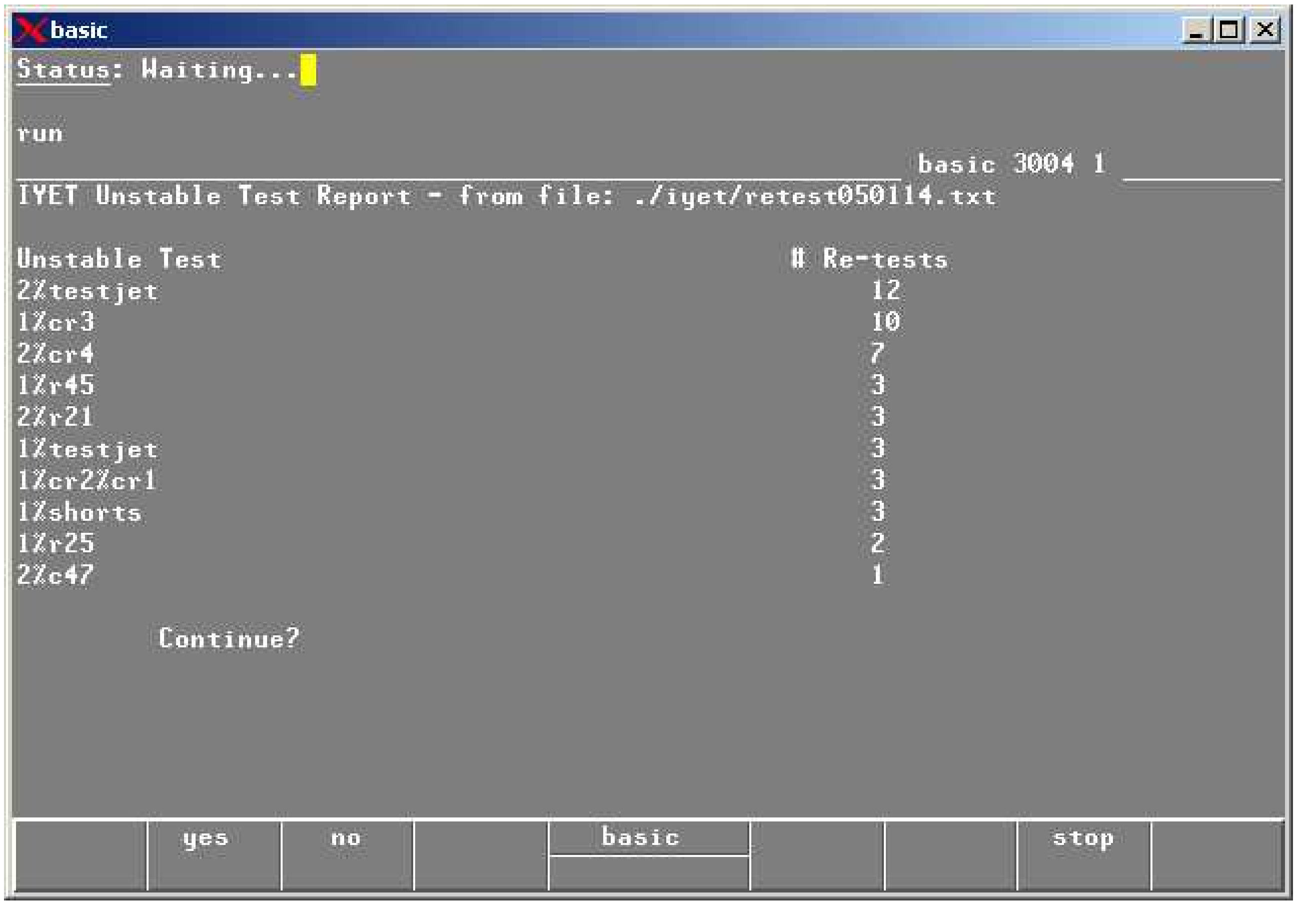


```
Status: Waiting...

run
                                                                    basic 3004 1

    Testing Preshorts
    Testing Shorts
    Testing Analog Unpowered
       Analog Unpowered failed, releasing panel...
         Continue?
          Re-testing 1%r44
          Re-testing 1%cr3
       Analog Unpowered failed, releasing panel...
         Continue?
          Re-testing 1%cr3
       Further testing on this panel is suspended due to failures on boards:
         1
    Setting Up Power Supplies
    Testing Digital Incircuit
    Testing Analog Powered and Mixed
    Testing Pin Contact
     **  P A N E L   F A I L E D  **
----------------------------------------
CMS Phoenix P5
```

Figure 3 BT-Basic screenshot of IYET test flow

Figure 3 shows the BT-Basic window during the normal execution of the testplan. Note that when an unpowered test subroutine fails, vacuum is released. During this “dead” time, the failure ticket is parsed and a list of failing tests are created. After an operator prompt, vacuum is turned on and the failures are re-tested. Do not comment this prompt unless your fixture is approved for actuation without operator intervention. See the section “IYET SUBROUTINE DESCRIPTIONS” for details.

If the flag “IYET_Report_On” is true and the “iyet” directory exists, then an “IYET - Unstable Tests Report” is displayed during initializations each time the testplan is run. Figure 4 shows a screenshot.



```
Status: Waiting...

run
                                                            basic 3004 1
IYET Unstable Test Report - from file: ./iyet/retest050114.txt

Unstable Test                                         # Re-tests
2%testjet                                                  12
1%cr3                                                      10
2%cr4                                                      7
1%r45                                                      3
2%r21                                                      3
1%testjet                                                  3
1%cr2%cr1                                                  3
1%shorts                                                   3
1%r25                                                      2
2%c47                                                      1

         Continue?
```

Figure 4 BT-Basic screenshot of IYET Unstable Test Report

The “IYET Unstable Tests Report” comes from a file listed in the title, in this case “./iyet/retest050114.txt”. Files are named “retest<date>.txt”. This file is appended with each failure ticket for the current date. If there are no retests or there is no file for the current date, then this screen displays the message, “No retests found or missing file”.

The file is parsed and the top 10 unstable tests are listed along with the total number of retests for that test that day. This display should be used to take corrective action on the fixture (replace the probes associated with the test, for example) or test program (add the “ed” option for a noisy test, for example).

Over time, the “iyet” directory will fill up with these re-test log files. The user can view, copy and delete them as needed.

For pay-per-use systems, this report may take several minutes to generate. If this is objectionable, set “IYET_Report_On” to false and review the re-test log files manually.

IYET Testplan Description

This section provides a detailed description of the IYET testplans. It is intended for those familiar with BT-Basic who need information to add IYET to existing testplans.

3070 testplans consist of two parts, the testmain and test subroutines. The testplan writer copies the testmain into the target board directory and appends the test subroutines. The IYET testmain is a lightly edited standard testmain with 7 subroutines (8 for panelized boards) appended. The naming convention for the additional IYET code is as follows:

- Subroutines - Name begins with “IYET”.
- Global variables - Contain the string “IYET”.
- Calls to these subroutines, supporting dimension statements, and global statements are interspersed in the testmain and labeled with the comment “! IYET”.

Figure 5 is a simplified map of a single board IYET testplan. The text in **blue** highlights the locations of additional code added to a standard testmain to support IYET.

Testmain

Dimension statements

Wait for start loop

Error and Break traps

Subroutines

Test_Sections, Reset_Board, Chek_Point_OK, Start_Logging, Get_Serial_Num$, Verify_Board_Names, IllegalChars_In_Board_Name, Create_Log_Queue, Setup_Logging, **Initializations, Print_Startup_Message,** Get_Board_Revision, Do_Version_Inits, Get_Version_Label, Initialize_Constants, **Set_Custom_Options,** Cleanup_Flash, Set_Log_Level

IYET Subroutines

Test Subroutines

Characterize, Preshorts, Shorts, Analog_Tests, TestJet, Connect_Check, Polarity_Check...

Figure 5 IYET Testplan Map

Each of the **blue** items in figure 5 is described in detail below:

• Dimension Statements

IYET requires three arrays to process failure tickets and report re-test history. They are 40 characters wide since that is the limit of the report printer. They are dimensioned at the top of the testplan as follows:

```
! IYET
dim IYET_Failing_Tests$(0:2047)[40]
dim IYET_Retest_Failing_Devices$(0:8191)[40]
dim IYET_Top_Retests$(0:8191,2)[40]
```

• Sub Initializations

IYET is initialized at the end of this subroutine. A global statement is also required.

```
! IYET
global Using_IYET
...
! IYET
if Using_IYET then call IYET_Initialize
Subend
```

• Sub Print_Startup_Message

The following code is used to print the IYET startup message at the end of this subroutine. A global statement is also required.

```
! IYET
global Using_IYET, IYET_Report_On
...
! IYET
O3$="          "
if Using_IYET then
  O3$=O3$&" Intelligent Yield Enhancement Test"
  if IYET_Report_On then O3$=O3$&"-Report"
end if
print O3$
print using "2/"
Subend
```

• Sub Test_Sections (Single Board)

IYET works on test subroutines called from the “Test_Sections” subroutine. Note that the flag “Using_IYET” is used to direct calls the core IYET subroutine for the unpowered test subroutines “Preshorts”, “Shorts”, “Analog_Tests” and “TestJet”. If the flag is not set, this subroutine behaves like a standard testplan.

The "Analog_Tests" call also includes checks to make sure that IYET is not used during learning or sampling. This ensures that re-test data does not corrupt datalogging.

```
! IYET
global Using_IYET, Sampling, IYET_Preshorts_Attempts
global IYET_Shorts_Attempts, IYET_Analog_Tests_Attempts, IYET_TestJet_Attempts

if Using_IYET then
  call IYET (PreshortsMsg$, IYET_Preshorts_Attempts)
else
  call Pre_Shorts (Failed_In_Preshorts, Mode$ & PreshortsMsg$)
end if
if boardfailed then subexit

if Using_IYET then
  call IYET (ShortsMsg$, IYET_Shorts_Attempts)
else
   call Shorts (Failed_In_Shorts, Mode$ & ShortsMsg$)
end if
if boardfailed then subexit

if Logging or not learning then
  if Using_IYET and not learning and not Sampling then
     call IYET (AnalogMsg$, IYET_Analog_Tests_Attempts)
  else
     call Analog_Tests (Failed_In_Analog, Mode$ & AnalogMsg$)
  end if
  if boardfailed then subexit
end if

if Using_IYET then
  call IYET (TestJetMsg$, IYET_TestJet_Attempts)
else
  call TestJet (Failed_In_TestJet, Mode$ & TestJetMsg$)
end if
if boardfailed then subexit
```

• Sub Test_Sections (Panelized Board)

IYET works on test subroutines called from the "Test_Sections" subroutine. Note that the flag "Using_IYET" is used to direct calls the core IYET subroutine for the unpowered test subroutines "Preshorts", "Shorts", "Analog_Tests" and "TestJet". If the flag is not set, this subroutine behaves like a standard testplan.

The "Analog_Tests" call also includes checks to make sure that IYET is not used during learning or sampling. This ensures that re-test data does not corrupt datalogging.

```
! IYET
global Using_IYET, Sampling, IYET_Preshorts_Attempts
global IYET_Shorts_Attempts, IYET_Analog_Tests_Attempts, IYET_TestJet_Attempts

if Using_IYET then
  call IYET (PreshortsMsg$, IYET_Preshorts_Attempts)
else
  call Pre_Shorts (Failed_In_Preshorts, Mode$ & PreshortsMsg$)
end if
if All_Failed then subexit

if Using_IYET then
  call IYET (ShortsMsg$, IYET_Shorts_Attempts)
```

```
Else
   call Shorts (Failed_In_Shorts, Mode$ & ShortsMsg$)
end if
if All_Failed then subexit

if Logging or not learning then
  if Using_IYET and not learning and not Sampling then
     call IYET (AnalogMsg$, IYET_Analog_Tests_Attempts)
  else
     call Analog_Tests (Failed_In_Analog, Mode$ & AnalogMsg$)
  end if
  if All_Failed then subexit
end if

if Using_IYET then
  call IYET (TestJetMsg$, IYET_TestJet_Attempts)
else
  call TestJet (Failed_In_TestJet, Mode$ & TestJetMsg$)
end if
if All_Failed then subexit
```

• Sub Update_Status (Panelized Board)

For panelized boards using IYET, the subroutine “Update_Status”, must be bypassed during the re-test sequence and re-executed at the end of the re-test sequence. This is accomplished by the following code:

```
! IYET
global IYET_Skip_Update_Status

if IYET_Skip_Update_Status then subexit
```

IYET Subroutine Descriptions

This is a high-level description IYET subroutines. In most cases, you will not need to modify these subroutines. They are described here for reference.

• Sub IYET_Initialize

This subroutine is called at the start of the testplan to initialize the paths, files, constants and messages used by IYET. It is called each time the testplan is run.

A directory called "iyet" is created in the board directory if it is not present. This directory will be used to store two files; the current failure ticket ("failure.txt") and a re-test log file ("retest<today's date>.txt"). The re-test log file includes the day's date in its name. The current failure ticket is appended to this file for each re-test sequence. Thus, the re-test log file tracks all failure tickets for a given day. Over time, this directory will fill up with re-test log files. The user can view, copy and delete these files as required.

If the flag "IYET_Report_On" is on, then the subroutine "IYET_Create_Report" is called (see below).

IYET will cycle the vacuum between retests. The delay for vacuum release and actuate are controlled by 2 parameters, "IYET_Vacuum_Off_Delay" and "IYET_Vacuum_On_Delay". These are set to 1.5 seconds and can be adjusted.

IYET will alter the behavior of Chek-Point (the "pins" test). If the testplan flag "Chek_Point_Mode" is set to "Failures", then IYET will set the "Max_Times_To_Fix_Contact" to 1. Since vacuum is cycled by IYET on failure, this prevents additional vacuum cycles when the "pins" test is run. Chek-Point is unchanged if the "Chek_Point_Mode" is "Off" or "Pretest".

• Sub IYET_Create_Report

If the flag "IYET_Report_On" is on, then this subroutine is called at the start of the testplan and displays the top 10 re-tested tests from the current re-test log file. The number of tests displayed is controlled by the "IYET_Max_Report_Count" parameter and can be adjusted. If there are no re-tests or the file does not exist, then a warning message will be displayed.

The re-test log file is named with the current day. Thus, only the re-test history for the current day is displayed. If there are no re-tests or the file does not exist, then a warning message will be displayed.

• Sub IYET_Vacuum_Off, IYET_Vacuum_On

These control vacuum actuation during the IYET re-test sequence. These routines assume that vacuum for the fixture is controlled using the "faon" and "faoff" commands. If you are using the other vacuum commands ("fbon/fboff", "auxconnects/disconnects", custom press or board handler commands, etc.) edit this subroutine to match your fixture set up. If you do edit these routines, be aware that IYET processes the re-test failure data during the delay for vacuum release. In the "IYET_Vacuum_Off" subroutine, the parameter "IYET_Start_Vacuum_Off" is set to the current "msec" count. This is in turn passed to the "IYET_Vacuum_On" subroutine and used to enforce the "IYET_Vacuum_Off_Delay".

The "IYET_Vacuum_On" subroutine will prompt the operator prior to actuating the vacuum during the re-test sequence using the following line of code:

```
question IYET_Cont_Msg$,Continue | if not Continue then stop
```

Do not comment this line unless your fixture is approved for actuation without operator intervention.

• Sub IYET (TestType$, Maximum_Number_Attempts)

This is the core subroutine for IYET. If the flag "Using_IYET" is set, it is called from the testplan subroutine "Test_Sections" in place of the normal calls to "Preshorts", "Shorts", "Analog_Tests", and "TestJet". The "Analog_Tests" call also includes checks to make sure that IYET is not used during learning or sampling. This ensures that re-test data does not corrupt datalogging.

This subroutine does the following:

- Checks that "Maximum_Number_Attempts" is greater than or equal to 1 and rounds to the nearest integer. The variable "Retests_Remaining" is now set to "Maximum_Number_Attempts".
- Turns off logging. It is restored to the correct level prior to the last re-test sequence by calling the subroutine, "Cleanup_Flash".
- Forces buffered reporting during the re-test sequence. If it is off to begin with, at the end of

subroutine the report buffer(s) is(are) dumped and buffered reporting is turned back off.

- If this is a panelized board, the “IYET_Skip_Update_Status” flag is set. This causes the panelized testplan subroutine, “Update_Status” to exit without processing the “Board_Set(*)” array during the re-test sequence.
- The test subroutine (“Preshorts”, “Shorts”, “ Analog_Tests”, “TestJet”) is called for the first time based on the string, “TestType$”.
- The variable, “Retests_Remaining” is decremented.
- The re-test loop is now entered. It does the following:
    - Exits if the initial test subroutine passes.
    - Exits if “Retests_Remaining” = 0.
    - Turns off vacuum, calls “IYET_Get_Failed_Tests”, turns on vacuum.
    - Exits if “IYET_Get_Failed_Tests” returns an “Abort” flag.
    - If “Retests_Remaining” = 1, restores logging by calling “Cleanup_Flash”.
    - Calls “IYET_Retest_Failed_Tests”. This re-tests all the failed tests.
    - Decrements “Retests_Remaining”.
- After the re-test loop, some cleanup is done prior to exiting the subroutine. Some cleanup activities could be redundant:
    - Restore logging by calling “Cleanup_Flash”.
    - If buffered reporting was initially off, dump the report buffer(s) and turn off buffered reporting.
    - For panelized boards, clear the “IYET_Skip_Update_Status” flag and call “Update_Status”.

IYET executes re-tested tests individually. Be sure that the tests in each test subroutine will pass “on their own”. Watch out for gprelay connections, variables, additional “unpowered” statements, or other setups that are not in the test source.

• Sub IYET_Get_Failed_Tests (IYET_Abort)

This subroutine does the following:

- Saves the failure ticket to a failure file and appends it to the re-test log file.
- Parses the failure file to get a list of failing tests in the array, “IYET_Failing_Tests$(*)”
- If there are no failing tests or capacitance compensation needs to be learned, then the “IYET_Abort” flag is set.
- If the “IYET_Abort” is not set, then the report buffer(s) and failure flag(s) are cleared.

•Sub IYET_Retest_Failed_Tests

This subroutine tests all the tests in the array “IYET_Failing_Tests$(*)”. For panelized boards, the “board number is” flag is set.

For more information about Agilent Technologies products, solutions, applications or services in electronics manufacturing, visit our website:
www.agilent.com/see/pcb

You can also contact one of the following centers and ask for a test and measurement sales representative.

United States:
(tel) 1 800 447 8378

Canada:
(tel) 1 800 447 8378
(fax) 1 905 282 6300

Europe:
(Tel) +49 69 9530 7917
(Fax) +49 7031 464 8219

Latin America:
(tel) 011 52 (333) 134 5854
(fax) 011 52 (333) 134 5851

Asia Pacific:
(tel) (65) 6215 8383
(fax) (65) 6822 8636

Data subject to change without notice.

© Agilent Technologies, Inc. 2005
Printed in the U.S.A.
July 15, 2005
5989-3498EN

Agilent Technologies