出國報告（出國類別：洽公）

赴 DRS 廠家參與測試工作見證

服務機關：台灣電力公司核四廠

姓名職稱：顏楨原 電腦維護專員

派赴國家：美　　國

出國期間：95.12.30 至 96.6.11

報告日期：96.7.26

行政院及所屬各機關出國報告提要

出國報告名稱：赴 DRS 廠家參與測試工作見證

頁數 92 含附件：■是□否

出國計畫主辦機關/聯絡人/電話：台灣電力公司 / 陳德隆 / (02)2366-7685

出國人員姓名/服務機關/單位/職稱/電話

顏楨原/台灣電力公司/核四廠/電腦維護專員/（02）24902401＃2939

出國類別：□1 考察□2 進修□3 研究□4 實習■5 其他 (洽公)

出國期間：95.12.30 至 96.6.11　　出國地區：美國

報告日期：96.7.26

分類號/目：儀控工程　ABWR

關鍵詞：儀控工程 核能電廠 安全系統 DRS

內容摘要：

由於 DRS 區塊屬於安全有關系統，其各組件之設計及測試過程對將來整廠安全儀控及運轉極為重要，鑒於目前 DRS 正進行 FID 等多項測試，派員前往 DRS 廠家參與測試工作見證。

赴美洽公期間，獲致以下成果：

1. 瞭解一號機 FID 之建置及測試狀況
2. 瞭解 VDU 相關軟硬體測試情形
3. 瞭解 DRS 設備在工地之安裝、測試之相關準備工作
4. 研讀 DRS 廠家提供之 Plus32 O&M manual，做為將來本廠之維護訓練的基本教材。

本文電子檔已傳至出國報告資訊網（http：//report.gsn.gov.tw）

目　　錄

壹、國外公務之內容與過程

一、目的

(一) 出國任務

赴負責發展核四廠安全系統之廠家 DRS 參與測試工作見證，出國期間自九十五年十二月三十日至九十六年六月十一日，共計 164 天。

(二) 緣起及目標

1. 核四廠分散式控制暨資訊系統(DCIS)由奇異公司負責設計核島區儀控系統,其中特殊安全設施(SSLC/ESF)（主要包含洩漏偵測隔離系統及緊急爐心冷卻系統)由奇異公司設計，並交由其次包商 DRS 廠家製造系統設備，來執行全廠之保護、控制及顯示功能；鑒於目前 DRS 正進行 FID 等多項測試，派員前往 DRS 廠家參與測試工作見證；目標如下：

   (1) 瞭解一號機 FID 之建置及測試狀況

   (2) 瞭解 VDU 相關軟硬體測試情形

   (3) 瞭解 DRS 設備在工地之安裝、測試之相關準備工作

   (4) 研讀 DRS 廠家提供之 Plus32 O&M manual，做為將來本廠之維護訓練的基本教材。

二、過程

奉派至美國 DRS 廠家(Danbury, CT) 參與測試工作見證，為期 164 天，詳細過程及工作內容如下表：

起始日	迄止日	地點	工作內容
951230	951231		往程(台北-紐約-Danbury)
960101	960608	Danbury	參與測試工作見證
960609	960611		返程（Danbury-紐約-台北）

在 DRS 廠家執行出國任務期間，瞭解各項測試進度及相關準備工作，並針對 DRS 設備未來在工地安裝、測試之技術問題進行討論。另要求廠家提供 O＆M manual，做為將來本廠之維護訓練的基本教材。

貳、執行過程與內容

一、 PLµS32 控制系統

1.系統介紹

Plµs32 控制系統是以 PERFORMNET（Performance Enhanced Redundant Fiber Optical Replicated Memory Network）為通訊網路主幹，配合各軟硬體組件達成通訊與控制功能。

- 硬體主要組件包括:
  - PLµS32 機櫃

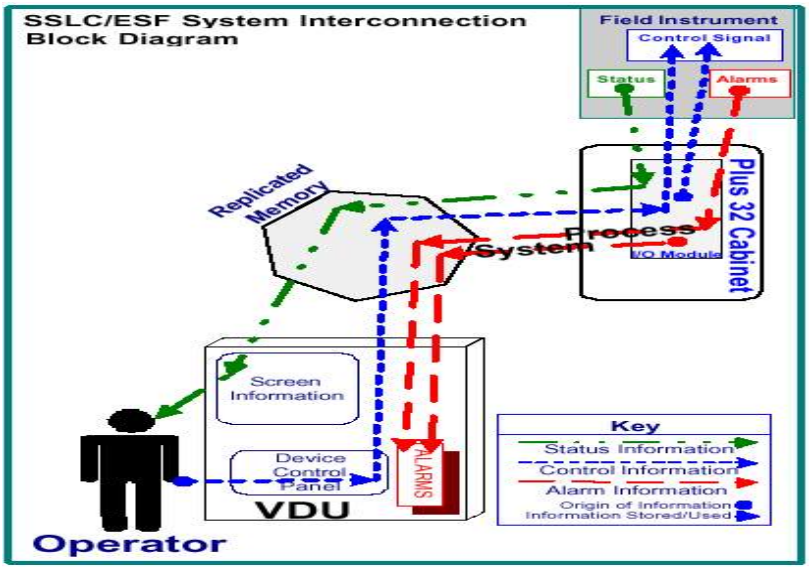
    負責訊號輸出入及系統控制。
  - VDU 人機介面工作站

    提供運轉員可以操控 PLµS32 機櫃內的輸出入控制模組的平台。
  - Communication Interface Module (CIM)

    負責安全系統各區間及與其他安全儀控系統(如 NUMAC)通訊。
  - Bridge Transfer Module (BTM)

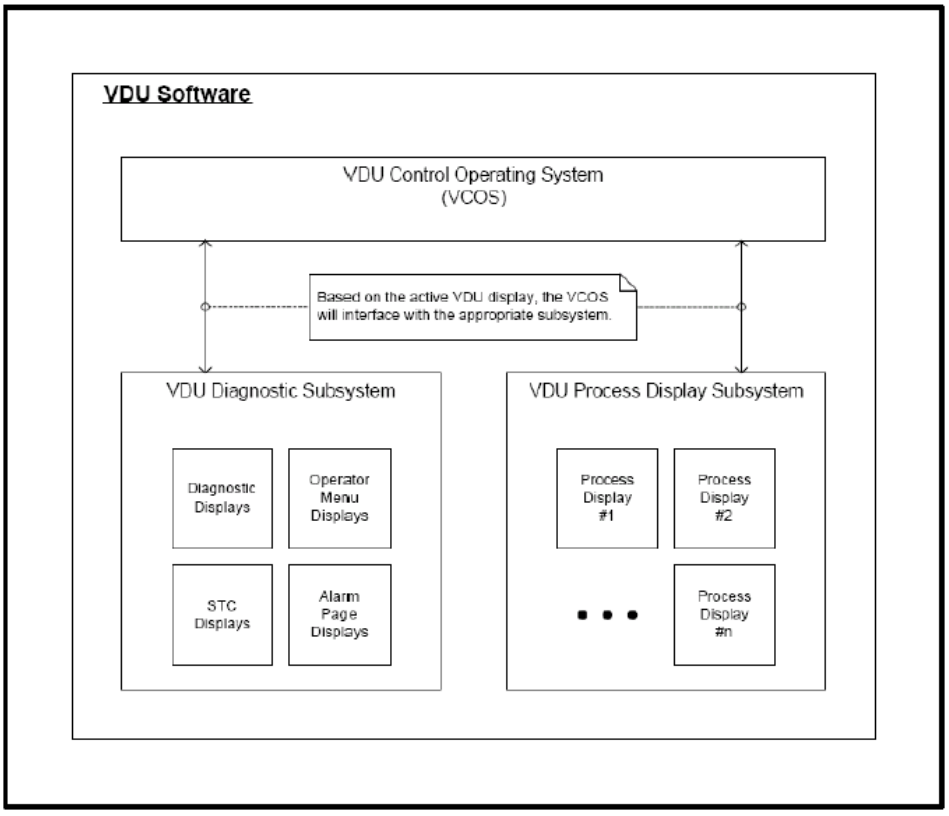
    負責與非安全儀控系統通訊 。

各主要設備之間的通訊傳輸方式如下圖所示。



- 軟體主要組件包括:
  - FID(Functional Interconnect Diagram)設計軟體
    - Control Algorithm Software
    - Control Operating System (COS) Software
    - Functional Interconnect Diagram (FID) Compiler
    - Network Interface Module (NIM) Software
    - Communications Interface Module (CIM) Software
  - VDU 軟體
    - VDU Control Operating System (VCOS)
    - VDU Diagnostic Subsystem
    - VDU Process Display Subsystem

VDU 之軟體架構如下圖所示。



2.PERFORMNET 網路

Plus 32 PERFORMNET 網路，為一真正 real-time 通訊網路，replicated memory 觀念類似共用記憶體系統，所有資料均存於一 memory bank，並複製至網路中每一 node，其為一完全雙重獨立、隔離之網路，單一故障下軟體會自動使用另一網路，且對反應時間或控制演算式不造成影響。

每一個 PERFORMNET 上最多可存在 128 nodes，每個 node 都有一個 node number(0~127)，目前龍門設計的 node number 分配請參考附件一，每個 node 有自己的 4Kbyte 記憶體可運用(可寫入)，故組成 512Kbyte。512Kbyte 的記憶體對每個 node 而言都是可讀取。(核四廠總共有 9 個 PERFORMNET)

每個 node 所擁有記憶體地址，是依 node number 依序排列下來，例如: node 0 (0~4095)，node 1 (4096~8191)...，基本上每一個 node 只使用前 3Kbyte 記憶體，後 1Kbyte 做為 error message 使用。

2.1 SSLC Cabinet or RMU Cabinet 計算方式

利用 NIM 卡(有 DIP SW 可設定 node number)與 PERFORMNET 其他 nodes 通訊，盤面上有 48 slots 可插 48 片 control I/O modules，每一片 control I/O module 分配 64 byte 作為 FID 圖使用，其配置也是依照 slot 順序(1~48)依序排列，FID 圖上之 Digital/Analog Interlock 有標明 offset(1~64)來表示所使用之區間，依上述原則，即可計算出 FID 圖上 Digital/Analog Interlock 使用到的絕對位址，公式簡述如下：

PERFORMNET Address=[Node * 4096]+[(Slot-1)*64]+[Interlock offset-1]

Node={0,1,2...127}

Slot ={1,2,3...48}

Interlock offset={1,2,3...64} for Digital Interlock(因為佔 1 byte)

={1,5,9...61} for Analog Interlock(因為佔 3 byte)

2.2 CIM 計算方式

CIM(有 DIP SW 可設定 node number)是插在 SSLC(Div.1~4)或 RMU(Div.0)盤內，負責與其他 Division 之資料交換(2 條單向傳輸)、接受 RTIF/NMS 之資料輸入(單向輸入)或輸出至 Division 0(單向輸出)；其可寫入之前 3Kbyte 記憶體，可在 FID 圖上看到 xCIMyy-zzzz 的 input interlock 或 output interlock，其中：

xCIMyy-zzzz

x 代表機組{0,1,2}

yy 代表 CIM number{11,12,21,22,31,32,41,42,51,52 }

zzzz 代表 Interlock offset{1~3072}

PERFORMNET Address=[Node * 4096]+[Interlock offset-1]

2.3 BTM 計算方式

BTM(有 DIP SW 可設定 node number)是插在 SSLC(Div.1~4)或 RMU(Div. 0)盤內，FPGA 負責將記憶體(512KByte)做全體複製(PERFORMNET side to SCRAMNET side)，其擁有之 4Kbyte 記憶體僅使用最後 1Kbyte 儲存本身之 error message，其餘 3Kbyte 並不寫入其他資料或進行資訊交換，記憶體之配置同 CIM。

2.4 VDU 計算方式

VDU 使用之記憶體空間是 node 0~3(共有 16 Kbyte)，所以在 FID 圖上 input interlock 標示有 VDU 及 offset 其對應記憶體的地址為：

PERFORMNET Address= [Interlock offset-1]

Interlock offset:

1~3072	(node 0)
4097~7168	(node 1)
8193~11264	(node 2)
12289~15360	(node 3)

3.PLμS32 機櫃

主要分三種機櫃:SSLC/TEST 機櫃、RMU 機櫃及 Display 機櫃，SSLC/TEST 機櫃主要是放置在控制室背盤，主要執行控制功能；而 RMU 機櫃主要是在放置在現場，主要執行訊號收集及輸出功能；Display 機櫃主要是在放置在輔助燃料廠房，做為現場顯示及控制用（另外盤 0H23-PL-2305 亦包含備用柴油機控制功能）。

SSLC/Test 及 RMU 機櫃，其上方右邊有兩張 NIM(Network Interface Module)卡作為與 PERFORMNET 通訊，每一機櫃是由 NIM、CIM、BTM、輸出入及控制模組和雙重電源供應器組成，機櫃間以雙重之 NIM 通訊，同一機櫃內控制模組以 Backplane 通訊，每 20 ms 掃描所有輸入點一次，以執行控制功能及更新輸出，模組反應時間從輸入至輸出為小於 50 ms，控制輸出入模組含線上偵測，結果顯示在人機介面工作站。

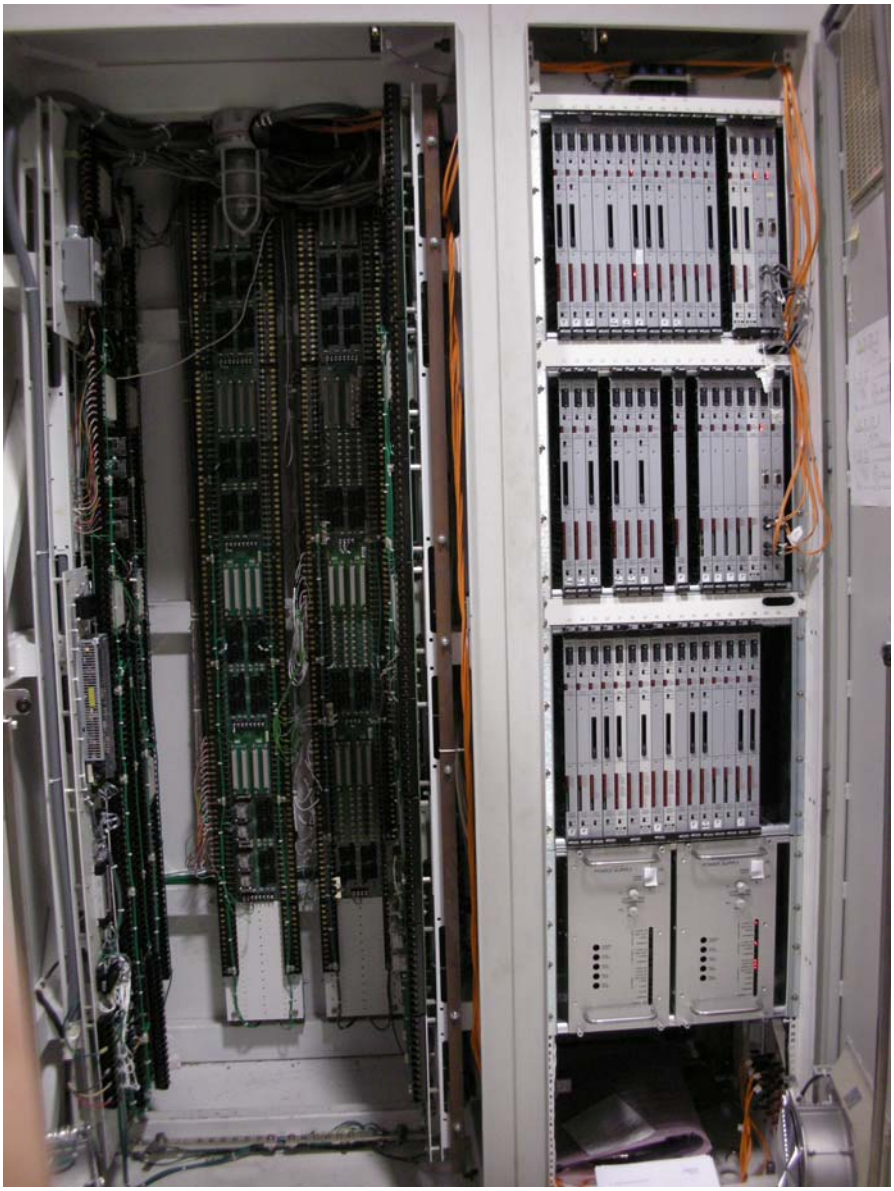
3.1 SSLC/TEST 機櫃

- SSLC
  - 機櫃內右上方兩張 NIM、右中兩張 CIM、右下層兩張 BTM 及兩張 CIM 與 PERFORMNET 通訊。
  - 其餘位置為控制模組(ACM、DCM1)。
  - 最下層為雙重電源供應器。
  - 機櫃間以雙重之 NIM 通訊，同一機櫃內控制模組以背板 RS-485 通訊(LP 匯流排)。
- TEST
  - 從上而下為 SDG control(暫放)、MCM (Media Conversion Module, 連到 SSLC 之 CIM)、Quad switch、STC、EPG/SAG、電源供應器。

下圖為 SSLC/TEST 機櫃

3.2 RMU 機櫃

- Logic Cabinet
  - 右上方兩張 NIM 與 PERFORMNET 通訊。
  - 餘位置為控制、輸出入模組(AIM、DCM3、AOM、RTD、T/C)。
  - 最下層為雙重電源供應器。
- Terminator Cabinet
  - 為端子板，接現場信號。



3.3 Display 機櫃

- 盤 0H23-PL-2305

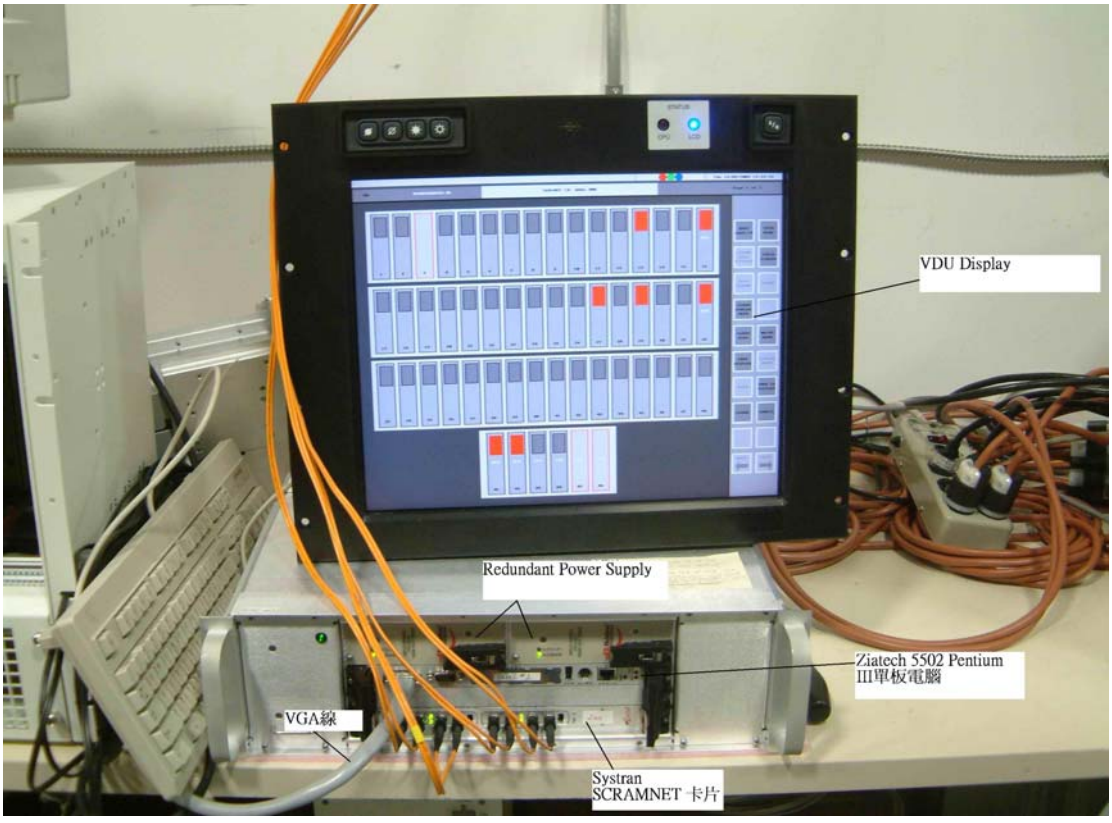
  上方為一 VDU，下方為備用柴油機控制盤。

4.VDU 人機介面工作站

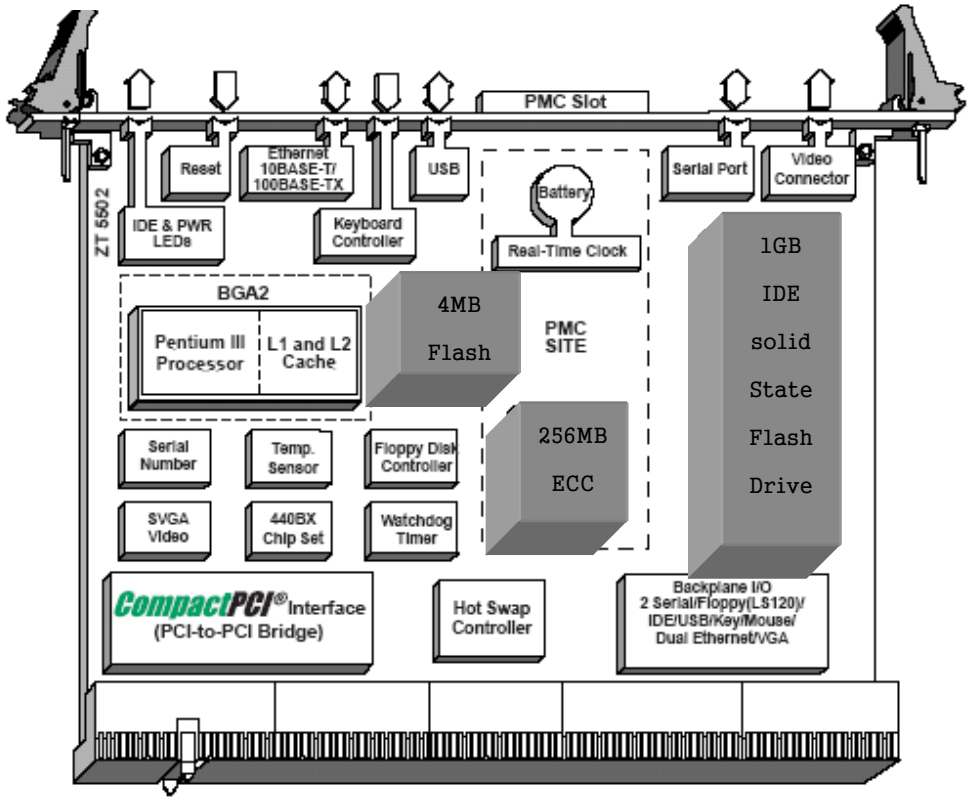
VDU 人機介面工作站可分為: VDU 顯示器及 VDU 顯示控制器兩組件。

- VDU 顯示器：為平板液晶螢幕、觸控螢幕面板及喇叭，觸控螢幕面板為唯一的使用者輸入元件。
- VDU 顯示控制器:匯流排界面為 CompactPCI®界面，由下列組件所組成：
  - Ziatech 5502 Pentium III 工業級單板電腦
  - 兩片 SYSTRAN PMC(PCI Mezzanine Card)之 SCRAMNET 卡裝在一塊 CompactPCI® 2-Slot PMC Carrier Board 上

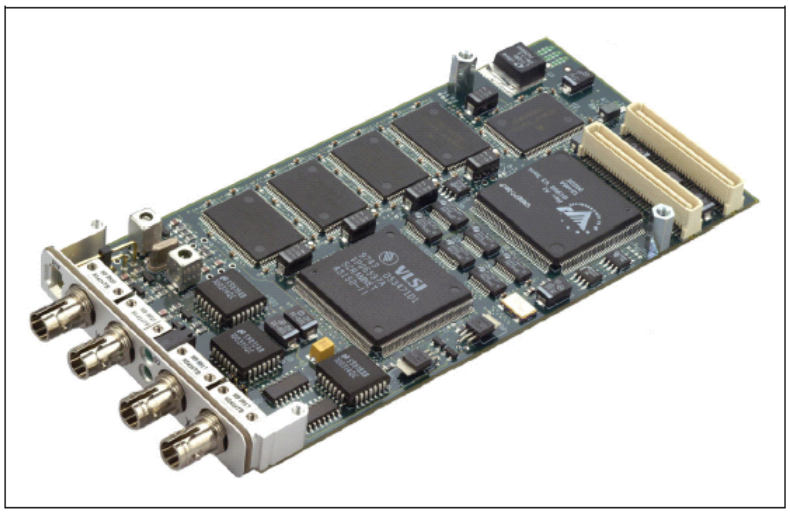
下圖為 VDU 人機介面工作站



下圖為 Ziatech 5502 工業級單板電腦



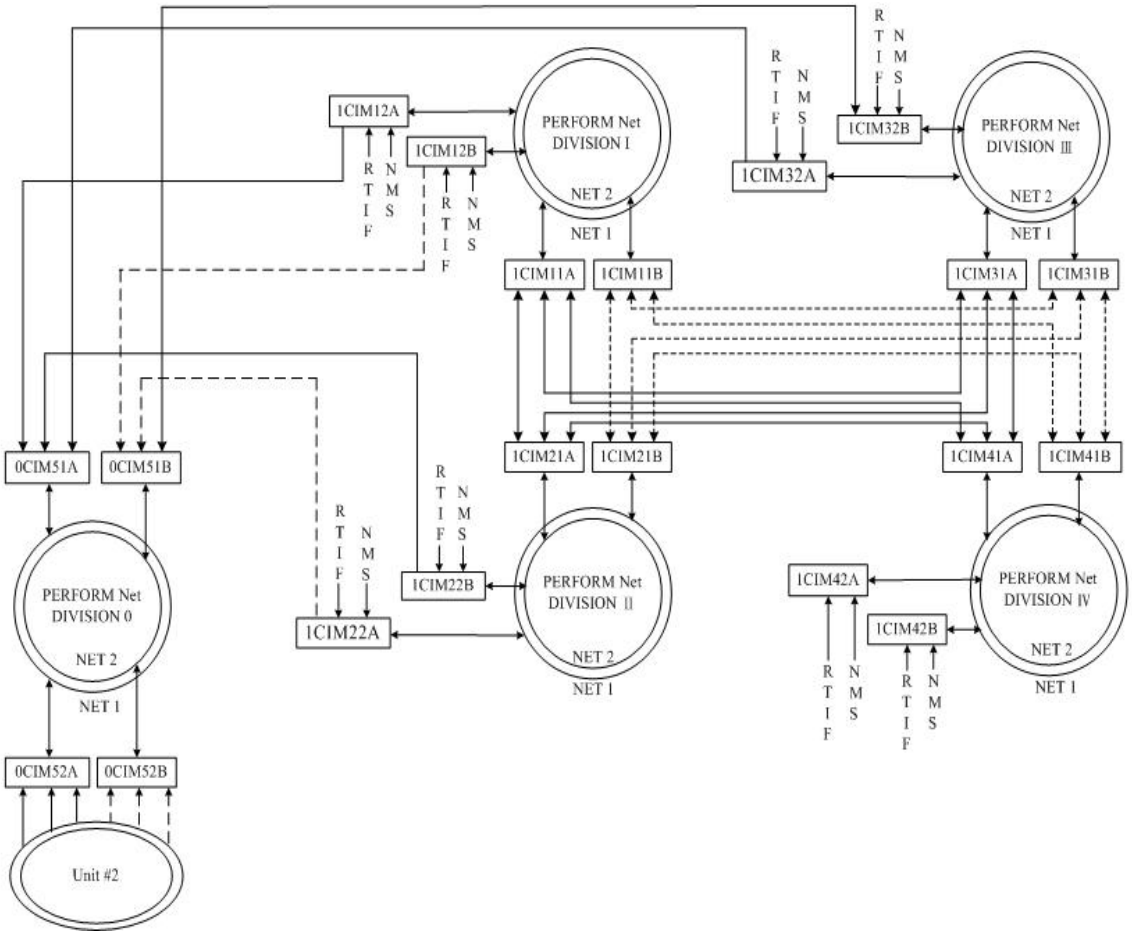
下圖為 SYSTRAN PMC SCRAMNET



5.CIM

- 模組特性大致與 NIM 相同。
- 有三個 RS-485 port 與 NUMAC 設備及其他 DIV 相連
  - 每個 RS-485 port 傳輸與接收分開。
  - DIV 1/2/3/4 之間雙向傳輸。
  - DIV 1/2/3/4 與 NUMAC 單向接收。
  - DIV 1/2/3 與 DIV 0 單向傳輸。
  - 每 500ms 接受 NUMAC 設備(NMS、RTIF)訊號。
- CIM(NUMAC)→MCM(NUMAC)~MCM(TEST)→CIM(SSLC)

下圖為安全相關系統網路架構



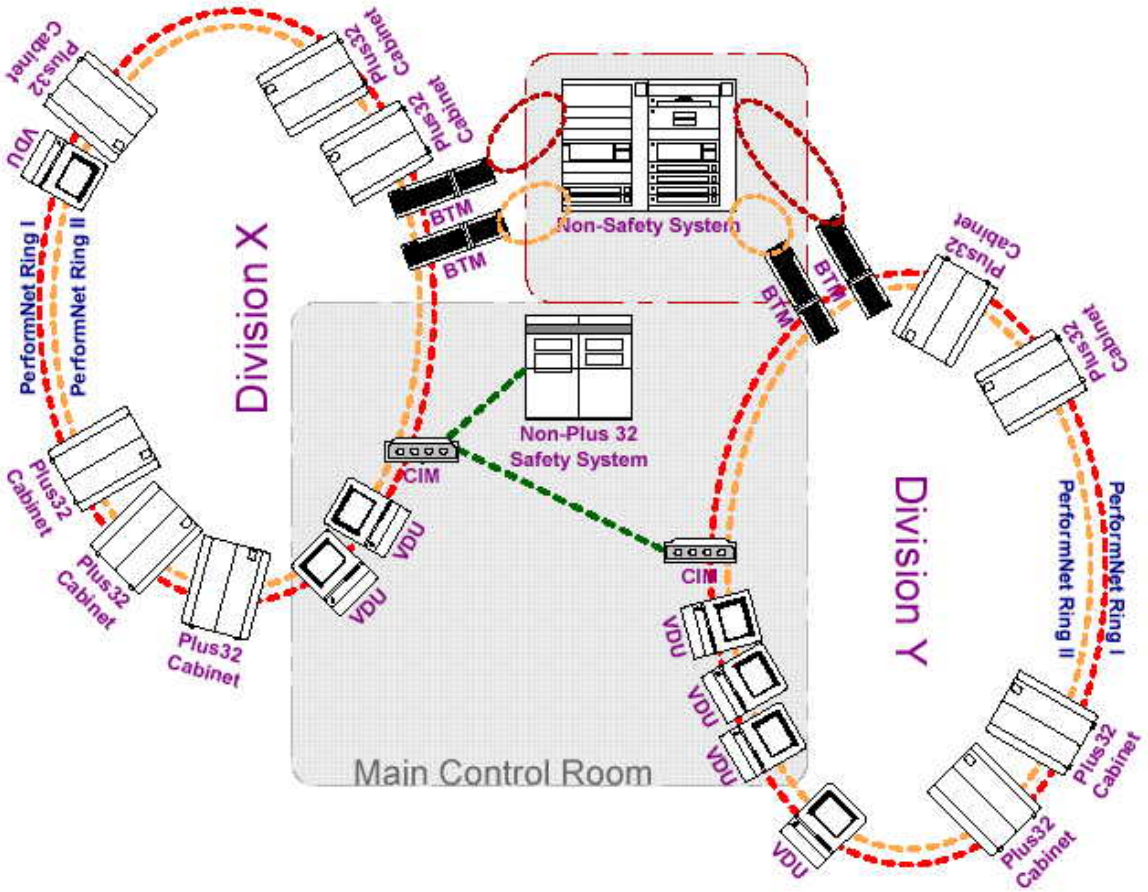
6.BTM

BTM 主要用途為安全與非安全網路之間傳送資料，在安全網路稱為 PERFORMNET，而非安全網路稱為 SCRAMNET。

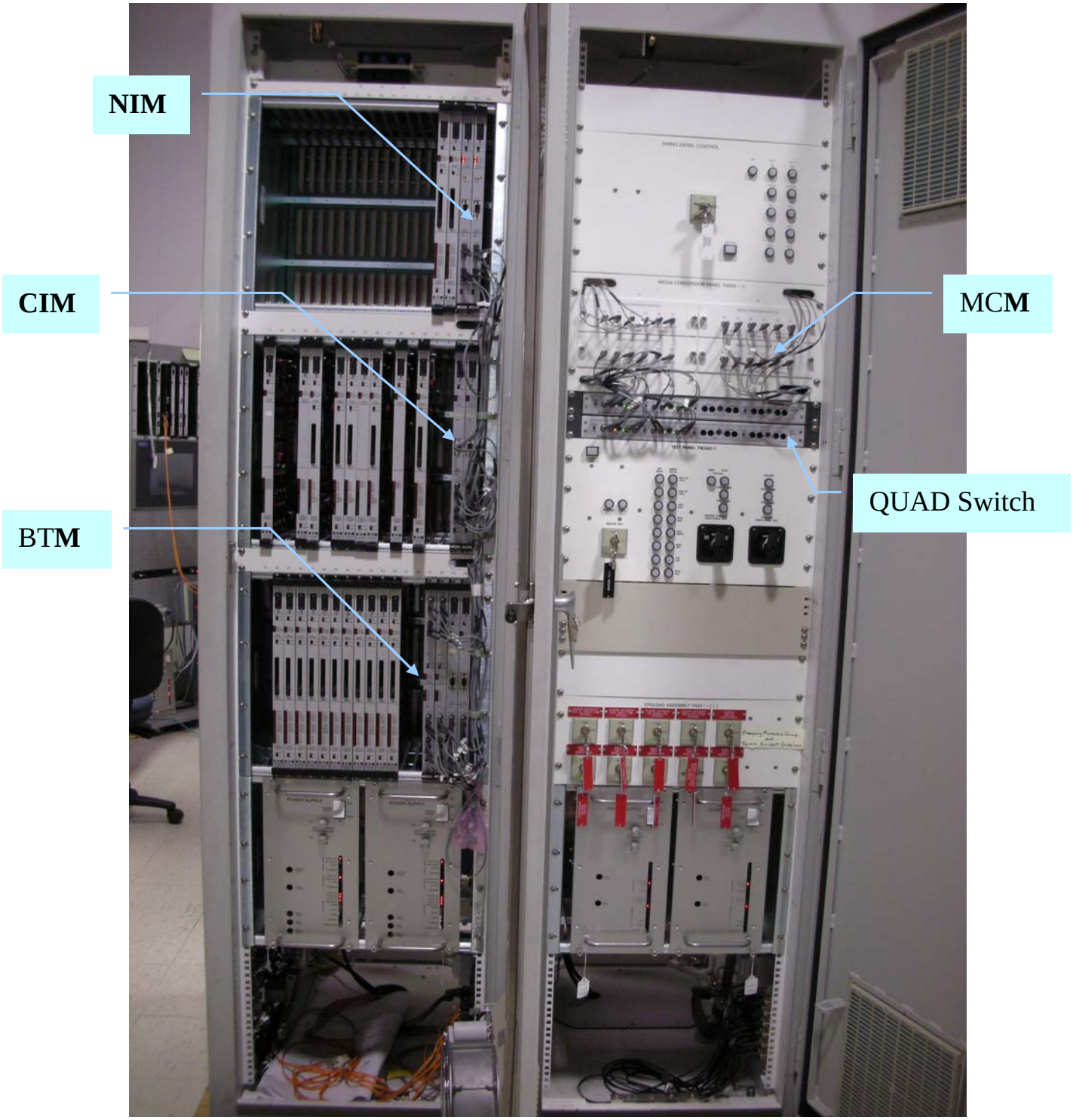
BTM 是包含 A、B 兩個網路通訊區，A 通訊區與安全網路通訊，B 通訊區與非安全網路通訊。

基本運作是由 A 通訊區收到 PERFORMNET 資料，它將寫到其記憶體，然後再轉移到 B 通訊區的記憶體，最後 B 通訊區對 SCRAMNET 網路發送。

下圖為安全相關系統各 Division 與非安全相關系統連結示意圖



下圖為網路介面說明



二、 VDU 軟體設計

1.介紹

VDU 軟體設計，可分為 3 個子系統

- VDU 作業系統(VDU Control Operating System, VCOS)
- VDU 診斷子系統(VDU Diagnostic Subsystem)
- VDU 流程顯示子系統(VDU Process Display Subsystem)

這 3 個子系統獨立運作，但彼此互動以符合 VDU 之軟體需求，其互相關係請參考附件二，其功能將簡述如下：

2.VDU 作業系統

此作業系統的軟體分兩部份：核心(Kernel)及作業系統(OS)，主要功能是在 VDU POWER-UP 或 RESTART 時，對 VDU 畫面控制器(Display Controller)之硬體執行起始化(initialize)與介面控制(Interface)，並與 VDU 診斷子系統及流程顯示子系統互動以更新畫面在最新狀態。

3.VDU 診斷子系統

診斷子系統主要是顯示 DRS 的診斷畫面，大部份的畫面由 DRS 來定義，以提供 DRS PLμS32 之設備及組件(NIM、CIM、BTM、I/O modules、VDU)的詳細狀態，另外有關 VDU 組態設定(Configuration)畫面、觸控螢幕校正(Touch Screen Calibration)、運轉員選單(Operator Menu)、清潔螢幕(Clean Screen)、偵測試驗控制器(STC)、警報(Alarm)等畫面，也在此子系統下執行。

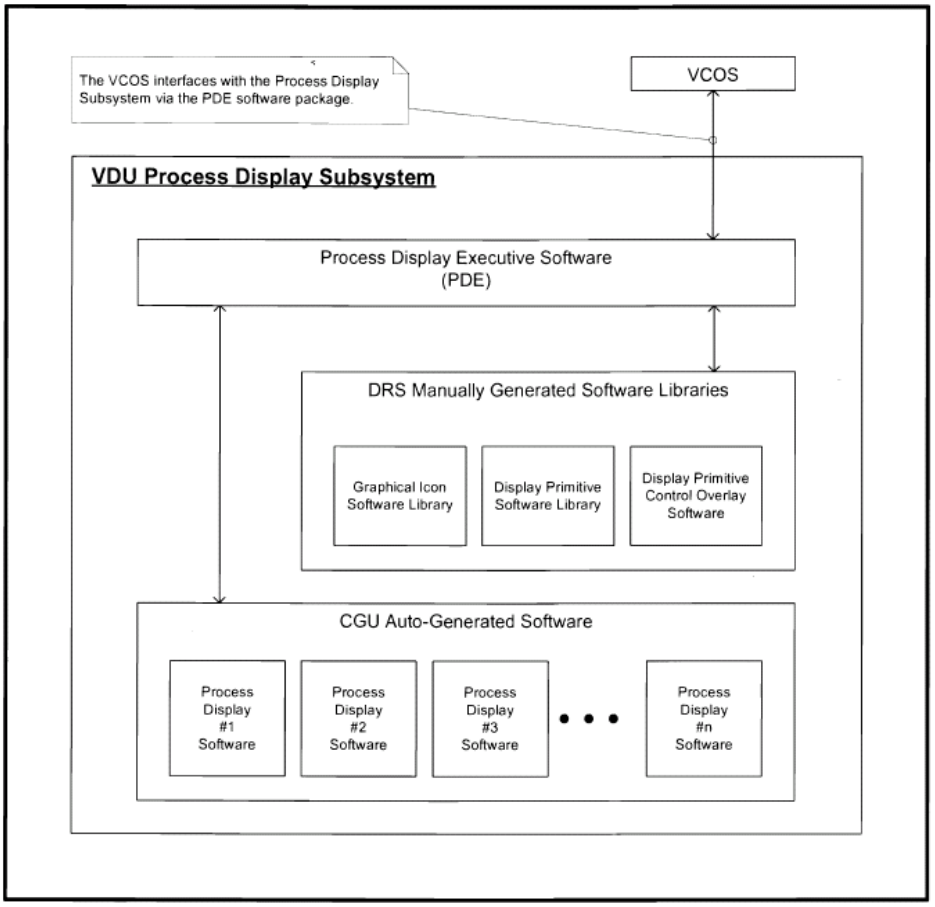
4.VDU 流程顯示子系統

流程顯示子系統是負責顯示 GE 所提供之流程畫面。流程畫面是用來監視及控制現場設備，畫面中分為靜態組件(Static Component)及動態組件(Dynamic Component)，而動態組件會改變顯示或/且可以選擇加以控制，而靜態組件則否。

流程顯示子系統是由：流程顯示執行軟體(Process Display Executive Software)、CGU 自動產生軟體(CGU Automatically generated software)及 DRS 人工產生軟體(DRS manually generated software)組合而成。DRS 人工產生軟體又可分為以下幾種軟體：

1. Display Primitive Software Library(DP)
2. Display Primitive Control Overlay Software(CO)
3. Graphical Icon Software Library(GIL)

流程顯示子系統之軟體架構如下圖所示。



三、 FID 測試進度

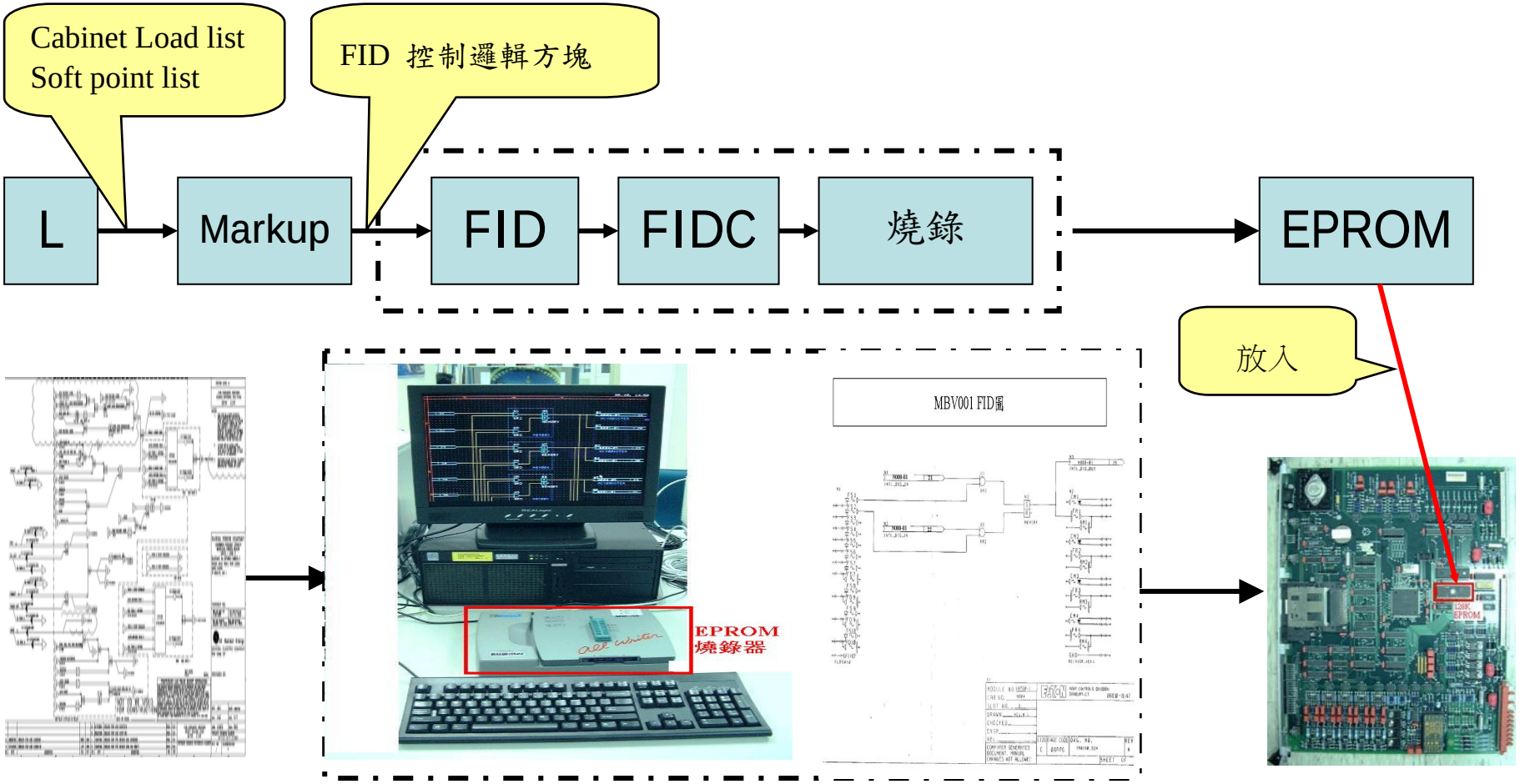
FID 乃 DRS 根據邏輯圖，I/O Database，DCT 等多項資料庫產生而成，主要做為現場設備邏輯處理與控制用。

DRSPLμS32 控制系統已經將許多控制邏輯方塊設計為標準的 FID (ORCAD) 邏輯方塊，而每一 FID 邏輯方塊應用於控制模組需依據 FID 控制邏輯設計法則及基準，才能使 FID 邏輯方塊正確使用於控制模組。

實際 Implement 各 Module FID 時,係將此已經定義好的 Template 先叫出來後,再增加或刪減必要元件,成為真正的 FID 圖，實際的系統 FID 圖號是以盤面名稱加上 MODULE 的位置號碼來表示，如 1H12PL1109A-31, 1H23PL0301A-05 等。各模組類別如下:

Module Type	Template Identifier
6N760-1	DCM for digital control module with both control and field
6N761-1	DCM for digital control module with control section only
6N762-1	DCM for digital control module with field section only
6N763-1	DOM for digital output module
6N765-1	ACM for analog control module
6N766-1	AIM for analog input module
6N768-1	AOM for analog output module
6N769-1	T/C for T/C input module
6N770-1	RTD/0-2Kfor RTD/0-2K input module

下圖為 FID 製作示意圖



在職本次任務期間，DRS 執行包含 3rd 與 4th round FID 修改及測試及測試工作，在這之中常因 GE 提供之資料不完整，造成 FID 功能不正確或必須多次修改，另因 DRS 本身文件處理流程繁瑣，致使 FID 測試進度有所延誤。

截至職返國前 3rd Round 之 302 張 FID 與 4th round 之 100 張 FID 已全部測完測試完成(附件二、三) ，FID 測試過程與進度說明請參照附件十一。

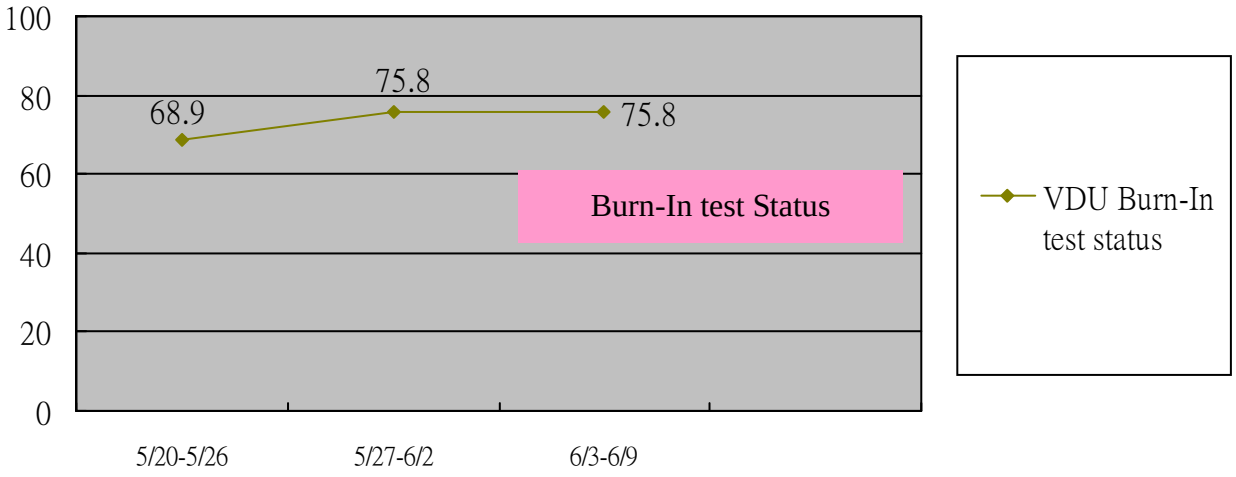
四、 VDU 軟硬體測試

1. VDU Software Unit Test

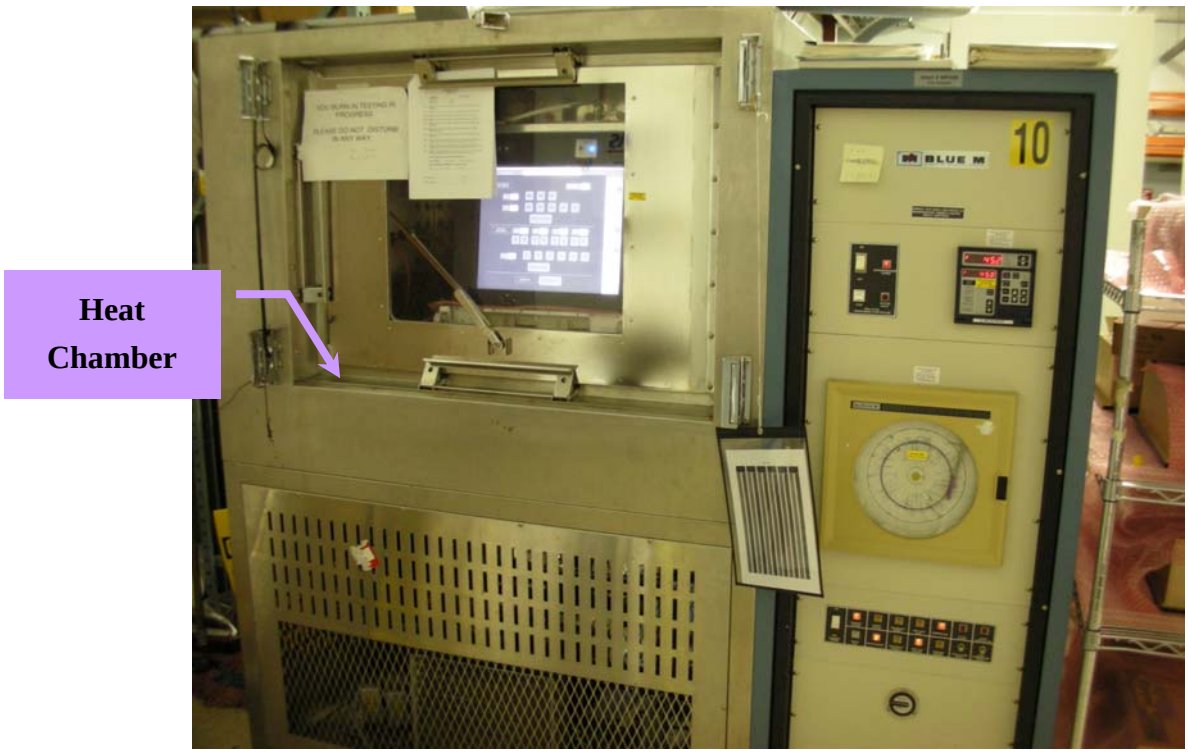
Unit test 為在純軟體的環境下，針對 VDU 各 Group 相關軟體之撰寫及其正確性進行驗證，截至職返國前測試仍持續進行，測試範例如附件四（以 ABV-1 Control Overlay 為例）。

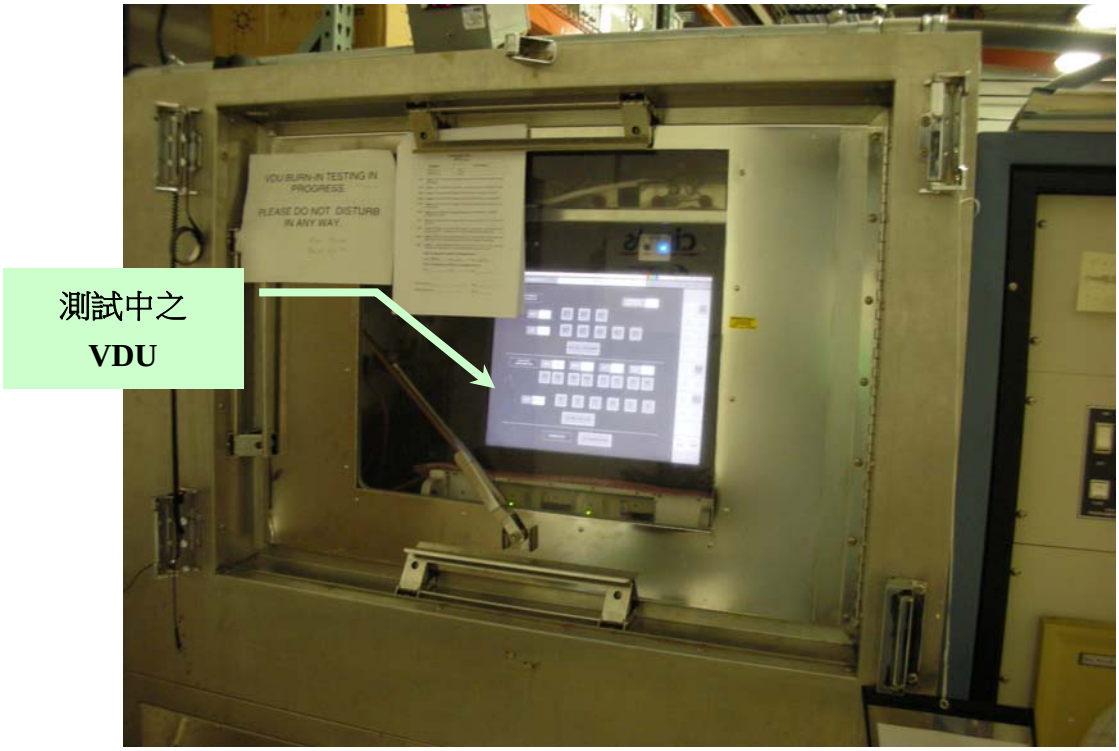
2. VDU Burn-in Test

Bur-In test 為將待測物放置於一溫度為 104°F（±5°F）的 heat chamber 中，並持續對待測物送電達 24 小時（以上）後，再觀察待測物功能是否正常的一項測試，截至職返國前完成進度如下：



下圖為 VDU Burn-in test 說明

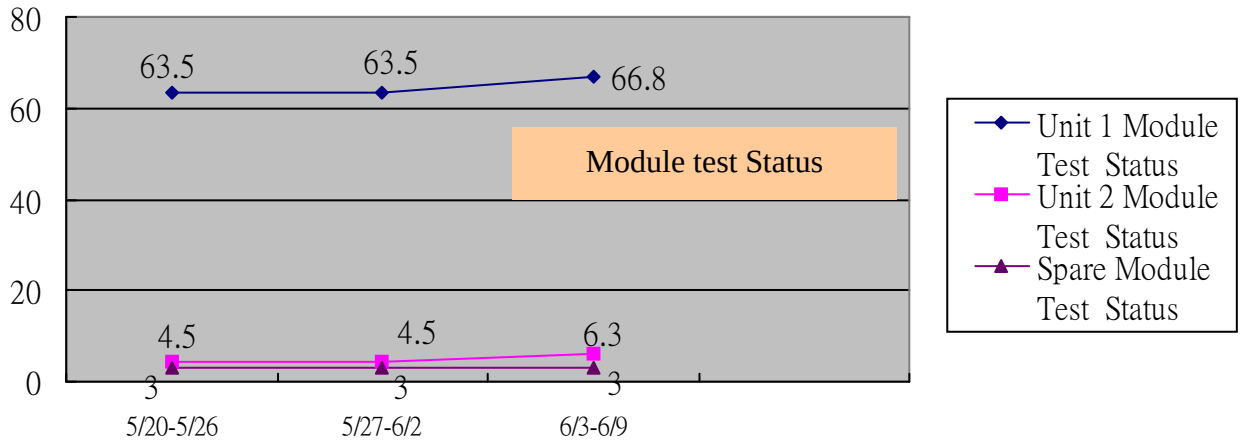




五、其他測試進度

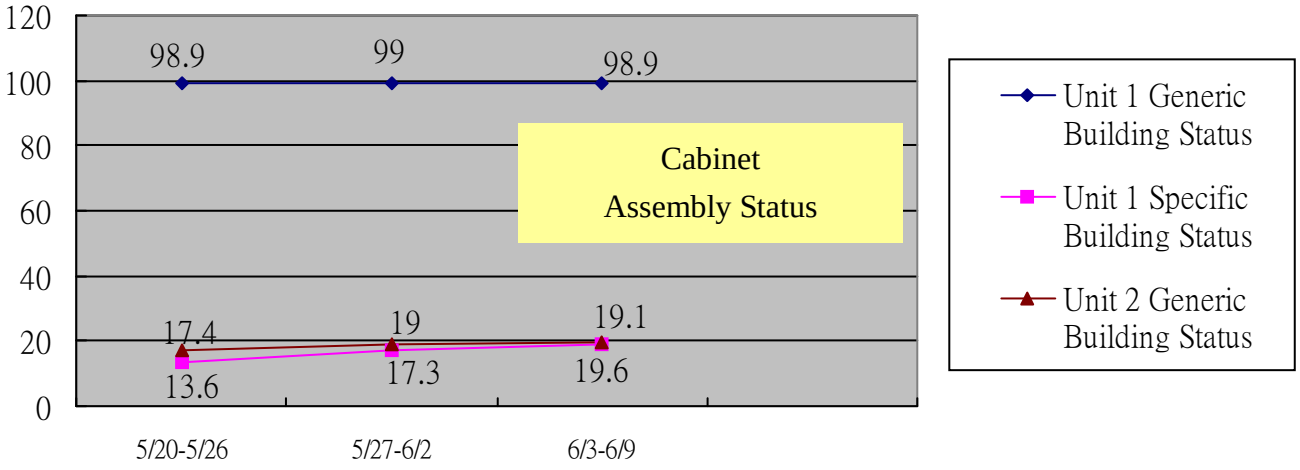
1. Module Test

Network Module 及 I/O Module Test 原本為待 Generic Assembly 完成後，執行 Acceptance Test 時方會進行之測試項目。其主要內容為將各種測試用之 PROM 放置於 module 上，配合 DRS 測試設備，送給 module 不同信號，驗證 module 上之 LED 燈號顯示是否正常，進而判定 module 功能是否正常之一項測試（邏輯測試於 FID Test 執行），截至職返國前 module test 進度如下：



2.SSLC/RMU Cabinet Generic/Specific Assembly

SSLC/RMU Cabinet Generic Assembly 為不含 Module 之盤體組裝工作，截至職返國前 cabinet assembly 進度如下：



六、技術問題討論

1. Skew Test

DRS PERFORMNET 在傳送與接收都分別用兩芯光纖來達到 150Mbps 的傳輸頻寬，但在經過長距離的佈線後，會產生兩芯光纖不等長的現象，此光纖不等長的狀況會造成信號由傳送端送出後抵達接收端時間的不一致，此一時間的落差即所謂 Skew Delay，而 Skew Delay 所造成的影響，輕微會使的網路效能變差，若 delay 過大，更可能會造成網路不通。雖然 DRS PERFORMNET 之設計可以允許每二芯光纖間可有小於 8” 差距，但在 PERFORMNET 佈線完成後，仍必須進行 Skew Test 來確保 PERFORMENT 可以正常工作，另外適當的補償也可以使的 PERFORMNET 的效能更為理想化。一般除了可以用 Skew meter 來進行 skew test 外，DRS 亦提供一種以手動方式進行之測試及調整方法，說明如下：

- Network Skew Test Tools

QTY	Part Number	Description
1	N/A	8” Fiber Optic Patch Cable
1	N/A	10” Fiber Optic Patch Cable
1	N/A	12” Fiber Optic Patch Cable
1	N/A	14” Fiber Optic Patch Cable
1	N/A	16” Fiber Optic Patch Cable
2	N/A	45” Fiber Optic Patch Cable
4	KH113A (AMP P/N:504021-1)	Fiber Optic Coupling (ST-ST)

- Network Skew Testing Procedure

本測試主要是利用連接光纖兩端之 Network Module(NIM/CIM/BTM)上的指示燈號作為測試結果之判定，再搭配測試工具，決定該被補償的光纖及應補償之長度，以使得網路效能達到最佳化。另因測試主要針對每一個 Division ring 作測試，所以在測試前須將所有的 Quad Switch 設定為 isolate mode(如附件五)，使得 AFB 及 SWPH 自成一獨立的小型 ring，以利測試。

A. Network Skew Testing Initial Setup

1. 檢查所有的 Network Module 是否已安裝在適當的 card rack 上。
2. 確認 QUAD Switch 設定在 isolate mode。
3. 將要用作測試結果指示之 Network Module 上之 reset switch 切換至”RESET”位置(附件六)，將待測的光纖接至 Rx port（另一端應接於另一個 Network Module 之 Tx port）。
4. 將 Network Module 上之 reset switch 切換至”NORMAL”位置。
5. 若”NET FAULT LED”亮起，則執行”B. Skew Testing, Net Fault Exists”相關步驟,若”NET FAULT LED”沒亮,則執行”C. Skew Testing, Net Fault Does Not Exist”相關步驟。

B. Skew Testing, Net Fault Exists

1. 將要用作測試結果指示之 Network Module 上之 reset switch 切換至”RESET”位置。
2. 拔除與此 Network Module 連接之光纖。
3. 將一 45”之 dual fiber patch cable 以 loop back 的方式連接到此 Network Module 上（一端接於 Tx，一端接於 Rx）。
4. 將 reset switch 切換至”NORMAL”。
5. 若”NET FAULT LED”亮起，表示此 Network Module 故障，更換另一組相同的 Network Module 後，從 A.3 步驟開始執行，若燈沒亮，則繼續 B.6 步驟。
6. 將 Tx 端光纖接上，並選定 Rx 端之其中一芯光纖（cable A）作為測試及補償，依照附件七逐段增加 fiber optic patch 之長度。
7. 在替換 patch 長度及測試過程中，必須重複將 reset switch 切換至”RESET”及”NORMAL”。
8. 在每次切換回”NORMAL 後，需等待約 30 秒，再將”NET FAULT LED”之狀態記錄於附件七。
9. 測試需等到附件七之各 patch 長度組合都替換完畢，或”NET FAULT LED”燈號清除後為止。
10. 如果”NET FAULT LED”燈號被清除了，則依附件七繼續替換 patch，直到”NET FAULT LED”燈號再次亮起為止，並記錄其值。
11. 若”NET FAULT LED”燈號於步驟 6～10 被清除後又亮起，則依”D. Skew

Correction, NET FAULT Existed”步驟計算該補償之 fiber optic 長度。

12. 若在完成附件七之表格後”NET FAULT LED”燈號仍未被清除，則表示步驟 6 進行補償之光纖選定錯誤，需選定另一芯光纖（cable B）後，再依附件八重複步驟 6～11，直到”NET FAULT LED”燈號清除又亮起後，依”D. Skew Correction, NET FAULT Existed”步驟計算該補償之 fiber optic 長度。

C. Skew Testing, Net Fault Does Not Exist

1. 選定一芯光纖（cable A）作為測試及補償，按附件七逐段增加 fiber optic patch 之長度。
2. 在替換 patch 長度及測試過程中，必須重複將 reset switch 切換至”RESET”及”NORMAL”，在每次切換回”NORMAL 後，需等待約 30 秒，再將”NET FAULT LED”之狀態記錄於附件七。
3. cable A 測試需等到”NET FAULT LED”燈號亮起後為止。
4. 再選定另一芯光纖（cable B），依附件八重複步驟 1～3 並記錄其值。
5. 直到 cable B 測試之”NET FAULT LED”燈號亮起後，依”E. Skew Correction, NET FAULT Did Not Exist”步驟計算該補償之 fiber optic 長度。

D. Skew Correction, NET FAULT Existed

1. 該補償的長度為第一段 pass 的 patch 長度與最後一段 pass 的 patch 長度之平均值。
2. 以下例子說明其計算方式：

```
Add 88”  to  Cable A: NET FAULT = FAIL
Add 90”  to  Cable A: NET FAULT = PASS
Add 92”  to  Cable A: NET FAULT = PASS
    :
Add 114”  to  Cable A: NET FAULT = PASS
Add 116”  to  Cable A: NET FAULT = PASS
Add 118”  to  Cable A: NET FAULT = FAIL

First Pass Length = 90” ; Last Pass Length = 116”
Correction = ((90” +116” )/2 = 103”
Correction = Cable A 加上 103” 或 Cable B 減 103”
```

E. Skew Correction, NET FAULT Did Not Existed

1. 該補償的長度為 Cable A 最後一段 pass 的 patch 長度與 Cable B 最後一段 Pass 的 patch 長度之差值平均。
2. 以下例子說明其計算方式：

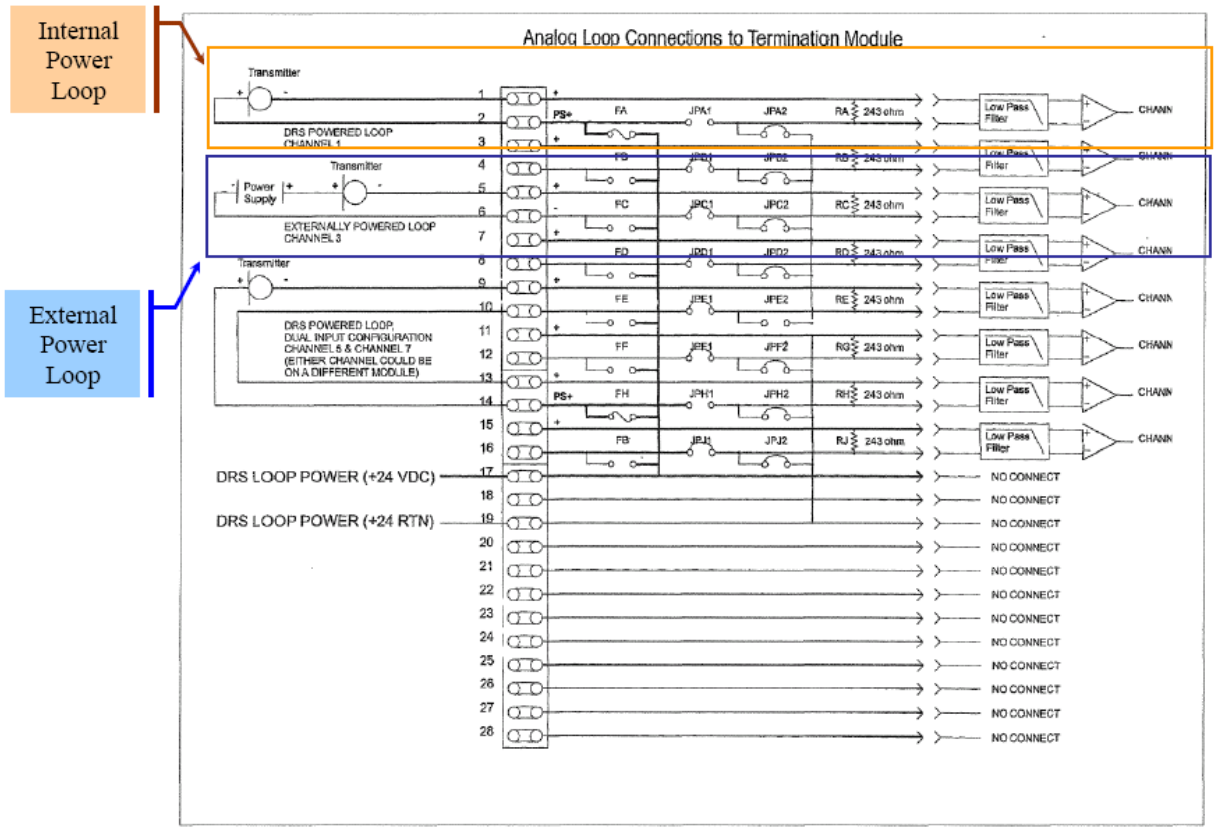
Add 14” to Cable A: NET FAULT = PASS
Add 16” to Cable A: NET FAULT = PASS
Add 18” to Cable A: NET FAULT = FAIL
:
Add 2” to Cable B: NET FAULT = PASS
Add 4” to Cable B: NET FAULT = PASS
Add 6” to Cable B: NET FAULT = FAIL

Last Pass Length of Cable A and Cable B = 16” and 4”
Correction = ((16” +(-4)”)/2 = 6”
Correction = Cable A 加上 6” 或 Cable B 減 6”

2. Hard Input Power Sources Change

根據 DRS 原先設計，部分現場之 transmitter 採 Internal Power Source，由 DRS 之 RMU 提供 power，然在 GE 的設計，這些 transmitter 所需的 power 已有其它的 cabinet 提供，經 GE 與 DRS 澄清後，由 DRS 將這些 transmitter 改為 External Power Source 方式（Hard Input Power Sources Changed FID List 如附件九），變更方式為將 DRS RMU 內 AP(Auxiliary Panel)之 Internal Power 接線改為 External Power 接線方式（範例說明詳如附件十）。

下圖為 Internal Power Loop 與 External Power Loop 示意圖



參、心得與感想

一、 此次至 DRS 參加 FID 測試見證期間，感覺 DRS 在測試流程與文件管制作業十分嚴謹，當與 GE 在設計或提供文件上有疑問時，皆能與 GE 充分溝通後，才繼續進行後續工作，對於 DRS 嚴謹的態度，相信對於未來的測試會更有信心。

二、 DRS 進行 FID 測試時，乃是依據 FID 建立 Test Matrix 後，透過 DRS FID Test bed 及 NI 程式進行自動測試，其速度快且準確率高，可為 TPC 自行建立測試平台時參考。

三、 隨著後續 VDU 及 Division FAT 進行，DRS 仍可能進行 FID 修改及測試，需密切注意後續測試進行結果。

四、 根據 DRS 表示，各 FID module 在使用期間會因為 module 上之 component（例如電容、二極體…等）造成 module 不可用，DRS 建議將 module 送回 DRS 進行檢修，然依韓電經驗，他們與韓國當地具修理安全等級設備之廠家合作，進行 module 檢修及測試，如此可省去將 module 送回 DRS 所需花費的時間及較高維修經費，這樣的維修方式可為本公司參考。

肆、建議

一、因 David Kulp 對較細部之技術問題並不見得了解，建議技術性問題儘量在取得 DRS 同意後，與相關負責工程師（如 Victor Silva、Steven Conner）直接對談，以減少時間浪費且可獲得取得較正確資訊。

二、經見證期間研讀 DRS 提供之 O＆M Manual，對 DRS 系統之安裝及維護有更進一步瞭解，然 O＆M Manual 內容仍有蠻多不足部分，目前 DRS 仍持續進行改版中，應儘速要求 DRS 提供改版後之 O&M Manual。

三、DRS 備品價格昂貴且交貨期長，DRS 系統備品預算編列及相關準備工作宜儘早進行，另有關 VDU 之備品準備，DRS 表示需以 Display Header 及 VDU controller 為單位備料，而無法以各組件（例如 power supply、SCRAMNET card…等）進行備料，此類備品亦需特別考量其備料方式。

四、因 DRS 的嚴謹態度及工作上的堅持，常會造成測試進度的延遲，此類題相信在往後 VDU test 及 Division FAT 期間亦可能遇到，建議遇到此類問題時可與 DRS 人員充分溝通，期使測試得以順利進行。

五、有關 Hard Input Power Source Change 變更之項目，日後在做相關測試時必須特別注意，避免輸入電源引用不適當，造成測試結果產生錯誤。

六、以手動方式作 skew test 及補償看似繁瑣耗時，但 DRS 認為 Skew Test 可能會發生問題的光纖，主要是長距離，一般短距離控制盤與控制盤之間的光纖並不會有太大 Skew Delay，所以不一定得買 Skew meter，以人工方式進行測試亦為可行，另在執行測試時，亦可搭配相關儀器（例如量測光傳導時間之儀器），先行判斷出該補償的光纖後再進行相關測試及補償動作，可避免因判斷錯誤所耗費的測試時間。進行與 Quad Switch 連接之 Skew Test 時，由於 Quad Switch 無法顯示通訊是否正常，執行時要將其連接至鄰近的 Cabinet 上，以協助診斷 skew delay 情形。

伍、附件

附件一 CABINET CROSS REFERENCE TABLE

NODE#	DIVISION	NETWORK	GE CAB Number for Unit 1	Cab Ref#	Assy Ref Number	DRS Cab Type
Node 0		1&2		VDU USE	RESERVE INSTRUMENT	
Node 1		1&2		VDU USE	RESERVE INSTRUMENT	
Node 2		1&2		VDU USE	RESERVE INSTRUMENT	
Node 3		1&2		VDU USE	RESERVE INSTRUMENT	
Node 4		1&2	QUAL SSLC		RESERVE TEST	9N189-1
Node 5		1&2	QUAL RMU		RESERVE TEST	9N188-1
Node 6		1&2	QUAL VDU		RESERVE TEST	5N132-1
Node 7		1&2			RESERVE TEST	
Node 8		1&2			RESERVE TEST	
Node 9						
Node 10	I	1&2	1H23PL0301A		RMU	9N188-3
Node 11	I	1&2	1H23PL0302A		RMU	9N188-3
Node 12	I	1&2	1H23PL0303A		RMU	9N188-3
Node 13	I	1&2	1H23PL0304A		RMU	9N188-3
Node 14	I	1&2	1H23PL0305A		RMU	9N188-3
Node 15	I	1&2	1H23PL0306A		RMU	9N188-3
Node 16						
Node 17						
Node 18	I	1&2	1H23PL0501A		RMU	9N188-3
Node 19						
Node 20	I	1&2	1H23PL0602A		RMU	9N188-3
Node 21						
Node 22	I	1&2	1H23PL1301A		RMU	9N188-3
Node 23	I	1&2	1H23PL1302A		RMU	9N188-3
Node 24	I	1&2	1H23PL1401A		RMU	9N188-1
Node 25	I	1&2	1H23PL2503A		RMU	9N188-1
Node 26	I	1&2	1H23PL5070A		RMU	9N188-3
Node 27	I	1&2	1H23PL5071A		RMU	9N188-3
Node 28						
Node 29						
Node 30						
Node 31	II	1&2	1H23PL0301B		RMU	9N188-3
Node 32	II	1&2	1H23PL0302B		RMU	9N188-3

NODE#	DIVISION	NETWORK	GE CAB Number for Unit 1	Cab Ref#	Assy Ref Number	DRS Cab Type
Node 33	II	1&2	1H23PL0303B		RMU	9N188-3
Node 34	II	1&2	1H23PL0304B		RMU	9N188-3
Node 35	II	1&2	1H23PL0305B		RMU	9N188-3
Node 36	II	1&2	1H23PL0306B		RMU	9N188-3
Node 37						
Node 38						
Node 39	II	1&2	1H23PL0501B		RMU	9N188-3
Node 40						
Node 41	II	1&2	1H23PL0602B		RMU	9N188-3
Node 42	II	1&2	1H23PL1301B		RMU	9N188-3
Node 43	II	1&2	1H23PL1302B		RMU	9N188-3
Node 44	II	1&2	1H23PL1303B		RMU	9N188-3
Node 45	II	1&2	1H23PL1401B		RMU	9N188-1
Node 46	II	1&2	1H23PL5070B		RMU	9N188-3
Node 47	II	1&2	1H23PL5071B		RMU	9N188-3
Node 48						
Node 49						
Node 50						
Node 51	III	1&2	1H23PL0301C		RMU	9N188-3
Node 52	III	1&2	1H23PL0302C		RMU	9N188-3
Node 53	III	1&2	1H23PL0303C		RMU	9N188-3
Node 54	III	1&2	1H23PL0304C		RMU	9N188-3
Node 55	III	1&2	1H23PL0305C		RMU	9N188-3
Node 56						
Node 57						
Node 58	III	1&2	1H23PL0501C		RMU	9N188-3
Node 59	III	1&2	1H23PL0602C		RMU	9N188-3
Node 60	III	1&2	1H23PL1301C		RMU	9N188-3
Node 61	III	1&2	1H23PL1302C		RMU	9N188-3
Node 62	III	1&2	1H23PL1303C		RMU	9N188-3
Node 63	III	1&2	1H23PL1401C		RMU	9N188-1
Node 64	III	1&2	1H23PL5070C		RMU	9N188-3
Node 65	III	1&2	1H23PL5071C		RMU	9N188-3
Node 66						
Node 67						
Node 68	IV	1&2	1H23PL1301D		RMU	9N188-3
Node 69	IV	1&2	1H23PL0301D		RMU	9N188-3

NODE#	DIVISION	NETWORK	GE CAB Number for Unit 1	Cab Ref#	Assy Ref Number	DRS Cab Type
Node 70	IV	1&2	1H23PL1401D		RMU	9N188-1
Node 71						
Node 72						
Node 73						
Node 74						
Node 75						
Node 76						
Node 77						
Node 78	I	1&2	1H12PL1109A		SSLC	9N189-1
Node 79	I	1&2	1H12PL1109B		SSLC	9N189-1
Node 80	I	1&2	1H12PL1109C		SSLC	9N189-1
Node 81						
Node 82	II	1&2	1H12PL1209A		SSLC	9N189-1
Node 83	II	1&2	1H12PL1209B		SSLC	9N189-1
Node 84	II	1&2	1H12PL1209C		SSLC	9N189-1
Node 85						
Node 86	III	1&2	1H12PL1309A		SSLC	9N189-1
Node 87	III	1&2	1H12PL1309B		SSLC	9N189-1
Node 88	III	1&2	1H12PL1309C		SSLC	9N189-1
Node 89	IV	1&2	1H12PL1409		SSLC	9N189-1
Node 90	I	1	PART OF PL1109A	1H12PL1109A	1CIM11A	6N757-1
Node90	I	2	PART OF PL1109A	1H12PL1109A	1CIM11B	6N757-1
Node 91	I	1	PART OF PL1109A	1H12PL1109A	1CIM12A	6N757-1
Node 91	I	2	PART OF PL1109A	1H12PL1109A	1CIM12B	6N757-1
Node 92	II	1	PART OF PL1209A	1H12PL1209A	1CIM21A	6N757-1
Node 92	II	2	PART OF PL1209A	1H12PL1209A	1CIM21B	6N757-1
Node 93	II	1	PART OF PL1209A	1H12PL1209A	1CIM22A	6N757-1
Node 93	II	2	PART OF PL1209A	1H12PL1209A	1CIM22B	6N757-1
Node 94	III	1	PART OF PL1309A	1H12PL1309A	1CIM31A	6N757-1
Node 94	III	2	PART OF PL1309A	1H12PL1309A	1CIM31B	6N757-1
Node 95	III	1	PART OF PL1309A	1H12PL1309A	1CIM32A	6N757-1
Node 95	III	2	PART OF PL1309A	1H12PL1309A	1CIM32B	6N757-1
Node 96	IV	1	PART OF PL1409	1H12PL1409	1CIM41A	6N757-1
Node 96	IV	2	PART OF PL1409	1H12PL1409	1CIM41B	6N757-1
Node 97						
Node 98						
Node 99	IV	1	PART OF PL1409	1H12PL1409	1CIM42A	6N757-1

NODE#	DIVISION	NETWORK	GE CAB Number for Unit 1	Cab Ref#	Assy Ref Number	DRS Cab Type
Node 99	IV	2	PART OF PL1409	1H12PL1409	1CIM42B	6N757-1
Node 100	I	1	PART OF PL1109A	1H12PL1109A	1BTM1A	6N756-1
Node 100	I	2	PART OF PL1109A	1H12PL1109A	1BTM1B	6N756-1
Node 101	II	1	PART OF PL1209A	1H12PL1209A	1BTM2A	6N756-1
Node 101	II	2	PART OF PL1209A	1H12PL1209A	1BTM2B	6N756-1
Node 102	III	1	PART OF PL1309A	1H12PL1309A	1BTM3A	6N756-1
Node 102	III	2	PART OF PL1309A	1H12PL1309A	1BTM3B	6N756-1
Node 103	IV	1	PART OF PL1409	1H12PL1409	1BTM4A	6N756-1
Node 103	IV	2	PART OF PL1409	1H12PL1409	1BTM4B	6N756-1
Node 104						
Node 105						
Node 106						
Node 107						
Node 108						
Node 109						
Node 110	I	1&2	1H23PL2501		DISPLAY	9N193-2
Node 111	I	1&2	1H11PL1703/G	WDP	VDU	5N132-1
Node 112	I	1&2	1H11PL1705/B	WDP	VDU	5N132-1
Node 113	I	1&2	1H11PL1700/C	MCC	VDU	5N132-1
Node 114	II	1&2	1H11PL1703/F	WDP	VDU	5N132-1
Node 115	II	1&2	1H11PL1705/C	WDP	VDU	5N132-1
Node 116	II	1&2	1H11PL1700/B	MCC	VDU	5N132-1
Node 117	III	1&2	1H11PL1703/D	WDP	VDU	5N132-1
Node 118	III	1&2	1H11PL1705/D	WDP	VDU	5N132-1
Node 119	III	1&2	1H11PL1700/A	MCC	VDU	5N132-1
Node 120	IV	1&2	1H11PL1703/C	WDP	VDU	5N132-1
Node 121						
Node 122						
Node 123						
Node 124						
Node 125						
Node 126					Reserved	
Node 127					Reserved	

附件二 3rd ROUIND FID測試進度

Cabinet	Qty FIDs	Tested	%Tested	Status	Comments	DIV	Test Complete
0H23PL1401S	1	1	100.00	Test Complete		0	1
0H23PL1402S	1	1	100.00	Test Complete		0	1
0H23PL2301S	3	3	100.00	Test Complete		0	1
0H23PL2302S	0	0	100.00	Test Complete		0	1
0H23PL2303S	0	0	100.00	Test Complete		0	1
0H23PL2306S	5	5	100.00	Test Complete		0	1
1H12PL1109A	5	5	100.00	Test Complete		1	1
1H12PL1109B	14	14	100.00	Test Complete		1	1
1H12PL1109C	6	6	100.00	Test Complete		1	1
1H23PL0301A	2	2	100.00	Test Complete		1	1
1H23PL0302A	15	15	100.00	Test Complete		1	1
1H23PL0303A	13	13	100.00	Test Complete		1	1
1H23PL0304A	0	0	100.00	Test Complete		1	1
1H23PL0305A	4	4	100.00	Test Complete		1	1
1H23PL0306A	8	8	100.00	Test Complete		1	1
1H23PL0501A	2	2	100.00	Test Complete		1	1
1H23PL0602A	11	11	100.00	Test Complete		1	1
1H23PL1301A	2	2	100.00	Test Complete		1	1
1H23PL1302A	8	8	100.00	Test Complete		1	1
1H23PL1401A	3	3	100.00	Test Complete		1	1
1H23PL2503A	1	1	100.00	Test Complete		1	1
1H23PL5070A	3	3	100.00	Test Complete		1	1
1H23PL5071A	5	5	100.00	Test Complete		1	1
1H12PL1209A	7	7	100.00	Test Complete		2	1
1H12PL1209B	13	13	100.00	Test Complete		2	1
1H12PL1209C	11	11	100.00	Test Complete		2	1
1H23PL0301B	3	3	100.00	Test Complete		2	1
1H23PL0302B	13	13	100.00	Test Complete		2	1
1H23PL0303B	8	8	100.00	Test Complete		2	1
1H23PL0304B	2	2	100.00	Test Complete		2	1
1H23PL0305B	6	6	100.00	Test Complete		2	1
1H23PL0306B	5	5	100.00	Test Complete		2	1
1H23PL0501B	2	2	100.00	Test Complete		2	1
1H23PL0602B	4	4	100.00	Test Complete		2	1
1H23PL1301B	6	6	100.00	Test Complete		2	1

1H23PL1302B	1	1	100.00	Test Complete		2	1
1H23PL1303B	3	3	100.00	Test Complete		2	1
1H23PL1401B	4	4	100.00	Test Complete		2	1
1H23PL5070B	4	4	100.00	Test Complete		2	1
1H23PL5071B	7	7	100.00	Test Complete		2	1
1H12PL1309A	4	4	100.00	Test Complete		3	1
1H12PL1309B	15	15	100.00	Test Complete		3	1
1H12PL1309C	6	6	100.00	Test Complete		3	1
1H23PL0301C	2	2	100.00	Test Complete		3	1
1H23PL0302C	10	10	100.00	Test Complete		3	1
1H23PL0303C	2	2	100.00	Test Complete		3	1
1H23PL0304C	5	5	100.00	Test Complete		3	1
1H23PL0305C	17	17	100.00	Test Complete		3	1
1H23PL0501C	2	2	100.00	Test Complete		3	1
1H23PL0602C	2	2	100.00	Test Complete		3	1
1H23PL1301C	6	6	100.00	Test Complete		3	1
1H23PL1302C	4	4	100.00	Test Complete		3	1
1H23PL1303C	1	1	100.00	Test Complete		3	1
1H23PL1401C	1	1	100.00	Test Complete		3	1
1H23PL5070C	3	3	100.00	Test Complete		3	1
1H23PL5071C	6	6	100.00	Test Complete		3	1
1H12PL1409	4	4	100.00	Test Complete		4	1
1H23PL0301D	1	1	100.00	Test Complete		4	1
1H23PL1301D	0	0	100.00	Test Complete		4	1
1H23PL1401D	0	0	100.00	Test Complete		4	1
2H23PL2503A	0	0	100.00	Test Complete			1
	302	302	100.00				61

附件三 4th ROUIND FID測試進度

Cabinet	Qty FIDs	Tested	%Tested	Status	Cabinet Changed	DIV	Test Complete
0H23PL1401S	0	0	100.00	Test Complete	0	0	1
0H23PL1402S	0	0	100.00	Test Complete	0	0	1
0H23PL2301S	1	1	100.00	Test Complete	1	0	1
0H23PL2302S	0	0	100.00	Test Complete	0	0	1
0H23PL2303S	0	0	100.00	Test Complete	0	0	1
0H23PL2306S	1	1	100.00	Test Complete	1	0	1
1H12PL1109A	0	0	100.00	Test Complete	0	1	1
1H12PL1109B	6	6	100.00	Test Complete	1	1	1
1H12PL1109C	1	1	100.00	Test Complete	1	1	1
1H23PL0301A	3	3	100.00	Test Complete	1	1	1
1H23PL0302A	5	5	100.00	Test Complete	1	1	1
1H23PL0303A	0	0	100.00	Test Complete	0	1	1
1H23PL0304A	0	0	100.00	Test Complete	0	1	1
1H23PL0305A	1	1	100.00	Test Complete	1	1	1
1H23PL0306A	4	4	100.00	Test Complete	1	1	1
1H23PL0501A	1	1	100.00	Test Complete	1	1	1
1H23PL0602A	1	1	100.00	Test Complete	1	1	1
1H23PL1301A	3	3	100.00	Test Complete	1	1	1
1H23PL1302A	0	0	100.00	Test Complete	0	1	1
1H23PL1401A	3	3	100.00	Test Complete	1	1	1
1H23PL2503A	0	0	100.00	Test Complete	0	1	1
1H23PL5070A	0	0	100.00	Test Complete	0	1	1
1H23PL5071A	1	1	100.00	Test Complete	1	1	1
1H12PL1209A	1	1	100.00	Test Complete	1	2	1
1H12PL1209B	3	3	100.00	Test Complete	1	2	1
1H12PL1209C	3	3	100.00	Test Complete	1	2	1
1H23PL0301B	5	5	100.00	Test Complete	1	2	1
1H23PL0302B	8	8	100.00	Test Complete	1	2	1
1H23PL0303B	1	1	100.00	Test Complete	1	2	1
1H23PL0304B	0	0	100.00	Test Complete	0	2	1
1H23PL0305B	2	2	100.00	Test Complete	1	2	1
1H23PL0306B	2	2	100.00	Test Complete	1	2	1
1H23PL0501B	1	1	100.00	Test Complete	1	2	1
1H23PL0602B	0	0	100.00	Test Complete	0	2	1
1H23PL1301B	1	1	100.00	Test Complete	1	2	1

1H23PL1302B	3	3	100.00	Test Complete	1	2	1
1H23PL1303B	3	3	100.00	Test Complete	1	2	1
1H23PL1401B	4	4	100.00	Test Complete	1	2	1
1H23PL5070B	0	0	100.00	Test Complete	0	2	1
1H23PL5071B	0	0	100.00	Test Complete	0	2	1
1H12PL1309A	1	1	100.00	Test Complete	1	3	1
1H12PL1309B	7	7	100.00	Test Complete	1	3	1
1H12PL1309C	2	2	100.00	Test Complete	1	3	1
1H23PL0301C	3	3	100.00	Test Complete	1	3	1
1H23PL0302C	2	2	100.00	Test Complete	1	3	1
1H23PL0303C	1	1	100.00	Test Complete	1	3	1
1H23PL0304C	2	2	100.00	Test Complete	1	3	1
1H23PL0305C	1	1	100.00	Test Complete	1	3	1
1H23PL0501C	0	0	100.00	Test Complete	0	3	1
1H23PL0602C	0	0	100.00	Test Complete	0	3	1
1H23PL1301C	0	0	100.00	Test Complete	0	3	1
1H23PL1302C	3	3	100.00	Test Complete	1	3	1
1H23PL1303C	2	2	100.00	Test Complete	1	3	1
1H23PL1401C	2	2	100.00	Test Complete	1	3	1
1H23PL5070C	0	0	100.00	Test Complete	0	3	1
1H23PL5071C	0	0	100.00	Test Complete	0	3	1
1H12PL1409	4	4	100.00	Test Complete	1	4	1
1H23PL0301D	0	0	100.00	Test Complete	0	4	1
1H23PL1301D	0	0	100.00	Test Complete	0	4	1
1H23PL1401D	2	2	100.00	Test Complete	1	4	1
2H23PL2503A	0	0	100.00	Test Complete	0		1
	100	100	100.00		39		61

附件四 VDU SOFTWARE UNIT TEST範例

四.1 測試程式

```
                    05/31/2007    10:39:38 AM    t_e_DP_ABV1_configureControlOverlay.c

/* MODULE NAME: t_e_DP_ABV1_configureControlOverlay.c - Unit Test wrapper for e_DP_ABV1_configureC
ontrolOverlay()

   MODULE DESCRIPTION
   ------------------
   Test wrapper file for e_DP_ABV1_configureControlOverlay.

   PUBLIC SYMBOL LIST
   ------------------
   rg_e_DP_ABV1_configureControlOverlay - structure holding xml file & function information for wr
apper
   rg_top_of_stack                      - global declared in setjmp.h to find the top of the progr
am stack if execution is interrupted
   rgp_progname                         - global holding the name of the module (ie. t_e_xxxx_xx )

   PUBLIC SUBPROGRAM LIST
   ----------------------
   e_catch_signal                          - Interrupt handler routine called when program executi
on is halted (usually cntr     - c)
   e_CO_button_activate                    - Stub that stashes away the control overlay button val
ue
   e_CO_button_inactivate                  - Stub that stashes away the control overlay button val
ue
   e_parse_button_call                     - Function that parses the array that holds the button
values passed to stub routines
   e_setStatics                            - routine to use static variables defined in the wrappe
r
   e_setup_DP_ABV1_configureControlOverlay - Sets up Inputs and Expected Outputs for each Test Cas
e
   e_test_DP_ABV1_configureControlOverlay  - Calls setup routine and compares results
   main                                    - Entry level routine for this wrapper

   PRIVATE SYMBOL LIST
   -------------------
   gu_maxButtonNameLength                                          - max buttonName length
   gu_max_CO_button_activate_calls                                 - max number of calls to e_CO_butt
on_activate
   ru_actual_controlOverlayData                                    - actual second argument passed to
 UUT
   ru_actual_DP_ABV                                                - memory allocated for the pointer
 used by the UUT
   ru_e_CO_button_activate_rlp_controlOverlayButton                - addresses of buttons passed to e
_CO_button_activate
   ru_e_CO_button_inactivate_rlp_controlOverlayButton              - addresses of buttons passed to e
_CO_button_activate
   ru_expected_controlOverlayData                                  - expected second argument passed
to UUT
   ru_expected_DP_ABV                                              - memory allocated for the UUT ret
urn value
   ru_expected_e_CO_button_activate_cntr                           - expected counter value increment
ed in the called function
   ru_expected_e_CO_button_inactivate_cntr                         - expected counter value increment
ed in the called function
   ru_expected_e_DP_ABV1_controlOverlayInit_cntr                   - expected counter value increment
ed in the called function
   ru_expected_e_DP_isValid_cntr                                   - expected counter value increment
ed in the called function
   ru_expected_e_DP_isValid_rl_algorithm                           - instance of value expected to be
 an arg to function called by UUT
   ru_first_test_case                                              - number of the first test case to
 run
   ru_last_test_case                                               - number of the last test case to
run
   rup_actual_controlOverlayData                                   - pointer to actual second argumen
t passed to UUT
   rup_actual_DP_ABV                                               - pointer to the DP structure pass
ed to the UUT
   rup_expected_controlOverlayData                                 - pointer to expected second argum
ent passed to UUT
   rup_expected_DP_ABV                                             - pointer to the expected DP struc
ture
   rup_expected_e_CO_button_activate_rlp_controlOverlayButton      - expected addresses of buttons pa
ssed to e_CO_button_inactivate

                                                                                                1
```

```
   rup_expected_e_CO_button_inactivate_rlp_controlOverlayButton - expected addresses of buttons pa
ssed to e_CO_button_inactivate
   ru_test_case_failures                                        - the number of comparison failure
s for all comparisons

   PRIVATE SUBPROGRAM LIST
   -----------------------
   N/A

   EXTERNAL EFFECTS
   ----------------
   Unit Test Output is printed to:
   1) the screen
   2) t_e_DP_ABV1_configureControlOverlay_TestResults.txt

   REVISIONS
   ---------
   $Log$
   Revision 1.3.2.2  2007/03/27 11:46:43  hills
   Fixed header/body mismatches

   Revision 1.3.2.1  2007/03/26 22:09:31  hills
   Unit test updated for Group1 Eng9.
   For bad N/W inputs, the tag permit buttons are not enabled if tagPermit is false.

   Revision 1.2.120.4  2006/08/14 22:27:37  hills
   Added default case for switch in e_catch_signal

   Revision 1.2.120.3  2006/08/14 14:04:09  hills
   Added code so pass/fail is returned to OS

   Revision 1.2.120.2  2006/08/10 22:08:57  hills
   Updated to test the changed tag-off button mask. Eng7, DPDS is CD40 ( Att T 68 )

   Revision 1.2.120.1  2006/08/10 19:48:57  bala
   Brought in Files from Eng5 and made mods to ensure that the unit test builds and
   runs.

   Revision 1.2.64.3  2006/01/23 20:02:10  krishb
   Made mods as per reviewer comments

   Revision 1.2.64.2  2006/01/23 17:51:22  krishb
   Wrapper done, tests complete. Ready for review.

*/

/* INCLUDE FILES */
#include <libgen.h> /* for basename() */
#include <setjmp.h> /* for setjmp() et.al. */
#include <signal.h> /* for SIGINT et.al. */
#include <stdio.h>  /* for sprintf() */
#include <stdlib.h> /* for exit() */
#include <string.h> /* for strcmp() */

#include "pFramework.h"                      /* for tg_test_suite */
#include "t_e_DP_ABV1_configureControlOverlay.h" /* for e_setup_DP_ABV1_configureControlOverlay()
*/
#include "p_dp_abv1.c"                          /* for access to unit-scope button structures */

/* VARIABLE AND CONSTANT DECLARATIONS */
tg_test_suite rg_e_DP_ABV1_configureControlOverlay = {
   { 1, "UnitTest/Source/Group1/t_e_DP_ABV1_configureControlOverlay_TestInput.xml" },
   { 1, "UnitTest/Source/Group1/t_e_DP_ABV1_configureControlOverlay_ExpectedOutput.xml" },
   e_setup_DP_ABV1_configureControlOverlay,
   e_test_DP_ABV1_configureControlOverlay
};

#define qu_max_CO_button_activate_calls (10)
#define qu_maxButtonNameLength (30)

char                                * rgp_progname;
jmp_buf                               rg_top_of_stack;
```

```
extern tg_DP_algorithm                  rg_e_DP_isValid_rl_algorithm;
extern tg_boolean                       rg_e_DP_isValid_return;
extern tg_uint_32                       rg_e_CO_button_activate_cntr;
extern tg_uint_32                       rg_e_CO_button_inactivate_cntr;
extern tg_uint_32                       rg_e_DP_ABV1_controlOverlayInit_cntr;
extern tg_uint_32                       rg_e_DP_isValid_cntr;

static struct tg_controlOverlayButton * ru_e_CO_button_activate_rlp_controlOverlayButton[qu_max_CO
_button_activate_calls];
static struct tg_controlOverlayButton * ru_e_CO_button_inactivate_rlp_controlOverlayButton[qu_max_
CO_button_activate_calls];
static struct tg_controlOverlayButton * rup_expected_e_CO_button_activate_rlp_controlOverlayButton
[qu_max_CO_button_activate_calls];
static struct tg_controlOverlayButton * rup_expected_e_CO_button_inactivate_rlp_controlOverlayButt
on[qu_max_CO_button_activate_calls];
static struct tg_controlOverlayData     ru_actual_controlOverlayData;
static struct tg_controlOverlayData     ru_expected_controlOverlayData;
static struct tg_controlOverlayData   * rup_actual_controlOverlayData;
static struct tg_controlOverlayData   * rup_expected_controlOverlayData;
static tg_DP_ABV                        ru_actual_DP_ABV;
static tg_DP_ABV                        ru_expected_DP_ABV;
static tg_DP_ABV                      * rup_actual_DP_ABV;
static tg_DP_ABV                      * rup_expected_DP_ABV;
static tg_DP_algorithm                  ru_expected_e_DP_isValid_rl_algorithm;
static tg_int_32                        ru_first_test_case;
static tg_int_32                        ru_last_test_case;
static tg_uint_32                       ru_expected_e_CO_button_activate_cntr;
static tg_uint_32                       ru_expected_e_CO_button_inactivate_cntr;
static tg_uint_32                       ru_expected_e_DP_ABV1_controlOverlayInit_cntr;
static tg_uint_32                       ru_expected_e_DP_isValid_cntr;
static tg_uint_32                       ru_test_case_failures;


/* SUBROUTINE NAME: e_setStatics - routine to use static variables defined in the wrapper

   SUBROUTINE DESCRIPTION
   ----------------------
   Routine to use static variables defined in the wrapper

   RETURN VALUE DESCRIPTION
   ------------------------
   void

   FORMAL PARAMETER LIST
   ---------------------
   N/A

   LOCAL SYMBOLS
   -------------
   N/A
*/
void e_setStatics( void )
{
   ru_bi_openPermits.negativeLogic = cg_false;
   ru_bi_openPermitsCoordinates.x = 0;
   ru_exitButton.numberOfTextLines = 0;
}


/* SUBROUTINE NAME: e_CO_button_activate - stub for e_CO_button_activate

   SUBROUTINE DESCRIPTION
   ----------------------
   Stub for e_CO_button_activate

   RETURN VALUE DESCRIPTION
   ----------------------
   void

   FORMAL PARAMETER LIST
   ---------------------
   rlp_controlOverlayButton - pointer to controlOverlayButton
```

```
   LOCAL SYMBOLS
   -------------
   N/A
*/
void e_CO_button_activate ( tgp_controlOverlayButton rlp_controlOverlayButton )
{
   ru_e_CO_button_activate_rlp_controlOverlayButton[rg_e_CO_button_activate_cntr] = rlp_controlOve
rlayButton;
   e_write_output( "Count `%u' in stub `%s'\n", ++rg_e_CO_button_activate_cntr, __func__ );
}


/* SUBROUTINE NAME: e_CO_button_inactivate - stub for e_CO_button_inactivate

   SUBROUTINE DESCRIPTION
   ----------------------
   Stub for e_CO_button_inactivate

   RETURN VALUE DESCRIPTION
   ------------------------
   void

   FORMAL PARAMETER LIST
   ---------------------
   rlp_controlOverlayButton - pointer to controlOverlayButton

   LOCAL SYMBOLS
   -------------
   N/A
*/
void e_CO_button_inactivate ( tgp_controlOverlayButton rlp_controlOverlayButton )
{
   ru_e_CO_button_inactivate_rlp_controlOverlayButton[rg_e_CO_button_inactivate_cntr] = rlp_contro
lOverlayButton;
   e_write_output( "Count `%u' in stub `%s'\n", ++rg_e_CO_button_inactivate_cntr, __func__ );
}


/* SUBROUTINE NAME: e_parse_button_call - parse the button call names

   SUBROUTINE DESCRIPTION
   ----------------------
   Parse the button call names. Then based on the input name, set the address of the ru variable i
n the
   rlp_button.

   RETURN VALUE DESCRIPTION
   ------------------------
   void

   FORMAL PARAMETER LIST
   ---------------------
   rlp_callName - the base name in the XML file
   rlp_button   - array of addresses to buttons

   LOCAL SYMBOLS
   -------------
   rl_buttonName  - storage for the button names read from the XML
   rl_i           - counter variable
   rlp_buttonName - pointer to base of button name storage
*/
void e_parse_button_call( tg_int_8 * rlp_callName,  struct tg_controlOverlayButton * rlp_button[]
)
{
   tg_int_8    rl_buttonName[qu_max_CO_button_activate_calls][qu_maxButtonNameLength];
   tg_int_8  * rlp_buttonName = &rl_buttonName[0][0];
   tg_uint_32  rl_i;

   memset(rl_buttonName, '\0', qu_max_CO_button_activate_calls * qu_maxButtonNameLength);
   e_parse_string_length_ptr( qu_maxButtonNameLength, &rlp_buttonName, rlp_callName, 0 );
   for (rl_i = 0; rl_i < qu_max_CO_button_activate_calls; rl_i++)
   {
      if (strcmp( &rl_buttonName[rl_i][0], "ru_openButton") == 0)
```

```
      {
         rlp_button[rl_i] = &ru_openButton;
      }
      if (strcmp( &rl_buttonName[rl_i][0], "ru_closeButton") == 0)
      {
         rlp_button[rl_i] = &ru_closeButton;
      }
      if (strcmp( &rl_buttonName[rl_i][0], "ru_tagOnButton") == 0)
      {
         rlp_button[rl_i] = &ru_tagOnButton;
      }
      if (strcmp( &rl_buttonName[rl_i][0], "ru_tagOffButton") == 0)
      {
         rlp_button[rl_i] = &ru_tagOffButton;
      }
   }
}


/* SUBROUTINE NAME: e_setup_DP_ABV1_configureControlOverlay - setup all data for each unit test

   SUBROUTINE DESCRIPTION
   ----------------------
   Gets the input and expected values for each test.

   RETURN VALUE DESCRIPTION
   ------------------------
   void

   FORMAL PARAMETER LIST
   ---------------------
   rl_test_number  - the test case number

   LOCAL SYMBOLS
   -------------
   rl_i - loop counter

*/

void e_setup_DP_ABV1_configureControlOverlay( int rl_test_number )
{
   tg_uint_32 rl_i;

   /* Make the pointers point to allocated storage   */
   rup_actual_DP_ABV = &ru_actual_DP_ABV;
   rup_expected_DP_ABV = &ru_expected_DP_ABV;
   rup_actual_controlOverlayData = &ru_actual_controlOverlayData;
   rup_expected_controlOverlayData = &ru_expected_controlOverlayData;

   /* initialize array where stubs copy argument values into */
   for (rl_i = 0; rl_i < qu_max_CO_button_activate_calls; rl_i++)
   {
      ru_e_CO_button_activate_rlp_controlOverlayButton[rl_i] = 0;
      rup_expected_e_CO_button_activate_rlp_controlOverlayButton[rl_i] = 0;
      ru_e_CO_button_inactivate_rlp_controlOverlayButton[rl_i] = 0;
      rup_expected_e_CO_button_inactivate_rlp_controlOverlayButton[rl_i] = 0;
   }

   /* initialize the global TestCase string */
   e_set_testcase( rl_test_number );

   /* set the global TestInput filename */
   e_set_xml_filename( rg_e_DP_ABV1_configureControlOverlay.input.filename );

   /* read in the actual input values from the xml file */
   e_parse_DP_ABV_ptr( &rup_actual_DP_ABV, "input_DP_ABV_ptr", 0 );
   e_parse_controlOverlayData_ptr( &rup_actual_controlOverlayData, "input_controlOverlayData_ptr",
 0 );

   /* Read in the initial value of the counters which are incremented in the stubs called by the U
UT */
   e_parse_uint_32( &rg_e_DP_isValid_cntr, "rg_e_DP_isValid_cntr", 0);
```

```
   e_parse_uint_32( &rg_e_DP_ABV1_controlOverlayInit_cntr, "rg_e_DP_ABV1_controlOverlayInit_cntr",
0);
   e_parse_uint_32( &rg_e_CO_button_inactivate_cntr, "rg_e_CO_button_inactivate_cntr", 0);
   e_parse_uint_32( &rg_e_CO_button_activate_cntr, "rg_e_CO_button_activate_cntr", 0);

   /* initialize the return value(s) */
   e_parse_boolean( &rg_e_DP_isValid_return, "e_DP_isValid_return", 0 );

   /* start out with the expected the same as the input */
   /* only copy if the pointers do, in fact, point to allocated storage */
   if ( rup_actual_DP_ABV == &ru_actual_DP_ABV
      && rup_expected_DP_ABV == &ru_expected_DP_ABV )
   {
      e_copy_DP_ABV( rup_actual_DP_ABV, rup_expected_DP_ABV );
   }
   if ( rup_actual_controlOverlayData == &ru_actual_controlOverlayData
      && rup_expected_controlOverlayData == &ru_expected_controlOverlayData )
   {
      e_copy_controlOverlayData( rup_actual_controlOverlayData, rup_expected_controlOverlayData );
   }

   /* set the global ExpectedOutput filename */
   e_set_xml_filename( rg_e_DP_ABV1_configureControlOverlay.expected_output.filename );

   /* read in the expected values from the xml file */
   e_parse_DP_ABV_ptr( &rup_expected_DP_ABV, "expected_DP_ABV_ptr", 0);
   e_parse_controlOverlayData_ptr( &rup_expected_controlOverlayData, "expected_controlOverlayData_
ptr", 0 );

   /* Read in the expected value of the counters which are incremented in the stubs called by the
UUT */
   e_parse_uint_32( &ru_expected_e_DP_isValid_cntr, "ru_expected_e_DP_isValid_cntr", 0);
   e_parse_uint_32( &ru_expected_e_DP_ABV1_controlOverlayInit_cntr, "ru_expected_e_DP_ABV1_control
OverlayInit_cntr", 0);
   e_parse_uint_32( &ru_expected_e_CO_button_inactivate_cntr, "ru_expected_e_CO_button_inactivate_
cntr", 0);
   e_parse_uint_32( &ru_expected_e_CO_button_activate_cntr, "ru_expected_e_CO_button_activate_cntr
", 0);

   /* read in the expected args of the function(s) called by the UUT */
   e_parse_DP_algorithm( &ru_expected_e_DP_isValid_rl_algorithm, "expected_e_DP_isValid_rl_algorit
hm", 0 );

   /* read in the expected names of unit-scope buttons passed to e_CO_button_activate called by th
e UUT */
   e_parse_button_call( "e_CO_button_activate_call", &rup_expected_e_CO_button_activate_rlp_contro
lOverlayButton[0] );
   e_parse_button_call( "e_CO_button_inactivate_call", &rup_expected_e_CO_button_inactivate_rlp_co
ntrolOverlayButton[0] );
}

/* SUBROUTINE NAME: e_test_DP_ABV1_configureControlOverlay - runs all unit tests

   SUBROUTINE DESCRIPTION
   ----------------------
   This function calls the setup routine, the UUT and compares results.

   RETURN VALUE DESCRIPTION
   ----------------------
   void

   FORMAL PARAMETER LIST
   ----------------------
   void

   LOCAL SYMBOLS
   -------------
   rl_i                    - loop counter
   rl_j                    - loop counter
   rl_jump_result          - variable that allows test to be interrupted
   rl_number_cases         - the total number of test cases
   rl_passing_test         - boolean indicating if the complete test was successful
   rl_run_tests            - boolean indicating whether tests should be run
```

```
   rl_string               - string variable
*/
void e_test_DP_ABV1_configureControlOverlay(void)
{
   int            rl_jump_result;
   tg_boolean     rl_passing_test;
   tg_boolean     rl_run_tests = qg_true;
   tg_int_32      rl_i = 0;
   tg_int_32      rl_j;
   tg_int_32      rl_number_cases = 0;
   tg_int_8       rl_string[128];  /* string holding first arg to e_compare_hex() */

   rl_jump_result = setjmp( rg_top_of_stack );
   if( rl_jump_result == 0 )
   {
      /* have just started the execution, non zero rl_jump_result means a thrown exception */
      rl_number_cases = get_number_test_cases( &rg_e_DP_ABV1_configureControlOverlay. );
      ru_test_case_failures = 0;
     /* If the last test case number is defined then stop running test cases at that point */
      if (ru_last_test_case > 0 && ru_last_test_case < rl_number_cases)
      {
         rl_number_cases = ru_last_test_case;
      }
      /* If the single test case number to run doesn't make sense, inform user and stop running te
sts */
      else if ((ru_last_test_case > rl_number_cases) ||
         (ru_first_test_case <= 0))
      {
         e_write_output("\nCannot run test case number %d.\nLimits are: 1<= Test case number <=%d.
\n\n", ru_last_test_case, rl_number_cases);
         e_write_output("\n**************************************************************************
*****\n\n");
         ru_test_case_failures = 1;
         rl_run_tests = qg_false;
      }
      for( rl_i = ru_first_test_case; (rl_i <= rl_number_cases) && (rl_run_tests == qg_true); rl_i
++ )
      {
         e_write_output( "\nRunning test case `%d' of `%d'\n", rl_i, rl_number_cases);

         /* Setup the inputs to and the expected outputs for the test */
         rg_e_DP_ABV1_configureControlOverlay.setup_single( rl_i );

         /* Run one test on the UUTsubroutine  */
         e_DP_ABV1_configureControlOverlay( (tgp_DP)rup_actual_DP_ABV, rup_actual_controlOverlayDa
ta );

         rl_passing_test = qg_true;
         /* compare all the structures and variables passed in */
         if (e_compare_DP_ABV( rup_expected_DP_ABV, rup_actual_DP_ABV) == qg_false)
         {
            rl_passing_test = qg_false;
         }
         if (e_compare_controlOverlayData( rup_expected_controlOverlayData, rup_actual_controlOver
layData) == qg_false)
         {
            rl_passing_test = qg_false;
         }
         /* Compare counters incremented by stubs called from the UUT */
         if (e_compare_uint_32( "rg_e_DP_isValid_cntr",
               ru_expected_e_DP_isValid_cntr, rg_e_DP_isValid_cntr) == qg_false)
         {
            rl_passing_test = qg_false;
         }
         if (e_compare_uint_32( "rg_e_DP_ABV1_controlOverlayInit_cntr",
               ru_expected_e_DP_ABV1_controlOverlayInit_cntr, rg_e_DP_ABV1_controlOverlayInit_cnt
r) == qg_false)
         {
            rl_passing_test = qg_false;
         }
         if (e_compare_uint_32( "rg_e_CO_button_inactivate_cntr",
               ru_expected_e_CO_button_inactivate_cntr, rg_e_CO_button_inactivate_cntr) == qg_fal
se)
```

```
        {
            rl_passing_test = qg_false;
        }
        if (e_compare_uint_32( "rg_e_CO_button_activate_cntr",
                ru_expected_e_CO_button_activate_cntr, rg_e_CO_button_activate_cntr) == qg_false)
        {
            rl_passing_test = qg_false;
        }
        /* check the expected args of the function(s) called by the UUT */
        if (e_compare_DP_algorithm( &ru_expected_e_DP_isValid_rl_algorithm, &rg_e_DP_isValid_rl_a
lgorithm ) == qg_false)
        {
            rl_passing_test = qg_false;
        }
        for (rl_j = 0; rl_j < qu_max_CO_button_activate_calls; rl_j++)
        {
            /* XML file entries are 1 based, so report a 1 based number in the comparison */
            sprintf( rl_string, "ru_e_CO_button_activate_rlp_controlOverlayButton[%d]", (int)(rl_j
 + 1));
            if (e_compare_hex( rl_string, (int)rup_expected_e_CO_button_activate_rlp_controlOverla
yButton[rl_j],
                (int)ru_e_CO_button_activate_rlp_controlOverlayButton[rl_j]) == qg_false)
            {
                rl_passing_test = qg_false;
            }
        }
        for (rl_j = 0; rl_j < qu_max_CO_button_activate_calls; rl_j++)
        {
            /* XML file entries are 1 based, so report a 1 based number in the comparison */
            sprintf( rl_string, "ru_e_CO_button_inactivate_rlp_controlOverlayButton[%d]", (int)(rl
_j + 1));
            if (e_compare_hex( rl_string, (int)rup_expected_e_CO_button_inactivate_rlp_controlOver
layButton[rl_j],
                (int)ru_e_CO_button_inactivate_rlp_controlOverlayButton[rl_j]) == qg_false)
            {
                rl_passing_test = qg_false;
            }
        }
        /* Check if any comparisons have failed */
        if ( rl_passing_test == qg_false )
        {
            ru_test_case_failures++;
            e_write_output("\n\nTest Case `%d' has failed.\n\n", rl_i);
        }
        else
        {
            e_write_output("\n\nTest Case `%d' has passed.\n\n", rl_i);
        }
        e_write_output("\n*********************************************\n");
    }
  }
  else
  {
     e_write_output( "Failed some aspect of test case `%d'\n", (int)rl_i);
     e_write_output("\n*********************************************\n");
     ru_test_case_failures++;
  }
}

/* SUBROUTINE NAME: e_catch_signal - signal handling routine

   SUBROUTINE DESCRIPTION
   ----------------------
   Interrupt handler routine called when program execution is
   halted (usually cntr - c)

   RETURN VALUE DESCRIPTION
   ----------------------
   void

   FORMAL PARAMETER LIST
   ----------------------
   sig - the interrupt signal received
```

```
   LOCAL SYMBOLS
   -------------
   N/A
*/
void e_catch_signal( int sig )
{
   switch( sig )
   {
      case SIGINT:
         e_write_output( "Ctrl-C received.\n" );
         exit( sig );
      default:
         e_write_output( "sig `%d' received.\n" );
         exit( sig );
   }
}

/* SUBROUTINE NAME: main - unit test entry point

   SUBROUTINE DESCRIPTION
   ----------------------
   Entry level routine for this wrapper

   RETURN VALUE DESCRIPTION
   ----------------------
   int - indicates a return value of zero on success

   FORMAL PARAMETER LIST
   ----------------------
   argc - number command line parameters passed to the routine
   argv - an array of command line parameters, in which the first parameter is name of the module

   LOCAL SYMBOLS
   -------------
   rl_return - zero on success, 1 on failure

*/
int main( int argc, char * argv[] )
{
   int rl_return;

   (void)signal( SIGINT, e_catch_signal );

   rgp_progname = basename( argv[0] );

   if (argc > 1)
   {
      ru_first_test_case = atoi(argv[1]);
      ru_last_test_case = ru_first_test_case;
   }
   else
   {
      ru_first_test_case = 1;
      /* the last test case is set to zero and will get filled in the e_test_XXX routine in this m
odule */
      ru_last_test_case = 0;
   }

   e_print_results( rg_e_DP_ABV1_configureControlOverlay.input.filename,
                    rg_e_DP_ABV1_configureControlOverlay.expected_output.filename );
   rg_e_DP_ABV1_configureControlOverlay.runall();

   e_write_output( "Final test status for `%s'", rgp_progname );
   if (ru_test_case_failures == 0)
   {
      e_write_output( "  PASSED\n" );
      rl_return = 0;
   }
   else
   {
      e_write_output( " FAILED\n" );
      rl_return = 1;
```

```
    }
    e_write_output("\n*********************************************\n");
    exit( rl_return );
}
```

四.2 Test Input程式

```
v       ,     05/31/2007    10:39:51 AM    t_e_DP_ABV1_configureControlOverlay_TestInput.xml

<?xml version="1.0" encoding="UTF-8"?>
<!--

   $Log$
   Revision 1.4.2.1  2007/03/26 22:09:31  hills
   Unit test updated for Group1 Eng9.
   For bad N/W inputs, the tag permit buttons are not enabled if tagPermit is false.

   Revision 1.3.120.1  2006/08/10 19:48:57  bala
   Brought in Files from Eng5 and made mods to ensure that the unit test builds and
   runs.

   Revision 1.3.64.2  2006/01/23 17:51:22  krishb
   Wrapper done, tests complete. Ready for review.

-->
<TestSuite>
   <num_of_test_cases value="21"/>

   <TestCase01>
      <!-- DP_ABV pointer non-zero, set in wrapper
         controlOverlayData pointer non-zero, set in wrapper
         e_DP_isValid returns false
      -->
      <!-- Initialize counter values for functions called by the UUT -->
      <rg_e_DP_isValid_cntr value = "0"/>
      <rg_e_DP_ABV1_controlOverlayInit_cntr value = "0"/>
      <rg_e_CO_button_inactivate_cntr value = "0"/>
      <rg_e_CO_button_activate_cntr value = "0"/>
      <!-- return values of functions called by UUT -->
      <e_DP_isValid_return boolean = "qg_false"/>
   </TestCase01>

   <TestCase02>
      <!-- DP_ABV pointer non-zero, set in wrapper
         controlOverlayData pointer zero
         e_DP_isValid returns true
      -->
      <input_controlOverlayData_ptr enum = "qg_null_ptr"/>
      <!-- Initialize counter values for functions called by the UUT -->
      <rg_e_DP_isValid_cntr value = "0"/>
      <rg_e_DP_ABV1_controlOverlayInit_cntr value = "0"/>
      <rg_e_CO_button_inactivate_cntr value = "0"/>
      <rg_e_CO_button_activate_cntr value = "0"/>
      <!-- return values of functions called by UUT -->
      <e_DP_isValid_return boolean = "qg_true"/>
   </TestCase02>

   <TestCase03>
      <!-- DP_ABV pointer non-zero
         controlOverlayData pointer non-zero
         rlp_DP_ABV->intializedFlag is false
         rlp_DP_ABV->common.networkInputsValid is false
         rlp_DP_ABV->common.tagPermitted is false
         e_DP_isValid returns true
      -->
      <input_DP_ABV_ptr link = "input_DP_ABV"/>
      <input_DP_ABV>
         <common>
            <networkInputsValid boolean = "qg_false"/>
            <tagPermitted boolean = "qg_false"/>
         </common>
      </input_DP_ABV>
      <input_controlOverlayData_ptr link = "input_controlOverlayData"/>
      <input_controlOverlayData>
         <initializedFlag boolean = "qg_false"/>
      </input_controlOverlayData>
      <!-- Initialize counter values for functions called by the UUT -->
      <rg_e_DP_isValid_cntr value = "0" />
      <rg_e_DP_ABV1_controlOverlayInit_cntr value = "0"/>
      <!-- rg_e_CO_button_inactivate_cntr is used in the wrapper as an array index and should star
t at 0 -->
      <rg_e_CO_button_inactivate_cntr value = "0"/>
```

```
      <!-- rg_e_CO_button_activate_cntr is used in the wrapper as an array index and should start
at 0 -->
      <rg_e_CO_button_activate_cntr value = "0"/>
      <!-- return values of functions called by UUT -->
      <e_DP_isValid_return boolean = "qg_true"/>
   </TestCase03>

   <TestCase04>
      <!-- DP_ABV pointer non-zero
         controlOverlayData pointer non-zero
         rlp_DP_ABV->intializedFlag is true
         rlp_DP_ABV->common.networkInputsValid is true
         rlp_DP_ABV->common.tagPermitted is true
         e_DP_isValid returns true
         image pattern as per columns in table
           4.1.6.of DPDS is 0 0 0 0
      -->
      <input_DP_ABV_ptr link = "input_DP_ABV"/>
      <input_DP_ABV>
         <common>
            <networkInputsValid boolean = "qg_true"/>
            <tagPermitted boolean = "qg_true" />
         </common>
         <sig_valveFullOpen>
            <image value = "0" />
         </sig_valveFullOpen>
         <sig_valveFullClosed>
            <image value = "0" />
         </sig_valveFullClosed>
         <sig_valveFailedToOpen>
            <image value = "0" />
         </sig_valveFailedToOpen>
         <sig_tagout>
            <image value = "0" />
         </sig_tagout>
      </input_DP_ABV>
      <input_controlOverlayData_ptr link = "input_controlOverlayData"/>
      <input_controlOverlayData>
         <initializedFlag boolean = "qg_true"/>
      </input_controlOverlayData>
      <!-- Initialize counter values for functions called by the UUT -->
      <rg_e_DP_isValid_cntr value = "0"/>
      <rg_e_DP_ABV1_controlOverlayInit_cntr value = "0"/>
      <!-- rg_e_CO_button_inactivate_cntr is used in the wrapper as an array index and should star
t at 0 -->
      <rg_e_CO_button_inactivate_cntr value = "0"/>
      <!-- rg_e_CO_button_activate_cntr is used in the wrapper as an array index and should start
at 0 -->
      <rg_e_CO_button_activate_cntr value = "0"/>
      <!-- return values of functions called by UUT -->
      <e_DP_isValid_return boolean = "qg_true"/>
   </TestCase04>

   <TestCase05>
      <!-- DP_ABV pointer non-zero
         controlOverlayData pointer non-zero
         rlp_DP_ABV->intializedFlag is true
         rlp_DP_ABV->common.networkInputsValid is true
         rlp_DP_ABV->common.tagPermitted is true
         e_DP_isValid returns true
         image pattern as per columns in table
           4.1.6.of DPDS is 0 0 0 1
      -->
      <input_DP_ABV_ptr link = "input_DP_ABV"/>
      <input_DP_ABV>
         <common>
            <networkInputsValid boolean = "qg_true"/>
            <tagPermitted boolean = "qg_true" />
         </common>
         <sig_valveFullOpen>
            <image value = "0" />
         </sig_valveFullOpen>
         <sig_valveFullClosed>
```

```
            <image value = "0" />
         </sig_valveFullClosed>
         <sig_valveFailedToOpen>
            <image value = "0" />
         </sig_valveFailedToOpen>
         <sig_tagout>
            <image value = "1" />
         </sig_tagout>
      </input_DP_ABV>
      <input_controlOverlayData_ptr link = "input_controlOverlayData"/>
      <input_controlOverlayData>
         <initializedFlag boolean = "qg_true"/>
      </input_controlOverlayData>
      <!-- Initialize counter values for functions called by the UUT -->
      <rg_e_DP_isValid_cntr value = "0"/>
      <rg_e_DP_ABV1_controlOverlayInit_cntr value = "0"/>
      <!-- rg_e_CO_button_inactivate_cntr is used in the wrapper as an array index and should star
t at 0 -->
      <rg_e_CO_button_inactivate_cntr value = "0"/>
      <!-- rg_e_CO_button_activate_cntr is used in the wrapper as an array index and should start
at 0 -->
      <rg_e_CO_button_activate_cntr value = "0"/>
      <!-- return values of functions called by UUT -->
      <e_DP_isValid_return boolean = "qg_true"/>
   </TestCase05>

   <TestCase06>
      <!-- DP_ABV pointer non-zero
         controlOverlayData pointer non-zero
         rlp_DP_ABV->intializedFlag is true
         rlp_DP_ABV->common.networkInputsValid is true
         rlp_DP_ABV->common.tagPermitted is true
         e_DP_isValid returns true
         image pattern as per columns in table
           4.1.6.of DPDS is 0 0 1 0
      -->
      <input_DP_ABV_ptr link = "input_DP_ABV"/>
      <input_DP_ABV>
         <common>
            <networkInputsValid boolean = "qg_true"/>
            <tagPermitted boolean = "qg_true" />
         </common>
         <sig_valveFullOpen>
            <image value = "0" />
         </sig_valveFullOpen>
         <sig_valveFullClosed>
            <image value = "0" />
         </sig_valveFullClosed>
         <sig_valveFailedToOpen>
            <image value = "1" />
         </sig_valveFailedToOpen>
         <sig_tagout>
            <image value = "0" />
         </sig_tagout>
      </input_DP_ABV>
      <input_controlOverlayData_ptr link = "input_controlOverlayData"/>
      <input_controlOverlayData>
         <initializedFlag boolean = "qg_true"/>
      </input_controlOverlayData>
      <!-- Initialize counter values for functions called by the UUT -->
      <rg_e_DP_isValid_cntr value = "0"/>
      <rg_e_DP_ABV1_controlOverlayInit_cntr value = "0"/>
      <!-- rg_e_CO_button_inactivate_cntr is used in the wrapper as an array index and should star
t at 0 -->
      <rg_e_CO_button_inactivate_cntr value = "0"/>
      <!-- rg_e_CO_button_activate_cntr is used in the wrapper as an array index and should start
at 0 -->
      <rg_e_CO_button_activate_cntr value = "0"/>
      <!-- return values of functions called by UUT -->
      <e_DP_isValid_return boolean = "qg_true"/>
   </TestCase06>

   <TestCase07>
```

```
      <!-- DP_ABV pointer non-zero
         controlOverlayData pointer non-zero
         rlp_DP_ABV->intializedFlag is true
         rlp_DP_ABV->common.networkInputsValid is true
         rlp_DP_ABV->common.tagPermitted is true
         e_DP_isValid returns true
         image pattern as per columns in table
           4.1.6.of DPDS is 0 0 1 1
      -->
      <input_DP_ABV_ptr link = "input_DP_ABV"/>
      <input_DP_ABV>
         <common>
            <networkInputsValid boolean = "qq_true"/>
            <tagPermitted boolean = "qq_true" />
         </common>
         <sig_valveFullOpen>
            <image value = "0" />
         </sig_valveFullOpen>
         <sig_valveFullClosed>
            <image value = "0" />
         </sig_valveFullClosed>
         <sig_valveFailedToOpen>
            <image value = "1" />
         </sig_valveFailedToOpen>
         <sig_tagout>
            <image value = "1" />
         </sig_tagout>
      </input_DP_ABV>
      <input_controlOverlayData_ptr link = "input_controlOverlayData"/>
      <input_controlOverlayData>
         <initializedFlag boolean = "qq_true"/>
      </input_controlOverlayData>
      <!-- Initialize counter values for functions called by the UUT -->
      <rg_e_DP_isValid_cntr value = "0"/>
      <rg_e_DP_ABV1_controlOverlayInit_cntr value = "0"/>
      <!-- rg_e_CO_button_inactivate_cntr is used in the wrapper as an array index and should star
t at 0 -->
      <rg_e_CO_button_inactivate_cntr value = "0"/>
      <!-- rg_e_CO_button_activate_cntr is used in the wrapper as an array index and should start
at 0 -->
      <rg_e_CO_button_activate_cntr value = "0"/>
      <!-- return values of functions called by UUT -->
      <e_DP_isValid_return boolean = "qq_true"/>
   </TestCase07>

   <TestCase08>
      <!-- DP_ABV pointer non-zero
         controlOverlayData pointer non-zero
         rlp_DP_ABV->intializedFlag is true
         rlp_DP_ABV->common.networkInputsValid is true
         rlp_DP_ABV->common.tagPermitted is true
         e_DP_isValid returns true
         image pattern as per columns in table
           4.1.6.of DPDS is 0 1 0 0
      -->
      <input_DP_ABV_ptr link = "input_DP_ABV"/>
      <input_DP_ABV>
         <common>
            <networkInputsValid boolean = "qq_true"/>
            <tagPermitted boolean = "qq_true" />
         </common>
         <sig_valveFullOpen>
            <image value = "0" />
         </sig_valveFullOpen>
         <sig_valveFullClosed>
            <image value = "1" />
         </sig_valveFullClosed>
         <sig_valveFailedToOpen>
            <image value = "0" />
         </sig_valveFailedToOpen>
         <sig_tagout>
            <image value = "0" />
         </sig_tagout>
```

```
      </input_DP_ABV>
      <input_controlOverlayData_ptr link = "input_controlOverlayData"/>
      <input_controlOverlayData>
         <initializedFlag boolean = "qg_true"/>
      </input_controlOverlayData>
      <!-- Initialize counter values for functions called by the UUT -->
      <rg_e_DP_isValid_cntr value = "0"/>
      <rg_e_DP_ABV1_controlOverlayInit_cntr value = "0"/>
      <!-- rg_e_CO_button_inactivate_cntr is used in the wrapper as an array index and should star
t at 0 -->
      <rg_e_CO_button_inactivate_cntr value = "0"/>
      <!-- rg_e_CO_button_activate_cntr is used in the wrapper as an array index and should start
at 0 -->
      <rg_e_CO_button_activate_cntr value = "0"/>
      <!-- return values of functions called by UUT -->
      <e_DP_isValid_return boolean = "qg_true"/>
   </TestCase08>

   <TestCase09>
      <!-- DP_ABV pointer non-zero
         controlOverlayData pointer non-zero
         rlp_DP_ABV->intializedFlag is true
         rlp_DP_ABV->common.networkInputsValid is true
         rlp_DP_ABV->common.tagPermitted is true
         e_DP_isValid returns true
         image pattern as per columns in table
           4.1.6.of DPDS is 0 1 0 1
      -->
      <input_DP_ABV_ptr link = "input_DP_ABV"/>
      <input_DP_ABV>
         <common>
            <networkInputsValid boolean = "qg_true"/>
            <tagPermitted boolean = "qg_true" />
         </common>
         <sig_valveFullOpen>
            <image value = "0" />
         </sig_valveFullOpen>
         <sig_valveFullClosed>
            <image value = "1" />
         </sig_valveFullClosed>
         <sig_valveFailedToOpen>
            <image value = "0" />
         </sig_valveFailedToOpen>
         <sig_tagout>
            <image value = "1" />
         </sig_tagout>
      </input_DP_ABV>
      <input_controlOverlayData_ptr link = "input_controlOverlayData"/>
      <input_controlOverlayData>
         <initializedFlag boolean = "qg_true"/>
      </input_controlOverlayData>
      <!-- Initialize counter values for functions called by the UUT -->
      <rg_e_DP_isValid_cntr value = "0"/>
      <rg_e_DP_ABV1_controlOverlayInit_cntr value = "0"/>
      <!-- rg_e_CO_button_inactivate_cntr is used in the wrapper as an array index and should star
t at 0 -->
      <rg_e_CO_button_inactivate_cntr value = "0"/>
      <!-- rg_e_CO_button_activate_cntr is used in the wrapper as an array index and should start
at 0 -->
      <rg_e_CO_button_activate_cntr value = "0"/>
      <!-- return values of functions called by UUT -->
      <e_DP_isValid_return boolean = "qg_true"/>
   </TestCase09>

   <TestCase10>
      <!-- DP_ABV pointer non-zero
         controlOverlayData pointer non-zero
         rlp_DP_ABV->intializedFlag is true
         rlp_DP_ABV->common.networkInputsValid is true
         rlp_DP_ABV->common.tagPermitted is true
         e_DP_isValid returns true
         image pattern as per columns in table
           4.1.6.of DPDS is 0 1 1 0
```

```
      -->
      <input_DP_ABV_ptr link = "input_DP_ABV"/>
      <input_DP_ABV>
         <common>
            <networkInputsValid boolean = "qg_true"/>
            <tagPermitted boolean = "qg_true" />
         </common>
         <sig_valveFullOpen>
            <image value = "0" />
         </sig_valveFullOpen>
         <sig_valveFullClosed>
            <image value = "1" />
         </sig_valveFullClosed>
         <sig_valveFailedToOpen>
            <image value = "1" />
         </sig_valveFailedToOpen>
         <sig_tagout>
            <image value = "0" />
         </sig_tagout>
      </input_DP_ABV>
      <input_controlOverlayData_ptr link = "input_controlOverlayData"/>
      <input_controlOverlayData>
         <initializedFlag boolean = "qg_true"/>
      </input_controlOverlayData>
      <!-- Initialize counter values for functions called by the UUT -->
      <rg_e_DP_isValid_cntr value = "0"/>
      <rg_e_DP_ABV1_controlOverlayInit_cntr value = "0"/>
      <!-- rg_e_CO_button_inactivate_cntr is used in the wrapper as an array index and should star
t at 0 -->
      <rg_e_CO_button_inactivate_cntr value = "0"/>
      <!-- rg_e_CO_button_activate_cntr is used in the wrapper as an array index and should start
at 0 -->
      <rg_e_CO_button_activate_cntr value = "0"/>
      <!-- return values of functions called by UUT -->
      <e_DP_isValid_return boolean = "qg_true"/>
   </TestCase10>

   <TestCase11>
      <!-- DP_ABV pointer non-zero
         controlOverlayData pointer non-zero
         rlp_DP_ABV->intializedFlag is true
         rlp_DP_ABV->common.networkInputsValid is true
         rlp_DP_ABV->common.tagPermitted is true
         e_DP_isValid returns true
         image pattern as per columns in table
           4.1.6.of DPDS is 0 1 1 1
      -->
      <input_DP_ABV_ptr link = "input_DP_ABV"/>
      <input_DP_ABV>
         <common>
            <networkInputsValid boolean = "qg_true"/>
            <tagPermitted boolean = "qg_true" />
         </common>
         <sig_valveFullOpen>
            <image value = "0" />
         </sig_valveFullOpen>
         <sig_valveFullClosed>
            <image value = "1" />
         </sig_valveFullClosed>
         <sig_valveFailedToOpen>
            <image value = "1" />
         </sig_valveFailedToOpen>
         <sig_tagout>
            <image value = "1" />
         </sig_tagout>
      </input_DP_ABV>
      <input_controlOverlayData_ptr link = "input_controlOverlayData"/>
      <input_controlOverlayData>
         <initializedFlag boolean = "qg_true"/>
      </input_controlOverlayData>
      <!-- Initialize counter values for functions called by the UUT -->
      <rg_e_DP_isValid_cntr value = "0"/>
      <rg_e_DP_ABV1_controlOverlayInit_cntr value = "0"/>
```

```
      <!-- rg_e_CO_button_inactivate_cntr is used in the wrapper as an array index and should star
t at 0 -->
      <rg_e_CO_button_inactivate_cntr value = "0"/>
      <!-- rg_e_CO_button_activate_cntr is used in the wrapper as an array index and should start
at 0 -->
      <rg_e_CO_button_activate_cntr value = "0"/>
      <!-- return values of functions called by UUT -->
      <e_DP_isValid_return boolean = "qg_true"/>
   </TestCase11>

   <TestCase12>
      <!-- DP_ABV pointer non-zero
         controlOverlayData pointer non-zero
         rlp_DP_ABV->intializedFlag is true
         rlp_DP_ABV->common.networkInputsValid is true
         rlp_DP_ABV->common.tagPermitted is true
         e_DP_isValid returns true
         image pattern as per columns in table
           4.1.6.of DPDS is 1 0 0 0
      -->
      <input_DP_ABV_ptr link = "input_DP_ABV"/>
      <input_DP_ABV>
         <common>
            <networkInputsValid boolean = "qg_true"/>
            <tagPermitted boolean = "qg_true" />
         </common>
         <sig_valveFullOpen>
            <image value = "1" />
         </sig_valveFullOpen>
         <sig_valveFullClosed>
            <image value = "0" />
         </sig_valveFullClosed>
         <sig_valveFailedToOpen>
            <image value = "0" />
         </sig_valveFailedToOpen>
         <sig_tagout>
            <image value = "0" />
         </sig_tagout>
      </input_DP_ABV>
      <input_controlOverlayData_ptr link = "input_controlOverlayData"/>
      <input_controlOverlayData>
         <initializedFlag boolean = "qg_true"/>
      </input_controlOverlayData>
      <!-- Initialize counter values for functions called by the UUT -->
      <rg_e_DP_isValid_cntr value = "0"/>
      <rg_e_DP_ABV1_controlOverlayInit_cntr value = "0"/>
      <!-- rg_e_CO_button_inactivate_cntr is used in the wrapper as an array index and should star
t at 0 -->
      <rg_e_CO_button_inactivate_cntr value = "0"/>
      <!-- rg_e_CO_button_activate_cntr is used in the wrapper as an array index and should start
at 0 -->
      <rg_e_CO_button_activate_cntr value = "0"/>
      <!-- return values of functions called by UUT -->
      <e_DP_isValid_return boolean = "qg_true"/>
   </TestCase12>

   <TestCase13>
      <!-- DP_ABV pointer non-zero
         controlOverlayData pointer non-zero
         rlp_DP_ABV->intializedFlag is true
         rlp_DP_ABV->common.networkInputsValid is true
         rlp_DP_ABV->common.tagPermitted is true
         e_DP_isValid returns true
         image pattern as per columns in table
           4.1.6.of DPDS is 1 0 0 1
      -->
      <input_DP_ABV_ptr link = "input_DP_ABV"/>
      <input_DP_ABV>
         <common>
            <networkInputsValid boolean = "qg_true"/>
            <tagPermitted boolean = "qg_true" />
         </common>
         <sig_valveFullOpen>
```

```
            <image value = "1" />
         </sig_valveFullOpen>
         <sig_valveFullClosed>
            <image value = "0" />
         </sig_valveFullClosed>
         <sig_valveFailedToOpen>
            <image value = "0" />
         </sig_valveFailedToOpen>
         <sig_tagout>
            <image value = "1" />
         </sig_tagout>
      </input_DP_ABV>
      <input_controlOverlayData_ptr link = "input_controlOverlayData"/>
      <input_controlOverlayData>
         <initializedFlag boolean = "qg_true"/>
      </input_controlOverlayData>
      <!-- Initialize counter values for functions called by the UUT -->
      <rg_e_DP_isValid_cntr value = "0"/>
      <rg_e_DP_ABV1_controlOverlayInit_cntr value = "0"/>
      <!-- rg_e_CO_button_inactivate_cntr is used in the wrapper as an array index and should star
t at 0 -->
      <rg_e_CO_button_inactivate_cntr value = "0"/>
      <!-- rg_e_CO_button_activate_cntr is used in the wrapper as an array index and should start
at 0 -->
      <rg_e_CO_button_activate_cntr value = "0"/>
      <!-- return values of functions called by UUT -->
      <e_DP_isValid_return boolean = "qg_true"/>
   </TestCase13>

   <TestCase14>
      <!-- DP_ABV pointer non-zero
         controlOverlayData pointer non-zero
         rlp_DP_ABV->intializedFlag is true
         rlp_DP_ABV->common.networkInputsValid is true
         rlp_DP_ABV->common.tagPermitted is true
         e_DP_isValid returns true
         image pattern as per columns in table
           4.1.6.of DPDS is 1 0 1 0
      -->
      <input_DP_ABV_ptr link = "input_DP_ABV"/>
      <input_DP_ABV>
         <common>
            <networkInputsValid boolean = "qg_true"/>
            <tagPermitted boolean = "qg_true" />
         </common>
         <sig_valveFullOpen>
            <image value = "1" />
         </sig_valveFullOpen>
         <sig_valveFullClosed>
            <image value = "0" />
         </sig_valveFullClosed>
         <sig_valveFailedToOpen>
            <image value = "1" />
         </sig_valveFailedToOpen>
         <sig_tagout>
            <image value = "0" />
         </sig_tagout>
      </input_DP_ABV>
      <input_controlOverlayData_ptr link = "input_controlOverlayData"/>
      <input_controlOverlayData>
         <initializedFlag boolean = "qg_true"/>
      </input_controlOverlayData>
      <!-- Initialize counter values for functions called by the UUT -->
      <rg_e_DP_isValid_cntr value = "0"/>
      <rg_e_DP_ABV1_controlOverlayInit_cntr value = "0"/>
      <!-- rg_e_CO_button_inactivate_cntr is used in the wrapper as an array index and should star
t at 0 -->
      <rg_e_CO_button_inactivate_cntr value = "0"/>
      <!-- rg_e_CO_button_activate_cntr is used in the wrapper as an array index and should start
at 0 -->
      <rg_e_CO_button_activate_cntr value = "0"/>
      <!-- return values of functions called by UUT -->
      <e_DP_isValid_return boolean = "qg_true"/>
```

```
   </TestCase14>

   <TestCase15>
      <!-- DP_ABV pointer non-zero
         controlOverlayData pointer non-zero
         rlp_DP_ABV->intializedFlag is true
         rlp_DP_ABV->common.networkInputsValid is true
         rlp_DP_ABV->common.tagPermitted is true
         e_DP_isValid returns true
         image pattern as per columns in table
           4.1.6.of DPDS is 1 0 1 1
      -->
      <input_DP_ABV_ptr link = "input_DP_ABV"/>
      <input_DP_ABV>
         <common>
            <networkInputsValid boolean = "qq_true"/>
            <tagPermitted boolean = "qq_true" />
         </common>
         <sig_valveFullOpen>
            <image value = "1" />
         </sig_valveFullOpen>
         <sig_valveFullClosed>
            <image value = "0" />
         </sig_valveFullClosed>
         <sig_valveFailedToOpen>
            <image value = "1" />
         </sig_valveFailedToOpen>
         <sig_tagout>
            <image value = "1" />
         </sig_tagout>
      </input_DP_ABV>
      <input_controlOverlayData_ptr link = "input_controlOverlayData"/>
      <input_controlOverlayData>
         <initializedFlag boolean = "qq_true"/>
      </input_controlOverlayData>
      <!-- Initialize counter values for functions called by the UUT -->
      <rg_e_DP_isValid_cntr value = "0"/>
      <rg_e_DP_ABV1_controlOverlayInit_cntr value = "0"/>
      <!-- rg_e_CO_button_inactivate_cntr is used in the wrapper as an array index and should star
t at 0 -->
      <rg_e_CO_button_inactivate_cntr value = "0"/>
      <!-- rg_e_CO_button_activate_cntr is used in the wrapper as an array index and should start
at 0 -->
      <rg_e_CO_button_activate_cntr value = "0"/>
      <!-- return values of functions called by UUT -->
      <e_DP_isValid_return boolean = "qq_true"/>
   </TestCase15>

   <TestCase16>
      <!-- DP_ABV pointer non-zero
         controlOverlayData pointer non-zero
         rlp_DP_ABV->intializedFlag is true
         rlp_DP_ABV->common.networkInputsValid is true
         rlp_DP_ABV->common.tagPermitted is true
         e_DP_isValid returns true
         image pattern as per columns in table
           4.1.6.of DPDS is 1 1 0 0
      -->
      <input_DP_ABV_ptr link = "input_DP_ABV"/>
      <input_DP_ABV>
         <common>
            <networkInputsValid boolean = "qq_true"/>
            <tagPermitted boolean = "qq_true" />
         </common>
         <sig_valveFullOpen>
            <image value = "1" />
         </sig_valveFullOpen>
         <sig_valveFullClosed>
            <image value = "1" />
         </sig_valveFullClosed>
         <sig_valveFailedToOpen>
            <image value = "0" />
         </sig_valveFailedToOpen>
```

```
         <sig_tagout>
            <image value = "0" />
         </sig_tagout>
      </input_DP_ABV>
      <input_controlOverlayData_ptr link = "input_controlOverlayData"/>
      <input_controlOverlayData>
         <initializedFlag boolean = "qg_true"/>
      </input_controlOverlayData>
      <!-- Initialize counter values for functions called by the UUT -->
      <rg_e_DP_isValid_cntr value = "0"/>
      <rg_e_DP_ABV1_controlOverlayInit_cntr value = "0"/>
      <!-- rg_e_CO_button_inactivate_cntr is used in the wrapper as an array index and should star
t at 0 -->
      <rg_e_CO_button_inactivate_cntr value = "0"/>
      <!-- rg_e_CO_button_activate_cntr is used in the wrapper as an array index and should start
at 0 -->
      <rg_e_CO_button_activate_cntr value = "0"/>
      <!-- return values of functions called by UUT -->
      <e_DP_isValid_return boolean = "qg_true"/>
   </TestCase16>

   <TestCase17>
      <!-- DP_ABV pointer non-zero
         controlOverlayData pointer non-zero
         rlp_DP_ABV->intializedFlag is true
         rlp_DP_ABV->common.networkInputsValid is true
         rlp_DP_ABV->common.tagPermitted is true
         e_DP_isValid returns true
         image pattern as per columns in table
           4.1.6.of DPDS is 1 1 0 1
      -->
      <input_DP_ABV_ptr link = "input_DP_ABV"/>
      <input_DP_ABV>
         <common>
            <networkInputsValid boolean = "qg_true"/>
            <tagPermitted boolean = "qg_true" />
         </common>
         <sig_valveFullOpen>
            <image value = "1" />
         </sig_valveFullOpen>
         <sig_valveFullClosed>
            <image value = "1" />
         </sig_valveFullClosed>
         <sig_valveFailedToOpen>
            <image value = "0" />
         </sig_valveFailedToOpen>
         <sig_tagout>
            <image value = "1" />
         </sig_tagout>
      </input_DP_ABV>
      <input_controlOverlayData_ptr link = "input_controlOverlayData"/>
      <input_controlOverlayData>
         <initializedFlag boolean = "qg_true"/>
      </input_controlOverlayData>
      <!-- Initialize counter values for functions called by the UUT -->
      <rg_e_DP_isValid_cntr value = "0"/>
      <rg_e_DP_ABV1_controlOverlayInit_cntr value = "0"/>
      <!-- rg_e_CO_button_inactivate_cntr is used in the wrapper as an array index and should star
t at 0 -->
      <rg_e_CO_button_inactivate_cntr value = "0"/>
      <!-- rg_e_CO_button_activate_cntr is used in the wrapper as an array index and should start
at 0 -->
      <rg_e_CO_button_activate_cntr value = "0"/>
      <!-- return values of functions called by UUT -->
      <e_DP_isValid_return boolean = "qg_true"/>
   </TestCase17>

   <TestCase18>
      <!-- DP_ABV pointer non-zero
         controlOverlayData pointer non-zero
         rlp_DP_ABV->intializedFlag is true
         rlp_DP_ABV->common.networkInputsValid is true
         rlp_DP_ABV->common.tagPermitted is true
```

```
      e_DP_isValid returns true
      image pattern as per columns in table
        4.1.6.of DPDS is 1 1 1 0
   -->
   <input_DP_ABV_ptr link = "input_DP_ABV"/>
   <input_DP_ABV>
      <common>
         <networkInputsValid boolean = "qq_true"/>
         <tagPermitted boolean = "qq_true" />
      </common>
      <sig_valveFullOpen>
         <image value = "1" />
      </sig_valveFullOpen>
      <sig_valveFullClosed>
         <image value = "1" />
      </sig_valveFullClosed>
      <sig_valveFailedToOpen>
         <image value = "1" />
      </sig_valveFailedToOpen>
      <sig_tagout>
         <image value = "0" />
      </sig_tagout>
   </input_DP_ABV>
   <input_controlOverlayData_ptr link = "input_controlOverlayData"/>
   <input_controlOverlayData>
      <initializedFlag boolean = "qq_true"/>
   </input_controlOverlayData>
   <!-- Initialize counter values for functions called by the UUT -->
   <rg_e_DP_isValid_cntr value = "0"/>
   <rg_e_DP_ABV1_controlOverlayInit_cntr value = "0"/>
   <!-- rg_e_CO_button_inactivate_cntr is used in the wrapper as an array index and should star
t at 0 -->
   <rg_e_CO_button_inactivate_cntr value = "0"/>
   <!-- rg_e_CO_button_activate_cntr is used in the wrapper as an array index and should start
at 0 -->
   <rg_e_CO_button_activate_cntr value = "0"/>
   <!-- return values of functions called by UUT -->
   <e_DP_isValid_return boolean = "qq_true"/>
</TestCase18>

<TestCase19>
   <!-- DP_ABV pointer non-zero
      controlOverlayData pointer non-zero
      rlp_DP_ABV->intializedFlag is true
      rlp_DP_ABV->common.networkInputsValid is true
      rlp_DP_ABV->common.tagPermitted is true
      e_DP_isValid returns true
      image pattern as per columns in table
        4.1.6.of DPDS is 1 1 1 1
   -->
   <input_DP_ABV_ptr link = "input_DP_ABV"/>
   <input_DP_ABV>
      <common>
         <networkInputsValid boolean = "qq_true"/>
         <tagPermitted boolean = "qq_true" />
      </common>
      <sig_valveFullOpen>
         <image value = "1" />
      </sig_valveFullOpen>
      <sig_valveFullClosed>
         <image value = "1" />
      </sig_valveFullClosed>
      <sig_valveFailedToOpen>
         <image value = "1" />
      </sig_valveFailedToOpen>
      <sig_tagout>
         <image value = "1" />
      </sig_tagout>
   </input_DP_ABV>
   <input_controlOverlayData_ptr link = "input_controlOverlayData"/>
   <input_controlOverlayData>
      <initializedFlag boolean = "qq_true"/>
   </input_controlOverlayData>
```

```
      <!-- Initialize counter values for functions called by the UUT -->
      <rg_e_DP_isValid_cntr value = "0"/>
      <rg_e_DP_ABV1_controlOverlayInit_cntr value = "0"/>
      <!-- rg_e_CO_button_inactivate_cntr is used in the wrapper as an array index and should star
t at 0 -->
      <rg_e_CO_button_inactivate_cntr value = "0"/>
      <!-- rg_e_CO_button_activate_cntr is used in the wrapper as an array index and should start
at 0 -->
      <rg_e_CO_button_activate_cntr value = "0"/>
      <!-- return values of functions called by UUT -->
      <e_DP_isValid_return boolean = "qg_true"/>
   </TestCase19>

   <TestCase20>
      <!-- DP_ABV pointer non-zero
         controlOverlayData pointer non-zero
         rlp_DP_ABV->intializedFlag is true
         rlp_DP_ABV->common.networkInputsValid is true
         rlp_DP_ABV->common.tagPermitted is false
         e_DP_isValid returns true
         image pattern as per columns in table
           4.1.6.of DPDS is 1 1 0 1
         This test case is essentially Test case 17 with tagPermitted false
      -->
      <input_DP_ABV_ptr link = "input_DP_ABV"/>
      <input_DP_ABV>
         <common>
            <networkInputsValid boolean = "qg_true"/>
            <tagPermitted boolean = "qg_false" />
         </common>
         <sig_valveFullOpen>
            <image value = "1" />
         </sig_valveFullOpen>
         <sig_valveFullClosed>
            <image value = "1" />
         </sig_valveFullClosed>
         <sig_valveFailedToOpen>
            <image value = "0" />
         </sig_valveFailedToOpen>
         <sig_tagout>
            <image value = "1" />
         </sig_tagout>
      </input_DP_ABV>
      <input_controlOverlayData_ptr link = "input_controlOverlayData"/>
      <input_controlOverlayData>
         <initializedFlag boolean = "qg_true"/>
      </input_controlOverlayData>
      <!-- Initialize counter values for functions called by the UUT -->
      <rg_e_DP_isValid_cntr value = "0"/>
      <rg_e_DP_ABV1_controlOverlayInit_cntr value = "0"/>
      <!-- rg_e_CO_button_inactivate_cntr is used in the wrapper as an array index and should star
t at 0 -->
      <rg_e_CO_button_inactivate_cntr value = "0"/>
      <!-- rg_e_CO_button_activate_cntr is used in the wrapper as an array index and should start
at 0 -->
      <rg_e_CO_button_activate_cntr value = "0"/>
      <!-- return values of functions called by UUT -->
      <e_DP_isValid_return boolean = "qg_true"/>
   </TestCase20>

   <TestCase21>
      <!-- DP_ABV pointer non-zero
         controlOverlayData pointer non-zero
         rlp_DP_ABV->intializedFlag is false
         rlp_DP_ABV->common.networkInputsValid is false
         rlp_DP_ABV->common.tagPermitted is true
         e_DP_isValid returns true
      -->
      <input_DP_ABV_ptr link = "input_DP_ABV"/>
      <input_DP_ABV>
         <common>
            <networkInputsValid boolean = "qg_false"/>
            <tagPermitted boolean = "qg_true"/>
```

```
         </common>
      </input_DP_ABV>
      <input_controlOverlayData_ptr link = "input_controlOverlayData"/>
      <input_controlOverlayData>
         <initializedFlag boolean = "qq_false"/>
      </input_controlOverlayData>
      <!-- Initialize counter values for functions called by the UUT -->
      <rg_e_DP_isValid_cntr value = "0" />
      <rg_e_DP_ABV1_controlOverlayInit_cntr value = "0"/>
      <!-- rg_e_CO_button_inactivate_cntr is used in the wrapper as an array index and should star
t at 0 -->
      <rg_e_CO_button_inactivate_cntr value = "0"/>
      <!-- rg_e_CO_button_activate_cntr is used in the wrapper as an array index and should start
at 0 -->
      <rg_e_CO_button_activate_cntr value = "0"/>
      <!-- return values of functions called by UUT -->
      <e_DP_isValid_return boolean = "qq_true"/>
   </TestCase21>

</TestSuite>
```

四.3 Expected Output程式

```
        05/31/2007   10:40:29 AM   t_e_DP_ABV1_configureControlOverlay_ExpectedOutput.xml
<?xml version="1.0" encoding="UTF-8"?>
<!--

   $Log$
   Revision 1.4.2.1  2007/03/26 22:09:31  hills
   Unit test updated for Group1 Eng9.
   For bad N/W inputs, the tag permit buttons are not enabled if tagPermit is false.

   Revision 1.3.120.2  2006/08/10 22:08:57  hills
   Updated to test the changed tag-off button mask. Eng7, DPDS is CD40 ( Att T 68 )

   Revision 1.3.120.1  2006/08/10 19:48:57  bala
   Brought in Files from Eng5 and made mods to ensure that the unit test builds and
   runs.

   Revision 1.3.64.2  2006/01/23 17:51:22  krishb
   Wrapper done, tests complete. Ready for review.

-->
<TestSuite>

   <TestCase01>
      <!-- Input DP_ABV gets copied to expected DP_ABV in wrapper -->
      <!-- Input controlOverlayData gets copied to expected controlOverlayData in wrapper -->
      <!-- Read in expected counter values for functions called by the UUT -->
      <ru_expected_e_DP_isValid_cntr value = "1"/>
      <ru_expected_e_DP_ABV1_controlOverlayInit_cntr value = "0"/>
      <ru_expected_e_CO_button_inactivate_cntr value = "0"/>
      <ru_expected_e_CO_button_activate_cntr value = "0"/>
      <!-- expected value(s) for args of functions called by the UUT -->
      <expected_e_DP_isValid_rl_algorithm enum = "qg_DP_ABV1"/>
   </TestCase01>

   <TestCase02>
      <!-- Input DP_ABV gets copied to expected DP_ABV in wrapper -->
      <expected_controlOverlayData_ptr enum = "qg_null_ptr"/>
      <!-- Read in expected counter values for functions called by the UUT -->
      <ru_expected_e_DP_isValid_cntr value = "1"/>
      <ru_expected_e_DP_ABV1_controlOverlayInit_cntr value = "0"/>
      <ru_expected_e_CO_button_inactivate_cntr value = "0"/>
      <ru_expected_e_CO_button_activate_cntr value = "0"/>
      <!-- expected value(s) for args of functions called by the UUT -->
      <expected_e_DP_isValid_rl_algorithm enum = "qg_DP_ABV1"/>
   </TestCase02>

   <TestCase03>
      <!-- Input DP_ABV gets copied to expected DP_ABV in wrapper -->
      <!-- Input controlOverlayData gets copied to expected controlOverlayData in wrapper -->
      <!-- Read in expected counter values for functions called by the UUT -->
      <ru_expected_e_DP_isValid_cntr value = "1"/>
      <ru_expected_e_DP_ABV1_controlOverlayInit_cntr value = "1"/>
      <ru_expected_e_CO_button_inactivate_cntr value = "4"/>
      <ru_expected_e_CO_button_activate_cntr value = "2"/>
      <!-- expected value(s) for args of functions called by the UUT -->
      <expected_e_DP_isValid_rl_algorithm enum = "qg_DP_ABV1"/>
      <!-- identify which buttons are passed to e_CO_button_inactivate -->
      <e_CO_button_inactivate_call array = "e_CO_button_inactivate">
         <e_CO_button_inactivate_Dimension value = "4"/>
         <e_CO_button_inactivate_01 value = "ru_openButton"/>
         <e_CO_button_inactivate_02 value = "ru_closeButton"/>
         <e_CO_button_inactivate_03 value = "ru_tagOnButton"/>
         <e_CO_button_inactivate_04 value = "ru_tagOffButton"/>
      </e_CO_button_inactivate_call>
      <!-- identify which buttons are passed to e_CO_button_activate -->
      <e_CO_button_activate_call array = "e_CO_button_activate">
         <e_CO_button_activate_Dimension value = "2"/>
         <e_CO_button_activate_01 value = "ru_openButton"/>
         <e_CO_button_activate_02 value = "ru_closeButton"/>
      </e_CO_button_activate_call>
   </TestCase03>

   <TestCase04>
      <!-- Input DP_ABV gets copied to expected DP_ABV in wrapper -->
```

```
    <!-- Input controlOverlayData gets copied to expected controlOverlayData in wrapper -->
    <!-- Read in expected counter values for functions called by the UUT -->
    <ru_expected_e_DP_isValid_cntr value = "1"/>
    <ru_expected_e_DP_ABV1_controlOverlayInit_cntr value = "0"/>
    <ru_expected_e_CO_button_inactivate_cntr value = "4"/>
    <ru_expected_e_CO_button_activate_cntr value = "3"/>
    <!-- expected value(s) for args of functions called by the UUT -->
    <expected_e_DP_isValid_rl_algorithm enum = "qg_DP_ABV1"/>
    <!-- identify which buttons are passed to e_CO_button_inactivate -->
    <e_CO_button_inactivate_call array = "e_CO_button_inactivate">
       <e_CO_button_inactivate_Dimension value = "4"/>
       <e_CO_button_inactivate_01 value = "ru_openButton"/>
       <e_CO_button_inactivate_02 value = "ru_closeButton"/>
       <e_CO_button_inactivate_03 value = "ru_tagOnButton"/>
       <e_CO_button_inactivate_04 value = "ru_tagOffButton"/>
    </e_CO_button_inactivate_call>
    <!-- identify which buttons are passed to e_CO_button_activate -->
    <e_CO_button_activate_call array = "e_CO_button_activate">
       <e_CO_button_activate_Dimension value = "3"/>
       <e_CO_button_activate_01 value = "ru_openButton"/>
       <e_CO_button_activate_02 value = "ru_closeButton"/>
       <e_CO_button_activate_03 value = "ru_tagOnButton"/>
    </e_CO_button_activate_call>
</TestCase04>

<TestCase05>
    <!-- Input DP_ABV gets copied to expected DP_ABV in wrapper -->
    <!-- Input controlOverlayData gets copied to expected controlOverlayData in wrapper -->
    <!-- Read in expected counter values for functions called by the UUT -->
    <ru_expected_e_DP_isValid_cntr value = "1"/>
    <ru_expected_e_DP_ABV1_controlOverlayInit_cntr value = "0"/>
    <ru_expected_e_CO_button_inactivate_cntr value = "4"/>
    <ru_expected_e_CO_button_activate_cntr value = "1"/>
    <!-- expected value(s) for args of functions called by the UUT -->
    <expected_e_DP_isValid_rl_algorithm enum = "qg_DP_ABV1"/>
    <!-- identify which buttons are passed to e_CO_button_inactivate -->
    <e_CO_button_inactivate_call array = "e_CO_button_inactivate">
       <e_CO_button_inactivate_Dimension value = "4"/>
       <e_CO_button_inactivate_01 value = "ru_openButton"/>
       <e_CO_button_inactivate_02 value = "ru_closeButton"/>
       <e_CO_button_inactivate_03 value = "ru_tagOnButton"/>
       <e_CO_button_inactivate_04 value = "ru_tagOffButton"/>
    </e_CO_button_inactivate_call>
    <!-- identify which buttons are passed to e_CO_button_activate -->
    <e_CO_button_activate_call array = "e_CO_button_activate">
       <e_CO_button_activate_Dimension value = "1"/>
       <e_CO_button_activate_01 value = "ru_tagOffButton"/>
    </e_CO_button_activate_call>
</TestCase05>

<TestCase06>
    <!-- Input DP_ABV gets copied to expected DP_ABV in wrapper -->
    <!-- Input controlOverlayData gets copied to expected controlOverlayData in wrapper -->
    <!-- Read in expected counter values for functions called by the UUT -->
    <ru_expected_e_DP_isValid_cntr value = "1"/>
    <ru_expected_e_DP_ABV1_controlOverlayInit_cntr value = "0"/>
    <ru_expected_e_CO_button_inactivate_cntr value = "4"/>
    <ru_expected_e_CO_button_activate_cntr value = "1"/>
    <!-- expected value(s) for args of functions called by the UUT -->
    <expected_e_DP_isValid_rl_algorithm enum = "qg_DP_ABV1"/>
    <!-- identify which buttons are passed to e_CO_button_inactivate -->
    <e_CO_button_inactivate_call array = "e_CO_button_inactivate">
       <e_CO_button_inactivate_Dimension value = "4"/>
       <e_CO_button_inactivate_01 value = "ru_openButton"/>
       <e_CO_button_inactivate_02 value = "ru_closeButton"/>
       <e_CO_button_inactivate_03 value = "ru_tagOnButton"/>
       <e_CO_button_inactivate_04 value = "ru_tagOffButton"/>
    </e_CO_button_inactivate_call>
    <!-- identify which buttons are passed to e_CO_button_activate -->
    <e_CO_button_activate_call array = "e_CO_button_activate">
       <e_CO_button_activate_Dimension value = "1"/>
       <e_CO_button_activate_01 value = "ru_closeButton"/>
    </e_CO_button_activate_call>
```

```
</TestCase06>

<TestCase07>
   <!-- Input DP_ABV gets copied to expected DP_ABV in wrapper -->
   <!-- Input controlOverlayData gets copied to expected controlOverlayData in wrapper -->
   <!-- Read in expected counter values for functions called by the UUT -->
   <ru_expected_e_DP_isValid_cntr value = "1"/>
   <ru_expected_e_DP_ABV1_controlOverlayInit_cntr value = "0"/>
   <ru_expected_e_CO_button_inactivate_cntr value = "4"/>
   <ru_expected_e_CO_button_activate_cntr value = "1"/>
   <!-- expected value(s) for args of functions called by the UUT -->
   <expected_e_DP_isValid_rl_algorithm enum = "qg_DP_ABV1"/>
   <!-- identify which buttons are passed to e_CO_button_inactivate -->
   <e_CO_button_inactivate_call array = "e_CO_button_inactivate">
      <e_CO_button_inactivate_Dimension value = "4"/>
      <e_CO_button_inactivate_01 value = "ru_openButton"/>
      <e_CO_button_inactivate_02 value = "ru_closeButton"/>
      <e_CO_button_inactivate_03 value = "ru_tagOnButton"/>
      <e_CO_button_inactivate_04 value = "ru_tagOffButton"/>
   </e_CO_button_inactivate_call>
   <!-- identify which buttons are passed to e_CO_button_activate -->
   <e_CO_button_activate_call array = "e_CO_button_activate">
      <e_CO_button_activate_Dimension value = "1"/>
      <e_CO_button_activate_01 value = "ru_tagOffButton"/>
   </e_CO_button_activate_call>
</TestCase07>

<TestCase08>
   <!-- Input DP_ABV gets copied to expected DP_ABV in wrapper -->
   <!-- Input controlOverlayData gets copied to expected controlOverlayData in wrapper -->
   <!-- Read in expected counter values for functions called by the UUT -->
   <ru_expected_e_DP_isValid_cntr value = "1"/>
   <ru_expected_e_DP_ABV1_controlOverlayInit_cntr value = "0"/>
   <ru_expected_e_CO_button_inactivate_cntr value = "4"/>
   <ru_expected_e_CO_button_activate_cntr value = "2"/>
   <!-- expected value(s) for args of functions called by the UUT -->
   <expected_e_DP_isValid_rl_algorithm enum = "qg_DP_ABV1"/>
   <!-- identify which buttons are passed to e_CO_button_inactivate -->
   <e_CO_button_inactivate_call array = "e_CO_button_inactivate">
      <e_CO_button_inactivate_Dimension value = "4"/>
      <e_CO_button_inactivate_01 value = "ru_openButton"/>
      <e_CO_button_inactivate_02 value = "ru_closeButton"/>
      <e_CO_button_inactivate_03 value = "ru_tagOnButton"/>
      <e_CO_button_inactivate_04 value = "ru_tagOffButton"/>
   </e_CO_button_inactivate_call>
   <!-- identify which buttons are passed to e_CO_button_activate -->
   <e_CO_button_activate_call array = "e_CO_button_activate">
      <e_CO_button_activate_Dimension value = "2"/>
      <e_CO_button_activate_01 value = "ru_openButton"/>
      <e_CO_button_activate_02 value = "ru_tagOnButton"/>
   </e_CO_button_activate_call>
</TestCase08>

<TestCase09>
   <!-- Input DP_ABV gets copied to expected DP_ABV in wrapper -->
   <!-- Input controlOverlayData gets copied to expected controlOverlayData in wrapper -->
   <!-- Read in expected counter values for functions called by the UUT -->
   <ru_expected_e_DP_isValid_cntr value = "1"/>
   <ru_expected_e_DP_ABV1_controlOverlayInit_cntr value = "0"/>
   <ru_expected_e_CO_button_inactivate_cntr value = "4"/>
   <ru_expected_e_CO_button_activate_cntr value = "1"/>
   <!-- expected value(s) for args of functions called by the UUT -->
   <expected_e_DP_isValid_rl_algorithm enum = "qg_DP_ABV1"/>
   <!-- identify which buttons are passed to e_CO_button_inactivate -->
   <e_CO_button_inactivate_call array = "e_CO_button_inactivate">
      <e_CO_button_inactivate_Dimension value = "4"/>
      <e_CO_button_inactivate_01 value = "ru_openButton"/>
      <e_CO_button_inactivate_02 value = "ru_closeButton"/>
      <e_CO_button_inactivate_03 value = "ru_tagOnButton"/>
      <e_CO_button_inactivate_04 value = "ru_tagOffButton"/>
   </e_CO_button_inactivate_call>
   <!-- identify which buttons are passed to e_CO_button_activate -->
   <e_CO_button_activate_call array = "e_CO_button_activate">
```

```
      <e_CO_button_activate_Dimension value = "1"/>
      <e_CO_button_activate_01 value = "ru_tagOffButton"/>
   </e_CO_button_activate_call>
</TestCase09>

<TestCase10>
   <!-- Input DP_ABV gets copied to expected DP_ABV in wrapper -->
   <!-- Input controlOverlayData gets copied to expected controlOverlayData in wrapper -->
   <!-- Read in expected counter values for functions called by the UUT -->
   <ru_expected_e_DP_isValid_cntr value = "1"/>
   <ru_expected_e_DP_ABV1_controlOverlayInit_cntr value = "0"/>
   <ru_expected_e_CO_button_inactivate_cntr value = "4"/>
   <ru_expected_e_CO_button_activate_cntr value = "1"/>
   <!-- expected value(s) for args of functions called by the UUT -->
   <expected_e_DP_isValid_rl_algorithm enum = "qg_DP_ABV1"/>
   <!-- identify which buttons are passed to e_CO_button_inactivate -->
   <e_CO_button_inactivate_call array = "e_CO_button_inactivate">
      <e_CO_button_inactivate_Dimension value = "4"/>
      <e_CO_button_inactivate_01 value = "ru_openButton"/>
      <e_CO_button_inactivate_02 value = "ru_closeButton"/>
      <e_CO_button_inactivate_03 value = "ru_tagOnButton"/>
      <e_CO_button_inactivate_04 value = "ru_tagOffButton"/>
   </e_CO_button_inactivate_call>
   <!-- identify which buttons are passed to e_CO_button_activate -->
   <e_CO_button_activate_call array = "e_CO_button_activate">
      <e_CO_button_activate_Dimension value = "1"/>
      <e_CO_button_activate_01 value = "ru_closeButton"/>
   </e_CO_button_activate_call>
</TestCase10>

<TestCase11>
   <!-- Input DP_ABV gets copied to expected DP_ABV in wrapper -->
   <!-- Input controlOverlayData gets copied to expected controlOverlayData in wrapper -->
   <!-- Read in expected counter values for functions called by the UUT -->
   <ru_expected_e_DP_isValid_cntr value = "1"/>
   <ru_expected_e_DP_ABV1_controlOverlayInit_cntr value = "0"/>
   <ru_expected_e_CO_button_inactivate_cntr value = "4"/>
   <ru_expected_e_CO_button_activate_cntr value = "1"/>
   <!-- expected value(s) for args of functions called by the UUT -->
   <expected_e_DP_isValid_rl_algorithm enum = "qg_DP_ABV1"/>
   <!-- identify which buttons are passed to e_CO_button_inactivate -->
   <e_CO_button_inactivate_call array = "e_CO_button_inactivate">
      <e_CO_button_inactivate_Dimension value = "4"/>
      <e_CO_button_inactivate_01 value = "ru_openButton"/>
      <e_CO_button_inactivate_02 value = "ru_closeButton"/>
      <e_CO_button_inactivate_03 value = "ru_tagOnButton"/>
      <e_CO_button_inactivate_04 value = "ru_tagOffButton"/>
   </e_CO_button_inactivate_call>
   <!-- identify which buttons are passed to e_CO_button_activate -->
   <e_CO_button_activate_call array = "e_CO_button_activate">
      <e_CO_button_activate_Dimension value = "1"/>
      <e_CO_button_activate_01 value = "ru_tagOffButton"/>
   </e_CO_button_activate_call>
</TestCase11>

<TestCase12>
   <!-- Input DP_ABV gets copied to expected DP_ABV in wrapper -->
   <!-- Input controlOverlayData gets copied to expected controlOverlayData in wrapper -->
   <!-- Read in expected counter values for functions called by the UUT -->
   <ru_expected_e_DP_isValid_cntr value = "1"/>
   <ru_expected_e_DP_ABV1_controlOverlayInit_cntr value = "0"/>
   <ru_expected_e_CO_button_inactivate_cntr value = "4"/>
   <ru_expected_e_CO_button_activate_cntr value = "1"/>
   <!-- expected value(s) for args of functions called by the UUT -->
   <expected_e_DP_isValid_rl_algorithm enum = "qg_DP_ABV1"/>
   <!-- identify which buttons are passed to e_CO_button_inactivate -->
   <e_CO_button_inactivate_call array = "e_CO_button_inactivate">
      <e_CO_button_inactivate_Dimension value = "4"/>
      <e_CO_button_inactivate_01 value = "ru_openButton"/>
      <e_CO_button_inactivate_02 value = "ru_closeButton"/>
      <e_CO_button_inactivate_03 value = "ru_tagOnButton"/>
      <e_CO_button_inactivate_04 value = "ru_tagOffButton"/>
   </e_CO_button_inactivate_call>
```

```
   <!-- identify which buttons are passed to e_CO_button_activate -->
   <e_CO_button_activate_call array = "e_CO_button_activate">
      <e_CO_button_activate_Dimension value = "1"/>
      <e_CO_button_activate_01 value = "ru_closeButton"/>
   </e_CO_button_activate_call>
</TestCase12>

<TestCase13>
   <!-- Input DP_ABV gets copied to expected DP_ABV in wrapper -->
   <!-- Input controlOverlayData gets copied to expected controlOverlayData in wrapper -->
   <!-- Read in expected counter values for functions called by the UUT -->
   <ru_expected_e_DP_isValid_cntr value = "1"/>
   <ru_expected_e_DP_ABV1_controlOverlayInit_cntr value = "0"/>
   <ru_expected_e_CO_button_inactivate_cntr value = "4"/>
   <ru_expected_e_CO_button_activate_cntr value = "1"/>
   <!-- expected value(s) for args of functions called by the UUT -->
   <expected_e_DP_isValid_rl_algorithm enum = "qg_DP_ABV1"/>
   <!-- identify which buttons are passed to e_CO_button_inactivate -->
   <e_CO_button_inactivate_call array = "e_CO_button_inactivate">
      <e_CO_button_inactivate_Dimension value = "4"/>
      <e_CO_button_inactivate_01 value = "ru_openButton"/>
      <e_CO_button_inactivate_02 value = "ru_closeButton"/>
      <e_CO_button_inactivate_03 value = "ru_tagOnButton"/>
      <e_CO_button_inactivate_04 value = "ru_tagOffButton"/>
   </e_CO_button_inactivate_call>
   <!-- identify which buttons are passed to e_CO_button_activate -->
   <e_CO_button_activate_call array = "e_CO_button_activate">
      <e_CO_button_activate_Dimension value = "1"/>
      <e_CO_button_activate_01 value = "ru_tagOffButton"/>
   </e_CO_button_activate_call>
</TestCase13>

<TestCase14>
   <!-- Input DP_ABV gets copied to expected DP_ABV in wrapper -->
   <!-- Input controlOverlayData gets copied to expected controlOverlayData in wrapper -->
   <!-- Read in expected counter values for functions called by the UUT -->
   <ru_expected_e_DP_isValid_cntr value = "1"/>
   <ru_expected_e_DP_ABV1_controlOverlayInit_cntr value = "0"/>
   <ru_expected_e_CO_button_inactivate_cntr value = "4"/>
   <ru_expected_e_CO_button_activate_cntr value = "1"/>
   <!-- expected value(s) for args of functions called by the UUT -->
   <expected_e_DP_isValid_rl_algorithm enum = "qg_DP_ABV1"/>
   <!-- identify which buttons are passed to e_CO_button_inactivate -->
   <e_CO_button_inactivate_call array = "e_CO_button_inactivate">
      <e_CO_button_inactivate_Dimension value = "4"/>
      <e_CO_button_inactivate_01 value = "ru_openButton"/>
      <e_CO_button_inactivate_02 value = "ru_closeButton"/>
      <e_CO_button_inactivate_03 value = "ru_tagOnButton"/>
      <e_CO_button_inactivate_04 value = "ru_tagOffButton"/>
   </e_CO_button_inactivate_call>
   <!-- identify which buttons are passed to e_CO_button_activate -->
   <e_CO_button_activate_call array = "e_CO_button_activate">
      <e_CO_button_activate_Dimension value = "1"/>
      <e_CO_button_activate_01 value = "ru_closeButton"/>
   </e_CO_button_activate_call>
</TestCase14>

<TestCase15>
   <!-- Input DP_ABV gets copied to expected DP_ABV in wrapper -->
   <!-- Input controlOverlayData gets copied to expected controlOverlayData in wrapper -->
   <!-- Read in expected counter values for functions called by the UUT -->
   <ru_expected_e_DP_isValid_cntr value = "1"/>
   <ru_expected_e_DP_ABV1_controlOverlayInit_cntr value = "0"/>
   <ru_expected_e_CO_button_inactivate_cntr value = "4"/>
   <ru_expected_e_CO_button_activate_cntr value = "1"/>
   <!-- expected value(s) for args of functions called by the UUT -->
   <expected_e_DP_isValid_rl_algorithm enum = "qg_DP_ABV1"/>
   <!-- identify which buttons are passed to e_CO_button_inactivate -->
   <e_CO_button_inactivate_call array = "e_CO_button_inactivate">
      <e_CO_button_inactivate_Dimension value = "4"/>
      <e_CO_button_inactivate_01 value = "ru_openButton"/>
      <e_CO_button_inactivate_02 value = "ru_closeButton"/>
      <e_CO_button_inactivate_03 value = "ru_tagOnButton"/>
```

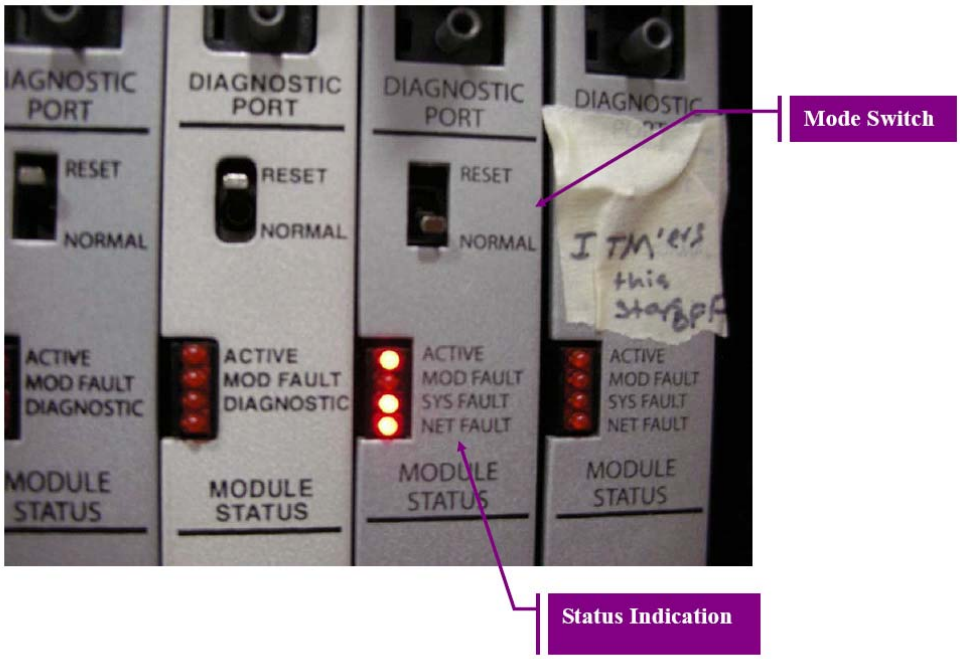
```
      <e_CO_button_inactivate_04 value = "ru_tagOffButton"/>
   </e_CO_button_inactivate_call>
   <!-- identify which buttons are passed to e_CO_button_activate -->
   <e_CO_button_activate_call array = "e_CO_button_activate">
      <e_CO_button_activate_Dimension value = "1"/>
      <e_CO_button_activate_01 value = "ru_tagOffButton"/>
   </e_CO_button_activate_call>
</TestCase15>

<TestCase16>
   <!-- Input DP_ABV gets copied to expected DP_ABV in wrapper -->
   <!-- Input controlOverlayData gets copied to expected controlOverlayData in wrapper -->
   <!-- Read in expected counter values for functions called by the UUT -->
   <ru_expected_e_DP_isValid_cntr value = "1"/>
   <ru_expected_e_DP_ABV1_controlOverlayInit_cntr value = "0"/>
   <ru_expected_e_CO_button_inactivate_cntr value = "4"/>
   <ru_expected_e_CO_button_activate_cntr value = "0"/>
   <!-- expected value(s) for args of functions called by the UUT -->
   <expected_e_DP_isValid_rl_algorithm enum = "qg_DP_ABV1"/>
   <!-- identify which buttons are passed to e_CO_button_inactivate -->
   <e_CO_button_inactivate_call array = "e_CO_button_inactivate">
      <e_CO_button_inactivate_Dimension value = "4"/>
      <e_CO_button_inactivate_01 value = "ru_openButton"/>
      <e_CO_button_inactivate_02 value = "ru_closeButton"/>
      <e_CO_button_inactivate_03 value = "ru_tagOnButton"/>
      <e_CO_button_inactivate_04 value = "ru_tagOffButton"/>
   </e_CO_button_inactivate_call>
   <!-- identify which buttons are passed to e_CO_button_activate -->
   <e_CO_button_activate_call array = "e_CO_button_activate">
      <e_CO_button_activate_Dimension value = "0"/>
   </e_CO_button_activate_call>
</TestCase16>

<TestCase17>
   <!-- Input DP_ABV gets copied to expected DP_ABV in wrapper -->
   <!-- Input controlOverlayData gets copied to expected controlOverlayData in wrapper -->
   <!-- Read in expected counter values for functions called by the UUT -->
   <ru_expected_e_DP_isValid_cntr value = "1"/>
   <ru_expected_e_DP_ABV1_controlOverlayInit_cntr value = "0"/>
   <ru_expected_e_CO_button_inactivate_cntr value = "4"/>
   <ru_expected_e_CO_button_activate_cntr value = "1"/>
   <!-- expected value(s) for args of functions called by the UUT -->
   <expected_e_DP_isValid_rl_algorithm enum = "qg_DP_ABV1"/>
   <!-- identify which buttons are passed to e_CO_button_inactivate -->
   <e_CO_button_inactivate_call array = "e_CO_button_inactivate">
      <e_CO_button_inactivate_Dimension value = "4"/>
      <e_CO_button_inactivate_01 value = "ru_openButton"/>
      <e_CO_button_inactivate_02 value = "ru_closeButton"/>
      <e_CO_button_inactivate_03 value = "ru_tagOnButton"/>
      <e_CO_button_inactivate_04 value = "ru_tagOffButton"/>
   </e_CO_button_inactivate_call>
   <!-- identify which buttons are passed to e_CO_button_activate -->
   <e_CO_button_activate_call array = "e_CO_button_activate">
      <e_CO_button_activate_Dimension value = "1"/>
      <e_CO_button_activate_01 value = "ru_tagOffButton"/>
   </e_CO_button_activate_call>
</TestCase17>

<TestCase18>
   <!-- Input DP_ABV gets copied to expected DP_ABV in wrapper -->
   <!-- Input controlOverlayData gets copied to expected controlOverlayData in wrapper -->
   <!-- Read in expected counter values for functions called by the UUT -->
   <ru_expected_e_DP_isValid_cntr value = "1"/>
   <ru_expected_e_DP_ABV1_controlOverlayInit_cntr value = "0"/>
   <ru_expected_e_CO_button_inactivate_cntr value = "4"/>
   <ru_expected_e_CO_button_activate_cntr value = "1"/>
   <!-- expected value(s) for args of functions called by the UUT -->
   <expected_e_DP_isValid_rl_algorithm enum = "qg_DP_ABV1"/>
   <!-- identify which buttons are passed to e_CO_button_inactivate -->
   <e_CO_button_inactivate_call array = "e_CO_button_inactivate">
      <e_CO_button_inactivate_Dimension value = "4"/>
      <e_CO_button_inactivate_01 value = "ru_openButton"/>
      <e_CO_button_inactivate_02 value = "ru_closeButton"/>
```

```
      <e_CO_button_inactivate_03 value = "ru_tagOnButton"/>
      <e_CO_button_inactivate_04 value = "ru_tagOffButton"/>
   </e_CO_button_inactivate_call>
   <!-- identify which buttons are passed to e_CO_button_activate -->
   <e_CO_button_activate_call array = "e_CO_button_activate">
      <e_CO_button_activate_Dimension value = "1"/>
      <e_CO_button_activate_01 value = "ru_closeButton"/>
   </e_CO_button_activate_call>
</TestCase18>

<TestCase19>
   <!-- Input DP_ABV gets copied to expected DP_ABV in wrapper -->
   <!-- Input controlOverlayData gets copied to expected controlOverlayData in wrapper -->
   <!-- Read in expected counter values for functions called by the UUT -->
   <ru_expected_e_DP_isValid_cntr value = "1"/>
   <ru_expected_e_DP_ABV1_controlOverlayInit_cntr value = "0"/>
   <ru_expected_e_CO_button_inactivate_cntr value = "4"/>
   <ru_expected_e_CO_button_activate_cntr value = "1"/>
   <!-- expected value(s) for args of functions called by the UUT -->
   <expected_e_DP_isValid_rl_algorithm enum = "qg_DP_ABV1"/>
   <!-- identify which buttons are passed to e_CO_button_inactivate -->
   <e_CO_button_inactivate_call array = "e_CO_button_inactivate">
      <e_CO_button_inactivate_Dimension value = "4"/>
      <e_CO_button_inactivate_01 value = "ru_openButton"/>
      <e_CO_button_inactivate_02 value = "ru_closeButton"/>
      <e_CO_button_inactivate_03 value = "ru_tagOnButton"/>
      <e_CO_button_inactivate_04 value = "ru_tagOffButton"/>
   </e_CO_button_inactivate_call>
   <!-- identify which buttons are passed to e_CO_button_activate -->
   <e_CO_button_activate_call array = "e_CO_button_activate">
      <e_CO_button_activate_Dimension value = "1"/>
      <e_CO_button_activate_01 value = "ru_tagOffButton"/>
   </e_CO_button_activate_call>
</TestCase19>

<TestCase20>
   <!-- Input DP_ABV gets copied to expected DP_ABV in wrapper -->
   <!-- Input controlOverlayData gets copied to expected controlOverlayData in wrapper -->
   <!-- Read in expected counter values for functions called by the UUT -->
   <ru_expected_e_DP_isValid_cntr value = "1"/>
   <ru_expected_e_DP_ABV1_controlOverlayInit_cntr value = "0"/>
   <ru_expected_e_CO_button_inactivate_cntr value = "4"/>
   <ru_expected_e_CO_button_activate_cntr value = "0"/>
   <!-- expected value(s) for args of functions called by the UUT -->
   <expected_e_DP_isValid_rl_algorithm enum = "qg_DP_ABV1"/>
   <!-- identify which buttons are passed to e_CO_button_inactivate -->
   <e_CO_button_inactivate_call array = "e_CO_button_inactivate">
      <e_CO_button_inactivate_Dimension value = "4"/>
      <e_CO_button_inactivate_01 value = "ru_openButton"/>
      <e_CO_button_inactivate_02 value = "ru_closeButton"/>
      <e_CO_button_inactivate_03 value = "ru_tagOnButton"/>
      <e_CO_button_inactivate_04 value = "ru_tagOffButton"/>
   </e_CO_button_inactivate_call>
   <!-- identify which buttons are passed to e_CO_button_activate -->
   <e_CO_button_activate_call array = "e_CO_button_activate">
      <e_CO_button_activate_Dimension value = "0"/>
   </e_CO_button_activate_call>
</TestCase20>

<TestCase21>
   <!-- Input DP_ABV gets copied to expected DP_ABV in wrapper -->
   <!-- Input controlOverlayData gets copied to expected controlOverlayData in wrapper -->
   <!-- Read in expected counter values for functions called by the UUT -->
   <ru_expected_e_DP_isValid_cntr value = "1"/>
   <ru_expected_e_DP_ABV1_controlOverlayInit_cntr value = "1"/>
   <ru_expected_e_CO_button_inactivate_cntr value = "4"/>
   <ru_expected_e_CO_button_activate_cntr value = "4"/>
   <!-- expected value(s) for args of functions called by the UUT -->
   <expected_e_DP_isValid_rl_algorithm enum = "qg_DP_ABV1"/>
   <!-- identify which buttons are passed to e_CO_button_inactivate -->
   <e_CO_button_inactivate_call array = "e_CO_button_inactivate">
      <e_CO_button_inactivate_Dimension value = "4"/>
      <e_CO_button_inactivate_01 value = "ru_openButton"/>
```

```
        <e_CO_button_inactivate_02 value = "ru_closeButton"/>
        <e_CO_button_inactivate_03 value = "ru_tagOnButton"/>
        <e_CO_button_inactivate_04 value = "ru_tagOffButton"/>
      </e_CO_button_inactivate_call>
      <!-- identify which buttons are passed to e_CO_button_activate -->
      <e_CO_button_activate_call array = "e_CO_button_activate">
        <e_CO_button_activate_Dimension value = "4"/>
        <e_CO_button_activate_01 value = "ru_openButton"/>
        <e_CO_button_activate_02 value = "ru_closeButton"/>
        <e_CO_button_activate_03 value = "ru_tagOnButton"/>
        <e_CO_button_activate_04 value = "ru_tagOffButton"/>
      </e_CO_button_activate_call>
   </TestCase21>

</TestSuite>
```

附件五 MODE SWITCH OF QUAD SWITCH

附件六 MODE SWITCH AND STATUS INDICATION



附件七 SKEW TESTING CABLE A LENGTHS

CABLE A ADJUSTMENT								
Step	Cable A #1	Cable A #2	Cable A Total	Cable B #1	Cable B #2	Cable B Total	Diff	Tested P/F
1	10	0	10	8	0	8	2	
2	12	0	12	8	0	8	4	
3	14	0	14	8	0	8	6	
4	16	0	16	8	0	8	8	
5	10	0	10	0	0	0	10	
6	12	0	12	0	0	0	12	
7	14	0	14	0	0	0	14	
8	16	0	16	0	0	0	16	
9	10	8	18	0	0	0	18	
10	12	8	20	0	0	0	20	
11	14	8	22	0	0	0	22	
12	16	8	24	0	0	0	24	
13	10	16	26	0	0	0	26	
14	12	16	28	0	0	0	28	
15	14	16	30	0	0	0	30	
16	45	0	45	16	0	16	29	
17	45	0	45	14	0	14	31	
18	45	0	45	12	0	12	33	
19	45	0	45	10	0	10	35	
20	45	0	45	8	0	8	37	
21	45	10	55	16	0	16	39	
22	45	12	57	16	0	16	41	

CABLE A ADJUSTMENT								
Step	Cable A #1	Cable A #2	Cable A Total	Cable B #1	Cable B #2	Cable B Total	Diff	Tested P/F
23	45	14	59	16	0	16	43	
24	45	0	45	0	0	0	45	
25	45	10	55	8	0	8	47	
26	45	12	57	8	0	8	49	
27	45	14	59	8	0	8	51	
28	45	8	53	0	0	0	53	
29	45	10	55	0	0	0	55	
30	45	12	57	0	0	0	57	
31	45	14	59	0	0	0	59	
32	45	16	61	0	0	0	61	
33	45	45	90	16	12	28	62	
34	45	45	90	16	10	26	64	
35	45	45	90	16	8	24	66	
36	45	45	90	14	8	22	68	
37	45	45	90	12	8	20	70	
38	45	45	90	10	8	18	72	
39	45	45	90	16	0	16	74	
40	45	45	90	14	0	14	76	
41	45	45	90	12	0	12	78	
42	45	45	90	10	0	10	80	
43	45	45	90	8	0	8	82	
44	45+10	45	100	16	0	16	84	
45	45+10	45	100	14	0	14	86	
46	45+10	45	100	12	0	12	88	

CABLE A ADJUSTMENT								
Step	Cable A #1	Cable A #2	Cable A Total	Cable B #1	Cable B #2	Cable B Total	Diff	Tested P/F
47	45	45	90	0	0	0	90	
48	45+16	45	106	14	0	0	92	
49	45+16	45	106	12	0	0	94	
50	45+16	45	106	10	0	0	96	
51	45+8	45	98	0	0	0	98	
52	45+10	45	100	0	0	0	100	
53	45+12	45	102	0	0	0	102	
54	45+14	45	104	0	0	0	104	
55	45+16	45	106	0	0	0	106	
56	45+10	45+8	108	0	0	0	108	
57	45+12	45+8	110	0	0	0	110	
58	45+14	45+8	112	0	0	0	112	
59	45+16	45+8	114	0	0	0	114	
60	45+16	45+10	116	0	0	0	116	
61	45+16	45+12	118	0	0	0	118	
62	45+16	45+14	120	0	0	0	120	

附件八 SKEW TESTING CABLE B LENGTHS

CABLE B ADJUSTMENT								
Step	Cable B #1	Cable B #2	Cable B Total	Cable A #1	Cable A #2	Cable A Total	Diff	Tested P/F
1	10	0	10	8	0	8	2	
2	12	0	12	8	0	8	4	
3	14	0	14	8	0	8	6	
4	16	0	16	8	0	8	8	
5	10	0	10	0	0	0	10	
6	12	0	12	0	0	0	12	
7	14	0	14	0	0	0	14	
8	16	0	16	0	0	0	16	
9	10	8	18	0	0	0	18	
10	12	8	20	0	0	0	20	
11	14	8	22	0	0	0	22	
12	16	8	24	0	0	0	24	
13	10	16	26	0	0	0	26	
14	12	16	28	0	0	0	28	
15	14	16	30	0	0	0	30	
16	45	0	45	16	0	16	29	
17	45	0	45	14	0	14	31	
18	45	0	45	12	0	12	33	
19	45	0	45	10	0	10	35	
20	45	0	45	8	0	8	37	
21	45	10	55	16	0	16	39	
22	45	12	57	16	0	16	41	

CABLE B ADJUSTMENT								
Step	Cable B #1	Cable B #2	Cable B Total	Cable A #1	Cable A #2	Cable A Total	Diff	Tested P/F
23	45	14	59	16	0	16	43	
24	45	0	45	0	0	0	45	
25	45	10	55	8	0	8	47	
26	45	12	57	8	0	8	49	
27	45	14	59	8	0	8	51	
28	45	8	53	0	0	0	53	
29	45	10	55	0	0	0	55	
30	45	12	57	0	0	0	57	
31	45	14	59	0	0	0	59	
32	45	16	61	0	0	0	61	
33	45	45	90	16	12	28	62	
34	45	45	90	16	10	26	64	
35	45	45	90	16	8	24	66	
36	45	45	90	14	8	22	68	
37	45	45	90	12	8	20	70	
38	45	45	90	10	8	18	72	
39	45	45	90	16	0	16	74	
40	45	45	90	14	0	14	76	
41	45	45	90	12	0	12	78	
42	45	45	90	10	0	10	80	
43	45	45	90	8	0	8	82	
44	45+10	45	100	16	0	16	84	
45	45+10	45	100	14	0	14	86	
46	45+10	45	100	12	0	12	88	

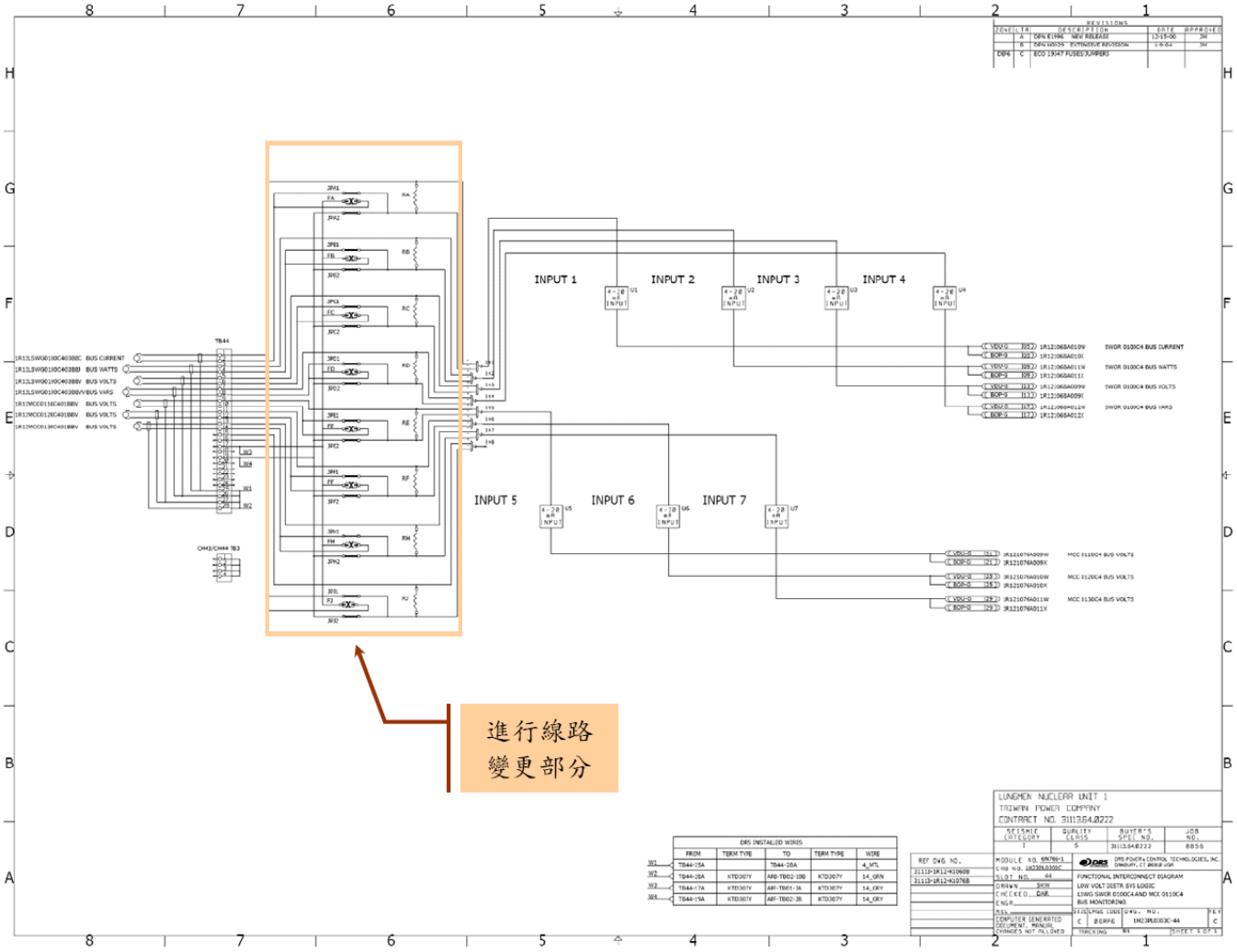
CABLE B ADJUSTMENT								
Step	Cable B #1	Cable B #2	Cable B Total	Cable A #1	Cable A #2	Cable A Total	Diff	Tested P/F
23	45	14	59	16	0	16	43	
24	45	0	45	0	0	0	45	
25	45	10	55	8	0	8	47	
26	45	12	57	8	0	8	49	
27	45	14	59	8	0	8	51	
28	45	8	53	0	0	0	53	
29	45	10	55	0	0	0	55	
30	45	12	57	0	0	0	57	
31	45	14	59	0	0	0	59	
32	45	16	61	0	0	0	61	
33	45	45	90	16	12	28	62	
34	45	45	90	16	10	26	64	
35	45	45	90	16	8	24	66	
36	45	45	90	14	8	22	68	
37	45	45	90	12	8	20	70	
38	45	45	90	10	8	18	72	
39	45	45	90	16	0	16	74	
40	45	45	90	14	0	14	76	
41	45	45	90	12	0	12	78	
42	45	45	90	10	0	10	80	
43	45	45	90	8	0	8	82	
44	45+10	45	100	16	0	16	84	
45	45+10	45	100	14	0	14	86	
46	45+10	45	100	12	0	12	88	

附件九 HARD INPUT POWER SOURCES CHANGED FID LIST

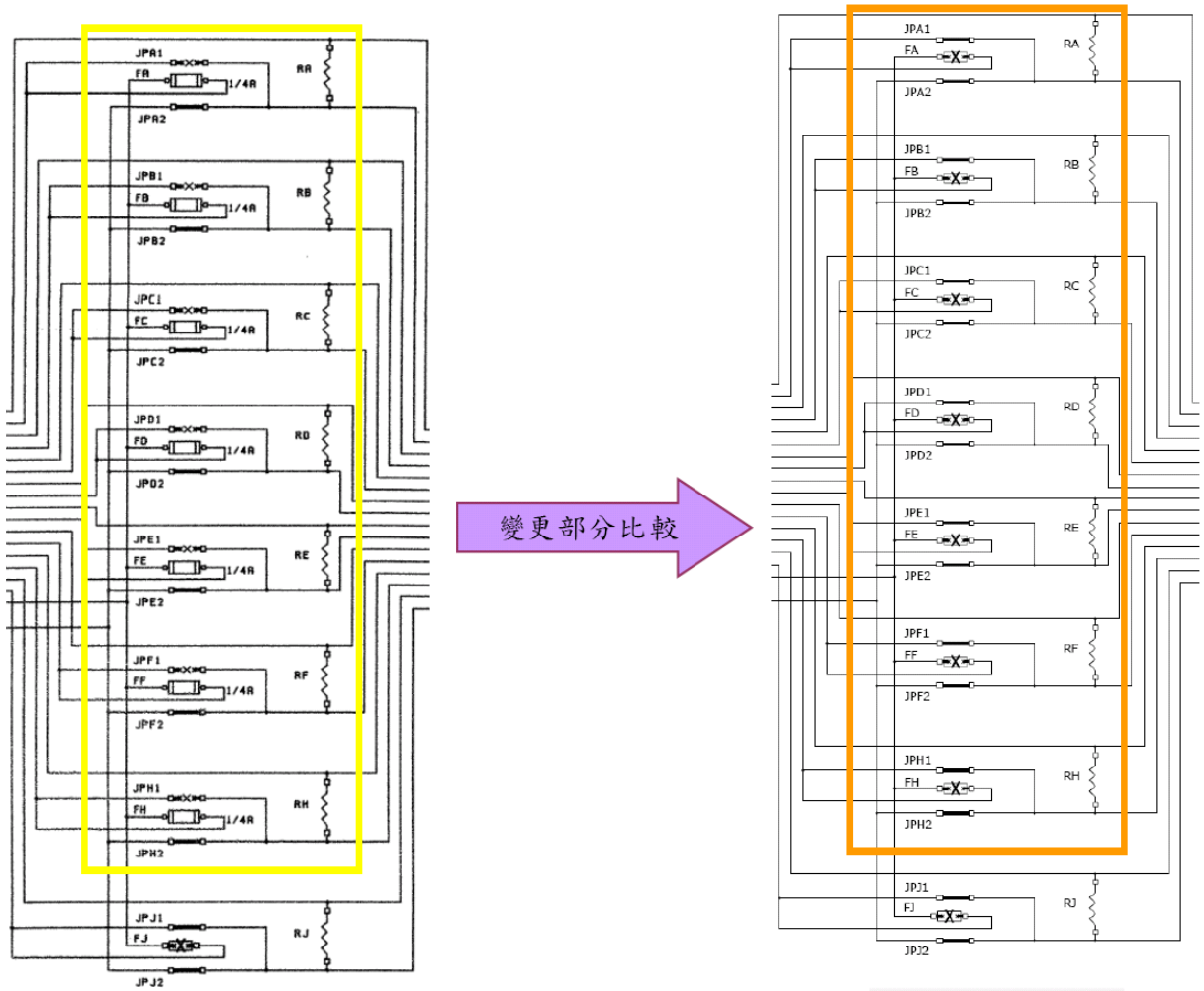
ITEM	FID	REVISION	SYSTEM	MODULE TYPE
1	1H23PL0302C-41	E	1T41	6N766-1
2	1H23PL0303C-44	C	1R12	6N766-1
3	1H23PL0305A-47	D	1R12	6N766-1
4	1H23PL0305B-35	D	1T41	6N766-1
5	1H23PL0306A-43	E	1T41	6N766-1
6	1H23PL0306B-45	C	1R12	6N766-1
7	1H23PL1301A-38	B	1R12	6N766-1
8	1H23PL1301A-41	C	1P21	6N766-1
9	1H23PL1301A-43	C	1P21	6N766-1
10	1H23PL1301B-41	C	1R12	6N766-1
11	1H23PL1302B-36	C	1T43	6N766-1
12	1H23PL1302B-37	C	1T43	6N766-1
13	1H23PL1302C-37	C	1T43	6N766-1
14	1H23PL1302C-43	D	1T43	6N766-1
15	1H23PL1302C-48	C	1R12	6N766-1
16	1H23PL1303B-40	C	1P21	6N766-1
17	1H23PL1303B-43	D	1P21	6N766-1
18	1H23PL1303C-41	C	1P21	6N766-1
19	1H23PL1303C-43	C	1P21	6N766-1
20	1H23PL1401A-05	F	1C74	6N762-1
21	1H23PL1401B-05	F	1C74	6N762-1
22	1H23PL1401C-04	C	1C74	6N762-1
23	1H23PL1401D-01	C	1C74	6N762-1

附件十 HARD INPUT POWER SOURCES CHANGE FID範例說明

十.1 以1H23PL0303C-44為例 (一)



十.2 以1H23PL0303C-44為例 (二)



變更前 (Rev. B)

變更後 (Rev. C)

附件十一 FID測試見證與進度說明

十一.1 概述

職本次奉派至廠家參與測試見證期間，主要見證項目為 3rd 與 4th FID test，另外亦參與部分 Cabinet Assembly Test、VDU Burn-in Test、VDU Software Unit Test 及 Module Test 等測試，本附件就 FID test 測試方式與進度作一說明。

原 DRS 設計之 FID 已經過 1st 與 2nd round 測試，然期間經歷過 GE 設計文件修改及設計變更，致使部分 FID 須進行修改或重新設計，也使得此類 FID 必須重新進行測試，於職至 DRS 見證期間，正值廠家進行 3rd 與 4th round FID test，FID 測試數量與時間說明如下：

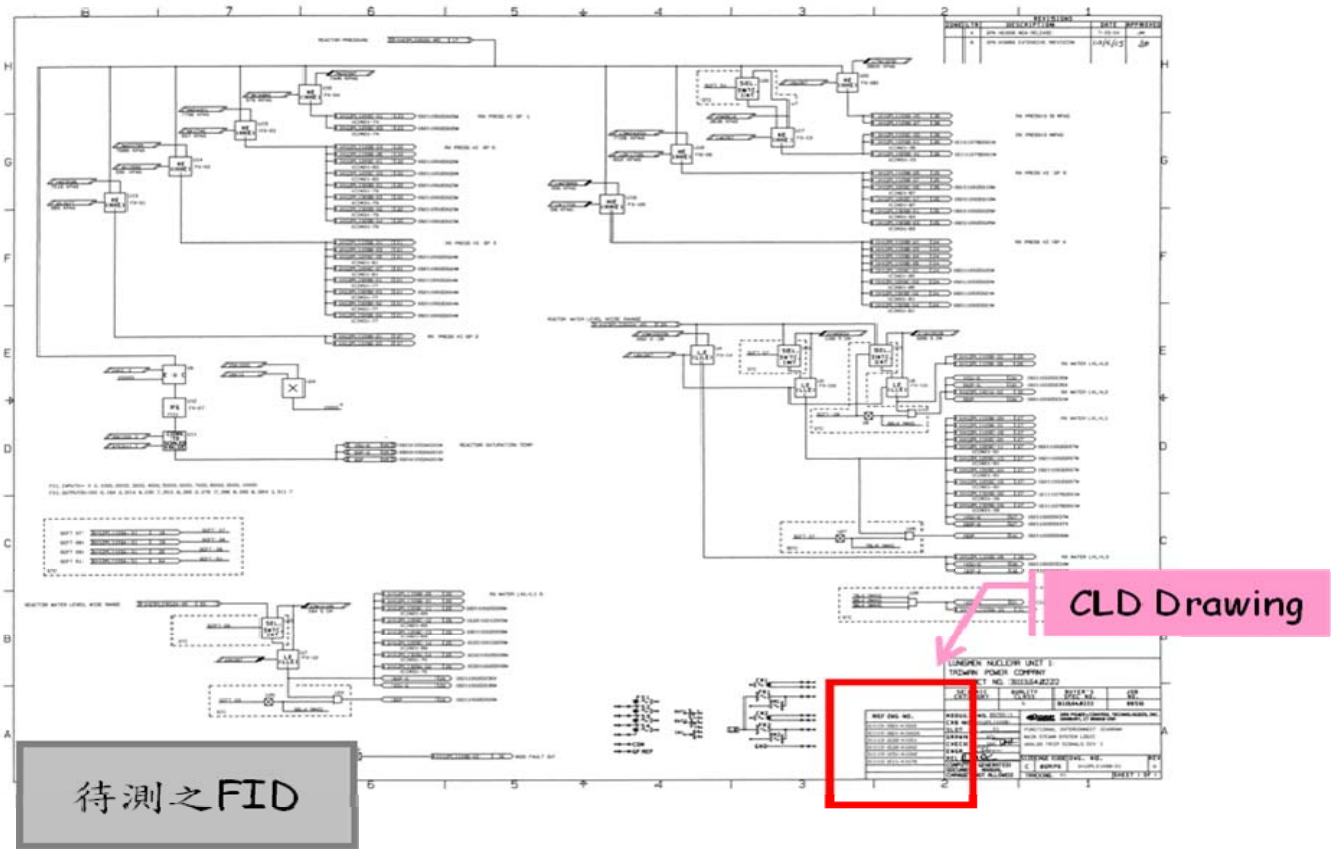
	Date		Total Quantity of FID
	Started	Completed	
1st round	Sep-2003	Dec-2004	1544
2nd round	Jan-2005	Jun-2006	1603
3rd round	Jan-2007	Feb-2007	302
4th round	Feb-2007	May-2007	100

十一.2 FID製作與測試流程說明

FID 乃 DRS 根據 GE 提供之邏輯圖，I/O Database，DCT 等多項資料庫，以 OrCAD 軟體繪製而成，主要做為現場設備邏輯處理與控制用。在 FID 製作過程中，當原始 OrCAD 檔案繪製完畢後，須經過編譯等過程，將相關之 Library、module 相關資訊及 COS(Control Operating System)一同燒入 EPROM 後，方成一完整可執行控制功能之 FID。在經過適當測試後，將該 EPROM 放置於適當 module 後便可進行現場設備控制。以下以 FID 1H12PL1109B-01 (rev.D)為例，說明 DRS 進行 FID 製作與測試情形，步驟說明如下：

1. 選定待測試FID
2. 利用待測試之FID建立Test Matrix (.xls檔)
3. 將待測的FID燒入EPROM
4. 將PROM放入module並將之放入FID test bed
5. 將module上mode switch轉至normal 後，完成硬體設定/連接
6. 進入測試畫面
7. 選定待測試之test matrix檔案
8. 設定” Get User Info.”相關欄位資訊
9. 按”DONE”開始測試
10. 測試完成出現測試結果(PASS/FAIL)，並自動產生test result (.xls檔)
11. 完成測試紀錄

- 選定待測試FID (1H12PL1109B-01 rev.D)



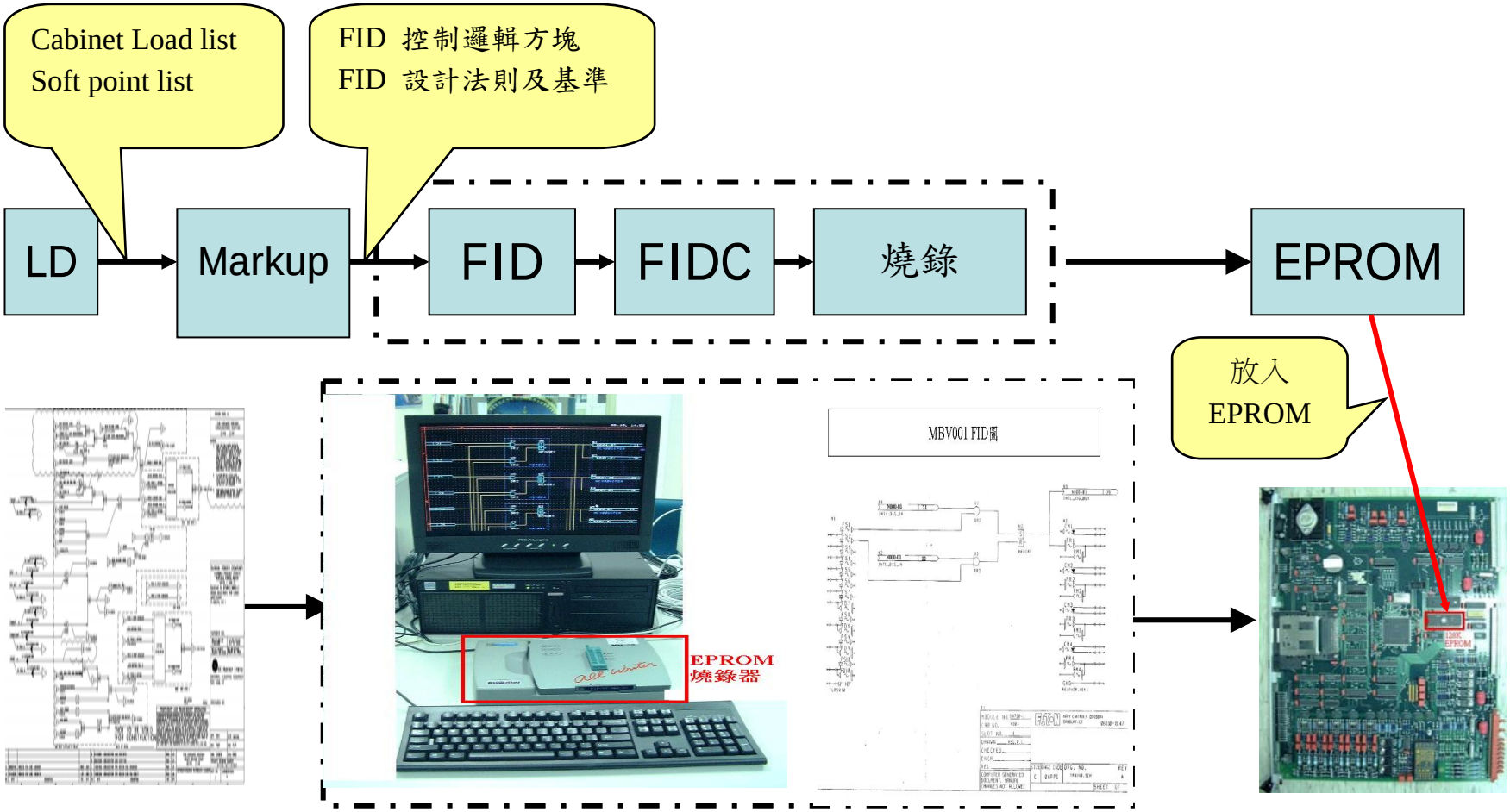
- 利用待測試之FID建立Test Matrix (1H12PL1109B-01 Test Matrix D7.xls檔)

Test Matrix: 1H12PL1109B-01 Test Matrix D7-1.xls
FID Test Procedure: KAY1317_16 Rev G
Test Engineer/Developer: J. Shaw, R. SHEA
Signature/Date - Test Engineer/Developer:
Signature/Date - Test Engineer/Reviewer:

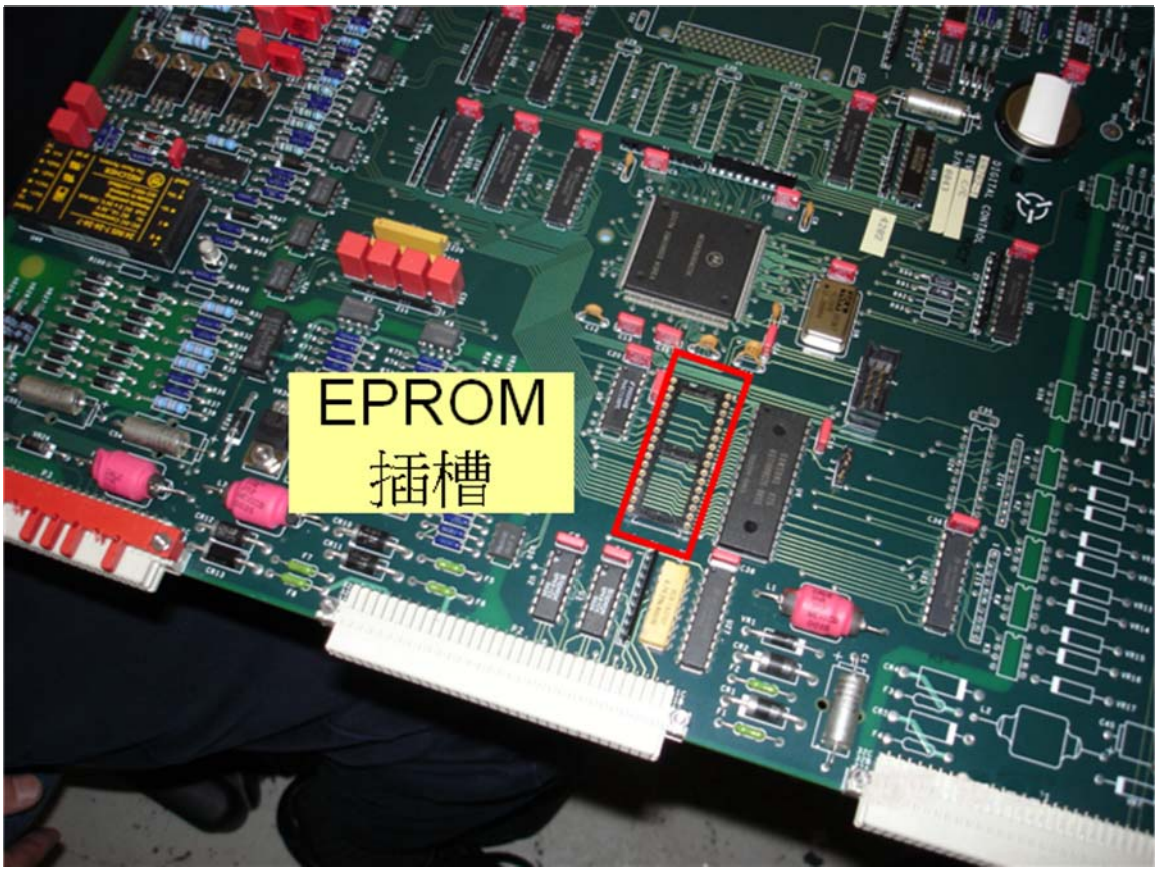
	INPUTS											OUTPUTS																
SIGNAL NAME	REACTOR PRESSURE	EUC	FG OUT	REACTOR WATER LEVEL WIDE RANGE	REACTOR WATER LEVEL WIDE RANGE	SCFT 07	SCFT 08	SCFT 09	SCFT 51	MOD FAULT IN		RX PRESS HI GP 1	RX PRESS HI GP 6	RX PRESS HI GP 3	RX PRESS HI GP 2	RX PRESS>3.9 MPAG	RX PRESS>3 MPAG	RX PRESS HI GP 5	RX PRESS HI GP 4	REACTOR SATURATION TEMP	RX WATER LVL<L2	RX WATER LVL<L2	RX WATER LVL<L1	RX WATER LVL<L1	RX WATER LVL<L3	RX WATER LVL<L1.5	RX WATER LVL<L1.5	BLKS RMVD	MOD FAULT OUT
SIGNAL TYPE	INTLCK_A			INTLCK_A	INTLCK_A	INTLCK_D	INTLCK_D	INTLCK_D	INTLCK_D	INTLCK_D		INTLCK_D	INTLCK_D	INTLCK_D	INTLCK_D	INTLCK_D	INTLCK_D	INTLCK_D	INTLCK_D	INTLCK_A	INTLCK_D	INTLCK_D	INTLCK_D	INTLCK_D	INTLCK_D	INTLCK_D	INTLCK_D	INTLCK_D	INTLCK_D
TEST POINT	N11 S45 O17			N11 S45 O29	N11 S45 O33	N78 S31 O18	N78 S31 O19	N78 S31 O20	N78 S31 O64	N78 S44 O12		N79 S01 O23	N79 S01 O22	N79 S01 O21	N79 S48 O37	N79 S48 O38	N79 S01 O26	N79 S01 O25	N79 S01 O24	N79 S01 O05	N79 S48 O28	N79 S01 O30	N79 S01 O27	N79 S01 O10	N79 S01 O39	N79 S01 O29	N79 S01 O9	N79 S01 O31	N79 S48 O36
1	800	0.00	100.00	7800	800	L	L	L	L	H		L	L	L	L	L	L	L	L	800	L	L	L	L	L	H	H	H	H
2	F00	625.00	152.63	7100	F00	L	L	L	H	L		L	L	L	L	L	H	L	L	23D7	L	L	L	L	L	H	H	H	L
3	1600	1250.00	191.88	6A00	1600	L	L	H	L	H		L	L	L	L	L	L	L	L	389B	L	L	L	L	L	H	L	L	H
4	1D00	1875.00	211.06	6300	1D00	L	H	L	L	H		L	L	L	L	L	L	L	L	42C1	H	L	L	L	L	H	H	L	H
5	2400	2500.00	225.30	5C00	2400	L	L	L	L	H		L	L	L	L	L	L	L	L	4A4A	L	L	L	L	L	H	H	H	H
6	2B00	3125.00	237.71	5500	2B00	H	L	L	L	H		L	L	L	L	L	H	L	L	50DB	L	L	H	L	L	L	L	L	H
7	3200	3750.00	247.78	4E00	3200	L	L	H	L	H		L	L	L	L	L	H	L	L	562E	L	L	L	L	L	H	L	L	H
8	3900	4375.00	256.83	4700	3900	L	L	L	L	H		L	L	L	L	H	H	L	L	5AF7	L	L	L	L	H	L	L	H	H
9	4000	5000.00	265.20	4000	4000	L	L	L	L	H		L	L	L	L	H	H	L	L	5F86	H	H	L	L	H	L	L	H	H
10	4700	5625.00	272.39	3900	4700	L	H	L	L	H		L	L	L	L	H	H	L	L	6333	H	L	L	L	H	L	L	L	H
11	4E00	6250.00	279.23	3200	4E00	L	L	L	L	H		L	L	L	L	H	H	L	L	66D1	H	H	L	L	H	L	L	H	H
12	5500	6875.00	285.54	2B00	5500	L	L	L	L	H		L	L	L	L	H	H	L	L	6A28	H	H	L	L	H	L	L	H	H
13	5C00	7500.00	291.30	2400	5C00	L	L	L	L	H		H	L	L	L	H	H	L	L	6D35	H	H	L	L	H	L	L	H	H
14	6300	8125.00	296.84	1D00	6300	L	L	L	L	H		H	H	H	H	H	H	H	H	7023	H	H	L	L	H	L	L	H	H
15	6A00	8750.00	302.03	1600	6A00	H	L	L	L	H		H	H	H	H	H	H	H	H	72E1	H	H	H	L	H	L	L	L	H
16	7100	9375.00	306.95	F00	7100	L	L	L	L	H		H	H	H	H	H	H	H	H	757C	H	H	H	H	H	L	L	H	H
17	7800	10000.00	311.70	800	7800	L	L	L	L	H		H	H	H	H	H	H	H	H	7800	H	H	H	H	H	L	L	H	H
												5B65	5F51	5CF7	5C2E	33AE	258C	5E88	5DBF		409A		1AB2		4B8B	28E9			
	0	1000	2000	3000	4000	5000	6000	7000	8000	9000	10000																		
	100	184.2	214.9	235.7	251.8	265.2	276.7	286.8	295.8	304.1	311.7																		

Note: Prior to running this test, the "REACTOR PRESSURE" must be set to a value between 4-20 mA.
From ScramNET Monitor, go to address BB10 hex, and enter value 807D0078 hex.
Reboot card to turn MOD FAULT LED OFF. Then run test as normal.

- 將待測的FID燒入EPROM(FID製作流程說明)



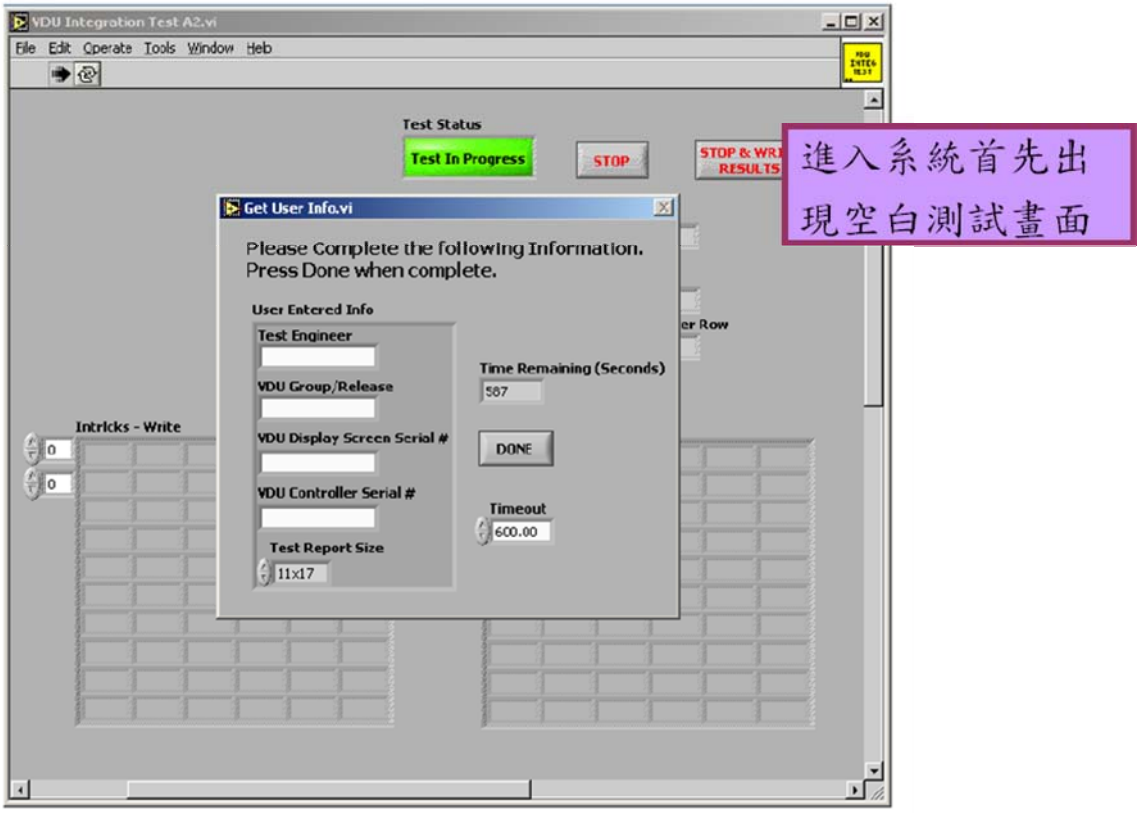
- 將EPROM放入module



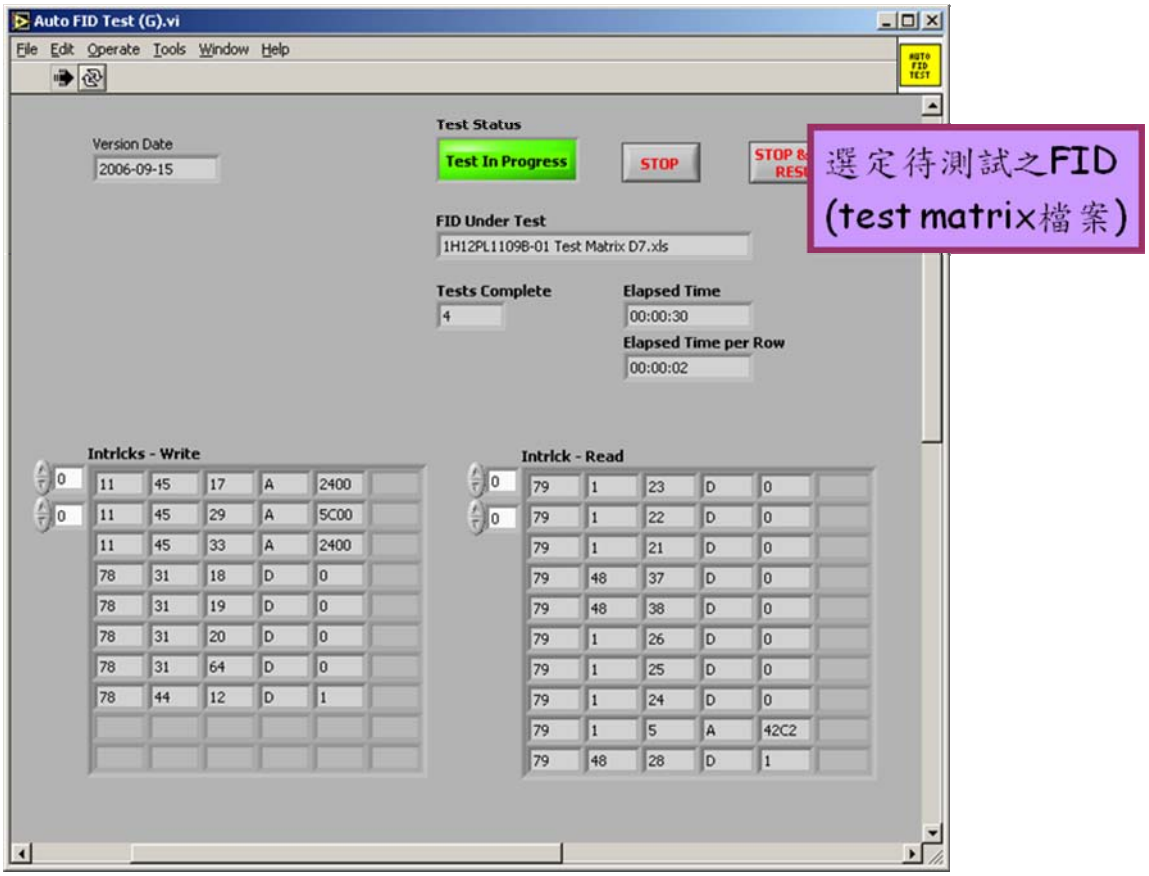
- 將module放入Test Bed

- 將module上mode switch轉至normal 後，完成硬體設定/連接

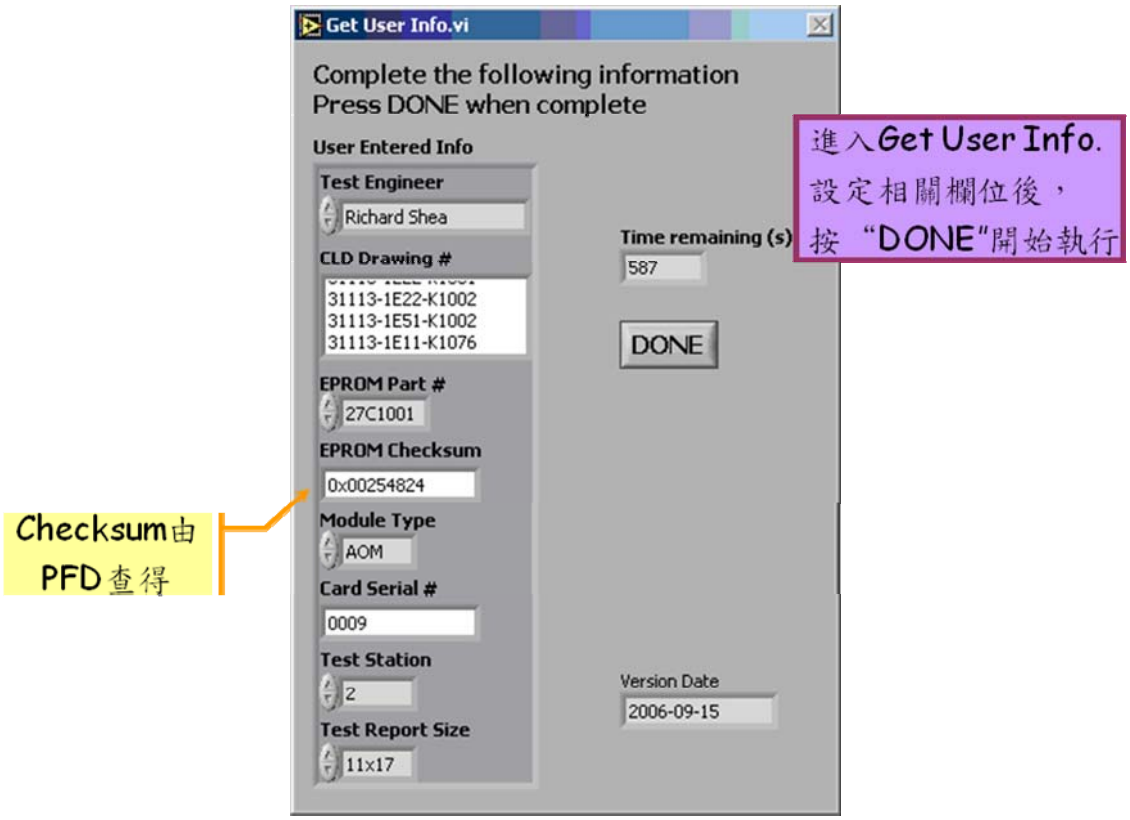
- 進入測試畫面



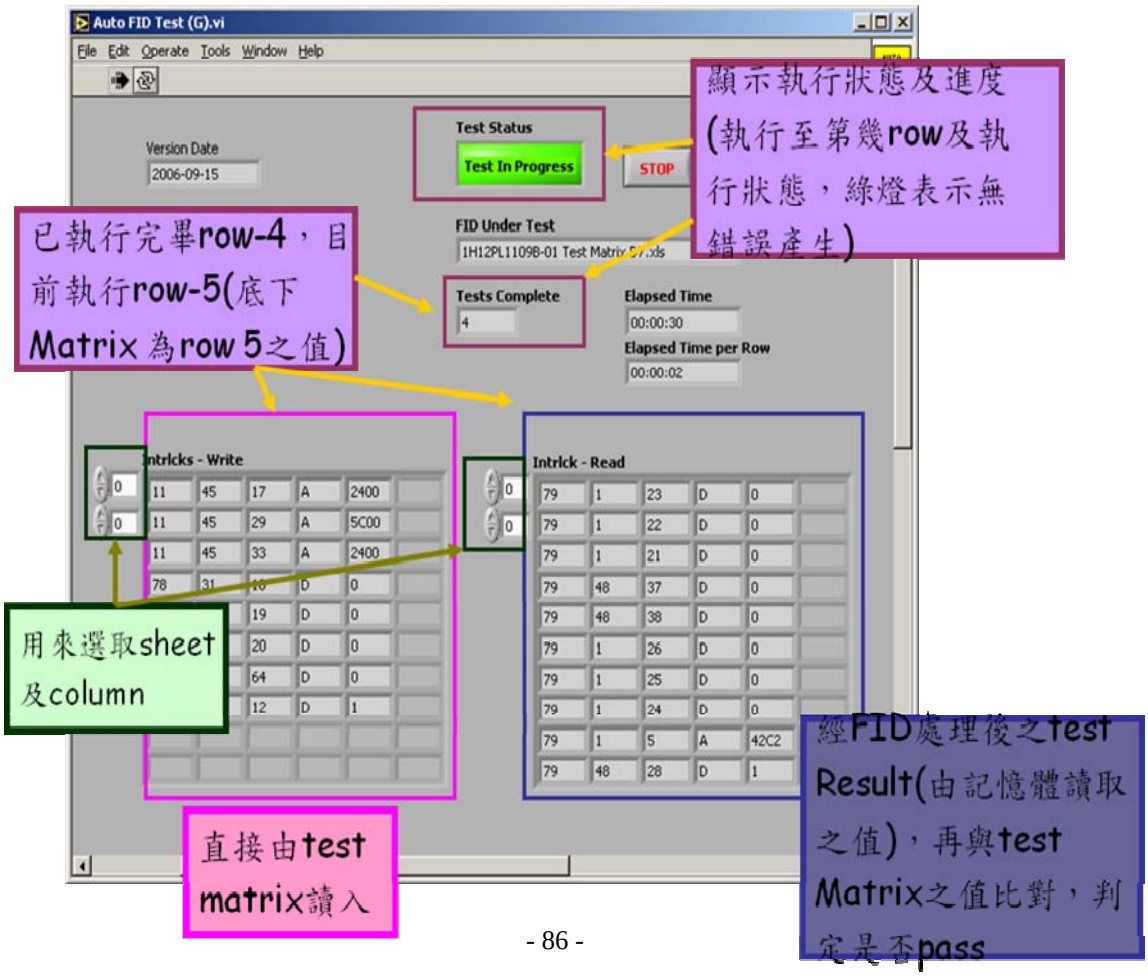
- 選定待測試之test matrix檔案



- 設定” Get User Info.”相關欄位資訊



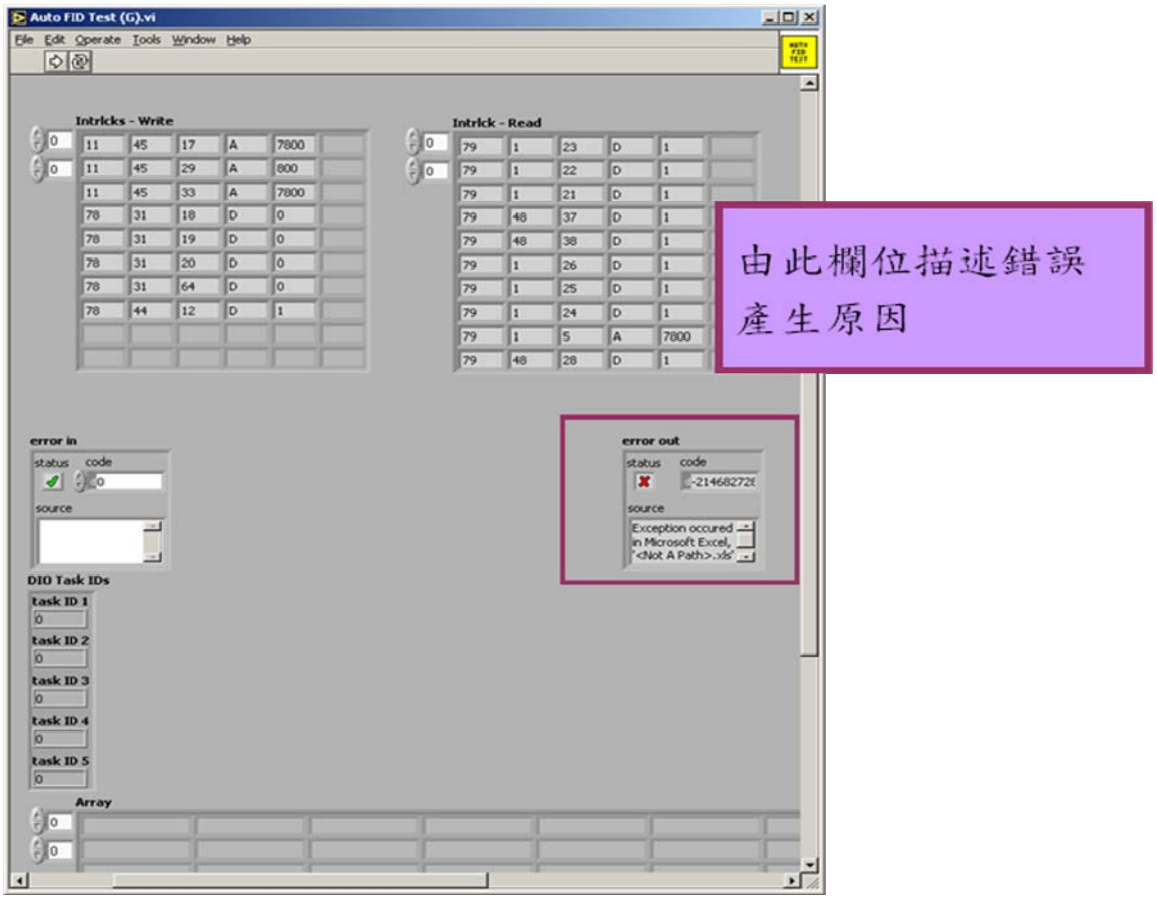
- 按”DONE”開始測試



- 測試完成並完成測試結果
  - 若PASS則自動產生test result，完成測試

  - 若FAIL則依據錯誤指示，重新檢查FID繪製及Text Matrix建立是否正確(1)

■ 依據錯誤指示，重新檢查FID繪製及Text Matrix建立是否正確(2)



■ 測試結束，自動產生test result (.xls檔) 完成測試紀錄(1)

Test Result Filename: 1H12PL1109B-01 D7 Test Results 200704170911-1.xls

FID Test Procedure	KAY1317_16 Rev G
Test Engineer	Richard Shea
Date of Test	2007/4/17
Time of Test	09:11
FID Dwg.(Rev.)	1H12PL1109B-01(D)
CLD Dwg.	31113-1B21-K1002 31113-1B21-K1002A 31113-1E22-K1001 31113-1E22-K1002 31113-1E51-K1002 31113-1E11-K1076
EPROM Part Number	27C1001
EPROM Checksum	0x00254824
Module Type	Analog Output Module
Module Part Number	6N768-1
Card Serial Number	9
Test Station	Test Station 2
Test Duration	00:00:52
Test Results	PASS

Signature/Date - Test Engineer _____

Signature/Date - Test Engineer/Reviewer _____

Signature/Date - NQA Reviewed/Witnessed _____

測試完畢自動產生 Test Result檔案

■ 測試結束，自動產生test result (.xls檔) 完成測試紀錄(2)

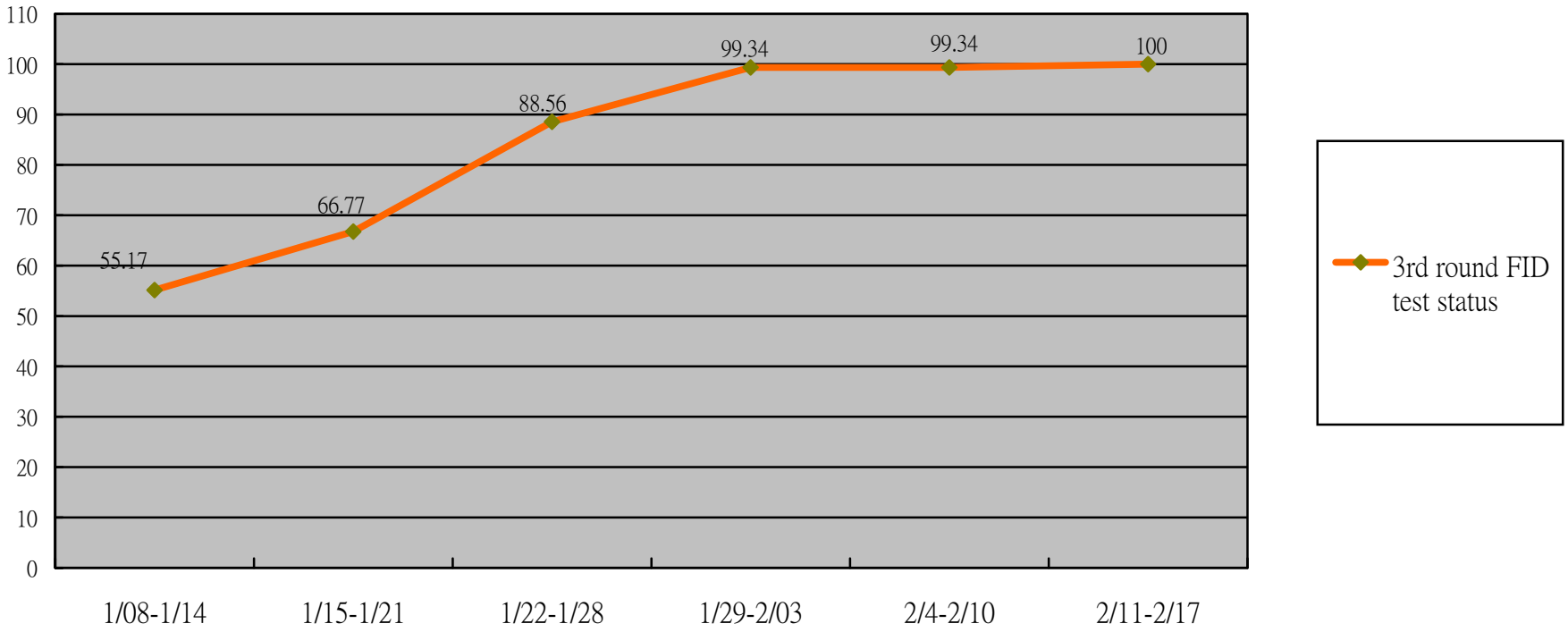
測試完畢自動產生
Test Result檔案

Test Result Filename: 1H12PL1109B-01 D7 Test Results 200704170911-2.xls

	INPUTS											OUTPUTS - ACTUAL																		RESULTS	
SIGNAL NAME	REACTOR PRESSURE	EUC	FG OUT	REACTOR WATER LEVEL WIDE RANGE	REACTOR WATER LEVEL WIDE RANGE	SCFT 07	SCFT 08	SCFT 09	SCFT 51	MOD FAULT IN		RX PRESS HI GP 1	RX PRESS HI GP 6	RX PRESS HI GP 3	RX PRESS HI GP 2	RX PRESS> 3.9 MPAG	RX PRESS> 3 MPAG	RX PRESS HI GP 5	RX PRESS HI GP 4	REACTOR SATURATION TEMP	RX WATER LVL<L2	RX WATER LVL<L2	RX WATER LVL<L1	RX WATER LVL<L1	RX WATER LVL<L3	RX WATER LVL<L1.5	RX WATER LVL<L1.5	BLKS RMVD	MOD FAULT OUT		
SIGNAL TYPE	INTLCK_A			INTLCK_A	INTLCK_A	INTLCK_D	INTLCK_D	INTLCK_D	INTLCK_D	INTLCK_D	DELAY	INTLCK_D	INTLCK_D	INTLCK_D	INTLCK_D	INTLCK_D	INTLCK_D	INTLCK_D	INTLCK_D	INTLCK_A	INTLCK_D	INTLCK_D	INTLCK_D	INTLCK_D	INTLCK_D	INTLCK_D	INTLCK_D	INTLCK_D	INTLCK_D		
TEST POINT	N11 S45 O17			N11 S45 O29	N11 S45 O33	N78 S31 O18	N78 S31 O19	N78 S31 O20	N78 S31 O64	N78 S44 O12		N79 S01 O23	N79 S01 O22	N79 S01 O21	N79 S48 O37	N79 S48 O38	N79 S01 O26	N79 S01 O25	N79 S01 O24	N79 S01 O05	N79 S48 O28	N79 S01 O30	N79 S01 O27	N79 S01 O10	N79 S01 O39	N79 S01 O29	N79 S01 O9	N79 S01 O31	N79 S48 O36		
1	800	0	100	7800	800	L	L	L	L	H		L	L	L	L	L	L	L	L	800	L	L	L	L	L	H	H	H	H	PASS	
2	F00	625	152.62	7100	F00	L	L	L	H	L		L	L	L	L	L	H	L	L	23D7	L	L	L	L	L	H	H	H	L	PASS	
3	1600	1250	191.87	6A00	1600	L	L	H	L	H		L	L	L	L	L	L	L	L	389B	L	L	L	L	L	H	L	L	H	PASS	
4	1D00	1875	211.06	6300	1D00	L	H	L	L	H		L	L	L	L	L	L	L	L	42C2	H	L	L	L	L	H	H	L	H	PASS	
5	2400	2500	225.3	5C00	2400	L	L	L	L	H		L	L	L	L	L	L	L	L	4A4A	L	L	L	L	L	H	H	H	H	PASS	
6	2B00	3125	237.71	5500	2B00	H	L	L	L	H		L	L	L	L	L	H	L	L	50DB	L	L	H	L	L	L	L	L	H	PASS	
7	3200	3750	247.77	4E00	3200	L	L	H	L	H		L	L	L	L	L	H	L	L	562E	L	L	L	L	L	H	L	L	H	PASS	
8	3900	4375	256.82	4700	3900	L	L	L	L	H		L	L	L	L	H	H	L	L	5AF8	L	L	L	L	H	L	L	H	H	PASS	
9	4000	5000	265.2	4000	4000	L	L	L	L	H		L	L	L	L	H	H	L	L	5F66	H	H	L	L	H	L	L	H	H	PASS	
10	4700	5625	272.38	3900	4700	L	H	L	L	H		L	L	L	L	H	H	L	L	6334	H	L	L	L	H	L	L	L	H	PASS	
11	4E00	6250	279.22	3200	4E00	L	L	L	L	H		L	L	L	L	H	H	L	L	66D2	H	H	L	L	H	L	L	H	H	PASS	
12	5500	6875	285.53	2B00	5500	L	L	L	L	H		L	L	L	L	H	H	L	L	6A29	H	H	L	L	H	L	L	H	H	PASS	
13	5C00	7500	291.3	2400	5C00	L	L	L	L	H		H	L	L	L	H	H	L	L	6D35	H	H	L	L	H	L	L	H	H	PASS	
14	6300	8125	296.83	1D00	6300	L	L	L	L	H		H	H	H	H	H	H	H	H	7023	H	H	L	L	H	L	L	H	H	PASS	
15	6A00	8750	302.02	1600	6A00	H	L	L	L	H		H	H	H	H	H	H	H	H	72E2	H	H	H	L	H	L	L	L	H	PASS	
16	7100	9375	306.95	F00	7100	L	L	L	L	H		H	H	H	H	H	H	H	H	757D	H	H	H	H	H	L	L	H	H	PASS	
17	7800	10000	311.7	800	7800	L	L	L	L	H		H	H	H	H	H	H	H	H	7800	H	H	H	H	H	L	L	H	H	PASS	

十一.3 FID測試進度說明

■ 3rd round FID 測試進度說明



■ 4th round FID 測試進度說明