UNIVERSITATEA „POLITEHNICA” TIMIŞOARA
FACULTATEA DE ELECTRONICĂ ŞI TELECOMUNICAŢII
DEPARTAMENTUL DE ELECTRONICĂ APLICATĂ

Ing. DAN SIMU

Contribuţii la configurarea unor structuri de testare automată cu aplicaţii în avionică

TEZĂ DE DOCTORAT

CONDUCĂTOR ŞTIINŢIFIC: PROF. DR. POPESCU VIOREL

TIMIŞOARA 2006

Cuprins

Introducere

Cap. I Stadiul actual in testarea automată pentru avionică.

Cap. II Testarea funcțională pentru avionică. Portabilitatea programelor de testare.

Cap. III Echipamente de testare automată pentru avionică.

Introducere

Avionica reprezintă domeniul tehnologic al dispozitivelor electronice utilizate în industria de aviaţie. Aparatura de aviaţie face parte din categoria dispozitivelor cu fiabilitatea ridicată, necesară pentru siguranţa pasagerilor şi aeronavelor. Verificarea periodică pentru confirmarea siguranţei în utilizare a diverselor aparate sau subansamble din componenţa aeronavelor a devenit obligatorie prin reglementările din domeniu. Aceste consideraţii sunt motivul obiectiv care a accelerat dezvoltarea sectorului de testare pentru avionică. Fiecare producător de dispozitive folosite în aviaţie parcurge numeroase etape de testare şi validare pe toată durata serviciului respectivului dispozitiv.

Această teză are ca obiect domeniul testării funcţionale automate pentru avionică. Este un domeniu îngust şi deosebit de divers în acelaşi timp.

Este îngust pentru că se adresează testării dispozitivelor de aviaţie, un sector foarte specific şi cu o diversitate relativă de unităţi. Astfel există doi mari producători de aeronave în lume Boeing şi AirBus care sunt şi producătorii importanţi de avionică. Cele două companii impun standardele şi stilul de realizare a unităţilor de aviaţie coagulând în jurul lor un număr nu foarte mare de subcontractori. Unităţile folosite sunt foarte asemănătoare din punct de vedere funcţional existând subcontractori care produc dispozitive pentru mai mult companii producătoare de avioane.

Este divers deoarece complexitatea unităţilor impune un echipament de testare foarte complicat, cu un număr mare de instrumente, cu instrumente particulare pentru verificarea unor tehnologii specifice domeniului. Diversitatea este menţinută şi de durata mare de utilizare a unei aeronave fapt care impune coexistenţa unor dispozitive din diverse generaţii realizate cu tehnologii diferite.

Traficul aerian civil a crescut foarte mult în ultimele decenii şi implicit cel al aeronavelor şi al unităţilor de avionică care sunt testate. Acest lucru a determinat înfiinţarea unor mari centre de testare a unităţilor de aviaţie. În general majoritatea companiilor aeriene mari au propria structură care se ocupă de verificarea şi întreţinerea unităţilor. Doresc să menţionez aici centrul de testare al companiei British Airways Avionic Engineering de la Cardiff, probabil unul dintre cele mai mari din lume care verifică avionica de pe avioanele companiei britanice dar şi ale altor companii aeriene. Acest centru este deosebit de interesant deoarece poti vedea în funcţiune o mulţime de testoare din diverse perioade, destinate unei singure unităţi sau unui grup de unităţi sau testoare de uz general. Acestea din urmă realizate de marii competitori în domeniu reflectă opţiunile proprii cu privire la arhitectura hardware şi software a unui astfel de testor.

Această arhitectură, tendinţele de standardizare în acest domeniu precum şi opinia autorului referitoare la diverse tehnici şi soluţii în acest proces de coagulare a tehnologiilor din domeniu reprezintă linia directoare după care afost elaborată această teză.

Autorul a participat la elaborarea documentaţiei de testare pentru mai mult de 200 de unităţi de avionică fiind implicat în toate fazele procesului de testare funcţională. Autorul a participat la proiectarea testoarelor, la analiza unităţilor, la proiectarea interfeţelor şi programelor de testare fiind în acelaşi timp conectat la eforturile comunităţii ştiinţifice din domeniu pentru realizarea portabilităţii programelor de testare

odată cu modernizarea echipamentelor şi instrumentelor. Această activitate diversă şi complexă a generat în diverse faze ale procesului încecări şi proiecte pilot pentru verificarea unor soluţii referitoare la uneltele şi aplicaţiile soft folosite.

Teza a fost alcătuită pe scheletul a cinci capitole care punctează elementele importante din ansamblul testării funcţionale pentru avionică:

- Capitolul I "Stadiul actual în testarea automată pentru avionică" trece în revistă domeniul şi componentele sale importante cu exemple din activitatea autorului.
- Capitolul II „ Testarea funcţională pentru avionică. Portabilitatea programelor de testare" defineşte procesul de testare funcţională din faza de descriere a testelor până la execuţia propriu-zisă. Se analizează elementele care asigură portabilitatea testelor funcţionale de la un echipament de testare la altul. Capitolul II continuă prezentarea domeniului testării funcţionale pentru avionică începută de Capitolul I precizând interesul autorului pentru portabilitate.
- Capitolul III „ Echipamente de testare automată pentru avionică" analizează în detaliu tipurile de echipamente, arhitectura acestora, instrumentele frecvent utilizate, standardele din domeniu şi prezintă unele contribuţii ale autorului.
- Capitolul IV „ Structura programelor de testare funcţională. Exemple specifice pentru avionică" defineşte procesul de testare în avionică şi prezintă exemple de arhitectură a unui program de testare, exemple de teste specifice. In final se analizează sintetic gradul de standardizare a diverselor instrumente.
- Capitolul V „Mediul de testare, limbaje de programare, drivere" este o prezentare de detaliu a componentelor majore software din structura unui echpament de testare automată pentru avionică. Autorul sublinează interdependenţele şi contribuţia acestor componente la atingerea dezideratului de portabilitate. De asemenea prezintă opiniile şi contribuţiile personale la diverse elemente din structură.

Teza se încheie cu capitolul de concluzii care subliniază elementelor cruciale pentru portabilitatea programelor de testare.

Teza este completată cu o bogată listă de anexe care exemplifică procesul de testare automată cu programe din experienţa autorului.

Portabilitatea programelor de testare automată pentru avionică a devenit foarte importantă odată cu creşterea numărului de avioane şi cu apariţia unor generaţii noi de unităţi şi instrumente de măsură. Trecerea testelor funcţionale de la o generaţie de echipament la alta s-a dovedit extrem de costisitoare. La aceasta s-a adăugat şi necesitatea proiectării testelor cu o mai mare viteză. Toate aceste tendinţe au presat în direcţia standardizării procesului şi conservarea unei arhitecturi deschise a sistemului pentru a permite evoluţia acestuia.

Teza a avut câteva izvoare bibliografice importante. Cele mai importante au fost manualele şi documentaţiile unităţilor testate, editate de companiile producătoare, care cuprind mii de pagini de informaţii tehnice. In aceeaşi categorie se află documentaţiile instrumentelor folosite pe diverse staţii, cu o subliniere pentru simulatoarele VXI pentru testarea comunicaţiilor. Un sector bibliografic special l-au reprezentat standardele şi proiectele de standarde ale diverselor grupuri de standardizare. Aici menţionam în special grupul SCC20 de la IEEE din cadrul secţiunii de instrumentaţie si măsuri, grupurile care

s-au ocupat de standardele ARINC şi consorţiul IVI pentru clasele de instrumente. Multe din informaţiile cuprinse în aceste standarde sau documentaţii au fost cuprinse în comunicări şi lucrări prezentate la AUTOTESTCON care este manifestarea ştiinţifică anuală în domeniul testarii automate.

Teza a fost elaborată sub îndrumarea atentă, profesională şi colegială a Domnului prof. Dr. Ing. Viorel Popescu căruia autorul îi mulţumeşte pentru efortul şi sprijinul acordat.

Autorul le mulţumeşte colegilor de la Timteh Electronics şi celor de la Institul Politehnic „Traian Vuia" cu care a colaborat la realizarea a numeroase proiecte în domeniul testării.

Pe parcusul elaborării tezei autorul a beneficiat de colaborarea unor colegi ingineri de o deosebită calitate profesională şi intelectuală, dintre care mulţi s-au stabilit în ţări occidentale. Autorul doreşte să le mulţumească pentru sprijinul, pentru ideile şi competenţa lor, dar şi pentru calitatea umană de care au dat dovadă pe parcursul colaborării .

Autorul mulţumeşte familiei pentru solidaritate şi suport necondiţionat.

CAP. I

Stadiul actual in testarea automată pentru avionică.[15][88][90][91][107][125][132][151][164][165][167][174][183][187]

În acest capitol autorul prezintă în formă succintă câteva elemente legate de definirea, aplicarea şi interpretarea unor noţiuni specifice activităţii de testare a unor dispozitve electronice complexe. Aparatura electronică de la bordul unei aeronave reprezintă în cel mai înalt grad nivelul de proiectare, realizare şi funcţionare în domeniul electronicii aplicate. Ca urmare a complexităţii unui astfel de echipament, rezultă ca o necesitate asigurarea fiabilităţii maxime a acestuia. Un mijloc de a asigura fiabilitatea maximă a echipamentului de la bordul unei aeronave constă în elaborarea de tehnologii de testare avansate.

Avionica este sectorul din industria electronică care cuprinde dispozitivele electronice construite pentru industria de aviaţie si aeronautică. Dispozitvele din acest grup sunt dispozitive electronice complexe şi foarte diverse conform necesităţilor unei aeronave şi sunt considerate dispozitive cu grad critic de siguranţă.

Testarea funcţională a acestor dispozitive decide utilizarea lor in timpul serviciului aeronavei, evidenţiază defectele unităţii testate şi permite diagnosticarea.

Testarea se face prin intermediul unor standuri de testare. Aceste standuri se numesc echipamente de testare automată deoarece procesul de testare este controlat de un program rulat de un calculator.

Programele de testare se scriu in limbaje specializate care descriu testul şi permit comanda echipamentului de testare pentru execuţia testului. Aceste limbaje au evoluat in timp devenind tot mai complexe.

Execuţia testelor se face prin intermediul unor aplicaţii de tip „test executiv" care leagă programul de testare, de instrumentele de stimulare şi măsură, de unitatea testată.

Concluziile la acest capitol sublinează dezvoltarea accelerată din acest domeniu şi noile direcţii care se conturează pentru testarea automată a dispozitivelor electronice utilizate în construcţia avioanelor.Se remarcă importanţa standardizării în procesul de testare funcţională pentru portabilitatea acestor teste şi pentru dezvoltarea unitară a domeniului.

1.1 Avionică. Tipuri de dispozitive. [21][36]-[52][55]-[80][152][100]

Dispozitivele electronice folosite de o aeronavă sunt mijlocul prin care nava este controlată. Gama de dispozitive este diversă. Pe aeronavele moderne se pilotează conform principiului "fly by wire", asta semnificând că toate comenzile se transmit prin intermediul dispozitvelor electronice şi atunci când se pilotează manual. În figura 1.1 este ilustrat acest lucru:

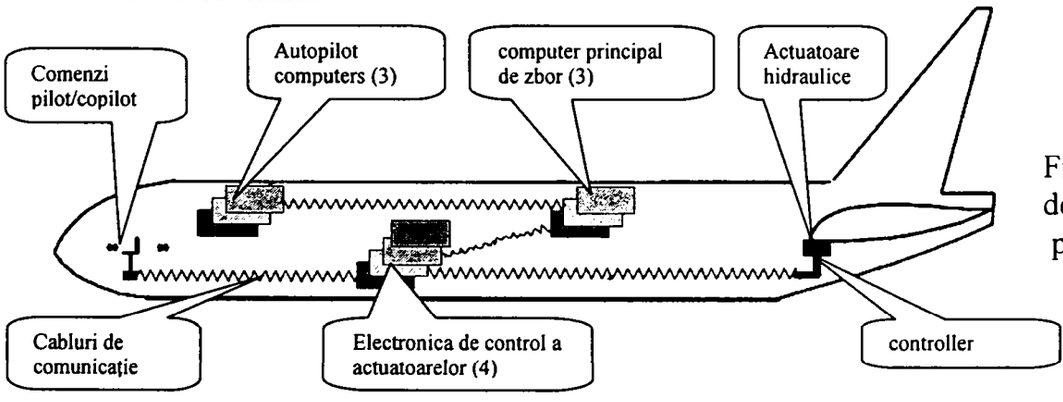


Fig 1.1 dispozitivele de navigaţie conectate prin fire cu pilotul

Pentru un avion din clasa modernă a avioanelor B777 figura 1.1 devine în detaliu figura 1.2

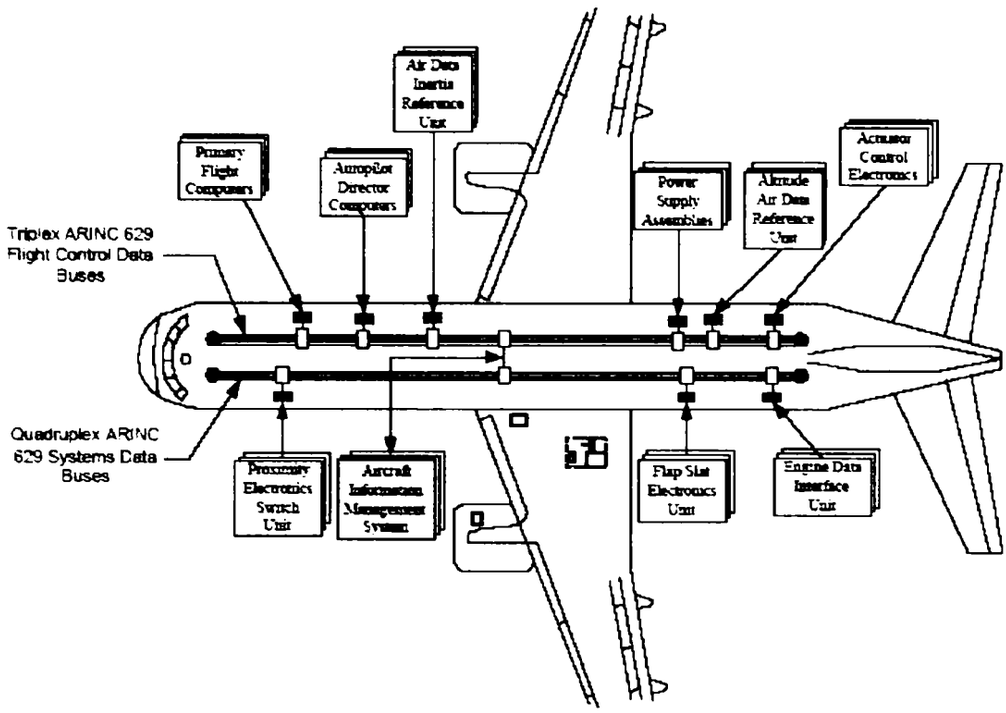


Fig. 1.2 Unităţile din sistemul de navigaţie şi din sistemul stocare şi întreţinere al unei aeronave moderne.

În figura 1.2 se observă cele două sisteme de navigaţie şi întreţinere legate prin magistralele de comunicaţie de tip ARINC 629[21] (care se prezintă în capitolele 3 şi 5).

In figura 1.3 se pot observa dispozitivele din cabina piloţilor (pentru un avion de generaţie nouă cu avionică de tip modular) care sunt în mare măsură, dispozitive de comadă şi afişare. Dispozitivul din lateral este MCDU (Main Control Display Unit) văzut din faţă şi spate . După cum se remarcă aceste dispozitive sunt construite în cutii metalice cu conectori pentru interconectare în sistem.

Fig.1.3 Vedere cabina piloţilor. Vedere MCDU

Ele sunt împarţite în dispozitve de putere, de comunicaţie, de reglare şi control, de calcul, de navigaţie . Există[81][82] mai multe tipuri de circuite care prezintă interes din punct de vedere al procesului de testare.

Energia este furnizată în avion de catre un generator care produce 115V AC si 400 Hz. Exista un dispozitiv electronic care controlează atât excitaţia cat şi indusul generatorului pentru a asigura o amplitudine şi o frecvenţă constantă. Această unitate numită "GCU" (Generator Control Unit) asigură funcţionarea corectă a sistemului de alimentare cu energie electrica. Toate dispozitivele electronice vor avea un circuit de alimentare care va transforma tensiunea alternativă de 115V AC si 400Hz într-o tensiune continua, în general de 28V. Dispozitivele electronice care controlează acţionarea unor elemente de control ale avionului cum ar fi miscarea flaps-urilor sau puterea de tractiune a motoarelor sunt dispozitive care au etaje de putere necesare pentru a acţiona un servomotor sau un cuplaj magnetic. O astfel de unitate este cea denumită "Autothrotle", care asigură controlul tracţiunii motoarelor, iar o altă unitate, "Antiskid" controleaza frânarea la aterizare. Există de semenea unităţi electronice care controlează poziţia avionului cum ar fi Pitch şi Roll Computers sau dispozitive care controlează navigaţia cum ar fi FMC (Flight Management Computer) si numeroase altele.

Dispozitivele pot fi clasificate corespunzător cu generaţia din care fac parte. Există dispozitive realizate cu circuite analoge sau altele mai recente realizate digital. Deoarece aeronavele au durate de viaţă de cel puţin 20 de ani astfel de dispozitive coexistă. O categorie separată de circuite sunt cele de comunicaţie. În avionică există standarde şi protocoale de comunicaţie[86] specifice acestei industrii. In ultima vreme au apărut standarde noi cum ar fi ARINC 629[21] sau AFDX. Unul dintre standardele cele mai folosite este de exemplu ARINC 429[17] care asigura comunicaţia intre dispozitive.

Acest standard permite dispozitivelor electronice pe baza unor adrese standard sa schimbe informaţii prin intermediul unui bus de comunicaţie. O situaţie particulară spre exemplu este transmiterea valorii unor marimi analoge ciclic pe bus permiţând astfel receptia in timp real de catre alte dispozitive cu o mare stabilitate la perturbaţii. Acest gen de circuite impune pentru testare utilizarea unor circuite specializate.

1.2 Testarea funcţională. [1]-[7][18]-[20][25]166][195][197][200][136][137]

Testarea unui dispozitiv electronic folosit pe un avion care se află in serviciu (numit şi LRU (Line Replaceble Unit)) se face periodic pentru a verifica dacă acesta poate fi folosit în siguranţă. Procesul de testare este de tip functional. Unitatea este conectată la un testor prin intermediul conectorului său. Testorul aplică stimuli şi măsoară răspunsurile unităţii verificând într-o anumită ordine funcţiile acesteia.

Testarea funcţională, prin definiţie, este testarea unui dispozitiv considerat cutie neagră prin intermediul conectorului său cu exteriorul, în raport cu o specificaţie dată. În mod normal se verifică funcţiile acestuia şi nu circuitele acestuia. Totuşi uneori se pot verifica şi anumite circuite şi diagnostica funcţionarea sau defectarea unor componente.

Procesul de verificare are loc într-o anumită ordine a funcţiilor unităţii. Mai întâi se testează anumite continuităţi fără a aplica tensiunea de alimentare a unităţii . Se testează apoi consumul de curent, existenţa unor tensiuni astfel încât tot circuitul de alimentare să fie verificate inclusiv circuitele de protecţie. Testarea propriu-zisă a

funcţiilor unităţii se face fie individual fie pe grupe de funcţii. Un test este compus dintr-o parte pregătitoare numită "set-up" în care se aduce unitatea în momentul funcţional necesar şi verificarea propriu-zisă. Partea pregătitoare poate fi uneori foarte laborioasă fiind necesară parcurgerea unor paşi anteriori. Există unităţi a căror testare durează şi 8-9 ore. Din acest motiv testarea unităţii se face folosind aplicaţii de testare care permit executarea unui singur test, a mai multor teste sau executarea tuturor testelor. De asemenea există variante de testare care permit diagnosticarea circuitelor sau componentelor.

Decizia legată de buna funcţionare a unităţii este luată pe baza verificării tuturor testelor prin rularea a ceea ce se numeşte "Full Functional Test" (Test Funcţional Complet). Dacă raportul testării nu are nici o eroare se consideră că unitatea este aptă de serviciu. În general acest test complet se rulează de mai multe ori (cel puţin 3) pentru ca rezultatul să fie considerat cert şi stabil.

Mediul de execuţie al unui program de testare, cuprinde opţiuni pentru operator, de testare parţială sau completă in variantă de parcurgere fără oprire sau cu oprire şi diagnostic la erori.

Structura unui program de testare permite gruparea testelor astfel ca ele să poată fi executate separat. Aceste grupuri se numesc teste parţiale. Ele sunt utile pentru verificarea şi mai ales diagnosticarea unor unităţi al căror test complet este foarte lung.

Programarea testelor funcţionale se bazează pe teste punctuale (test cases) care sunt grupate şi ordonate astfel încât testarea funcţională să acopere cât mai mult din verificarea completă a unităţii respective. Realizarea arborelui de teste necesar pentru verificarea unei unităţi se poate face în mai multe feluri. Astfel documentul descriptiv al testelor, elaborat de proiectantul unităţii poate fi manualul de întreţinere CMM (Component Maintenance Manual) [36]-[52][55]-[70] al unităţii sau un document mult mai formalizat care se numeşte TRD (Test Requirement Document). Uneori proiectantul testelor funcţionale nu are la dispoziţie decât documentaţia dispozitivului (scheme şi descrierea funcţionării). În acest caz el are sarcina să dezvolte şi documentul anterior programării testelor, respectiv TRD-ul. Definirea testelor şi a condiţiilor de testare reprezintă un proces iterativ în timp care are scopul de acoperi cat mai multe din erorile posibile într-un timp cât mai scurt.

Există mai multe criterii de abordare a generării seriei de teste:

- în funcţie de generaţia din care face parte unitatea;
- de măsura în care o parte din teste au fost integrate în autotestele pe care unitatea le face automat la pornire, aşa numitul BIST (Built In Self Test);
- de existenţa unor funcţii speciale de testare pe care unitatea le are implementate din proiectare;
- de existenţa (pentru unităţile inteligente) unor programe speciale de testare care se încarcă în momentul testării.

1.3 Sisteme de testare automată[3][15][18][19][22][23][24][88][91][136][158][159][175][196][200][13]

Intregul sistem de testare automată (ATS – Automatic Test System) cuprinde Echipamentul de Testare Automată (ATE – Automatic Test Equipment), interfaţa cu unitatea testată (TUA – Test Unit Adapter) şi dispozitivul electronic testat (UUT – Unit Under Test). Sistemele de testare automată au apărut ca o necesitate la numărul mare de

dispozitive electronice care alcătuiesc sistemul electronic al unei aeronave. Cu ani în urmă fiecare dispozitiv avea un testor specific. Acest lucru se întâmplă şi în prezent dar numărul mare al acestor testoare care ocupă spaţiu şi personal specializat a generat necesitatea înlocuirii acestora cu testoare de uz general. Apariţia testoarelor de uz general a generat la rândul ei necesitatea standardizarii procesului de testare pe diversele nivele care îl alcătuiesc.

Un echipament de testare automată este alcătuit din mai multe entităţi caracteristice[110]-[120]. Se prezintă in figura 1.4 configuraţia testorului SMART CATS (produs de compania Rada Electronic Industries Ltd.) pe care s-au dezvoltat şi integrat *(cu participarea şi sub coordonarea autorului)* programe de testare pentru peste 200 de unită i de _ e mai multe ti _ uri de avioane.

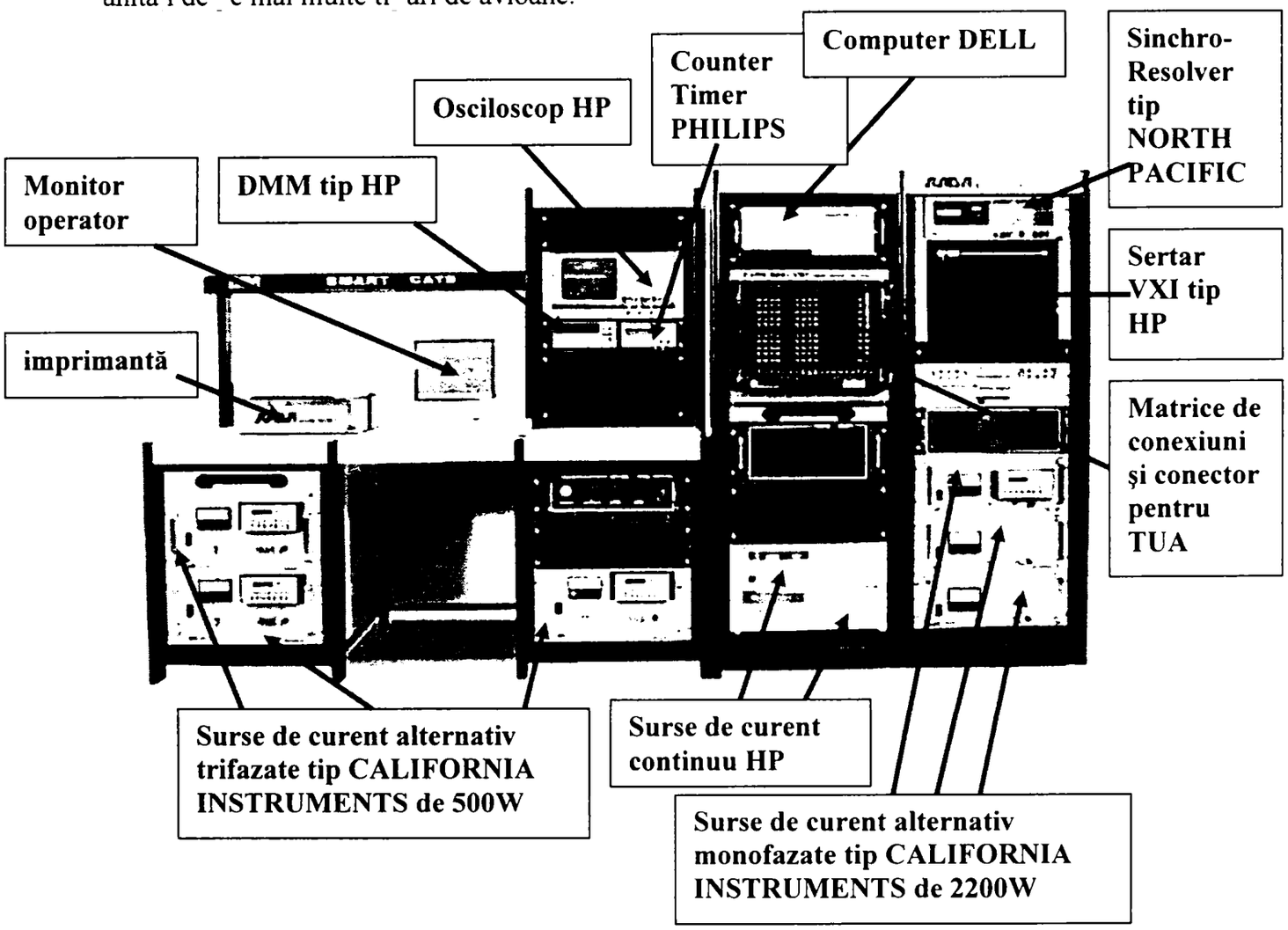


Fig. 1.4 Staţie de testare tip SMART CATS

Testorul este în principiu alcătuit dintr-un computer care comandă printr-o magistrală de tip GPIB (General Purpose Instrument Bus)[23][24] instrumentele de măsură sau generatoarele de stimuli. Acestea sunt conectate la o matrice de conexiuni. Un element deosebit este sertarul VXI[131][153] care conţine diverse instrumente si generatoare

de stimuli în format de plăci VXI sau VME. Sertarul este comandat prin intermediul unei plăci controller care face legatura intre magistrala GPIB şi magistrala sertarului VXI.

1.4 Limbaje pentru programele de testare [1][2][4][7][122][123][124][25][160][71][87]

Ansamblul de programe care definesc testul funcţional complet al unei unitaţi testate se numeşte TPS (Test Program Set) . Aceste programe au fost realizate utilizând diverse editoare şi diverse limbaje de programare. TPS-ul rezolvă descrierea testelor în secvenţa lor logică, rezolvă legătura cu echipamentul de testare automată şi rezolvă legătura cu operatorul.

Limbajele de programare utilizate au evoluat în timp spre limbaje de nivel înalt sau chiar spre variante grafice. Primele programe de testare au fost scrise în limbaje tradiţionale cum ar fi Basic standard sau C. Firmele au dezvoltat librării proprii pentru aplicaţiile proprii de testare.

Un astfel de exemplu sunt programele de testare scrise pentru Concorde. Echipamentul de testare pentru Concorde denumit şi UTE (Universal Test Equipment produs de British Aerospace) construit în perioada anului 1975, conform structurii standard pentru echipamentele de testare automată utiliza un computer HP 2116 pe care rula un interpretor BASIC. Proiectanţii sistemului au construit o suită de subrutine care programau echipamentul de testare, astfel încât inginerii de testare sa aibă posibilitatea dezvoltării programelor de testare. Spre exemplu programarea unei surse AC de tip Elgar era realizată astfel:

```
CALL (27,0)          ;configurează sursa în mod trifazat
CALL (28,400,115)    ;programează cele trei faze la 115V şi 400Hz
```

Aşa cum se remarcă programul este relativ dificil de citit din punct de vedere al testului iar legatura cu sursa este realizată intr-un mod particular. Remarc aceste idei deorece după aproape 30 de ani, compania BAAE (British Airways Avionic Engineering) componentă a grupului BA (British Airways) care se ocupă de testarea şi întreţinerea echipamentelor electronice a lansat un proiect de înlocuire a testoarelor. Tranziţia de la sistemul vechi la sistemul nou a implicat atât înlocuirea echipamentului cât şi a programelor şi mai ales revalidarea tuturor testelor (pentru toate unităţile). *Se va prezenta un translator de programe în cap. 4 la a cărui proiectare şi realizarea a participat autorul.*

Limbajele de programare a testelor au evoluat în mai multe planuri. Astfel s-au dezvoltat limbaje care sunt specializate pentru testare şi care descriu acţiunea de măsurare sau stimulare care se execută. Un astfel de limbaj a dezvoltat compania Rada Electronic Ind. Cele două linii de program de mai sus (care programează sursa AC) arată în acest caz astfel:

```
INIT-PS-N    PS1    ADR   ALONE      ;iniţializează sursa PS1 la adresa
                                     ;ADR in mod independent
PS-AC-V      PS1    115VAC  400Hz    ;setează tensiunea şi frecvenţa
```

Un pas important s-a făcut când companiile aeriene prin consorţiul ARINC care se ocupă cu standardizarea în domeniul aviaţiei a introdus limbajul ATLAS (Abreviated Test Leanguage for All Systems). Acesta a ordonat structura unui program de testare şi a stabilit sintaxa limbajului. Astfel mai întâi se declară toate aparatele de măsură şi stimulare, apoi se declară conexiunile iar în final se programează testele. Cele două linii de mai sus (programarea sursei AC) în limbajul ATLAS conform standardului ARINC 626 vor fi in secţiuni diferite şi vor arăta astfel:

```
001100 REQUIRE, 'AC-SUPPLY', SOURCE, AC SIGNAL,
       CONTROL,
          VOLTAGE RANGE 0V TO 200V CONTINUOUS,
          FREQ RANGE 100 HZ TO 600 HZ CONTINUOUS,
       LIMIT,
          CURRENT MAX 1.5 A,
       CNX HI  LO $

032000 DEFINE, 'POWER', SIGNAL, AC SIGNAL USING 'AC-SUPPLY',
       VOLTAGE () RANGE 0 V TO 180 V,
       FREQ 400 HZ,
       CURRENT MAX 1.3 A,
       CNX HI  'J1C-04 115VAC'
            LO  'J1C-01 115VAC RET'  $

210000 APPLY, 'POWER', 115  $
```

1.5 Execuţia testelor.[71][8][2]-[6][160][121][122][12]

Există două moduri de a executa efectiv un program de testare.

Un mod constă în compilarea programului împreună cu alte librării şi generarea unui executabil. În momentul în care aplicaţia generată este rulată ea execută acţiunile programate. Există desigur diverse combinaţii arhitecturale ale acestui gen de aplicaţie cum ar fi compilarea testelor sub formă de librării dinamice şi integrarea lor într-o aplicaţie de uz general care implementează funcţionalitatea sistemului sau pur şi simplu integrarea codului într-un cod template şi compilarea codului rezultat.

Un alt mod constă în utilizarea unui interpretor de limbaj care citeşte programul şi îl execută pas cu pas.

Toate sistemele trebuie să asigure câteva funcţii de bază. Acestea sunt :

- Să asigure legătura cu instrumentele;
- Să asigure legatura cu operatorul;
- Să implemeteze structura necesară pentru testarea funcţională a dispozitivelor electronice de tip avionică;
- Să asigure generarea şi stocarea rapoartelor;

În continuare se prezintă câteva exemple de aplicaţii de testare ale unor companii care au dezvoltat sisteme de testare automată pentru care autorul a dezvoltat programe de

testare. Exemplele sunt prezentate în ordine cronologică din punct de vedere al evoluţiei mediului de testare.

Sistemul de testare dezvoltat de catre BAE (British Aerospace) pentru BAAE (British Airways Avionic Engineering) se numea UTE (Universal Test Equipment)[71] şi folosea o aplicaţie Basic de tip interpretor carea avea incluse rutinele de comanda a instrumentelor. Aplicaţia avea implementată o interfaţă de tip consolă cu cateva funcţii disponibile pentru operator. De asemenea putea genera un raport. Sistemul funcţiona sub un sistem de operare care nu mai este actual, totuşi se menţionează pentru că a fost sistemul de testare pentru avionul Concorde, singurul supersonic pe o linie comercială din lume.

Sistemul de testare[8] dezvoltat de Rada Electronic Industries Ltd. funcţionează cu un testor modern, pe o platformă Windows. Aplicaţia software a fos dezvoltată sub DOS ca sistem de operare dar portată apoi sub Windows. Test Executivul dezvoltat de RADA este un interpretor al unui limbaj propriu orientat spre masurători. Aplicaţia realizează legătura cu instrumentele prin drivere specifice incluse în executabil. Astfel orice instrument nou adăugat sistemului implica recompilarea aplicaţiei şi generarea unui nou executabil.

Sistemul de testare dezvoltat de Aerospatiale împrună cu Arinc inc.[2]-[6] reprezintă un moment de tranziţie spre aplicaţiile deschise în care se pot adăuga module noi. Totuşi acest sistem este construit mai aproape de o aplicaţie de tip informatic bazată pe limbajul C, fiecare test fiind de fapt o recompilare cu generare de noi aplicaţii.

Sistemul de testare dezvoltat Tyx corp. sub denumirea de PAWS[160] utilizează ca limbaj de programare limbajul ATLAS. Acesta are implementată la nivel de sistem o librărie de nivel scăzut de operare cu porturile sistemului. Cu funcţiile acestei librării se dezvoltă de către utilizator o suită de drivere de nivel înalt care sunt apoi folosite pentru executarea programului. Sistemul reprezintă un progres important deoarece leagă în mod practic entităţi cum ar fi semnal, capabilitate, conexiune, care aparţin de un mod mai general de abordare a testării. Mediul PAWS este dezvoltat atât sub platforma Windows cât şi sub UNIX şi utilizează mai multe subset-uri de limbaj pentru diverse variante de ATLAS. PAWS este larg folosit în aplicaţii de testare militare.

Sistemele de testare bazate pe aplicaţia TestStand[121][122] dezvoltată de National Instruments reprezintă o abordare modernă de dezvoltare a unui mediu de testare. Test Stand permite existenţa unui limbaj de testare de nive înalt care poate fi creat de către utilizator. Un element nou îl reprezintă posibilitatea creerii unui „Process Model". Execuţia unui test este un proces complex care implică existenţa unei succesiuni de acţiuni simple. Acest proces se numeşte „Process Model", el poate fi declarat la nivel de sistem şi aplicat tuturor testelor în mod automat. Sistemul permite de asemenea existenţa unor acţiuni concurente precum şi adăugarea unor module noi, compilate în afara mediului. Practic TestStand permite crearea unor aplicaţii de testare care să includă toate noile abordări care sunt speccificate în standardul IEEE 1641[200].

Un capitol foarte important în executarea programelor de testare îl reprezintă legătura cu instrumetele. Aceasta se face prin driverele de instrument. În ultimii ani există tendinţa de a standardiza instrumentele prin definirea unor clase de instrumente. O astfel de clasă defineşte capabilităţile de bază pe care un instrument din categoria respectivă trebuie să le îndeplinească. Această standardizare permite în final utilizarea de către

programele de testare a unor instrumente virtuale nimite IVI (Interchangeable Virtual Instruments). Această abordare permite înlocuirea instrumentului fizic cu altul care aparţine clasei respective fără a modifica prgramul de testare.

1.6 Concluzii.

Domeniul testării automate se află într-un proces de transformare fundamentală la nivelul filosofiei de testare. Există câteva direcţii în care s-au creat mari consorţii internaţionale formate din marile companii, universităţi, instituţii de standardizare. Aceste consorţii au ca obiectiv crearea unor documente care să gideze activitatea de testare.

Din studierea standardelor şi consorţiilor industriale active şi valide în prezent autorul a constatat:

- Necesitatea descrierii specificaţiei de testare independent de platforma software şi hardware, a fost concretizată în standardul „IEEE 716[122] Test language for all Systems – Common/Abbreviated Test Language for All Syistems„ care defineşte un limbaj de nivel înalt pentru testare. Acest limbaj este destinat să descrie testele într-un mod independent de orice sistem specific de testare. Standardul IEEE 716 defineşte testul funcţional ca structură şi entităţile participante la proces.
- Necesitatea descrierii procesului de testare orientată spre semnale este acoperită de standardul "IEEE 1641-2004[200] Standard for Signal and Test Definition". De asemenea acest standard propune un set comun de semnale de bază, care pot fi conectate prin algoritmi matematici astfel încât să se poată sintetiza semnale complexe utilizabile pe orice platformă de testare.
- Necesitatea unui management coerent al procesului de proiectare a testelor, care este acoperită de standardul "ARINC – 625[19] Quality management process for test procedure generation " care specifică regulile de proiectare a unui program de testare funcţională pentru avionică.
- Existenţa unui deosebit interes pentru standardizarea domeniului instrumentaţiei virtuale. Acest lucru s-a concretizat în proiectele de standarde generate de consorţiul IVI[26]-[35] (consorţiu pentru standardizarea în domeniul instrumentaţiei virtuale) dintre care autorul le aminteşte pe cele mai frecvent utilizate:

IVI-3.1: Driver Architecture Specification
IVI-4.1: IviScope Class Specification
IVI-4.2: IviDmm Class Specification
IVI-4.3: Fgen Class Specification
IVI-4.4: IviDCPwr Class Specification
IVI-4.6: IviSwtch Class Specification
IVI-4.7: IviPwrMeter Class Specification
IVI-4.8: IviSpecAn Class Specification
IVI-4.10: IviRFSigGen Class Specification

Aceste standarde definesc câteva clase de instrumente pe care le respectă toţi producătorii de instrumente .Toate aceste documente de standardizare elaborate prin efortul comunităţii stiinţifice sublinează orientarea în domeniul construcţiei de testoare spre elemente standardizate utilizabile în diverse aplicaţii sub formă generică.

În domeniul limbajelor de programare autorul a constat o orientare spre limbaje de nivel înalt cu folosirea semnalelor ca entităţi descriptive care permit conservarea programelor de testare indiferent de platformă.

Autorul a constat un deosebit interes pentru portabilitatea programelor de testare funcţională şi pentru structurile de testare deschise care permit integrarea şi extinderea cu noi domenii tehnologice.

CAP. II
Testarea funcţională pentru avionică. Portabilitatea programelor de testare [22][18][7][55]-[70][93][98][111][125]

În acest capitol se prezintă procesul de testare funcţională în avionică: definiţie, descrirea testelor, programarea testelor,alocarea resurselor.

Testarea funcţională a unei unităţi electronice de aviaţie este un proces de testare succesivă a funcţiilor unităţii respective în raport cu o specificaţie a acestora.

Programul de testare are ca sursă de informaţie diverse documente generate în procesul de proiectare, fabricaţie şi utilizare. Autorul abordează structurarea proceselor de testare astfel încât să se poată obţine un set de programe şi fişiere portabile în anumite condiţii.

Testarea funcţională cuprinde şase entităţi semnificative: TPS (Test Program Set), ATE software (care cuprinde aplicaţiile şi procesele care sunt rezidente pe testor şi care execută programul de testare), ATE Instrumente (care cuprinde instrumentele de măsură şi stimulare), Matricea de Conexiuni împreună cu Conectorul testorului şi cablarea acestuia, Interfaţa de Testare , Unitatea Testată.

Autorul descrie procesele, aplicaţiile şi documentele care participă la realizarea testului funcţional accentuînd interfeţele dintre domenii şi condiţiile obligatorii pentru portabilitatea testării.

2.1 Testarea funcţională în avionică.[20][200][197][199][205]

Dispozitivele electronice din alcătuirea unei aeronave, sunt verificate funcţional periodic, pentru a putea fi utilizate. **Testarea funcţională, este procesul de verificare a unui dispozitiv electronic, din punct de vedere al capabilităţii de execuţie a tuturor funcţiilor pentru care a fost proiectat, în condiţii cât mai apropiate de cele reale.** Testarea funcţională este un şir de experimente succesive în care funcţiile unităţii sunt verificate. Un astfel de experiment verifică o funcţie sau un grup de funcţii dependente una de alta şi se numeşte test. Această succesiune de teste, are ca obiectiv descoperirea eventualelor erori de funcţionare a unităţii respective. Un test este un proces independent, alcătuit din două părţi, prima care crează condiţiile (set-up) de verificare, a doua în care se fac verificările propriu-zise. În final testul se încheie cu o decizie de validare sau invalidare a verificărilor (GO, NOGO). În cazul unui test funcţional o unitate este aptă de serviciu dacă a trecut toate testele. Testele se aleg astfel încât să nu interfere între ele. Dacă un test cade (se termină cu erori), acest lucru nu trebuie să blocheze execuţia altor teste. Acest lucru permite evaluarea gradului de funcţionalitate al unităţii testate.

Procesul de testare funcţională este un proces complex la care participă componente software şi hardware care se clasifică în următoarele domenii de interes :

- **TPS (Test Program Software)** – reprezintă suma programelor de testare şi a fişierelor de configurare specifice unui test funcţional, care sunt proiectate pentru o anumită unitate.
- **ATE software** – reprezintă suma aplicaţiilor software rezidente pe echipamentul de testare automată care execută programele de testare şi celelalte funcţii legate de testare.

- **ATE Instrumente** – reprezintă toate instrumentele care alcătuiesc echipamentul de testare şi care generează stimulii sau măsoară diverse mărimi.
- **Conectorul Echipamentului de Testare** automată şi legăturile acestuia cu resursele testorului. Acest conector se defineşte în mod particular pentru un domeniu tehnologic (spre exemplu avionica)
- **Interfaţa de Testare** – reprezintă conexiunile între conectorul testorului şi unitatea testată, precum şi resursele suplimentare necesare. Aceasta este însoţită de obicei de un document descriptiv.
- **Unitatea Testată** – reprezintă obiectul testării şi este însoţită de mai multe documente care o descriu, şi instrucţiuni de operare în timpul testării.

Acest capitol abordează problema portabilităţii testelor funcţionale din zona tehnologică a avionicii. Autorul analizează pe parcursul tezei procesul de testare automată pentru avionică cu obiectivul de prezentare a acestui domeniu tehnologic, asociat cu preocupările şi cercetările personale în domeniu pe diverse paliere şi convergent spre definirea unei structuri portabile pentru testarea automată în avionică.

Domeniile enumerate sunt analizate din punct de vedere al portabilităţii şi al interdependenţelor necesare pentru atingerea acesteia.

2.2 Descrierea testelor funcţionale. (TPS)

Procesul de proiectare a testelor care alcătuiesc un program de testare funcţională cuprinde mai multe etape. Programul de testare este scris pe baza informaţiilor provenite dintr-un document care descrie testele. Acest document este creat pe baza diagramelor funcţionale definite de proiectantul dispozitivului sau mai recent de către inginerul care proiectează testele funcţionale. Diversitatea acestor documente descriptive a dus la necesitatea unei abordări mai formale a acestui proces. Avionica este un domeniu în care testarea este critică pentru siguranţa utilizării avioanelor. Din acest motiv s-a acordat o mare atenţie modului de generare a documentelor care descriu testarea. La unităţile de pe avioanele mai vechi, procesul de testare este descris doar în CMM (manualul de întreţinere al unităţii). Manualul unităţii este o documentaţie completă care cuprinde schemele şi descrierea unităţii, precum şi echipamentul şi procesul de testare. Cu toate acestea manualele au diferenţe în modul de descriere a unor teste similare, fapt care impune găsirea unui mod mai precis de descriere a testelor şi mai ales a unui mod independent de echipamentul de testare. Astfel apare standardul ARINC 416 care defineşte limbajul ATLAS (Abreviated Test Language for Avionic Systems) care precizează modul de descriere a testelor, a senzorilor şi stimulilor necesari pentru testare precum şi modul de organizare a programului de testare. Precizăm că ARINC 416 este un standard care priveşte testarea funcţională pentru avionică în toate dimensiunile acestui proces, nu doar formalismul limbajului ATLAS. Pentru un timp s-a folosit acest standard, după care s-atrecut la ARINC 626 (care defineşte ATLAS ca fiind "Abreviated Test

Language for All Systems", deci se extinde domeniul de aplicare) , cu echivalentul IEEE 716. În paralel cu limbajul ATLAS s-a definit şi structura unui document numit TRD (Test Requirement Document) care este foarte apropiat de ATLAS şi care la unităţile noi a devenit document sursă pentru generarea descrierii ATLAS. Prezentăm în figura 2.1 succesiunea activităţilor necesare pentru descrierea în format ATLAS a şirului de verifcări care alcătuiesc testul funcţional complet (Full Functional Test) al unităţii.

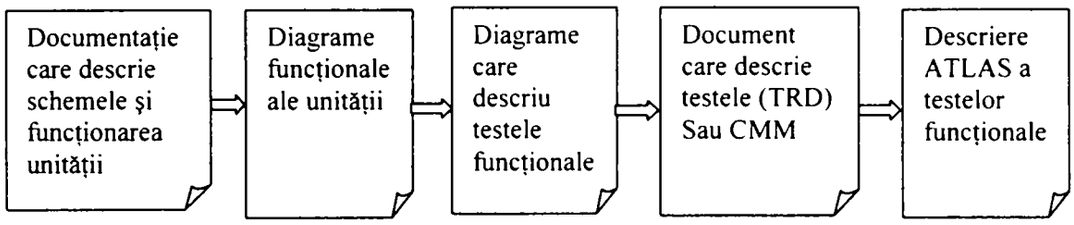


Fig. 2.1 Generarea documentaţiei descriptive a testelor funcţionale

Se remarcă faptul că **ATLAS** a fost iniţial **definit ca un limbaj descriptiv**. După crearea descrierii ATLAS a unui test funcţional, acesta poate deveni sursă de informaţii pentru programe de testare indiferent de limbajul de programare a testelor. Datorită caracterului descriptiv ATLAS a fost conceput de la început ca un sistem deschis permiţând adăugarea de noi domenii de tehnologie (deci verbe şi atribute noi). Directiva "EXTEND" a fost foarte cotroversată datorită posibilităţii de a introduce în mod necontrolat elemente de limbaj noi de către oricine şi deci pierderea caracterului de standard. Din acest motiv au apărut subseturile de limbaj care pot fi modificate numai cu aprobarea organismului de standardizare.

2.3 Programarea testelor. (TPS)

Programul de testare este entitatea care implementează descrierea testelor pe un echipament de testare. În realitate se crează un set de fişiere (Test Program Set) care permit execuţia testelor de către echipamentul de testare. Acest set de fişiere descrie într-un format special elementele virtuale cu care programul operează şi legătura acestora cu elementele fizice ale staţiei. Prima procesare asupra programului de testare după scrierea acestuia este verificarea sintactică. Verificarea sintactică este legată de limbaj şi de forma de editare a limbajului (uneori verificarea se face automat la editare). Verificarea sintactică este o operaţie care se execută şi asupra documentului descriptiv atunci când acesta este o descriere de tip ATLAS.

După acest pas urmează alocarea resurselor. Obiectelor virtuale din program li se asociază elemente fizice din structura echipamentului. Alocarea resurselor s-a făcut şi se face încă în majoritatea cazurilor manual de către programator. Există aplicaţii care fac automat alocarea din documentaţia de descriere (când este ATLAS) dar greşesc uneori, implicând astfel o post verificare. Acest lucru se întâmplă din cauza capabilităţilor pe care documentul de descriere le asociază senzorilor sau stimulilor. Din cauza caracterului general pe care documentul descriptiv îl are uneori, alocatorul automat găseşte mai multe opţiuni la dispoziţie. Procesul de alocare a resurselor genereză o descriere a legăturii dintre resursele virtuale şi cele reale.

O resursă particulară în structura unui echipament de testare automată este matricea de conexiuni. Aceasta face legătura dintre diverse puncte de conexiune ale unităţii de testare şi porturile instrumentelor. Alocarea acestor conexiuni este dificilă înainte de execuţia propriu-zisă a programului. Alocarea înainte de execuţie exclude conexiunile parametrizate. Acest lucru creşte excesiv dimensiunea programelor. Conexiunile parametrizate impun existenţa unui alocator dinamic în timpul execuţiei. Dacă operaţiunea de alocare se face static este nevoie de o descriere a conexiunilor posibile sau de un algoritm de găsire a acestora. Pe baza acestei descrieri la compilarea programului de testare se includ şi conexiunile reale în structura executabilului generat. Se prezintă în figura 2.2 paşii necesari până la execuţia programului de testare

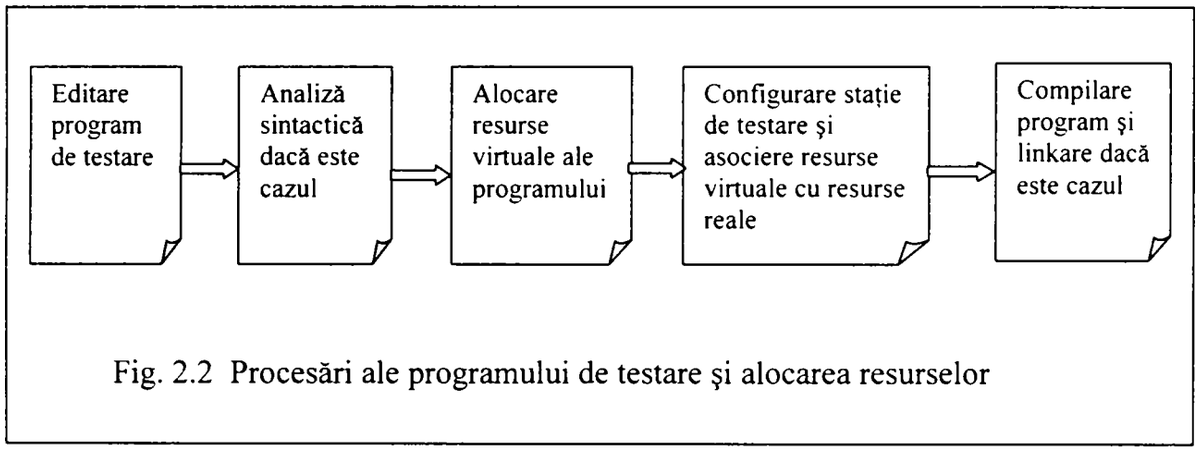


Fig. 2.2 Procesări ale programului de testare şi alocarea resurselor

2.4 Administrarea echipamentului de testare şi execuţia programelor. (ATE software)

Legătura dintre programul de testare şi echipamentul fizic se realizează de către aplicaţiile care administrează procesul de testare. Acestea, mai multe la număr, permit configurarea testorului, instalarea de instrumente, editarea diverselor fişiere, rularea programului, generarea de rapoarte, conectarea la baze de date şi la o reţea externă. Tot acest ansamblu de aplicaţii pe o platforma de operare se numeşte mediu de testare. Cel mai important aspect îl reprezintă legătura cu instrumentele. Această legătură se realizează prin drivere. În ultimii ani datorită necesităţii de interschimbabilitate a instrumentelor, prin evoluţia acestora şi modernizarea testoarelor, s-au dezvoltatat protocoale şi standarde noi referitoare la drivere. Astfel a apărut consorţiul IVI (Instrumente Virtuale Interschimbabile) care defineşte modul de construcţie a driverelor IVI, dar mai ales defineşte noţiunea de clase de instrumente şi capabilităţile asociate acestora. Acest lucru genereză posibilitatea proiectării unor testoare a căror configurare se face la nivel de clase de instrumente. Aplicaţiile care administrează procesele de pe echipamentul de testare pentru a conserva portabilitatea programelor de testare trebuie să accepte limbaje independente de hardware, legătura cu acesta fiind făcută prin drivere de nivel înalt legate la rândul lor cu drivere specifice ale instrumentelor. Alocarea dinamică a conexiunilor în principiu nu afectează portabilitatea, algoritmul local de conectare dacă este văzut printr-un driver IVI nu are importanţă. Uneori totuşi nivelul de complexitate al driverelor IVI nu este suficient şi este necesară adăugarea unui modul suplimentar care

gestionează o matrice complexă deasupra acestor drivere. Acest modul devine parte din setul portabil.

2.5 Instrumente folosite la realizarea testoarelor (ATE Instruments) şi modul de conectare a acestora la Conectorul Testorului.

Instrumentele care se folosesc astăzi sunt de două feluri: de sine stătătoare (care pot fi folosite independent) sau instrumente sub formă de plăci electronice în sertare VXI sau PXI sau în computere industriale. În avionică se folosesc anumite tipuri de instrumente datorită unei anumite tipologii specifice a circutelor electronice testate. Aceste instrumente sunt comandate prin intermediul unor magistrale de către drivere. Instrumentele sunt conectate prin matricea de conexiuni şi fire la un conector imens, care este conectorul testorului. Se prezintă în figura 2.3 legătura funcţională dintre test executiv (aplicaţia care execută programul de testare) şi conectorul testorului:

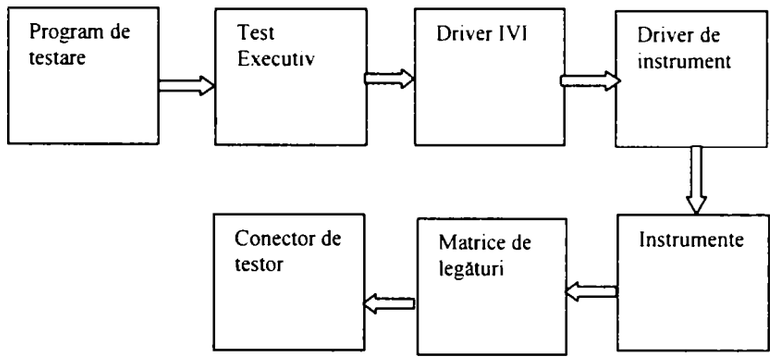


Fig. 2.3 Schema de principu a propagării comenzilor de la programul de testare transformate în semnale spre conectorul testorului

Se remarcă alături de driverele şi clasele de instrumente IVI necesitatea conectorului standard al testorului. Acesta este definit pentru avionică de către standardul ARINC 608.

2.6 Interfaţa de Testare şi Unitatea Testată

Interfaţa de testare (dacă testorul ar avea toate resursele necesare) este doar un cablu de conexiuni între acesta şi unitatea testată. De multe ori este necesară completarea resurselor testorului cu resurse care se adaugă în interfaţă. Acestea însă trebuie văzute ca aparţinând testorului. Pentru asigurarea portabilităţii singura informaţie care se adaugă setului portabil este descriptorul conexiunilor între conectorul testorului şi conectorul unităţii testate. Adăugăm la schema din figura 2.3 conexiunea la interfaţa de testare şi cu unitatea testată şi schema devine cea din figura 2.4:

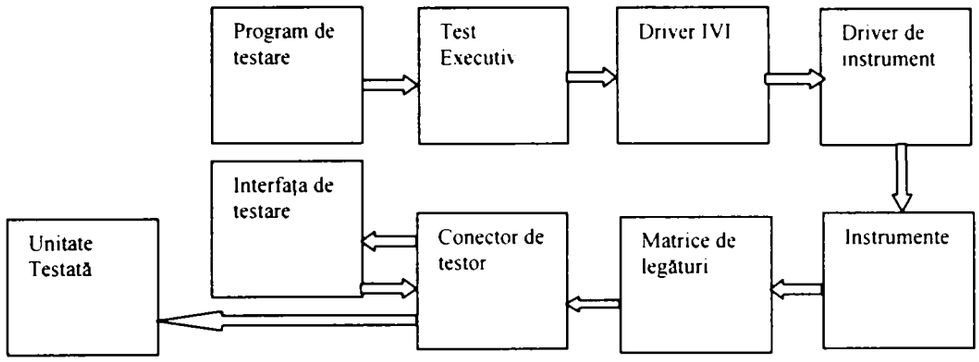


Fig. 2.4 Schema de principu a propagării comenzilor de la programul de testare spre unitatea testată

După cum se remarcă interfaţa de testare este transparentă pentru portabilitate.

2.7 Consideraţii asupra portabilităţii programelor de testare[203][204][206].

Procesul de alocare a resurselor a fost şi este problema fundamentală în faţa portabilităţii programelor de testare. Portabilitatea programelor de testare este un obiectiv fundamental pentru producătorii de testoare şi de avionică, atingerea acestuia implicând reduceri foarte mari de costuri şi de timp. **Portabilitatea programelor de testare se defineşte ca fiind procesul de transportare a unui set de programe şi documente asociate descrise într-un mod independent de un context hardware şi software de la o platformă de testare la alta**. Portabilitatea este cu atât mai bună cu cât operaţiunile de adaptare la noua platformă sunt mai puţine sau nu sunt necesare.

Portabilitatea programelor de testare este mai dificilă decât cea a aplicaţiilor software de sine stătătoare care sunt dependente doar de sistemul de operare. Problema portabilităţii este precedată de problema alocării resurselor. La descrierea testelor se folosesc pentru stimuli şi senzori entităţi cu capabilităţi foarte diverse. Cea mai precisă descriere, cea realizată în ATLAS, acceptă un domeniu foarte larg de entităţi fizice şi capabilităţi asociate. În prezent nici o aplicaţie nu este capabilă să discearnă cu precizie şi mai ales să selecteze un obiect real asociat. Această necesară suprapunere dintre descriere şi mulţimea reală a resurselor unui testor a fost abordată la momente diferite din două unghiuri diferite sau cel puţin independente unul de altul. Un punct de vedere a fost cel al descrierii testelor care a încercat să folosească entităţi virtuale care să facă descrierile independente de platformă. Un alt punct de vedere este cel al dezvoltatorilor de drivere de instrumente care urmărind interschimbabilitatea acestora au creat reguli şi proceduri independente pentru atingerea acestui obiectiv. Mulţimea instrumentelor virtuale a fost împărţită pe clase de instrumente. Fiecărei clase de instrumente i se asociază ca în programarea clasică pe obiecte un model funcţional, atribute şi metode care toate construiesc capabilităţile clasei respective de instrumente. După cum se remarcă trecerea de la descriere la programul de execuţie în acest moment este imposibil de realizat automat fiind necesară această legătură între obiectele virtuale cu care operează cele două. Tehnologia IVI (Instrumente Virtuale Interschimbabile) descrie foarte precis clasele de instrumente şi capabilităţile asociate, producătorii de instrumente punând la

dispoziţie deja drivere IVI, şi de aceea asigură portabilitatea între dispozitivele din clasa respectivă. Standardul IEEE 716 C/ATLAS este mai puţin precis din punct de vedere al legăturii cu echipamentele reale. Totuşi există numeroase încercări de al transforma într-un limbaj de programare. Atunci când sinteza instrumentelor din instrumente simple va fi posibilă pe scară largă, există posibilitatea ca procesul de alocare să se realizeze direct deşi varietatea şi multitudinea de posibilităţi va face foarte complex acest proces şi probabil se vor aduce modificări standardului. Rezolvarea acestei rupturi în fond între planul teoretic, descriptiv al testelor şi planul real al structurii echipamentelor de testare se face prin intercalarea unor procesări ajutătoare între programul de testare şi execuţia acestuia. Primul pas este configurarea testorului utilizând clase de instrumente instrumente virtuale. Al doile pas este transformarea limbajului descriptiv într-un limbaj intermediar care translatează elementele virtuale descriptive în elemente virtuale IVI şi transformă verbele cu acţiune multiplă în verbe simple. Translatarea dintr-un limbaj în altul este un proces automat, dar realizarea legăturii dintre cele doua mulţimi virtuale de resurse este un proces manual sau în cel mai bun caz semiautomat. Totuşi această descriere se face o singură dată la integrarea programului de testare şi devine parte din setul portabil.

Portabilitatea conexiunilor prezintă unele elemente specifice. În primul rând alocarea căilor de conectare trebuie să fie dinamică pentru a păstra generalitatea şi posibilităţile de parametrizare ale programului. Gasirea căilor de conectare este un proces dependent de structura matricii de conexiuni. Este necesar un descriptor al matricii de conexiuni care să fie utilizat împreună cu driverele IVI de către alocatorul dinamic. Se prezintă în figura 2.5 (transformarea figurii 2.2 pentru o structură portabilă) procesele de translatare a programului sursă şi de configurare şi alocare a conexiunilor:

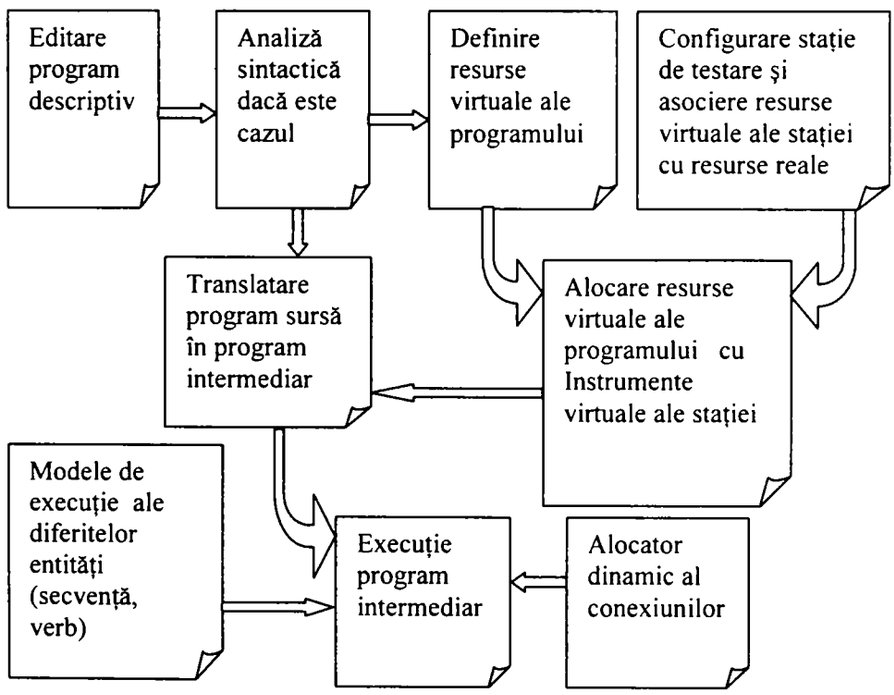


Fig. 2.5 Procesări ale programului de testare şi alocarea resurselor în varianta portabilă

Portabilitatea testelor funcţionale nu se reduce la programele propriu-zise. Echipamentul staţiei şi mai ales interfeţele acestuia trebuie să asigure portabilitatea. Primele experimente legate de portabilitate s-au făcut odată cu trecerea la o nouă generaţie de testoare şi apariţia necesităţii de translaţie a unor teste vechi pe noile echipamente. Autorul a participat la translaţia a peste 20 de programe de testare pentru avionică din sistemul avionului Concorde de pe testorul vechi pe un testor nou. Translaţia dintr-un limbaj în altul a fost simplă, problemele apărând la alocarea instrumentelor şi reproiectarea interfeţelor. Procesul a impus crearea unui set de instrumente virtuale ale testorului cu care să funcţioneze programul translatat. Tot acest experiment a avut loc într-o perioadă când consorţiul IVI era la început şi nu s-au folosit clasele de instrumente standardizate. Totuşi principiul de translaţie a fost acelasi : S-au creat instrumente virtuale la nivel de program, la nivel de testor, apoi s-a realizat legătura între aceste două mulţimi, s-a translatat programul sursă în noul limbaj şi în final s-a validat noul program. Efortul de migrare a programelor de testare a fost mult redus, dar inexistenţa unei standardizări la nivelul instrumentaţiei de pe noul testor (sub formă de clase de instrumente) şi inexistenţa standardizării la nivel de instrumente şi de conector al echipamentului pe vechiul testor a impus proiectarea acestor entităţi şi evident creşterea timpului necesar translaţiei.

Efortul de standardizare şi de structurare a diverselor segmente tehnologice din procesul de testare automată pentru avionică a implicat şi implică în continuare un număr important de companii, institute de cercetare şi universităţi. Portabilitatea programelor de testare este un proces care se va modifica şi perfecţiona în permanenţă. Portabilitatea este legată şi de modernizarea tehnologică, de apariţia unor noi domenii tehnologice. Dezideratul este realizarea unor structuri şi proceduri de proiectare a echipamentelor de testare care să permită extinderea ariei tehnologice de utilizare. Proiectul pentru o nouă versiune de ATLAS (ATLAS 2000) cuprinde noţiunea de domeniu tehnologic şi specifică entităţile şi interfeţele necesare la adăugarea unui nou domeniu. Pe de altă parte proiectanţii de unităţi dau tot mai multă importanţă testabilităţii acestora. Noile unităţi sunt digitale şi încapsulează o parte din teste în softul propriu sau pot să fie încărcate cu aplicaţii de testare, testul propriu-zis transformându-se într-un schimb de informaţii pe un canal de comunicaţie special între unitate şi testor.

Autorul a participat la analiza unui mare număr de unităţi de aviaţie din diverse perioade de timp, fapt care i-a permis să facă o selecţie a tipurilor şi procedurilor frecvente din testarea de avionică. De asemenea a participat la construcţia şi proiectarea unor testoare fapt care a permis o evaluare de detaliu a resurslor de instrumentaţie şi o analiză a modului de structurare a echipamentelor de testare automată pentru avionică. Autorul a participat sau a proiectat şi cercetat variante pe cont propriu de aplicaţii software, componente ale mediului de testare. De asemenea a proiectat variante proprii de limbaje intermediare bazate pe experienţa şi documentarea din domeniul instrumentaţiei virtuale. Date fiind aceste activităţi de studiu şi cercetare, autorul îşi propune în această teză parcurgerea domeniului testării automate pentru avionică prin prezentarea tuturor componentelor necesare acestui proces însoţite de exemple din activitatea proprie, de soluţii personale pentru unele dintre ele sau de clasificări bazate pe experienţele proprii din domeniu. Toate aceste componente sunt prezentate cu scopul de a identifica o structură deschisă de sistem de testare şi elementele necesare pentru asigurarea portabilităţii programelor de testare. In figura 2.7 se prezintă cele şase zone de interes ale

domeniului tehnologic al testării automate de aparatură de aviaţie cu toate procesările, interdependeţele şi punctele critice, pe care autorul le va urmări şi mai departe în teză :

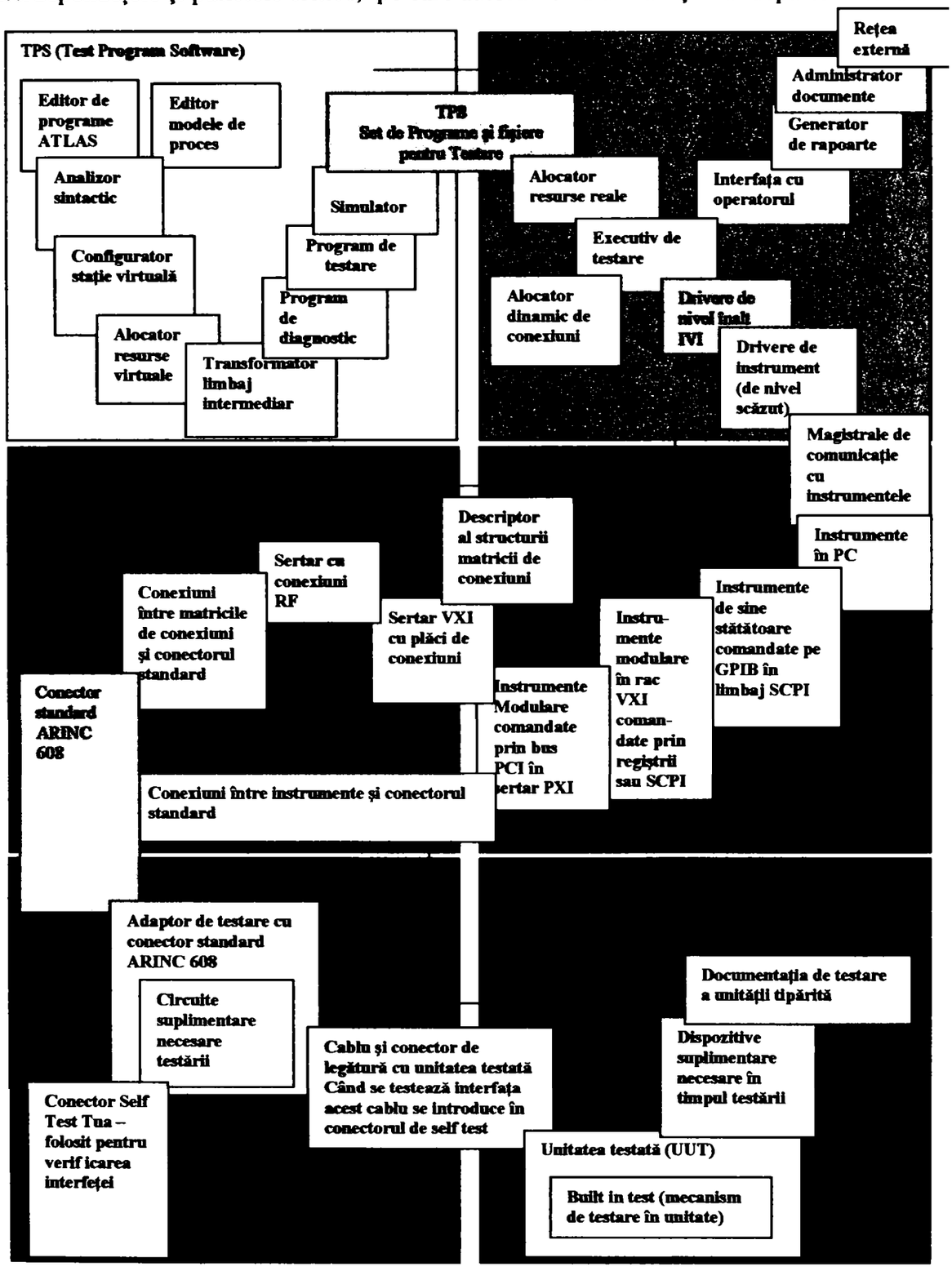


Fig. 2.6 Diagrama cu componetele procesului de testare automată în succesiunea participării lor la proces şi cu interdependenţele dintre domenii

2.8 Concluzii

Capitolul II clarifică aspectele legate de obiectivele acestei teze :

- Obiectivul principal al acestei teze este găsirea unei structuri de testare pentru domeniul avionicii care să asigure portabilitatea programelor de testare funcţională.
- Autorul defineşte domeniile de interes pentru o astfel de structură.
- Autorul analizează componentele participante la procesul de testare constatând discontinuitatăţile dintre planul descriptiv al testelor, planul software şi planul instrumentaţiei reale.
- Autorul constată necesitatea unor limbaje şi procesări intermediare care să elimine discontinuităţile şi să conserve portabilitatea.
- Autorul prezintă o diagramă a componentelor participante la procesul de testare automată în avionică cu interdependenţele dintre domenii. Definirea acestor componente permite identificarea elementelor critice pentru portabilitate.

Capitolele 3,4,5, prezintă în continuare în detaliu domeniul tehnologic al testării automate în avionică cu exemple şi contribuţii rezultate din activitatea practică a autorului, cu sublinierea elementelor importante pentru portabilitate.

CAP. III
Echipamente de testare automată pentru avionică.[17][18][21][23][24][90][96][110]-[120]

Acest capitol prezintă echipamente de testare automată folosite în industria de aviaţie pentru dispozitivele electronice (avionică), utilitatea acestora, structura precum şi elementele constitutive. Echipamentele prezentate, precum şi clasificările, arhitecturile, conceptele comentate şi uneori completate sunt rezultatul unei activităţi în domeniul testării efectuate sau coordonate de autor pentru mai mult de 200 de unităţi de avionică utilizând câteva tipuri de testoare.

Arhitectura sistemelor de testare automată cuprinde un computer, o matrice pentru conexiuni, instrumente de măsură şi stimulare şi interfaţa de testare. Această lucrare se bazează pe experienţa acumulată prin utilizarea unor astfel de sisteme la dezvoltarea de programe de testare şi interfeţe de testare pentru dispozitive electronice din componenţa unor aeronave de tip Boeing, Concorde, Airbus şi altele.

Aparatele de măsură uzuale sunt în general aparate care pot fi utilizate ca instrumente de laborator individuale şi independente sau pot conversa pe o magistrală de comunicaţie cu un computer şi pot fi comandate de acesta. Nici un testor nu cuprinde toate resursele posibile, dar construcţia sa este concepută să permită extinderea numărului de resurse în funcţie de mulţimea şi complexitatea unităţilor testate.

Un echipament specializat şi necesar pentru testele executate în regim automat este matricea de măsură. Aceasta asigură mobilitatea sistemului asupra unei mulţimi de puncte de măsură sau de injecţie de stimuli prin intermediul unui conector standardizat pentru avionică.

Testarea sistemelor de comunicaţie în avionică este un element foarte important care a impus proiectarea unor aparate de măsură şi stimulare specializate pentru comunicaţie.

Alimentarea cu energie electrică a aeronavelor se face cu generatoare electrice care alimentează dispozitivele electronice. Simularea acestor surse precum şi testarea unităţilor care controlează aceste generatoare impun folosirea unor surse de curent alternative speciale.

Structura echipamentului de testare precum şi necesităţile de simulare a mediului de funcţionare de pe avion impun uneori tehnici particulare de măsură.

3.1 Arhitectura unui echipament de testare automată [90][91][92][105][182][183][184][205]

Un echipament de testare automată este alcătuit dintr-un computer care prin intermediul uneia sau mai multor magistrale specializate comandă instrumentele care alcătuiesc ansamblul de testare.

Instrumentele pot fi de sine stătătoare şi comandate printr-o magistrală GPIB[23][24][99][128],sau instrumente de tip placă instalate într-un sertar VXI[95][96][97][129][163][166][168][190], sau plăci direct pe magistrala computerului, sau un sertar PXI[177] cu plăci specializate. Pe lângă aceste instrumente există comunicaţie serială RS232 sau RS485 şi comunicaţie de reţea tip Ethernet furnizate de computer.

Există diverse variante de construire a testoarelor dar în general ele includ câteva elemente comune:

- Computer;
- Instrumente de masură şi stimulare sub diverse forme de realizare;
- Magistralele de comunicaţie cu instrumentele;
- Matricea de conexiuni;

Un element interesant îl reprezintă magistralele de comunicaţie posibile dintre instrument şi computer. Obiectivul principal este atingerea unor viteze de comunicaţie cât mai rapide şi utilizarea unei magistrale pentru cat mai multe instrumente.

Testorul folosit în majoritatea timpului de către autor a fost un testor de tip SMART CATS[8] produs de compania RADA ELECTRONICS LTD. Acest testor are o structură standard pe care o reîntâlnim şi la alţi producători de astfel de testoare. Structura reflectă de fapt un concept de interconectare a instrumentelor de măsură şi stimulare care să permită extensia ulterioară a echipamentului precum şi automatizarea operaţiilor de testare.

Matricea de conexiuni este o componentă importantă şi complexă a sistemului. Pentru a face posibilă standardizarea s-a definit un conector al sistemelor de testare pentru avionică.

Autorul a participat la dezvoltarea de programe de testare pentru avionică de pe avioanele Boeing atât din categoria generaţiilor mai vechi (B737, B747) cât şi din generaţia foarte nouă de Boeing 777. Un astfel de proiect în favoarea companiei BAAE (British Airways Avionic Engineering) care are la Cardiff unul dintre cele mai mari centre de testare din lume, a cuprins peste 30 de unităţi pentru B777 care s-au testat pe testorul companiei Rada Electronic Industries. Testorul se prezintă in figura 3.1.

Fig. 3.1 Testor folosit la testarea de avionică de pe avionul B777

Un testor similar s-a folosit şi pentru translatarea şi integrarea a peste 30 de unităţi pentru avionul Concorde. La acest testor s-a adăugat o extensie care a permis reutilizarea unor interfeţe de la testorul vechi precum şi adaptarea la o conectică nestandardizată la vremea proiectării acestei aeronave.

În acest proiect s-au evidenţiat carenţele unor proiecte care nu s-au preocupat de standardizarea şi supravieţuirea în timp a echipamentelor de testare şi a procesului de testare . Lipsa de portabilitate a programelor a făcut necesară practic reproiectarea multor teste precum şi un proces de revalidare laborios.

Se prezintă în continuare în figura 3.2 o structură de testor care este o structură standard, dar care include echipamentul din figura de mai sus. Structura este folosită de toţi producătorii de astfele de echipamente, doar resursele utilizate pot fi diverse.

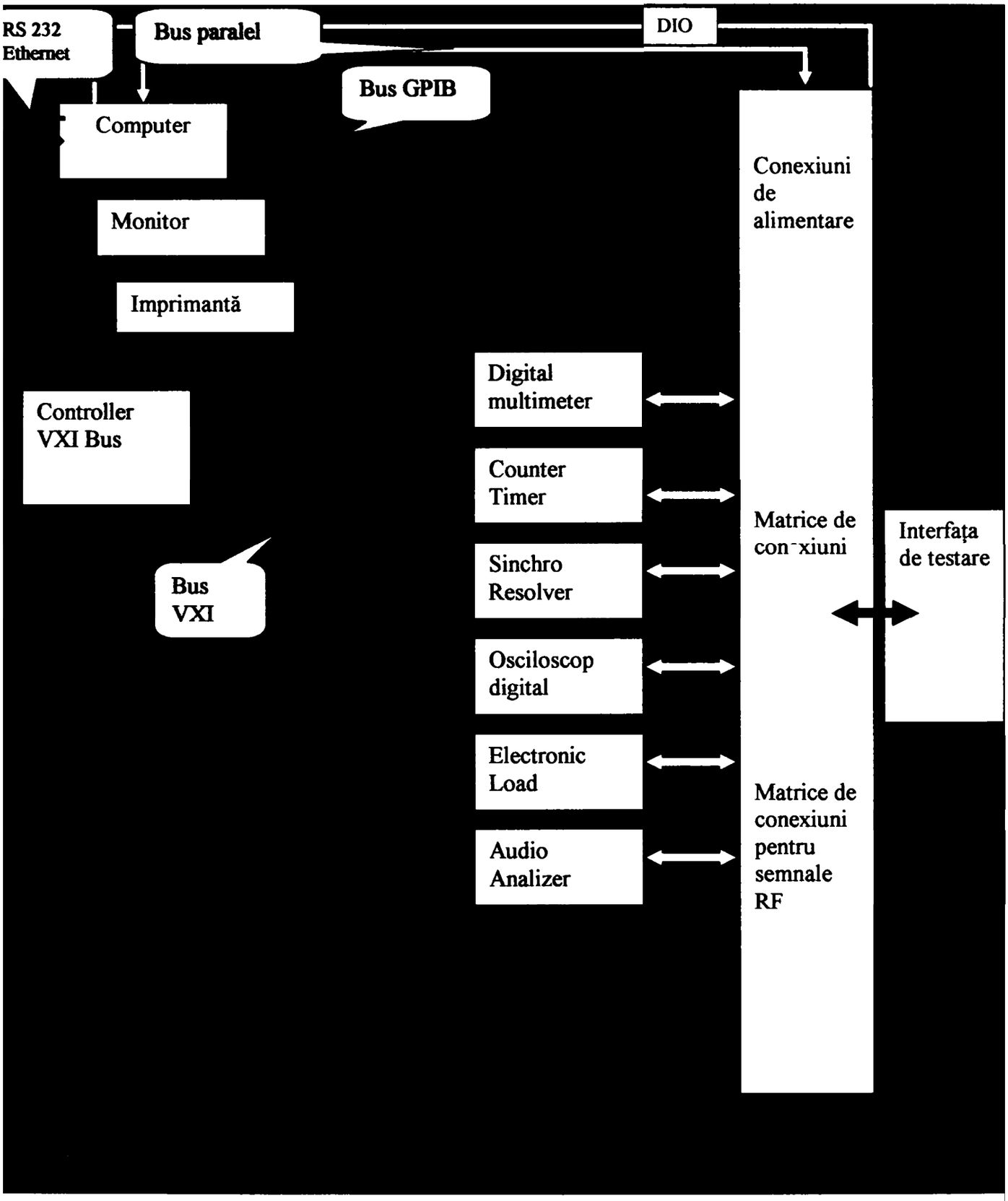


Fig. 3.2 Schemă bloc echipament de testare automată pentru avionică

Schema bloc din figura 3.2 cuprinde instrumentaţie şi pentru domeniul RF deoarece o parte din testele pentru B777 au fost în aceasta zonă. Se analizează în continuare elementele importante din această schemă bloc.

3.2 Magistrale de comunicaţie pentru controlul instrumentelor.[23][24][11][101][105][110]-[120]

Computerul care controlează instrumentele componente ale testorului are posibilitatea conectării cu acestea prin mai multe tipuri de legături.

3.2.1 Placi pe magistrale interne ale computerului.

Unele instrumente pot fi plăci care se introduc în interiorul computerului şi sunt controlate de acesta prin intermediul unei magistrale interne cum ar fi ISA (o variantă veche care nu se mai foloseşte dar care există la unele testoare mai vechi) sau mult mai frecvent PCI sau PCI+. Conexiunea la instrumentul propriu-zis pe magistrala PCI se face printr-un circuit numit pod (Bridge) care leagă o magistrală internă a plăcii respective de bus-ul PCI al computerului. Este o conexiune foarte rapidă şi producătorii de instrumentaţie odată cu miniaturizarea au extins foarte mult gama de produse. Există pentru aplicaţii mai restrînse testoare care au toată instrumentaţia sub formă de plăci în interiorul computerului. Acestea sunt testoare automate dar în general dedicate pentru un anumit tip de unităţi deoarece partea de matrice de conexiuni este mai voluminoasă. *Autorul a proiectat şi realizat un astfel de testor bazat pe un sertar de PC industrial (figura 3.3, 3.4, 3.5, 3.6) cu plăci de instrumente pe magistrala PCI a computerului, pentru unităţi de înregistrare video:*

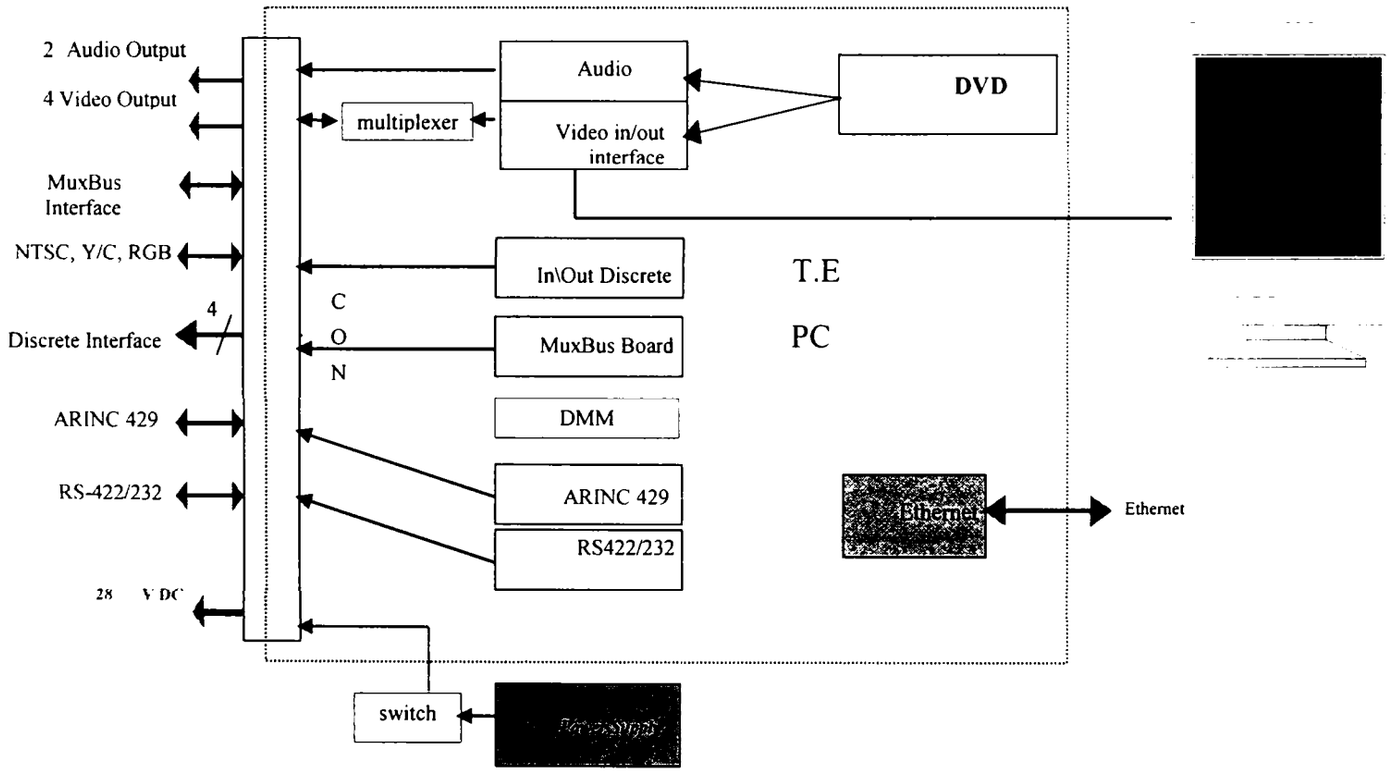


Fig. 3.3 Schemă bloc testor realizat pe structura unui PC industrial

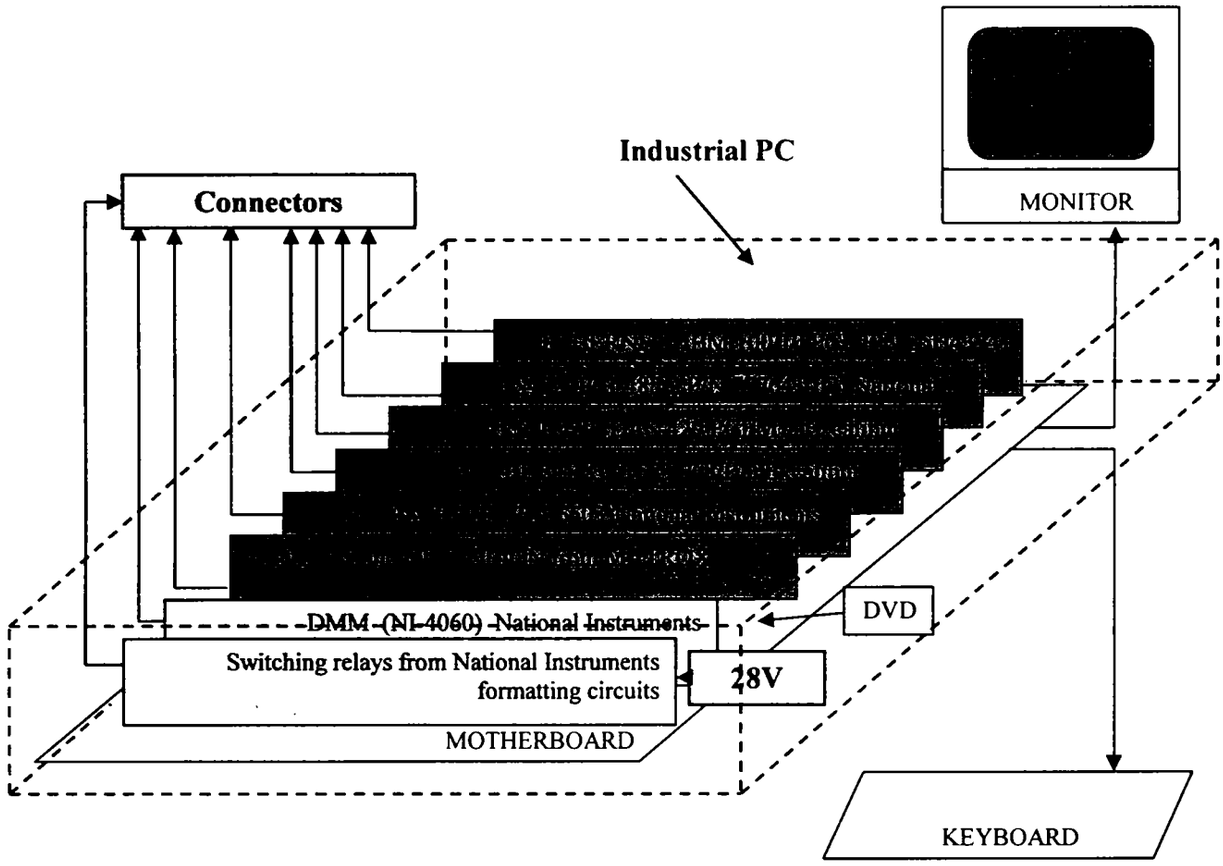


Fig. 3.4 Componenţa testorului cu prezentarea plăcilor
pe magistrala PCI a computerului

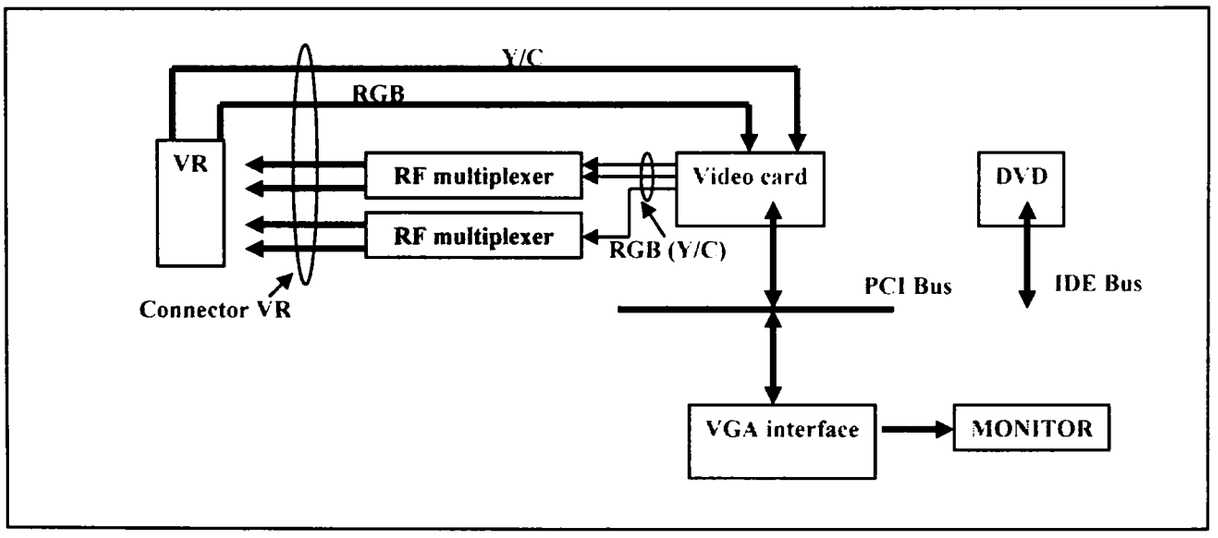


Fig. 3.5 Schema de testare a DVR (Digital Video Recorder)
Pentru partea de video.

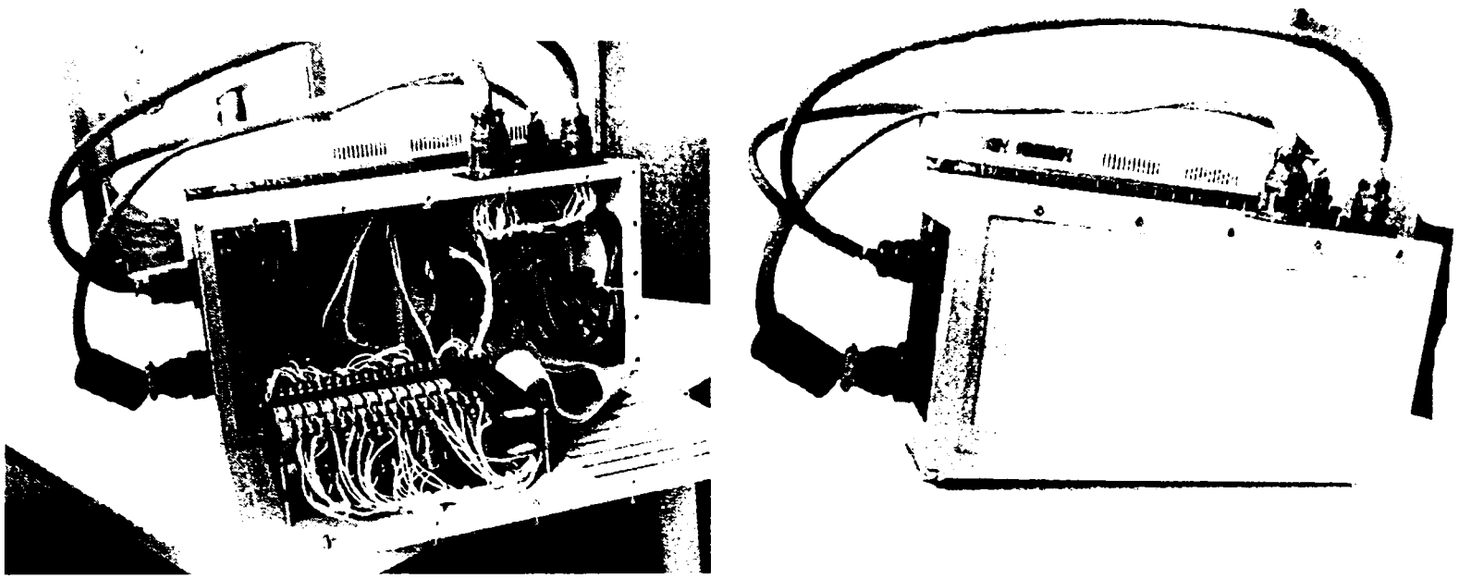


Fig. 3.6 Testorul proiectat şi realizat

3.2.2 Magistrala IEEE 488 (GPIB) [23][24][126]

"General Purpose Interface Bus" sau GPIB cum mai este cunoscut bus-ul de comunicaţie între un calculator şi diverse instrumente de măsură şi definit de standardul IEEE 488.1 a fost prima oară propus de către compania americana Hewlett-Packard. În 1975 IEEE a publicat standardul 488-1975 cunoscut sub numele de interfaţă digitală standard pentru instrumente programabile. Acesta specifică elementele electrice, mecanice şi funcţionale pentru sistemul de interfaţă. Acest standard nu specifică protocolul de comunicaţie si nici nu stabileşte convenţii de sintaxă şi format al comunicţiei.

Aceste elemente au fost adaugate printr-un standard suplimentar IEEE 488.2 numit "Coduri, formate, protocoale şi comenzi comune".In 1990 standardul IEEE 488.2 a fost completat cu un document numit "Comenzi Standard pentru Instrumente Programabile" sau SCPI care defineşte comenzi specifice pe care orice clasă de instrumente trebuie să le respecte. Magistrala GPIB este cea mai folosită de către producătorii de testoare deoarece foarte multe instrumente profesionale sunt construite cu o interfaţă compatibilă. Standardul IEEE 488 este compus din ansamblul IEEE 488.1 şi IEEE 488.2.

3.2.2.1 Specificaţia IEEE 488.1 [23]

Standardul IEEE 488.1 stabileşte că o interfaşă GPIB este o interfaţă digitală paralelă de 8 biţi care permite rate de transfer de pînă la 1 Mbyte/s. Bus-ul suportă un controller de sistem şi pină la 14 instrumente. Acestea sunt legate intre ele prin intermediul unui cablu cu 24 de fire ecranate, care are la capete un conector special. Acesta este în acelaşi timp şi conector mamă şi conector tată permiţând astfel racordarea în continuare a altor aparate. Standardul IEEE 488.1 include 16 linii de semnal si 8 linii de masă. Cele 16 linii se impart in 8 linii de date si 5 linii de comanda si 3 linii de "handshake":

CATEGORIA	LINIA	NUME
8 linii de date	**DIO 1-8**	**Data Input Output**
3 linii "handshake"	**DAV**	**Data Valid**
	NRFD	**Not Ready For Data**
	NDAC	**Not Data Accepted**
5 linii de management al interfeţei	**REN**	**Remote Enable**
	ATN	**Attention**
	IFC	**Interface Clear**
	SRQ	**Service Clear**
	EOI	**End Or Identify**

Sistemul GPIB foloseşte o interblocare pe trei linii care asigură transmisia şi recepţia de date pe cele 8 linii de date fară eroare.

Specificaţia electrică şi fizică impune câteva limite:

- Lungimea maximă a cablului folosit este de 20 m;
- Distanţa maximă între două aparate de 4 m, dar o distanţă medie de 2 m;
- Maxim 15 instrumente conectate pe bus cu cel puţin 2/3 alimentate electric;
- Semnalele sunt transmise cu nivele TTL şi în logică negativă, fiecare instrument are drivere de ieşire care pot să ducă 48 mA iar capacitatea de intrare nu este mai mare de 50pF;

La interfaţa GPIB linia ATN este utilizată pentru a distinge intre date pentru instrumente si mesaje de interfaţă. Astfel comenzile de programare a instrumentelor şi valoarea masuratorilor returnate vor fi intodeauna transmise cu ATN=o. Mesajele de interfaţă sunt trimise cu ATN=1 şi servesc unui scop bine definit de standardul GPIB. Bus-ul GPIB foloseşte o logică negativă aşa cum se observă din tabel:

NIVEL DE TENSIUNE		MESAJ		EXEMPLU
High	> 2 Volti	Fals	0	cind DAV line este Low
Low	< 0.8 Volti	Adevarat	1	mesajul DAV este True

Adiţional, procedura de "handshake" GPIB foloseşte conceptul de "SAU CABLAT" pentru liniile NRFD si NDAC. Aceasta inseamnă că NRFD trece in "High" când nici un dispozitiv nu trage această linie "Low", NRFD (Not Ready For Data) este negatul lui RFD (Ready For Data). Ca resultat sursa nu va primi RFD="TRUE" până când

toţi acceptorii nu vor trimite RFD="TRUE". De aceea se spune că un acceptor va trimite un mesaj "PASIV TRUE". Acelaşi sistem se aplică şi la Not Data Accept (NDAC) care poartă mesajul Data Accept (DAC). După ce controller-ul a adresat sursa (vorbitor) şi unul sau mai mulţi ascultători (acceptori), liniile de "handshake" sunt la nivelul iniţial iar nivelul lui DAV, NRFD şi NDAC sunt cele din figură. Sursa (vorbitorul) poate să înceapă acum sa transmită date catre acceptori:

1. Când sursa are un nou octet disponibil (nba) ea plasează data pe cele 8 linii de date;
2. Dupa un timp de stabilire T1, sursa trimite DAV="TRUE" (linia DAV = low) pentru a semnala acceptorilor că există date pe bus;
3. Ca resultat toţi acceptorii trimit mesajul RFD="FALSE" respectiv pun linia NRFD = "LOW";
4. Când unul din dispozitive a acceptat datele trimite DAC= "PASSIVE TRUE".
5. Când toate dispozitivele au acceptat datele, NDAC trece in "HIGH" Sursa primeşte acum mesajul DAC="TRUE";
6. Ca răspuns sursa trimite DAV="FALSE" (linia DAV="HIGH") şi mută datele de pe Bus;
7. Ca rezultat toţi acceptorii trimit DAC="FALSE" şi linia NDAC trece in "LOW".
8. Deşi datele au fost acceptate de toţi acceptorii nu inseamnă că toate dispozitivele sunt gata sa primească un nou octet de date. Acestea ţin RFD= "FALSE" până când sunt gata să primească un nou octet de date. Atunci RFD este trimis ca "PASSIVE TRUE";
9. Când toţi acceptorii au trimis RFD="TRUE", NRFD trece în "HIGH". Aceasta implică faptul că sursa primeşte mesajul RFD ="TRUE" şi toţi acceptorii sunt gata sa primească noul octet;
10. Sursa poate trimite acum următorul octet de date şi să reia ciclul de la punctul 1;

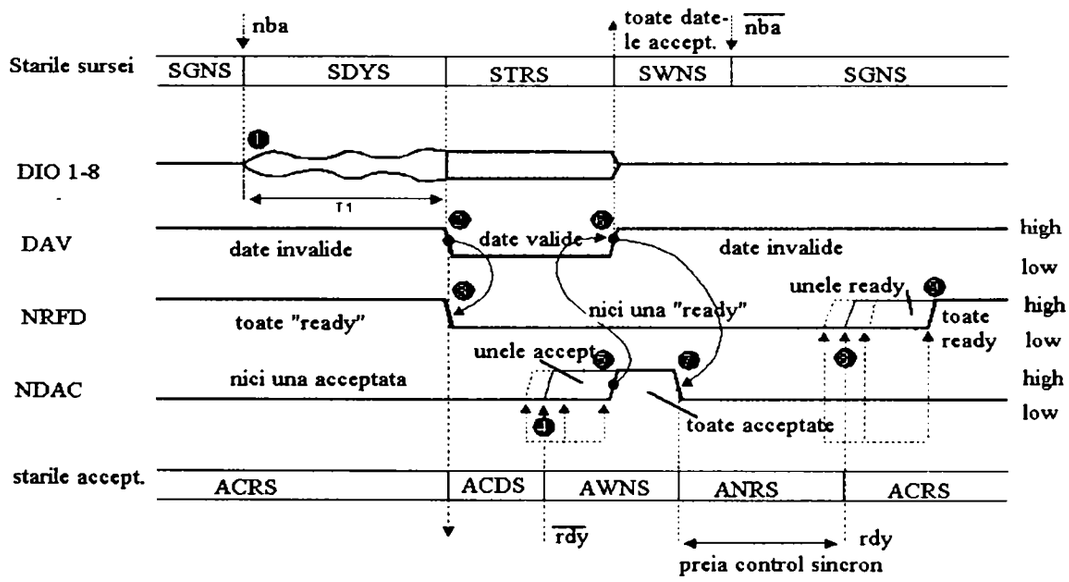


Fig. 2.6 Diagrame comunicaţie GPIB

3.2.2.2 Conceptul IEEE 488.2[24]

Standardul IEE 488.2 stabileşte sintaxa mesajelor. Acest standard asigură condiţia unei comunicaţii cu instrumentele care nu se blocheaza în nici o circumstanţă. De aceea a fost introdus protocolul MEP (Message Exchange Protocol) care se bazează pe principiul simplu că un instrument nu trimite date dacă nu se solicită acest lucru. IEEE 488.2 asigură de asemenea protocoale pentru: raportarea erorilor şi status-ului, sincronizarea între programul controller-ului şi acţiunile şi evenimentele instrumentului, sincronizarea între diferite instrumente. Definiţia semanticii mesajelor nu este parte din IEEE 488.2. Aceasta a fost lansată printr-un limbaj de nivel înalt cum este SCPI (Standard Commands for Programmable Instruments).

3.2.3 Magistrala VXI (IEEE 1155) [153]

Standardul VXI a fost dezvoltat de un consorţiu in 1987. IEEE a acceptat să introducă specificaţia pentru VXI sub forma IEEE 1155. VXI este folosit in aplicaţii de masură, ATE şi chiar în automatizări. Standardul VXI a redus dimensiunile plăcilor, a crescut viteza de transfer a datelor şi a permis implementarea instrumentului virtual.

Fizic un echipament VXI este construit dintr-un sertar în care se introduc cardurile. Specificaţia de construire a sertarului (bazată pe standardul industrial pentru VME (IEEE-1014)) include modul de construire a plăcilor, compatibilitatea electromagnetică, distribuţia puterii, răcirea şi circulaţia aerului de răcire. Modulele sunt instalate in ghidaje şi toate elementele de comunicare (LED-uri, Switch-uri, conectori) sunt instalate pe panoul frontal.

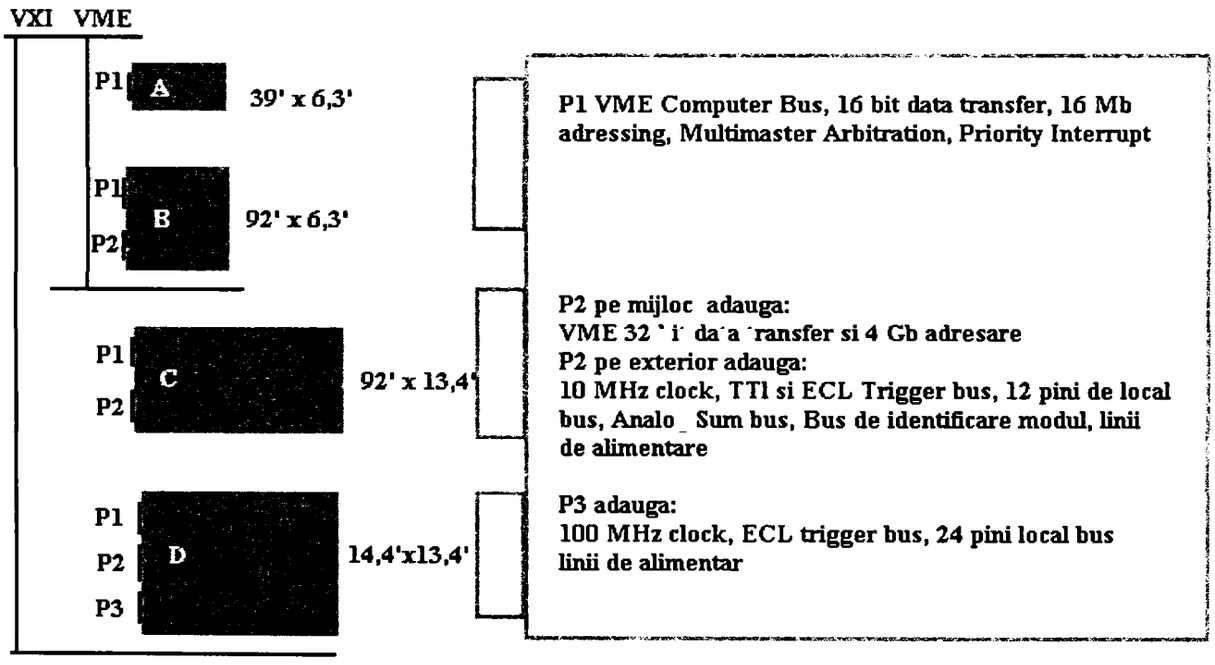


Fig. 3.7 Structură sertar VXI comparative cu VME

Fluxul de aer se trimite de jos in sus (de la P3 la P1 (fig. 3.7)).Specificația de răcire va fi astfel intocmită pentru toate modulele încât punctul de funționare să fie specificat pentru cel mai scăzut flux de aer.

Plăcile dintr-un sistem VXI nu se influențează prin radiație electromagnetică una pe alta. Dimensinea spațiului alocat pentru o placă s-a modificat de la 0.8 in. la 1.2 in. Fiecare placă are propria cutie de metal care este împământată la fundul de sertar. Specificația de VXI are precizări in legatura cu zgomotul sursei. De asemenea fiecare modul nu poate contribui la radiația de zgomot la distanță mai mare de 1/n din total (unde n este nr. de plăci in sistem). Pentru a minimiza zgomotul şi interferența la nivel de linii de trigger si de clock este necesar un fund de sertar monolitic şi singular pentru fiecare conector.

Plăcile VXI au un set de registre la adrese specifice după cum se vede in figură. Cei 16 KB superiori din spațiul de adresă de 16 biti (64KB) sint rezervați pentru dispozitivele VXI. Fiecare dispozitiv are o adresă logică de 8 biți care specifică unde sunt plasate registrele in spațiul de adresă. Un sistem VXI poate avea 256 de dispozitive. Adresa logică a unui dispozitiv VXI poate fi configurată de utilizator. În figura 3.8 se prezintă spațiul de configurare (adresele regiştrilor şi destinația lor) pentru o placă VXI :

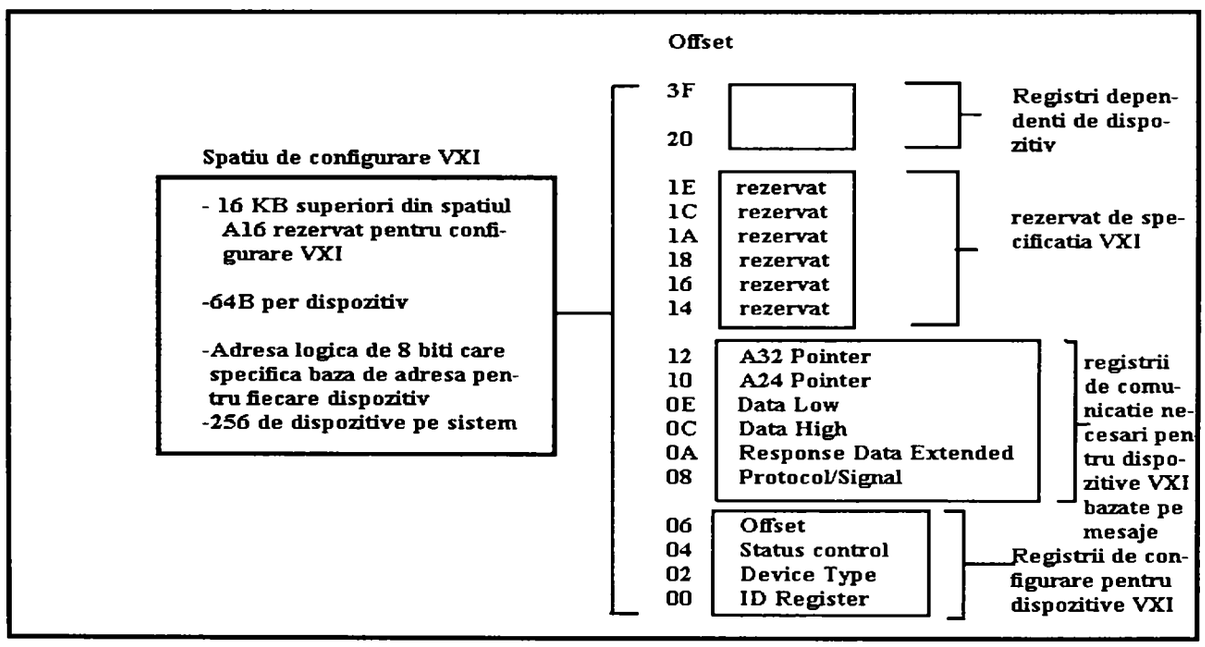


Fig. 3.8 Alocare spațiu de memorie standard VXI

Datorită registrelor de configurare VXI care sunt necesare pentru toate dispozitivele VXI sistemul poate identifica fiecare tip de dispozitiv VXI: tip, producator, spațiu de adresă, memorie. Dispozitivele VXI care au doar această capabilitate minimă se

numesc dispozitive bazate pe registre. Cu acest set comun de registre RM (managerul de resurse) care este un modul software poate automat configura sistemul şi memoria la iniţializare.

Suplimentar faţă de dispozitivele bazate pe registre specificaţia de VXI defineste dispozitivele pe baza de mesaj. Toate dispozitivele pe bază de mesaj pot comunica la un nivel minim folosind "Word Serial Protocol". Când un minimum de comunicaţie este posibilă, canale de comunicaţie cu performanţă ridicată pot fi stabilite, asa cum ar fi canale cu memorie comună care sa folosească viteza de transfer a sistemului VXI.

Protocolul VXI WSP ("Word Serial Protocol") este similar protocolului folosit de IEEE-488 care transferă date sub forma de mesaje de la un dispozitiv la altul sub forma de cuvinte. Toate dispozitivele VXI bazate pe mesaje folosesc WSP pentru a comunica standard. În general dispozitivele au un registru DATA IN şi un registru DATA OUT precum şi un registru de stare care conţine starea acestor registre. Sistemul functionează similar cu comunicaţia serială printr-un port de tip UART.

Standardul VXI defineşte un protocol Master/Slave permiţând construirea unor ierarhii . Ierarhia poate fi arborescentă, master putând fi orice dispozitiv care are alte dispozitive in subordine.

Un dispozitiv poate fi şi master şi sclav in acelasi timp. Un dispozitiv master are controlul comunicaţiei şi al configurării dispozitivelor din subordine. Un dispozitiv poate avea un singur master. Un dispozitiv master comunică cu dispozitivele subordonate prin registrele de comunicaţie ale acestora dacă acestea sunt dispozitive bazate pe mesaj sau prin registrele de configurare dacă sunt dispozitive bazate pe registre.

Dispozitivele subordonate pot comunica starea sau evenimentele către master folosind întreruperile hard sau semnale către registrul de semnale al master-ului. Dispozitivele care au numai statut de sclav transmit astfel de informaţii numai prin întreruperi în timp ce dispozitivele care sunt şi master pot utiliza fie calea întreruperilor fie a semnalelor. Specificaţia de VXI are comenzi definite care permit master-ului să înţeleagă capabilitaţile dispozitivelor subordonate, să le configureze şi să transmită întreruperi sau semnale intr-un mod particular. In protocolul WSP există posibilitatea stabilirii de legături pereche între două dispozitive fie utilizând o memorie comună fie prin registre.

Poziţia cea mai din stânga într-un sertar de VXI este folosită pentru câteva funcţii speciale. Dispozitvul din această poziţie (controller-ul de sertar) generează unele semnale cum ar fi: clock, sincronizari, etc.

Managerul de Resurse (RM) este un modul soft care poate să se afle pe oricare dispozitiv din sertar sau chiar extern acestuia. RM impreună cu controller-ul de sertar identifică fiecare dispozitiv din sertar şi stabileşte ierarhia sertarului. Dupa aceasta RM generează comanda de operare normală la dispozitivele master de cel mai înalt nivel. RM poate în anumite condiţii să oprească sistemul sau să-l reconfigureze.

Există mai multe configuraţii posibile pentru controlul unui sistem VXI. O primă configuraţie este aceea a sistemelor alcătuite dintr-un sertar VXI conectat la un controller extern printr-o conexiune GPIB. Controllerul comunică cu un modul de interfaţă aflat în

poziţia 0 în sertar. Interfaţa GPIB-VXI translatează protocolul GPIB către şi de la protocolul WSP.

A doua configuraţie este aceea a sistemelor care au în poziţia 0 un computer (embedded computer) care este de fapt un dispozitiv VXI şi este direct conectat la bus.

O altă configuraţie o reprezintă o conexiune pe un bus MXIla un computer.

În figura 3.9 este prezentat un exemplu de controller de sertar VXI. Acesta este un sistem de sine stătator care poate funcţiona ca TCC (test control computer) sau ca un simplu translator al mesajelor GPIB în mesaje de comandă pentru instrumentele VXI. In general în aplicaţiile de testare se foloseşte ca translator al mesajelor de comandă. Adesea acest controller conţine driver-ele necesare pentru conversia de la programarea în mod registru la limbajul SCPI (exemple Generatorul Arbitrar de Funcţii).

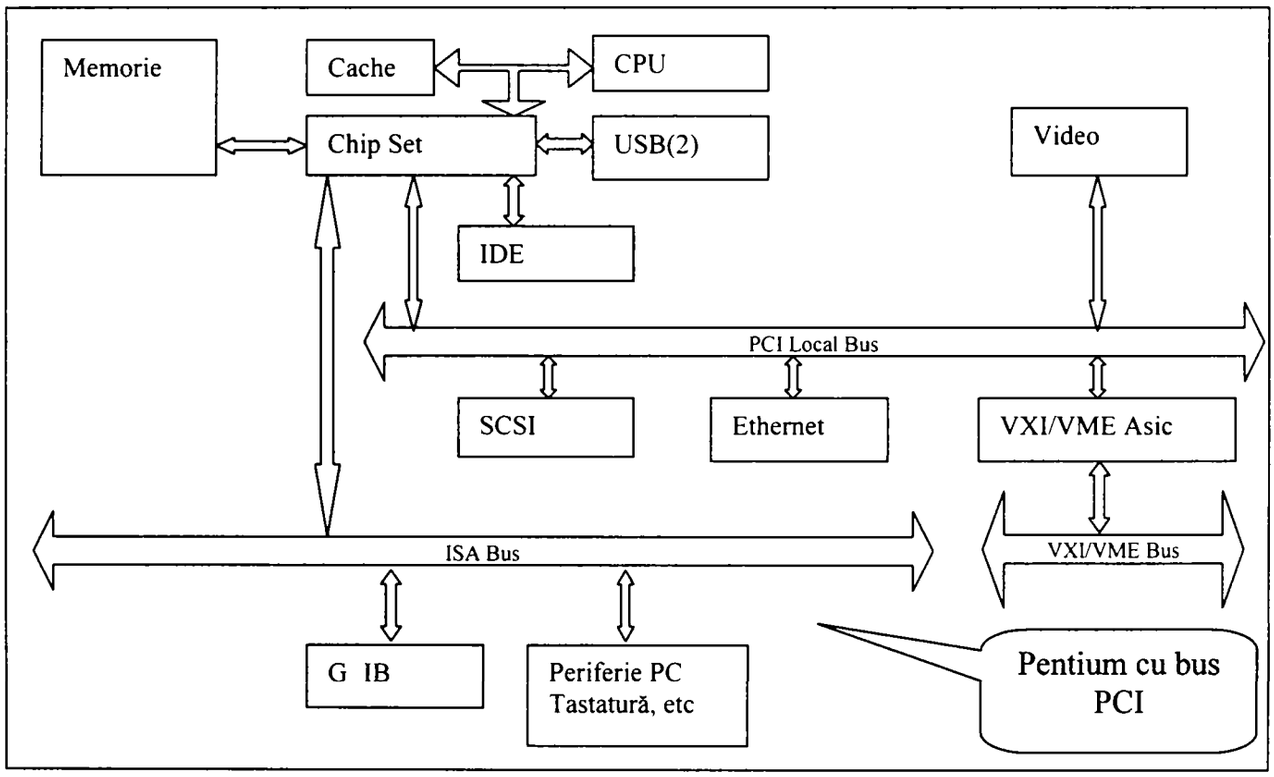


Fig. 3.9 Schema bloc a controller-ului de magistrală VXI

Schema bloc cuprinde câteva circuite importante:

1. Circuitul Video este o placă pentru bus-ul PCI care poate adresa 2MB de RAM şi foloseşte un bus de date intern de 64 de biţi.
2. Interfaţa IDE pentru transferurile către drive-ul de hard disk.
3. Circuitul de interfaţă între bus-ul PCI şi bus-ul VXI/VME. De obicei companiile producătoare folosesc un circuit ASIC.
4. Circuitul de conexiune la reţea de tip Ethernet pentru bus-ul PCI.
5. Interfaţa GPIB este de obicei un circuit pentru bus-ul ISA şi este de asemenea realizată cu un circuit ASIC.
6. Periferia pentru PC o reprezinta porturile pentru conectarea tastaturii, a mouse-ului, a unităţilor de disketă etc.

7. Sistemul I/O este un bloc care face legatura între bus-ul PCI şi bus-ul ISA. Include logica de arbitrare şi de DMA precum şi controller-ul de întreruperi.
8. Circuitul de interfaţă pentru bus-ul de tip SCSI care permite utilizarea unor dispozitive externe cum ar fi un CD-ROM.

3.2.4 Alte tipuri de magistrale.

Prin modificarea protocolului de conectare stadardul HS488 creşte viteza de comunicaţie de la IEEE 488.

Un alt bus folosit pentru instrumentaţie este bus-ul PXI care este de fapt o extensie a bus-ului intern al PC-ului (PCI) către un sertar cu instrumente de tip Eurocard. Prezentăm în figura 3.10 schema bloc a unui sistem bazat pe o conexiune PXI:

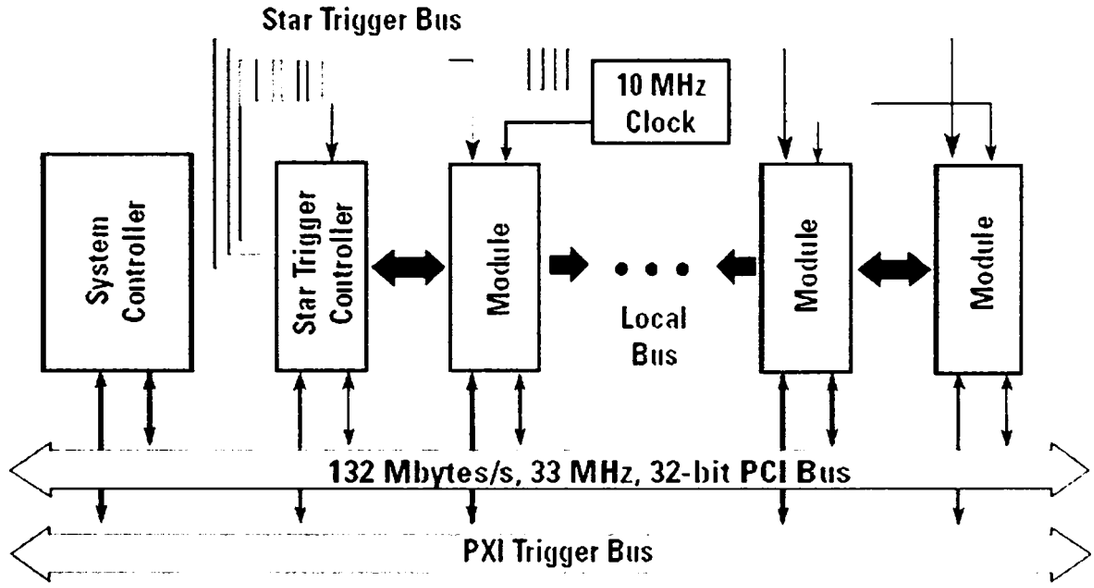


Fig. 3.10 Sistem PXI

După cum se remarcă bus-ul PCI este transferat către sistemul de măsură şi prin intermediul său sunt controlate instrumentele care au şi o magistrală locală pentru transferal unor mărimi analoge cât şi al unor comenzi.

USB (Universal Serial Bus) a devenit o conexiune populară între instrumente şi PC, iar versiunea USB 2.0 a stabilit o nouă categorie de dispozitve de vitză (480 Mbits/sec). De asemenea specificaţia pentru Clasa de Testare şi Masură USB (USBTMC) a introdus un protocol care permite o comunicaţie ca şi cea folosită cu protocolul GPIB.

LAN (Local Area Network) este un standard de reţea folosit în sistemele de măsură distribuite sau pentru instrumente aflate la distanţă. LXI (LAN eXtensions for Instrumentation) defineşte un subset al standardului LAN pentru instrumente de reţea de sine stătătoare bazat pe standardul IEEE 1588.

3.3 Aparate de măsură şi stimulare uzuale.

3.3.1 Multimetrul Digital (DMM).[119][29]

Multimetrul digital produs de Hewlett-Packard (acum denumit Agilent) tip HP 34401A este unul dintre cele mai folosite multimetre digitale în echipamentele de testare automată. Multe din teste implică măsurători de continuităţi, tensiuni continue sau alternative, frcvenţe, impedanţe, mărimi care toate se măsoară cu DMM-ul. Se prezintă în figura 3.11 schema bloc a acestui aparat:

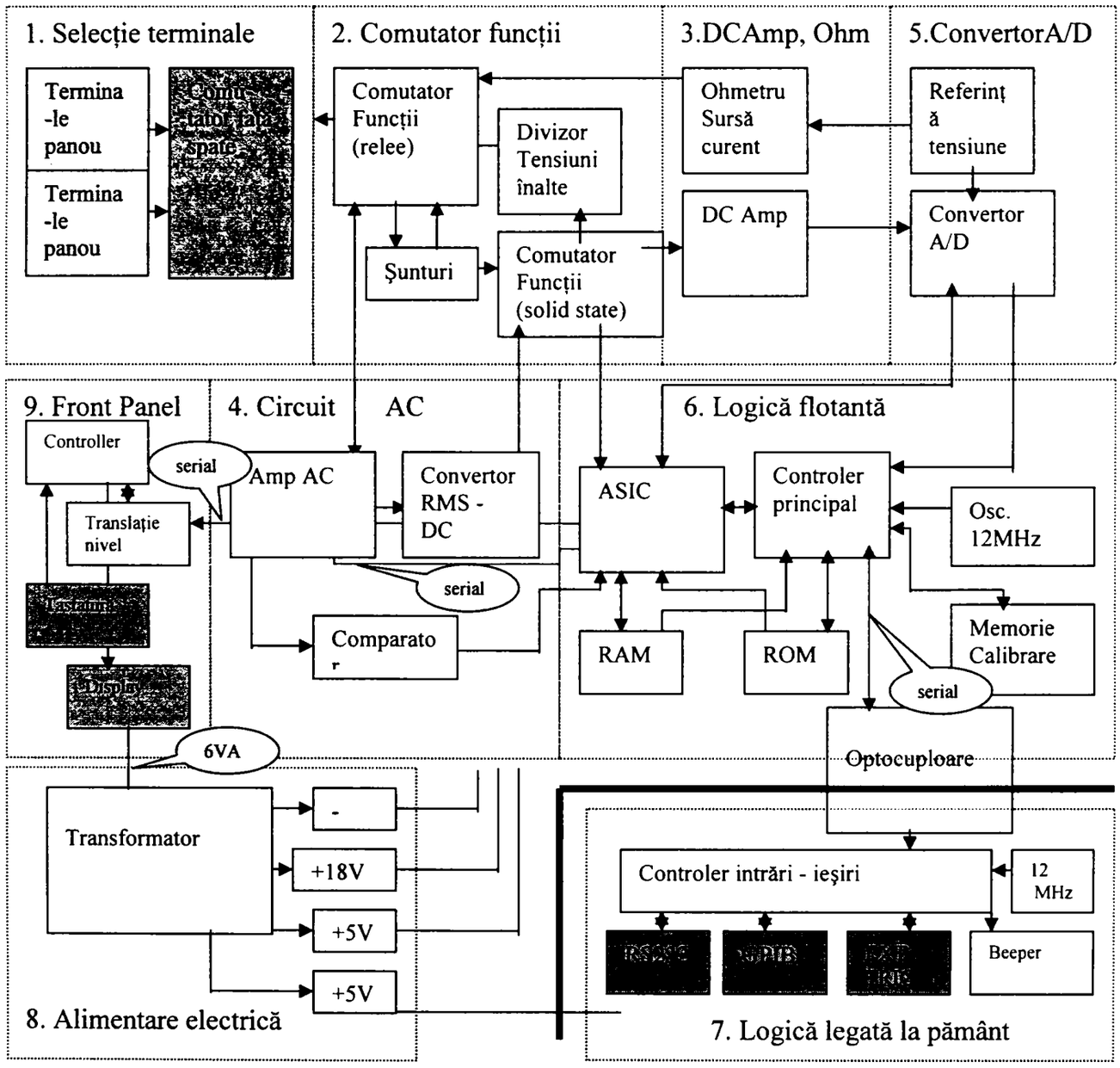


Fig. 3.11 Schema bloc DMM HP34401A

Din această schemă bloc se observă că aparatul are o parte din circuite flotante şi o parte legată la pământ. Această proprietate (de a fi flotant faţă de punctele de măsură) îl face extrem de util în testoarele automate unde DMM-ul este mutat prin intermediul matricii de conectare în diverse puncte de măsură ale unităţii testate.

Tipurile de erori ale aparatului sunt cele cunoscute: termice, de încărcare, de scurgeri de curent, datorate rejecţiei normale (când DMM-ul integrează şi masoară valoarea medie pe un interval de timp), datorate rejecţiei de mod comun imperfecte (rezistenţa de izolaţie este măsurabilă), datorate buclelor de câmp magnetic, datorate buclelor electrice spre pamânt. Erorile pot fi diminuate prin circuite de măsură corecte care reduc mărimea acestora comparativ cu valoarea utilă şi prin utilizarea funcţiei şi domeniului de măsură potrivit.

Funcţiile acestui DMM sunt următoarele:

- Tensiuni DC în domenii de la 100 mV la 1000 V;
- Rezistenţa (măsurată) pe 2 fire în domenii de la 100 Ohm la 100Mohm;
- Rezistenţa (măsurată) pe 4 fire în domenii de la 100 Ohm la 100Mohm;
- Curent continuu în domenii de la 10 mA la 3 A;
- Tensiuni AC True RMS în domenii de la 100 mV la 750V şi frecveţă de la 3 Hz la 300 KHz;
- Curenţi AC True RMS în domenii de la 1 A la 3 A şi frecvenţe de la 3 Hz la 5 KHz;
- Frecvenţa pentru domenii de la 3 Hz la 300 KHz şi tensiuni de la 100 mV la 750V;
- Perioada pentru domenii de la 1 sec la 10ms şi tensiuni de la 100 mV la 750V;

Toate aceste funcţii pot fi setate manual de la tastatură sau prin intermediul conexiunilor RS 232 sau GPIB. Aparatul permite comunicarea folosind limbajul SCPI (Standard Commands for Programmable Instruments). HP 34401A implementează şi comenzi suplimentare specifice aparatului.
Comanda DMM-ului de la distanţă poate fi executată după următoarea secvenţă:

1. Se setează DMM-ul într-o stare cunoscută (ex. RESET);
2. Se schimbă setarea aparatului în configuraţia dorită;
3. Se setează condiţiile de declanşare (Trigger);
4. Se iniţializează (armează) DMM-ul petru o măsurătoare;
5. Se declanşează măsurătoarea;
6. Se extrage valoarea măsurată (citită) din buffer-ul instrumentului;
7. Se mută datele măsurate la distanţă;

Măsurătoarea propriu-zisă se execută de la pasul 3 la pasul 5. Se prezintă în fig. 2.12 digrama operaţiunilor executate la o măsurătoare declanşată (cu trigger). Această diagramă este modelul de comportare a DMM-ului ca aparat generic în cazul măsurătorilor declanşate. Toţi producătorii de astfel de dispozitive care se aliniază la grupul capabilităţilor de bază vor respecta această diagramă.

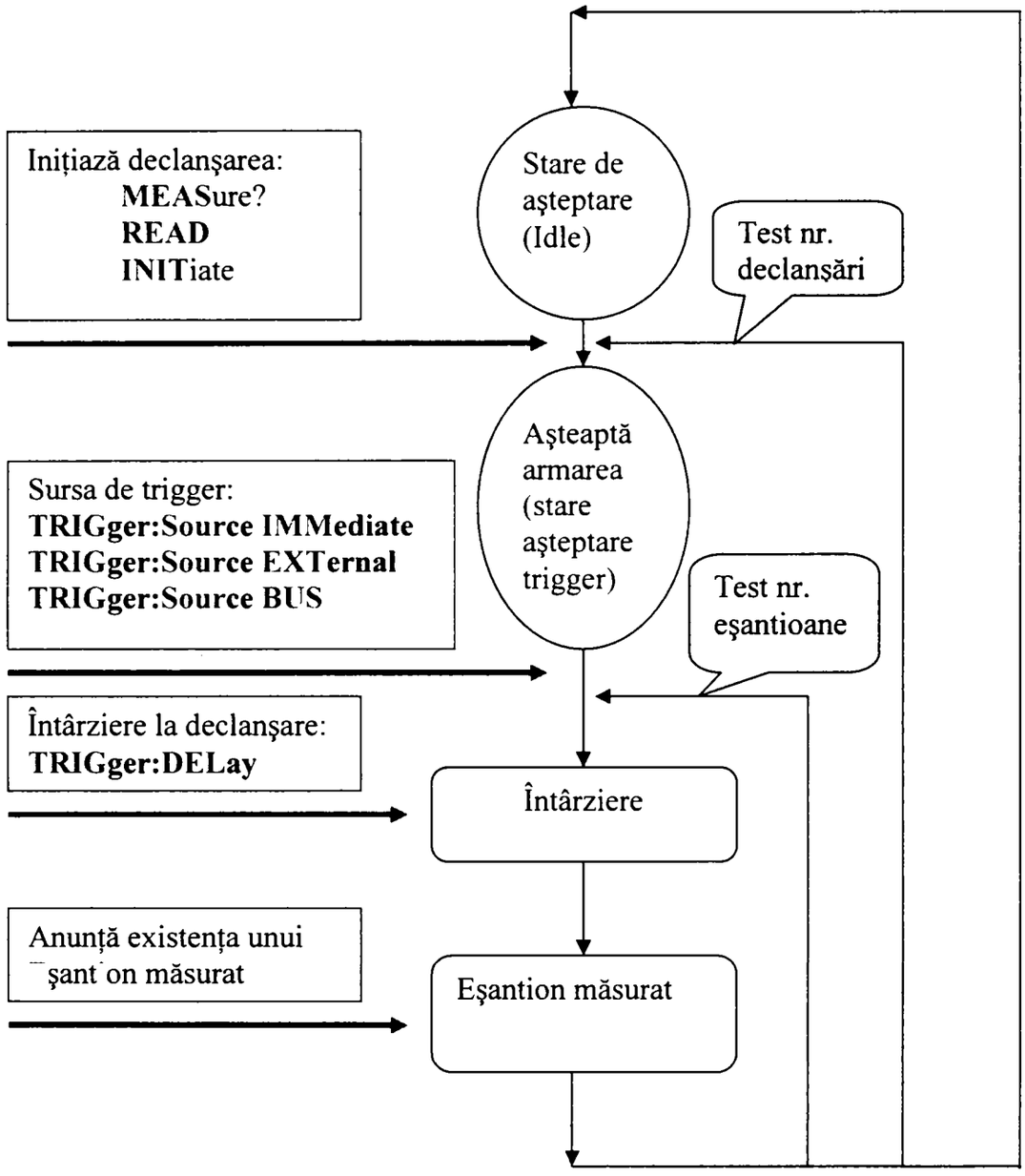


Fig. 3.12 Model de funţionare cu declanşare a DMM-ului

După cum se remarcă în comentariile din stânga (în fig. 3.12) secvenţa in limbaj SCPI este următoarea:

```
RST                          ;pune DMM-ul în starea de reset
CONF: VOLT: DC 10, 0.003     ;îl configurează
INIT                         ;declanşează masurarea
FETC                         ;extrage valoarea măsurată
```

Limbajul SCPI are o structura arborescentă. Un capitol important îl reprezintă sistemul de stări SCPI prin care se poate afla în detaliu starea aparatului prin citirea registrului de evenimente, de stare şi de date. Acestea pot fi citite sau setate cu un set special de instrucţii numite şi Comenzi Comune IEEE-488.2.

3.3.1.1 Capabilitățile de bază ale clasei DMM[29]

DMM-ul ca instrument generic (denumită şi clasa DMM) conform definiției consorțiului IVI (consorțiul pentru Instrumente Virtuale Interschimbabile) este un instrument care poate măsura valori scalare ale semnalului de intrare. Tipic acesta măsoară tensiuni, curenți sau rezistențe. Cu toate acestea clasa DMM suportă şi instrumente care măsoară şi alte mărimi precum temperatura sau frecvența. Clasa DMM suportă un grup de capabilități de bază şi mai multe grupuri de extensii de capabilități. Grupul capabilităților de bază este folosit pentru a configura DMM-ul pentru o măsurătoare tipică (setarea funcției de măsură, domeniului, rezoluției, sursei de declanşare), inițierea măsurătorii şi returnarea valorii măsurate. Capabilitățile de bază grupează un set de atribute şi funcții cu care se operează asupra acestora, valabile pentru toate instrumentele din clasa DMM. Grupul capabilităților de bază definesc următoarele atribute:

- Funcția, care specifică tipurile de măsurători pe care le poate efectua instrumentul (măsurători de tensiuni DC şi AC, măsurători de curenți AC si DC, măsurători de rezistențe pe 2 şi pe 4 fire);
- Domeniul, specifică domeniul de măsură şi unitățile de măsură;
- Rezoluția absolută, specifică rezoluția în unități de măsură absolute;
- Întârzierea la declanşare, specifică timpul între momentul declanşării şi momentul citirii valorii măsurate;
- Sursa de declanşare, specifică modul de declanşare al măsurătorii (imediat, extern, soft);

Grupul capabilităților de bază defineşte următoarele funcții:

- Abort, anulează acțiunea de măsurare;
- Configure Measurement, setează instrumentul;
- Configure Trigger, setează condițiile de declanşare;
- Fetch, extrage valoare măsurată;
- Initiate, inițiază măsurătoarea;
- Is Over Range, sesizează existența unei depăşiri de domeniu;
- Read, citeşte măsurătoarea din buffer;

Acest subcapitol nu şi-a propus să analizeze implementarea formală a capabilităților de bază, dar sublinează funcțiile şi performanțele unui DMM generic din punct de vedere al inginerului care implementează testul. Dacă s-ar construi un DMM generic interfața cu operatorul ar arăta ca în figura 3.13:

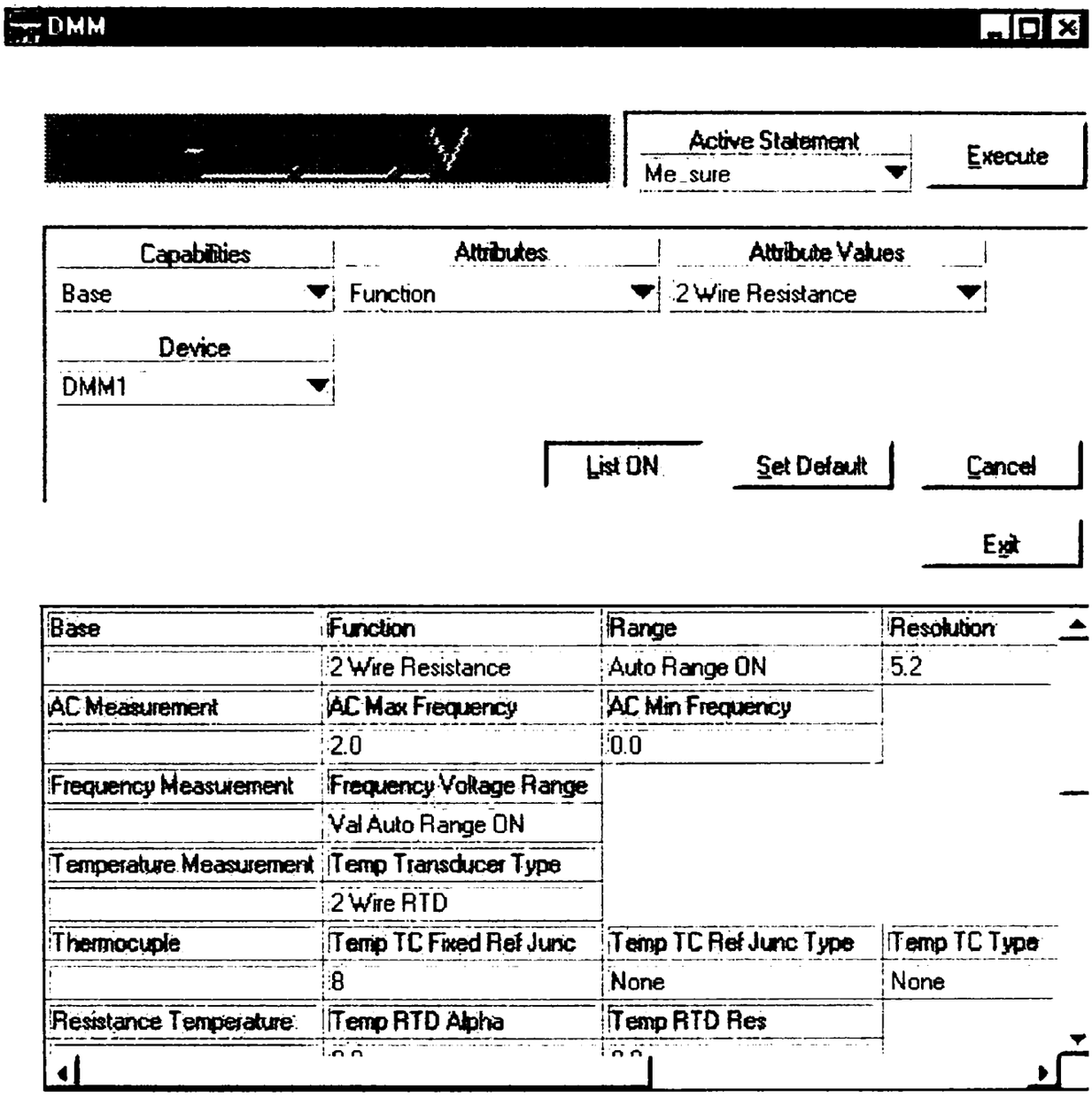


Fig.3.13 Interfaţă grafică DMM generic

Acest DMM a fost implementat ca instrument generic de către autor. Un instrument generic asemănător a funcţionat de la distanţă, operatorul aflându-se la Hamburg iar instrumentul propriu-zis la Tel Aviv. Autorul a participat şi la implementarea acestei aplicaţii pentru un instrument generic cu capabilităţiile de bază şi extensiile. În momentul demonstraţiei, instrumentul fizic a fost DMM-ul produs de HP dar ar fi putut să fie oricare alt DMM care îndeplinea capabilităţile de bază şi avea o interfaţă conformă cu IEEE 488.

3.3.2 Generatorul de funcții arbitrare (AFG).[30][112]

Generatorul de funcții arbitrare tip HP E1340A[112] este construit ca dispozitiv VXI. Acesta poate fi introdus într-un sertar VXI şi comandat prin intermediul bus-ului VXI. AFG este un dispozitiv indispensabil pentru un testor pentru avionică. AFG poate genera semnale de forme standard (sinus, triunghi, dreptunghi) sau poate genera forme de undă arbitrare (conform unei descrieri tabelare a semnalului) periodice sau aperiodice. Schema bloc simplificată a dispozitivului este prezentată în figura 3.14:

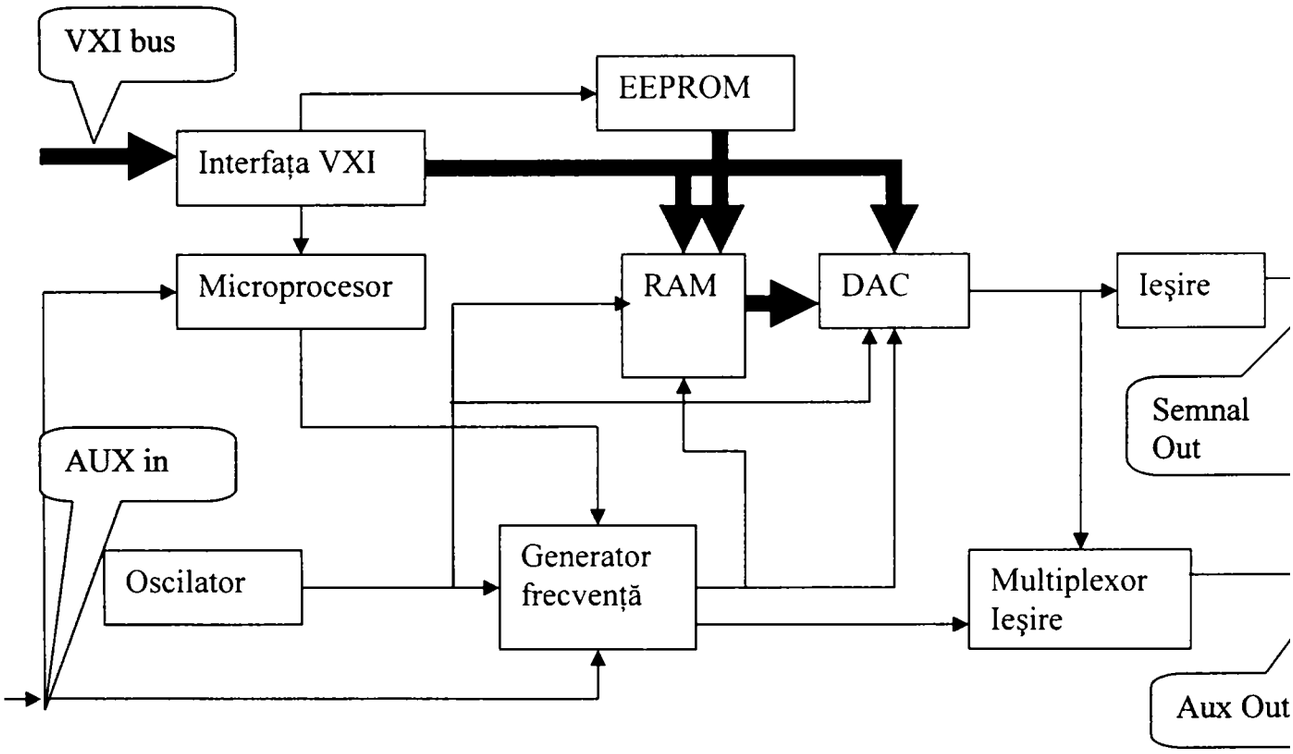


Fig. 3.14 Schemă bloc AFG

Placa E1340A este un dispozitiv bazat pe registre. Pentru a folosi limbajul SCPI este necesară instalarea unui driver in modulul de comadă al sertarului VXI. Acest driver converteşte comenzile SCPI în comenzi către registrele plăcii. Set-ul de comenzi SCPI implementează inclusiv grupul comezilor de stare, astfel utizatorul vede un instrumet de sine stătător programabil SCPI.

AFG este un instrument foarte important deoarece permite (mai ales pentru unitățile analoge) generarea unor forme de undă particulare, descrise prin fişiere sau forme sinusoidale şi triunghiulare prin care se injectează diverse semnale în timpul testării. AFG permite şi generarea unor semnale periodice cu perametrii de timp, formă şi amplitudine programabili. Uneori AFG poate fi folosit chiar şi pentru transmiterea unor mesaje pe diverse canale de comunicație prin simularea semnalului .

3.3.2.1 Capabilităţile de bază ale Generatorului de Funcţii (FG)

Clasa de instrumente „Generator de Funcţii", defineşte un instrument capabil să genereze o formă de semnal electric. Semnalul de ieşire are o formă definită (ex. Sinusoidal sau dreptunghiular). Unele instrumente sunt capabile să genereze şi forme arbitrare de semnal. Dacă generatorul de funcţii este capabil să genereze forme arbitrare, atunci semnalul de ieşire este o secvenţă de forme arbitrare repetate. Clasa Fgen este divizată între grupul capabilităţilor de bază şi grupuri de extensie. Grupul de bază se referă doar la semnalele de ieşire standard.

Schema bloc simplificată a unui generator de funcţii este prezentată în figura 3.15:

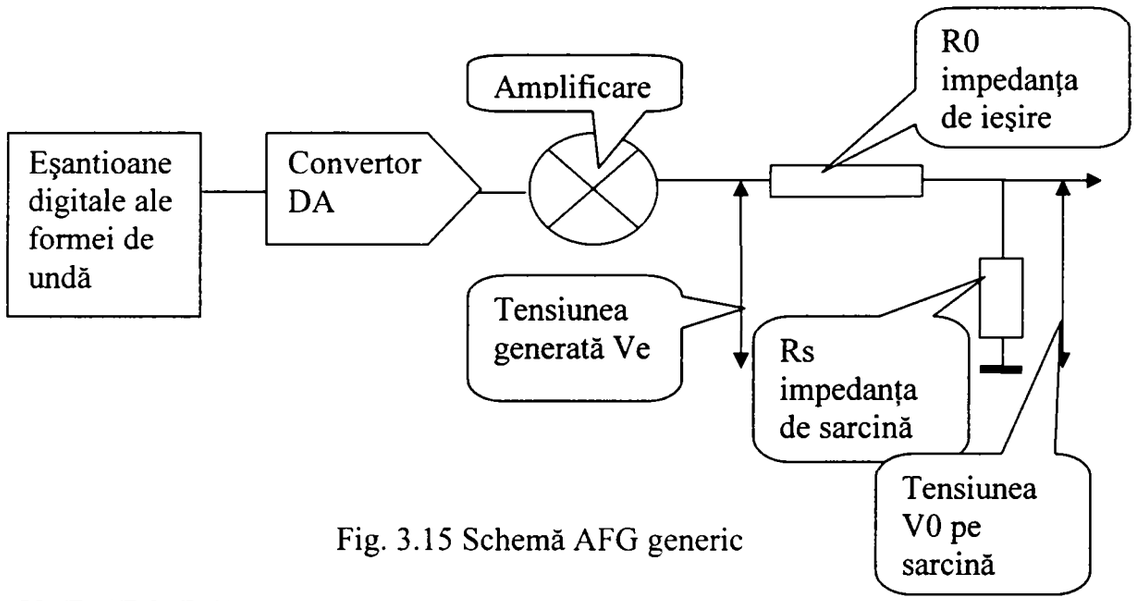


Fig. 3.15 Schemă AFG generic

$V0 = Ve(Rs/(R0+Rs))$

În funcţie de raportul între R0 şi Rs, V0 are diverse valori
(de ex. Pt R0=Rs V0=Ve/2 sau pentru R0<<RS V0=Ve)

Pentru grupul capabilităţilor de bază se definesc următoarele atribute:

- **Output Count** - defineşte numărul de canale ale aparatului;
- **Operation Mode** - defineşte modul de producere a semnalului continuu sau trenuri de semnal(burst);
- **Output Enabled** - specifică dacă semnalul produs apare la conectorul de ieşire;
- **Output Impedance** - setează impedanţa de ieşire în Ohm;
- **Output Mode** - specifică modul de producere a semnalului(standard, arbitrar, secvenţial);
- **Output Name** - returnează numele fizic pentru canalul respectiv;
- **Reference Clock Source** - specifică sursa ceasului de referinţă la generarea diverselor semnale;

Funcţiile cu care se operază asupra atributelor sunt:

- **Abort Generation** - anulează generarea unui semnal care a fost iniţiată;
- **Configure Operation Mode** – acţionează asupra canalului cu numele specificat modificând modul de operare;
- **Configure Output Enable** – acţionează asupra atributului care validează pentru canalul cu numele specificat prezenţa semnalului la conectorul de ieşire;
- **Configure Output Impedance** – modifică impedanţa de ieşire;
- **Configure Output Mode** – setează modul de ieşire: continuu, burst;
- **Configure Reference Clock Sorce** – setează sursa de ceas;
- **Get Channel Name** – solicită numele canalului;
- **Initiate Generation** – iniţiază generarea semnalului după configurare;

Generatorul de funcţii are câteva extensii din care o să amintim extensia IviFgenStdFunc care adaugă grupului de bază funcţii prin care se modifică proprietăţile formei de semnal (frecvenţă, amplitudine, DC offset şi defazaj). Aceste funcţii operează asupra formelor de semnal repetitive şi considerate standard de către producători:

- sinusoidal;
- dreptunghiular;
- triunghiular;
- rampă crescătoare;
- rampă descrescătoare;
- DC;

Atributele modificate de această extensie sunt:

- Amplitudune;
- DC offset;
- Factor de umplere;
- Frecvenţă;
- Defazaj;
- Formă de semnal;

Aceste atribute sunt modificate cu o singură funcţie: ConfigureStandardWaveform.
Aceasta extensie adaugă la modelul de comportare al AFG definit de grupul capabilităţilor de bază doar această funcţie care modifică atributele suplimentare dar care nu modifică modelul.

3.3.3 Osciloscopul Digital (DSO).[118][28]

Osciloscopul Digital este un alt instrument de măsură fundamental pentru un echipament de testare automată deoarece permite măsurarea unui număr important de mărimi electrice, precum şi implementarea unor teste particulare pentru evaluarea unor semnale cu forme deosebite. Se prezintă în figura 2.16 o schemă bloc a osciloscopului HP 54825A produs de Hewllet Packard utilizat pe testorul SMART CATS:

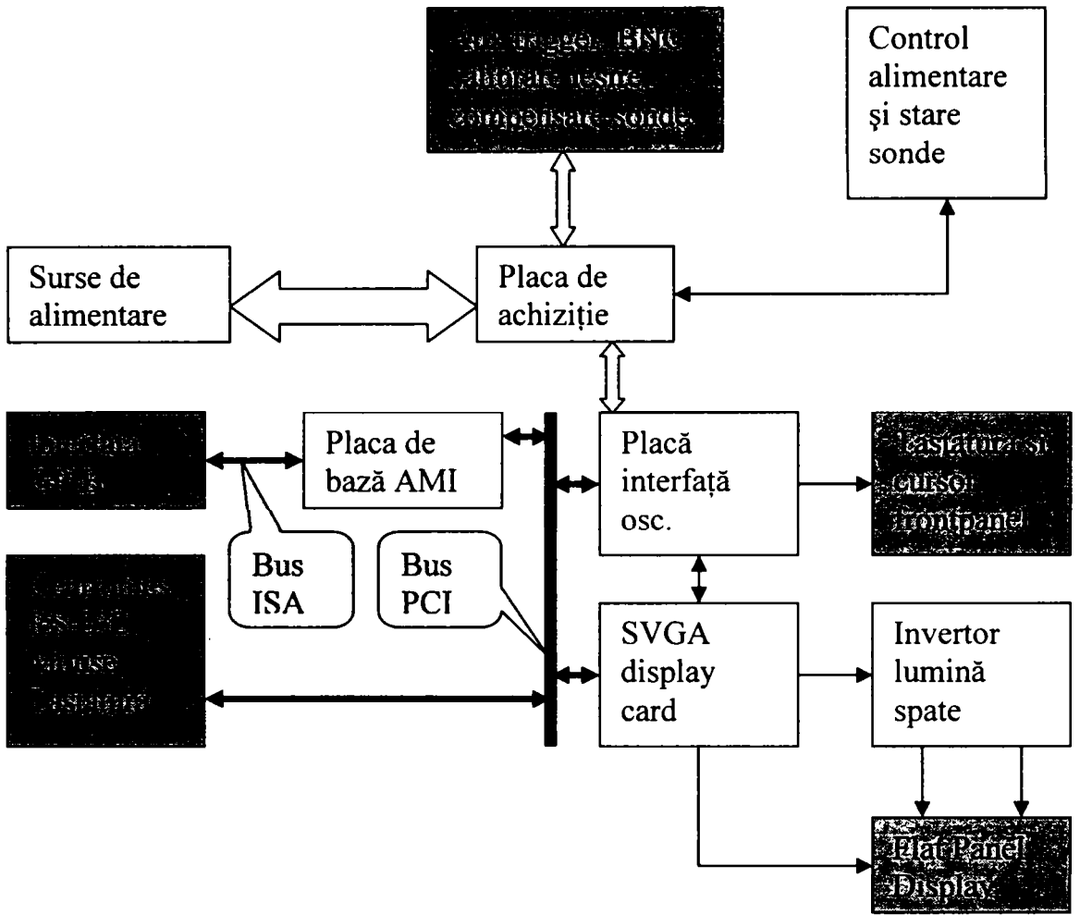


Fig.3.16 Schema bloc a osciloscopului digital HP 54825A

După cum se remarcă, instrumentul este construit pe structura unei plăci de PC la care s-au adăugat interfeţe specializate. Acest tip de osciloscop are 4 canale care funcţionează individual cu 2 Gsa/s. Fiecare canal este salvat într-o memorie de 32KB. Instrumentul poate să achiziţioneze simultan pe toate cele 4 canale şi are o bandă analogă de 500 MHz. Porturile, conectorii şi în general legăturile cu utilizatorul sunt figurate cu albastru în figură. Din punct de vedere al testării automate este importantă conexiunea GPIB (IEEE 488) care permite controlul osciloscopului de la distanţă. Instrumentul comunică cu un computer prin intermediul unor mesaje conform sintaxei specificate de standardul 488.2.

3.3.3.1 Structura de raportare a stării instrumentului[118]

La fel ca şi DMM-ul şi AFG-ul şi acest osciloscop are implementată o structură de raportare a stării instrumentului care poate fi utilizată prin intermediul comenzilor SCPI comune. Se prezintă în figura următoare schema bloc de raportare a stării (se aplică pentru toate instrumentele care au implementat SCPI).

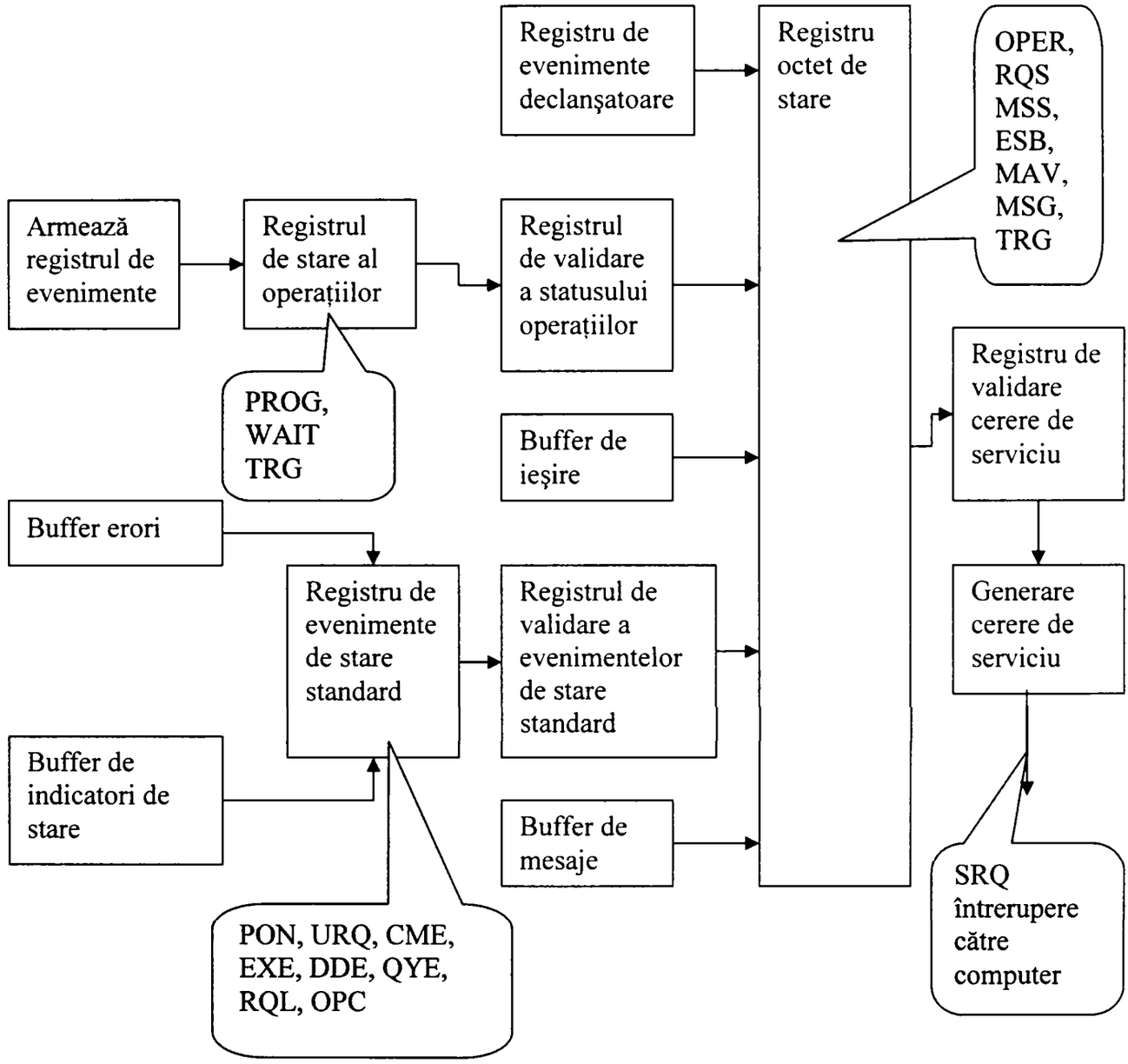


Fig. 3.17 Structura de raportare a stării osciloscopului.

Biţ-ii care raportează starea sunt următorii:

PON – power on;
URQ – nefolosit;
CME – Command Error;
EXE – Execution Error;
DDE – Device Dependent Error;
QYE – Query Error;
RQL – Request Control;

OPC – Operation Complete;
OPER – Operation status register;
RQS – Request Service;
MSS – Master Summary Status;
ESB – Events Status Bit;
MAV – Message Available;
USR – User Event Register;
TRG – Trigger;
LCL – Local;
FAIL – Fail;
COMP – Complete;
WAIT TRIG – Trigger;

Toţi aceşti biţi de stare pot fi citiţi prin intermediul comenzilor SCPI comune:

***CLS** – şterge regiştrii de stare şi eroare;
***ESE** – setează biţii de evenimente standard în registrul de validare ev. de stare;
***ESR?** – citeşte conţinutul reg. de evenimente;
***IDN?** – returnează identificatorii producătorului;
***LRN?** – reurnează setarea aparatului;
***OPC** – setează bit-ul de operaţie completă cânnd toate operaţiile dependente de dispozitiv s-au terminat ;
***OPT** – returnează lista opţiunilor instalate;
***RCL** – reface o setare a instrumentului salvată anterior;
***RST** – setează instrumentul într-o stare cunoscută;
***SAV** – salvează starea curentă;
***SRE** – setează biţii registrului pentru cereri de servicii;
***STB?** – returnează conţinutul registrului de stare;
***TRG** – achiziţionează semnalul dacă există condiţii de declanşare;
***TST?** – iniţiază un self test şi returnează un mesaj de eroare;
***WAI** – opreşte instrumentul de la excutarea unor comenzi noi până la terminarea celor în curs ;

Pe lângă comenzile comune instrumentul are implementate două categorii de comenzi cu care execută funcţiile specifice ale osciloscopului:

1. Comenzile de nivel primar (Root Level Commands) care implementează operaţiile de bază ale instrumentului, echivalente cu comenzile manuale de pe panoul frontal.
2. Comenzile de subsistem care sub un nod cuprind toate comenzile referitoare la un subsistem funcţional (ex. Baza de timp).

Osciloscopul este unul dintre instrumentele cele mai frcvent folosite de către sistemele de testare automată pentru complexitatea posibilă a testelor.

3.3.3.2 Capabilităţile de bază ale Oscilocopului Digital (SCO) [28]

Clasa instrumentelor de tip osciloscop (Scope) este clasa instrumentelor care pot achiziţiona forme de semnal care se modifică în timp. Clasa Scope este împărţită în grupul capabilităţilor de bază şi extensii. Funcţiile şi atributele grupului de bază sunt folosite pentru achiziţii tipice de semnal (setează canalul, achiziţia, şi subsistemul de declanşare), iniţiază achiziţia şi returnează forma de semnal. Grupul capabilităţilor de bază suportă doar declanşarea pe front pentru osciloscoape care pot achiziţiona pe mai multe canale.

Grupul capabilităţilor de bază defineşte atributele şi funcţiile care configurează trei subsiteme: canalul, achiziţia şi declanşarea.
Astfel pentru subsistemul de canal se definesc atributele:

- Validare canal;
- Atenuare sondă;
- Mod de cuplare (AC,DC);
- Offset;
- Domeniu;

Pentru subsistemul de achiziţie se definesc atributele:

- Timp de start la achiziţie;
- Tip de achiziţie;
- Număr minim de puncte orizontale;
- Durata unei înregistrări;

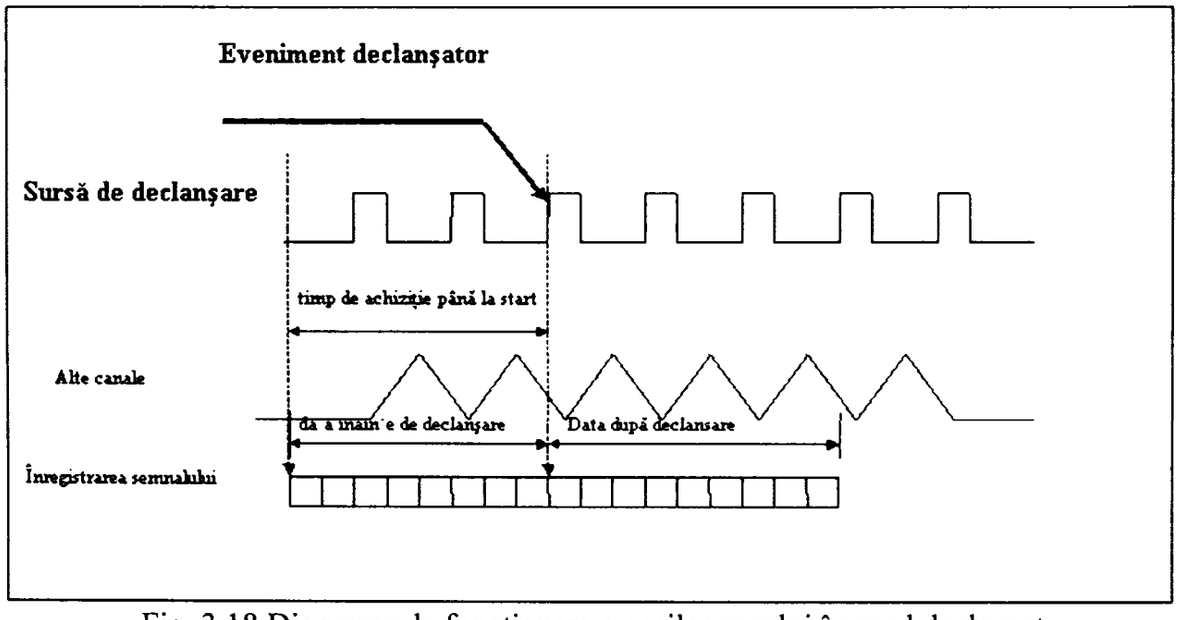


Fig. 3.18 Diagrama de funcţionare a osciloscopului în mod declanşat

Pentru subsistemul de declanşare se definesc următoarele attribute:

- modul de cuplare al sursei de declanşare;
- timpul după prima declanşare în care instrumentul nu poate fi declanşat din nou;
- tipul de eveniment care declanşează;

Declanşarea pe front se configurează modificând următoarele attribute:

- nivelul de declanşare;
- sursa de declanşare;
- frontul (crescător, descrescător) de declanşare;

Atributele sunt modificate cu următoarele funcţii:

- **Abort** – anulează achiziţia;
- **Acquisition Status** – returnează starea achiziţiei;
- **Actual Record Length** – returneză nr. de puncte/canal;
- **Configure Acquisition Record** – configurează subsistemul de achiziţie;
- **Configure Acquisition Type** – configurează modul de achiziţie;
- **Configure Channel** – configurează atributele canalului;
- **Configure Channel Characteristics** – configurează atributele electrice ale canalului;
- **Configure Edge Trigger Source** – configurează sursa de declanşare;
- **Configure Trigger** – configurează declanşare;
- **Configure Trigger Coupling** – configurează modul de cuplare al semnalului de declanşare;
- **GetChannelName** – returneză identificatorul de canal;
- **Fetch Waveform** – returnază forma de semnal ca arie de elemente;
- **Initiate Acquisition** – iniţiază execuţia achiziţiei;
- **Is Invalid Waveform Element** – testează elementele din aria returnată;
- **Read Waveform** – execută achiziţia şi returnează aria de elemente;
- **Sample Rate** – returnează rata de eşantionare;

Se prezintă în figura 3.19 modelul de funcţionare al osciloscopului pentru capabilităţile de bază:

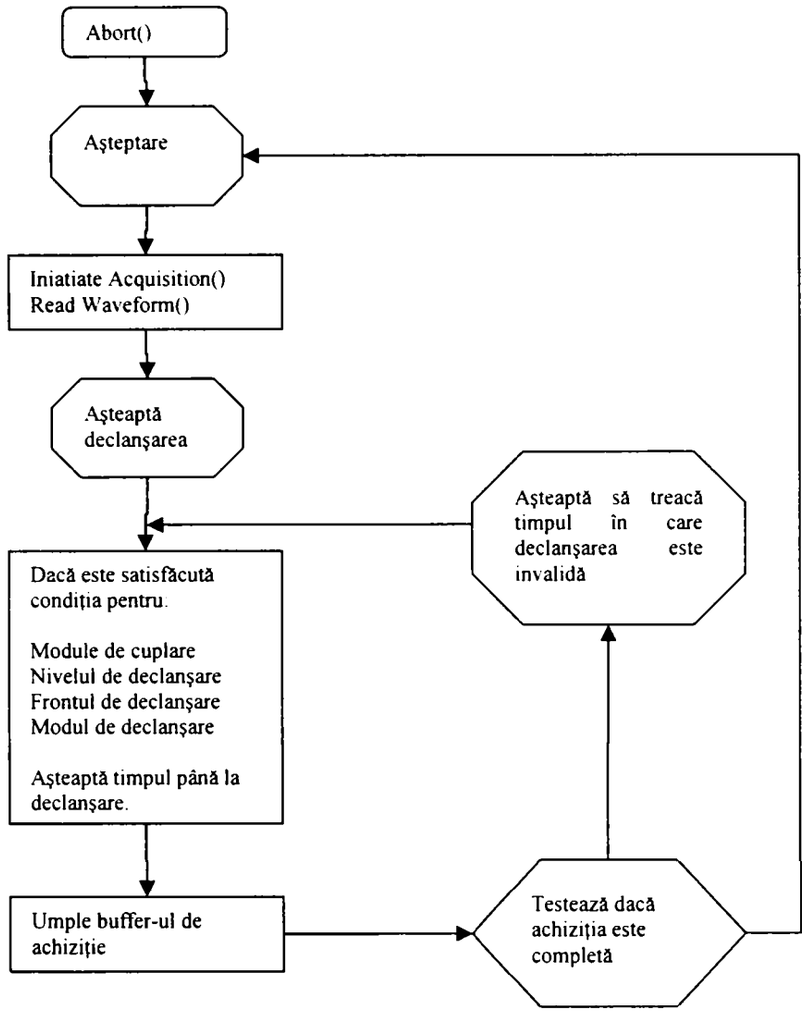


Fig. 3.19 Modelul de funcţionare al osciloscopului generic în mod declanşat.

3.4 Sistemul de inteconectare. Matricea de conexiuni.[18][32][105][125][8]

Matricea de conexiuni împreună cu sistemul de interconectare realizează legătura între instrumentele testorului, interfaţa de testare (TUA –Test Unit Adapter) şi unitatea testată (UUT – Unit Under Test). Dată fiind importanţa conexiunilor între resursele testorului şi unitatea testată, modul de conectare, ansamblul care face legătura între testor şi unitate a devenit obiectul unui proiect de standard numit "IEEE P1505 Receiver Fixture Interface System Standard". O remarcabilă prezentare a structurii unui echipament de testare automată este făcută în acest document, prezentare pe care autorul o va folosi ca ghid în analiza unor elemente din sistemul de conectare. Această prezentare (din figura 3.20) se numeşte "Aplicaţie tipică de sistem de interfaţare pentru testare" .

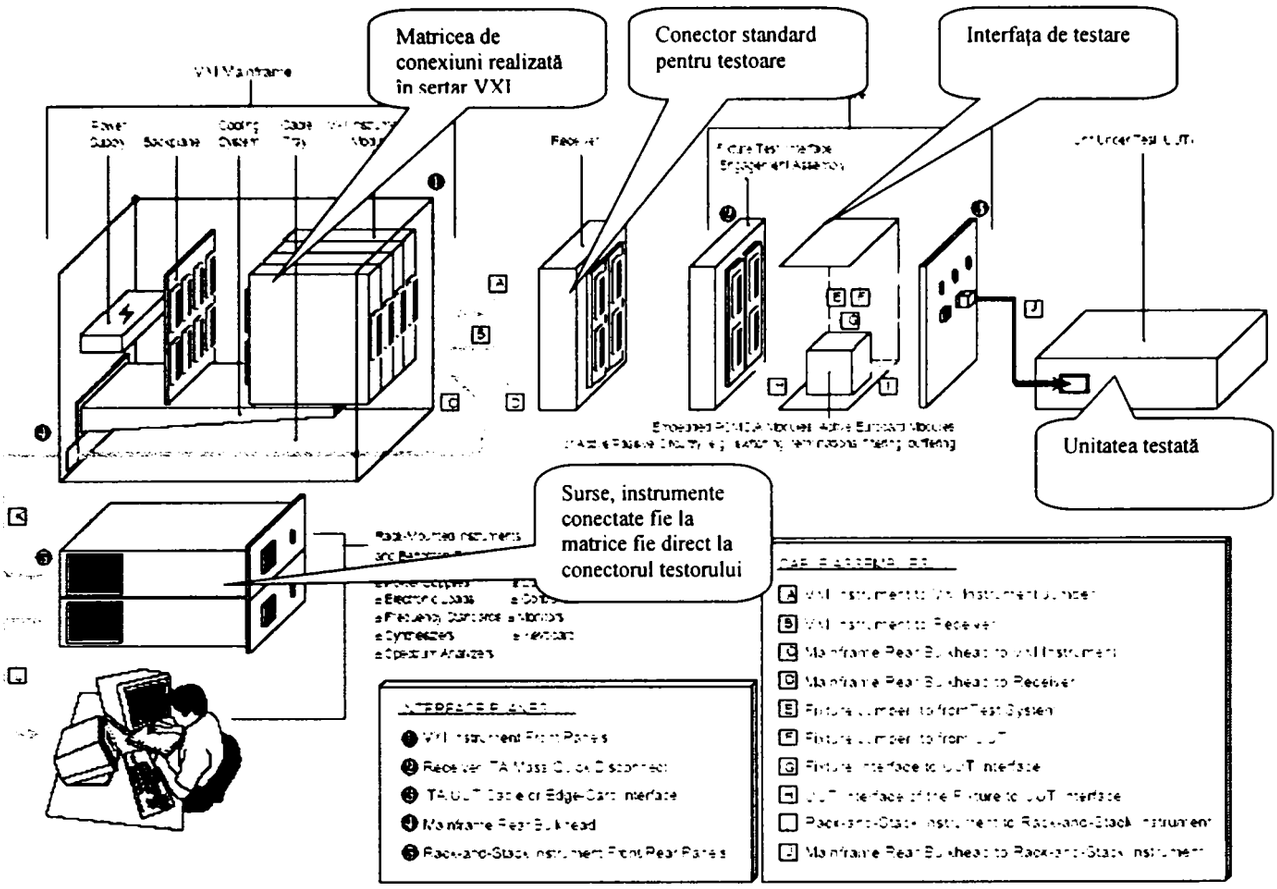
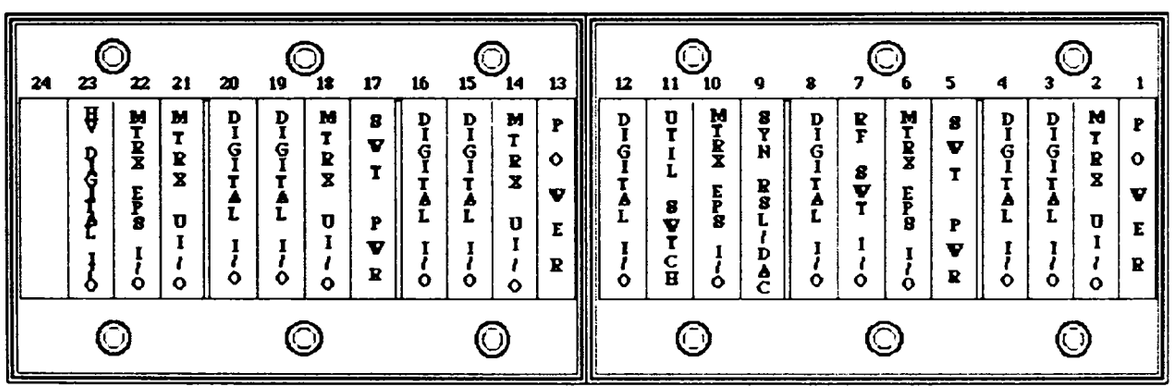


Fig. 3.20 Un sistem de testare automată conform IEEE P1505

Conectorul testorului este construit conform standardului ARINC 608 şi preluat şi de IEEE P1505 şi are 24 de conectori fiecare cu două secţiuni A şi B. Harta acestui conector a fost standardizată la rândul ei şi practic aceasta defineşte arhitectura testorului:



1 - Program DC Power 1-6, Sense 1-6 (16 High, 28 Low power pins)
2 - Matrix Universal I/O (96 low frequency signal pins)
3 - Digital I/O 1-64 Channels (128-192 signal pins)
4 - Digital I/O 65-128 Channels (128-192 signal pins)
5 - Switch DC Power, Sense (16 High, 28 Low power pins)
6 - Matrix I/O (24 low RF coax pins)
7- RF Switch I/O (12 high RF coax pins)
8 - Digital I/O 129-192 Channels (128-192 signal pins)
9 - Synchro Resolver Simulator/Indicator and Digital to Analog (DAC) Converters 1-12 (50-200 signal pins)
10 - Matrix I/O (24 low RF coax pins)
11 - Low Frequency Utility Switching (128-192 signal pins)
12 - Digital I/O 193-256 Channels (128-192 signal pins)
13 - Program DC Power 1-6, Sense 1-6 (16 High, 28 Low power pins)
14 - Matrix Universal I/O (96 low frequency signal pins)
15 - Digital I/O 257-320 Channels (128-192 signal pins)
16 - Digital I/O 321-384 Channels (128-192 signal pins)
17 - Switch DC Power, Sense (16 High, 28 Low power pins)
18 - Matrix Universal I/O (96 low frequency signal pins)
19 - Digital I/O 385 - 448 Channels (128-192 signal pins)
20 - Digital I/O 449-512 Channels (128-192 signal pins)
21 - Matrix Universal I/O (96 low frequency signal pins)
22 - Matrix I/O (24 low RF coax pins)
23 - High Voltage Digital I/O 1-64 Channels (128-192 signal pins)
24 - Digital I/O 513-536 Channels (128-192 signal pins)

Fig. 3.21 Conectorul ARINC 608

După cum se remarcă (din figura 3.21) acest conector practic leagă o parte din resurse direct iar o parte prin matricea de conexiuni la conectorul testorului. Sursele de alimentare precum şi o parte din stimuli sunt aduse direct la acest conector, dar de obicei unele dintre ele sunt trecute prin contacte de releu şi eventual circuite de protecţie. Cea mai mare parte din instrumente sunt conectate prin matricea de conexiuni. Aceasta alocare a conectorului este dedicată domeniului tehnologic al testării aparatelor electronice folosite pe aeronave.

3.4.1 Matricea de conexiuni

Matricea de conexiuni leagă resursele testorului de conectorul acestuia. Posibilităţile de interconectare sunt limitate. Tipurile de conexiuni pe care un testor le are de realizat pot fi clasificate astfel:

- conexiuni de putere (300 VDC, 28ADC, rezistenţa max. a contactului 10 mOhm).
- conexiuni de semnal (300 VDC, 2ADC, rezistenţa max. a contactului 10 mOhm, capacitatea mai mică de 10 pF, zgomot „crosstalk" mai mic de 60 dB la 10 MHz şi mai mic de 40 dB la 100MHz).
- conexiuni de semnal RF de joasă frcvenţă (banda până la 1,2 GHz).
- conexiuni de semnal RF de înaltă frecvenţă (bandă pâna la 26,5 GHz).

In general conexiunile de putere sunt mai simple şi nu sunt duse prin matrice, iar cele de RF sunt duse printr-o matrice specială mai mică. Matricea care realizează conexiunile de semnal este complexă cu un număr mare de interconexiuni posibile. Prezint în continuare arhitectura unui modul de matrice de conexiuni realizat de firma RADA Electronic Ind. Ltd[8].:

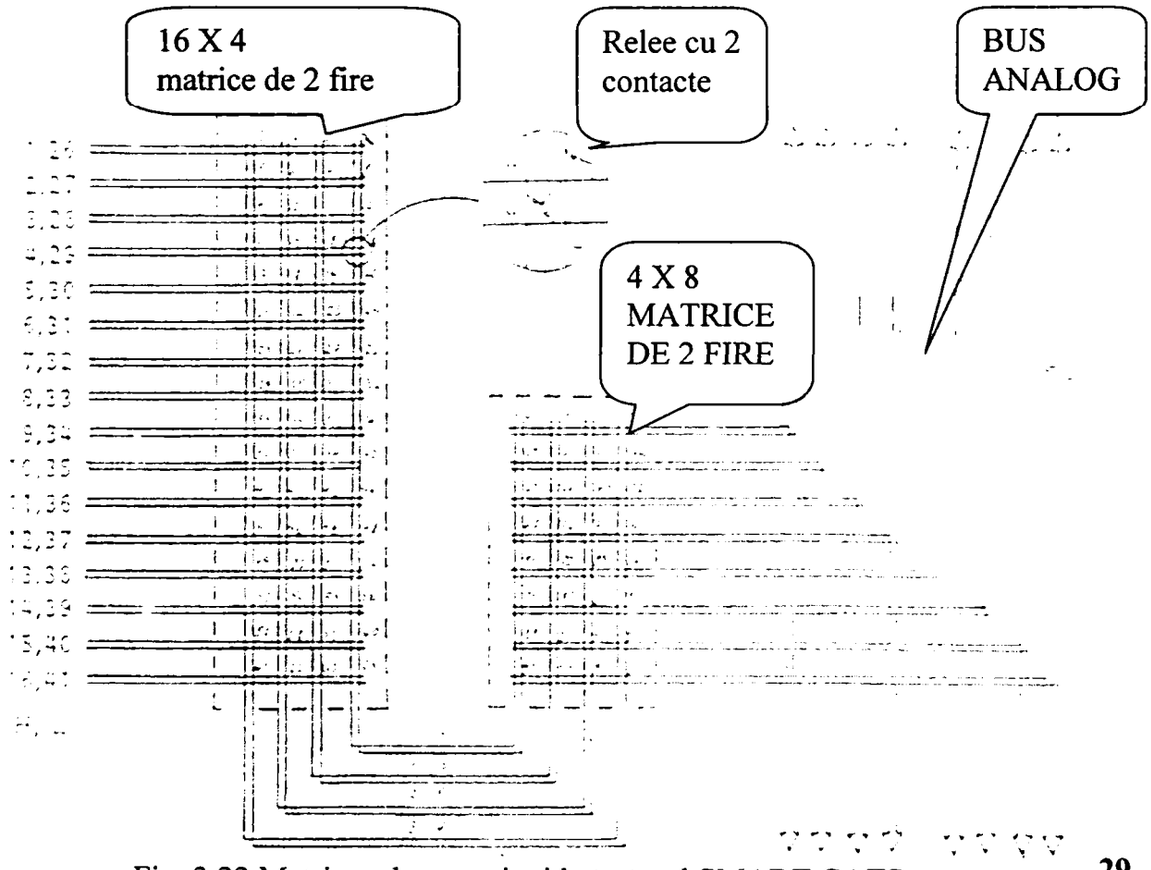


Fig. 3.22 Matricea de conexiuni la testorul SMART CATS

Acest gen de matrice este realizat cu 96 de relee pe o placă VXI. Matricea poate realiza de 2 ori 4 legături pe 2 fire sau de 2 ori 2 legături pe 4 fire. Pe bus-urile analoge se leagă instrumentele (senzori sau stimuli) acestea putând fi conectate fie pe partea A fie pe partea B a conectorului (figura 3.23).

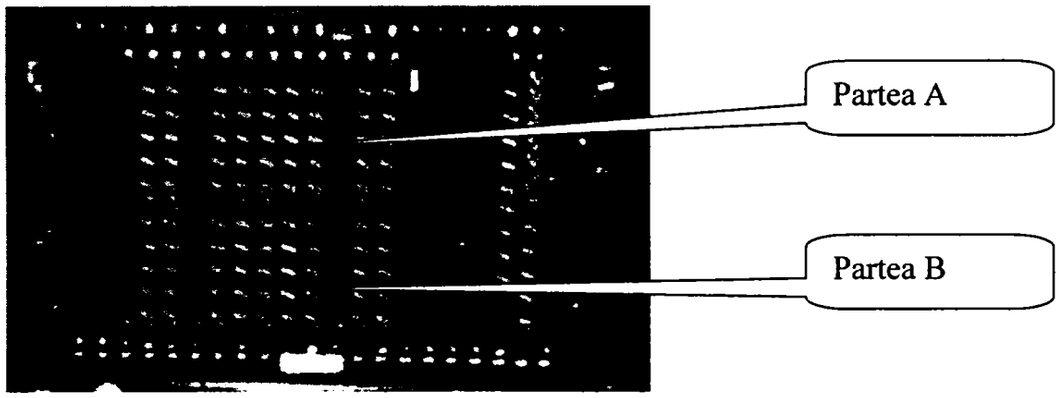


Fig. 3.23 Conectorul testorului SMART CATS

Racal Instruments construieşte plăcile VXI cu relee în diverse configuraţii cu care se pot construi matrici de conexiuni de dimensiuni mari. Astfel seria 1260A Opt 01T permite conectarea în serie a unui număr de 12 plăci pe care controlorul de bus VXI le vede ca pe un singur modul. In realitate există un modul master legat printr-un bus local (conexiune exterioară prin faţă) cu celelalte module. Un astfel de lanţ poate fi comandat fie prin registre, fie mai elegant in limbaj "SCPI". Sistemul include structura care descrie starea conform IEEE 488.2. Pe lângă comenzile comune există două comenzi importante: "CLOSE" şi "OPEN".

Aceste comenzi specifică adresa precisă a releului care trebuie închis sau deschis, precizând adresa modulului VXI (de la 1 la 12) şi adresa canalului (releului de pe placă). Există posibilitatea monitorizării contactului închis prin citirea unor contacte auxiliare în registre de control. Sistemul poate controla cu mare viteză mai multe relee în paralel sincron sau asincron. Plăcile pot fi matrici de relee sau grupuri de relee de putere sau chiar plăci de intrări-ieşiri digitale. Există posibilitatea unor execuţii în lanţ cu specificarea operaţiilor succesive. Astfel un releu poate fi deschis înainte ca următorul să fie închis (BBM Break Before Make) sau un releu poate fi deschis după ce următorul a fost închis (MBB Make Before Break).

Toate aceste operaţiuni sunt extrem de importante în timpul testării, fie pentru a descărca o capacitate parazită, fie pentru a proteja un circuit. In general astfel de operaţiuni sunt necesar să fie standardizate şi automatizate, un astfel de sertar fiind unul din elementele foarte scumpe dintr-un testor şi foarte expus accidentelor.

3.4.2 Capabilităţile de bază ale elementelor de conexiune (Switch) [32]

Clasa instrumentelor de conexiune defineşte acele dispozitive care pot realiza o conexiune între două puncte, indiferent dacă sunt necesare mai multe relee sau dacă nu sunt necesare toate releele avute la dispoziţie. Vom defini în continuare termenii utilizaţi în descrierea conexiunilor:

- **Canal**: un punct de conexiune la modulul de conexiuni pe care utilizatorul îl poate accesa. Un canal nu specifică un număr de fire, el putând consta în 1,2,3,4 fire spre exemplu;
- **Pereche de canale**: două nume de canale legate prin simbolul "->";
- **Comun**: numele canalului de ieşire la un multiplexor;
- **Canal de configurare**: este un canal care nu este accesibil direct dar poate fi folosit pentru crearea unei legături între două canale;
- **Modul de matrice de conexiuni**: un modul de conexiuni cu intrări şi ieşiri multiple organizat sub forma unei matrici în care fiecare linie poate fi conectată la oricare coloană;
- **Modul de multiplexare conexiuni**: un modul de conexiuni cu multiple intrări dar cu o singură ieşire;
- **Cale**: este legătura între două canale. De remarcat că modulul de conexiuni trebuie să ştie care căi sunt valide, invalide sau folosite;
- **Modul de conexiuni de tip scanner**: are capabilitatea de a scana canalele;
- Canal sursă: un canal direct accesibil de către utilizator pentru o conexiune externă;
- **Modul de conexiune**: un dispozitiv controlat de utilizator;

Atributele clasei de conexiuni pentru grupul de bază sunt următoarele:

- **AC Current Carry Max** – curentul (AC) maxim pe care îl duce canalul;
- **AC Current Switching Max** – curentul(AC) maxim care poate fi comutat;
- **AC Power Carry Max** – Puterea (AC) maximă admisă prin canal;
- **AC Power Switching Max** – Puterea (AC) maximă comutată pe canal;
- **AC Voltage Max** – Tensiunea(AC) maximă admisă pe canal;
- **Bandwith** – Banda de frecvenţă transferată de canal cu atenuare mai mică de 3 dB;
- **Channel Count** – returnează numărul canalelor disponibile;
- **Channel Item** – returnează un pointer la proprietţile canalului pe baza numelui;
- **Channel Name** – returnează numele pe baza unui index;
- **Characteristic Impedance** – returnează impedanţa canalului;
- **DC Current Carry Max**– curentul (DC) maxim pe care îl duce canalul;
- **DC Current Switching Max** – curentul(DC) maxim care poate fi comutat;

- **DC Power Carry Max** – Puterea (DC) maximă admisă prin canal;
- **DC Power Switching Max** – Puterea (CC) maximă comutată pe cana;

- **DC Voltage Max** – Tensiunea(DC) maximă admisă pe canal;
- **Is configuration Channel** – specifică utilizarea canalului pentru crearea de căi de conexiune, considerat legătură internă;
- **Is debounced** – raportează stabilizarea conexiunii după comandă şi terminarea tuturor oscilaţiilor contactelor;
- **Is Source Channel** – canalul declarat canal sursă;
- **Settling Time** – timpul după care semnalul prin canalul respectiv este considerat stabil din punct de vedere al conexiunii;
- **Wire Mode** – descrie numărul de fire pentru canalul respectiv;

Funcţiile cu care aceste atribute pot fi modificate sunt următoarele:

- **Can Connect** - verifică posibilitatea creerii căii de conexiuni;
- **Connect** - realizează o cale de conexiuni între două canale;
- **Disconnect** - desface o cale de conexiuni între două canale;
- **Disconnect All** - desface toate căile connectate la acel moment;
- **Get Channel Name** - returnează numele canalului pe baza unui index;
- **Get Path** - returnează o listă a perechilor de canale utilizate pentru realizarea unei cai de conexiune între două canale;
- **Is Debounced** - confirmă terminarea vibraţiilor pe calea respectivă şi ca semnalul pe acea cale este stabil conectat;
- **Set Path** - permite utilizatorului să seteze căile de configurare pentru a realiza o anumită cale de conexini. Uzual acest lucru este realizat automat de modulul de conexiuni;
- **Wait For Debounce** - setează timpul de stabilizare a semnalului, iar dacă acesta este depăşit returnează eroare;

3.5 Testarea comunicaţiei şi instrumente pentru testarea dispozitivelor de comunicaţie.[21][8][100][110]

Numeroasele dispozitive electronice care se găsesc în alcătuirea unei aeronave comunică între ele folosind câteva tipuri de comunicaţie specifice acestei industrii. In figura 3.24 sunt prezentate dispozitivele conectate pe bus-urile Arinc 629 şi reţeaua LAN pe avionul Boing 777. Se remarcă existenţa a două bus-uri A629, unul pentru controlul zborului şi unul pentru interconectarea sistemelor şi transferul datelor. Al doilea poate fi folosit şi pentru reîncărcarea de noi versiuni de soft operaţional în timpul operaţiunilor de întreţinere la sol.

Pentru noile arhitecturi de aeronave comunicaţia este esenţială. Standardul Arinc 629 este un standard nou, foarte eficient în utizarea şi alocarea canalului de comunicaţie.

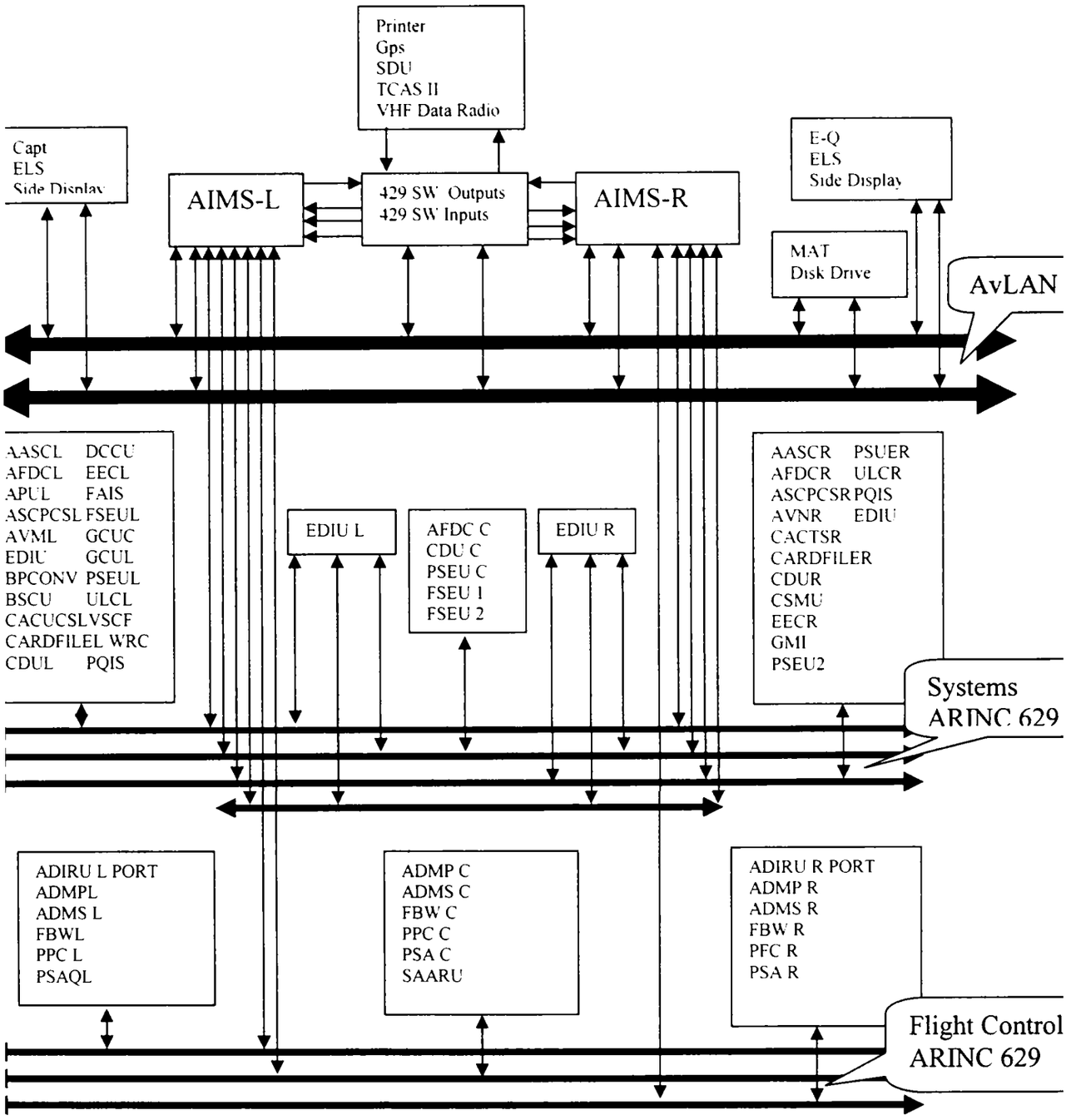


Fig. 3.24 Schema cu unitățile conectate la bus-urile 629 pe avionul Boing 777

Majoritatea dispozitivelor testate au prin construcție circuite destinate comunicației, testarea acestor funcții de comunicație fiind absolut necesară, existența unor instrumente specializate în arhitectura testorului şi cunoaşterea acestor tipuri de comunicație este de asemenea necesară. Deoarece standardele ARINC 429 şi ARINC 629 sunt implementate aproape pe toate unitățile, se prezintă aceste standarde precum şi instrumentele folosite pentru testare.

3.5.1 ARINC 629[21][110]

Arinc 629 este un bus serial digital dezvoltat de compania Boeing pentru transferul datelor şi controlul informaţiei în interiorul avionului. Un bus A629 poate fi folosit de 120 de transmiţătoare fără nici un controlor de bus. Fiecare dispozitiv se interfaţează cu aces bus printr-un "terminal controller", un modul de interfaţă serială (SIM) şi un cuplor de current. Bus-ul propriu-zis este o pereche de fire torsadate cu 130 Ohm la fiecare capăt (terminatori). Terminal controller-ul este programat cu un ROM de personalitate. Un dispozitiv va împarţi o zonă de memorie cu terminal controller-ul.

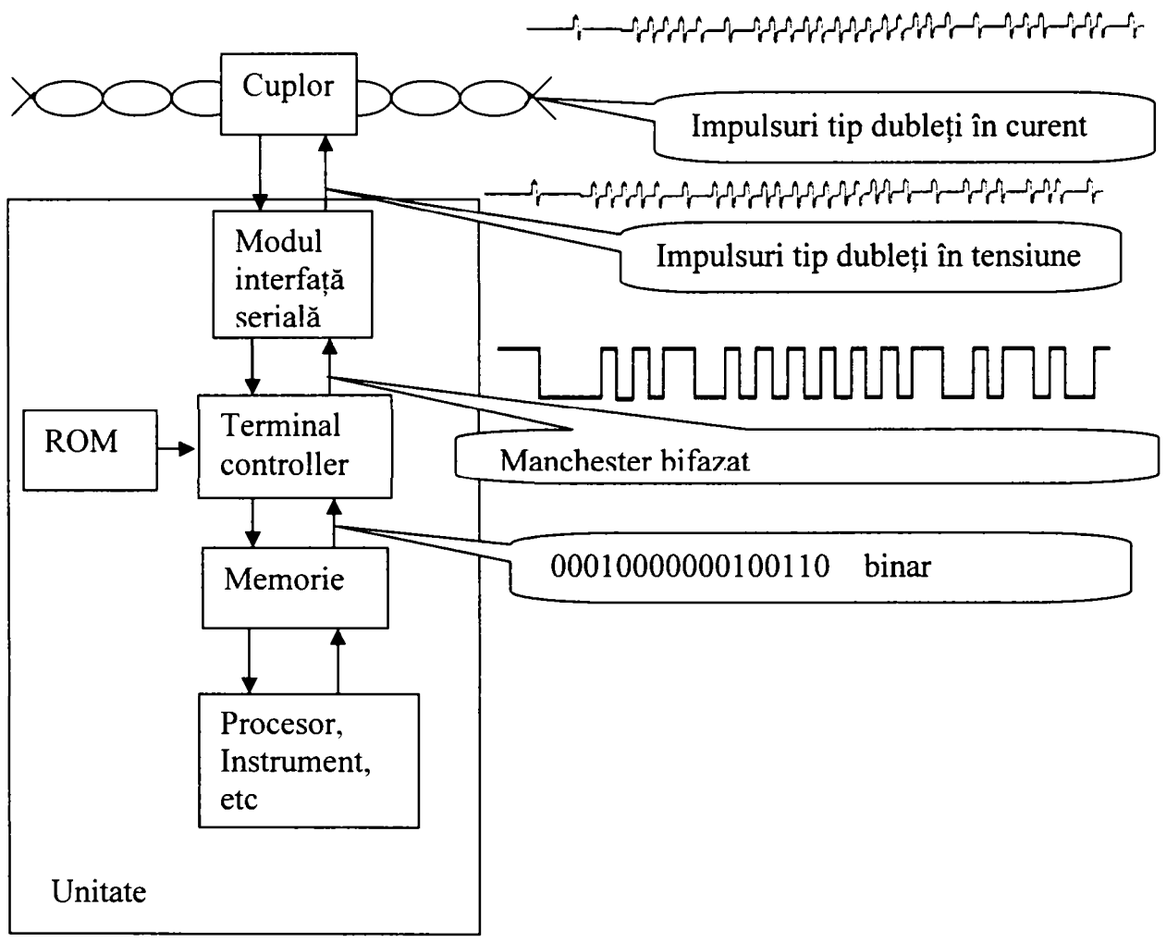


Fig. 3.25 Schema circuitului de conectare la bus-ul ARINC 629 a unei unităţi

Fiecare terminal monitorizează activitatea pe bus şi menţine 3 „timere" pentru a evita coliziunile, a permite acces egal tuturor unităţilor, şi pentru o utilizare maximă a benzii fără un controlor de bus.

Un terminal trimite un mesaj odată. Un mesaj este alcătuit din 31 de şiruri de cuvinte. Un şir poate fi compus din maxim 257 de cuvinte. Un cuvânt are 3 biţi de sincronizare, 16 biţi de date şi un bit de paritate. Fiecare şir are în faţă o etchetă de

identificare din 16 biţi din care 12 identifică şirul şi 4 biţi mai semnificativi identifică canalul. Între două şiruri consecutive din acelaşi mesaj există un spaţiu de timp egal cu 4 biţi.

Fiecare controler de terminal are trei „timere":
- Interval de transmisie (TI);
- Intervale între terminale (Terminal Gap TG);
- Interval de sincronizare (Syinchronization Gap SG);

Toate terminalele folosesc acelaşi TI. Dacă bus-ul nu este supraîncărcat fiecare terminal va transmite odata în intervalul TI. Fiecare terminal va avea un TG diferit eliminând astfel posibilitatea coliziunii. Un terminal transmite dacă după TG-ul său bus-ul este liber. SG garantează accesul fiecărui terminal la bus. SG este mai lung decât cel mai mare TG. SG este un interval după transmisie în care terminalul respectiv nu poate să transmită.

3.5.2 Modulul de comunicaţie VX4469A

Modulul de comunicaţie ARINC 629 de tip VX4469A produs de Tekronix este o placă VXI bazată pe procesorul Intel 80186 şi care este programată in mod registru. Nu s-au definit capabilităţi pentru acest instrument. *Autorul a construit un driver sub mediul PAWS pentru această placă* . Se prezintă schema bloc a cestei plăci în figura 3.26.

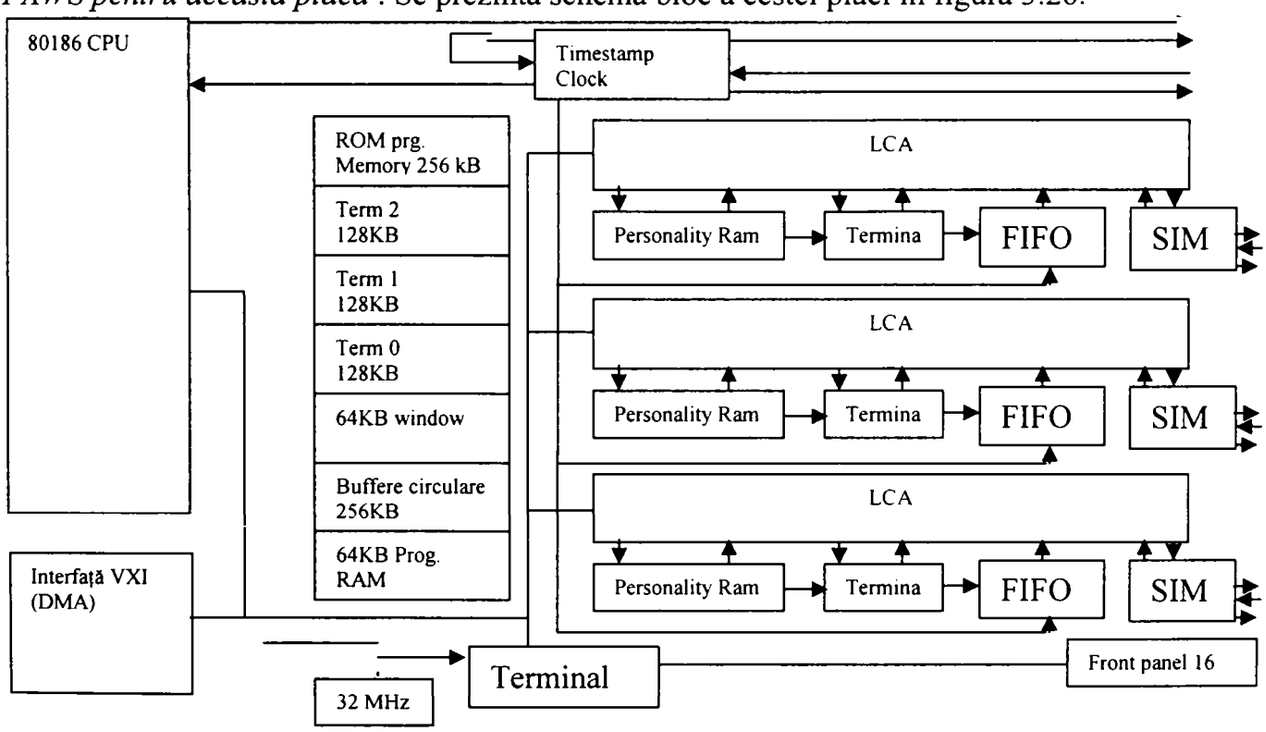


Fig. 3.26 Schema bloc a plăcii VX4469A

3.5.3 ARINC 429[17]

ARINC 429 numit şi "Mark 33 Digital Information Transfer System (DITS)" este cel mai răspândit sistem de comunicaţie, între dispozitivele electronice de pe avioane, folosit în acest moment. Acest standard descrie un sistem în care un dispozitiv electronic care are de transmis o informaţie, o face de la un port desemnat, printr-o pereche de fire torsadate şi ecranate, către toate sistemele care au nevoie de informaţia respectivă. Nu este permisă transmisia bidirecţională. Semnalul transmis este un semnal diferenţial.

Modulul TVXI/429 (placă VXI) produs de TASCO are schema bloc:

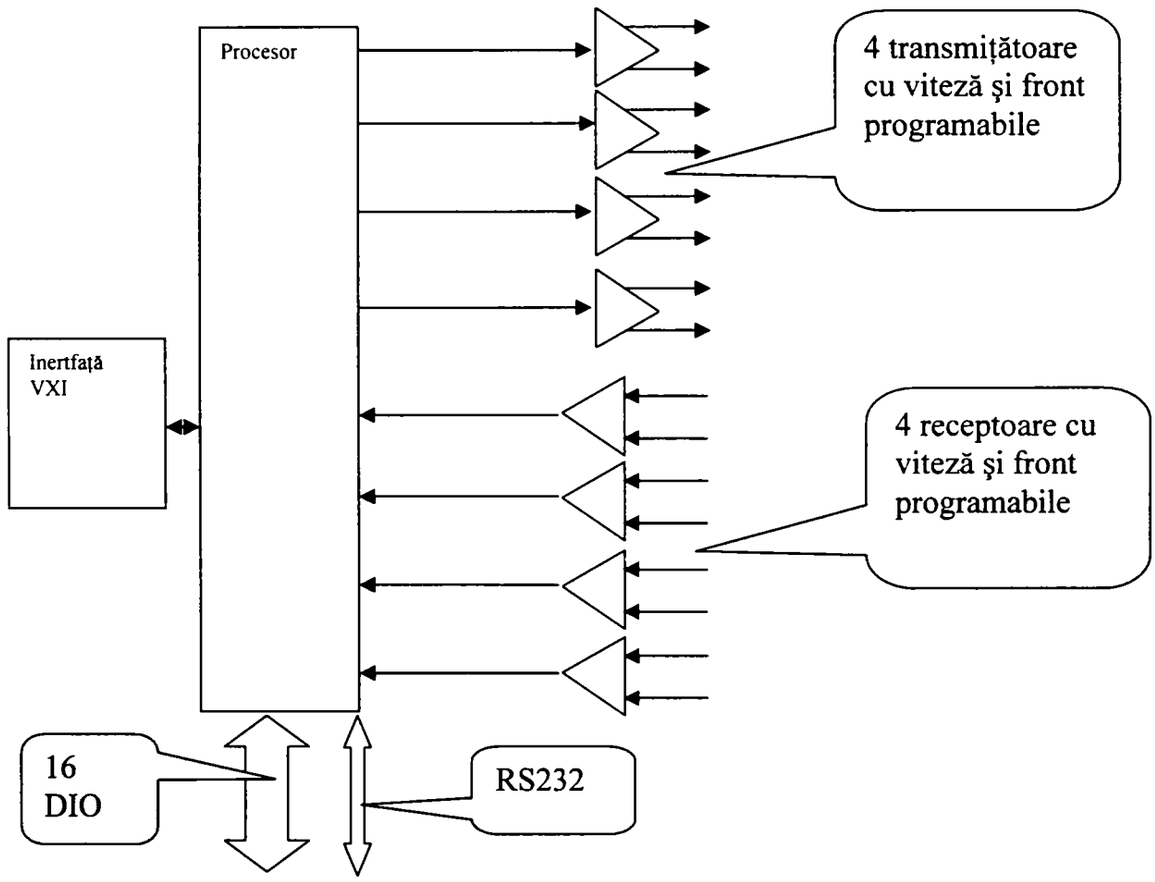


Fig. 3.27 Schema bloc a plăcii TVXI/429

Protocolul Arinc 429 operează cu cuvinte de 32 de biţi construite ca în figura de mai jos:

32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	1
P	SSM		Data																	Pad		SDI		LABEL	

P = paritate
SSM = "Sign / Status Matrix" specifică modul de operare

DATA PAD = datwele propriu-zise
SDI = Soursă / Identificare Destinaţie (dacă sunt mai multe destinaţii sau surse posibile)
LABEL = tipul de date (acestea sunt foarte precise în standardul A429)

Din punct de vedere electric cuvintele sunt trimise în mod diferenţial pe două fire faţă de masă:

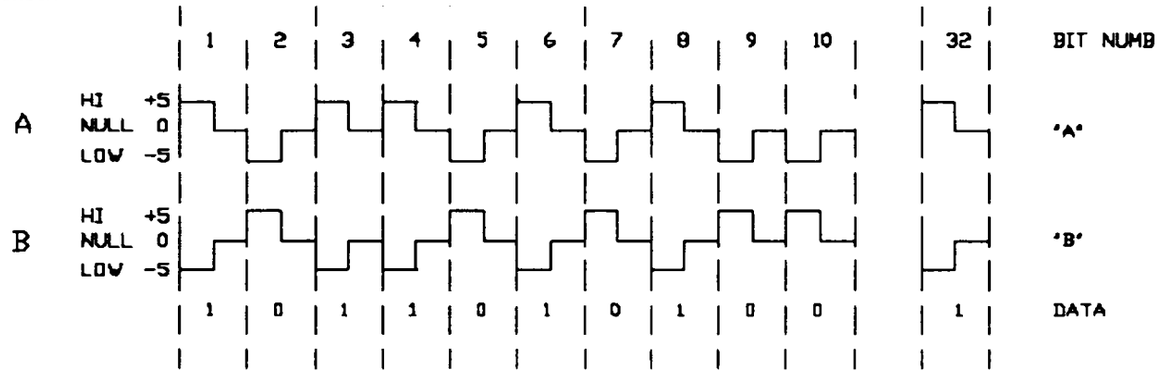


Fig. 3.28 diagramele de timp ale semnalelor pe busul Arinc 429

După cum se observă între A şi B avem +10V pentru 1 logic şi +10V pentru 0 logic. Semnalul este de tipul RZ (Return to Zero). In detaliu acesta arată astfel:

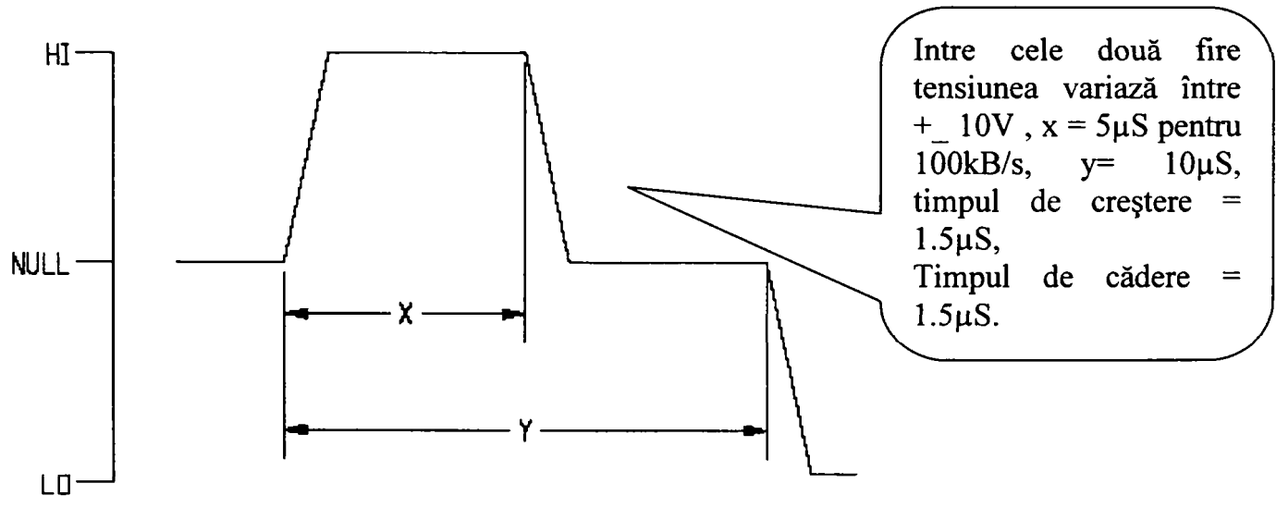


Fig. 3.29 Amplitudinea şi duratele semnalului pe bus-ul ARINC 429

3.6 Surse de alimentare.[31][158][202]

Sursele de alimentare pot fi AC sau DC. Pentru avionică se folosesc alimentări de 28 VDC sau de 115VAC şi 400 Hz. Aceste instrumente se folosesc atât pentru alimentarea propriu-zisă cât şi pentru generarea unor stimuli.

Foarte des folosite în construcţia testoarelor sunt sursele AC produse de California Instruments de tip 503T[X]. Acestea sunt surse trifazate (3 faze şi nul) care acoperă frecvenţe între 5 Hz şi 10 KHz cu puteri de maxim 170 VA pe fază. Sursa este comandată GPIB, are limbajul SCPI implementat şi poate controla separat pe fiecare fază amplitudinea, faza, frecvenţa şi curentul. De asemenea sursa poate genera rampe de tensiune sau de frecvenţă. Nu există definit un instrument generic pentru sursele de curent alternativ şi nici capabilităţi de bază.

Producătorii de surse DC sunt mai numeroşi, sursele de tip HP603xA produse Hewlett Packard sunt printre cele mai folosite. Valorile electrice acoperite de aceste surse sunt variate în funcţie de tipul ales. Aceste surse pot fi controlate GPIB şi pot îndeplini următoarele funcţii:

- programarea tensiunii;
- programarea curentului;
- posibilitatea de declanşare;
- controlul prezenţei tensiunii la ieşire;
- limitarea curentului şi atensiunii;
- raportarea stării;
- protecţie tip "foldback";
- măsurarea tensiunii;
- măsurarea curentului;
- self test;

3.6.1 Capabilităţi de bază pentru sursele DC[31].

O sursă DC este considerată o sursă de semnal DC. Definim în continuare câţiva termeni uzuali în descrierea surselor DC:

- **Nivel de tensiune** - tensiunea pe care sursa încearcă să o genereze;
- **OVP** - protecţie la supratensiune (Over Voltage Protection). Dacă ac easta este validată, sursa opreşte generarea de tensiune la ieşire;
- **Current Limit** - limită de curent. Există două situaţii posibile: fie sursa opreşte generarea de tensiune, fie trece în regim de sursă de curent;
- **Mod de tensiune constantă** - sursa reglează curentul şi menţine tensiunea constantă;
- **Mod de curent constant** - reglează tensiunea şi menţine curentul constant;
- **Mod neregulat** - când sursa nu reuşeşte să regleze la valorile programate nici curentul nici tensiunea;

Diagrama tensiune curent pentru o sursă DC este următoarea:

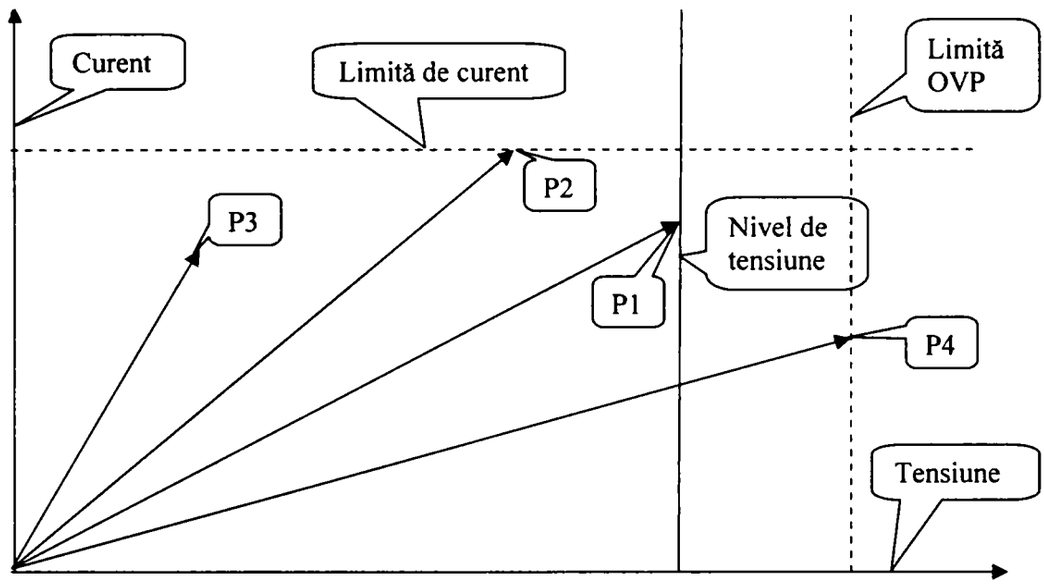


Fig. 3.30 Diagrama tesiune – curent pentru sursele DC

Punctul P1 - sursa se află în mod de tensiune constantă;
Punctul P2 - sursa se află în mod de curent constant;
Punctul P3 - sursa se află în mod neregulat;
Punctul P4 - sursa a atins limita de protecţie la supra tensiune;

Atributele definite pentru grupul de bază sunt următoarele:

- Limita de curent;
- Comportarea la limita de curent;
- Validare ieşire;
- Validare OVP;
- Nivel de tensiune;
- Numărul canalului de ieşire;
- Numele canalului;

Funcţiile cu care se operează asupra acestor atribute sunt următoarele:

- Configure Current Limit;
- Configure Output Enabled;
- Configure Output Range;
- Configure OVP;
- Configure Voltage Level;

- Get OutputChannelName;
- Query Current Limit Max;
- Query Voltage Level Max;
- Query Output State;
- Reset Output Protection;

3.7 Interfaţa de testare (TUA)[55-80][18]

Adaptorul de testare este un subansamblu care face legătura între conectorul standard al testorului şi conectorul unităţii testate, dar cuprinde şi resursele suplimentare necesare pentru testare care nu există în structura testorului.

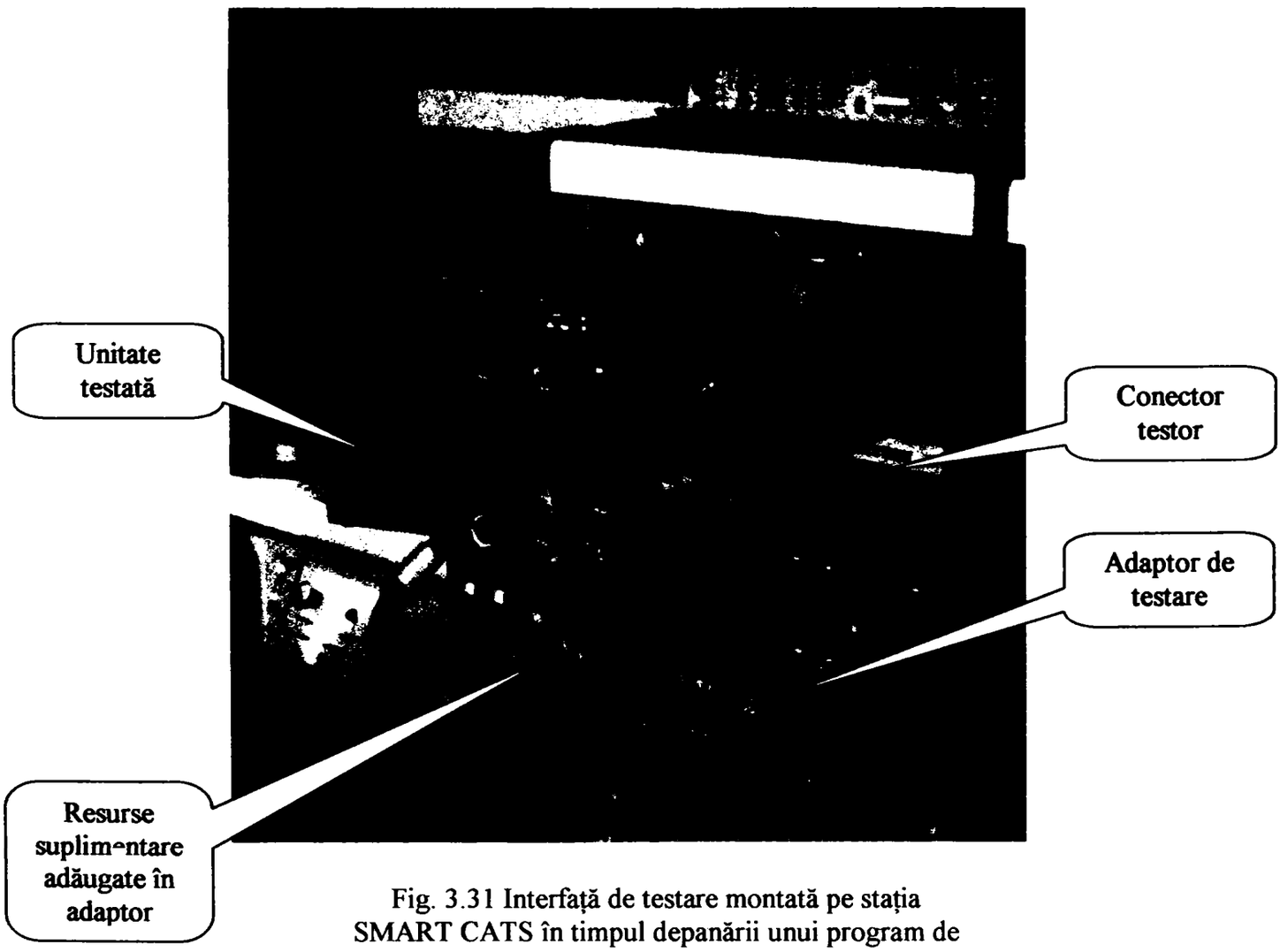


Fig. 3.31 Interfaţă de testare montată pe staţia SMART CATS în timpul depanării unui program de testare

Adaptorul de testare se construieşte respectând câteva reguli:

- Acesta are întodeauna un mod de identificare astfel încât eroare de conectare a unui adaptor cu o unitate nepotrivită să nu distrugă unitatea;
- Adaptorul are un program de testare propriu şi în consecință este proiectat cu posibilitatea de măsurare a resurselor pe care le conține;

3.7.1 Interfața de testare pentru unitatea CSCP (Cabin System Control Panel) [40]

CSCP este o unitatea principală din sistemul de management al cabinei. Aceasta este o consolă (tip touch screen) folosită de personalul de însoțire pentru diverse comenzi cum ar fi aprinderea luminilor, monitorizare şi control, aflarea adresei pasagerilor, etc. Tot această unitate este folosită de personalul de întreținere pentru încărcarea softului operațional şi a bazelor de date, realizând testarea şi diagnosticarea sistemului precum şi funcții generale de întreținere şi reconfigurare. CSCP este interfațat cu unități precum CSMU şi EMC şi cu serverul de date pentru cabină. CSCP conține două sisteme cu procesorul INTEL 80486, unul operând sub UNIX şi unul sub Windows, fiind legat cu două conexiuni Ethernet cu rețeau şi conținând conexiuni audio, video, etc. Se prezintă câteva circuite folosite pentru testarea aceastei unitați, care sunt uzuale:

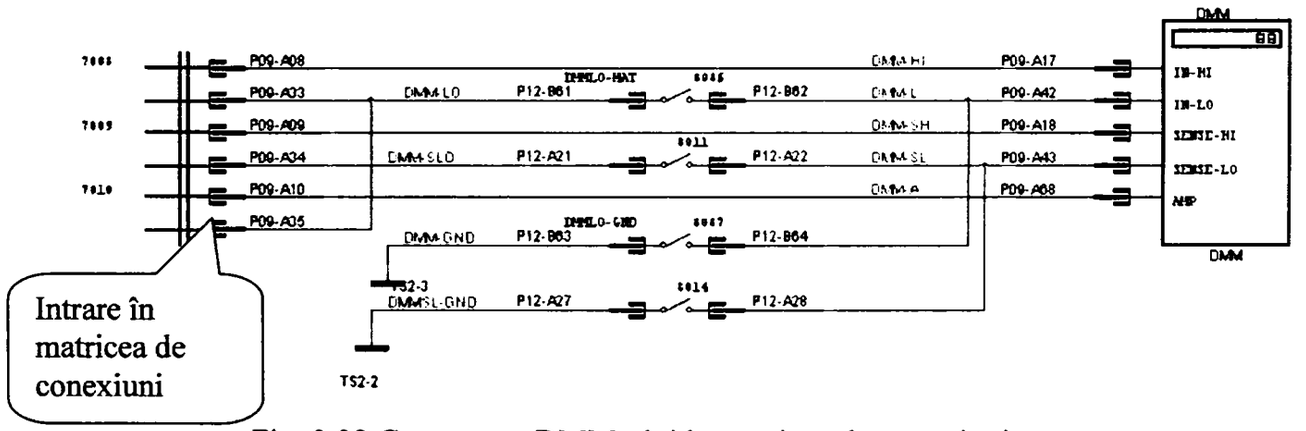


Fig. 3.32 Conectarea DMM-ului la matricea de conexiuni.

Modul de legare al DMM-ului la matricea de conexiuni permite diverse tipuri de măsurători (pe patru fire, pe două fire, scanare cu un fir legat la masă).

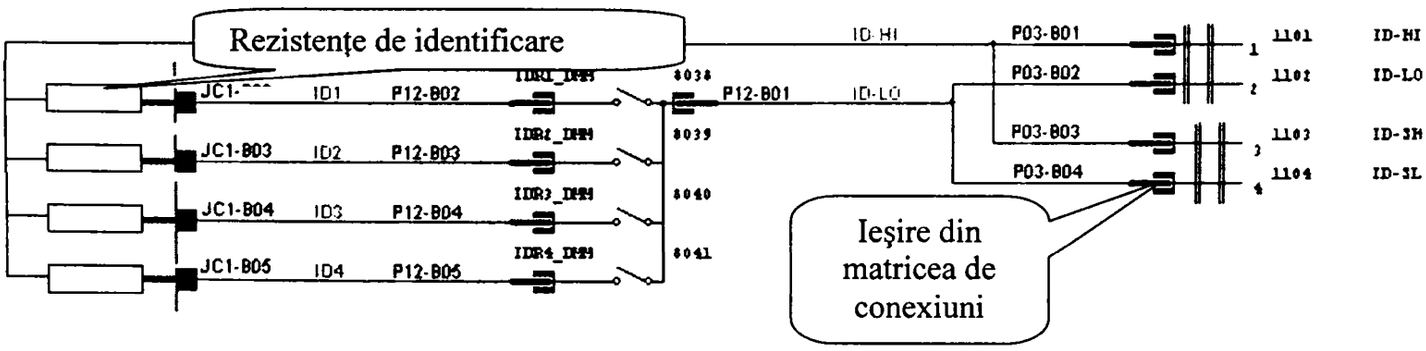


Fig. 3.33 Conectarea rezistențelor de identificare ale interfeței la matricea de conexiuni.

Schema de identificare permite măsurarea cu DMM-ul pe 4 fire a patru rezistenţe care sunt unice pentru unitatea respectivă. Cele 4 rezistenţe sunt plasate în adaptor. O situaţie mai dificilă se întâlneşte la verificarea unităţilor, care nu au fost construite pe acest principiu şi inginerul care proiectează testul imaginează un test de verificare măsurând rezistenţa unor circuite la rece, test care se efectuează înainte de alimentarea unităţii.

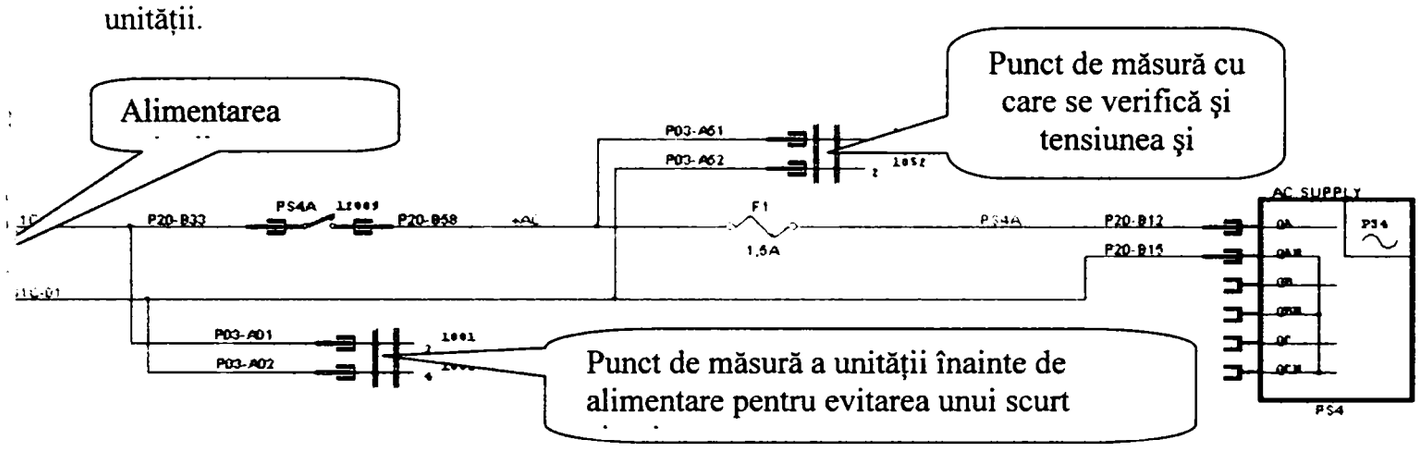


Fig. 3.34 Alimentarea unităţii testate cu tensiune AC.

Alimentarea unităţii cu tensiune alternativă de 115 V şi 400 de Hz se face de la una din fazele unei surse de tip California Instruments de 500 VA.

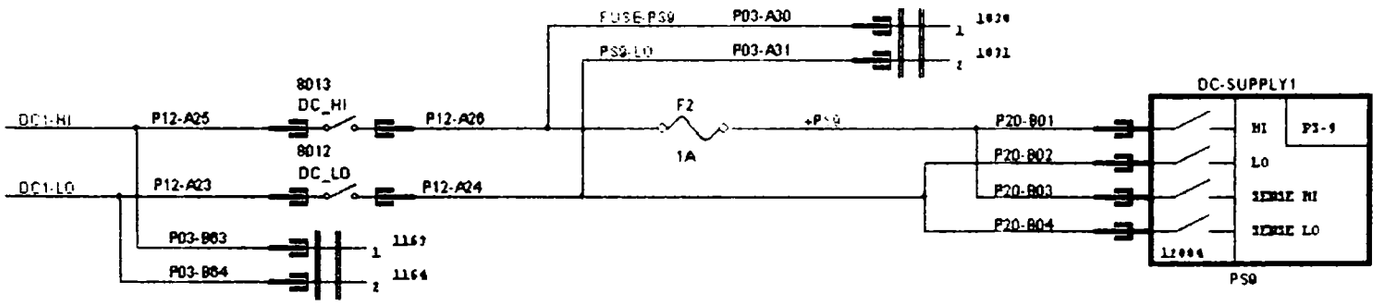


Fig. 3.35 Alimntarea unităţii testate cu tensiune DC.

Alimentarea cu curent continuu este similară cu cea de curent alternativ.

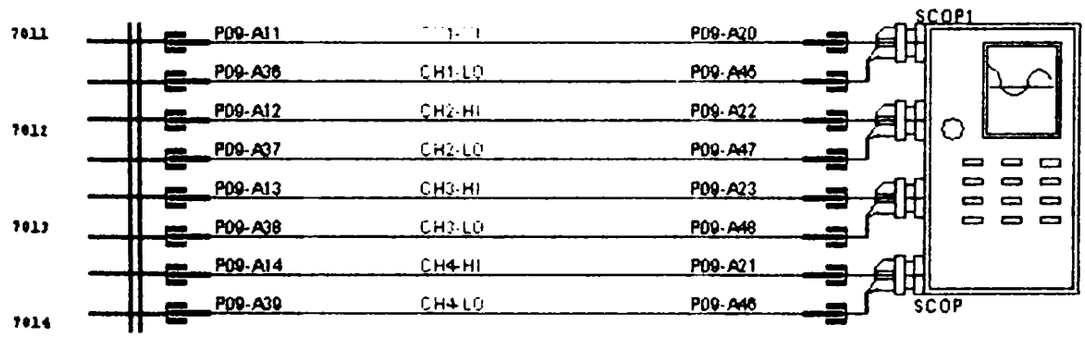


Fig. 3.36 Conectarea osciloscopului cu 4 canale la matricea de conexiuni.

Figura 3.36 prezintă conectarea osciloscopului cu toate cele 4 canale la matricea de conexiuni. După cum se remarcă acesta ocupă 4 canale analoge în varianta în care 2 canale sunt conectate simultan, altfel ocupând chiar mai multe. Cum de multe ori sunt necesare şi alte resurse pe matrice când se măsoară cu osciloscopul testele care folosesc toate canalele sunt dificil de administrat.

3.8 Concluzii

Echipamentele de testare automată folosite pentru testarea dispozitivelor electronice din aviaţie sunt supuse unui intens proces de standardizare pe diversele paliere care alcătuiesc aceste sisteme complexe.

Gama instrumentelor virtuale interschimbabile (IVI) se diversifică odată cu diversificarea numărului de dispozitive. Această diversificare este primul pas spre necesitatea realizării instrumentelor sintetizate. Instrumentele sintetizate sunt instrumente generice realizate prin intermediul unor instrumente simple ale căror operaţii vor fi integrate de către driver-ele IVI.

Tipurile de instrumente folosite se grupează pe categorii generice care devin clase de instrumente şi care permit realizarea unor programe de testare independente de suportul hardware folosit.

Matricea de conectare devine tot mai complexă permiţând tot mai multe conexiuni simultane.

Prin încercarea diverselor arhitecturi de testare s-a constat necesitatea interschimbabilităţii instrumentelor fără modificarea programelor de testare precum şi posibilitatea extinderii resurselor testorului cu instrumente noi, acest lucru semnificând crearea unor arhitecturi deschise.

În secţiunea 3.1 se prezintă o *structură de echipament de testare automată, utilizată practic de autor*, care subliniază importanţa magistralelor de comunicaţie cu resursele staţiei în asigurarea dezvoltării ulterioare cu noi resurse precum şi modificarea acestora în sensul evoluţiei către forme mai complexe sau elemente de sinteză. Toate aceste modificări conservă programele de testare şi asigură legătura între generaţiile de echipament.

În secţiunea 3.1 se prezintă şi *un testor proiectat şi realizat de către autor* pentru testarea unui înregistrator video pentru elicoptere.

În secţiunea 3.2 se trec în revistă magistralele de comunicaţie cu instrumentele, accentuându-se standardele IEEE 488, VXI, PXI care sunt cele mai folosite în present şi pe care autorul le-a folosit în diversele programe de testare dezvoltate.

În secţiunea 3.3 se prezintă câteva instrumente uzuale cum ar fi DMM-ul (multimetrul digital), AFG-ul (generatorul de funcţii arbitrare), DSO-ul (osciloscopul digital) în dublă ipostază: ca instrumente specifice ale unor producători conscraţi dar şi ca instrumente generice cu atributele şi funcţiile de modificare a acestora din grupul capabilităţilor de bază respective. Se prezintă interfaţa grafică a unui *instrument virtual de tip DMM generic realizat de autor*, cu comandă de la distanţă.

În secţiunea 3.4 se prezintă matricea de conxiuni, element fundamental de legătură între mulţimea senzorilor şi stimulilor testorului şi mulţimea punctelor de măsură şi injecţie de stimuli ai unităţii, *matrice prezentată în varianta practică utilizată de autor*

precum şi varianta generică cu capabilităţile de bază la care s-au adagat elementele de standardizare din standardul ARINC 608 şi proiectul IEEE P1505.

Secţiunea 3.5 prezintă testarea unor circuite specifice în avionică şi anume circuitele de comunicaţie între unităţi. Se pune accent pe standardele ARINC 629 şi ARINC 429 cu prezentarea propriu-zisă a standardelor respective (extrem de sintetic şi cu accent pe partea de formă a semnalelor, partea de protocol şi de convenţii fiind extrem de laborioasă) precum şi pe prezentarea unor păci tip VXI specifice pe care autorul le-a folosit şi la care a dezvoltat drivere. Autorul *a participat la dezvoltarea unui driver pentru bus-ul ARINC 629.*

Secţiunea 3.6 prezintă sursele de current alternativ şi continuu ca un grup distinct în rândul instrumentelor cu sublinierea standardizării surselor de current continuu ca instrumente generice.

Secţiunea 3.7 prezintă câteva scheme de conectare a instrumentelor uzuale prin intermediul adaptorului de testare la unitatea testată şi câteva reguli practice de realizare, folosite la proiectarea interfeţelor de testare de către autor.

CAP. IV
Structura programelor de testare funcţională. Exemple specifice pentru avionică[22][18][7][55]-[70][93][98][111][125]

În acest capitol se prezintă procesul de testare funcţională în avionică: mod de proiectare, structură, utilitatea acestuia.

Testarea funcţională a unei unităţi electronice de aviaţie este un proces de verificare succesivă a funcţiilor unităţii respective prin intermediul conectorului exterior al acesteia.

Programul de testare are ca sursă de informaţie diverse documente generate în procesul de proiectare, fabricaţie şi utilizare.

Structura programelor de testare prezentată în diagrama de testare implementează mai multe grupe de teste permiţând teste complete, parţiale sau diagnosticarea.

Rezultatul testării complete este raportul de testare, un fişier cu rezultatele tuturor testelor care trebuiesc parcurse de unitate pentru a rămâne în serviciu.

4.1 Testarea funcţională în avionică.

Dispozitivele electronice folosite de o aeronavă sunt verificate din punct de vedere funcţional periodic pentru a putea fi utilizate. Testarea funcţională este importantă pentru avionică, acest domeniu tehnologic fiind critic din punct de vedere al securităţii pasagerilor. Din acest motiv testele funcţionale se fac periodic pentru a verifica dacă o unitate este capabilă de serviciu. Testele funcţionale permit estimarea cu precizie a gradului de funcţionare a unui dispozitiv datorită modului de proiectare a testelor, ca proceduri independente care caracterizează o zonă bine delimitată funcţional şi constructiv a unităţii testate. Un test funcţional furnizează informaţii şi despre circuitele defecte. În regim de diagnostic un test funcţional poate indica componenta defectă.

Proiectarea programului de testare funcţională începe odată cu proiectarea avionului, continuă cu proiectarea unităţii este validat şi apoi folosit pe toată perioada de folosire a a vionului. Un astfel de ciclu a stat la baza proiectului de standard care specifică criteriile şi procedurile de calitate pentru elaborarea setului de programe funcţionale pentru o unitate. După cum se remarcă în figura 4.1 definiţia funcţionalităţii unităţii începe cu proiectarea de principu a avionului, continuă cu proiectarea propriu-zisă a unităţii, continuă cu proiectarea testelor, cu validarea acestora şi cu utilizarea lor pe durata de viaţă a avionului. Acest proces este un proces permanent care corectează pe toată durata de viaţă a avionului specificaţiile şi programul de testare.

Procesul de dezvoltare al unui TPS (Test Program Set, care reprezintă totalitatea documentelor necesare pentru execuţia testelor) a fost standardizat din punct de vedere al procedurilor şi al calităţii în standardul ARINC 625. Acesta specifică toate activităţile şi documentele necesare pentru realizarea unui program de testare funcţională.

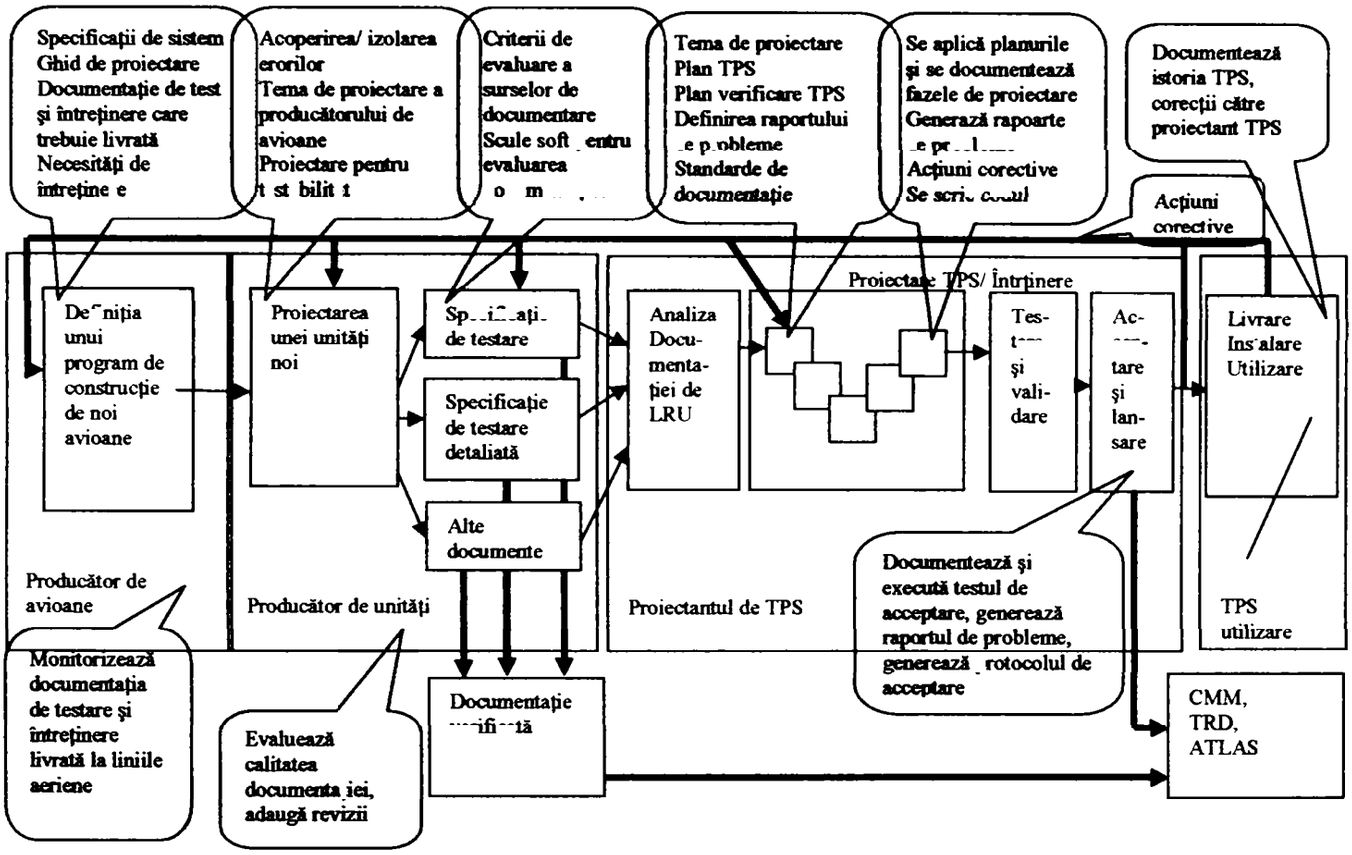


Fig. 4.1 Ciclul de existenţă al unui TPS[14]

4.2 Moduri de descriere a testelor. [182] Moduri de execuţie atestelor.

Documentul care descrie testarea funcţională a fost la unităţile mai vechi, manualul de întreţinere CMM (Component Maintenance Manual). CMM-ul are o secţiune de testare în care se precizează echipamentul de testare, testele care trebuie parcurse şi toleranţele verificărilor. Acest gen de document s-a adresat unităţilor care aveau echipamente de testare specifice şi unde tehniceanul executa manual testele. Generarea unor programe de testare funcţională după aceste documente este dificilă deoarece nu oferă o informaţie precisă, de multe ori aceasta fiind cuprinsă în modul de construcţie al testorului. Trecerea unor astfel de unităţi pe testoare de uz general implică multe revalidări ale procesului de testare.

Documentul, care este utilizat astăzi şi care se construieşte conform standardului MIL-STD-1519, se numeşte TRD (Test Requirement Document). TRD-ul în construcţia sa defineşte cel mai bine testul funcţional. TRD-ul este sursa pentru descrierea testelor în

limbajul ATLAS conform standardului IEEE 416. Testele în ATLAS sunt mai formal prezentate dar conservă cu exactitate informaţia din TRD.

TRD-ul va oferi informaţia necesară pentru a testa unitatea cu minimum de interfeţe suplimentare. Prin definiţie, TRD-ul furnizează informaţia necesară pentru a testa performanţele unităţii, pentru a detecta toate erorile şi depaşirile de toleranţe, pentru a face ajustările şi reglările când este posibil.

Secvenţa de testare este complexul de acţiuni necesare pentru a verifica o funcţiune independentă a unităţii. Secvenţele sunt astfele alese şi ordonate încât o secvenţă care are erori să nu influenţeze testarea celorlate secvenţe.

Testele sunt efectuate pe categorii. Se disting câteva categorii mai importante:

- **Teste anterioare testului funcţional de verificare a continuităţilor şi la scurt circuit**. Acestea sunt teste pentru punctele unde se aplică surse sau stimuli (înainte de aplicarea acestora);
- **Teste funcţionale propriu-zise**. Acestea sunt teste parcurse complet pentru a revela orice degradare în funcţionare;
- **Teste de diagnosticare**. Acestea sunt teste parcurse dupa detectarea unei erori de testele funcţionale pentru găsirea elementelor fizice defecte sau uneori pentru reglaje;
- **Teste parţiale**. Acestea sunt teste singulare sau grupuri de teste parcurse pentru precizarea unor defecţiuni.

4.3 Proiectarea programelor de testare funcţională.

Proiectarea programelor de testare funcţională cuprinde mai multe etape:

- analiza documentaţiei referitoare la unitate;
- proiectarea diagramelor de testare ;
- proiectarea diagramei programului de testare;
- proiectarea interfeţei de testare;
- proiectarea programului de autotestare a interfeţei;
- proiectarea programului de testare;
- validarea interfeţei şi a programului de autotestare;
- validarea programului de testare pe mai multe unităţi funcţionale;

Programul de testare este, fie inclus într-o structură care asigură un proces unitar de execuţie, fie asigură el acest proces. Am ales două unităţi testate, *dintre cele la care autorul a participat la proiectarea programelor şi interfeţelor de testare*, pentru care se prezintă diagramele de testare funcţională .

Prima se numeşte ADC (Air Data Computer) [203] şi este o unitate care furnizează informaţii despre altitudine, viteză şi alte mărimi.

A doua unitate se numeşte AT (Autothrotle)[204] şi controlează tracţiunea motoarelor avionului.

Cele două unităţi au avut doar manual de întreţinere fapt care a făcut mai dificilă generarea programului de testare. Prima ordinogramă include şi ordinograma pentru testul interfeţei.

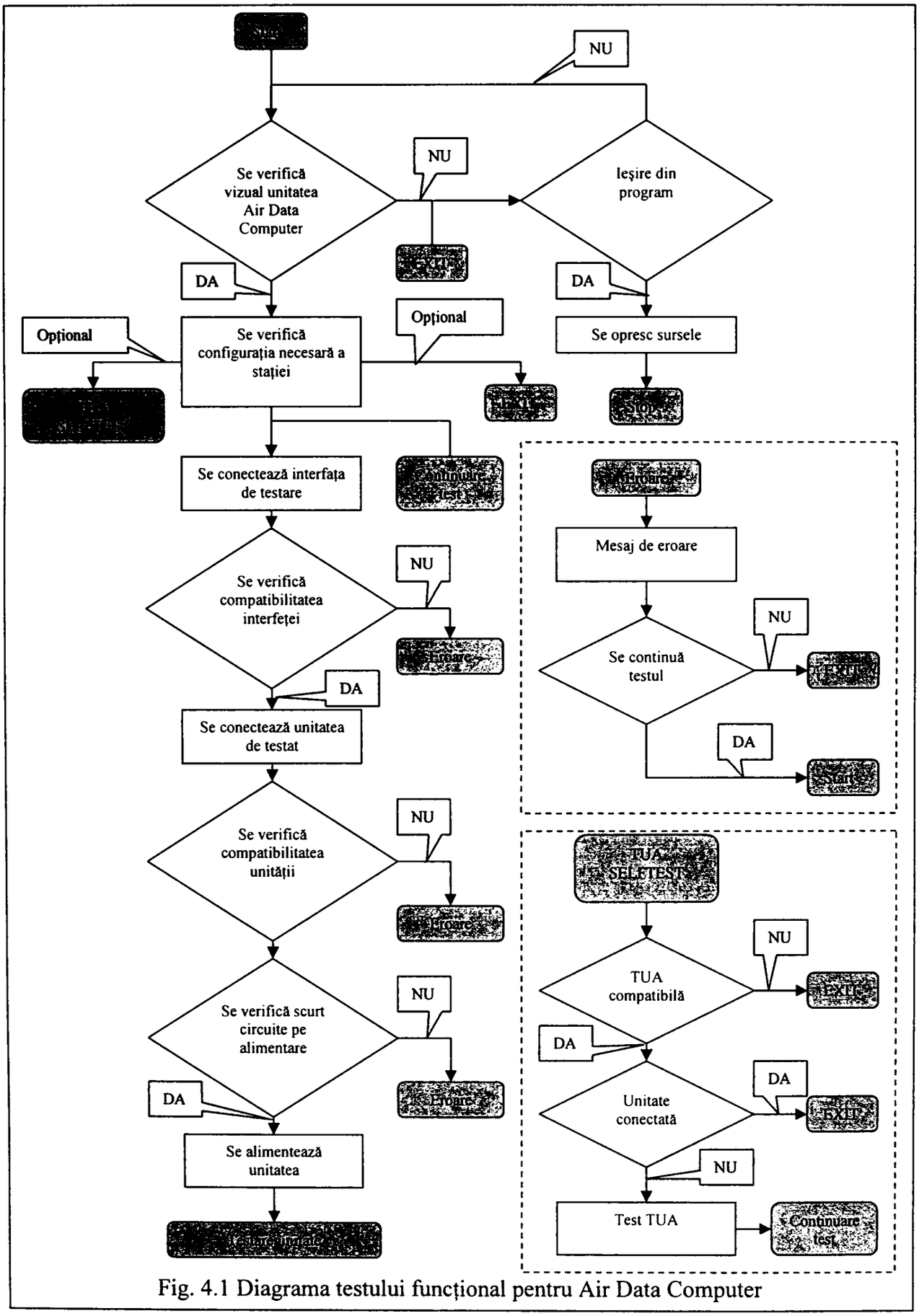


Fig. 4.1 Diagrama testului funcţional pentru Air Data Computer

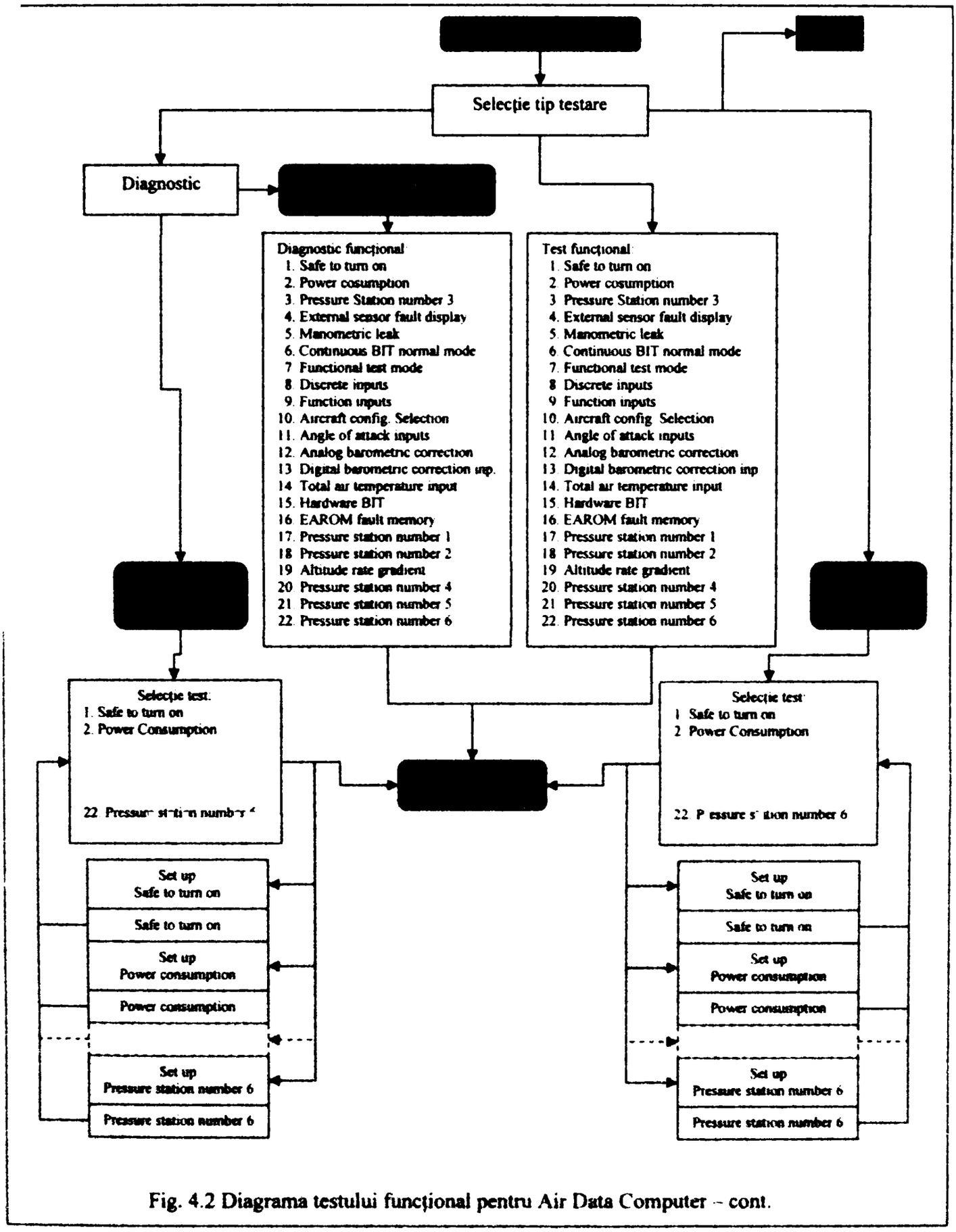


Fig. 4.2 Diagrama testului funcţional pentru Air Data Computer – cont.

În figura 4.1 este prezentată ordinograma părţii de program care iniţializează testarea propriu-zisă. Se remarcă procedurile de verificare a interfeţei şi a unităţii de testare înainte de aplicarea tensiunilor de alimentare. Interfaţa de testare poate fi testată separat înainte de începerea testului unităţii, sau dacă sunt necesare calibrări care se verifică înainte de fiecare test. Programul de testare al interfeţei este similar cu cel al unităţii ca structură.

In figura 4.2 este prezentată ordinograma programului de testare pentru unitatea Air Data Computer. După cum se observă există mai multe opţiuni.

Operatorul poate opta pentru modul **test funcţional complet** care reprezintă validarea unităţii respective pentru serviciu. De multe ori se face un test complet pentru a avea o imagine de ansamblu asupra stării unităţii după care se repetă testele specifice unde au apărut erori. De asemenea din interpretarea tuturor erorilor se pot evidenţia mai uşor circuitele cu probleme.

1.1 BONDING
1.2 POWER CHECK
1.3 CURRENT LIMIT
1.4 FLIGHT FAULTS
1.5 SET DISPLAY
2.1 CPU USTEP
2.2 STORE RAM UPPER
2.3 STORE RAM LOWER
2.4 STORE PRG CONTS
3.1 PERIPH TEST 0
3.2 PERIPH TEST 1
3.3 PERIPH TEST 2
3.4 PERIPH TEST 3
3.5 PERIPH TEST 4
3.6 PERIPH TEST 5
3.7 PERIPH TEST 6
4.1 PERIPH TEST 7A
4.2 PERIPH TEST 7B
4.3 PERIPH TEST 7C
4.4 PERIPH TEST 8
5.1 PARTS A TO E
5.2 LEFT PRTS F L
5.3 RIGHT PRTS F L
5.4 PARTS M TO Q
5.5 PARTS R TO V
5.6 PARTS W TO X
5.7 PARTS Y TO Z
5.8 PARTS AA TO AF
6.1 WATCHDOG
6.2 RELOAD
6.3 TEST F
7.1 NIM CONV
7.2 NIM SUPP
7.3 NIM TRIP
7.4 NIM COMM
7.5 THRUST ONE
7.6 THRUST TWO
8.1 ARINC REC
8.2 ARINC TX
8.3 ARINC WAVE
8.4 APPS ENTRY
8.5 BITE TEST
8.6 RAM WIPT

Fig. 4.3 Lista testelor funcţionale pentru Autothrotle

De asemenea operatorul poate opta pentru o secţiune din teste numită **test funcţional parţial**. Menţionăm că testele în testarea funcţională sunt astfel construite încât să fie cât de independente posibil şi să definească cu precizie un anumit circuit al unităţii conjugat cu o anumită funcţionalitate. Această calitate permite continuarea testării dacă unul din teste a raportat erori. Există desigur şi **teste obligatorii** pentru unitate pentru a putea continua procesul de testare. Aceste teste pot genera erori catastrofale (un exemplu sunt testele pentu circuitele de alimentare cu energie electrică) care trebuiesc rezolvate pentru a continua testul.

Operatorul poate opta pentru testarea în mod **diagnostic** care la rândul ei poate fi realizată complet sau parţial. În regim diagnostic la programul de testare propriu-zis se adaugă secţiuni care indică circuitele sau componentele defecte şi dacă este necesar şi posibil permit anumite calibrări.

În figura 3.3 se prezintă doar lista testelor din testul funcţional complet pentru o altă unitate testată AT (Autothrotle). Ordinograma de testare este similară cu cea de la ADC (Air Data Computer).

Fiecare test din această listă are o diagramă de testare asociată şi o secvenţă (program) de testare care se execută de către echipamentul de testare automată. Se analizează in continuare pentru exemplificare blocul de teste nr. 2 de la unitatea AT din punct de vedere al documentaţei descriptive, al diagramei de testare şi al secvenţei de programare asociate.

Blocul nr. 2 de teste verifică procesorul (CPU) unităţii testate din punct de vedere al executării microinstrucţiilor, al funcţionării celor două memorii

RAM, precum şi din punct de vedere al funcţionării programelor. Se prezintă în continuare schema de testare necesară.

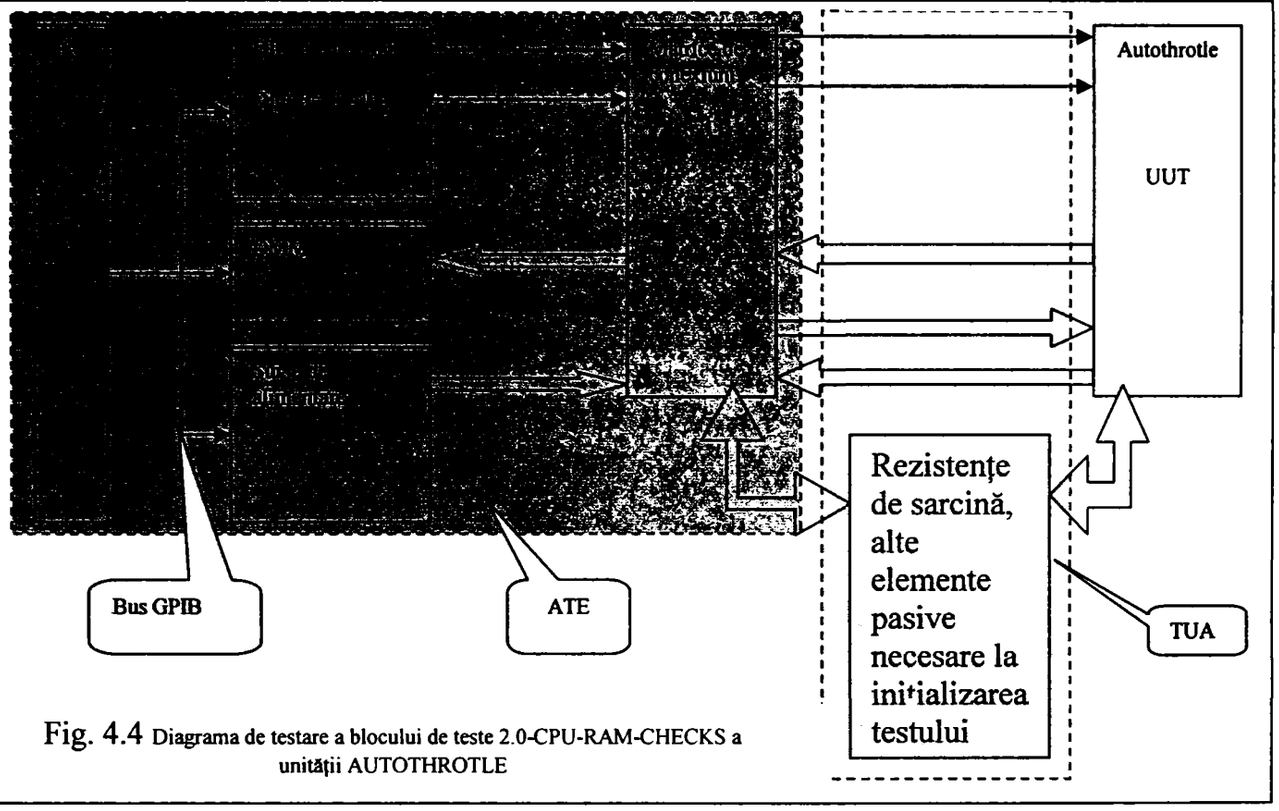


Fig. 4.4 Diagrama de testare a blocului de teste 2.0-CPU-RAM-CHECKS a unităţii AUTOTHROTLE

Testul funcţionează astfel. Se iniţializează unitatea prin aplicarea tutror sarcinilor, se alimentează unitatea. Se încarcă prin cele două conexiuni LOAD şi DATA serial diverse fişiere in unitate. Se lansează programele încărcate prin punerea scurtă la masă a unei intrări a unităţii. Se verifică cuvinte de stare măsurabile la pinii unităţii cu DMM-ul. Se generează raport de testare. Unitatea are descise la acest bloc, patru teste:

- 2.1 CPU-UPSTEP, care verifică execuţia microinstrucţiilor în unitatea centrală.
- 2.2 STORE-RAM-UPPER, care verifică funcţionarea blocului de memorie numit Ram Upper.
- 2.3 STORE-RAM-LOWER, care verifică funcţionarea blocului de memorie numit Ram Lower.
- 2.4 STORE-PROGRAMME-CONTENTS, care încarcă soft-ul operaţional într-un banc de memorie şi soft-ul de autotest la sol BITE în celălalt banc de memorie şi verifică această operaţie.

Se prezintă în **Anexa 1** definiţia testelor prezentată într-o variantă de ATLAS 616 (o variantă de limbaj ATLAS folosită pentru descrierea testelor), schema detaliată a interfeţei de testare, programul de testare pentru secvenţa de teste nr. 2 scris într-un limbaj de tip interpretor, şi raportul de testare pentru această secvenţă.

4.4 Tipuri de teste utilizate frecvent în avionică [55]-[70]

În decursul acestei activităţi *autorul a participat la testarea a peste 200 de unităţi* de avionică. O parte din aceste unităţi sunt listate în **Anexa 5**. Această activitate a permis o privire de ansamblu asupra tipurilor de circuite testate şi posibilitatea unei clasificări a testelor uzuale pentru avionică. Clasificarea a luat în considerare atât schema electronică a dispozitivului precum şi structura unui program de testare funcţională.

4.4.1 Măsurarea conexiunilor şi componentelor pasive.

Testarea funcţională începe întodeauna cu un şir de măsurători de rezistenţă între pinii unităţii sau alte componente pasive aflate între pinii unităţii. Acest gen de măsurători permite excluderea din start a unor defecţiuni catastrofale precum şi identificarea unităţii pe baza unor valori caracteristice acesteia înainte de alimentarea cu energie electrică.

4.4.2 Verificarea circuitelor de alimentare la rece

Circuitele de alimentare electrică ale unităţii sunt verificate la rece pentru a evita existenţa unor scurtcircuite sau a altor defecţiuni majore. Astfel pentru circuitele de curent alternativ se măsoară impedanţa de intrare a transformatoarelor şi eventual componentele din circuitele de comutaţie unde este cazul. Pentru circuitele de curent continuu se verifică componentele pasive (mai ales condensatorii) şi active.

4.4.3 Alimentarea unităţii cu energie electrică. Măsurarea puterii consumate.

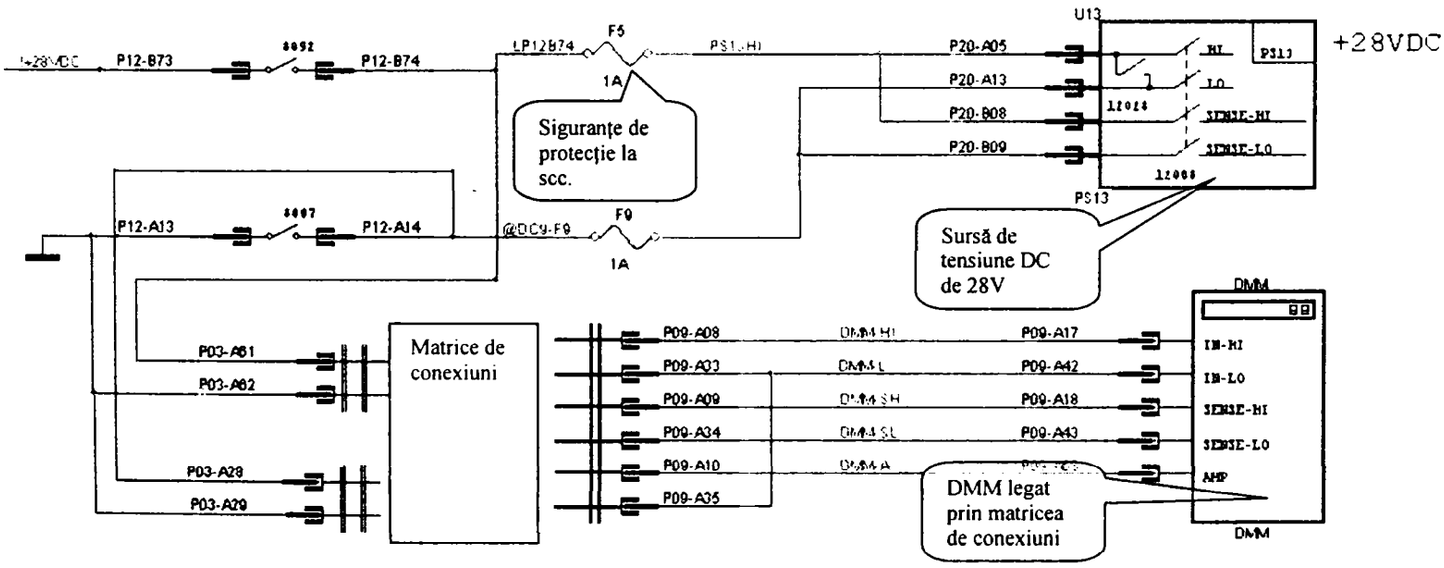


Fig. 4.5 Alimentarea unei unităţi şi măsurarea mărimilor electrice

Se remarcă protejarea sursei la scurt circuit precum şi posibilitatea de a măsura tensiunea şi curentul furnizate cu ajutorul DMM-ului. Sursele de DC au capabilitatea în

general să măsoare şi curentul furnizat şi tensiunea şi uneori şi puterea, dar nu întodeauna cu precizia necesară. Nu acelaşi lucru se poate spune despre sursele AC unde schema de mai sus este foarte utilă. De asemenea la măsurarea curentului prin matrice trebuie avut în vedere curentul maxim admis de către aceasta.

4.4.4 Verificarea canalelor de comunicaţie de tip ARINC 429

De obicei verificarea canalelor de comunicaţie cuprinde mai multe tipuri de teste. Se prezintă diagrama de testare pentru ARINC 429:

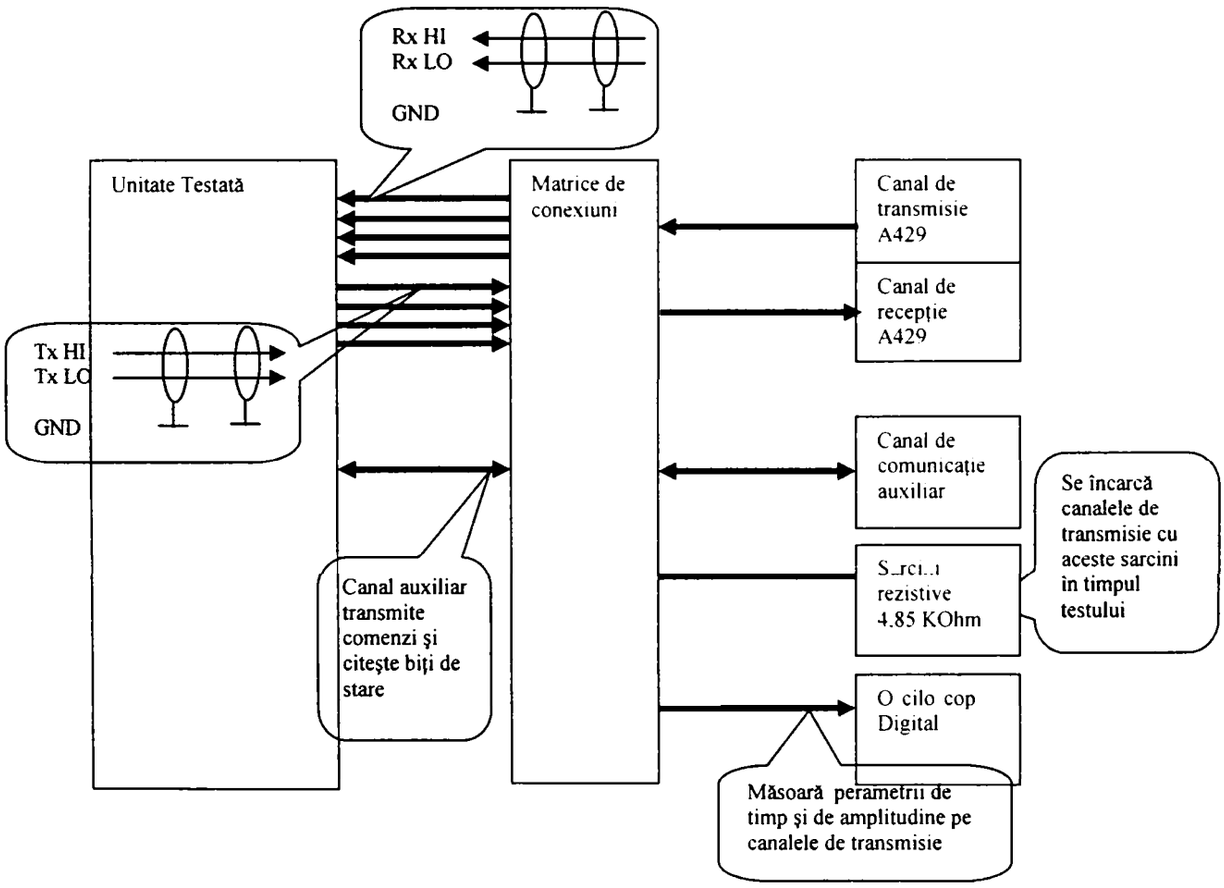


Fig. 4.6 Schema de principiu pentru testul canalelor A429

Testul cuprinde verificarea transmiţătoarelor şi a receptoarelor. În general o parte din teste se fac utilizând o conexiune serială (RS 232, RS 485, sau un canal A429 de test) auxiliară prin care se verifică mesajele recepţionate de unitate sau valoarea anumitor biţi de stare din unitate.

Tot prin această conexiune serială se comandă transmisia unor cuvinte sau mesaje de către unitate. În general acest lucru se face prin încărcarea canalului cu o impedanţă de sarcină.

O altă parte din test verifică forma de undă a semnalelor. Această verificare se face cu osciloscopul. Parametrii măsuraţi se prezintă în figura 3.7.

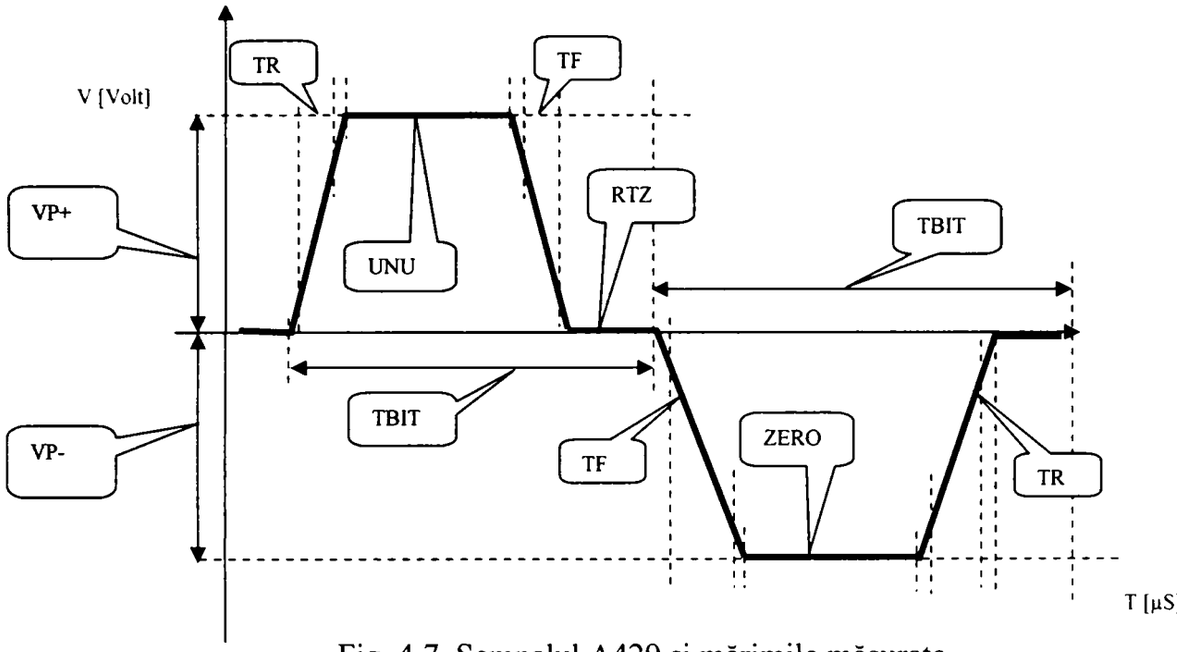


Fig. 4.7 Semnalul A429 şi mărimile măsurate

Verificarea se face astfel: se transmite către unitate comanda de emisie a unui singur cuvânt de 32 biţi format numai din UNU (FFFFFFFF Hex). Se declanşează osciloscopul pe frontul crescător şi se memorează forma de undă. Se măsoară VP+(peak voltage), TR (rise time), TF (fall time). Se transmite un cuvânt format numai din ZERO (00000000 Hex). Se măsoară VP- (peak voltage), TR, TF. Pentru măsurarea lui TBIT care este durata unui bit se transmite din nou un cuvânt format numai din UNU după care se măsoara anvelopa ca durată şi se împarte la 32. Se opbţine astfel o valoare medie. Măsurătorile se fac prin funcţii specializate ale osciloscopului. La nivel de utilizator fie se văd funcţii distincte fie se transmite o comandă gen: „VERIFY 429 WAVEFORM". În acest moment nu există un instrument de tip IVI (cu capabilităţi definite) pentru A429. RTZ semnifică „Return To Zero" şi sublinează un bit se termină întodeauna cu un palier de nivel 0 V.

4.4.5 Verificarea canalelor de comunicaţie de tip ARINC 629.

Comunicaţia ARINC 629 a fost inventată de compania Boeing şi folosită pentru prima dată pe generaţia Boeing 777. A629 este o comunicaţie robustă care foloseşte canalul de comunicaţie la maxim şi care acordă acces egal la bus tuturor participanţilor.

Se prezintă în figura 3.8 o schemă de testare pentru unităţile care au o astfel de comunicaţie.

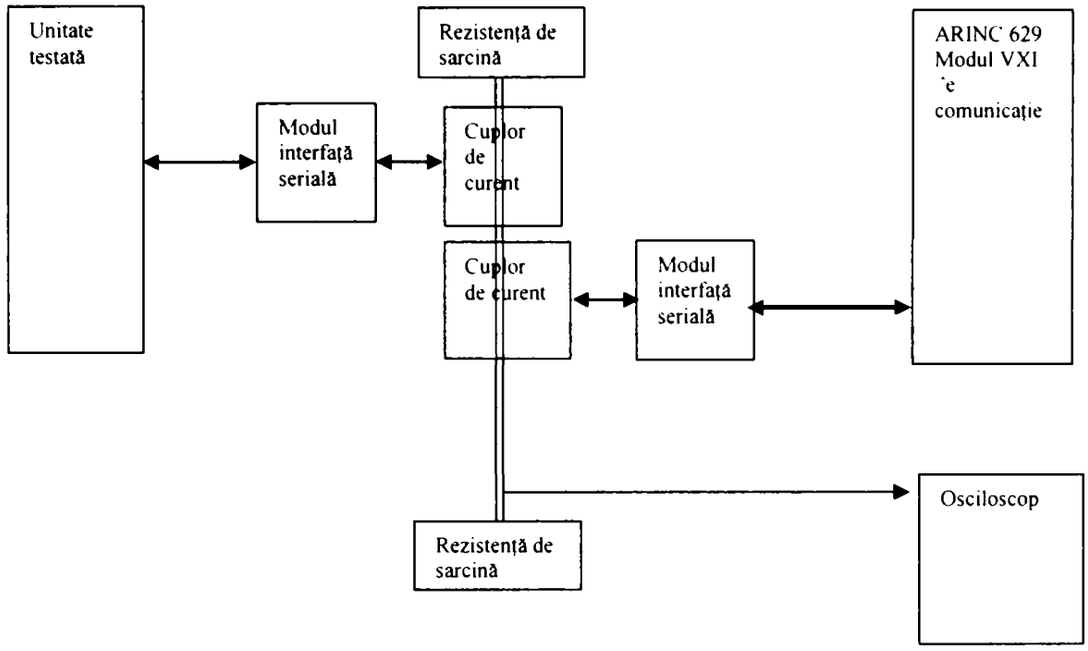


Fig. 4.8 schemă de testare comunicaţia ARINC 629

La fel ca în cazul A429 şi la A629 se măsoară câteva mărimi de timp şi de amplitudine cu osciloscopul care se prezintă în figura 4.9:

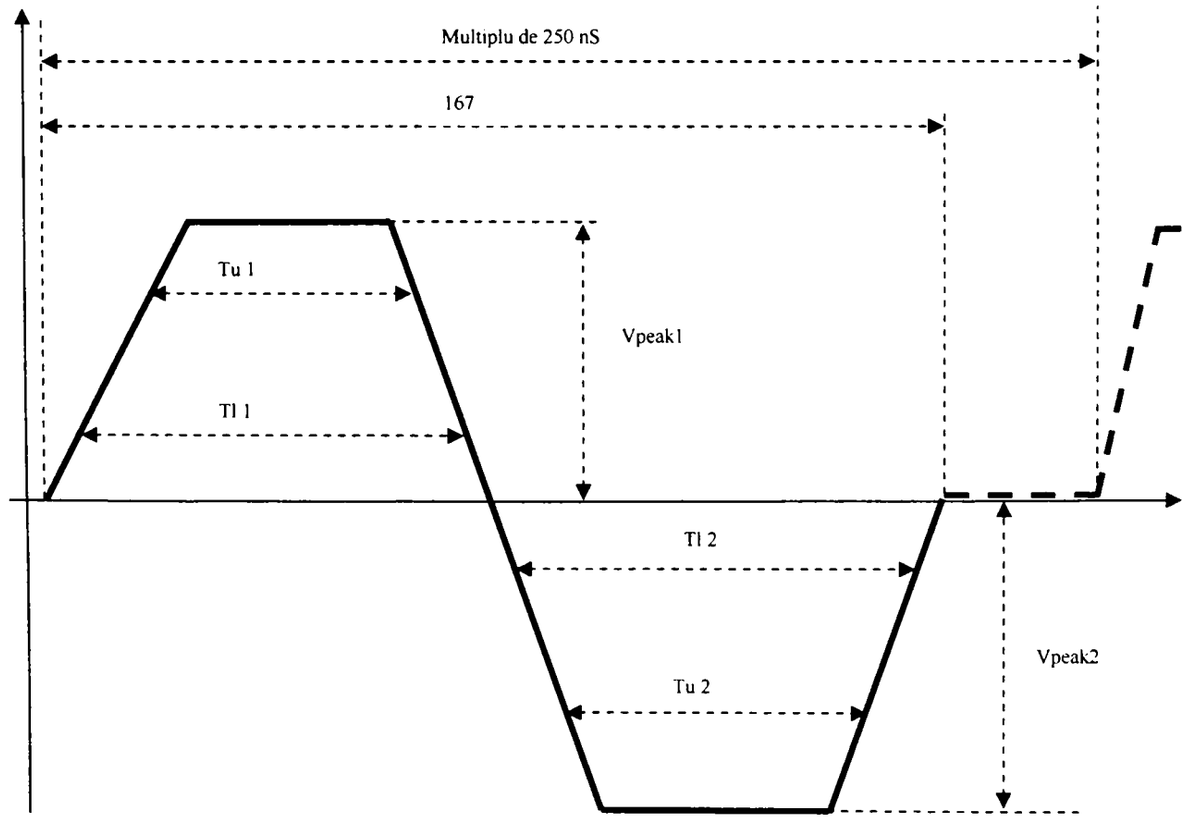


Fig. 4.9 Dubletul A629 şi mărimile măsurate

Acest semnal se numeşte dublet şi poate fi ca cel din figura 4.9 sau în oglindă în funcţie de tranziţia semnalului. Valorile pe care trebuie să le îndeplinească măsurătoarea sunt următoarele:

59.0 nS < Tl 1, Tl 2 <77.4 nS măsurat la +- 1.0 V
52.6 nS < Tu 1, Tu 2 <70.9 nS măsurat la +- 2.25 V
| 3.96 V| < Vpeak < | 5.5 V|

Pe lângă aceste măsurători calitative măsurate cu osciloscopul pe cele două fire torsadate care reprezintă fizic bus-ul , se fac şi verificări de corectitudine de funcţionare a protocolului precum şi respectarea intervalelor de timp între mesajele puse pe bus de diverse dispozitive. Vom analiza în capitolul 4 un driver pentru o placă Tektronix VXI pentru comunicaţia A629. Nici pentru A629 nu există un instrument generic şi un driver IVI.

4.4.6 Verificare comunicaţie LAN (Ethernet).

Verificarea comunicaţei de tip Ethernet, pe bază de protocol TCP/IP se face prin intermediul unui hub, prin transmiterea unor mesaje şi verificarea corctitudinii acestora. Nu există drivere de tip IVI deşi acest gen de comunicaţie este standardizat.

4.4.7 Verificarea utilizând testarea pe contur „Boundary scan"(J-TAG).[146][101]

Unităţile noi sunt proiectate pentru a putea fi testate aplicând standardul IEEE 1149.1 sau J-TAG. Se încarcă pe magistrala J-TAG vectorii de testare, după care se verifică diverse funcţii ale unităţii. În principiu schema de testare se prezintă în figura 4.10:

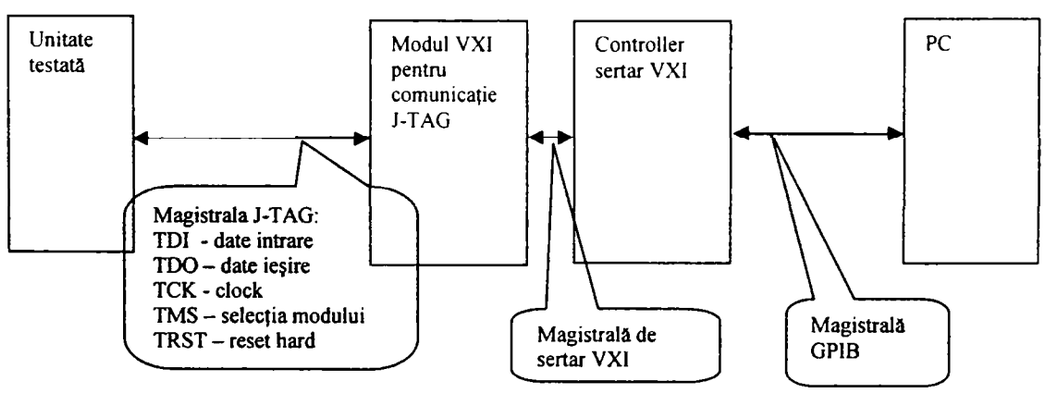


Fig. 4.10 Schema de testare pe magistrala J-TAG

După cum se remarcă este o schemă de testare simplă . Testarea este o succesiune de încărcări şi descărcari de vectori de testare precum şi măsurarea unor pini ai unităţii testate în corelaţie cu vectorul încărcat. Din punct de vedere hard procedeul este simplu se conectează cablul de legătura pentru magistrala J-TAG între testor şi unitate. Din punct de vedere al programelor care rulează pe diversele entităţi din lanţ (unitate, modul VXI, Controller de sertar, PC) construcţia software este complexă, iar scrierea driver-elor este laborioasă. Nu există un instrument generic pentru J-TAG şi nici drivere tip IVI.

4.4.8 Verificări care utilizează simulatorul de LVDT (Linear Variable Differential Transformer)[113][130]

LVDT-ul este folosit foarte mult în industria de aviaţie pentru diverse traductoare de poziţie. În figura 4.11 este prezentată o secţiune printr-un LVDT:

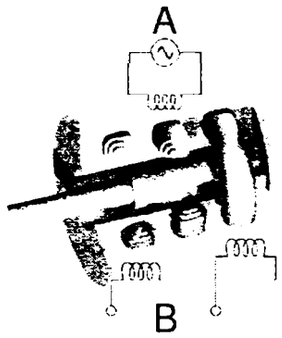


Fig. 4.11 LVDT

Primarul este alimentat cu o tensiune alternativă de aproximetiv 10 KHz. Mişcarea miezului schimbă valoarea inductanţelor mutuale. Diferenţa semnalelor la ieşirea va fi proporţională cu poziţia miezului şi în fază sau antifază după cum acesta se află la stânga sau dreapta primarului. Acest gen de stimul (de la simulator spre unitate) se foloseşte foarte des în testarea avionicii, numeroase unităţi care controlează ozi ia unor subansamble având circuite conectate la astfel de traductoare.

Pentru testare se folosec simulatoare. Există plăci VXI care au mai multe canale care simulează LVDT –uri (ex placa 7836 VXI LVDT Simulator produsă de North Atlantic Instruments Inc.). Schema bloc a unui astfel de simulator este următoarea:

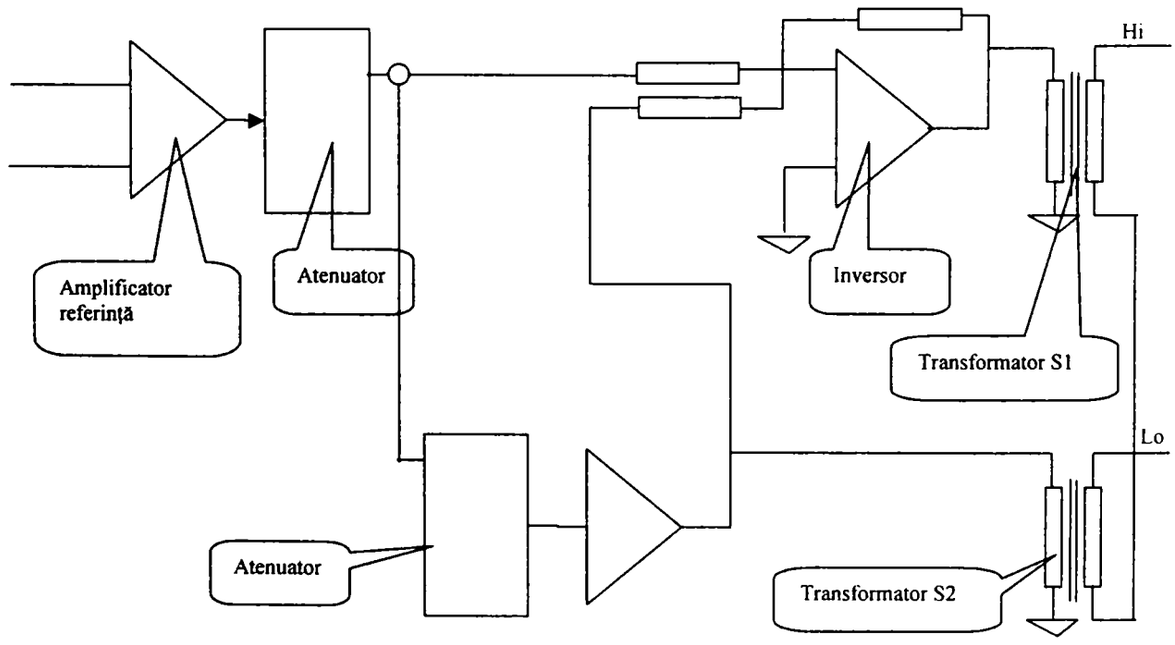


Fig. 4.12 Schemă simulator LVDT

Placa simulator de LVDT poate fi comandată prin limbaj SCPI specificând raportul între referinţă si mărimea de ieşire cu semn. Conectarea în ansamblul echipamentului de testare se face între PC, Controller VXI, Simulator LVDT, Matrice de conexiuni, Interfaţă de adaptare, Unitate testată. Nu există pentru Simulatorul LVDT definit instrument generic şi nici driver IVI.

4.4.9 Verificări folosind Synchro/Resolver-ul.[114]

Synchro/Resolverul este un dispozitiv foarte util (la fel ca şi LVDT-ul) care funcţionează în condiţii difcile şi care poate da informaţii de poziţie la rândul său. Multe unităţi sunt conectate la dispoţitive tip synchro-resolver. Pentru a le testa este necesar un simulator de Synchro/Resolver. Un astfel de instrument este placa VXI produsă de North Atlantic Instruments Inc. care conţine cel puţin 2 simulatoare de synchro/resolver şi un circuit API (Angle Position Indicator).

Synchro/Resolverele sunt folosite pentru multe unităţi de pe o aeronavă: unităţile din sistemul inerţial de navigaţie, unităţile de supraveghere contra incendiilor, unităţile responsabile de poziţionarea pe suprafaţa de zbor, unităţile care controlează acţionarea flapsuri-lor şi altele. O schemă obişnuită de testare este cea din figura 4.13:

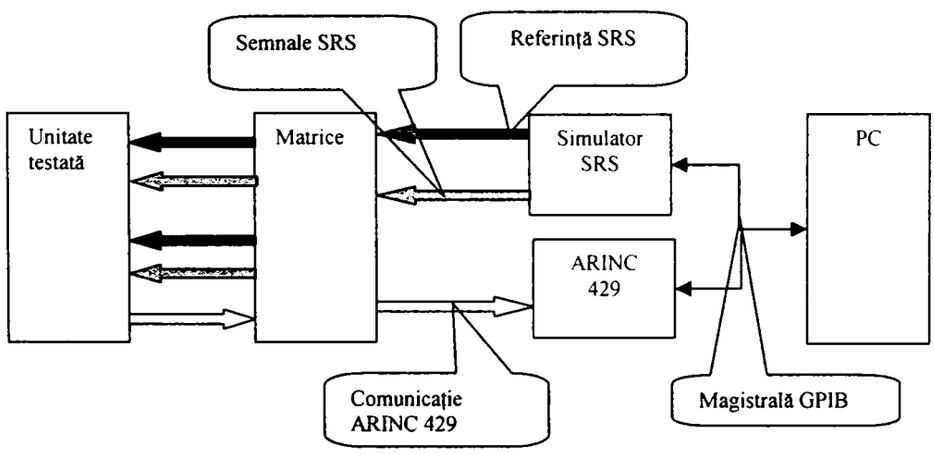


Fig. 4.13 Diagramă de testare cu Synchro/Resolver

În diagrama de testare din figură, unitatea primeşte de la traductorul de poziţie de tip synchro/resolver (simulat în diagramă) semnale defazate cu numărul de grade corespunzător faţă de referinţă (ex. Poziţia flapsurilor). Unitatea transmite valoarea sub formă numerică pe magistrala ARINC 429. Se compară de către programul de testare care rulează pe PC valoarea prescrisă şi cea retransmisă de unitate.

Synchro/Resolverele folosesc referinţe de 115V (sau 26V) şi 400Hz. Aceste semnale sunt aplicate prin matrice unităţii. Instrumentul este un dispozitiv VXI comandat prin regiştrii. Driver-ul aflat pe controller-ul de magistrală VXI permite comanda in grade şi implementează limbajul SCPI. Nu există definite capabilităţi pentru instrument generic şi nici driver IVI pentru acest instrument.

4.4.10 Verificări uzuale folosind PGA (Programmable Gain Amplifier)

PGA-ul este o placă VXI care are mai multe canale de amplificare care pot fi comandate în amplitudine. Aceasta se foloseşte la multe teste, pentru dimensionarea semnalelor la valoarea dorită. Nu are definite capabilităţi sau driver IVI

4.4.11 Verificări uzuale de teste folosind Osciloscopul.

Osciloscopul digital este unul dintre cele mai utilizate instrumente în testarea pentru avionică. După cum am prezentat în exemplele de comunicaţie de mai sus osciloscopul este utilizat pentru măsurarea formelor de undă. Un caz des întâlnit este măsurarea rampelor foarte lente de ordinul secundelor sau al zecilor de secunde. Deoarece instrumentul poate fi declanşat de un semnal treaptă pe alt canal sau poate fi declaşat soft aceste rampe pot fi memorate şi evaluate apoi punct cu punct.

Osciloscopul este definit ca instrument generic de clasa Scope şi există drivere IVI realizate pentru el.

4.4.12 Verificări uzuale folosind „Timer-Counter"-ul.

Timer/Counter-ul este un instrument folosit pentru măsurarea duratelor de timp, a perioadelor sau a frecvenţelor. Situaţia cea mai folosită este aceea de a măsura distanţa în timp pentru două evenimente electrice. Instrumentul are două canale cu parametrii de declanşare programabili pe care sunt monitorizate evenimentele care interesează. Timpii sunt măsuraţi cu precizie foarte bună. În testarea automată există tendinţa de a folosi timer-e soft implementate pe computerul pe care rulează programul de testare. Acestea au o precizie relativă în raport cu evenimentele care au loc în unitate şi de aceea sunt folosite cel mult pentru evenimente foarte lungi de ordinul zecilor de secunde.

Timer/Counter-ul este un instrument cu capabilităţi generice definite şi cu drivere IVI dezvoltate de producători şi alte companii.

4.4.13 Verificări uzuale folosind AFG (Arbitrary Function Generator).

Arbitrary Function Generator este un instrument folosit la generarea diferitelor forme de undă utilizate pentru testarea unităţilor. Spre exemplu atunci când există moduri de comunicaţie simple, dar nestandardizate sau mai vechi, mesajele transmise (mai precis cuvintele) pot fi construite cu ajutorul AFG şi răspunsurile recepţionate cu ajutorul osciloscopului şi interpretate cu acesta. De asemenea forme de undă mai deosebite sau semnale periodice cu formă nestadard pot fi generate cu acest instrument.

AFG este de obicei un instrument VXI, există capabilităţi generice definite şi sunt construite drivere IVI pentru acest instrument.

4.4.14 Verificări uzuale folosind DIO (Data Input Output).

DIO semnifică intrări şi ieşiri digitale. Numeroase instrumente au definite şi câte un port de intrări/ieşiri digitale. Se folosec de asemea plăci specializate cu mai multe porturi paralele de intrări/ ieşiri. Acestea sunt foarte utile la realizarea condiţiilor de test pentru anuimite teste, la verificarea unor ieşiri sau la simularea unor comunicaţii seriale.

Nu există o standardizare pentru acest gen de instrumente.

4.4.15 Verificări în domeniul RF[143][144]

Instrumentele cele mai utilizate pentru măsurători în domeniul frecvenţelor înalte (RF) sunt Generatorul RF (RF Gen), analizorul de spectru (Spectrum Analyzer), analizorul de reţea (Network analyzer) şi instrumentul de măsurarea a puterii (RF Powermeter) . Legătura între aceste instrumente şi unitate nu se poate face prin matricea de conexiuni definită de ARINC 608 şi nici prin cablări obişnuite. De obicei se adaugă o matrice suplimentară (sau doar relee speciale comandate de soft) construită special pentru conexiuni RF (cabluri şi conectori coaxiali). De asemenea se adaugă o interfaţă specială pentru partea de RF şi de multe ori şi programul de testare este separat. O schemă simplificată este prezentată în figura 4.14:

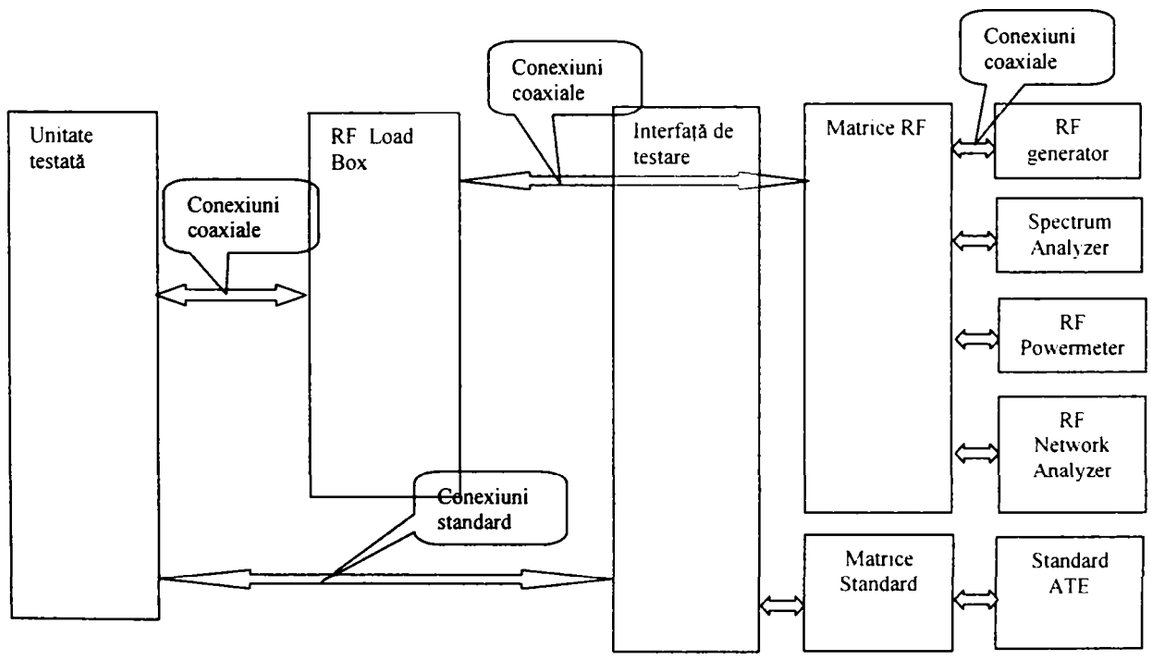


Fig. 4.14 Extinderea testorului cu echipament pentru domeniul RF

În general pentru partea de RF se proiectează un program de testare separat.

4.4.16 Verificarea unităţilor prin încărcarea de programe de testare[111]

Unităţile mai noi, digitale, echipate cu procesoare complexe sunt testate prin încărcarea unor programe de test care execută anumite funcţii. Acestea sunt verificate fie prin măsurători ale unor mărimi electrice la conectorul unităţii, fie prin mesajele pe care le transmite unitatea către testor.

Exemplul cel mai sugestiv este sistemul de încărcare a programelor de testare pentru avionica de tip modular care echipează avioanele din clasa Boeing 777. Se prezintă modul de funcţionare al unităţilor (în general) din acest punct de vedere în fig. 4.15:

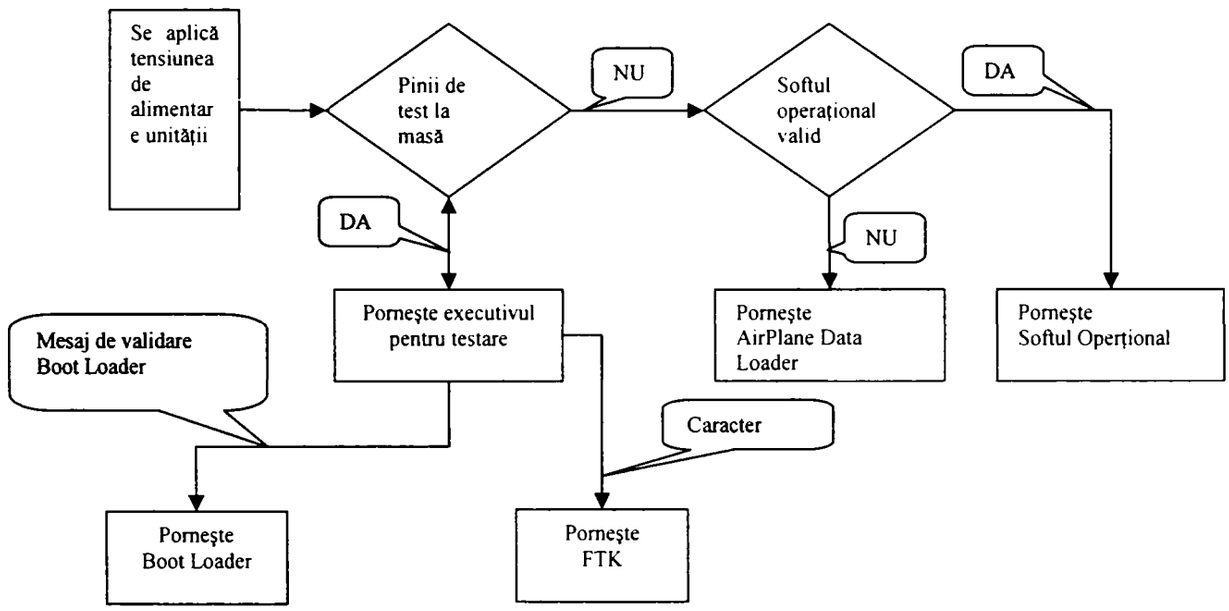


Fig. 4.15 Diagrama de iniţializare a unităţilor din clasa B777

La aplicarea tensiunii procesorul unităţii verifică cei doi pini de test. Dacă aceştia nu sunt legaţi la masă se verifică dacă este valid softul operaţional. Dacă acesta este valid unitatea intră în regim de funcţionare normală. Dacă acest soft nu este valid se lansează o aplicaţie de încărcare a softului. Această operaţie poate fi făcută şi la bordul avionului prin intermediul unităţii CMC (Cabin Management Controller) şi a magistralei de date, permiţându-se astfel modificarea softului de pe unităţi cu revizii noi.

Dacă pinii de testare sunt la masă se lansează un executiv de testare care aşteaptă 5 secunde un mesaj pe comunicaţia de test RS232 a unităţii. Acest mesaj poate fi o cerere de validare iniţializare a transferului de fişiere (Boot Loader) sau o lansare a nucleului de testare funcţională (FTK).

În testarea funcţională se utilizează funcţia de încărcare şi lansare a unor executabile, prin intermediul cărora se verifică diverse funcţii.

Această operaţie nu este standardizată neexistând capabilităţi generice. În mod obişnuit atunci când apare necesitatea încărcării unei aplicaţii de acest gen, programul de

testare comută controlul unui program extern furnizat de producător cu care se face operaţia. *Autorul a implementat lansarea „loader"-ului ca aplicaţie DOS (produs de firma BOEING) sub sistemul de operare UNIX (varianta UnixWare) din aplicaţia de testare şi revenirea la aceasta. Procedura este destul de complexă deoarece aplicaţia este un executabil care rulează sub UNIX şi care se leagă cu porturile computerului prin nucleul sistemului de operare (transformat şi îngheţat în Real Time Operating System de o aplicaţie a firmei Venix) în timp ce „loader"-ul rulează pe un emulator de DOS dar se conectează la portul serial tot prin sistemul de operare.* La unităţi mai vechi, se furnizează protocolul de comunicaţie, programul de testare încarcă executabilul, iar lansarea în execuţie a acestuia se face prin resetarea procesorului. O astfel de unitate este FMC (Flight Management Computer) produsă de firma Smith Ind. care are o arhitectură multiprocesor (trei plăci procesor) şi care este testată prin încărcarea programelor de testare şi resetarea plăcii testate. După resetare se măsoară diverşi parametrii sau se citesc zone de memorie conform documentaţei de testare. *Autorul a proiectat interfaţa de testare şi programul de testare pentru această unitate.*

4.5 Raportul de testare şi interpretarea acestuia.

Raportul de testare se generează automat după parcugerea testului final al unităţii. Acesta certifică şi validează unitatea pentru serviciu. Raportul se încheie întodeauna cu numărul de erori cumulat pentru toate testele funcţionale şi durata testului. Se menţionează aici că un test funcţional folosit pentru validarea unităţii este un program aprobat de producătorul unităţii şi nu poate fi schimbat decât cu aprobarea acestuia.

Din punct de vedere al testului şi al diagnosticului, testul funcţional complet furnizează o imagine asupra stării unităţii. Aşa cum am menţionat, testul funcţional se proiectează în aşa fel încât fiecare modul de testare să fie independent, să reflecte funcţionalitatea unei părţi din unitate şi să nu inflenţeze rezultatele altor teste. Acest lucru permite după rularea testului complet, prin simpla citire a raportului, identificarea circuitelor defecte sau dereglate. După această etapă se rulează programele de diagnostic care permit identificare precisă a componentelor defecte sau reglarea conform procedurii producătorului. După aceste operaţiuni şi remedierea unităţii se rulează din nou testul funcţional complet.

4.6 Gradul de standardizare al unui echipament de testare automată.

Echipamentul de testare automată construit cu instrumentele pe care le-am evaluat mai sus converge spre un echipament cu elemente generice care permite generarea unor programe de testare independente de platformă. Realitatea este mai puţin perfectă. Diverse consorţii industriale încearcă să definească şi să standardizeze mulţimea de elemente care alcătuiesc un testor pentru avionică. Procedura de generare a unui standard industrial este deosebit de complexă. Astfel datorită necesităţilor economice sau tehnice un grup de companii importante într-un domeniu alcătuiesc un consorţiu care generează primele documente. Dacă subiectul prezintă interes atunci numărul membrilor consorţiului creşte. După mai multe revizii consorţiul se adresează unei instituţii de standardizare (in acest caz IEEE sau ARINC) care la rândul ei organizează un grup de

lucru şi generează noi variante de proiect. *Autorul a participat la revizia internă a standardului ARINC 625 referitor la TPS (Test Programm Set)* elaborat de compania RADA Electronic Ind. standard în vigoare în prezent. Prezentăm o structură de echipament în funcţie gradul de standardizare a instrumentelor:

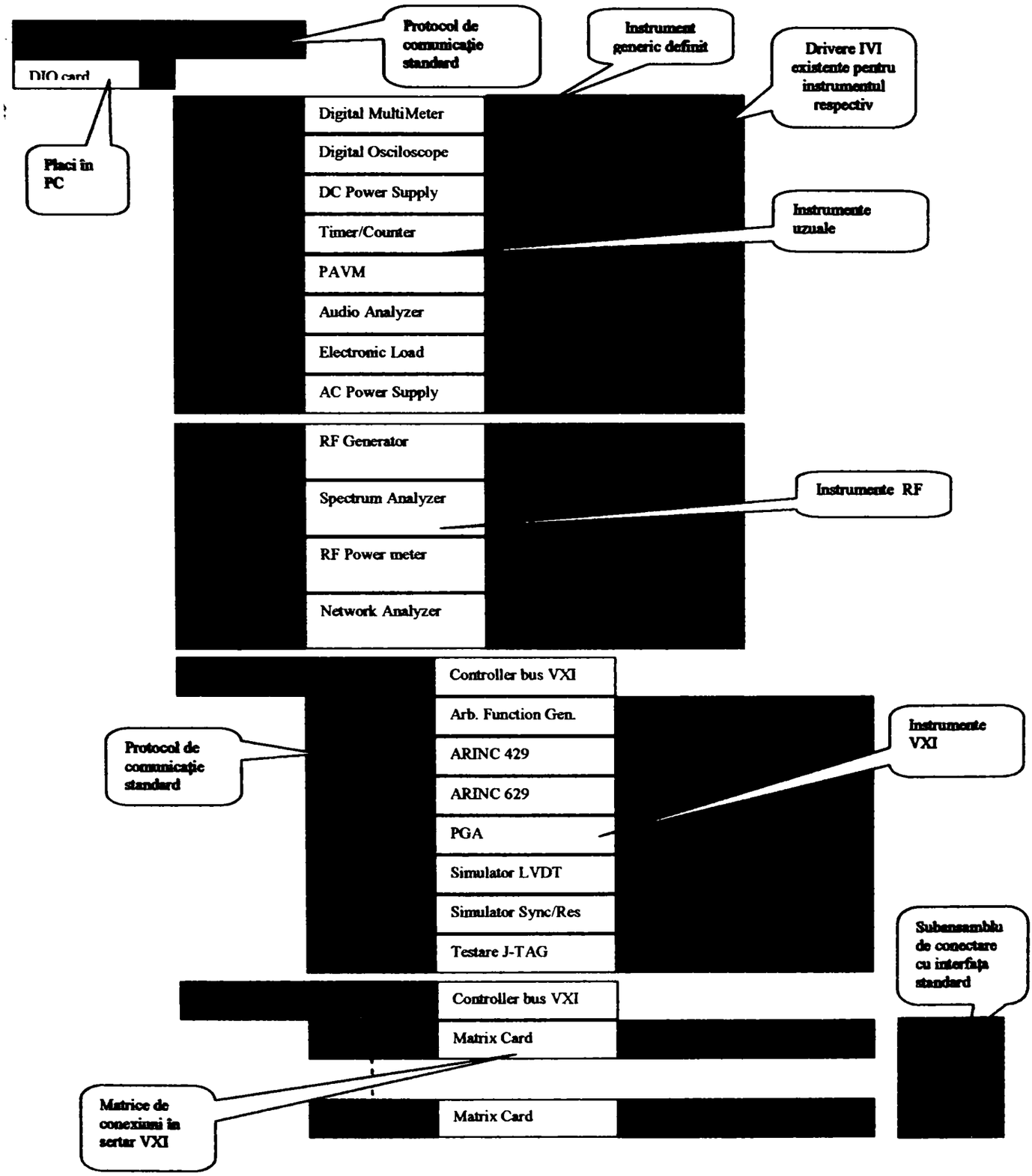


Fig. 4.16 Imagine sintetică asupra gradului de implementare a diverselor standarde pentru un testor de avionică

După cum se remarcă nu toate instrumentele au definite capabilităţi de clasă. Definţia mulţimii de instrumente generice este dificilă deoarece apar în permanenţă necesităţi noi odată cu apariţia unor tehnologii noi. Conceptul de structură implică existenţa unor proceduri care să permită adăugarea de noi instrumente fără a compromite generalitatea. Aceste lucruri se vor analiza în detaliu în capitolul 5. Obiectivul autorului în această teză este de a se apropia de o structură ideală din punct de vedere al portabilităţii programelor de testare, dar şi de evidenţierea unui mecanism care să permită acest lucru.

4.7 Concluzii

Acest capitol defineşte în secţiunea 4.1 modurile de lucru în testarea funcţională în avionică, prezentând şi ciclul complet al unui program de testare din momentul proiectării până în momentul scoaterii din servicu al avionului. De altfel autorul *a participat la proiectarea standardului ARINC 625* în faza de revizie internă a acestuia de către compania RADA Electronic Ind. care a fost iniţiatorul standardului.În secţiunea 4.2 se prezintă modul de descriere a testelor iar în secţiunea 4.3 modul de proiectare a unui test funcţional. Toate aceste observaţii şi definiţii sunt documentate cu aplicaţii practice pe două unităţi. Autorul a participat la *dezvoltarea testelor pentru cele două unităţi*..

Secţiunea 4.4 este o trecere în revistă a tipurilor de teste şi de instrumente care sunt frecvent utilizate în domeniul testării avionicii. Această evaluare este *realizată pe baza situaţiilor de testare reale* cu care autorul s-a întâlnit în activitatea sa. Autorul prezintă câteva soluţii de testare pentru verificări uzuale în avionică subliniind dificultăţile sau implementările mai deosebite precum şi caracterul de noutate datorită în special inovaţiilor aduse de avionca modulară.

Secţiunea 4.5 sublinează importanţa raportului de testare pentru acest gen de teste.

Secţiunea 4.6 încearcă să sintetizeze concluziile din secţiunea 4.5 şi *generează imaginea reală a unui testor de avionică* în prezent precum şi obiectivele care mai sunt de completat în acest domeniu. Prin aceste obiective se înţeleg completările la definirea unor capabilităţi generice acolo unde ele nu există, definirea modului de generare a drivere-lor, posibilităţile de extindere a testorului fără a compromite ceea ce există în prezent. *Această analiză a autorului* permite în capitolul 5. integrarea componentelor de software astfel încât sa conserve caracterul de generalitate al structurii unui sistem de testare pentru avionică.

CAP. V
Mediul de testare, limbaje de programare, drivere[7][8][121][122][160][26][123][10]

Acest capitol prezintă procesul de testare funcţională în avionică, din punct de vedere al tuturor componentelor software care participă la el inclusiv limbajele de programare.

Se prezintă în acest capitol, pentru a avea o imagine de ansamblu, structura mediului de dezvoltare cu toate componentele sale.

Limbajele de programare a evoluat spre variante de nivel înalt care permit portabilitatea acestora pe diverse platforme şi echipamente de testare.

Dezvoltarea driverelor pentru instrumente interschimbabile virtuale permite proiectarea de teste care utilizează elemente generice permiţând trecerea de la o generaţie de testoare la alta fără a reproiecta testele.

Translaţia unor aplicaţii vechi de testare către echipamente moderne fără a reproiecta propriu-zis testele este un procedeu complex dar care permite conservarea unor programe de testare cu efort mult redus.

Aplicaţia care execută programul de testare se numeşte test executiv. Acest gen de aplicaţie a evoluat foarte mult permiţând execuţia programelor de testare în mod unitar cu definirea proceselor de testare prin modelare.

Programele de testare care au fost dezvoltate pe numeroase unităţi au permis verificarea diverselor variante de limbaj, drivere, test executiv.

5.1 Mediul de testare, limbaje, drivere. Privire generală[8][22][2][3][4][5]

Pentru testarea funcţională a unei unităţi sunt necesare mai multe aplicaţii şi programe software. Prezentăm în figura 5.1 o diagramă a activităţilor şi programelor asociate necesare la testarea unei unităţi.

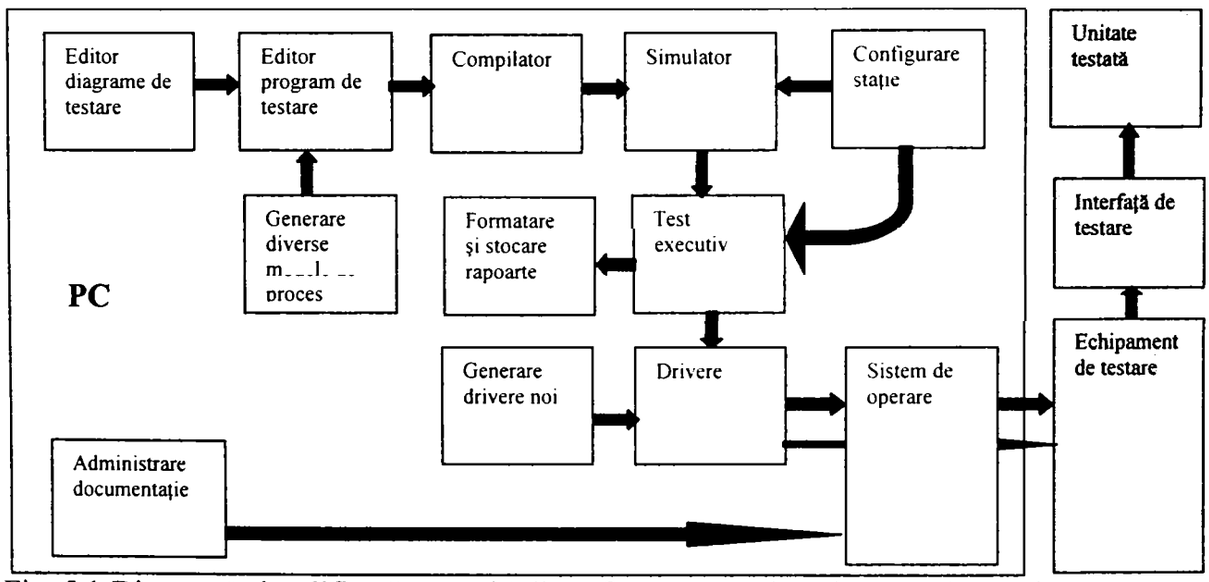


Fig. 5.1 Diagramă simplificată cu entităţile software care alcătuiesc un sistem de testare

Dacă evaluăm figura 5.1 de la stânga la dreapta, precum şi considerând cele prezentate în capitolele anterioare, se constată că ordinea acţiunilor pentru a efectua testarea unei unităţi este următoarea:

- Se analizează documentaţia de testare furnizată de producătorul unităţii;
- Se proiectează diagramele de testare;
- Se proiectează interfaţa de testare şi se configurează staţia de testare;
- Se scrie programul de testare într-un limbaj adecvat;
- Se verifică sintactic programul fie prin compilare fie editorul are o astfel de funcţie;
- Se verifică existenţa driverelor de instrumente necesare, conform configuraţiei staţiei, se construiesc driverele care lipsesc;
- Se verifică programul pe un simulator;
- Se obţine o unitate funcţională, verificată anterior prin alte mijloace;
- Se pune în funcţiune interfaţa de testare, impreună cu programul de verificare a acesteia;
- Se conectează unitatea la interfaţa de testare;
- Se integrează programul de testare pas cu pas.
- Se rulează programul de testare;
- Se verifică testele parţiale;
- Se integrează programul de diagnostic;
- Se rulează testul complet de cel puţin 3 ori şi se generează rapoarte de testare;
- Se generează raport de diferenţe între documentaţia furnizată de producător şi condiţiile de testare rezultate în urma integrării programului.
- Se obţine aprobarea producătorului pentru modificări sau se reintegrează testele care nu corespund;
- După obţinerea aprobării se îngheaţă versiunea funcţională care devine validă pentru verificarea şi declararea unităţii apte de serviciu.

Dintre aceste acţiuni, editarea diagramelor de testare şi editarea programului de testare sunt primele pe care le execută inginerul care dezvoltă programul. Dacă diagramele de testare pot fi şi desenate cu creionul sau editate într-un mediu consacrat (Orcad, ViewLogic sau o aplicaţie specializată pentru testare), programul se editează într-un limbaj specializat. În timp s-au dezvoltat limbaje de diverse nivele de complexitate. Dintre acestea amintim limbajul ATLAS (Abreviated Test Leanguage for All Systems) care încearcă să îmbine calităţile descriptive cu cele formale ale unui limbaj de programare.

Test executivul[10] este de obicei o aplicaţie care execută programul de test. Această aplicaţie este în acest caz un interpretor care execută instrucţiile pas cu pas. Există şi situaţia în care programul de testare este integrat cu o parte comună tuturor programelor şi compilate împreună, iar aplicaţia rezultată este specifică, particulară. Mai există şi alte variante de legare a testului de test executiv. Există câteva aplicaţii deosebit de cunoscute cum ar fi Test Stand[121] dezvoltat de National Instruments, ATEasy dezvoltat de Marvin Test Systems, Inc. şi mai ales PAWS[160] dezvoltat de TYX Corp. care este specializat pentru avionică.

O secţiune deosebită pentru sistemele de testare sunt driver-ele. Driverele sunt cele care leagă programele de instrumente şi în final de unitatea testată. De fapt diversitatea acestora este cea care a împiedicat standardizarea echipamentelor de testare automată pentru multă vreme. Consorţiul IVI (Interchangeble Virtual Instruments) a iniţiat definirea unor clase de instrumente generice, a unor proceduri de dezvoltare a drivere-lor. **Driver-ele IVI**[26]-[35][83]-[85][89][102]-[106][133]-[135] au modificat fundamental concepţia despre programele de testare dar standardizarea tuturor instrumentelor şi dezvoltarea de drivere pentru ele reprezintă un efort foarte mare.

Am menţionat **translaţia** unor programe existente ca fiind importantă. Există multe programe de testare validate de o utilizare îndelungată şi care sunt încă foarte utile dar echipamentele vechi fac imposibilă utilizarea lor. Schimbarea echipamentelor duce implicit la noi programe. Translaţia automată a celor existente permite un proces de validare al noilor programe mult mai rapid.

Pe lângă editorul de programe, test executiv, drivere există o serie de aplicaţii ajutătoare care simplifică dezvoltarea programelor de testare. O astfel de aplicaţie este **simulatorul**[53][54] de testare. Există diverse variante de simulatoare. Cele mai simple permit rularea programului cu introducerea manuală a valorilor măsurate. Acest gen de simulator permite verificarea tuturor ramurilor programului, a mesajelor, a formatării raportului. O formă mult mai avansată ar fi verificarea pe un model al unităţii, dar după informaţiile autorului nu există o variantă care să funcţioneze pentru testarea funcţională, modelarea şi existenţa unor drivere care să poată fi legate de model fiind proces complexe.

Generatorul de rapoarte permite formatarea rapoartelor de de testare sub diverse forme şi eventual publicarea acestora. De asemenea există şi posibilitatea ca stocarea rapoartelor să fie realizată tot de generatorul de rapoarte. Dacă există o reţea conectată la mai multe sisteme de testare automate atunci aplicaţia de administrare va fi mult mai complexă legată la o bază de date.

Un subiect sensibil al arhitecturii software pentru sistemele automate de testare este posibilitatea extinderii şi eventual adăugării de noi elemente (instrumente şi drivere pentru ele, noi tipuri de semnale şi chiar noi domenii tehnologice de testare). Dorinţa de a acoperi la modul general toate problemele, a făcut ca noul standard ATLAS pe care subcomitetul IEEE SCCC20 trebuie să-l propună în variantă finală a fost primit circumspect de către industrie fiind considerat dificil de aplicat. Totuşi aplicaţii precum Test Stand permit dezvoltarea de programe într- arhitectură deschisă, implementarea unui sistem bazat pe instrucţiile de bază din ATLAS fiind apropiată de standard. Implementările făcute în mediul PAWS se bazează pe variantele ATLAS subseturile 626, 716[20][22]. În acest caz driverele sunt dezvoltate sub mediu şi deci nu au statutul de drivere IVI. Variantele mai noi de PAWS permit includerea de drivere IVI dezvoltate sub LabWindows/CVI[9] sau alte compilatoare dar numai pentru sistemul de operare Windows. Aceste medii de testare permit adăugarea de module nestadard , definite ca module externe. Aceast gen de structură deschisă a mediilor de testare au opţiunea ca atunci când apar domenii tehnologice noi, până la conscrarea şi implicit standardizarea acestora, testele corespunzătoare să fie adăugate ca module externe.

Autorul îşi propune să analizeze câteva variante pe care le-a aplicat la testarea unor unităţi. Cum în timp s-au folosit mai multe variante, concluziile ne-au permis să

putem opta pentru o variantă care încearcă să păstreze elementele de standardizare dar să simplifice anumite elemente. Din acest motiv vom analiza mai întâi limbajele utilizate.

5.2 Limbaje de programare a testelor.

Limbajul de programare concentrează simbolic arhitectura hardware şi software a sistemului de testare automată. Din acest motiv inventarea unui limbaj pentru testare s-a dovedit o sarcină deosebit de dificilă. În timp limbajele de programare folosite au evoluat odată cu celelalte entităţi ale sistemului. Dat fiind că instrumentele (inclusiv căile de conectare), conectorul unităţii, precum şi acţiunile executate asupra acestora devin elemente de program, s-a dovedit a fi mult mai dificilă crearea unui limbaj pentru testare. Multe din acţiunile executate la momentul testării au fost ascunse de programul propriu-zis, ele fiind implicite şi specifice echipamentului sau integrate în codul aplicaţiilor, invizibile pentru proiectantul de test.

Astfel deosebim mai multe tipuri de limbaj. **Limbajul de nivel mai scăzut** care este specific unui producător de echipament foarte asemănător cu BASIC-ul care include în vocabular şi comenzile spre instrumente. În acest caz extinderea echipamentului este dificilă şi necesită integrarea noilor comenzi prin modificarea executabilelor. Proiectantul de test operează în acest caz cu comenzi legate de echipamentul pe care este instalat test executivul asociat. Un astfel de limbaj este limbajul dezvoltat de compania RADA Electronic Industries Ltd[8].. Editorul de programe în acest caz are şi sarcina de a verifica sintactic programul.

Structura unui astfele de program este construită pe un sistem de graf prezentat în figura 5.2:

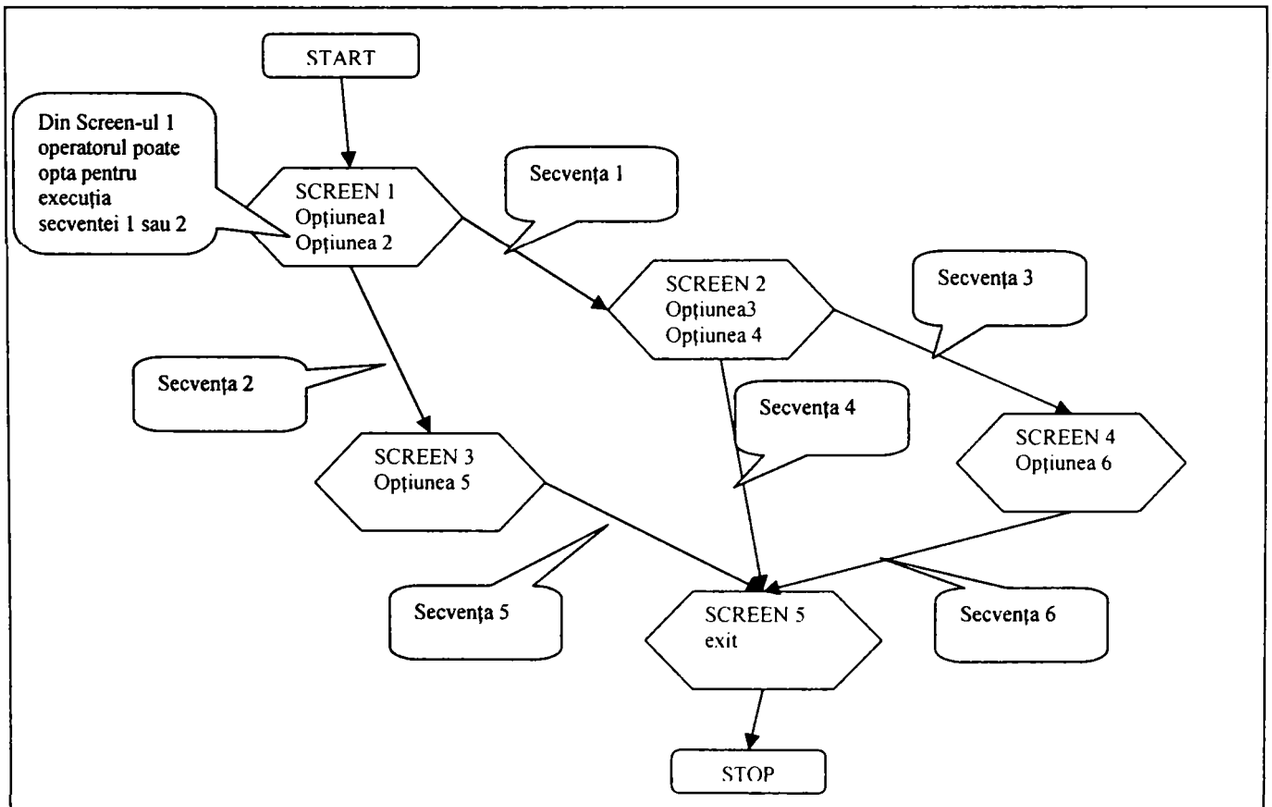


Fig. 5.2 graful ecranelor şi secvenţelor pentru un program scris în TPG.

După cum se remarcă proiectantul are la dispoziţie blocuri de decizie numite screen-uri (ecrane) unde operatorul are afişate mesaje şi poate selecta opţiuni. Selectarea unei opţiuni însemnă transferul la alt screen după executarea unei secvenţe. Secvenţele sunt de fapt programele executate de testor.

Se prezintă în figura 5.3 o secvenţă de testare a tensiunilor sursei de alimentare de la unitatea FMC (Flight Management Computer):

INSTR.	PARAM 1	PARAM 2	PARAM 3	PARAM 4	TOL
[					
'	power test - verify the values of the dc power sup				
]					
[					
P	POWER SUPPLY TEST				
]					
LOC_LINE	3	2			
WR_LINE	POWER SUPPLY TEST				
WAIT_SEC	1				
M_VDC	J2-8	GND-J2-8	REG	28V	
JMP	FAIL	LINE	PWRTST-1		
M_VDC	J2-6	GND-J2-6	REG	19.5V	
JMP	FAIL	LINE	PWRTST-2		
M_VDC	J2-7	GND-J2-7	REG	-19.5V	
JMP	FAIL	LINE	PWRTST-2		
M_VDC	J2-2	GND-J2-2	REG	5V	
JMP	FAIL	LINE	PWRTST-1		
M_VDC	J2-3	GND-J2-3	REG	15V	
JMP	FAIL	LINE	PWRTST-1		
M_VDC	J2-5	GND-J2-5	REG	-15V	
JMP	FAIL	LINE	PWRTST-1		
JMP	UNC	LINE	LEAVE-PWT		
:	PWRTST-1				
LOADF	FATAL FAIL	1			
JMP	UNC	SEQ	FATAL F		
:	PWRTST-2				
LOADF	FATAL FAIL	2			
JMP	UNC	SEQ	FATAL F		
:	LEAVE-PWT				
[					
P	POWER SUPPLY TEST PASS				
]					
LOC_LINE	3	2			
WR_LINE	POWER SUPPLY TEST PASS				

comentariu

Comentariu Tipărit

Mesaj pentru operator

Măsurătoare de tensiune între J2-8 şi GND-J2-8 cu valoarea în REG şi toleranţa în registrul 28V

Instrucţie de salt condiţionat

Încărcarea registrului FATALFAIL cu valoarea 1

Fig. 5.3 Secvenţă scrisă în TPG

Secvenţa este alcătuită din linii de program. O linie are o instrucţie, patru câmpuri pentru parametrii şi un câmp pentru toleranţe. Dacă analizăm linia care măsoară tensiunea remarcăm că aceasta execută mai multe operaţii: face conexiunile, setează instrumentul pentru tipul de măsurătoare, execută măsurătoarea, depune valoarea într-un registru, o compară cu toleranţele definite de un alt registru (valoarea acestora nu se vede) şi modifică un fanion care semnalează dacă testul a trecut sau nu. Multe din aceste operaţiuni sunt ascunse proiectantului, altele trebuie cunoscute pentru ca testul să păstreze caracteristicile unui test funcţional. De asemenea din acest exemplu nu rezultă posibilitatea utilizării unor instrumente diverse. De fapt instrumentul este HP 34401 şi nu poate fi folosit altul pentru că test executivul are driver-ul înglobat.

Un alt exemplu este utilizarea unor limbaje consacrate (ex. BASIC) , iar legătura cu instrumentele se face prin apelarea unor subrutine (care de fapt sunt driver-ele). In realitate acesta nu este cazul unui limbaj de testare, nu avem instrucţii specifice domeniului.

Limbajele au evoluat spre variante mult mai complexe, de nivel înalt care permit o abordare unitară a programelor de testare. Un astfele de limbaj este limbajul ATLAS. Limbajul ATLAS este în realitate rezultatul eforturilor de creere a unui limbaj de nivel înalt, orientat spre semnale şi care îndeplineşte simultan cele două deziderate:

- formalizarea descrierii testelor;
- independenţa de echipament;

Limbajul ATLAS este descris de către standardul ARINC 626 (rev. 3)[20] sau IEEE 716-1995 (C/ATLAS)[22].

5.2.1 Limbajul ATLAS[139][140][141][142] (Abreviated Test Language for All Systems) pentru testare modulară.

Specificaţia de testare de tip ATLAS cuprinde un set de documente care descriu testele necesare pentru o unitate într-o manieră independentă de echipamentul de testare. Structura specificaţei de tip ATLAS cuprinde trei entităţi: structura programului ATLAS, structura modulului ATLAS şi structura modulelor non-ATLAS.

Structura unui program ATLAS cuprinde următoarele părţi:

- directiva BEGIN ATLAS;
- structura de tip preambul program;
- directiva COMMENCE MAIN;
- structura procedurală principală;
- directiva TERMINATE ATLAS;

Structura unui modul ATLAS cuprinde:

- directiva BEGIN ATLAS MODULE;
- preambul modul;
- directiva TERMINATE ATLAS MODULE;

O structură non-ATLAS are aceaşi compoziţie ca o structră de modul ATLAS, cu observaţia că aceasta este dependentă de platfomă şi este scrisă într-un alt limbaj.

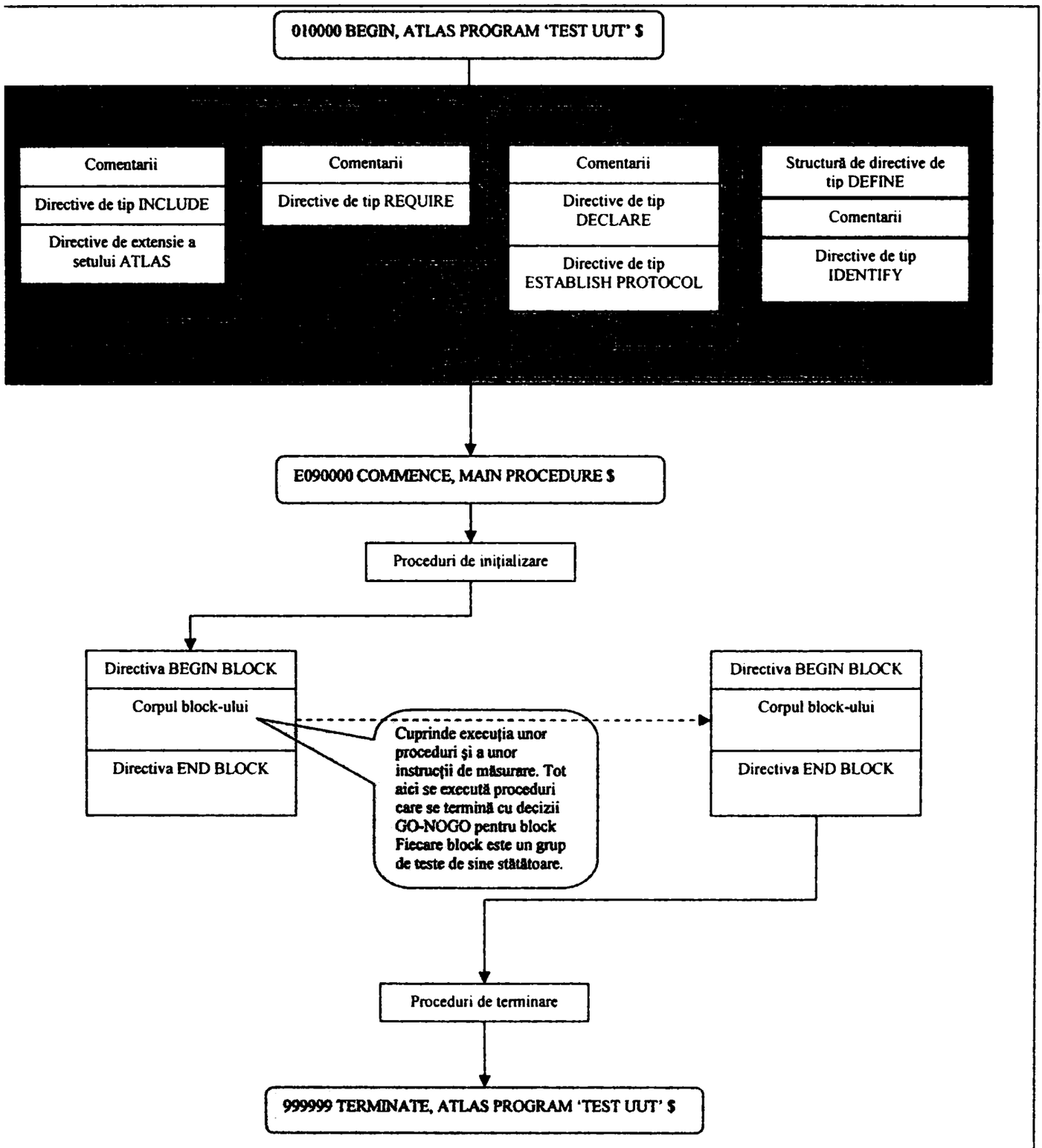


Fig. 5.4 Exemplu de structură a unui program ATLAS

După cum se remarcă în exemplul din figura 5.4 preambulul unui program ATLAS cuprinde o serie de acţiuni care definesc contextul testului.

Prin directivele de tip INCLUDE se definesc mai multe module ATLAS care pot face parte din programul respectiv. O proprietate a acestei directive este posibilitatea de extindere a limbajului cu noi subseturi sau cu noi sbstantive.

Directiva de tip REQUIRE este specială în structura programului. Ea defineşte practic mulţimea senzorilor şi stimulilor necesari pentru testare, sau altfel spus defineşte instrumentele şi resursele necesare pentru test. Se prezintă o astfel de instrucţie care defineşte necesitatea unei surse de tensiune DC.

```
001000 REQUIRE, '+28-VDC-POWER-SUPPLY', SOURCE, DC SIGNAL,
      CONTROL,
        VOLTAGE +28.0 V ERRLMT +-0.1 V,
      LIMIT,
        CURRENT MAX 2 A,
      CNX HI LO $
C$
```

După cum se remarcă instrucţia are următoarea structură:

- index : 001000;
- cuvânt cheie: REQUIRE;
- nume semnal: '+28-VDC-POWER-SUPPLY';
- tip de semnal: SOURCE, DC SIGNAL;
- capabilităţi: CONTROL,
  VOLTAGE +28.0 V ERRLMT +-0.1 V;
- limite: LIMIT,
  CURRENT MAX 2 A,
- conectori: CNX HI LO;
- terminator instrucţie: $;

Directiva de tip DECLARE este folosită pentru a declararea variabilelor folosite de program:

```
DECLARE, VARIABLE, 'AC-VOLT'  IS DECIMAL $
```

Directiva de tip ESTABLISH PROTOCOL defineşte în ansamblul său o conexiune pe care se schimbă informaţii după un anumit protocol:

```
020000 ESTABLISH, BUS-PROTOCOL 'XMT-BUS',
       SPEC 'ARINC-429',
       TEST-EQUIP-ROLE MASTER,MONITOR,SLAVE,
       TEST-EQUIP-MONITOR DATA,
     BUS-MODE TALKER-LISTENER,
       TALKER TEST-EQUIP,
       LISTENER UUT,
       DATA (STRING(32) OF BIT),
       STANDARD PRIMARY BUS,
     CNX TRUE COMPL $
```

Şi aici identificăm mai multe elemente:

- index : 020000;

- cuvânt cheie: ESTABLISH;
- numele busului şi protocolului: BUS-PROTOCOL 'XMT-BUS';
- specificaţia care defineşte protocolul: SPEC 'ARINC-429';
- capabilităţile instrumentelor de comunicaţie:

```
TEST-EQUIP-ROLE MASTER,MONITOR,SLAVE,
TEST-EQUIP-MONITOR DATA,
BUS-MODE TALKER-LISTENER,
TALKER TEST-EQUIP,
LISTENER UUT,
DATA (STRING(32) OF BIT),
STANDARD PRIMARY BUS,
```

- pinii de conexiune: CNX TRUE COMPL ;
- terminator de instrucţie: $;

Directiva de tip DEFINE este utilizată pentru definirea detaliată şi specifică a unui eveniment cum este de exeplu transmisia a două cuvinte folosind protocolul definit mai sus şi denumit 'XMT-BUS':

```
031500 DEFINE, 'SEND-W', EXCHANGE, PROTOCOL 'XMT-BUS',
                    BUS-MODE TALKER-LISTENER,
                    TALKER TEST-EQUIP,
                    LISTENER UUT,
                    DATA 'SEND-WORD'(1 THRU 2) $
```

Tot la categoria DEFINE se găsesc şi procedurile comune (subrutine) care practic sunt declarate şi definite aici după cum se vede din exemplul următor:

```
C ********************************************************************** * $
C *  PROCEDURE: 'DATE-TIME'                                              * $
C *---------------------------------------------------------------------- * $
C *  PURPOSE: READS THE DATE USING THE ATLAS DATE FUNCTION AND           * $
C *        RETURNS THE DATE IN THE FORM YY/MM/DD AND RETURNS             * $
C *        THE TIME IN THE FORM HH:MM:SS                                 * $
C *---------------------------------------------------------------------- * $
C *  GLOBAL VARIABLES USED:                                              * $
C *    -VAR-       -TYPE-          -FUNCTION-                            * $
C *    'D'         STRING(15) OF CHAR   RETURNS THE DATE                 * $
C *    'T'         STRING(15) OF CHAR   RETURNS THE TIME                 * $
C *                                                                      * $
C *  PROCEDURES: NONE                                                    * $
C *                                                                      * $
C ********************************************************************** * $
C$
 045000 DEFINE, 'DATE-TIME',
       PROCEDURE RESULT( 'D', 'T' IS STRING(15) OF CHAR) $
       DECLARE, VARIABLE, 'TODAY' IS STRING(15) OF CHAR    $
       DECLARE, VARIABLE, 'D1', 'T1' IS STRING(10) OF CHAR $
       DECLARE, VARIABLE, 'D2', 'T2' IS STRING(12) OF CHAR $
       CALCULATE, 'TODAY' = DATE $
       CALCULATE, 'D1' = COPY('TODAY', 1, 6 ),
          'D2' = INSERT(C'/', 'D1', 2),
          'D'  = INSERT(C'/', 'D2', 5) $
       CALCULATE, 'T1' = COPY('TODAY', 7, 6 ),
          'T2' = INSERT(C':', 'T1', 2),
          'T'  = INSERT(C':', 'T2', 5) $
       END, 'DATE-TIME' $
 C$
```

Procedurile din secţiunea DEFINE a programului sunt rutine globale care pot fi apelate de oriunde în program.

După partea de preambul care descrie practic contestul de testare propriu-zis, urmează secţiunea principală (MAIN) care conţine testele funcţionale grupate în blocuri. Blocurile pot avea şi ele o parte de preambul după care urmează programul propriuzis, după cum se poate vedea şi din exemplu:

```
C ************************************************************************ $
C *      BLOCK 1.3  POWER-UP TESTS                                       * $
C ************************************************************************ $
C$
C$
     OUTPUT, TEXT, FROM
       'SPACE-BEFORE',
       C'\LF\',
       C' --- BLOCK 1.3  POWER-UP TESTS ---\LF\',
       'SPACE-AFTER' $
C$
     OUTPUT, TEXT TO 'LOG-FILE', FROM
       C'\LF\',
       C' --- BLOCK 1.3  POWER-UP TESTS ---\LF\\LF\' $
C$
 130000 APPLY, DC SIGNAL USING '+28-VDC-POWER-SUPPLY',
      VOLTAGE +28.0 V ERRLMT +-0.1 V,
      CURRENT MAX 2 A,
      CNX HI J1-10
        LO EARTH $
C$
     WAIT FOR, TIME 5 SEC  $
C$
 130010 VERIFY, (CURRENT), DC SIGNAL USING 'DC-AMMETER',
           LE 0.5 A,
           CURRENT RANGE 0 A TO 2 A,
           CNX VIA J1-10A $
C$
     PERFORM, 'CHECK-NOGO' (C'STATNO 130010 ', 0.05, 'MEASUREMENT', 0.5, C'A') $
C$
 130020 VERIFY, (VOLTAGE), DC SIGNAL USING 'DC-VOLTMETER',
       UL 28.5 V  LL 27.5 V,
       VOLTAGE RANGE 0V TO 30V,
       CNX  HI  J1-10
          LO  EARTH  $
C$
     PERFORM, 'CHECK-NOGO' (C'STATNO 130020 ', 27.5, 'MEASUREMENT', 28.5, C'VOLT') $
C$
 130030 VERIFY, (VOLTAGE), DC SIGNAL USING 'DC-VOLTMETER',
       UL 28.5 V  LL 27.5 V,
       VOLTAGE RANGE 0V TO 30V,
       CNX  HI  J1-16
          LO  EARTH  $
C$
     PERFORM, 'CHECK-NOGO' (C'STATNO 130030 ', 27.5, 'MEASUREMENT', 28.5, C'VOLT') $
C$
C ---------- Initiate the NORMAL MODE for ACP ---------- $
C$
     ENABLE, EXCHANGE-CONFIGURATION USING 'XMT',
       PROTOCOL 'XMT-BUS',
     CNX TRUE J1-17 COMPL J1-18 $
C$
     WAIT FOR, TIME 0.1 SEC  $
C$
     CALCULATE, 'SEND-WORD'(1) =  'SCC-ACP-CT-1' $
     CALCULATE, 'SEND-WORD'(2) =  'SCC-ACP-CT-2' $
C$
     DO, EXCHANGE USING 'XMT', TEST-EQUIP-ROLE MASTER,
```

```
      (EXCHANGE 'SEND-W',
        TEST-EQUIP-ROLE MASTER,
        DELAY 40 MSEC),
     PROCEED $
C$
C---------- 5 VAC POWER-SUPPLY ---------- $
C$
     APPLY, AC SIGNAL USING '5-VAC-POWER-SUPPLY',
          VOLTAGE 5V,
          FREQ 400HZ,
          CURRENT MAX 0.5A,
          CNX HI J1-22
            LO J1-23 $
C$
 130040 VERIFY, (VOLTAGE), AC SIGNAL USING 'AC-VOLTMETER',
       UL 5.2 V  LL 4.5 V,
       VOLTAGE RANGE 0V TO 6V,
       CNX  HI  J1-22
          LO  J1-23  $
C$
     PERFORM, 'CHECK-NOGO' (C'STATNO 130040 ', 4.5, 'MEASUREMENT', 5.2, C'VOLT') $
C$
 130050 VERIFY, (CURRENT), AC SIGNAL USING 'AC-AMMETER',
     UL 0.5A LL 0.005A,
     CURRENT RANGE 0A TO 1A,
     CNX VIA J1-22A $
C$
     PERFORM, 'CHECK-NOGO' (C'STATNO 130050 ', 0.05, 'MEASUREMENT', 0.5, C'A') $
C$
 199999 END, BLOCK '1.0 PRE-POWER/POWER-UP TESTS' $
C$
```

Exemplul de mai sus elaborat de autor face parte din programul de testare al unităţii ACP (ANEXA 2). *Acest program a fost validat de producătorul avionului* şi din punct de vedere al testului, dar şi ca document descriptiv pentru această unitate. Acest sub-bloc verifică mărimile electrice referitoare la alimentarea unităţii. Procedura 'CHECK-NOGO' apare de fiecare dată când se fac măsurători care se publică în raport. Unitatea are 9 blocuri. Unele blocuri au mai multe subblocuri.

Instrucţia ATLAS este construită din punct de vedere sintactic astfel:

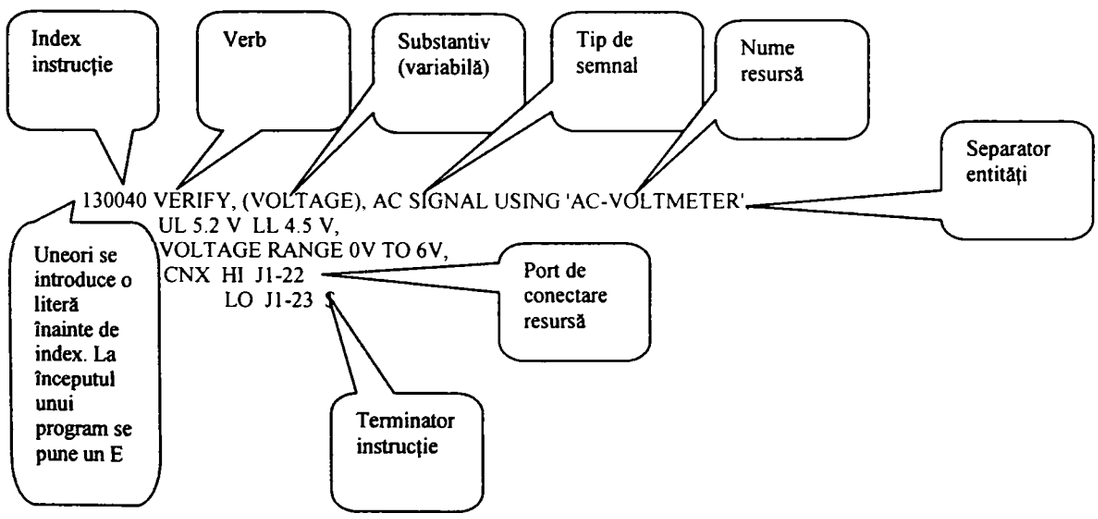


Fig. 4.5 structura unei instrucţii ATLAS

Enumerăm cele mai folosite verbe în ATLAS :

- DEFINE
- IDENTIFY
- REQUIRE
- APPLY
- REMOVE
- MEASURE
- MONITOR
- VERIFY
- READ
- INITIATE
- SETUP
- CONNECT
- DISCONNECT
- ARM
- FETCH
- CHANGE
- RESET
- WAIT_FOR
- STIMULATE
- SENSE
- PROVE
- DO
- EXCHANGE

La acestea se mai pot adăuga extensii pentru comunicaţie sau alte domenii tehnologice. Lista de mai sus cuprinde cele mai utilizate verbe, standardul ARINC 626 fiind mult mai întins. Dorim să subliniem că acţiunile verbelor pot fi simple sau complexe. Ne vom referi la instrucţiile orientate spre semnal. Acestea constau în acţiuni simple sau multiple asupra surselor, senzorilor şi sarcinilor legate de semnalele la conectorul unităţii testate. Resursele sunt considerate virtuale. Ele pot avea diverse stări:

- UNALLOCATED: resursa nu este folosită pentru nici funcţie a unităţii testate şi este disponibilă;
- SET: resursa alocată dar neconectată la unitate;
- PRPARED: resursa conectată la unitate;
- ARMED: stare validă doar pentru senzori şi implică disponibilitatea acestuia de a face o măsurătoare;
- UNARMED: indică faptul că senzorul nu este gata să măsoare.
- READY: toate condiţiile necesare pentru ca o resursă să-şi schimbe starea spre staraea dorită sunt îndeplinite;
- APPLIED: stare aplicată stimulilor şi sarcinilor care specifică aplicarea semnalului necesar unităţii;
- MEASURED: senzorul a executata măsurătoarea;
- COMPLETED : senzorul a executat măsurătoare, este în continuare conectat la unitate şi aşteaptă instrucţiuni.

Am definit aceste stări pentru a prezenta următoarele diagrame care definesc aplicarea unor semnale sau măsurarea lor folosind verbe simple şi multiple în condiţii de sincronizare cu un evenimet sau fără sincronizare:

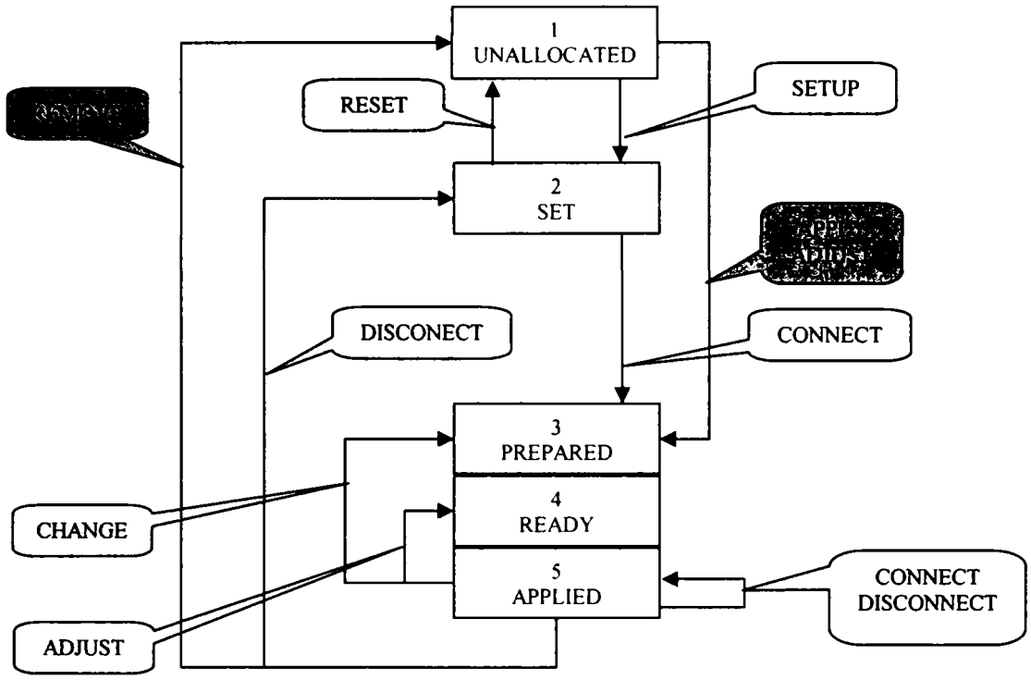


Fig. 5.6 Digrama aplicării unui stimul sau sarcină nesicron cu un eveniment folosind verbe simple şi multiple

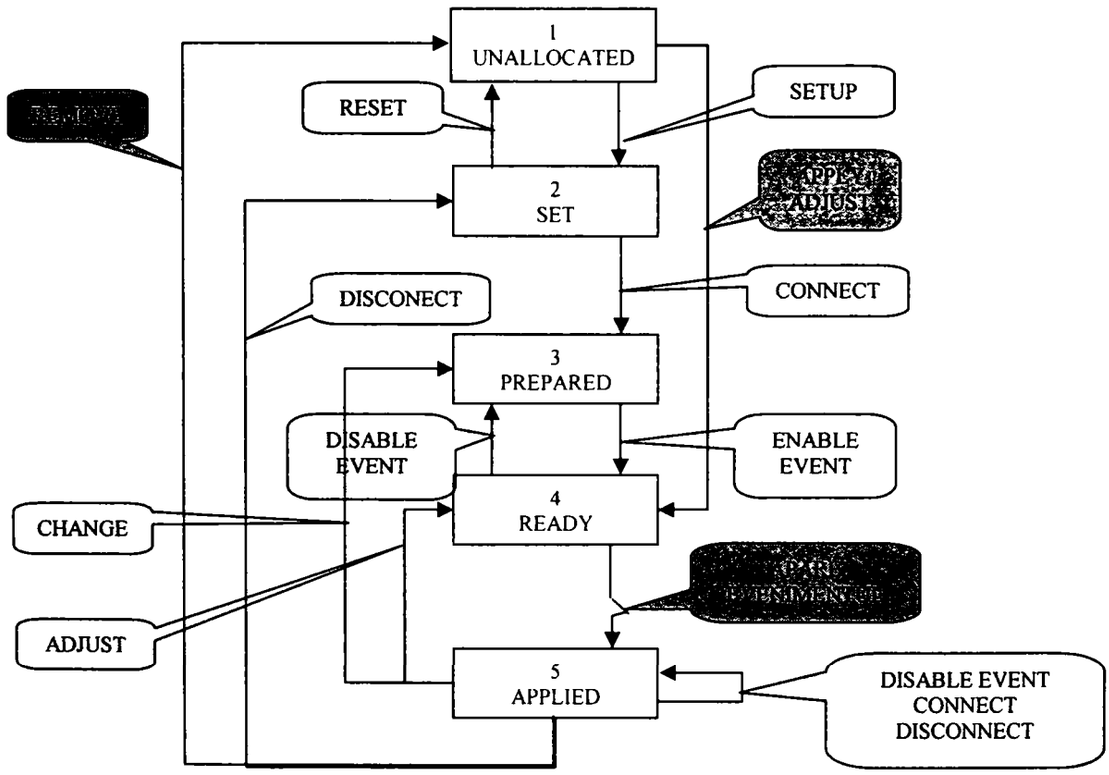


Fig. 5.7 Digrama aplicării unui stimul sau sarcină sicron cu un eveniment folosind verbe simple şi multiple

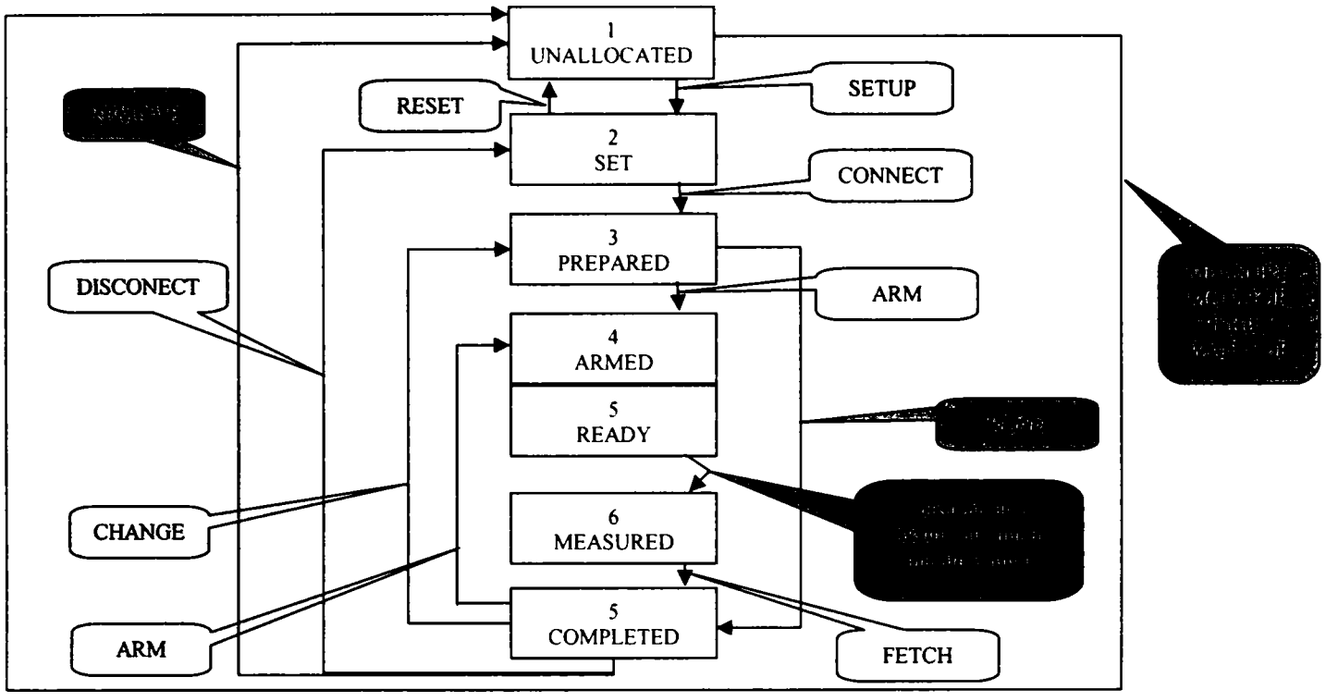


Fig. 5.8 Digrama măsurării cu un senzor, nesicron cu un eveniment folosind verbe simple şi multiple

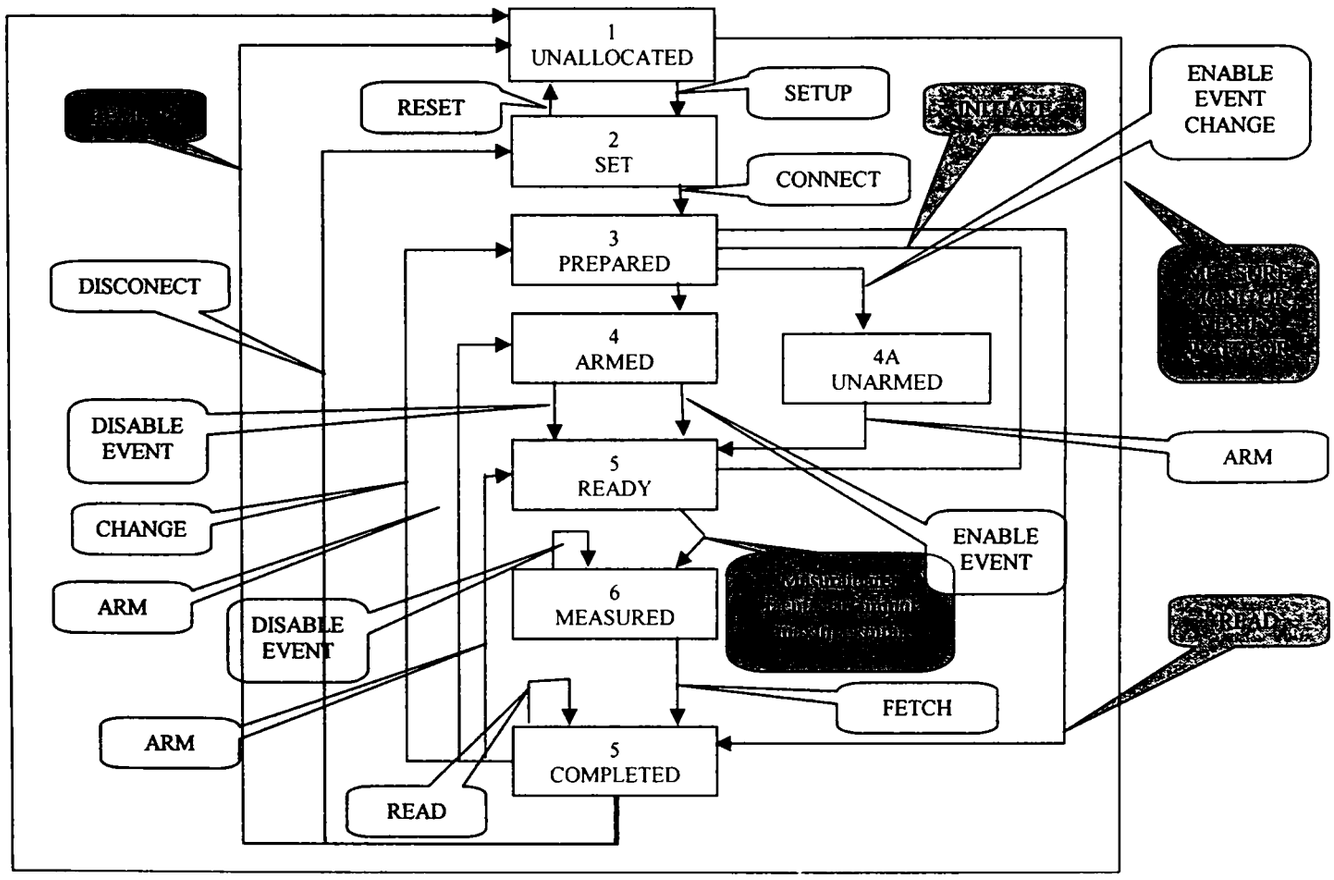


Fig. 5.9 Digrama măsurării cu un senzor, sicron cu un eveniment folosind verbe simple şi multiple

Am făcut această prezentare detaliată a programului ATLAS pentru a explica nu numai structura acestuia bazată pe exemple din activitatea autorului dar şi pentru a prezenta dilema care o crează. Acest limbaj a fost creat iniţial pentru a descrie mai formal testele funcţionale la care este supusă o unitate. O dată cu automatizarea procesului de testare a apărut şi necesitatea unui limbaj pentru scrierea programelor de testare. Dar acest limbaj exista şi se folosea pentru descriere. Trecerea de la un limbaj descriptiv la un limbaj de programare se dovedeşte în cazul acesta (când o parte din entităţile cu care acesta operează sunt elemente hardware) o sarcină mult mai dificilă.

Primul limbaj prezentat este un limbaj de programare (de nivel BASIC) care nu poate fi folosit pentru descriere în timp ce ATLAS este un limbaj descriptiv care se încearcă a fi folosit pentru programare. Acest lucru a reuşit, dar numai parţial cu multe simplificări şi particularizări, dar mai ales cu mutarea unor informaţii în alte fişiere pe care mediul le foloseşte în detrimentul generalităţii programului.

5.3 Drivere pentru instrumente.[26]-[35][84][102]-[104][106][123]

Pc-ul aşa cum am văzut în capitolul II se leagă de instrumente prin diverse magistrale şi protocoale. Programul, care face legătura între programul de testare şi instrumente, fie prin funcţii ale sistemului de operare, fie prin funcţii care operează direct asupra porturilor, se numeşte driver. Dată fiind diversitatea instrumentelor şi resurselor hardware crearea unui concept unitar la acest nivel al driverelor este foarte dificil. Cea mai directă abordare este crearea unei subrutine în programul de test care să comande direct portul computerului şi mai departe instrumentul. Această abordare exclude posibilitatea schimbării instrumentelor pe staţie şi portabilitatea aplicaţiei. Există foarte multe programe de testare care au fost concepute astfel.

Pentru un mediu de testare care utilizează un interpretor de comenzi legătura dintre programul de testare şi instrument poate fi prezentată astfel:

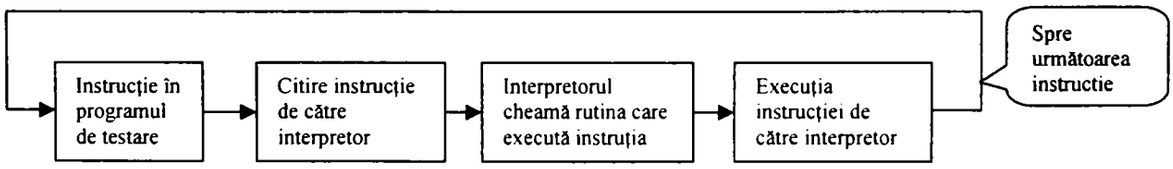


Fig. 5.10 Ciclul simplificat al execuţiei unei instrucţii de către un interpretor

După cum se remarcă în figura 5.10 interpretorul este o aplicaţie care conţine toate operaţiile de testare cu excepţia editării programului. Este genul de program care a funcţionat sub sistemul de operare DOS.

Mediul de testare PAWS al companiei TYX foloseşte ca limbaj de programare, limbajul ATLAS. Modul în care este legat programul de testare de instrumente este complex. Prezentăm în figura 5.11 modul cum este structurat acest mediu precum şi legăturile dintre dversele componente ale mediului:

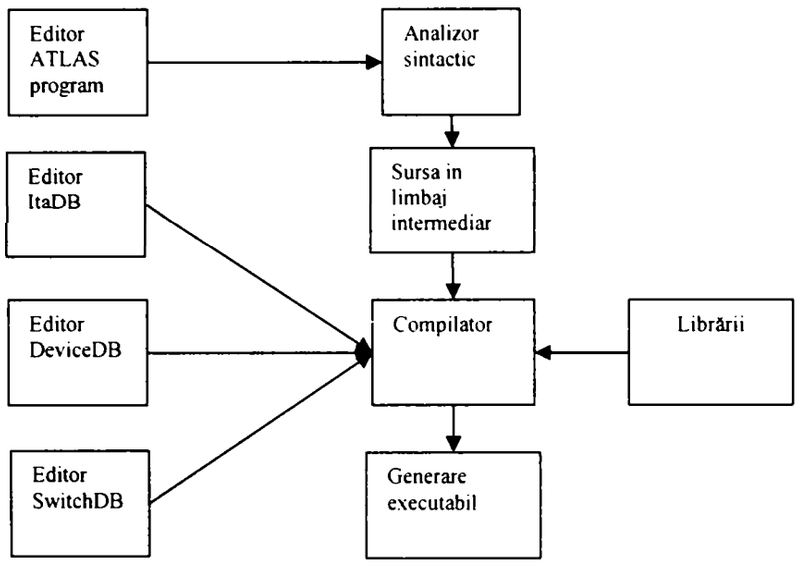


Fig. 5.11 Digramă simplificată a unui mediu de testare care utilizează ATLAS ca limbaj

ItaDb este un fişier care descrie interfaţa cu unitatea, SwitchDB descrie matricea de conexiuni iar DeviceDB conţine driverele de nivel înalt. Acestea se leagă cu funţii de nivel mai scăzut care prin intermediul sistemului de operare acţionează asupra porturilor sistemului. Aceste funcţii se găsesc gata compilate în librării.

Fişierul DeviceDb conţine toate driverele de nivel înalt. Fiecare instrument are mai multe zone alocate. Structura fişierului este prezentată în figura 5.12 . Se remarcă zona de declararea a funcţiilor pe care dezvoltatorul de driver le are la dispoziţie, zona dinamică unde se declară variabilele utilizate şi funcţiile noi cu care se operează şi zona statică în care se declară capabilităţile instrumentului precum şi verbele simple. O operaţie pe care o face mediul înainte de compilare este alocarea instrumentelor. Alocatorul are misiunea de a identifica instrumentul potrivit dintre resursele descrise în DeviceDB pentru un instrument din secţiunea de REQUIREMENT a programului ATLAS. Evident se întâmplă foarte des ca alocatorul să găsească un instrument nepotrivit sau pur şi simplu să nu identifice nici un instrument potrivit. În aceste cazuri se face o alocare forţată a instrumentului specificând într-un alt fişier asocierea funcţiei din DeviceDB (FNC) cu instrumentul din ATLAS. Procesul de alocare este greoi şi consumă mult timp din procesul de integrare a programului de testare. Un rol separat îl are driverul de conectare. Deoarece conform figurii 5.11 în final se generează un executabil toate conexiunile trebuie definite în programul de testare fiind imposibil de utilizat conexiuni variabile în funcţie de diversele rezultate ale testului. Alocatorul face alocarea înainte de compilare şi lucrează numai cu valori precise ale pinilor. Dar foarte multe programe ATLAS au fost scrise cu scop descriptiv şi conexiunile sunt adesea variabile şi modificate folosind subrutine sau bucle. Acestea se modifică prin explodare, adică fiecare subrutină se rescrie de câte ori este necesară cu valorile specifice momentului. Acest lucru se întâmplă mai ales la rutinele de comunicaţie folosite pentru mai multe canale la care semnalele sunt mutate de la o pereche de pini la alţii. *In Anexa 3 se prezintă un fişier DeviceDB funcţional pentru câteva instrumente (DMM, Audio Analyzer, Power supply) inclusiv partea de conexiuni la a cărui dezvoltare a participat şi autorul.*

```
def, fnc, Cat    ==      1;      ** ConcatenateStrings
def, fnc, Cmp    ==      2;      ** Compare Strings
.
.
def, fnc, CkSum  ==     57;      ** ClearGetCheckSum
```

Se declară funcţiile interne implementate în librării

```
dcl
{
        txt DMM_Buffer 80;
        int DMM_itemp;
}
```

ZONA DINAMICĂ
Se declară variabilele cu care se operează

```
def, mac, DMM_TalkBus();
{
        Apnd("\r\n",DMM_Buffer);
        Talk(DMM_Buffer);
        Rtn(DMM_itemp);

        if(DMM_itemp<0)
                DMM_BusDeviceError("BUS TALK ERROR");

        Cpy("",DMM_Buffer);
}
```

ZONA DINAMICĂ
Se descriu funcţiile folosite de driver

```
begin DEV DMM using DMMhp;
        reset DMM_ResetDevice();
        init DMM_Inx();

        cnx hi DMM1_HI, lo DMM1_LO;

begin; ** dc & resistance functions - NPLC aplicable
  begin; **voltage functions
   setup DMM_Setup( $FNC, voltage $RNG:0, 2);
   fetch DMM_Fetch( voltage result);
   sensor (voltage) dc signal;
   capability
   {
    test-equip-imp-magnitude min 1E6 ohm;
    ac-comp max 0.1 v;
   }
   begin FNC=11;
        control
        {
         voltage range -1.0e3 v to 1.0e3 v by 1.0e-3 v
errlmt +-0.05 pc RNG=8,
                range -1.0e2 v to 1.0e2 v by 1.0e-4 v
errlmt +-0.05 pc RNG=6,
                range -1.0e1 v to 1.0e1 v by 1.0e-5 v
errlmt +-0.05 pc RNG=5,
                range -1.0e0 v to 1.0e0 v by 1.0e-6 v
errlmt +-0.05 pc RNG=3,
                range -0.1e0 v to 0.1e0 v by 1.0e-7 v
errlmt +-0.05 pc RNG=2;
                 **  6 1/2 digits resolution
        }
   end;
end; ** voltage funct
```

ZONA STATICĂ
Se descriu capabilităţile instrumentului după care se face asocierea cu instrumentul din zona de REQUIREMENTS din ATLAS şi inclusiv alocarea şi se definesc verbele simple.

Fig. 5.12 Secţiunile care definesc un driver în fişierul DeviceDB (exemplul DMM-ului)

5.4 Drivere IVI[26]-[35][149][179][180][172]

Instrumentele virtuale interschimbabile reprezintă un progres semnificativ în această tentativă de standardizare a echipamentelor şi proceselor de testare. Instrumentul virtual interschimbabil este ceea ce am denumit mai devreme instrument generic. Instrumentul generic pentru a i se ataşa o entitate software a fost definit ca o clasă de instrumente care au proprietăţi şi metode prin care aceste proprietăţi pot fi schimbate. Driver-ul pune la dispoziţia aplicaţei de nivel superior o interfaţă prin care se pot utiliza metodele respective pentru modificarea unor parametrii. Aceste metode sunt aceleaşi pentru toate instrumentele din clasa respectivă care au aceleaşi capabilităţi, dar obligatoriu capabilităţile de bază.

Capabilităţile unei clase de instrumente cuprind următoarele secţiuni:

- prezentare generală: descrie scopul şi modul de folosire a grupului de capabilităţi;
- atribute: defineşte atributele care fac parte din grup. Se defineşte numele fiecărui atribut, tipul de date folosit, accesul la atribut (scriere,citire) , o scurtă descriere, domeniul de valori;
- funcţii: prezintă funcţiile din grup. Pentru fiecare se defineşte numele, descrierea, parametrii de intrare, parametrii de ieşire;
- modelul de comportare: descrie legătura dintre atributele şi funcţiile clasei şi comportamentul instrumentului;
- note de complianţă: specifică eventualele excepţii ale clasei care trebuie îndeplinite.

Din punct de vedere al driver-ului ne interesează atributele şi funcţiile. Pentru o prezentare mai sugestivă vom lua în considerare un driver pentru un DMM care are doar capabilităţile de bază.

Atributele DMM-ului sunt următoarele:

a. FUNCTION

tip de dată	**acces**	**se aplică la**	**coerciţie**	**funcţie de nivel înalt**
ViInt32	R/W	N/A	N/A	ConfigureMeasurement

Numele complet al atributului este: **IVIDMM_ATTR_FUNCTION**
Valorile pe care le poate lua atributul sunt următoarele:
IVIDMM_VAL_DC_VOLTS: setează DMM-ul pentru măsurarea de tensiuni DC;
IVIDMM_VAL_AC_VOLTS: setează DMM-ul pentru măsurarea de tensiuni AC;
IVIDMM_VAL_DC_CURRENT: setează DMM-ul pentru măsurarea curentului DC;
IVIDMM_VAL_AC_CURRENT: setează DMM-ul pentru măsurarea curentului AC;
IVIDMM_VAL_2_WIRE_RES: setează DMM-ul pentru măsurarea rezistenţelor pe 2 fire;
IVIDMM_VAL_4_WIRE_RES: setează DMM-ul pentru măsurarea rezistenţelor pe 4 fire;
IVIDMM_VAL_FREQ: setează DMM-ul pentru măsurarea frecvenţei;
IVIDMM_VAL_PERIOD: setează DMM-ul pentru măsurarea perioadei;
IVIDMM_VAL_TEMPERATURE: setează DMM-ul pentru temperaturii;

b. RANGE

tip de dată	**acces**	**se aplică la**	**coerciţie**	**funcţie de nivel înalt**
ViReal64	R/W	N/A	în sus	ConfigureMeasurement

Numele complet al atributului este: **IVIDMM_ATTR_RANGE**
Valorile pe care le poate lua atributul sunt următoarele:
IVIDMM_VAL_AUTO_RANGE_ON: DMM-ul selectează automat domeniul;
IVIDMM_VAL_AUTO_RANGE_OFF: DMM-ul rămâne pe domeniul cel mai recent;
IVIDMM_VAL_AUTO_RANGE_ONCE: DMM-ul selectează domeniul înaintea următoarei măsurători. Toate măsurătorile care urmează vor fi executate în acelaşi domeniu;

c. RESOLUTION ABSOLUTE

tip de dată	**acces**	**se aplică la**	**coerciţie**	**funcţie de nivel înalt**
ViReal64	R/W	N/A	în jos	ConfigureMeasurement

Numele complet al atributului este: **IVIDMM_ATTR_RESOLUTION_ABSOLUTE**
Valorile pe care le poate lua atributul sunt următoarele:
IVIDMM_ATTR_RESOLUTION_ABSOLUTE: DMM-ul măsoară cu valori absolute;

d. TRIGGER DELAY

tip de dată	**acces**	**se aplică la**	**coerciţie**	**funcţie de nivel înalt**
ViReal64	R/W	N/A	nu	ConfigureMeasurement

Numele complet al atributului este: **IVIDMM_ATTR_TRIGGER_DELAY**
Valorile pe care le poate lua atributul sunt următoarele:
IVIDMM_VAL_AUTO_DELAY_ON: DMM-ul selectează automat întârzierea după declanşare, înainte de fiecare măsurătoare;
IVIDMM_VAL_AUTO_DELAY_OFF: DMM-ul selectează ultima setare pentru toate măsurătorile care urmează;

e. TRIGGER SOURCE

tip de dată	**acces**	**se aplică la**	**coerciţie**	**funcţie de nivel înalt**
ViReal64	R/W	N/A	nu	ConfigureMeasurement

Numele complet al atributului este: **IVIDMM_ATTR_TRIGGER_SOURCE**
Valorile pe care le poate lua atributul sunt următoarele:

IVIDMM_VAL_IMMEDIATE: DMM-ul nu aşteaptă declanşarea, execută imediat măsurătoarea;
IVIDMM_VAL_EXTERNAL: DMM-ul aşteaptă o declanşare externă pentru a părăsi starea de WAIT-FOR-TRIGGER conform diagramei de comportare.
IVIDMM_VAL_SOFTWARE_TRIG: DMM-ul aşteaptă declanşarea softaware;

Atributele de mai sus pot fi modificate prin funcţii. Funcţiile definite pentru clasa de instrumente DMM, grupul capabilităţilor de bază sunt următoarele:

A. Abort

Funcţia abandonează măsurătoarea iniţiată şi DMM-ul se reîntoarce în starea de aşteptare conform modelului de comportare.
Prototipul funcţiei în ANSI C este:

ViStatus IviDMM_Abort (ViSession Vi);

Parametrii de intrare: Vi – instrument handler;
Tipul: ViSession (este un tip folosit de National Instruments);
Funcţia returnează coduri de status (daca acţiunea a reuşit sau coduri de eroare).

B. Configure Measurement

Funcţia configurează atributele comune ale DMM-ului.
Prototipul funcţiei în ANSI C este:

ViStatus IviDmm_ConfigureMeasurement (ViSession Vi, ViInt32 Function, Real64 Range, ViReal64 Resolution);

Parametrii de intrare: Vi – instrument handler;
Function – setează atributul FUNCTION
Range – setează atributul RANGE
Resolution – setează atributul RESOLUTION
Funcţia returnează coduri de status (daca acţiunea a reuşit sau coduri de eroare).

C. Configure Trigger

Funcţia configurează atributele comune de declanşare ale DMM-ului.
Prototipul funcţiei în ANSI C este:

ViStatus IviDmm_ConfigureTrigger (ViSession Vi, ViInt32 TriggerSource, ViReal64 TriggerDelay);

Parametrii de intrare: Vi – instrument handler;
TriggerSource – setează atributul TRIGGER SOURCE
TriggerDelay – setează atributul TRIGGER DELAY
Funcţia returnează coduri de status (daca acţiunea a reuşit sau coduri de eroare).

D. Fetch

Funcţia returnează o măsurătoare pe care o iniţiază funcţia Initiate.
Prototipul funcţiei în ANSI C este:

ViStatusIviDmm_Fetch (ViSessionVi, ViInt32 MaxTimeMilliseconds,
ViReal64 *Reading);

Parametrii de intrare: Vi – instrument handler;
MaxTimeMilliseconds – setează timpul maxim de execuţie a funcţiei;
*Reading – pointer către valoarea măsurată;
Funcţia returnează „Over Range" dacă valoarea domeniului este depăşită sau "Max Time Exceeded" dacă timpul este depăşit.

E. Initiate

Funcţia iniţializează o măsurătoare cu DMM-ul. Funcţia nu verifică statusul instrumentului.
Prototipul funcţiei în ANSI C este:

ViStatus IviDmm_Initiate (ViSession Vi);

Parametrii de intrare: Vi – instrument handler;
Funcţia returnează coduri de status (daca acţiunea a reuşit sau coduri de eroare).

F. Is Over Range

Funcţia determină pentru o măsurătoare cu DMM-ul executată folosind FETCH sau READ dacă pentru valoarea citită există sau nu condiţia de depăşire a domeniului..
Prototipul funcţiei în ANSI C este:

ViStatus IviDmm_IsOverRange (ViSession Vi, ViReal64 MeasurementValue, ViBoolean *IsOverRange);

Parametrii de intrare: Vi – instrument handler;
MeasurementValue – valoarea măsurată
Funcţia returnează TRUE dacă domeniul a fost depăşit sau FALSE dacă măsurătoarea este în domeniu prin valoarea parametrului IsOverRange.
Funcţia returnează coduri de status (daca acţiunea a reuşit sau coduri de eroare).

G. Read

Funcţia iniţializează o măsurătoare cu DMM-ul, aşteaptă reîntoarcerea DMM-ului în starea de IDLE şi returnează valoarea măsurată.
Prototipul funcţiei în ANSI C este:

ViStatus IviDmm_Read (ViSession Vi, ViInt32 MaxTimeMilliseconds,
ViReal64 *Reading);

Parametrii de intrare: Vi – instrument handler;
MaxTimeMilliseconds – setează timpul maxim de execuţie a funcţiei;
*Reading – pointer către valoarea măsurată;
Funcţia returnează coduri de status (daca acţiunea a reuşit sau coduri de eroare).

Acestea sunt atributele şi funcţiile cu care programul de testare operează asupra clasei DMM. Pentru a evidenţia această legătură vom continua cu prezentarea structurii unui driver în general şi cu detalii pentru DMM. După cum se remarcă în figura 5.13 există mai multe legături care se stabilesc pentru comanda unui instrument.

API (application programming interface) este o interfaţă către aplicaţia care utilizează driver-ul. Aceasta expune mai multe funcţii de nivel înalt şi care definesc clasa de instrumente (instrumentul generic).

Tot la acest nivel este o interfaţă grafică folosită de proiectant pentru punerea la punct a driver-ului. Această interfaţă nu este absolut necesară dar este foarte utilă şi în exploatarea ulterioară a driver-ului. Sistemele tot mai complexe care permit construirea unui testor printr-o simplă configurare a softului pot avea şi facilitatea utilizării instrumentelor generice în regim individual sau de sine stătător, situaţie în care această interfaţă este foarte utilă.

Ambele interfeţe operează prin intermediul funcţiilor expuse asupra unor funcţii interne ale driver-ului care la rândul lor interacţionează cu un motor IVI. Acest motor IVI asigură respectarea unor reguli la execuţia unor comenzi către instrument.

VISA (Virtual Instrumentation Software Architecture) asigură utilizarea unor tipuri de date standard la construirea driverelor IVI. Astfel atributele unei clase de instrumente sunt exprimate într-un format unitar.

Driverele din categoria IVI completează proprietăţile driver-elor din categoria plug&play. Cele mai importante caracteristici ale driver-elor sunt :

- interschimbabilitatea (asigură înlocuirea unui instrumet aparţinând unei clase cu altul din aceiaşi clasă;
- simularea (driver-ul returnează şi valori simulate pentru testarea unor programe de testare fără instrumente);
- monitorizarea stării instrumentului evitând astfel comenzi redundante
- permit crearea unei interfeţe grafice cu instrumentul pentru configurarea şi întreţinerea acestuia.

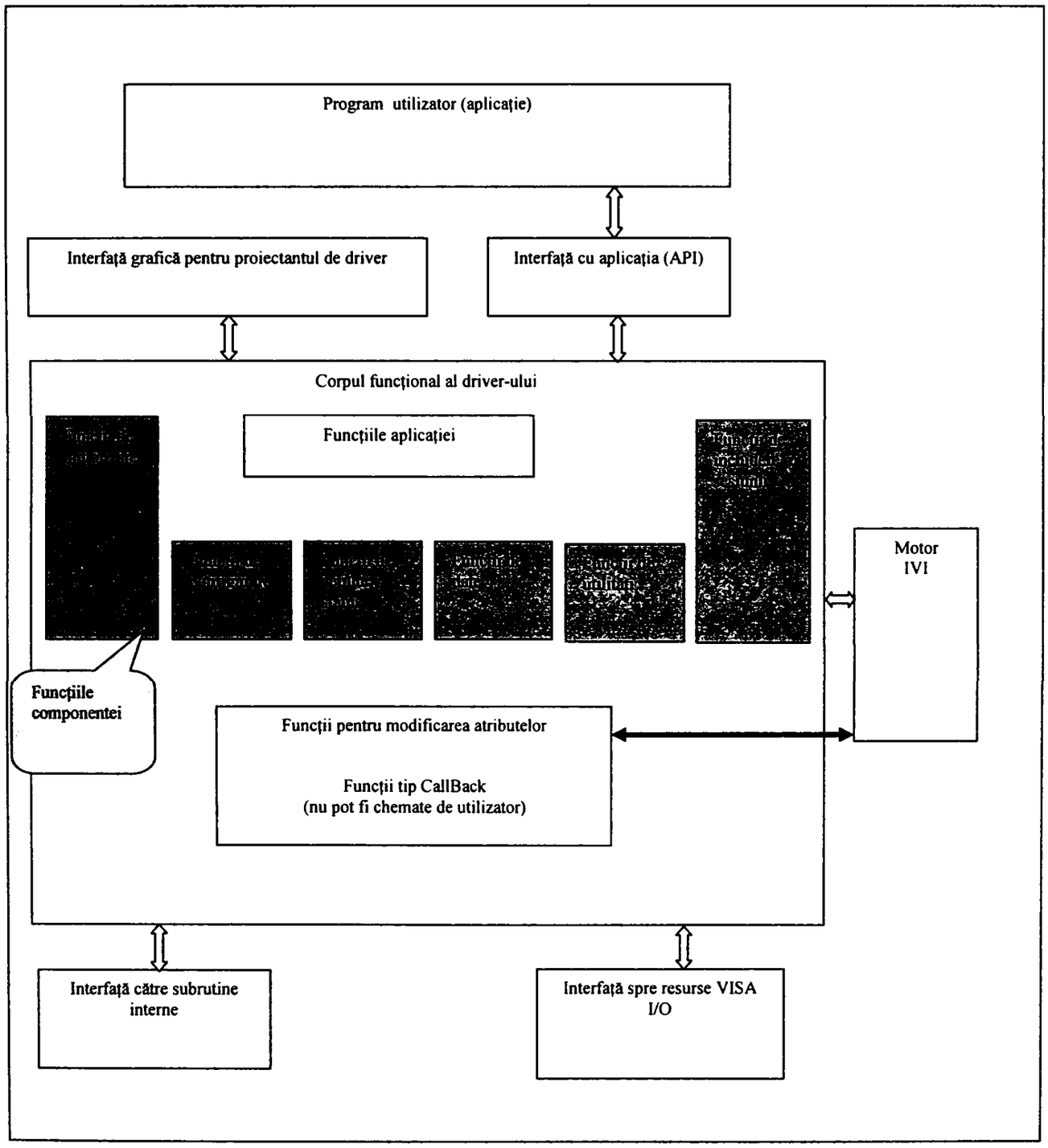


Fig. 5.13 Diagrama unui driver cu componentele sale

Se prezintă în figura 5.14 legătura dintre driver şi motorul IVI (care este un .dll (librărie dinamică) pe 32 de biţi invocat de driver).

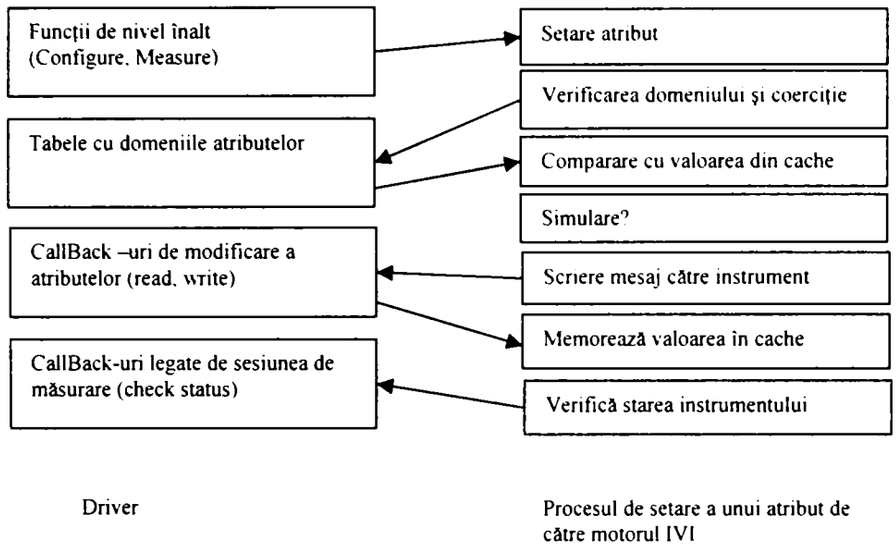


Fig. 5.14 Relaţia dintre driver şi motorul IVI în procesul de setare a unui atribut

Prezentăm în rândurile următoare prototipurile funcţiilor construite pentru un driver IVI al DMM-ului cu ajutorul unui utilitar al companiei National Instruments[123].

```
ViStatus _VI_FUNC  dmm_IviInit (ViRsrc resourceName, ViBoolean IDQuery,
                   ViBoolean reset, ViSession vi);
ViStatus _VI_FUNC  dmm_IviClose (ViSession vi);

  /*- Locking Functions ----------------------------------------------*/

ViStatus _VI_FUNC  dmm_LockSession (ViSession vi, ViBoolean *callerHasLock);
ViStatus _VI_FUNC  dmm_UnlockSession (ViSession vi, ViBoolean *callerHasLock);

  /*- Basic Instrument Operation -------------------------------------*/

ViStatus _VI_FUNC  dmm_ConfigureMeasurement (ViSession vi, ViInt32 function,
                                             ViReal64 range, ViReal64 resolution);
ViStatus _VI_FUNC  dmm_ConfigureTrigger    (ViSession vi, ViInt32 triggerSource,
                                            ViReal64 triggerDelay);
ViStatus _VI_FUNC  dmm_Read  (ViSession vi, ViInt32 maxTime,
                              ViReal64 *reading);
ViStatus _VI_FUNC  dmm_Fetch (ViSession vi, ViInt32 maxTime, ViReal64 *reading);
ViStatus _VI_FUNC  dmm_Abort (ViSession vi);
ViStatus _VI_FUNC  dmm_Initiate (ViSession vi);
ViStatus _VI_FUNC  dmm_IsOverRange (ViSession vi, ViReal64 measurementValue,
                                    ViBoolean *isOverRange);

  /*- Error Functions ------------------------------------------------*/

ViStatus _VI_FUNC  dmm_error_query (ViSession vi, ViInt32 *errorCode,
                                    ViChar errorMessage[]);
```

```
ViStatus _VI_FUNC  dmm_GetErrorInfo (ViSession vi, ViStatus *primaryError,
                        ViStatus *secondaryError, ViChar errorElaboration[256]);
ViStatus _VI_FUNC  dmm_ClearErrorInfo (ViSession vi);
ViStatus _VI_FUNC  dmm_error_message (ViSession vi, ViStatus errorCode,
                                        ViChar errorMessage[256]);

  /*- Utility Functions --------------------------------------------------*/

ViStatus _VI_FUNC  dmm_reset (ViSession vi);
ViStatus _VI_FUNC  dmm_self_test (ViSession vi, ViInt16 *selfTestResult,
                                                    ViChar selfTestMessage[]);
ViStatus _VI_FUNC  dmm_revision_query (ViSession vi,
                    ViChar instrumentDriverRevision[], ViChar firmwareRevision[]);
ViStatus _VI_FUNC  dmm_WriteInstrData (ViSession vi, ViConstString writeBuffer);
ViStatus _VI_FUNC  dmm_ReadInstrData  (ViSession vi, ViInt32 numBytes,
                                    ViChar rdBuf[], ViInt32 *bytesRead);
```

Acestea sunt funcţiile necesare pentru a folosi capabilităţile de bază ale DMM-ului. Acest driver construit cu un utilitar are implementată toată structura IVI. Dar comenzile propriu-zise sunt implementate de proiectant. Astfel ca exemplu pentru funcţia **dmm_IviInit** comanda în limbaj SCPI este executată în subrutina:

```
static ViStatus dmm_DefaultInstrSetup (ViSession vi) {
  ViStatus    error = VI_SUCCESS;

  /* Invalidate all attributes */
  checkErr( Ivi_InvalidateAllAttributes (vi));

  if (!Ivi_Simulating(vi)){
     ViSession   io = Ivi_IOSession(vi); /* call only when locked */
     checkErr( Ivi_SetNeedToCheckStatus (vi, VI_TRUE));
     viCheckErr( viPrintf (io, "*CLS;*ESE 1;*SRE 32"));
     viCheckErr( viPrintf (io, ":HEADERS OFF"));
  }
Error:
  return error;
}
```

Se transmite sirul de comenzi SCPI pentru DMM

Un element important la aceste funcţii este prefixul. In cazul de mai sus am folosit prefixul „dmm" dar mai potrivit ar fi fost „hp34401" pentru că acesta este instrumentul folosit în realitate. După cum se remarcă indiferent de instrumentul din clasa dmm folosit doar prefixul diferă funcţiile fiind aceleaşi. Acest lucru permite invocarea funcţiilor respective din aplicaţia de nivel superior doar prin cunoaşterea prefixului. Acesta este de fapt mecanismul interschimbabilităţii instrumentelor.

Reluînd structura unui echipament de testare automată până la nivelul aplicaţiei de nivel superior se defineşte o arhitectură care asigură interschimbabilitatea şi posibilitatea de

extindere a sistemului:

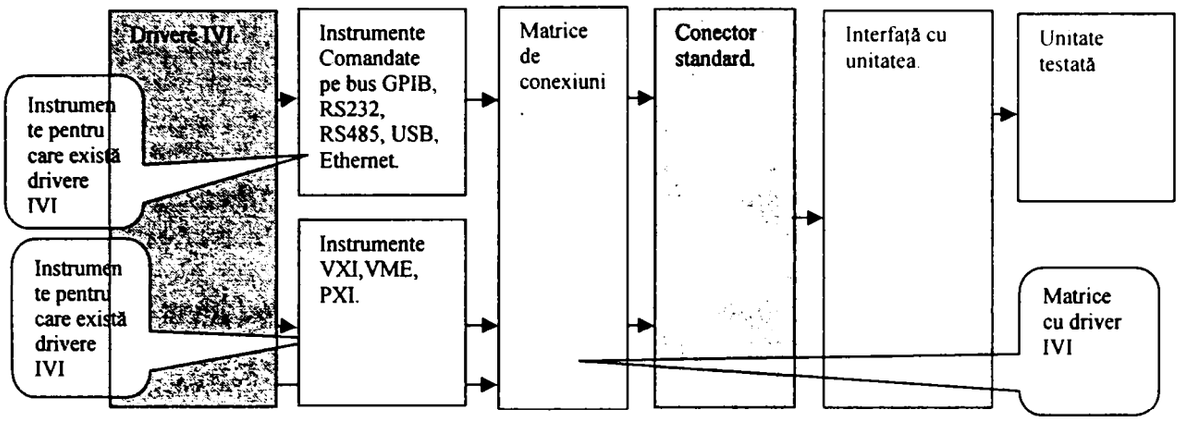


Fig. 5.15 Diagrama unui testor până la nivelul driverelor cu elemente standardizate

După cum se remarcă singurele elemente care nu sunt standardizate sunt interfaţa şi unitatea testată. Teoretic până la acest nivel transferul perechii interfaţă-unitate la un alt testor care respectă structura de mai sus nu afectează rezultatul testului.

5.5 Mediul de testare[157][156][154]

Se analizează în continuare mediul de testare din punctul de vedere al legăturii dintre programul de testare şi drivere, al modelului de proces pentru execuţia instrucţiilor şi secvenţelor şi al funcţionării test executivelor. Analiza se axează în special pe programe sursă ATLAS legate de instrumente care au un driver IVI.

Mediul de testare este cel care implementează modul de execuţie a instrucţiilor. Dacă privim diagramele 5.6-5.9 remarcăm multitudinea de operaţii care stau în spatele unor instrucţii ATLAS. Aceste diagrame subliniază faptul că ATLAS este un limbaj de nivel înalt. În general toate mediile de testare translatează programele ATLAS într-un program bazat pe un limbaj intermediar de nivel scăzut cu instrucţii simple. Aplicaţia care are ca entitate de intrare programul de testare şi care execută acest program se numeşte test executiv. Acest element din lanţul testării automate nu a fost standardizat deşi există standarde şi reglementări referitoare la procesul de testare automată. Fiecare companie care produce aplicaţii software doreşte să-şi protejeze propriile produse şi „know-how"-ul pe care îl posedă.

Un mediu deosebit de complex este mediul „TestStand" produs de compania National Instrumens. *Autorul a participat la proiectarea unui prototip de mediu de testare pentru avionică bazat pe drivere IVI şi modele de proces dezvoltat pe structură TestStand.*. Autorul prezintă în continuare un exemplu de implementare a unei instrucţii ATLAS bazat pe subinstrucţii elementare (pas)(numite conform variantei în limba engleză „step") cu referinţe la mediul „TestStand" dar cu pretenţia de generalitate a structurii .

Mediul TestStand a introdus câteva noţiuni interesante pentru managementul unui program de testare. Astfel a apărut noţiune de model de proces. Modelul de proces poate exista la nivel de program de testare, de bloc de testare, de secvenţă şi chiar la nivel de instrucţie sau pas. Modelul de proces specifică o succesiune de operaţii care se fac înainte şi după execuţia propriu-zisă a blocului, secvenţei, instrucţiei sau pasului. Modelul de proces poate fi editat şi modificat de către utilizator. Modelul de proces este o pseudo secvenţă care ordonează acţiunile test executivului la diverse nivele. În această secvenţă sunt chemate toate rutinele care trebuie executate înainte şi după bloc, secvenţă, instrucţie. Nu vom analiza implementarea din TestStand, dar menţionăm posibilitatea implementării fluxului programului de testare la acest nivel. Ceea ce ne interesează în legătura dintre ATLAS şi limbajul de nivel intermediar, este modul de implementare a instrucţiilor ATLAS. Analizăm o instrucţie simplă cum ar fi instrucţia SETUP. In ATLAS un exemplu de instrucţie este următorul:

```
100000 SETUP,  DC SIGNAL USING 'DC-VOLTMETER', VOLTAGE RANGE 0V TO 30V $
```

Evident instrucţia de mai sus va trebui translatată într-un limbaj mai precis. Astfel de aici aflăm ca se acţionează asupra unui DMM şi că este o măsurătoare în curent continuu iar domeniul este de la o la 30 V, deci se măsoară o tensiune. Să considerăm instrucţia de mai sus la nivel IVI şi sa o convertim în atribute şi funcţii din clasa DMM intr-o ordine conformă cu diagrama 5.6. Acest lucru semnifică iniţializarea instrumentului şi setarea acestuia:

```
1. IviDmm_Initiate (DCVolt);
2. IviDmm_ConfigureMeasurement
    (DCVolt, IVIDMM_VAL_DC_VOLTS, IVIDMM_VAL_AUTO_RANGE_ON, 1);
```

Aceste două funcţii vor fi executate la instrucţia SETUP. Dacă SETUP va fi definit ca un pas atunci cele două funcţii vor fi un modul extern chemat de pasul SETUP. Pasul va mai avea o serie de proprietăţi asociate:

- existenţa unei precondiţii care va fi testată de test executiv;
- existenţa unei postcondiţii care va fi testată de test executiv;
- existenţa modulului simulare care forţează execuţia cu succes a pasului;
- forţarea în mod eroare a execuţiei pasului;
- evitarea execuţiei pasului;

În concluzie pasul ATLAS va arăta astfel:

```
SETUP ( DCVolt, DC_VOLTS, AUTO, mV) $
```

Luăm în continuare spre analiză o instrucţie cu acţiune multiplă:

```
100000 VERIFY, (VOLTAGE), DC SIGNAL USING 'DC-VOLTMETER',
     UL 28.5 V  LL 27.5 V,
     VOLTAGE RANGE 0V TO 30V,
     CNX  HI  J1-16
          LO  EARTH  $
```

Conform standardului ARINC 626, instrucţia VERIFY se descompune în următoarele verbe simple:

SETUP
CONNECT
ARM
ENABLE, EVENT
FETCH
COMPARE
DISABLE, EVENT
DISCONNECT
RESET

Dacă am definit verbele simple ATLAS ca fiind paşi, atunci verbele multiple cum este şi VERIFY vor fi secvenţe. Dintre acţiunile simple de mai sus o problemă aparte o reprezintă CONNECT şi DISCONNECT. Aceste verbe operează asupra matricii de conexiuni. Informaţia din instrucţia ATLAS se referă doar la portul de destinaţie la care se conectează instrumentul considerându-se implicit cunoscut portul instrumentului. De fapt acest lucru semnifică faptul că procesul de alocare a fost realizat. Procesul de alocare asociază instrumentelor virtuale, instrumente reale cu porturile de conectare la matrice şi defineşte lanţurile de conexiuni necesare până la portul de destinaţie. Date fiind cele două acţiuni există o alocare a instrumentelor staţiei pentru resursele virtuale şi o alocare a căilor de conexiuni. **Alocarea** căilor de conexiuni se poate face **static** sau **dinamic**. **Alocarea statică** (se face în faza de compilare a programului de testare) implică un program de testare mai lung şi mai greoi fără posibilitatea utilizării conexiunilor variabile. Driver-ul pentru controlul căilor de conexiuni este mai complex în cazul alocării dinamice când o imagine oglindă a tuturor conexiunilor trebuie memorată. Utilizarea unui driver IVI conform clasei de conexiuni nu asigură toate aceste optimizări ale căilor precum şi memorarea stărilor matricei, fapt care implică utilizarea unui modul intermediar. Acest modul va fi însă specific matricei utilizate. Configurarea modulului va fi însă relativ simplă deoarece conectorul este standardizat şi practic impune numărul şi tipurile de conexiuni avute la dispoziţie. In cazul mediului TestStand pentru exemplul de mai sus, prima alocare a instrumentelor se va face manual, iar driverul de conexiuni va fi o problemă dacă alocarea se face dinamic deoarece acesta va trebui să fie permanent activ (un motor sau un serviciu la care test executivul să se conecteze în mod dinamic (ex. prin controale activeX)). Pornind de la aceste concluzii autorul *a proiectat o arhitectură a mediului software* care aduce independenţa programului de testare faţă de hardware. Această arhitectură încearcă să definească componentele unui astfel de sistem şi să stabilească care dintre acestea se impune să fie standardizate. O astfel de arhitectură operează cu toate componentele pe care le-am descris în acest capitol :

- program de testare scris in ATLAS;
- clase de instrumente;
- test executiv;
- alocator de resurse;
- limbaj intermediar;
- alocator dinamic de conexiuni;
- interfaţa grafică cu operatorul;
- gestionarea documentaţiei;

- configurarea staţiei;
- modele de proces;

Schema bloc a unui astfel de sistem se prezintă în figura 5.16

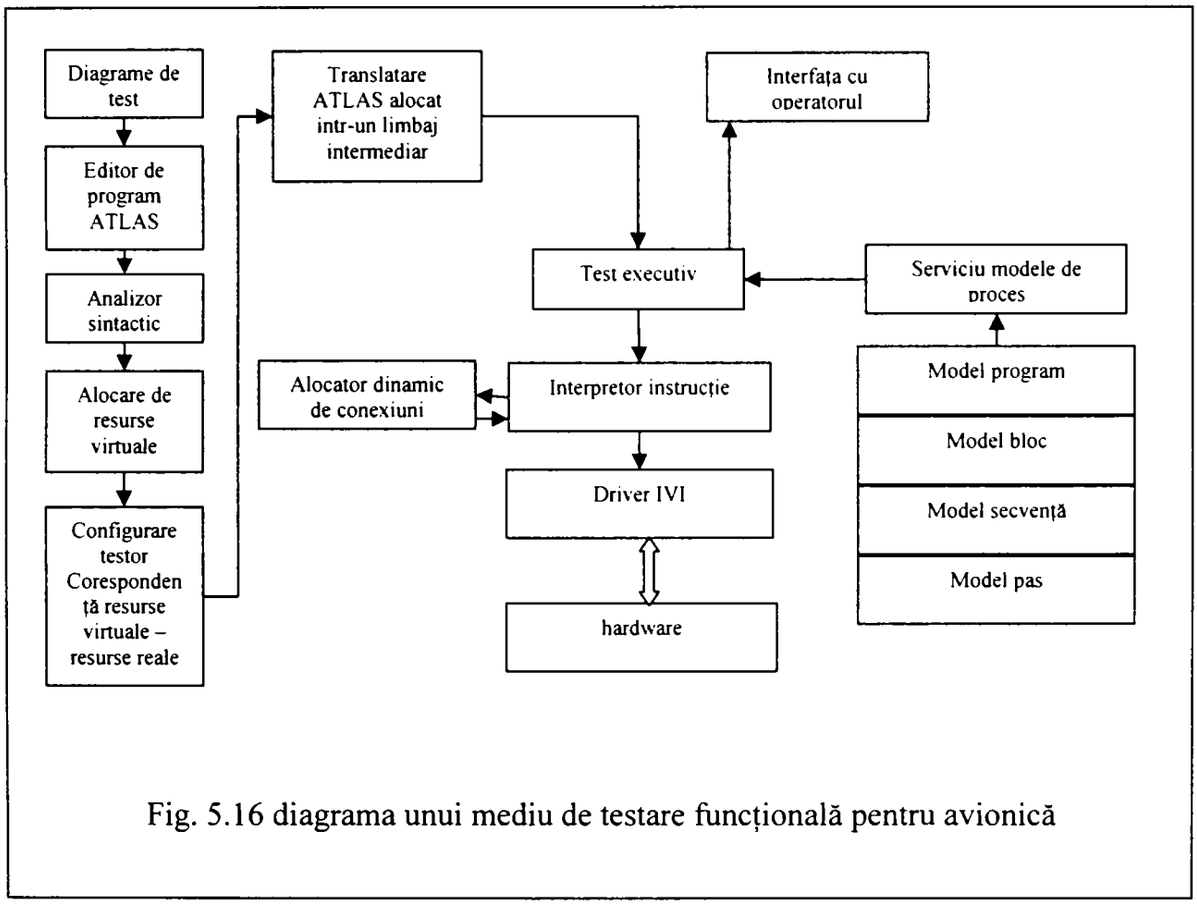


Fig. 5.16 diagrama unui mediu de testare funcţională pentru avionică

Dacă analizăm figura 5.16 remarcăm câteva procese care nu sunt rezolvate complet în acest moment. Informaţia pentru test executiv trebuie grupată şi ordonată pentru ca acesta să execute programul de testare. Astfel semnalelor virtuale definite de ATLAS, este necesar să li se asocieze instrumente IVI şi acestora instrumente reale. Blocurile şi secvenţele vor fi identificate şi marcate astfel încât test executivul să permită operatorului să iniţieze opţiunile care definesc un test funcţional pentru avionică. Limbajul intermediar evidenţiază legătura cu instrumentele dar permite urmărirea programului ATLAS. **Alocatorul dinamic** va funcţiona ca un serviciu care va pune tot timpul la dispoziţie starea căilor de conexiune ocupate şi libere. Serviciul care pune la dispoziţie modelele pentru bloc, secvenţă şi pas permite de fapt executarea unor secvenţe standard înainte şi după elementul de program sau returnarea unor fanioane referitoare la statusul elementului de program. Din această listă de

acţiuni se observă că portabilitatea este asigurată la nivelul unui grup de fişiere care furnizează următoarele informaţii:

- programul ATLAS verificat sintactic conform subsetului de instrucţii folosit;
- lista de instrumente IVI conţinută de testor (configuraţia staţiei);
- lista de alocare a instrumentelor virtuale la instrumente reale;
- lista alocării porturilor de instrument la conectorul standard. Aceasta nu este obligatorie, dar dacă sunt conexiuni opţionale atunci acestea trebuie specificate;
- dacă alocatorul dinamic suportă diverse configraţii pentru matricea de conexiuni, această configurare face parte din setul de documente.
- fişierele cu modelele de proces;
- fisierele cu formatul de raport;
- lista de conexiuni între conectorul standard al staţiei şi pinii unităţii

Din punct de vedere hardware portabilitatea este asigurată de interfaţa de testare şi de conectorul testorului care este standard.

Limbajul intermediar[204][205] este foarte important mai ales în faza de punere în funcţiune a testului. Autorul a dezvoltat în decursul timpului o variantă de limbaj bazată pe verbele ATLAS , cu atribute parţial inspirate din definiţia claselor de instrumente. Astfe o instrucţie ILS (Intermediary Language Set) *are în varianta autorului următoarea structură*:

Index, verb (clasa_instrument (prefix_instrument_alocat-> (atribut1=val1, atribut2= val2, ..., atributn=valn), val->(nume1 ,...) tol-> (atribut1=val1, atribut2=val2...), cnx-> (atribut1=val1, atribut2=val2,...));

Să considerăm exemplul ATLAS:

```
100000 VERIFY, (VOLTAGE), DC SIGNAL USING 'DC-VOLTMETER'.
    UL 28.5 V  LL 27.5 V,
    VOLTAGE RANGE 0V TO 30V,
    CNX  HI  J1-16
         LO  EARTH  $
```

Care devine:

100000, verify (dmm (dc_voltmeter->(FUNCTION = DC_VOLT, RANGE = 30V), val->(voltage), tol-> (UL = 28.5, LL = 27.5), cnx-> (HI = J1-16, LO = EARTH));

prin similitudine cu structura ATLAS, verify devine:

100000, setup // care se transformă spre driver în:

Funcţii care sunt apelate din driverul IVI pentru execuţia instrucţiei „setup"

```
IviDmm_Initiate (DCVolt);
IviDmm_ConfigureMeasurement (DCVolt, IVIDMM_VAL_DC_VOLTS, 30);
```

connect // care apelează alocatorul dinamic şi găseşte căile HI(dmm) – (J1-16)
// în echivalent la conectorul standard (listă de echivalenţe declarată)
arm // iniţializează instrumentul
enable, event // validează declanşarea
fetch // citeşte valoarea
compare // compară valoarea cu limitele
disable, event // invalidează declanşarea
disconect // deconectează instrumentul şi alocatorul dinamic îşi modifică
// informaţia memorată
reset // resetează instrumentul şi îl aduce în starea de aşteptare

Fiecare verb simplu are implementată o structură similară cu „setup" la nivel de legătură cu driver-ul IVI.

Autorul a realizat o aplicaţie transformator din ATLAS subsetul ARINC 626 în limbaj intermediar. Structura de directoare a aplicaţiei este următorul:

A2T
- ATLAS // conţine fisierul sursă ATLAS
- BIN // conţine executabilele
- ConfigTemplate // conţine fişiere de configurare în special pentru subset
- Projects // conţine proiectele nou create (ex. OEU_N)
  - OEU_N // director proiect nou creat
    - Cfg // director configurare
    - Out // director fişiere intermediare şi fişier rezultat
    - Temp // director fişiere temporare

Fişierele de configurare descriu practic limbajul ATLAS, limbajul intermediar şi legăturile dintre elementele celor două limbaje . Inerfaţa aplicaţei se prezintă în fig. 5.17 :

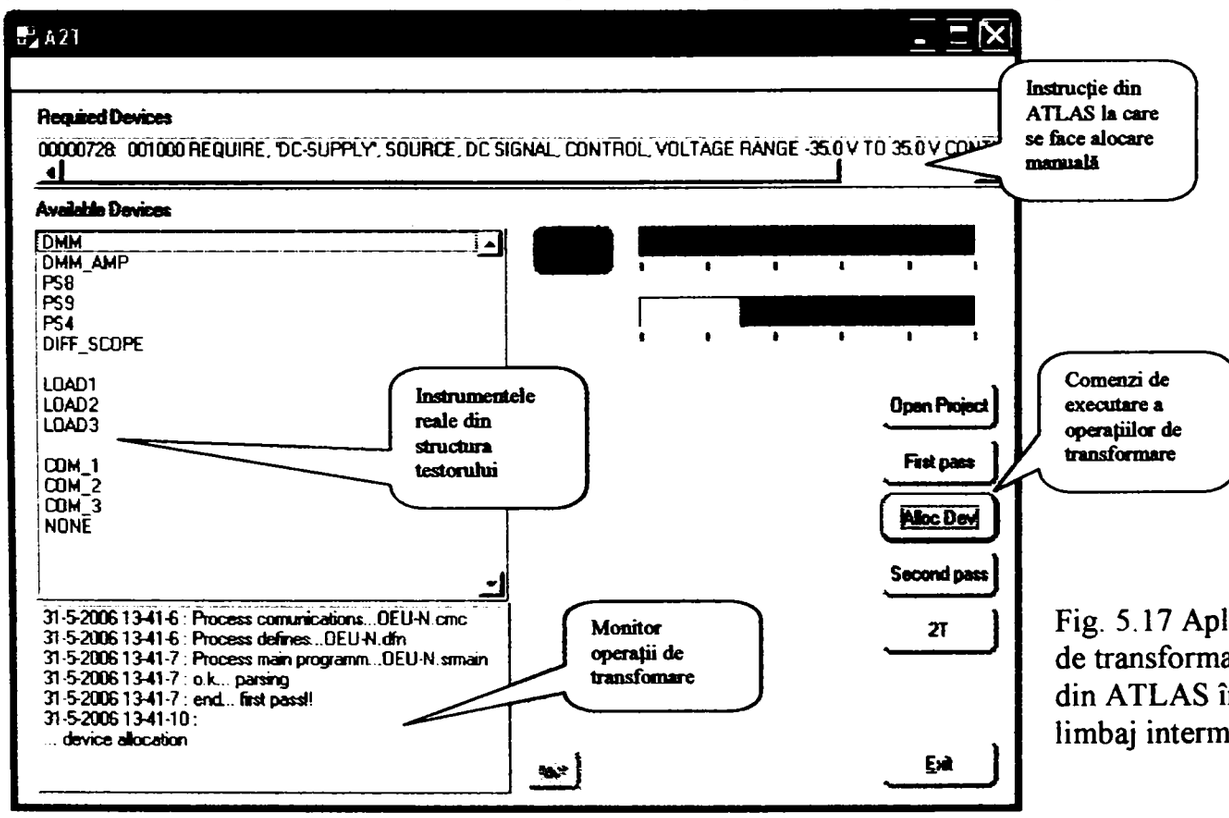


Fig. 5.17 Aplicaţie de transformare din ATLAS în limbaj intermediar

Fişierele de configurare, fişierul de intrare ATLAS şi fişierul rezultat după transformare sunt prezentate în Anexa 4.

Prezentăm în figura 5.18 interfaţa grafică a unei aplicaţii (test executiv) *la al cărei proiect autorul a participat:*

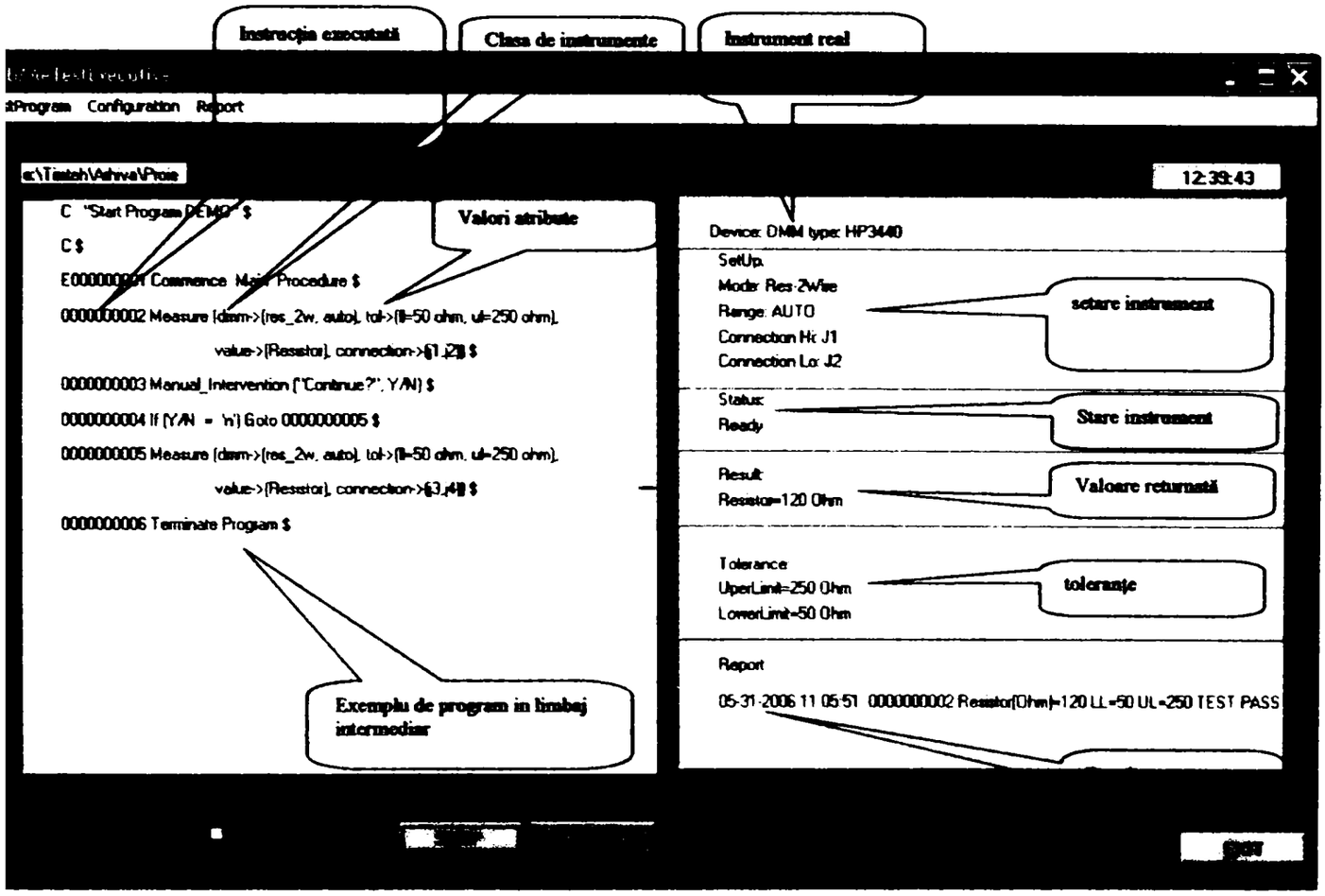


Fig. 5.18 Test executiv cu programul sursă în limbaj intermediar

In secţiunea de status se vizualizează toate verbele intermediare prin care trece instrumentul. Alocările de instrumente pentru acest prgram se fac manual din configurare şi se memorează într-un fişier staţie.ini.

Prin aceste aplicaţii autorul a verificat arhitectura din fig. 5.16 pentru un mediu de dezvoltare pentru programe de testare funcţională. În figura 3.13 din capitolul 3 este prezentată interfaţa grafică a unui driver IVI pentru DMM. *Toate aceste aplicaţii verifică structura de mediu de testare prezentată de autor ca fiind portabilă de la o generaţie la alta de echipamente.*

5.6 Translatoare de programe de testare.[108][199]

În prezent există foarte multe programe de testare pentru unităţi mai vechi, dar care sunt încă în serviciu. Modernizarea echipamentelor de testare impune şi transformarea acestor programe în programe compatibile cu noile echipamente. În cele mai multe situaţii se reproiectează şi interfeţele de testare. Validarea acestor programe este un proces lung şi anevoios. Astfel fiecare test trebuie verificat şi validat. Din acest motiv realizarea unor translatoare de programe este foarte utilă şi din punct de vedere al dezvoltării noului program şi din punct de vedere al validării sale. Translaţia se face în două planuri: hard şi soft. Translaţia hard presupune identificarea diagramelor de test, a resurselor din aceste diagrame şi proiectarea unor noi diagrame de testare cu echipamentul nou care să se suprapună funcţional peste diagrama veche. După acest moment se crează liste de echivalenţă între pinii conectorului testorului vechi şi pinii conectorului testorului nou. După crearea unor echivalenţe între circuitele funcţionale ale celor două testoare se trece la echivalarea rutinelor sau secvenţelor de măsură . Se identifică subrutinele de măsură din programul mai vechi şi se crează echivalentul lor în forma mai nouă. La terminarea acestor operaţiuni se valideaza noile runtine de măsură. Cu toate aceste informaţii se poate face translaţia programului vechi în programul nou. *La realizarea unui astfel de translator a participat autorul în cadrul unui proiect de portare a testelor pentru unităţi ale avionului CONCORDE.* Tema de proiectare a fost translaţia programelor vechi de testare care rulau pe un testor construit de Bae (British Aerospace) cu mulţi ani în urmă pe un testor modern (Smart Cats al companiei RADA Electronic Industries). Translaţia s-a făcut conform ordinogramei din figura:

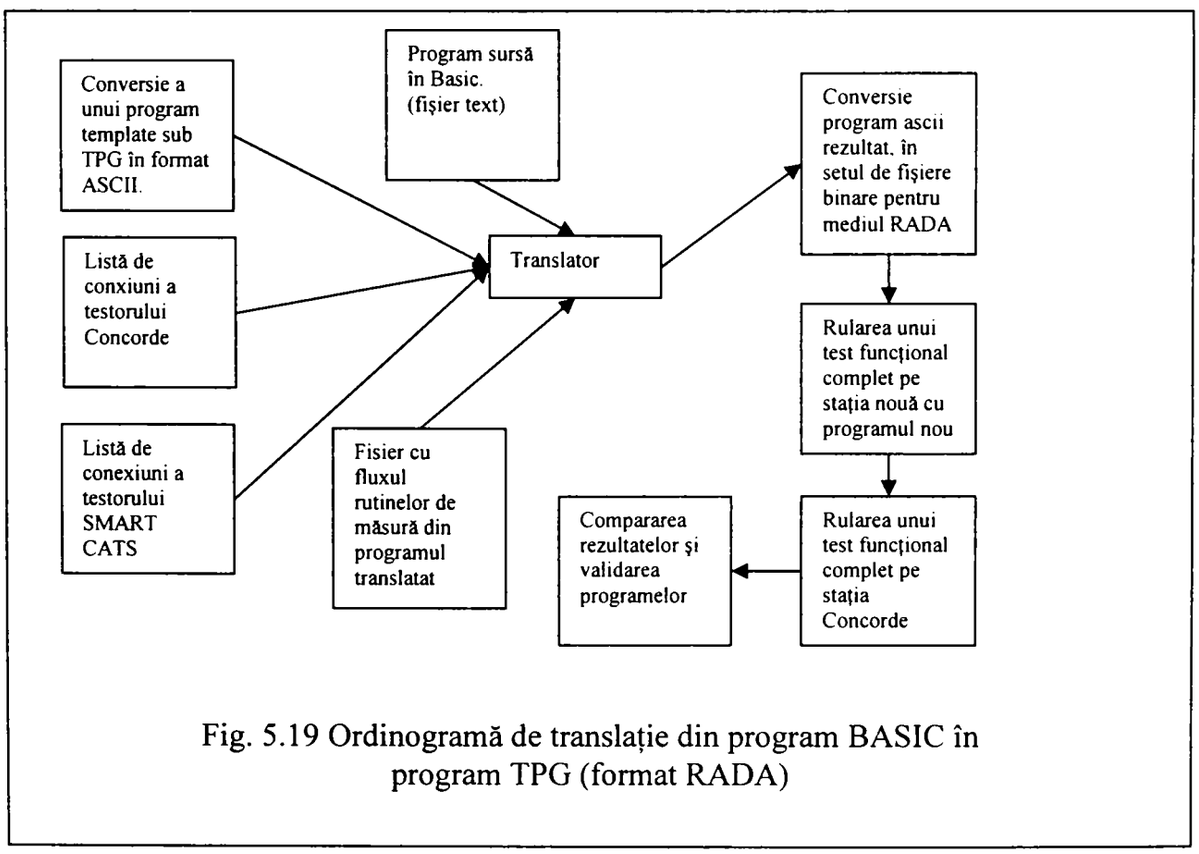


Fig. 5.19 Ordinogramă de translaţie din program BASIC în program TPG (format RADA)

Se prezintă în continuare în figura 5.20 un fragment din programul supus translaţiei şi un fragment din programul rezultat după translaţie:

```
1 COM C[2],D[2],F[1],X[4].P[1],R[1],S[15].T[3],Z[1],V[20],A[1
2 IF T[1]=3901 THEN 4
3 CALL (50,1,326,1)
4 LET T[1]=T[1]+1
6 REM
7  REM * FILE 3901   OUTER ELEVON NEUTRALIZATI
COMPUTER   *
8  REM * BAC/COEN/73901/AB  AMPLIFIER DEMODULAT
14-11-86 *
9 REM
10 GOTO A[1]
34 PRINT
35 PRINT
36 PRINT
37 PRINT
38 RETURN
47 LET X[1]=X
48 LET X[2]=H
49 LET X[3]=L
50 LET X[4]=W
51 LET A[1]=47
52 LET T[1]=21
53 CALL (50,2,T[1],1)
54 RETURN
CALL (32,1,326)
1099  CALL (32,1,325)
```

```
   3903_0001 COM C[2],D[2],F[1],X[4].P[1],R[1],S[15],T[3],Z[1],V[20],A[1]
   3903_0002 CMPJMP  ,EQ     ,T[1]    ,S3903_BAS ,L3903_0004
   3903_0003 JMP     ,UNCOND  ,SEQUENCE ,S0326_BAS
         REM3
L        LABEL    ,L3903_0004
   3903_0004 ADD     ,T[1]    ,1       ,RES_1
         LOADF    ,T[1]    ,RES_1
         REM1
   3903_0006 REM     ,*********************************************************
   3903_0007 REM     ,* FILE 3903   OUTER ELEVON NEUTRALIZATION COMPUTER
   3903_0008 REM     ,* BAC/COEN/73903/AC  NEUT.ZN DECISION LOGIC  17-12-86 *
   3903_0009 REM     ,*********************************************************
         REM2
   3903_0010 JMP     ,UNCOND  ,LINE    ,##########
         REM3
L        LABEL    ,L3903_0034
         REM1
   3903_0034 PRINT   ,
   3903_0035 PRINT   ,
         REM2
         DISP_LINE ,1      ,-      ,-      ,UUT.HDR
         DISP_LINE ,2      ,-      ,-      ,UUT.HDR
         REM3
L        LABEL    ,L3903_0036
         REM1
   3903_0036 PRINT   ,
   3903_0037 PRINT   ,
         REM2
         DISP_LINE ,3      ,-      ,-      ,UUT.HDR
         DISP_LINE ,4      ,-      ,-      ,UUT.HDR
   3903_0038 RETURN  ,2
         REM3
L        LABEL    ,L3903_0047
   3903_0047 LOADF   ,X[1]    ,X
   3903_0048 LOADF   ,X[2]    ,H
   3903_0049 LOADF   ,X[3]    ,L
   3903_0050 LOADF   ,X[4]    ,W
   3903_0051 LOADF   ,A[1]    ,47
   3903_0052 LOADF   ,T[1]    ,S0021_BAS
   3903_0053 CALL    ,UNCOND  ,T[1]    ,FIRST-LINE,NOT-PRINT
   3903_0054 RETURN  ,2
```

Programul sursă BASIC

Indicatorul rutinei BASIC

Linia BASIC

Fişierul rezultat TPG

Fig. 5.20 Program BASIC supus translaţiei comparativ cu program TPG rezultat (format RADA)

5.7 Concluzii

Acest capitol prezintă în secţiunea 5.1 o imagine de ansamblu asupra structurii mediilor de programare pentru testarea avionicii. Se prezintă entităţile componente şi importanţa fiecăruia în ansamblul sistemului de testare.

În secţiunea 5.2 se abordează problema limbajelor de programare din punctul de vedere al evoluţiei lor în timp precum şi prin prisma experienţei autorului. Se prezintă fracţiuni de program din programul de testare al unităţii Flight Management Computer , program proiectat, verificat şi validat , de către autor. O deosebită şi largă abordare se acordă limbajului ATLAS devenit limbaj de programare a testelor şi standard de testare. Se prezintă câteva secvenţe din programul pentru *unitatea ACP la elaborarea căruia autorul a participat*, *program ATLAS* care a fost *validat de producătorul avionului*. Câteva fragmente sunt prezentate şi în Anexa 2.

În secţiunea 5.3 se analizează modul de construcţie a driverelor de instrumente. Se *descrie funcţionarea practică a driverelor proiectate* în mediul PAWS, sub UNIXWARE ca sistem de operare, pentru *testarea unor unităţi din seria Boeing 777*.

În secţiunea 5.4 se prezintă structura driverelor IVI a căror dezvoltare a permis reconsiderarea problemei portabilităţii programelor de testare. Autorul analizează în detaliu structura unui astfel de driver cu *exemplificări din proiectul pentru DMM la care autorul a participat.*

În secţiunea 5.5 se dezvoltă o analiză a diverselor momente din procesul de testare luînd în considerare constrângerile şi interdependeţele care apar datorită numeroaselor componente ale mediului precum şi legăturii dintre software şi hardware. Mediul de testare care administrează sistemul de testare şi asigură mijloacele pentru dezvoltarea programelor este şi entitatea cea mai dificil de standardizat datorită complexităţii sale. Autorul urmăreşte legătura dintre programul ATLAS şi instrument pe baza unor exemple care permit *propunerea unei arhitecturi a sistemului de către autor. De asemenea autorul propune o structură de limbaj intermediar* care este *verificată* pas cu pas prin câteva *programe de transformare şi execuţie proiectate de autor.*

In secţiunea 5.6 se abordează o problemă importantă a momentului de tranziţie de la testoare mai vechi la noile arhitecturi. *Autorul a participat la translaţia şi validarea a peste 20 de programe de testare pentru avionul Concorde* la varianta pentru testorul SMART CATS. Translatorul la al cărui proiect autorul a participat, a permis translatarea semiautomată a programelor de testare. In secţiune se prezintă comparativ fişierul translatat de program şi fişierul rezultat pentru comparare precum şi o ordinogramă a momentelor unei astfel de translaţii.

O concluzie generală a acestui capitol este caracterul aplicat al prezentării subiectelor precum şi aplicarea unor idei noi de structurare a sistemului. Dintre acestea *se menţionează proiectul de limbaj intermediar* care permite o mai bună şi mai simplă legare de driver şi conservarea caracterului descriptiv al limbajului de nivel înalt ATLAS.

Cap. VI Concluzii

Concluzia sintetică a acestei teze este că setul de fisiere şi programe (TPS), driverele de tip IVI şi conectorul standard pentru interfaţa de testare sunt elementele care asigură portabilitatea programelor de testare în domeniul avionicii.

Dacă cele trei entităţi sunt obligatorii pentru portabilitate şi celelate module trebuie să îndeplinescă diverse criterii şi reguli. Autorul prezintă în cele patru capitole tehnologiile care iau parte la ansamblul unui sistem de testare automată pentru aavionică cu observaţiile şi experienţele personale obţinute de-a lungul unui şir de participări la proiecte din acest domeniu.

În capitolul I autorul prezintă domeniul tehnologic al echipamentelor de aviaţie cu obiectivele şi problemele acestei industrii. Astfel se remarcă diversitatea dispozitivelor cuprinse sub denumirea generică de avionică şi complexitatea sistemelor de testare. Se subliniază de asemenea importanţa testării funcţionale pentru acceptarea în serviciu a unei unităţi de aviaţie.

În capitolul II autorul prezintă domeniul testării funcţionale din perspectiva portabilităţii, subliniind elementele portabile necesare în tranziţia de la o platformă la alta. Autorul evidenţiază zonele de discontinuitate între descriere şi implementarea fizică cu limitările şi standardizările necesare, cu operaţiunile ajutătoare necesare şi cu descrierile suplimentare adăugate la TPS. *Capitolul II a fost introdus de către autor pentru a sublinia scopul acestei teze de a găsi printr-o analiză de detaliu a fiecărei componente din proces, a unei structuri şi ordini de execuţie a diverselor faze care să asigure portabilitatea testelor funcţionale în cadrul standardelor valide astăzi.*

În capitolul III autorul abordează echipamentele de testare automată din punct de vederea a structurii acestora, a instrumentelor şi dispozitivelor folosite. Se prezintă tipuri de testoare care au fost utilizate sau proiectate de autor precum şi tipuri de magistrale sau de instrumente utilizate. Se prezintă de asemenea noţiunea de clasă de instrument cu exemple aplicate trecându-se în revistă câteva instrumente cunoscute şi capabilităţile clasei din care acestea fac parte. Se prezintă aspecte specifice testării aparaturii de aviaţie cum ar fi conectorul standardizat (şi în general structura testorului) pentru testarea automată în aviaţie precum şi dispozitive de comunicaţie folosite în această industrie.

În capitolul IV se explică procesul de testare funcţională cu exemple de documentare a testelor precum şi diagrama unui test funcţional. Autorul exemplifică fiecare aspect al prezentării cu aplicaţii realizate. Tot în acest capitol se face o trecere în revistă a diverselor verificări utilizate la testarea avionicii. În final autorul prezintă sintetic tipurile de instrumente şi gradul acestora de standardizare.

În capitolul V se analizează problema structurii mediilor de testare pentru sisteme automate şi în special pentru avionică precum şi limbajele folosite, driverele şi alte componente software. În acest capitol se subliniază importanţa standardului ARINC 626 (IEEE C/ATLAS) care defineşte un limbaj şi un proces de testare. Autorul prezintă lanţul de procesări care au loc cu exemplificări sau aplicaţii proprii şi soluţii în plan teoretic. Dezvoltarea driverelor IVI este prezentată în detaliu cu exemple practice. Se subliniază în mod deosebit necesitatea unui limbaj intermediar şi se prezintă o variantă proprie de limbaj, de translator de la ATLAS la varianta intermediară şi o aplicaţie de tip testexecutiv pentru acesta. Translatoarele sunt o aplicţie utilă în perioada de tranziţie de la

vechile programe de testare la noua generaţie pe care autorul o prezintă din punctul de vedere al experienţei proprii.

Cele cinci capitole prezintă în detaliu şi în ordine firească tehnologiile care participă la acest complex proces de testare precum şi orientările acestei industrii. Dintr-o astfel de analiză se extrag concluzii şi pentru industria producăoare de avionică care poate proiecta orientat spre testare luînd în considerare tehnologiile, standardele, protocoalele utilizate de echipamentele de testare.

În diagrama 2.1 autorul înlănţuie sintetic toate componentele sistemului pornind de la proiectarea şi verificarea TPS-ului, cu execuţia şi alocarea resurselor de catre ATE, cu comanda instrumentelor prin driverele IVI, cu legarea instrumentelor şi matricii de conexiuni de interfaţa de testare legată la rândul ei de unitatea testată.

In concluzie autorul a prezentat în această teză o imagine de ansamblu a tehnologiei de testare automată a aparatelor de aviaţie cu detalii şi exemple pentru elementele decisive pentru portabilitatea programelor precum şi experienţe şi proiecte proprii care au încercat să soluţioneze anumite verigi tehnice din sistem.

Cap. VII Contribuţii

Teza de doctorat „**Contribuţii la configurarea unor structuri de testare automată cu aplicaţii în avionică**" sintetizează şi reflectă activitatea autorului în perioada perfecţionării profesionale prin activitatea de doctorand dar şi preocupările profesionale cotidiene ale acestuia. Autorul a studiat numeroase documentaţii ale unor aparate folosite pe aeronave, standarde şi documente de lucru ale diverselor comitete şi consorţii profesionale care activează în domeniu, publicaţii în reviste şi jurnale ale asociaţiilor profesionale din domeniul instrumentaţiei şi măsurătorilor sau aeronauticii, cărţi de referinţă ale unor personalităţi din domeniu.

Activitatea autorului s-a axat pe încercarea de a proiecta o structură de sistem care să permită portabilitatea programelor de testare precum şi pe cercetarea şi verificarea prin soluţii proprii a unor module din arhitectura sistemului.

Se analizează în continuare pe capitole contribuţiile autorului la aprofundarea şi dezvoltarea unor module din complexul sistem al echipamentelor de testare automată pentru avionică.

1. Contribuţii la cap.I „Stadiul actual la testarea automată pentru avionică"

Acest capitol introductiv prezintă domeniul de aplicaţie al echipamentelor de testare automată şi structura hardware şi software a acestora. Pe baza studiului bibliografic şi a experienţei în testarea avionicii autorul a constat:

- Noţiunea de avionică conţine o mare diversitate de dispozitive electronice proiectate pentru o perioadă îndelungată de timp;
- Dispozitivele electronice care se utilizează pe o aeronavă pot fi clasificate ca provenind din perioada analogă, din epoca digitală sau din perioada dispozitivelor modular-digitale;
- Dispozitivele pot fi clasificate după funcţia pe care o îndeplinesc în sistemul aeronavei, dispozitivele de comunicaţie fiind cele mai numeroase;
- Echipamentele de testare pot fi specifice pentru un dispozitiv sau grup de dispozitive electronice sau pot fi echipamente de uz general care permit testarea unui mare număr de unităţi de avionică;
- Testarea funcţională este procedeul prin care se decide dacă o unitate de avionică poate fi folosită in sistemul aeronavei pentru serviciu sau nu;
- Programul de testare se proiectează pe baza datelor din documentele de testare ale producătorului prezentate sub formă de manual de întreţinere, document cu specificaţiile de testare (Test Requirement Document) sau ATLAS.
- Echipamentul de testare cuprinde computerul, instrumentele pentru măsură şi stimulare, matricea de conexiuni, conectorul testorului şi interfaţa de testare;
- Programele de testare pot fi scrise sub mai multe forme şi limbaje. Există programe în limbaje de nivel scăzut a căror portabilitate este complicată şi programe în limbaje de nivel înalt cum este ATLAS, limbaj cu o mare

independenţă faţă de mediul de dezvoltare şi drivere. Autorul exemplifică programarea în diverse limbaje;

- Există diverse sisteme care execută programele. Autorul aminteşte câteva tipuri de mediu de lucru pentru testarea avionicii. Exemplele cuprind mediul dezvoltat la BA (British Airways) pentru avionul Concorde, mediul folosit de RADA Electronic Ind., mediul folosit de TYX Corp., mediul TestStand de la national Instruments, mediul ATEasy de la Marvin Test Systems şi mediul dezvoltat de ARINC, inc. Autorul a dezvoltat proiecte pe toate aceste medii.
- Standardizarea în domeniul testării de avionică este deosebit de densă. Se amintesc câteva standarde ARINC (compania liniilor aeriene care administrează standardizarea în domeniu) sau IEEE care reglementează procesul de testare. Autorul a participat la reviziile interne pentru ARINC 625, standard care defineşte procesul de proiectare a testelor din punct de vedere al calităţii.
- Introducerea driverelor tip IVI a făcut posibilă portarea programelor de testare. Autorul trece în revistă instrumentele pentru care au fost definite capabilităţi de clasă de către consorţiul IVI.

2. Contribuţii la cap. II „Testarea funcţională pentru avionică. Portabilitatea programelor de testare. "

Capitolul defineşte conceptul de testare funcţională şi domeniile concurente semnificative participante la acest proces. Autorul prezintă procesul de testare funcţională în domeniul avionicii din perspectiva portabilităţii testelor cu următoarele constatări şi remarci personale:

- Domeniul testării funcţionale în avionică poate fi împărţit în şase sectoare de interes care cuprind atât zona de software cât şi zone legate de arhitectura hardware a sistemului.
- Descrierea testelor funcţionale a suferit un proces continuu de standardizare.
- Descrierea testelor a evoluat formal până la limita de tranziţie dintr-un document descriptiv într-un program de testare.
- Din mulţimea aplicaţiilor care participă la execuţia unui program de testare, cele care fac legătura dintre programul de testare şi instrumente condiţionează portabilitatea.
- Conceptul de instrumentaţie virtuală interschimbabilă a accelerat drumul către portabilitate.
- Interfaţa de testare nu trebuie să afecteze portabilitatea.
- Alocarea dinamică a căilor de conexiune este un proces specific fiecărui testor, dar un descriptor al matricii de conexiuni face parte din setul portabil de documente.
- Necesitatea alocării resurselor virtuale din descriere la resursele virtuale ale testorului şi necesitatea tranziţiei de la programul descriptiv de nivel înalt la un program intermediar în faza de execuţie.

- Prezentarea drumului parcurs de la descrierea testelor până la execuţia acestora cu tranziţia de la un sector la altul, de la software la hardware, cu sublinierea elementelor obligatorii pentru portabilitate.

3. Contribuţii la cap. III „Echipamente de testare automată pentru avionică"

Capitolul III prezintă arhitecturi de staţii de testare, instrumentele uzuale folosite pentru testare în avionică şi scheme de utilizare a acestora. Din experienţa utilizării a numeroase staţii de testare precum şi a proiectării de testoare, din aprofundarea standardului ARINC 608 (IEEE P1505) autorul a constat sau a contribuit cu următoarele observaţii, experienţe proprii şi cercetări la definirea unei arhitecturi de staţie de testare:

- Structura sistemelor de testare este supusă unui proces de standardizare intens la diferite nivele cu scopul portabilităţii programelor de testare şi pentru crearea unor structuri deschise care permit extinderea sistemului cu conservarea programelor dezvoltate anterior;
- Autorul prezintă staţia SMART CATS şi o schemă bloc de testor pentru avionică folosită pentru testarea dispozitivelor de tip modular utilizate pe noile aeronave din clasa B777.
- Schema bloc a testorului subliniază categoriile de echipament care alcătuiesc un astfel de sistem.
- Autorul prezintă o altă variantă de testor realizată de autor pe structura unui PC industrial dar care conservă structura unui testor de uz general. Testorul a fost proiectat pentru testarea unui recorder video care echipează elicoptere (la a cărui proiectare autorul a participat de asemenea), dar după cum se observă din schemă a fost echipat cu module de comunicaţie, cu matrice de conexiuni astfel încât el poate fi utilizat şi pentru testarea altor unităţi.
- Magistrala de comunicaţie pentru instrumentaţie definită de standardul IEEE 488 în conexiune cu standardul VXI, este folosită pe scară largă în construcţia testoarelor. Cele două standarde au permis dezvoltarea unui limbaj pentru comanda instrumentelor, SCPI, care este folosit fie pentru instrumente de sine stătătoare fie pentru instrumente modulare comandate prin drivere instalate pe controlorul de sertar. Această arhitectură a permis dezvoltarea instrumentaţiei virtuale interschimbabile.
- Instrumentele cele mai utilizate sunt analizate din punct de vedere al schemei funcţionale şi al existenţei capabilităţilor clasei instrumentului respectiv definite conform specificaţiei IVI.
- Autorul a realizat un instrument generic cu o interfaţă grafică care permite accesul la capabilităţile clasei DMM. Instrumentul propriu-zis poate fi oricare instrument ale cărui drivere permit conectarea la un driver de nivel mai înalt care realizează interfaţa IVI.
- Se prezintă instrumentele şi din punct de vedere al modelului de funcţionare care trebuie corelat cu modelele implementate pentru verbe multiple la nivel de limbaj ATLAS.

- Analiza modului de raportare a stării pentru osciloscop permite observarea modului de conexiune între driverul IVI şi toate instrumentele care au implementat limbajul SCPI la nivel de control al stării instrumentului. Acest proces este legat la nivel mai înalt cu structura verbelor cu acţiune multiplă ATLAS;
- Aprofundarea standardului ARINC 608 care este decisiv pentru realizarea portabilităţii programelor de testare şi în a cărui structură se remarcă conectorul standard pentru avionică împreună cu matricea de conexiuni.
- Se prezintă o schemă de matrice de conexiuni utilizată de autor, în care se remarcă limitările tehnologice asupra modelului teoretic, care se vor reflecta în toate documentele şi programele referitoare la conexiuni.
- Autorul remarcă standardizarea elementelor de conexiune cu reprezentarea detaliată a căilor de conexiune care permit construirea unei matrici complexe. Se constată amânarea standardizării problemei alocării căilor de conexiune şi conservarea ei la nivel de mediu de testare ca opţiune specifică producătorului.
- Prezentarea în detaliu a sistemului de comunicaţie de pe o aeronavă modernă şi analizarea modulului de simulare şi testare pentru comunicaţia de tip ARINC 629.
- Realizarea unui driver pentru modulul de comunicaţie A629 de nivel înalt precum şi testarea unor unităţi cu acest tip de driver.
- Analiza comunicaţiei de tip A429 care este consacrată în avionică . Autorul a participat la realizarea unui driver pentru A429 pentru placa VXI Tektronix VX4469 dar şi pentru modulul produs de firma TASCO.
- Se aprofundează standardizarea IVI pentru sursele de alimentare şi se constată că este realizată doar pentru sursele DC.
- Modul de proiectare a interfeţei de testare permite interconectarea unităţii la testorul de uz general şi se subliniază necesitatea verificării interfeţei printră-un program de testare propriu. La realizarea unei structuri standard de program de testare a interfeţei precum şi la proiectarea unor circuite care au devenit permanente a participat şi autorul în baza experienţei acumulate.
- Autorul prezintă câteva scheme de conectare a instrumentelor uzuale la matricea de conexiuni, scheme de protecţie a surselor, modul de identificare a interfeţei de testare sau legarea unui osciloscop cu patru canale pentru o utilizare optimă a canalelor analoge din matricea de conexiuni;

4. Contribuţii la cap. IV „Structura programelor de testare funcţională. Exemple specifice pentru avionică"

Autorul a analizat la elaborarea numeroaselor programe de testare la care a participat, procesul de testare funcţională, structura programelor , tipologia măsurătorilor. Contribuţiile şi constatările referitoare la acest capitol sunt:

- Din experienţa proprie şi din studierea unor programe elaborate de companii sau producători, autorul aprofundează în limitele domeniului testare de avionică conceptul de testare funcţională.
- Proiectarea programelor de testare funcţională este un proces laborios pe care autorul îl prezintă în ordinea sa firească. Autorul prezintă o ordinogramă de testare funcţională verificată pe programele mai multor unităţi, dar cu aplicaţie pe unitatea Air Data Computer. Diagrama fixează paşii necesari în procesul de testare funcţională.
- Autorul detaliază prezentarea cu schema de testare a blocului 2 din testarea unităţii AirData Computer. In Anexa 1 se prezintă un pachet de documentaţie complet pentru blocul 2 care cuprinde documentul descriptiv (în format ATLAS, care aici este folosit ca limbaj de descriere), schemele aferente din interfaţa de testare, programul de testare şi rezultatele acestei testări într-un fişier de raport;
- Autorul analizează în acest capitol câteva scheme de conectare consacrate ale instrumentelor de măsură. Astfel se prezintă ordinea de efectuare a măsuratorilor (cu schemele aferente) cum ar fi: verificarea la „rece" a unităţii, conectarea DMM-ului, schema de principiu pentru conectarea canalelor de A429, schema de testare a comunicaţiei A629, testarea J-TAG, simulatoarele de LVDT şi Synchro/Resolver. Autorul prezintă o schemă bloc pentru testarea în domeniul RF adăugată ca extensie la testarea obişnuită.
- Se constată la avionica de tip modular utilizarea unor programe speciale de testare care se încarcă în unităţi în momentul testului. Autorul prezintă experienţa proprie în acest gen de testare care integrează aplicaţii specifice de încărcare cu un mediu de testare de uz general.
- Se trec în revistă tipurile de instrumente utilizate frecvent cu gradul de integrare pe care îl are fiecare într-un sistem care permite portabilitatea programelor de testare.
- Proiectarea programelor de testare pentru avionică este obiectul unor eforturi susţinute de standardizare printre care şi eleborarea standardului ARINC 625 care stabileşte criteriile de calitate pentru acest proces. Autorul a făcut observaţii asupra acestui standard în perioada de elaborare a acestuia bazate pe experienţa proprie.

5. Contribuţii la cap. V „Mediul de testare, limbaje de programare, drivere"

Acest capitol cuprinde elementele cele mai importante, decisive pentru realizarea unor structuri de testare portabile. Prin prisma numeroaselor programe de testare elaborate, a diverselor încercări de realizare de componente software care să permită atingerea acestui deziderat al portabilităţii, a bibliografiei şi cercetărilor din domeniu autorul a constatat elemente de interes sau a contribuit cu observaţii sau proiecte personale:

- Se prezintă o schemă bloc a entităţilor software care compun sistemul de testare funcţională şi activităţile asociate în ordinea cronologică a utilizării lor. Se subliniază componentele esenţiale ale sistemului.
- Se analizează comparativ limbajele de programare de nivel scăzut, cu o exemplificare a limbajului folosit de aplicaţia TPG a companiei RADA şi limbajul ATLAS.
- Se observă faptul că limbajul ATLAS a devenit dintr-un limbaj folosit pentru descrierea testelor un limbaj de programare şi un standard pentru testarea funcţională orientat spre semnale.
- Prezentarea limbajului se face cu exemple din activitatea de programare de teste a autorului pentru diverse unităţi de aviaţie.
- Se prezintă asociat cu exemplele diagramele de funcţionare a verbelor simple şi multiple cu declanşare sau fără declanşare.
- Elaborarea programelor ATLAS cuprinde două situaţii. Prima şi cea mai frecventă este aceea când proiectanţii de teste folosesc un Program ATLAS sursă furnizat de proiectantul unităţii care trebuie integrat cu sistemul de testare, alocate instrumentele şi căile de conectare şi revalidat programul. A doua situaţie mult mai dificilă este aceea a elaborării programului sursă ATLAS în baza unor alte documente ale producătorului cu integrarea acestuia cu sistemul şi validarea programului. Autorul prezintă în anexă fragmente dintr-un astfel de program la a cărui proiectare a participat.
- Se evidenţiază importanţa driverelor de instrumente pentru portabilitatea programelor de testare. Analiza cuprinde drivere legate de programe scrise în limbajul ATLAS pentru mai multe tipuri de programe de execuţie. Pentru Mediul Paws se prezintă fragmente din driverele de nive înalt la a căror elaborare autorul a participat.
- Driverele de tip IVI au reprezentat momentul de cotitură în evoluţia sistemelor de testare automată introducând conceptul de clase de instrumente şi de instrumente interschimbabile. Se evidenţiază structura unui driver pe exemplul driver-ului pentru DMM la a cărui proiectare autorul a participat.
- Se evidenţiază structura unui testor până la nivelul driverelor remarcâdu-se faptul că singurul element specific este perechea unitate de testare – interfaţă cu testorul.
- Execuţia instrucţiilor ATLAS este realizată prin intermediul programelor tip testexecutiv şi al driverelor IVI. Autorul exemplifică legătura dintre verbele simple şi multiple ATLAS cu driverele IVI. Se sublinează de asemenea importanţa modelelor de proces pentru execuţia programelor de testare funcţională.
- Din această analiză rezultă concluzia existenţei mai multor componente sftware precum şi necesitatea utilizării unui limbaj intermediar între ATLAS şi drivere. Se prezintă necesitatea existenţei unui alocator dinamic pentru conexiuni.
- Autorul propune formatul unui limbaj intermediar care conservă caracterul descriptiv al limbajului ATLAS dar include utilizarea

claselor de instrumente. Se exemplifică pentru DMM formatul şi modul de translaţie de la ATLAS la limbajul intermediar. Se prezintă o aplicaţie realizată de autor pentru transformare.

- Autorul a testat acest limbaj, utilizând un test executiv pe care l-a proiectat cu acest scop şi care reflectă în prezentarea grafică toate stările prin care trece sistemul la execuţia unei măsurători.
- Există utilitare importante pentru conservarea programelor de testare proiectate mai de mult dar încă utile pentru testarea avionicii aflate în serviciu. O astfel de categorie o reprezintă translatoarele de programe. Autorul prezintă o astfel de aplicaţie pentru programele de testare de la unităţi de pe avionul Concorde la a cărui proiectare a participat şi care a fost utilizat la mai mult de 20 de astfel de programe.

BIBLIOGRAFIE

[1] IEE TC8, SCCC20: "Atlas 2000 Architecture", 1997.
[2] Arinc Aeronautical Radio, Inc.: SMART Getting Started, May 1992.
[3] Arinc Aeronautical Radio, Inc.: Smart Configuration Model Language, May 1992.
[4] Arinc Aeronautical Radio, Inc.: Smart Key Word Language, May 1992.
[5] Arinc Aeronautical Radio, Inc.: Smart Test Unit Adapter Description Language, May 1992.
[6] Arinc Aeronautical Radio, Inc.: Smart Resource Description Keyword Language, May 1992.
[7] Arinc Aeronautical Radio,inc.: Atlas 626-3 specification.
[8] Smart Cats manul, Rada Electronic Industries Ltd.
[9] National Instruments: LabWindows/CVI user manual, 1998.
[10] National Instruments: LabWindows/CVI test executive toolkit reference manual, 1998.
[11] National Instruments: VXI/VME PC 600 series user manual. 1998
[12] IEEE Aerospace and Electronic systems - Integrating system engineering software tools. nov. 1998
[13] IEEE Aerospace and Electronic systems - Www leads an ATE revolution. june 1998.
[14] IEEE Aerospace and Electronic systems - Applying independent verification and validation to the ATE life cycle vol. 13 nr. 7.
[15] **Dan Simu**, An overview of automatic test equipment, third international conference on technical informatics CONTI'98 , Buletinul stiinţific al Universităţii „Politehnica" Timişoara, seria Automatică şi Calculatoare pag. 104-108, - 1998.
[16] Miron Abramovici, Melvin a. Breuer, Arthur d. Friedman, Digital systems testing and testable design - 1990.
[17] Arinc specification 429p3-18 mark 33 digital information transfer system (dits)
[18] Arinc report 608a, Design guidance for avionics test equipment – 1993.
[19] Arinc report 625-1, Quality management process for test procedure generation – 1999
[20] Arinc report 626-3 - Standard atlas language for modular test – 1995
[21] Arinc report 629, Multi-transmitter data bus – 1999
[22] IEEE 716 Standard, Test Language for All Systems – Common / Abbreviated Test Language for All Systems (c/atlas) – 1995
[23] IEEE 488.1 standard, Digital interface for programmable instrumentation – 1988
[24] IEEE 488.2 standard, Codes, formats, protocols, and common commands for use with IEEE std 488.1-1987, IEEE standard digital interface for programmable instrumentation – 1992
[25] IEEE 771 IEEE Guide to the use of the atlas specification – 1998
[26] IVI-3.1: Driver architecture specification - 2006
[27] IVI-3.4: Api style guide -2005
[28] IVI-4.1: Iviscope class specification - 2004
[29] IVI-4.2: Ividmm class specification -2004

[30] IVI-4.3: Fgen class specification - 2004
[31] IVI-4.4: Ividcpwr class specification -2004
[32] IVI-4.6: Iviswtch class specification -2004
[33] IVI-4.7: Ivipwrmeter class specification -2004
[34] IVI-4.8: Ivispecan class specification -2004
[35] IVI-4.10: ivirfsiggen class specification -2004
[36] CMM ATA 23-31-05 for ambient noise sensor p/n 285w0030 – 1998
[37] CMM ATA 23-92-01 for arinc signal gateway p/n 285w0020- 1999
[38] CMM ATA 23-39-14 for touch screen/lcd assembly p/n 285w0129 - 1999
[39] CMM ATA 23-42-0 cabin attendant handset p/n 285w0024-1 –1998
[40] CMM ATA 23-39-12 cabin system control panel p/n 285w0011-2 – 1999
[41] CMM ATA 23-39-11 cabin system management unit p/n285w0034-1 – 1996
[42] CMM ATA 21-27-04 environmental control system miscellaneous card p/n 285w0019-101 - 1996
[43] CMM ATA 33-16-08 master dim and test pwa p/n 285w0191-1 1996
[44] CMM ATA 23-33-11 overhead electronics unit p/n 285w0029-3 – 1995
[45] CMM ATA 23-93-01 overhead panel bus controller p/n 283w0219-1 – 1998
[46] CMM ATA 23-39-03 passenger address / cabin interphone p/n 285w0062-2 – 1997
[47] CMM ATA 31-09-01 linear/monitor card p/n 285w0037-101 – 1997
[48] CMM ATA 31-09-02 pre-regulator pwa for cardfile power supply p/n 285w0038-101 – 1998
[49] CMM ATA 23-12-04 radio tuning panel p/n 285w0114-1 -1997
[50] CMM ATA 23-31-02 speaker drive module p/n 285w0025 – 1997
[51] CMM ATA 23-34-05 seat electronics unit p/n 285w0035-1 1998
[52] CMM ATA 31-51-43 warning electronic unit p/n 285w0015-101 – 1996
[53] **Dan Simu** - Simulation based testing of complex electronic systems – Transactions on Electronics and Communications – buletinul stiintific al Universitatii " Politehnica" Timisoara - tom 49(63), fascicola 1, 2004.
[54] **Dan Simu** - Simulation based environment for automatic test and error injection - scientific bulletin "Politehnica" University of Timisoara, romania, transactions on automatic control and computer science, vol. 49(63) 2004 no. 1,2,3,4 / issn 1224-600x
[55] CMM ATA 27-59-01 flap/slat electronics unit p/n 285w0023-1 – 1998
[56] CMM ATA 23-39-04 zone management unit p/n 285w0027-1 -1996
[57] CMM ATA 23-92-02 overhead panel interface card p/n 285w0218-1 – 1996
[58] CMM ATA 26-10-04 cargo smoke detector p/n s218w301-1 - 1994
[59] CMM ATA 23-39-01 zone power converter p/n s906-70463-1 - 1994
[60] CMM ATA 27-20-02 rudder trim indicator p/n s231w231-2 - 1994
[61] CMM ATA 23-50-43 audio management unit p/n s222w101 – 1995
[62] CMM ATA 28-48-01 boeing 777 integrated refuel panel p/n s345w001-020 – 1995
[63] CMM ATA 23-34-02 entertainment multiplexer/controller p/n 285w0013-1 1994
[64] CMM ATA 29-11-02 hydraulic interface module p/n 285w0017- 101 – 1996

[65] CMM ATA 29-11-06 logic and speed control unit p/n s271w130- 33 – 1994
[67] CMM ATA 21-32-44 cabin pressure valve control unit p/n 2704474-2 – 1995
[68] CMM ATA 23-93-03 panel data concentrator unit p/n 285w0259-1 – 1995
[69] CMM ATA 23-39-50 cabin control panel p/n 285w0863 – 1999
[70] CMM ATA 23-39-04 css zone management unit p/n 285w0027-101 - 1999
[71] B.A.C. Universal test equipment manuals - 1975
[72] CMM ATA 27-32-44 artificial feel computer – concorde- 1975
[73] CMM ATA 33-15-11 master warning control unit – concorde- 1975
[74] CMM ATA 71-61-74 aicu computer unit – concorde - 1975
[75] CMM ATA 71-61-60 aitu computer unit – concorde - 1975
[76] CMM ATA 22-12-11 pitch computer unit – concorde- 1975
[77] CMM ATA 22-13-11 azimuth computer unit – concorde- 1975
[78] CMM ATA 22-22-11 auto stab computer unit – concorde- 1975
[79] CMM ATA 22-31-11 auto-throttle computer unit – concorde- 1975
[80] CMM ATA 78-31-80 nozzle angle scheduling unit – concorde- 1975
[81] Jahn Luke, Replacement strategy for aging avionics computers, ieee aes systems magazine, march, 1999
[82] Wilcock, G. Totten, T. Gleave, A. Wilson, r., The application of cots technology in future modular avionic systems, Electronics & communication engineering Journal vol. 13, nr. 4, - 2001
[83] Bode, F., Ivi comes of age: an overview of ivi specifications with current status, Autotestcon proceedings, 2002. ieee – 2002
[84] Cheij, D., Using interchangeable virtual instrument (ivi) drivers to increase test system performance, Aerospace and Electronic Systems magazine, IEEE, vol. 16, nr. 7, - 2001.
[85] Gutterman, L.,Integrating visa, ivi and ateasy to migrate legacy test systems, Aerospace and Electronic Systems magazine,IEEE, vol. 20, nr. 6 2005.
[86] Dowling, D. Rupinski, T., Avionics maintenance 2010, : Selected areas in communications, IEEE journal on, vol. 4, nr. 7 - 1986
[87] Cashar, E.E., Development of a tps reuse library using cots tools, Aerospace and Electronic Systems magazine, IEEE, vol. 12, nr. 10 - 1997.
[88] Gooding, M. Cohen, L., Evaluation of three ate test environments, Aerospace and Electronic Systems magazine, IEEE, vol. 12, nr. 9 – 1997
[89] Gutterman, L., Integrating visa, ivi and ateasy to migrate legacy test systems, Aerospace and Electronic Systems magazine, ieee, vol. 20, nr. 6 - 2005
[90] Pizzica, S., Open systems architecture solutions for military avionics testing, Aerospace and Electronic Systems magazine, IEEE, vol. 16, nr. 8 - 2001
[91] Rajsuman, R. Masuda, N. Yamashita, K., Architecture and design of an open ate to incubate the development of third-party instruments, instrumentation and measurement, ieee transactions on, vol. 54, nr. 5 - 2005
[92] Mueller, J. Oblad, R., Architecture drives test system standards, spectrum, IEEE – sept. 2000.
[93] Rolain, Y. Van Moer, W., Block-oriented instrument software design, instrumentation and measurement, IEEE transactions on, vol. 53, nr. 3 – 2004
[94] Craig, R.W., A methodology for addressing support equipment obsolescence, Aerospace and Electronic Systems magazine, ieee, vol.17, nr. 5 - 2002

[95] Cleary, R.T., A new deterministic vxi highway interconnect, Aerospace and Electronic Systems magazine, IEEE, vol. 11, nr. 3 - 1996
[96] Hill, G. , VXIbus revision 2.0-what's new?, Instrumentation & Measurement magazine, IEEE, vol. 2, nr. 2 – 1999.
[97] Dettmer, R., The vxi bus-an open standard for modulator instrumentation, : IEE review, vol. 35, nr.9 – 1989.
[98] Neblett, B., Implementing reusable, instrument independent test programs in the factory, Aerospace and Electronic Systems magazine, IEEE, vol. 12, nr. 6 – 1997.
[99] Mussmann, S.P., Gpib and the battle of incompatible languages, Instrumentation and measurement, IEEE transactions on, vol. 37, nr. 4 – 1988.
[100] Andrade, L. Tenning, C., Design of boeing 777 electric system, Aerospace and Electronic Systems magazine, ieee, vol. 7, nr. 7 – 1992.
[101] Eklow, B. Parker, K.P. Barnhart, c.f., IEEE 1149.6: a boundary-scan standard for advanced digital networks, Design & test of computers, IEEE , vol. 20, nr.5 – 2003.
[102] Pasquarette, John: Using ivi drivers to buildhardware - Independent test systems with LabView and LabWindows /CVI, National Instruments 1998
[103] Pasquarette, John: Using IVI drivers to build hardware-independent test systems with LabView and LabWindows/CVI, National Instruments 1998
[104] Pasquarette, John: Using IVI drivers to simulate your instrumentation hardware in LabView and LabWindows/CVI, National Instruments 1998
[105] Long-term instrument control and connectivity solutions – National Instruments 2006
[106] How ivi-c instrument driver technology enables system longevity and platform portability – National Instruments 2006
[107] The future for new bus technologies in instrument control and connectivity– National Instruments 2006
[108] Guo De-Gui, Liu Lei, Li Wen-Jin, Transformation from test language atlas to c++, the fifth international conference on computer and information technology (cit'05), 2005
[109] B.Blanc, G.Durrieu, A.Lakehal, O.Laurent, B.Marre, I. Parissis, C.Seguin, V. Wiels, Automated functional test case generation fromdata flow specifications using structural coverage criteria – Airbus france erts 2006
[110] Vx4469a Arinc 629 communication module, User manual 070-9147-00 - 1994
[111] 777 data load user requirements d220w110 - 1991
[112] HP e1340a arbitrary function generator user's manual -1992
[113] 7836 vxi lvdt simulator manual, North Atlantic Instruments -1993
[114] VXI 5390/5393 synchro/resolver processor, North Atlantic Instruments -1993
[115] Operators Manual model 2250 degital analyzing voltmeter, North Atlantic Instruments -1991
[116] 777 cas software load system requirements and interface control document, d906-70658 - 1994
[117] HP 75000 series c, c-size VXIbus systems, Installation and getting started guide - 1993
[118] HP 54503a 500 mhz digitizing oscilloscope , Programmer's reference - 1993

[119] HP 34401a multimeter, User's guide - 1992
[120] HP 8903b audio analyzer, Operation and calibration manual - 1989
[121] TestStand reference manual, Natioanal Instruments - 1993
[122] Using TestStand, Manual, National Instruments - 2003
[123] LabWindows/CVI Instrument driver developers guide, National Instruments - 2003
[124] Measurement studio user manual, National Instruments - 2006
[125] Stora, M.J. Droste, D.,"ATE open system platform" ieee-p1552 structured architecture for test systems (sats), Autotestcon 2003. IEEE systems readiness technology conference. proceedings – 2003
[126] Purcell, A., The search for a gpib replacement, Autotestcon '99. IEEE systems readiness technology conference, 1999. IEEE - 1999
[127] Rawnsley, D.J. Hummels, D.M. Segee, B.E., A Virtual instrument bus using network programming, Instrumentation and measurement technology conference, 1997. Imtc/97. proceedings. 'sensing, processing, networking'., IEEE - 1997
[128] Cram, R.S., Gpib compliance testing in a large test and measurement company, Autotestcon '94. ieee systems readiness technology conference. 'cost effective support into the next century', conference proceedings. – 1994
[129] Emmert, G.T., Considerations for implementing high performance vxi test systems, Autotestcon '98. IEEE systems readiness technology conference., 1998 IEEE-1998
[130] Johnson, M.W. Dayton, D.W., VXI lvdt simulation cots resource augments legacy commercial airline industry ate, Autotestcon proceedings, 2001. IEEE systems readiness technology conference – 2001
[131] Hetherington, D. , The vxibus: an introduction, buses for instruments: vxi and beyond, IEE colloquium on – 1989
[132] Nair, C., Modular test architectures for the aerospace industry, autotestcon proceedings, 2002. IEEE - 2002
[133] Rajendran, R., User-focused ivi-com driver development, Autotestcon 2003. IEEE systems readiness technology conference. proceedings - 2003
[134] Hulett, J., IVI drivers - new requirements for ivi conformance, Autotestcon 2004. proceedings – 2004
[135] Mueller, J.E., Achieving instrument interchangeability with ivi instrument drivers, Autotestcon 2003. IEEE systems readiness technology conference. proceedings - 2003
[136] Functional test requirements – Cargo system controller d906-70143 – 1993
[137] Functional test requirements – Master dim and test pwa d906-70372 – 1994
[138] Jones, S., A signal server software architecture, Autotestcon proceedings, 2000 IEEE - 2000
[139] Cherfas, A. Reeves, W., Atlas 2000 signal and method classification and modeling, Autotestcon '98. IEEE systems readiness technology conference., 1998 IEEE - 1998
[140] Hulme, A.M.B., The atlas language-panacea or pariah? does it only specify the task or does it really drive the tester?, Autotestcon '95. 'systems readiness: test technology for the 21st century'. conference record - 1995

[141] Oishi, R.T., Atlas extend, its effect on ate system software, Autotestcon '88. IEEE international automatic testing conference, futuretest. symposium proceedings - 1988
[142] Timcho, T.J., Bridging the gap between atlas and c: an open-systems approach to TPS re-host, Autotestcon '98. ieee systems readiness technology conference., 1998 IEEE - 1998
[143] Krizman, K.J. Duvall, J.A., Rf synthetic instrumentation: ats technology insertion and implications, Autotestcon 2003. IEEE systems readiness technology conference. proceedings - 2003
[144] Mesibov, c., Mms response to rapid rf ate development needs, Autotestcon '93. IEEE systems readiness technology conference. proceedings -1993
[145] Margaret Cadogan, Teresa Lopes, Specifying an ivi class for digital test instrumentation, Teradyne inc.
[146] IEEE standard protocol 1149.5. a standard module test and maintenance bus.- 1995
[147] Dave Rolince, Simplifying tps development and execution using a pc, web based environment, Teradyne, inc.
[148] Philip Stern, High-performance component software changes the rules for configuring ate, Assembly test division Teradyne inc.
[149] Dave Rolince, Applying virtual test principles to digital test program development, Teradyne, inc.
[150] Prashant S. Parikh, Miron Abramovici, Testability-based partial scan analysis, Journal of electronic testing: theory and applications - 1995
[151] Albert Helfrick, Principles of avionics, avionics communications inc - 2004
[152] Cary R. Spitzer, The avionics handbook, electrical handbook series , crc press - 2000
[153] IEEE std 1155-1992, IEEE standard for vmebus extensions for instrumentation: vxibus -1992
[154] Pierluigi san Pietro, Angelo Morzenti, Sandro Morasca, Generation of execution sequences for modular time critical systems, IEEE transactions on software engineering, vol. 26, nr. 2 - 2000
[155] A perspective on the state of research on fault injection techniques, University of Virginia, center for safety - critical systems - 2002
[156] Cheng-Wei chen, Jenq Kuen Lee, Case study: an infrastructure for C/Atlas environments with object-oriented design and xml representation, Journal of system and software - 2002
[157] Jan Peleska, Klemens Brumm, Gunnar Jonas, Tobias Hartmann, Advancement in automated simulation and testing technology for safety-critical avionic systems, University of Bremen; center of information technolgy - 2006
[158] Elgar operating manual, model sw 5250a -1996
[159] HP electronic load family, Programming reference manual -1991
[160] Paws user's guide dos/ms-windows, tyx corp. - 1995
[161] Paolo Donzelli, Roberto Marozza, Customizing the software process to support avionics systems enhancements, the journal of defense software engineering - 2001

[162] Barry, T. Scheffer, T. Small, L.R., An environment for the integration and test of the space station distributed avionics systems, Aerospace and Electronic Systems magazine, IEEE - 1988
[163] Pete Faulkner, Flexible, mixed architecture automated test systems uses vme/vxi/compactpci/pxi, Vmebus systems - 2003
[164] Chris Goringe, EADS, The pitfalls of replacing obsolete instrumentation, EE-evaluation engineering - 2006
[165] Douglass, K.; Worley, J.; Stehle, C., Choosing software and replacing ate: lessons learned, Aerospace and Electronic systems magazine, IEEE, vol 19, nr. 9 - 2004
[166] Ted Miller, Thomas J. Gallagher, VXI-based functional ate, ee-evaluation engineering - 1998
[167] Nate D'anna, Cots software prevents ate obsolescence, Test & measurement world - 2006
[168] Eric Sacher, A system view of vxi ate design, ee-evaluation engineering - 1997
[169] Sergio M Perez, The critical need for open ate architecture, International test conference (itc'04) - 2004
[170] Burnell G. West, Michael F. Jones, Digital synchronization for reconfigurable ate, International test conference (itc'04) - 2004
[171] Shanrui Zhang, Minsu Choi, Nohpill Park, Fabrizio Lombardi, Probabilistic balancing of fault coverage and test cost in combined built-in self-test/automated test equipment testing environment, 19th IEEE international symposium on defect and fault tolerance in vlsi systems (dft'04) - 2004
[172] M. Miegler, W. Wolz, Development of test programs in a virtual test environment, 14th IEEE vlsi test symposium (vts '96) - 1996
[173] Guangfan Shi, Guangming Yan, Jigang Li, Guanran Wang, Zeguo Cheng, The design and implement of virtual instrument based on computing technique and usb platform, Third international conference on information technology and applications (icita'05) volume 1 - 2005
[174] Armando Carbonari, Avionic systems overview, Proceedings of the 17th symposium on integrated circuits and system design (sbcci '04) - 2004
[175] Bob Stasonis, pxi - A new architecture for many testing requirements, International test conference 2003 - 2003
[176] Standards, IEEE design and test of computers - 2001
[177] Eric Starkloff, Tim Fountain, Garth Black, the pxi modular instrumentation architecture, International test conference 2003 -2003
[178] Wolfe, R, Virtual instruments in vxi, Autotestcon '93. ieee systems readiness technology conference. proceedings - 1993
[179] Cristaldi, L.; Ferrero, A.; Piuri, V., Programmable instruments, virtual instruments, and distributed measurement systems: what is really useful, innovative and technically sound?, Instrumentation & measurement magazine, IEEE, vol. 2, nr. 3 - 1999
[180] Rabe, D. Miller, J., Applying software process to virtual instrument based test program set development, : Autotestcon '97. 1997 IEEE autotestcon proceedings - 1997

[181] Bearse, T.M. Lynch, M.L., Translating test programs using a model-based process, Autotestcon '99.IEEE systems readiness technology conference, 1999. IEEE - 1999
[182] Deng, B.; Glauert, W., Formal description of test specification and ate architecture for mixed-signal test, test conference, 2004. proceedings. Itc 2004. international – 2004.
[183] Perez, S.M., The consequences of an open ate architecture, Test conference, 2002. Proceedings. international - 2002
[184] West, B.G., Open ate architecture: key challenges, Test conference, 2002. Proceedings. international - 2002
[185] Orlidge, L.A. Stoll, E.D., Measurement hardware emulator: synthetic instrumentation and cass, : Autotestcon '99. IEEE systems readiness technology conference, 1999. IEEE - 1999
[186] Crowe, D. Matysek, G. , The need for multi-service test equipment standardization, autotestcon '91. IEEE systems readiness technology conference. improving systems effectiveness in the changing environment of the '90s, conference record. - 1991
[187] Kennedy, C., Sustainment of legacy automatic test systems: lessons learned on TPS transportability, aerospace and electronic systems magazine, ieee - 2005
[188] Proceedings international test conference 2001 (cat. no.01ch37260), Test conference, 2001. proceedings. international - 2001
[189] Simpson, W.R.; Sheppard, J.W., Developing intelligent automatic test equipment, aerospace and electronics conference, 1991. naecon 1991., proceedings of the IEEE 1991 national -1991
[190] Jian Hou; Weiguo Feng, A vxi-based automatic test solution for avionics: issues and implementation, Autotestcon proceedings, 2000 IEEE - 2000
[191] Pestana, P.A., C and C/Atlas: the logical solution for the future, Autotestcon '95. 'systems readiness: test technology for the 21st century'. conference record -1995
[192] Linn, R.J., jr., Conformance evaluation methodology and protocol testing, Selected areas in communications, IEEE journal on – 1989
[193] Ramachandran, N., The role of (software) standards in test (avionics testing), Autotestcon 2003. IEEE systems readiness technology conference. proceedings - 2003
[194] Sacher, E.; Lonngren, D., Nondenominational digital test programming for functional test, Autotestcon 2003. IEEE systems readiness technology conference. proceedings – 2003
[195] Ellis, K.; Delaney, d., Signal definition and test description - an IEEE standard, Autotestcon proceedings, 2002. ieee - 2002
[196] Blair, M., Tedl - a new test interface standard from the ieee, Autotestcon '97. 1997 IEEE autotestcon proceedings – 1997
[197] Seavey, M., The complex signal C/Atlas structure: a signal description construct for the future, Autotestcon '92. IEEE systems readiness technology conference, conference record - 1992
[198] Jurcak, T., An instrument-independent test software framework allows both hardware and software reuse, Autotestcon '97. 1997 IEEE autotestcon proceedings - 1997

[199] Hudis, E.; Greenspan, A.M., Atlas2000 object oriented programming capabilities, Autotestcon '98. IEEE systems readiness technology conference., 1998 ieee - 1998.
[200] IEEE standard for signal and test definition, IEEE std 1641-2004 – 2005
[201] Stoica, S., System design verification tests - an overview, Test conference, 1999. proceedings. international - 1999
[202] HP 603xA user's manual.
[203] **Dan Simu** , Avionics functional test programs portability, Analele Universităţii din Oradea, fascicola Electrotehnică, secţiunea electronică, pag. 214-218, -2006
[204] **Dan Simu** , Useful Intermediate language for running and translating test programs, Analele Universităţii din Oradea, fascicola Electrotehnică, secţiunea electronică, pag. 219-223, -2006
[205] **Dan Simu**, Echipamente de testare automata, referat doctorat – 1999
[206] **Dan Simu**, Portabilitatea programelor de testare.Clase de instrumente virtuale, referat doctorat - 1999

Nota: CMM = Component Maintenance Manual
ATA = Air Transport Association
XX-XX-XX = numărul de înregistrare ATA

ANEXA 1

Descrierea în limbaj ATLAS 616 a testelor din blocul 2.0-CPU-RAM-CHECKS pentru unitatea AUTOTHROTTLE tip 10-62017-21 pentru avioanele Boeing 737-500 produs de firma Smith Industries Aerospace

```
C$
C ********************************************************** $
C *                    MAIN  PROCEDURE                     * $
C ********************************************************** $
C$
C$
C ********************************************************** $
C **                                                      ** $
 200000 BEGIN, BLOCK, '2.0-CPU-RAM-CHECKS' $
C **                                                      ** $
C ********************************************************** $
C *                                                        * $
C *      FUNCTIONAL DESCRIPTION                            * $
C *        THIS BLOCK CARRIES OUT THE CPU MICROSTEP, STORE RAM  $
C *        UPPER, STORE RAM LOWER AND PROGRAM CONTENTS CHECKS   $
C *        IF SELECTED VIA THE OPTIONS PROCEDURE.          * $
C *        THE FAULT IS GIVEN.                             * $
C *                                                        * $
C *        ---------------------- TEST DIRECTORY ---------- * $
C *        SUB-BLOCK                                STATEMENT * $
C *        NUMBER          BLOCK TITLE              NUMBER    * $
C *        ------------------------------------------------- * $
C *        2.1             '2.1-CPU-USTEP'          210000    * $
C *        2.2             '2.2-STORE-RAM-UPPER'    220000    * $
C *        2.3             '2.3-STORE-RAM-LOWER'    230000    * $
C *        2.4             '2.4-STORE-PROGRAMME-CONTENTS' 240000 * $
C *        ------------------------------------------------- * $
C ********************************************************** $
C *                                                        * $
C *        PROCEDURES USED                                 * $
C *                        'INITCHECK'              210300    * $
C *                                                 220300    * $
C *                                                 230300    * $
C *                                                 240300    * $
C *                        'LOADCHECK'              210400    * $
C *                                                 220400    * $
C *                                                 230400    * $
C *                                                 240400    * $
C *                        'READ-DVM'               211100    * $
C *                                                 212000    * $
C *                                                 213010    * $
C *                                                 248000    * $
C *                        'FAILSTOP'               211950    * $
C *                                                 212750    * $
C *                                                 215750    * $
C *                                                 217100    * $
C *                                                 222600    * $
C *                                                 232600    * $
C *                                                 246600    * $
C *                                                 243406    * $
C *                                                 243520    * $
C *                        'GO-AROUND'              212000    * $
C *                                                 221010    * $
C *                                                 231010    * $
C *                                                 246200    * $
C *                        'SET-707070/'            221000    * $
C *                                                 231000    * $
C *                        'RAM-CHECK'              221110    * $
C *                                                 231110    * $
C *                        'SRAM-PRINT'             221120    * $
C *                                                 231120    * $
C *                        'PRINT-TITLE'            246340    * $
C *                                                 247130    * $
C *                                                 248400    * $
C *                        'CHECK'                  248110    * $
C *                        'BLOCK-PES'              247410    * $
C ********************************************************** $
C$
C$
C        ------------------------------------------------- $
C        I LEAVE BLOCK IF NONE OF THE TESTS HAVE BEEN SELECTED  I $
C        I I.E. 'OPTION' (1 TO 4) ALL AT 0                      I $
C        ------------------------------------------------- $
C$
C$
C ********************************************************** $
C *       BLOCK 2.1 CPU MICROSTEP                          * $
C ********************************************************** $
C *       THIS BLOCK CARRIES OUT THE CPU MICROSTEP AS FOLLOWS  * $
C *       THE UUT IS SET TO INITIAL CONDITIONS AND THE CPU     * $
C *       DOWNLOAD PROGRAMME IS LOADED INTO THE UUT.           * $
C *       IF THE LOAD IS OK THEN TEST PROCEEDS AS FOLLOWS      * $
C *                                                            * $
C *       OUTPUT WORDS 240 AND 220 BITS 15 THROUGH 4 ARE CHEKED * $
C *       THAT THEY ARE INITIALLY ALL IN THE ON (LOGIC 1) STATE. * $
C *                                                            * $
C *       A GO-AROUND IS THEN CARRIED OUT AND BITS 15 THROUGH 4 * $
C *       OF WORD 240 AND BITS 3 THROUGH 4 OF WORD 220 GO TO THE * $
C *       OFF (LOGIC 0) STATE. THE UUT THEN PROCEEDS TO STEP   * $
C *       THROUGH ITS CPU INSTRUCTION SET OUTPUTTING THE       * $
C *       INSTRUCTION NUMBER ON BITS 15 THROUGH 11 OF WORD 240 * $
C *       AND AT THE END ALL THE BITS RETURN TO THE ON STATE.  * $
C *                                                            * $
C *       IF A FAILURE OCCURS THE CPU INSTRUCTION SET COUNT HALTS * $
C *       WITH AN INSTRUCTION NUMBER ON BITS 15 THROUGH 11 OF  * $
C *       WORD 240                                             * $
C *                                                            * $
C *       IF NO PROGRAMME CORRUPTION OCCURED (I.E. LR17, LR18  * $
C *       AND LR42 ALL AT 0V) THEN THIS INSTRUCTION NUMBER IS  * $
C *       THE NUMBER OF THE INSTRUCTION THAT FAILED.           * $
C *                                                            * $
C *       IF A PROGRAMME CORRUPTION DID OCCUR (I.E. ANY OF LR17, * $
C *       LR18 OR LR42 NOT AT 0.0V) THEN THIS INSTRUCTION NUMBER * $
C *       IS THE NUMBER OF THE LAST SUCCESSFUL INSTRUCTION STEP * $
C *       TEST.                                                * $
C *                                                            * $
C *       THE FOLLOWING TABLE PROVIDES A CROSS REFERENCE BETWEEN * $
C *       WORDS 220 AND 240 AND THEIR ASSOCIATED UUT PINS. ALSO * $
C *       INCLUDED IN THE TABLE ARE THE ASSOCIATED LAMP NUMBERS * $
C *       AS USED IN THE BENCH TEST SET (BTS) ATP FOR THE A/T. * $
C *                                                            * $
C *       ------------------------------------------------------ * $
C *       I                                                    I * $
C *       I    OUTPUT  UUT     BTS     LOGIC 1(ON)    LOGIC 0(OFF) I * $
C *       I WORD BIT   PIN     LAMP    PIN     LAMP   PIN     LAMP I * $
C *       I ------- --- ------ ------  ------- ------ ------- ------ I * $
C *       I 220  15    LL6     20     28.0V    ON    00.0V    OFF  I * $
C *       I 220  14    LR38    21     28.0V    ON    00.0V    OFF  I * $
C *       I 220  13    LR28    22     28.0V    ON    00.0V    OFF  I * $
C *       I 220  12    LR27    23     28.0V    ON    00.0V    OFF  I * $
C *       I 220  11    LR45    24     00.0V    ON    28.0V    OFF  I * $
C *       I 220  10    LR46    25     00.0V    ON    28.0V    OFF  I * $
C *       I 220   9    LR23    26     00.0V    ON    28.0V    OFF  I * $
C *       I 220   8    LL23    27     00.0V    ON    28.0V    OFF  I * $
C *       I 220   7    LR25    40     00.0V    ON    28.0V    OFF  I * $
C *       I 220   6    LR32    41     00.0V    ON    28.0V    OFF  I * $
C *       I 220   5    IR12    42     00.0V    ON    28.0V    OFF  I * $
C *       I 220   4    IR16    43     00.0V    ON    28.0V    OFF  I * $
C *       I                                                    I * $
C *       I 240  15    LR6     28     00.0V    ON    28.0V    OFF  I * $
C *       I 240  14    LR7     29     00.0V    ON    28.0V    OFF  I * $
C *       I 240  13    LR15    30     00.0V    ON    28.0V    OFF  I * $
C *       I 240  12    LR44    31     00.0V    ON    28.0V    OFF  I * $
C *       I 240  11    LR36    32     00.0V    ON    28.0V    OFF  I * $
C *       I 240  10    LR43    33     00.0V    ON    28.0V    OFF  I * $
C *       I 240   9    LR26    34     00.0V    ON    28.0V    OFF  I * $
C *       I 240   8    LR14    35     00.0V    ON    28.0V    OFF  I * $
```

ANEXA 1

Descrierea în limbaj ATLAS 616 a testelor din blocul 2.0-CPU-RAM-CHECKS pentru unitatea AUTOTHROTTLE tip 10-62017-21 pentru avioanele Boeing 737-500 produs de firma Smith Industries Aerospace

```
C *       I 240   7  LR33   36      00.0V    ON    28.0V    OFF  I    * $
C *       I 240   6  LR24   37      00.0V    ON    28.0V    OFF  I    * $
C *       I 240   5  LR35   38      00.0V    ON    28.0V    OFF  I    * $
C *       I 240   4  LR34   39      00.0V    ON    28.0V    OFF  I    * $
C *       I                                                     I    * $
C *       I                                                     I    * $
C *       I VOLTAGES PRESENT ON THE UUT PINS IN THE LOGIC 0 OR  I    * $
C *       I OFF STATE ARE PROVIDE BY PULL UP OR DOWN LOADS      I    * $
C *       I                                                     I    * $
C *       ------------------------------------------------------     * $
C *                                                                  * $
C *       THE FOLLOWING TABLE IS A LIST OF THE CPU MICROSTEP         * $
C *       TESTS ASSOCIATED WITH EACH OCTAL TEST NUMBER.              * $
C *                                                                  * $
C *       ---------  ------------------------------------           * $
C *       I                                                 I        * $
C *       I OCTAL                                           I        * $
C *       I NUMBER   TEST                                   I        * $
C *       I --------  ------------------------------------   I        * $
C *       I 174000   START                                  I        * $
C *       I 004000   SATURATED ADD                          I        * $
C *       I 010000   SATURATED SUBTRACT                     I        * $
C *       I 014000   BIT GROUP SKIP                         I        * $
C *       I 020000   SKIP REGISTER LESS THAN OR EQUAL TO STORE I     * $
C *       I 024000   ADD CARRY HOLD                         I        * $
C *       I 030000   SUBTRACT CARRY HOLD                    I        * $
C *       I 034000   LOGICAL AND                            I        * $
C *       I 040000   LOGICAL OR                             I        * $
C *       I 044000   EXCLUSIVE OR                           I        * $
C *       I 050000   MULTIPLY                               I        * $
C *       I 054000   DIVIDE                                 I        * $
C *       I 060000   LEFT  SHIFT ARITHMETIC (SATURATING)    I        * $
C *       I 064000   LEFT  SHIFT ARITHMETIC A REGISTER      I        * $
C *       I 070000   LEFT  SHIFT ARITHMETIC B REGISTER      I        * $
C *       I 074000   RIGHT SHIFT LOGICAL    A REGISTER     I        * $
C *       I 100000   RIGHT SHIFT LOGICAL    B REGISTER     I        * $
C *       I 104000   RIGHT SHIFT ARITHMETIC                 I        * $
C *       I 110000   LOAD OPPOSITE STATE/CHANGE STATE       I        * $
C *       I 114000   FETCH 20 BITS LOOK AT A REGISTER       I        * $
C *       I 120000   FETCH 20 BITS LOOK AT B REGISTER       I        * $
C *       I 124000   LOGICAL AND SPECIAL REGISTER           I        * $
C *       I 130000   STORE SPECIAL REGISTER                 I        * $
C *       I 134000   MODIFY INDIRECT ADDRESSING             I        * $
C *       I 140000   INDIRECT ADDRESSING                    I        * $
C *       I 144000   MODIFIED ADDRESSING                    I        * $
C *       I 150000   INDIRECT MODIFIED ADDRESSING           I        * $
C *       I 154000   RELATIVE JUMP (GO/NO GO)               I        * $
C *       I 160000   SUBTRACT FROM SPECIAL REGISTER         I        * $
C *       I 164000   DECREMENT SPECIAL REGISTER AND JUMP IF  I        * $
C *       I          NOT EQUAL TO 0                         I        * $
C *       I 170000   SKIP IF REGISTER NOT EQUAL TO STORE    I        * $
C *       I          (GO/NO GO)                             I        * $
C *       I 174000   SUBTRACT                               I        * $
C *       ---------  ------------------------------------   I        * $
C *                                                                  * $
C ****************************************************************** $
C$
 210000   BEGIN, BLOCK, '2 1-CPU-USTEP' $
C$
C          ------------------------------------------------------    $
C          I EXIT BLOCK IF TEST NOT SELECTED                  I $
C          ------------------------------------------------------    $
C$
 210100     IF, 'OPTION'(1) NE 1, THEN $
      20      LEAVE, BLOCK $
      30     END, IF $
 210200     CALCULATE, 'J' = 1 $
      10     CALCULATE, 'RESLTS'('J') = 1 $
      20     FILL, 'TNAM', C'CPU MICROSTEP          ' $
C$
```

```
C          ------------------------------------------------------    $
C          I INITIALISE UUT.   EXIT BLOCK IF FAILURE OCCURS   I $
C          ------------------------------------------------------    $
C$
 210300     PERFORM, 'INITCHECK', C'CPU MICROSTEP          ' $
      10   IF, 'RESLTS'('J') LT 0, THEN $
             DISPLAY, MESSAGE, CPU MICROSTEP INITIALISATION FAILED $
      30     DISPLAY, MESSAGE,                                    $
      40     PRINT,   MESSAGE, CPU MICROSTEP INITIALISATION FAILED $
 210350      PRINT,   MESSAGE,                                    $
      60     REMOVE, ALL $
      70     LEAVE, BLOCK $
      80    END, IF $
C$
C          ------------------------------------------------------    $
C          I DOWLOAD PROGRAM.  EXIT BLOCK IF FAILURE OCCURS   I $
C          ------------------------------------------------------    $
C$
 210400     PERFORM, 'LOADCHECK', C'CPU MICROSTEP          ' $
      10   IF, 'RESLTS'('J') LT 0, THEN $
             DISPLAY, MESSAGE, CPU MICROSTEP DOWNLOAD       FAILED $
      30     DISPLAY, MESSAGE,                                    $
      40     PRINT,   MESSAGE, CPU MICROSTEP DOWNLOAD       FAILED $
 210450      PRINT,   MESSAGE,                                    $
      60     REMOVE, ALL $
      70     LEAVE, BLOCK $
      80    END, IF $
C$
C          ------------------------------------------------------    $
C          I VERIFY WORDS 220 AND 240 BITS 15 THROUGH 4 ALL ON I $
C          ------------------------------------------------------    $
C$
 211000     FILL,   'G1',
                          (1)  28,
                          (2)  28,
                          (3)  28,
                          (4)  28,
                          (5)   0,
                          (6)   0,
                          (7)   0,
                          (8)   0,
                          (9)   0,
                         (10)   0,
                         (11)   0,
                         (12)   0,
                         (13)   0,
                         (14)   0,
                         (15)   0,
                         (16)   0,
                         (17)   0,
                         (18)   0,
                         (19)   0,
                         (20)   0,
                         (21)   0,
                         (22)   0,
                         (23)   0,
                         (24)   0 $
 211010     FILL, 'PINS',
                          (1) C'LL6 ',
                          (2) C'LR38',
                          (3) C'LR28',
                          (4) C'LR27',
                          (5) C'LR45',
                          (6) C'LR46',
                          (7) C'LR23',
                          (8) C'LL23',
                          (9) C'LR6 ',
                         (10) C'LR7 ',
                         (11) C'LR15',
                         (12) C'LR44',
                         (13) C'LR36',
```

ANEXA 1

Descrierea în limbaj ATLAS 616 a testelor din blocul 2.0-CPU-RAM-CHECKS pentru unitatea AUTOTHROTTLE tip 10-62017-21 pentru avioanele Boeing 737-500 produs de firma Smith Industries Aerospace

```
                    (14) C'LR43',
                    (15) C'LR26',
                    (16) C'LR14',
                    (17) C'LR33',
                    (18) C'LR24',
                    (19) C'LR35',
                    (20) C'LR34',
                    (21) C'LR25',
                    (22) C'LR32',
                    (23) C'UR12',
                    (24) C'UR16' $
C0
 211100     PERFORM, 'READ-DVM', TABLE 3 VAR 24 TIMES,
              LL6,          UR7,          1,
              LR38,         UR7,          2,
              LR26,         UR7,          3,
              LR27,         UR7,          4,
              LR45,         UR7,          5,
              LR46,         UR7,          6,
              LR23,         UR7,          7,
              LL23,         UR7,          8,
              LR6,          UR7,          9,
              LR7,          UR7,          10,
              LR15,         UR7,          11,
              LR44,         UR7,          12,
              LR36,         UR7,          13,
              LR43,         UR7,          14,
              LR26,         UR7,          15,
              LR14,         UR7,          16,
              LR33,         UR7,          17,
              LR24,         UR7,          18,
              LR35,         UR7,          19,
              LR34,         UR7,          20,
              LR25,         UR7,          21,
              LR32,         UR7,          22,
              UR12,         UR7,          23,
              UR16,         UR7,          24 $
C             -----         ------        ---- $
C           HI PIN        LO PIN        INDEX $
C           ------        ------          --- $
C9
C9
C
C           ------------------------------------------------------------ $
C           I DISPLAY RESULTS AND SET RESULTS BUFFER IF FAIL           I $
C           ------------------------------------------------------------ $
C9
 211200     DISPLAY, MESSAGE, ('MESS24PINS') $
     10     CALCULATE, 'PRINT' = 'OPTION'(26) $
 211300     FOR, 'INDEX1' = 1 THRU 24, THEN $
     10       CALCULATE, 'Z1' = 'C1'('INDEX1') $
     20       CALCULATE, 'Z2'    'B'('INDEX1') $
C8
 211400       IF, ABS('Z2' - 'Z1') GT 3.0, THEN $
     10         FILL, 'FRESULT', C'FAILED' $
     20         CALCULATE, 'RESLTS'(1) = -1 $
     30         CALCULATE, 'PRINT' = 1 $
     40       ELSE $
 211450         FILL, 'FRESULT', C'PASSED' $
     60       END, IF $
C8
     70       DISPLAY, MESSAGE, ('MESSCPURES', 'PINS'('INDEX1'),
                                 'Z1', 'Z2', 'FRESULT') $
     80     END, FOR $
C9
C           ------------------------------------------------------------ $
C           I IF 'PRINT'=1 I.E. TEST FAILED OR FULL PRINTOUT MODE      I $
C           I GIVE A PRINTOUT OF THE RESULTS                           I $
C           ------------------------------------------------------------ $
C8
 211500     IF, 'PRINT' EQ 1, THEN $
     10       PRINT, MESSAGE, ('MESS24PINS') $
```

```
 211600       FOR, 'INDEX1' = 1 THRU 24, THEN $
     10         CALCULATE, 'Z1' = 'C1'('INDEX1') $
     20         CALCULATE, 'Z2' =  'B'('INDEX1') $
C8
 211700         IF, ABS('Z2' - 'Z1') GT 3.0, THEN $
     10           FILL, 'FRESULT', C'FAILED' $
     20         ELSE $
     30           FILL, 'FRESULT', C'PASSED' $
     40           IF, 'OPTION'(26) EQ 0, THEN $
 211750             RESUME, FOR $
     60           END, IF $
     70         END, IF $
C8
     80         PRINT, MESSAGE, ('MESSCPURES', 'PINS'('INDEX1'),
                                 'Z1', 'Z2', 'FRESULT') $
     90       END, FOR $
 211800     END, IF $
C8
C           ------------------------------------------------------------ $
C           I IF FAILED ABORT THE TEST AFTER A FAILSTOP                I $
C           ------------------------------------------------------------ $
C8
 211900   0 IF, 'RESLTS'(1) LT 0, THEN $
     10       CALCULATE, 'J' = 1 $
     20       FILL, 'TNAM', C' CPU MICROSTEP TEST      ' $
     30       PERFORM, 'BLOCK-RES' $
     40       REMOVE, ALL $
 211950       LEAVE, BLOCK $
     60     END, IF $
C8
C9
C           ------------------------------------------------------------ $
C           I PERFORM A GO-AROUND AND VERIFY THAT CPU INSTRUCTION      I $
C           I SET COUNT THROUGH HAS BEEN INITIATED.                    I $
C           I INITIATION OF COUNT SEQUENCE CAN BE VERIFIED BY          I $
C           I CHECK BITS 15 AND 10 OF WORD 240 BOTH GO TO LOGIC 0      I $
C           I FROM LOGIC 1  2 SECONDS AFTER PERFORMING GO AROUND       I $
C           ------------------------------------------------------------ $
C8
 212000     PERFORM, 'GO-AROUND' $
     10     DELAY, 2SEC $
     20     PERFORM, 'READ-DVM', TABLE 3 VAR 2 TIMES,
              LR43,     UR7,      1,
              LR6,      UR7,      2 $
C             -----     ------    ---- $
C           HI PIN    LO PIN    INDEX $
C           ------    ------    ----- $
C8
 212100     DISPLAY, MESSAGE,                                            $
     10     DISPLAY, MESSAGE, ------------------  ----------  ---------- $
     20     DISPLAY, MESSAGE, CPU MICROSTEP TEST                         $
     30     DISPLAY, MESSAGE, CHECK FOR INITIATION OF CPU COUNT          $
     40     DISPLAY, MESSAGE,                                            $
 212150     DISPLAY, MESSAGE, 2 SECONDS AFTER GO-AROUND                  $
     60     DISPLAY, MESSAGE, LR43 AND LR6 SHOULD BOTH BE 28+-2V         $
     70     DISPLAY, MESSAGE, ------------------  ----------  ---------- $
     80     FILL, 'PINS',
                    (1) C'LR43',
                    (2) C'LR6 ' $
C8
 212200     FOR, 'INDEX1' = 1 THRU 2, THEN $
     10       CALCULATE, 'Z1' = 'B'('INDEX1') $
C8
 212300       IF, ABS('Z1'-28) GT 2, THEN $
     10         FILL, 'FRESULT', C'FAILED' $
     20         CALCULATE, 'RESLTS'(1) = -1 $
     30         CALCULATE, 'PRINT' = 1 $
     40       ELSE $
 212350         FILL, 'FRESULT', C'PASSED' $
     60       END, IF $
     70       DISPLAY, MESSAGE, ('MSCPU2', 'PINS'('INDEX1'), 'Z1',
```

ANEXA 1

Descrierea în limbaj ATLAS 616 a testelor din blocul 2.0-CPU-RAM-CHECKS pentru unitatea AUTOTHROTTLE tip 10-62017-21 pentru avioanele Boeing 737-500 produs de firma Smith Industries Aerospace

```
                                    'FRESULT') $
     80        END, FOR $
C$
C              --------------------------------------------------- $
C              I GIVE PRINTOUT IF 'PRINT' = 1                    I $
C              I I.E. FULL PRINTOUT MODE OR THIS TEST FAILED     I $
C              --------------------------------------------------- $
C$
 212400        IF, 'PRINT' EQ 1, THEN $
     10          PRINT, MESSAGE,                                    $
     20          PRINT, MESSAGE,      ------------------------      $
     30          PRINT, MESSAGE, CPU MICROSTEP TEST                 $
     40          PRINT, MESSAGE, CHECK FOR INITIATION OF CPU COUNT  $
 212450          PRINT, MESSAGE,                                    $
     60          PRINT, MESSAGE, 2 SECONDS AFTER GO-AROUND          $
     70          PRINT, MESSAGE, LR43 AND LR6 SHOULD BOTH BE 28+-2V $
     80          PRINT, MESSAGE, ---------------------------------- $
C$
 212500          FOR, 'INDEX1' = 1 THRU 2, THEN $
     10            CALCULATE, 'Z1' = 'B'('INDEX1') $
C$
 212600            IF, ABS('Z1'-20) GT 2, THEN $
     10              FILL, 'FRESULT', C'FAILED' $
     20            ELSE $
     30              FILL, 'FRESULT', C'PASSED' $
     40            END, IF $
 212650            DISPLAY, MESSAGE, ('MSCPU2', 'PINS'('INDEX1'), 'Z1',
                                       'FRESULT') $
     60          END, FOR $
     70        END, IF $
C$
 212700        IF, 'RESLTS'(1) LT 0, THEN $
     10          DISPLAY, MESSAGE, CPU COUNT HAS NOT STARTED    FAILED $
     20          DISPLAY, MESSAGE,                                     $
     30          PRINT,   MESSAGE, CPU COUNT HAS NOT STARTED    FAILED $
     40          PRINT,   MESSAGE,                                     $
 212750          PERFORM, 'FAILSTOP' $
     60          REMOVE, ALL $
     70          LEAVE, BLOCK $
 212800        ELSE $
     10          DISPLAY, MESSAGE, CPU COUNT HAS STARTED               $
     20          DISPLAY, MESSAGE,                                     $
     30          IF, 'OPTION'(26) EQ 1, THEN $
     40            PRINT, MESSAGE, CPU COUNT HAS STARTED               $
 212850            PRINT, MESSAGE,                                     $
     60          END, IF $
     70        END, IF $
C$
C              --------------------------------------------------- $
C              I CPU INSTRUCTION SET CYCLE WILL BE COMPLETE WITHIN I $
C              I 30 SECONDS. DELAY 30 SECONDS AND THEN            I $
C              I VERIFY WORDS 220 AND 240 BITS 15 THROUGH 4 ALL ON I $
C              I ALSO READ VOLTAGES PRESENT ON LR42 LR17 AND LR18 I $
C              --------------------------------------------------- $
C$
 213000        FILL, 'PINS',
                                 (1) C'LL6 ',
                                 (2) C'LR38' $
 213010        PERFORM, 'READ-DVM', TABLE ) VAR 27 TIMES,
                   LL6,         UR7,         1,
                   LR38,        UR7,         2,
                   LR28,        UR7,         3,
                   LR27,        UR7,         4,
                   LR45,        UR7,         5,
                   LR46,        UR7,         6,
                   LR23,        UR7,         7,
                   LL23,        UR7,         8,
                   LR6,         UR7,         9,
                   LR7,         UR7,        10,
                   LR15,        UR7,        11,
                   LR44,        UR7,        12,
                   LR36,        UR7,        13,
                   LR43,        UR7,        14,
                   LR26,        UR7,        15,
                   LR14,        UR7,        16,
                   LR33,        UR7,        17,
                   LR24,        UR7,        18,
                   LR35,        UR7,        19,
                   LR34,        UR7,        20,
                   LR25,        UR7,        21,
                   LR32,        UR7,        22,
                   UR12,        UR7,        23,
                   UR16,        UR7,        24,
                   LR42,        UR7,        25,
                   LR17,        UR7,        26,
                   LR18,        UR7,        27   $
C                  ------       ------      ----- $
C                  HI PIN       LO PIN      INDEX $
C                  ------       ------      ----- $
C$
C              ---------------------------------------------------------- $
C              I CHECK OUTPUT WORDS 240 AND 220 BITS 15 TO 4 ALL ON     I $
C              I IF NOT SET Z TO OCTAL CODE OF INSTRUCTION PRESENT      I $
C              I ON OUTPUT PINS                                         I $
C              I INITIALLY SET Z TO 177777                              I $
C              I SET PRINT INDICATOR TO 1 IF FAIL OR FULL PRINT         I $
C              I                                                        I $
C              I IF OUTPUT IS INCORRECT THEN                            I $
C              I SETTING OF Z TO OCTAL CODE ON PINS LR6 TO LR36         I $
C              I VOLTAGE ON PIN      LR6     LR7   LR15  LR44  LR36     I $
C              I STORED  IN 'B'      (9)     (10)  (11)  (12)  (13)     I $
C              I OCTAL VALUE IF 28V  0       0     0     0     0        I $
C              I OCTAL VALUE IF  0V  100000  40000 20000 10000 4000     I $
C              ---------------------------------------------------------- $
C$
 213100        CALCULATE, 'PRINT' = 'OPTION'(26) $
     10        CALCULATE, 'Z' = 177777 $
C$
 213200        FOR, 'INDEX1' = 1 THRU 24, THEN $
     10          CALCULATE, 'Z1' = 'C1'('INDEX1') $
     20          CALCULATE, 'Z2' =  'B'('INDEX1') $
     30          IF, ABS('Z2' - 'Z1') GT 3.0, THEN $
 213300            IF, 'Z' EQ 177777, THEN $
     10              CALCULATE, 'Z' = 0 $
 213400              IF, 'B'(9) LT 3, THEN $
     10                CALCULATE, 'Z' = 'Z' + 100000 $
     20              END, IF $
 213450              IF, 'B'(10) LT 3, THEN $
     60                CALCULATE, 'Z' = 'Z' + 40000 $
     70              END, IF $
 213500              IF, 'B'(11) LT 3, THEN $
     10                CALCULATE, 'Z' = 'Z' + 20000 $
     20              END, IF $
 213550              IF, 'B'(12) LT 3, THEN $
     60                CALCULATE, 'Z' = 'Z' + 10000 $
     70              END, IF $
 213600              IF, 'B'(13) LT 3, THEN $
     20                CALCULATE, 'Z' = 'Z' + 4000 $
     30              END, IF $
     40            END, IF $
 213650            LEAVE, FOR $
 213700          END, IF $
     10        END, FOR $
C$
 213800        IF, 'Z' NE 177777, THEN $
     10          CALCULATE, 'PRINT' = 1 $
     20        END, IF $
C$
C              ---------------------------------------------------------- $
C              I DISPLAY AND (IF REQUIRED) PRINT RESULT                 I $
C              ---------------------------------------------------------- $
C$
```

12

ANEXA 1

Descrierea în limbaj ATLAS 616 a testelor din blocul 2.0-CPU-RAM-CHECKS pentru unitatea AUTOTHROTTLE tip 10-62017-21 pentru avioanele Boeing 737-500 produs de firma Smith Industries Aerospace

```
214000      DISPLAY, MESSAGE,                                                $
    10      DISPLAY, MESSAGE, ------------------------------------------    $
    20      DISPLAY, MESSAGE, CPU CHECK                                      $
    30      DISPLAY, MESSAGE, 30 SECONDS AFTER GO AROUND                     $
    40      DISPLAY, MESSAGE, OUTPUT STATUS CHECK                            $
214050      DISPLAY, MESSAGE, ------------------------------------------    $
    60      DISPLAY, MESSAGE,                                                $
    70      DISPLAY, MESSAGE, ('MESS24PINS') $
C$
214100      IF, 'PRINT' EQ 1, THEN $
    10        PRINT,   MESSAGE,                                              $
    20        PRINT,   MESSAGE, ------------------------------------------  $
    30        PRINT,   MESSAGE, CPU CHECK                                    $
    40        PRINT,   MESSAGE, 30 SECONDS AFTER GO AROUND                   $
214150        PRINT,   MESSAGE, OUTPUT STATUS CHECK                          $
    60        PRINT,   MESSAGE, ------------------------------------------  $
    70        PRINT,   MESSAGE,                                              $
    80        PRINT,   MESSAGE, ('MESS24PINS') $
    90      END, IF $
C$
214200      FOR, 'INDEX1' = 1 THRU 24, THEN $
    10        CALCULATE, 'PRINT' = 'OPTION'(26) $
    20        CALCULATE, 'Z1' = 'Z1'('INDEX1') $
    30        CALCULATE, 'Z2' =  'B'('INDEX1') $
C$
214300        IF, ABS('Z2' - 'Z1') GT 3.0, THEN $
    10          FILL, 'FRESULT', C'FAILED' $
    20          CALCULATE, 'PRINT' = 1 $
    30        ELSE $
    40          FILL, 'FRESULT', C'PASSED' $
214350        END, IF $
C$
214400        DISPLAY, MESSAGE, ('MESSCPUPB3', 'PINS'('INDEX1'),
                                'Z1', 'Z2', 'FRESULT') $
    10        IF, 'PRINT' EQ 1, THEN $
    20          PRINT, MESSAGE, ('MESSCPUPB3', 'PINS'('INDEX1'),
                                 'Z1', 'Z2', 'FRESULT') $
    30        END, IF $
    40      END, FOR $
C$
C           ------------------------------------------------------------- $
C           I OUTPUT PATTERN CHECK COMPLETED                              I $
C           I NOW CHECK LR42 LR17 AND LR18                                I $
C           I LET B(1) = VOLTAGE ON LR42 = B(25)   AND PINS(1)=LR42       I $
C           I LET B(2) = VOLTAGE ON LR17 = B(26)   AND PINS(2)=LR17       I $
C           I LET B(3) = VOLTAGE ON LR18 = B(27)   AND PINS(3)=LR18       I $
C           I                                                             I $
C           ------------------------------------------------------------- $
C$
214500      CALCULATE, 'B'(1) = 'B'(25) $
    10      CALCULATE, 'B'(2) = 'B'(26) $
    20      CALCULATE, 'B'(3) = 'B'(27) $
    30      FILL, 'PINS',
                  (1) C'LR42',
                  (2) C'LR17',
                  (3) C'LR18' $
C$
C           ------------------------------------------------------------- $
C           I IF OUTPUT PATTERN PASSED (Z=177777) CHECK LR42 LR17   I $
C           I AND LR18 ALL AT 20V   DISPLAY AND PRINT RESULTS AND   I $
C           I IF PASS LEAVE BLOCK                                   I $
C           ------------------------------------------------------------- $
C$
215000      IF, 'Z' EQ 177777, THEN$
    10        CALCULATE, 'PRINT' = 'OPTION'(26) $
    20        FOR, 'INDEX' = 1 THRU 3, THEN $
    30          IF, 'B'('INDEX') LT 25, THEN $
    40            CALCULATE, 'RESLTS'(1) = -1 $
215050            CALCULATE, 'PRINT' = 1 $
    60          END, IF $
    70        END, FOR $
C$
215100      DISPLAY, MESSAGE,                                                $
    10      DISPLAY, MESSAGE, LR42, LR17 AND LR18                            $
    20      DISPLAY, MESSAGE, SHOULD ALL BE GREATER THAN 25.0V               $
    30      DISPLAY, MESSAGE, MEASURED VALUES ARE                            $
C$
215200      IF, 'PRINT' EQ 1, THEN $
    10        PRINT, MESSAGE,                                                $
    20        PRINT, MESSAGE, LR42, LR17 AND LR18                            $
    30        PRINT, MESSAGE, SHOULD ALL BE GREATER THAN 25.0V               $
    40        PRINT, MESSAGE, MEASURED VALUES ARE                            $
215250      END, IF $
C$
215300      FOR, 'INDEX' = 1 THRU 3, THEN $
    10        CALCULATE, 'PRINT' = 'OPTION'(26) $
    20        CALCULATE, 'Z1' = 'B'('INDEX') $
215400        IF, 'Z1' LT 25, THEN $
    10          CALCULATE, 'RESLTS'(1) = -1 $
    20          CALCULATE, 'PRINT' = 1 $
    30          FILL, 'FRESULT', C'PASSED' $
    40        ELSE $
215450          FILL, 'FRESULT', C'FAILED' $
    60        END, IF $
C$
215500        DISPLAY, MESSAGE, VOLTAGE ON ('PINS'('INDEX')) =     $
    10        DISPLAY, MESSAGE,  ('Z1' FORMAT +99.99)V ('FRESULT') $
    20        IF, 'PRINT' EQ 1, THEN $
    30          PRINT, MESSAGE, VOLTAGE ON ('PINS'('INDEX')) =     $
    40          PRINT, MESSAGE,  ('Z1' FORMAT +99.99)V ('FRESULT') $
215550        END, IF $
    60      END, FOR $
C$
C           ------------------------------------------------------------- $
C           I GIVE PASS OR FAIL MESSAGE AND LEAVE BLOCK                   I $
C           ------------------------------------------------------------- $
C$
215600      CALCULATE, 'J' = 1 $
    10      PERFORM, 'BLOCK-RES' $
    20      REMOVE, ALL $
    30      LEAVE, BLOCK $
    40    END, IF $
C$
C           ------------------------------------------------------------- $
C           I OUTPUT CODE HAS FAILED                                      I $
C           I CHECK VOLTAGES ON LR42 LR17 AND LR18                        I $
C           I IF ALL AT 0V CPU HAS FAILED TEST INDICATED BY Z             I $
C           I OTHERWISE TEST INDICATED BY Z WAS THE LAST TEST             I $
C           I TO PASS AND THE PROGRAMME HAS CORRUPTED DURING              I $
C           I THE NEXT ONE                                                I $
C           ------------------------------------------------------------- $
C$
216000      FOR, 'INDEX' = 1 THRU 3, THEN $
    10        IF, 'B'('INDEX') GT 3, THEN $
    20          CALCULATE, 'RESLTS'(1) = -1 $
    30        END, IF $
    40      END, FOR $
C$
216100      IF, 'RESLTS'(1) EQ -1, THEN $
    10        CALCULATE, 'Z1' = 'Z' + 40000 $
C$
C           ------------------------------------------------------------- $
C           I PROGRAMME HAS CORRUPTED                                     I $
C           ------------------------------------------------------------- $
C$
216200      DISPLAY, MESSAGE,                                                $
    10      DISPLAY, MESSAGE, ------------------------------------------    $
    20      DISPLAY, MESSAGE, CPU MICROPROGRAM TEST          FAILED          $
    30      DISPLAY, MESSAGE, LAST TEST TO PASS                              $
    40      DISPLAY, MESSAGE,            STEP ('Z' FORMAT +999999)           $
216250      DISPLAY, MESSAGE, PROGRAM CORRUPTED AT                           $
    60      DISPLAY, MESSAGE,          STEP ('Z1' FORMAT +999999)            $
```

ANEXA 1

Descrierea în limbaj ATLAS 616 a testelor din blocul 2.0-CPU-RAM-CHECKS pentru unitatea AUTOTHROTTLE tip 10-62017-21 pentru avioanele Boeing 737-500 produs de firma Smith Industries Aerospace

```
     70       DISPLAY, MESSAGE, ------------------------------------- $
 216300       PRINT,   MESSAGE,                                        $
     10       PRINT,   MESSAGE, ------------------------------------- $
     20       PRINT,   MESSAGE, CPU MICROPROGRAM TEST          FAILED  $
     30       PRINT,   MESSAGE, LAST TEST TO PASS                      $
     40       PRINT,   MESSAGE,       STEP ('Z' FORMAT +999999)        $
 216350       PRINT,   MESSAGE, PROGRAM CORRUPTED AT                   $
     60       PRINT,   MESSAGE,       STEP ('Z1' FORMAT +999999)       $
     70       PRINT,   MESSAGE, ------------------------------------- $
     80     ELSE $
     90       CALCULATE, 'RESLTS'(1) = -1 $
C$
C            ------------------------------------------------------------ $
C            I PROGRAMME HAS FAILED ON INDICATED TEST                  I $
C            ------------------------------------------------------------ $
C$
 216400       DISPLAY, MESSAGE,                                        $
     10       DISPLAY, MESSAGE, ------------------------------------- $
     20       DISPLAY, MESSAGE, CPU MICROPROGRAM TEST          FAILED  $
     30       DISPLAY, MESSAGE, PROGRAM FAILED AT                      $
     40       DISPLAY, MESSAGE,       STEP ('Z1' FORMAT +999999)       $
 216450       DISPLAY, MESSAGE, ------------------------------------- $
 216500       PRINT,   MESSAGE,                                        $
     10       PRINT,   MESSAGE, ------------------------------------- $
     20       PRINT,   MESSAGE, CPU MICROPROGRAM TEST          FAILED  $
     30       PRINT,   MESSAGE, PROGRAM FAILED AT                      $
     40       PRINT,   MESSAGE,       STEP ('Z1' FORMAT +999999)       $
 216550       PRINT,   MESSAGE, ------------------------------------- $
     60     END, IF $
C$
 216600     CALCULATE, 'J' = 1 $
     10     PERFORM, 'BLOCK-RES' $
     20     REMOVE, ALL $
 216998   END, BLOCK, 'CPU UNTEP' $
C$
C$
C ***************************************************************** $
C *       BLOCK 2.2 STORE RAM UPPER                                 * $
C ***************************************************************** $
C *                                                                 * $
C *       THIS BLOCK CARRIES OUT THE STORE RAM UPPER TEST AS        * $
C *       FOLLOWS                                                   * $
C *       THE UUT IS SET TO INITIAL CONDITIONS AND THE STORE RAM    * $
C *       UPPER DOWNLOAD PROGRAMME IS LOADED INTO THE UUT.          * $
C *       IF THE LOAD IS OK THEN TEST PROCEEDS AS FOLLOWS           * $
C *                                                                 * $
C *       THE FOLLOWING CODE (7070707 OCTAL) IS THEN SET UP ON      * $
C *       THE INPUT PINS                                            * $
C *                                                                 * $
C *       ------------------------------------------------------    * $
C *       I CODE 7070707                                        I   * $
C *       I                                                     I   * $
C *       I    INPUT    UUT    BTS              STATUS         I   * $
C *       I WORD BIT    PIN    SWITCH   LOGIC   PIN(V)   SWITCH I   * $
C *       I ---------   ----   ------   -----   ------   ------ I   * $
C *       I 1240  12    LL36   47         1      28.0     UP     I   * $
C *       I 1240  11    LR22   48         1      0/C      UP     I   * $
C *       I 1240  10    LL22   49         1      0/C      UP     I   * $
C *       I 1240   9    LL35   50         0      0/C      DOWN   I   * $
C *       I 1240   8    LL34   51         0      0/C      DOWN   I   * $
C *       I                                                     I   * $
C *       I 1300  15    UL12   52         0      0/C      DOWN   I   * $
C *       I 1300  14    UL3    53         1      28.0     UP     I   * $
C *       I 1300  13    UL2    54         1      28.0     UP     I   * $
C *       I 1300  12    LR37   55         1      0/C      UP     I   * $
C *       I 1300  11    LR30   56         0      0/C      DOWN   I   * $
C *       I 1300  10    LL30   57         0      0/C      DOWN   I   * $
C *       I 1300   9    UR20   58         0      0/C      DOWN   I   * $
C *       I 1300   8    LL2    59         1      28.0     UP     I   * $
C *       I 1300   7    UL33   60         1      28.0     UP     I   * $
C *       I 1300   6    UL32   61         1      28.0     UP     I   * $
C *       I 1300   5    LR48   62         0      0/C      DOWN   I   * $
C *       I 1300   4    LL48   63         0      0/C      DOWN   I   * $
C *       I                                                     I   * $
C *       I 1240   7    UL36   64         0      0/C      DOWN   I   * $
C *       I 1240   6    UL35   65         1      28.0     UP     I   * $
C *       I 1240   5    UL34   66         1      28.0     UP     I   * $
C *       I 1240   4    UL25   67         1      28.0     UP     I   * $
C *       I                                                     I   * $
C *       I BTS SWITCHES REFER TO THE EQUIVALENT BENCH TEST      I   * $
C *       I SET SWITCH AS USED IN THE BENCH TEST ATP AND ARE     I   * $
C *       I INCORPORATED FOR CROSS REFERENCING PURPOSES          I   * $
C *       I BETWEEN THE ATLAS AND THE ATP                        I   * $
C *       ------------------------------------------------------    * $
C *                                                                 * $
C *                                                                 * $
C *       A GO-AROUND IS THEN PERFORMED AND THE UUT OUTPUTS         * $
C *       LR6 LR7 AND LR15 ARE CHECKED                              * $
C *                                                                 * $
C *       IF THE TEST HAS PASSED THEN THEY WILL ALL BE 0V           * $
C *       IF A FAILURE OCCURS THEN AN APPARENT RAM MALFUNCTION      * $
C *       HAS OCCURED AS FOLLOWS                                    * $
C *                                                                 * $
C *       LR6  FAILED = RAM DECODE BOARD LOCATION C3 MALFUNCTION    * $
C *       LR7  FAILED = RAM DECODE BOARD LOCATION C2 MALFUNCTION    * $
C *       LR15 FAILED = RAM DECODE BOARD LOCATION C1 MALFUNCTION    * $
C *                                                                 * $
C ***************************************************************** $
C$
 220000   BEGIN, BLOCK, '2.2-STORE-RAM-UPPER' $
C$
C            ------------------------------------------------------------ $
C            I EXIT BLOCK IF TEST NOT SELECTED                         I $
C            ------------------------------------------------------------ $
C$
 220100     IF, 'OPTION'(2) NE 1, THEN $
     20       LEAVE, BLOCK $
     30     END, IF $
 220200     CALCULATE, 'J' = 2 $
     10     CALCULATE, 'RESLTS'('J') = 1 $
     20     FILL, 'THAM', C'STORE RAM UPPER          ' $
C$
C            ------------------------------------------------------------ $
C            I INITIALISE UUT.   EXIT BLOCK IF FAILURE OCCURS          I $
C            ------------------------------------------------------------ $
C$
 220300     PERFORM, 'INITCHECK', C'STORE RAM UPPER          ' $
     10     IF, 'RESLTS'('J') LT 0, THEN $
     20       LEAVE, BLOCK $
     30     END, IF $
C$
C            ------------------------------------------------------------ $
C            I DOWLOAD PROGRAM.  EXIT BLOCK IF FAILURE OCCURS          I $
C            ------------------------------------------------------------ $
C$
 220400     PERFORM, 'LOADCHECK', C'STORE RAM UPPER          ' $
     10     IF, 'RESLTS'('J') LT 0, THEN $
     20       LEAVE, BLOCK $
     30     END, IF $
C$
C            ------------------------------------------------------------ $
C            I SET UP CODE 7070707 THEN APPLY GO AROUND                I $
C            ------------------------------------------------------------ $
C$
 221000     PERFORM, 'SET-7070707' $
     10     PERFORM, 'GO-AROUND' $
C$
C            ------------------------------------------------------------ $
C            I AFTER SHORT DELAY VERIFY LR6 LR7 AND LR15 ALL AT 0V     I $
C            ------------------------------------------------------------ $
C$
 221100     DELAY, 2 SEC $
     10     PERFORM, 'RAM-CHECK' $
```

ANEXA 1

Descrierea în limbaj ATLAS 616 a testelor din blocul 2.0-CPU-RAM-CHECKS pentru unitatea AUTOTHROTTLE tip 10-62017-21 pentru avioanele Boeing 737-500 produs de firma Smith Industries Aerospace

```
     20     PERFORM, 'SRAM-PRINT' $
 221200     IF, 'RESLTS'(2) EQ 1, THEN $
     10       CALCULATE, 'J' = 2 $
     20       PERFORM, 'BLOCK-RES' $
     30       REMOVE, ALL $
     40       LEAVE, BLOCK $
 221250     END, IF $
C$
C          ------------------------------------------------------------ $
C          I TEST FAILED GIVE DIAGNOSTICS                             I $
C          ------------------------------------------------------------ $
C$
 222000     DISPLAY, MESSAGE,                                            $
     10     DISPLAY, MESSAGE, ('TNAM') TEST FAILED                       $
     20     DISPLAY, MESSAGE, RAM DECODE BOARD SUSPECT AT FOLLOWING      $
     30     DISPLAY, MESSAGE, LOCATIONS                                  $
 222050     PRINT,   MESSAGE,                                            $
     60     PRINT,   MESSAGE, ('TNAM') TEST FAILED                       $
     70     PRINT,   MESSAGE, RAM DECODE BOARD SUSPECT AT FOLLOWING      $
     80     PRINT,   MESSAGE, LOCATIONS                                  $
C$
 222100     IF, 'B'(1) GT 2.0, THEN $
     10       DISPLAY, MESSAGE, LOCATION C3 $
     20       PRINT,   MESSAGE, LOCATION C3 $
     30     END, IF $
C$
 222200     IF, 'B'(2) GT 2.0, THEN $
     10       DISPLAY, MESSAGE, LOCATION C2 $
     20       PRINT,   MESSAGE, LOCATION C2 $
     30     END, IF $
C$
 222300     IF, 'B'(3) GT 2.0, THEN $
     10       DISPLAY, MESSAGE, LOCATION C1 $
     20       PRINT,   MESSAGE, LOCATION C1 $
     30     END, IF $
C$
 229000     CALCULATE, 'J' = 2 $
     10     PERFORM, 'BLOCK-RES' $
     20     REMOVE, ALL $
 229990     END, BLOCK, '2.2-STORE RAM-UPPER' $
C$
C$
C ********************************************************************** $
C *        BLOCK 2.3 STORE RAM LOWER                                   * $
C ********************************************************************** $
C *        THIS BLOCK CARRIES OUT THE STORE RAM LOWER TEST AS          * $
C *        FOLLOWS                                                     * $
C *        THE UUT IS SET TO INITIAL CONDITIONS AND THE STORE RAM      * $
C *        LOWER DOWNLOAD PROGRAMME IS LOADED INTO THE UUT.            * $
C *        IF THE LOAD IS OK THEN TEST PROCEEDS AS FOLLOWS             * $
C *                                                                    * $
C *        THE FOLLOWING CODE (7070707 OCTAL) IS THEN SET UP ON        * $
C *        THE INPUT PINS                                              * $
C *                                                                    * $
C *        ------------------------------------------------------      * $
C *        I CODE 7070707                                        I     * $
C *        I                                                     I     * $
C *        I    INPUT    UUT    BTS            STATUS           I     * $
C *        I WORD BIT    PIN    SWITCH  LOGIC  PIN(V)  SWITCH   I     * $
C *        I ------- ---  ---    ------  -----  ------  ------   I     * $
C *        I 1240  12    LL36   47        1     28.0    UP       I     * $
C *        I 1240  11    LR22   48        1     O/C     UP       I     * $
C *        I 1240  10    LL22   49        1     O/C     UP       I     * $
C *        I 1240   9    LL35   50        0     O/C     DOWN     I     * $
C *        I 1240   8    LL34   51        0     O/C     DOWN     I     * $
C *        I                                                     I     * $
C *        I 1300  15    UL12   52        0     O/C     DOWN     I     * $
C *        I 1300  14    UL3    53        1     28.0    UP       I     * $
C *        I 1300  13    UL2    54        1     28.0    UP       I     * $
C *        I 1300  12    LR37   55        1     O/C     UP       I     * $
C *        I 1300  11    LR30   56        0     O/C     DOWN     I     * $
C *        I 1300  10    LL30   57        0     O/C     DOWN     I     * $
C *        I 1300   9    UR28   58        0     O/C     DOWN     I     * $
C *        I 1300   8    LL2    59        1     28.0    UP       I     * $
C *        I 1300   7    UL30   60        1     28.0    UP       I     * $
C *        I 1300   6    UL32   61        1     28.0    UP       I     * $
C *        I 1300   5    LR48   62        0     O/C     DOWN     I     * $
C *        I 1300   4    LL48   63        0     O/C     DOWN     I     * $
C *        I                                                     I     * $
C *        I 1240   7    UL36   64        0     O/C     DOWN     I     * $
C *        I 1240   6    UL35   65        1     28.0    UP       I     * $
C *        I 1240   5    UL34   66        1     28.0    UP       I     * $
C *        I 1240   4    UL25   67        1     28.0    UP       I     * $
C *        I                                                     I     * $
C *        I BTS SWITCHES REFER TO THE EQUIVALENT BENCH TEST     I     * $
C *        I SET SWITCH AS USED IN THE BENCH TEST ATP AND ARE    I     * $
C *        I INCORPORATED FOR CROSS REFERENCING PURPOSES         I     * $
C *        I BETWEEN THE ATLAS AND THE ATP                       I     * $
C *        ------------------------------------------------------      * $
C *                                                                    * $
C *                                                                    * $
C *        A GO-AROUND IS THEN PERFORMED AND THE UUT OUTPUTS           * $
C *        LR6 LR7 AND LR15 ARE CHECKED                                * $
C *                                                                    * $
C *        IF THE TEST HAS PASSED THEN THEY WILL ALL BE 0V             * $
C *        IF A FAILURE OCCURS THEN AN APPARENT RAM MALFUNCTION        * $
C *        HAS OCCURED AS FOLLOWS                                      * $
C *                                                                    * $
C *        LR6  FAILED = RAM DECODE BOARD LOCATION C3 MALFUNCTION      * $
C *        LR7  FAILED = RAM DECODE BOARD LOCATION C2 MALFUNCTION      * $
C *        LR15 FAILED = RAM DECODE BOARD LOCATION C1 MALFUNCTION      * $
C *                                                                    * $
C ********************************************************************** $
C$
 230000     BEGIN, BLOCK, '2.3-STORE RAM-LOWER' $
C$
C            ------------------------------------------------------------ $
C            I EXIT BLOCK IF TEST NOT SELECTED                          I $
C            ------------------------------------------------------------ $
C$
 230100     IF, 'OPTION'(3) NE 1, THEN $
     20       LEAVE, BLOCK $
     30     END, IF $
 230200     CALCULATE, 'J' = 3 $
     10     CALCULATE, 'RESLTS'('J') = 1 $
     20     FILL, 'TNAM', C'STORE RAM LOWER      ' $
C$
C            ------------------------------------------------------------ $
C            I INITIALISE UUT.   EXIT BLOCK IF FAILURE OCCURS           I $
C            ------------------------------------------------------------ $
C$
 230300     PERFORM, 'INITCHECK', C'STORE RAM LOWER      ' $
     10     IF, 'RESLTS'('J') LT 0, THEN $
     20       LEAVE, BLOCK $
     30     END, IF $
C$
C            ------------------------------------------------------------ $
C            I DOWLOAD PROGRAM.  EXIT BLOCK IF FAILURE OCCURS           I $
C            ------------------------------------------------------------ $
C$
 230400     PERFORM, 'LOADCHECK', C'STORE RAM LOWER      ' $
     10     IF, 'RESLTS'('J') LT 0, THEN $
     20       LEAVE, BLOCK $
     30     END, IF $
C$
C            ------------------------------------------------------------ $
C            I SET UP CODE 7070707 THEN APPLY GO-AROUND                 I $
C            ------------------------------------------------------------ $
C$
 231000     PERFORM, 'SET-7070707' $
     10     PERFORM, 'GO-AROUND' $
C$
```

ANEXA 1

Descrierea în limbaj ATLAS 616 a testelor din blocul 2.0-CPU-RAM-CHECKS pentru unitatea AUTOTHROTTLE tip 10-62017-21 pentru avioanele Boeing 737-500 produs de firma Smith Industries Aerospace

```
C          ------------------------------------------------------------ $
C          I AFTER SHORT DELAY VERIFY LR6 LR7 AND LR15 ALL AT 0V      I $
C          ------------------------------------------------------------ $
C$
 231100      DELAY, 2 SEC $
     10      PERFORM, 'RAM-CHECK' $
     20      PERFORM, 'SRAM-PRINT' $
 231200      IF, 'RESLTS'(3) EQ 1, THEN $
     10        CALCULATE, 'J' = 3 $
     20        PERFORM, 'BLOCK-RES' $
     30        REMOVE, ALL $
     40        LEAVE, BLOCK $
 231290      END, IF $
C$
C          ------------------------------------------------------------ $
C          I TEST FAILED GIVE DIAGNOSTICS                             I $
C          ------------------------------------------------------------ $
C$
 232000      DISPLAY, MESSAGE,                                          $
     10      DISPLAY, MESSAGE, ('TNAM') TEST FAILED                     $
     20      DISPLAY, MESSAGE, RAM DECODE BOARD SUSPECT AT FOLLOWING    $
     30      DISPLAY, MESSAGE, LOCATIONS                                $
 232050      PRINT,   MESSAGE,                                          $
     60      PRINT,   MESSAGE, ('TNAM') TEST FAILED                     $
     70      PRINT,   MESSAGE, RAM DECODE BOARD SUSPECT AT FOLLOWING    $
     80      PRINT,   MESSAGE, LOCATIONS                                $
C$
 232100      IF, 'B'(1) GT 2.0, THEN $
     10        DISPLAY, MESSAGE, LOCATION C3 $
     20        PRINT,   MESSAGE, LOCATION C3 $
     30      END, IF $
C$
 232200      IF, 'B'(2) GT 2.0, THEN $
     10        DISPLAY, MESSAGE, LOCATION C2 $
     20        PRINT,   MESSAGE, LOCATION C2 $
     30      END, IF $
C$
 232300      IF, 'B'(3) GT 2.0, THEN $
     10        DISPLAY, MESSAGE, LOCATION C1 $
     20        PRINT,   MESSAGE, LOCATION C1 $
     30      END, IF $
C$
 239000      CALCULATE, 'J' = 3 $
     10      PERFORM, 'BLOCK-RES' $
     20      REMOVE, ALL $
 239090      END, BLOCK, '2.3 STORE-RAM-LOWER' $
C$
C$
C ********************************************************************* $
C *       BLOCK 2.4 STORE PROGRAMME CONTENTS TEST                     * $
C ********************************************************************* $
C *       THIS BLOCK CARRIES OUT THE STORE PROGRAMME CONTENTS TEST     * $
C *       AS FOLLOWS                                                  * $
C *       THE UUT IS SET TO INITIAL CONDITIONS AND THE STORE          * $
C *       PROGRAMME CONTENTS DOWNLOAD IS LOADED INTO THE UUT.         * $
C *       IF THE LOAD IS OK THEN TEST PROCEEDS AS FOLLOWS             * $
C *                                                                   * $
C *       THERE ARE TWO CRC CHECKS TO BE PERFORMED, ONE FOR THE       * $
C *       FLYING PROGRAM (BANK 0) AND ONE FOR THE GROUND BITE         * $
C *       PROGRAMME (BANK 1).                                         * $
C *                                                                   * $
C *       THE APPROPRIATE BANK IS SELECTED BY EITHER SETTING THE      * $
C *       GEAR DOWN INPUT (LL4) TO THE APPROPRIATE LEVEL              * $
C *                                                                   * $
C *        BANK 0 = FLYING PROGRAM - GEAR UP   = LL4 OPEN CIRCUIT     * $
C *        BANK 1 = GROUND BITE    - GEAR DOWN = LL4 GROUNDED         * $
C *                                                                   * $
C *       THE LENGTH OF THE PROGRAMME TO BE CHECKED IS THEN           * $
C *       ENTERED INTO THE UUT ON WORD 1300 BITS 15 THRU 9            * $
C *       AND WORD 1240 BITS 7 THRU 4 AS FOLLOWS                      * $
C *                                                                   * $
```

```
C *  ------------------------------------------------------------------
C *  I                                                                 I
C *  I   INPUT    UUT    BTS     BIT=0   BIT=1   SWITCH  STATE        I
C *  I WORD BIT   PIN    SWITCH  PIN(V)  PIN(V)  BIT=0   BIT=1        I
C *  I ---------  ----   ------  ------  ------  ------  -----        I
C *  I 1300  15  UL12   52      O/C     28.0    DOWN    UP           I
C *  I 1300  14  UL3    53      O/C     28.0    DOWN    UP           I
C *  I 1300  13  UL2    54      O/C     28.0    DOWN    UP           I
C *  I 1300  12  LR37   55      00.0    O/C     DOWN    UP           I
C *  I 1300  11  LR30   56      O/C     28.0    DOWN    UP           I
C *  I 1300  10  LL30   57      O/C     28.0    DOWN    UP           I
C *  I 1300   9  UR28   58      O/C     28.0    DOWN    UP           I
C *  I 1300   8  LL2    59      O/C     28.0    DOWN    UP           I
C *  I 1300   7  UL53   60      O/C     28.0    DOWN    UP           I
C *  I 1300   6  UL32   61      O/C     28.0    DOWN    UP           I
C *  I 1300   5  LR48   62      O/C     28.0    DOWN    UP           I
C *  I 1300   4  LL48   63      O/C     28.0    DOWN    UP           I
C *  I                                                             I
C *  I 1240   7  UL36   64      O/C     28.0    DOWN    UP           I
C *  I 1240   6  UL35   65      O/C     28.0    DOWN    UP           I
C *  I 1240   5  UL34   66      O/C     28.0    DOWN    UP           I
C *  I 1240   4  UL25   67      O/C     28.0    DOWN    UP           I
C *  I                                                             I
C *  I BTS SWITCHES REFER TO THE EQUIVALENT BENCH TEST             I
C *  I SET SWITCH AS USED IN THE BENCH TEST ATP AND ARE            I
C *  I INCORPORATED FOR CROSS REFERENCING PURPOSES                 I
C *  I BETWEEN THE ATLAS AND THE ATP                               I
C *  ------------------------------------------------------------------
C *
C *  TREATING THE ABOVE 16 DISCRETES WITH UL12 BEING THE
C *  MSB AND UL25 THE LSB THE OCTAL LENGTHS OF THE
C *  PROGRAMMES ARE AS FOLLOWS
C *
C *    7555UE1-1     BANK 0 = 040000     BANK 1 = 040000
C *    7555UE2-2     BANK 0 = 040000     BANK 1 = 040000
C *    7555UE2-3     BANK 0 = 040000     BANK 1 = 040000
C *    7565UE1-1     BANK 0 = 040000     BANK 1 = 040000
C *    7565UE2-1     BANK 0 = 040000     BANK 1 = 040000
C *    7565UE3-2     BANK 0   040000     BANK 1 = 040000
C *    7565UE3-3     BANK 0 = 040000     BANK 1   040000
C *    7565UE4-2     BANK 0   040000     BANK 1 = 040000
C *    7565UE4-3     BANK 0 = 040000     BANK 1 = 040000
C *
C *  A GO-AROUND IS THEN PERFORMED WHICH CAUSES THE UUT TO
C *  START ITS CRC CHECK. UUT OUTPUT PIN LR10 IS THEN
C *  MONITORED AND WHEN THE CRC CHECK HAS BEEN COMPLETED
C *  IT CHANGES FROM -10V TO +10V WHICH SHOULD HAPPEN
C *  WITHIN 20SECONDS OF THE GO-AROUND (THIS IS EQUIVALENT
C *  TO WAITING FOR LAMP 19 TO GO FROM RED TO GREEN ON THE
C *  BENCH TEST SET9
C *
C *  ON COMPLETION OF THE CRC CHECK THE CRC CODE WILL APPEAR
C *  ON UUT OUTPUT WORDS 220 AND 240 AS FOLLOWS
C *
C *  ------------------------------------------------------------------
C *  I                                                                 I
C *  I  OUTPUT    UUT    BTS     BIT=0   BIT=1    LAMP STATE         I
C *  I WORD BIT   PIN    LAMP    PIN(V)  PIN(V)  BIT=0   BIT=1        I
C *  I ---------  ----   ------  ------  ------  ------  -----        I
C *  I 220   11  LR45   24      28.0    00.0    OFF     ON           I
C *  I 220   10  LR46   25      28.0    00.0    OFF     ON           I
C *  I 220    9  LR23   26      28.0    00.0    OFF     ON           I
C *  I 220    8  LL23   27      28.0    00.0    OFF     ON           I
C *  I 240   15  LR6    28      28.0    00.0    OFF     ON           I
C *  I 240   14  LR7    29      28.0    00.0    OFF     ON           I
C *  I 240   13  LR15   30      28.0    00.0    OFF     ON           I
C *  I 240   12  LR44   31      28.0    00.0    OFF     ON           I
C *  I 240   11  LR36   32      28.0    00.0    OFF     ON           I
C *  I 240   10  LR43   33      28.0    00.0    OFF     ON           I
C *  I 240    9  LR26   34      28.0    00.0    OFF     ON           I
C *  I 240    8  LR14   35      28.0    00.0    OFF     ON           I
```

ANEXA 1

Descrierea în limbaj ATLAS 616 a testelor din blocul 2.0-CPU-RAM-CHECKS pentru unitatea AUTOTHROTTLE tip 10-62017-21 pentru avioanele Boeing 737-500 produs de firma Smith Industries Aerospace

```
C *        I 240     7  LR33  36         28.0   00.0   OFF     ON     I    * $
C *        I 240     6  LR24  37         28.0   00.0   OFF     ON     I    * $
C *        I 240     5  LR35  38         28.0   00.0   OFF     ON     I    * $
C *        I 240     4  LR34  39         28.0   00.0   OFF     ON     I    * $
C *        I 220     7  LR25  40         28.0   00.0   OFF     ON     I    * $
C *        I 220     6  LR22  41         28.0   00.0   OFF     ON     I    * $
C *        I 220     5  UR12  42         28.0   00.0   OFF     ON     I    * $
C *        I 220     4  UR16  43         28.0   00.0   OFF     ON     I    * $
C *        I                                                          I    * $
C *        I BT6 LAMPS     REFER TO THE EQUIVALENT BENCH TEST         I    * $
C *        I SET LAMP    AS USED IN THE BENCH TEST ATP AND ARE        I    * $
C *        I INCORPORATED FOR CROSS REFERENCING PURPOSES              I    * $
C *        I BETWEEN THE ATLAS AND THE ATP                            I    * $
C *        ----------------------------------------------------------      * $
C *                                                                        * $
C *        THE BINARY BIT PATTERN EXPECTED ON THE OUTPUTS FOR EACH         * $
C *        CODE AND BANK ARE AS FOLLOWS (WHERE LR45 IS THE MSB             * $
C *        AND UR16 IS THE LSB)                                            * $
C *                                                                        * $
C *         CODE        MSB     BANK 0      LSB  MSB     BANK 1      LSB   * $
C *         --------    --------------------------------------------       * $
C *         755SUE1 1   01100000100011100110  01100000001011001100         * $
C *         755SUE2 2   10010110011100101010  00110010000110101011         * $
C *         755SUE2 3   10010110011100101010  01010111111101000000         * $
C *         756SUE1 1   01100000100011100110  01100000001011001100         * $
C *         756SUE2 1   01100000100011100110  01100000001011001100         * $
C *         756SUE3-2   10010110011100101010  00110010000110101011         * $
C *         756SUE3 3   10010110011100101010  01010111111101000000         * $
C *         756SUE4 2   10010110011100101010  00110010000110101011         * $
C *         756SUE4 3   10010110011100101010  01010111111101000000         * $
C *                                                                        * $
C ************************************************************************** $
C$
 240000    BEGIN, BLOCK, '2.4-STORE-PROGRAMME-CONTENTS' $
C$
C            -----------------------------------------------------------  $
C            I EXIT BLOCK IF TEST NOT SELECTED                          I $
C            -----------------------------------------------------------  $
C$
 240100      IF, 'OPTION'(4) NE 1, THEN $
     20        LEAVE, BLOCK $
     30      END, IF $
 240200      CALCULATE, 'J' = 4 $
     10      CALCULATE, 'RESLTS'('J') = 1 $
     20      FILL, 'TNAM', C'STORE PROGRAMME CONTENTS ' $
C$
C            -----------------------------------------------------------  $
C            I INITIALISE UUT.    EXIT BLOCK IF FAILURE OCCURS          I $
C            -----------------------------------------------------------  $
C$
 240300      PERFORM, 'INITCHECK', C'STORE PROGRAMME CONTENTS ' $
     10      IF, 'RESLTS'('J') LT 0, THEN $
     20        LEAVE, BLOCK $
     30      END, IF $
C$
C            -----------------------------------------------------------  $
C            I DOWLOAD PROGRAM.  EXIT BLOCK IF FAILURE OCCURS           I $
C            -----------------------------------------------------------  $
C$
 240400      PERFORM, 'LOADCHECK', C'STORE PROGRAMME CONTENTS ' $
     10      IF, 'RESLTS'('J') LT 0, THEN $
     20        LEAVE, BLOCK $
     30      END, IF $
C$
C            -----------------------------------------------------------  $
C            I 'GB-BANK0' CONTAINS EXPECTED CONTENTS FOR BANK 0         I $
C            I 'GB-BANK1' CONTAINS EXPECTED CONTENTS FOR BANK 1         I $
C            I BOTH WORDS ARE 20 BIT WIDE DIGITAL WORDS                 I $
C            I WITH LEFT MOST BIT BEING THE MSB.                        I $
C            I SET 'C' (1-20) WITH EQUIVALENT OF BANK 0 WORD            I $
C            I SET 'C1'(1-20) WITH EQUIVALENT OF BANK 1 WORD            I $
C            I WHERE 'C'(1)  AND 'C1'(1)  ARE THE MSB OF THE WORDS      I $
C            I   AND 'C'(20) AND 'C1'(20) ARE THE LSB OF THE WORDS      I $
C            I WHERE THE VALUES OF 'C' AND 'C1' ARE SET AS FOLLOWS      I $
C            I   = 28 IF ASSOCIATED DIGITAL BIT IS A '0'                I $
C            I   =  0 IF ASSOCIATED DIGITAL BIT IS A '1'                I $
C            -----------------------------------------------------------  $
C$
 241000      CALCULATE, 'INDEX7' = 0 $
 241100      FOR, 'INDEX8' = 20 THRU 1 BY -1, THEN $
     10        CALCULATE, 'INDEX7' = 'INDEX7' + 1 $
 241200        IF, EXTRACT('GB-BANK0', 'INDEX8', 1) EQ B'1', THEN $
     10          CALCULATE, 'C'('INDEX7') = 0 $
     20        ELSE $
     30          CALCULATE, 'C'('INDEX7') = 28 $
     40        END, IF $
 241300        IF, EXTRACT('GB-BANK1', 'INDEX8', 1) EQ B'1', THEN $
     10          CALCULATE, 'C1'('INDEX7') = 0 $
     20        ELSE $
     30          CALCULATE, 'C1'('INDEX7') = 28 $
     40        END, IF $
 241350      END, FOR $
C$
C$
C            -----------------------------------------------------------  $
C            I LENGTHS OF ALL CRC CHECKS ARE THE SAME (I.E. 040000)     I $
C            I SO SET THE INPUTS UP TO THIS.                            I $
C            I N.B. INPUTS HAVE BEEN INITIALLY SET TO 000000 BY         I $
C            I POWER UP ROUTINE                                         I $
C            -----------------------------------------------------------  $
C$
 245000      CONNECT, 'DC-28V', CNX HI UL3  LO UR1 $
C$
C ----------------------------------------------------------------------  $
C I         NOW CHECK OUTPUTS FOR BOTH BANKS                            I $
C ----------------------------------------------------------------------  $
C$
 246000      FOR, 'BANK' = 0 THRU 1, THEN $
 246100        IF, 'BANK' EQ 1, THEN $
C$
C                -------------------------------------------------------  $
C                I BANK 1 SELECTED (GROUND BITE TEST)                   I $
C                I GROUND GEAR DOWN LL4 TO EARTH( ON GROUND)            I $
C                I FILL 'C' WITH EXPECTED VALUES CONTAINED IN 'C1'      I $
C                -------------------------------------------------------  $
C$
     10          CONNECT, SHORT, CNX HI LL4  LO EARTH $
     20          FOR, 'INDEX' = 1 THRU 20, THEN $
     30            CALCULATE, 'C'('INDEX') = 'C1'('INDEX') $
     40          END, FOR $
 246150        END, IF $
     60        FILL,        'PINRES',
                    (1)     C'PASS ',
                    (2)     C'PASS ',
                    (3)     C'PASS ',
                    (4)     C'PASS ',
                    (5)     C'PASS ',
                    (6)     C'PASS ',
                    (7)     C'PASS ',
                    (8)     C'PASS ',
                    (9)     C'PASS ',
                   (10)     C'PASS ',
                   (11)     C'PASS ',
                   (12)     C'PASS ',
                   (13)     C'PASS ',
                   (14)     C'PASS ',
                   (15)     C'PASS ',
                   (16)     C'PASS ',
                   (17)     C'PASS ',
                   (18)     C'PASS ',
                   (19)     C'PASS ',
                   (20)     C'PASS ' $
```

ANEXA 1

Descrierea în limbaj ATLAS 616 a testelor din blocul 2.0-CPU-RAM-CHECKS pentru unitatea AUTOTHROTTLE tip 10-62017-21 pentru avioanele Boeing 737-500 produs de firma Smith Industries Aerospace


```
C          ------------------------------------------------------------ $
C          I PERFORM GO-AROUND AND VERIFY THAT PIN LR10 STARTS  I $
C          ' AT LESS THAN -5V(CRC CHECK NOT YET COMPLETE        I $
C          ------------------------------------------------------------ $
CS
 246200        PERFORM, 'GO-AROUND' $
    10         MEASURE, (VOLTAGE INTO 'B'(1)), DC SIGNAL USING 'DVM1',
                 VOLTAGE RANGE -50.0V TO 50.0V,
                 CNX HI LR10 LO UR7 $
 246300        IF, 'B'(1) GE -5.0, THEN $
    10           CALCULATE, 'RESLTS'(4) = -1 $
    30           IF, 'OPTION'(26) NE 1, THEN $
    40             PERFORM, 'PRINT-TITLE' $
 246350          END, IF $
 246400          DISPLAY, MESSAGE,                                    $
    10           DISPLAY, MESSAGE, STORE PROGRAMME CONTENTS TEST      $
    20           DISPLAY, MESSAGE, BANK ('BANK' FORMAT +9) SELECTED   $
    30           DISPLAY, MESSAGE,                                    $
    40           DISPLAY, MESSAGE, LR10 CHECK              FAILED     $
 246450          DISPLAY, MESSAGE, LR10 GT -5.0V IMMEDIATELY          $
    60           DISPLAY, MESSAGE, AFTER GO-AROUND                    $
 246500          PRINT,   MESSAGE,                                    $
    10           PRINT,   MESSAGE, STORE PROGRAMME CONTENTS TEST      $
    20           PRINT,   MESSAGE, BANK ('BANK' FORMAT +9) SELECTED   $
    30           PRINT,   MESSAGE,                                    $
    40           PRINT,   MESSAGE, LR1  CHECK              FAILED     $
 246550          PRINT,   MESSAGE,  R10 GT -5.0V IMMEDIATELY          $
    60           PRINT,   MESSAGE, AFTER GO-AROUND                    $
 246600          PERFORM, 'FAILSTOP' $
    10           REMOVE, ALL $
    20           LEAVE, BLOCK $
    30           END, IF $
CS
C            ------------------------------------------------------------ $
C            I VERIFY CRC CHECK COMPLETES WITHIN 20 SECONDS       I $
C            I LR10 GOES GREATER THAN +5V                         I $
C            ------------------------------------------------------------ $
CS
 247000        WAIT FOR, (VOLTAGE), DC SIGNAL USING 'DVM1',
                 GT 5.0V,
                 POS-SLOPE,
                 MAX-TIME 209EC,
                 CNX HI LR10 LO UR7 $
 247100          IF, MAX-TIME, THEN $
    10             CALCULATE, 'RESLTS'(4) = -1 $
    20             IF, 'OPTION'(26) NE 1, THEN $
    30               PERFORM, 'PRINT-TITLE' $
    40             END, IF $
CS
 247200          DISPLAY, MESSAGE,                                    $
    10           DISPLAY, MESSAGE, STORE PROGRAMME CONTENTS TEST      $
    20           DISPLAY, MESSAGE, BANK ('BANK' FORMAT +9)  SELECTED  $
    30           DISPLAY, MESSAGE,                                    $
    40           DISPLAY, MESSAGE, LR10 CHECK              FAILED     $
 247250          DISPLAY, MESSAGE, 20 SECONDS AFTER GO-AROUND         $
    60           DISPLAY, MESSAGE, LR1  STILL NOT ABOVE +5.0VDC       $
    70           DISPLAY, MESSAGE, I.E. CRC CHECK STILL NOT COMPLETED $
 247300          PRINT,   MESSAGE,                                    $
    10           PRINT,   MESSAGE, STORE PROGRAMME CONTENTS TEST      $
    20           PRINT,   MESSAGE, BANK ('BANK' FORMAT +9)  SELECTED  $
    30           PRINT,   MESSAGE,                                    $
    40           PRINT,   MESSAGE, LR10 CHECK              FAILED     $
 247350          PRINT,   MESSAGE, 20 SECONDS AFTER GO-AROUND         $
    60           PRINT,   MESSAGE, LR1  STILL NOT ABOVE +5.0VDC       $
    70           PRINT,   MESSAGE, I.E. CRC CHECK STILL NOT COMPLETED $
 247400          CALCULATE, 'J' = 4 $
    10           PERFORM, 'BLOCK-RES' .
    20           LEAVE, BLOCK $
    30           END, IF $
CS
```

```
C            I MEASURE OUTPUT VOLTAGES INTO LIST  $                  I $
C            ------------------------------------------------------------ $
C9
 248000        PERFORM, 'READ-DVM', TABLE 3 VAR 20 TIMES,
                 LR45,     UR7,     1,
                 LR46,     UR7,     2,
                 LR23,     UR7,     3,
                 LL23,     UR7,     4,
                 LR6,      UR7,     5,
                 LR7,      UR7,     6,
                 LR15,     UR7,     7,
                 LR44,     UR7,     8,
                 LR36,     UR7,     9,
                 LR43,     UR7,     10,
                 LR26,     UR7,     11,
                 LR14,     UR7,     12,
                 LR33,     UR7,     13,
                 LR24,     UR7,     14,
                 LR35,     UR7,     15,
                 LR34,     UR7,     16,
                 LR25,     UR7,     17,
                 LR32,     UR7,     18,
                 UR12,     UR7,     19,
                 UR16,     UR7,     20 $
                 ------    ------   ---- $
C                HI PIN    LO PIN   INDEX $
C
C9
C            ------------------------------------------------------------ $
C            I COMPARE MEASURED VALUES WITH EXPECTED VALUES       I $
C            I DISPLAY AND (IF REQUIRED) PRINT RESULTS            I $
C            ------------------------------------------------------------ $
CS
 248100          CALCULATE, 'PF' = 0 $
    10           PERFORM, 'CHECK', 20 $
    20           CALCULATE, 'PRINT' = 'OPTION'(26) $
 248150          IF, 'PF' GT 0.5, THEN $
    60             CALCULATE, 'PRINT' = 1 $
    70           END, IF $
 248200          DISPLAY, MESSAGE,                                    $
    10           DISPLAY, MESSAGE, STORE PROGRAMME CONTENTS CHECK     $
    20           DISPLAY, MESSAGE, BANK ('BANK' FORMAT +9) SELECTED   $
    30           DISPLAY, MESSAGE,                                    $
    40           DISPLAY, MESSAGE, ('MESC1A',
                   'C'(1),'C'(2),'C'(3),'C'(4),'C'(5), 'C'(6), 'C'(7),
                   'C'(8), 'C'(9), 'C'(10),
                   'B'(1),'B'(2),'B'(3),'B'(4),'B'(5),'B'(6),'B'(7),
                   'B'(8),'B'(9), 'B'(10),
                   'PINRES'(1), 'PINRES'(2), 'PINRES'(3), 'PINRES'(4),
                   'PINRES'(5), 'PINRES'(6), 'PINRES'(7), 'PINRES'(8),
                   'PINRES'(9), 'PINRES'(10)) $
 248250          DISPLAY, MESSAGE, ('MESC1B',
                   'C'(11), 'C'(12), 'C'(13), 'C'(14), 'C'(15),
                   'C'(16), 'C'(17), 'C'(18), 'C'(19), 'C'(20),
                   'B'(11), 'B'(12), 'B'(13), 'B'(14), 'B'(15),
                   'B'(16), 'B'(17), 'B'(18), 'B'(19), 'B'(20),
                   'PINRES'(11), 'PINRES'(12), 'PINRES'(13),
                   'PINRES'(14), 'PINRES'(15), 'PINRES'(16),
                   'PINRES'(17), 'PINRES'(18), 'PINRES'(19),
                   'PINRES'(20)) $
CS
 248300          IF, 'PRINT' EQ 0, THEN $
    10             RESUME, FOR $
    20           END, IF $
C9
C            ------------------------------------------------------------ $
C            I FAILED OR FULL PRINT MODE GIVE PRINTOUT            I $
C            ------------------------------------------------------------ $
CS
 248400          PERFORM, 'PRINT-TITLE' $
    10           PRINT,   MESSAGE,                                    $
```

ANEXA 1

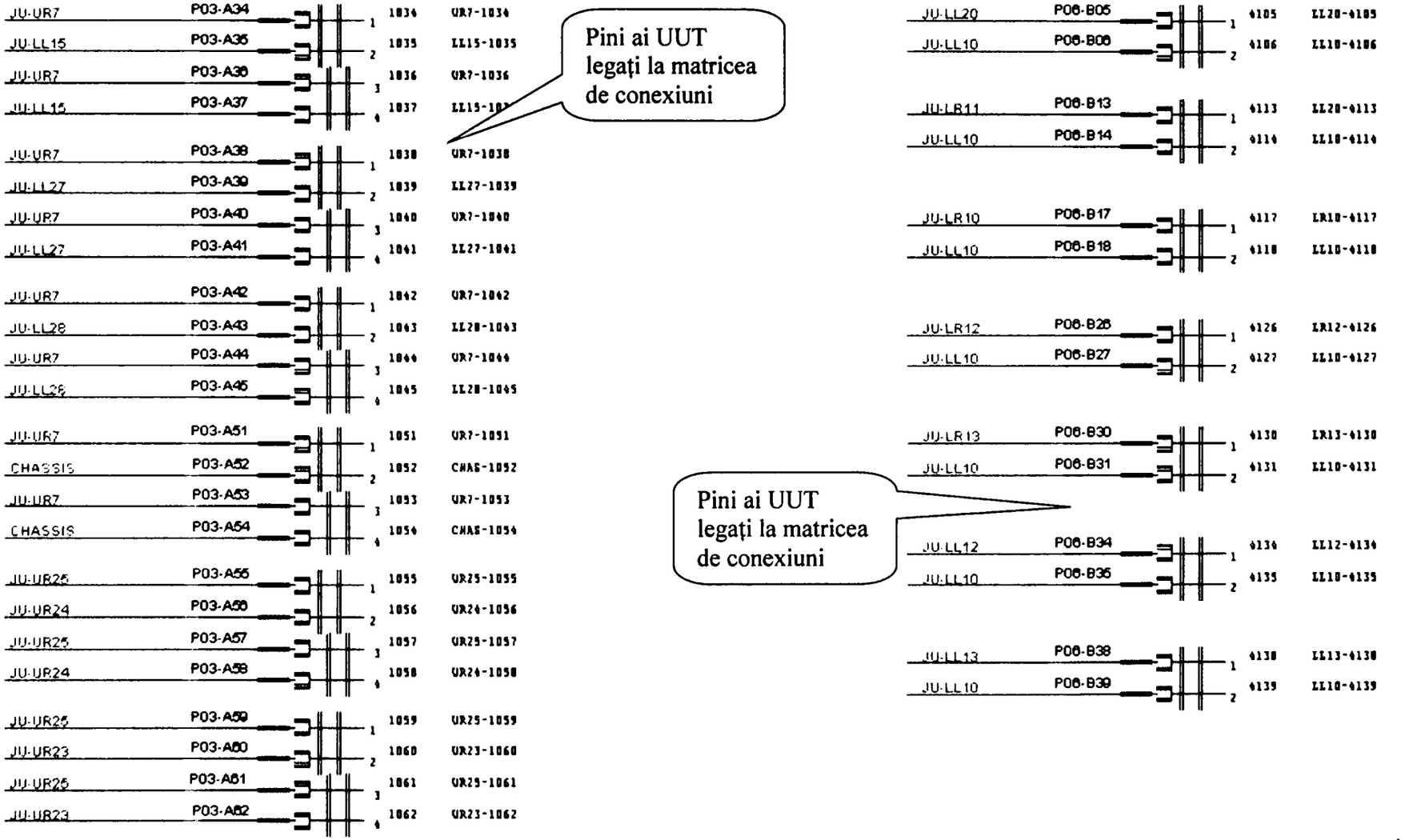
Descrierea în limbaj ATLAS 616 a testelor din blocul 2.0-CPU-RAM-CHECKS pentru unitatea AUTOTHROTTLE tip 10-62017-21 pentru avioanele Boeing 737-500 produs de firma Smith Industries Aerospace

```
                                                                    )
      20        PRINT,   MESSAGE, STORE PROGRAMME CONTENTS CHECK          $
      30        PRINT,   MESSAGE, BANK ('BANK' FORMAT +9) SELECTED        $
      40        PRINT,   MESSAGE,                                         $
  248450        PRINT,   MESSAGE, ('MESC1A',
                  'C'(1),'C'(2),'C'(3),'C'(4),'C'(5), 'C'(6), 'C'(7),
                  'C'(8), 'C'(9), 'C'(10),
                  'B'(1),'B'(2),'B'(3),'B'(4),'B'(5),'B'(6),'B'(7),
                  'B'(8),'B'(9), 'B'(10),
                  'PINRES'(1), 'PINRES'(2), 'PINRES'(3), 'PINRES'(4),
                  'PINRES'(5), 'PINRES'(6), 'PINRES'(7), 'PINRES'(8),
                  'PINRES'(9), 'PINRES'(10)) $
      60        PRINT,   MESSAGE, ('MESC1B',
                  'C'(11), 'C'(12), 'C'(13), 'C'(14), 'C'(15),
                  'C'(16), 'C'(17), 'C'(18), 'C'(19), 'C'(20),
                  'B'(11), 'B'(12), 'B'(13), 'B'(14), 'B'(15),
                  'B'(16), 'B'(17), 'B'(18), 'B'(19), 'B'(20),
                  'PINRES'(11), 'PINRES'( 2), 'PINRES'(13),
                  'PINRES'(14), 'PINRES'(15), 'PINRES'(16),
                  'PINRES'(17), 'PINRES'(18), 'PINRES'(19),
                  'PINRES'(20)) $
C$
C           ------------------------------------------------------------ $
C           I IF FAIL AND FAILSTOP SELECTED HALT PROGRAMME          I $
C           ------------------------------------------------------------ $
C$
  248500        IF, 'FF' GT 0.5, THEN $
      10          IF, 'OPTION'(27) EQ 1, THEN $
      20            PERFORM, 'FAILSTOP' $
      30          END, IF $
      40        END, IF $
  248600    END, FOR $
C$
C           ------------------------------------------------------------ $
C           I  GIVE OVERALL RESULT OF TEST                          I $
C           ------------------------------------------------------------ $
C$
  249000    CALCULATE, 'J' = 4 $
      10    PERFORM, 'BLOCK-RES' $
      20    REMOVE, ALL $
  249990 END, BLOCK, '2.4-STORE-PROGRAMME-CONTENTS-TEST' $
C$
  290000    REMOVE, ALL $
  299990 END, BLOCK, '2.0-CPU-RAM-CHECKS' $
C$
```

ANEXA 1

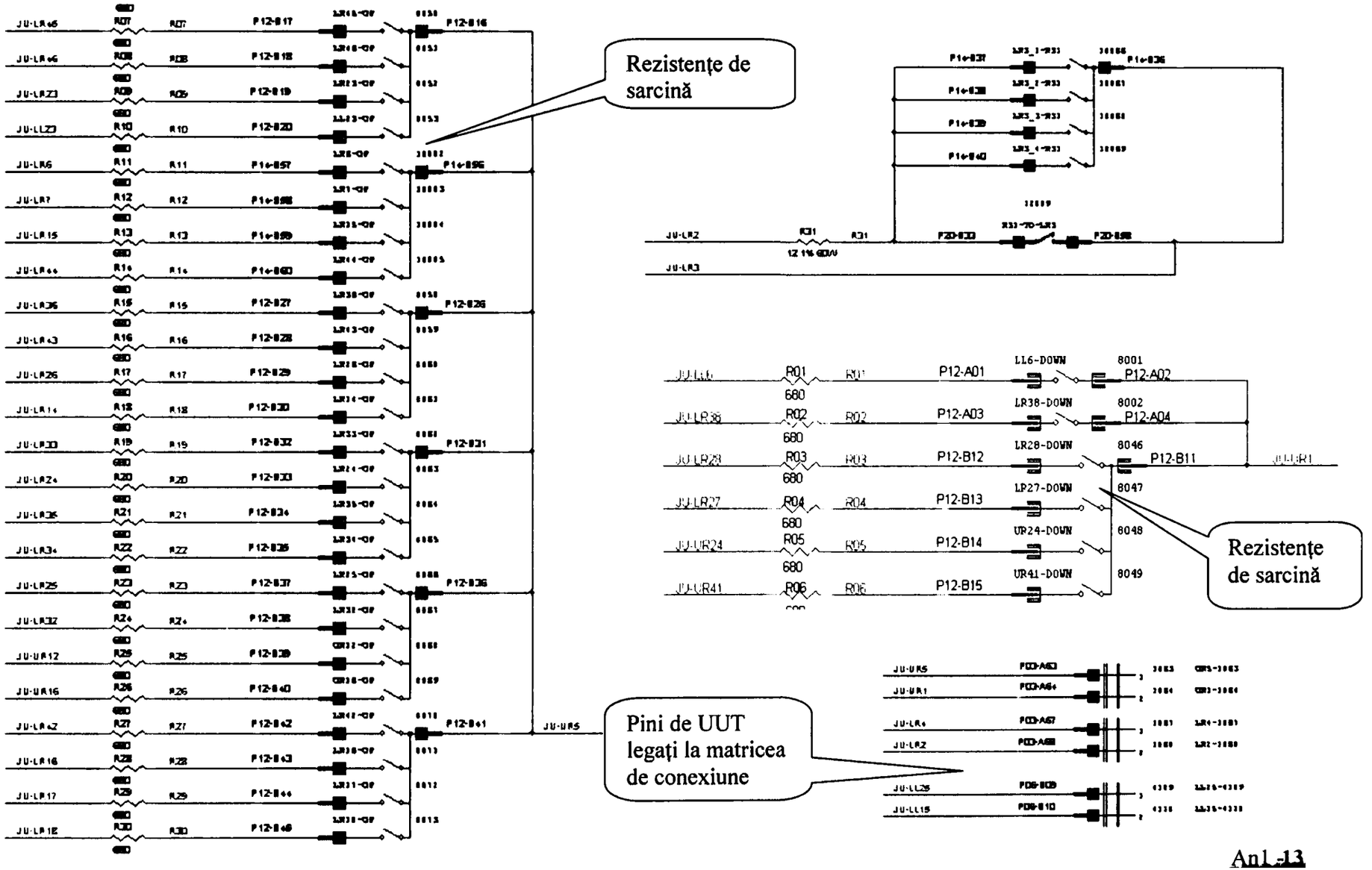
Scheme interfață de testa e (pentru testul 2) pentru unitatea

AUTOTHROTTLE tip 10-62017-21 pentru avioanel Boeing 737-500 produs de firma Smith Industries Aerospace



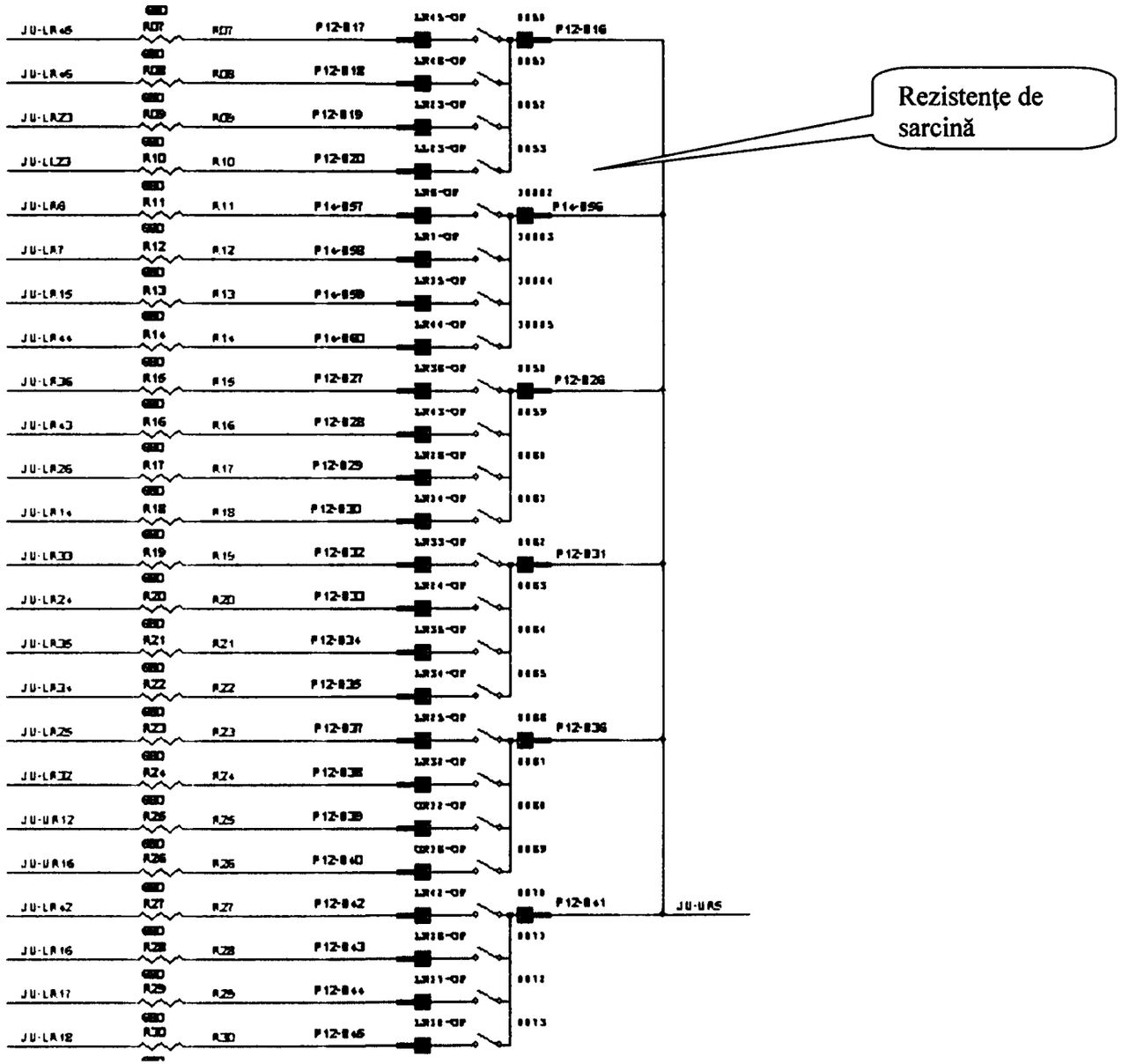
ANEXA 1

Scheme interfață de testare (pentru testul 2) pentru unitatea
AUTOTHROTTLE tip 10-62017-21 pentru avioanele Boeing 737-500 produs de firma Smith Industries Aerospace



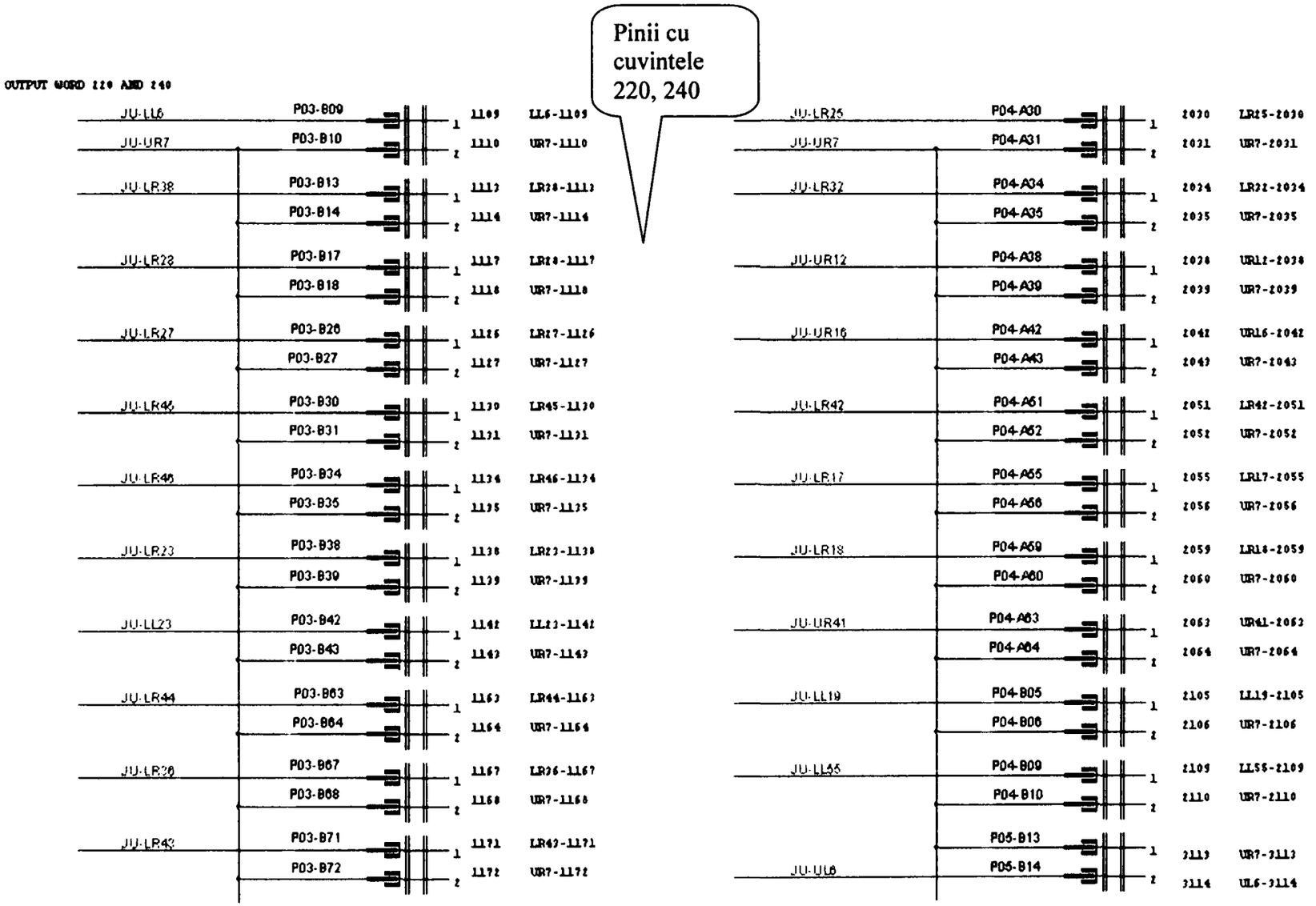
ANEXA 1

Scheme interfață de testare (pentru testul 2) pentru unitatea
AUTOTHROTTLE tip 10-62017-21 pentru avioanele Boeing 737-500 produs de firma Smith Industries Aerospace



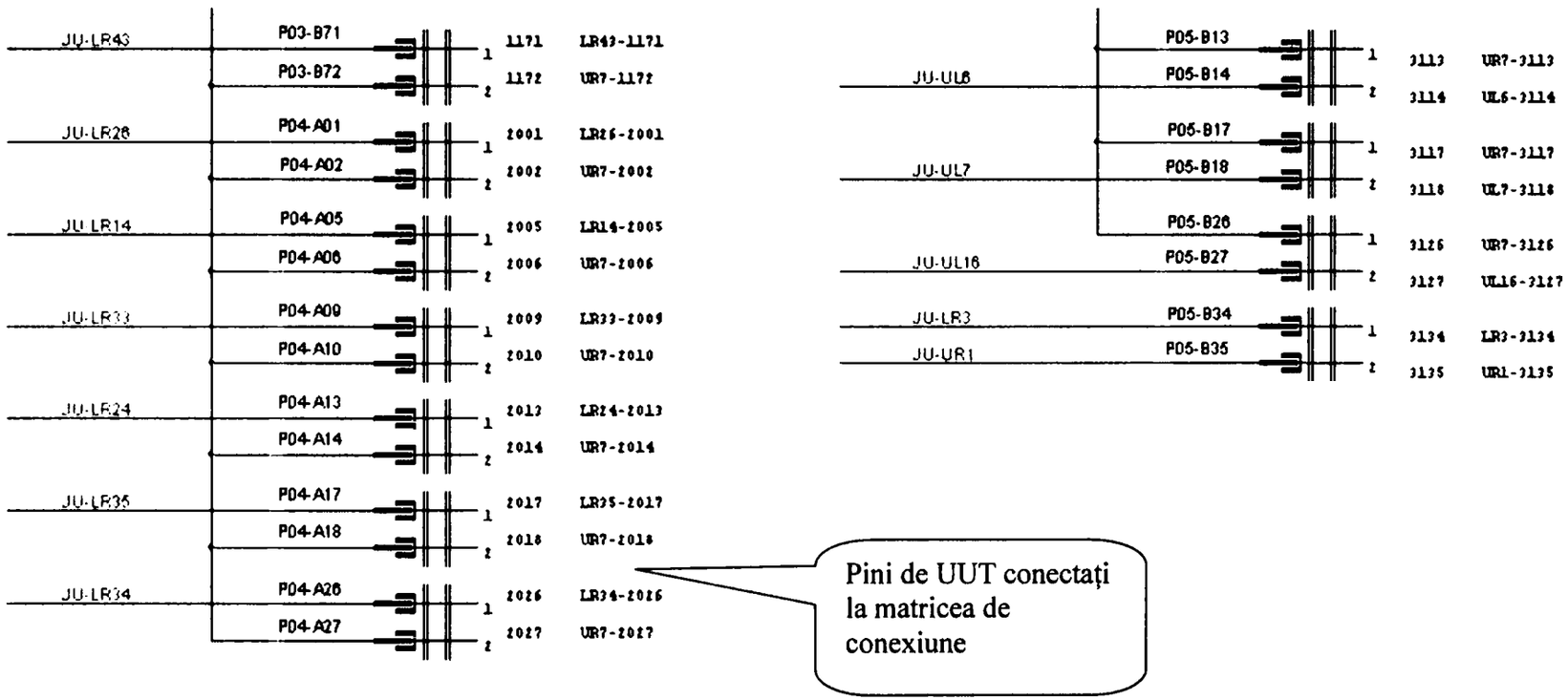
ANEXA 1

Scheme interfață de testare (pentru testul 2) pentru unitatea
AUTOTHROTTLE tip 10-62017-21 pentru avioanele Boeing 737-500 produs de firma Smith Industries Aerospace



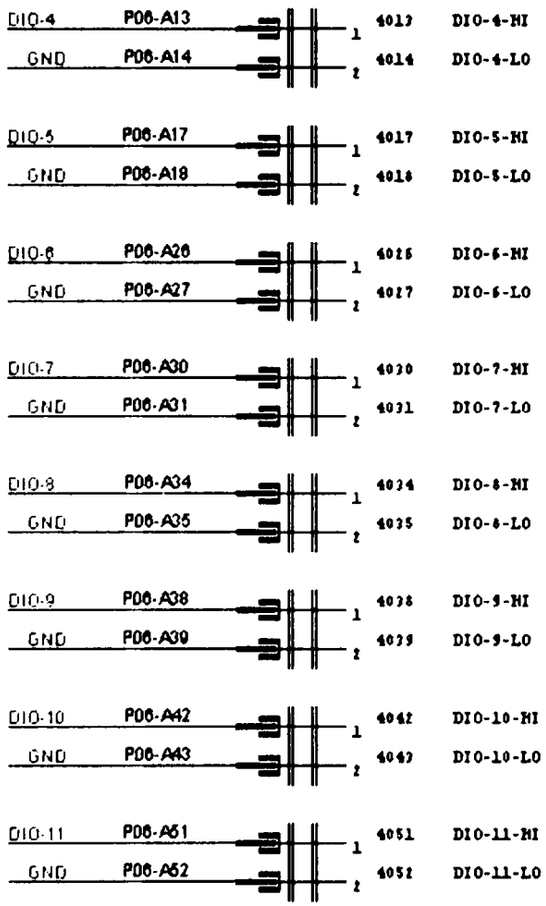
ANEXA 1

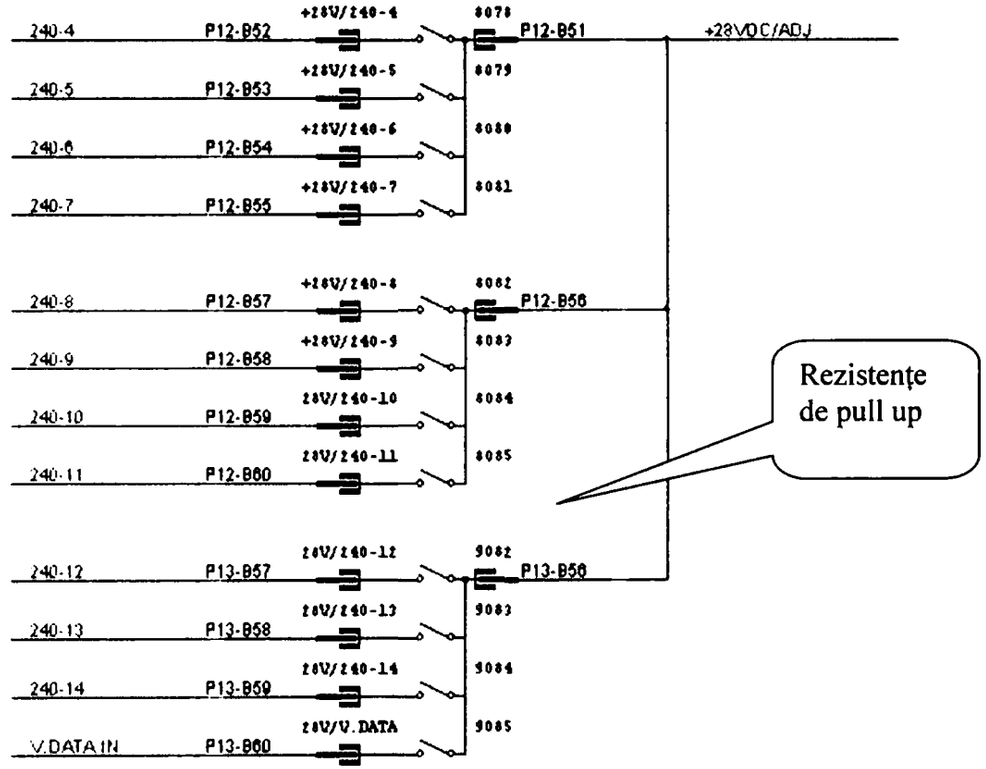
Scheme interfață de testare (pentru testul 2) pentru unitatea
AUTOTHROTTLE tip 10-62017-21 pentru avioanele Boeing 737-500 produs de firma Smith Industries Aerospace



ANEXA 1

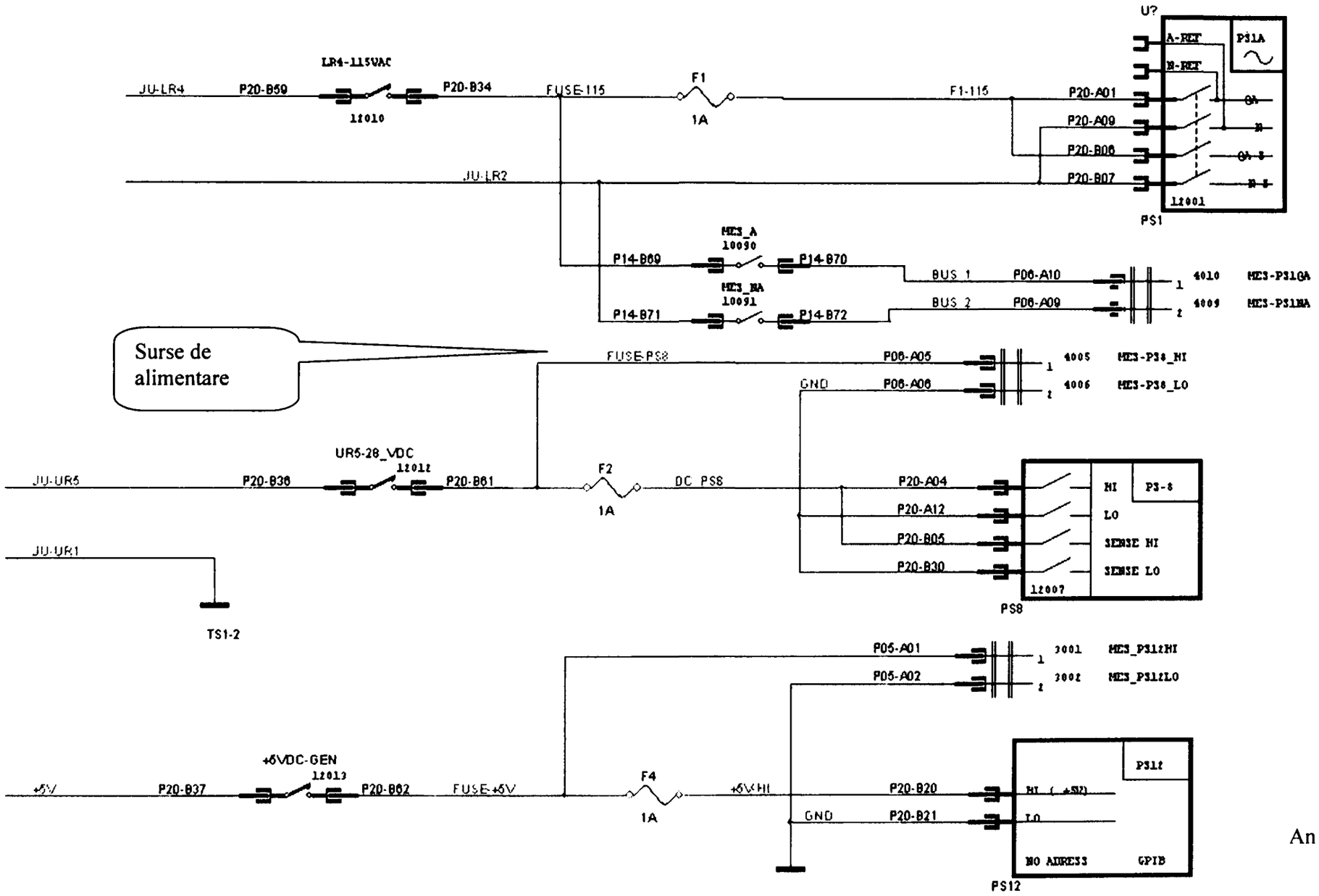
Scheme interfață de testare (pentru testul 2) pentru unitatea
AUTOTHROTTLE tip 10-62017-21 pentru avioanele Boeing 737-500 produs de firma Smith Industries Aerospace





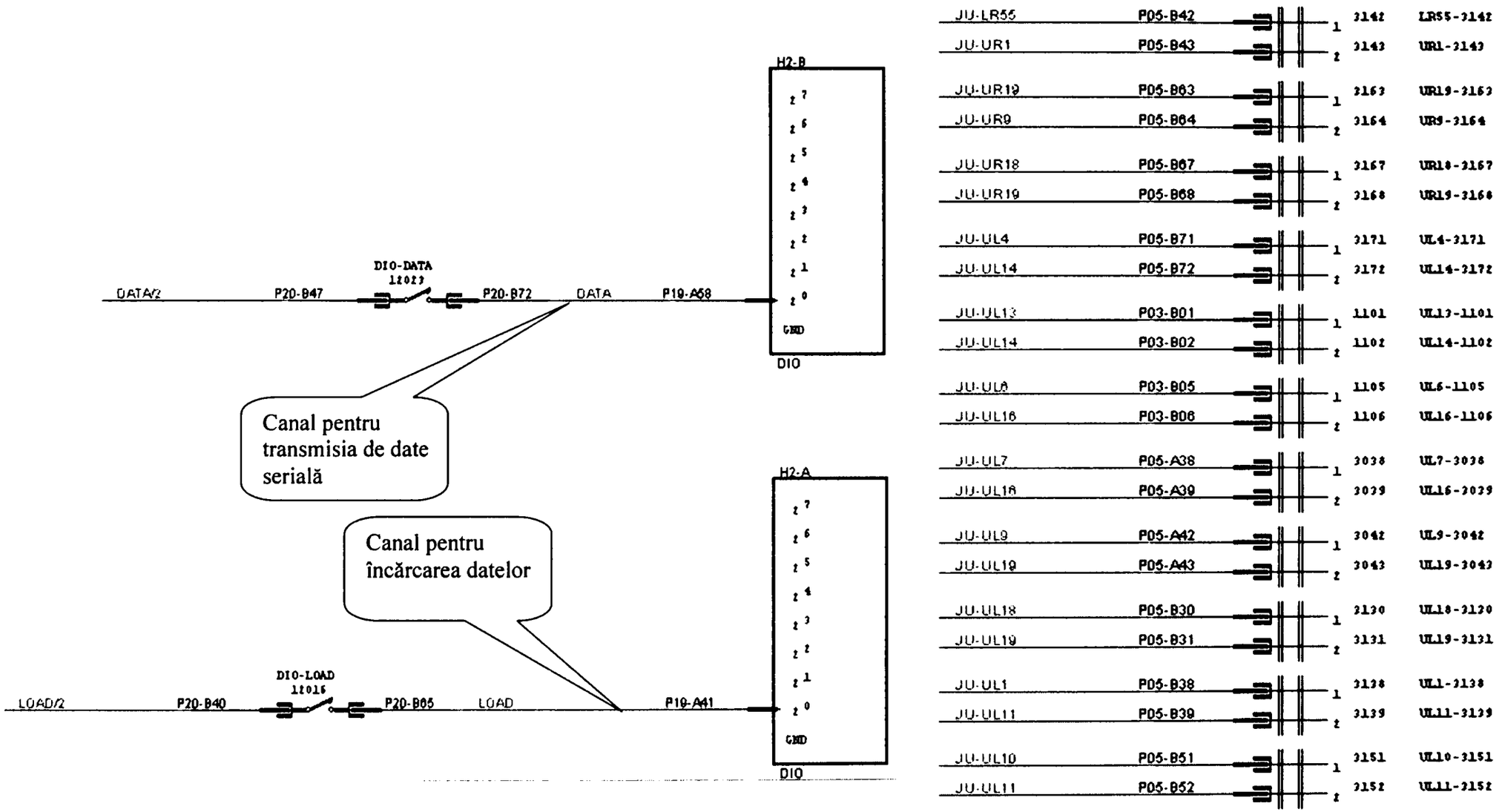
ANEXA 1

Scheme interfață de testare (pentru testul 2) pentru unitatea
AUTOTHROTTLE tip 10-62017-21 pentru avioanele Boeii g 737-500 produs de firma Smith Industries Aerospace



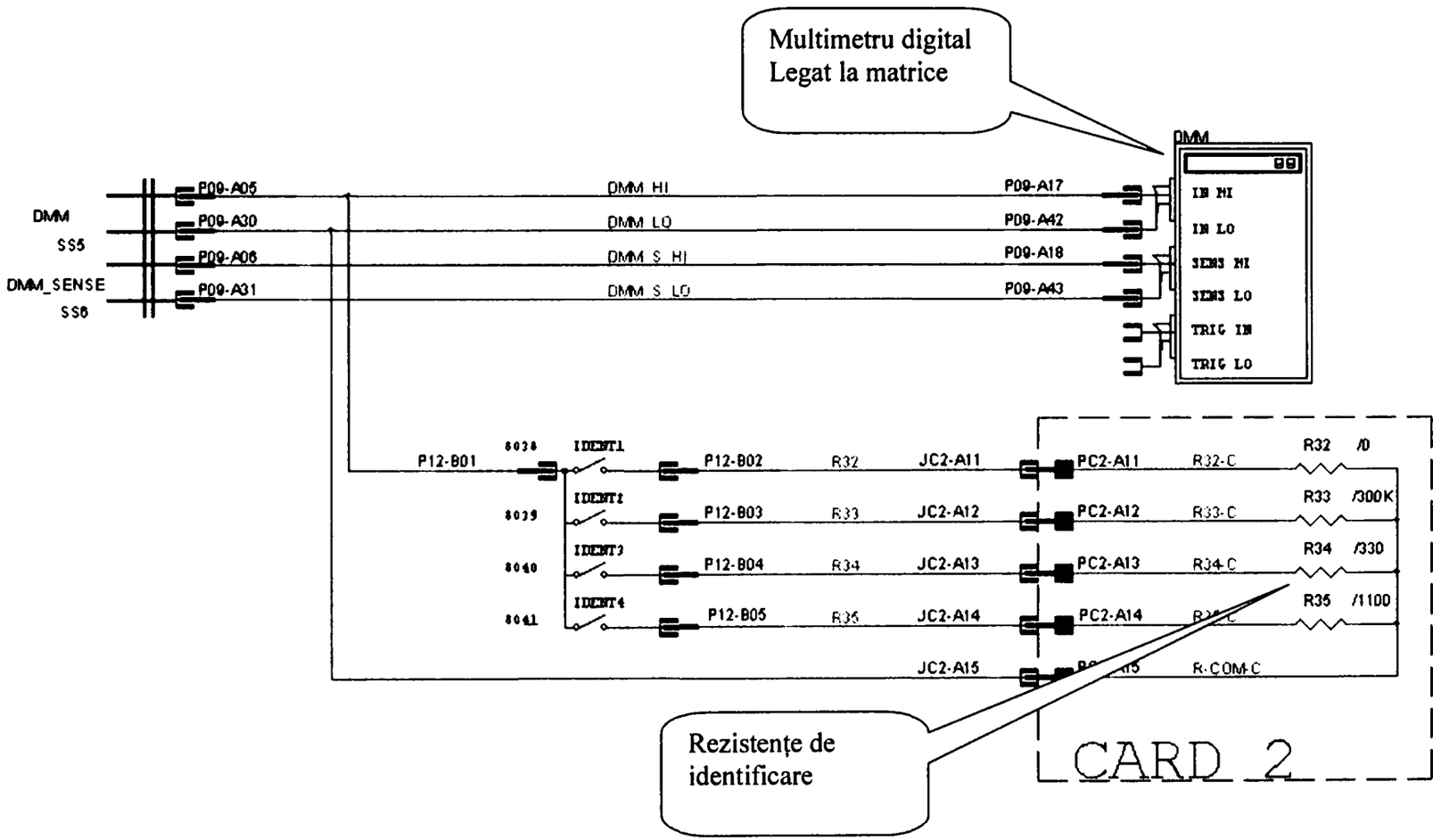
ANEXA 1

Scheme interfață de testare (pentru testul 2) pentru unitatea
AUTOTHROTTLE tip 10-62017-21 pentru avioanele Boeing 737-500 produs de firma Smith Industries Aerospace



ANEXA 1

Scheme interfață de testare (pentru testul 2) pentru unitatea
AUTOTHROTTLE tip 10-62017-21 pentru avioanele Boeing 737-500 produs de firma Smith Industries Aerospace



```
REM1
PRN_LINE          2.1 CPU MICROSTEP
REM2
SCREEN_R          DISP-SCR  SHOW    -      -
DISP_LINE         8     2     4     TESTS.HDR
DISP_LINE         9     3     4     TESTS.HDR
WAIT_SEC          3
REM1
PRN_LINE          BLOCK 2.1 CPU MICROSTEP          STEP 210000
REM       THIS BLOCK CARRIES OUT THE CPU MICROSTEP AS FOLLOWS
REM       THE UUT IS SET TO INITIAL CONDITIONS AND THE CPU
REM       DOWNLOAD PROGRAMME IS LOADED INTO THE UUT
REM2
REM1
REM       INITIALISE UUT.
REM       ABORT TESTING IF FAILURE OCCURS.
REM2
REM1
REM       APPLICATION OF PULLUP AND PULLDOWN LOADS
REM2
CALL      UNC     PULLUP&DW  FIRST-LINE NOT-PRINT
REM1
REM       CONNECT MOTORS
REM2
CALL      UNC     CON-MOT   FIRST-LINE NOT-PRINT
REM1
REM       PERFORM 'POWER UP'
REM2
CALL      UNC     POWER UP  FIRST-LINE NOT-PRINT
REM1
REM       DOWNLOAD 'CPU MICROSTEP' INTO UUT
REM       OPTION = 1. CPU MICROSTEP
REM2
DISP_LINE         32    5     20    MSG.HDR
LOADF     OPTION  1
CALL      UNC     PLOAD    NOT-PRINT  FIRST-LINE
SCREEN_R          DISP-SCR  SHOW    -      -
DISP_LINE         8     2     4     TESTS.HDR
DISP_LINE         9     3     4     TESTS.HDR
REM1
REM       VERIFY DOWNLOAD LOAD INTO UUT OK , 'PL-FLAG'=1
REM       IF FAIL INDICATE SO AND ABORT TEST.
REM2
CMPJMP            EQ      PL-FLAG  1      T2.1_C1
REM1
PRN_LINE          PROGRAMME DOWNLOAD FAILED
REM2
CALL      UNC     REMOVE ALL FIRST-LINE NOT-PRINT
BEEP      1
SCREEN_R  MSG-SCR  SHOW    -      -
DISP_LINE 2       2       4       TESTS.HDR
DISP_LINE 7       3       4       TESTS.HDR
DISP_LINE 9       4       25      MSG.HDR
DISP_LINE 2       5       25      MSG.HDR
SCREEN_R  MSG-SCR  ENTER   -      123456
CMPJMP    NE      COND_JMP  ON-FAIL  T2.1_S1
SCREEN_R  LOAD-FAIL ENTER    -      -
LABEL     T2.1_S1
JMP       UNC     SCR     100
LABEL     T2.1_C1
REM1
PRN_LINE  VERIFY WORDS 220 AND 240 BITS 15 THROUGH 4 ALL ON
PRN_LINE  STEP 211000
REM2
LOADF     Z1-Z2   0
REM1
PRN_LINE  VERIFY BIT 15 OF WORD 220 IS ON
REM2
M_VDC     LL6-1109  UR7-1110  =           28ñ3_VDC
JMP       SUC     LINE    T2.1_C2
```

```
LOADF     Z1-Z2    1
LABEL     T2.1_C2
REM1
PRN_LINE    VERIFY BIT 14 OF WORD 220 IS ON .
REM2
M_VDC     LR38-1113 UR7-1114  =          28ñ3_VDC
JMP       SUC     LINE     T2.1_C3
LOADF     Z1-Z2    1
LABEL     T2.1_C3
REM1
PRN_LINE    VERIFY BIT 13 OF WORD 220 IS ON
REM2
M_VDC     LR28-1117 UR7-1118  =          28ñ3_VDC
JMP       SUC     LINE     T2.1_C4
LOADF     Z1-Z2    1
LABEL     T2.1_C4
REM1
PRN_LINE    VERIFY BIT 12 OF WORD 220 IS ON .
REM2
M_VDC     LR27-1126 UR7-1127  =          28ñ3_VDC
JMP       SUC     LINE     T2.1_C5
LOADF     Z1-Z2    1
LABEL     T2.1_C5
REM1
PRN_LINE    VERIFY BIT 11 OF WORD 220 IS ON
REM2
M_VDC     LR45-1130 UR7-1131  =          0ñ3_VDC
JMP       SUC     LINE     T2.1_C6
LOADF     Z1-Z2    1
LABEL     T2.1_C6
REM1
PRN_LINE    VERIFY BIT 10 OF WORD 220 IS ON .
REM2
M_VDC     LR46-1134 UR7-1135  =          0ñ3_VDC
JMP       SUC     LINE     T2.1_C7
LOADF     Z1-Z2    1
LABEL     T2.1_C7
REM1
PRN_LINE    VERIFY BIT 9 OF WORD 220 IS ON .
REM2
M_VDC     LR23-1138 UR7-1139  =          0ñ3_VDC
JMP       SUC     LINE     T2.1_C8
LOADF     Z1-Z2    1
LABEL     T2.1_C8
REM1
PRN_LINE    VERIFY BIT 8 OF WORD 220 IS ON .
REM2
M_VDC     LL23-1142 UR7-1143  =          0ñ3_VDC
JMP       SUC     LINE     T2.1_C9
LOADF     Z1-Z2    1
LABEL     T2.1_C9
REM1
PRN_LINE    VERIFY BIT 15 OF WORD 240 IS ON .
REM2
M_VDC     LR6-2159  UR7-2160  =          0ñ3_VDC
JMP       SUC     LINE     T2.1_C10
LOADF     Z1-Z2    1
LABEL     T2.1_C10
REM1
PRN_LINE    VERIFY BIT 14 OF WORD 240 IS ON .
REM2
M_VDC     LR7-2163  UR7-2164  =          0ñ3_VDC
JMP       SUC     LINE     T2.1_C11
LOADF     Z1-Z2    1
LABEL     T2.1_C11
REM1
PRN_LINE    VERIFY BIT 13 OF WORD 240 IS ON .
REM2
M_VDC     LR15-2167 UR7-2168  =          0ñ3_VDC
JMP       SUC     LINE     T2.1_C12
LOADF     Z1-Z2    1
LABEL     T2.1_C12
REM1
```

```
PRN_LINE   VERIFY BIT 12 OF WORD 240 IS ON
REM2
M_VDC     LR44-1163 UR7-1164  =          0ñ3_VDC
JMP       SUC     LINE     T2.1_C13
LOADF     Z1-Z2    1
LABEL     T2.1_C13
REM1
PRN_LINE   VERIFY BIT 11 OF WORD 240 IS ON
REM2
M_VDC     LR36-1167 UR7-1168  =          0ñ3_VDC
JMP       SUC     LINE     T2.1_C14
LOADF     Z1-Z2    1
LABEL     T2.1_C14
REM1
PRN_LINE   VERIFY BIT 10 OF WORD 240 IS ON.
REM2
M_VDC     LR43-1171 UR7-1172  =          0ñ3_VDC
JMP       SUC     LINE     T2.1_C15
LOADF     Z1-Z2    1
LABEL     T2.1_C15
REM1
PRN_LINE   VERIFY BIT 9 OF WORD 240 IS ON.
REM2
M_VDC     LR26-2001 UR7-2002  =          0ñ3_VDC
JMP       SUC     LINE     T2.1_C16
LOADF     Z1-Z2    1
LABEL     T2.1_C16
REM1
PRN_LINE   VERIFY BIT 8 OF WORD 240 IS ON.
REM2
M_VDC     LR14-2005 UR7-2006  =          0ñ3_VDC
JMP       SUC     LINE     T2.1_C17
LOADF     Z1-Z2    1
LABEL     T2.1_C17
REM1
PRN_LINE   VERIFY BIT 7 OF WORD 240 IS ON.
REM2
M_VDC     LR33-2009 UR7-2010  =          0ñ3_VDC
JMP       SUC     LINE     T2.1_C18
LOADF     Z1-Z2    1
LABEL     T2.1_C18
REM1
PRN_LINE   VERIFY BIT 6 OF WORD 240 IS ON.
REM2
M_VDC     LR24-2013 UR7-2014  =          0ñ3_VDC
JMP       SUC     LINE     T2.1_C19
LOADF     Z1-Z2    1
LABEL     T2.1_C19
REM1
PRN_LINE   VERIFY BIT 5 OF WORD 240 IS ON
REM2
M_VDC     LR35-2017 UR7-2018  =          0ñ3_VDC
JMP       SUC     LINE     T2.1_C20
LOADF     Z1-Z2    1
LABEL     T2.1_C20
REM1
PRN_LINE   VERIFY BIT 4 OF WORD 240 IS ON.
REM2
M_VDC     LR34-2026 UR7-2027  =          0ñ3_VDC
JMP       SUC     LINE     T2.1_C21
LOADF     Z1-Z2    1
LABEL     T2.1_C21
REM1
PRN_LINE   VERIFY BIT 7 OF WORD 220 IS ON.
REM2
M_VDC     LR25-2030 UR7-2031  =          0ñ3_VDC
JMP       SUC     LINE     T2.1_C22
LOADF     Z1-Z2    1
LABEL     T2.1_C22
REM1
PRN_LINE   VERIFY BIT 6 OF WORD 220 IS ON.
REM2
M_VDC     LR32-2034 UR7-2035  =          0ñ3_VDC
```

```
JMP      SUC     LINE     T2.1_C23
LOADF     Z1-Z2    1
LABEL     T2.1_C23
REM1
PRN_LINE    VERIFY BIT 5 OF WORD 220 IS ON .
REM2
M_VDC     UR12-2038 UR7-2039  =            0ñ3_VDC
JMP      SUC     LINE     T2.1_C24
LOADF     Z1-Z2    1
LABEL     T2.1_C24
REM1
PRN_LINE    VERIFY BIT 4 OF WORD 220 IS ON
REM2
M_VDC     UR16-2042 UR7-2043  =            0ñ3_VDC
JMP      SUC     LINE     T2.1_C25
LOADF     Z1-Z2    1
LABEL     T2.1_C25
CMPJMP     EQ      Z1-Z2    0       T2.1_C26
REM1
PRN_LINE    WORDS 220 AND 240 BITS 15 THROUGH 4 NOT ALL ON.
PRN_LINE    TESTING ABORTED.
REM2
CALL      UNC      REMOVE ALL FIRST-LINE NOT-PRINT
BEEP      1
SCREEN_R    MSG-SCR   SHOW     -       -
DISP_LINE   8       2       4       TESTS.HDR
DISP_LINE   9       3       4       TESTS.HDR
DISP_LINE   11      4       10       MSG.HDR
DISP_LINE   2       5       10      MSG.HDR
SCREEN_R    MSG-SCR   ENTER    -       123456
CMPJMP     NE      COND_JMP  ON-FAIL   T2.1_TR1
BEEP      1
SCREEN_R    TRBL-SCR2 SHOW     -       -
DISP_LINE   10      3       18      TRBL.HDR
DISP_LINE   11      4       18      TRBL.HDR
DISP_LINE   12      5       18      TRBL.HDR
SCREEN_R    TRBL-SCR2 ENTER    -       123456
LABEL     T2.1_TR1
JMP      UNC      SCR      100
LABEL     T2.1_C26
REM1
REM       PERFORM A GO-AROUND AND VERIFY THAT CPU INSTRUCTION
REM       SET COUNT THROUGH HAS BEEN INITIATED.
REM2
CALL      UNC      GO-AROUND  FIRST-LINE NOT-PRINT
WAIT_SEC    2
REM1
PRN_LINE    CHECK FOR AND LR6 SHOULD INITIATION OF CPU COUNT.
PRN_LINE    LR43 BOTH BE 28ñ2_VDC. STEP 212220
REM2
LOADF     Z1-Z2    0
REM1
PRN_LINE    VERIFY BIT 10 OF WORD 240 IS OFF .
REM2
M_VDC     LR43-1171 UR7-1172  =            28ñ2_VDC
JMP      SUC     LINE     T2.1_C27
LOADF     Z1-Z2    1
LABEL     T2.1_C27
REM1
PRN_LINE    VERIFY BIT 15 OF WORD 240 IS OFF .
REM2
M_VDC     LR6-2159  UR7-2160  =            28ñ2_VDC
JMP      SUC     LINE     T2.1_C28
LOADF     Z1-Z2    1
LABEL     T2.1_C28
CMPJMP     EQ      Z1-Z2    0       T2.1_C29
REM1
PRN_LINE    CPU COUNT HAS NOT STARTED.
PRN_LINE    TESTING ABORTED.
REM2
CALL      UNC      REMOVE ALL FIRST-LINE NOT-PRINT
BEEP      1
SCREEN_R    MSG-SCR   SHOW     -       -
```

```
DISP_LINE  8     2     4     TESTS.HDR
DISP_LINE  9     3     4     TESTS.HDR
DISP_LINE  12    4     25    MSG.HDR
DISP_LINE  2     5     25    MSG.HDR
SCREEN_R   MSG-SCR  ENTER   -     123456
CMPJMP     NE     COND_JMP  ON-FAIL  T2.1_TR2
BEEP       1
SCREEN_R   TRBL-SCR2  SHOW   -     -
DISP_LINE  10    3     18    TRBL.HDR
DISP_LINE  11    4     18    TRBL.HDR
DISP_LINE  12    5     18    TRBL.HDR
SCREEN_R   TRBL-SCR2  ENTER   -     123456
LABEL      T2.1_TR2
JMP        UNC     SCR     100
LABEL      T2.1_C29
REM1
REM        CPU INSTRUCTION SET CYCLE WILL BE COMPLETE WITHIN
REM        30 SECONDS. DELAY 30 SECONDS AND THEN VERIFY WORDS
REM        220 AND 240 BITS 15 THROUGH 4 ALL ON, ALSO READ
REM        VOLTAGES PRESENT ON LR42, LR17, AND LR18
REM2
REM1
PRN_LINE   VERIFY WORDS 220 AND 240 BITS 15 THROUGH 4 ALL ON
PRN_LINE   STEP 213000
REM2
WAIT_SEC   30
LOADF      Z       177777
LOADF      Z1-Z2   0
REM1
PRN_LINE   VERIFY BIT 15 OF WORD 220 IS ON
REM2
M_VDC      LL6-1109  UR7-1110  =          28ñ3_VDC
JMP        SUC     LINE    T2.1_C30
LOADF      Z1-Z2   1
LABEL      T2.1_C30
REM1
PRN_LINE   VERIFY BIT 14 OF WORD 220 IS ON .
REM2
M_VDC      LR38-1113  UR7-1114  =          28ñ3_VDC
JMP        SUC     LINE    T2.1_C31
LOADF      Z1-Z2   1
LABEL      T2.1_C31
REM1
PRN_LINE   VERIFY BIT 13 OF WORD 220 IS ON .
REM2
M_VDC      LR28-1117  UR7-1118  =          28ñ3_VDC
JMP        SUC     LINE    T2.1_C32
LOADF      Z1-Z2   1
LABEL      T2.1_C32
REM1
PRN_LINE   VERIFY BIT 12 OF WORD 220 IS ON
REM2
M_VDC      LR27-1126  UR7-1127  =          28ñ3_VDC
JMP        SUC     LINE    T2.1_C33
LOADF      Z1-Z2   1
LABEL      T2.1_C33
REM1
PRN_LINE   VERIFY BIT 11 OF WORD 220 IS ON .
REM2
M_VDC      LR45-1130  UR7-1131  =          0ñ3_VDC
JMP        SUC     LINE    T2.1_C34
LOADF      Z1-Z2   1
LABEL      T2.1_C34
REM1
PRN_LINE   VERIFY BIT 10 OF WORD 220 IS ON .
REM2
M_VDC      LR46-1134  UR7-1135  =          0ñ3_VDC
JMP        SUC     LINE    T2.1_C35
LOADF      Z1-Z2   1
LABEL      T2.1_C35
REM1
PRN_LINE   VERIFY BIT 9 OF WORD 220 IS ON .
REM2
```

```
M_VDC     LR23-1138 UR7-1139 =          0ñ3_VDC
JMP       SUC     LINE    T2.1_C36
LOADF     Z1-Z2   1
LABEL     T2.1_C36
REM1
PRN_LINE  VERIFY BIT 8 OF WORD 220 IS ON.
REM2
M_VDC     LL23-1142 UR7-1143 =          0ñ3_VDC
JMP       SUC     LINE    T2.1_C37
LOADF     Z1-Z2   1
LABEL     T2.1_C37
REM1
PRN_LINE  VERIFY BIT 15 OF WORD 240 IS ON
REM2
M_VDC     LR6-2159  UR7-2160 =          0ñ3_VDC
JMP       SUC     LINE    T2.1_C38
LOADF     B(9)    1
LOADF     Z1-Z2   1
LABEL     T2.1_C38
REM1
PRN_LINE  VERIFY BIT 14 OF WORD 240 IS ON.
REM2
M_VDC     LR7-2163  UR7-2164 =          0ñ3_VDC
JMP       SUC     LINE    T2.1_C39
LOADF     B(10)   1
LOADF     Z1-Z2   1
LABEL     T2.1_C39
REM1
PRN_LINE  VERIFY BIT 13 OF WORD 240 IS ON.
REM2
M_VDC     LR15-2167 UR7-2168 =          0ñ3_VDC
JMP       SUC     LINE    T2.1_C40
LOADF     B(11)   1
LOADF     Z1-Z2   1
LABEL     T2.1_C40
REM1
PRN_LINE  VERIFY BIT 12 OF WORD 240 IS ON.
REM2
M_VDC     LR44-1163 UR7-1164 =          0ñ3_VDC
JMP       SUC     LINE    T2.1_C41
LOADF     B(12)   1
LOADF     Z1-Z2   1
LABEL     T2.1_C41
REM1
PRN_LINE  VERIFY BIT 11 OF WORD 240 IS ON.
REM2
M_VDC     LR36-1167 UR7-1168 =          0ñ3_VDC
JMP       SUC     LINE    T2.1_C42
LOADF     B(13)   1
LOADF     Z1-Z2   1
LABEL     T2.1_C42
REM1
PRN_LINE  VERIFY BIT 10 OF WORD 240 IS ON.
REM2
M_VDC     LR43-1171 UR7-1172 =          0ñ3_VDC
JMP       SUC     LINE    T2.1_C43
LOADF     Z1-Z2   1
LABEL     T2.1_C43
REM1
PRN_LINE  VERIFY BIT 9 OF WORD 240 IS ON.
REM2
M_VDC     LR26-2001 UR7-2002 =          0ñ3_VDC
JMP       SUC     LINE    T2.1_C44
LOADF     Z1-Z2   1
LABEL     T2.1_C44
REM1
PRN_LINE  VERIFY BIT 8 OF WORD 240 IS ON.
REM2
M_VDC     LR14-2005 UR7-2006 =          0ñ3_VDC
JMP       SUC     LINE    T2.1_C45
LOADF     Z1-Z2   1
LABEL     T2.1_C45
REM1
```

```
PRN_LINE   VERIFY BIT 7 OF WORD 240 IS ON .
REM2
M_VDC      LR33-2009 UR7-2010 =          0ñ3_VDC
JMP        SUC     LINE     T2.1_C46
LOADF      Z1-Z2    1
LABEL      T2.1_C46
REM1
PRN_LINE   VERIFY BIT 6 OF WORD 240 IS ON .
REM2
M_VDC      LR24-2013 UR7-2014 =          0ñ3_VDC
JMP        SUC     LINE     T2.1_C47
LOADF      Z1-Z2    1
LABEL      T2.1_C47
REM1
PRN_LINE   VERIFY BIT 5 OF WORD 240 IS ON
REM2
M_VDC      LR35-2017 UR7-2018 =          0ñ3_VDC
JMP        SUC     LINE     T2.1_C48
LOADF      Z1-Z2    1
LABEL      T2.1_C48
REM1
PRN_LINE   VERIFY BIT 4 OF WORD 240 IS ON
REM2
M_VDC      LR34-2026 UR7-2027 =          0ñ3_VDC
JMP        SUC     LINE     T2.1_C49
LOADF      Z1-Z2    1
LABEL      T2.1_C49
REM1
PRN_LINE   VERIFY BIT 7 OF WORD 220 IS ON .
REM2
M_VDC      LR25-2030 UR7-2031 =          0ñ3_VDC
JMP        SUC     LINE     T2.1_C50
LOADF      Z1-Z2    1
LABEL      T2.1_C50
REM1
PRN_LINE   VERIFY BIT 6 OF WORD 220 IS ON
REM2
M_VDC      LR32-2034 UR7-2035 =          0ñ3_VDC
JMP        SUC     LINE     T2.1_C51
LOADF      Z1-Z2    1
LABEL      T2.1_C51
REM1
PRN_LINE   VERIFY BIT 5 OF WORD 220 IS ON
REM2
M_VDC      UR12-2038 UR7-2039 =          0ñ3_VDC
JMP        SUC     LINE     T2.1_C52
LOADF      Z1-Z2    1
LABEL      T2.1_C52
REM1
PRN_LINE   VERIFY BIT 4 OF WORD 220 IS ON .
REM2
M_VDC      UR16-2042 UR7-2043 =          0ñ3_VDC
JMP        SUC     LINE     T2.1_C53
LOADF      Z1-Z2    1
LABEL      T2.1_C53
REM1
PRN_LINE   CHECK LR42, LR17 AND LR18
REM2
LOADF      Z1       0
LOADF      RESLTS   0
REM1
PRN_LINE   VERIFY LR42 IS 28_VDC
REM2
M_VDC      LR42-2051 UR7-2052 =          28ñ3_VDC
JMP        SUC     LINE     T2.1_C54
LOADF      RESLTS   1
LOADF      Z1-Z2    1
LABEL      T2.1_C54
REM1
PRN_LINE   VERIFY LR17 IS 28_VDC
REM2
M_VDC      LR17-2055 UR7-2056 =          28ñ3_VDC
JMP        SUC     LINE     T2.1_C55
```

```
LOADF     RESLTS   1
LOADF     Z1-Z2    1
LABEL     T2.1_C55
REM1
PRN_LINE  VERIFY LR18 IS 28_VDC
REM2
M_VDC     LR18-2059 UR7-2060 =          28ñ3_VDC
JMP       SUC     LINE     T2.1_C56
LOADF     RESLTS   1
LOADF     Z1-Z2    1
LABEL     T2.1_C56
CMPJMP    EQ      Z1-Z2    0       T2.1_C57
REM1
PRN_LINE  OUTPUT CODE HAS FAILED.
PRN_LINE  TESTING ABORTED.
REM2
CALL      UNC     REMOVE ALL FIRST-LINE NOT-PRINT
LOADF     Z       0
CMPJMP    NE      B(9)     1       T2.1_C58
ADD       Z       100000   Z
LABEL     T2.1_C58
CMPJMP    NE      B(10)    1       T2.1_C59
ADD       Z       40000    Z
LABEL     T2.1_C59
CMPJMP    NE      B(11)    1       T2.1_C60
ADD       Z       20000    Z
LABEL     T2.1_C60
CMPJMP    NE      B(12)    1       T2.1_C61
ADD       Z       10000    Z
LABEL     T2.1_C61
CMPJMP    NE      B(13)    1       T2.1_C62
ADD       Z       4000     Z
LABEL     T2.1_C62
CMPJMP    NE      RESLTS   1       T2.1_C63
ADD       Z       40000    Z1
LABEL     T2.1_C63
REM1
PRN_LINE  PROGRAMME HAS CORRUPTED       STEP 216200
PRN_LINE  LAST TEST TO PASS STEP:
REM2
BETWEEN   0       Z       8.999E+36       LOG
REM1
PRN_LINE  PROGRAM CORRUPTED AT :
REM2
BETWEEN   0       Z1      8.999E+36       LOG
BEEP      1
SCREEN_R  MSG-SCR  SHOW    -       -
DISP_LINE 8       2       4       TESTS.HDR
DISP_LINE 7       3       4       TESTS.HDR
DISP_LINE 13      4       25      MSG.HDR
DISP_LINE 2       5       25      MSG.HDR
SCREEN_R  MSG-SCR  ENTER   -       123456
CMPJMP    NE      COND_JMP ON-FAIL  T2.1_TR3
BEEP      1
SCREEN_R  TRBL-SCR2 SHOW   -       -
DISP_LINE 10      3       18      TRBL.HDR
DISP_LINE 11      4       18      TRBL.HDR
DISP_LINE 12      5       18      TRBL.HDR
SCREEN_R  TRBL-SCR2 ENTER  -       123456
LABEL     T2.1_TR3
JMP       UNC     SCR      100
LABEL     T2.1_C57
CALL      UNC     REMOVE ALL FIRST-LINE NOT-PRINT
NOP



REM1
PRN_LINE  2.2 STORE RAM UPPER
REM2
LOADF     TEST_NAME 14
```

```
SCREEN_R   DISP-SCR  SHOW     -      -
DISP_LINE  8       2       4       TESTS.HDR
DISP_LINE  10      3       4       TESTS.HDR
WAIT_SEC   3
REM1
PRN_LINE   BLOCK 2.2 STORE RAM UPPER          STEP 220000
REM        THIS BLOCK CARRIES OUT THE STORE RAM UPPER TEST.
REM2
REM1
REM        INITIALISE UUT.
REM        ABORT TESTING IF FAILURE OCCURS.
REM2
REM1
REM        APPLICATION OF PULLUP AND PULLDOWN LOADS
REM2
CALL       UNC      PULLUP&DW  FIRST-LINE NOT-PRINT
REM1
REM        CONNECT MOTORS
REM2
CALL       UNC      CON-MOT    FIRST-LINE NOT-PRINT
REM1
REM        PERFORM 'POWER UP'
REM2
CALL       UNC      POWER UP   FIRST-LINE NOT-PRINT
REM1
REM        DOWNLOAD 'STORE RAM UPPER' INTO UUT
REM        OPTION = 2. STORE RAM UPPER
REM2
DISP_LINE  32      5       20      MSG.HDR
LOADF      OPTION   2
CALL       UNC      PLOAD     NOT-PRINT  FIRST-LINE
SCREEN_R   DISP-SCR  SHOW     -      -
DISP_LINE  8       2       4       TESTS.HDR
DISP_LINE  10      3       4       TESTS.HDR
REM1
REM        VERIFY DOWNLOAD LOAD INTO UUT OK , 'PL-FLAG'=1
REM        IF FAIL INDICATE SO AND ABORT TEST.
REM2
CMPJMP     EQ       PL-FLAG   1       T2.2_C1
REM1
PRN_LINE   PROGRAMME DOWNLOAD FAILED
REM2
CALL       UNC      REMOVE ALL FIRST-LINE NOT-PRINT
BEEP       1
SCREEN_R   MSG-SCR   SHOW     -      -
DISP_LINE  2       2       4       TESTS.HDR
DISP_LINE  7       3       4       TESTS.HDR
DISP_LINE  9       4       25      MSG.HDR
DISP_LINE  2       5       25      MSG.HDR
SCREEN_R   MSG-SCR   ENTER    -       123456
CMPJMP     NE       COND_JMP  ON-FAIL   T2.2_S1
SCREEN_R   LOAD-FAIL  ENTER    -      -
LABEL      T2.2_S1
JMP        UNC      SCR      100
LABEL      T2.2_C1
REM1
REM        SET UP CODE 7070707 THEN APPLY GO-AROUND
REM2
CALL       UNC      SET7070707 FIRST-LINE NOT-PRINT
CALL       UNC      GO-AROUND  FIRST-LINE NOT-PRINT
REM1
PRN_LINE   AFTER SHORT DELAY VERIFY LR6, LR7 AND LR15 ALL AT 0_V
PRN_LINE   STEP 221100
REM2
WAIT_SEC   2
LOADF      RESLTS    1
REM1
PRN_LINE   VERIFY LR6 AT 0_V
REM2
M_VDC      LR6-2159  UR7-2160  B(1)           0ñ2_VDC
JMP        SUC      LINE     T2.2_C2
LOADF      RESLTS   -1
LABEL      T2.2_C2
```

```
REM1
PRN_LINE   VERIFY LR7 AT 0_V
REM2
M_VDC      LR7-2163  UR7-2164  B(2)          0ñ2_VDC
JMP        SUC     LINE     T2.2_C3
LOADF      RESLTS   -1
LABEL      T2.2_C3
REM1
PRN_LINE   VERIFY LR15 AT 0_V
REM2
M_VDC      LR15-2167 UR7-2168  B(3)          0ñ2_VDC
JMP        SUC     LINE     T2.2_C4
LOADF      RESLTS   -1
LABEL      T2.2_C4
CMPJMP     EQ      RESLTS   1       T2.2_C8
REM1
PRN_LINE   TEST FAILED GIVE DIAGNOSTICS       STEP 222000
PRN_LINE   : RAM DECODE BOARD SUSPECT AT FOLOWING LOCATIONS
REM2
CMPJMP     LESS     B(1)     2       T2.2_C5
REM1
PRN_LINE   LOCATION C3
REM2
LABEL      T2.2_C5
CMPJMP     LESS     B(2)     2       T2.2_C6
REM1
PRN_LINE   LOCATION C2
REM2
LABEL      T2.2_C6
CMPJMP     LESS     B(3)     2       T2.2_C7
REM1
PRN_LINE   LOCATION C1
REM2
LABEL      T2.2_C7
LOADF      TRBL_LINE 15
JMP        UNC      WHERE_JMP  VALUE_JMP
LABEL      T2.2_C8
REM1
REM        REMOVE 'SET-7070707'
REM2
CON        LR22-GND
CON        LL22-GND
CON        LR37-GND
DISCON     LL36-28V
DISCON     UL3-28V
DISCON     UL2-28V
DISCON     LL2-28V
DISCON     UL53-28V
DISCON     UL32-28V
DISCON     UL35-28V
DISCON     UL34-28V
DISCON     UL25-28V
CALL       UNC      REMOVE ALL FIRST-LINE NOT-PRINT
NOP



REM1
PRN_LINE   2.3 STORE RAM LOWER
REM2
LOADF      TEST_NAME 17
SCREEN_R   DISP-SCR  SHOW     -       -
DISP_LINE  8       2       4       TESTS.HDR
DISP_LINE  11      3       4       TESTS.HDR
WAIT_SEC   3
REM1
PRN_LINE   BLOCK 2.3 STORE RAM LOWER          STEP 230000
REM        THIS BLOCK CARRIES OUT THE STORE RAM LOWER TEST.
REM2
REM1
REM        INITIALISE UUT.
REM        ABORT TESTING IF FAILURE OCCURS.
REM2
```

ANEXA 1

Secvențele de testare ale testelor din blocul 2.0-CPU-RAM-CHECKS pentru unitatea AUTOTHROTTLE tip 10-62017-21 pentru avioanele Boeing 737-500 produs de firma Smith Industries Aerospace

```
REM1
REM      APPLICATION OF PULLUP AND PULLDOWN LOADS
REM2
CALL     UNC     PULLUP&DW  FIRST-LINE NOT-PRINT
REM1
REM      CONNECT MOTORS
REM2
CALL     UNC     CON-MOT   FIRST-LINE NOT-PRINT
REM1
REM      PERFORM 'POWER UP'
REM2
CALL     UNC     POWER UP   FIRST-LINE NOT-PRINT
REM1
REM      DOWNLOAD 'STORE RAM LOWER' INTO UUT
REM      OPTION = 3. STORE RAM LOWER
REM2
DISP_LINE  32     5      20      MSG HDR
LOADF     OPTION   3
CALL      UNC     PLOAD    NOT-PRINT  FIRST-LINE
SCREEN_R   DISP-SCR  SHOW    -      -
DISP_LINE  8      2      4       TESTS.HDR
DISP_LINE  11     3      4       TESTS HDR
REM1
REM      VERIFY DOWNLOAD LOAD INTO UUT OK , 'PL-FLAG'=1
REM      IF FAIL INDICATE SO AND ABORT TEST.
REM2
CMPJMP    EQ      PL-FLAG  1       T2.3_C1
REM1
PRN_LINE   PROGRAMME DOWNLOAD FAILED
REM2
CALL      UNC     REMOVE ALL FIRST-LINE NOT-PRINT
BEEP      1
SCREEN_R   MSG-SCR   SHOW    -      -
DISP_LINE  2      2      4       TESTS.HDR
DISP_LINE  7      3      4       TESTS.HDR
DISP_LINE  9      4      25      MSG.HDR
DISP_LINE  2      5      25      MSG.HDR
SCREEN_R   MSG-SCR   ENTER    -      123456
CMPJMP    NE      COND_JMP  ON-FAIL   T2.3_S1
SCREEN_R   LOAD-FAIL  ENTER    -      -
LABEL     T2.3_S1
JMP      UNC     SCR     100
LABEL     T2.3_C1
REM1
REM      SET UP CODE 7070707 THEN APPLY GO-AROUND
REM2
CALL      UNC     SET7070707 FIRST-LINE NOT-PRINT
CALL      UNC     GO-AROUND  FIRST-LINE NOT-PRINT
REM1
PRN_LINE   AFTER SHORT DELAY VERIFY LR6, LR7 AND LR15 ALL AT 0_V
PRN_LINE   STEP 231100
REM2
WAIT_SEC   2
LOADF     RESLTS   1
REM1
PRN_LINE   VERIFY LR6 AT 0_V
REM2
M_VDC     LR6-2159  UR7-2160  B(1)          0ñ2_VDC
JMP      SUC     LINE     T2.3_C2
LOADF     RESLTS   -1
LABEL     T2.3_C2
REM1
PRN_LINE   VERIFY LR7 AT 0_V
REM2
M_VDC     LR7-2163  UR7-2164  B(2)          0ñ2_VDC
JMP      SUC     LINE     T2.3_C3
LOADF     RESLTS   -1
LABEL     T2.3_C3
REM1
PRN_LINE   VERIFY LR15 AT 0_V
REM2
M_VDC     LR15-2167 UR7-2168  B(3)          0ñ2_VDC
JMP      SUC     LINE     T2.3_C4
```

```
LOADF     RESLTS   -1
LABEL     T2.3_C4
CMPJMP    EQ       RESLTS   1       T2.3_C8
REM1
PRN_LINE  TEST FAILED GIVE DIAGNOSTICS       STEP 232000
PRN_LINE  RAM DECODE BOARD SUSPECT AT FOLOWING LOCATIONS
REM2
CMPJMP    LESS     B(1)     2       T2.3_C5
REM1
PRN_LINE  LOCATION C3
REM2
LABEL     T2.3_C5
CMPJMP    LESS     B(2)     2       T2.3_C6
REM1
PRN_LINE  LOCATION C2
REM2
LABEL     T2.3_C6
CMPJMP    LESS     B(3)     2       T2.3_C7
REM1
PRN_LINE  LOCATION C1
REM2
LABEL     T2.3_C7
LOADF     TRBL_LINE 18
JMP       UNC      WHERE_JMP VALUE_JMP
LABEL     T2.3_C8
REM1
REM       REMOVE 'SET-7070707'
REM2
CON       LR22-GND
CON       LL22-GND
CON       LR37-GND
DISCON    LL36-28V
DISCON    UL3-28V
DISCON    UL2-28V
DISCON    LL2-28V
DISCON    UL53-28V
DISCON    UL32-28V
DISCON    UL35-28V
DISCON    UL34-28V
DISCON    UL25-28V
CALL      UNC      REMOVE ALL FIRST-LINE NOT-PRINT
NOP


REM1
PRN_LINE  2.4 STORE PROGRAMME CONTENTS
REM2
LOADF     TEST_NAME 20
SCREEN_R  DISP-SCR SHOW    -       -
DISP_LINE 8       2       4       TESTS.HDR
DISP_LINE 12      3       4       TESTS.HDR
WAIT_SEC  3
REM1
PRN_LINE  BLOCK 2.4 STORE PROGRAMME CONTENTS   STEP 240000
REM       THIS BLOCK CARRIES OUT THE STORE PROGRAMME CONTENTS
REM       TEST.
REM2
REM1
REM       INITIALISE UUT.
REM       ABORT TESTING IF FAILURE OCCURS.
REM2
REM1
REM       APPLICATION OF PULLUP AND PULLDOWN LOADS
REM2
CALL      UNC      PULLUP&DW  FIRST-LINE NOT-PRINT
REM1
REM       CONNECT MOTORS
REM2
CALL      UNC      CON-MOT    FIRST-LINE NOT-PRINT
REM1
REM       PERFORM 'POWER UP'
REM2
CALL      UNC      POWER UP   FIRST-LINE NOT-PRINT
```

```
REM1
REM       DOWNLOAD 'STORE PROGRAMME CONTENTS' INTO UUT
REM       OPTION = 4. STORE PROGRAMME CONTENTS
REM2
DISP_LINE  32     5      20      MSG.HDR
LOADF      OPTION   4
CALL       UNC      PLOAD    NOT-PRINT  FIRST-LINE
SCREEN_R   DISP-SCR  SHOW     -       -
DISP_LINE  8      2      4       TESTS.HDR
DISP_LINE  12     3      4       TESTS.HDR
REM1
REM       VERIFY DOWNLOAD LOAD INTO UUT OK , 'PL-FLAG'=1
REM       IF FAIL INDICATE SO AND ABORT TEST.
REM2
CMPJMP     EQ       PL-FLAG   1       T2.4_C1
REM1
PRN_LINE   PROGRAMME DOWNLOAD FAILED
REM2
CALL       UNC      REMOVE ALL FIRST-LINE NOT-PRINT
BEEP       1
SCREEN_R   MSG-SCR   SHOW     -       -
DISP_LINE  8      2      4       TESTS.HDR
DISP_LINE  12     3      4       TESTS.HDR
DISP_LINE  9      4      25      MSG.HDR
DISP_LINE  2      5      25      MSG.HDR
SCREEN_R   MSG-SCR   ENTER    -       123456
CMPJMP     NE       COND_JMP  ON-FAIL   T2.4_S1
SCREEN_R   LOAD-FAIL  ENTER    -       -
LABEL      T2.4_S1
JMP        UNC      SCR      100
LABEL      T2.4_C1
REM1
REM       LENGTHS OF ALL CRC CHECKS ARE THE SAME (I.E.040000)
REM       SO SET THE INPUTS UP TO THIS.
REM2
CON        UL3-28V
REM1
PRN_LINE   CHECK OUTPUTS FOR BOTH BANKS (0 AND 1).  STEP 246000
REM2
REM1
PRN_LINE   BANK 0
REM2
CALL       UNC      GO-AROUND  FIRST-LINE NOT-PRINT
REM1
PRN_LINE   BANK 0, LR10 < -5V IMMEDIATELY AFTER GO-AROUND.
REM2
M_VDC      LR10-2134  UR7-2135  =              <-5_VDC
JMP        SUC      LINE     T2.4_C2
REM1
PRN_LINE   BANK 0 SELECTED, LR10 CHECK FAILED,
PRN_LINE   LR10 GREATER -5_VDC IMMEDIATELY AFTER GO-AROUND.
REM2
CALL       UNC      REMOVE ALL FIRST-LINE NOT-PRINT
BEEP       1
SCREEN_R   MSG-SCR   SHOW     -       -
DISP_LINE  2      2      4       TESTS.HDR
DISP_LINE  7      3      4       TESTS.HDR
DISP_LINE  14     4      25      MSG.HDR
DISP_LINE  2      5      25      MSG.HDR
SCREEN_R   MSG-SCR   ENTER    -       123456
LOADF      TRBL_LINE 21
JMP        UNC      WHERE_JMP  VALUE_JMP
JMP        UNC      SCR      100
LABEL      T2.4_C2
REM1
PRN_LINE   BANK 0, LR10 > +5V WITHIN 20 SECONDS AFTER GO-AROUND.
REM2
ELAPS_TIME  0       0
LABEL      T2.4_L1
CON_MAT    DMM      LR10-2134  UR7-2135
READ_DMM   VDC      MEAS_VOLT
DISCON_MAT  DMM      LR10-2134  UR7-2135
ELAPS_TIME  1       TIME
```

ANEXA 1

Secvențele de testare ale testelor din blocul 2.0-CPU-RAM-CHECKS pentru unitatea AUTOTHROTTLE tip 10-62017-21 pentru avioanele Boeing 737-500 produs de firma Smith Industries Aerospace

```
CMPJMP    GREATER  TIME     20       T2.4_C3
CMPJMP    GREATER  MEAS_VOLT 5       T2.4_C4
JMP       UNC      LINE     T2.4_L1
LABEL     T2.4_C3
BETWEEN   5        MEAS_VOLT 50             LOG
REM1
PRN_LINE  BANK 0 SELECTED, LR10 CHECK FAILED,
PRN_LINE  20 SECONDS AFTER GO-AROUND LR10 STILL NOT ABOVE +5_V
REM2
CALL      UNC      REMOVE ALL FIRST-LINE NOT-PRINT
BEEP      1
SCREEN_R  MSG-SCR  SHOW     -       -
DISP_LINE 2        2        4        TESTS.HDR
DISP_LINE 7        3        4        TESTS.HDR
DISP_LINE 14       4        25       MSG.HDR
DISP_LINE 2        5        25       MSG.HDR
SCREEN_R  MSG-SCR  ENTER    -        123456
LOADF     TRBL_LINE 21
JMP       UNC      WHERE_JMP VALUE_JMP
JMP       UNC      SCR      100
LABEL     T2.4_C4
BETWEEN   5        MEAS_VOLT 50             LOG
REM1
PRN_LINE  VERIFY OUTPUT WORD BIT FOR BANK 0   STEP 248000
PRN_LINE  OFF=28_VDC
PRN_LINE  ON=0_VDC
REM2
REM1
PRN_LINE  VERIFY BIT 20 (MSB) OF BANK 0 IS ON
REM2
M_VDC     LR45-1130 UR7-1131  =            0ñ2_VDC
LOADF     TRBL_LINE 21
JMP       COND_JMP  WHERE_JMP VALUE_JMP
REM1
PRN_LINE  VERIFY BIT 19 OF BANK 0 IS OFF
REM2
M_VDC     LR46-1134 UR7-1135  =            28ñ2_VDC
LOADF     TRBL_LINE 21
JMP       COND_JMP  WHERE_JMP VALUE_JMP
REM1
PRN_LINE  VERIFY BIT 18 OF BANK 0 IS OFF
REM2
M_VDC     LR23-1138 UR7-1139  =            28ñ2_VDC
LOADF     TRBL_LINE 21
JMP       COND_JMP  WHERE_JMP VALUE_JMP
REM1
PRN_LINE  VERIFY BIT 17 OF BANK 0 IS ON
REM2
M_VDC     LL23-1142 UR7-1143  =            0ñ2_VDC
LOADF     TRBL_LINE 21
JMP       COND_JMP  WHERE_JMP VALUE_JMP
REM1
PRN_LINE  VERIFY BIT 16 OF BANK 0 IS OFF
REM2
M_VDC     LR6-2159  UR7-2160  =            28ñ2_VDC
LOADF     TRBL_LINE 21
JMP       COND_JMP  WHERE_JMP VALUE_JMP
REM1
PRN_LINE  VERIFY BIT 15 OF BANK 0 IS ON
REM2
M_VDC     LR7-2163  UR7-2164  =            0ñ2_VDC
LOADF     TRBL_LINE 21
JMP       COND_JMP  WHERE_JMP VALUE_JMP
REM1
PRN_LINE  VERIFY BIT 14 OF BANK 0 IS ON
REM2
M_VDC     LR15-2167 UR7-2168  =            0ñ2_VDC
LOADF     TRBL_LINE 21
JMP       COND_JMP  WHERE_JMP VALUE_JMP
REM1
PRN_LINE  VERIFY BIT 13 OF BANK 0 IS OFF
REM2
M_VDC     LR44-1163 UR7-1164  =            28ñ2_VDC
```

ANEXA 1

Secvenţele de testare ale testelor din blocul 2.0-CPU-RAM-CHECKS pentru unitatea AUTOTHROTTLE tip 10-62017-21 pentru avioanele Boeing 737-500 produs de firma Smith Industries Aerospace

```
LOADF     TRBL_LINE 21
JMP       COND_JMP  WHERE_JMP VALUE_JMP
REM1
PRN_LINE    VERIFY BIT 12 OF BANK 0 IS OFF
REM2
M_VDC     LR36-1167 UR7-1168  =           28ñ2_VDC
LOADF     TRBL_LINE 21
JMP       COND_JMP  WHERE_JMP VALUE_JMP
REM1
PRN_LINE    VERIFY BIT 11 OF BANK 0 IS ON
REM2
M_VDC     LR43-1171 UR7-1172  =           0ñ2_VDC
LOADF     TRBL_LINE 21
JMP       COND_JMP  WHERE_JMP VALUE_JMP
REM1
PRN_LINE    VERIFY BIT 10 OF BANK 0 IS ON
REM2
M_VDC     LR26-2001 UR7-2002  =           0ñ2_VDC
LOADF     TRBL_LINE 21
JMP       COND_JMP  WHERE_JMP VALUE_JMP
REM1
PRN_LINE    VERIFY BIT 9 OF BANK 0 IS ON
REM2
M_VDC     LR14-2005 UR7-2006  =           0ñ2_VDC
LOADF     TRBL_LINE 21
JMP       COND_JMP  WHERE_JMP VALUE_JMP
REM1
PRN_LINE    VERIFY BIT 8 OF BANK 0 IS OFF
REM2
M_VDC     LR33-2009 UR7-2010  =           28ñ2_VDC
LOADF     TRBL_LINE 21
JMP       COND_JMP  WHERE_JMP VALUE_JMP
REM1
PRN_LINE    VERIFY BIT 7 OF BANK 0 IS OFF
REM2
M_VDC     LR24-2013 UR7-2014  =           28ñ2_VDC
LOADF     TRBL_LINE 21
JMP       COND_JMP  WHERE_JMP VALUE_JMP
REM1
PRN_LINE    VERIFY BIT 6 OF BANK 0 IS ON
REM2
M_VDC     LR35-2017 UR7-2018  =           0ñ2_VDC
LOADF     TRBL_LINE 21
JMP       COND_JMP  WHERE_JMP VALUE_JMP
REM1
PRN_LINE    VERIFY BIT 5 OF BANK 0 IS OFF
REM2
M_VDC     LR34-2026 UR7-2027  =           28ñ2_VDC
LOADF     TRBL_LINE 21
JMP       COND_JMP  WHERE_JMP VALUE_JMP
REM1
PRN_LINE    VERIFY BIT 4 OF BANK 0 IS ON
REM2
M_VDC     LR25-2030 UR7-2031  =           0ñ2_VDC
LOADF     TRBL_LINE 21
JMP       COND_JMP  WHERE_JMP VALUE_JMP
REM1
PRN_LINE    VERIFY BIT 3 OF BANK 0 IS OFF
REM2
M_VDC     LR32-2034 UR7-2035  =           28ñ2_VDC
LOADF     TRBL_LINE 21
JMP       COND_JMP  WHERE_JMP VALUE_JMP
REM1
PRN_LINE    VERIFY BIT 2 OF BANK 0 IS ON
REM2
M_VDC     UR12-2038 UR7-2039  =           0ñ2_VDC
LOADF     TRBL_LINE 21
JMP       COND_JMP  WHERE_JMP VALUE_JMP
REM1
PRN_LINE    VERIFY BIT 1 (LSB) OF BANK 0 IS OFF
REM2
M_VDC     UR16-2042 UR7-2043  =           28ñ2_VDC
LOADF     TRBL_LINE 21
```

ANEXA 1

Secvenţele de testare ale testelor din blocul 2.0-CPU-RAM-CHECKS pentru unitatea AUTOTHROTTLE tip 10-62017-21 pentru avioanele Boeing 737-500 produs de firma Smith Industries Aerospace

```
JMP      COND_JMP  WHERE_JMP  VALUE_JMP
REM1
PRN_LINE   BANK 1
REM2
CON      LL4-GND
CALL     UNC     GO-AROUND  FIRST-LINE NOT-PRINT
REM1
PRN_LINE   BANK 1, LR10 < -5V IMMEDIATELY AFTER GO-AROUND.
REM2
M_VDC    LR10-2134 UR7-2135  =            <-5_VDC
JMP      SUC     LINE     T2.4_C5
REM1
PRN_LINE   BANK 1 SELECTED, LR10 CHECK FAILED,
PRN_LINE   LR10 GREATER -5_VDC IMMEDIATELY AFTER GO-AROUND.
REM2
CALL     UNC     REMOVE ALL FIRST-LINE NOT-PRINT
BEEP     1
SCREEN_R   MSG-SCR   SHOW    -      -
DISP_LINE  2      2      4      TESTS.HDR
DISP_LINE  7      3      4      TESTS.HDR
DISP_LINE  14     4      25     MSG.HDR
DISP_LINE  2      5      25     MSG.HDR
SCREEN_R   MSG-SCR   ENTER    -      123456
LOADF    TRBL_LINE 21
JMP      UNC     WHERE_JMP  VALUE_JMP
JMP      UNC     SCR      100
LABEL    T2.4_C5
REM1
PRN_LINE   BANK 1, LR10 > +5V WITHIN 20 SECONDS AFTER GO-AROUND.
REM2
ELAPS_TIME 0       0
LABEL    T2.4_L2
CON_MAT    DMM     LR10-2134 UR7-2135
READ_DMM   VDC     MEAS_VOLT
DISCON_MAT DMM     LR10-2134 UR7-2135
ELAPS_TIME 1       TIME
CMPJMP     GREATER  TIME     20      T2.4_C6
CMPJMP     GREATER  MEAS_VOLT 5      T2.4_C7
JMP      UNC     LINE     T2.4_L2
LABEL    T2.4_C6
BETWEEN    5       MEAS_VOLT 50            LOG
REM1
PRN_LINE   BANK 1 SELECTED, LR10 CHECK FAILED,
PRN_LINE   20 SECONDS AFTER GO-AROUND LR10 STILL NOT ABOVE +5_V
REM2
CALL     UNC     REMOVE ALL FIRST-LINE NOT-PRINT
BEEP     1
SCREEN_R   MSG-SCR   SHOW    -      -
DISP_LINE  2      2      4      TESTS.HDR
DISP_LINE  7      3      4      TESTS.HDR
DISP_LINE  14     4      25     MSG.HDR
DISP_LINE  2      5      25     MSG.HDR
SCREEN_R   MSG-SCR   ENTER    -      123456
LOADF    TRBL_LINE 21
JMP      UNC     WHERE_JMP  VALUE_JMP
JMP      UNC     SCR      100
LABEL    T2.4_C7
BETWEEN    5       MEAS_VOLT 50            LOG
REM1
PRN_LINE   VERIFY OUTPUT WORD BIT FOR BANK 1   STEP 248000
PRN_LINE   OFF=28_VDC
PRN_LINE   ON=0_VDC
REM2
REM1
PRN_LINE   VERIFY BIT 20 (MSB) OF BANK 1 IS OFF
REM2
M_VDC    LR45-1130 UR7-1131  =            28ñ2_VDC
LOADF    TRBL_LINE 21
JMP      COND_JMP  WHERE_JMP  VALUE_JMP
REM1
PRN_LINE   VERIFY BIT 19 OF BANK 1 IS ON
REM2
M_VDC    LR46-1134 UR7-1135  =            0ñ2_VDC
```

ANEXA 1

Secvențele de testare ale testelor din blocul 2.0-CPU-RAM-CHECKS pentru unitatea AUTOTHROTTLE tip 10-62017-21 pentru avioanele Boeing 737-500 produs de firma Smith Industries Aerospace

```
LOADF     TRBL_LINE 21
JMP       COND_JMP  WHERE_JMP VALUE_JMP
REM1
PRN_LINE    VERIFY BIT 18 OF BANK 1 IS OFF
REM2
M_VDC     LR23-1138 UR7-1139  =           28ñ2_VDC
LOADF     TRBL_LINE 21
JMP       COND_JMP  WHERE_JMP VALUE_JMP
REM1
PRN_LINE    VERIFY BIT 17 OF BANK 1 IS ON
REM2
M_VDC     LL23-1142 UR7-1143  =           0ñ2_VDC
LOADF     TRBL_LINE 21
JMP       COND_JMP  WHERE_JMP VALUE_JMP
REM1
PRN_LINE    VERIFY BIT 16 OF BANK 1 IS OFF
REM2
M_VDC     LR6-2159  UR7-2160  =           28ñ2_VDC
LOADF     TRBL_LINE 21
JMP       COND_JMP  WHERE_JMP VALUE_JMP
REM1
PRN_LINE    VERIFY BIT 15 OF BANK 1 IS ON
REM2
M_VDC     LR7-2163  UR7-2164  =           0ñ2_VDC
LOADF     TRBL_LINE 21
JMP       COND_JMP  WHERE_JMP VALUE_JMP
REM1
PRN_LINE    VERIFY BIT 14 OF BANK 1 IS ON
REM2
M_VDC     LR15-2167 UR7-2168  =           0ñ2_VDC
LOADF     TRBL_LINE 21
JMP       COND_JMP  WHERE_JMP VALUE_JMP
REM1
PRN_LINE    VERIFY BIT 13 OF BANK 1 IS ON
REM2
M_VDC     LR44-1163 UR7-1164  =           0ñ2_VDC
LOADF     TRBL_LINE 21
JMP       COND_JMP  WHERE_JMP VALUE_JMP
REM1
PRN_LINE    VERIFY BIT 12 OF BANK 1 IS ON
REM2
M_VDC     LR36-1167 UR7-1168  =           0ñ2_VDC
LOADF     TRBL_LINE 21
JMP       COND_JMP  WHERE_JMP VALUE_JMP
REM1
PRN_LINE    VERIFY BIT 11 OF BANK 1 IS ON
REM2
M_VDC     LR43-1171 UR7-1172  =           0ñ2_VDC
LOADF     TRBL_LINE 21
JMP       COND_JMP  WHERE_JMP VALUE_JMP
REM1
PRN_LINE    VERIFY BIT 10 OF BANK 1 IS ON
REM2
M_VDC     LR26-2001 UR7-2002  =           0ñ2_VDC
LOADF     TRBL_LINE 21
JMP       COND_JMP  WHERE_JMP VALUE_JMP
REM1
PRN_LINE    VERIFY BIT 9 OF BANK 1 IS ON
REM2
M_VDC     LR14-2005 UR7-2006  =           0ñ2_VDC
LOADF     TRBL_LINE 21
JMP       COND_JMP  WHERE_JMP VALUE_JMP
REM1
PRN_LINE    VERIFY BIT 8 OF BANK 1 IS OFF
REM2
M_VDC     LR33-2009 UR7-2010  =           28ñ2_VDC
LOADF     TRBL_LINE 21
JMP       COND_JMP  WHERE_JMP VALUE_JMP
REM1
PRN_LINE    VERIFY BIT 7 OF BANK 1 IS ON
REM2
M_VDC     LR24-2013 UR7-2014  =           0ñ2_VDC
LOADF     TRBL_LINE 21
```

ANEXA 1

Secvențele de testare ale testelor din blocul 2.0-CPU-RAM-CHECKS pentru unitatea AUTOTHROTTLE tip 10-62017-21 pentru avioanele Boeing 737-500 produs de firma Smith Industries Aerospace

```
JMP      COND_JMP  WHERE_JMP VALUE_JMP
REM1
PRN_LINE   VERIFY BIT 6 OF BANK 1 IS OFF
REM2
M_VDC     LR35-2017 UR7-2018  =          28ñ2_VDC
LOADF     TRBL_LINE 21
JMP      COND_JMP  WHERE_JMP VALUE_JMP
REM1
PRN_LINE   VERIFY BIT 5 OF BANK 1 IS OFF
REM2
M_VDC     LR34-2026 UR7-2027  =          28ñ2_VDC
LOADF     TRBL_LINE 21
JMP      COND_JMP  WHERE_JMP VALUE_JMP
REM1
PRN_LINE   VERIFY BIT 4 OF BANK 1 IS OFF
REM2
M_VDC     LR25-2030 UR7-2031  =          28ñ2_VDC
LOADF     TRBL_LINE 21
JMP      COND_JMP  WHERE_JMP VALUE_JMP
REM1
PRN_LINE          VERIFY BIT 3 OF BANK 1 IS OFF
REM2
M_VDC             LR32-2034 UR7-2035  =          28ñ2_VDC
LOADF             TRBL_LINE 21
JMP      COND_JMP  WHERE_JMP VALUE_JMP
REM1
PRN_LINE          VERIFY BIT 2 OF BANK 1 IS OFF
REM2
M_VDC             UR12-2038 UR7-2039  =          28ñ2_VDC
LOADF             TRBL_LINE 21
JMP      COND_JMP  WHERE_JMP VALUE_JMP
REM1
PRN_LINE          VERIFY BIT 1 (LSB) OF BANK 1 IS OFF
REM2
M_VDC             UR16-2042 UR7-2043  =          28ñ2_VDC
LOADF             TRBL_LINE 21
JMP      COND_JMP  WHERE_JMP VALUE_JMP
DISCON            LL4-GND
DISCON            UL3-28V
CALL     UNC      REMOVE ALL FIRST-LINE NOT-PRINT
NOP
```

ANEXA 1
Raport de testare pentru blocul 2.0-CPU-RAM-CHECKS pentru unitatea AUTOTHROTTLE tip 10-62017-21 pentru avioanele Boeing 737-500 produs de firma Smith Industries Aerospace

UUT NAME: AUTOTHROTTLE
UUT P/N:
UUT S/N:
TAIL NO:
TPS VERSION: V1.0
OPERATOR NAME:
DATE: 12-17-1997
ATS S/N:

LINE COMMAND PRM 1/3 PRM 2/4 UNIT MINIMUM RESULT MAXIMUM
TEST CHAPTER: 2.1 CPU-USTEP

2.1 CPU MICROSTEP
BLOCK 2.1 CPU MICROSTEP STEP 210000
VERIFY WORDS 220 AND 240 BITS 15 THROUGH 4 ALL ON
STEP 211000
VERIFY BIT 15 OF WORD 220 IS ON .
69 M_VDC LL6-1109 UR7-1110 VOLT 25 26.112421 31-
=
VERIFY BIT 14 OF WORD 220 IS ON
76 M_VDC LR38-1113 UR7-1114 VOLT 25 26.8395443 31-
=
VERIFY BIT 13 OF WORD 220 IS ON .
83 M_VDC LR28-1117 UR7-1118 VOLT 25 26.8419495 31-
=
VERIFY BIT 12 OF WORD 220 IS ON .
90 M_VDC LR27-1126 UR7-1127 VOLT 25 26.8456097 31-
=
VERIFY BIT 11 OF WORD 220 IS ON .
97 M_VDC LR45-1130 UR7-1131 VOLT -3 0.73245 3-
=
VERIFY BIT 10 OF WORD 220 IS ON
104 M_VDC LR46-1134 UR7-1135 VOLT -3 0.7358608 3-
=
VERIFY BIT 9 OF WORD 220 IS ON .
111 M_VDC LR23-1138 UR7-1139 VOLT -3 0.7368449 3-
=
VERIFY BIT 8 OF WORD 220 IS ON .
118 M_VDC LL23-1142 UR7-1143 VOLT -3 0.7322261 3-
=
VERIFY BIT 15 OF WORD 240 IS ON .
125 M_VDC LR6-2159 UR7-2160 VOLT -3 0.738808 3-
=
VERIFY BIT 14 OF WORD 240 IS ON
132 M_VDC LR7-2163 UR7-2164 VOLT -3 0.7364804 3-
=
VERIFY BIT 13 OF WORD 240 IS ON .
139 M_VDC LR15-2167 UR7-2168 VOLT -3 0.7384716 3-
=
VERIFY BIT 12 OF WORD 240 IS ON .
146 M_VDC LR44-1163 UR7-1164 VOLT -3 0.7470198 3-
=
VERIFY BIT 11 OF WORD 240 IS ON
153 M_VDC LR36-1167 UR7-1168 VOLT -3 0.731269 3-
=
VERIFY BIT 10 OF WORD 240 IS ON .
160 M_VDC LR43-1171 UR7-1172 VOLT -3 0.7345287 3-
=
VERIFY BIT 9 OF WORD 240 IS ON .
167 M_VDC LR26-2001 UR7-2002 VOLT -3 0.7373979 3-
=
VERIFY BIT 8 OF WORD 240 IS ON .
174 M_VDC LR14-2005 UR7-2006 VOLT -3 0.732877 3-
=
VERIFY BIT 7 OF WORD 240 IS ON .
181 M_VDC LR33-2009 UR7-2010 VOLT -3 0.7390559 3-
=
VERIFY BIT 6 OF WORD 240 IS ON .
188 M_VDC LR24-2013 UR7-2014 VOLT -3 0.7404972 3-
=
VERIFY BIT 5 OF WORD 240 IS ON .

195 M_VDC LR35-2017 UR7-2018 VOLT -3 0.7293049 3-
=
VERIFY BIT 4 OF WORD 240 IS ON .
202 M_VDC LR34-2026 UR7-2027 VOLT -3 0.733178 3-
=
VERIFY BIT 7 OF WORD 220 IS ON .
209 M_VDC LR25-2030 UR7-2031 VOLT -3 0.7295308 3-
=
VERIFY BIT 6 OF WORD 220 IS ON .
216 M_VDC LR32-2034 UR7-2035 VOLT -3 0.7451702 3-
=
VERIFY BIT 5 OF WORD 220 IS ON .
223 M_VDC UR12-2038 UR7-2039 VOLT -3 0.7312242 3-
=
VERIFY BIT 4 OF WORD 220 IS ON .

230 M_VDC UR16-2042 UR7-2043 VOLT -3 0.7355608 3-
=
CHECK FOR INITIATION OF CPU COUNT,
LR43 AND LR6 SHOULD BOTH BE 28ñ2_VDC. STEP 212220
VERIFY BIT 10 OF WORD 240 IS OFF
271 M_VDC LR43-1171 UR7-1172 VOLT 26 27.8678493 30-
=
VERIFY BIT 15 OF WORD 240 IS OFF .
278 M_VDC LR6-2159 UR7-2160 VOLT 26 27.847353 30-
=
VERIFY WORDS 220 AND 240 BITS 15 THROUGH 4 ALL ON
STEP 213000
VERIFY BIT 15 OF WORD 220 IS ON .
321 M_VDC LL6-1109 UR7-1110 VOLT 25 26.1604214 31-
=
VERIFY BIT 14 OF WORD 220 IS ON .
328 M_VDC LR38-1113 UR7-1114 VOLT 25 26.8843021 31-
=
VERIFY BIT 13 OF WORD 220 IS ON .
335 M_VDC LR28-1117 UR7-1118 VOLT 25 26.8868122 31-
=
VERIFY BIT 12 OF WORD 220 IS ON .
342 M_VDC LR27-1126 UR7-1127 VOLT 25 26.8901596 31-
=
VERIFY BIT 11 OF WORD 220 IS ON .
349 M_VDC LR45-1130 UR7-1131 VOLT -3 0.7287195 3-
=
VERIFY BIT 10 OF WORD 220 IS ON .
356 M_VDC LR46-1134 UR7-1135 VOLT -3 0.7331102 3-
=
VERIFY BIT 9 OF WORD 220 IS ON .
363 M_VDC LR23-1138 UR7-1139 VOLT -3 0.7341257 3-
=
VERIFY BIT 8 OF WORD 220 IS ON .
370 M_VDC LL23-1142 UR7-1143 VOLT -3 0.7286966 3-
=
VERIFY BIT 15 OF WORD 240 IS ON .
377 M_VDC LR6-2159 UR7-2160 VOLT -3 0.7359419 3-
=
VERIFY BIT 14 OF WORD 240 IS ON .
385 M_VDC LR7-2163 UR7-2164 VOLT -3 0.731872 3-
=
VERIFY BIT 13 OF WORD 240 IS ON .
393 M_VDC LR15-2167 UR7-2168 VOLT -3 0.7345392 3-
=
VERIFY BIT 12 OF WORD 240 IS ON .
401 M_VDC LR44-1163 UR7-1164 VOLT -3 0.7431404 3-
=
VERIFY BIT 11 OF WORD 240 IS ON .
409 M_VDC LR36-1167 UR7-1168 VOLT -3 0.7268137 3-
=
VERIFY BIT 10 OF WORD 240 IS ON .
417 M_VDC LR43-1171 UR7-1172 VOLT -3 0.7317303 3-
=
VERIFY BIT 9 OF WORD 240 IS ON .

424 M_VDC LR26-2001 UR7-2002 VOLT -3 0.734612 3-

=
VERIFY BIT 8 OF WORD 240 IS ON .
431 M_VDC LR14-2005 UR7-2006 VOLT -3 0.7285498 3-
=
VERIFY BIT 7 OF WORD 240 IS ON .
438 M_VDC LR33-2009 UR7-2010 VOLT -3 0.7353046 3-
=
VERIFY BIT 6 OF WORD 240 IS ON .
445 M_VDC LR24-2013 UR7-2014 VOLT -3 0.7368209 3-
=
VERIFY BIT 5 OF WORD 240 IS ON .
452 M_VDC LR35-2017 UR7-2018 VOLT -3 0.7250672 3-
=
VERIFY BIT 4 OF WORD 240 IS ON .
459 M_VDC LR34-2026 UR7-2027 VOLT -3 0.7289549 3-
=
VERIFY BIT 7 OF WORD 220 IS ON .
466 M_VDC LR25-2030 UR7-2031 VOLT -3 0.7265377 3-
=
VERIFY BIT 6 OF WORD 220 IS ON .
473 M_VDC LR32-2034 UR7-2035 VOLT -3 0.7436664 3-
=
VERIFY BIT 5 OF WORD 220 IS ON .
480 M_VDC UR12-2038 UR7-2039 VOLT -3 0.7282999 3-
=
VERIFY BIT 4 OF WORD 220 IS ON .
487 M_VDC UR16-2042 UR7-2043 VOLT -3 0.7332811 3-
=
CHECK LR42, LR17 AND LR18
VERIFY LR42 IS 28_VDC
499 M_VDC LR42-2051 UR7-2052 VOLT 25 27.3969402 31-
=
VERIFY LR17 IS 28_VDC
507 M_VDC LR17-2055 UR7-2056 VOLT 25 27.39715 31-
=
VERIFY LR18 IS 28_VDC
515 M_VDC LR18-2059 UR7-2060 VOLT 25 27.3970451 31-
=
POWER OFF

TEST CHAPTER: 2.2 STORE-RAM-UPPER

2.2 STORE RAM UPPER
BLOCK 2.2 STORE RAM UPPER STEP 220000
AFTER SHORT DELAY VERIFY LR6, LR7 AND LR15 ALL AT 0_V
STEP 221100
VERIFY LR6 AT 0_V
74 M_VDC LR6-2159 UR7-2160 VOLT -2 0.7003402 2-
B(1)
VERIFY LR7 AT 0_V
81 M_VDC LR7-2163 UR7-2164 VOLT -2 0.6977606 2-
B(2)
VERIFY LR15 AT 0_V
88 M_VDC LR15-2167 UR7-2168 VOLT -2 0.6993331 2-
B(3)
POWER OFF

TEST CHAPTER: 2.3 STORE-RAM-LOWER

2.3 STORE RAM LOWER
BLOCK 2.3 STORE RAM LOWER STEP 230000
AFTER SHORT DELAY VERIFY LR6, LR7 AND LR15 ALL AT 0_V
STEP 231100
VERIFY LR6 AT 0_V
74 M_VDC LR6-2159 UR7-2160 VOLT -2 0.6984708 2-
B(1)
VERIFY LR7 AT 0_V
81 M_VDC LR7-2163 UR7-2164 VOLT -2 0.6958922 2-
B(2)
VERIFY LR15 AT 0_V
88 M_VDC LR15-2167 UR7-2168 VOLT -2 0.6974835 2-
B(3)
POWER OFF

TEST CHAPTER: 2.4 STORE-PRG-CONTS

ANEXA 1

Raport de testare pentru blocul 2.0-CPU-RAM-CHECKS pentru unitatea AUTOTHROTTLE tip 10-62017-21 pentru avioanele Boeing 737-500 produs de firma Smith Industries Aerospace

```
    2.4 STORE PROGRAMME CONTENTS
BLOCK 2.4 STORE PROGRAMME CONTENTS   STEP 240000
CHECK OUTPUTS FOR BOTH BANKS (0 AND 1)  STEP 246000
BANK 0
BANK 0, LR10 < -5V IMMEDIATELY AFTER GO-AROUND
76    M_VDC  LR10-2134  UR7-2135    VOLT      -50 -10.020056      -5-
             =
BANK 0, LR10 > +5V WITHIN 20 SECONDS AFTER GO-AROUND
124   BETWEEN       5  MEAS_VOLT   LOGIC       5 10.0031834      50-
           50
VERIFY OUTPUT WORD BIT FOR BANK 0    STEP 248000
 OFF=28_VDC
 ON=0_VDC
VERIFY BIT 20 (MSB) OF BANK 0 IS ON
133   M_VDC  LR45-1130  UR7-1131    VOLT      -2  0.707321      2-
             =
VERIFY BIT 19 OF BANK 0 IS OFF
139   M_VDC  LR46-1134  UR7-1135    VOLT      26 27.6986446     30-
             =
VERIFY BIT 18 OF BANK 0 IS OFF
145   M_VDC  LR23-1138  UR7-1139    VOLT      26 27.6984367     30-
             =
VERIFY BIT 17 OF BANK 0 IS ON
151   M_VDC  LL23-1142  UR7-1143    VOLT      -2  0.7068336     2-
             =
VERIFY BIT 16 OF BANK 0 IS OFF
157   M_VDC   LR6-2159  UR7-2160    VOLT      26 27.7206059     30-
             =
VERIFY BIT 15 OF BANK 0 IS ON
163   M_VDC   LR7-2163  UR7-2164    VOLT      -2  0.7076824     2-
             =
VERIFY BIT 14 OF BANK 0 IS ON
169   M_VDC  LR15-2167  UR7-2168    VOLT      -2   0.70933      2-
             =
VERIFY BIT 13 OF BANK 0 IS OFF
175   M_VDC  LR44-1163  UR7-1164    VOLT      26 27.7196655     30-
             =
VERIFY BIT 12 OF BANK 0 IS OFF
181   M_VDC  LR36-1167  UR7-1168    VOLT      26 27.6959267     30-
             =
VERIFY BIT 11 OF BANK 0 IS ON
187   M_VDC  LR43-1171  UR7-1172    VOLT      -2  0.7058786     2-
             =
VERIFY BIT 10 OF BANK 0 IS ON
193   M_VDC  LR26-2001  UR7-2002    VOLT      -2  0.7088832     2-
             =
VERIFY BIT 9 OF BANK 0 IS ON
199   M_VDC  LR14-2005  UR7-2006    VOLT      -2  0.7042644     2-
             =
VERIFY BIT 8 OF BANK 0 IS OFF
205   M_VDC  LR33-2009  UR7-2010    VOLT      26 27.7030373     30-
             =
VERIFY BIT 7 OF BANK 0 IS OFF
211   M_VDC  LR24-2013  UR7-2014    VOLT      26 27.7028275     30-
             =
VERIFY BIT 6 OF BANK 0 IS ON
217   M_VDC  LR35-2017  UR7-2018    VOLT      -2  0.7006401     2-
             =
VERIFY BIT 5 OF BANK 0 IS OFF
223   M_VDC  LR34-2026  UR7-2027    VOLT      26 27.7024097     30-
             =
VERIFY BIT 4 OF BANK 0 IS ON
229   M_VDC  LR25-2030  UR7-2031    VOLT      -2  0.7040603     2-
             =
VERIFY BIT 3 OF BANK 0 IS OFF
235   M_VDC  LR32-2034  UR7-2035    VOLT      26 27.6986446     30-
             =
VERIFY BIT 2 OF BANK 0 IS ON
241   M_VDC  UR12-2038  UR7-2039    VOLT      -2  0.7058349     2-
             =
VERIFY BIT 1 (LSB) OF BANK 0 IS OFF
```

```
247    M_VDC  UR16-2042  UR7-2043    VOLT       26 27.7001095      30-
              =
BANK 1
BANK 1, LR10 < -5V IMMEDIATELY AFTER GO-AROUND
258    M_VDC  LR10-2134  UR7-2135    VOLT       -50 -10.019607     -5-
              =
BANK 1, LR10 > +5V WITHIN 20 SECONDS AFTER GO-AROUND
306  BETWEEN       5 MEAS_VOLT   LOGIC       5 10.0041313      50-
            50
VERIFY OUTPUT WORD BIT FOR BANK 1    STEP 248000
 OFF=28_VDC
 ON=0_VDC
VERIFY BIT 20 (MSB) OF BANK 1 IS OFF
315    M_VDC  LR45-1130  UR7-1131    VOLT       26 27.6804485      30-
              =
VERIFY BIT 19 OF BANK 1 IS ON
321    M_VDC  LR46-1134  UR7-1135    VOLT       -2  0.7144393      2-
              =
VERIFY BIT 18 OF BANK 1 IS OFF
327    M_VDC  LR23-1138  UR7-1139    VOLT       26 27.6796112      30-
              =
VERIFY BIT 17 OF BANK 1 IS ON
333    M_VDC  LL23-1142  UR7-1143    VOLT       -2  0.7064701      2-
              =
VERIFY BIT 16 OF BANK 1 IS OFF
339    M_VDC  LR6-2159   UR7-2160    VOLT       26 27.6978073      30-
              =
VERIFY BIT 15 OF BANK 1 IS ON
345    M_VDC  LR7-2163   UR7-2164    VOLT       -2  0.7093654      2-
              =
VERIFY BIT 14 OF BANK 1 IS ON
351    M_VDC  LR15-2167  UR7-2168    VOLT       -2  0.7118346      2-
              =
VERIFY BIT 13 OF BANK 1 IS ON
357    M_VDC  LR44-1163  UR7-1164    VOLT       -2  0.7202235      2-
              =
VERIFY BIT 12 OF BANK 1 IS ON
363    M_VDC  LR36-1167  UR7-1168    VOLT       -2  0.7037645      2-
              =
VERIFY BIT 11 OF BANK 1 IS ON
369    M_VDC  LR43-1171  UR7-1172    VOLT       -2  0.7075053      2-
              =
VERIFY BIT 10 OF BANK 1 IS ON
375    M_VDC  LR26-2001  UR7-2002    VOLT       -2  0.7105099      2-
              =
VERIFY BIT 9 OF BANK 1 IS ON
381    M_VDC  LR14-2005  UR7-2006    VOLT       -2   0.705637      2-
              =
VERIFY BIT 8 OF BANK 1 IS OFF
387    M_VDC  LR33-2009  UR7-2010    VOLT       26 27.6846313      30-
              =
VERIFY BIT 7 OF BANK 1 IS ON
393    M_VDC  LR24-2013  UR7-2014    VOLT       -2  0.7141612      2-
              =
VERIFY BIT 6 OF BANK 1 IS OFF
399    M_VDC  LR35-2017  UR7-2018    VOLT       26 27.6840038      30-
              =
VERIFY BIT 5 OF BANK 1 IS OFF
405    M_VDC  LR34-2026  UR7-2027    VOLT       26 27.6844215      30-
              =
VERIFY BIT 4 OF BANK 1 IS OFF
411    M_VDC  LR25-2030  UR7-2031    VOLT       26 27.6894417      30-
              =
VERIFY BIT 3 OF BANK 1 IS OFF
417    M_VDC  LR32-2034  UR7-2035    VOLT       26 27.6892338      30-
              =
VERIFY BIT 2 OF BANK 1 IS OFF
423    M_VDC  UR12-2038  UR7-2039    VOLT       26  27.689024      30-
              =
VERIFY BIT 1 (LSB) OF BANK 1 IS OFF
429    M_VDC  UR16-2042  UR7-2043    VOLT       26 27.6894417      30-
              =
     POWER OFF
```

Fragment ATLAS ACP

```
000001 BEGIN, ATLAS PROGRAM 'ACP-ATL'                                    $
C                                                                        $
C                                                                        $
C                                                                        $
C                                                                        $
C 1 0 INTRODUCTION                                                       $
C                                                                        $
C     THIS DOCUMENT CONTAINS COMPLETE TEST                               $
C     SPECIFICATIONS WRITTEN IN ARINC 626 ATLAS                          $
C                                                                        $
C     CMM ATA NUMBER     23-50-42                                        $
C     UUT NAME           AUDIO CONTROL PANEL                             $
C     UUT AVTECH P/N     5701-1-1                                        $
C     UUT BOEING P/N     S222W101-202                                    $
C                                                                        $
C     ORIGINATOR         TIMTEH LTD                                      $
C                                                                        $
C                                                                        $
C     ATLAS SUBSET USED  STANDARD ATLAS FOR MODULAR TEST                 $
C                        (ARINC SPECIFICATION 626-3)                     $
C                                                                        $
C ***********************************************************************$
C                                                                        $
C                REVISION HISTORY                                        $
C                ---------------                                         $
C                                                                        $
C REV   DATE    ENGINEER        CHANGE SUMMARY/REASON                    $
C -- ------ -------  ------------------                                  $
C -   20/12/96              -                                           $
C                                                                        $
C                                                                        $
C ***********************************************************************$
C                                                                        $
C 1.1  PURPOSE                                                           $
C                                                                        $
C     THE PURPOSE OF THIS DOCUMENT IS TO DELINEATE THE REQUIREMENTS      $
C     NEEDED TO PERFORM A COMPLETE ACCEPTANCE TEST PROCEDURE (ATP)       $
C     FOR THE AVTECH 5701-1-1 ACP.                                       $
C                                                                        $
C 1.2  SCOPE                                                             $
C                                                                        $
C     THE TESTS DEFINED BY THIS DOCUMENT SHALL VERIFY THE                $
C     FUNCTIONALITY OF THE LRU                                           $
C                                                                        $
C 1 3  APPLICABLE DOCUMENTS                                              $
C                                                                        $
C     1.3.1 23-50-42     CMM DOCUMENTATION (AVTECH, SEATTLE USA)         $
C                                                                        $
C     1 3 2 TR5701-9     Schematic Tester ACP (AVTECH)                   $
C                                                                        $
C     1.3.3 5701-1500    SOFTWARE REQ. FOR THE AUDIO CONTROL PANEL       $
C                        of the B777 DIGITAL CONTROL AUDIO SYSTEM        $
C                        AVTECH Part Number 5701-1                       $
C                                                                        $
C 1 4  ACRONYMS USED                                                     $
C                                                                        $
C     ACP    AUDIO CONTROL PANEL                                       $
C     APP    APPROACH                                                  $
C     ATT    ATTENDANT                                                 $
C     BITE   BUILT IN TEST EQUIPMENT                                   $
C     C      CENTER                                                    $
C     CCW    COUNTER CLOCK-WISE                                        $
C     CW     CLOCK-WISE                                                $
C     FLT    FLIGHT                                                    $
C     INT    INTERPHONE                                                $
C     L      LEFT                                                      $
C     M      MEGA                                                      $
C     MIC    MICROPHONE                                                $
C     MUX    MULTIPLEXER                                               $
C     NAV    NAVIGATION                                                $
C     NVM    NON-VOLATILE MEMORY                                       $
C     PA     PASSENGER ADRESS                                          $
C     P/N    PART NUMBER                                               $
C     PTT    PUSH-TO-TALK                                              $
C     R      RIGHT                                                     $
C     RF     RADIO FREQENCY                                            $
C     RTN    RETURN                                                    $
C     SCC    STATION CARD CONTROLLER                                   $
C     SEL    SELECT                                                    $
C     S/N    SERIAL NUMBER                                             $
C     SPKR   SPEAKER                                                   $
C     UUT    UNIT UNDER TEST                                           $
C     VBR    VOICE/BOTH/RANGE                                          $
C     VOR    VHF OMNIDIRECTIONAL AND RADIO RANGE                       $
C     VHF    VERY HIGH FREQENCY                                        $
C                                                                      $
C                                                                      $
C 2 0  GENERAL DESCRIPTIONS                                            $
C                                                                      $
C      THE 5701-1-1 AUDIO CONTROL PANEL (ACP) IS THE COCKPIT           $
C     INTERFACE THAT CONTROLS THE AUDIO MANAGEMENT UNIT (AMU).         $
C     THE AMU CONTROLS THE COMMUNICATIONS, NAVIGATION RADIOS,          $
C     PHONE, PUBLIC ADRESS SYSTEM, AUDIO FILTERING, VOLUME OF          $
C     MONITORED CHANNELS AND SELECTION OF THE CABIN INTERPHONE.        $
```

ANEXA 2
Fragment Exemplu Program ATLAS ACP (Audio Control Panel)

```
C                                                                                $
C 2.1  SUBSYSTEM OVERVIEW                                                        $
C                                                                                $
C    SEE 2.0 ABOVE                                                               $
C                                                                                $
C 2.2  TEST DIRECTORY                                                      $
C                                                                                $
C    BLOCK                                                                       $
C    NUMBER                                                                      $
C    ------                                                                      $
C     1.0   PRE-POWER/POWER-UP TESTS                        100000               $
C      1.1  PRE-POWER TESTS                                 110000               $
C      1.2  INITIAL CONDITIONS                              120000               $
C      1.3  POWER-UP TESTS                                  130000               $
C                                                                                $
C     2.0   INDICATOR TEST                                  200000               $
C                                                                                $
C     3.0   CLEANUP NVM                                     300000               $
C      3.1   READING NVM CONTENS                 310000                  $
C      3.2   REMOVIG NON-VALID FAULT CODES                 320000               $
C                                                                                $
C     4.0   BITE TESTS                                      400000               $
C      4.1   CONTINUOS BITE TEST                           410000               $
C      4.2   SHOP INITIATED BITE TEST            420000                  $
C                                                                                $
C     5.0   TEST SWITCHES AND SWITCH INDICATORS  500000                  $
C      5.1  MICROPHONE SWITCH AND INDICATOR TEST 510000                  $
C      5.2  VOLUME CONTROL SWITCH AND INDICATOR TEST       520000               $
C                                                                                $
C     6.0   ARINC 429 TESTS                                 600000               $
C      6.1.1 CALL INDICATORS TEST                611000                  $
C      6.1.2 AMU TO ACP CALL INTERLINK TEST                612000               $
C      6.2  ARINC 429 TO SWITCH INTERFACE TEST             620000               $
C                                                                                $
C     7.0   VOLUME INDICATOR TEST               700000                  $
C                                                                                $
C     8.0   READ THE PART NUMBER AND SERIAL NUMBER 800000               $
C                                                                                $
C     9.0   POWER INTERUPT TEST                             900000               $
C                                                                                $
C                                                                                $
C                                                                                $
C 3.0  GENERAL TEST REQUIREMENTS                                           $
C                                                                                $
C                                                                                $
C 3.1  UUT POWER AND GROUND REQUIREMENTS                                         $
C                                                                                $
C    See Applicable Document 1.3.1, 1.3.2                                        $
C                                                                                $
C 3.2  UUT INPUT/ OUTPUT REQUIREMENTS                                      $
C                                                                                $
C    See Applicable Document 1.3.1, 1.3.2                                        $
C                                                                                $
C 3.3  PIN CONNECTIONS                                                $
C                                                                          $
C    See Applicable Document 1.3.1, 1.3.2                          $
C                                                                          $
C 3.4  ARINC-429 REQUIREMENTS                                              $
C                                                                          $
C    See Applicable Document 1.3.3                                         $
C                                                                                $
C                                                                                $
C 3.5  FUNCTIONAL TEST REQUIREMENTS                                        $
C                                                                                $
C    THE FUNCTIONAL TEST REQUIREMENTS ARE COMPLETELY DEFINED                 $
C    BY THIS ATLAS TEST SPECIFICATION.  ALL EMBEDDED COMMENTS                $
C    WITHIN THE PREAMBLE SHALL BE CONSIDERED PART OF THE TEST                $
C    SPECIFICATION WITH THE EXCEPTION OF DESCRIPTIVE COMMENTS                $
C    SUCH AS HEADER INFORMATION                                              $
C                                                                                $
C                                                                                $
C 4.0  ACCEPTANCE TEST REQUIREMENTS                                        $
C                                                                                $
C    EACH NEW-BUILT FACTORY PRODUCTION AND UNITS RETURNED FOR                $
C    REPAIR SHALL HAVE THE FUNCTIONAL TEST PER PARAGRAPH 4.1                 $
C                                                                                $
C 4.1  FUNCTIONAL TEST                                                     $
C                                                                                $
C    EACH PRODUCTION UNIT OR REPAIRED UNIT SHALL BE FUNCTIONAL               $
C    TESTED PER THE ATLAS TEST SPECIFICATION REQUIREMENTS DESCRIBED          $
C    IN 3.5 ABOVE AND CONTAINED HEREIN.                                      $
C                                                                                $
C 4.2  VISUAL INSPECTION                                                         $
C                                                                                $
C    EACH PRODUCTION UNIT SHALL BE SUBJECTED TO IN-PROCESS VISUAL            $
C    INSPECTIONS OF SUBASSEMBLIES DURING FABRICATION & ASSEMBLY TO           $
C    ENSURE COMPLIANCE WITH DRAWINGS AND PROCESS SPECIFICATIONS              $
C                                                                                $
C                                                                                $
C                                                                                        $
C ************************************************************************       $
C * 001000          REQUIRE STATEMENTS                 *                         $
C ************************************************************************       $
C                                                                                        $
C ------------------------------------------------------------------------       $
C *              TEST EQUIPMENT REQUIREMENTS            *                         $
C ------------------------------------------------------------------------       $
C                                                                                $
C    THE ERROR LIMITS (ERRLMT) GIVEN IN 'REQUIRE' STATEMENTS                 $
C    SHALL APPLY TO ALL STATEMENTS REFERENCING THE VIRTUAL                   $
C    RESOURCE THROUGH THE 'USING' FIELD UNLESS THE ERROR LIMIT               $
```

```
C     IS STATED                                                          $
C ------------------------------------------------------------ $
C                                                                        $
C---------------- POWER SUPPLY ------------------------------- $
C                                                                        $
 001000 REQUIRE, '+28-VDC-POWER-SUPPLY', SOURCE, DC SIGNAL,
       CONTROL,
        VOLTAGE +28.0 V ERRLMT +-0 1 V,
       LIMIT,
        CURRENT MAX 2 A,
       CNX HI LO $
C                                                                        $
 001010 REQUIRE, '5-VAC-POWER-SUPPLY', SOURCE, AC SIGNAL,
       CONTROL,
        VOLTAGE RANGE 0V TO 10 V BY 0.1 V,
        FREQ RANGE 300HZ TO 500 HZ BY 1 HZ,
       LIMIT,
        CURRENT MAX 0 5 A,
       CNX HI LO $
C                                                                        $
C---------------- DMM ------------------------------------------ $
C                                                                        $
 001100 REQUIRE, 'DC-VOLTMETER', SENSOR (VOLTAGE), DC SIGNAL,
       CAPABILITY,
        VOLTAGE RANGE -30.0 V TO 30 0 V,
       CNX HI LO $
C                                                                        $
 001110 REQUIRE, 'AC-VOLTMETER', SENSOR (VOLTAGE), AC SIGNAL,
       CONTROL,
        VOLTAGE RANGE 0.0 V TO 10.0 V,
       CAPABILITY,
        FREQ RANGE  200 HZ TO 1000 HZ,
       CNX HI LO $
C                                                                        $
C                                                                        $
 001130 REQUIRE, 'OHMMETER', SENSOR (RES), IMPEDANCE,
       CONTROL,
        RES RANGE 0 0 OHM TO 1E8 OHM ERRLMT +- 2 PC,
        DMM-MODE SLOW  MEDIUM  FAST,
       CNX HI LO $
C                                                                        $
 001140 REQUIRE, 'DC-AMMETER', SENSOR (CURRENT), DC SIGNAL,
       CONTROL,
        CURRENT RANGE 0.0 A TO 3.0  A BY 1 MA,
                DMM-MODE SLOW  MEDIUM  FAST,
       CNX VIA $
C                                                                        $
 001150 REQUIRE, 'AC-AMMETER', SENSOR (CURRENT), AC SIGNAL,
              CONTROL,
               CURRENT RANGE 0.0 A TO 1.0 A BY 1 MA,
              CNX VIA $
C                                                                        $
C---------------- FGEN ------------------------------------ $
 001200 REQUIRE, 'FUNCTION-GENERATOR',  SOURCE, PULSED DC,
              CONTROL,
                        VOLTAGE RANGE 0V TO 10V,
                        DC-OFFSET RANGE -5V TO 5V,
                        PERIOD RANGE 0.1 SEC TO 10 SEC,
                        PULSE-COUNT RANGE 0 TIMES TO 100 TIMES,
                        CNX HI LO                                        $
C                                                                        $
C---------------------- LOADS ------------------------------- $
C                                                                        $
 001400 REQUIRE, 'RID1', LOAD, IMPEDANCE,
       CONTROL, RES  0 OHM  ERRLMT +- 1 PC,
       CNX HI LO  $
C                                                                        $
      REQUIRE, 'RID2', LOAD, IMPEDANCE,
       CONTROL, RES  300000 OHM  ERRLMT +- 1 PC,
       CNX HI LO  $
C                                                                        $
      REQUIRE, 'RID3', LOAD, IMPEDANCE,
       CONTROL, RES  620 OHM  ERRLMT +- 1 PC,
       CNX HI LO  $
C                                                                        $
      REQUIRE, 'RID4', LOAD, IMPEDANCE,
       CONTROL, RES  270000 OHM  ERRLMT +- 1 PC,
       CNX HI LO  $
C                                                                        $
      REQUIRE, 'R105', LOAD,  IMPEDANCE,
       CONTROL, RES 2200 OHM ERRLMT +- 2 PC,
       CNX HI LO  $
C                                                                        $
      REQUIRE, 'R106', LOAD,  IMPEDANCE,
       CONTROL, RES 2200 OHM ERRLMT +- 2 PC,
       CNX HI LO  $
C                                                                        $
      REQUIRE, 'R107', LOAD,  IMPEDANCE,
       CONTROL, RES 10000 OHM ERRLMT +- 2 PC,
       CNX HI LO  $
C                                                                        $
      REQUIRE, 'R108', LOAD,  IMPEDANCE,
       CONTROL, RES 10000 OHM ERRLMT +- 2 PC,
       CNX HI LO  $
C                                                                        $
      REQUIRE, 'SHORT1',
          LOAD, SHORT, CNX $
C                                                                        $
      REQUIRE, 'SHORT2',
          LOAD, SHORT, CNX $
C                                                                        $
```

ANEXA 2
Fragment Exemplu Program ATLAS ACP (Audio Control Panel)

```
    REQUIRE, 'SHORT3',
        LOAD, SHORT, CNX $
C$
 001500 REQUIRE, 'GROUND', LOAD, EARTH, CNX $
C$
C$
C ********************************************************************** $
C *              DECLARE STATEMENTS                    * $
C ********************************************************************** $
C$
C$
 010000 DECLARE, VARIABLE, 'LOOP', 'CNT' IS INTEGER $
C$
    DECLARE, VARIABLE, 'AC-VOLT'  IS DECIMAL $
C$
C---    DECLARE, VARIABLE, 'WORD-RATE' IS DECIMAL INITIAL = 700 $
C$
    DECLARE, VARIABLE, 'WRDIN'    IS ARRAY (1 THRU 256)
                  OF STRING(32) OF BIT
                INITIAL = (256 OF X'0') $
C$
    DECLARE, VARIABLE, 'RCV-WORD'   IS ARRAY (1 THRU 256)
                        OF STRING(32) OF BIT
                          INITIAL = (256 OF X'0') $
C$
    DECLARE, VARIABLE, 'SEND-WORD'  IS ARRAY(1 THRU 2)
                        OF STRING(32) OF BIT $
C$
    DECLARE, VARIABLE, 'FAULT-WORD'  IS ARRAY (1 THRU 99)
                        OF STRING(32) OF BIT $
C$
            DECLARE, VARIABLE, 'DIG-VOL' IS STRING(6) OF BIT $
C$
    DECLARE, VARIABLE, 'LABEL', 'OCT-LABEL', 'EXP-LABEL' IS STRING(32) OF BIT  $

    DECLARE, VARIABLE, 'WORDS'      IS ARRAY (1 THRU 100)
                        OF STRING(32) OF BIT
                          INITIAL = (100 OF X'0') $

    DECLARE, VARIABLE, 'WRDOUT', 'WRD'     IS STRING(32) OF BIT $
    DECLARE, VARIABLE, 'EXP', 'EXP-WORD'   IS STRING(32) OF BIT $
    DECLARE, VARIABLE, 'WORD', 'SLICE', 'SSM'IS STRING(32) OF BIT $
    DECLARE, VARIABLE, 'EXP-BIT', 'BIT'    IS STRING(32) OF BIT $
    DECLARE, VARIABLE, 'WORD-33'        IS STRING(33) OF BIT $
C$
    DECLARE, VARIABLE, 'CHAR', 'CHR'  IS STRING (1) OF CHAR $

    DECLARE, VARIABLE, 'CHR-1'      IS STRING (1) OF CHAR $
    DECLARE, VARIABLE, 'CHR-2'      IS STRING (2) OF CHAR $
    DECLARE, VARIABLE, 'CHR-3'      IS STRING (3) OF CHAR $
    DECLARE, VARIABLE, 'CHR-4'      IS STRING (4) OF CHAR $
    DECLARE, VARIABLE, 'CHR-5'      IS STRING (5) OF CHAR $
    DECLARE, VARIABLE, 'CHR-6'      IS STRING (6) OF CHAR $
    DECLARE, VARIABLE, 'CHR-7'      IS STRING (7) OF CHAR $
    DECLARE, VARIABLE, 'CHR-8'      IS STRING (8) OF CHAR $
    DECLARE, VARIABLE, 'CHR-9'      IS STRING (9) OF CHAR $
    DECLARE, VARIABLE, 'CHR-10'     IS STRING (10) OF CHAR $
    DECLARE, VARIABLE, 'CHR-11'     IS STRING (11) OF CHAR $
    DECLARE, VARIABLE, 'CHR-12'     IS STRING (12) OF CHAR $
    DECLARE, VARIABLE, 'CHR-13'     IS STRING (13) OF CHAR $
    DECLARE, VARIABLE, 'CHR-14'     IS STRING (14) OF CHAR $

    DECLARE, VARIABLE, 'S-N', 'EXP S-N'  IS STRING (6) OF CHAR $

    DECLARE, VARIABLE, 'P-N', 'EXP P-N'  IS STRING (15) OF CHAR $
C$
    DECLARE, VARIABLE, 'TYPE'         IS STRING (15) OF CHAR $
C$
            DECLARE, VARIABLE, 'BIT-MSG', 'DSP-MSG' IS STRING(80) OF CHAR $
C$
    DECLARE, VARIABLE, 'ARINC-ERR'      IS BOOLEAN INITIAL = FALSE $
C$
            DECLARE, VARIABLE, 'POS', 'VOL' IS INTEGER $
C$
    DECLARE, VARIABLE, 'I', 'SIZE', 'NBIT', 'TRY',
              'BIT-NO', 'NBWRD', 'EXPNB'  IS INTEGER $
C$
    DECLARE, VARIABLE, 'INDEX'  IS INTEGER $
C$
C----------------------------------------------------------------$
C$
    DECLARE, VARIABLE, 'SCC-ACP-CT-1'      IS STRING(32) OF BIT
                          INITIAL = (X'80000003') $
C$
    DECLARE, VARIABLE, 'SCC-ACP-CT-2'      IS STRING(32) OF BIT
                          INITIAL = (X'00000083') $
C$
    DECLARE, VARIABLE, 'SCC-ACP-PN-SN'     IS STRING(32) OF BIT
                          INITIAL = (X'80002083') $
C$
    DECLARE, VARIABLE, 'SCC-ACP-IBITE-CT'  IS STRING(32) OF BIT
                          INITIAL = (X'094E5043') $
C$
    DECLARE, VARIABLE, 'SHOP-MODE-ENABLE'  IS STRING(32) OF BIT
                          INITIAL = (X'95E194A3') $
C$
    DECLARE, VARIABLE, 'SH-ACP-SD-FA-DAT'  IS STRING(32) OF BIT
                          INITIAL = (X'8509E0E3') $
C$
    DECLARE, VARIABLE, 'SH-ACP-CL-FA-REC'  IS STRING(32) OF BIT
                          INITIAL = (X'80C18C93') $
```

```
C$
    DECLARE, VARIABLE, 'SH-ACP-IBITE-CT'   IS STRING(32) OF BIT
                          INITIAL = (X'9CA56453') $
C$
    DECLARE, VARIABLE, 'ACP-SH-BITE-ST'    IS ARRAY (1 THRU 4)
                          OF STRING(32) OF BIT $
C$
    DECLARE, VARIABLE, 'SH-ACP-PN-SN'    IS ARRAY (1 THRU 8)
                          OF STRING(32) OF BIT $
    DECLARE, VARIABLE, 'FAULT-RECORD' IS ARRAY (1 THRU 100)
    OF STRING(32) OF BIT $
C$
    DECLARE, VARIABLE, 'SERIAL-NUMBER' IS ARRAY (1 THRU 8)
    OF STRING(32) OF BIT $
C$
    DECLARE, VARIABLE, 'ACP-STATUS' IS ARRAY (1 THRU 8)
    OF STRING(32) OF BIT $
C$
    DECLARE, VARIABLE, 'AMU-STATUS' IS ARRAY (1 THRU 2)
                      OF STRING(32) OF BIT  $
C$
    DECLARE, VARIABLE, 'MIC-LABEL'  IS STRING(32) OF BIT
                          INITIAL = O'270' $
    DECLARE, VARIABLE, 'CALL-LABEL' IS STRING(32) OF BIT
                          INITIAL = O'272' $
    DECLARE, VARIABLE, 'VOL-LABEL'  IS STRING(32) OF BIT
                          INITIAL = O'000' $
C$
C-------------------------------------------------------------------$
C$
    DECLARE, CONSTANT, 'SPACE-BEFORE' IS
    C'\LF\\LF\\LF\\LF\\LF\\LF\\LF\\LF\\LF\\LF\\LF\\LF\\LF\\LF\\LF\
**********************************************************************\LF\'$
C$
    DECLARE, CONSTANT, 'SPACE-AFTER' IS
            C'\LF\**********************************************************************\LF\'$
C$
    DECLARE, VARIABLE, 'FAILURE-COUNTER' IS INTEGER INITIAL = 0  $
C$
    DECLARE, VARIABLE, 'LOG' IS BOOLEAN INITIAL = TRUE $
C$
    DECLARE, VARIABLE, 'DSPL' IS BOOLEAN INITIAL = TRUE $
C$
    DECLARE, VARIABLE, 'LOG-NAME' IS STRING(14) OF CHAR INITIAL = C'ACP-REP' $
C$
    DECLARE, VARIABLE, 'LOG-FILE', 'ACP-FILE' IS FILE OF TEXT $
C$
    DECLARE, VARIABLE, 'UUT-NAME' IS STRING(10) OF CHAR INITIAL = C'ACP' $
C$
    DECLARE, VARIABLE, 'UUT-FULL-NAME' IS STRING(80) OF CHAR
     INITIAL = C' ACP P/N 571-1-1' $
C$
    DECLARE, VARIABLE, 'SWB' IS STRING(1) OF CHAR $
C$
    DECLARE, VARIABLE, 'A429_LABEL' IS STRING(9) OF BIT  $
C$
C ********************************************************************** $
C *            ESTABLISH PROTOCOL STATEMENTS                 * $
C ********************************************************************** $
C$
 020000 ESTABLISH, BUS-PROTOCOL 'XMT-BUS',
        SPEC 'ARINC-429',
        TEST-EQUIP-ROLE MASTER,MONITOR,SLAVE,
        TEST-EQUIP-MONITOR DATA,
      BUS-MODE TALKER-LISTENER,
        TALKER TEST-EQUIP,
        LISTENER UUT,
        DATA (STRING(32) OF BIT),
        STANDARD PRIMARY BUS,
    CNX TRUE COMPL $
C$
C$
 020100 ESTABLISH, BUS-PROTOCOL 'RCV-BUS',
        SPEC 'ARINC-429',
        TEST-EQUIP-ROLE MONITOR,
        TEST-EQUIP-MONITOR DATA,
      BUS-MODE TALKER-LISTENER,
        TALKER UUT,
        LISTENER TEST-EQUIP,
        DATA (STRING(32) OF BIT),
        STANDARD PRIMARY BUS,
    CNX TRUE COMPL $
C$
 020110 ESTABLISH, BUS-PROTOCOL 'RCV-BUS-L',
        SPEC 'ARINC-429',
        TEST-EQUIP-ROLE MONITOR,
        TEST-EQUIP-MONITOR DATA,
      BUS-MODE TALKER-LISTENER,
        TALKER UUT,
        LISTENER TEST-EQUIP,
        COMMAND (STRING(8) OF BIT),
        DATA (STRING(32) OF BIT),
        STANDARD PRIMARY BUS,
    CNX TRUE COMPL $
C$
C$
C ********************************************************************** $
C *              IDENTIFY STATEMENTS                 * $
C ********************************************************************** $
C$
C$
```

```
C ********************************************************************** $
C *            DEFINE EXCHANGE STATEMENTS                 * $
C ********************************************************************** $
C$
 031000 DEFINE, 'XMT', EXCHANGE CONFIGURATION
     'XMT-BUS' $
 031100 DEFINE, 'RCV', EXCHANGE CONFIGURATION
     'RCV-BUS' $
 031110 DEFINE, 'RCV-L', EXCHANGE CONFIGURATION
     'RCV-BUS-L' $
C$
C$
C$
 031500 DEFINE, 'SEND-W', EXCHANGE, PROTOCOL 'XMT-BUS',
        BUS-MODE TALKER-LISTENER,
        TALKER TEST-EQUIP,
        LISTENER UUT,
        DATA 'SEND-WORD'(1 THRU 2) $
C$
 031510 DEFINE, 'SEND-1W', EXCHANGE, PROTOCOL 'XMT-BUS',
        BUS-MODE TALKER-LISTENER,
        TALKER TEST-EQUIP,
        LISTENER UUT,
        DATA 'SEND-WORD'(1) $
C$
 031520 DEFINE, 'SEND-PN', EXCHANGE, PROTOCOL 'XMT-BUS',
        BUS-MODE TALKER-LISTENER,
        TALKER TEST-EQUIP,
        LISTENER UUT,
        DATA 'SH-ACP-PN-SN'(1 THRU 8) $
C$
 031600 DEFINE, 'RCV-W', EXCHANGE, PROTOCOL 'RCV-BUS',
        BUS-MODE TALKER-LISTENER,
        TALKER UUT,
        LISTENER TEST-EQUIP,
        DATA 'RCV-WORD'(1 THRU 32)  $
C$
 031610 DEFINE, 'RCV-WL', EXCHANGE, PROTOCOL 'RCV-BUS-L',
        BUS-MODE TALKER-LISTENER,
        TALKER UUT,
        LISTENER TEST-EQUIP,
        DATA 'RCV-WORD'(1 THRU 100),
        COMMAND  'EXP-LABEL'      $
C$
C$
C ********************************************************************** $
C * 040000       DEFINE PROCEDURE STATEMENTS              * $
C ********************************************************************** $
C                                                        $
C ********************************************************************** $
C * PROCEDURE 'CLEAR-SCREEN'                             * $
C *-------------------------------------------------------* $
C ********************************************************************** $
C                                                        $
 040000 DEFINE, 'CLEAR-SCREEN', PROCEDURE $
C$
     OUTPUT, TEXT, FROM C'\LF\\LF\\LF\\LF\\LF\\LF\\LF\\LF\\LF\\LF\' $
     OUTPUT, TEXT, FROM C'\LF\\LF\\LF\\LF\\LF\\LF\\LF\\LF\\LF\\LF\' $
     OUTPUT, TEXT, FROM C'\LF\\LF\\LF\\LF\\LF\\LF\\LF\\LF\\LF\\LF\' $
     OUTPUT, TEXT, FROM C'\LF\\LF\\LF\\LF\\LF\\LF\\LF\\LF\\LF\\LF\' $
C$
 040999 END,   'CLEAR-SCREEN'                            $


.
C$
C ********************************************************************** $
C * PROCEDURE 'DATE-TIME'                                * $
C *-------------------------------------------------------* $
C * PURPOSE: READS THE DATE USING THE ATLAS DATE FUNCTION AND     * $
C *       RETURNS THE DATE IN THE FORM YY/MM/DD AND RETURNS     * $
C *       THE TIME IN THE FORM HH:MM:SS                 * $
C *-------------------------------------------------------* $
C * GLOBAL VARIABLES USED:                               * $
C *   -VAR-       -TYPE-          -FUNCTION-          * $
C *   'D'         STRING(15) OF CHAR   RETURNS THE DATE     * $
C *   'T'         STRING(15) OF CHAR   RETURNS THE TIME     * $
C *                                                    * $
C * PROCEDURES: NONE                                   * $
C *                                                    * $
C ********************************************************************** $
C$
 045000 DEFINE, 'DATE-TIME',
      PROCEDURE RESULT( 'D', 'T' IS STRING(15) OF CHAR)  $
     DECLARE, VARIABLE, 'TODAY' IS STRING(15) OF CHAR    $
     DECLARE, VARIABLE, 'D1', 'T1' IS STRING(10) OF CHAR  $
     DECLARE, VARIABLE, 'D2', 'T2' IS STRING(12) OF CHAR  $
     CALCULATE,  'TODAY' = DATE $
     CALCULATE,  'D1' = COPY('TODAY', 1, 6 ),
        'D2' = INSERT(C'/', 'D1', 2),
        'D'  = INSERT(C'/', 'D2', 5)  $
     CALCULATE,  'T1' = COPY('TODAY', 7, 6 ),
        'T2' = INSERT(C':', 'T1', 2),
        'T'  = INSERT(C':', 'T2', 5)  $
     END, 'DATE-TIME' $
C$
C ********************************************************************** $
C **   PROCEDURE 'WAIT-MI'                              ** $
C ********************************************************************** $
C * PURPOSE - WAIT FOR MANUAL INTERVENTION                * $
C ********************************************************************** $
```

```
C * LABEL DEFINITIONS-                                    * $
C * ----LABEL----    ----TYPE----   ------FUNCTION------     * $
C ************************************************************** $
C$
C$
  05 DEFINE, 'WAIT-MI', PROCEDURE $
C$
    OUTPUT, TEXT, FROM
            C'Manual Intervention to continue  \LF\' $
    ENABLE, MANUAL INTERVENTION $
    WAIT FOR, MANUAL INTERVENTION $
    DISABLE, MANUAL INTERVENTION $
    END, 'WAIT-MI'  $
C$
C$
C ************************************************************** $
C * PROCEDURE: 'ENABLE-IO'                                  * $
C *-------------------------------------------------------* $
C * PURPOSE: ENABLES SELECTED PERIPHPERAL DEVICES          * $
C *-------------------------------------------------------* $
C * GLOBAL VARIABLES USED:                                  * $
C *   -VAR-        -TYPE-        -FUNCTON-          * $
C *   LOG          BOOLEAN       LOG FLAG           * $
C *   LOG-NAME     STRING        FILE NAME          * $
C *   LOG-FILE     FILE          LOG FILE           * $
C *                                                 * $
C * PROCEDURES: NONE                                * $
C *                                                 * $
C ************************************************************** $
C$
  10 DEFINE, 'ENABLE-IO', PROCEDURE $
C$
    IF, 'LOG', THEN $
     ENABLE, OUTPUT TO NEW 'LOG-NAME', VIA 'LOG-FILE' $
    END, IF $
C$
    END, 'ENABLE-IO' $
C $



C$
C ************************************************************** $
C * PROCEDURE: 'CHECK-TUA'                                  * $
C *-------------------------------------------------------* $
C * PURPOSE: Identifies Test Unit Adapter and checks that all   * $
C *       connectors are plugged in                         * $
C *-------------------------------------------------------* $
C * GLOBAL VARIABLES USED:                                  * $
C *   -VAR-        -TYPE-        -FUNCTION-         * $
C *                                                 * $
C * PROCEDURES: NONE                                * $
C *                                                 * $
C ************************************************************** $
C$
  40 DEFINE, 'CHECK-TUA', PROCEDURE $
C$
    DECLARE, VARIABLE, 'CNT', 'FAIL' IS INTEGER $
C$
    CALCULATE, 'CNT' = 0, 'FAIL' = 4 $
    WHILE, ('CNT' LT 10) AND ('FAIL' NE 0), THEN $
     CALCULATE, 'CNT' = 'CNT' +1 $
     CALCULATE, 'FAIL' = 0 $
C ------------------- VERIFY IDR1 ------------------- $
    APPLY, IMPEDANCE USING 'RID1',
     RES  0 OHM  ERRLMT +- 1 PC,
    CNX HI  ID-HI
      LO  ID-LO $
C$
    VERIFY, (RES), IMPEDANCE USING 'OHMMETER',
                        UL 10 OHM  LL 0 OHM,
                        RES RANGE 0 OHM TO 100 OHM,
                        CNX HI ID-HI
        LO ID-LO $
C$
    REMOVE, IMPEDANCE USING 'RID1',
     RES  0 OHM  ERRLMT +- 1 PC,
    CNX HI  ID-HI
      LO  ID-LO $
C$
     IF, NOGO, THEN $
       CALCULATE, 'FAIL' = 'FAIL' + 1 $
       OUTPUT, TEXT, FROM
       'SPACE-BEFORE',
       C'The TUA is not correctly installed or it is not the TUA\LF\',
       C'for testing AUDIO CONTROL PANEL P/N 5701-1-1\LF\\LF\',
       C'Please, if you wish to perform the test,\LF\',
       C'install properly the required TUA\LF\\LF\',
              C'*************************************************\LF\',
              C'*        PRESS "T" TO CONTINUE, "F" TO QUIT     *\LF\',
              C'*************************************************\LF\' $
       INPUT, GO-NOGO $
       IF, NOGO, THEN $
         PERFORM, 'ABORT-TEST' $
       END, IF $
     END, IF $
C$
C ------------------- VERIFY IDR2 ------------------- $
    APPLY, IMPEDANCE USING 'RID2',
     RES  300000 OHM  ERRLMT +- 1 PC,
    CNX HI  ID-HI
```

```
      LO  ID-LO $
C$
    VERIFY, (RES), IMPEDANCE USING 'OHMMETER',
                        UL 303000 OHM  LL 297000 OHM,
                        RES RANGE 0 OHM TO 1000000 OHM,
                        CNX HI ID-HI
        LO ID-LO $
C$
    REMOVE, IMPEDANCE USING 'RID2',
     RES  300000 OHM  ERRLMT +- 1 PC,
    CNX HI  ID-HI
      LO  ID-LO $
C$
C$
     IF, NOGO, THEN $
        CALCULATE, 'FAIL' = 'FAIL' + 1 $
        OUTPUT, TEXT, FROM
        'SPACE-BEFORE',
        C'The TUA is not correctly installed or it is not the TUA\LF\',
        C'for testing AUDIO CONTROL PANEL P/N 5701-1-1\LF\\LF\',
        C'Please, if you wish to perform the test,\LF\',
        C'install properly the required TUA\LF\\LF\',
                C'*****************************************************\LF\',
                C'*       PRESS "T" TO CONTINUE, "F" TO QUIT          *\LF\',
                C'*****************************************************\LF\' $
        INPUT, GO-NOGO  $
        IF, NOGO, THEN  $
          PERFORM, 'ABORT-TEST' $
        END, IF $
      END, IF $
C$
C ------------------ VERIFY IDR3 ------------------------ $
    APPLY, IMPEDANCE USING 'RID3',
     RES  620 OHM  ERRLMT +- 1 PC,
    CNX HI  ID-HI
      LO  ID-LO $
C$
    VERIFY, (RES), IMPEDANCE USING 'OHMMETER',
                        UL 626 OHM  LL 614 OHM,
                        RES RANGE 0 OHM TO 1000 OHM,
                        CNX HI ID-HI
        LO ID-LO $
C$
    REMOVE, IMPEDANCE USING 'RID3',
     RES  620 OHM  ERRLMT +- 1 PC,
    CNX HI  ID-HI
      LO  ID-LO $
C$
     IF, NOGO, THEN $
        CALCULATE, 'FAIL' = 'FAIL' + 1 $
        OUTPUT, TEXT, FROM
        'SPACE-BEFORE',
        C'The TUA is not correctly installed or it is not the TUA\LF\',
        C'for testing AUDIO CONTROL PANEL P/N 5701-1-1\LF\\LF\',
        C'Please, if you wish to perform the test,\LF\',
        C'install properly the required TUA\LF\\LF\',
                C'*****************************************************\LF\',
                C'*       PRESS "T" TO CONTINUE, "F" TO QUIT          *\LF\',
                C'*****************************************************\LF\' $
        INPUT, GO-NOGO  $
        IF, NOGO, THEN  $
          PERFORM, 'ABORT-TEST' $
        END, IF $
      END, IF $
C$
C ------------------ VERIFY IDR4 ------------------------ $
    APPLY, IMPEDANCE USING 'RID4',
     RES  270000 OHM  ERRLMT +- 1 PC,
    CNX HI  ID-HI
      LO  ID-LO $
C$
    VERIFY, (RES), IMPEDANCE USING 'OHMMETER',
       UL 277100 OHM  LL 262900 OHM,
                        RES RANGE 0 OHM TO 1000000 OHM,
                        CNX HI ID-HI
        LO ID-LO $
C$
    REMOVE, IMPEDANCE USING 'RID4',
     RES  270000 OHM  ERRLMT +- 1 PC,
    CNX HI  ID-HI
      LO  ID-LO $
C$
     IF, NOGO, THEN $
        CALCULATE, 'FAIL' = 'FAIL' + 1 $
        OUTPUT, TEXT, FROM
        'SPACE-BEFORE',
        C'The TUA is not correctly installed or it is not the TUA\LF\',
        C'for testing AUDIO CONTROL PANEL P/N 5701-1-1\LF\\LF\',
        C'Please, if you wish to perform the test,\LF\',
        C'install properly the required TUA\LF\\LF\',
                C'*****************************************************\LF\',
                C'*       PRESS "T" TO CONTINUE, "F" TO QUIT          *\LF\',
                C'*****************************************************\LF\' $
        INPUT, GO-NOGO  $
        IF, NOGO, THEN  $
          PERFORM, 'ABORT-TEST' $
        END, IF $
      END, IF $
C$
    END, WHILE $
C$
```

```
    IF, 'CNT' GE 10, THEN $
      OUTPUT, TEXT, FROM
        'SPACE-BEFORE',
        C'The TUA is not correctly installed or it is not the TUA\LF\',
        C'for testing AUDIO CONTROL PANEL P/N 5701-1-1\LF\',
        'SPACE-AFTER' $
        PERFORM, 'WAIT-MI' $
        PERFORM, 'ABORT-TEST' $
    END, IF $
C-------------------------------------------------------------------$
    END, 'CHECK-TUA' $
C$
C-------------------------------------------------------------------$


C ********************************************************************* $
C * PROCEDURE 'START-PROG'                                            * $
C *-----------------------------------------------------------------* $
C * PURPOSE: Establishes initial conditions for the program         * $
C *       Sets global variables, sets start of run time             * $
C *-----------------------------------------------------------------* $
C * GLOBAL VARIABLES USED:                                          * $
C *   -VAR-        -TYPE-         -FUNCTION-                        * $
C *                                                                 * $
C * PROCEDURES: 'SET-RUN-OPTIONS'                                   * $
C *         'ENABLE-IO'                                             * $
C *                                                                 * $
C ********************************************************************* $
C$
   45 DEFINE, 'START-PROG', PROCEDURE $
C$
    CALCULATE, 'FAILURE-COUNTER' = 0 $
C$
    PERFORM, 'ENABLE-IO'    $
C$
    PERFORM, 'PROG-HEADER'    $
C$
    PERFORM, 'CHECK-TUA'    $
C$
          OUTPUT, TEXT, FROM
    'SPACE-BEFORE',
    C'OPERATOR ACTION:\LF\\LF\',
    C'Do you wish that results will be displayed during test?\LF\',
    C'Press [t] if yes, [f] if not. \LF\',
    'SPACE-AFTER' $
    ENABLE, MANUAL INTERVENTION $
        INPUT, GO-NOGO  $
    DISABLE, MANUAL INTERVENTION $
        IF, NOGO, THEN  $
          CALCULATE, 'DSPL' = FALSE $
        END, IF $
    END, 'START-PROG'       $


C$
C$
C ********************************************************************* $
C *   PROCEDURE: 'RECEIVE-429-WORD'                                 * $
C *-----------------------------------------------------------------* $
C * PURPOSE:    PERFORMS RECEPTION & DETECTION OF AN EXPECTED WORD * $
C *-----------------------------------------------------------------* $
C * PARAM-IN :  'EXP-LABEL'    EXPECTED LABEL OF WORD             * $
C ********************************************************************* $
C *                                                                 * $
C * PARAM-OUT : 'OCT-LABEL'    OCTAL LABEL OF FINDED WORDS        * $
C *         'WORDS'('INDEX') ARRAY OF FINDED WORDS                * $
C *                                                                 * $
C * LOCAL-USE :  NONE                                               * $
C *                                                                 * $
C * PROCEDURES: 'REVERSE-LABEL'                                     * $
C *                                                                 * $
C ********************************************************************* $
C$
    DEFINE, 'RECEIVE-429-WORD', PROCEDURE ('EXP-LABEL' IS STRING (32) OF BIT) $
C$
C-------------------------------------------------------------------- $
C$
    PERFORM, 'REVERSE-LABEL' ('EXP-LABEL') $
C$
    FOR, 'I' = 1 THRU 32, THEN $
    CALCULATE, 'RCV-WORD'('I') =  X'00000000' $
    END, FOR $
C$
    CALCULATE, 'INDEX' = 1 $
    CALCULATE, 'ARINC-ERR' = 'TRUE'$
    CALCULATE, 'BIT-MSG' = C' '$
C$
C-------------------------------------------------------------------- $
C$
    ENABLE, EXCHANGE-CONFIGURATION USING 'RCV',
      PROTOCOL 'RCV-BUS',
    CNX TRUE J1-20 COMPL J1-21 $
C$
    WAIT FOR, TIME 200 MSEC $
C$
    DO, EXCHANGE USING 'RCV', TEST-EQUIP-ROLE MONITOR,
      (EXCHANGE 'RCV-W',
       TEST-EQUIP-MONITOR DATA 'RCV-WORD'(1)),
    PROCEED, MAX-TIME 40 MSEC $
```

```
C$
    WAIT FOR, TIME 50 MSEC $
C$
    DO, EXCHANGE USING 'RCV', TEST-EQUIP-ROLE MONITOR,
      (EXCHANGE 'RCV-W',
       TEST-EQUIP-MONITOR DATA 'RCV-WORD'(1 THRU 32)),
    PROCEED, MAX-TIME 50 MSEC $
C$
C---------------------------------------------------------------- $
C$
    FOR, 'I' = 1 THRU 32, THEN $
C$
    CALCULATE, 'WRD' = 'RCV-WORD'('I') AND X'000000FF' $
C$
      IF,('LABEL' EQ 'WRD'), THEN  $
C$
      CALCULATE, 'ARINC-ERR' = 'FALSE'$
      CALCULATE, 'BIT-MSG' = C'FIND WORD'$
C$
    CALCULATE, 'WRDOUT'   = 'RCV-WORD'('I') $
    CALCULATE, 'WORDS'(1) = 'RCV-WORD'('I') $
C$
    LEAVE, FOR $
C$
      ELSE $
C$
        IF, 'I' LT 32, THEN $
C$
    RESUME, FOR $
C$
        ELSE $
C$
        CALCULATE, 'BIT-MSG' = C'NOT FIND LABEL'$
        CALCULATE, 'ARINC-ERR' = 'TRUE'$
C$
        END, IF $
      END, IF $
    END, FOR $
C$
   END, 'RECEIVE-429-WORD' $


C$
C$
C ********************************************************************** $
C *              END OF PREAMBLE                       * $
C ********************************************************************** $
C$
C ********************************************************************** $
C               MAIN PROCEDURE                         $
C ********************************************************************** $
C$
E090000 COMMENCE, MAIN PROCEDURE $
C$
    PERFORM, 'START-PROG'    $
C$
C$
C ********************************************************************** $
C *     BLOCK 1.0  PRE-POWER/POWER-UP TESTS              * $
C ********************************************************************** $
C$
C$
E100000 BEGIN, BLOCK, '1.0 PRE-POWER/POWER-UP TESTS'  $
C$
C$
C ********************************************************************** $
C *     BLOCK 1.1  PRE-POWER TESTS                       * $
C ********************************************************************** $
C$
    OUTPUT, TEXT, FROM
      'SPACE-BEFORE',
      C'\LF\',
      C' *** BLOCK 1.0  PRE-POWER/POWER-UP TESTS ***',
      C'\LF\',
      'SPACE-AFTER' $
C$
    OUTPUT, TEXT, FROM
      'SPACE-BEFORE',
      C'\LF\',
      C' — BLOCK 1.1  PRE-POWER TESTS —',
      C'\LF\',
      'SPACE-AFTER' $
C$
    OUTPUT, TEXT TO 'LOG-FILE', FROM
      C'\LF\\LF\',
      C' *** BLOCK 1.0 PRE-POWER/POWER-UP TESTS ***',
      C'\LF\\LF\',
      C' — BLOCK 1.1  PRE-POWER TESTS —\LF\\LF\' $
C$
C$
C —— Verify if not short on +28VDC pins J1-10, J1-16 refer to EARTH —— $
C$
 110020 MEASURE, (RES INTO 'MEASUREMENT'), IMPEDANCE USING 'OHMMETER',
    RES RANGE 0 OHM TO 30000 OHM,
    CNX HI  J1-10
      LO  EARTH $
C$
    IF, 'MEASUREMENT' LT 100 , THEN $
C$
       OUTPUT, TEXT, FROM
```

```
        'SPACE-BEFORE',
        C'There is a SHORT between J1-10 and EARTH, or it is not the UUT\LF\',
        C'ACP, P/N 5701-1-1\LF\',
        'SPACE-AFTER' $
        PERFORM, 'WAIT-MI' $
        PERFORM, 'ABORT-TEST' $
C$
    END, IF $
C$
 110030 MEASURE, (RES INTO 'MEASUREMENT'), IMPEDANCE USING 'OHMMETER',
    RES RANGE 0 OHM TO 30000 OHM,
    CNX HI  J1-16
      LO  EARTH $
C$
    IF, 'MEASUREMENT' LT 100 , THEN $
C$
       OUTPUT, TEXT, FROM
        'SPACE-BEFORE',
        C'There is a SHORT between J1-16 and EARTH, or it is not the UUT\LF\',
        C'ACP, P/N 5701-1-1\LF\',
        'SPACE-AFTER' $
        PERFORM, 'WAIT-MI' $
        PERFORM, 'ABORT-TEST' $
C$
    END, IF $
C$
C ----- Verify if not short on 5VAC pins J1-22 refer to J1-23 ------ $
C$
 110040 MEASURE, (RES INTO 'MEASUREMENT'), IMPEDANCE USING 'OHMMETER',
    RES RANGE 0 OHM TO 30000 OHM,
    CNX HI  J1-10
      LO  EARTH $
C$
    IF, 'MEASUREMENT' LT 100 , THEN $
C$
       OUTPUT, TEXT, FROM
        'SPACE-BEFORE',
        C'There is a SHORT between J1-22 and J1-23, or it is not the UUT\LF\',
        C'ACP, P/N 5701-1-1\LF\',
        'SPACE-AFTER' $
        PERFORM, 'WAIT-MI' $
        PERFORM, 'ABORT-TEST' $
C$
    END, IF $
C$
C$
C ********************************************************************** $
C *      BLOCK 1.2  INITIAL CONDITIONS                                 * $
C ********************************************************************** $
C$
C$
    OUTPUT, TEXT, FROM
      'SPACE-BEFORE',
      C'\LF\',
      C' --- BLOCK 1.2  INITIAL CONDITIONS ---\LF\',
      'SPACE-AFTER' $
C$
C     OUTPUT, TEXT TO 'LOG-FILE', FROM
      C'\LF\\LF\',
      C' --- BLOCK 1.2  INITIAL CONDITIONS ---\LF\\LF\' $
C$
C------- MIC PTT IN = OFF ( switch OPEN ) ---------------$
C------- INT PTT IN = OFF ( switch OPEN ) ---------------$
C------- LAMP TEST  = OFF ( switch CLOSED to 28 V) -------$
C$
 120010 APPLY, SHORT USING 'SHORT3',
      CNX K1-4
        J1-14 $
C$
C------- 28VDC IND LTS.....HIGH ---------------- $
C$
   20 APPLY, SHORT USING 'SHORT1',
      CNX J1-16
        R107  $
C$
        OUTPUT, TEXT, FROM
    'SPACE-BEFORE',
    C'\LF\',
    C' OPERATOR ACTION:', C'\LF\\LF\',
    C'  - Set all the ACP potentiometers FULLY CCW          \LF\\LF\',
    C'  - Set the ACP rotary switches as indicated below.   \LF\\LF\',
    C'-------------------------------------------------------\LF\',
    C'|      SWITCH        |      POSITION        |\LF\',
    C'|-------------------------------------------------------|\LF\',
    C'|   VOR ADF  SWITCH      |       VOR L          |\LF\',
    C'|   V/B/R    SWITCH      |       V              |\LF\',
    C'|   APP      SWITCH      |       APP L          |\LF\',
    C'-------------------------------------------------------\LF\',
    'SPACE-AFTER' $
C$
    PERFORM, 'WAIT-MI'$
C$
C$
C ********************************************************************** $
C *      BLOCK 1.3  POWER-UP TESTS                                     * $
C ********************************************************************** $
C$
C$
    OUTPUT, TEXT, FROM
      'SPACE-BEFORE',
      C'\LF\',
```

```
      C' — BLOCK 1.3  POWER-UP TESTS —\LF\',
      'SPACE-AFTER' $
C$
    OUTPUT, TEXT TO 'LOG-FILE', FROM
      C'\LF\',
      C' — BLOCK 1.3  POWER-UP TESTS —\LF\\LF\' $
C$
130000 APPLY, DC SIGNAL USING '+28-VDC-POWER-SUPPLY',
     VOLTAGE +28.0 V ERRLMT +-0.1 V,
     CURRENT MAX 2 A,
     CNX HI J1-10
       LO EARTH $
C$
    WAIT FOR, TIME 5 SEC  $
C$
130010 VERIFY, (CURRENT), DC SIGNAL USING 'DC-AMMETER',
         LE 0.5 A,
         CURRENT RANGE 0 A TO 2 A,
         CNX VIA J1-10A $
C$
    PERFORM, 'CHECK-NOGO' (C'STATNO 130010 ', 0.05, 'MEASUREMENT', 0.5, C'A') $
C$
130020 VERIFY, (VOLTAGE), DC SIGNAL USING 'DC-VOLTMETER',
     UL 28.5 V  LL 27.5 V,
     VOLTAGE RANGE 0V TO 30V,
     CNX  HI  J1-10
       LO  EARTH  $
C$
    PERFORM, 'CHECK-NOGO' (C'STATNO 130020 ', 27.5, 'MEASUREMENT', 28.5, C'VOLT') $
C$
130030 VERIFY, (VOLTAGE), DC SIGNAL USING 'DC-VOLTMETER',
     UL 28.5 V  LL 27.5 V,
     VOLTAGE RANGE 0V TO 30V,
     CNX  HI  J1-16
       LO  EARTH  $
C$
    PERFORM, 'CHECK-NOGO' (C'STATNO 130030 ', 27.5, 'MEASUREMENT', 28.5, C'VOLT') $
C$
C ---------- Initiate the NORMAL MODE for ACP ---------- $
C$
    ENABLE, EXCHANGE-CONFIGURATION USING 'XMT',
      PROTOCOL 'XMT-BUS',
    CNX TRUE J1-17 COMPL J1-18 $
C$
    WAIT FOR, TIME 0.1 SEC  $
C$
    CALCULATE, 'SEND-WORD'(1) =  'SCC-ACP-CT-1' $
    CALCULATE, 'SEND-WORD'(2) =  'SCC-ACP-CT-2' $
C$
    DO, EXCHANGE USING 'XMT', TEST-EQUIP-ROLE MASTER,
      (EXCHANGE 'SEND-W',
        TEST-EQUIP-ROLE MASTER,
        DELAY 40 MSEC),
    PROCEED $
C$
C---------- 5 VAC POWER-SUPPLY  ---------- $
C$
    APPLY, AC SIGNAL USING '5-VAC-POWER-SUPPLY',
         VOLTAGE 5V,
         FREQ 400HZ,
         CURRENT MAX 0.5A,
         CNX HI J1-22
           LO J1-23 $
C$
130040 VERIFY, (VOLTAGE), AC SIGNAL USING 'AC-VOLTMETER',
     UL 5.2 V  LL 4.5 V,
     VOLTAGE RANGE 0V TO 6V,
     CNX  HI  J1-22
       LO  J1-23  $
C$
    PERFORM, 'CHECK-NOGO' (C'STATNO 130040 ', 4.5, 'MEASUREMENT', 5.2, C'VOLT') $
C$
130050 VERIFY, (CURRENT), AC SIGNAL USING 'AC-AMMETER',
    UL 0.5A LL 0.005A,
    CURRENT RANGE 0A TO 1A,
    CNX VIA J1-22A $
C$
    PERFORM, 'CHECK-NOGO' (C'STATNO 130050 ', 0.05, 'MEASUREMENT', 0.5, C'A') $
C$
199999 END, BLOCK '1.0 PRE-POWER/POWER-UP TESTS' $


C$
C ********************************************************************** $
C *      BLOCK 2.0  INDICATOR TEST                                     * $
C ********************************************************************** $
C$
C$
E200000 BEGIN, BLOCK '2.0 INDICATOR TEST' $
C$
    OUTPUT, TEXT, FROM
      'SPACE-BEFORE',
      C'\LF\',
      C' *** BLOCK 2.0  INDICATOR TEST ***\LF\',
      'SPACE-AFTER' $
C$
    OUTPUT, TEXT TO 'LOG-FILE', FROM
      C'\LF\\LF\',
      C' *** BLOCK 2.0  INDICATOR TEST ***\LF\\LF\' $
C$
```

```
     OUTPUT, TEXT, FROM
       'SPACE-BEFORE',
       C'\LF\',
       C' OPERATOR ACTION :', C'\LF\\LF\',
       C'  - Watch the indicator lamps from the UUT front panel \LF\',
       C'  - All indicators must turn ON for a 5 sec. period, then OFF \LF\',
       'SPACE-AFTER' $
C$
     PERFORM, 'WAIT-MI' $
C$
C------- LAMP TEST = ON ( switch CLOSED to EARTH ) ---------$
C$
     DISCONNECT, SHORT USING 'SHORT3',
       CNX K1-4 J1-14 $
C$
     CONNECT, SHORT USING 'SHORT3',
       CNX EARTH J1-14 $
C$
             WAIT FOR, TIME 5 SEC $
C$
C-------- LAMP TEST = OFF ( switch CLOSED to EARTH ) --------$
C$
     DISCONNECT, SHORT USING 'SHORT3',
       CNX EARTH J1-14 $
C$
     CONNECT, SHORT USING 'SHORT3',
       CNX K1-4 J1-14 $
C$
 200100 OUTPUT, TEXT, FROM
       'SPACE-BEFORE',
       C'\LF\',
       C' OPERATOR ACTION :', C'\LF\\LF\',
       C'  - Did the lights turned  ON / OFF ? \LF\',
       'SPACE-AFTER' $
C$
     PERFORM , 'OPERATOR-CONFIRM' (C'STATNO 200100',
                     C'ALL INDICATOR LIGHTS TURNES ON / OFF') $
C$
C$
 299999 END, BLOCK '2 0 INDICATOR TEST' $



C$
E900000 BEGIN, BLOCK '9.0 POWER INTERUPT TEST' $
C$
C---- SET THE INDICATORS AS SHOWN IN TABLE 113 PAG 118 CMM $
C$
     OUTPUT, TEXT, FROM
         'SPACE-BEFORE',
         C'\LF\',
         C' OPERATOR ACTION :', C'\LF\\LF\',
         C'  - Set the following indicator lamps ON and all others OFF\LF\',
         C'   by pressing the appropriate switches \LF\\LF\',
         C'     VHF L VOL      VHF C VOL       VHF R VOL\LF\',
         C'     FLT VOL        CAB VOL         PA VOL\LF\',
         C'     HF L VOL       SAT 1 VOL       VOR ADF VOL\LF\',
         C'     SPKR VOL       VHF L MIC/CALL\LF\\LF\',
              C'*************************************************************************\LF\' $
C$
     PERFORM, 'WAIT-MI' $
C$
C---- 200 MSEC POWER INTERRUPT $
C$
 900100        APPLY, PULSED DC USING 'FUNCTION-GENERATOR',
                         VOLTAGE 5V,
                         DC-OFFSET -5V,
                         PERIOD 0.4 SEC,
                         PULSE-COUNT 1 TIMES,
               CNX HI RELAY-HI
                 LO RELAY-LO $
C$
C---- ALL INDICATORS MUST GO BACK TO THEIR SETTING PRIOR TO PWR INTERRUPT.
    REFER TO TABLE 113 PAG118 CMM  $
C$
     OUTPUT, TEXT, FROM
         'SPACE-BEFORE',
              C'\LF\',
         C' OPERATOR ACTION :', C'\LF\\LF\',
         C'  - Look at the indicator lamps \LF\',
         C'  - Following indicators must be ON and all others OFF\LF\\LF\',
         C'     VHF L VOL      VHF C VOL       VHF R VOL\LF\',
         C'     FLT VOL        CAB VOL         PA VOL\LF\',
         C'     HF L VOL       SAT 1 VOL       VOR ADF VOL\LF\',
         C'     SPKR VOL       VHF L MIC/CALL\LF\\LF\',
         C'  - Did the indicators configuration CORRECT ?\LF\',
              C'*************************************************************************\LF\' $
C$
     PERFORM, 'OPERATOR-CONFIRM' (C'STATNO 900100', C'THE INDICATORS CONFIGURATION ') $
C$
C--- Set all the indicators and potentiometers as in INITIAL CONDITIONS $
C$
     OUTPUT, TEXT, FROM
         'SPACE-BEFORE',
         C'\LF\',
         C' OPERATOR ACTION :', C'\LF\\LF\',
         C'  - Turn OFF all the indicator lamps, by pressing the appropriate switches.\LF\',
         C'  - Ensure that all potentiometers are in FULL CCW position. \LF\',
         'SPACE-AFTER' $
C$
     PERFORM, 'WAIT-MI' $
```

```
C$
 999000 DISABLE. EXCHANGE-CONFIGURATION USING 'XMT' $
C$
 999990 END. BLOCK '9 0 POWER INTERUPT TEST' $
C$
C ********************************************************************** $
C              END OF TEST                          $
C ********************************************************************** $
C$
 999995 REMOVE. ALL $
 999997 PERFORM. 'TERMINATE' $
      PERFORM. 'DISABLE-IO' $
C$
C$
 999999 TERMINATE. ATLAS PROGRAM 'ACP-ATL' $
```

Exemplu de drivere de nivel înalt sub mediul PAWS varianta pentru UnixWare

```
*=============================================================================
*                                                                            *
*                    DeviceDB DINAMIC DESCRİPTION                            *
*                                                                            *
*=============================================================================


*****  Built-In Function Definitions - Short names  *****

def, fnc, Cat  ==    1;    ** ConcatenateStrings
def, fnc, Cmp  ==    2;    ** Compare Strings
def, fnc, Cpy  ==    3;    ** CopyStrings
def, fnc, Ilc  ==    4;    ** InsertLeadingCharacters
def, fnc, Itc  ==    5;    ** InsertTrailingCharacters
def, fnc, Itoc ==    6;    ** IntegerToNumericString
def, fnc, Tup  ==    7;    ** LowerCaseToUpperCase
def, fnc, Slw  ==    8;    ** RemoveLeadingWhiteSpace
def, fnc, Stw  ==    9;    ** RemoveTrailingWhiteSpace
def, fnc, Rev  ==    10;   ** ReverseString
def, fnc, Sfw  ==    11;   ** SetCurrentFieldWidth
def, fnc, Tlo  ==    12;   ** UpperCaseToLowerCase
def, fnc, Bic  ==    13;   ** BusInterfaceClear
def, fnc, Bto  ==    14;   ** SetBusTimeOut
def, fnc, Wcmd ==    15;   ** WriteCommandString
def, fnc, Wdat ==    16;   ** WriteDataString
def, fnc, Rtn  ==    17;   ** GetCurrentFunctionReturn
def, fnc, Print ==   18;   ** Print
def, fnc, Abort ==   19;   ** Abort
def, fnc, Stop ==    20;   ** Stop
def, fnc, Apnd ==    21;   ** AppendString
def, fnc, Split ==   22;   ** ParseCharacters
def, fnc, Gnum ==    23;   ** ParseDecimalString
def, fnc, Gint ==    24;   ** ParseNumericString
def, fnc, GCla ==    25;   ** GetBusControllerListenAddress
def, fnc, GCta ==    26;   ** GetBusControllerTalkAddress
def, fnc, GDla ==    27;   ** GetBusDeviceListenAddress
def, fnc, GDid ==    28;   ** GetBusDeviceName
def, fnc, GDta ==    29;   ** GetBusDevciceTalkAddress
def, fnc, Sdev ==    30;   ** SetBusDeviceNumber
def, fnc, Seoi ==    31;   ** SetEoiLineFlag
def, fnc, Talk ==    32;   ** TalkToBusDevice
def, fnc, Ctof ==    33;   ** DecimalStringToFloat
def, fnc, Ftoc ==    34;   ** FloatToDecimalString
def, fnc, Ctoi ==    35;   ** NumericStringToInteger
def, fnc, Sdp  ==    36;   ** SetDecimalPointFlag
def, fnc, Sexp ==    37;   ** SetExponentIndicator
def, fnc, SexpS ==   38;   ** SetExponentSignFlag
def, fnc, SexpW ==   39;   ** SetExponentWidth
def, fnc, SfW  ==    40;   ** SetFractionWidth
def, fnc, SiW  ==    41;   ** SetIntegerWidth
```

```
def, fnc, SmanS ==    42;   ** SetNumberSignFlag
def, fnc, Listn ==    43;   ** ListenToBusDevice
def, fnc, Read  ==    44;   ** ReadDataString
def, fnc, Dsp   ==    45;   ** Display
def, fnc, GpK   ==    46;   ** GetPathCount
def, fnc, GpD   ==    47;   ** GetPathData
def, fnc, GiK   ==    48;   ** IndirectPathCount
def, fnc, GiD   ==    49;   ** IndirectPathToDirect
def, fnc, RBin  ==    50;   ** ReadBinary
def, fnc, WBin  ==    51;   ** WriteBinary
def, fnc, XtoD  ==    52;   ** HexToDigital
def, fnc, DtoX  ==    53;   ** DigitalToHex
def, fnc, FmtDtoS  == 54;   ** FormatDataToString        (NEW)
def, fnc, FmtIBPut == 55;   ** FormattedBufferedBusOutput  (NEW)
def, fnc, CclrF ==    56;   ** ClearBusClearFlag
def, fnc, CkSum ==    57;   ** ClearGetCheckSum


*****************************************************************************


*****  Built-In Function Definitions - Long names  *****

def, fnc, ConcatenateStrings          == 1;
def, fnc, CompareStrings              == 2;
def, fnc, CopyString                  == 3;
def, fnc, InsertLeadingCharacters     == 4;
def, fnc, InsertTrailingCharacters    == 5;
def, fnc, IntegerToNumericString      == 6;
def, fnc, LowerCaseToUpperCase        == 7;
def, fnc, RemoveLeadingWhiteSpace     == 8;
def, fnc, RemoveTrailingWhiteSpace    == 9;
def, fnc, ReverseString               == 10;
def, fnc, SetFieldWidth               == 11;
def, fnc, UpperCaseToLowerCase        == 12;
def, fnc, InterfaceClear              == 13;
def, fnc, SetBusDeviceTimeout         == 14;
def, fnc, WriteCommandString          == 15;
def, fnc, WriteDataString             == 16;
def, fnc, GetFunctionReturn           == 17;
def, fnc, AppendString                == 21;
def, fnc, ParseCharacters             == 22;
def, fnc, ParseDecimalString          == 23;
def, fnc, ParseNumericString          == 24;
def, fnc, GetBusControllerListenAddress == 25;
def, fnc, GetBusControllerTalkAddress   == 26;
def, fnc, GetBusDeviceListenAddress     == 27;
def, fnc, GetBusDeviceName            == 28;
def, fnc, GetBusDeviceTalkAddress     == 29;
def, fnc, SetBusDeviceNumber          == 30;
def, fnc, SetEoiLineFlag              == 31;
def, fnc, TalkToBusDevice             == 32;
def, fnc, DecimalStringToFloat        == 33;
def, fnc, FloatToDecimalString        == 34;
def, fnc, NumericStringToInteger      == 35;
def, fnc, SetDecimalPointFlag         == 36;
```

```
def, fnc, SetExponentIndicator        == 37;
def, fnc, SetExponentSignFlag         == 38;
def, fnc, SetExponentWidth            == 39;
def, fnc, SetFractionWidth            == 40;
def, fnc, SetIntegerWidth             == 41;
def, fnc, SetNumberSignFlag           == 42;
def, fnc, ListenToBusDevice           == 43;
def, fnc, ReadDataString              == 44;
def, fnc, Display                     == 45;
def, fnc, GetPathCount                == 46;
def, fnc, GetPathData                 == 47;
def, fnc, IndirectPathCount           == 48;
def, fnc, IndirectPathToDirect        == 49;
def, fnc, ReadBinary                  == 50;
def, fnc, WriteBinary                 == 51;
def, fnc, HexStringToDigital          == 52;
def, fnc, DigitalToHexString          == 53;
def, fnc, EncodeString                == 54;
def, fnc, EncodeBuf                   == 54;
def, fnc, FTalk                       == 55;
def, fnc, SetClearFlag                == 56;
def, fnc, SetGetChecksum              == 57;
def, fnc, CheckSRQLine                == 58;
def, fnc, GetToken                    == 59;
def, fnc, WriteToGPIO                 == 91;
def, fnc, ReadFromGPIO                == 92;
def, fnc, TalkVXI                     == 93;
def, fnc, ListenVXI                   == 94;
def, fnc, ReadSTB                     == 95;
def, fnc, OpenSerial                  == 101;
def, fnc, CloseSerial                 == 102;
def, fnc, WriteAsciiSerial            == 103;
def, fnc, ReadAsciiSerial             == 104;
def, fnc, SetTimeoutSerial            == 105; *
def, fnc, WritePXB                            == 110; ** Added for RADA 608
def, fnc, InitPXB                     == 111; ** Added for RADA 608



*****************************************************************************
*                       DUMMY MACRO                                         *
*****************************************************************************

def, mac, DummyMacro( );

{
;
}

*****************************************************************************
*               SWITCHING DRIVER DYNAMIC DESCRIPTION                        *
*****************************************************************************

dcl {
        ***** SWITCH GLOBALS
     int simulation;
```

```
        int PATHINFO(5);
        bln SwxBool;
        int matrix_relay, bus_relay, dist_relay, resc_relay;
        int sec_dist_relay, sec_resc_relay;
        int card_address(8), trip;
        int path_sum_before(6);
        int path1_after(8), path2_after(8), path3_after(8), path4_after(8);
        int active_paths;
        int debug;
        int relay_tmp;
        int index;

        int BusGrp(4);
        int RscGrp(4);

        int src_swx_cnt, pwr_swx_cnt, matrix_path_seen;
        int src_relays(20);
        int pwr_relays(20);

        int dis_matrix_relay, dis_bus_relay, dis_dist_relay, dis_resc_relay;
        int dis_sec_bus_relay, dis_sec_dist_relay, dis_sec_resc_relay;

        int PreviousMod(20), PreviousPth(20);
        *** end SWITCH

        *** Other Globals
  int  NULLPATH;        ** needed for switching
        bln  BLN_VAR;         ** needed for interrupts
  int  itmp;            ** temp integer for work
        txt  ttmp 50;         ** temp txt buffer for work
        txt  cmd_buf 250;     ** Command Buffer used to buffer strings
        txt  tbuf 100;        ** Temp Buffer used for work

        txt resource_ids(20) 50;
        int resource_seen(20);
        txt swx_action 10;

     int LineHandle(2);     ** /dev/tty00, /dev/tty01 Lines
        int CurrentLine;

   }

*******************************************************************************
*******************************************************************************
*******************************************************************************
***
***
***       SWITCH Macros

def, mac, StartConnect();
{
        SETDEVICE(<SWX>);
        trip = 1;
        src_swx_cnt = 0;
        pwr_swx_cnt = 0;
        bus_relay = 0;
```

```
matrix_path_seen = 0;      **** JMH
}
def, mac, StartDisConnect();
{
        SETDEVICE(<SWX>);
        trip = 1;
        src_swx_cnt = 0;
        pwr_swx_cnt = 0;
        bus_relay = 0;
matrix_path_seen = 0;
}


***
*************************************************************************
*** This macro will send the Power Switch Closures
***

def, mac, SendPowerSwx(action);
int i;
int action;
{
        for(i = 1; i <= pwr_swx_cnt; i = i + 1)
                WritePXB(action, pwr_relays(i), simulation);
}

*************************************************************************
***
*** This macro will send the Source Switch Closures
***

def, mac, SendSourceSwx(action);
int i;
int action;
{
        for(i = 1; i <= src_swx_cnt; i = i + 1)
                WritePXB(action, src_relays(i), simulation);
}

*************************************************************************
***
*** This macro will process and buffer the Source Switchs.
***

def, mac, ProcessSrcPwrSwitchs(src_pwr_mod, src_pwr_pth);
int src_pwr_mod, src_pwr_pth;
{
        if((src_pwr_mod / 100) == 20)
        {pwr_swx_cnt = pwr_swx_cnt + 1;
         pwr_relays(pwr_swx_cnt) = ((src_pwr_mod % 100) * 1000) + src_pwr_pth;
        }
        else
        {src_swx_cnt = src_swx_cnt + 1;
         src_relays(src_swx_cnt) = ((src_pwr_mod % 100) * 1000) + src_pwr_pth;
        }
}
```

```
*************************************************************************
***
*** This macro will prepare, during the CONNECT, for the DISCONNECT.
***       This is necessary due to the fact that if a conflict is detected
***       during processing of the CONNECT and the relays must be ReCalculated,
***       the cooresponding DISCONNECT will need to know which path was choosen.
***
def, mac, PrepareDISCONpaths();
          {
          .


     }

*****************************************************************************
**
**     Compute a ResourceGroup based upon Relay Number
**
def, mac, ComputeIndex(relay, ref); int relay; int ref;
          {
           if (relay >= 0)
             {
              index = relay % 4;
              if (index == 0) index = 4;

****Display("Relay=", relay, " Index=", index, " @", ref, "\n");
              }
           else
              {
               Display("ERROR: Invalid Relay # ", relay, "\n");
               Stop();
              }
     }


*****************************************************************************
**
**     Select final Distributor and Resource Relays based upon
**     availablility of Resource Group
**
def, mac, SelDisRscRly(@ dist_relay, @ resc_relay);
int dist_relay, resc_relay;
int    check_count;
      {



}


def, mac, CheckBusGroups();
int tier, tier2;
int relay_tmp;
int i;
```

```
        {
        .


}

*************************************************************************
*************************************************************************
*************************************************************************

def, mac, ExecuteMatrixDisconnect(@ pathX_after); int pathX_after(8);
        {
        .


    }



def, mac, ExecuteDISCON();
int current_path_sum;
int i, path_pointer;
        {
        .


}

*************************************************************************
*************************************************************************
*************************************************************************

def, mac, ProcessTriplets();
int mod, pth;
txt tmp_buf 250;
txt resource_buf 250;
txt resource_dist 20;
int plug, i;
bln skip;
{



}

*************************************************************************
*************************************************************************
*************************************************************************
def, mac, PATH();
{
 ;
}

*************************************************************************
*************************************************************************
```

```
**************************************************************************
def, mac, ResetSwx();
int i;
{
        SETDEVICE(<SWX>);

        matrix_relay = 0;
        bus_relay = 0;
        dist_relay = 0;
        resc_relay = 0;
        sec_dist_relay = 0;
        sec_resc_relay = 0;

        active_paths = 0;

        src_swx_cnt = 0;
        pwr_swx_cnt = 0;

        RscGrp(1) = 0; RscGrp(2) = 0; RscGrp(3) = 0; RscGrp(4) = 0;
        BusGrp(1) = 0; BusGrp(2) = 0; BusGrp(3) = 0; BusGrp(4) = 0;
}

**************************************************************************
**************************************************************************
**************************************************************************

def, mac, DispatchCON();
{
        if(src_swx_cnt != 0)
                SendSourceSwx(30);
        if(pwr_swx_cnt != 0)
                SendPowerSwx(30);
        if(matrix_path_seen == 1)
                CheckBusGroups();
}

**************************************************************************
**************************************************************************
**************************************************************************
def, mac, InitSWX();
bln SwxBool;
{
    InitPXB(simulation);
        ResetSwx();
        ** CONNECT
        enable 40 StartConnect     SwxBool; ** Start Connection Path
        enable 41 ProcessTriplets  SwxBool; ** Process Triplets
        enable 42 DispatchCON      SwxBool; ** Manage Bus and Matrix
        ** DISCONNECT
        enable 43 StartDisConnect  SwxBool; ** Start DisConnection Path
        enable 44 ProcessTriplets  SwxBool; ** Process Triplets
        enable 45 ExecuteDISCON    SwxBool; ** Manage Bus and Matrix

}
```

```
*****************************************************************************
*            MANUAL INTERVENTION DINAMIC DESCRIPTION              *
*****************************************************************************

dcl
{
  int MIMaxTime;
}

def, mac, InxMIEvent( T);
dec T;  * MAX-TIME
{
    MIMaxTime = T + 0.5;
    'manual-intervention' = FALSE;
}

def, mac, FthMIEvent( @R);
bln R;  * EVENT-INDICATOR
{
    while(( 'manual-intervention' == FALSE) && ( MIMaxTime > 0))
    {
      MIMaxTime = MIMaxTime - 1;
      SETDELAY(1);
    }
    R = 'manual-intervention;
}




*****************************************************************************
** TIMTEH S.R.L.                                          **
** DMM Driver Dinamic Description                                 **
** Version 5                                  **
** Author:                                          **
*****************************************************************************

dcl
{
          txt DMM_Buffer 80;
          int DMM_itemp;
          txt DMM_ttemp 20;
          int DMM_chars_read;
          txt DMM_commands(15) 8;
     dec DMM_ranges(10);
     dec DMM_resolutions(3);
}

def, mac, DMM_BusDeviceError(error_msg);
          txt error_msg 20, device 10;
{
          GDid(device);
          Dsp(device," = ", error_msg,"\n");
          Stop();
}
```

```
def, mac, DMM_AppendNR3( value);
        dec value;
{
        Sfw( 15);
        SmanS(1);
        SiW( 1);
        Sdp( 0);
        SfW( 7);
        Sexp( "E");
        SexpS( 1);
        SexpW( 3);
        Ftoc( value, DMM_ttemp);
        Apnd( DMM_ttemp, DMM_Buffer);
}


def, mac, DMM_AppendInt( inum);
        int inum;
{
        Itoc( inum, DMM_ttemp);
        Apnd( DMM_ttemp, DMM_Buffer);
}

*****************************************************************************
** IEEE Controller communication **

def, mac, DMM_TalkBus();
{
        Apnd("\r\n",DMM_Buffer);
        Talk(DMM_Buffer);
        Rtn(DMM_itemp);

        if(DMM_itemp<0)
                DMM_BusDeviceError("BUS TALK ERROR");

        Cpy("",DMM_Buffer);
}


def, mac, DMM_ListenBus();
{
        Listn(DMM_Buffer);
        Rtn(DMM_itemp);

        if(DMM_itemp < 0)
                DMM_BusDeviceError("BUS LISTEN ERROR");
        else
                DMM_chars_read = DMM_itemp;
}

def, mac, DMM_Status();                 * Call after REMOVE ALL, TERMINATE ...
         int stat;
{
         Cpy( "*STB?", DMM_Buffer);     * poll status byte
         DMM_TalkBus();
         DMM_ListenBus();
```

```
        Ctoi( DMM_Buffer, stat);

        if( stat != 0)
        {
         Dsp( "   ERROR !    Final DMM_Status Byte = ", stat, "\n");
         DMM_BusDeviceError("DMM_Status ERROR");
        }
}


def, mac, DMM_Sta();                * Call after FETCH
        int sta;
{
 Cpy( "*STB?", DMM_Buffer);     * poll status byte
 DMM_TalkBus();
 DMM_ListenBus();

 Ctoi( DMM_Buffer, sta);

 if ((sta & x'20') != 0)
 {
        Cpy( "*ESR?", DMM_Buffer);      * poll Standard Event
        DMM_TalkBus();
        DMM_ListenBus();
        Ctoi( DMM_Buffer, DMM_itemp);
        if((DMM_itemp & x'64') != 0)          * operation complete is ignored
        {
         Dsp( "   ERROR !   Standard Event Register = ", DMM_itemp, "\n");
         DMM_BusDeviceError("DMM_Status ERROR");
    }
 }
 if ((sta & x'08') != 0)
 {
        Cpy( "STAT:QUES:EVEN?", DMM_Buffer);          * poll Quest.Data
        DMM_TalkBus();
        DMM_ListenBus();
        Ctoi( DMM_Buffer, DMM_itemp);
        if( DMM_itemp != 512)         * Ohms overload is ignored
        {
         Dsp( "   ERROR !   Questionable Data Register = ", DMM_itemp, "\n");
         DMM_BusDeviceError("DMM_Status ERROR");
        }
 }
}


def, mac, DMM_ResetDevice();
{
        Cpy( "*RST", DMM_Buffer);       * Reset
        DMM_TalkBus();
        Cpy( "*CLS", DMM_Buffer);    * Clear status
        DMM_TalkBus();
}
```

```
def, mac, DMM_AMP1_Reset();
{
     WritePXB(30, 8032, simulation);   * Close the current path
     SETDELAY(0.5);                    * Wait for bouncing
     WritePXB(31, 8033, simulation);   * Open Ampmeter port
     WritePXB(31, 8034, simulation);   * Open Ampmeter port
     DMM_ResetDevice();
}

def, mac, DMM_AMP2_Reset();
{
     WritePXB(30, 8035, simulation);   * Close the current path
     SETDELAY(0.5);                    * Wait for bouncing
     WritePXB(31, 8036, simulation);   * Open Ampmeter port
     WritePXB(31, 8037, simulation);   * Open Ampmeter port
     DMM_ResetDevice();
}


def, mac, DMM_CommFill();
{
          Cpy( "VOLT:DC", DMM_commands( 1));
          Cpy( "CURR:DC", DMM_commands( 2));
          Cpy( "RES",     DMM_commands( 3));
          Cpy( "FRES",    DMM_commands( 4));
          Cpy( "VOLT:AC", DMM_commands( 5));
          Cpy( "CURR:AC", DMM_commands( 6));
     DMM_resolutions(1) = 1.0e6;
     DMM_resolutions(2) = 1.0e5;
     DMM_resolutions(3) = 1.0e4;
     DMM_ranges(1)  = 0.010;
     DMM_ranges(2)  = 0.100;
     DMM_ranges(3)  = 1.000;
     DMM_ranges(4)  = 3.000;
     DMM_ranges(5)  = 10.00;
     DMM_ranges(6)  = 100.0;
     DMM_ranges(7)  = 750.0;
     DMM_ranges(8)  = 1.0e3;
     DMM_ranges(9)  = 1.0e4;
     DMM_ranges(10) = 1.0e5;
}

def, mac, DMM_Init();
{
          SETDEVICE(<DMM>);
          DMM_CommFill();
          Cpy("",DMM_Buffer);
     Cpy( "*SRE 0", DMM_Buffer);          * Service enable reg: Disable all
     DMM_TalkBus();
     Cpy( "*ESE 255", DMM_Buffer);        * Standard Event enab.reg: Enab. all
     DMM_TalkBus();
          Cpy( "STAT:QUES:ENAB 515", DMM_Buffer);      * Enab. all overload bits
          DMM_TalkBus();
     DMM_AMP1_Reset();
     DMM_AMP2_Reset();
*    DMM_ResetDevice();
```

```
*         DMM_Status();
*         Cpy( "ROUT:TERM?", DMM_Buffer);       * Check the active terminals (FRONT / REAR)
*         DMM_TalkBus();
*         DMM_ListenBus();
*   Dsp( "Active terminals: ", DMM_Buffer);
}


*****************************************************************************
** SETUP ** ATLAS single verb handling

def, mac, DMM_Setup( function, range_val, speed);
int function, range_val, speed;
dec range, resolution;
int fnc, mode;
{
     fnc  = function % 10;
     mode = (int) function / 10;
          Cpy( "CONF:", DMM_Buffer);
          Apnd( DMM_commands(fnc), DMM_Buffer);
     Apnd(" ", DMM_Buffer);
*********************************************************
     if ((fnc == 3) || (fnc == 4))
      range_val = 0;                          ** auto ranging for resistance
*********************************************************
          if( range_val == 0)
             Apnd( "DEF,DEF", DMM_Buffer);                  ** AUTO ranging
          else
        {
         range = DMM_ranges(range_val);
         if ((fnc == 3) || (fnc == 4))
           range = range*1000.0;                    ** resistance measurement
         if ((range_val == 4) || (range_val == 7))
           range_val = range_val + 1;
         resolution = DMM_ranges(range_val);
         if ((fnc == 3) || (fnc == 4))
          resolution = resolution*1000.0;             ** resistance measurement
         resolution = resolution / DMM_resolutions(mode);
              DMM_AppendNR3( range);
         Apnd(",", DMM_Buffer);
         DMM_AppendNR3( resolution);
        }
     DMM_TalkBus();

     if((fnc == 5) || (fnc == 6))
     {
      if (speed == 1) Cpy("DET:BAND 3", DMM_Buffer);
      if (speed == 2) Cpy("DET:BAND 20", DMM_Buffer);
      if (speed == 3) Cpy("DET:BAND 200", DMM_Buffer);
      DMM_TalkBus();
     }

     if((fnc == 1) || (fnc == 2) || (fnc == 3) || (fnc == 4))
     {
      Cpy(DMM_commands(fnc), DMM_Buffer);
      Apnd(":NPLC ", DMM_Buffer);
```

```
        if (speed == 1) Apnd("100", DMM_Buffer);
        if (speed == 2) Apnd("10", DMM_Buffer);
        if (speed == 3) Apnd("1", DMM_Buffer);
        DMM_TalkBus();
      }
}

def, mac, DMM_AMP1_Setup( function, range_val, speed);
int function, range_val, speed;
dec range;
{
      DMM_Setup(function, range_val, speed);
      WritePXB(30, 8033, simulation);    * Close Ampmeter port
      WritePXB(30, 8034, simulation);    * Close Ampmeter port
      SETDELAY(0.5);                     * Wait for bouncing
      WritePXB(31, 8032, simulation);    * Open the current path
}

def, mac, DMM_AMP2_Setup( function, range_val, speed);
int function, range_val, speed;
dec range;
{
      DMM_Setup(function, range_val, speed);
      WritePXB(30, 8036, simulation);    * Close Ampmeter port
      WritePXB(30, 8037, simulation);    * Close Ampmeter port
      SETDELAY(0.5);                     * Wait for bouncing
      WritePXB(31, 8035, simulation);    * Open the current path
}


*****************************************************************************
** INITIATE ** ATLAS single verb handling

** ARM ** ATLAS single verb handling

def, mac, DMM_Inx();
{
          Cpy( "INIT", DMM_Buffer);
          DMM_TalkBus();
}

*****************************************************************************
** FETCH ** ATLAS single verb handling

def, mac, DMM_Fetch( @target);
dec target;
{
          Cpy( "FETC?", DMM_Buffer);
          DMM_TalkBus();
          DMM_ListenBus();

      Ctof( DMM_Buffer, target);
      DMM_Sta();
}
```

```
*****************************************************************************
*                COMMON RESERVED MACROS                                     *
*****************************************************************************

def, mac, RESETALL();
{
        itmp = 0;                    ** reset switch
        InitSWX();

        SETDEVICE(<DMM>);
        DMM_ResetDevice();
        DMM_Status();
    DMM_AMP1_Reset();
    DMM_AMP2_Reset();

}

def, mac, INITIALIZE();
{
        Bto(10);                     ** Bus Timeout - 10 Sec
        Bic();                       ** Interface Clear

** initialize switching driver
        simulation = 0;
        InitSWX();
        debug = 0;

** initialize all ATE devices
        DMM_Init();

        AUD-Initialize();            ** initialize AUD

    RESETALL;
}

def, mac, TERMINATE();
{
    RESETALL();
}


*=============================================================================
======
*                                                                           *
*                    STATIC DESCRIPTION                                     *
*                                                                           *
*=============================================================================
======


*****************************************************************************
*          MANUAL INTERVENTION STATIC DESCRIPTION                           *
*****************************************************************************
begin DEV NUL;
```

```
  begin;
    cnx event-out NUL-EV;

    init InxMIEvent( max-time: 0);
    fetch FthMIEvent( event-indicator result);

    event monitor (manual-intervention) event;

     control
     {
       event-out;
       event-indicator;
       max-time max 3000 sec;
     }
   end;
end;  ** DEV NUL


*****************************************************************************
*              STATIC DESCRIPTION                                           *
*****************************************************************************

begin dev EARTH1;
 cnx from EARTH,
    to EARTH;
   load earth;
end;


*****************************************************************************
** TIMTEH S.R.L.                                          **
** DMM Driver Static Description                                 **
** Version                                          **
** Author:                                          **
*****************************************************************************

begin DEV DMM using DMMhp;
          reset DMM_ResetDevice();
          init DMM_Inx();

          cnx hi DMM1_HI, lo DMM1_LO;

begin; ** dc & resistance functions - NPLC aplicable
 begin; **voltage functions
  setup DMM_Setup( $FNC, voltage $RNG:0, 2);
  fetch DMM_Fetch( voltage result);
  sensor (voltage) dc signal;
  capability
  {
   test-equip-imp-magnitude min 1E6 ohm;
   ac-comp max 0.1 v;
  }
  begin FNC=11;
          control
          {
           voltage range -1.0e3 v to 1.0e3 v by 1.0e-3 v errlmt +-0.05 pc RNG=8,
```

```
                range -1.0e2 v to 1.0e2 v by 1.0e-4 v errlmt +-0.05 pc RNG=6,
                range -1.0e1 v to 1.0e1 v by 1.0e-5 v errlmt +-0.05 pc RNG=5,
                range -1.0e0 v to 1.0e0 v by 1.0e-6 v errlmt +-0.05 pc RNG=3,
                range -0.1e0 v to 0.1e0 v by 1.0e-7 v errlmt +-0.05 pc RNG=2;
          ** 6 1/2 digits resolution
        }
end;
begin FNC=21;
        control
        {
         voltage range -1.0e3 v to 1.0e3 v by 1.0e-2 v errlmt +-0.05 pc RNG=8,
                range -1.0e2 v to 1.0e2 v by 1.0e-3 v errlmt +-0.05 pc RNG=6,
                range -1.0e1 v to 1.0e1 v by 1.0e-4 v errlmt +-0.05 pc RNG=5,
                range -1.0e0 v to 1.0e0 v by 1.0e-5 v errlmt +-0.05 pc RNG=3,
                range -0.1e0 v to 0.1e0 v by 1.0e-6 v errlmt +-0.05 pc RNG=2;
          ** 5 1/2 digits resolution
        }
end;
begin FNC=31;
        control
        {
         voltage range -1.0e3 v to 1.0e3 v by 1.0e-1 v errlmt +-0.05 pc RNG=8,
                range -1.0e2 v to 1.0e2 v by 1.0e-2 v errlmt +-0.05 pc RNG=6,
                range -1.0e1 v to 1.0e1 v by 1.0e-3 v errlmt +-0.05 pc RNG=5,
                range -1.0e0 v to 1.0e0 v by 1.0e-4 v errlmt +-0.05 pc RNG=3,
                range -0.1e0 v to 0.1e0 v by 1.0e-5 v errlmt +-0.05 pc RNG=2;
          ** 4 1/2 digits resolution
        }
end;
end; ** voltage funct

begin; ** 2 wires resistance functions
 setup DMM_Setup( $FNC, res $RNG:0, 2);
 fetch DMM_Fetch( res result);
 sensor (res) impedance;
 limit
 {
  pwr-lmt max 1 mw;
 }
 begin FNC=13;
    control
        {
         res range 0.0 ohm to 1.0e8 ohm by 1.00e2 ohm errlmt +-0.81 pc RNG=10,
            range 0.0 ohm to 1.0e7 ohm by 1.00e1 ohm errlmt +-0.04 pc RNG=9,
            range 0.0 ohm to 1.0e6 ohm by 1.0e-0 ohm errlmt +-0.01 pc RNG=8,
            range 0.0 ohm to 1.0e5 ohm by 1.0e-1 ohm errlmt +-0.01 pc RNG=6,
            range 0.0 ohm to 1.0e4 ohm by 1.0e-2 ohm errlmt +-0.01 pc RNG=5,
            range 0.0 ohm to 1.0e3 ohm by 1.0e-3 ohm errlmt +-0.01 pc RNG=3,
            range 0.0 ohm to 1.0e2 ohm by 1.0e-4 ohm errlmt +-0.01 pc RNG=2;
        ** - 6 1/2 digits resolution
    }
 end;
 begin FNC=23;
    control
        {
         res range 0.0 ohm to 1.0e8 ohm by 1.00e3 ohm errlmt +-0.81 pc RNG=10,
```

```
            range 0.0 ohm to 1.0e7 ohm by 1.00e2 ohm errlmt +-0.04 pc RNG=9,
            range 0.0 ohm to 1.0e6 ohm by 1.00e1 ohm errlmt +-0.01 pc RNG=8,
            range 0.0 ohm to 1.0e5 ohm by 1.0e-0 ohm errlmt +-0.01 pc RNG=6,
            range 0.0 ohm to 1.0e4 ohm by 1.0e-1 ohm errlmt +-0.01 pc RNG=5,
            range 0.0 ohm to 1.0e3 ohm by 1.0e-2 ohm errlmt +-0.01 pc RNG=3,
            range 0.0 ohm to 1.0e2 ohm by 1.0e-3 ohm errlmt +-0.01 pc RNG=2;
         ** - 5 1/2 digits resolution
      }
   end;
   begin FNC=33;
      control
         {
          res range 0.0 ohm to 1.0e8 ohm by 1.00e4 ohm errlmt +-0.81 pc RNG=10,
            range 0.0 ohm to 1.0e7 ohm by 1.00e3 ohm errlmt +-0.04 pc RNG=9,
            range 0.0 ohm to 1.0e6 ohm by 1.00e2 ohm errlmt +-0.01 pc RNG=8,
            range 0.0 ohm to 1.0e5 ohm by 1.00e1 ohm errlmt +-0.01 pc RNG=6,
            range 0.0 ohm to 1.0e4 ohm by 1.0e-0 ohm errlmt +-0.01 pc RNG=5,
            range 0.0 ohm to 1.0e3 ohm by 1.0e-1 ohm errlmt +-0.01 pc RNG=3,
            range 0.0 ohm to 1.0e2 ohm by 1.0e-2 ohm errlmt +-0.01 pc RNG=2;
         ** - 4 1/2 digits resolution
      }
   end;
end; * 2 wires resistance functions

begin; ** current functions
  sensor (current) dc signal;
  fetch DMM_Fetch( current result);

  begin;  ** AMP1 device
         cnx hi AMP1;
     setup DMM_AMP1_Setup( $FNC, current $RNG:0, 2);
     reset DMM_AMP1_Reset();
         begin FNC=12;
            control
            {
         current range -3.0e+0 a to 3.0e+0 a by 1.0e-5 a errlmt +-0.50 pc RNG=4,
                       range -1.0e+0 a to 1.0e+0 a by 1.0e-6 a errlmt +-0.10 pc RNG=3,
                       range -1.0e-1 a to 1.0e-1 a by 1.0e-7 a errlmt +-0.05 pc RNG=2,
                       range -1.0e-2 a to 1.0e-2 a by 1.0e-8 a errlmt +-0.07 pc RNG=1;
            * - 6 1/2 digits resolution
            }
         end;
         begin FNC=22;
            control
            {
         current range -3.0e+0 a to 3.0e+0 a by 1.0e-4 a errlmt +-0.50 pc RNG=4,
                       range -1.0e+0 a to 1.0e+0 a by 1.0e-5 a errlmt +-0.10 pc RNG=3,
                       range -1.0e-1 a to 1.0e-1 a by 1.0e-6 a errlmt +-0.05 pc RNG=2,
                       range -1.0e-2 a to 1.0e-2 a by 1.0e-7 a errlmt +-0.07 pc RNG=1;
            * - 5 1/2 digits resolution
            }
         end;
         begin FNC=32;
            control
            {
         current range -3.0e+0 a to 3.0e+0 a by 1.0e-3 a errlmt +-0.50 pc RNG=4,
```

```
                range -1.0e+0 a to 1.0e+0 a by 1.0e-4 a errlmt +-0.10 pc RNG=3,
                range -1.0e-1 a to 1.0e-1 a by 1.0e-5 a errlmt +-0.05 pc RNG=2,
                range -1.0e-2 a to 1.0e-2 a by 1.0e-6 a errlmt +-0.07 pc RNG=1;
          * - 4 1/2 digits resolution
          }
        end;
  end;  ** AMP1 device
  begin;  ** AMP2 device
        cnx hi AMP2;
    setup DMM_AMP2_Setup( $FNC, current $RNG:0, 2);
    reset DMM_AMP2_Reset();

        begin FNC=12;
          control
          {
       current range -3.0e+0 a to 3.0e+0 a by 1.0e-5 a errlmt +-0.50 pc RNG=4,
                range -1.0e+0 a to 1.0e+0 a by 1.0e-6 a errlmt +-0.10 pc RNG=3,
                range -1.0e-1 a to 1.0e-1 a by 1.0e-7 a errlmt +-0.05 pc RNG=2,
                range -1.0e-2 a to 1.0e-2 a by 1.0e-8 a errlmt +-0.07 pc RNG=1;
          * - 6 1/2 digits resolution
          }
        end;
        begin FNC=22;
          control
          {
       current range -3.0e+0 a to 3.0e+0 a by 1.0e-4 a errlmt +-0.50 pc RNG=4,
                range -1.0e+0 a to 1.0e+0 a by 1.0e-5 a errlmt +-0.10 pc RNG=3,
                range -1.0e-1 a to 1.0e-1 a by 1.0e-6 a errlmt +-0.05 pc RNG=2,
                range -1.0e-2 a to 1.0e-2 a by 1.0e-7 a errlmt +-0.07 pc RNG=1;
          * - 5 1/2 digits resolution
          }
        end;
        begin FNC=32;
          control
          {
       current range -3.0e+0 a to 3.0e+0 a by 1.0e-3 a errlmt +-0.50 pc RNG=4,
                range -1.0e+0 a to 1.0e+0 a by 1.0e-4 a errlmt +-0.10 pc RNG=3,
                range -1.0e-1 a to 1.0e-1 a by 1.0e-5 a errlmt +-0.05 pc RNG=2,
                range -1.0e-2 a to 1.0e-2 a by 1.0e-6 a errlmt +-0.07 pc RNG=1;
          * - 4 1/2 digits resolution
          }
        end;
  end;  ** AMP2 device
 end; ** current functions
end; ** dc & resistance functions - NPLC aplicable

*****************************************

begin; ** ac functions
  control
    {
     freq range 3 hz to 20000 hz   RNG = 1,
        range 20 hz to 20000 hz  RNG = 2,
        range 200 hz to 20000 hz RNG = 3;
     bandwidth range 3 hz to 20000 hz   RNG = 1,
           range 20 hz to 20000 hz  RNG = 2,
```

```
            range 200 hz to 20000 hz RNG = 3;
    }
  begin FNC=15;
            setup DMM_Setup( $FNC, voltage $RNG:0, bandwidth $RNG: freq $RNG: 2);
            fetch DMM_Fetch( voltage result);
            sensor (voltage) ac signal;
            control
            {
             voltage range 0.0 v to 750.0 v by 1.0e-3 v errlmt +-0.1 pc RNG=7,
                        range 0.0 v to 100.0 v by 1.0e-4 v errlmt +-0.1 pc RNG=6,
                        range 0.0 v to 10.00 v by 1.0e-5 v errlmt +-0.1 pc RNG=5,
                        range 0.0 v to 1.000 v by 1.0e-6 v errlmt +-0.1 pc RNG=3,
                        range 0.0 v to 0.100 v by 1.0e-7 v errlmt +-0.1 pc RNG=2;
              ** 6 1/2 digits resolution
            }
  end;
  begin FNC=15;
        setup DMM_Setup( $FNC, voltage-trms $RNG:0, bandwidth $RNG: freq $RNG: 2);
        fetch DMM_Fetch( voltage-trms result);
            sensor (voltage-trms) ac signal;
            control
            {
         voltage-trms range 0.0 v to 750.0 v by 1.0e-3 v errlmt +-0.1 pc RNG=7,
                        range 0.0 v to 100.0 v by 1.0e-4 v errlmt +-0.1 pc RNG=6,
                                range 0.0 v to 10.00 v by 1.0e-5 v errlmt +-0.1 pc RNG=5,
                                range 0.0 v to 1.000 v by 1.0e-6 v errlmt +-0.1 pc RNG=3,
                                range 0.0 v to 0.100 v by 1.0e-7 v errlmt +-0.1 pc RNG=2;
                    ** 6 1/2 digits resolution
            }
  end;
  begin FNC = 16; ** ac current functions
    fetch DMM_Fetch( current result);
    sensor (current) ac signal;
    control
    {
            current range 0 a to 3 a by 1.0e-5 a errlmt +-0.23 pc RNG=4,
                        range 0 a to 1 a by 1.0e-6 a errlmt +-0.16 pc RNG=3;
            freq range 20 hz to 5000 hz;
    }
    begin;  ** AMP1 device
            cnx hi AMP1;
      setup DMM_AMP1_Setup( $FNC, current $RNG:0, bandwidth $RNG: freq $RNG: 2);
      reset DMM_AMP1_Reset();
    end;    ** AMP1 device
    begin;  ** AMP2 device
            cnx hi AMP2;
      setup DMM_AMP2_Setup( $FNC, current $RNG:0, bandwidth $RNG: freq $RNG: 2);
      reset DMM_AMP2_Reset();
    end;    ** AMP2 device
   end; ** ac current functions
end; ** ac functions

end; ** DEV DMM
```

Fişier DeviceDescription.cfg

```
DMM:
        VERIFY:     <VERIFY, (%W%)>: DATA = %
        VERIFY:     <DC SIGNAL>: DC_VOLT = DC_VOLT
        VERIFY:     <AC SIGNAL>: AC_VOLT = AC_VOLT
        VERIFY:     <IMPEDANCE>: IMPEDANCE = IMPEDANCE
        VERIFY:     <VOLTAGE RANGE %W%>: RANGE1 = %
        VERIFY:     <TO %W%>: RANGE2 = %
        VERIFY:     <LL %W%>: LL = LL
        VERIFY:     <UL %W%>: UL = UL
        VERIFY:     <CNX HI %S%LO>: HI = %
        VERIFY:     <LO %S%>: LO = %

        MEASURE:    <INTO %W%>: DATA = %
        MEASURE:    <DC SIGNAL>: DC_VOLT = DC_VOLT
        MEASURE:    <AC SIGNAL>: AC_VOLT = AC_VOLT
        MEASURE:    <IMPEDANCE>: IMPEDANCE = IMPEDANCE
        MEASURE:    <VOLTAGE RANGE %W%>: RANGE1 = %
        MEASURE:    <TO %W%>: RANGE2 = %
        MEASURE:    <LL %W%>: LL = LL
        MEASURE:    <UL %W%>: UL = UL
        MEASURE:    <CNX HI %S%LO>: HI = %
        MEASURE:    <LO %S%>: LO = %
END DMM

DMM_AMP:
        VERIFY:     <VERIFY, (%W%)>: DATA = %
        VERIFY:     <DC SIGNAL>: DC_CURRENT = DC_CURRENT
        VERIFY:     <AC SIGNAL>: AC_CURRENT = AC_CURRENT
        VERIFY:     <VOLTAGE RANGE %W%>: RANGE1 = %
        VERIFY:     <TO %W%>: RANGE2 = %
        VERIFY:     <LL %W%>: LL = LL
        VERIFY:     <UL %W%>: UL = UL
        VERIFY:     <CNX HI %S%LO>: HI = %
        VERIFY:     <LO %S%>: LO = %

        MEASURE:    <INTO %W%>: DATA = %
        MEASURE:    <DC SIGNAL>: DC_VOLT = DC_VOLT
        MEASURE:    <AC SIGNAL>: AC_VOLT = AC_VOLT
        MEASURE:    <VOLTAGE RANGE %W%>: RANGE1 = %
        MEASURE:    <TO %W%>: RANGE2 = %
        MEASURE:    <LL %W%>: LL = LL
        MEASURE:    <UL %W%>: UL = UL
        MEASURE:    <CNX HI %S%LO>: HI = %
        MEASURE:    <LO %S%>: LO = %

END DMM_AMP

PS8:
        APPLY: <DC SIGNAL>: DC_VOLT = DC_VOLT
        APPLY: <VOLTAGE %W%>: VOLTAGE = %
        APPLY: <VOLTAGE RANGE %W%>: RANGE1 = %
        APPLY: <TO %W%>: RANGE2 = %
```

```
        APPLY: <CNX HI %S%LO>: HI = %
        APPLY: <LO %S%>: LO = %

        REMOVE:     <DC SIGNAL>: DC_VOLT = DC_VOLT
        REMOVE:     <VOLTAGE %W%>: VOLTAGE = %
        REMOVE:     <VOLTAGE RANGE %W%>: RANGE1 = %
        REMOVE:     <TO %W%>: RANGE2 = %
        REMOVE:     <CNX HI %S%LO>: HI = %
        REMOVE:     <LO %S%>: LO = %
END PS8

LOAD1
        APPLY: <RES %W%>: RES = %
        APPLY: <RANGE %W%>: RANGE1 = %
        APPLY: <TO %W%>: RANGE2 = %
        APPLY: <CNX HI %S%LO>: HI = %
        APPLY: <LO %S%>: LO = %
        APPLY: <CNX %S%>: HI = %

        REMOVE:     <RES %W%>: RES = %
        REMOVE:     <RANGE %W%>: RANGE1 = %
        REMOVE:     <TO %W%>: RANGE2 = %
        REMOVE:     <CNX %S%>: HI = %

        CONNECT:    <CNX HI %S%LO>: HI = %
        REMOVE:     <LO %S%>: LO = %

END LOAD1

LOAD2
        APPLY: <RES %W%>: RES = %
        APPLY: <RANGE %W%>: RANGE1 = %
        APPLY: <TO %W%>: RANGE2 = %
        APPLY: <CNX HI %S%LO>: HI = %
        APPLY: <LO %S%>: LO = %

        REMOVE:     <RES %W%>: RES = %
        REMOVE:     <RANGE %W%>: RANGE1 = %
        REMOVE:     <TO %W%>: RANGE2 = %
        REMOVE:     <CNX HI %S%LO>: HI = %
        REMOVE:     <LO %S%>: LO = %

END LOAD2

PS4:
        APPLY: <AC SIGNAL>: AC_VOLT = AC_VOLT
        APPLY: <VOLTAGE %W%>: VOLTAGE = %
        APPLY: <FREQ %W%>: FREQ = %
        APPLY: <CURRENT MAX %W%>: RANGE1 = %
        APPLY: <CNX HI %S%LO>: HI = %
        APPLY: <LO %S%>: LO = %

        REMOVE:     <AC SIGNAL>: AC_VOLT = AC_VOLT
        REMOVE:     <VOLTAGE %W%>: VOLTAGE = %
        REMOVE:     <FREQ %W%>: FREQ = %
        REMOVE:     <CURRENT MAX %W%>: RANGE1 = %
```

```
        REMOVE:     <CNX HI %S%LO>: HI = %
        REMOVE:     <LO %S%>: LO = %
END PS4

DIFF_SCOPE:

        VERIFY:     <VERIFY, (%S%)>: DATA = %
        VERIFY:     <VOLTAGE-P-POS>: PEAK_DETECT = POSITIVE_PEAK
        VERIFY:     <VOLTAGE-P-NEG>: PEAK_DETECT = NEGATIVE_PEAK
        VERIFY:     <SQUARE WAVE>: CHANNEL_A = CHANNEL_1
        VERIFY:     <SQUARE WAVE>: CHANNEL_B = CHANNEL_4
        VERIFY:     <MAX %W%>: VERT_RANGE = %
        VERIFY:     <MIN %W%>: VERT_RANGE = %
        VERIFY:     <SQUARE WAVE>: TRIG_SOURCE = CHANNEL_1
        VERIFY:     <SQUARE WAVE>: TRIG_LEVEL = ??
        VERIFY:     <SQUARE WAVE>: TRIG_COUPLING = DC
        VERIFY:     <SQUARE WAVE>: TRIG_LEVEL = EDGE

        VERIFY:     <LL %W%>: LL = LL
        VERIFY:     <UL %W%>: UL = UL
        VERIFY:     <CNX TRUE %S%COMP>: TRUE = %
        VERIFY:     <COMP %S%LO>: COMP = %
        VERIFY:     <LO %S%>: LO = %

END DIFF_SCOPE


COM_1:
        ENABLE, EXCHANGE:  <USING '%S%',>: CONFIG = %
        ENABLE, EXCHANGE:  <CNX HI %S%LO>: HI = %
        ENABLE, EXCHANGE:  <LO %S%>: LO = %

END COM_1

COM_2:
        ENABLE, EXCHANGE:  <USING '%S%',>: CONFIG = %
        ENABLE, EXCHANGE:  <CNX HI %S%LO>: HI = %
        ENABLE, EXCHANGE:  <LO %S%>: LO = %

END COM_2

COM_3:
        ENABLE, DIGITAL:      <CONFIGURATION '%S%'>: CONFIG = %
END COM_3

FLOW:
        CALCULATE:   <CALCULATE, %S%>: EXPRESSION = %
        PERFORM:     <PERFORM, '%S%'>: SUBROUTINE = %
        PERFORM:     <(%nW%)>: ITEM = %
        WAIT FOR:    <WAIT FOR, TIME%S%$>: DELAY = %
        BEGIN, BLOCK:          <BEGIN, BLOCK, '%S%'>: ENTRY = %
        END, BLOCK:  <END, BLOCK, '%S%'>: ENTRY = %
        IF,:      <NOGO>: IF_VAR = NOGO
        IF,:      <EQ>: IF_TEST = EQ
        IF,:      <NOGO>: IF_VALUE = TRUE
        END, IF:        <END, IF>: ENDIF = ?
```

```
        OUTPUT:         <OUTPUT, TEXT>: OUT_DESTIN = stdio
        OUTPUT:         <TEXT, FROM C'%nW%'>: OUTPUT = %
        INPUT, FROM:    <INPUT,>: INPUT =
        INPUT, FROM:    <FROM '%s%'>: FROM = %
        INPUT, FROM:    <INTO '%S%'>: IN_DESTIN = %
        INPUT, TEXT,:   <INTO '%S%' >: DEST = %
        COMPARE:        <COMPARE, '%S%'>: COMP_DATA = %
        COMPARE:        <UL %S% >: COMP_UL = %
        COMPARE:        <LL %S% >: COMP_LL = %
        WHILE:<WHILE, %S% (>: WHILE_COND = %
        WHILE:< (%S%),>: WHILE_VAL = %
        ENABLE, INPUT:          <FROM%S%,>: SOURCE = %
        ENABLE, INPUT:          <VIA '%S%'>: DEST = %
        DISABLE,:       <DISABLE, '%S%'> HANDLER = %
END FLOW:
```

Fişier DeviceModifiers.cfg

```
DMM
        result->data = DATA;
        device->name = DMM;
        device->function = DC_VOLT;
        device->function = AC_VOLT;
        device->function = IMPEDANCE;
        device->range = RANGE1;
        device->range = RANGE2;
        device->resolution = ;
        device->trigger = ;
        tolerance->ul = UL;
        tolerance->ll = LL;
        connection->hi = HI;
        connection->lo = LO;

END DMM

DMM_AMP
        result->data = DATA;

        device->name = DMM_AMP;
        device->function = DC_CURRENT;
        device->function = AC_CURRENT;
        device->range = RANGE1;
        device->range = RANGE2;
        device->resolution = ;
        device->trigger = ;

        tolerance->ul = UL;
        tolerance->ll = LL;

        connection->hi = HI;
        connection->lo = LO;
END DMM_AMP

PS8
        device->name = PS8;
```

```
        device->function = DC_VOLT;
        device->control = VOLTAGE;
        device->range = RANGE1;
        device->range = RANGE2;
        connection->hi = HI;
        connection->lo = LO;
END PS8

LOAD1
        device->name = LOAD1;
        device->data = RES;
        device->range = RANGE1;
        device->range = RANGE2;
        connection->hi = HI;
        connection->lo = LO;

END LOAD1

LOAD2
        device->name = LOAD2;
        device->data = RES;
        device->range = RANGE1;
        device->range = RANGE2;
        connection->hi = HI;
        connection->lo = LO;

END LOAD2

PS4
        device->name = PS4;
        device->function = AC_VOLT;
        device->voltage = VOLTAGE;
        device->freq = FREQ;
        device->current_max = RANGE1;
        connection->hi = HI;
        connection->lo = LO;
END PS4


DIFF_SCOPE

        result->data = DATA;

        device->name = DIFF_SCOPE;
        device->aquisition_type = PEAK_DETECT;
        device->channel1 = CHANNEL_A;
        device->channel2 = CHANNEL_B;
        device->vertical_range = VERT_RANGE;
        device->trig_source = TRIG_SOURCE;
        device->trig_level = TRIG_LEVEL;
        device->trig_coupl = TRIG_COUPLING;
        device->trig_slope = TRIG_SLOPE;
        device->trig_type = TRIG_TYPE;

        tolerance->ul = UL;
        tolerance->ll = LL;
```

```
        connection->true = TRUE;
        connection->comp = COMP;
        connection->lo = LO;

END DIFF_SCOPE

COM_1
        device->name = COM_1;
        config->type = CONFIG;
        connection->hi = HI;
        connection->lo = LO;

END COM_1

COM_2
        device->name = COM_2;
        config->type = CONFIG;
        connection->hi = HI;
        connection->lo = LO;

END COM_2

COM_3
        device->name = COM_3;
        config->type = CONFIG;

END COM_3

FLOW
        expression->data = EXPRESSION;
        subroutine->name = SUBROUTINE;
        parameters->item = ITEM;
        delay->value = DELAY;
        entry->name = ENTRY;
        out_dest->type = OUT_DESTIN;
        data->content = OUTPUT;
        data->type = INPUT;
        source->id = FROM;
        in_dest->id = IN_DESTIN;
        variable->name = IF_VAR;
        condition->type = IF_TEST;
        value->test = IF_VALUE;
        nothing->no = ENDIF;
        data->value = COMP_DATA;
        limit->ul = COMP_UL;
        limit-> = COMP_LL;
        cond->type = WHILE_COND;
        cond->data = WHILE_DATA;
        cond->value = WHILE_VAL;
        source->name = SOURCE;
        destianation->name = DEST;
        handler->name = HANDLER;
END FLOW
```

Fişier Language.cfg

```
VERIFY = verify (result, device, tolerance, connection);
APPLY = apply (device, connection);
REMOVE = remove (device, connection);
MEASURE = measure (result, device, connection);
CONNECT = connect (device, connection);
DISCONNECT = disconnect (device, connection);
DO, EXCHANGE = do_exchange (device, parameters);
IF, = if (variable, condition, test);
END, IF = end_if ();
BEGIN, BLOCK = begin_block (entry);
END, BLOCK = end_block (entry);
OUTPUT = output (out_dest, data);
INPUT, FROM = input (data, source, in_dest);
INPUT, TEXT, = input_text (destination);
INPUT, GO-NOGO = input_gonogo ();
CALCULATE = calculate (expression);
PERFORM = perform (subroutine, parameters);
WAIT FOR = wait (delay);
COMPARE = compare (data, limit);
FINISH = goto_end ();
WHILE = while (cond);
END, WHILE = end_while ();
TERMINATE = terminate ();
REMOVE, ALL = reset_all ();
ENABLE, EXCHANGE = enable_exchange (device, config, connection);
ENABLE, DIGITAL = enable_dig (device, config);
ENABLE, INPUT = enable_input (source, destination);
DISABLE, = close (handler);
COMMENCE, = main ();
```

Fişier station.cfg

```
DMM
DMM_AMP
PS8
PS9
PS4
DIFF_SCOPE

LOAD1
LOAD2
LOAD3

COM_1
COM_2
COM_3
NONE
```

Instrucţie ATLAS din program sursa:

```
050060 VERIFY,(VOLTAGE),DC SIGNAL USING
                              'DC-VOLTMETER',
       UL 28 V LL 24 V,
       VOLTAGE RANGE -100 V TO +100 V,
       CNX HI `J1-1`
            LO `J1-2` $
```

Instrucţie ILS translatată :

050060 verify (dmm (HP344 ->(function = DC_VOLT , range = 100 V, resolution = , trigger =)), val -> (voltage), tol ->(ul = 28 V , ll = 24 V), cnx ->(hi = J1-1' , lo = 'J1-2'));

Listă de unități

1. Ambient Noise Sensor - Boeing
2. Arinc Signal Gateway - Boeing
3. Touch Screen/Lcd Assembly - Boeing
4. Cabin Attendant Handset - Boeing
5. Cabin System Control panel - Boeing
6. Cabin System Management unit - Boeing
7. Environmental Control System Miscellaneous Card - Boeing
8. Master Dim and Test Pwa - Boeing
9. Overhead Electronics Unit - Boeing
10. Overhead Panel Bus Controller - Boeing
11. Passenger Address / Cabin Interphone - Boeing
12. Linear/Monitor Card - Boeing
13. Pre-Regulator Pwa for cardfile power Supply - Boeing
14. Radio Tuning Panel - Boeing
15. Speaker Drive Module - Boeing
16. Seat Electronics Unit - Boeing
17. Warning Electronic - Boeing
18. Flap/Slat Electronics Unit - Boeing
19. Zone Management Unit - Boeing
20. Overhead Panel Interface Card - Boeing
21. Cargo Smoke Detector - Boeing
22. Zone Power Converter - Boeing
23. Rudder Trim Indicator - Boeing
24. Audio Management Unit - Boeing
25. Integrated Refuel Panel - Boeing
26. Entertainment Multiplexer/Controller - Boeing
27. Hydraulic Interface Module - Boeing
28. Logic and Speed Control Unit - Boeing
29. Cabin Pressure Valve Control Unit - Boeing
30. Panel Data Concentrator unit - Boeing
31. Cabin Control Panel - Boeing
32. Css Zone Management Unit - Boeing
33. Artificial Feel Computer – concorde
34. Master Warning Control unit – concorde
35. Aicu Computer Unit – concorde
36. Aitu Computer Unit – concorde
37. Pitch Computer Unit – concorde
38. Azimuth Computer Unit – concorde
39. Auto Stab Computer Unit – concorde
40. Auto-throttle Computer Unit – concorde
41. Nozzle Angle Scheduling unit – concorde

Listă de termeni şi acronime

ATE – Automatic Test Equipment, Echipament de testare automată
TUA – Test Unit Adapter, Interfață de testare
IEEE – Institute of Electrical / Eectronics Engineers
UUT – Unit Under Test, Unitate testată
GPIB – General Purpose Instrument Bus, Magistrala de instrumentație
VME - VersaModule Eurocard BUS, magistrală pentru instrumente modulare deăvoltată în 1980 de Motorola, Signetics, Mostek and şi Thompson CSF.
VXI - VME EXtensions for Instrumentation, extensie a magistralei VME pentru instrumentație
ATLAS – Abreviated Test Language for All Systems, Limbaj de test pentru toate sistemele
TRD – Test Requirement Document, Document care conține documentația de testare
ARINC – Aeronautical Radio, Inc., fondată de companiile aeriene pentru a reglementa inițial comunicația radio, domeniu extins mai târziu.
IVI – Interchangeble Virtual Instrument, Instrument Virtual Interschimbabil
PXI - PCI EXtensions for Instrumentation, extensie a magistralei PCI pentru instrumentație.
DMM – Digital Multimeter, Multimetru Digital
AFG – Arbitrary Function Generator, Generator de funcții arbitrare
DSO – Digital Scope
SCPI - Standard Commands for Programmable Instruments.
J-TAG - Joint Test Action Group, numele uzual pentru standardul IEEE1149.1
TPS – Test Program Set, Set de programe şi fişiere necesare pentru testarea unei unități
DVR – Digital Video Recorder
C/ATLAS - Common/Abbreviated Test Language for All Systems, standardul IEEE 716-1995
WSP – Word Serial Protocol, un protocol care permite comunicarea prin mesaje între modulele dintr-un sertar VXI
SCSI - Small Computer System Interface, o interfață standard paralelă care permite viteze până la 80 Mbytes/sec
IDE - Intelligent Drive Electronics or Integrated Drive Electronics, interfață pentru hard discuri.
PCI - Peripheral Component Interconnect, magistrală între CPU şi periferice
LVDT - Linear-Voltage Differential Transformer , senzor de poziție
UNIXWARE – Sistem de operare UNIX pentr PC-uri
CVI – C for Virtual Instrumentation
TPG – Test Program Generator, aplicație a firmei RADA pentru editarea de programe într-un limbaj propriu, uneori se foloseşte denumirea şi pentru limbaj în sine
B777 – Avionul de tip Boeing 777
PAWS – mediu de testare al companiei americane TYX Corp.
CMM – Component Maintenance Manual , Manual de întrținere pentru o unitate de aviație