FHT

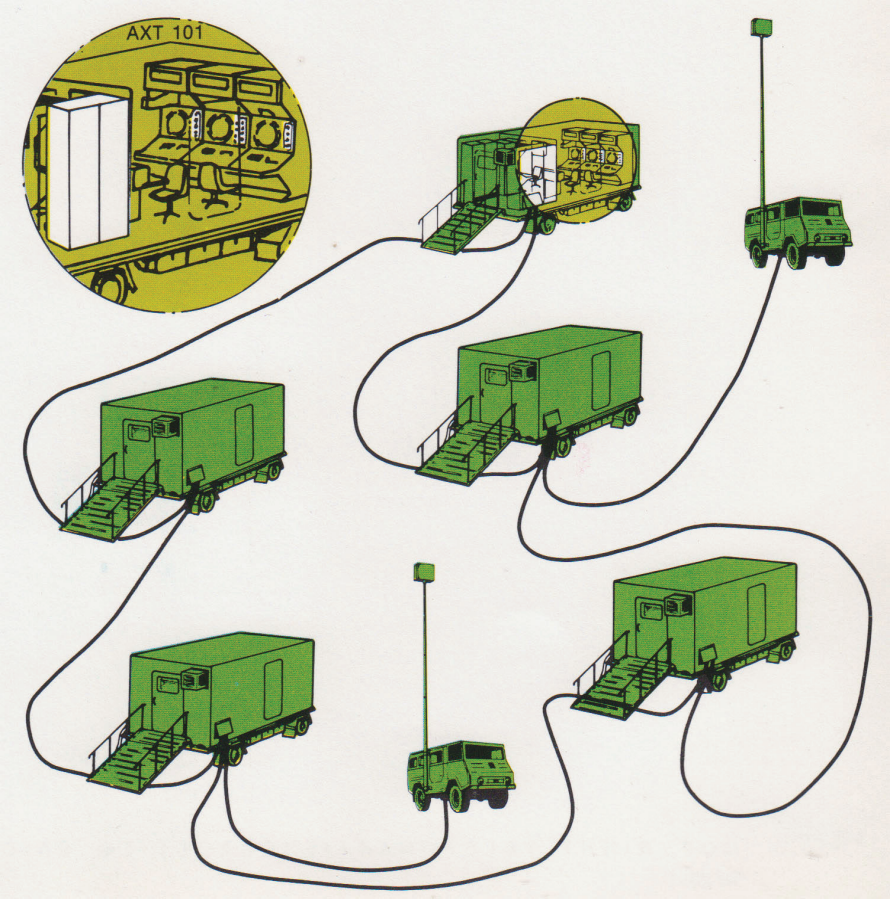
Försvarets Historiska Telesamlingar
Flygvapnet

2016-03-10

DATORSYSTEMET I AXT-VÄXLARNA

Göte Brunberg

F 03/16



Förord

Inför Försvarets Historiska Telesamlingars 30-årsjubileum 2014-09-25 sammanställdes dokumentet *Försvarselektronik från svenska leverantörer, Företagen – produkterna – samhällsnyttan,* som översiktligt redovisar de större elektronikföretagen och deras produkter. I dessa produkter kom datorer och programvara att i allt högre grad svara för funktionaliteten – och för den totala livstidskostnaden. Dator- och programutvecklingen, utvecklingen av programmeringsmetoder och hjälpmedel, liksom kompetensuppbyggnaden i företagen, fick dock inte den framträdande roll i sammanställningen som man kunde önska sig.

Målet för FHT:s verksamhet är dels att dokumentera utrustningar, apparater och system som avvecklats och dels att medverka vid urvalet av vad som ska bevaras vid museer. Hitintills har dock dokumenteringen av program och programproduktion skett i väldigt begränsad omfattning och i stort har ingen programvara (främst källkod) till de avvecklade systemen bevarats (arkiverats vid Krigsarkivet eller de stora museerna).

För att inte värdefull kunskap om den stora utvecklingen inom dator- och programvaruområdet för de teletrafiksystem som anskaffats till försvaret ska falla i glömska togs kontakt med Göte Brunberg, tidigare anställd vid Ericsson, för att om möjligt få hjälp med ”historieskrivningen” vad avser AXT-växlarna. Detta dokument hade svårligen kunnat tas fram utan hans medverkan.

Vi hoppas också att dokumentet kan vara till nytta när man längre fram i tiden vill undersöka hur datorer och programvaror för realtidsapplikationer utvecklades under perioden från 1970 fram till 2000.

Göte Brunberg var som anställd på Ericsson under den större delen av sitt yrkesverksamma liv – i mer än 25 år – engagerad i utvecklingen av AXT-växlarna, då främst abonnentväxlar av typ AXT 101. Han kom med redan 1978 när arbetet med att utveckla den första generationen av växlar för PS 860 och RIR påbörjades. Sedan deltog han i den fortsatta utvecklingen av AXT 101 för leveranser till bland annat radargruppcentraler, luftförsvarscentraler och flygbaser. Efter det att utvecklingsarbetet på AXT 101 var avslutat, i och med leveransen av Tvx 420, deltog han i slutförandet av leveransen av nätväxlar AXT 121.

Göran Kihlström

Innehållsförteckning

1. INLEDNING

I slutet av 1970-talet beställde FMV från Ericsson ett telekommunikationssystem för radarsystem PS-860 och rörliga stridsledningscentraler RIR. Systemet, som av Ericsson gavs produktnamnet AXT 101, skulle, tillsammans med övrig teknisk utrustning och arbetsplatser för operatörer, placeras i transportabla vagnar, ”hyddor”. Systemet skulle ge operatörerna avancerade telefonitjänster, såväl vid interna samtal som vid samtal med externa operatörer. Transmissionen skulle vara digital, baserad på PCM, och tjänsterna skulle realiseras i programvara i ett datorsystem. Tekniska och taktiska tjänster skulle administreras via inmatning från bildskärmsterminal.

De fanns tre olika typer av hyddor: ”radarhydda”, ”telehydda” och ”operatörshydda”. Systemrealiseringen innebar att i varje hydda placerades en växel, som i sig innehöll all funktionalitet som krävdes för att hyddan skulle kunna tjänstgöra som en självständigt arbetande enhet. En avancerad egenskap var att hyddorna kunde kopplas ihop till ett större system, exempelvis bestående av en radarhydda och två operatörshyddor. Detta realiserades, som visas i bilden på framsidan, genom att växlarna kopplades ihop med varandra i en ring med 30/32 kanalers PCM-system. På detta sätt skapades ett större system, en ”konfiguration”, som sett från operativ synpunkt, men även från drift- och underhållssynpunkt, kunde betraktas och hanteras som ett enda system. Operatörerna kunde placera sig på godtycklig arbetsplats i konfigurationen. De externa förbindelserna var en gemensam resurs som kunde utnyttjas av samtliga operatörer. Ett bortfall av en växel påverkade endast den hydda där växeln var placerad.

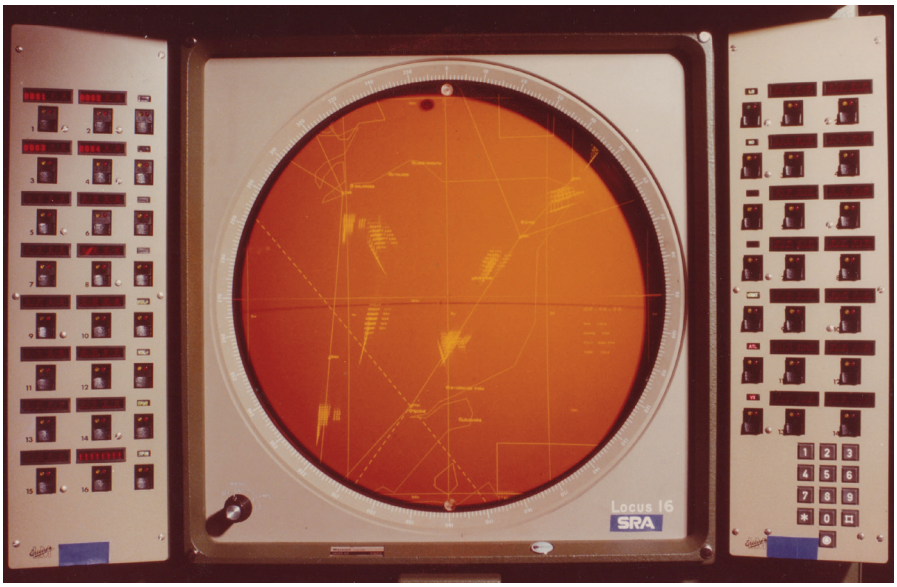
I varje växel fanns ett datorsystem, som bestod av en central processor och ett antal regionala processorer, som var placerade i direkt anslutning till enheterna i växelns taltransmissionsdel. Systemets ”intelligens” var realiserad i programmet i centralprocessorn, som kommunicerade med regionalprocessorerna via ett buss-system.

Programmet i centralprocessorn laddades vid uppstart in från en bandspelare. Därefter angav man från en bildskärmsterminal vilka växlar som ingick i konfigurationen. Detta resulterade i att förberedda data, ”stationsdata”, för den aktuella konfigurationen laddades in från bandspelaren.

En grunddel i stationsdatat var anläggningsdatat, som angav för datorsystemet hur växeln i den aktuella hyddtypen var maskinvarumässigt bestyckad. På en högre nivå låg tekniska och taktiska stationsdata, som kunde kompletteras och ändras under drift genom inmatning i menyer, ”tablåer”, från en bildskärm i godtycklig hydda i anläggningen. Dataändringarna spreds till de övriga växlarna, så att samtliga hyddor hade identiska data, gällande för hela konfigurationen.

För att ge operatörerna avancerade samtalstjänster fanns operatörspaneler, som var utformade på olika sätt för att täcka behovet av tjänster på den specifika platsen. I figur 1 visas den typ av panel som användes på en radaroperatörsplats i RIR: en ”vänster”-panel och en ”höger”-panel på ömse sidor om radarskärmen. Panelerna var försedda med omkastare med alfanumeriska displayer, som visade den aktuella motabonnentens identitet. Operatörerna kunde placera sig på en godtycklig plats och genom inloggning få ”sina” förbindelser knutna till platsen.

Den första beställningen av AXT-växlar följdes under ett antal år av beställningar för andra typer av anläggningar, som Rrgc, Lfc och flygbaser. I samband med de sista beställningarna, för Tvx 420 och växlar i ATL-nätet, skedde en uppgradering av datorsystemet med en kraftfullare centralprocessor och ett modernare I/O-system för program- och stationsdatahantering.



Figur 1. Operatörsplats i RIR-hydda.

2. AXT-SYSTEMET, ÖVERSIKTLIG SYSTEMSTRUKTUR

Systembasering på AXE 10

Allmänt

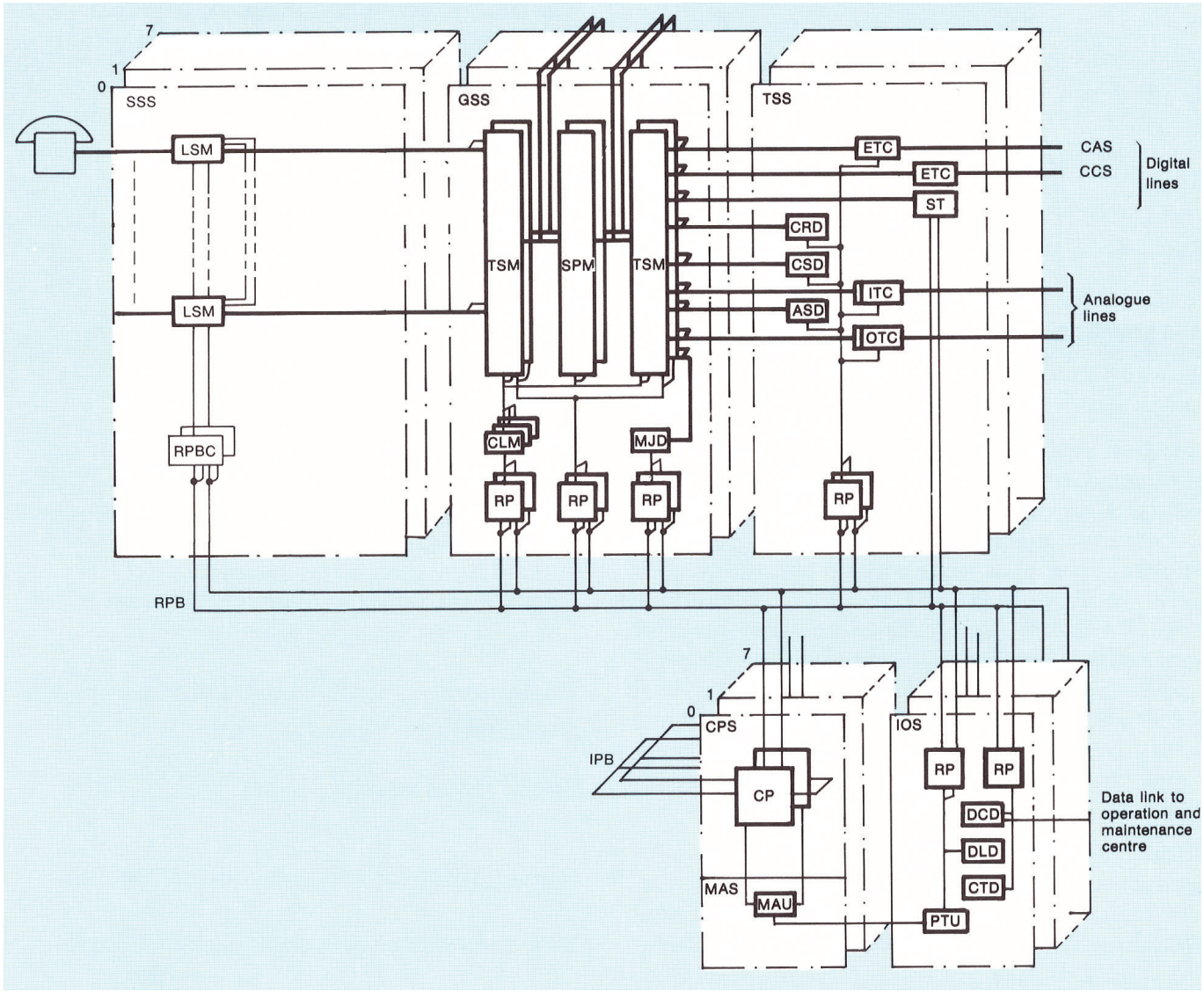
Telekommunikationssystemet för PS-860/RIR, som av Ericsson gavs produktnamnet AXT 101 01 (i fortsättningen benämnt AXT), baserades på maskinvara från Ericssons publika digitala telefonväxelsystem AXE 10. Detta fick en fundamental inverkan på utformningen av AXT-systemet, inte bara på växeldelen, ”kopplingssystemet”, utan även på datorsystemet, ”styrsystemet”. Den styrsystemhierarki som användes i AXE 10, med centrala och regionala processorer, tvingade fram en liknande hierarki i AXT. Detta motiverar en översiktlig beskrivning av AXE 10, som bakgrund till utformningen av AXT.

Kopplingssystemet

Kopplingssystemet i AXE 10 var PCM-orienterat, vilket innebar att systemet hanterade grupper om 32 kanaler, där för varje kanal tal (eller data) kopplades igenom med hastigheten 64 kbit/s.

Den centrala komponenten i kopplingssystemet var ett digitalt väljarsystem GSS (se figur 2). Detta bestod av ett antal spärrfria väljarenheter TSM med 512 ingångar (16 grupper med vardera 32 kanaler). För att skapa väljare med högre kapacitet – upp till tiotusentals abonnenter – kopplades ett erforderligt antal TSM ihop med varandra via en rumsväljare SPM. En klockenhet CLM styrde takten i dataöverföringen genom väljaren och mot anslutna länkar. För att öka tillförlitligheten var väljarsystemet dubblerat i två plan, A och B. Klockenheten CLM var triplerad.

För anpassning mot analoga linjer användes terminalenheter LSM, till vilka kunde anslutas 128 abon-



Figur 2. AXE 10, maskinvarustruktur.

nenter. Anslutning av PCM-system med kanalassocierad signalering skedde via terminalenheter ETCA och PCM-system med gemensam kanalsignalering enligt CCITT No. 7 via ETCC. I GSS ingick också en digital konferensenhet MJC-D.

Styrsystemet

Den totala systemfunktionaliteten i AXE 10 var – i tillägg till maskinvaran i kopplingssystemet – realiserad i programvara i styrsystemet. Detta var baserat på en hierarki med två typer av datorer (på Ericssonspråk kallade ”processorer”): en (eller flera) dubblerade centralprocessorer CP och ett antal regionala processorer RP, även de dubblerade. De analyserande och beslutsfattande uppgifterna låg hos CP. RP skötte de mer repetitiva uppgifterna, som att upptäcka tillståndsförändringar i linjer som var anslutna till växeln och att ändra tillståndet i dessa på order från CP. Kommunikationen mellan CP och RP i de styrda enheterna, med ett samlingsbegrepp kallade utökningsmoduler EM, skedde via två dubblerade buss-system: RP-bussen RPB mellan CP och RP och EM-bussen mellan RP och enheterna (se figurerna 2 och 4). Gränssnittet i enheterna var ett kretskort EMC-6.

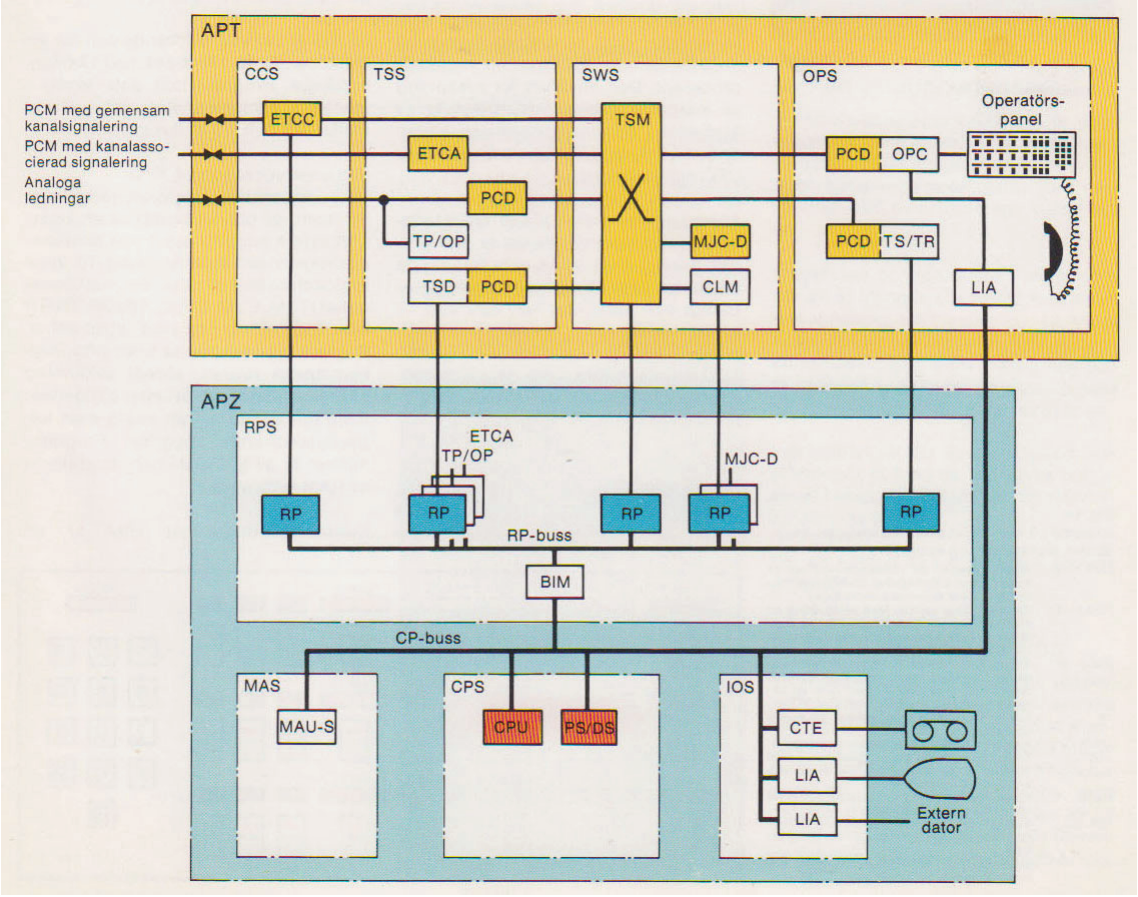
Vid utvecklingen av AXE 10 skapade man en funktionell systemstruktur, som på den högsta nivån innehöll styrsystemet APZ 210 och kopplingssystemet APT 210. I styrsystemet ingick maskinvarumässigt CP och RP med tillhörande bussystem, och dessutom I/O-system och system för övervakning av processorerna och deras funktion. I kopplingssystemet ingick maskinvarumässigt det som gjorde AXE 10 till en telefonväxel, nämligen väljardelen med anslutna enheter.

Det nya i betraktelsesättet var att de trafikala programmen, fastän de fysiskt låg i styrsystemet, ändå i den funktionella strukturen betraktades som tillhörande kopplingssystemet. Styrsystemet med dess maskinvara och programvara utgjorde endast den miljö där programvaran i kopplingssystemet var placerad.

I både APZ och APT ingick ett antal delsystem som i sin tur bestod av funktionella enheter, moduler. En modul kunde innehålla maskinvara, central programvara och regional programvara.

Tillämpning i AXT

För tillämpningen i AXT valdes som väljar- och anpassningssystem mot analoga och digitala linjer delar ur AXE 10:s väljarsystem enligt figur 3. Kapaciteten hos en enda väljarenhet TSM var tillräcklig, varför det inte krävdes någon rumsväljare SPM. Som anpassning mot externa PCM-system med kanalassocierad signalering och gemensam kanalsignalering användes samma enhe-



Figur 3. AXT 101 01, maskinvarustruktur (ur Ericsson Rewiew, nr 1 1980).

ter, ETCA respektive ETCC, som i AXE 10. Även klockenheten CLM och konferensenheten MJC-D hämtades från AXE 10. För anpassning mot sextrådiga analoga linjer användes PCD för taltransmission och TP/OP för signalering.

Styrsystemet i AXE 10 var på alla sätt (utrymmes-, kostnads- och kapacitetsmässigt) överdimensionerat för tillämpning i AXT. Som centralprocessor CP valdes istället en minidator APN 163, som hade utvecklats inom Ericsson och redan användes i andra produkter. Systemstrukturen förenklades på så sätt att EM-bussen slopades och på EMC-6-kortets plats i de styrda enheterna placerades istället en enkorts RP. Den hade utvecklats inom ett pilotprojekt i AXE 10, som hade till syfte att ta fram en enklare systemstruktur för – i detta sammanhang – små abonnentväxlar. Som anpassning mellan APN 163 och den nya typen av RP-buss konstruerades en bussanpassningsenhet BIM. Detta gav en styrsystemstruktur i AXT enligt figur 4.

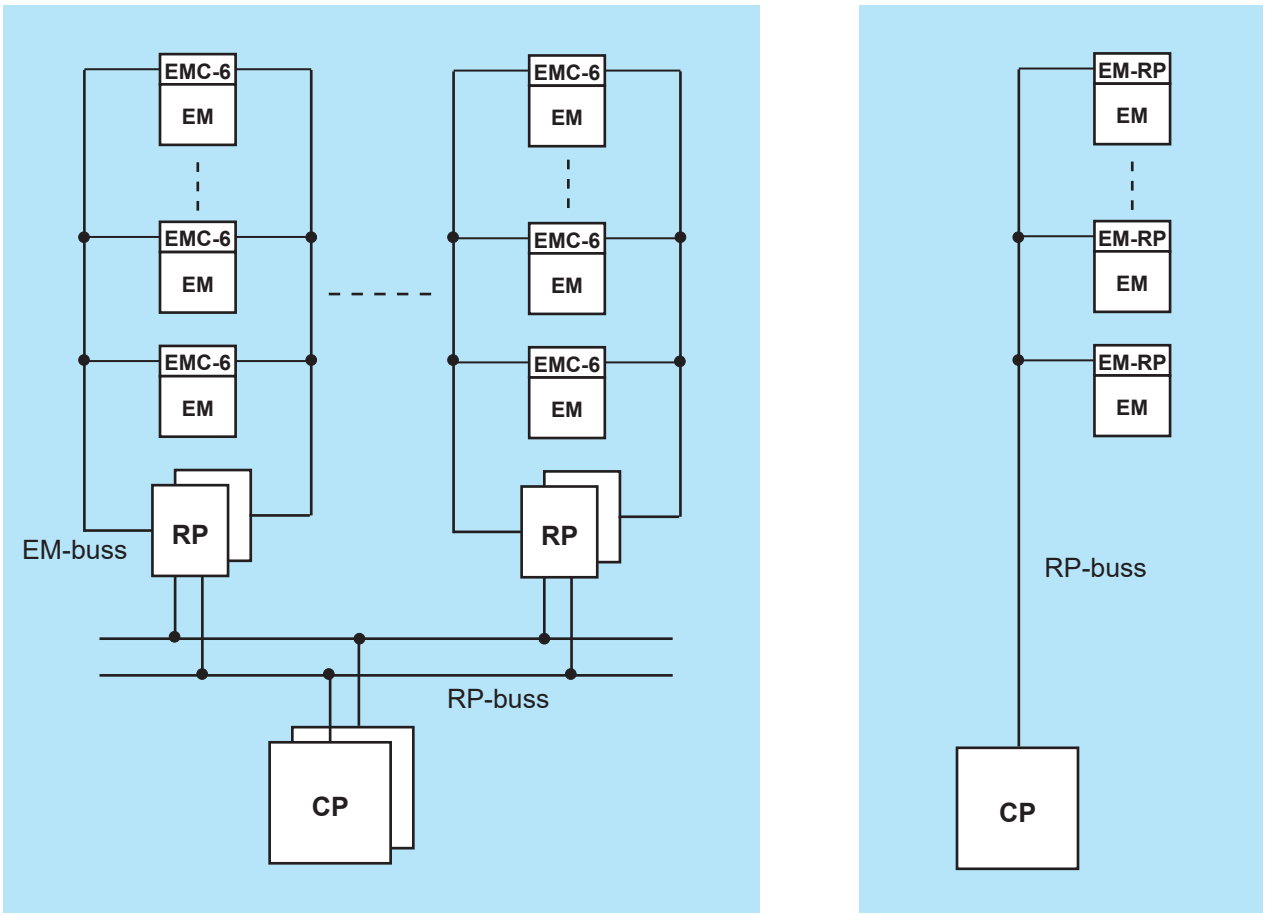
”Tänket” med APZ- och APT-systemen följde med från AXE 10 till AXT. Detta gällde även den modulära uppbyggnaden i delsystem inom varje system. I styrsystemet, med produktnamnet APZ 101, ingick centralprocessordelsystem CPS, regionalprocessordelsystem RPS, övervakningsdelsystem MAS och in/utmatningsdelsystem IOS. Kopplingssystemet fick namnet APT 101. Det beskrivs inte i detta dokument.

Vidareutveckling av AXT

Den första leveransen av AXT följdes under de följande åren av leveranser av olika typer av system till bland annat Rrgc, Lfc och flygbaser. Denna systemfamilj hade samlingsnamnet AXT 101. Systemuppbyggnaden användes även i nätväxlar av typ AXT 121, som levererades till ATL.

I takt med att AXE 10-systemet utvecklades skedde en kontinuerlig modernisering och miniatyrisering av de ingående maskinvaruenheterna. De nya produkterna fördes då in i AXT 101 och AXT 121 i samband med nybeställningar.

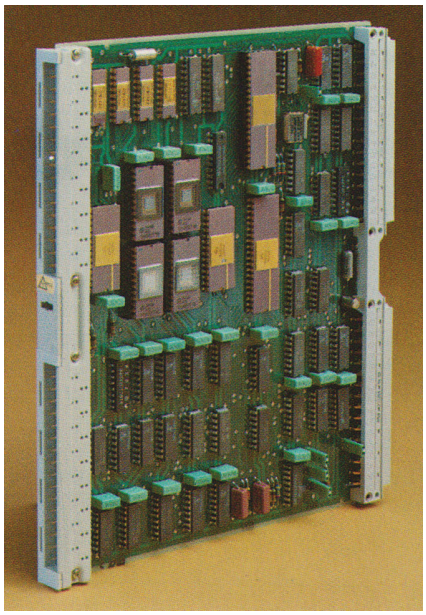
Ett stort utvecklingssteg togs för styrsystemet i samband med beställningen av AXT 101 för tillämpning i Tvx 420 och av AXT 121 för modernisering av ATL-nätet. För dessa system infördes ett nytt styrsystem, där centralprocessor APN 163 ersattes med APN 167, som hade högre avverkningskapacitet och modernare I/O-system. Den överordnade systemstrukturen var oförändrad, varför processorbytet till övervägande endast påverkade styrsystemet självt.



Figur 4. Styrsystemstruktur i AXE 10 och AXT.

3. BYGGSÄTT

Mekaniskt var ett AXT-system byggt med komponenter ur Ericssons ”standard”-byggsätt BYB 101, som togs fram i samband med utvecklingen av AXE-systemet. Detta gällde både för styrsystemet och kopplingssystemet. Styrsystemet var alltså mekaniskt integrerat i växelsystemet och inte en ”burk vid sidan om”.



Kretskort

Magasin

En funktionsenhet bestod mekaniskt av ett eller flera kretskort. Dessa var placerade i magasin som i sin tur var monterade i stativ.

Enheterna förbands med varandra med kontaktdonsförsedda kablar som anslöts till fronten på kretskorten. Beroende av trådantalet fanns det ”kvartsdons”-, ”halvdons”-, ”trekvartsdons-” och ”heldons”-kablar. Endast för interna förbindelser mellan korten i en och samma enhet användes trådar i magasinets bakplan (ofta användes även kablar för detta). Kablarna var ”trådräta”, vilket innebar att tillverkningen kunde standardiseras. När ett flertal enheter skulle anslutas till ett gemensamt buss-system skedde detta genom multiplicering: kabeln till nästa enhet anslöts till fronten av den inkommande kabeln.

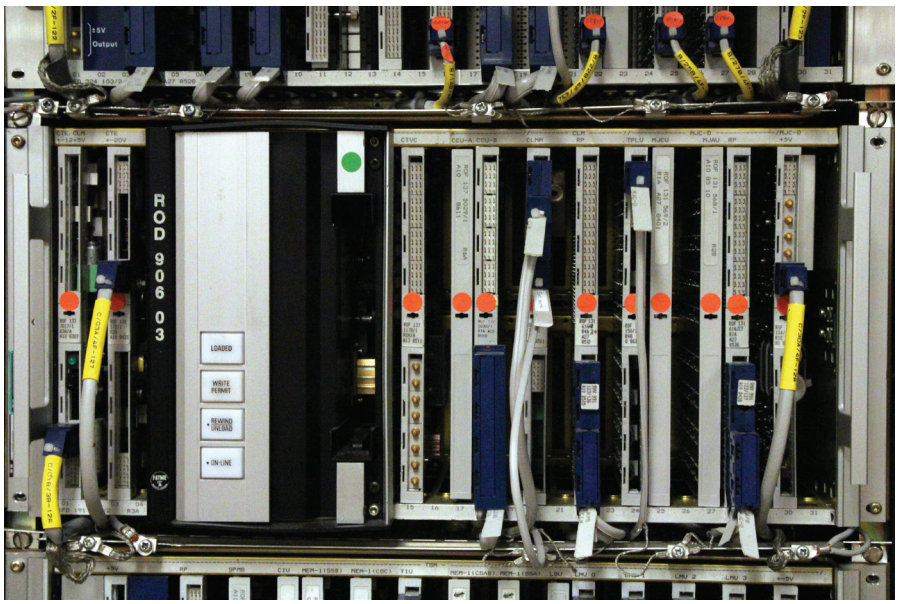
Kretskorten, som hade typbeteckningen ROF 13, var 222 mm höga och 178 mm djupa. Korten hade kontaktdon i bakkant för förbindning mot magasinets bakplan. På fronten fanns kontaktdon av olika utföranden för anslutning av kablar. I kontaktdonen kunde även sättas ”byglingsproppar”. Dessa användes bland annat till att ge varje enhet en unik identitet i systemet. Exempelvis tilldelades på detta sätt varje regionalprocessor en adress på RP-bussen.

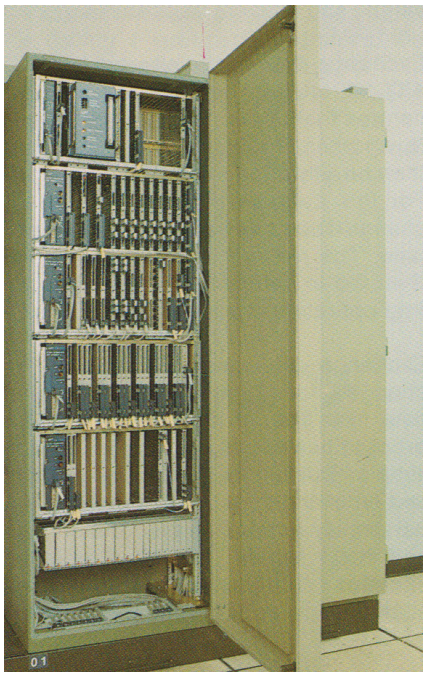
På fronten kunde, i tillägg till kontaktdon. även placeras knappar, omkastare och lysdioder.

Magasinen hade måtten 244 x 488 x 220 mm (höjd x bredd x djup) och rymde ett varierande antal kretskort, beroende av kretskortens komponenthöjd och effektutveckling. Det fanns även magasin med dubbel höjd. I magasinen fanns spänningsomvandlare för omvandling av -48 V till erforderliga spänningar till enheterna.

Magasinen monterades i stativ. Dessa placerades på anläggningarna i skåp med forcerad kylning. Vid krav på att utrustningen skulle vara skyddad mot elektromagnetisk strålning (EMC), som i PS-860/ RIR, användes en speciell typ av skåp enligt figur 5.

Skåp





Figur 5. Byggsätt.

4. STYRSYSTEM APZ 101 MED CENTRALPROCESSOR APN 163

Struktur

Styrsystemet APZ 101 var indelat i fyra delsystem. I varje delsystem ingick maskinvara enligt figur 3 och central programvara, i regionalprocessordelsystemet även regional programvara.

- Centralprocessordelsystem CPS innehöll som central del en processor av typ APN 163, som hade utvecklats speciellt för realtidsapplikationer. De analyserande och beslutsfattande funktionerna i programvaran för styrsystemet och kopplingssystemet exekverades i CPS.
- Regionalprocessordelsystem RPS. De rutinmässiga, kapacitetskrävande funktionerna i programvaran för kopplingssystemet exekverades i RPS, som innehöll en regionalprocessor RP för varje styrd enhet.

  Regionalprocessorerna var via RP-bussen RPB och en anpassningsenhet BIM anslutna till centralprocessorn.
- I/O-delsystemet IOS hanterade man-maskinkommunikationen och in- och utmatning av program och data. Som lagringsmedium för program och semipermanenta data (”stationsdata”) användes kassettband av så kallad 3M-typ.
- Underhållsdelsystemet MAS hade till uppgift att övervaka APZ 101 och APT 101 och vidta åtgärder i händelse av fel. I MAS ingick även funktioner för start och omstart av det totala systemet.

Som tidigare nämnts bestod CPS mekaniskt av kretskort som var placerade i magasin. Figur 13 visar som exempel ett magasin som innehåller större delen av styrsystemet. Endast kassettbandspelaren med styrlogik saknas.

Centralprocessordelsystem CPS

Allmänt

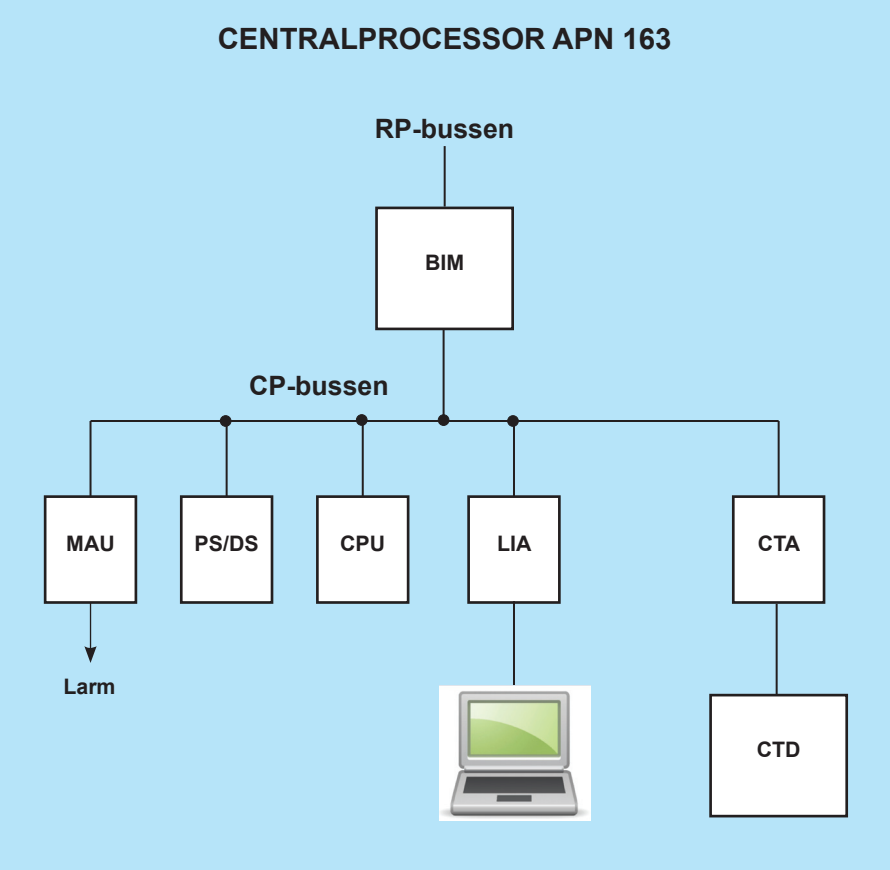
CPS bestod av maskinvarumässigt av centralprocessorenheten CPU med programminne PS och dataminne DS. Fysiskt låg PS och DS i samma skriv- och läsbara minne. CPU kommunicerade med PS/DS och perifera enheter i andra delsystem i APZ 101 (och även några enheter i kopplingssystemet APT 101) via ett bussystem, CP-bussen. Se figur 6.

Centralprocessorn var händelsestyrd, vilket innebar att den normalt låg i ett vilotillstånd. Detta tillstånd kunde avbrytas, ”interrupteras”, av en enhet som var ansluten till CP-bussen. Detta ledde till att en – mer eller mindre omfattande – programsekvens för hantering av avbrottet exekverades. När detta var klart återgick CPU till vilotillståndet i väntan på nästa interrupt.

Programavverkningen i centralprocessorn administrerades av ett exekutivsystem, EXEC 163. Programmen var skrivna i PL 163, ett programspråk som kombinerade effektiviteten hos kod skriven i assembler med satser hämtade från högnivåspråk.

CP-bussen

Bussen var asynkron och dubbelriktad. Den innehöll en adressdel och en datadel, vardera med 16 trådar, och en styrdel med 14 trådar. Det normala tillståndet på bussen var att den användes av CPU för läsning och skrivning i PS/DS och övriga anslutna enheter, men det fanns även två speciella moder: interrupt och direkt bussaccess DBA.



Figur 6. CPS med centralprocessor APN 163.

Interrupt användes när en händelse i växeln krävde bearbetning i CP, exempelvis att BIM hade tagit emot ett meddelande från en RP. BIM sände då en interruptbegäran till CPU. Detta resulterade i att exekutivsystemet avbröt eventuell pågående programexekvering och hämtade meddelandet. Det bearbetades av en interruptprocedur som vidarebefordrade meddelandet som en signal till exekutivsystemet. I och med detta var hanteringen av själva interrupten avslutad.

Varje enhet som var ansluten till CP-bussen hade en unik identitet som bestämdes av en byglingspropp. På så sätt anropades en speciell interruptprocedur för varje typ av enhet.

Med hjälp av kretskort BTX-L och BTX-E kunde CP-bussen förlängas till andra magasin än det där CPU var placerad.

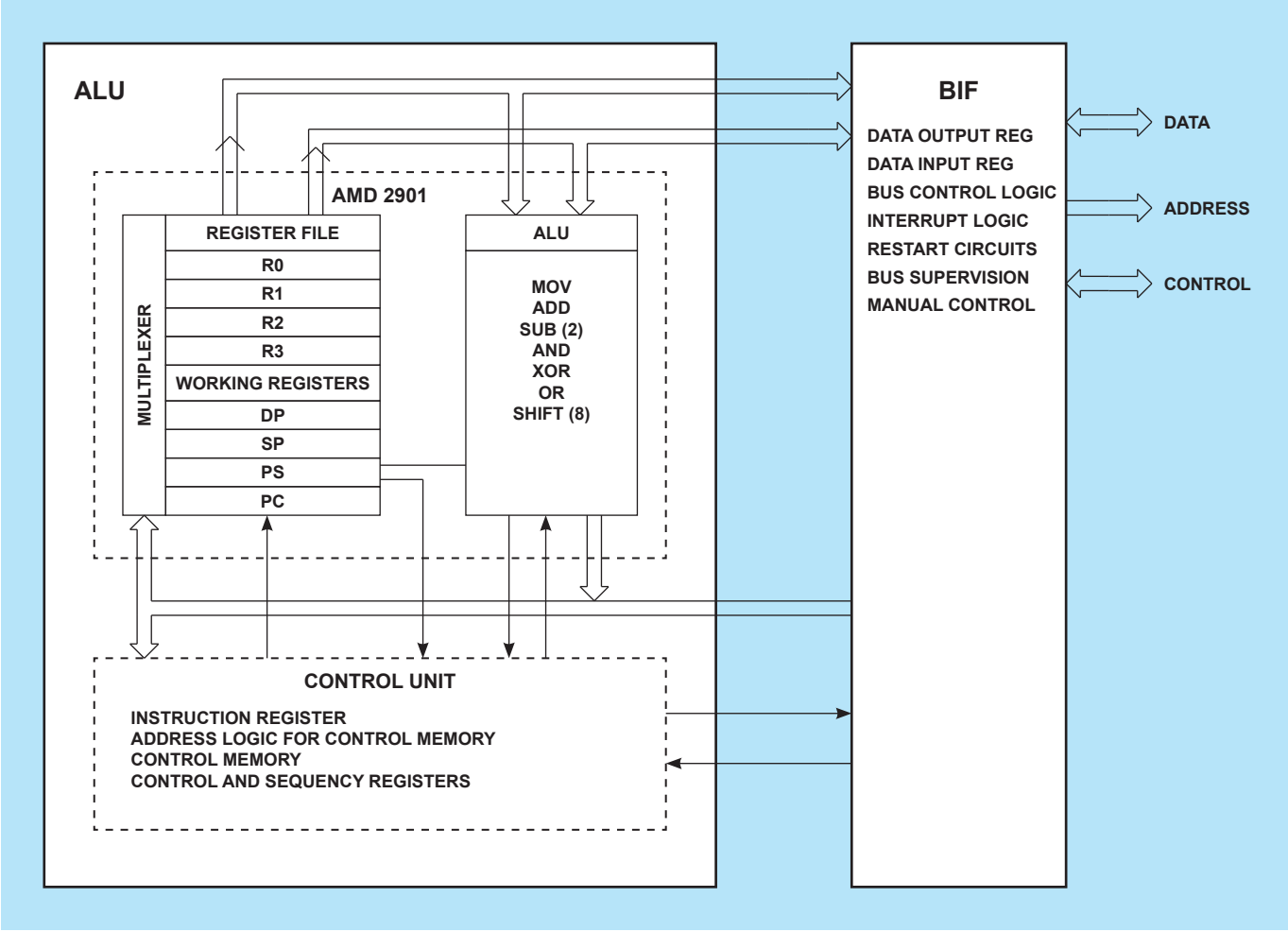
Direkt bussaccess DBA användes av kassettbandspelarstyrenheten CTA för överföring av data mellan enheten och centralprocessorminnet utan medverkan av CPU.

Centralprocessor APN 163

APN 163 bestod av två kretskort: aritmetisk/logisk enhet ALU och bussanpassningsenhet BIF. Se figur 7. Processorn hanterade adresser och data med 16 bitars bredd.

ALU utförde databehandling på operander som hade hämtats från interna register och CP-bussen (minnet eller perifera enheter). Enhetens grundstomme var en så kallad bit-sliceprocessor av typ AMD 2901. Denna processortyp användes även av tillverkare av minidatorer, bland andra i PDP-11 från Digital Equipment. Processorkretsen, som kunde utföra enkla aritmetiska och logiska operationer på fyra bitars data, seriekopplades i APN 163 till 16 bitars bredd. Den innehöll även 16 register.

Bit-sliceprocessorns instruktionsrepertoar var, som framgår under ALU i figur 7, ganska begränsad. För att CPU skulle kunna hantera programinstruktioner av typ ”kopiera innehållet i den adress i centralprocessorminnet som pekas ut av register R1 till adressen A i minnet” krävdes en påbyggnad i form av en kontrollenhet som utvidgade instruktionsrepertoaren. Kontrollenheten innehöll ett mikroinstruktionsminne med storleken 1024 ord x 24 bitar, ett instruktionsregister med avkodare samt styr- och sekvensregister. Kontrollenheten initierade att CPU hämtade en



Figur 7. Centralprocessor APN 163, blockschema.

instruktion från programminnet. Instruktionen avkodades varefter kontrollenheten startade en av instruktionen beroende mikroinstruktionssekvens, som styrde exekveringen av instruktionen i de olika delarna av CPU. De allra enklaste instruktionerna omfattade endast ett ord i programminnet men i de flesta fall omfattade de två eller flera ord.

Av registren i ALU var R0-R3 allmänt användbara register som kunde adresseras i programmet. Dessutom fanns ett antal register som användes av exekutivsystemet EXEC 163. Systemstackpekare SP och datastackpekare DP pekade mot motsvarande stack i dataminnet (se vidare avsnittet Exekvering av processer). Programstatusregister PS innehöll bland annat ett antal statusindikatorer som sattes beroende av resultatet av operationerna i ALU. Programräknare PC innehöll adressen till nästa instruktion som skulle exekveras.

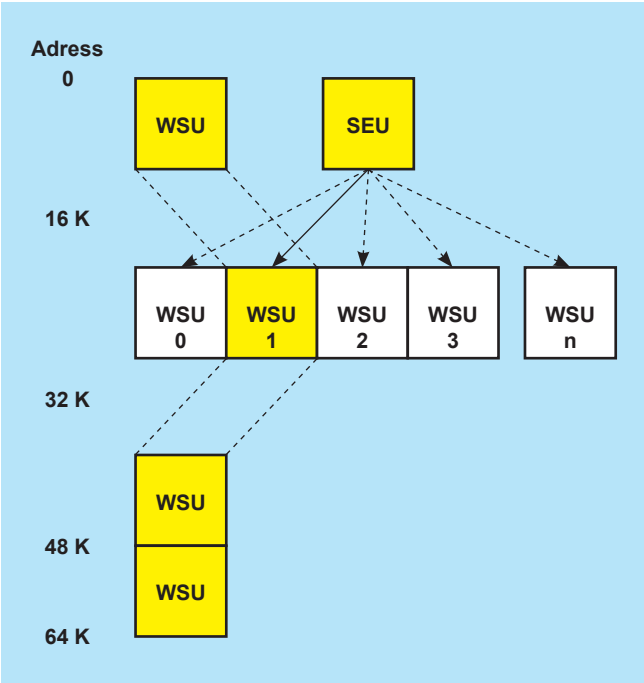
Det totala antalet instruktionstyper var 58. Operationer kunde ske på hela ord, ordfält, bitfält och enstaka bitar. Tiden för att avverka en instruktion var, beroende av komplexiteten, 1,5-4,0 mikrosekunder.

BIF var den del av CPU som hanterade CP-bussen. Den innehöll register för in- och utdata och kretsar för styrning av bussen. På BIF fanns även ett antal knappar för manuell kontroll av CPU (start/stopp och mikroinstruktionsstegning).

Programminne PS och dataminne DS

PS och DS låg i en gemensam minnesenhet WSU, som var ansluten till CP-bussen. Eftersom CPU arbetade med 16-bitars ord var adresseringskapaciteten endast 64 kord (1 kord = 1 K = 1 024 ord), men genom användning av en minnesutökningsfunktion kunde den totala minnesvolymen utökas till 1024 kord.

Fysiskt bestod WSU av ett eller flera kretskort. I den första leveransen av AXT 101 innehöll varje kretskort 16 kord skriv- och läsbart minne, men i takt med att det kom fram minneskretsar med större kapacitet rymde minneskorten mer och mer så att i den sista leveransen av AXT 101 med APN 163 bestod minnet av ett enda kretskort med 1 024 kord, WSU1024K.



Figur 8. Principen för minnesutökning.

Utökningen av minnesområdet från 64 kord till 1024 kord utfördes med hjälp av en minnesutökningsenhet SEU. Den var till en början placerad på ett eget kretskort men var i WSU 1024K placerad på minneskortet självt.

I figur 8 visas hur det tillgängliga adressområdet 0–64 kord disponerades och principen för minnesutökningen. För området 0–16K (hexadecimalt 0000-3FFF) och 32K–64K (#8000-FFFF), kallat ”gemensamt minne”, pekade adressen entydigt på ett ord i minnet. Inom området 16K–32K (#4000-7FFF) pekades en adress ut i en av upp till 60 alternativa sidor i minnet (”utökningsmoduler”). Exekutivsystemet bestämde vilken sida som skulle adresseras genom att sidnumret skrevs in i register i SEU.

Det gemensamma minnet var en dyrbar resurs och användes därför i första hand för lagring av program och data som tillhörde exekutivsystemet. Applikationsprogrammen i kopplingssystemet APT 101, med tillhörande data, låg till övervägande del på utökningsmoduler.

För att skydda de delar av minnet som innehöll program och semipermantenta data (”stationsdata”) mot obehörig skrivning gick det att skrivskydda minnet i steg om 4 kord.

Den högsta delen av adressområdet, #FC00-FFFF, var reserverad för de perifera enheterna på CP-bussen. Dessa kunde därför adresseras av CPU på samma sätt som minnet.

Programspråk PL 163

I samband med grundutvecklingen av APN 163 skapades en programutvecklingsmiljö på värddatorer av typ VAX/VMS. Programmen skrevs i assemblerkod, som i värddatorn översattes av en assembler till maskinkod för att göra den körbar i CPU.

Program som är skrivna i assemb-

lerkod genererar effektiv maskinkod men är besvärliga att strukturera, överblicka och underhålla. I en grupp inom Ericsson, som sysslade med utveckling av programspråk, fick man därför idén att utveckla ett så kallat högnivåassemblerspråk, PL 163, för APN 163. Inspirationskällan var programspråket PL360 för IBM/360-datorer, som hade utvecklats av Niklaus Wirth, skaparen av programspråket Pascal.

PL 163 innehöll som grund assemblerinstruktionerna, men uttryckta på ett mer lättförståeligt sätt, och kombinerade detta med med möjligheten hos ett högnivåspråk att skriva program som är väl strukturerade och därför lätta att prova och underhålla. PL 163 innehöll därför satser av typ IF ... THEN ... ELSE, WHILE ... DO och CASE OF .... Dessutom ingick i språket ett antal satser för anrop av exekutivsystemet EXEC 163, bland andra SEND och WAIT för signalsändning respektive signalmottagning och GETFREE och PUTFREE för att begära och lämna tillbaka buffertar i minnet.

Programstruktur

En väsentlig egenskap hos ett högnivåspråk är att det är modulärt, vilket innebär att det går att skapa ett helt programsystem av mindre programsegment, som vart och ett utför en speciell uppgift. I PL 163 ingick därför fyra typer av programsegment som var ordnade i en hierarki enligt figur 9.

Processer (PROCESS) och procedurer (PROCEDURE) kunde betraktas som ”huvudprogram” respektive ”subrutiner”. De innehöll programkod och data, de senare endast tillgängliga för processen eller proceduren själv.

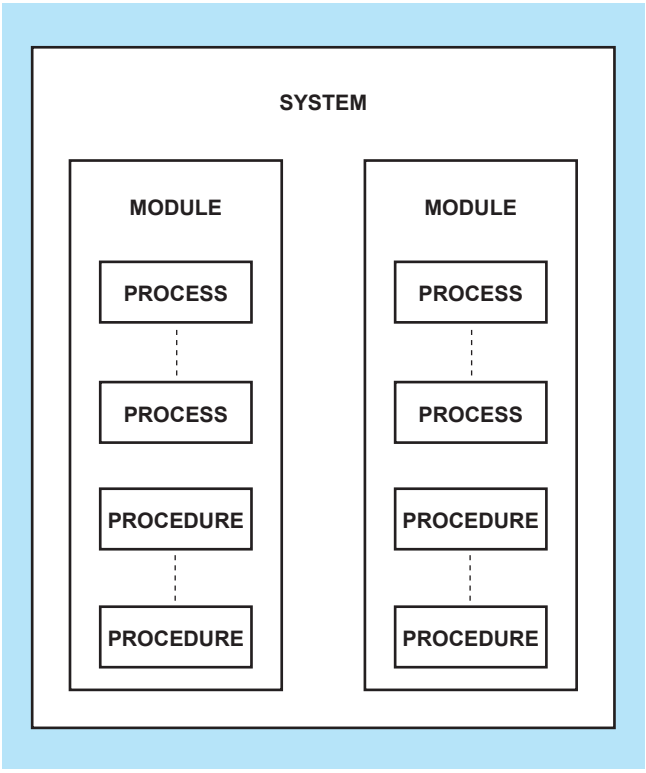
EXEC 163 var baserat på exekvering av processer. Dessa fick tillgång till CPU enligt den prioritet som de var tilldelade. Processerna kommunicerade med varandra genom sändning och mottagning av signaler. Se vidare avsnittet Exekutivsystem EXEC 163.

En process eller procedur innehöll programvara (program, statiska data och variabla data) för en viss funktion. Processer och procedurer, som tillsammans bildade en överordnad funktion, samlades i en modul (MODULE). Modulen innehöll data, som endast var tillgängliga från processer och procedurer inom modulen.

Högst upp i programhierarkin låg systemsegmentet SYSTEM. Detta innehöll data som var tillgängliga från alla processer och procedurer i det totala programsystemet.

På alla nivåer fanns också möjlighet att deklarera konstanter, till vilka referenser kunde göras i underliggande segment.

I kopplingssystemet i AXT tillämpades programstrukturen på så sätt att telefonibegrepp som ”linjeanpassning”, ”samtalsuppkoppling”, ”nummeranalys”, ”samtalsövervakning” och ”väljarstyrning” realiserades i form av moduler. Dessa innehöll (i de flesta fall) en process och ett antal procedurer. Exempelvis hanterades telefonabonnentlinjer av modulen BTKR, som innehöll processen KRCALL och ett antal procedurer för bland annat hantering av anrop, siffermottagning och tonsändning.



Figur 9. PL 163, programstruktur.

Exekutivsystem EXEC 163

Programavverkningen i CPU administrerades av exekutivsystemet EXEC 163. Detta var optimerat för tillämpning i realtidssystem, där yttre händelser som kräver bearbetning i programsystemet uppträder slumpmässigt och ibland sker samtidigt. Eftersom centralprocessorn är en flaskhals i systemet måste någon form av prioritering ske i hanteringen av uppgifterna. Detta skedde genom att huvudprogrammen, processerna, tilldelades olika prioriteter för att ges tillgång till CPU. EXEC 163 gav, när flera uppgifter samtidigt krävde bearbetning, i varje läge den process som hade den högsta prioritetet tillgång till CPU.

Den centrala delen i EXEC 163 var

modulen EXEC, som hanterade kommunikationen mellan processerna, tidmätning och minneshantering. För varje typ av anrop till exekutivsystemet innehöll EXEC en procedur, exempelvis XSEND och XWAIT för hantering av SEND respektive WAIT.

Exekvering av processer

I de flesta programmodulerna ingick en process, som var huvudprogrammet för realiseringen av modulens funktion. Till en process hörde programkod, statiska data och variabla data.

Vid systemuppstart skapade EXEC 163 för varje process i systemet en så kallad processinstans. Denna bestod av en databuffert, vars huvudsakliga delar var processhuvudet och system- och datastackarna. Processhuvudet innehöll data som användes av exekutivsystemet. Systemstacken användes för tillfällig lagring av bland annat innehållet i registren i samband med interrupt och vid subrutinanrop. Datastacken användes för överföring av data vid subrutinanrop.

Varje processinstans kunde i vissa avseenden betraktas som om den vore en egen, ”virtuell”, processor med full tillgång till CPU. I verkligheten kunde naturligtvis programkod i endast en instans i taget exekveras. EXEC 163 innehöll därför en funktion för prioritering av i vilken ordning som detta skulle ske. I programkoden för processerna fanns en prioritetsnivå 0–15 deklarerad. Om flera instanser med samma prioritetsnivå krävde exekvering, ordnades de i en kö för varje prioritetsnivå.

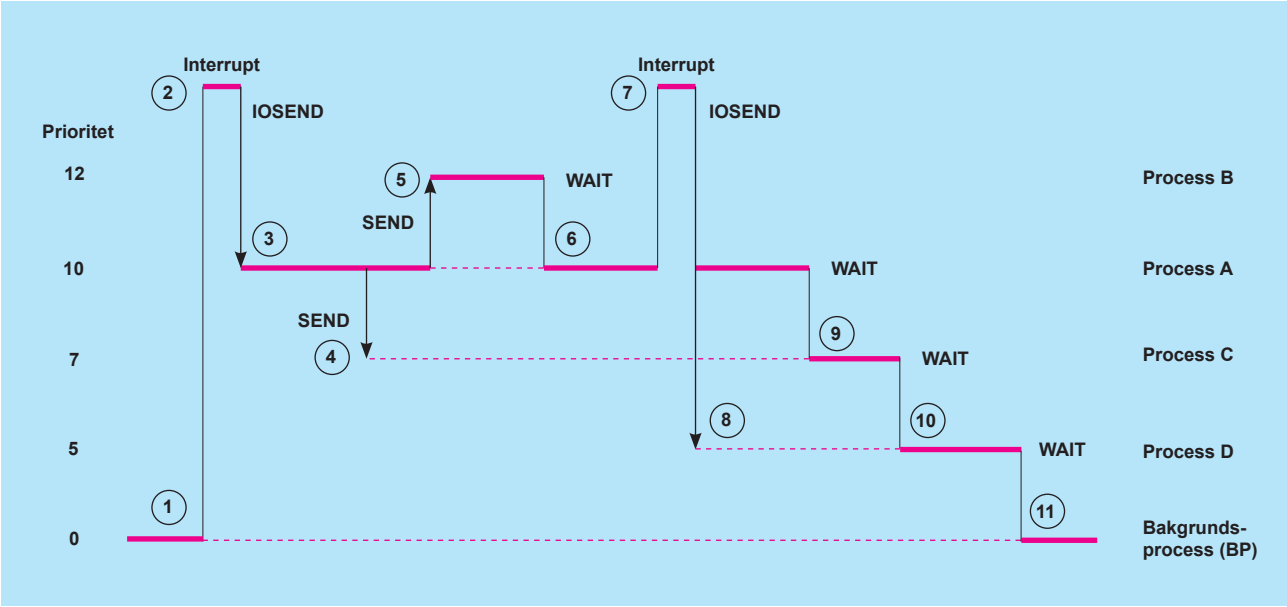
Exekveringen av en process kunde avbrytas, exempelvis av ett interrupt eller av att en process med högre prioritet kallades in. Då sparades registerinnehåll, programpekare och system- och datastackpekare undan i aktiveringsposten. Dessa data plockades sedan fram när processen åter skulle börja exekveras.

Processkommunikation

Processerna samarbetade med varandra genom sändning och mottagning av signaler på så kallade signalbanor. Rent konkret var en signal en databuffert av varierande storlek och en signalbana en dataarea med tre ord. Mot en signalbana kunde länkas adresser till signaler och processinstanser.

En process avslutade exekveringen och lade sig i ”viloläge” på en signalbana med instruktionen WAIT. Detta resulterade i att innehållet i, bland annat, programräknare och register hämtades från CPU och lagrades i systemstacken, varefter adressen till instansen lades in i signalbanan. När processen åter skulle börja exekveras laddades dessa data tillbaka till CPU.

En process, som var under exe-



1. CPU i viloläge. Bakgrundsprocessen (BP) exekveras.
2. Processen avbryts av interrupt.
3. Interruptproceduren sänder signal till process A, som har högre priortet. BP sätts på väntelistan på nivå 0 och A börjar exekveras.
4. A sänder signal till C, som har lägre prioritet än A och sätts på väntelistan på nivå 7.
5. A sänder signal till B, som har högre prioritet. A sätts på väntelistan på nivå 10 och B börjar exekveras.
6. B avslutas. A, som har högst prioritet av de väntande processerna, fortsätter att exekveras.
7. A avbryts av interrupt.
8. Interruptproceduren sänder signal till D, som har lägre prioritet än A och sätts på väntelistan på nivå 5.
9. A avslutas. C, som har högst prioritet av de väntande processerna, exekveras.
10. C avslutas. D exekveras.
11. D avslutas. BP börjar åter exekveras.

Figur 10. Principen för processexekvering i EXEC 163.

kvering, kunde med instruktionen SEND sända signaler på signalbanor till andra processer. Även interruptprocedurer kunde sända signaler.

En typisk sekvens i exekveringen av en process var *vänteläge – signalmottagning – bearbetning – signalsändning med SEND – fortsatt bearbetning – vänteläge med WAIT*.

Prioriteringsfunktionen trädde in i samband med signalsändningen. Om den sändande processen hade samma prioritet – eller högre – än den väntande fortsatte den att exekveras. I det fall att den mottagande processen hade högre prioritet avbröts exekveringen av den sändande processen och processinstansen länkades till kön för prioritetsnivån. Detsamma gällde om exekveringen av en process avbröts av att en interruptprocedur gjorde en IOSEND.

En konsekvens av principen med prioritering var att i ett givet ögonblick var det alltid den processinstans som hade den högsta prioriteten som exekverades.

I figur 10 visas principen för hur EXEC 163 hanterade interrupt och prioritering av processer på olika prioritetsnivåer.

Implementering av exekutivanropen

Exekutivanropen SEND, WAIT, GETFREE och PUTFREE var de mest frekvent förekommande i applikationsprogrammen, både i programkoden och vid programexekveringen, men det fanns totalt över tjugo typer av exekutivanrop. Metoden för att implementera dessa var att de vid kompilering av PL 163-koden översattes till en instruktion TRAP(xx), där xx var en för anropet gällande vektor. Exempelvis översattes WAIT till TRAP(54) och SEND till TRAP(60). Vid exekveringen av TRAP-instruktionen initierades en ”mjukvaru”-interrupt av CPU.

Programutveckling

Utvecklingsmiljö

Utvecklingsmiljön för APN 163 fanns i datorer av typ VAX/VMS. Den innehöll funktioner för editering och kompilering av enskilda programsegment och uppbyggnad av programsystem. Under de sista åren av AXT-utvecklingen, när tillverkningen av VAX-datorerna hade lagts ned av leverantören, användes en VAX/VMS-emulator av typ Charon.

Under den första tiden av utvecklingsarbetet för AXT 101 01 var stödet för programutveckling ganska primitivt. Koden skrevs för hand på stansunderlag som lämnades till Ericssons dataavdelning för stansning av hålkort, som via en hålkortsläsare lästes in i utvecklingssystemet. Senare kom möjligheten att skriva in koden direkt i utvecklingssystemet, först med en radeditor och senare med en fullskärmseditor. Som ordbehandlingsprogram användes det av Ericsson utvecklade EXCO.

Kompilering av PL 163-koden

”Byggklotsarna” i ett programsystem för en viss applikation var modulerna (MODULE). Varje modul hade en unik identifikation, bestående av ett produktnummer och ett revisionsläge, exempelvis CAA 113 54 R3A. Därmed kunde flera moduler utvecklas parallellt i utvecklingsmiljön. Programkoden för modulen med ingående processer, procedurer och data skrevs i PL 163. Vid kompileringen av modulen översattes PL 163-instruktionerna till assemblerkod. Interna referenser inom modulen, exempelvis när en procedur refererade till data som var deklarerade i modulen, löstes upp. Externa referenser, exempelvis till SYSTEM-segmentet, fick symboliska namn. Dessutom upptäcktes syntaktiska fel i programkoden, till exempel rena felstavningar eller att interna referenser i modulen saknades eller var felaktiga.

Resultatet av kompileringen blev en flyttbar (”relokerbar”) objektmodul.

Systemgenerering

Resultatet av genereringen av ett programsystem var en laddmodul med exekverbar maskinkod som från en bandspelare laddades in i centralprocessorn.

Systemgenereringen började ofta (alltid när det gällde system som skulle levereras) med att samtliga i systemet ingående programmoduler kompilerades. Programsystemet hade en identifikation, bestående av ett produktnummer och ett revisionsläge, exempelvis LZY 207 1023/1 R1A. Vilka moduler som skulle kompileras framgick av en modullista, där produktnumret och revisionsläget var angivet för varje modul. Sedan länkades modulerna ihop till programsystemet. Detta styrdes av en länklista, som i detalj beskrev hur varje i programsystemet ingående komponent (modul, process, procedur) skulle placeras i centralprocessorminnet (i ”gemensamt minne” eller i en utökningsmodul).

Resultatet av länkningen blev en så kallad ”total” laddmodul med exekverbar maskinkod. Den matades ut till ett kassettband som vid systemstart av en anläggning laddades in i centralprocessorn från en bandspelare. Se vidare avsnittet I/O-delsystem IOS.

Länkningen av ett helt programsystem var en resurskrävande process i värddatorn i utvecklingssystemet och kunde ta lång tid. Under utveckling och utprovning av ett programsystem användes därför ofta ett enklare sätt att byta ut enstaka programmoduler. Detta utfördes genom att en speciell typ av objektmodul skapades vid kompileringen. I denna var de externa referenserna angivna på ett sådant sätt att de kunde lösas upp av ett program i centralprocessorn, LOADER, när modulen laddades in i systemet.

Regionalprocessordelsystem RPS

Allmänt

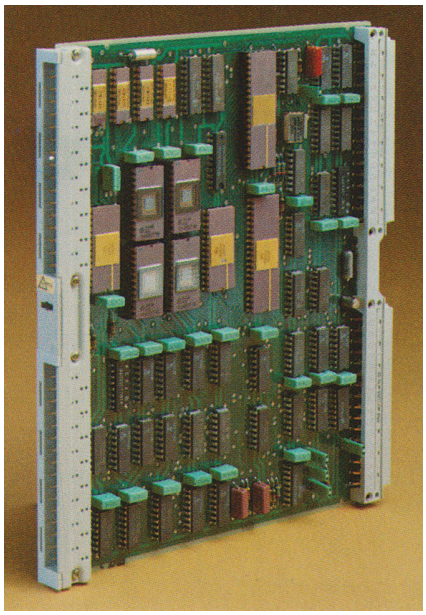
Rutinmässiga, kapacitetskrävande programvarufunktioner i kopplingssystemet APT 101 var realiserade i programvara i RPS, som kunde betraktas som en ”förlängning” av maskinvaran i kopplingssystemet. RPS innehöll en regionalprocessor RP per styrd enhet i kopplingssystemet. Regionalprocessorerna kommunicerade med centralprocessorn via RP-bussen och anpassningsenheten BIM, som var ansluten till CP-bussen.

RP-bussen

RP-bussen var en balanserad likströmsbuss med fyra trådpar. Överföringshastigheten var 250 kbit/s. Dataöverföringen skedde i form av meddelanden enligt det så kallade HDLC-protokollet. Varje RP hade ett unikt nummer, som bestämdes av en byglingspropp i fronten på kretskortet.

Regionalprocessor RP

Regionalprocessorn, som tillhörde Ericssons produktfamilj APN 165, bestod av ett kretskort (se figur 11) och var baserad på kretsar ur Motorolas mikroprocessorfamilj MC6800. Dess kärna var centralenheten MPU med programminne PS och dataminne DS.



Figur 11. Regionalprocessor RP.

Programminnet PS hade storleken 8 kord x 8 bitar och var realiserat i EPROM (programmerbart minne som kan raderas med UV-ljus). Detta innebar att PS inte förlorade sitt innehåll vid spänningsbortfall. Dataminnet DS hade storleken 2 kord x 8 bitar och var skriv- och läsbart.

Det fanns även en nyare RP-variant, som innehöll 16 kord programminne och 4 kord dataminne.

Programavverkningen i RP styrdes av ett exekutivsystem, EXEC 165. Detta innehöll funktioner för programinkallning, tidmätning och hantering av gränssnittet mot RP-bussen och den styrda enheten.

Bussanpassninqsenhet BIM

BIM var anpassningsenhet mellan centralprocessorn CP och RP-bussen och utförde följande:

- Mottagning av meddelanden från CP och överföring av meddelandena till avsedd RP enligt det protokoll som gällde på RP-bussen.
- Avsökning av RP för att upptäcka om dessa hade meddelanden att sända till CP. Om så var fallet, hämtade BIM meddelandet, varefter CP interrupterades.

BIM var baserad på kretsar ur mikroprocessorfamiljen MC6800 och bestod av en anpassningsdel mot CP-bussen, en anpassningsdel mot RP-bussen och en styrdel.

Anpassningsdelen mot RP-bussen innehöll kretsar för elektrisk och signalmässig anpassning mot bussen. Från BIM utgick två bussgrenar, som elektriskt var åtskilda men signalmässigt ihopslagna. Till varje bussgren kunde anslutas upp till 32 RP. Den totala kapaciteten för en BIM var alltså 64 RP.

Styrdelen i BIM administrerade meddelandesändning och -mottagning mot RP-bussen och CPU.

I/O-delsystem IOS

Allmänt

IOS innehöll maskinvara och central programvara för in- och utmatning av program och semipermanenta data (”stationsdata”) och för bildskärmskommunikation.

För lagring av program och stationsdata användes kassettdrivenheten CTD med tillhörande styrenhet CTE. För kommunikation med bildskärrnsterrninalerna användes terminalanpassningsenheter LIA.

Kassettbandspelaranpassningsenhet CTE

CTE utförde anpassning mellan CPU och en kassettdrivenhet CTD för så kallade 3M-kassetter (DC300A). Kassetterna användes för lagring av program och semipermanenta data (”stationsdata”).

CTE var mekaniskt uppdelad i två delar: En del, innehållande kretskortet CTA, var placerad i CP-magasinet medan övrig utrustning var placerad i samma magasin som drivenheten CTD (se figur 12). CTA var ansluten till CP-bussen.

Program och stationsdata lagrades på bandet i form av block, där varje block kunde innehålla upp till 2048 ord om 8 bitar. Till varje block hörde en checksumma som genererades vid inskrivning och kontrollerades vid utläsning. Överföringshastigheten till och från bandet var 48 kbit/s.

Överföringen av data mellan centralprocessorn och bandet skedde block för block. Datat i ett block som lästes från bandet lagrades temporärt i en buffert i CTE. Det överfördes till CPU med användning direkt bussaccess, DBA, på CP-bussen. Denna initierades genom att CTE, genom en signal på en styrtråd i CP-bussen, begärde att få tillgång till bussen. När detta beviljades övertog CTE kontrollen över bussen och kopierade datat i bufferten, ord för ord, till en area i centralprocessorminnet. När överföringen

var klar interrupterades CPU för att initiera bearbetningen av datat.

Vid skrivning av data på bandet användes i princip samma metod som vid läsning.

På kassettbandet fanns fyra spår. Programmet var lagrat på det första och det fjärde spåret. Vid tillslag av spänningen till systemet sände övervakningsenheten MAU-S i MAS en signal till CTE. Signalen initierade att CTE läste ut det första blocket på det första spåret och placerade innehållet i programminnet med början på adressen #8000. Blocket innehöll programmodulen BOOT, som sedan övertog den fortsatta inladdningen av programmet från kassettbandet till minnet.

Stationsdatahantering

I AXT-systemen fanns en inbyggd möjlighet att konfigurera växeln för olika driftfall. Detta möjliggjordes genom att det fanns semipermanenta data, ”stationsdata”, som bestämde hur den enskilda växeln (och även en total konfiguration bestående av flera växlar) skulle fungera i det aktuella driftläget. Som exempel på stationsdata kan nämnas abonnentdata, ledningsdata, viadata och nummeranalysdata. Dessa data laddades in i centralprocessorns minne vid uppstart.

På kassettbandets andra spår fanns ett antal filer och på bandets tredje spår en fil för lagring av stationsdata. Normalt utfördes vid systemuppstart och -omstart laddning av stationsdata från filen på det tredje spåret (”återstartdata”). Det var dock möjligt att med kommando från en bildskärm istället ladda in stationsdatat från någon av filerna på det andra spåret (”nystartdata”). Detta utnyttjades bland annat för att sätta anläggningen i olika operativa drifttillstånd.

Figur 12. CTD med styrlogik.

Efter det att en stationsdataändring hade utförts i centralprocessorminnet, var det möjligt att mata ut stationsdatat från minnet till filen på kassettbandets tredje spår. Detta beordrades manuellt genom användning av en meny på bildskärmen men kunde även initieras av en signal från MAU-S när enheten hade detekterat låg batterispänninge till växeln.

Terminalanpassningsenhet LIA

LIA hanterade kommunikationen mellan CPU och periferiutrustningar via seriella dataförbindelser.

I LIA utfördes parallell-serieomvandling av data från CPU till periferiutrustningen och serie-parallellomvandling av data från periferiutrustningen till CPU. Överföring av data mellan CPU och LIA skedde tecken för tecken och initierades genom interrupt från LIA varje gång ett tecken hade sänts eller tagits emot.

Den primära användningen av LIA var som anpassning mot växelns bildskärm för drift och underhåll, men i AXT 101 användes enheten även för kommunikation med operatörspanelerna. Mot bildskärmen användes gränssnitt enligt CCITT V.24/V.28 mot operatörsplatsutrustningarna med strömslinga.

Vid behov av att koppla om en LIA-enhet mellan två bildskärmar användes ett kretskort V24-MUX.

Underhållsdelsystem MAS

Allmänt

MAS övervakade centralprocessordelsystemet CPS och initierade vid feltillstånd återstart av centralprocessorn. I MAS ingick även funktioner för lagring och presentation av feltillstånd i APZ 101 och APT 101 samt sändning av larm.

I MAS ingick maskinvaruenheten MAU-S och central programvara för insamling och lagring av felrapporter, styrning av larmsändning och beordring av systemåterstart.

Övervakninqsenhet MAU-S

MAU-S utförde bland annat följande:

- ”Watch-dog”-övervakning av programavverkningen i CP. I enheten fanns två tidsövervakningskretsar, en med en kort tid (1 s) och en annan med en längre (10 s). Kretsarna nollställdes av CP-programmets lågprioriterade bakgrundsprocess, när denna exekverades. Om tiden löpte ut indikerade detta ett fel i programavverkningen, exempelvis att programmet löpte i en slinga. Ifall den korta tidsövervakningen löpte ut beordrade MAU-S så kallad liten återstart av växeln. Denna innebar att programsystemet återstartades utan program- och stationsdataladdning. Om detta inte åtgärdade felet löpte den långa tidsövervakningen ut. Då beordrade MAU-S, genom en signal till kassettbandspelarenheten CTE, total omstart av växeln med programladdning från kassettbandet.
- Sändning av larm. Tre olika larmklasser, A, B och C, kunde sändas. Dessutom sändes så kallat akustiskt larm när ett nytt fel inträffade i någon enhet i växeln.
- Identifiering av växelns kod. Med en byglingspropp på fronten av kretskortet gavs växeln en bokstavskod (A, B, C osv).

Koden användes i tillämpningar när flera växlar kopplades ihop till att bestämma växelns plats i gruppen.

- Val mellan manuell och automatisk start av växeln. Detta styrdes av en omkastare MAN/AUT på fronten av kretskortet.

  Vid manuell start kunde man, via bildskärmen, synkronisera uppstartprocessen i växlarna i en konfiguration. När samtliga växlar hade rapporterat att de hade laddat in programmet, beordrades inmatning av stationsdata från en av filerna. När detta var klart i samtliga växlar beordrades systemet att gå över till trafikal drift.

  Vid automatisk återstart skedde ingen synkronisering. Stationsdata laddades från återstartfilen varefter varje växel för sig gick över till trafikal drift.

- Manuell omstart av centralprocessorn. På fronten av kretskortet fanns en tryckknapp för beordring av omstart. En kort tryckning resulterade i liten omstart och en längre (mer än en sekund) i stor återstart med programladdning.

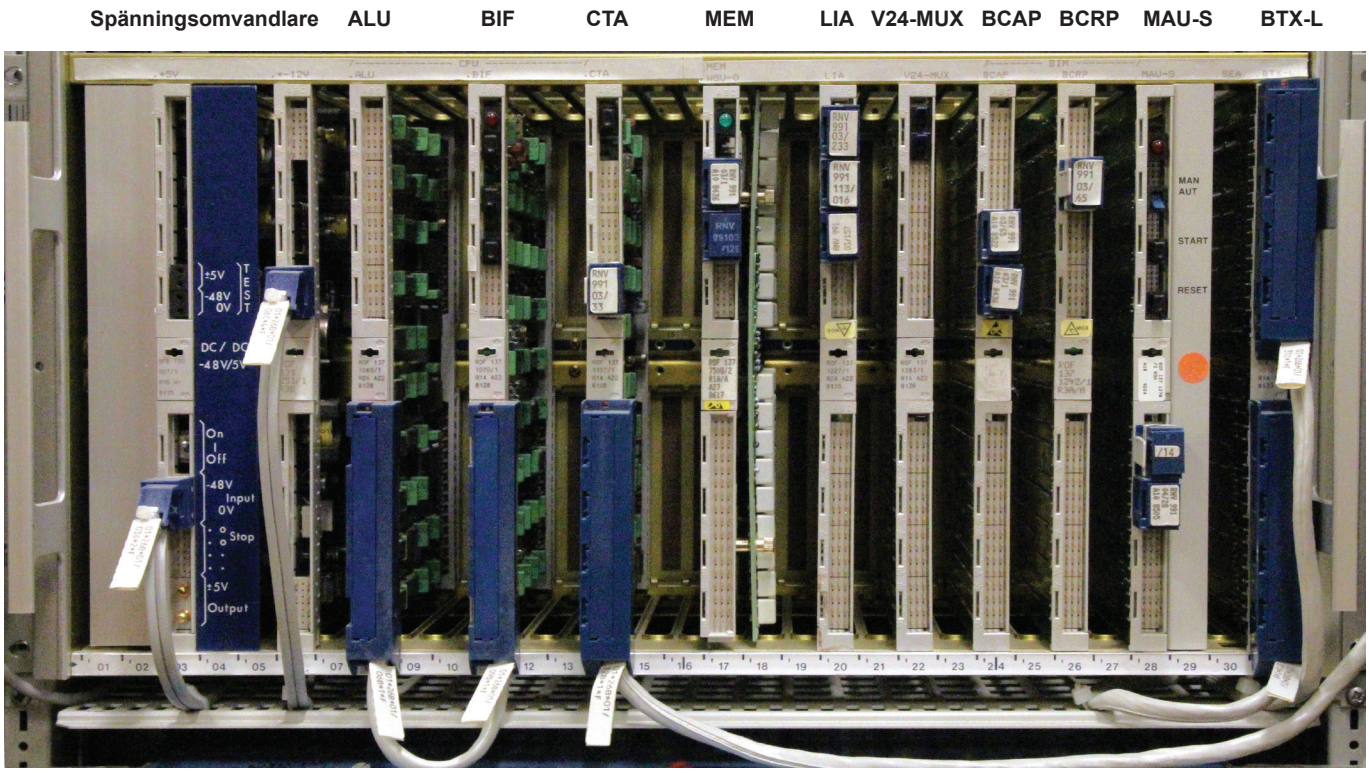
Kontrollfunktioner i programvaran

Applikationsprogrammen innehöll inbyggda kontroller för att upptäcka onormala tillstånd i programavverkningen. Till exempel övervakades att data hade rimliga värden. Vid fel anropades en felhanteringsfunktion i exekutivsystemet EXEC 163 med instruktionen TRAP(82). Detta resulterade i att en interruptprocedur lagrade en felrapport.

Vid allvarligare fel, till exempel att en process inte tilldelades en begärd databuffert, användes instruktionen TRAP(80). Detta resulterade i att en felrapport lagrades, men även i att liten återstart utfördes.

Lagring av felrapporter för enheter

Övervakningsfunktionerna i APZ 101 och APT 101 sände rapporter om inträffade feltillstånd i enheter i växeln till den centrala programvaran i MAS. Felrapporterna lagrades för att presenteras i felmenyerna på bildskärmsterminalen. Dessutom sändes larm från MAU-S till externt larminsamlingssystem.



Figur 13. Magasin med kretskort ingående i CPS med APN 163.

5. STYRSYSTEM APZ 101 MED CENTRALPROCESSOR APN 167

Allmänt

AXT 101 för Televäxel 420 och nätväxlar AXT 121 levererades med ett styrsystem som var baserat på en centralprocessor av typ APN 167. Denna, som hade utvecklats av Ericsson för i första hand realtidstillämpningar i hjälpsystem till AXE 10, hade högre programavverkningskapacitet än APN 163, men framför allt hade den ett modernare I/O-system. Kassettbandspelaren för program- och stationsdataladdning i APN 163 var i APN 167 ersatt med en hårddisk/diskettenhet.

Styrsystemstrukturen med de fyra delsystemen CPS, RPS, IOS och MAS var densamma som i APN 163-versionen. Eftersom en stor del av systemstrukturen och funktionaliteten var oförändrad beskrivs här endast av de nya delarna.

Centralprocessordelsystem CPS

Allmänt

Televäxel 420, med Ericsson-benämningen AXT 101 05, var bestyckad med centralprocessor APN 167 11. Denna användes även i nätväxlar AXT 121 02 och AXT 121 05. I den sista leveransen av nätväxlar, AXT 121 08, användes en vidareutvecklad processor, APN 167 21.

APN 167 var baserad på kretsar ur Motorolas 68000-familj. Enligt samma princip som för APN 163 och dess CP-buss var externa enheter anslutna till CPU via ett centralt bussystem, VME-bussen. Denna var en standard som användes i många datorsystem som var baserade på 68000-kretsar.

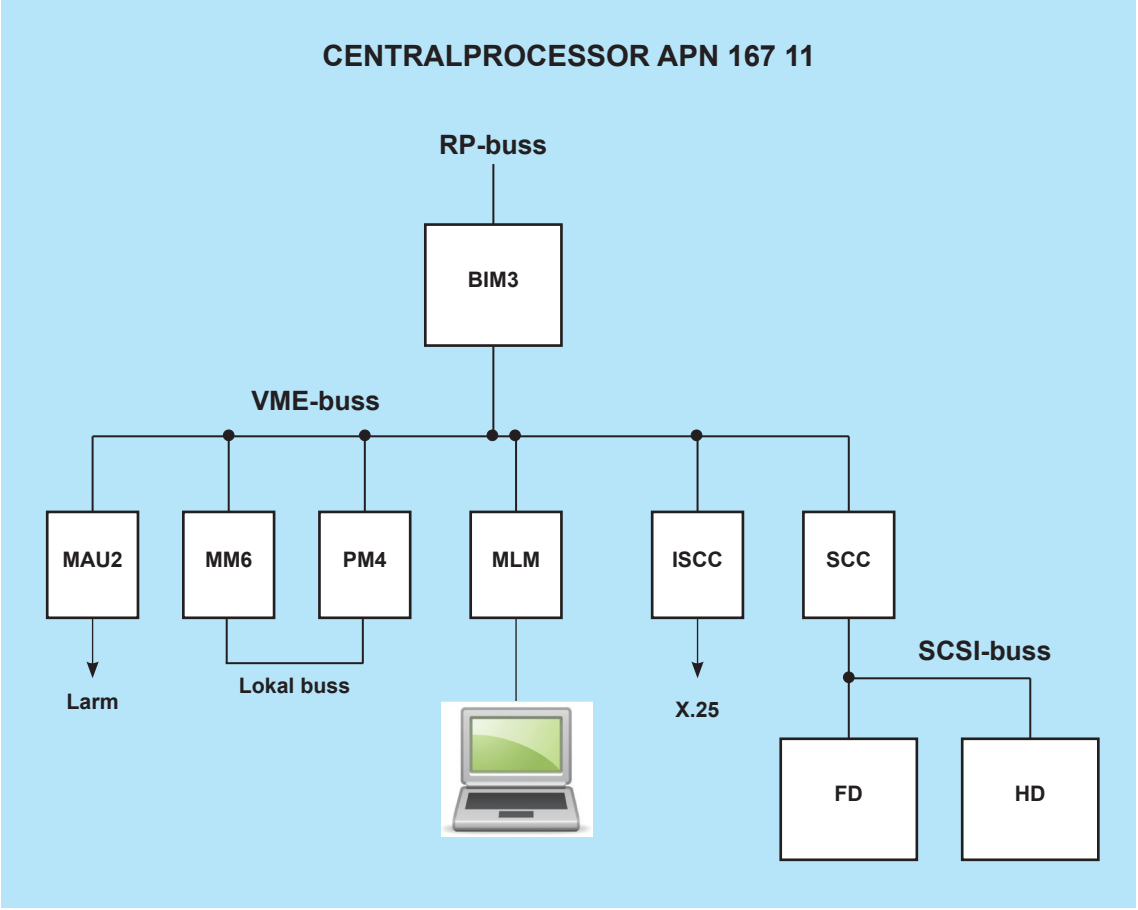
Exekutivsystemet EXEC 163 var i APN 167 ersatt med ett mer utvecklat operativsystem, EriOS. Exekutivdelen i detta hade stora likheter med motsvarande funktioner i EXEC 163. Detta var inte en tillfällighet, eftersom det var samma konstruktionsinstans som hade tagit fram de båda systemen. Programspråket var EriPascal, en utveckling av ”standard” Pascal med realtidsfunktioner.

Centralprocessor APN 167 11

APN 167 11 bestod av två kretskort: processorenheten PM4 och minnesenheten MM6 (se figur 14). Enheterna anslöts till varandra över en lokal buss.

Processorenhet PM4

PM4 var baserad på en mikroprocessor av typ Motorola MC68020



Figur 14. CPS med centralprocessor APN 167 11.

och en flyttalsprocessor av typ MC68881. Den senare användes inte i AXT-tillämpningen. Klockfrekvensen var 16 MHz.

MC68020 kunde hantera ord med databredden 8, 16 och 32 bitar. För att snabba upp exekveringshastigheten användes så kallad pipe-lining. Denna innebar att flera operationer kunde utföras parallellt i mikroprocessorn. MC68020 innehöll även ett så kallat cache-minne, från vilket utläsning kunde ske snabbt. I detta lagrades de senast exekverade programinstruktionerna. Innan en instruktion hämtades från minneskortet MM6 undersöktes om den redan fanns lagrad i cache-minnet. Om så var fallet hämtades instruktionen istället därifrån.

PM4 innehöll i övrigt följande:

- Ett PROM-minne, som innehöll det boot-program som vid spänningspåslag till centralprocessorn startade inladdningen av programmet från hårddiskenheten till minnesenheten MM6.
- Anpassningskretsar mot VME-bussen. Bussen innehöll 24 adresstrådar och 16 datatrådar samt ett antal styrtrådar.

  Anslutna enheter kunde interruptera APN 167 11. Detta innebar, som i APN 163, att den ordinarie programexekveringen avbröts och ett interruptprogram, som var knutet till den aktuella enhetstypen, började exekveras.

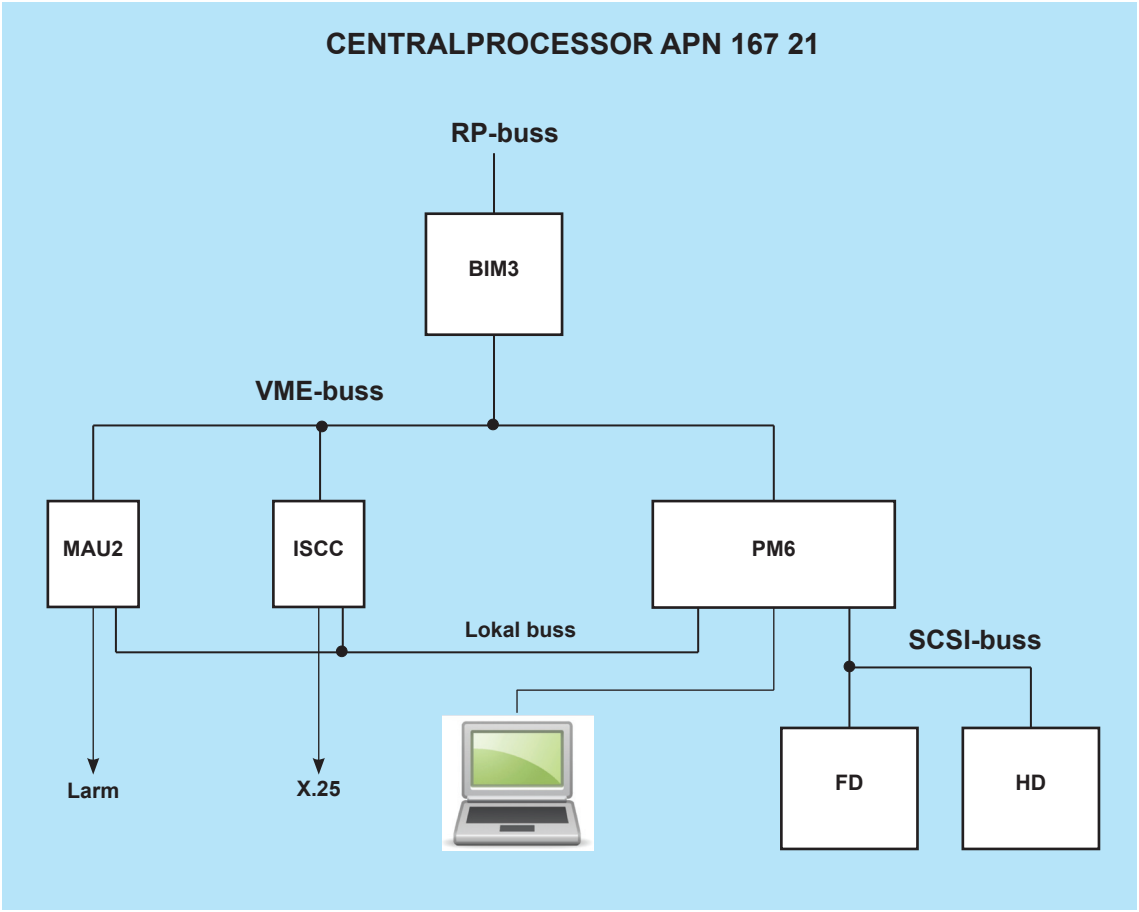
  Anslutna enheter kunde även ta över kontrollen över VME-bussen från APN 167 11 genom så kallad DMA (Direct Memory Access). Metoden användes i första hand för att snabbt överföra data mellan minnesenheten MM6 och hårddisk- och disketttenheterna.
- Anpassningskretsar mot minnesenheten MM6. För att snabba upp överföringshastigheten mellan enheterna användes en lokal buss med 24 adresstrådar och 32 datatrådar.
- Gränssnitt mot en bildskärmsterminal för felsökning (inte visat i figur 14).

Minnesenhet MM6

MM6 kunde lagra 8 Mbyte program och data. Minnet var av typ RAM (Random Access Memory), vilket innebar att det förlorade datat vid spänningsfrånslag.

Centralprocessor APN 167 21

I APN 167 21 var funktionerna hos PM4 och MM6 integrerade i ett enda kretskort, PM6 (se figur 15). Processorn var baserad på mikroprocessor MC68040 och flyttalsprocessor MC6882. Klockfrekvensen var 25 MHz och minneskapaciteten 16 Mbyte.



Figur 15. CPS med centralprocessor APN 167 21.

Kretsarna för anslutning till bildskärmsterminalen för felsökning och SCSI-bussen var integrerade på PM6-kortet.

Hela datorsystemet rymdes i ett enda magasin (se figur 16).

Programspråk

Programspråket i systemen med APN 167 var EriPascal. Detta var baserat på standard Pascal men hade kompletterats med funktioner som krävs i realtidssystem.

Ett speciellt problem vid införandet av APN 167 i AXT-systemen var den programvara i kopplingssystemet som under årens lopp hade skrivits i PL 163. Skulle man ”för hand” skriva om den i EriPascal, något som skulle kräva en stor arbetsinsats men också innebar en risk att introducera fel i fungerande kod? Lösningen blev att bibehålla PL 163-koden och att med en kompilator översätta assemblerkod i PL 163 till assemblerkod i EriPascal. De speciella exekutivsystemanropen i PL163, till exempel SEND och WAIT, gjordes i kompilatorn om till motsvarande EriOS-anrop. På detta sätt kunde existerande program som var skrivna i PL 163 flyttas till APN 167.

Återanvändningen av PL 163-koden underlättades av att det fanns stora likheter i programstrukturen i de båda språken. Den i figur 9 visade strukturen för PL 163-programmen gällde i det stora hela även för EriPascal. Segmentet SYSTEM fanns inte.

Operativsystem EriOS

EriOS bestod av ett exekutivsystem och ett antal funktioner. Exekutivsystemet fungerade som gränssnitt mot maskinvaran i APN 167.

Exekutivsystemet fungerade på nästan exakt samma sätt som i EXEC 163.

Förutom exekutivsystemet innehöll EriOS bland annat ett filhanteringssystem för lagring av data på hårddisken och diskettenheten.

Regionalprocessordelsystem RPS

Delsystemet var till sin struktur oförändrat med en regionalprocessor RP per styrd enhet i kopplingssystemet. Regionalprocessorerna kommunicerade via regionalprocessorbussen RPB och en anpassningsenhet BIM med centralprocessorn.

Regionalprocessor RP

RP bestod, som i APZ 101 med APN 163, av ett kretskort och var baserad på kretsar ur Motorolas mikroprocessorfamilj MC6800. Eftersom det fanns olika typer av gränssnitt mot den styrda maskinvaran i kopplingssystemet fanns det tre regionalprocessorvarianter: RP-2, RP-3 och RP-4.

Bussanpassninqsenhet BIM

BIM var anpassningsenhet mellan centralprocessorn CP och RP-bussen och hade samma uppgift som BIM i APZ 101 med APN 163. Enheten var ansluten till VME-bussen.

BIM bestod av kretskortet BIM3 och var baserad på en mikroprocessor av typ Motorola MC68000.

Figur 16. Magasin med kretskort ingående i CPS med APN 167 21.

I/O-delsystem IOS

Allmänt

IOS innehöll maskinvara och central programvara för in- och utmatning av program och stationsdata samt för bildskärmskommunikation.

För lagring av program och stationsdata användes i första hand hårddiskenheten HD. För back-up av program och stationsdata användes diskettenheten FD. Dataöverföringen mellan centralprocessorenheten och HD/FD styrdes av anpassningsenheten SCC via ett standardiserat bussystem, SCSI-bussen.

För kommunikation med bildskärrnsterminaler och operatörspaneler användes terminalanpassningsenheten MLM.

SCSI-bussanpassninqsenhet SCC

SCC styde överföringen av data mellan APN 167 11 och hårddisk- och diskettenheterna.

Kommunikationen skedde över ett bussystem, SCSI-bussen (Small Computer Systems Interface). Detta bussystem är en standardiserad buss för anslutning av I/O-enheter. Syftet med bussen är att (inom vissa gränser) införa ett maskinvaruoberoende gränssnitt mot den värddator som skall kommunicera med en I/O-enhet. Hårddiskenheten HD och diskettenheten FD innehöll därför ”intelligent” styrlogik som hanterade det speciella lagringsmediet. Kommunikationen mellan datorn och I/O-enheten skedde med kommandon. När ett datablock skulle överföras mellan APN 167 och exempelvis hårddisken, kopplade processorn, i samarbete med SCC, upp en förbindelse från minneskortet MM6 via PM4, VME-bussen, SCC och SCSI-bussen till hårddiskens styrlogikenhet. Dataöverföringen skeddde sedan genom DMA-förfarande på de båda bussarna. SCC innehöll ett buffertminne, som utjämnade för de olika överföringshastigheterna på de båda bussarna.

Hårddiskenhet HD och diskettenhet FD

Hårddiskenheten HD och diskettenheten FD utgjorde en mekanisk enhet. I varje enhet ingick styrlogik för anpassning mot SCSI-bussen.

Disketterna var av typ 3,5 tum med en lagringskapacitet av 2,88 Mbyte.

För att erhålla utökad driftsäkerhet var HD försedd med två hårddiskar med parallell lagring av datat.

I de sista leveranserna av AXT 121-växlar innehöll HD inte hårddiskar, utan en ”flash”-minnesenhet av i princip samma utförande som dagens SSD-diskar. Den hade så goda driftsäkerhetsdata att den inte behövde dubbleras.

Terminalanpassningsenheter MLM och CLIA

MLM anpassade mot bildskärmsterminaler enligt samma princip som LIA i APZ 101 med APN 163. Gränssnittet var CCITT V.24/V.28.

MLM utförde parallell-serieomvandling av data från centralprocessorn till terminalen och serie-parallellomvandling av data från terminalen till centralprocessorn. Dataöverföringen mellan centralprocessorn och MLM skedde tecken för tecken och initierades genom att MLM interrupterade processorn varje gång ett tecken hade sänts eller tagits emot.

I AXT 121 med APN 167 21 var MLM-funktionen integrerad i processorkortet PM6.

För anpassning mot operatörsplatsutrustningarna i Televäxel 420 användes istället för LIA en RP-bussansluten terminalanpassningsenhet, CLIA. Denna hanterade tecken-för-tecken-kommunikationen med operatörsplatsutrustningarna och minskade därmed belastningen på centralprocessorn.

Underhållsdelsystem MAS

Allmänt

MAS övervakade funktionen i CPS och initierade återstart av centralprocessorn vid feltillstånd. I MAS ingick även funktioner för lagring och presentation av feltillstånd i APZ 101 och APT 101 samt sändning av larm.

I MAS ingick maskinvaruenheten MAU2 och central programvara för insamling och lagring av felrapporter, styrning av larmsändning och beordring av systemåterstart.

Övervakningsenhet MAU2

MAU2 utförde samma uppgifter som MAU-S i APZ 101 med APN 163. Den var ansluten till VME-bussen.

Kontrollfunktioner i programvaran

Applikationsprogrammen innehöll inbyggda kontroller för att upptäcka onormala tillstånd i programavverkningen. Till exempel övervakades att data hade rimliga värden. När ett onormalt tillstånd upptäcktes resulterar det i att en felrapport lagrades. Beroende av felets art kunde även återstart av växeln utföras.

Lagring av felrapporter för enheter

Övervakningsfunktionerna i APZ 101 och APT 101 sände rapporter om inträffade feltillstånd i enheter till den centrala programvaran i MAS.

Felen lagrades för att presenteras i felmenyerna på bildskärmsterminalen. Dessutom sändes larm från MAU2 till externt larminsamlingssystem.

6. ANVÄNDA FÖRKORTNINGAR

ALU	Kretskort med aritmetisk-logisk del i APN 163.
APN	Ericssons produktfamilj för datorer (processorer).
APT	Ericssons produktfamil för kopplingssystem.
APZ	Ericssons produktfamilj för styrsystem.
AXE	Ericssons produktfamilj för publika digitala växlar.
AXT	Ericssons produktfamilj för militära digitala växlar.
BIF	Kretskort i APN 163 för anpassning mot centralprocessorbussen.
BIM	Enhet för anpassning mot regionalprocessorbussen i APN 163.
BYB	Ericssons byggsätt för AXx-växlar.
CLM	Klockenhet för styrning av överföringshastigheten i väljare och externa länkar.
CPB	Centralprocessorbuss.
CPS	Centralprocessordelsystem.
CPU	Centralprocessorenhet.
CTD	Bandspelarenhet för 3M-kassetter.
CTE	Styrenhet för bandspelarenheten.
DBA	Direkt bussaccess i APN 163.
DMA	Direkt minnesaccess i APN 167.
DS	Dataminne i APN 163.
EMC-6	Gränssnitt mellan regionalprocessor och styrd enhet.
EPROM	Elektriskt programmerbart minne.
EriOS	Operativsystemet i APN 167.
EriPascal	Programspråket i APN 167.
ETCA	Terminalenhet för anslutning av PCM-system med kanalassocierad signalering (CAS).
ETCC	Terminalenhet för anslutning av PCM-system med gemensam kanalsignalering (CCS)
FD/HD	Diskett- och hårddiskenhet i APN 167.
HDLC	Protokoll för överföring av data.
ISCC	Terminalenhet för datasignalering enligt protokoll X.25.
IOS	Delsystem för hantering av in- och utdata.
LIA	Terminalenhet för datasignalering enligt gränssnitt V.24/V.28 eller strömslinga i system med APN 163..
MAS	Delsystem för drift och underhåll.
MAUx	Enhet för övervakning av CPS.
MJC-D	Enhet för konferensuppkoppling.
MLM	Enhet för datasignalering enligt gränssnitt V.24/V.28 eller strömslinga i system med APN 167..
MM6	Minnesenhet i APN 167 11.
PCD	Terminalenhet för anslutning av analoga linjer.
PCM	Pulskodmodulering.
PL 163	Programspråk i APN 163.
PM4	Processorenhet i APN 167 11.
PM6	Kombinerad processor- och minnesenhet i APN 167 21.
PS	Programminne i APN 163.
RP	Regionalprocessor.
RPB	Regionalprocessorbuss.
RPS	Regionprocessordelsystem.
SCC	Enhet för anpassning mot SCSI-bussen i APN 167 11.
SCSI	Standardiserat gränssnitt för kommunikation mellan datorer och IO-enheter
SEU	Enhet för minnesutökning i APN 163.
SPM	Rumsväljarenhet.
TP/OP	Test- och operationspunktsenhet för signalering mot analoga linjer.
TSM	Tidsväljarenhet.
VAX/ VMS	Dator- och operativsystem tillverkade av Digital Equipment.
VME	Standardiserat protokoll för buss-system med mikroprocessor MC 68000.
WSUx	Minneskort i APN 163.
X.25	Protokoll för paketförmedling av data.