DIE ONTWIKKELING VAN 'N DATAKOMMUNIKASIESTELSEL VIR DIE BEHEER VAN AFGELEË TOESTELLE MET BEPERKTE TERUGVOERING VAN TELEMETRIESE DATA

D.M.L. DE KOKER

DIE ONTWIKKELING VAN 'N DATAKOMMUNIKASIESTELSEL VIR DIE BEHEER VAN AFGELEë TOESTELLE MET BEPERKTE TERUGVOERING VAN TELEMETRIESE DATA

deur

DAWID MATTHYS LODEWICKUS DE KOKER

Skripsie voorgelê ter gedeeltelike voldoening aan die vereistes vir die

MEESTERSDIPLOMA IN TEGNOLOGIE

Elektriese Ingenieurswese (Swakstroom)

in die Fakulteit Ingenieurswese aan die Technikon O.V.S.

Datum van inhandiging: FEBRUARIE 1993.

Studieleier: Mnr. G.D. Jordaan

Central University of Technology, Free State

BEDANKINGS

Ek wil graag die volgende persone bedank sonder wie se hulp en ondersteuning die voltooiing van hierdie projek onmoontlik sou wees:

Die studieleier, mnr. G.D. Jordaan, vir hulp en leiding gedurende die verloop van die projek.

Technikon O.V.S. vir die beskikbaarstelling van fondse om komponente aan te koop vir voltooiing van die hardeware van die projek.

My bestuurder vir die belangstelling in die projek en die tyd aan my toegestaan vir die voltooiing daarvan.

My kollega, mnr. A.C. Bos, vir die hulp, belangstelling en beskikbaarstelling van sagteware en toetsapparaat ter voltooiing van die projek.

My ouers vir die opvoeding en geleenthede aan my verskaf om tot op hierdie vlak te vorder.

My vrou, Rina, vir haar ondersteuning en aanvaarding vir die tyd wat ek van haar moes ontneem om die projek te voltooi.

UITTREKSEL

Die meeste datakommunikasiestelsels maak van die RS232 datakommunikasiestandaard gebruik om toegang tot afgeleë eenhede te verkry deur van modems aan beide eindpunte gebruik te maak. In meer gesofistikeerde stelsels (bv. LANs) word daar van relatief duur apparatuur gebruik gemaak om datakommunikasie tussen verskeie afgeleë gebruikers en die meestereenheid self te bewerkstellig.

'n Behoefte aan 'n alternatiewe lokale datakommunikasiestelsel tussen 'n hoofrekenaar (meester) en 'n aantal intelligente slaafeenhede is dus geïdentifiseer.

'n Multitap (multidrop) datakommunikasiestelsel is ontwikkel om die fasiliteit daar te stel vir die beheer van 'n aantal afgeleë eenhede wat aan 'n hoofrekenaar gekoppel is. 'n Afgeleë eenheid is ontwikkel om met die hoofrekenaar verbind te word wat die basis vorm waarop hierdie datakommunikasiestelsel funksioneer.

Die navorsing is in 4 fases uitgevoer, nl.:

□ *Ontwikkeling van die afgeleë node:* Die node is 'n mikrorekenaar wat ontwikkel is deur van die INTEL 8031 mikrobeheerder gebruik te maak. Dit beskik oor 16K LAG en 8K WLSG. Die SN75176 geïntegreerde kring is gebruik om die RS485 datakommunikasiekoppelvlak daar te stel wat aan die hoofrekenaar verbind moes word. Die mikrorekenaar is d.m.v. 'n optiese isoleerderkring van die datakommunikasielyn geïsoleer. Die sagteware van die mikrorekenaar is met 'n C pakket ontwikkel om as die BIOS program vir die mikrorekenaar te dien en om beheer uit te oefen oor die datakommunikasie tussen die hoofrekenaar en die node.

□ *Ontwikkeling van kommunikasie omgewing:* 'n RS232-na-RS485 omsetter is ontwikkel om as koppelvlak tussen die hoofrekenaar en die mikrorekenaar te dien.

Hierdie kring maak ook gebruik van die SN75176 geïntegreede kring om die RS232 dataseine na die RS485 standaard om te skakel.

□ *Ontwikkeling van hoofrekenaarfunksies:* Die sagteware wat deur die hoofrekenaar gebruik word is ontwikkel deur van Turbo Pascal 6.0 gebruik te maak. Dit verskaf al die databasisse om algemene opstellingsparameters, noderesultate, ens. te stoor. Dit vorm ook die sagtewarekoppelvlak om datakommunikasie moontlik te maak.

□ *Evaluering van datakommunikasiestelsel:* 'n Toetsmodule is ontwikkel wat aan die 8255PPI van die mikrorekenaarnode gekoppel is wat voorsiening maak vir visuele bepaling van die logikatoestand van twee uitsetpoorte, en van 'n insetpoort voorsien is. Data is in beide rigtings oorgedra en die geldigheid daarvan het die korrekte werking van die stelsel bevestig.

Die belangrikste gevolgtrekking wat uit hierdie navorsingsprojek voortgespruit het, is dat 'n RS485 multitap datakommunikasiestelsel 'n baie effektiewe standaard is wat datakommunikasie tussen 'n hoofrekenaar en 'n aantal slaafeenhede moontlik maak. Dit is ook 'n baie koste effektiewe en betroubare stelsel wat oorweeg behoort te word vir hierdie tipe van toepassings.

SYNOPSIS

Presently most communication systems make use of RS232 communication standards to access remote units by utilising modems at both ends. In more sophisticated systems (e.g. LANs) use is made of rather expensive equipment to provide communication between various remote users and the master unit itself.

The problem was therefore to develop an alternative local communication system to facilitate data communication between a master computer and a number of intelligent slave units using state-of-the-art components and the latest design techniques.

A multipoint (multidrop) data communication system has been developed to facilitate the requirements for the control of a number of remote units connected to a master computer. A remote unit has been developed and built to interface with the master computer to form the basis on which the data communication system operates.

The research was carried out in 4 phases, namely:

□ *Development of slaves:* The slave unit is a microcomputer which was developed using the INTEL 8031 microcontroller. It has 16K ROM and 8K RAM. The SN75176 integrated circuit was used to provide the RS485 data communication interface for connecting the slave unit to the master computer. The microcomputer is isolated from the data communication line by making use of an optic isolator circuit. The software of the microcomputer was developed using a C package to form the BIOS program for the node and to control the data communication between the master computer and the node.

□ *Development of communication environment:* A RS232-to-RS485 converter was developed to form an interface between the master computer and the microcomputer's RS485 data communication interface. This circuit also utilises the SN75176 integrated circuit to convert the RS232 data signals to the RS485 standard.

□ *Development of master functions:* The software used by the main computer was developed using Turbo Pascal 6.0. It provides the necessary data bases to save the general setup parameters, node results, etc. It also forms the software interface to facilitate data communication between the master and slave units.

□ *Evaluation of communication system:* A test module was developed and connected to the 8255PPI integrated circuit of the microcomputer node to provide visible outputs for the two output ports and inputs for the input port. Data was transferred in both directions and the validity of the received data proved the correct operation of the data communication system.

The most important conclusion that can be made from this project is that the RS485 multipoint data communication standard is a very effective system to facilitate data communication between a master computer and a number of slave units. It is also a very cost efficient and reliable system that should be considered for applications of this nature.

INHOUD

BYLAES

DATAKOMMUNIKASIESTELSEL HARDEWARE

DATAKOMMUNIKASIESTELSEL SAGTEWARE

MIKROREKENAAR SAGTEWARE

LYS VAN FIGURE

HOOFSTUK 2

HOOFSTUK 3

HOOFSTUK 4

LYS VAN TABELLE

HOOFSTUK 2

HOOFSTUK 3

HOOFSTUK 4

HOOFSTUK 1

INLEIDING

1.1 Probleemstelling

Datakommunikasiestelsels word al hoe meer geïmplementeer om meer as een gebruiker toegang tot dieselfde hoofrekenaarfunksies te bied. Hierdie stelsels vereis normaalweg hoë installering- en onderhoudskostes wat nie algemeen regverdig kan word nie.

Oor die algemeen word daar van die RS232 seriale datakommunikasiestandaard gebruik gemaak om afgeleë eenhede te beheer vanaf 'n hoofrekenaar. Aangesien daar van modems aan beide eindpunte gebruik gemaak word, word die koste verbonde aan die stelsel baie groot.

'n Behoefte het dus onstaan om 'n alternatiewe datakommunikasiestelsel te ontwikkel wat op 'n plaaslike vlak gebruik kan word deur gebruik te maak van spesiaal ontwikkelde datakommunikasiekomponente en die nuutste ontwerptegnieke.

'n Moontlike toepassing van sodanige kommunikasiestelsel is vir sentrale beheer van afgeleë nodes in 'n rekenaarbeheerde besproeiingstelsel.

1.2 Doelwit van die ondersoek

Die doelwit van hierdie projek word onder die volgende punte opgesom:

- Die bepaling van 'n geskikte datakommunikasiestandaard waarop die stelsel gegrond kan word. Die RS485 datakommunikasiestandaard is as die mees geskikte standaard vir hierdie kommunikasiestelsel geïdentifiseer.
- Die implementering van die RS485 datakommunikasiestandaard, waaronder:
  - Die vasstelling van die sagtewareprotokols om datakommunikasie tussen die hoofrekenaar en mikrorekenaarnode moontlik te maak.
  - Die ontwikkeling van 'n hardeware omsettermodule wat die RS232 standaard dataseine van die hoofrekenaar na die RS485 standaard omskakel sodat dit met die node kan kommunikeer.
- Die ontwikkeling van die afgeleë node, waaronder:
  - Die ontwikkeling van die mikrorekenaarhardeware met voorsiening vir RS485 datakommunikasie.
  - Die ontwikkeling van sagteware om as BIOS program en datakommunikasiebeheerprogram van die node te dien.

1.3 Hipotese

'n Multitap datakommunikasiestelsel kan ontwikkel word om aan die dataoordragbehoeftes te voldoen vir die beheer van 'n aantal intelligente afgeleë nodes deur 'n hoofrekenaar.

'n Persoonlike rekenaar is as die hoofrekenaar gebruik en 'n mikrorekenaarnode is ontwikkel om as afgeleë node te dien. Voorsiening is gemaak vir 'n totaal van 16 individueel adresseerbare nodes.

Sodanige node funksioneer net soos 'n terminaal in 'n gewone datakommunikasiestelsel deurdat daar 'n unieke adres aan dit toegeken is waarop dit reageer op versoeke van die hoofrekenaar.

Daar was egter nie net vir 16 adresse (4 bisse) in die sagteware voorsiening gemaak nie, maar wel vir 32 (5 bisse) om 'n groter mate van buigsaamheid aan die stelsel te verleen.

1.4 Afbakening van die studieterrein

Na intensiewe ondersoek ingestel is na die vermoë van verskeie seriale datakommunikasiestandaarde, is die volgende tekortkominge in die RS232, RS423 en RS422 standaarde uitgewys:

- *RS232*: Buiten die nadeel van die beperkte maksimum datakommunikasieafstand - as gevolg daarvan dat hierdie standaard van die enkelent modus gebruik maak - kan daar slegs een sender en een ontvanger in so 'n stelsel aan mekaar verbind word.
- *RS423:* Hierdie standaard maak wel voorsiening vir langer kommunikasieafstande, hoewel dit ook van die enkelent modus gebruik maak, maar daar kan slegs een sender en tien ontvangers aan so 'n stelsel verbind word.
- *RS422*: Hierdie standaard funksioneer in 'n differensiële modus. Dit kan egter net een sender en tien ontvangers vir so 'n stelsel bied.

Nie een van bogenoemde standaarde het dus aan die behoefte van die stelsel wat in hierdie projek ontwikkel sou word voldoen nie. As laaste alternatief is die RS485 datakommunikasiestandaard oorweeg. Hierdie standaard maak van 'n differensiële modus vir datakommunikasie gebruik en beskik oor die vermoë om 32 senders en 32 ontvangers te akkommodeer.

Die projek behels derhalwe die implemetering van die RS485 standaard in 'n halfduplekse toepassing tussen 'n persoonlike rekenaar as hoofrekenaar en mikrobeheerder gebaseerde nodes as eindpunte.

1.5 Metode van navorsing

Die navorsing is in vier fases uitgevoer, nl.:

- Ontwikkeling van die afgeleë node,
- Ontwikkeling van die datakommunikasieomgewing,
- Ontwikkeling van datakommunikasiesagteware,
- Evaluering van die datakommunikasiestelsel.

1.5.1 Ontwikkeling van die afgeleë node

Die afgeleë node is as 'n mikrorekenaar eenheid ontwikkel deur van 'n mikrobeheerder gebruik te maak. Hierdie mikrorekenaar is toegerus met eksterne LAG en WLSG, asook 'n Inset/Uitset (I/U) module waarop 'n RS232-na-RS485 omsetterkring voorkom.

Hierdie module bevat ook 'n 8255PPI (programmeerbare apparatuur koppelvlak) kring wat dit moontlik maak om parallelle insette na die mikrorekenaar te voer en ook parallelle uitsette te lewer. Dit bevat ook optiese isoleerders wat die mikrorekenaar van die datakommunikasiemedium isoleer.

Die mikrorekenaar funksioneer as 'n intelligente alleenstaande eenheid wat sy eie datakommunikasie, geheueorganisering en algemene funksies kan uitvoer.

1.5.2 Ontwikkeling van die datakommunikasieomgewing

Aangesien die mikrorekenaarnode van die RS485 datakommunikasiestandaard gebruik maak, moes 'n RS232-na-RS485 omsetter ontwikkel word wat aan die hoofrekenaar gekoppel kon word om datakommunikasie tussen hierdie twee eenhede moontlik te maak .

Daar is van 'n RS485 sender/ontvanger kring (SN75176) gebruik gemaak om hierdie koppelvlak te verskaf. Hierdie kring kan in die halfduplekse modus gebruik word, wat kabelkoste verminder aangesien slegs 2 geleiers benodig word om tweerigtingkommunikasie te bewerkstellig.

1.5.3 Ontwikkeling van datakommunikasiesagteware

Sagtweware is ontwikkel om die algemene opstellingsparameters, databasisse, ens. op die hoofrekenaar te beheer. Hierdie sagteware beheer ook die datakommunikasie na en vanaf die mikrorekenaarnode. Dit maak voorsiening vir die onttrekking van inligting vanaf die node en die programmering van die mikrorekenaarnode om sekere vasgestelde funksies, volgens data wat verskaf word vanaf die hoofrekenaar, uit te voer.

Die mikrorekenaarsagteware dien as BIOS en datakommunikasiesagteware van die node. Hierdie sagteware maak voorsiening daarvoor om data vanaf die hoofrekenaar te ontvang om verdere parallelle inset/uitset funksies te verrig. Dit kan ook data van die node se huidige staat aan die hoofrekenaar verskaf indien die hoofrekenaar dit sou versoek.

Datakommunikasie word verder gesinkroniseer deurdat die hoofrekenaar sowel as die node van sekere identifikasieseine gebruik maak om funksies aan te vra en om bevestiging aan mekaar te verskaf.

1.5.4 Evaluering van die datakommunikasiestelsel

Evaluering van die sagteware is gedoen tydens die ontwikkeling, en na die voltooiing daarvan, op sowel die hoofrekenaar as die mikrorekenaarnode.

Die stelsel is tot 'n sekere mate aan elektriese ruis blootgestel om die betroubaarheid en geldigheid van die gestuurde en ontvangde data na en vanaf die hoofrekenaar te evalueer. Data is vanaf die hoofrekenaar gestuur en by die node getoets vir geldigheid. Hierdie data is weer teruggestuur na die hoofrekenaar waar die geldigheid daarvan weereens getoets is om 'n betroubare stelsel te verseker.

'n Toetsmodule is ontwikkel om verdere evaluering van die afgeleë mikrorekenaarnode te bewerkstellig.

Hierdie module maak voorsiening vir visuele indikasie (d.m.v. LEDs) aan die navorser van elke uitsetbis van die twee uitsetpoorte van die I/U module 8255PPI om korrekte uitsette te verseker. Dit bied ook die fasiliteit om 8 enkelbis insette aan die mikrorekenaar te verskaf om die korrekte parallelle invoering van data te evalueer.

1.6 Probleme ondervind

Vervolgens word die belangrikste probleme soos ondervind tydens die uitvoering van die projek uitgelig.

- Die RS485 sender/ontvanger het terugvoering aan die plaaslike ontvanger gegee tydens datatransmissie vanaf die hoofrekenaar. Die kring is vervolgens in die ontvang-gereed toestand geplaas, en die sender is slegs geïnisieer indien datatransmissie verlang is.
- Polsing na die nodes kon weens 'n onderbrekingsteenstrydigheid in die DOS stelsel nie ten volle geïmplementeer word nie. Polsing d.m.v. handbeheer is egter ten volle geïmplementeer.
- 'n Aantal hardeware probleme het ook tydens die ontwikkeling van die mikrorekenaarnode ontstaan en is gedurende die ontwikkelingsproses opgelos.

HOOFSTUK 2

DATAKOMMUNIKASIESTELSELS - 'N TEORETIESE OORSIG

Die basiese elemente van 'n datakommunikasiestelsel word as volg uiteengesit:

- Datakommunikasietoerusting, waaronder:
  - 'n Rekenaar,
  - Datakommunikasiehardeware om data te stuur en te ontvang,
  - 'n Kommunikasiemedium waaroor data gestuur/ontvang kan word.
- Datakommunikasiesagteware wat hierdie stelsel beheer,
- Iemand/iets om mee te kommunikeer, waaronder:
  - 'n Ander rekenaar met aanpasbare datakommunikasietoerusting, hierna na verwys as mikrorekenaarnodes.
- Geduld (Rettke, 1990:10).

Verskeie datakommunikasiekonfigurasies kom voor wat almal van 'n algemene konfigurasie afgelei is. Figuur 2.1 toon die blokdiagram van die algemene datakommunikasiestelsel soos deur Schweber (1988:6).

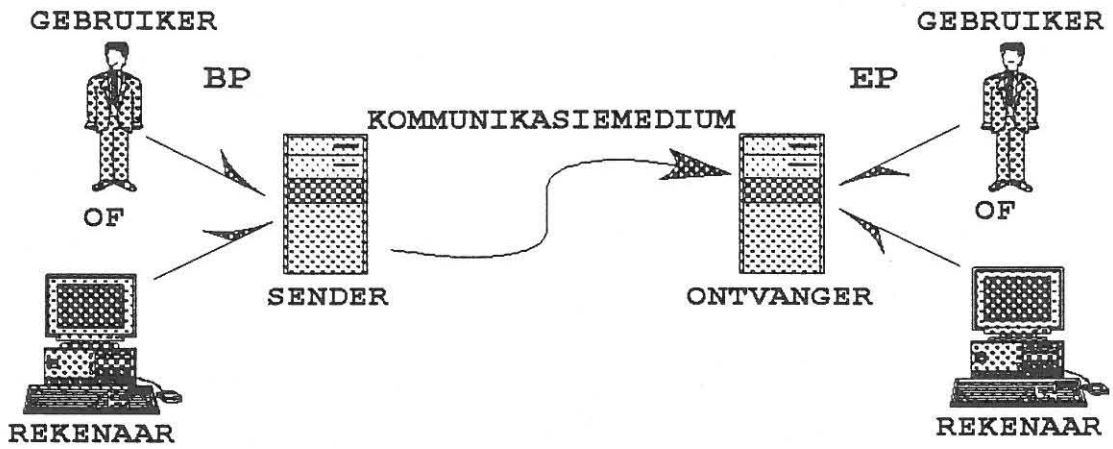


Fig. 2.1 - Blokdiagram van 'n algemene datakommunikasiestelsel

Vanuit Figuur 2.1 kan gesien word dat data vanaf 'n beheerpunt (BP) na/vanaf die eindpunt (EP) via 'n datakommunikasiemedium gestuur/ontvang moet word. Verskeie datakommunikasiestandaarde, bv. parallelle en seriale datakommunikasietegnieke, maak die oordrag van data tussen die BP en EP moontlik.

Die beheer van hierdie oordrag word deur sagteware bewerkstellig. Sagteware speel dus so 'n belangrike rol dat dit moeilik is om dit te skei van die hardeware in hierdie stelsel. Om die aanbieding in die skripsie te vergemaklik word dit egter geskei en die volgende afdelings word vervolgens afsonderlik bespreek:

- Mikrorekenaarnodes,
- Datakommunikasie,
- Sagteware vir datakommunikasiebeheer.

2.1 Mikrorekenaarnodes

Die eerste vraag wat na vore kom indien daar na mikrorekenaars verwys word is: wat is mikrorekenaars? Mikrorekenaars is stelsels van een of meer geïntegreerde kringe, wat van halfgeleier tegnologie en digitale logika gebruik maak, om groot rekenaar funksies op 'n klein skaal te implementeer (Intel, 1985:1-1).

Nodes word in drie kategorieë ingedeel nl. onintelligente nodes, mikroprosesseerdernodes en mikrobeheerdernodes.

'n Onintelligente node kan geen funksies vanself uitvoer nie en is afhanklik van beheer vanaf 'n beheerpunt. Dit reageer slegs wanneer dit versoek word om 'n sekere funksie(s) uit te voer en is verder staties in operasionele staat.

Mikroprosesseerdernodes daarenteen is intelligente nodes wat voorafbepaalde funksies kan uitvoer. 'n Voorbeeld van 'n mikroprosesseerder is die INTEL 8086 geïntegreerde kring. Daar is drie hoofelemente in 'n mikrorekenaar wat met 'n mikroprosesseerder toegerus is, nl.:

- Die sentrale verwerkingseenheid (SVE),
- Die eksterne geheue,
- Die inset/uitset (I/U) poorte/apparatuur (Intel, 1985:1-1).

Die SVE is die hart van die mikrorekenaar. Dit voer numeriese prosessering, logiese bewerkings en tydhoufunksies uit. Die eksterne geheue word gebruik om programme en data te stoor. Die I/U poorte/apparatuur is die enigste manier waarop die mikroprosesseerder met die buitewêreld kan kommunikeer (Intel, 1985:1-2).

Die mikrobeheerdernodes funksioneer soortgelyk aan die mikroprosesseerdernodes, maar verskil in sekere opsigte daarvan. Voorbeelde van mikrobeheerders is die Intel 8031/51 reeks mikrobeheerder geïntegreerde kringe.

Hierdie kringe verskil in die volgende opsigte van die mikroprosesseerderkringe:

- SVE is ontwerp vir beheertoepassings,
- Dit het interne geheue en hoef nie noodwendig van eksterne geheue gebruik te maak nie,
- Dit beskik oor 'n seriale poort en tipies 32 tweerigting, onafhanklik-adresseerbare I/U lyne (Intel, 1991a:5-3).

Daar kan gesien word dat die toepassing die tipe mikrorekenaarnode sal bepaal. Indien daar van seriale datakommunikasie gebruik gemaak word, en daar word besluit om 'n mikroprosesseerderkring vir die toepassing te gebruik, moet daar addisionele kringe ontwerp word om hierdie tipe van kommunikasie te bewerkstellig. Addisionele

sagteware sal ook ontwikkel moet word om hierdie deel van die ontwerp afsonderlik te beheer.

Die mikrobeheerder daarenteen beskik reeds oor hierdie fasiliteit en slegs sagteware vir die verwerking van die seriale data hoef ontwikkel te word. Indien daar van direkte parallelle I/U kommunikasie gebruik gemaak word kom die probleem van addisionele komponente weer voor. Mikrobeheerders word meestal in beheertoepassings gebruik en mikroprosesseerders in meer komplekse funksies as beheeruitoefening.

Vir die moontlike toepassings van die tegnieke ontwikkel tydens die uitvoering van hierdie navorsingsprojek, is besluit om van 'n mikrobeheerder gebruik te maak in die mikrorekenaarkonfigurasie. Vervolgens word daar op hierdie konfigurasie gekonsentreer.

Na deeglike oorweging is daar besluit om van die Intel 8031/51 reeks mikrobeheerderkringe gebruik te maak in hierdie projek. Die eerste rede hiervoor is dat hierdie kringe reeds oor 'n RS232 koppelvlak beskik wat ekstra komponente in die ontwerp uitskakel en dus die ontwerp vereenvoudig (Intel, 1991b:2-11). Dit vergemaklik ook die sagtewareontwerp om datakommunikasie moontlik te maak.

Elke mikrorekenaarnode moet oor 'n unieke adres beskik sodat dit soos 'n rekenaarnode in 'n rekenaarnetwerk geadresseer kan word. Die beskikbaarheid van verskeie I/U lyne wat direk adresseerbaar is, sonder om van eksterne konfigurasies gebruik te maak, het ook aanleiding gegee tot die gebruik van hierdie komponente - aangesien die unieke adres direk op een van die I/U poorte aangebring kan word (Intel, 1991b:2-10).

Verdere oorweging is geskenk aan die moontlike gebruik van onderbreekinsette (interrupt inputs) om van eksterne gebeure rekening te kan gee. Die 8031/51 reeks

kringe beskik dan ook oor eksterne onderbreekinsette om so 'n moontlikheid te vergemaklik (Intel, 1991a:5-3). Die seriale poort verskaf ook onderbreekaanduiding wat die gebruik van onderbrekingsprosedures (interrupt procedures) vir sagtewarebeheer van dié poort moontlik maak (Intel, 1991b:2-13).

2.2 Datakommunikasie

Kommunikasie is die ruiling van gedagtes of opinies d.m.v. woorde, briewe of boodskappe (Tugal & Tugal, 1989:1). Verskeie vorme van inligting kan voorkom en in rekenaartoepassings staan hierdie inligting as data bekend (Tugal & Tugal, 1989:3).

Om kommunikasie tussen rekenaars moontlik te maak moet 'n datakommunikasiemedium daarvoor voorsien word. 'n Datakommunikasieskakel/medium word gedefinieer as 'n weg/medium waarlangs elektriese seine tussen twee of meer stasies/terminale op 'n enkeldraad, 'n groep drade, 'n koaksiale kabel, 'n spesiale deel van die radiofrekwensiespektrum of 'n optiese vesel kabel gesend/ontvang kan word (Tugal & Tugal, 1989:3).

Drie basiese tipes datakommunikasie kom voor, nl.:

- Parallel,
- Seriaal,
- Netwerke.

Daar bestaan verskillende gebruike en standaarde vir bg. tipes datakommunikasie. Alle tipes datakommunikasienetwerke, hetsy parallelle of seriale konfigurasies, moet egter aan spesifieke protokols voldoen ten einde 'n mate van beheer uit te oefen oor die gebruik daarvan. Tabel 2.1 dui die protokolstandaarde vir seriale en netwerk kommunikasie aan.

Tabel 2.1 - Tabel van 7-vlak ISO-OSI protokols

FUNKSIONELE VLAK	FUNKSIE	BENADERINGS VOORBEELDE
1. Fisies	Bistransmissie oor 'n fisiese medium.	EIA RS 232 EIA RS 422 EIA RS 449 CCITT X.21
2. Dataskakel	Maak die betroubare ruiling van logiese sekwensiële boodskappe oor 'n enkele fisiese dataskakel moontlik.	ISO--HDLC ANSI--ADCCP IBM--SDLC,BSC DEC-DDCMP CCITT X.25
3. Netwerkbeheer	Intranetwerk operasies, adressering en roetering. Terminaalkoppeling - skakelingbeheer.	CCITT X.25, X.21 ANSI X281
4. Datavloeibeheer	Punt-na-punt beheer en inligting ruiling. Betroubaarheid en foutbeheer.	3 & 4 WORD DIKWELS SAAMGEVOEG
5. Sessiebeheer	Ondersteuning van sessie-dialoog indien programme en dienste beskikbaar is.	ARPANET FTP DECNET DAP
6. Aanbiedingsbeheer	Kompaktering, enkripsie, randapparatuurkodering en formatering.	STELSEL GEDEFINIEERDE BENODIGDHEDE
7. Prosesbeheer	Toepassing en stelselaktiwiteitsbeheer.	GEBRUIKER GEDEFINIEERDE STAPPE

Hierdie standaarde is deur The International Standards Organization (ISO) voorgestel en staan bekend as the Reference Model of Open Systems Interconnection waarna kortliks as ISO-OSI verwys word (Sinnema & McGovern, 1986:451).

Die ISO-OSI is slegs 'n stel riglyne en nié 'n goed gedefinieerde standaard nie (Sinnema & McGovern, 1986:451). Vervolgens word die sewe vlakke van 'n totale datakommunikasienetwerk kortliks bespreek.

□ *Vlak 1 : Fisies.*

Protokols word hier verskaf vir die seinspanningswaai en bisduur, bepaling of die transmissie simpleks, halfdupleks of voldupleks is en hoe koppelings by die eindpunte verkry word.

□ *Vlak 2 : Dataskakel.*

Op hierdie vlak word uitgaande boodskappe in rame saamgestel en daar word op bevestigings gewag nadat elke boodskap gestuur is. Uitgaande rame bevat ook 'n destinasieadres en 'n adres van afkoms indien nodig, asook foutopsporingsdata en foutkorreksiedata.

□ *Vlak 3 : Netwerkbeheer.*

Op hierdie vlak word uitgaande boodskappe in pakkette verdeel. Inkomende boodskappakkette word saamgestel vir gebruik in hoër vlakke en roetering definieer die destinasie van die pakket asook die orde van transmissie.

□ *Vlak 4 : Datavloeibeheer.*

Hierdie vlak bepaal watter kanale vir datatransmissie gebruik moet word.

□ *Vlak 5 : Sessiebeheer.*

Op hierdie vlak bewerkstellig die gebruiker 'n stelsel-na-stelsel verbinding. Dit beheer die aan- en aftekening van die gebruiker asook die gebruikeridentifikasie, rekeningtelling en sessiebestuur.

□ *Vlak 6 : Aanbiedingsbeheer.*

Dit beheer funksies waarna die gebruiker gereeld vra en regverdig dus algemene behandeling. Funksies soos biblioteekroetines, enkripsie, kode-omskakeling, ens. word hier verteenwoordig.

□ *Vlak 7 : Prosesbeheer.*

Dit is die vlak soos gesien deur die individuele gebruiker. Geen bekendstelling van die stelsel se detail en hardeware word aan die gebruiker gegee nie, maar wel die toepassings daarvan (Sinnema & McGovern, 1986:451-454).

Hieruit kan dus gesien word dat 'n totale datakommunikasienetwerk deur hierdie sewe-vlak protokol voorgestel word. In hierdie navorsingsprojek sal uitsluitlik op vlak 1 gekonsentreer word. Dit verteenwoordig seriale en parallelle datakommunikasie. Daar word ook na Lokale Area Netwerke (LANs) verwys om die verskille duideliker uit te wys.

Vervolgens word bogenoemde afsonderlik bespreek.

2.2.1 Parallelle datakommunikasie

Verskeie standaarde kom voor onder parallelle datakommunikasie, waaronder die volgende:

- Mikroverwerker S100 bus,
- Mikroverwerker 6800 bus,
- IEEE-488 algemene koppelvlakbus,
- IEEE-583 CAMAC koppelvlakbus (Zaks & Lesea, 1977:322).

Parallelle datakommunikasie is baie bruikbaar waar daar hoëspoed module-na-module kommunikasie - in die geval van mikrorekenaarbusse, en stelsel-na-stelsel kommunikasie - in die geval van IEEE-488, benodig word. CAMAC is die enigste uitsondering van bostaande standaarde omdat dit alle kommunikasie vanaf die komponentvlak opwaarts kan hanteer (Zaks & Lesea, 1977:323).

Parallelle busse dra alle bisse van 'n datawoord gelyktydig oor via afsonderlike datalyne. Lyne moet verskaf word vir die databus, adresbus en vir die beheerbus in 'n basiese mikrorekenaarstelsel (Zaks & Lesea, 1977:323). 'n Basiese mikrorekenaar sal bv. 8 bisse nodig hê om een ASCII karakter voor te stel. 'n Voordeel van parallelle datakommunikasie is dat dit baie maklik is om te gebruik

- anders as seriale datakommunikasie waar daar baie probleme kan voorkom (Horowitz & Hill, 1980:730).

Parallelle datakommunikasie is duurder as seriale datakommunikasie aangesien meer verbindings voorsien moet word om kommunikasie moontlik te maak. Dit is egter ook beperk tot 'n kort afstand waaroor daar gekommunikeer kan word.

Genoemde standaarde is egter reeds verouderd en die meer populêre standaarde word as volg gelys:

- Centronics parallelle datakommunikasiepoort,
- SCSI datakommunikasiebus,
- IPI datakommunikasiebus,
- IEEE-488 GPIB datakommunikasiebus.

Vervolgens word bogenoemde protokols kortliks bespreek.

2.2.1.1 Centronics parallelle datakommunikasiepoort

Hierdie datakommunikasiepoort is 'n enkelvoudige woordlengte (8 bisse), tweerigting parallelle poort wat oor aansluitbevestiging (handshaking signals) beskik. Dit is deur Centronics daargestel en word uitsluitlik vir drukkertoepassings gebruik (Horowitz & Hill, 1980:730). Dit beskik oor 8 datalyne, 3 beheerlyne en 6 statuslyne om die drukkerstatus te kan weergee (Horowitz & Hill, 1980:731).

2.2.1.2 SCSI datakommunikasiebus

Daar bestaan universele parallelle koppelvlakstandaarde om skywe (hardeskyf- & sagteskyfhardeware) en ander hoëverrigting randapparatuur aan mikrorekenaars te koppel. SCSI (Small Computer System Interface) is 'n 8-bis parallelle koppelvlak met aansluitbevestiging en protokols om meer as een gasheer (host) en meer as een randapparaat te kan hanteer (Horowitz & Hill, 1980:733).

Dit beskik oor beide asinkrone en sinkrone modusse en voorafgedefinieerde sagtewareprotokols. SCSI het die voordeel dat dit alle mikrorekenaars aanpasbaar maak met alle tipes randapparatuur. Dit kan datasnelhede van tot 1.5 Mgreep/s (asinkrone) of 4 Mgreep/s (sinkrone) met 'n kabellengte van 6m (enkelent) of 24m (differensieel) hanteer (Horowitz & Hill, 1980:734).

2.2.1.3 IPI datakommunikasiebus

IPI (Intelligent Peripheral Interface) funksioneer soortgelyk aan SCSI, met die verskil dat dit 'n 16-bis koppelvlak is om vir nuwe-generasie skywe voorsiening te maak. Dit spesifiseer 'n koppelvlak wat teen 10 Mgreep/s (5 MHz oordrag verhouding) funksioneer en kan ook soos SCSI meer as een gasheer en randapparaat hanteer (Horowitz & Hill, 1980:734).

2.2.1.4 IEEE-488 GPIB datakommunikasiebus

Hierdie datakommunikasiebus is ontwerp om stelsels, eerder as modules, met mekaar te verbind. Apparatuur soos bv. rekenaars, multimeters, kragbronne

en frekwensiegenerators kan met 'n IEEE-488 bus toegerus word. Die 488 bus het tot stand gekom na 3 jaar van onderhandeling tussen Hewlett Packard (HP) en IEC (International Electrotechnical Commission). In 1974 het die IEEE hierdie ontwerp goedgekeur wat dan aanleiding gegee het tot die totstandkoming van die IEEE-488-1975 standaard (Zaks & Lesea, 1977:336).

Hierdie koppelvlakbus staan algemeen bekend as die GPIB (general purpose interface bus) of 488-bus. Dit het die universele digitale koppelvlak in laboratoriums geword om instrumentasie aan mekaar te verbind. Instrumente van alle maatskappye kan op dieselfde GPIB gekoppel word met 'n mikrorekenaar wat die beheer daaroor uitoefen (Horowitz & Hill, 1980:734).

Die basiese bus word verbind aan apparatuur wat een van die volgende funksies kan uitvoer :

- Om ander eenhede te beheer - *beheerder*,
- Neem inligting aan vanaf die beheerder - *luisteraar (listener)*,
- Verskaf inligting aan die beheerder - *prater (talker)*.

Hierdie bus beskik oor 8 tweerigting datalyne, 3 greepoordrag beheerlyne en 5 algemene beheerlyne.

2.2.2 Seriale datakommunikasie

Onder seriale datakommunikasie word die volgende bespreek:

- Seriale datakommunikasietegnieke, waaronder:
  - Asinkrone datakommunikasie,

- Sinkrone datakommunikasie.
- Datatransmissiespoed (DSR),
- Datatransmissierigtings,
- Datakommunikasielynstrukture,
- Seriale datakommunikasieprotokols, waaronder:
  - EIA (Electronics Industry Association) RS232 datakommunikasie,
  - EIA RS423 datakommunikasie,
  - EIA RS422 datakommunikasie,
  - EIA RS485 datakommunikasie.
- Foutopsporing in seriale datakommunikasiestelsels, waaronder:
  - Pariteit, waaronder:
    - Vertikale oortolligheidskontrole (vertical redundancy check),
    - Horisontale oortollighiedskontrole (longitudinal redundancy check).
  - Sikliese oortolligheidskontrole (cyclic redundancy check).

2.2.2.1 Seriale datakommunikasietegnieke

Twee tipes seriale datakommunikasietegnieke kom voor en word vervolgens bespreek.

2.2.2.1.1 Asinkrone datakommunikasie

Asinkrone datakommunikasie is waar bisse in groepe van 0e of 1e as individuele karakters gestuur word. Die dataseinprosessering word deur 'n voorafbepaalde klok bepaal. Tydvertragings, ook genoem gapings, mag in die datastring voorkom terwyl die stelsel op die volgende karakter wag om geprosesseer te word. Elke karakter word omring deur

spesiale bisse wat die begin en einde daarvan aandui. Hierdie bisse staan as die beginbis en stopbis bekend.

'n Pariteitstoetsbis word op gereelde intervalle in die datastring ingesluit en word gebruik om moontlike foute in die gestuurde data uit te wys. Die meeste mikrorekenaars maak van hierdie tegniek van datakommunikasie gebruik.

Figuur 2.2 illustreer die vorm van die datastring/woord soos dit voorkom in asinkrone datakommunikasie (Rettke, 1990:14).

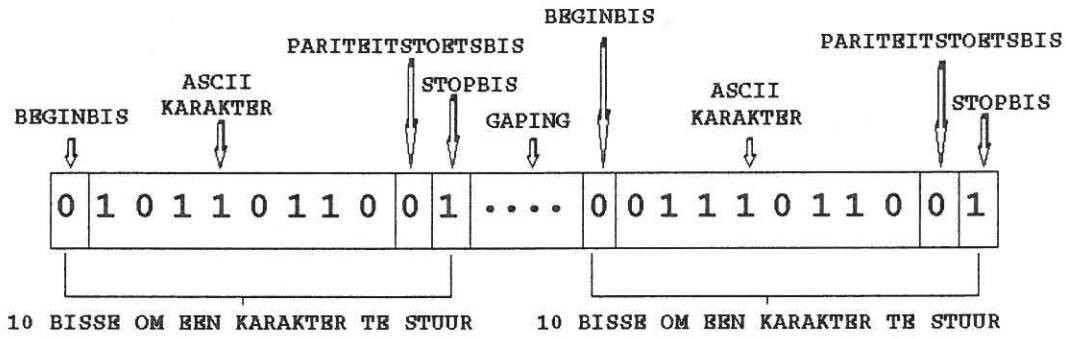


Fig. 2.2 - Asinkrone datastring/woord samestelling

2.2.2.1.2 Sinkrone datakommunikasie

In hierdie modus word data teen 'n voorafbepaalde tempo gesend as 'n kontinue string van groepe karakters. Tydhoudende seine word in die datastring ingesluit om die send- en ontvangstoerusting met mekaar te sinkroniseer. Die inligting word in 'n pakket gestuur met presiese tydhouding bewerkstellig tussen die sender en ontvanger. Sinkrone

datakommunikasie is vinniger as asinkrone datakommunikasie en word gewoonlik deur groot rekenaarstelsels gebruik.

Die datastring/woord samestelling van sinkrone datakommunikasie word in Figuur 2.3 aangetoon (Rettke, 1990:15).

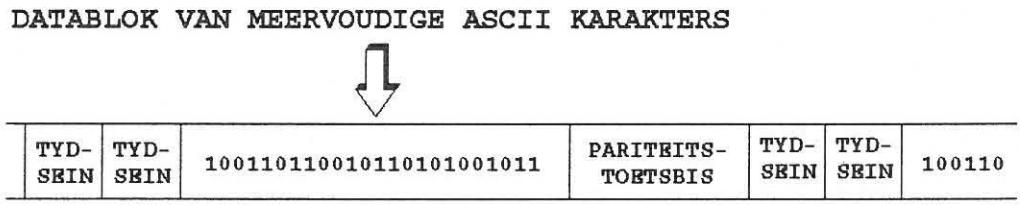


Fig. 2.3 - Sinkrone datastring/woord samestelling

Bogenoemde tegnieke word algemeen in al die protokols gebruik en die gebruike daarvan word deur die tipe datakommunikasiestelsel bepaal.

2.2.2.2 Datatransmissiespoed

Datatransmissiespoed is die tempo waarteen data gestuur en ontvang word in 'n datakommunikasiestelsel. Dit word deur die volgende formule gegee vir parallelle, sowel as seriale datakommunikasie (Freeman, 1985:1065):

$$DSR = \sum_{i=1}^{m} \frac{1}{Ti} log_2 Ni$$

waar

DSR = datatransmissietempo,

m = aantal parallelle kanale,

T_i = minimum interval vir die i ste kanaal - in sekondes uitgedruk,

N_i = aantal beduidende kondisies van modulasie in die i ste kanaal.

Hierdie datatransmissiespoed word in bisse per sekonde aangedui. Bogenoemde formule sal dus nou as volg daar uitsien vir seriale datakommunikasie:

$$DSR=\frac{1}{T}log_2 n$$

Met 'n 2-kondisie modulasie (n=2), wat 'n logika '0' of '1' kan wees, is:

$$DSR=\frac{1}{T}$$

2.2.2.3 Datatransmissierigtings

Datatransmissierigting word geklassifiseer as

- Simpleks,
- Halfdupleks,
- Voldupleks.

Bogenoemde word vervolgens kortliks bespreek.

2.2.2.3.1 Simplekse datatransmissie

Simplekse datatransmissie is eenrigting transmissie. 'n Voorbeeld hiervan is radio uitsendings. Daar kan na die uitsendings geluister word, maar geen inligting kan teruggestuur word nie. Net so kan 'n simplekse

rekenaarterminaal slegs 'n send-alleen of ontvang-alleen terminaal wees. Hierdie tipe rekenaardatatransmissie is baie beperk en word baie selde gebruik (Rettke, 1990:16).

2.2.2.3.2 Halfduplekse datatransmissie

Halfduplekse datatransmissie is tweerigting transmissie. Data kan beide gestuur en ontvang word, maar slegs een modus kan op 'n slag aktief wees. 'n Voorbeeld hiervan is 'n tweerigting radio. Slegs een persoon kan inligting op 'n slag stuur. Slegs indien die eerste individu datatransmissie afgehandel het kan die ander data stuur. Hierdie tipe datatransmissie word gebruik waar daar 'n beperkte aantal datakommunikasieverbindings voorkom (Rettke, 1990:16).

2.2.2.3.3 Volduplekse datatransmissie

Volduplekse datatransmissie is ook tweerigting transmissie, maar beide partye kan gelyktydig data stuur *en* ontvang. 'n Enkel datakommunikasiekanaal beskik oor die vermoë om data gelyktydig en onafhanklik te stuur en te ontvang. Volduplekse datatransmissie word algemeen gebruik vir hoofraam-na-hoofraam rekenaarstelselverbindings (Rettke, 1990:16).

2.2.2.4 Datakommunikasielynstrukture

Wanneer datakommunikasieterminale aan rekenaars verbind word val die datakommunikasiemedium in een van die volgende kategorieë:

- Punt-tot-punt of
- Multitap.

Bogenoemde word vervolgens kortliks bespreek.

2.2.2.4.1 Punt-tot-punt datakommunikasielynstruktuur

Figuur 2.4 toon die blokdiagram vir 'n punt-tot-punt datakommunikasiestelsel soos deur Sinnema & McGovern (1986:36).

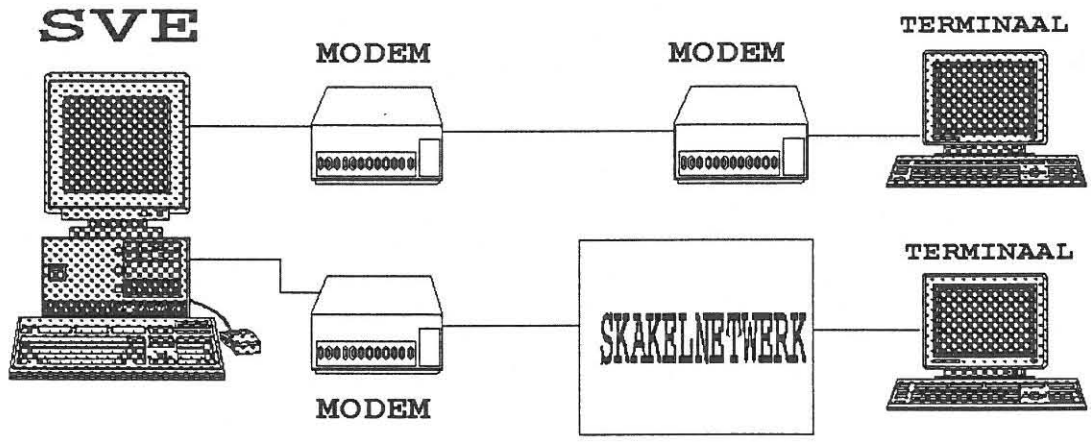


Fig. 2.4 - Blokdiagram 'n van punt-tot-punt datakommunikasiestelsel

In hierdie konfigurasie word twee punte aan mekaar verbind deur van private lyne of skakelnetwerke gebruik te maak (Sinnema & McGovern, 1986:35). Een van die terminale word gewoonlik as die primêre terminaal aangewys

sodat dit oor die outoriteit beskik om data te kan stuur. Die ander terminaal word as die sekondêre terminaal aangewys wat toegelaat word om data te stuur slegs op versoek/toelating van die primêre terminaal. Indien daar van volduplekse datatransmissie gebruik gemaak word beskik beide terminale oor dieselfde outoriteit.

2.2.2.4.2 Multitaplyn datakommunikasie

Figuur 2.5 toon die blokdiagram van 'n multitap datakommunikasiestelsel.

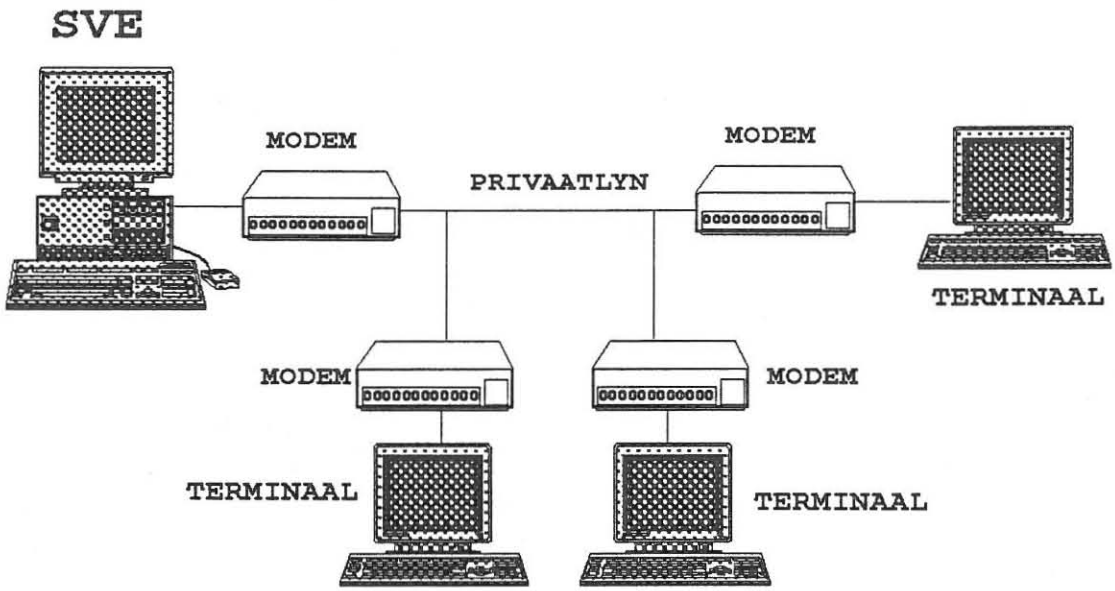


Fig. 2.5 - Blokdiagram van 'n multitap datakommunikasiestelsel

In 'n multitapkonfigurasie deel 2 of meer terminale/rekenaars 'n gemeenskaplike 2-draad of 4-draad privaatlyn. Hoewel 2 of meer terminale boodskappe gelyktydig kan ontvang, kan slegs een terminaal data stuur op 'n slag. Soos in die geval van die punt-tot-punt

konfigurasie word een terminaal/rekenaar as die primêre stasie aangewys en die ander as die sekondêre terminale/rekenaars.

Buskontensie (bus contention) - wanneer meer as een terminaal/rekenaar probeer om beheer oor die datakommunikasielyn oor te neem, of wanneer meer as een terminaal/rekenaar probeer om data gelyktydig te stuur - word oorkom deur tegnieke te implementeer wat as pols en selekteer (poll and select) bekend staan. Datatransmissie kan in halfduplekse of volduplekse modusse voorkom.

Elke stasie wat aan die datakommunikasielyn verbind is word 'n unieke adres toegeken. Indien 'n primêre stasie terminale pols (poll) vra dit die terminale in 'n spesifieke volgorde of dit data het om te stuur al dan nie. Elke sekondêre terminaal analiseer die ontvangde boodskap en evalueer die unieke identifikasieadres. Slegs een sekondêre terminaal sal die unieke adres aanvaar/herken en op die pols reageer.

Indien hierdie terminaal oor geen data beskik om te versend nie, stel dit die primêre stasie dienooreenkomstig in kennis en die volgende terminaal kan gepols word. Indien die sekondêre terminaal wel data het om te stuur, stuur dit 'n positiewe respons terug na die primêre stasie.

Die sekondêre stasie kan op hierdie stadium voortgaan om data na die primêre stasie te stuur, of dit kan op moontlike verdere instruksies vanaf die primêre stasie wag - soos in Figuur 2.6 geïllustreer (Sinnema & McGovern, 1986:37).

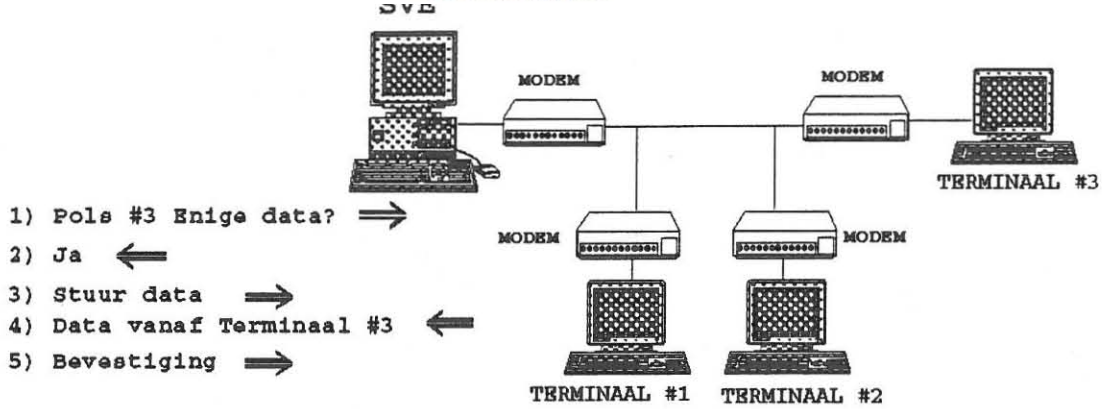


Fig. 2.6 - Polsing (Polling) - 'n tipiese blokdiagram

Met die ontvangs van data vanaf die sekondêre stasie stuur die primêre stasie 'n bevestiging (acknowledgement) aan die sekondêre stasie terug om aan te dui of die data korrek ontvang is of nie. Indien die data korrek ontvang is kan die polsroetine voortgaan.

Selektering kan ook gebruik word om buskontensie uit te skakel. In hierdie tegniek selekteer die primêre stasie slegs een sekondêre stasie - nie opeenvolgens al die nodes nie - wat dit dan adresseer en vra of dit data het om na die primêre stasie terug te stuur. Die terminaal wat d.m.v. die unieke adres geselekteer is antwoord met 'n positiewe of negatiewe respons.

Indien die terminaal positief antwoord, stuur dit data terug na die primêre stasie. Indien die antwoord vanaf die terminaal negatief is kan die primêre stasie dit op 'n latere geleentheid weer selekteer. Indien die primêre stasie die data korrek ontvang, stel dit die sekondêre stasie daarvan in kennis en dit kan met die volgende seleksie of polsing voortgaan. Figuur 2.7 toon die blokdiagram van 'n tipiese

selekteringskonfigurasie aan soos deur Sinnema & McGovern (1986:37).

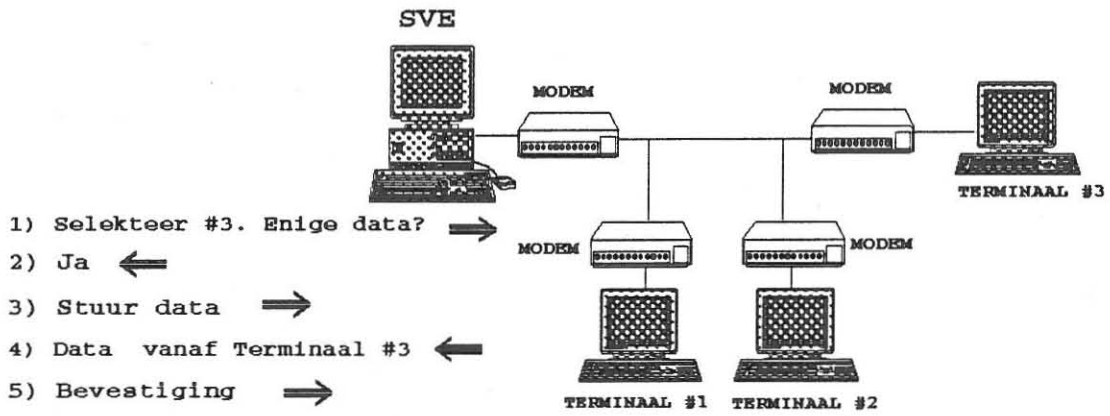


Fig. 2.7 - 'n Tipiese selekteringskonfigurasieblokdiagram

2.2.2.5 Seriale datakommunikasieprotokols

Seriale datakommunikasiestandaarde bepaal die afstand, datatransmissiespoed, ens. van 'n datakommunikasiestelsel en word vervolgens bespreek.

2.2.2.5.1 EIA-RS232 datakommunikasie

Die RS232 seriale datakommunikasiestandaard is in 1969 neergelê en dit is as die RS232C standaard geklassifiseer. In 1986 is die RS232D standaard daargestel. Beide hierdie standaarde maak gebruik van bipolêre spanningseine om kommunikasie moontlik te maak.

Die RS232 standaard spesifiseer die kwaliteitstandaarde vir beide drywers en ontvangers. 'n Drywer *moet* +5V tot +25V (logika LAAG uitset) en -5V tot -25V (logika HOOG uitset) in 'n las van 3k tot 7k met 'n swaaitempo (slew rate) van minder as 30V/µs genereer en terselfdertyd in staat wees om 'n kortsluiting op enige ander uitset te kan weerstaan - wat tot ±5V @ 500mA kan wees. Die ontvanger *moet* 'n lasweerstand van 3k tot 7k voorhou terwyl dit 'n inset van +3V tot +25V vir 'n logika LAAG moet omskakel asook -3V tot -25V vir 'n logika HOOG (Pasahow, 1981:121).

Dit moet egter in ag geneem word dat die RS232 drywer die logika '1' omkeer na die negatiewe vlak en dit staan as die "merk" bekend. Die logika '0' is die positiewe vlak en staan as die "spasie" bekend. In stroomlustransmissie vloei stroom gedurende die logika '1' tydperk en eindig gedurende die logika '0' vlak.

RS232 ontvangers het gewoonlik spanningshisterese by die inset en sekere tipes het die fasiliteit om die reaksiespoed d.m.v. 'n kapasitor te beperk sodat dit minder sensitief is vir ruispulse. RS232 werk baie goed tot 38400 baud oor 'n afstand van 15m. Vir korter datakommunikasieafstande word dit tot 115200 baud gebruik (Horowitz & Hill, 1980:723). Die standaard spesifiseer ook die verbindertipe (connector type) en pentoewysings.

RS232 is ontwerp om DTEs (dataterminaalapparaat) en DCEs (datakommunikasieapparaat) met mekaar te verbind. 'n Terminaal word beskou as 'n DTE en 'n modem as 'n DCE, maar ander randapparatuur kan beide wees.

Tabel 2.2 som die RS232 seine op wat tussen 'n DTE en DCE voorkom soos voorgehou deur Horowitz & Hill (1980:724).

Tabel 2.2 - Die RS232 standaard dataseine (DTE ⇔ DCE)

	PEN NOMMER		RIGTING	FUNKSIE (SOOS GESIEN DEUR DTE)	
SEIN	25 PEN	9 PEN	DTE ⇔ DCE		
TD	2	3	⇨	SEND DATA	DATAPAAR
RD	3	2	⇦	ONTVANG DATA	
RTS	4	7	⇨	SENDVERSOEK-SEIN	AANSLUIT-BEVEST I-GINGS-PAAR
CTS	5	8	⇦	GEREEDHEID-SEIN	
DTR	20	4	⇨	DATA-TERMINAAL-GEREED	AANSLUIT-BEVEST I-GINGS-PAAR
DSR	6	6	⇦	DATASTEL-GEREED	
DCD	8	1	⇦	DATADRAER-BESPEUR	BEWERK - STELLIG DTE INSET
RI	22	9	⇦	LUI AANDUI	
FG	1	-		RAAMGROND	
SG	7	5		SEINGROND	

Vervolgens word 'n kort beskrywing van elke lyn gegee.

TD en RD is die datapaar. Data word via hierdie twee lyne gestuur en ontvang. RTS is die sendversoekseinlyn (request to send) en CTS die

gereedheidseinlyn (clear to send). DTR is dataterminaalgereed (data terminal ready), DSR is datastelgereed (data set ready) en DCD is datadraerbespeur (data carrier detect). Daar kom ook 2 grondkoppelings voor, nl. raamgrond (frame/chassis ground) en seingrond (signal ground). Hierdie twee lyne word gewoonlik aan mekaar verbind.

Die bespreekte seine wat nie datalyne is nie, word die aansluitbevestigingseine genoem. Figuur 2.8 illustreer die interaksie van die aansluitbevestigingseine kragtens Held (1991:74).

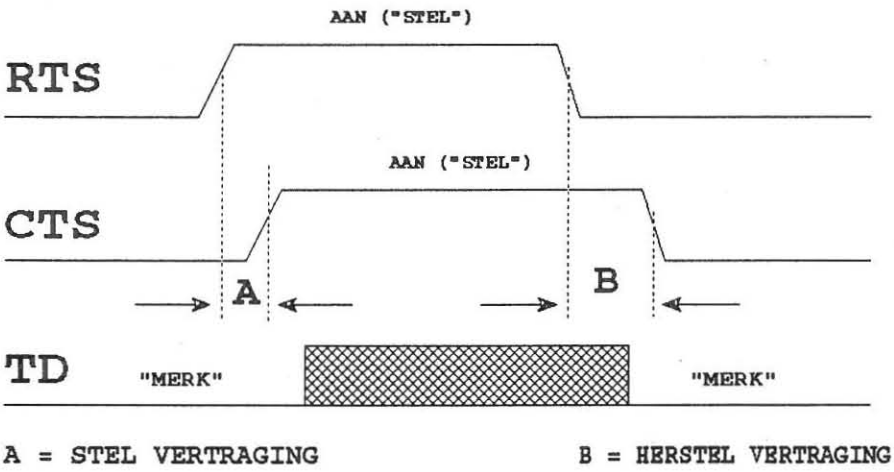


Fig. 2.8 - RS232 aansluitbevestigingseininteraksie

RTS word gestel deur die DTE om aan te dui dat dit data het om te stuur en dat die DCE gereed moet maak om die data te ontvang. Omdat RTS gestel is, stel die DCE sy draersein. Na 'n kort vertraging stel die DCE sy CTS-lyn wat die DTE in kennis stel dat dit gereed is om data te ontvang.

Indien die DTE geen data het om meer te stuur nie, herstel dit die RTS-lyn. Na 'n kort vertraging - om seker te maak dat alle data wat die

koppelvlakpunt gekruis het, gestuur is - herstel die DCE die CTS-lyn. Indien die RTS-lyn herstel is, mag dit nie weer gestel word voordat die CTS-lyn deur die DCE herstel is nie.

Alle seinlyne maak van RS232 bipolêre seinvlakke gebruik met die datalyne (TD, RD) wat *negatief* gestel word en die beheerlyne (RTS, CTS, DSR, DTR, DCD) wat *positief* gestel word.

RS232 kan in die meeste toepassings gebruik word. Daar is egter 'n paar tekortkominge wat aanleiding gegee het tot die ontwikkeling van ander seriale datakommunikasiestandaarde wat later bespreek word. Die beperkings op RS232 word deur Schweber (1988:292-293) as volg gelys:

□ *Afstand*: Die 15m kommunikasieafstand is oor die algemeen nie voldoende vir klein industriële aanlegte of groot kantore nie,
□ *Datatempo:* Die 20 kbaud spoed (somtyds na 38.4 kbaud verhoog - buite die spesifikasies) maak nie voorsiening daarvoor dat 'n groot hoeveelheid data vinnig genoeg oorgedra kan word nie,
□ *Meervoudige gebruikers:* Die behoefte onstaan dat dieselfde lyn deur meer as een gebruiker gedeel kan word weens die hoë koste verbonde aan die kommunikasiemedia. Die RS232 standaard is slegs ontwerp om voorsiening te maak vir 2 gebruikers in 'n konfigurasie waar een gebruiker direk aan 'n ander verbind is (punt-na-punt),
□ *Algehele doeltreffendheid en buigsaamheid:* Baie stelsels benodig 'n koppelvlak wat meer doeltreffend is (afstand, spoed, ens.), en met 'n groter mate van buigsaamheid. Die verlangde buigsaamheid sluit in dat verskeie interkoppelings aanvaarbaar is, maar dat die mees geskikte een vir die toepassing gekies kan word.

koppelvlakpunt gekruis het, gestuur is - herstel die DCE die CTS-lyn. Indien die RTS-lyn herstel is, mag dit nie weer gestel word voordat die CTS-lyn deur die DCE herstel is nie.

Alle seinlyne maak van RS232 bipolêre seinvlakke gebruik met die datalyne (TD, RD) wat *negatief* gestel word en die beheerlyne (RTS, CTS, DSR, DTR, DCD) wat *positief* gestel word.

RS232 kan in die meeste toepassings gebruik word. Daar is egter 'n paar tekortkominge wat aanleiding gegee het tot die ontwikkeling van ander seriale datakommunikasiestandaarde wat later bespreek word. Die beperkings op RS232 word deur Schweber (1988:292-293) as volg gelys:

□ *Afstand*: Die 15m kommunikasieafstand is oor die algemeen nie voldoende vir klein industriële aanlegte of groot kantore nie,

□ *Datatempo:* Die 20 kbaud spoed (somtyds na 38.4 kbaud verhoog - buite die spesifikasies) maak nie voorsiening daarvoor dat 'n groot hoeveelheid data vinnig genoeg oorgedra kan word nie,

□ *Meervoudige gebruikers:* Die behoefte onstaan dat dieselfde lyn deur meer as een gebruiker gedeel kan word weens die hoë koste verbonde aan die kommunikasiemedia. Die RS232 standaard is slegs ontwerp om voorsiening te maak vir 2 gebruikers in 'n konfigurasie waar een gebruiker direk aan 'n ander verbind is (punt-na-punt),

□ *Algehele doeltreffendheid en buigsaamheid:* Baie stelsels benodig 'n koppelvlak wat meer doeltreffend is (afstand, spoed, ens.), en met 'n groter mate van buigsaamheid. Die verlangde buigsaamheid sluit in dat verskeie interkoppelings aanvaarbaar is, maar dat die mees geskikte een vir die toepassing gekies kan word.

□ *Ruisimmuniteit:* RS232 is baie kwesbaar vir die voorkoms van ruis - selfs oor kort afstande. Dit beteken dat die stelsel datafoute oplewer. Dit is wel moontlik om hierdie foute op te spoor, maar dit beïnvloed die algehele effektiwiteit van die datakommunikasiestelsel. 'n Koppelvlak wat die waarskynlikheid van datafoute verminder sal dus verkieslik wees.

□ *Spanningsvlakke:* Die +3V tot +25V en -3V tot -25V spanningstrek van RS232 is nie direk aanpasbaar met vermoëns van tipiese digitale elektroniese kringe nie. Spesiale kragbronne moet ontwerp word om hierdie spannings te lewer en dit veroorsaak 'n toename in koste, fisiese ruimte en onbetroubaarheid t.o.v. die elektronika.

Daar bestaan egter geen datakommunikasiekoppelvlak wat al bogenoemde tekortkominge kan oplos nie. Verbeterings op die RS232 standaard bestaan egter en hierdie standaarde word vervolgens bepreek.

2.2.2.5.2 EIA-RS423 datakommunikasie

Die RS423 datakommunikasiestandaard brei die RS232 standaard uit deurdat dit meervoudige ontvangers, groter kommunikasieafstand en vinniger datatransmissiesnelhede toelaat. Dit is net soos RS232 'n enkelentprotokol wat een dataverbinding (eenrigting) en 'n grondkoppeling benodig om datakommunikasie moontlik te maak. Hierdie grondkoppeling is gemeenskaplik tot al die kringe in die konfigurasie.

Die meervoudige ontvangers maak RS423 egter nie 'n volledige multitapstelsel (multidrop system) nie, aangesien daar nie meervoudige senders mag voorkom nie. 'n Tipiese toepassing van RS423 is waar data slegs na die ontvangers gestuur moet word en geen terugsending van data benodig word nie.

Figuur 2.9 toon die blokdiagram van 'n tipiese RS423 datakommunikasiestelsel.

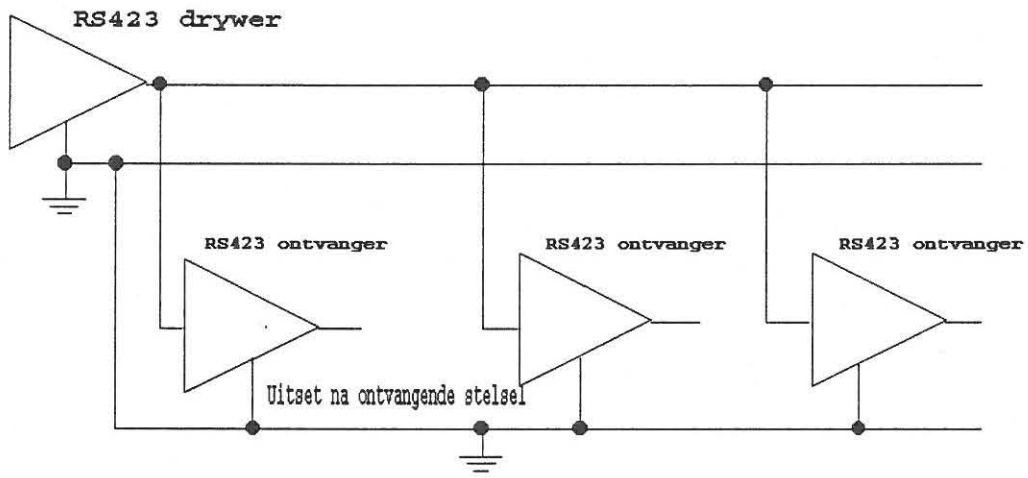


Fig. 2.9 - Blokdiagram van 'n tipiese RS423 datakommunikasiestelsel

Die elektriese spesifikasies vir RS423 bepaal die volgende:

- Datakommunikasieafstande tot 1200m,
- Datatransmissiesnelhede tot 100 kbis/s,
- 'n Maksimum van 10 ontvangers mag aan die stelsel verbind word.

Die spanningsvlakke wat vir RS423 gebruik word is baie laer as dié wat vir RS232 gebruik word. Dit dra gedeeltelik by tot die hoër

datatransmissiesnelhede wat bereik kan word. Vanweë die hoë seinspannings van RS232 word hoë swaaitempos (slew rates) vir hoër datatransmissiesnelhede benodig (Schweber, 1988:299).

Verskeie datakommunikasiestelsels kan egter nie hierdie swaaitempo lewer nie as gevolg van die komplekse drywerkringkonfigurasie. Die kapasitansie van die datakommunikasiemedium beperk ook die swaaitempo. Laer spanningsvlakke benodig proporsioneel laer swaaitempos vir dieselfde aantal bisse per sekonde wat versend moet word. Dieselfde swaaitempo kan dus 'n baie hoër datatransmissiespoed lewer indien die seinspanningsvlak laer is.

RS423 maak gebruik van spanningsvlakke van +3.6V tot +6V om 'n binêre '0' voor te stel en -3.6V tot -6V vir die voorstelling van 'n binêre '1'. Dieselfde swaaitempo as wat op RS232 van toepassing is, verskaf ongeveer 'n viervoudige verhoging in die datatransmissiespoed t.o.v. RS423 a.g.v. die laer seinspanningsvlakke.

Die besluitnemingspunte (decision points) vir binêre '0' en '1' waardes is laer by die ontvanger in die RS423 datakommunikasiestelsel as vir RS232. 'n Spanning van +200mV word deur die ontvanger as 'n binêre '0' beskou en 'n -200mV spanning as 'n binêre '1'. Dit beteken dat proporsioneel groter spanningsverliese in die RS423 datakommunikasiestelsel mag voorkom sonder enige verlies aan data.

Met behulp van die volgende berekeninge kan die effek van datatransmissiespanningsvlakke en besluitnemingspunte by die ontvanger teenoor die swaaitempovermoë aangetoon word:

Indien 'n drywer bv. oor die vermoë beskik om 'n sein te lewer met 'n swaaitempo van 1000V/s sal die RS232 oorgang van -25V tot +25V die volgende tydsvertraging oplewer:

$$\frac{50\,V}{1000\,V/s}=0.05s$$

Dieselfde sein het slegs 0.012s nodig om van -6V tot +6V oor te gaan (12 / (1000V/s)) in die RS423 standaard. Dieselfde drywer swaaitempo maak dus vir 'n baie hoër datatransmissiesnelheid voorsiening indien laer dataseinspanningsvlakke gebruik word (Schweber, 1988:300).

Dieselfde berekeninge kan op die ontvangskant van die datakommunikasiestelsel gedoen word om die swaaitempovermoë te bepaal. Die datalyndrywer mag oor die vermoë beskik om baie hoë swaaitempos te lewer, maar die kapasitansie en induktansie van die datakommunikasiemedium kan die swaaitempo beperk by die ontvangskant.

Vir 'n vasgestelde swaaitempo neem dit slegs 6.7% van die totale tydsduur vir die sein om vanaf -200mV tot +200mV oor te skakel inteenstelling met die -3V na +3V strek van RS232 waar dit 12% neem. Die datakommunikasiemedium lewer dus 'n kleiner beperking op die swaaitempo.

2.2.2.5.3 EIA-RS422 datakommunikasie

Baie ooreenkomste en verskille bestaan tussen die RS232 en die RS422 standaarde. Dit kan net soos RS423 een sender en 10 ontvangers akkommodeer en kan tot 'n datakommunikasieafstand van 1200m gebruik word. Die seinspanningsvlakke is ook soortgelyk aan dié van RS423. Die seinspanningsvlak van die sender hoef egter slegs tussen +2V en -2V te wees.

RS422 maak van differensiële seine gebruik terwyl RS423 van enkelenttransmissie gebruik maak. Figuur 2.10 toon 'n tipiese RS422 datakommunikasiestelselkonfigurasie soos voorgestel deur Schweber (1988:300).

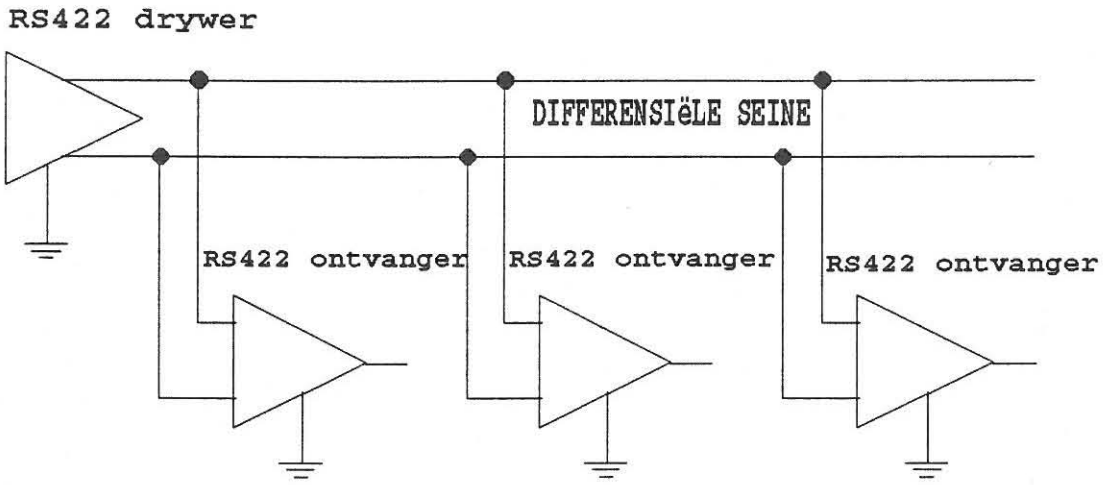


Fig. 2.10 - Blokdiagram van 'n tipiese RS422 datakommunikasiestelsel

Daar moet egter op gelet word dat hoewel die datalyne nie van 'n gemeenskaplike grondkoppeling gebruik maak nie, die sender en ontvanger *nie* van mekaar geïsoleer is nie.

Die voordeel van differensiële seine is dat die seinkwaliteit van die datakommunikasielyn baie beter beheer kan word. Ruisimmuniteit word baie hierdeur verbeter. Die effek van grondprobleme op die seinvorm word ook verminder, aangesien die grondkoppeling nie gebruik word om enige seinenergie te dra nie.

Die verbetering in seinkwaliteit, gekombineer met laer seinspanningsvlakke, gee aanleiding daartoe dat RS422 teen baie hoër datatransmissiesnelhede gebruik kan word. Hierdie standaard kan datatransmissiesnelhede van tot 10 Mbis/s lewer.

'n Tipiese toepassing van RS422 is waar 'n rekenaar groot hoeveelhede data na verskeie terminale moet stuur. Hierdie hoë datatransmissiesnelhede word benodig waar tyd 'n kritiese faktor in die toepassing is.

2.2.2.5.4 EIA-RS485 datakommunikasie

Die RS485 datakommunikasiestandaard beskik oor die hoogste prestasiefaktor en buigsaamheid van die 4 genoemde EIA-gerugsteunde datakommunikasiestandaarde.

Dit maak ook gebruik van differensiële seine om datakommunikasie moontlik te maak en laat datatransmissie toe tot 'n afstand van 1200m (soos RS422). Die grootste verbetering is dat RS485 ware multitaptoepassings moontlik maak, en dat dit drietoestand (tristate) uitsette kan lewer. Daar kan 32 senders en 32 ontvangers op een enkelpaar verbinding gekoppel word, mits hierdie senders en ontvangers

aan die RS485 spesifikasie voldoen (Schweber, 1988:301). Hierdie spesifikasie bepaal watter seine gebruik moet word en hoe die hoë-impedansie modus van die RS485 sender elektries daar moet uitsien.

Die blokdiagram van 'n tipiese RS485 datakommunikasiestelselkonfigurasie word in Figuur 2.11 getoon.

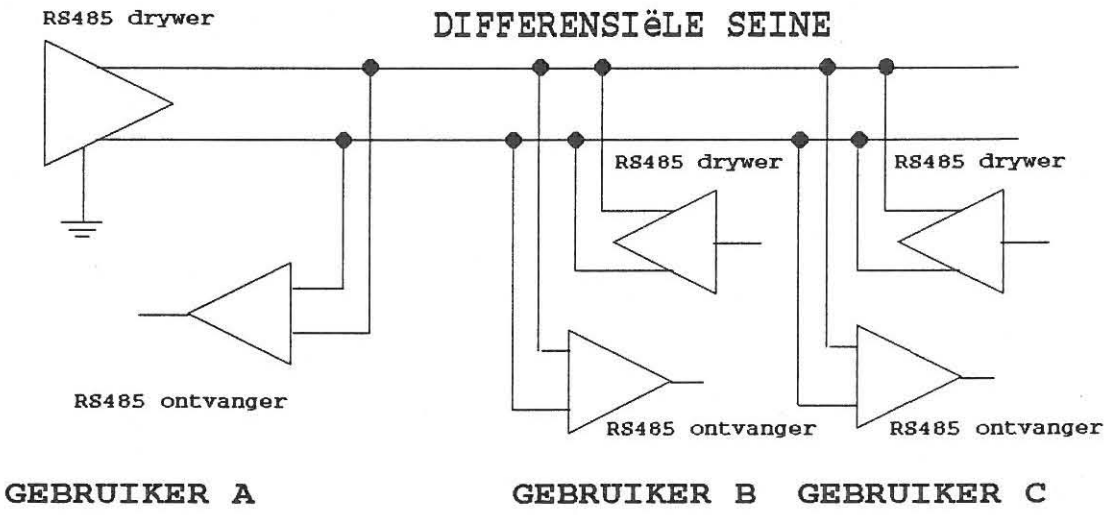


Fig. 2.11 - Blokdiagram van 'n tipiese RS485 datakommunikasiestelsel

Die seinspanningsvlakke verskil van dié van die RS422 standaard. 'n Seinspanningsvlak van +1.5V stel 'n logika '0', en -1.5V stel 'n logika '1' voor. Die ontvanger maak van +200mV en -200mV spanningsvlakke gebruik om te bepaal watter logikavlakke ontvang is. Die differensiële lyne verseker sodanige ruisimmuniteit dat hierdie klein besluitnemingsvlak voldoende is. Die maksimum datatransmissiesnelheid is 10 Mbis/s vir die RS485 standaard.

Tabel 2.3 gee 'n opsomming van genoemde standaarde, soos voorgehou deur Horowitz & Hill (1980:727).

Tabel 2.3 - Seriale datakommunikasiestandaarde & protokols

	RS232	**RS423**	**RS422**	**RS485**
Modus	enkelent	enkelent	differensieel	differensieel
Maks aantal drywers/ ontvang-ers	1 1	1 10	1 10	32 32
Maks kabel-lengte	15m	1200m	1200m	1200m
Maks data-spoed (bis/s)	20k	100k	10M	10M
Trans-missie-vlakke	± 5V min ±25V max	±3.6V min ±6.0V max	±2V min (diff'l)	±1.5V min
Ontvang-sensitiwi-teit	±3V	±0.2V	±0.2V	±0.2V
Lasimpe-dansie	3k tot 7k	450Ω min	100Ω min	60Ω min
Uitset-stroom limiet	500mA mbt VCC of GND	150mA mbt GND	150mA mbt GND	150mA mbt GND 250mA mbt -8 / -12V
Drywer Zuit, min (geen krag)	300Ω	60k	60k	120k

2.2.2.6 Foutopsporing in seriale datakommunikasiestelsels

Om 'n foutvrye datakommunikasiestelsel daar te stel moet daar tegnieke wees om te bepaal of foutiewe data ontvang word. Verskeie tegnieke is oor die jare ontwikkel en slegs die mees algemene word vervolgens bespreek.

2.2.2.6.1 Pariteit

'n Pariteitstoetsbis is 'n oortollige toetsbis wat in die datawoord/string ingevoeg is op so manier dat die totale aantal logika 1e in die datawoord/string, insluitende die pariteitsbis, 'n gelyke of ongelyke aantal verteenwoordig (Uffenbeck, 1987:513).

Pariteitsfoutopsporing kan in 'n vertikale sowel as horisontale rigting toegepas word. Beide tegnieke word vervolgens afsonderlik bespreek.

2.2.2.6.1.1 Vertikale oortolligheidskontrole

Vertikale oortolligheidskontrole - VRC (vertical redundancy check) - is 'n baie algemene vorm van foutopsporing in asinkrone seriale datakommunikasiestelsels. Een rede hiervoor is dat die sturende USART (universal synchronous/asynchronous receiver/transmitter) baie maklik 'n ekstra bis in die datawoord/string kan byvoeg. Hierdie bis kan met die minimum addisionele hardeware evalueer word by ontvangs. Indien dit saam met ASCII-gekodeerde data gebruik word, vorm hierdie bis die 8ste bis van die karakter om

sodoende die aantal logika 1e ewe of onewe te maak (Uffenbeck, 1987:514).

2.2.2.6.1.2 Longitudinale oortolligheidskontrole

Longitudinale oortolligheidskontrole - LRC (logitudinal redundancy check) - funksioneer anders as VRC wat slegs 'n ongelyke aantal foutbisse kan opspoor. Indien daar na die volgende voorbeeld gekyk word kan die probleem duidelik uitgewys word:

□ Indien die waarde D1H gestuur word, maar die waarde van D2H word ontvang, word die pariteit nie geaffekteer nie, hoewel daar 2 bisse foutief is. Daarom word VRC veral gebruik in stelsels waar meervoudige foutbisse onwaarskynlik is (Uffenbeck, 1987:515).

Tipies kan die transmissie van data tussen twee rekenaars oor 'n gedraaide paar (twisted pair) geleiers of afgeskermde kabel wees. Analise van datafoute oor lang verbindings het aan die lig gebring dat datafoute in sarsies (bursts) voorkom (Uffenbeck, 1987:515). Teen 'n datatransmissiespoed van 9600 baud sal 20 tot 25 bisse in 2ms gesend word. Dus word verskeie karakters deur 'n sarsie ruisfoute, byvoorbeeld in die geval van weerlig, geaffekteer.

Om hierdie rede word 'n oortollige blokkontrolekarakter (BCC - block checking charater) of kontrolesom (checksum) gebruik om meervoudige bisfoute op te spoor. Die blokkontrolekarakter kan as 'n LRC paritietswoord bereken word soos in Figuur 2.12 getoon, of dit kan deur sagteware as die 2s komplementsom van al die voorafgaande grepe in die datablok bereken word.

	← EEN DATABLOK →							
↑	b0	b0	b0	b0			p0	↑
	b1	b1	b1	b1			p1	
	b2	b2	b2	b2			p2	*LRC*
BIS	b3	b3	b3	b3			p3	
	b4	b4	b4	b4			p4	*PAR*
POSI-	b5	b5	b5	b5			p5	
SIES	b6	b6	b6	b6			p6	*KARA-*
	b7	b7	b7	b7			p7	*TERS*
↓	**P**	**P**	**P**	**P**	**P**	**P**		↓
	← *VRC PARITEITSBISSE* →							

Fig. 2.12 - VRC en LRC pariteitsfoutopsporing

Die volgende voorbeeld illustreer die gebruik van die kontrolesom op 4 heksadesimale grepe nl. 10H, 23H, 45H en 04H. Eerstens word die som van hierdie getalle bereken:

- 10H+23H+45H+04H=7CH.
- Die inverse van die getal 7CH word nou bepaal en daar word 1 hierby getel om die tweeskomplement te vorm: 83H+1H=84H.
- Hierdie toetssom word dan saam met die datablok gestuur om deur die ontvanger ontsyfer te word.

By die ontvangskant word die data as volg gedekodeer. Indien die volgende ontvang is : 10H, 23H, 45H, 04H, 84H, kan as volg bepaal word of die datablok korrek is al dan nie:

- Die ontvanger hoef slegs die 5 grepe op te tel - 10H+23H+45H+04H+84H=00H.

Indien die oordragvlag (carry flag) geïgnoreer word is die resultaat 00H - wat 'n foutvrye kondisie aandui. Enige ander antwoord dui op 'n datafout.

Hoewel die BCC oor die vermoë beskik om meervoudige bisfoute aan te dui, het dit ook 'n verdere voordeel. Vir 'n 255-greep datablok is slegs een greep (of 0.4%) oortolligheid nodig. VRC benodig egter een addisionele bis per greep. Dit bring 'n 12.5% oorhoofse oortolligheid mee. Die BCC is dus 'n baie geskikte tegniek om meervoudige bisfoute op te spoor sonder om die datatransmissiespoed noemenswaardig te beïnvloed.

BCC is egter nie perfek nie. Wanneer 'n fout opgespoor word moet die hele datablok weer gestuur word - anders as VRC waar slegs een karakter weer gestuur hoef te word. Daar is ook foutkombinasies wat nie deur BCC opgespoor kan word nie. Indien die vorige voorbeeld weer in oënskou geneem word en die data 45H na 44H verander en 04H na 05H, bly die BCC onveranderd. Algemeen gesproke, moet gelyke hoeveelhede bisse in dieselfde posisie verander om nie deur die BCC metode bespeur te word nie.

Figuur 2.12 dui die voorstelling van LRC en VRC aan. Soos uit Figuur 2.12 blyk word een VRC bis by elke karakter bygevoeg, en een LRC karakter per datablok.

'n Kombinasie van die LRC en VRC metodes word ook soms gebruik.

2.2.2.6.2 Sikliese oortolligheidskontrole

Sikliese oortolligheidskontrole, CRC (cyclic redundancy check), is nog 'n vorm van longitudinale oortolligheid wat op 'n blok data bereken word. Dit word algemeen gebruik wanneer data na/vanaf 'n slapskyf of starre/harde skyf (rigid disk) geskryf/gelees word en om die integriteit van data in programmeerbare leesalleengeheues te verseker. Dit word ook universeel gebruik om foute in sinkrone datakommunikasiestelsels op te spoor (Uffenbeck, 1987:517).

Anders as die kontrolesom/kontrolekarakter (BCC) metode is die CRC metode nie greep georïenteer nie. Die datablok word gesien as 'n string van seriale databisse. Die bisse in hierdie *n*-bis blok word beskou as die koëffisiënt van die karakteristieke polinoom - gewoonlik na verwys as M(X) - M van X.

M(X) het die volgende vorm:

$$M(X)=b_n+b_{n-1}X+b_{n-2}X^2+.....+b_1X^{n-1}+b_0X^n$$

waar:

b_0 die mins beduidende bis (LSB) en

b_n die mees beduidende bis (MSB) is.

Die volgende voorbeeld illustreer die gebruik van bogenoemde formule indien die polinoom M(X) van die 16-bis datastring 26F0H bepaal moet word:

□ Stap 1: *Omskakeling van datastring na binêr*:

Binêr = 0010 0110 1111 0000.

□ Stap 2: *Skryf M(X) as*:

$$M(X)=0+0X^1+1X^2+0X^3+0X^4+1X^5+1X^6+0X^7+1X^8 +1X^9+1X^{10}+1X^{11}+0X^{12}+0X^{13}+0X^{14}+0X^{15}$$

□ Stap 3: *Eliminering van al die 0-terme:*

$$M(X) = X^2 + X^5 + X^6 + X^8 + X^9 + X^{10} + X^{11}$$

Bogenoemde formule is 'n unieke weergawe van die data in die 16-bis datablok. Indien een of meer databisse sou verander, sou die polinoom ook verander. Die CRC word nou verkry deur die volgende formule toe te pas:

$$\frac{M(X)*X^n}{G(X)} = Q(X) + R(X)$$

waar:

G(X) die generator polinoom heet en

n is die aantal bisse in die blok.

Vir die bisinkrone (bisync) protokol word G(X) gegee as:

$$G(X) = X^{16} + X^{15} + X^2 + 1$$

Indien die deling plaasvind is die resultaat 'n kwosiënt (quotient) Q(X) en reswaarde (remainder) R(X). Die CRC tegniek behels die berekening van R(X) vir die datastring. Hierdie waarde word dan by die datastring gevoeg tydens versending. Indien R(X) deur die ontvanger bereken word, moet R(X) = 0. Let op dat hoewel G(X) 'n term tot die mag 16

bevat, het die reswaarde R(X) nooit 'n term met 'n hoër mag as 15 nie. Dit verteenwoordig dus 2 grepe, ongeag die lengte van die datablok.

Die volgende voorbeeld illustreer die gebruik van die bisinkrone tegniek om die reswaarde van die datablok 26F0H te bepaal. Vanuit die gegewens word die volgende vergelyking opgestel om die waarde van R(X) te bepaal:

$$\frac{M(X)*X^{16}}{G(X)}=\frac{X^{27}+X^{26}+X^{25}+X^{24}+X^{22}+X^{21}+X^{18}}{X^{16}+X^{15}+X^{2}+1}$$

Kragtens Uffenbeck lewer hierdie uitdrukking die reswaarde soos hieronder (1987:520).

$$R(X) = X^{15} + X^{13} + X^{9} + X^{8} + X^{6} + X^{4} + X^{3} + X + 1$$

Indien dit na binêr omgeskakel word, met inagneming daarvan dat die koëffisiënt van die hoogste mag die LSB word, word die volgende waarde gekry : 1101 1010 1100 0101 = DAC5H (Uffenbeck, 1987:519).

Indien die twee grepe DAC5H by 26F0H gevoeg word sal die ontvangde CRC berekening die resultaat van R(X)=0 lewer, wat geen bisfoute aandui nie. Indien daar enige foute voorkom moet die totale datablok weer hersend word, net soos in die geval van BCC.

Indien daar 'n CRC lengte van *n*-bisse aanvaar word, kan die effektiwiteit van CRC as volg saamgevat word :

- Alle datablokke met 'n ewe of onewe aantal foute word uitgewys, afhangend daarvan of *n* ewe of onewe is.

- Alle datablokke met sarsiefoute minder as *n*-bisse lank word geïdentifiseer.
- Alle datablokke met 'n totale aantal foutbisse minder as ongeveer *n/4* word bespeur.
- Van *alle* oorblywende foutpatrone is daar 1 foutpatroon uit 'n moontlike 2^n wat nie geïdentifiseer kan word nie.

Die SDLC (synchronous data link control) protokol gebruik die volgende uitdrukking vir G(X):

$$G(X) = X^{16} + X^{13} + X^5 + 1$$

In die praktyk word daar hedendaags van die Intel 8273 programmeerbare HDLC/SDLC protokolbeheerder gebruik gemaak om CRC foute aan die ontvangskant te bespeur. Dit voeg ook R(X) outomaties by die datastring gedurende datatransmissie. Die CRC word dus "deursigtig" (transparent) vir die gebruiker.

2.2.3 Netwerk datakommunikasie (LANs)

Voorheen het rekenaars in bondelmodus (batch mode) gefunksioneer. Hierdie rekenaars was baie kragtig, maar stadiger as selfs die mins-kragtige persoonlike rekenaars wat heedensdaags beskikbaar is. Hierdie rekenaars was ook besonder duur.

Hedendaags word daar van netwerke gebruik gemaak om toegang tot dieselfde stelsel aan meer as een gebruiker te verleen. Een van hierdie rekenaarnetwerke staan as lokale areanetwerke (LANs) bekend. 'n LAN

word gedefinieer as 'n netwerk wat spesifiek ontwerp is om oor 'n relatief klein fisiese area, bv. 'n kantoor of fabriek, te kan funksioneer (Schweber, 1988:386).

Met die ontwikkeling van LANs is die koste verbonde aan rekenaarnetwerke van soortgelyke aard aansienlik verminder. Hierdie LANs word spesifiek gebruik om hoë datasnelhede (bo 19200 baud) te hanteer.

2.3 Datakommunikasiesagteware

Soos reeds genoem word sagteware benodig om die datakommunikasiestelsel te beheer. Hierdie sagteware is baie nou verweef met die datakommunikasiehardeware en dus vorm dit 'n essensiële deel van die datakommunikasiestelselstruktuur.

Die datakommunikasiesagteware soos ontwikkel tydens die uitvoering van hierdie projek word in Hoofstuk 4 uiteengesit.

2.4 Opsomming

In datakommunikasie word die klem veral gelê op die afstand waaroor daar gekommunikeer moet word, datatransmissiespoed en ruisimmuniteit.

Parallelle datakommunikasie is geskik vir module-na-module verbindings waar daar nie oor lang afstande gekommunikeer moet word nie. Dit is ook relatief duur aangesien daar een verbinding per bis benodig word. Dit laat dus die weg oop om 'n besluit te neem oor die gebruik van seriale datakommunikasie of die gebruik van netwerkdatakommunikasie.

As gevolg van die beperkte omvang van die projek is LANs nie verder oorweeg tydens die uitvoering daarvan nie.

Aangesien RS232 baie kwesbaar is vir ruis en vanweë die hoë koste verbonde daaraan om modems te gebruik om verder as 15m te kan kommunikeer, is daar teen die gebruik van hierdie standaard besluit. Dit is ook onnodig geag om te moduleer aangesien dit 'n privaatlynkonfigurasie datakommunikasiestelsel gaan wees wat slegs oor 'n baie smal bandwydte gaan werk.

RS423 lewer wel die fasiliteit om oor 1200m te kommunikeer, maar dit kan slegs in 'n eenrigting datakommunikasiestelsel gebruik word. RS422 is 'n baie goeie keuse vir gebruik in 'n toepassing soos behandel in hierdie navorsingsprojek. Dit beskik egter nie oor die fasiliteit om multitapkonfigurasies moontlik te maak nie en daarom is daar ook hierteen besluit.

Die RS485 datakommunikasiestandaard is dus die gewenste keuse vir die gebruik in 'n stelsel soos ontwikkel aangesien dit in 'n multitapkonfigurasie gebruik kan word. Elke mikrorekenaarnode kan d.m.v. die selekteermodus geadresseer word om data daarheen te stuur en data daarvanaf te versoek. Hierdie standaard het ook 'n baie goeie ruisimmuniteitsfaktor, wat dit 'n baie goeie keuse vir 'n datakommunikasiestelsel maak. Hierdie protokol kan ook in die halfduplekse modus gebruik word wat dan net 2 fisiese verbindings benodig.

Daar is ook verwys na verskeie datafoutopsporingstegnieke vir moontlike gebruik in datakommunikasiestelsels. Hierdie navorsingsprojek gebruik egter geen datafoutopsporingstegniek nie aangesien die konfigurasie van so 'n aard is dat dit nie foutopsporing regverdig nie.

DATAKOMMUNIKASIESTELSEL HARDEWARE

Die blokdiagram van die datakommunikasiestelsel sien as volg daar uit :

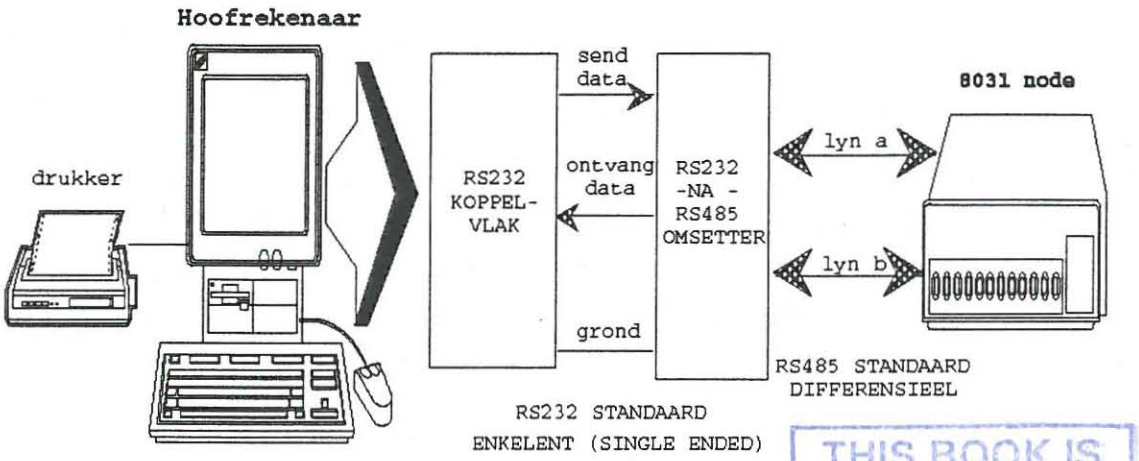


Fig. 3.1 - Blokdiagram van datakommunikasiestelsel

Hierdie blokdiagram verteenwoordig die totale datakommunikasiestelsel en funksioneer as volg :

Die hoofrekenaar bevat die datakommunikasiesagteware en databasishantering en -beheersagteware om beheer uit te oefen oor die 8031 nodes aan die eindpunte van die datakommunikasielyn. Die hoofrekenaar kommunikeer met die 8031 node d.m.v. die RS232 koppelvlak. Hierdie standaard dataseine word deur 'n RS232-na-RS485 omsetter na die RS485 standaard omgeskakel sodat dit aanvaarbaar is vir die 8031 node. Die 8031 node is 'n alleenstaande mikrorekenaar en voer vasgestelde kontrolefunksies uit. Dit stuur ook data terug na die hoofrekenaar op aanvraag.

Die hardeware bestaan basies uit die volgende modules:

- Meerdoelige kragbron,
- RS232-na-RS485 omsetter,
- 8031 mikrorekenaar,
- Inset/Uitset (I/U) module vir die 8031 node.

Die funksie, ontwerp en werking van elk word vervolgens bespreek.

3.1 Meerdoelige kragbron

Bylaag A.1 toon die skematiese voorstelling van die meerdoelige kragbron.

Die kragbron verskaf +5V en +15V spannings teen 'n maksimum stroomlewering van 500mA. Beide spannings word verkry deur gebruikmaking van LM317T spanningsreguleerders. Die +5V voorsien krag aan die geïntegreerde kringe en die +15V dryf die optiese isoleerderkringe aan.

Twee ligemissiediodes (LEDs) verskaf visuele bevestiging van die twee spannings. 'n 500 mA sekering aan die primêre kant van die transformator verskaf ook beskerming teen oorlading.

3.2 RS232-na-RS485 omsettermodule

Bylaag A.2 toon die skematiese voorstelling van die RS232-na-RS485 omsettermodule.

Hierdie module skakel die seriale RS232 standaard dataseine vanaf die hoofrekenaar om na RS485 standaard dataseine. In dié formaat word dit via die datakommunikasielyn na die verskillende mikrorekenaarnodes gestuur. Dit skakel ook die ontvangde RS485 dataseine vanaf die datakommunikasielyn terug na RS232 formaat vir koppeling aan die hoofrekenaar.

Die omskakelingsproses geskied deur middel van 'n SN75176 geïntegreerde kring. Figuur 3.2 toon die blokdiagram van hierdie kring:

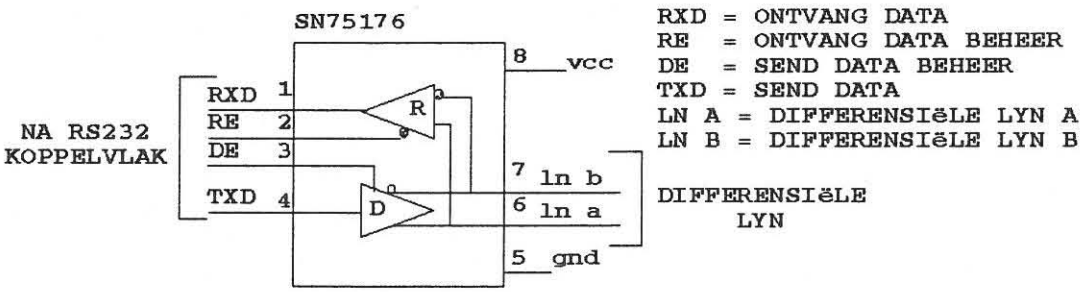


Fig. 3.2 - Blokdiagram van die SN75176 geïntegreerde kring

Hierdie kring is 'n gekombineerde halfduplekse differensiële sender/ontvanger en verskaf ook 'n hoë-impedansie uitset. Die basiese werking van die module is as volg:

Dit is voortdurend in die ontvang-gereed toestand sodat data vanaf die datakommunikasielyn deur die hoofrekenaar ontvang kan word. Hierdie toestand bepaal dat die RE/DE beheerlyne by 'n logika '0' vlak moet wees sodat die ontvanger in die gereed toestand kan wees en die sender onaktief is. Dit word bewerkstellig deurdat die logika '0' peil vanaf die Q-uitset van die 74121 geïntegreerde kring (Bylaag A.2, U3 - monostabiele multivibrator) verkry word in sy rustoestand. Indien daar data vanaf die hoofrekenaar na die nodes versend moet word sal die module as volg funksioneer:

Die send-data lyn (TXD) is by 'n logika '0' peil tydens normale lyntoestande. Wanneer die hoofrekenaar data stuur word 'n datawoord van 10 bisse saamgestel wat uit 1 beginbis (start bit), 8 databisse en 'n stopbis bestaan. Wanneer die beginbis gestuur word verander die lyn se logikavlak na 'n logika '1' peil. Hierdie sein word deur die optiese isoleerderkring (CNY74-4 - U1) gestuur en op pen 5 (pen B) van die 74121 kring aangelê. Weens die stygende peil van die sein word die 74121 gesneller en verskaf dit 'n uitsetpuls met 'n tydsduur van ongeveer 15ms. Hierdie tydsduur word bepaal deur die tydkonstante van R5 en C1 soos gekoppel in die skematiese voorstelling.

Hierdie tydsduur word deur die volgende formule gegee:

$$TW(UIT)=0.7*Rt*Cext$$ (ongeveer)

waar *Rt* die eksterne weerstand is (R5) en

Cext die eksterne kapasitor (C1), (Elektor, 1983:174).

Vir hierdie tydsduur word daar 'n logika '1' peil op die RE/DE beheerlyne aangelê wat die sender vir dié tydsduur aanskakel (send-toestand) en die data kan na die nodes gestuur word. Hierdie tydsduur verskaf 'n konstante uitset wat voorsiening maak daarvoor dat die hele tydsduur - of slegs 'n gedeelte daarvan - gebruik kan word om die 10 databisse te stuur. Dit maak dus voorsiening vir 'n lae dataoordragsnelheid, asook 'n hoë dataoordragsnelheid.

Sodra die send-toestand verstryk keer die module na die ontvang-gereed toestand terug en kan dataontvangs normaalweg geskied. Die differensiële uitset van die kring impliseer dat die spanning tussen die twee uitsetlyne (lyn-a en lyn-b) ontwikkel word en nie 'n grondkoppeling as verwysing gebruik nie. Die seinspanningsvlakinterpretasie by die ontvangskant is as volg :

'n Logika '1' word verteenwoordig deur 'n spanningsverskil van groter as 0.2V tussen lyn-a en lyn-b. Indien die verskil in spanning tussen lyn-a en lyn-b meer negatief is as -0.2V (lyn-b se potensiaal is groter as lyn-a se potensiaal) word 'n logika '0' aanvaar. Indien die verskil in spanning tussen die twee lyne kleiner is as 0.2V val hierdie waarde in die ongedefinieerde streek (Schweber, 1988:303).

Die optiese isoleerderkring (CNY 74-4 - U1) bied optiese isolasie tussen die inset- en uitsetlyne van die hoofrekenaar en die RS232-na-RS485 omsetter. Dit kan tot 5000V isolasie verskaf en die hoofdoel hiervan is om die hoofrekenaar teen lyntransiënte te beskerm. Daar moet gelet word op die feit dat die weerstande op die uitset van die optiese isoleerder kring (Bylaag A.2 - R3, R4) 560Ω is, inteenstelling met die 220Ω weerstande wat in die Inset/Uitset module kring (soos later bespreek) voorkom. Die rede hiervoor is dat die impedansie van die 74121 en SN75176 kringe dit genoodsaak het om hierdie waardes dienooreenkomstig te verander sodat die korrekte spanningsvlakke na die SN75176 gestuur kon word.

Die twee ligemissiediodes (LEDs) verskaf indikasie of data versend of ontvang word.

3.3 8031 mikrorekenaarnode

Die 8031 mikrorekenaarnode bestaan uit twee afsonderlike modules wat 'n eenheid vorm, nl.

- 8031 mikrorekenaar en
- Inset/Uitset module.

Vervolgens word die ontwerp, funksie en werking van hierdie twee kringe bespreek.

3.3.1 8031 mikrorekenaar

Bylaag A.3 toon die skematiese voorstelling van die 8031 mikrorekenaar.

Hierdie kring is die kern van die afgeleë node wat die data vanaf die hoofrekenaar moet ontvang, dit verwerk en daarvolgens reageer. Verder moet dit ook vasgestelde funksies - ooreenkomstig die programmering daarvan - verrig.

Hierdie kring bestaan basies uit die volgende :

- 'n Mikrobeheerder (8031 - Bylaag A.3, U1),
- Adreshoukring (LATCH 74LS373 - Bylaag A.3, U2),
- 16K Leesalleengeheue (LAG) (2764 - Bylaag A.3, U3),
- 8K Willekeurige toeganggeheue (WLSG) (6264 - Bylaag A.3, U4)
- Adresdekodeerder (74LS138 - Bylaag A.3, U5).

Die mikrobeheerder (Bylaag A.3, U1) wat gebruik word is die INTEL 8031 geïntegreerde kring. Dit beskik oor 4 inset/uitset poorte (P0 tot P3) waarvan P0 terselfdertyd die lae orde gemultiplekseerde adres/databus is en P2 die hoë orde adresbus verteenwoordig (Intel, 1991a:5-3,5-6). P1 kan as 'n meerdoelige inset/uitset poort gebruik word. P3 dien as die gespesialiseerde poort met 2 onderbreekinsette (interrupt inputs), 2 tellerinsette (counter inputs) en 'n volduplekse RS232 datakommunikasiepoort (Intel, 1991b:2-2).

In hierdie toepassing word bisse 0 tot 4 van P1 gebruik om die unieke adres van die node te verteenwoordig. Hierdie adres word verkry vanaf DIP-skakelaars wat op die Inset/Uitset module monteer is. Onmiddellik na aanskakeling word die adres van die node ingelees vanaf hierdie poort en gestoor sodat hierdie

node slegs sal reageer op versoeke vanaf die hoofrekenaar indien dit geadresseer word.

Bisse 5 en 6 word gebruik om die RE en DE beheerlyne van die RS232-na-RS485 omsetter op die Inset/Uitset module te beheer sodat dit data kan send of ontvang. Bis 5 word gebruik om die DE beheerlyn (Send data) te stel of te herstel afhangend van omstandighede. Bis 6 word vir die RE beheerlyn (Ontvang data) gebruik onder beheer waarvan data ontvang word.

Die mikrobeheerder gebruik 'n 11.0592 MHz kristal. Sodoende kan daar heelgetalwaardes verkry word vir die veranderlike wat in die sagteware gebruik moet word om die datakommunikasiesnelheid van die node te bepaal. Die volgende formule illustreer die bepaling van die veranderlike vir die datakommunikasiesnelheid van die node soos in hierdie toepassing van krag:

$$BR = (2^{smod})/32*(T1\_OFR)$$

waar

BR die datakommunikasiesnelheid is,

smod die bis in die PCON register is wat aandui of die datakommunikasiesnelheid verdubbel moet word, al dan nie en

T1_OFR teller 1 se oorvloeitempo aandui (Intel, 1991a:7-17).

Dus is T1_OFR die veranderlike wat hieruit bepaal moet word.

Die kring het ook 'n analoog hardewareherstelkring wat die node laat terugkeer na die oorspronklike staat indien dit verlang word. Hierdie kring maak ook voorsiening daarvoor dat die mikrobeheerder voldoende tyd het om by kragaanskakeling tot operasionele vlak te kom voordat dit die voorafbepaalde

aanskakelroetines begin uitvoer. Aangesien dit 'n analoog herstelkring is, kon dit nie direk gebruik word om die 8255PPI geïntegreerde programmeerbare randapparatuurkoppelvlak te herstel nie, en daarom is P1 bis nr. 7 vir hierdie doel ingespan. Vanaf hierdie pen word 'n TTL uitset verkry wat op die 8255PPI se herstelinset aangelê word. Sodra die 8031 mikrobeheerder begin met die aanskakelingsroetines, word hierdie pen na 'n logika '0' peil geskakel om die einde-van-herstel vir die 8255 geïntegreerde kring aan te dui.

Daar is ook 'n meerdoelige LED aan die mikrobeheerder verbind om voorsiening te maak vir foutindikasie en toetsdoeleindes. Hierdie LED word geadresseer deur 'n waarde van '0' of '1' na bis T0 van P3 te skryf.

Die adreshoukring (Bylaag A.3, U2) word op die gemultiplekseerde adres/databus geplaas sodat die lae orde adres van 'n bewerking of Inset/Uitsetadres op die uitset hiervan teenwoordig sal wees na die eerste klokpuls van 'n bewerking. Vanaf die tweede klokpuls word data op die inset hiervan aangetref. Die data kan egter nie die lae orde adres op die uitset van die kring affekteer nie.

Die leesalleengeheue (Bylaag A.3, U3) het 'n kapasiteit van 16K grepe (131072 bisse) en word gebruik om die BIOS-program (Basiese Inset/Uitset program), sowel as al die ander funksies wat die node moet verrig, te stoor. Indien die node herstel word, of vir die eerste keer bekragtig word, word die eerste instruksie op adres 0000H uitgevoer. Die adresstrek van hierdie geheue is vanaf 0000H tot 3FFFH wat 16K kapasiteit verteenwoordig.

Die willekeurige toeganggeheue (Bylaag A.3, U4) het 'n kapasiteit van 8K grepe (65536 bisse) en word gebruik om datastrukture en veranderlikes in te stoor. Die adresstrek van hierdie geheue is vanaf 4000H tot by 5FFFH.

Die adresdekodeerder (Bylaag A.3, U5) bepaal die adresse van die LAG en WLSG en ook van ander kringe wat aan hierdie node verbind word. Uitset Y6 word gebruik om die adresstrek vir die I/U poort op die I/U module te bepaal, nl. C000H tot FFFFH.

Die node word met 'n 26 pen koppelvlak (Bylaag A.3, J1) aan die I/U module verbind.

Soos blyk uit Bylaag A.3 is daar geen datalyndrywers in hierdie ontwerp teenwoordig nie. Aanvanklik is dit as onnodig geag, maar eksperimentering het die wenslikheid van die gebruik daarvan om die Inset/Uitset module aan te dryf na vore gebring. Derhalwe is die Inset/Uitset module met hierdie kringe toegerus om onnodige kringherbouing uit te skakel.

'n Verdere probleem wat na eksperimentering en analisering na vore gekom het is dat data aanvanklik nie korrek vanaf die WLSG gelees is nie. Nadat 'n logika analiseerder aan die kring gekoppel is, is vasgestel dat die leesbeheerlyn (RD-) onstabiel raak voordat die leessiklus ten einde kom.

Hierdie verskynsel kan in Bylaag A.8 gesien word. Hierdie probleem is oorkom deur die MRD- beheerlyn (i.p.v. RD-) - wat die LAG geïntegreerde kring vir leesdoeleindes beheer - (Bylaag A.3 - Pen 3 van U6A) ook op die WLSG kring aan te lê.

Bylaag A.4 toon die alternatiewe ontwerp - met insluiting van bg. buffers - van die node aan.

3.3.2 Inset/Uitset module

Bylaag A.5 toon die skematiese voorstelling van hierdie kring.

Die doel van hierdie module is om die inset- en uitsetfunksies van die 8031 mikrorekenaar (parallel en seriaal) te help beheer.

Die kring bestaan basies uit die volgende :

- 'n DIP-skakelaarkonfigurasie (Bylaag A.5, SW1),
- RS232-na-RS485 omsetter (Bylaag A.5, U1 -SN75176),
- Optiese isoleerderkring (Bylaag A.5, U2 -CNY74-4),
- Programmeerbare Randapparatuurkoppelvlak (Bylaag A.5, U3 -8255PPI),
- Tweerigting datalyndrywer (Bylaag A.5, U4 -74LS245),
- Eenrigting datalyndrywer (Bylaag A.5, U5 -74LS244).

Die DIP-skakelaarkonfigurasie (Bylaag A.5, SW1) bepaal die unieke adres van die mikrorekenaarnode - wat kan wissel van 00h tot 1FH - deur die skakelaars so te skakel dat die toepaslike adres verkry word.

Die RS232-na-RS485 omsetter funksioneer soortgelyk aan die RS232-na-RS485 omsettermodule wat reeds bespreek is, maar verskil as volg :

- Dit word uitsluitlik deur die mikrobeheerder beheer en bespeur nié outomaties of die node data wil stuur al dan nie.
- Die node plaas dit outomaties in die ontvang-gereed toestand en dit word eers in die send-gereed toestand geplaas indien die node data het om na die hoofrekenaar terug te stuur.

'n Probleem wat na vore gekom het was dat data ongeldig was by ontvangs op die RXD-inset van die mikrobeheerder. Na evaluering is vasgestel dat die

RS232 drywers van omkeerderdrywerkonfigurasies gebruik maak. Om hiervoor te kompenseer is daar 'n omkeerderkring op beide die TXD en RXD lyne in hierdie module aangebring (Bylaag A.5, U6a en U6b).

Die optiese isoleerder (Bylaag A.5, U2) isoleer die datakommunikasielyn vanaf die mikrorekenaarnode en funksioneer soos reeds bespreek. Twee LEDs word ook van hier gedryf om aan te dui of daar data deur die node gestuur of ontvang word. Die optiese isoleerders lewer 'n +5V uitset indien daar 'n logika '1' op die inset daarvan voorkom.

Die 8255 programmeerbare randapparatuurkoppelvlak (8255PPI), (Bylaag A.5, U3), word gebruik as parallelle dataskakel tussen die mikrorekenaar en die toetsmodule. Dit het tipies 3 inset/uitset poorte (Poort A tot Poort C) waarvan PA en PB slegs inset- of uitsetpoorte kan wees. Poort C verskaf die moontlikheid om as 2 vier-bis poorte saam met PA en PB gebruik te kan word. Dit verskaf ook die fasiliteit om hierdie komponent as 'n inset/uitset poort te kan gebruik met aansluitbevestiging (handshaking) indien dit verlang word (Verburg, 1989:231).

Hierdie kring word egter só geprogrammeer dat Poort A en Poort B as uitsetpoorte dien en Poort C as 'n eenvoudige insetpoort. Die rede hiervoor is dat 'n toetsmodule ontwerp is vir gebruik in hierdie konfigurasie.

Die tweerigting datalyndrywer (Bylaag A.5, U4) buffer die inkomende sowel as die uitgaande data op die databus. Dit word bewerkstellig deur die RD-sein op die DIR beheerlyn aan te lê om te bepaal of die data vanaf A na B gestuur moet word (RD- hoog) en of data vanaf B na A gestuur moet word (RD- laag). Hierdie buffer voorsien dat die seinvlak onderhou word en dus dat alle data op die databus na én vanaf die I/U module geldig en korrek sal wees.

Aanvanklik is die hek-beheerlyn (Bylaag A.5, U4 pen 19) van hierdie kring direk aan grond verbind, maar met die koppeling van die module aan die 8031 mikrorekenaarmodule is ondervind dat data nie vanaf die WLSG gelees kon word nie. Met behulp van 'n logika analiseerder is vasgestel dat die data op die databus onstabiel raak tydens die leessiklus, soos blyk uit Bylaag A.9.

Daar is vasgestel dat die oorsaak van die onstabiele data op die databus daaraan te wyte is dat hierdie drywer nie in sy hoë-impedansie modus geplaas word nie. Om die probleem te oorkom is die grondkoppeling verwyder vanaf die G pen en is dit aan CS- van die 8255PPI kring verbind. Slegs wanneer daar toegang tot die 8255PPI verkry moet word, sal hierdie kring dan geaktiveer word. In enige ander stadium sal daar 'n hoë-impedansie toestand op die inset, sowel as die uitset teenwoordig wees wat nié die res van die kring kan beïnvloed nie.

Die eenrigting datalyndrywer (Bylaag A.5, U5) buffer slegs die beheerseine aan die 8255PPI asook die TXD en RXD seine vanaf die mikrorekenaarnode. Dit verseker dus dat alle beheerseine vanaf die mikrorekenaarnode geldig is, maar ook dat die inkomende data op die seriale poort geldig en korrek aan die mikrorekenaar weergegee word.

Die doel van die twee 26 pen koppelvlakke is as volg :

- J1 (Bylaag A.5) verskaf die verbinding tussen die mikrorekenaarnode en die Inset/Uitset module.
- J2 (Bylaag A.5) verskaf die verbinding tussen die Inset/Uitset module en 'n toetsmodule vir evaluering van die stelsel.

Die alternatiewe I/U kring wat saam met Bylaag A.4 gebruik kan word kan in Bylaag A.6 gesien word. Hierdie ontwerp sluit datalyndrywers uit aangesien dit deel van die kring soos in Bylaag A.4 uitmaak.

3.4 Opsomming

Soos uit voorafgaande gesien kan word funksioneer al hierdie modules as 'n eenheid om die 8031 mikrorekenaarnode te vorm. Die kragbron verskaf die nodige spannings om die geïntegreerde kringe aan te dryf met 'n maksimum stroomlewering van 500mA. 'n Stroommeting is egter gedoen en die node gebruik 'n maksimum van ongeveer 280mA. Verdere uitbreidings is dus ook moontlik sonder om van 'n ander kragbron gebruik te maak.

Die 8031 mikrorekenaar en I/U module vorm die kern van die 8031 mikrorekenaarnode. Die RS485 datakommunikasie word moontlik gemaak deur 'n geïntegreerde kring, SN75176, wat in 'n halfduplekse modus funksioneer. Hierdie kring is altyd in die ontvang-gereed stand en word deur die mikrobeheerder in die send-toestand geplaas wanneer dit data het om te versend.

Die mikrorekenaar beskik oor 16K LAG en 8K WLSG. Dit beskik ook oor 'n 8255PPI wat parallelle datakommunikasie met 'n toetsmodule moontlik maak. Die geheuespasie word in vier 16K blokke opgedeel en die geheuekaart word deur Tabel 3.1 aangetoon.

Tabel 3.1 - Geheuekaart van 8031 mikrorekenaar

BEGIN ADRES	EIND ADRES	GEBRUIKER VAN SPASIE
0000H	3FFFH	LAG (ROM) 16K
4000H	5FFFH	LSG (RAM) 8K
6000H	7FFFH	VOORSIENING VIR 16K LSG
8000H	BFFFH	VRY
C000H	FFFFH	8255 PPI SPASIE
C000H	ENKEL	8255 PPI POORT A
C100H	ENKEL	8255 PPI POORT B
C200H	ENKEL	8255 PPI POORT C
C300H	ENKEL	8255 PPI BEHEER REGISTER

Na die modifikasies, soos bespreek in punt 3.3.2, aangebring is het die onstabiliteit op die databus nie meer voorgekom nie. Bylaag 3.10 toon 'n logika analiseerder uitdruk van die korrekte kringtoestand.

HOOFSTUK 4

DATAKOMMUNIKASIESTELSEL SAGTEWARE

Die ontwikkeling van sagteware vir rekenaarstelsels is steeds 'n wesenlike probleem aangesien daar geen vasgestelde standaard daarvoor gestel is nie. Dit het aanleiding gegee tot die daarstelling van die term "sagteware ingenieurswese" in die laat 1960's op 'n konferensie wat gehou is om die sogenaamde "sagteware krisis" te bespreek (Sommerville, 1982:1).

Die sagtewarelewenssiklus gee 'n basiese riglyn vir die periode van sagtewareontwerp en gebruik. Die volgende lys dui die trappe van die sagtewarelewenssiklus aan kragtens Sommerville (1982:3):

- *Spesifikasie:* Die sagtewarebenodigdhede - dit behels die stelselfunksies en operasionele begrensing (newevoorwaardes) - word bepaal en gespesifiseer.
- *Ontwerp:* 'n Sagtewareontwerp word afgelei vanaf 'n analise van die sagtewarebenodigdhede.
- *Implementering:* Die sagteware word ontwerp in 'n programmeringstaal wat deur die teikenstelsel aanvaarbaar is.
- *Toetsing:* Die sagteware wat geïmplementeer is word getoets om te verseker dat dit voldoen aan die gespesifiseerde behoefte.
- *Gebruik en instandhouding:* Die stelsel word geïnstalleer en intensief gebruik. Indien enige stelselfoute voorkom word dit gekorrigeer. Dit bring veranderings en soms sekere byvoegings tot die oorspronklike stelsel mee.

Die ontwikkeling van sagteware is 'n iteratiewe proses met elke fase in die lewenssiklus wat inligting terugvoer na die vorige fases. Die daarstelling van 'n riglyn vir sagteware ontwikkeling, nl. Bo-af ontwerp (Top Down Design), gee die nodige

metodologie sodat 'n toepaslike sagtewarepakket ontwikkel kan word vir die instandhouding van die sagtewarelewenssiklus.

In hierdie hoofstuk word die volgende oorweeg:

- Sagteware ontwikkelingsteorie, waaronder
  - Bo-af ontwerp,
  - Evaluering,
  - Dokumentering,
  - Vlakke van programmering.
- Hoofrekenaar sagteware,
- Mikrorekenaarnode sagteware.

Bogenoemde punte word vervolgens afsonderlik bespreek.

4.1 Sagteware ontwikkelingsteorie

4.1.1 Bo-af ontwerp

Die ontwikkeling van sagteware is 'n kreatiewe proses wat nie geformuleer kan word deur 'n stel rëels nie. Deur egter van 'n sistematiese ontwerpmetodologie gebruik te maak, word die ontwerpproses vergemaklik en sagteware wat verstaanbaar, verifieerbaar en betroubaar is gelewer. Een van hierdie metodologieë staan as bo-af ontwerp of stapgewyse verfyning (stepwise refinement) bekend. Bo-af ontwerp word deur Aron (1974:128) as volg gedefinieer: Dit is die tegniek om rekenaarsagteware vir elke vlak van die ontwerp te ontwikkel en te skryf soos die ontwerp vorder.

Dit maak van die mees fundamentele menslike probleemoplossingsfasiliteit gebruik, nl. abstraktheid. Die aanvanklike idee word as 'n abstrakte entiteit beskou sonder enige detail van hoe hierdie entiteit werklik gerealiseer gaan word. Soos die ontwerp vorder word elke komponent daarvan in sy eie fundamentele funksies opgebreek. Hierdie proses word voortgesit totdat die laevlak ontwerp geformuleer is.

Die formulering en beskrywing van 'n sagtewareontwerp behels 'n aantal fases en word as volg gelys (Sommerville, 1982:52):

- *Bestudeer en verstaan die probleemstelling.* Hiersonder is effektiewe sagtewareontwerp onmoontlik.
- *Indentifiseer bruto kenmerke van ten minste een moontlike oplossing.* Hier word gewoonlik 'n aantal moontlike oplossings geïdentifiseer en elkeen word geëvalueer. Die maklikste oplossing word hieruit verkry en gekies.
- *Konstrueer 'n datavloeidiagram wat die bruto datatransformasies aandui.* Indien dit nie uitgevoer kan word nie is dit 'n aanduiding dat die probleem nie volkome verstaan word nie.
- *Konstrueer 'n struktuurkaart deur gebruik te maak van die datavloeidiagram.* Hierdie struktuurkaart moet die verskillende eenhede aandui wat die oplossing verteenwoordig.
- *Beskryf elke abstraktheid wat in die oplossing gebruik is.* Dit kan gedoen word deur 'n taal soos PDL te gebruik.

Nadat die oplossing en beskrywing vir die hoogste vlak afgehandel is, word die proses voortgesit vir elke abstraktheid wat gebruik word. Hierdie proses van verfyning duur voort totdat die laevlak spesifikasie van elke abstraktheid bereik is.

Vir elke fase van die ontwerp bestaan die hoëvlak modules alreeds voordat die laevlak modules ontwikkel word. Daar word stompprosedures (stubs) in die program aangebring om die laer vlak modules aan te dui sodat dit die programontwerp vergemaklik. Figuur 4.1 gee 'n voorbeeld van hierdie tegniek soos deur Aron (1974:129).

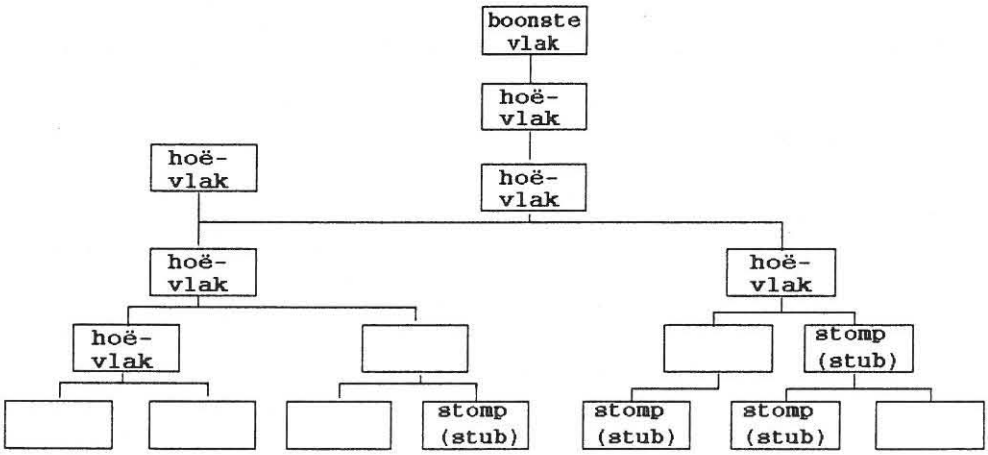


Fig. 4.1 - Bo-af ontwerp implementering

Figuur 4.1 illustreer die beginsel van bo-af ontwerp soos bespreek en hier kan gesien word hoe daar vanaf die heel boonste vlak na die laevalk prosedures gewerk word om die stelselontwerp te voltooi.

4.1.2 Evaluering

Die geldigheidsbepaling van 'n sagteware stelsel is 'n kontinue proses deur elke fase van die sagteware lewenssiklus. Programtoetsing is die deel van die geldigheidsbepalingsproses wat gewoonlik gedurende die implementeringsfase uitgevoer word - asook nadat die implementeringsfase voltooi is.

Programevaluering behels die toetsing van die sagteware deur die volgende data aan die program te verskaf om te verseker dat dit korrek funksioneer:

- *Tipiese data:* Hier word bepaal of die tipiese data waarmee die program werk korrek geïnterpreteer word en of die bewerkings daarop korrek is.
- *Grenswaardes:* Grenswaardes word aan die program verskaf om te verseker dat dit tot op die gespesifiseerde limiete korrek funksioneer.
- *Buite grenswaardes:* Hierdie data word aan die program verskaf om te verseker dat data verwerp word indien dit buite die spesifikasie is.

Dit word somtyds gestel dat programtoetsing en -ontfouting nie dieselfde proses is nie. Daar is wel baie ooreenkomste daartussen, maar tog is dit twee duidelik verskillende prosedures. Toetsing is die prosedure waar die moontlikheid van programfoute ondersoek word, terwyl ontfouting die prosedure is om die spesifieke voorkoms van die fout aan te dui en dit te korrigeer (Sommerville, 1982:152).

Dit is baie belangrik om te besef dat toetsing nooit kan uitwys of 'n program korrek is nie. Die moontlikheid bestaan altyd dat foute nog steeds kan voorkom, selfs na die mees intensiewe toetse. Programtoetsing kan slegs die teenwoordigheid van foute aandui en *nie* die afwesigheid daarvan nie.

Programtoetsing is 'n destruktiewe prosedure wat dikwels die program opsetlik op 'n sekere manier wil laat funksioneer waarvoor dit nie ontwerp is nie (buite grenswaardes). Daar moet voldoende tyd toegelaat word vir programtoetsing. Hierdie tyd moet ongeveer gelykstaande wees aan programmeringstyd (Van Tassel, 1978:271).

Deur dus van uitgebreide programtoetsing en -ontfouting gebruik te maak word die beste eindresultaat verkry en 'n betroubare stelsel verseker.

Die verskillende fases van programtoetsing word deur Sommerville (1982:156-157) as volg opgesom:

□ *Funksionele toetsing:* Dit is die basiese vlak van toetsing waar die funksies wat 'n module opmaak getoets word om te verseker dat dit korrek funksioneer.

□ *Module toetsing:* 'n Module bestaan uit 'n aantal funksies wat moontlik met mekaar kan saamwerk. Nadat elke funksie getoets is word die gesamentlike werking van die funksies getoets wanneer dit saamgevoeg is as 'n module. Dit moet moontlik wees om 'n module te toets as 'n alleenstaande entiteit, sonder die teenwoordigheid van ander stelselmodules.

□ *Substelsel toetsing*: 'n Substelsel word saamgestel uit verskeie modules wat saamgevoeg word. Net soos modules saamwerk en kommunikeer, so moet die toetsing van substelsels gerig wees op die toetsing van module koppelvlakke.

□S*telseltoetsing:* Stelseltoetsing (integrasie toetsing) word uitgevoer wanneer verskeie substelsels geïntegreer word onder 'n sekere beheerder om die totale stelsel te vorm. Op hierdie stadium word daar nie net gekonsentreer om foute in die ontwerp en kodering uit te wys nie, maar ook op die vraag of die stelsel voldoen aan die behoefte wat gestel is, en of die dinamiese karakteristieke van die stelsel voldoen aan dié benodig.

□ *Aanvaarding toetsing:* Tot op hierdie stadium word alle toetse gedoen deur van tipiese toetsdata gebruik te maak. Aanvaarding toetsing is die prosedure waar daar van werklike data - waarvoor die stelsel ontwerp is - gebruik gemaak word om die stelsel te toets. Hierdie prosedure bring dikwels nog foute in die aanvanklike spesifikasie van die stelsel aan die lig.

4.1.3 Dokumentering

Onder dokumentering word die volgende bespreek:

□ Die belangrikheid van sagtewaredokumentasie,

□ Tipes dokumentasie.

4.1.3.1 Die belangrikheid van sagtewaredokumentasie

Deur die unieke kompleksiteit van rekenaar gegronde stelsels te erken, word die behoefte aan goeie en akkurate dokumentasie duidelik uitgelig. Vier hoof redes bestaan kragtens Katzin (1985:2) - en word vervolgens afsonderlik bespreek:

□ *Produktiwiteit:* Die kompleksiteit van rekenaarstelsels laat gewoonlik nie toe dat een persoon alles van 'n stelsel sal weet nie. Deur van goeie en akkurate dokumentasie gebruik te maak word vertragings uitgeskakel en kan produktiwiteit gehandhaaf word.

□ *Verwysing:* Dokumenstasie dien as 'n primêre verwysingsleutel in 'n stelsel-georiënteerde organisasie. Alle konsepte van die organisasie word hierdeur weergegee en dit word as die bloudruk (blueprint) van die administrasie beskou.

□ *Kontinuïteit:* Dokumentasie moet voordurend op datum gebring word om te verseker dat die stelsel kontinu kan funksioneer in geval van moontilke stelselfalings.

□ *Ooreenstemming:* As gevolg van goeie dokumentasie word ooreenstemming in data, operasies, instruksies en resultate onderhou. Ooreenstemming beeld akkuraatheid uit wat weer op sy beurt tyd verminder in die geval van foutopsporing en -korrigering.

Bogenoemde punte het wel nie direk betrekking op hierdie projek nie, maar moet in ag geneem word in enige stelsel voor en gedurende die ontwikkeling daarvan.

4.1.3.2 Tipes dokumentasie

Die volgende tipes dokumentasie kom voor en word kortliks bespreek:

- Stelsel en programdokumentasie,
- Operasioneledokumentasie,
- Gebruikersdokumentasie.

4.1.3.2.1 Stelsel en programdokumentasie

Hierdie vorm van dokumentasie word grootliks deur die stelsel en programmeringsgroep gebruik en dit beskryf in detail hoe elke program en stelsel funksioneer. Dit moet nie eers 'n aanvang neem nadat die program/stelsel voltooi is nie, maar moet 'n kontinue proses wees wat saam met die probleemdefinisie begin (Van Tassel, 1978:102).

Hier word die doelwit en metode van elke programmodule beskryf. Dit word benodig wanneer veranderinge aan die program of rekenaarhardeware aangebring word of waar daar omskakelings benodig word.

4.1.3.2.2 Operasioneledokumentasie

Operasioneledokumentasie behels instruksies vir rekenaaroperateurs om hulle in kennis te stel watter tipe skyf of bandlêers gemonteer moet word op watter skyf of bandaandrywer, watter tipe papier om vir die drukkers te gebruik, watter insette vanaf die kontrolepaneel benodig word, ens.

Instruksies vir data-invoer en beheerseksies word ook somtyds by hierdie dokumentasie ingesluit (Katzin, 1985:3). Hierdie tipe dokumentasie (weens die aard daarvan) word nie benodig vir hierdie projek nie.

4.1.3.2.3 Gebruikersdokumentasie

Die gebruikershandleiding (user's manual) word geskryf vir die persone wat op die funksionele vlak werk waarvoor die rekenaarstelsel ontwerp is. Dit is minder tegnies van aard as bv. die stelsel- en operasionelehandleidings. Dit beskryf egter steeds in detail en konsep al die funksies van die stelsel vanaf die gebruiker se standpunt.

Gebruikersdokumentasie sluit gewoonlik 'n maklike verwysing in om individuele afdelings en instruksies te beskryf en verduidelik. Dit beskryf ook die uitsette van die stelsel in detail en verskaf 'n indeks vir maklike kruisverwysings.

Daar word nie 'n gebruikershandleiding vir die sagtewarepakket van hierdie navorsingsprojek saamgestel nie, aangesien dit nie kommersieël bemark gaan word nie en weens die gebruikersvriendelikhied van die pakket self.

4.1.4 Vlakke van programmering

Daar bestaan 3 vlakke van programmeringstale, nl. hoëvlak, middelvlak en laevlak programmering. 'n Hoëvlaktaal gee voorafbepaalde funksies om

programmering te vergemaklik. 'n Laevlaktaal is die laagste vlak van programmering en staan as saamsteltaal (assembly language) bekend. 'n Middelvlaktaal kombineer die elemente van 'n hoëvlaktaal met die funksionaliteit van saamsteltaal. Tabel 4.1 lys die vlakke van programmering met voorbeelde van die tale soos deur Schildt (1990:6).

Tabel 4.1 - Tabel van programmeringsvlakke

Hoëvlak	**Middelvlak**	**Laevlak**
Ada	C	Assembler
Modula-2	FORTH	
Pascal	Macro-assembler	
COBOL		
FORTRAN		
BASIC		

Hierdie tale kan verder in gestruktureerde en ongestruktureerde programmeringstale opgedeel word. 'n Blokgestruktureerde taal laat toe dat prosedures en funksies binne-in ander prosedures en funksies geskep kan word. Ongestruktureerde tale laat geen vorm van strukturaliteit toe nie.

Die belangrikste onderskeidingskenmerk van 'n gestruktureerde taal is *kompartementialisering* (compartmentalization) van die data en kode. Dit is die tegniek waar alle informasie en instruksies wat benodig word om sekere funksies uit te voer, van die res van die program afgebaken en verskuil word (Schildt, 1990:7).

Tabel 4.2 dui tipiese gestruktureerde en ongestruktureerde tale aan.

Gestruktureerde tale neig daartoe om nuwer te wees as ongestruktureerde tale en word daarom baie meer algemeen gebruik.

Tabel 4.2 - Gestruktureerde en ongestruktureerde programmeringstale

Gestruktureerde tale	Ongestruktureerde tale
Pascal	FORTRAN
Ada	BASIC
C	COBOL
Modula-2	

Alle programmeringstale is egter nie ewe verkieslik vir programmeerders nie. Nie-programmeerderstale is bv. COBOL en BASIC. COBOL was ontwikkel om die nie-programmeerder toe te laat om die program te kan lees en te verstaan. BASIC was essensieël ontwikkel om die nie-programmeerder in staat te stel om redelike eenvoudige probleme te kan oplos.

Hedendaags word veral Pascal en C as programmeringstale gebruik. Beide hierdie twee tale is vir die doel van die navorsingsprojek gebruik. Pascal is gebruik om die hoofrekenaar se sagteware te ontwikkel en C is gebruik om die mikrorekenaar se sagteware te ontwikkel.

4.2 Hoofrekenaar sagteware

Die doel van die hoofrekenaar se sagteware is om 'n koppelvlak te skep tussen die gebruiker en die hoofrekenaar se datakommunikasiemoontlikhede sodat hy toegang kan hê tot die funksies wat die mikrorekenaarnode bied. Al die hoofrekenaar sagteware is geskryf deur gebruik te maak van Turbo Pascal 6.0.

Hierdie sagteware bied kortliks die volgende fasiliteite aan die gebruiker:

□ Dit verskaf die opsie om die nodes se adresse in die hoofrekenaardatabasis te kan verander en vertoon.

□ Dit verskaf die opsie om die 24 8255PPI penkonfigurasie-inligting van die node - wat vir 5 moontlike aantye (uitset- en insetpenne) en 5 moontlike aftye van die uitsetpenne voorsiening maak - te redigeer, te vertoon, uit te wis en om dit aan die node te verskaf (programmeer node).

□ Dit bied die opsie om die huidige staat van die node se 8255PPI penne te onttrek.

□ Hierdie resultate kan vertoon, sorteer en gedruk word vir verwysingsdoeleindes. Hierdie inligting word slegs 'n sekere tydperk gestoor (soos deur die gebruiker gespesifiseer) en word daarna outomaties deur die sagteware uitgewis.

□ Die algemene opstellingsparameters soos bv. seriale poort, resultaatstoortydperk, ens. kan ook gespesifiseer en vertoon word.

Hierdie sagteware bestaan uit die volgende programme en modules:

□ Die hoofprogram, waaronder:

- □ *MAINPROG.PAS:* Hoofprogram wat die menu beheer en ander subprogramme roep.

□ Twee subprogramme, waaronder:

- □ *RES_UTIL.PAS:* Subprogram wat alle resultaat verwante funksies uitvoer.
- □ *NODEUTIL.PAS:* Subprogram wat alle 8255PPI pen verwante funksies uitvoer.

□ Vyf verwante modules, waaronder:

- □ *COMNDECL.PAS:* Module wat alle gemeenskaplike funksies en deklarasies bevat.
- □ *COM_UNIT.PAS:* Module wat datakommunikasiefunksies en prosedures bevat.
- □ *EDITLN.PAS:* Module wat 'n volle lynredigeerder (line editor) bied.
- □ *GETKEY.PAS:* Module wat die sleutel wat op die sleutelbord gedruk is aanvaar. Dit bevat ook prosedures om vir 'n muis voorsiening te maak.
- □ *SCRNUTIL.PAS:* Module wat alle skerm verwante funksies en prosedures bevat.

Die twee subprogramme word geroep deur die hoofprogram wat sekere parameters daarheen stuur sodat dit kan bepaal watter funksie uitgevoer moet word. Die vyf modules word by die programme ingesluit tydens die vertalingsproses (compilation). Hierdie modules word slegs aan die program/module verbind indien enige funksie daarvan in die program of ander module benodig word.

Die sagteware maak voorsiening vir die gebruik van 'n muis indien wenslik. Dit ondersteun menu seleksie met die muis en sleutelbord en is dus baie gebruikersvriendelik t.o.v. beskikbare funksies.

Die funksies, prosedures en deklarasies van die hoofprogram, die twee subprogramme en die vyf modules word vervolgens bespreek.

4.2.1 Hoofprogram - MAINPROG

Die bronkode-uitdruk van die hoofprogram kan in Bylaag B.1 gesien word. Hierdie program bestaan uit die volgende funksies en prosedures:

- *PollTime:* Hierdie funksie bereken en verskaf die huidige polstyd deur van die opstellingsdatabasis data gebruik te maak.
- *PollNodes:* Hierdie prosedure moet die nodes pols vir inligting indien die stelseltyd gelyk is aan die huidige polstyd. Al die nodes moet gepols word vir die resultate wat dit bevat. Hierdie prosedure word aan INT28H se onderbrekingsprosedure gekoppel om die nodige funksie uit te voer. Die polsing van die nodes kan egter nie direk vanaf hierdie prosedure uitgevoer word nie, aangesien daar teenstrydigheid in die onderbrekingsprosedures van die DOS stelsel voorkom. Dus is 'n soortgelyke prosedure wat nie van DOS se INT21H se funksies gebruik maak nie in plek van die polsprosedure geplaas om die wenslikheid van polsing op 'n sekere vasgestelde stadium aan te dui. Vir 'n

gedetaileerde bespreking verwys na Hoofstuk 5, punt 5.2 - Evaluering van hoofrekenaar sagteware.

□ *LegalPollTime:* Hierdie funksie bepaal of die huidige stelseltyd binne 1 minuut van die polsing roeping is wanneer 'n funksie deur die gebruiker uitgevoer word. Indien dit die geval is word die gebruiker versoek om later weer te probeer.

□ *ExtractResults:* Hierdie prosedure word gebruik om die resultate van 'n node te onttrek deur sekere parameters na die NODEUTIL program te stuur.

□ *ProgramNode:* Hierdie prosedure word gebruik om die node met die 8255PPI penkonfigurasiedata te programmeer. Daar word later in meer detail hierna verwys.

□ *SelectNode:* Hierdie prosedure word gebruik om een van 16 nodes vanaf 'n menu te kies waarna funksies daarmee uitgevoer kan word.

□ *DbaseChoose:* Hierdie prosedure word gebruik om 'n menu item te kies wanneer daar na die databasisfunksies verwys word.

□ *DoDbaseMenu:* Hierdie prosedure voer die databasismenu uit.

□ *UtilChoose:* Hierdie prosedure verskaf 'n menu waaruit keuses gemaak kan word vir nutsprogramme/funksies indien verlang.

□ *ViewGeneralParams:* Hierdie prosedure vertoon die algemene opstellingsparameters.

□ *EditGeneralParams:* Hierdie prosedure laat die gebruiker toe om sekere opstellingsparameters te verander, soos bv. die polstydinterval, ens.

□ *GenSetupChoose:* Hierdie prosedure verskaf 'n menu om tussen redigering en besigtiging van die algemene opstellingsparameters te kies.

□ *DoGenSetupMenu:* Hierdie prosedure vertoon die algemene opstellingsmenu.

□ *ViewNodeAddr:* Hierdie prosedure vertoon die nodes se adresse soos opgestel.

□ *EditNodeAddr:* Hierdie prosedure laat die gebruiker die nodes se adresse redigeer.

□ *NodeAddrChoose:* Hierdie prosedure laat die gebruiker kies tussen die redigering of vertoning van die nodes se adresse.

- *DoNodeAddrMenu:* Hierdie prosedure vertoon die node-adres menu.
- *NodeIdChoose:* Hierdie prosedure verskaf 'n keuse tussen die vertoning, redigering, drukstuk maak en uitwissing van die node se 8255PPI penkonfigurasiedata.
- *DoNodeIdMenu:* Hierdie prosedure vertoon die node-8255-pen menu.
- *SetupChoose:* Hierdie prosedure verskaf 'n keuse tussen die algemene opstellingsmenu, node-adres menu en node-8255-pen menu.
- *DoUtilMenu:* Hierdie prosedure vertoon die nutsprogrammenu.
- *DoSetupMenu:* Hierdie prosedure vertoon die opstellingsmenu.
- *MainChoose:* Hierdie prosedure verskaf 'n keuse tussen die nutsprogrammenu en die opstellingsmenu.
- *Dos_Safe_To_Use:* Hierdie prosedure is die onderbrekingsprosedure wat die polstyd en die stelseltyd op die skerm opdateer. Dit bepaal ook wanneer die nodes gepols moet word.
- *Int28ExitProc:* Hierdie prosedure is die uitgangsprosedure vir die onderbrekingsprosedure wat aan INT28H gekoppel is.
- *Set_Poll_Inthandler:* Hierdie prosedure inisieer die onderbrekingsprosedure wat aan INT28H gekoppel moet word.
- *DataExpires:* Hierdie funksie bepaal of die resultaatdata in die nodes se databasisse verstryk het volgens die algemene opstellingsveld *Hold* (Bylaag B.4: Lyn 50) en word elke keer met die programaanvang geroep.
- *CheckDbaseExpDates:* Hierdie prosedure wis die resultaatdata uit indien die houtyd daarvan verstryk het.
- *DoMainMenu:* Hierdie prosedure vertoon die hoofmenu.
- *Init:* Hierdie prosedure is die eerste prosedure in die hoofprogram wat uitgevoer word. Dit inisieer wysers (pointers), algemene veranderlikes, ens.

Die eerstevlak struktuurkaart van die hoofprogram (soos dit in Engels op die skerm vertoon word) kan in Figuur 4.2 gesien word.

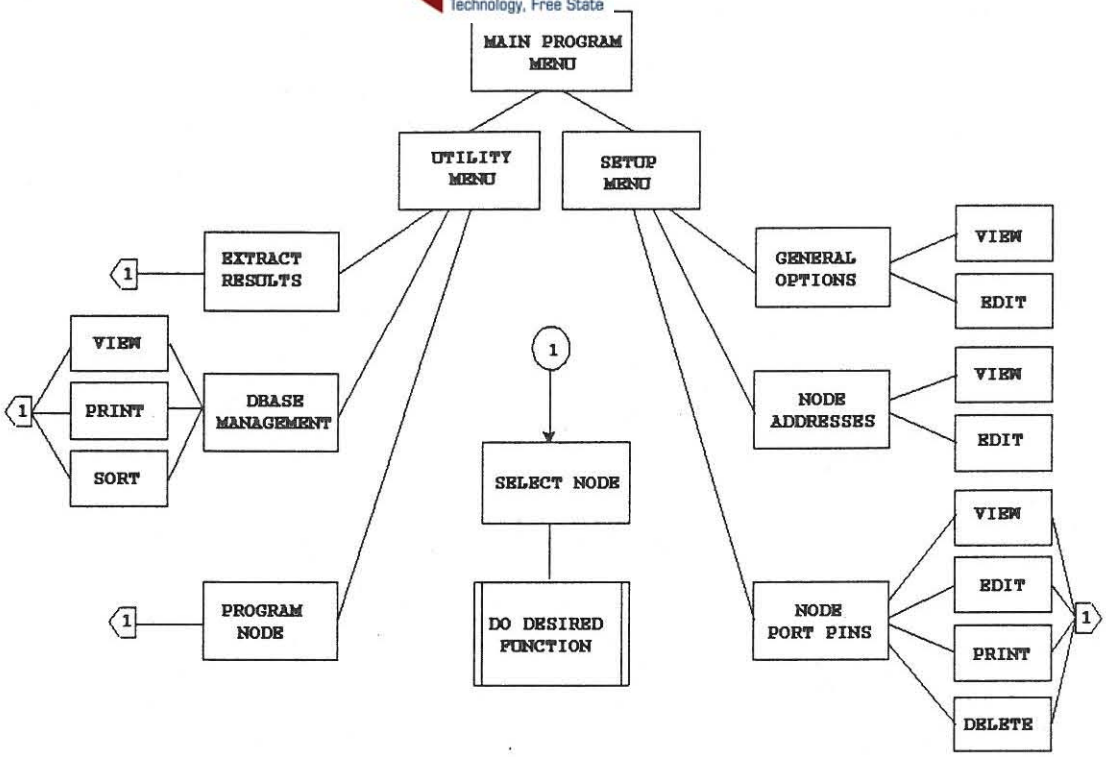


Fig. 4.2 - Eerstevlak struktuurkaart van hoofprogram

Hieruit kan gesien word hoe die menus vanaf die twee hoof keuses na onder versprei word. Nadat die keuse gemaak is word die funksie dienooreenkomstig uitgevoer.

Vanuit Figuur 4.2 kan gesien word dat die hoofmenu eerste op die skerm verskyn. Vanuit hierdie menu kan daar tussen 2 onderafdelings gekies word, nl.

- Nutsprogramme/funksies (utilities),
- Opstellingsprogramme/funksies (setup).

Vervolgens word 'n meer gedetaileerde beskrywing van elke funksie aan die hand van 'n struktuurkaart of vloeikaart gedoen.

4.2.1.1 Nutsprogramme/funksies

Vanuit die nutsprogram menu word die volgende keuses gemaak:

- Onttrek resultate,
- Databasis beheerfunksies,
- Programmeer node.

4.2.1.1.1 Onttrek resultate

Figuur 4.3 dui die basiese vloeikaart vir hierdie prosedure aan.

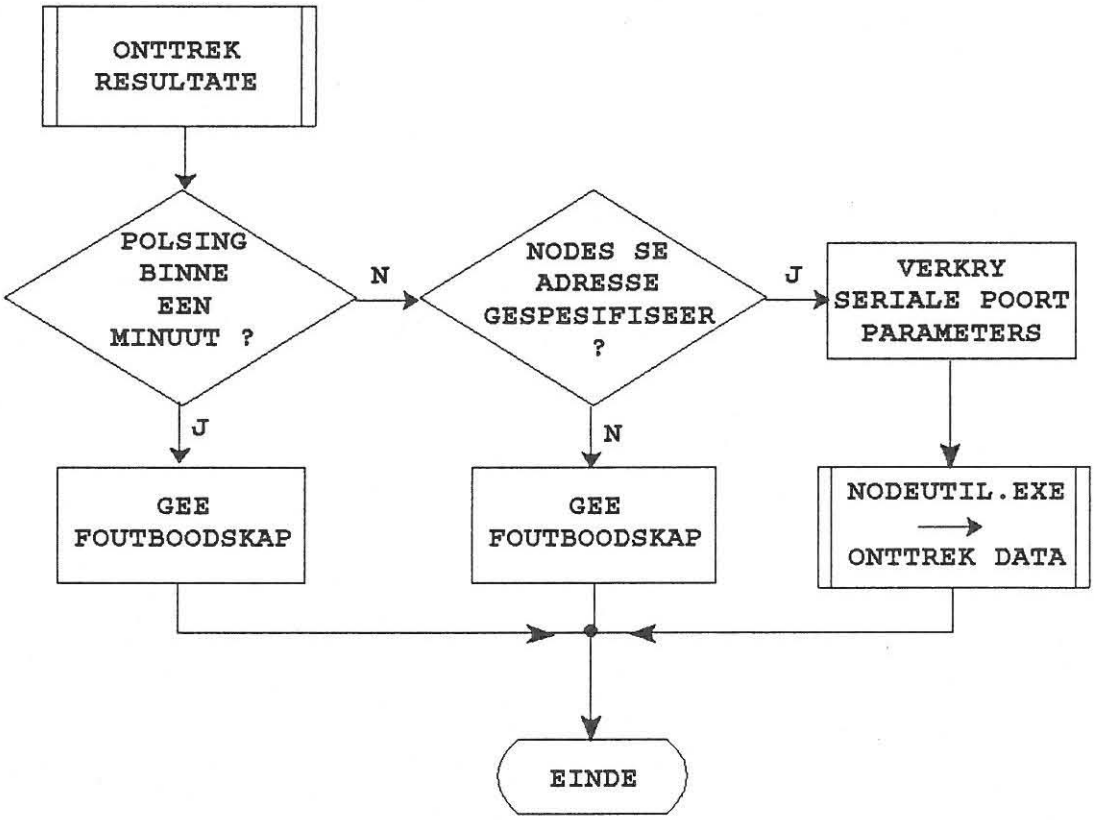


Fig. 4.3 - Vloeikaart van Onttrek Resultate prosedure

Vanuit Figuur 4.3 kan gesien word dat daar eers bepaal word of die polstyd onderbrekingsprosedure nie binne een minuut geaktiveer gaan word nie. Indien dit nie die geval is nie word verseker dat die nodes se adresse alreeds gespesifiseer is. Indien hieraan voldoen is word die seriale poort parameters van die algemene opstellingsdatabasis verkry.

Hierdie parameters word via die opdraglyn (command line) aan die subprogram NODEUTIL.EXE verskaf wat op sy beurt verdere prosessering uitvoer om die ontrekking van data moontlik te maak (Bylaag B.2: Lyn 677). Die polsonderbrekingsprosedure maak nie van hierdie tegniek gebruik om data van die nodes te onttrek nie, maar eerder van 'n ander wat later bespreek word.

4.2.1.1.2 Databasis beheerfunksies

Hierdie menu bied die volgende keuse t.o.v. die beheer van die resultaatdata van die nodes:

- Vertoon op skerm,
- Uitdruk na drukker,
- Sorteer in stygende volgorde.

4.2.1.1.2.1 Vertoon op skerm

Met hierdie opsie kan die resultate in die databasis van die geselekteerde node op die skerm vertoon word. Die hoofprogram stuur sekere parameters (Bylaag B.1: Lyn 324) na die subprogram RES_UTIL.EXE sodat die data op die skerm vertoon kan word.

Die prosedure wat hierdie funksie verrig heet *ViewResults* (Bylaag B.3: Lyn 134).

4.2.1.1.2.2 Uitdruk na drukker

Hierdie opsie funksioneer soortgelyk aan 4.2.1.1.2.1. Die enigste verskil is dat die parameters wat na RES_UTIL.EXE gestuur word hierdie subprogram beveel om die resultate na die drukker te lys en nie na die skerm nie.

Die prosedure wat hierdie funksie verrig heet *PrintResults* (Bylaag B.3: Lyn 221).

4.2.1.1.2.3 Sorteer in stygende volgorde

Hierdie funksie sorteer die resultate in stygende volgorde van pennommer indien dit verlang word. Die hoofprogram maak weereens van die subprogram RES_UTIL.EXE gebruik om hierdie funksie uit te voer.

Die prosedure wat hierdie funksie verrig heet *VerifySort* (Bylaag B.3: Lyn 120).

4.2.1.1.3 Programmeer node

Hierdie opsie word gebruik om die 8255PPI penkonfigurasiedata na die node oor te dra waar dit in die geheue gestoor word sodat bewerkings daarmee uitgevoer kan word. Die prosedure wat hierdie funksie verrig heet *ProgramNode* (Bylaag B.1: Lyn 183).

Die node word eerstens geadresseer sodat dit kan bevestig dat dit die korrekte adres vanaf die meester ontvang het. Indien die reaksie positief is vanaf die node gaan die meester voort om die data van die penkonfigurasies na 'n databuffer oor te dra en daarna na die node te stuur. Die databuffer/woord samestelling kan in Figuur 4.4 gesien word.

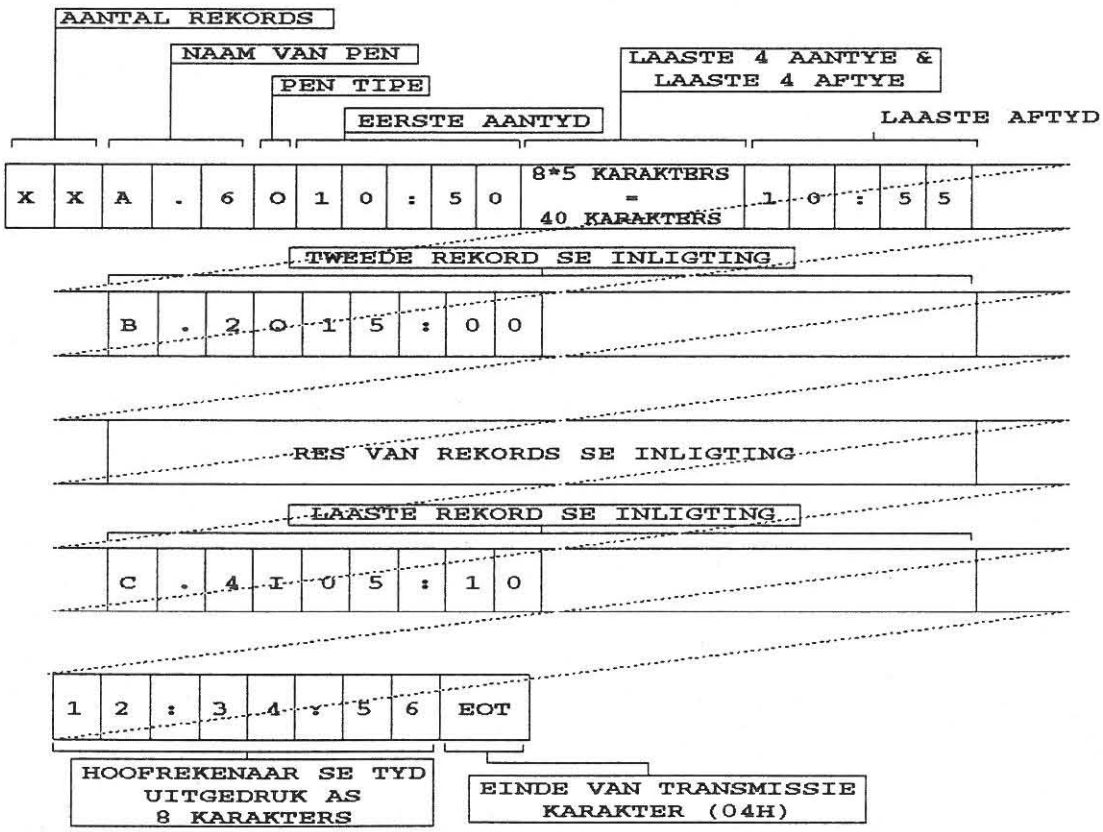


Fig. 4.4 - Databuffer/woord samestelling om node te programmeer

Uit Figuur 4.4 kan die volgende waargeneem word:

- Die eerste 2 karakters van die datawoord dui die aantal rekords waarvan die data na die node gestuur word aan, bv. 4 rekords = 04. Dit word verskaf sodat die node presies weet hoeveel penrekords se data verwerk moet word.
- Die volgende 3 karakters dui die naam van die pen aan, bv poort A pen nr. 1 = A.0, ens.
- Die volgende karakter dui die pentipe aan, d.w.s. of dit 'n uitsetpen ('O') of insetpen ('I') is.
- Die volgende 50 karakters dui die 5 aantye en 5 aftye van die pen aan. Hierdie tye word as 5 karakter stringe gestuur, bv. '10:50'.
- Elke pen se inligting word vervolgens in dieselfde formaat as bg. gestuur totdat die laaste pen se inligting gestuur is.
- Die huidige stelseltyd van die hoofrekenaar word nou in die string as 'n 8 karakter string gevoeg sodat die node se tyd met die hoofrekenaar se tyd gesinkroniseer kan word.
- Die laaste karakter in die datawoord/buffer is die einde-van-transmissie karakter wat as EOT gedefinieer is (Bylaag B.5: Lyn 27-30).

Nadat die datawoord/buffer aan die node veskaf is verwerk die node dit en stuur toepaslike boodskappe terug om die korrekte ontvangs van data te verifieer. Die node gaan daarna voort om opdatering te doen.

4.2.1.2 Opstellingsprogramme/funksies

Vanuit die opstelling menu kan die volgende keuses gemaak word:

- Algemene keuses/funksies,
- Node-adres funksies

- Node-8255PPI-pen funksies

4.2.1.2.1 Algemene keuses/funksies

Hierdie opsies bied die fasiliteit om die algemene opstellingsparameters op die skerm te vertoon en ook om dit te redigeer. Parameters wat vertoon word is:

- Die datakommunikasiepoort wat gebruik word op die hoofrekenaar,
- Die baud tempo - vasgestel op 1200 baud vir hierdie projek,
- Die tipe pariteit - vasgestel as GEEN vir hierdie projek,
- Die aantal stopbisse - vasgestel op 1 vir hierdie projek,
- Die aantal databisse - vasgestel op 8 vir hierdie projek,
- Die polstyd interval - 15 min, 30 min, 60 min of 120 min.
- Die resultaathoutyd - 24 uur, 36 uur, 48 uur of 72 uur.

Bogenoemde parameters kan geredigeer word om aan die behoefte van die gebruiker te voldoen. Daar moet egter op gelet word dat dit verkieslik is om na die volgende veld in die redigeer skerm te gaan voordat die F2 sleutel gedruk word om die nuwe waardes te stoor. Dit word vereis sodat die gebruiker die versekering kan hê dat die laaste parameter wat geredigeer is wel dienooreenkomstig verander word.

Daar moet ook op gelet word dat die vasgestelde parameters nie geredigeer kan word nie aangesien dit onnodig in hierdie projek is.

Die prosedure wat die vertoonfunksie verrig heet *ViewGeneralParams* (Bylaag B.1: Lyn 459) en dié wat die redigeerfunksie uitvoer heet *EditGeneralParams* (Bylaag B.1: Lyn 515).

4.2.1.2.2 Node-adres funksies

Hierdie opsie bied die moontlikheid om die nodes se adresse te kan verander indien 'n behoefte daarvoor bestaan. Aangesien daar vir 16 nodes in hierdie projek voorsiening gemaak word, moet die adresse so gespesifiseer word dat geen 2 nodes dieselfde adresse het nie.

Hierdie adresse word deur die redigeringsfunksie verander en kan vanaf 00H tot 1FH strek. Die rede hiervoor is dat slegs die 5 mins beduidende bisse in die node as die adres beskou word. Hierdie 5 bisse maak voorsiening vir 32 moontlike adreskonfigurasies om meer buigsaamheid aan die stelsel te verleen.

Hierdie adresse kan d.m.v. die vertoonfunksie op die skerm besigtig word om te verseker dat dit reg opgestel is. Die vertoonfunksie word uitgevoer deur *ViewNodeAddr* (Bylaag B.1: Lyn 633) en die redigeerfunksie deur *EditNodeAddr* (Bylaag B.1: Lyn 659).

4.2.1.2.3 Node-8255PPI-pen funksies

Hierdie opsie veskaf 'n menu wat die volgende fasiliteite bied:

- Vertoon penkonfigurasies,
- Redigeer 8255PPI penkonfigurasie-inligting,
- Druk penkonfigurasie-inligting op drukker,
- Wis penkonfigurasierekords uit.

4.2.1.2.3.1 Vertoon penkonfigurasies

Hierdie fasiliteit vertoon die bestaande 8255PPI penkonfigurasie-inligting op die skerm. Die volgende inligting word vertoon:

- 8255PPI pen, bv. 'Poort A.5',
- 'n Beskrywing van die gebruik van die pen - maks 30 karakters,
- Node geheueposisie - opsioneel. Daar is aanvanklik voorsiening gemaak dat elke 8255PPI penrekord 'n eie begin geheueposisie in die node sal hê, maar dit is later as onnodig beskou aangesien dit outomaties in die databasisstruktuur van die node geïnkorporeer word.
- Pentipe - hetsy dit 'n uitset- of insetpen is,
- Die 5 aantye en aftye van die pen soos benodig word.

Die prosedure wat hierdie funksie verrig heet *ViewPinConfig* (Bylaag B.2: Lyn 334).

4.2.1.2.3.2 Redigeer 8255PPI penkonfigurasie-inligting

Hierdie opsie bied die fasiliteit om die 8255PPI peninligting te redigeer. Dit skep outomaties enige nuwe rekords en redigeer bestaande rekords.

Die beskrywing van die gebruik van die pen is daar om die gebruiker die fasiliteit te bied om presies te weet waarvoor elke pen gedefinieer is sodat daar altyd beheer oor pentoewysings in die toepassing uitgeoefen kan word.

Soos reeds genoem is die geheueposisie in die node opsioneel en word dit nie na die node gestuur vir verwerking nie. Die pentipe spreek vanself - hetsy uitset of inset.

In die geval van 'n uitsetpen bepaal die 5 aantye die tye wanneer die betrokke pen se uitset na 'n logika '1' geskakel word. In die geval van 'n insetpen bepaal hierdie tye elk 'n een minuut periode waartydens daar voortdurend vanaf die insetpen gelees word om die status daarvan te bepaal.

Die 5 aftye bepaal die tye wanneer die uitsetpen se status na 'n logika '0' verander word. Hierdie tye het geen effek vir 'n insetpen nie aangesien daar slegs van die 5 aantye gebruik gemaak word in die opdateringsprosedure van die node.

Indien 'n aantyd of aftyd geïgnoreer moet word, word die string '**:**' ingevoer. Dit is ook die aanvanklike waarde van die veld tydens inisieëring. Daar moet egter op die formaat van die tyd gelet word aangesien die verantwoordelikheid by die gebruiker berus om dit korrek te hou (hh:mm).

Die prosedure wat hierdie funksie verrig heet *ConfigurePins* (Bylaag B.2: Lyn 226).

4.2.1.2.3.3 Druk penkonfigurasie-inligting op drukker

Hierdie opsie bied die fasiliteit om die 8255PPI penkonfigurasie-inligting na die drukker te lys. Die prosedure wat hierdie funksie verrig heet *PrintConfig* (Bylaag B.2: Lyn 434).

4.2.1.2.3.4 Wis penkonfigurasierekords uit

Hierdie opsie bied die fasiliteit om enkel rekords van die 8255PPI penkonfigurasies uit te wis. Die rekord se inligting word in 'n opsommende wyse vertoon en die gebruiker kies slegs die penrekord wat uitgewis moet word.

Die prosedure wat hierdie funksie verrig heet *ChooseRec* (Bylaag B.2: Lyn 501).

4.2.2 Subprogram1 - NODEUTIL

Die bronkode-uitdruk van hierdie subprogram kan in Bylaag B.2 gesien word. Die doel van hierdie program is om prosedures uit te voer wat direk en indirek betrekking het op die werking van die 8031 node. Dit sluit die 8255PPI penkonfigurasie funksies en node resultaatonttrekking in.

Hierdie opdragte word uitgevoer deurdat die hoofprogram sekere parameters daarheen stuur om die verlangde funksie uit te wys. Daar is reeds na al die funksies verwys behalwe die gedetaileerde beskrywing van resultaatontrekking

vanaf die node. Hierdie posedure word in die COMNDECL module (Bylaag B.4) aangetref en word bespreek na die funksie en prosedure opsommingslys.

Hierdie program (NODEUTIL) bestaan uit die volgende funksies en prosedures:

□ *Exchange:* Hierdie prosedure word deur die sorteer prosedure gebruik om twee rekords in die geskakelde lys (linked list) om te ruil.

□ *SortToMemory:* Hierdie prosedure sorteer die node 8255PPI penkonfigurasie-inligting in stygende volgorde van pennaam.

□ *VerifySort:* Hierdie prosedure bepaal of daar meer as een rekord in die databasis voorkom voordat dit enige data kan sorteer.

□ *PortHilight:* Hierdie prosedure word gebruik om die 8255PPI penstringe in die selekteermenu uit te lig (highlight).

□ *PortChoose:* Hierdie prosedure word gebruik om 'n 8255PPI pen uit die selekteermenu te kies.

□ *GetDbaseData:* Hierdie prosedure word gebruik om te bepaal of die rekord wat geredigeer word reeds bestaan al dan nie. Dit verskaf die databasisinligting aan die redigeerder indien dit alreeds bestaan, anders verskaf dit verstek (default) inligting aan die redigeerder.

□ *ConfigurePins:* Hierdie prosedure word gebruik om die 8255PPI penkonfigurasie-inligting te redigeer en te stoor.

□ *ViewPinConfig:* Hierdie prosedure word gebruik om die 8255PPI penkonfigurasie-inligting op die skerm te vertoon.

□ *PrintConfigHeader:* Hierdie prosedure druk 'n opskrif op elke bladsy van die drukstuk van die 8255PPI penrekords.

□ *PrintConfig:* Hierdie prosedure lys die 8255PPI penkonfigurasierekords na die drukker.

□ *DeleteRecord:* Hierdie prosedure wis die 8255PPI penrekord uit.

□ *ChooseRec:* Hierdie prosedure laat die gebruiker die 8255PPI penrekord kies wat uitgewis moet word.

□ *DeletePinConfig:* Hierdie prosedure bepaal of daar enige 8255PPI penrekords bestaan om uit te wis al dan nie.

□ *DoNodeMenu:* Hierdie prosedure inisieer die node-selekteer menu en analiseer die parameters vanaf die hoofprogram om te bepaal watter funksie verlang word om uit te voer.

Die onttreking van data word deur die volgende prosedure bewerkstellig, nl. *XtractInfo* (Bylaag B.4: Lyn 317). Die basiese werking kan as volg opgesom word:

□ Die node word geadresseer deur die hoofprogram.

□ Die onttrek-resultate id karakter word na die node gestuur nadat dit bevestiging van teenwoordigheid aan die meester gestuur het.

□ Die node bevestig dat dit gereed is om die data te stuur waarna die meester die databuffer inisieer om die inkomende data te *vang*.

□ Die node stuur die data nadat dit verwerk is en bevestig die einde van transmissie indien die databuffer na die meester gestuur is.

□ Die meester bepaal die aantal rekords gestuur en verwerk die res van die data.

□ Hierdie data word in die resultaatdatabasis gestoor.

4.2.3 Subprogram2 - RES_UTIL

Die bronkode-uitdruk van hierdie subprogram kan in Bylaag B.3 gesien word. Die hooffunksie van hierdie subprogram is om opdragte van die hoofprogram m.b.t. bestaande noderesulate uit te voer.

Daar is reeds na die funksies en prosedures van hierdie program verwys en vervolgens word 'n opsomming van die prosedures in hierdie program verskaf:

- *Exchange:* Hierdie prosedure word deur die sorteer prosedure gebruik om 2 rekords in die geskakelde lys (linked list) om te ruil.
- *SortToMemory:* Hierdie prosedure sorteer die node se resultaatinligting in stygende volgorde van pennaam indien dit verlang word.
- *VerifySort:* Hierdie prosedure bepaal of daar meer as een rekord in die databasis voorkom voordat dit enige data kan sorteer.
- *ViewResults:* Hierdie prosedure vertoon die bestaande resultate op die skerm van die rekenaar.
- *PrintHeader:* Hierdie prosedure druk 'n opskrif op elke bladsy van die drukstuk van die resultate van die node.
- *PrintResults:* Hierdie prosedure lys die geselekteerde node se bestaande resultaatrekords na die drukker.
- *ExecResultFunctions:* Hierdie prosedure analiseer die parameters vanaf die hoofprogram om te bepaal watter funksie uitgevoer moet word.

4.2.4 Module COMNDECL

Die bronkode-uitdruk van hierdie module kan in Bylaag B.4 gesien word. In hierdie module word die algemene verklarings van veranderlikes, funksies en prosedures gemaak. Die menu opskrifstringe word ook hier gedefinieer om maklike toegang daartoe te verkry deur al die modules en programme.

Verder word die databasisstrukture vir die opstelrekord (setup record), 8255PPI penkonfigurasierekord van die nodes en die resultaatrekord hier verklaar. Hierdie rekords word gebruik om die verwante data in databasisse te stoor sodat die programme toegang daartoe kan hê om verwerkings daarop uit te voer. Die beskrywings van die verskillende velde in die strukture kan uit die verduidelikende opmerkings gesien word.

'n Kort opsomming van die verskillende funksies en prosedures in hierdie module word vervolgens gegee:

□ *DosSafe:* Hierdie funksie bepaal of DOS besig is om ander funksies uit te voer indien dit onderbreek moet word om onderbrekingsroetines (interrupt routines) uit te voer.

□ *PrnPresent:* Hierdie funksie bepaal of die drukker aan die rekenaar verbind is en aangeskakel is wanneer drukstukke gemaak moet word.

□ *SetupFileExist:* Hierdie funksie bepaal of die algemene opstellingslêer alreeds bestaan al dan nie.

□ *NodeAddrFileExist:* Hierdie funksie bepaal of die nodes se adresse reeds gedefinieer is al dan nie.

□ *ComPortExist:* Hierdie funksie bepaal of die seriale poort wat gespesifiseer is wel teenwoordig is op die rekenaar al dan nie.

□ *NodeChoose:* Hierdie funksie bepaal of daar 'n node uit die node-selekteer menu gekies is of nie.

□ *NodeDbaseExist:* Hierdie funksie bepaal of daar 8255PPI penkonfigurasies vir die node bestaan al dan nie.

□ *ResFileExist:* Hierdie funksie bepaal of daar al resultate gelees is vanaf die spesifieke node.

□ *ConvStr2:* Hierdie funksie gee 'n 2 karakter string vanaf 'n heelgetalwaarde wat as parameter daaraan verskaf word.

□ *SystemDate:* Hierdie funksie gee die huidige stelseldatum van die hoofrekenaar as 'n stringwaarde.

□ *SystemTime:* Hierdie funksie gee die huidige stelseltyd van die hoofrekenaar as 'n stringwaarde.

□ *HexStrToByte:* Hierdie funksie skakel die heksadesimale string waardes van die nodes se adresse om na 'n desimale waarde tussen 0 en 255.

□ *XtractInfo:* Hierdie prosedure word gebruik om resultate vanaf nodes aan te vra en in die databuffer in te lees. Hierna word hierdie data verwerk en in die

resultaatdatabasis van die spesifieke node gestoor. Dit word in meer detail bespreek wanneer die programfunksie bespreek word.

4.2.5 Module COM_UNIT

Hierdie module is seker die belangrikste module van die hoofrekenaar se sagteware aangesien dit al die prosedures bevat om seriale datakommunikasie met die nodes moontlik te maak. Die bronkode-uitdruk kan in Bylaag B.5 gesien word.

Hier word die basisadres van elke moontlike seriale datakommunikasiepoort gedefinieer asook die sagteware onderbreekvektor (interrupt vector) wat daarmee saamgaan (Bylaag B.5: Lyn 17 - 25). Die afwyking (offset) van die verskillende seriale datakommunikasiepoortregisters word ook hier weergegee soos deur Dettman (1989:181) en die funksie van elkeen word vervolgens kortliks bespreek.

Die volgende registers word aangetref (Bylaag B.5: Lyn 10 - 16):

□ *THR:* "Transmit holding resgister (*THR) & Receive data register (*RDR)". Die *THR hou die datagreep wat oor die lyn gestuur moet word. Data word na hierdie register geskryf indien bis 5 van LSR aandui dat hierdie register leeg is. Die *RDR hou die mees onlangse datagreep wat vanaf die datakommunikasielyn ontvang is. Hierdie register word gelees indien bis 0 van LSR aandui dat 'n datagreep ontvang is.

□ *IER:* "Interrupt enable register". Die IER beheer die tipe onderbrekings wat deur die USART (Universal Synchronous Asynchronous Receiver & Transmitter) gegenereer word. Verskeie onderbrekings kan toegelaat word, afhangend van hoe die onderbrekingsprosedure (interrupt procedure) geskryf is. Daar moet egter op

gelet word dat daar 'n spesifieke aksie geneem moet word om 'n onderbreekaanduiding te herstel indien dit toegelaat word. Tabel 4.3 gee 'n opsomming van die bistoewysings van hierdie register om spesifieke onderbrekings toe te laat om voor te kom en ook die aksie wat geneem moet word om die bis te herstel.

□ *IIR:* "Interrupt identification register". Indien 'n onderbreking voorkom kan die datakommunikasieprogram dit identifiseer en die nodige aksie neem. Tabel 4.4 gee 'n opsomming van die betekenis van die bisse in die IIR.

□ *LCR:* "Line control register". Hierdie register is die primêre beheerregister van die seriale kommunikasielyn. Dit bepaal die datawoordlengte (5, 6, 7 of 8 bisse), die aantal stopbisse (1, 1.5 of 2 bisse) en die paritiet (ignoreer, onewe, ewe, merk of spasie).

□ *MCR:* "Modem control register". Hierdie register stel beheerlyne na die modem en stel die modem dienooreenkomstig in kennis of die rekenaar gereed is om data te stuur, te ontvang of beide.

□ *LSR:* "Line status register". Hierdie register gee die status weer van die datakommunikasielyn en algemene lynfoute kan hierdeur opgespoor word.

□ *MSR:* "Modem status register". Hierdie register dui die status van die modem aan, bv. of CTS gestel is al dan nie.

Tabel 4.3 - IER bis toewysings

Bis	Aktiveer	Aksie
0	Data ontvang	Lees *RDR
1	*THR leeg	Skryf na THR
2	Datafout of onderbreking	Lees LSR
3	MSR verandering	Lees MSR
4-7	Nie gebruik nie - altyd 0'e	

Tabel 4.4 - IIR bis toewysings

Bis	Betekenis
0	Meer as een onderbreking het voorgekom
1-2	Onderbreking identifikasie:
	00 - Verandering in MSR
	01 - *THR leeg
	10 - Data ontvang
	11 - Data ontvang fout of onderbreking

Tabel 4.3 en 4.4 word verskaf om 'n beter begrip te vestig oor die gebruik van onderbrekingsprosedures wanneer datakommunikasie plaasvind.

In hierdie module word *IntHandler* as die onderbrekingsprosedure gebruik om te reageer as data ontvang word (Bylaag B.5: Lyn 56). Hierdie prosedure word as 'n agtergrond (background) prosedure beskou omdat dit slegs in werking tree indien data op die seriale kommunikasielyn ontvang word. Dit lees data in 'n databuffer indien dit verlang word en toets ook sekere vlaggies wat gestel of herstel moet word gedurende datakommunikasie.

Vervolgens word die ander prosedures kortliks bespreek.

□ *InitPort:* Hierdie prosedure word gebruik om die gespesifiseerde seriale poort te inisieer. Die baud tempo word hier gespesifiseer (1200 baud vir hierdie projek), asook die aantal stop bisse, tipe pariteit en datawoordlengte (1200,1,N,8). Die onderbrekingsprosedure (IntHandler) word ook hier geïnstalleer om as agtergrondprosedure op te tree.

□ *SetDtrHigh:* Hierdie prosedure stel of herstel die DTR (dataterminaal gereed) lyn sodat data gestuur en ontvang mag word.

□ *TxString:* Hierdie prosedure stuur 'n datastring na die datakommunikasielyn met 'n maksimum lengte van 255 karakters.

□ *RestoreValues:* Hierdie prosedure herstel die oorspronklike datakommunikasieparameters wat tydens die InitPort prosedure verander is.

4.2.6 Module EDITLN

Die bronkode-uitdruk van hierdie module kan in Bylaag B.6 gesien word. Die funksie van hierdie module is om 'n lynredigeringsfasiliteit (line editor) daar te stel.

Hierdie prosedure heet *EditLine* en die volgende parameters kan daaraan voorsien word:

□ *S :* Die string wat ingevoer moet word,

□ *X :* Die X-koördinaat op die skerm,

□ *Y :* Die Y-koördinaat op die skerm,

□ *Len :* Die maksimum lengte van die string in karakters uitgedruk,

□ *LegalChars :* Die toelaatbare karakters wat ingevoer mag word,

□ *Term :* Die karakters/sleutels wat die einde van redigering aandui.

4.2.7 Module GETKEY

Die bronkode-uitdruk van hierdie module kan in Bylaag B.7 gesien word. Die funksie van hierdie module is om die sleutel/muis aksie vanaf die stelsel te bepaal en as 'n karakter aan die aktiewe funksie/prosedure terug te stuur.

Bylaag B.7: Lyn 10 - 29 definieer sekere sleutels en muis aksies soos deur dié module geïndentifiseer word. Die twee funksies *Keypressed* en *Readkey* maak gebruik van INT16H (interrupt 16H) in Turbo Pascal om die betrokke funksie uit

te voer. Aangesien INT28H gebruik word om onderbrekings te bewerkstellig vir die polsingsroetine (polling), moes hierdie twee funksies herskryf word om van INT21H gebruik te maak sodat INT28H kan voorkom indien daar vir sleutelbordinset gewag word.

Bogenoemde verskynsel is vasgestel nadat 'n kort Turbo C program geskryf is om INT28H te genereer. Hierdie program het korrek gefunksioneer, inteenstelling met die gewone Turbo Pascal program wat van INT16H gebruik maak vir sleutelbordfunksies. Daar is van Turbo Debugger gebruik gemaak om die verskille in die programme uit te wys en daarna is die nodige verandering in die Turbo Pascal program aangebring.

4.2.8 Module SCRNUTIL

Die bronkode-uitdruk van hierdie module kan in Bylaag B.8 gesien word. Hierdie module bevat die prosedures om skermfunksies uit te voer. Vervolgens word 'n kort opsomming van die prosedures gegee:

- *ColorSettings:* Hierdie prosedure bepaal of die skerm wat die rekenaar gebruik 'n kleur of monochroom skerm is en inisieer sekere standaard kleure vir bv. skrif, menu items, ens.
- *SaveScreen:* Hierdie prosedure kan 'n maksimum van 10 skerms se inligting stoor in die rekenaar se geheue sodat hierdie skerms na uitvoering van 'n funksie weer direk uit die video geheue teruggeplaas kan word op die skerm.
- *RestoreScreen:* Hierdie prosedure plaas die vorige gestoorde skerm inligting terug op die skerm deur direkte video geheuetoegang.
- *HideCursor:* Hierdie prosedure haal die merker (cursor) vanaf die skerm sodat dit net wys indien dit benodig word.

□ *RestCursor:* Hierdie prosedure plaas die merker (cursor) terug op die skerm in gewone teksmodus.

□ *SetEnterResident:* Hierdie prosedure plaas 'n reëleindekarakter direk in die sleutelbordbuffer sodat die menu onmiddelik op die skerm verskyn sonder om hierdie sleutel te druk. Daar moet egter op gelet word dat hierdie funksies slegs op AT masjiene met verbeterde sleutelborde (advanced keyboards) werk.

□ *SetTxtColor:* Hierdie prosedure word gebruik om die teks op die skerm 'n sekere voorgrond en agtergrond kleur te gee.

□ *WriteAt:* Hierdie prosedure word gebruik om teks by 'n sekere posisie op die skerm te skryf met 'n spesifieke kleur.

□ *WriteCenter:* Hierdie prosedure skryf teks in die middel van die skerm op die ry wat deur parameter Y aangewys word.

□ *ClearWindow:* Hierdie prosedure maak die gedeelte van die skerm skoon wat deur koördinate X1, Y1, X2,Y2 aangewys word.

□ *DrawBox:* Hierdie prosedure teken 'n omlynde reghoekige skerm op die huidige aktiewe skerm. Hier word ook van koördinate gebruik gemaak om die grootte en posisie van die skerm aan te dui.

□ *ErrorMsg:* Hierdie prosedure vertoon 'n sekere foutboodskap op die skerm en wag vir die gebruiker om 'n sleutel op die sleutelbord te druk.

□ *DispMessage:* Hierdie prosedure vertoon 'n teksboodskap op die skerm.

Die funksies en prosedures van bogenoemde modules word in die hoofprogram en twee subprogramme gebruik om die stelsel as 'n eenheid te laat funsioneer.

Die voordeel hiervan is dat die hoofprogram baie minder kompleks kan wees en dit bespaar ook die gebruik van rekenaargeheue aangesien slegs sekere dele van die programme op enige stadium geheue gebruik en nie alles op een slag nie.

4.3 Mikrorekenaarnode sagteware

Die bronkode-uitdruk van die 8031 node sagteware kan in Bylaag C gesien word. Hierdie sagteware dien as die BIOS (Basic Input/Output System) van die mikrorekenaar en beheer alle funksies van die mikrorekenaar.

Die volgende funksies en prosedures som die programstruktuur op:

- *Timer0:* Hierdie prosedure is die onderbrekingsprosedure wat die intydse horlosie (real-time clock) beheer. Die onderbrekingsvektor word outomaties na hierdie prosedure verwys met die aanvang van die program.
- *Serial:* Hierdie prosedure is die onderbrekingsprosedure wat die seriale datakommunikasie van ontvangde data beheer. Die onderbrekingsvektor word outomaties na hierdie prosedure verwys met die aanvang van die program.
- *Int_to_str:* Hierdie funksie verskaf die ASCII waarde van 'n heelgetal in 'n stringformaat.
- *Reset_misc_variables:* Hierdie prosedure inisieer die algemene veranderlikes en datastrukture van die program met die aanvang van die program.
- *Get_unique_addr:* Hierdie prosedure bepaal die unieke adres van die node en stoor dit in die eksterne geheue vir verwysingsdoeleindes.
- *Init_serial_interrupt:* Hierdie prosedure inisieer die seriale poort vir 1200 baud, geen pariteit, 1 stopbis en 8 databisse.
- *Init_timer0_interrupt:* Hierdie prosedure inisieer die intydse horlosie veranderlikes.
- *Init_8255ppi:* Hierdie prosedure inisieer die 8255PPI om 2 eenvoudige uitsetpoorte en 1 insetpoort te verskaf vir evalueringsdoeleindes.
- *Enable_interrupts:* Hierdie prosedure stel die seriale en intydse onderbrekings in staat om voor te kom.
- *Tx_char:* Hierdie prosedure stuur enkel karakters oor die seriale verbinding na die meester.

□ *Tx_string:* Hierdie prosedure stuur stringe karakters oor die seriale verbinding na die meester.

□ *Tx_lf_cr:* Hierdie prosedure stuur 'n nuwelyn karakter (new line) en 'n reëleindekarakter (carriage return) oor die seriale verbinding na die meester.

□ *Current_time:* Hierdie funksie verskaf die huidige stelseltyd van die node as 'n ASCII string van 8 karakters.

□ *Update_8255_status:* Hierdie prosedure dateer die status van die 8255PPI poorte op volgens die aan- en aftye van die pen en dateer ook die geheueveranderlikes wat vir resultaatterugvoering gebruik moet word op.

□ *Accept_programming_data:* Hierdie prosedure aanvaar die 8255PPI penkonfigurasie-inligting vanaf die meester en verwerk dit. Hierdie inligting word in 'n datastruktuur, in die eksterne geheue van die node, gestoor vir opdatering en resultaatterugvoering. Die stelseltyd van die meester word ook hier verwerk en die node se intydse horlosie word daarvolgens gesinkroniseer.

□ *Send_results_to_master:* Hierdie prosedure stel die datawoord/buffer saam om die resultaatinligting van die 8255PPI penne na die meester te stuur. Hierdie prosedure word later in detail bespreek.

□ *Stay_in_endless_loop:* Hierdie prosedure inisieer die opdateringsprosedure en bepaal watter funksies verrig moet word indien die meester dit adresseer. Hierdie prosedure vorm 'n eindlose lus en is dus die hoofprosedure van die node. Dit word later in meer detail beskryf.

Die basiese werking van die 8031 node kan as 'n vloeikaart in Figuur 4.5 gesien word.

Uit Figuur 4.5 blyk dat die vloeidiagram in 'n eindlose lus bly na afloop van die inisieëringsprosedures. Hierdie lusprosedure heet *Stay_in_endless_loop* soos reeds genoem.

Die vloeikaart van hierdie prosedure kan in Figuur 4.6 waargeneem word.

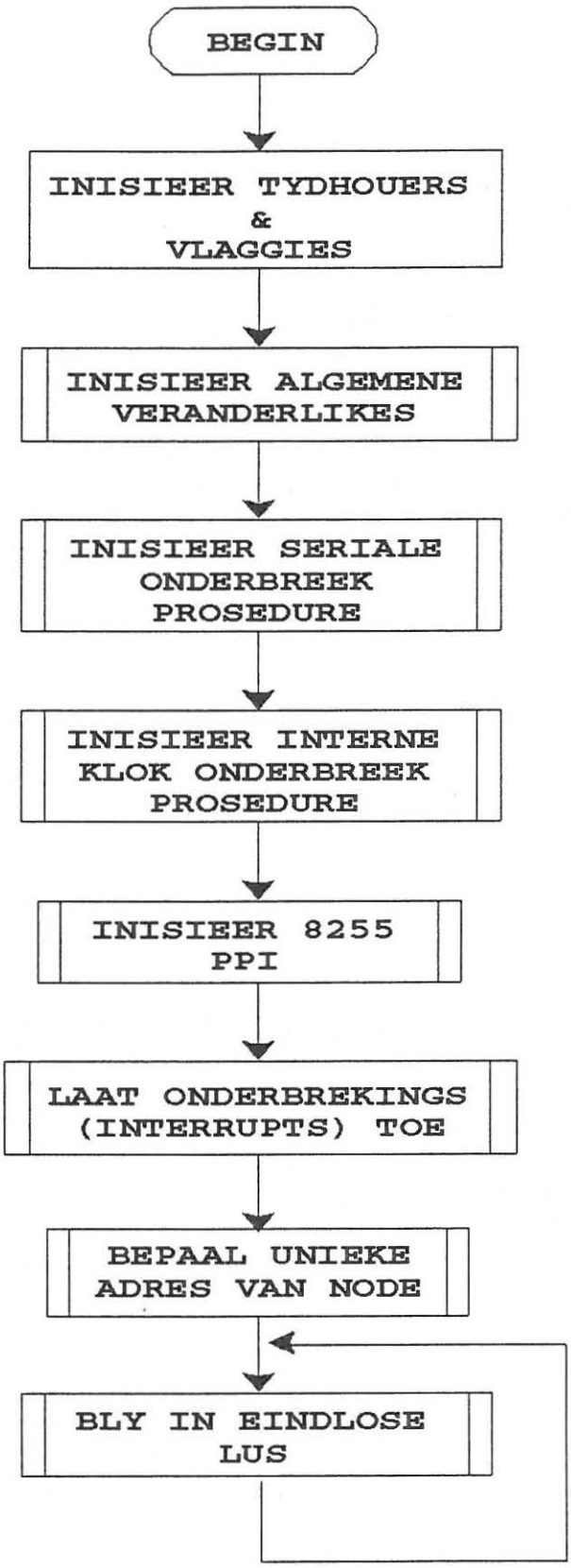


Fig. 4.5 - Vloeidiagram van 8031 node sagteware

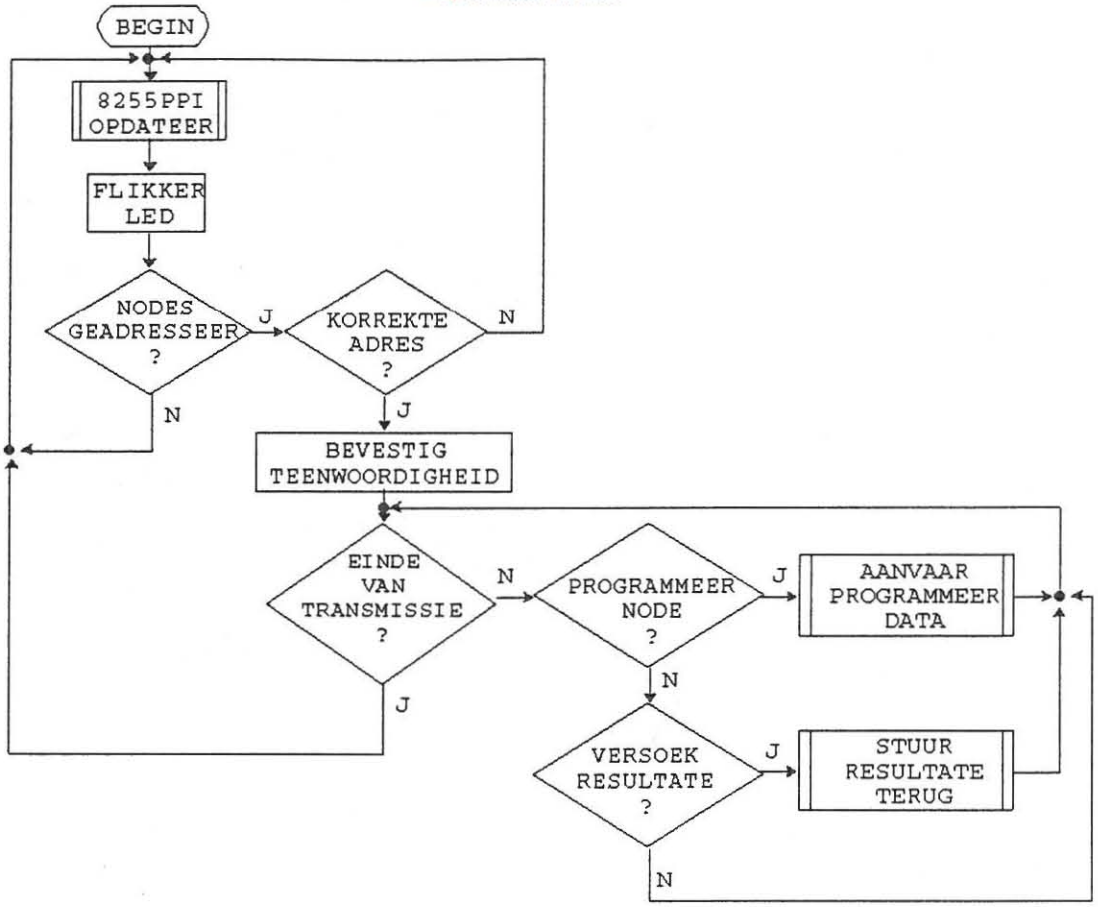


Fig. 4.6 - Vloeikaart van eindlose lus roetine

Vanuit Figuur 4.6 kan gesien word dat die opdateringsprosedure heel eerste uitgevoer word. Hierdie prosedure kyk die heeltyd of die meester die node adresseer al dan nie. Indien dit wel die geval is, bevestig die node sy teenwoordigheid. Die prosedure gaan dan in 'n lus van besluitnemings om te bepaal watter funksies deur die hoofrekenaar aangevra word.

Die node bly in hierdie staat totdat die meester die einde van transmissie aandui.

Indien die mikrorekenaar die versoek ontvang dat die hoofrekenaar dit wil programmeer voer dit die *Accept_programming_data* prosedure (Bylaag C: Lyn 364) uit.

Indien die hoofrekenaar die huidige status van die 8255PPI penne wil moniteer, word die versoek aan die node gerig en die prosedure *Send_results_to_master* (Bylaag C: Lyn 438) word uitgevoer. Hierdie prosedure stel die datawoord/buffer saam wat na die meester rekenaar gestuur word. Figuur 4.7 dui die datawoord/buffer samestelling van die resultaatdata aan.

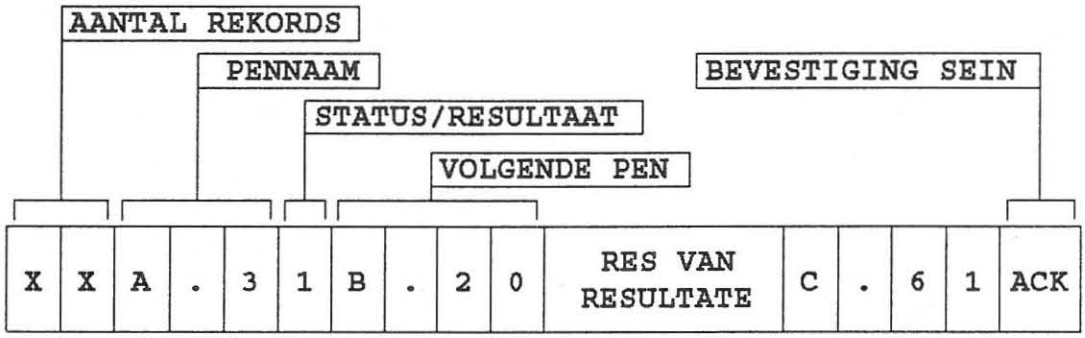


Fig. 4.7 - Datawoord/buffer samestelling van resultate deur die 8031 node

Vanuit Figuur 4.7 kan die volgende gesien word:

□ Die eerste 2 karakters dui die aantal rekords in die mikrorekenaar se geheue aan. dit word deur die hoofrekenaar benodig om te weet hoeveel 8255PPI penrekords se resultate verwerk moet word.

□ Die volgende 3 karakters dui die pennaam aan, bv. poort A pen 1 = A.0, ens.

□ Die volgende karakter dui die status/resultaat van die 8255PPI pen aan. 'n '0' verteenwoordig 'n *af* status en 'n '1' 'n *aan* status.

□ Die res van die rekords se inligting word verder in dieselfde formaat gestuur.

□ Die bevestigingskarakter aan die einde van die datawoord/buffer dui die einde van resultaattransmissie aan wat die hoofrekenaar in staat stel om die inligting te verwerk en in die resultaatdatabasis te stoor.

4.4 Opsomming

Die inhoud van hierdie hoofstuk wys duidelik die belangrikheid van sagteware in 'n rekenaarstelsel uit. Die kombinasie van goeie beplanning, dokumentasie, bronkode-ontwikkeling en evaluering vorm die basis van 'n goeie rekenaarstelsel.

Daar is in hierdie navorsingsprojek so ver moontlik gepoog om die sagteware so gebruikersvriendelik en eenvoudig moontlik te hou. Die interaksie van die 5 afsonderlike modules, die 2 subprogramme en die hoofprogram dui op 'n goed gedefinieerde stelsel wat maklik evalueerbaar is.

Die datakommunikasie wat tussen die hoofrekenaar en mikrorekenaarnode bewerkstellig is, vorm die grondslag van 'n goeie datakommunikasiestelsel. Die vermoë van beide die hoofrekenaar en mikrorekenaar sagteware om onderbrekingsprosedures te kan hanteer maak hierdie stelsel 'n baie betroubare en effektiewe stelsel om te implementeer in enige relatief laespoed prosesbeheertoepassing.

HOOFSTUK 5

EVALUERING VAN DATAKOMMUNIKASIESTELSEL

Die evaluering van die datakommunikasiestelsel het die volgende behels:

- Evaluering van mikrorekenaarnode,
- Evaluering van hoofrekenaarsagteware,
- Evaluering van totale datakommunikasiestelsel.

Bogenoemde punte word vervolgens afsonderlik bespreek.

5.1 Evaluering van mikrorekenaarnode

Tydens die ontwikkeling van die node sagteware is vrylik van simulasies daarvan gebruik gemaak om die korrekte werking daarvan te verseker.

Die korrekte werking van die sagteware is vervolgens d.m.v. 'n *Romulator* bevestig voordat dit in die betrokke LAG gestoor is om deur die stelsel gebruik te word.

Om die werking van die mikrorekenaar te monitor is 'n toetsmodule ontwikkel wat 'n visuele aanduiding verskaf van elke uitsetpoort en ook 'n fasiliteit bied om insette aan die node te verskaf. Die skematiese voorstelling van hierdie kring kan gesien word in Bylaag A.7.

Vanuit Bylaag A.7 blyk dit dat die kring basies uit twee dele bestaan, nl:

- Twee 8-bis uitsetpoorte met LED indikasie,
- Een 8-bis insetpoort vanaf 'n DIP-skakelaarkonfigurasie.

Die twee 8-bis uitsetpoorte word vanaf die 8255PPI kring in die I/U module gedryf en is aan Poort A en Poort B onderskeidelik verbind. Die binêre waarde wat op Poort A voorkom word as 'n waarde-indikasie op LED1 tot LED8 aangedui en dié van Poort B op LED9 tot LED16. Die doel hiervan is dat daar sekere waardes na die I/U module geskryf kan word en die geldigheid daarvan visueel bevestig kan word.

Die 8-bis insetpoort word voorsien deur dit aan Poort C van die 8255PPI op die I/U module te verbind. Die DIP-skakelaarkonfigurasie verskaf bis manipulasie deurdat elke skakelaar 'n bis verteenwoordig. Die doel hiervan is om te toets of die data wat vanaf die parallele poort op die I/U module aan die mikrorekenaarnode verskaf word geldig is.

5.2 Evaluering van hoofrekenaarsagteware

Die basiese beginsel om data vanaf die nodes te onttrek lewer twee moonlikhede nl.:

- Polsing d.m.v. handbeheer,
- Polsing d.m.v. 'n onderbrekingsprosedure.

In hierdie projek is gepoog om beide hierdie tegnieke te implementeer. Polsing d.m.v. handbeheer is foutvry geïmplementeer en data kan sonder probleme vanaf die nodes onttrek word.

Daar het egter 'n probleem ontstaan met die polsingstegniek wat van die onderbrekingsprosedure gebruik maak - ten spyte daarvan dat beide tegnieke van dieselfde prosedure (Bylaag B.4: Lyn 317) gebruik maak om data vanaf die nodes te onttrek. Die onderbrekingsprosedure *Dos_Safe_To_Use* (Bylaag B.1: Lyn 968) word aan onderbreking INT28H gekoppel en die stelseltyd deurentyd gemoniteer om te bepaal of die nodes vir resultate gepols moet word al dan nie.

Indien die nodes wel gepols moet word, word die huidige programfunksie onderbreek en die polsing moet vanaf hierdie prosedure geïnisieer word. Daar word egter van databasisfunksies in hierdie onderbrekingsprosedure gebruik gemaak om te bepaal of daar 8255PPI penkonfigurasiedata vir die betrokke node bestaan. Hierdie funksies maak van INT21H (algemene DOS onderbreking) gebruik om data vanaf leêrs te lees en daarheen te skryf.

Dit bring mee dat daar 'n onderbrekingsteenstrydigheid in die DOS stelsel voorkom en die prosedure laat die stelsel *hang*. Nadat daar van verskeie tegnieke gebruik gemaak is om hierdie probleem te probeer oplos het dit egter verdere teenstrydigheid in die onderbrekingsprosedure meegebring wat bv. die lêertoewysingstabel (FAT - file allocation table) onbruikbaar gemaak het.

Volgens Dettman (1989:580) kan daar nie van INT21H funksies in 'n onderbrekingsprosedure gebruik gemaak word as die onderbrekingsprosedure reeds in die geheue gelaai is nie (memory resident). Die gevolgtrekking is dus gemaak dat hierdie prosedure nie ten volle geïmplementeer kan word nie.

Daar is egter van 'n fopprosedure (dummy procedure) in plek hiervan gebruik gemaak om die wenslikheid van polsing op 'n sekere vasgestelde stadium aan te dui. Hierdie prosedure onderbreek die aktiewe programfunksie en gee 'n visuele aanduiding dat die nodes in daardie stadium gepols behoort te word.

Die res van die sagteware is noukeurig getoets en daar is oral van foutanalisering, met toepaslike foutboodskappe, gebruik gemaak om foute uit te wys. Daar is ook van Turbo Debugger gebruik gemaak om die laevlak kodering van sekere prosedures na te gaan om korrekte programwerking te verseker.

Turbo Pascal bied 'n geïntegreerde ontfouter (debugger) wat tydens die ontwikkelingsproses gebruik is om die korrekte werking van die stelsel as geheel te verseker.

5.3 Evaluering van totale datakommunikasiestelsel

Daar is van gedraaide paar (twisted pair) kabel gebruik gemaak om die multitap koppeling tussen die hoofrekenaar en die node te bewerkstellig. Hierdie stelsel is getoets om te verseker dat die data wat vanaf die hoofrekenaar aan die node verskaf word betroubaar en foutvry ontvang word.

Resultate is voortdurend vanaf die node na die hoofrekenaar teruggelees om die korrekte oordrag van data vanaf die node na die hoofrekenaar te verseker. Bevredigende resultate is behaal.

Die stelsel is ook tot 'n mindere mate aan elektriese ruis blootgestel en het korrek gefunksioneer. Die gevolgtrekking is gemaak dat hierdie stelsel oor goeie ruisimmuniteit by lae elektriese ruis uitstraling (agtergrondruis) beskik en dus baie betroubaar sal funksioneer in enige normale datakommunikasieomgewing.

Daar is ook van 'n 8 km koppellyn tussen die hoofrekenaar en die mikrorekenaarnode gebruik gemaak om hierdie stelsel te toets. Hierdie lang verbinding het terugvoering in die stelsel meegebring en datafoute het voorgekom.

Evaluering kon egter, as gevolg van 'n gebrek aan toerusting, nie oor 1200m - die teoretiese limiet volgens RS485 spesifikasie - plaasvind nie. Aangesien daar van die RS485 standaarde/komponente in die ontwerp gebruik gemaak is, is die oorwoë mening egter dat dit bevredigend tot op hierdie afstand sal funksioneer.

Die datatransmissiespoed van hierdie stelsel is vasgestel op 1200 baud, maar dit is egter met sukses teen 2400 baud en 4800 baud getoets.

5.4 Gevolgtrekkings

Die volgende gevolgtrekkings is na afhandeling van die projek gemaak:

- Die RS485 seriale standaard is uiters geskik vir multitap datakommunikasiestelsels tussen 'n sentrale beheereenheid en afgeleë nodes as gevolg van die volgende redes:
  - Die algemene beskikbaarheid van geskikte lyndrywer/ontvanger komponente en die lae koste daarvan,
  - Die werklike multitap vermoë waaroor RS485 beskik - dit gee 'n hoë-impedansie uitset,
  - Die differensiële spanningsmodus wat dit as datakommunikasiebeginsel gebruik.
- 'n Persoonlike rekenaar is 'n baie geskikte eenheid om as 'n hoofrekenaar te gebruik as gevolg van:
  - Die nie-toegewyde toepassing daarvan in so 'n stelsel - dit hoef slegs gebruik te word vir beheer van die nodes indien dit verlang word deur die gebruiker en kan andersins vir enige ander nie-toegewyde toepassing gebruik word,
  - Die hoëvlak van programmeerbaarheid waaroor dit beskik - komplekse programme kan ontwikkel word om in sulke toepassings gebruik te word,
  - Die beskikbaarheid van ondersteunende hardware en sagteware.
- Die INTEL 8031 mikrobeheerder is 'n uiters geskikte beheerelement in mikrorekenaarnodes as gevolg van:
  - Die seriale dataoordragvermoë waaroor dit beskik,
  - Die veskeidenheid inset/uitset poorte wat dit bied,
  - Die relatief lae komponenttelling van so 'n stelsel,

- Die beskikbaarheid van hoëvlak ontwikkelingsagteware vir hierdie familie van mikrobeheerders.

- Ruim geheue uitbreiding van die LAG en WSLG van die 8031 mikrobeheerderkonfigurasie word benodig as gevolg van:
  - Groot stoorkapasiteit wat benodig word om programme van hierdie aard te ontwikkel (LAG),
  - Groot stoorkapasiteit wat benodig word om die databasisstrukture en verskeie veranderlikes van die program te stoor (WSLG).
- Die ontwikkelde datakommunikasiestelsel kan met weinig aanpassing in enige prosesbeheertoepassing waar soortgelyke kommunikasie benodig word geïmplementeer word.

HOOFSTUK 6

SAMEVATTING

Die hoofdoel van hierdie projek was om 'n lae-koste datakommunikasiestelsel te ontwikkel wat oor relatief groot afstande kan funksioneer sonder die gebruik van modems.

Daar is weggebreek van die algemeen gebruikte RS232 tegniek en alternatiewe seriale standaarde is ondersoek om die bes moontlike datakommunikasiestelsel daar te stel. 'n Mikrorekenaar is ontwikkel wat van die RS485 datakommunikasiestandaard gebruik maak om as die eindpunt node op te tree en sodoende datakommunikasie tussen die meester (hoofrekenaar) en die node (mikrorekenaar) moontlik te maak.

'n Teoretiese oorsig van algemene datakommunikasiestelsels is in Hoofstuk 2 gelewer. Hier is na verskillende tipes datakommunikasie verwys, waaronder parallelle, seriale en netwerk kommunikasie. Seriale kommunikasie het die mees geskikte vir hierdie toepassing blyk te wees.

Seriale datakommunikasietegnieke, datatransmissiespoed en datakommunikasielynstrukture is oorweeg met spesiale verwysing na RS232, RS423, RS422 en RS485. Daar is ook ondersoek ingestel na die mees algemeen gebruikte foutopsporingstegnieke en na sikliese oortolligheidskontrole (CRC) verwys.

Om die datakommunikasietegniek te implementeer is daar hardeware ontwikkel om aan die vereistes van die RS485 standaard te voldoen. Hierdie hardeware word in Hoofstuk 3 bespreek. 'n Meerdoelige kragbron is ontwikkel wat later in die RS232-na-RS485 omsetter en die mikrorekenaarnode geïmplementeer is.

'n RS232-na-RS485 omsetter is ontwikkel om die RS232 dataseine van die hoofrekenaar na RS485 standaard dataseine om te skakel. 'n Doelspesifieke geïntegreerde kring - die SN75176 - wat as sender en ontvanger funksioneer, is gebruik om hierdie omskakeling te doen.

Soos genoem is 'n mikrorekenaarnode ontwikkel om as eindpunt van die datakommunikasiestelsel te funksioneer. Hierdie mikrorekenaar bestaan uit 'n mikrobeheerderkring (8031) en ondersteunende komponente.

'n Inset/Uitset module is ook ontwikkel om aan die 8031 node verbind te word. Hierdie module beskik oor die volgende fasiliteite:

- Dit skakel die RS232 standaard dataseine om na RS485 standaard,
- Dit bevat 'n 8255PPI geïntegreede kring wat parallelle uitsette en insette aan die mikrorekenaar bied. Hierdie kring is so geprogrammeer dat dit 2 uitsetpoorte bied en een insetpoort.
- Dit verskaf ook optiese isolering sodat die mikrorekenaar as 'n elektries geïsoleerde eenheid aan die hoofrekenaar verbind kon word.

Sagteware is ontwikkel om die hoofrekenaar en mikrorekenaarnode in staat te stel om oor die seriale verbinding te kan kommunikeer. Hierdie sagteware word in Hoofstuk 4 bespreek.

Die hoofrekenaarsagteware bied die fasiliteit om opstellingsdata te stoor en om datakommunikasie na en vanaf die node te beheer. Dit stoor ook data wat vanaf die node verkry word en is in staat om die node met nuwe data te programmeer.

Die mikrorekenaarnode se sagteware dien as 'n BIOS program vir hierdie afgeleë toestel. Dit hanteer kommunikasie na en vanaf die hoofrekenaar. Dit stuur data op

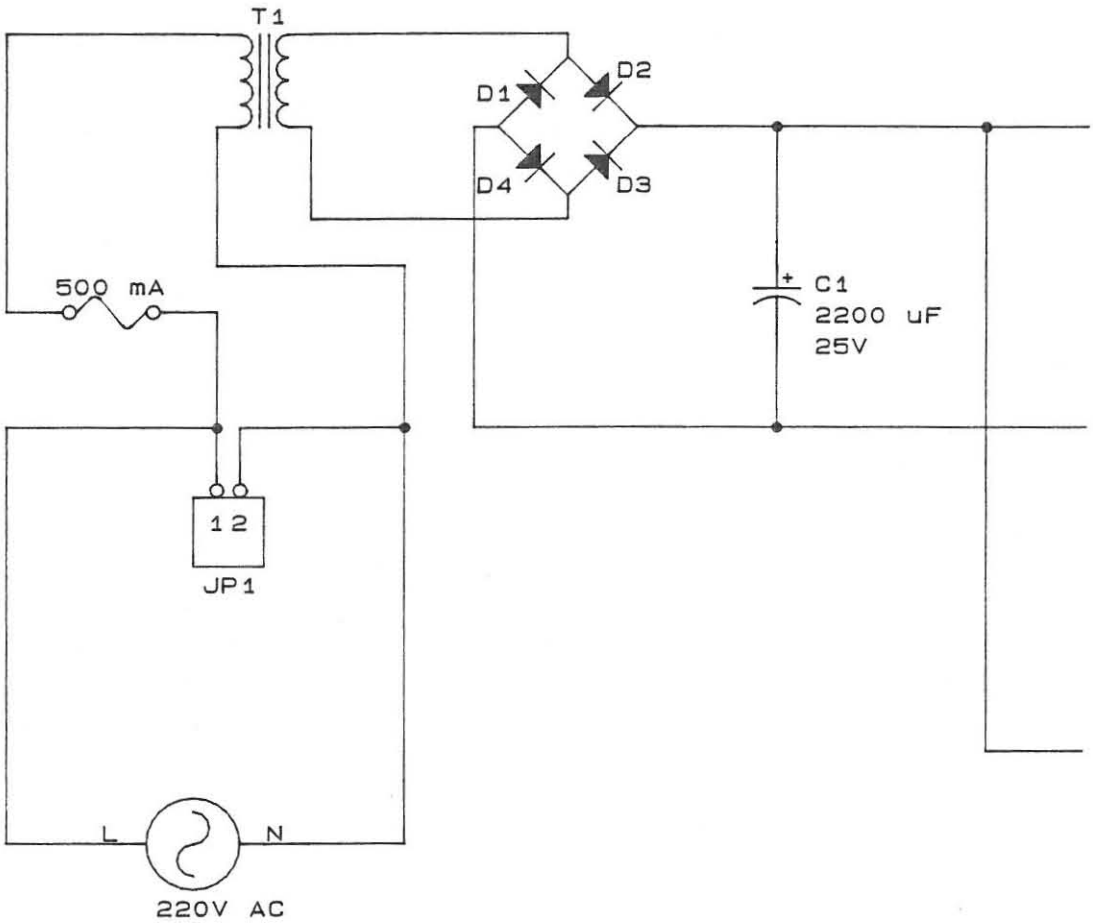
versoek na die hoofrekenaar terug en aanvaar programmeringsdata vanaf die node wat dit in eksterne geheue stoor vir evalueringsdoeleindes.

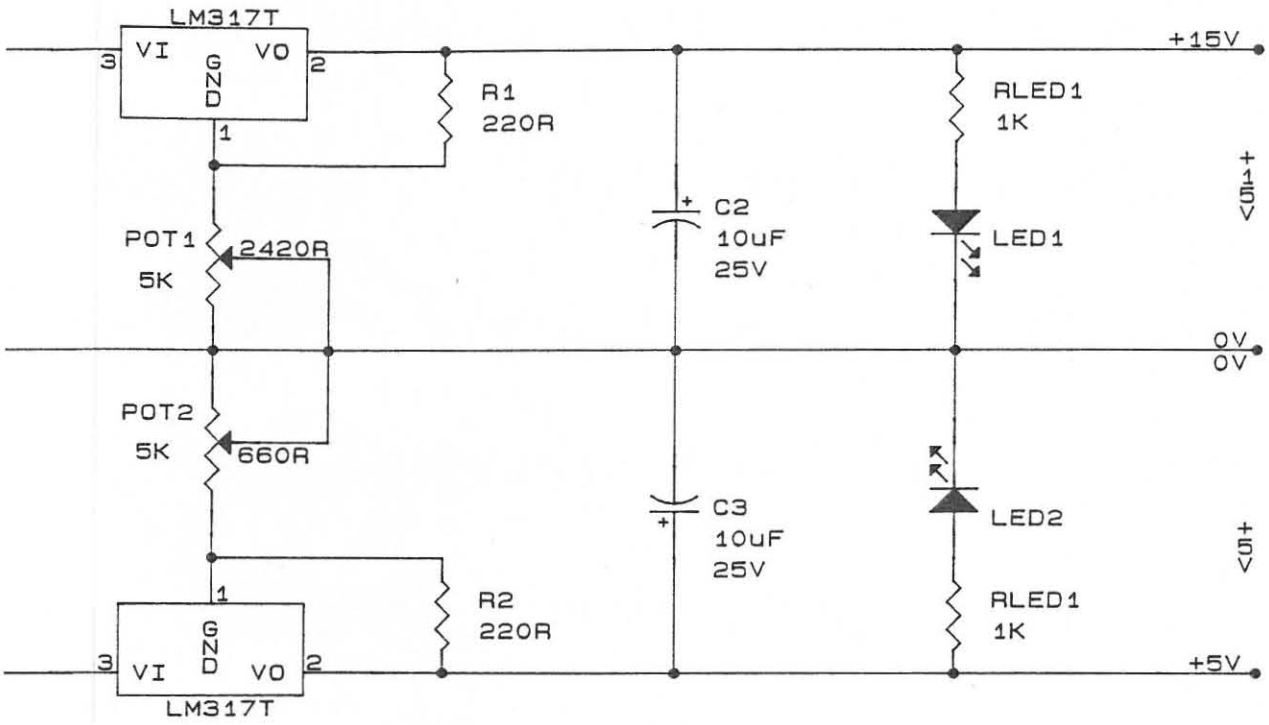
In die evalueringsfase is 'n toetsmodule ontwikkel wat aan die mikrorekenaar verbind word. Hierdie toetsmodule word in Hoofstuk 5 bespreek.

Dit lewer visuele indikasie van elke bis van die twee uitsetpoorte van die 8255PPI wat op die Inset/Uitset module voorkom, asook 'n 8 bis DIP-skakelaarkonfigurasie wat as insetfasiliteit na die een insetpoort van die 8255PPI dien.

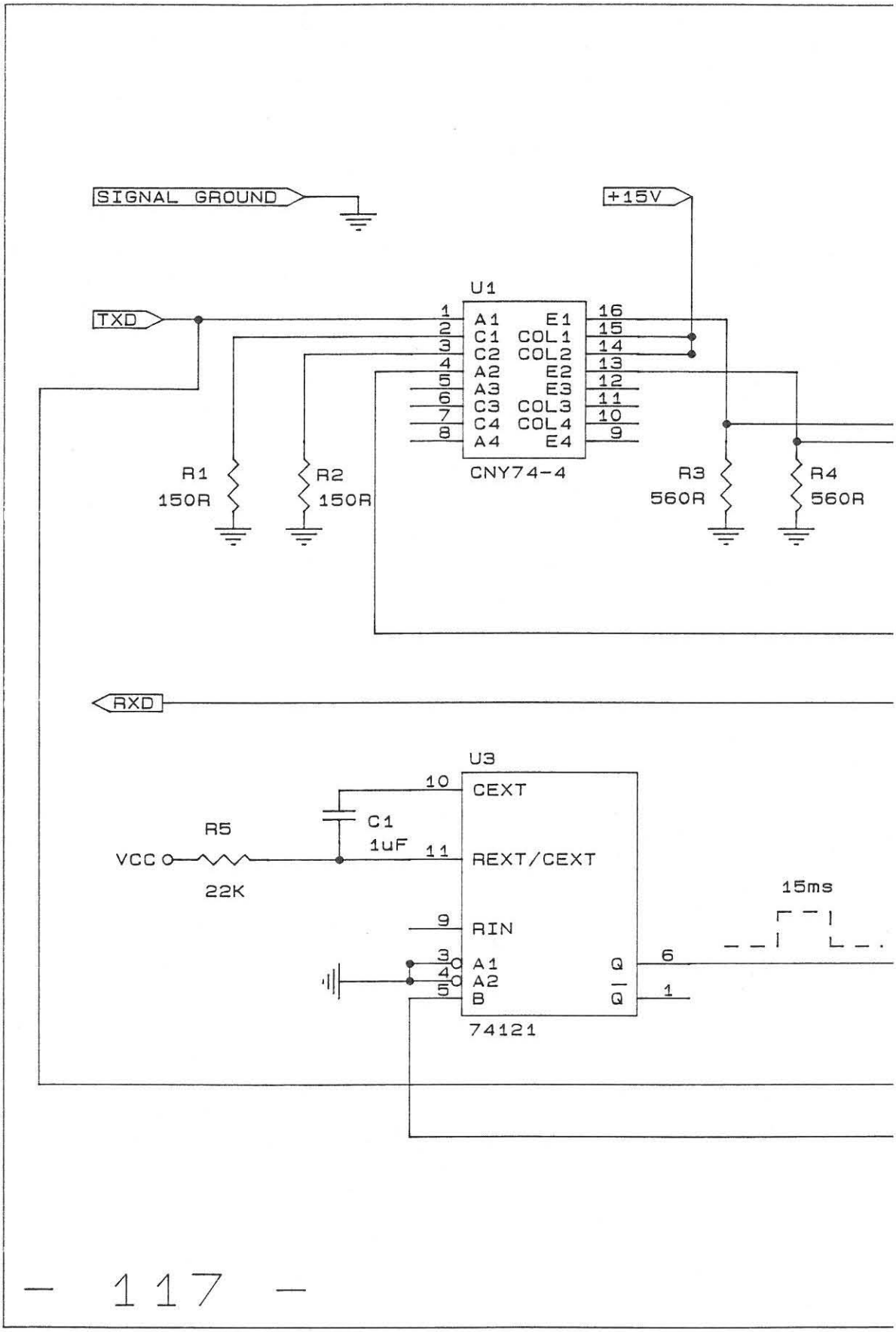
Die uitvoering van die projek het geïllustreer dat die RS485 datakommunikasiestandaard 'n baie geskikte tegniek is om in 'n multitap datakommunikasieomgewing te gebruik. Navorsingsprojekte wat vervolgens aangepak word kan meer op die volgende konsentreer:

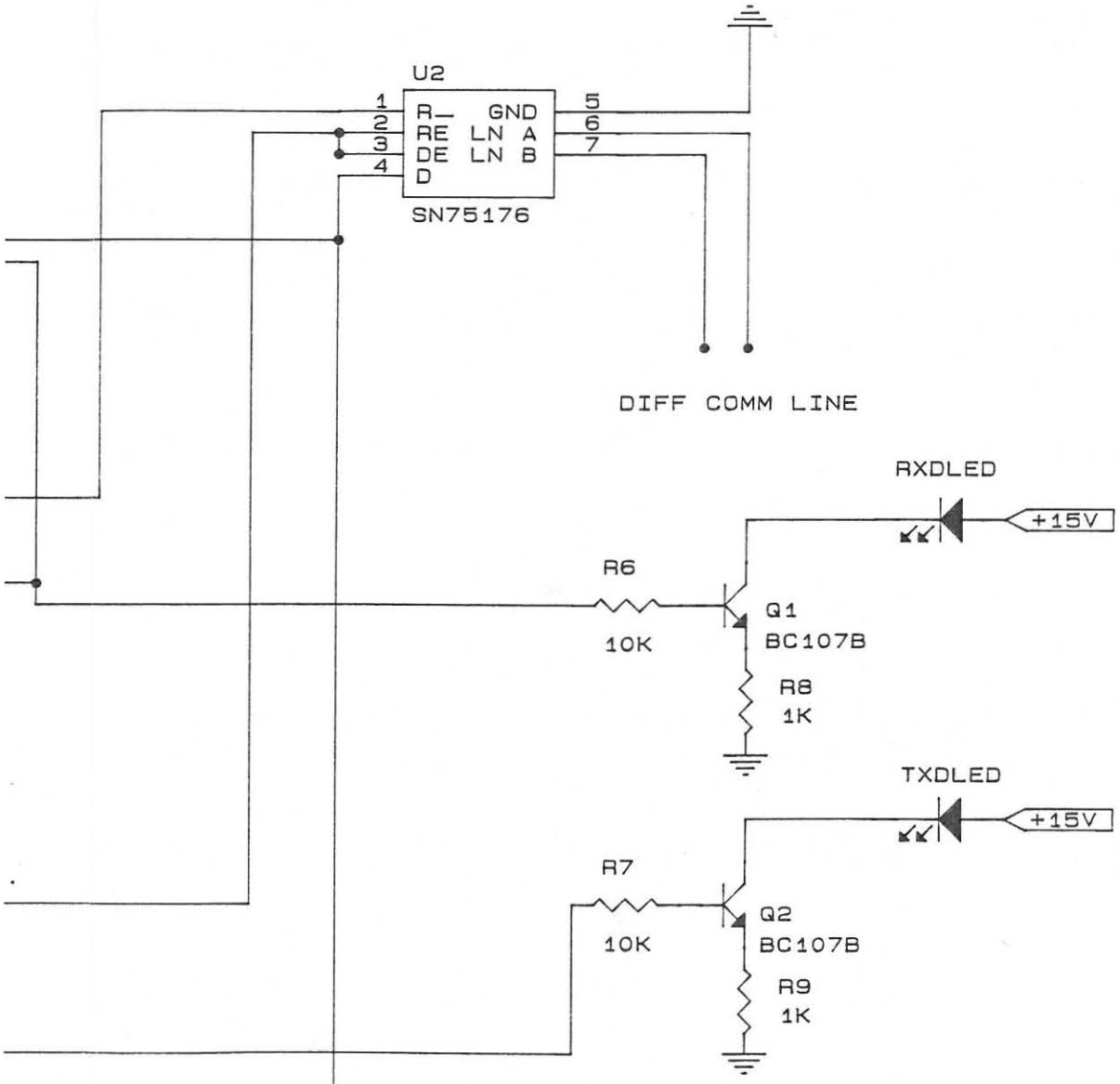
- Vinniger dataoordragsnelhede en
- Langer kommunikasieafstande.



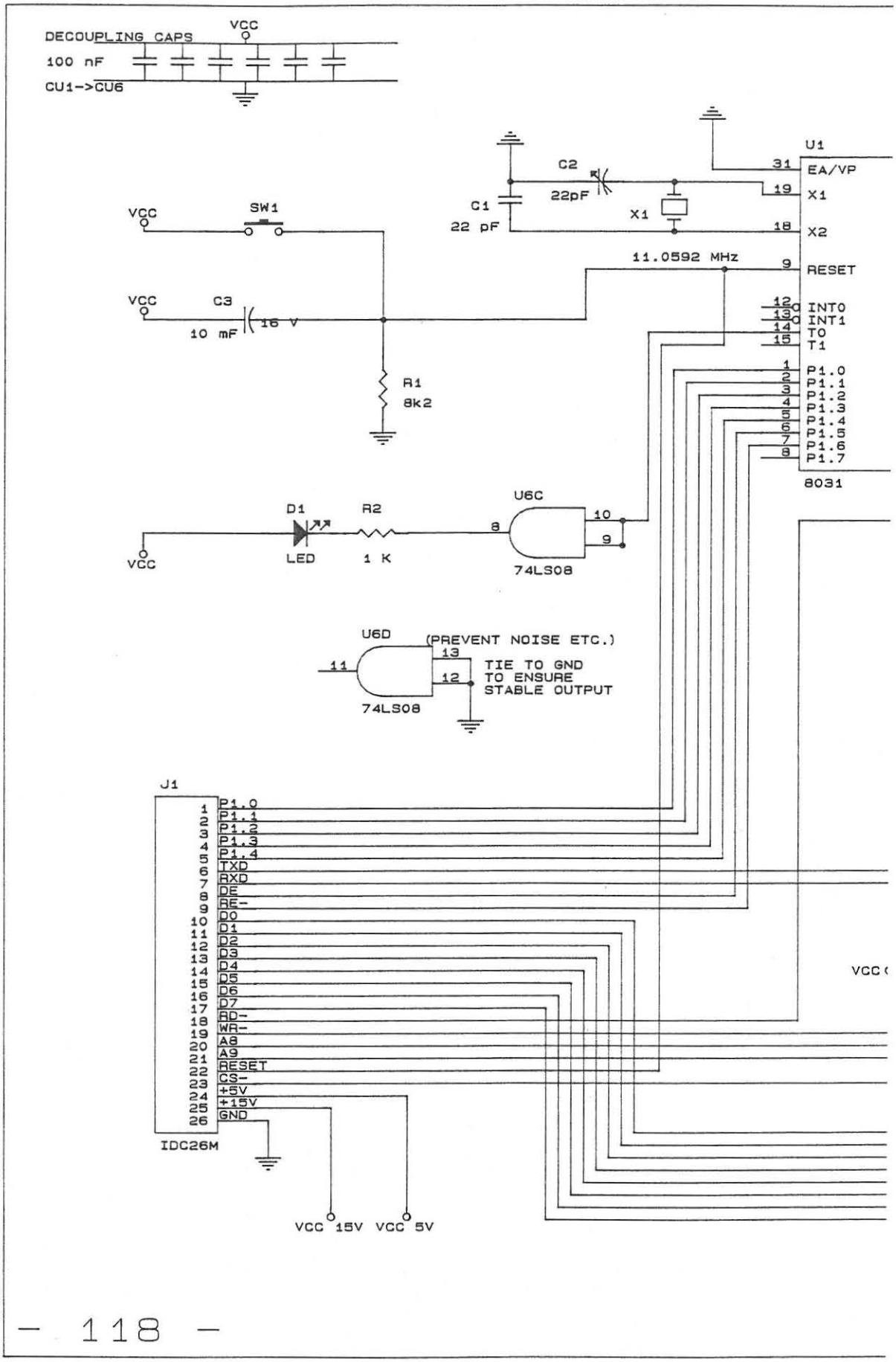


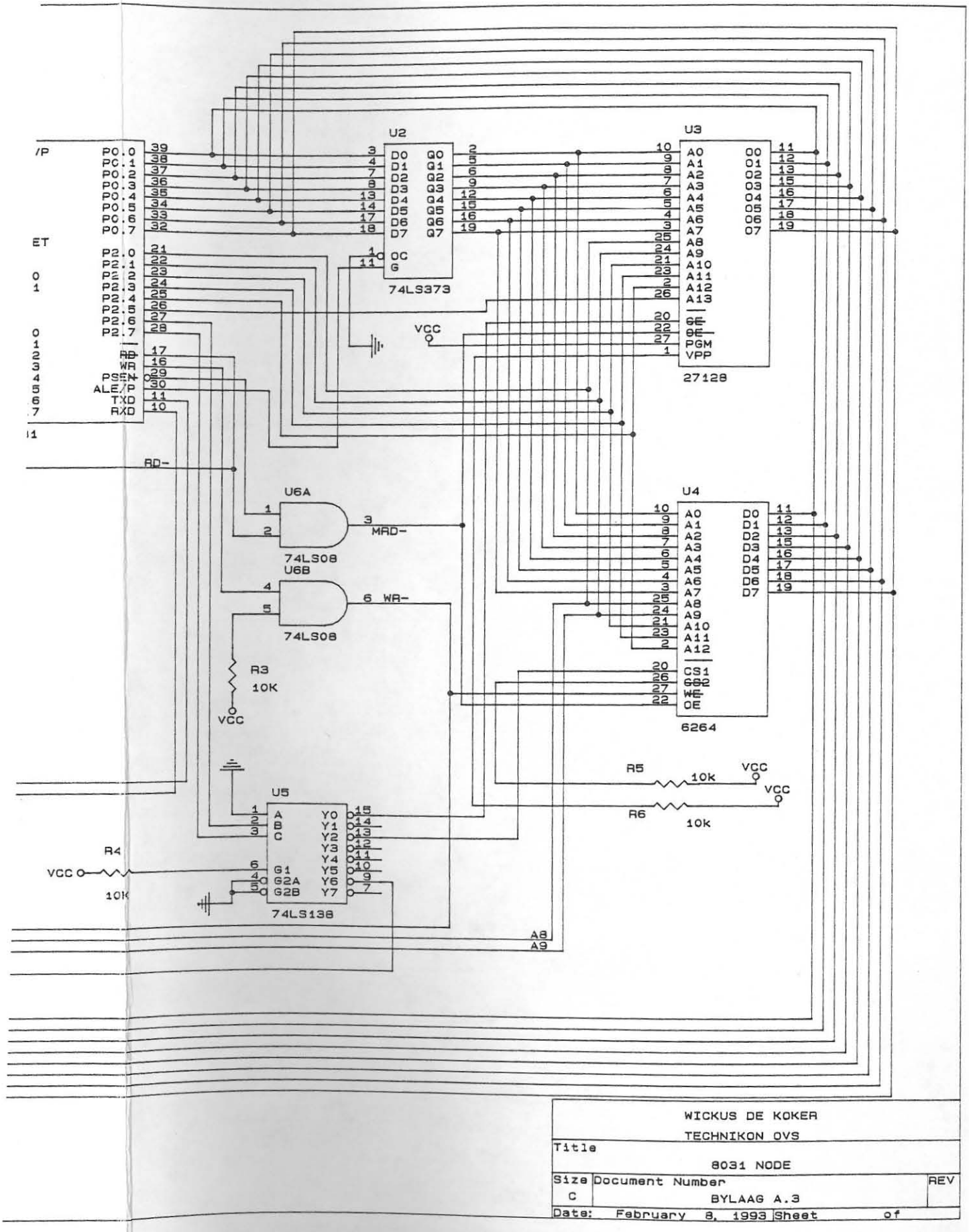
WICKUS DE KOKER TECHNIKON OVS		
Title +15V/+5V KRAGBRON		
Size A	Document Number BYLAAG A.1	REV
Date: February 8, 1993	Sheet of	

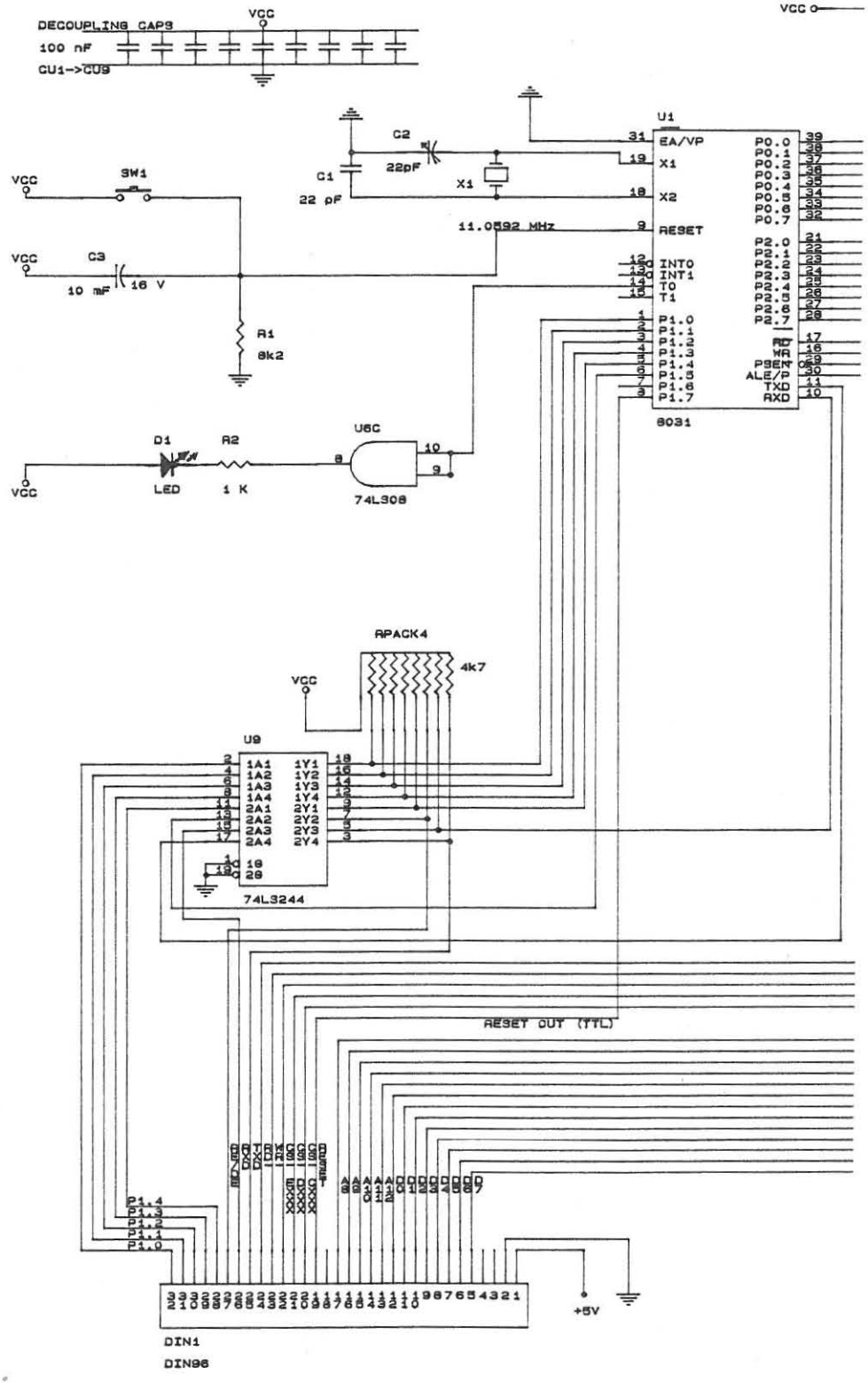


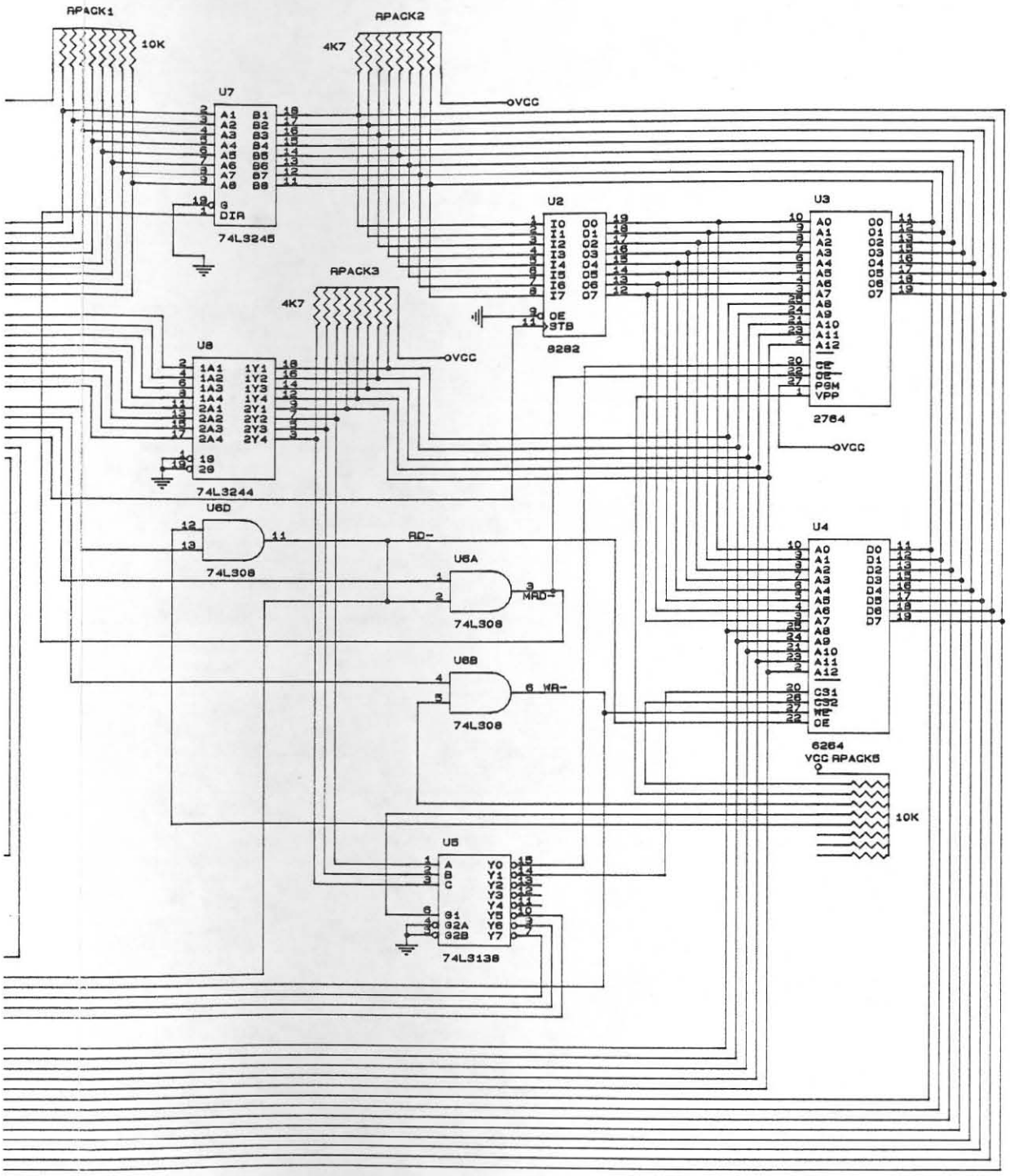


WICKUS DE KOKER TECHNIKON OVS		
Title RS232-NA-RS485 OMSETTER (HALF DUPLEKS)		
Size A	Document Number BYLAAG A.2	REV
Date: February 8, 1993	Sheet of	

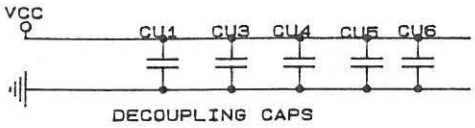




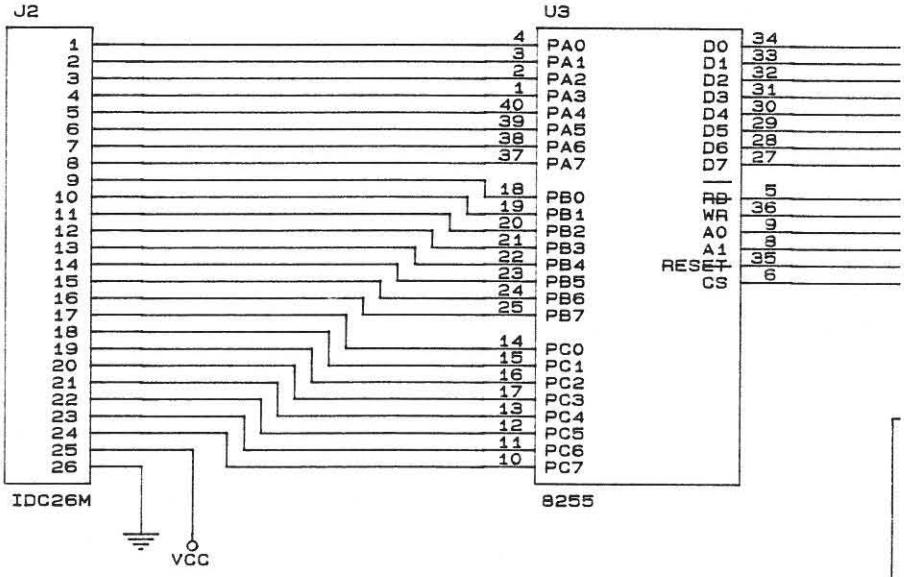


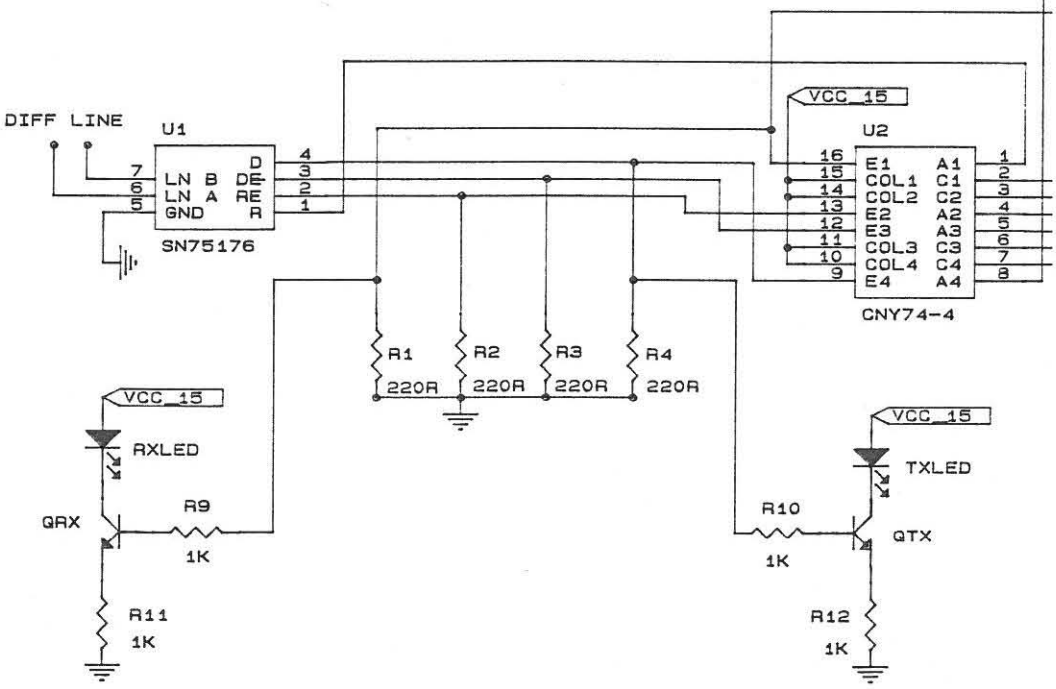


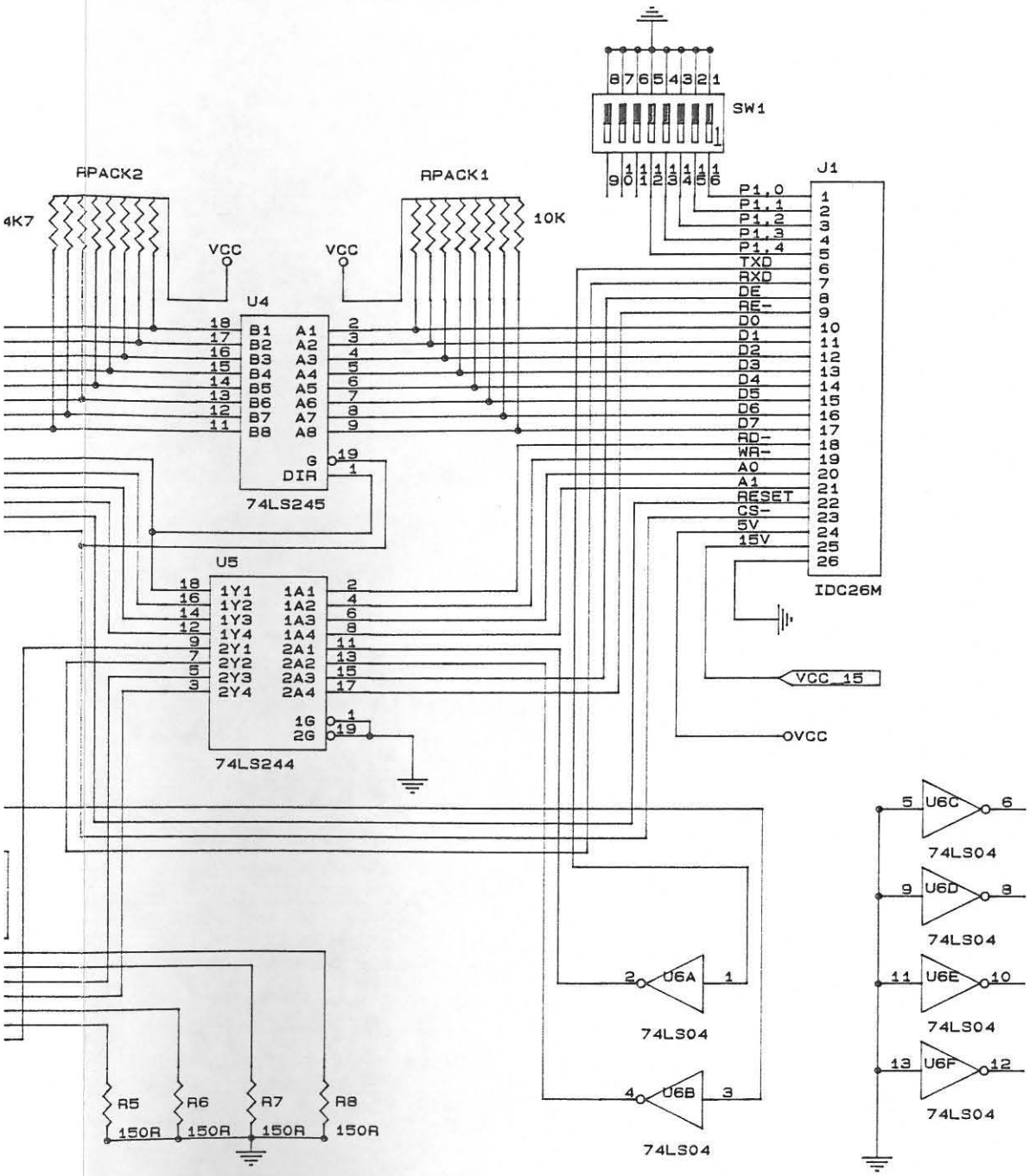
WICKUS DE KOKER TECHNIKON OVS		
Title: 8031 NODE (ALTERNATIEF)		
Size: E	Document Number: BYLAAG A.4	AE
Date: February 8, 1993	Sheet of	



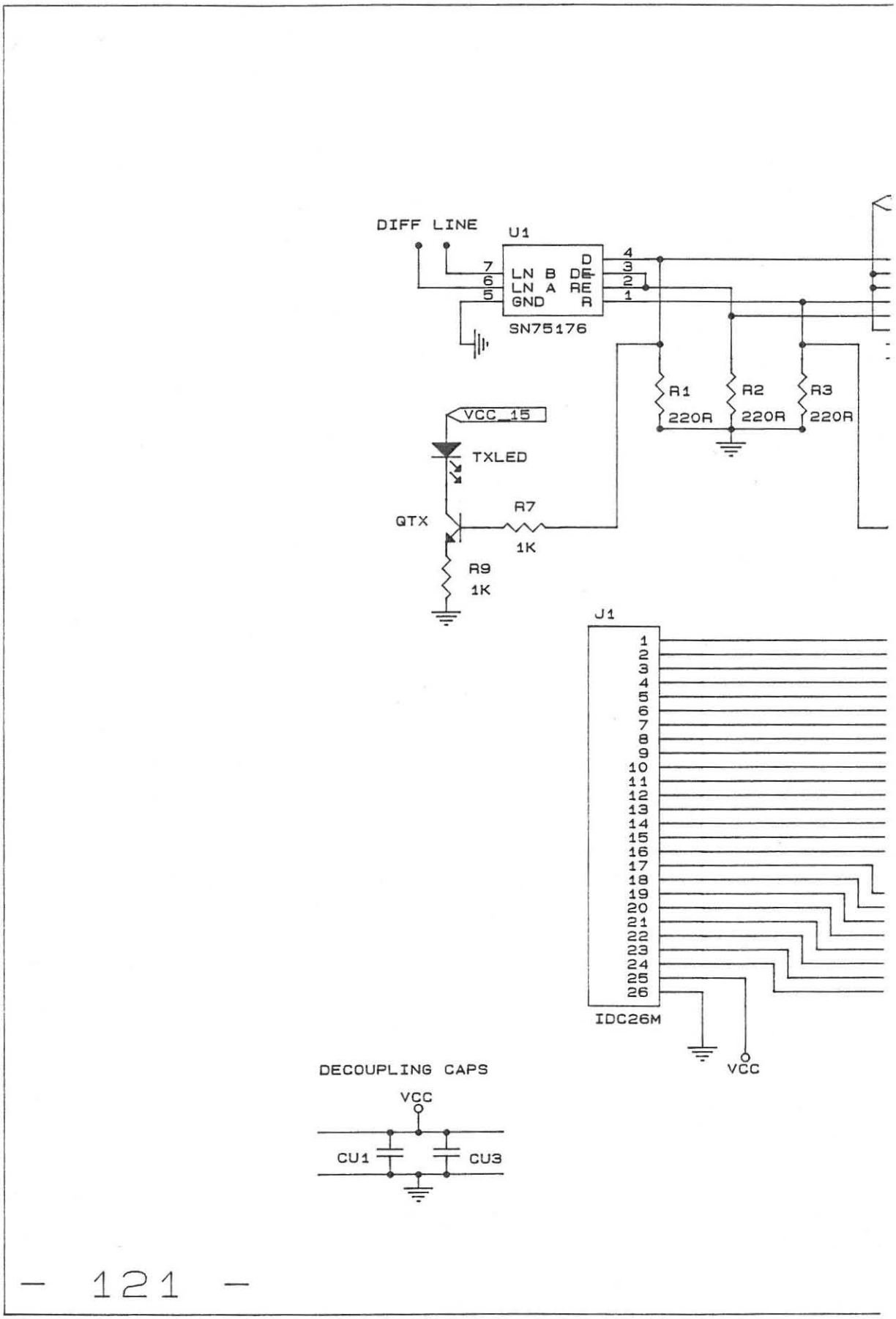
4K7

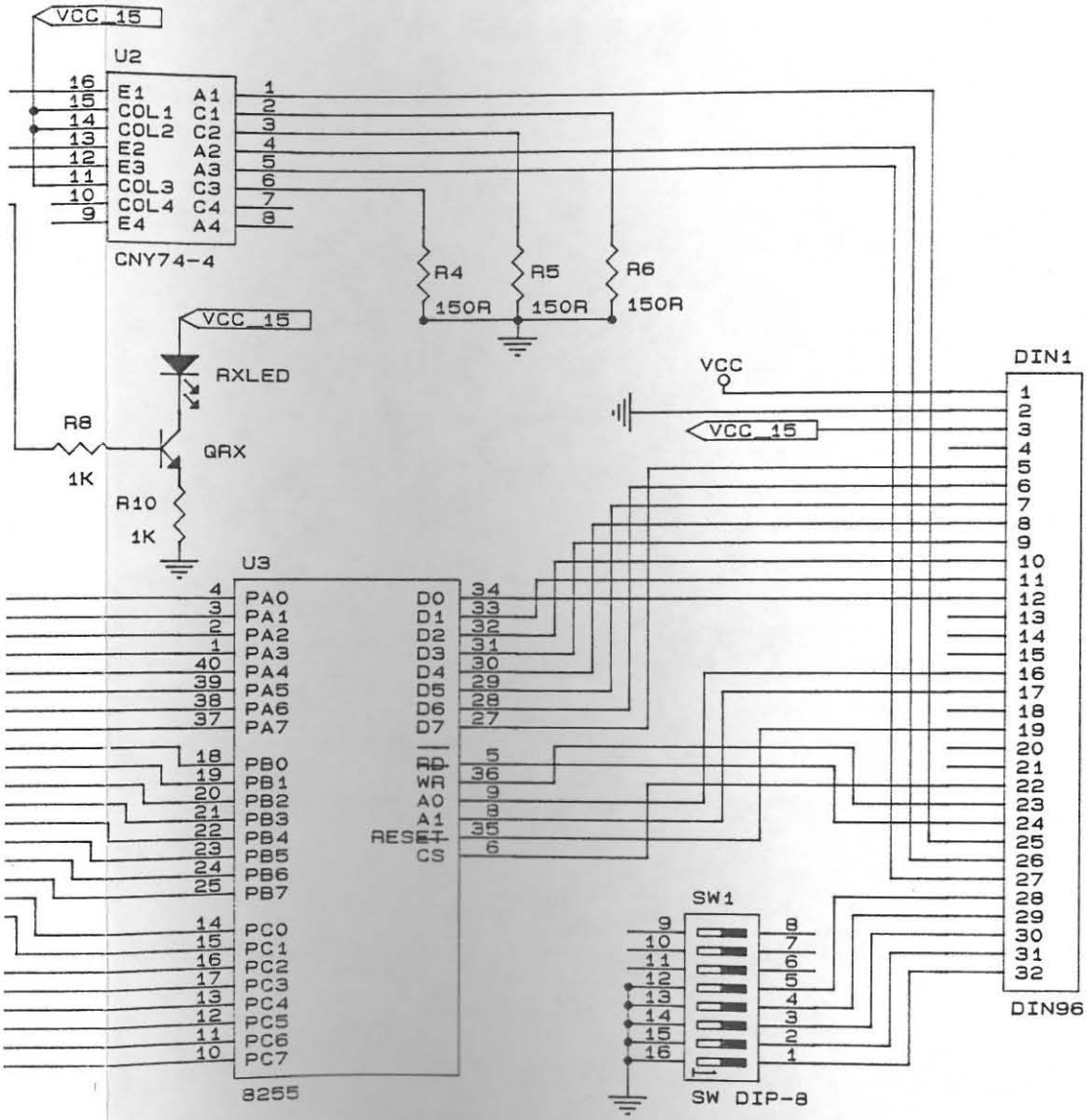




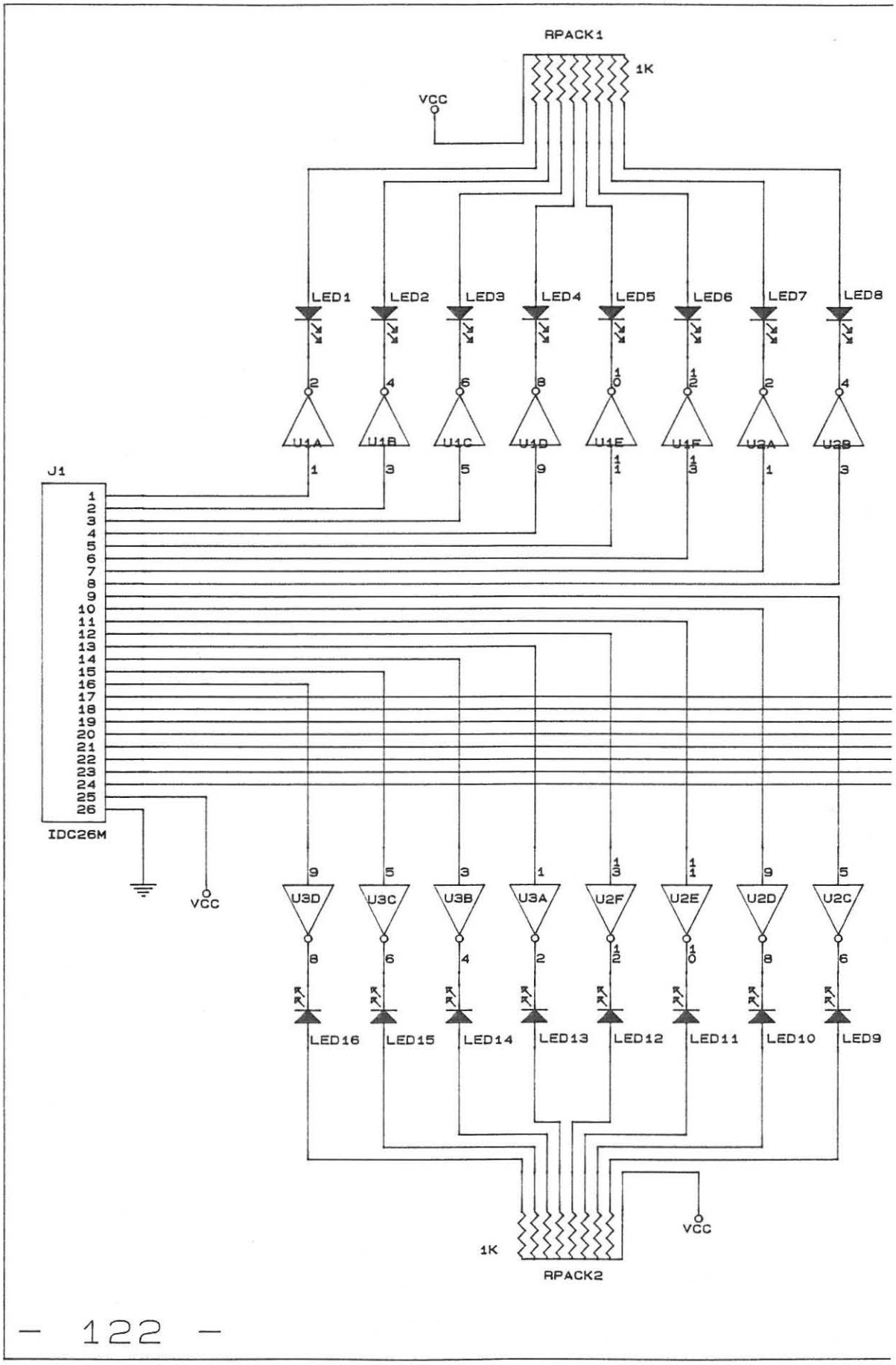


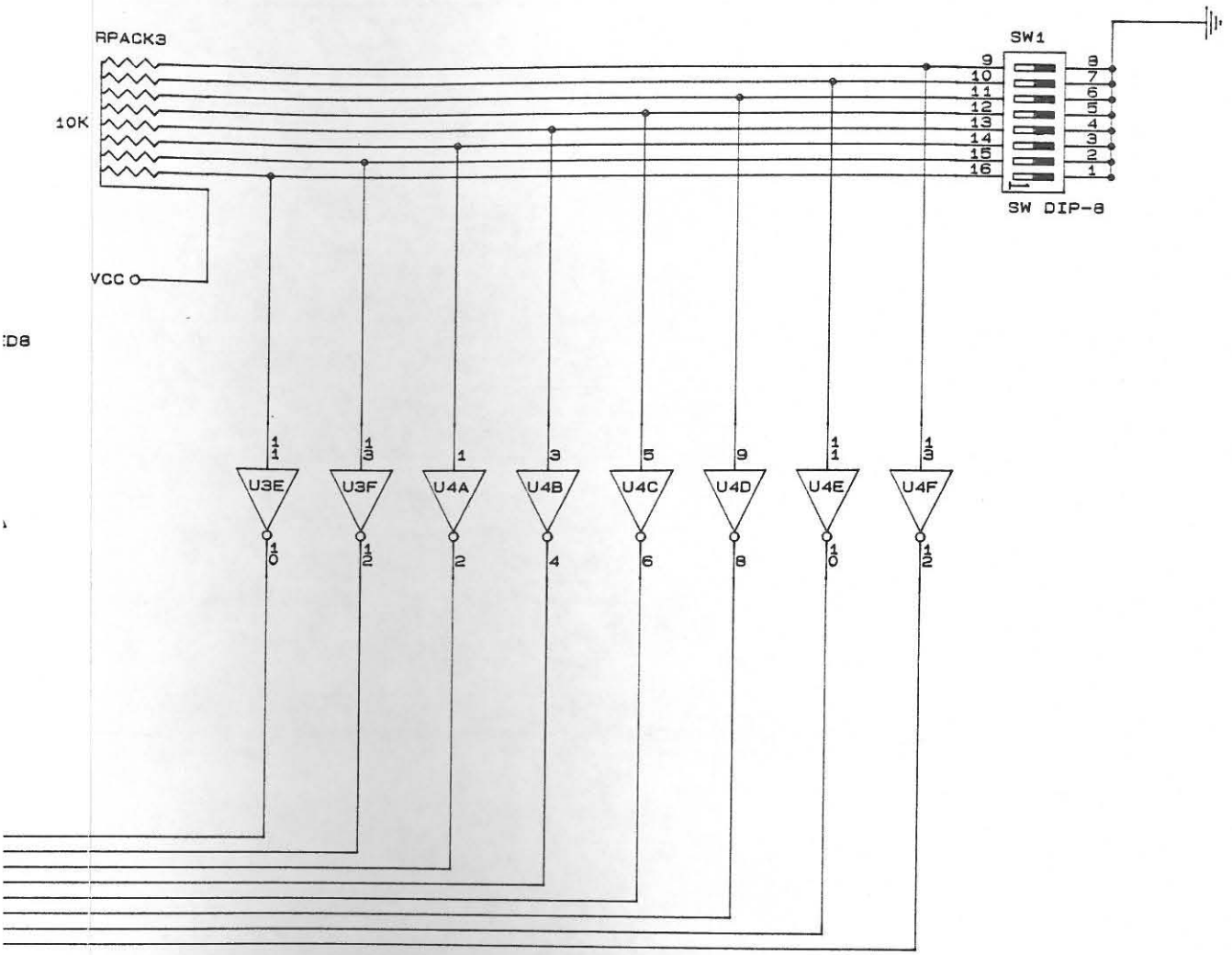
WICKUS DE KOKER TECHNIKON OVS		
Title 8031 NODE I/U MODULE		
Size C	Document Number BYLAAG A.5	REV
Date: February 8, 1993	Sheet of 1	

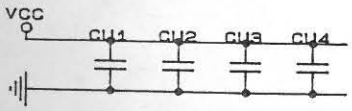




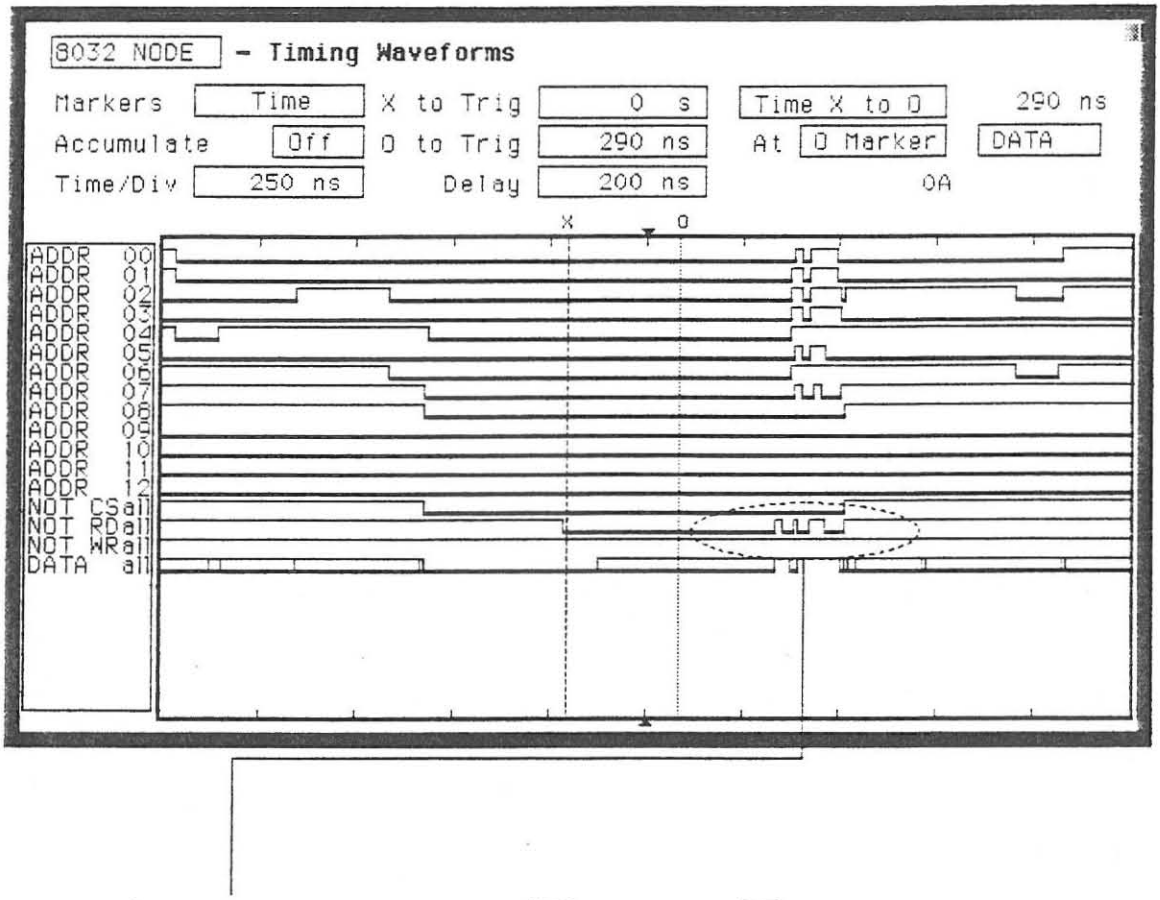
WICKUS DE KOKER TECHNIKON OVS		
Title 8031 NODE I/U MODULE (ALTERNATIEF)		
Size B	Document Number BYLAAG A.6	REV
Date: February 8, 1993	Sheet of	







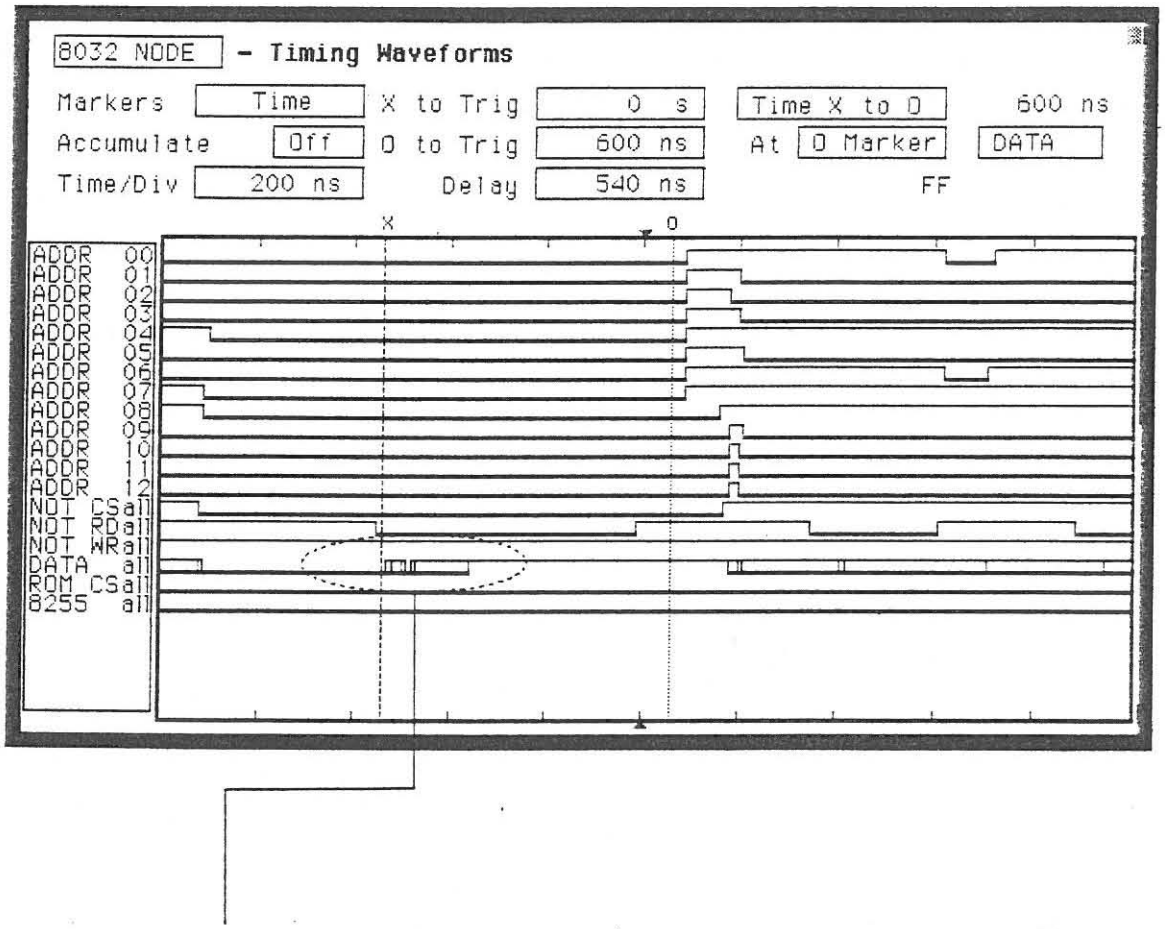
WICKUS DE KOKER TECHNIKON OVS		
Title: 8031 NODE - I/U MODULE TOETS MODULE		
Size: C	Document Number: BYLAAG A.7	REV
Date: February 8, 1993	Sheet of	



LSG (Bylaag A.3 - U4) se $\overline{OE}$ lyn aan $\overline{RD}$ verbind.

Onstabiliteit op $\overline{RD}$ lyn.

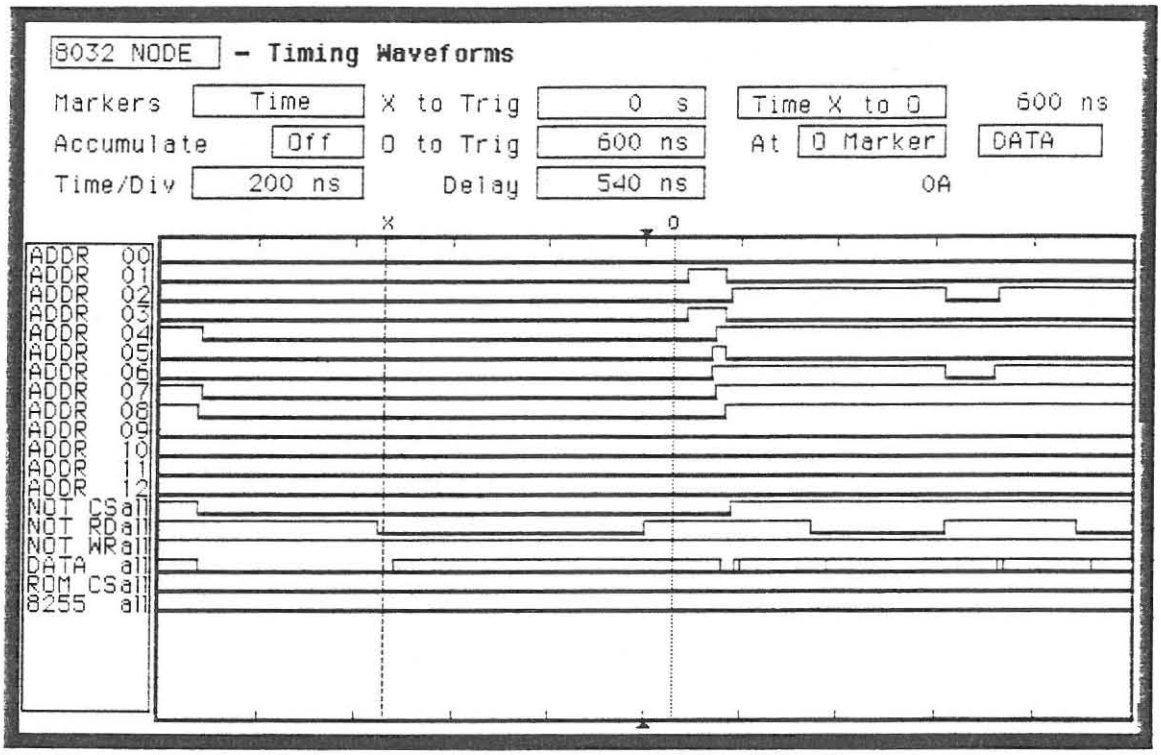
Bylaag A.8 Logika analiseerder uitdruk -:- RD- fout



Inset/Uitset module se databuffer (Bylaag A.4 - U4) $\overline{G}$ lyn direk aan grond verbind.

Onstabiliteit op databus.

Bylaag A.9 Logika analiseerder uitdruk -:- G- fout



Bylaag A.10 Logika analiseerder uitdruk -:- Foute gekorrigeer

Bylaag B.1

MAINPROG.PAS

```
(1)   {*************************************************************}
(2)   {PROGRAM:  8031 NODE MAIN PROGRAM                              }
(3)   {PURPOSE:  To call subprograms to control database management,}
(4)   {          data communications and menu control.               }
(5)   {*************************************************************}

(6)   {$M 10240,0,8192}                          {STACK SIZE = 10K, HEAP MIN = 0K, HEAP MAX = 8K}
(7)   Program MainProgRS485;

(8)                                              {Units to be linked to program}
(9)   Uses Crt,                                  {Pascal CRT unit}
(10)       Dos,                                  {Pascal DOS unit}
(11)       ComnDecl,                             {Common declarations}
(12)       Getkey,                               {Get key pressed/mouse action}
(13)       ScrnUtil,                             {Screen orientated functions => savescreen, etc.}
(14)       Com_Unit,                             {Communication functions}
(15)       EditLn;                               {Line editor}

(16)  Var
(17)    Main1,Main2,                             {Pointers to menu topics of}
(18)    Util1,Util2,                             {different menus.}
(19)    Setup1,Setup2,
(20)    NodeId1,NodeId2,
(21)    GenSetup1,GenSetup2,
(22)    NodeAddr1,NodeAddr2,
(23)    Dbase1,Dbase2          : ShortInt;
(24)    I,J                    : Byte;           {Misc variables for general use}

(25)    CurrPollTime           : String[5];      {String holding current poll time}

(26)    Int28Save,                               {DOS OK TO USE int saved vector}
(27)    OldExitProc            : Pointer;        {Old Exit Proc vector}
(28)    Int28new               : pointer;        {dummy pointer for debugging purpose}

(29)    ReConfig,                                {Reconfigure poll times?}
(30)    Handler_Active         : Boolean;        {Interrupt Handler invoked?}
(31)    Int28Count             : Integer;        {Counter to indicate number of int28H occurrences}
(32)    PrevSec, CurSec        : String[2];      {Previous & current second string variables}
(33)    Curtime                : String[8];      {Current time string}


(34)  {*************************************************************************}
(35)  {Return the calculated poll time using the database entries in the setup file}
(36)  {*************************************************************************}
(37)  Function PollTime(CurrPoll : String) : String;
(38)  Const
(39)      PollArr : Array[1..4] of Byte = (15,30,60,120);
```

Bylaag B.1

MAINPROG.PAS

```
(40)  Var
(41)    Hour,Min : Byte;
(42)    Error    : Integer;
(43)  Begin
(44)      Val(Copy(CurrPoll,1,2),Hour,Error);
(45)      Val(Copy(CurrPoll,4,2),Min,Error);
(46)      If ((Min+PollArr[SetupRec.Poll]) >= 60) then
(47)      Begin
(48)           Min := 0;
(49)           If (SetupRec.Poll in [1..3]) then
(50)           Begin
(51)                If ((Hour+1) = 24) then Hour := 0
(52)                Else Inc(Hour);
(53)           End;
(54)           If (SetupRec.Poll = 4) then
(55)           Begin
(56)                If ((Hour+2) >= 24) then
(57)                   If ((Hour+2) = 24) then Hour := 0
(58)                   Else
(59)                   If ((Hour+2) = 25) then Hour := 1
(60)                   Else
(61)                Else Inc(Hour,2);
(62)           End;
(63)      End
(64)      Else Inc(Min,PollArr[SetupRec.Poll]-(Min mod PollArr[SetupRec.Poll]));
(65)      PollTime := Concat(ConvStr2(Hour),':',ConvStr2(Min));
(66)  End;

(67)  {*********************************************************************}
(68)  {Poll nodes for data if the current system time = poll time calculated}
(69)  {NB: THIS PROCEDURE COULD NOT BE INCORPORATED COMPLETELY DUE TO       }
(70)  {    INTERRUPT CONFLICTIONS IN THE DOS SYSTEM. IT THEREFOR WAS MADE A }
(71)  {    DUMMY PROCEDURE TO INDICATE THE INTENTION OF IT'S USE.          }
(72)  {*********************************************************************}
(73)  {$F+}
(74)  Procedure PollNodes;
(75)  {$F-}
(76)  Var
(77)    NodeCount1,NodeCount2 : Byte;
(78)    ComStr,BaudStr        : String[4];
(79)    Cport,BaudRate        : Word;
(80)    Error                 : Integer;
(81)  Begin
(82)      SaveScreen;
(83)      Window(1,1,80,25);
(84)      Drawbox(10,10,70,15,LightRed,0);
(85)      WriteCenter(10,LightRed,15,' POLLING NODES FOR RESULTS.   PLEASE BE PATIENT. ');
```

Bylaag B.1

MAINPROG.PAS

```
(86)  {***********************************************}
(87)  {THE NEXT SEGMENT OF SOURCE CODE HAD TO PERFORM }
(88)  {THE POLLING OF NODES AS IT IS LINKED TO THE    }
(89)  {INTERRUPT HANDLER FOR POLLING THE NODES.       }
(90)  {***********************************************}
(91)  {
(92)      Str(SetupRec.Comport,ComStr);
(93)      Str(SetupRec.Baudrate,BaudStr);
(94)      DispMessage(True,13,Colors.HlBg,Colors.HlFg,'Initializing com port '+ComStr+','+BaudStr+',N,1');
(95)      InitPort(SetupRec.ComPort,SetupRec.BaudRate);

(96)      For NodeCount1 := 0 to 1 do
(97)          For NodeCount2 := 0 to 7 do
(98)          If NodeDbaseExist(Nodes[NodeCount1,NodeCount2]) then
(99)          Begin
(100)               ClearWindow(11,11,69,14);
(101)               Window(1,1,80,25);
(102)               NodesB := NodeCount1;
(103)               NodesD := NodeCount2;
(104)               WriteAt(12,12,15,0,'Current node to poll    ->'+Nodes[NodeCount1,NodeCount2]);
(105)               WriteAt(12,13,15,0,'Status                   : Extracting data.');
(106)               XtractInfo(Nodes[NodesB,NodesD],'/PNX');
(107)         End;
(108) }
(109) {*****************************************************}
(110) {THE FOLLOWING SOURCE CODE REPRESENTS THE DUMMY POLLING  }
(111) {PROCEDURE TO INDICATE THE INTENDED USE OF THIS FUNCTION }
(112) {*****************************************************}
(113)     For NodeCount1 := 0 to 1 do
(114)         For NodeCount2 := 0 to 7 do
(115)         Begin
(116)               ClearWindow(11,11,69,14);
(117)               Window(1,1,80,25);
(118)               NodesB := NodeCount1;
(119)               NodesD := NodeCount2;
(120)               WriteAt(12,12,15,0,'Current node to poll    ->'+Nodes[NodeCount1,NodeCount2]);
(121)               WriteAt(12,13,15,0,'Status                   : Extracting data.');
(122)               Delay(750);
(123)         End;

(124)     RestoreScreen;
(125)     CurrPollTime := PollTime(CurrPollTime);{Calculate new poll time}
(126) End;
```

Bylaag B.1

MAINPROG.PAS

```
(127) {*************************************************************}
(128) {Does system time fall within 1 minute of current polling time ?}
(129) {*************************************************************}
(130) Function LegalPollTime : Boolean;
(131) Var
(132)   HourVal,
(133)   HpollVal,
(134)   MinVal,
(135)   PollVal,
(136)   Error     : Integer;
(137) Begin
(138)     Val(Copy(SystemTime,1,2),HourVal,Error);
(139)     Val(Copy(CurrPollTime,1,2),HpollVal,Error);
(140)     Val(Copy(SystemTime,4,2),MinVal,Error);
(141)     Val(Copy(CurrPollTime,4,2),PollVal,Error);
(142)     If (HpollVal = 0) then HpollVal := 24;
(143)     If (PollVal = 0) then PollVal := 60;
(144)     If (SetupRec.Poll = 4) then            {120 min}
(145)     Begin
(146)          If ((HpollVal - HourVal) <= 1) then LegalPollTime := ((PollVal - MinVal) > 1)
(147)          Else
(148)          LegalPollTime := True;
(149)     End
(150)     Else
(151)     LegalPollTime := ((PollVal - MinVal) > 1);
(152) End;


(153) {*********************************}
(154) {Extract result for the selected node}
(155) {*********************************}
(156) Procedure ExtractResults;
(157) Var
(158)   ComStr,
(159)   BaudStr  : String;
(160) Begin
(161)     If (Not LegalPollTime) then
(162)     ErrorMsg('POLLING ABOUT TO COMMENCE IN LESS THAN 1 MIN !! PLEASE TRY LATER.')
(163)     Else
(164)     If (Not NodeAddrFileExist) then ErrorMsg('NODE ADDRESSES HAVEN''T BEEN DEFINED YET !!')
(165)     Else
(166)     Begin
(167)          If (Not SetupFileExist) then ErrorMsg('GENERAL SETUP HASN''T BEEN DONE YET !!')
(168)          Else
(169)          Begin
(170)               Window(1,1,80,25);
(171)               DispMessage(False,13,15,0,'Loading ...');
(172)               Str(SetupRec.Comport,ComStr);
```

Bylaag B.1
MAINPROG.PAS

```
(173)                Str(SetupRec.Baudrate,BaudStr);
(174)                                              {Call subprogram to get results from node}
(175)                                              {ComStr = comm port, BaudStr = baud rate}
(176)                Exec('NODEUTIL.EXE','NODE8031 /X '+ComStr+' '+BaudStr);
(177)             End;
(178)       End;
(179)  End;

(180)  {*********************************}
(181)  {Send 8255 pin config data to node}
(182)  {*********************************}
(183)  Procedure ProgramNode;
(184)  Var
(185)    Cport,BaudR : String[5];                  {Comport & baudrate strings}
(186)    Address,                                  {Node address}
(187)    SecCount    : Byte;                       {Second counter}
(188)    Pin, Wait   : Char;                       {Pin = Input/Ouput pin}
(189)    Fsize,                                    {Size of database in records}
(190)    CurrSec,                                  {Current second string}
(191)    PrevSec     : String[2];                  {Previous second string}
(192)    DelayEnd    : Boolean;                    {End of acknowledge delay?}
(193)    DataStream  : String;                     {Data buffer/stream}
(194)  Begin
(195)       Acknowledge := False;                  {No acknowledgement from node}
(196)       Reset(NaddrFile);                      {Get addresses from address database}
(197)       Read(NaddrFile,NaddrRec);
(198)       Close(NaddrFile);

(199)       Str(SetupRec.ComPort,Cport);
(200)       Str(SetupRec.BaudRate,BaudR);
(201)       DispMessage(True,13,Colors.HlBg,Colors.HlFg,'Initializing com port '+
(202)                   Cport+','+BaudR+',N,1');
(203)                                              {Initialize comm port}
(204)       InitPort(SetupRec.ComPort,SetupRec.BaudRate);
(205)       Drawbox(10,10,70,20,Colors.HlFg,Colors.TopicBg);
(206)       WriteCenter(10,Colors.TxtFg,Colors.TxtBg,' Programming node : '+Nodes[NodesB,NodesD]+' ');
(207)       Window(11,11,69,19);
(208)       SetDtrHigh(True);                      {DTR = high}

(209)       TxString(#32);                         {Ensure rxd buffers of nodes are clear}

(210)       {Convert hex string to byte value and add address id bit}
(211)       {eg: 0DH or 80H = 8DH => MSB set to indicate id bit send}
(212)       Address := HexStrToByte(NaddrRec.NodeArr[NodesB,NodesD]) or $80;

(213)       Write('Addressing node');
(214)       TxString(Char(Address));               {Transmit node address + id bit}
```

Bylaag B.1

MAINPROG.PAS

```
(215)     {Give node sufficient time to acknowledge - max 10 seconds}
(216)     PrevSec := Copy(SystemTime,7,2);
(217)     SecCount := 0;
(218)     DelayEnd := False;
(219)     While (Not Acknowledge) and (Not DelayEnd) do
(220)     Begin
(221)          CurrSec := Copy(SystemTime,7,2);
(222)          If (CurrSec <> PrevSec) and (Not Acknowledge) then
(223)          Begin
(224)               Write('.');
(225)               Inc(SecCount);
(226)               PrevSec := CurrSec;
(227)          End;
(228)          If (SecCount = 10) then DelayEnd := True;
(229)     End;

(230)     Delay(1000);
(231)     Writeln;

(232)     If Acknowledge then                    {Node acknowledged?}
(233)     Begin                                  {... Yes!}
(234)          Delay(1000);
(235)          Writeln('Transmitting program command ...');
(236)          TxString(#32+Char(Prg));          {Transmit program id char}
(237)          Delay(1000);
(238)          Reset(NodeFile);
(239)          Str(FileSize(NodeFile),Fsize);    {Get filesize from node config database}
(240)          If (FileSize(Nodefile) < 10) then Fsize := Concat('0',Fsize);
(241)          Writeln('Programming node ... ('+Fsize+' records)');
(242)          TxString(Fsize);                  {Transmit first 2 characters of datastream}
(243)          While Not Eof(NodeFile) do        {Add reset of node data to datastream}
(244)          Begin
(245)               Read(NodeFile,NodeRec);
(246)               DataStream := '';
(247)               With NodeRec do
(248)               Begin
(249)                    WriteAt(1,WhereY,15,0,'Pin :'+Port+'  '+Name+'  ');
(250)                    If (Ptype = Output) then
(251)                    Begin
(252)                         Write('Output');
(253)                         Pin := 'O';
(254)                    End
(255)                    Else
(256)                    Begin
(257)                         Write('Input ');
(258)                         Pin := 'I';
(259)                    End;
```

Bylaag B.1
MAINPROG.PAS

```
(260)                                       {Add pin id, eg. A.0 to stream}
(261)                    DataStream := Concat(DataStream,Copy(Port,7,3));
(262)                                       {Add (I)nput/(O)utput pin id to stream}
(263)                    DataStream := Concat(DataStream,Pin);
(264)                    For I := 1 to 2 do
(265)                        For J := 1 to 5 do  {Add on/off times to stream}
(266)                        DataStream := Concat(DataStream,Times[I,J]);
(267)                    TxString(DataStream);   {Transmit data stream}
(268)               End;
(269)          End;
(270)          Close(NodeFile);
(271)          Writeln;
(272)          Writeln('Syncronizing node time with master''s ...');
(273)          TxString(Copy(SystemTime,1,8));   {Sync node's time with master's}
(274)          Delay(500);
(275)          Txstring(Char(Eot));              {Transmit End Of Transmission (eot) id}
(276)          Writeln('End of transmission.');
(277)          Wait := GetKeyInput;              {Wait for key pressed}
(278)     End
(279)     Else                                   {Node doesn't acknowledge}
(280)     Writeln('No acknowledgement from node => Check addresses !!!');
(281)     Delay(1000);
(282)     SetDtrHigh(False);                     {Restore DTR}
(283)     RestoreValues;                         {Restore com attributes}
(284) End;

(285) {*********************************}
(286) {Select node to perform operations on}
(287) {*********************************}
(288) Procedure SelectNode;
(289) Var
(290)   NodeChosen : Boolean;
(291) Begin
(292)     If (Not NodeAddrFileExist) then ErrorMsg('NODE ADDRESSES HAVEN''T BEEN DEFINED YET !!')
(293)     Else
(294)     Begin
(295)          Drawbox(20,10,45,21,15,0);
(296)          WriteAt(22,20,15,0,'Please select node ...');
(297)          For I := 1 to 8 do WriteAt(22,I+10,Colors.TopicFg,Colors.TopicBg,Nodes[0,I-1]);
(298)          For I := 1 to 8 do WriteAt(34,I+10,Colors.TopicFg,Colors.TopicBg,Nodes[1,I-1]);
(299)          NodeChosen := NodeChoose;

(300)          If NodeChosen then                {Node selected = Yes!}
(301)             If (Not NodeDbaseExist(Nodes[NodesB,NodesD])) then ErrorMsg('NODE DATABASE DOESN''T EXIST YET !!')
(302)             Else
(303)             Begin
(304)                  If (Util2 = 0) then
(305)                  Begin
```

Bylaag B.1
MAINPROG.PAS

```
(306)                    If (Not LegalPollTime) then
(307)                    ErrorMsg('POLLING ABOUT TO COMMENCE IN LESS THAN 1 MIN !! PLEASE TRY LATER.')
(308)                    Else
(309)                    ExtractResults;      {Xtract results}
(310)               End;

(311)               If (Util2 = 2) then
(312)               Begin
(313)                    If (Not LegalPollTime) then
(314)                    ErrorMsg('POLLING ABOUT TO COMMENCE IN LESS THAN 1 MIN !! PLEASE TRY LATER.')
(315)                    Else
(316)                    ProgramNode;         {Program node}
(317)               End;

(318)               If (Util2 = 1) then       {Result database management}
(319)               If ResFileExist(Nodes[NodesB,NodesD]) then
(320)               Begin
(321)                    Window(1,1,80,25);
(322)                    DispMessage(False,13,15,0,'Loading ...');
(323)                    Case Dbase2 of       {View result for node}
(324)                         0 : Exec('RES_UTIL.EXE','NODE8031 /V '+ResFileName);
(325)                                           {Print results}
(326)                         1 : Exec('RES_UTIL.EXE','NODE8031 /P '+ResFileName);
(327)                                           {Sort results in ascending order}
(328)                         2 : Exec('RES_UTIL.EXE','NODE8031 /S '+ResFileName);
(329)                    End; {case dbase2}
(330)               End
(331)               Else ErrorMsg('NO RESULTS AVAILABLE FOR NODE !!');
(332)          End;
(333)     End;
(334) End;


(335) {*****************************************}
(336) {Select database management topics from menu}
(337) {*****************************************}
(338) Procedure DbaseChoose;
(339) Var
(340)   Dbase : Char;
(341)   Fin  : Boolean;

(342)   Procedure DbaseHilight;              {Menu topic highlight}
(343)   Begin
(344)        Window(32,Dbase1+10,32+Length(DbaseMenu[Dbase1]),Dbase1+10);
(345)        SetTxtColor(Colors.TopicFg,Colors.TopicBg);
(346)        Clrscr;
(347)        Write(DbaseMenu[Dbase1]);
(348)        Window(32,Dbase2+10,32+Length(DbaseMenu[Dbase2]),Dbase2+10);
```

Bylaag B.1

MAINPROG.PAS

```
(349)         SetTxtColor(Colors.HlBg,Colors.HlFg);
(350)         Clrscr;
(351)         Write(DbaseMenu[Dbase2]);
(352)         SetTxtColor(15,0);
(353)    End;

(354)  Begin                                          {Select topic}
(355)      Fin := False;
(356)      DbaseHilight;
(357)      Repeat
(358)            Dbase1 := Dbase2;
(359)            Dbase := GetkeyInput;
(360)            Case Dbase of
(361)                 Enter,MouseEnter  : SelectNode;
(362)                 CurDown,MouseDown : Inc(Dbase2);
(363)                 CurUp,MouseUp     : Dec(Dbase2);
(364)                 Esc,MouseEsc      : Fin := True;
(365)            End; {Case Dbase}
(366)            If (Dbase2) = -1 then Dbase2 := 2
(367)            Else
(368)            If (Dbase2 = 3) then Dbase2 := 0;
(369)            If (Not Fin) then DbaseHilight
(370)            Else
(371)            Begin
(372)                 Window(3,5,78,21);
(373)                 SetTxtColor(15,0);
(374)                 Clrscr;
(375)            End;
(376)      Until Fin or (Dbase = Enter) or (Dbase = MouseEnter);
(377)  End;

(378)  {*****************************************}
(379)  {Show database management menu and select topic}
(380)  {*****************************************}
(381)  Procedure DoDbaseMenu;
(382)  Begin
(383)      DrawBox(31,9,40,13,15,0);
(384)      For I := 0 to 2 do WriteAt(32,I+10,Colors.TopicFg,Colors.TopicBg,DbaseMenu[I]);
(385)      DbaseChoose;
(386)  End;

(387)  {***********************************}
(388)  {Utility menu topic selection procedure}
(389)  {***********************************}
(390)  Procedure UtilChoose;
(391)  Var
(392)    Util : Char;
(393)    Fin  : Boolean;
```

Bylaag B.1

MAINPROG.PAS

```
(394)    Procedure UtilHilight;                    {Utility menu topic highlight procedure}
(395)    Begin
(396)         Window(13,Util1+6,13+Length(UtilMenu[Util1]),Util1+6);
(397)         SetTxtColor(Colors.TopicFg,Colors.TopicBg);
(398)         Clrscr;
(399)         Write(UtilMenu[Util1]);
(400)         Window(13,Util2+6,13+Length(UtilMenu[Util2]),Util2+6);
(401)         SetTxtColor(Colors.HlBg,Colors.HlFg);
(402)         Clrscr;
(403)         Write(UtilMenu[Util2]);
(404)         SetTxtColor(15,0);
(405)    End;

(406)  Begin
(407)      Fin := False;                            {Choose utility needed}
(408)      UtilHilight;
(409)      Repeat
(410)            Util1 := Util2;
(411)            Util := GetkeyInput;
(412)            Case Util of
(413)                 Enter,MouseEnter  : Case Util2 of
(414)                                          0 : Begin
(415)                                                SaveScreen;
(416)                                                ExtractResults;
(417)                                                RestoreScreen;
(418)                                              End;
(419)                                          1 : Begin
(420)                                                SaveScreen;
(421)                                                DoDbaseMenu;
(422)                                                RestoreScreen;
(423)                                              End;
(424)                                          2 : Begin
(425)                                                SaveScreen;
(426)                                                SelectNode;
(427)                                                RestoreScreen;
(428)                                              End;

(429)                                     End;
(430)                 CurDown,MouseDown : Inc(Util2);
(431)                 CurUp,MouseUp     : Dec(Util2);
(432)                 Esc,MouseEsc      : Fin := True;
(433)                 CurLeft,MouseLeft : Begin
(434)                                          Dec(Main2);
(435)                                          Fin := True;
(436)                                          SetEnterResident;
(437)                                     End;
(438)                 CurRight,MouseRight : Begin
(439)                                           Inc(Main2);
```

Bylaag B.1
MAINPROG.PAS

```
(440)                                                Fin := True;
(441)                                                SetEnterResident;
(442)                                           End;
(443)           End; {Case Util}
(444)           If (Util2) = -1 then Util2 := 2
(445)           Else
(446)           If (Util2 = 3) then Util2 := 0;
(447)           If (Not Fin) then UtilHilight
(448)           Else
(449)           Begin
(450)                Window(3,5,78,21);
(451)                SetTxtColor(15,0);
(452)                Clrscr;
(453)           End;

(454)      Until Fin;
(455) End;

(456) {*******************************}
(457) {View the general setup parameters}
(458) {*******************************}
(459) Procedure ViewGeneralParams;
(460) Var
(461)   Wait : Char;
(462) Begin
(463)      If (Not SetupFileExist) then ErrorMsg('GENERAL SETUP HASN''T BEEN DONE YET !!')
(464)      Else
(465)      Begin
(466)           Drawbox(17,10,50,21,15,0);
(467)           WriteAt(25,10,Colors.HlBg,Colors.HlFg,' View General Setup ');
(468)           WriteAt(19,20,Colors.TxtFg,Colors.TxtBg,'<Esc> to quit ...');
(469)           WriteAt(19,12,15,0,'Com Port  -> ');
(470)           SetTxtColor(LightRed,0);
(471)           Write(SetupRec.Comport);
(472)           SetTxtColor(15,0);
(473)           WriteAt(19,13,15,0,'Baud Rate -> ');
(474)           SetTxtColor(LightRed,0);
(475)           Write(SetupRec.BaudRate);
(476)           SetTxtColor(15,0);
(477)           WriteAt(19,14,15,0,'Parity    -> ');
(478)           SetTxtColor(LightRed,0);
(479)           Write('NONE');
(480)           SetTxtColor(15,0);
(481)           WriteAt(19,15,15,0,'Stop Bits -> ');
(482)           SetTxtColor(LightRed,0);
(483)           Write('1');
(484)           SetTxtColor(15,0);
(485)           WriteAt(19,16,15,0,'Data Bits -> ');
```

Bylaag B.1
MAINPROG.PAS

```
(486)           SetTxtColor(LightRed,0);
(487)           Write('8');
(488)           SetTxtColor(15,0);
(489)           WriteAt(19,17,15,0,'Polling Time Interval -> ');
(490)           SetTxtColor(LightRed,0);
(491)           Case SetupRec.Poll of
(492)                 1 : Write('15 m');
(493)                 2 : Write('30 m');
(494)                 3 : Write('60 m');
(495)                 4 : Write('120 m');
(496)           End;
(497)           SetTxtColor(15,0);
(498)           WriteAt(19,18,15,0,'Database Exp Interval -> ');
(499)           SetTxtColor(LightRed,0);
(500)           Case SetupRec.Hold of
(501)                 1 : Write('24 h');
(502)                 2 : Write('36 h');
(503)                 3 : Write('48 h');
(504)                 4 : Write('72 h');
(505)           End;
(506)           SetTxtColor(15,0);
(507)           Repeat
(508)                 Wait := GetkeyInput;
(509)           Until (Wait = Esc);
(510)     End;
(511) End;

(512) {*************************************************************************}
(513) {Edit the general setup parameters => com port, poll intv, dbase exp intv}
(514) {*************************************************************************}
(515) Procedure EditGeneralParams;
(516) Var
(517)   S     : String;
(518)   Error : Integer;
(519) Begin
(520)     DrawBox(5,12,49,22,15,0);
(521)     WriteAt(20,12,Colors.HlBg,Colors.HlFg,' Do General Setup ');
(522)     WriteAt(7,14,Colors.TopicBg,Colors.TopicFg,'Com Port To Be Used   (port) (1..4)');
(523)     WriteAt(7,16,Colors.TopicBg,Colors.TopicFg,'Polling Time Interval  (min) (1..4)');
(524)     WriteAt(7,17,Colors.TopicFg,Colors.TopicBg,'1=15   2=30   3=60   4=120');
(525)     WriteAt(7,18,Colors.TopicBg,Colors.TopicFg,'Database Exp Interval (hour) (1..4)');
(526)     WriteAt(7,19,Colors.TopicFg,Colors.TopicBg,'1=24   2=36   3=48   4=72');
(527)     WriteAt(7,21,Colors.TxtFg,Colors.TxtBg,'F2 when done editing ...');

(528)     EditSave := False;
(529)     EditFin  := False;
(530)     SetTxtColor(LightRed,Black);
(531)     RestCursor;
```

Bylaag B.1

MAINPROG.PAS

```
(532)      Repeat
(533)            If (Not EditSave) and (Not EditFin) then
(534)            Begin
(535)                 Str(SetupRec.Comport,S);
(536)                 EditLine(S,45,14,1,['1'..'4'],[Enter,CurDown,Esc,F2]);
(537)            End;
(538)            If (Not EditSave) and (Not EditFin) then
(539)            Begin
(540)                 Val(S,SetupRec.Comport,Error);
(541)                 If (Not ComportExist(SetupRec.Comport)) then
(542)                 Begin
(543)                      ErrorMsg('WARNING : COM PORT SPECIFIED DOES NOT EXIST !!   PLEASE CORRECT ENTRY.');
(544)                      SetupRec.Comport := 1;
(545)                 End;
(546)                 Str(SetupRec.Poll,S);
(547)                 EditLine(S,45,16,1,['1'..'4'],[Enter,CurDown,Esc,F2]);
(548)            End;
(549)            If (Not EditSave) and (Not EditFin) then
(550)            Begin
(551)                 Val(S,SetupRec.Poll,Error);
(552)                 Str(SetupRec.Hold,S);
(553)                 EditLine(S,45,18,1,['1'..'5'],[Enter,CurDown,Esc,F2]);
(554)            End;
(555)            If (Not EditSave) and (Not EditFin) then Val(S,SetupRec.Hold,Error);
(556)      Until EditFin or EditSave;
(557)      SetTxtColor(15,0);
(558)      HideCursor(14,0);

(559)      If EditSave then
(560)      Begin
(561)           Rewrite(SetupFile);
(562)           Write(SetupFile,SetupRec);
(563)           Close(SetupFile);
(564)           Window(1,1,80,25);
(565)           CurrPollTime := PollTime(SystemTime);
(566)           RestoreScreen;
(567)           Dispmessage(True,13,15,0,'RE-CONFIGURING POLLING TIMES ...');
(568)           WriteCenter(2,15,0,'Next Poll -> '+CurrPollTime);
(569)           ReConfig := True;
(570)      End;
(571) End;

(572) {*********************************}
(573) {General setup submenu topic select}
(574) {*********************************}
(575) Procedure GenSetupChoose;
(576) Var
(577)   GenSetup : Char;
```

Bylaag B.1

MAINPROG.PAS

```
(578)    Fin  : Boolean;

(579)    Procedure GenSetupHilight;                {Menu topic highlight}
(580)    Begin
(581)         Window(50,GenSetup1+10,50+Length(GenSetupMenu[GenSetup1]),GenSetup1+10);
(582)         SetTxtColor(Colors.TopicFg,Colors.TopicBg);
(583)         Clrscr;
(584)         Write(GenSetupMenu[GenSetup1]);
(585)         Window(50,GenSetup2+10,50+Length(GenSetupMenu[GenSetup2]),GenSetup2+10);
(586)         SetTxtColor(Colors.HlBg,Colors.HlFg);
(587)         Clrscr;
(588)         Write(GenSetupMenu[GenSetup2]);
(589)         SetTxtColor(15,0);
(590)    End;

(591)  Begin                                       {Select topic}
(592)      Fin := False;
(593)      GenSetupHilight;
(594)      Repeat
(595)            GenSetup1 := GenSetup2;
(596)            GenSetup := GetkeyInput;
(597)            Case GenSetup of
(598)                 Enter,MouseEnter  : Case GenSetup2 of
(599)                                       0 : ViewGeneralParams;
(600)                                       1 : EditGeneralParams;
(601)                                     End;
(602)                 CurDown,MouseDown : Inc(GenSetup2);
(603)                 CurUp,MouseUp     : Dec(GenSetup2);
(604)                 Esc,MouseEsc      : Fin := True;
(605)            End; {Case GenSetup}
(606)            If (GenSetup2) = -1 then GenSetup2 := 1
(607)            Else
(608)            If (GenSetup2 = 2) then GenSetup2 := 0;
(609)            If (Not Fin) and
(610)               (GenSetup <> Enter) and
(611)               (GenSetup <> MouseEnter) then GenSetupHilight
(612)            Else
(613)            If Not ReConfig then
(614)            Begin
(615)                 Window(3,5,78,21);
(616)                 SetTxtColor(15,0);
(617)                 Clrscr;
(618)            End;

(619)      Until Fin or (GenSetup = Enter) or (GenSetup = MouseEnter);
(620)  End;
```

Bylaag B.1

MAINPROG.PAS

```
(621) {*********************************************}
(622) {Display general setup menu topics & select topic}
(623) {*********************************************}
(624) Procedure DoGenSetupMenu;
(625) Begin
(626)     DrawBox(49,9,56,12,15,0);
(627)     For I := 0 to 1 do WriteAt(50,I+10,Colors.TopicFg,Colors.TopicBg,GenSetupMenu[I]);
(628)     GenSetupChoose;
(629) End;

(630) {*******************************}
(631) {View the node addresses specified}
(632) {*******************************}
(633) Procedure ViewNodeAddr;
(634) Var
(635)   Wait : Char;
(636)   I,J  : Byte;
(637) Begin
(638)     If NodeAddrFileExist  then
(639)     Begin
(640)          Drawbox(13,9,48,21,15,0);
(641)          WriteAt(18,9,Colors.HlBg,Colors.HlFg,' View Node Address Config ');
(642)          WriteAt(17,20,Colors.TxtFg,Colors.TxtBg,'Press [Esc] to quit ...');
(643)          For I := 0 to 7 do
(644)          Begin
(645)               WriteAt(16,I+11,15,0,Nodes[0,I]);
(646)               WriteAt(26,I+11,Colors.HlBg,Colors.HlFg,NaddrRec.NodeArr[0,I]);
(647)               WriteAt(32,I+11,15,0,Nodes[1,I]);
(648)               WriteAt(42,I+11,Colors.HlBg,Colors.HlFg,NaddrRec.NodeArr[1,I]);
(649)          End;

(650)          Repeat
(651)                Wait := GetkeyInput;
(652)          Until (Wait = Esc);
(653)     End
(654)     Else ErrorMsg('NODE ADDRESS FILE DOESN''T EXIST YET !!');
(655) End;

(656) {*************************************}
(657) {Edit the unique addresses of the nodes}
(658) {*************************************}
(659) Procedure EditNodeAddr;
(660) Const
(661)     Sarr : Array[0..1,0..7] of String[2] =
(662)            (('00','01','02','03','04','05','06','07'),
(663)             ('08','09','0A','0B','0C','0D','0E','0F'));
(664) Var
(665)   S    : String;
```

Bylaag B.1

MAINPROG.PAS

```
(666)    I,J : Byte;
(667)  Begin
(668)      Drawbox(15,10,47,21,15,0);
(669)      WriteAt(20,10,Colors.HlBg,Colors.HlFg,' Setup Node Addresses ');
(670)      WriteAt(17,20,Colors.TxtFg,Colors.TxtBg,'F2 when done editing ...');
(671)      For I := 0 to 7 do
(672)      Begin
(673)           WriteAt(16,I+11,15,0,Nodes[0,I]);
(674)           WriteAt(32,I+11,15,0,Nodes[1,I]);
(675)      End;
(676)      EditFin := False;
(677)      EditSave := False;
(678)      SetTxtColor(LightRed,White);
(679)      RestCursor;
(680)      If (Not NodeAddrFileExist) then
(681)         For J := 0 to 1 do
(682)         For I := 0 to 7 do NaddrRec.NodeArr[J,I] := Sarr[J,I];

(683)      Repeat
(684)           For J := 0 to 1 do
(685)           For I := 0 to 7 do
(686)           If (Not EditFin) and (Not EditSave) then
(687)           Begin
(688)                S := NaddrRec.NodeArr[J,I];
(689)                EditLine(S,26+(J*16),I+11,2,['0'..'9','A'..'F'],[Enter,CurDown,Esc,F2]);
(690)                If (Not EditFin) then NaddrRec.NodeArr[J,I] := S;
(691)           End;
(692)      Until EditSave or EditFin;

(693)      HideCursor(14,0);
(694)      SetTxtColor(15,0);
(695)      If EditSave then
(696)      Begin
(697)           WriteAt(17,20,15,0,'Updating node address data ...');
(698)           Rewrite(NaddrFile);
(699)           Write(NaddrFile,NaddrRec);
(700)           Close(NaddrFile);
(701)      End;
(702)  End;

(703)  {*************************************}
(704)  {Node address menu topic select procedure}
(705)  {*************************************}
(706)  Procedure NodeAddrChoose;
(707)  Var
(708)    NodeAddr : Char;
(709)    Fin  : Boolean;
```

Bylaag B.1

MAINPROG.PAS

```
(710)    Procedure NodeAddrHilight;                {Menu topic highlight}
(711)    Begin
(712)         Window(50,NodeAddr1+10,50+Length(NodeAddrMenu[NodeAddr1]),NodeAddr1+10);
(713)         SetTxtColor(Colors.TopicFg,Colors.TopicBg);
(714)         Clrscr;
(715)         Write(NodeAddrMenu[NodeAddr1]);
(716)         Window(50,NodeAddr2+10,50+Length(NodeAddrMenu[NodeAddr2]),NodeAddr2+10);
(717)         SetTxtColor(Colors.HlBg,Colors.HlFg);
(718)         Clrscr;
(719)         Write(NodeAddrMenu[NodeAddr2]);
(720)         SetTxtColor(15,0);
(721)    End;

(722)  Begin                                       {Select menu topic}
(723)      Fin := False;
(724)      NodeAddrHilight;
(725)      Repeat
(726)            NodeAddr1 := NodeAddr2;
(727)            NodeAddr := GetkeyInput;
(728)            Case NodeAddr of
(729)                 Enter,MouseEnter  : Case NodeAddr2 of
(730)                                     {View}  0 : ViewNodeAddr;
(731)                                     {Edit}  1 : EditNodeAddr;
(732)                                       End;
(733)                 CurDown,MouseDown : Inc(NodeAddr2);
(734)                 CurUp,MouseUp     : Dec(NodeAddr2);
(735)                 Esc,MouseEsc      : Fin := True;
(736)            End; {Case NodeAddr}
(737)            If (NodeAddr2) = -1 then NodeAddr2 := 1
(738)            Else
(739)            If (NodeAddr2 = 2) then NodeAddr2 := 0;
(740)            If (Not Fin) then NodeAddrHilight
(741)            Else
(742)            Begin
(743)                 Window(3,5,78,21);
(744)                 SetTxtColor(15,0);
(745)                 Clrscr;
(746)            End;

(747)      Until Fin or (NodeAddr = Enter) or (NodeAddr = MouseEnter);
(748)  End;

(749)  {*****************************************************}
(750)  {Display node address topic strings an select menu topic}
(751)  {*****************************************************}
(752)  Procedure DoNodeAddrMenu;
(753)  Begin
(754)      DrawBox(49,9,56,12,15,0);
```

Bylaag B.1

MAINPROG.PAS

```
(755)      For I := 0 to 1 do WriteAt(50,I+10,Colors.TopicFg,Colors.TopicBg,NodeAddrMenu[I]);
(756)      NodeAddrChoose;
(757)  End;

(758)  {*****************************************}
(759)  {Node port pins menu topic selection procedure}
(760)  {*****************************************}
(761)  Procedure NodeIdChoose;
(762)  Var
(763)    NodeId : Char;
(764)    Fin  : Boolean;


(765)    Procedure NodeIdHilight;                    {Node port pins topic highlight procedure}
(766)    Begin
(767)         Window(48,NodeId1+10,48+Length(NodeIdMenu[NodeId1]),NodeId1+10);
(768)         SetTxtColor(Colors.TopicFg,Colors.TopicBg);
(769)         Clrscr;
(770)         Write(NodeIdMenu[NodeId1]);
(771)         Window(48,NodeId2+10,48+Length(NodeIdMenu[NodeId2]),NodeId2+10);
(772)         SetTxtColor(Colors.HlBg,Colors.HlFg);
(773)         Clrscr;
(774)         Write(NodeIdMenu[NodeId2]);
(775)         SetTxtColor(15,0);
(776)    End;

(777)  Begin
(778)      Fin := False;
(779)      NodeIdHilight;
(780)      Repeat
(781)            NodeId1 := NodeId2;
(782)            NodeId := GetkeyInput;
(783)            Case NodeId of
(784)                 Enter,MouseEnter  : Begin
(785)                                          Window(1,1,80,25);
(786)                                          WriteCenter(13,15,0,'Loading ...');
(787)                                          Case NodeId2 of
(788)                                        {view}  0 : Exec('NODEUTIL.EXE','NODE8031 /V');
(789)                                        {edit}  1 : Exec('NODEUTIL.EXE','NODE8031 /C');
(790)                                      {print}   2 : Exec('NODEUTIL.EXE','NODE8031 /P');
(791)                                    {delete}    3 : Exec('NODEUTIL.EXE','NODE8031 /D');
(792)                                          End;
(793)                                     End;
(794)                 CurDown,MouseDown : Inc(NodeId2);
(795)                 CurUp,MouseUp     : Dec(NodeId2);
(796)                 Esc,MouseEsc      : Fin := True;
(797)            End; {Case NodeId}
(798)            If (NodeId2) = -1 then NodeId2 := 3
```

Bylaag B.1
MAINPROG.PAS

```
(799)            Else
(800)            If (NodeId2 = 4) then NodeId2 := 0;
(801)            If (Not Fin) then NodeIdHilight
(802)            Else
(803)            Begin
(804)                 Window(3,5,78,21);
(805)                 SetTxtColor(15,0);
(806)                 Clrscr;
(807)            End;

(808)       Until Fin or (NodeId = Enter) or (NodeId = MouseEnter);
(809)  End;

(810)  {*******************************************************}
(811)  {Display node port pins topic strings and select menu topic}
(812)  {*******************************************************}
(813)  Procedure DoNodeIdMenu;
(814)  Begin
(815)       DrawBox(47,9,56,14,15,0);
(816)       For I := 0 to 3 do WriteAt(48,I+10,Colors.TopicFg,Colors.TopicBg,NodeIdMenu[I]);
(817)       NodeIdChoose;
(818)  End;

(819)  {**********************}
(820)  {Select setup menu topic}
(821)  {**********************}
(822)  Procedure SetupChoose;
(823)  Var
(824)    Setup : Char;
(825)    Fin   : Boolean;

(826)    Procedure SetupHilight;                    {Setup menu topic highlight procedure}
(827)    Begin
(828)         Window(56,Setup1+6,56+Length(SetupMenu[Setup1]),Setup1+6);
(829)         SetTxtColor(Colors.TopicFg,Colors.TopicBg);
(830)         Clrscr;
(831)         Write(SetupMenu[Setup1]);
(832)         Window(56,Setup2+6,56+Length(SetupMenu[Setup2]),Setup2+6);
(833)         SetTxtColor(Colors.HlBg,Colors.HlFg);
(834)         Clrscr;
(835)         Write(SetupMenu[Setup2]);
(836)         SetTxtColor(15,0);
(837)    End;

(838)  Begin                                        {Setup menu topic select}
(839)       Fin := False;
(840)       SetupHilight;
(841)       Repeat
```

Bylaag B.1

MAINPROG.PAS

```
(842)            Setup1 := Setup2;
(843)            Setup := GetkeyInput;
(844)            Case Setup of
(845)                 Enter,MouseEnter  : Case Setup2 of
(846)                                          0 : Begin
(847)                                                 SaveScreen;
(848)                                     {Gen setup menu} DoGenSetupMenu;
(849)                                                 If Not ReConfig then RestoreScreen
(850)                                                 Else  ReConfig := False;
(851)                                              End;
(852)                                          1 : Begin
(853)                                                 SaveScreen;
(854)                                     {Node addr menu} DoNodeAddrMenu;
(855)                                                 RestoreScreen;
(856)                                              End;
(857)                                          2 : Begin
(858)                                                 SaveScreen;
(859)                                     {Node id menu}   DoNodeIdMenu;
(860)                                                 RestoreScreen;
(861)                                              End;
(862)                                    End;
(863)                 CurDown,MouseDown : Inc(Setup2);
(864)                 CurUp,MouseUp     : Dec(Setup2);
(865)                 Esc,MouseEsc      : Fin := True;
(866)                 CurLeft,MouseLeft : Begin
(867)                                          Dec(Main2);
(868)                                          Fin := True;
(869)                                          SetEnterResident;
(870)                                    End;
(871)                 CurRight,MouseRight : Begin
(872)                                           Inc(Main2);
(873)                                           Fin := True;
(874)                                           SetEnterResident;
(875)                                       End;
(876)            End; {Case Setup}
(877)            If (Setup2) = -1 then Setup2 := 2
(878)            Else
(879)            If (Setup2 = 3) then Setup2 := 0;
(880)            If (Not Fin) then SetupHilight
(881)            Else
(882)            Begin
(883)                 Window(3,5,78,21);
(884)                 SetTxtColor(15,0);
(885)                 Clrscr;
(886)            End;

(887)       Until Fin;
(888)  End;
```

Bylaag B.1

MAINPROG.PAS

```
(889) {***********************************************}
(890) {Display utility menu topic strings & select topic}
(891) {***********************************************}
(892) Procedure DoUtilMenu;
(893) Begin
(894)      DrawBox(12,5,31,9,15,0);
(895)      For I := 0 to 2 do WriteAt(13,I+6,Colors.TopicFg,Colors.TopicBg,UtilMenu[I]);
(896)      UtilChoose;
(897) End;

(898) {***********************************************}
(899) {Display setup menu topic strings & select topic}
(900) {***********************************************}
(901) Procedure DoSetupMenu;
(902) Begin
(903)      DrawBox(55,5,73,9,15,0);
(904)      For I := 0 to 2 do WriteAt(56,I+6,Colors.TopicFg,Colors.TopicBg,SetupMenu[I]);
(905)      SetupChoose;
(906) End;

(907) {*********************************}
(908) {Main menu topic selection procedure}
(909) {*********************************}
(910) Procedure MainChoose;
(911) Var
(912)   Fin    : Boolean;
(913)   MainCh : Char;

(914)   Procedure MainHilight;                    {Main menu topic highlight procedure}
(915)   Begin
(916)        Window((40*Main1)+17,2,(40*Main1)+17+(Length(MainMenu[Main1])),2);
(917)        SetTxtColor(Colors.TopicFg,Colors.TopicBg);
(918)        Clrscr;
(919)        Write(MainMenu[Main1]);
(920)        Window((40*Main2)+17,2,(40*Main2)+17+(Length(MainMenu[Main2])),2);
(921)        SetTxtColor(Colors.HlBg,Colors.HlFg);
(922)        Clrscr;
(923)        Write(MainMenu[Main2]);
(924)        SetTxtColor(15,0);
(925)   End;

(926) Begin                                       {Main menu topic select}
(927)      Fin := False;
(928)      SetEnterResident;
(929)      Repeat
(930)           Main1 := Main2;
(931)           MainCh := GetKeyInput;
(932)           Case MainCh of
```

Bylaag B.1
MAINPROG.PAS

```
(933)                Enter,MouseEnter : Case Main2 of
(934)                {Utility menu}      0 : DoUtilMenu;
(935)                {Setup menu}        1 : DoSetupMenu;
(936)                                   End;
(937)                CurRight,MouseRight : Begin
(938)                                           Inc(Main2);
(939)                                           SetEnterResident;
(940)                                      End;
(941)                CurLeft,MouseLeft   : Begin
(942)                                           Dec(Main2);
(943)                                           SetEnterResident;
(944)                                      End;

(945)                Esc,MouseEsc : Begin
(946)                                     Fin := True;
(947)                                     Window(1,1,80,25);
(948)                                     Clrscr;
(949)                                     SetTxtColor(15,0);
(950)                               End;
(951)           End; {Case MainCh}

(952)           If (Main2 = 2) then Main2 := 0;
(953)           If (Main2 = -1) then Main2 := 1;
(954)           If (Not Fin) then MainHilight;
(955)     Until Fin;
(956) End;

(957) {*****************************************}
(958) {To set stolen int vector to previous pointer}
(959) {NOTE: This is a macro and NOT a procedure!! }
(960) {*****************************************}
(961) Procedure JmpOldISR(OldISR : Pointer);
(962) Inline($5B/$58/$87/$5E/$0E/$87/$46/$10/$89/
(963)       $EC/$5D/$07/$1F/$5F/$5E/$5A/$59/$CB);

(964) {$F+}
(965) {*****************************************************}
(966) {DOS_SAVE_TO_USE interrupt procedure -> hooked to INT28H}
(967) {*****************************************************}
(968) Procedure Dos_Safe_To_Use;
(969) INTERRUPT;
(970) Var
(971)   CurX1,CurX2,                              {Current screen attributes}
(972)   CurY1,CurY2,
(973)   CurX,CurY    : Byte;
(974)   CurTextAttr  : Word;
(975) Begin
(976)     Inline($FA);                           {CLI}
```

Bylaag B.1

MAINPROG.PAS

```
(977)     If (Not Handler_Active) then Handler_Active := True;

(978)     If (Handler_Active) then
(979)     Begin
(980)          Handler_Active := False;

(981)          Inc(Int28Count);                    {Interrupt occurred + 1}

(982)          If (Int28Count = 200) then          {Update time displayed on screen}
(983)          Begin
(984)               Curtime := Systemtime;
(985)               CurSec  := Copy(Curtime,7,2);
(986)               Int28count := 0;

(987)               If (CurSec <> PrevSec) then  {If 1 second passed -> check poll time}
(988)               Begin
(989)                    PrevSec := CurSec;
(990)                    CurX1 := Lo(WindMin)+1; {Save screen attributes}
(991)                    CurY1 := Hi(WindMin)+1;
(992)                    CurX2 := Lo(WindMax)+1;
(993)                    CurY2 := Hi(WindMax)+1;
(994)                    CurX  := WhereX;
(995)                    CurY  := WhereY;
(996)                    CurTextAttr := TextAttr;
(997)                    Window(1,1,80,25);
(998)                    Writecenter(3,15,0,' '+Curtime+' ');
(999)                    WriteCenter(2,15,0,'Next Poll -> '+CurrPollTime);
(1000)                        PrevSec := CurSec;
(1001)                                          {Restore screen attributes}
(1002)                        Window(CurX1,CurY1,CurX2,CurY2);
(1003)                        GotoXY(CurX,CurY);
(1004)                        TextAttr := CurTextAttr;
(1005)                                          {If current time = poll time => poll nodes for results}
(1006)                        If (Copy(Curtime,1,5) = CurrPollTime) and (Copy(Curtime,7,2) = '00') then
(1007)                        If NodeAddrFileExist then PollNodes
(1008)                        Else
(1009)                        ErrorMsg('CANNOT POLL NODES WITHOUT NODE ADDRESS DATABASE !!');
(1010)                   End;
(1011)              End;
(1012)         End;
(1013)         Inline($FB);                         {STI}
(1014)         JmpOldISR(Int28Save);                {Pass int on}
(1015)    End;
```

Bylaag B.1

MAINPROG.PAS

```
(1016)      {*****************************}
(1017)      {Interrupt 28H exit procedure }
(1018)      {*****************************}
(1019)      Procedure Int28ExitProc;
(1020)      Begin
(1021)          ExitProc := OldExitProc;
(1022)          SetIntVec($28,Int28Save);
(1023)      End;
(1024)      {$F-}

(1025)      {************************************************}
(1026)      {Set INT28H int vector to DOS OK TO USE int handler}
(1027)      {************************************************}
(1028)      Procedure Set_Poll_Inthandler;
(1029)      Begin
(1030)          Int28Count := 0;                    {Interrupt occurred = 0}
(1031)          Handler_Active := False;            {Handler not active}
(1032)          OldExitProc := ExitProc;
(1033)          ExitProc := @Int28ExitProc;         {Set exit proc vector}
(1034)          CurrPollTime := PollTime(SystemTime); {Determine current poll time}
(1035)          GetIntvec($28,Int28Save);           {Get current int vector & save it}
(1036)          SetIntVec($28,@Dos_Safe_To_Use);    {Hook INT28H int vector to new int handler}
(1037)      End;


(1038)      {******************************************************}
(1039)      {Calculate the minutes between the results captured and }
(1040)      {the current time and date of the main computer and     }
(1041)      {return the expiration status of the data.              }
(1042)      {******************************************************}
(1043)      Function DataExpires(Date1,Time1,Date2,Time2 : String) : Boolean;
(1044)      Var
(1045)        Days          : Array[1..12] of Byte;
(1046)        ExpArr        : Array[1..3] of Byte;
(1047)        Error         : Integer;
(1048)        ExpiredMinutes,
(1049)        ExpInt        : LongInt;


(1050)        Year1,   Year2,
(1051)        Month1,  Month2,
(1052)        MonthDiv,
(1053)        Day1,    Day2,
(1054)        Hour1,   Hour2,
(1055)        Min1,    Min2,
(1056)        HrDiv,   MinDiv : Integer;
```

Bylaag B.1

MAINPROG.PAS

```
(1057)      Begin
(1058)          If (Date1 > Date2) then
(1059)          Begin
(1060)               Val(Copy(Time1,1,2),Hour1,Error);
(1061)               Val(Copy(Time1,4,2),Min1,Error);
(1062)               Val(Copy(Time2,1,2),Hour2,Error);
(1063)               Val(Copy(Time2,4,2),Min2,Error);

(1064)               Val(Copy(Date1,1,2),Year1,Error);
(1065)               Val(Copy(Date1,4,2),Month1,Error);
(1066)               Val(Copy(Date1,7,2),Day1,Error);


(1067)               Val(Copy(Date2,1,2),Year2,Error);
(1068)               Val(Copy(Date2,4,2),Month2,Error);
(1069)               Val(Copy(Date2,7,2),Day2,Error);

(1070)               For I := 1 to 12 do
(1071)               Case I of
                (1072)    : Days[I] := 31;
                (1073)         : Days[I] := 30;
                (1074)              : If ((Year2 mod 4) = 0) then Days[I] := 29
(1075)                                   Else Days[I] := 28;
(1076)               End; {Case}

(1077)               If (Year1 = Year2) then
(1078)               Begin
(1079)                    If (Month2 = Month1) then
(1080)                    Begin
(1081)                         ExpArr[1] := (Day1-Day2);

(1082)                         HrDiv := (Hour1-Hour2);
(1083)                         MinDiv := (Min1-Min2);
(1084)                         If (HrDiv >= 0) then ExpArr[2] := HrDiv
(1085)                         Else
(1086)                         Begin
(1087)                              Dec(ExpArr[1]);
(1088)                              If (ExpArr[1] = 255) then ExpArr[1] := 0;
(1089)                              ExpArr[2] := (24+HrDiv);
(1090)                         End;
(1091)                         If (MinDiv >= 0) then ExpArr[3] := MinDiv
(1092)                         Else
(1093)                         Begin
(1094)                              Dec(ExpArr[2]);
(1095)                              If (ExpArr[2] = 255) then
(1096)                              Begin
(1097)                                   ExpArr[2] := 23;
(1098)                                   Dec(ExpArr[1]);
```

Bylaag B.1
MAINPROG.PAS

```
(1099)                              End;
(1100)                              ExpArr[3] := (60+MinDiv);
(1101)                         End;
(1102)                    End
(1103)                    Else
(1104)                    Begin
(1105)                         Monthdiv := (Month1-Month2);
(1106)                         If ((Day1-Day2) < 0) then Dec(Monthdiv);
(1107)                         ExpArr[1] := (Days[Month2]-Day2);
(1108)                         For I := 1 to (Monthdiv-1) do Inc(ExpArr[1],Days[I+Month2]);
(1109)                         Inc(ExpArr[1],Day1+1);

(1110)                         HrDiv := (Hour1-Hour2);
(1111)                         MinDiv := (Min1-Min2);
(1112)                         If (HrDiv < 0) then
(1113)                         Begin
(1114)                              Dec(ExpArr[1]);
(1115)                              ExpArr[2] := (24+HrDiv);
(1116)                         End
(1117)                         Else ExpArr[2] := HrDiv;
(1118)                         If (MinDiv < 0) then
(1119)                         Begin
(1120)                              Dec(ExpArr[2]);
(1121)                              If (ExpArr[2] = 255) then
(1122)                              Begin
(1123)                                   ExpArr[2] := 23;
(1124)                                   Dec(ExpArr[1]);
(1125)                              End;
(1126)                              ExpArr[3] := (60+MinDiv);
(1127)                         End
(1128)                         Else ExpArr[3] := MinDiv;
(1129)                    End;
(1130)               End
(1131)               Else
(1132)               Begin
(1133)                    ExpArr[1] := (Days[Month2]-Day2);
(1134)                    For I := (Month2+1) to 12 do Inc(ExpArr[1],Days[I]);
(1135)                    If (Year1 mod 4) = 0 then Days[2] := 29
(1136)                    Else Days[2] := 28;
(1137)                    For I := 1 to (Month1-1) do Inc(ExpArr[1],Days[I]);
(1138)                    Inc(ExpArr[1],Day1);
(1139)                    HrDiv := (Hour1-Hour2);
(1140)                    MinDiv := (Min1-Min2);
(1141)                    If (HrDiv < 0) then
(1142)                    Begin
(1143)                         Dec(ExpArr[1]);
(1144)                         ExpArr[2] := (24+HrDiv);
(1145)                    End
```

Bylaag B.1
MAINPROG.PAS

```
(1146)                    Else ExpArr[2] := HrDiv;
(1147)                    If (MinDiv < 0) then
(1148)                    Begin
(1149)                         Dec(ExpArr[2]);
(1150)                         If (ExpArr[2] = 255) then
(1151)                         Begin
(1152)                              ExpArr[2] := 23;
(1153)                              Dec(ExpArr[1]);
(1154)                         End;
(1155)                         ExpArr[3] := (60+MinDiv);
(1156)                    End
(1157)                    Else ExpArr[3] := MinDiv;
(1158)               End;

(1159)               {minutes for each interval}
(1160)               Case SetupRec.Hold of
(1161)                     1 : ExpInt := 1440;   {24 hours}
(1162)                     2 : ExpInt := 2160;   {36 hours}
(1163)                     3 : ExpInt := 2880;   {48 hours}
(1164)                     4 : ExpInt := 4320;   {72 hours}
(1165)               End;
(1166)                                  {days*24h*60m     hours*60m      minutes}
(1167)               ExpiredMinutes := (ExpArr[1]*24*60)+(ExpArr[2]*60)+(ExpArr[3]);

(1168)               If (ExpiredMinutes >= ExpInt) then DataExpires := True
(1169)               Else DataExpires := False;
(1170)          End
(1171)          Else DataExpires := False;
(1172)     End;

(1173)     {*********************************************************************}
(1174)     {Check result database for results older than expiration interval specified}
(1175)     {*********************************************************************}
(1176)     Procedure CheckDbaseExpDates;
(1177)     Var
(1178)       ResBupF    : File of XtractRecord;
(1179)       NodeCount1,
(1180)       NodeCount2 : Byte;
(1181)       ExpireFlag : Boolean;
(1182)     Begin
(1183)          DrawBox(10,10,69,15,Colors.HlFg,0);
(1184)          WriteCenter(10,Colors.TxtFg,Colors.TxtBg,' CHECKING RESULT DATABASE FOR EXPIRED DATA ');
(1185)          For NodeCount1 := 0 to 1 do
(1186)               For NodeCount2 := 0 to 7 do
(1187)               If NodeDbaseExist(Nodes[NodeCount1,NodeCount2]) then
(1188)               If ResFileExist(Nodes[Nodecount1,NodeCount2]) then
(1189)               Begin
(1190)                    ClearWindow(11,11,68,12);
```

Bylaag B.1
MAINPROG.PAS

```
(1191)                  Window(1,1,80,25);
(1192)                  WriteAt(12,11,15,0,'Current node ->'+Nodes[NodeCount1,NodeCount2]);
(1193)                  WriteAt(12,12,15,0,'Status        : Scanning for expired data.');
(1194)                  Reset(ResFile);
(1195)                  ExpireFlag := False;
(1196)                  While (Not Eof(ResFile)) do
(1197)                  Begin
(1198)                       Read(ResFile,ResRec);
(1199)                       With ResRec do
(1200)                       Begin
(1201)                            WriteAt(12,13,15,0,'Result date   : '+Date);
(1202)                            WriteAt(12,14,15,0,'Result time   : '+Time);
(1203)                                               {If data expires => delete it}
(1204)                            If DataExpires(SystemDate,SystemTime,Date,Time) then
(1205)                            Begin
(1206)                                 Seek(ResFile,FilePos(ResFile)-1);
(1207)                                 Index := '#';
(1208)                                 Write(Resfile,ResRec);
(1209)                                 If (Not ExpireFlag) then Expireflag := True;
(1210)                            End;
(1211)                       End;
(1212)                  End;
(1213)                  Close(ResFile);
(1214)                  If ExpireFlag then           {Delete expired data}
(1215)                  Begin
(1216)                       WriteAt(12,12,15,0,'Status        : Deleting expired results.');
(1217)                       Assign(ResBupF,Copy(ResFileName,1,7)+'.BAK');
(1218)                       Rewrite(ResBupF);
(1219)                       Reset(ResFile);
(1220)                       While (Not Eof(ResFile)) do
(1221)                       Begin
(1222)                            Read(ResFile,ResRec);
(1223)                            If (ResRec.Index <> '#') then Write(ResBupF,ResRec);
(1224)                       End;
(1225)                       Close(ResFile);
(1226)                       Erase(ResFile);
(1227)                       If (FileSize(ResBupF) > 0) then
(1228)                       Begin
(1229)                            Close(ResBupF);
(1230)                            Rename(ResBupF,ResFileName);
(1231)                       End
(1232)                       Else
(1233)                       Begin
(1234)                            Close(ResBupF);
(1235)                            Erase(ResBupF);
(1236)                       End;
(1237)                  End;
(1238)             End;
```

Bylaag B.1

MAINPROG.PAS

```
(1239)          Delay(1000);
(1240)          ClearWindow(10,10,69,15);
(1241)      End;

(1242)      {*********************************************************}
(1243)      {Display MAIN menu screen and topic strings & select topics}
(1244)      {*********************************************************}
(1245)      Procedure DoMainMenu;
(1246)      Begin
(1247)          DrawBox(2,1,79,3,15,0);
(1248)          DrawBox(2,4,79,22,15,0);
(1249)          DrawBox(2,23,79,25,15,0);
(1250)          WriteCenter(1,Colors.TxtFg,Colors.TxtBg,' 8031 NODE COMMUNICATION MANAGEMENT SYSTEM ');
(1251)          WriteCenter(23,Colors.TxtFg,Colors.TxtBg,' Author : Wickus de Koker - Technikon OFS ');
(1252)          WriteCenter(25,Colors.TxtFg,Colors.TxtBg,' Error Window ');

(1253)          For I := 0 to 1 do WriteAt(40*I+17,2,Colors.TopicFg,Colors.TopicBg,MainMenu[I]);
(1254)          CheckDbaseExpDates;
(1255)          Set_Poll_Inthandler;                    {INSTALL THE INTERRUPT HANDLER TO DO POLLING ETC.}
(1256)          MainChoose;
(1257)      End;

(1258)      {*************************************************************}
(1259)      {Initialization routine for menu topic pointers & misc variables}
(1260)      {*************************************************************}
(1261)      Procedure Init;
(1262)      Begin
(1263)          Main1     := 0; Main2     := 0;
(1264)          Util1     := 0; Util2     := 0;
(1265)          Setup1    := 0; Setup2    := 0;
(1266)          NodeId1   := 0; NodeId2   := 0;
(1267)          GenSetup1 := 0; GenSetup2 := 0;
(1268)          NodeAddr1 := 0; NodeAddr2 := 0;
(1269)          Dbase1    := 0; Dbase2    := 0;

(1270)          ReConfig := False;
(1271)          PrevSec := '';
(1272)          CurTime := '';

(1273)          If Not SetupFileExist then              {No setup file ? => Set default values}
(1274)          Begin
(1275)               SetupRec.Comport := 1;
(1276)               SetupRec.Poll := 1;
(1277)               SetupRec.Hold := 1;
(1278)               SetupRec.BaudRate := 1200;
(1279)               Rewrite(SetupFile);
(1280)               Write(SetupFile,SetupRec);
(1281)               Close(SetupFile);
```

Bylaag B.1
MAINPROG.PAS

```
(1282)          End;

(1283)          HideCursor(14,0);
(1284)          Window(1,1,80,25);
(1285)          SetTxtColor(15,0);
(1286)          Clrscr;
(1287)          DoMainMenu;                          {Start program}

(1288)          RestCursor;                          {Put cursor back on screen at program termination}
(1289)      End;


(1290)      {******************************}
(1291)      {     START OF MAIN PROGRAM    }
(1292)      {******************************}
(1293)      Begin                                     {Do initialization at startup}
(1294)          Init
(1295)      End.
```

Bylaag B.2

NODEUTIL.PAS

```
(1)   {*****************************************}
(2)   {PROGRAM: NODE UTILITY MANAGER              }
(3)   {PURPOSE: Performs all node related functions.}
(4)   {*****************************************}

(5)   {$M 16384,0,16384}                          {STACK=16K, HEAP=16K -> MIN=0K}
(6)   Program xNodeUtil;
(7)   Uses Crt,Dos,ScrnUtil,Getkey,Editln,Printer,Com_Unit,ComnDecl;
(8)   Const
(9)       Sfile = 'SORTFILE.FIL';                 {Sortfile name}
                                          (10)        PPI pin description strings}
(11)      Pin8255   : Array[0..2,0..7] of String[9] =
(12)                  ((' Port A.0',' Port A.1',' Port A.2',' Port A.3',
(13)                    ' Port A.4',' Port A.5',' Port A.6',' Port A.7'),
(14)                   (' Port B.0',' Port B.1',' Port B.2',' Port B.3',
(15)                    ' Port B.4',' Port B.5',' Port B.6',' Port B.7'),
(16)                   (' Port C.0',' Port C.1',' Port C.2',' Port C.3',
(17)                    ' Port C.4',' Port C.5',' Port C.6',' Port C.7'));

(18)  Type
(19)     DtaPointr = ^SortRec;                    {Sort record memory pointer}

(20)     SortRec = Record                         {Sort record structure}
(21)                     S1    : String[9];      {String to sort with}
(22)                     Fpos  : Integer;        {Position in database}
(23)                     Next  : DtaPointr;      {Pointer to next record for linked list}
(24)               End;


(25)  Var
(26)    StartItem,                               {Linked list pointers}
(27)    NewItem,
(28)    OldItem,
(29)    CurItem ,
(30)    NextItem     : DtaPointr;
(31)    Sf           : File of NodePortConfig;   {Sort file}

(32)    Found        : Boolean;                  {Found pin record?}
(33)    FoundPtr     : LongInt;                  {Filepos of found pin record}

(34)    PinsA,PinsB,
(35)    PinsC,PinsD  : ShortInt;                 {Pointers to 8255 pin description strings}
```

Bylaag B.2

NODEUTIL.PAS

```
(36)  {*****************************}
(37)  {Exchange records in linked list}
(38)  {*****************************}
(39)  Procedure Exchange(RecA,RecB : DtaPointr);
(40)  Var
(41)    TempS1  : String[9];
(42)    TempPos : Integer;
(43)  Begin
(44)      With RecA^ do                          {RecA = RecB}
(45)      Begin
(46)           TempS1  := S1;
(47)           TempPos := Fpos;
(48)           S1      := RecB^.S1;
(49)           Fpos    := RecB^.Fpos;
(50)      End;

(51)      With RecB^ do                          {RecB = Old RecA}
(52)      Begin
(53)           S1   := TempS1;
(54)           Fpos := TempPos;
(55)      End;
(56)  End;

(57)  {*******************************}
(58)  {Sort pin config database to memory}
(59)  {*******************************}
(60)  Procedure SortToMemory(Fsize : Integer);
(61)  Var
(62)    I,J : Integer;
(63)  Begin
(64)      WriteAt(4,21,Colors.TxtFg,Colors.TxtBg,'Sorting the node id database ......   ');
(65)      Reset(NodeFile);
(66)      If (MemAvail < 100) then ErrorMsg('Not enough memory to perform SORT !!!')
(67)      Else
(68)      Begin
(69)           Read(NodeFile,NodeRec);
(70)           New(StartItem);
(71)           With StartItem^ do
(72)           Begin
(73)                Next := Nil;
(74)                StartItem^.S1 := NodeRec.Port;
(75)                Fpos := Filepos(NodeFile)-1;
(76)           End;
(77)           OldItem := StartItem;
(78)           While (Not Eof(NodeFile)) do
(79)           Begin
(80)                Read(NodeFile,NodeRec);
(81)                If (MemAvail < 100) then
```

Bylaag B.2

NODEUTIL.PAS

```
(36)  {*****************************}
(37)  {Exchange records in linked list}
(38)  {*****************************}
(39)  Procedure Exchange(RecA,RecB : DtaPointr);
(40)  Var
(41)    TempS1  : String[9];
(42)    TempPos : Integer;
(43)  Begin
(44)      With RecA^ do                          {RecA = RecB}
(45)      Begin
(46)           TempS1  := S1;
(47)           TempPos := Fpos;
(48)           S1      := RecB^.S1;
(49)           Fpos    := RecB^.Fpos;
(50)      End;

(51)      With RecB^ do                          {RecB = Old RecA}
(52)      Begin
(53)           S1   := TempS1;
(54)           Fpos := TempPos;
(55)      End;
(56)  End;

(57)  {*******************************}
(58)  {Sort pin config database to memory}
(59)  {*******************************}
(60)  Procedure SortToMemory(Fsize : Integer);
(61)  Var
(62)    I,J : Integer;
(63)  Begin
(64)      WriteAt(4,21,Colors.TxtFg,Colors.TxtBg,'Sorting the node id database ......   ');
(65)      Reset(NodeFile);
(66)      If (MemAvail < 100) then ErrorMsg('Not enough memory to perform SORT !!!')
(67)      Else
(68)      Begin
(69)           Read(NodeFile,NodeRec);
(70)           New(StartItem);
(71)           With StartItem^ do
(72)           Begin
(73)                Next := Nil;
(74)                StartItem^.S1 := NodeRec.Port;
(75)                Fpos := Filepos(NodeFile)-1;
(76)           End;
(77)           OldItem := StartItem;
(78)           While (Not Eof(NodeFile)) do
(79)           Begin
(80)                Read(NodeFile,NodeRec);
(81)                If (MemAvail < 100) then
```

Bylaag B.2
NODEUTIL.PAS

```
(82)                Begin
(83)                     ErrorMsg('Not enough memory to perform SORT !!!');
(84)                     Close(NodeFile);
(85)                     Exit;
(86)                End
(87)                Else
(88)                Begin
(89)                     New(NewItem);
(90)                     NewItem^.Next := OldItem^.Next;
(91)                     OldItem^.Next := NewItem;
(92)                     NewItem^.S1 := NodeRec.Port;
(93)                     NewItem^.Fpos := Filepos(NodeFile)-1;
(94)                     OldItem := NewItem;
(95)                End;
(96)           End;
(97)           Close(NodeFile);

(98)           For I := 1 to Fsize do
(99)           Begin
(100)               CurItem := StartItem;
(101)               NextItem := CurItem^.Next;
(102)               While (NextItem <> Nil) do
(103)               Begin
(104)                    If (NextItem^.S1 < CurItem^.S1) then Exchange(NextItem,CurItem);
(105)                       CurItem := CurItem^.Next;
(106)                       NextItem := CurItem^.Next;
(107)               End;
(108)          End;

(109)          CurItem := StartItem;
(110)          Assign(Sf,Sfile);
(111)          Rewrite(Sf);
(112)          Reset(NodeFile);
(113)          While (CurItem <> Nil) do
(114)          Begin
(115)               Seek(NodeFile,CurItem^.Fpos);
(116)               Read(NodeFile,NodeRec);
(117)               Write(Sf,NodeRec);
(118)               CurItem := CurItem^.Next;
(119)          End;
(120)          Close(NodeFile);
(121)          Erase(NodeFile);
(122)          Close(Sf);
(123)          Rename(Sf,NodeFileName);

(124)          Release(StartItem);
(125)     End;
(126) End;
```

Bylaag B.2

NODEUTIL.PAS

```
(127) {*****************************************}
(128) {Can sorting be done on pin config database?}
(129) {*****************************************}
(130) Procedure VerifySort;
(131) Var
(132)   Fsize : Integer;
(133) Begin
(134)     Reset(NodeFile);
(135)     Fsize := Filesize(NodeFile);
(136)     Close(NodeFile);
(137)                                          {Sort if database > 1 record}
(138)     If (Fsize > 1) then SortToMemory(Fsize);
(139) End;

(140) {*************************************************}
(141) {Menu topic highlight procedure for 8255 pin strings}
(142) {*************************************************}
(143) Procedure PortHilight;
(144) Begin
(145)     Window((10*PinsA)+3,5+PinsC,(10*PinsA)+3+Length(Pin8255[PinsA,PinsC]),5+PinsC);
(146)     SetTxtColor(Colors.TopicFg,Colors.TopicBg);
(147)     Clrscr;
(148)     Write(Pin8255[PinsA,PinsC]);
(149)     Window((10*PinsB)+3,5+PinsD,(10*PinsB)+3+Length(Pin8255[PinsB,PinsD]),5+PinsD);
(150)     SetTxtColor(Colors.HlBg,Colors.HlFg);
(151)     Clrscr;
(152)     Write(Pin8255[PinsB,PinsD]);
(153)     SetTxtColor(15,0);
(154) End;

(155) {*********************************}
(156) pin menu topic choose procedure}
(157) {*********************************}
(158) Function PortChoose : Boolean;
(159) Var
(160)   Fin,NoChoice      : Boolean;
(161)   PortCh    : Char;
(162)   PortStr : String[2];
(163)   Error     : Integer;
(164) Begin
(165)     Fin := False;
(166)     NoChoice := False;
(167)     PortHilight;
(168)     Repeat
(169)           PinsA := PinsB; PinsC := PinsD;
(170)           PortCh := GetkeyInput;
(171)           Case PortCh of
(172)                Enter,MouseEnter : ;
```

Bylaag B.2
NODEUTIL.PAS

```
(173)                CurRight,MouseRight : Inc(PinsB);
(174)                CurLeft,MouseLeft   : Dec(PinsB);
(175)                CurDown,MouseDown   : Inc(PinsD);
(176)                CurUp,MouseUp       : Dec(PinsD);
(177)                Esc,MouseEsc        : Begin
(178)                                        Fin := True;
(179)                                        NoChoice := True;
(180)                                      End;
(181)           End; {Case}

(182)           If (PinsB = 3) then PinsB := 0;
(183)           If (PinsB = -1) then PinsB := 2;
(184)           If (PinsD = 8) then PinsD := 0;
(185)           If (PinsD = -1) then PinsD := 7;
(186)           If (Not Fin) then PortHilight;
(187)     Until (PortCh = Enter) or (PortCh = MouseEnter) or (Fin);
(188)     PortChoose := (Not Fin);
(189) End;

(190) {*****************************}
(191) { Pin config data exist yet?    }
(192) {*****************************}
(193) Procedure GetDbaseData(NodeName,Pin : String);
(194) Var
(195)   I,J : Byte;
(196) Begin
(197)     Found := False;
(198)     If NodeDbaseExist(NodeName) then
(199)     Begin
(200)          Reset(NodeFile);
(201)          While (Not Found) and (Not Eof(NodeFile)) do
(202)          Begin
(203)               Read(NodeFile,NodeRec);
(204)               If (NodeRec.Port = Pin) then
(205)               Begin
(206)                    Found := True;
(207)                    FoundPtr := Filepos(NodeFile)-1;
(208)               End;
(209)          End;
(210)          Close(NodeFile);
(211)     End;
(212)                                          {Not existing? => set default pin data values}
(213)     If (Not Found) or (Not NodeDbaseExist(NodeName)) then
(214)     With NodeRec do
(215)     Begin
(216)          Port  := Pin8255[PinsB,PinsD];
(217)          Name  := 'NONE';
(218)          MemLoc := '****';
```

Bylaag B.2

NODEUTIL.PAS

```
(219)           For I := 1 to 2 do
(220)           For J := 1 to 5 do Times[I,J] := '**:**';
(221)       End;
(222)   End;

(223)   {**********************}
(224)   {Configure 8255 pin data}
(225)   {**********************}
(226)   Procedure ConfigurePins(NodeName : String);
(227)   Var
(228)     I          : Byte;
(229)     PortChosen : Boolean;
(230)     S          : String;
(231)     Verify     : Char;
(232)   Begin
(233)       PinsA := 0;  PinsB := 0;
(234)       PinsC := 0;  PinsD := 0;
(235)       Found := False;
(236)       Drawbox(2,4,33,15,15,0);
(237)       WriteAt(4,14,Colors.TxtFg,Colors.TxtBg,'Please choose 8255 pin ...');
(238)       For I := 0 to 7 do
(239)       Begin
(240)           WriteAt(3,I+5,Colors.TopicFg,Colors.TopicBg,Pin8255[0,I]);
(241)           WriteAt(13,I+5,Colors.TopicFg,Colors.TopicBg,Pin8255[1,I]);
(242)           WriteAt(23,I+5,Colors.TopicFg,Colors.TopicBg,Pin8255[2,I]);
(243)       End;
(244)       PortChosen := PortChoose;
(245)       If PortChosen then
(246)       Begin
(247)            RestCursor;
(248)            GetDbaseData(NodeName,Pin8255[PinsB,PinsD]);
(249)            Drawbox(2,4,50,22,15,0);
(250)            WriteAt(4,21,Colors.TxtFg,Colors.TxtBg,'[F2] = Save & Quit / [Esc] = Exit');
(251)            WriteAt(3,5,Colors.HlBg,Colors.HlFg,' Node ->'+NodeName+'    -:-   8255 pin ->'+NodeRec.Port+' ');
(252)            WriteAt(4,7,Colors.TopicFg,Colors.TopicBg,'Name of pin ? ');
(253)            WriteAt(4,8,Colors.TopicFg,Colors.TopicBg,'(I)nput/(O)utput pin ? ');
(254)            WriteAt(4,9,Colors.TopicFg,Colors.TopicBg,'Memory location in node (hex) ? ');
(255)            WriteAt(4,10,Colors.HlBg,Colors.HlFg,' On/Off times of pin (5 max) : ');
(256)            NodeRec.Index := '*';
(257)            Repeat                              {Input data from keyboard}
(258)                  SetTxtColor(Colors.HlBg,White);
(259)                  S := NodeRec.Name;
(260)                  EditLine(S,18,7,30,[#32..#126],[Enter,CurDown,Esc,F2]);
(261)                  If (Not EditFin) then
(262)                  Begin
(263)                       NodeRec.Name := S;
(264)                       S := 'O';
(265)                       If (Not EditSave) and (Not EditFin) then
```

Bylaag B.2
NODEUTIL.PAS

```
(266)                    Begin
(267)                         EditLine(S,27,8,1,['I','O'],[Enter,CurDown,Esc,F2]);
(268)                         If (Not EditFin) then
(269)                            If (S = 'I') then NodeRec.Ptype := Input
(270)                            Else NodeRec.Ptype := Output;
(271)                         If (Not EditFin) and (Not EditSave) then
(272)                         Begin
(273)                              S := NodeRec.MemLoc;
(274)                              EditLine(S,36,9,4,['0'..'9','A'..'F'],[Enter,CurDown,Esc,F2]);
(275)                              If (Not EditFin) or (Not EditSave) then NodeRec.MemLoc := S;
(276)                         End;
(277)                    End;
(278)                    SetTxtColor(White,Black);
(279)               End;

(280)               For I := 1 to 5 do
(281)               If (Not EditSave) and (Not EditFin) then
(282)               Begin
(283)                    WriteAt(5,I+10,Colors.TopicFg,Colors.TopicBg,'On :         Off :');
(284)                    SetTxtColor(Colors.HlBg,White);
(285)                    S := NodeRec.Times[1,I];
(286)                    EditLine(S,10,I+10,5,['0'..'9',':','*'],[Enter,CurDown,Esc,F2]);
(287)                    If (Not EditSave) and (Not EditFin) then
(288)                    Begin
(289)                         NodeRec.Times[1,I] := S;
(290)                         S := NodeRec.Times[2,I];
(291)                         EditLine(S,24,I+10,5,['0'..'9',':','*'],[Enter,CurDown,Esc,F2]);
(292)                         If (Not EditSave) and (Not EditFin) then NodeRec.Times[2,I] := S;
(293)                    End;
(294)                    SetTxtColor(15,0);
(295)               End;
(296)          Until EditFin or EditSave;
(297)          HideCursor(14,0);
(298)     End;
(299)     SetTxtColor(15,0);

(300)     If EditSave and PortChosen then
(301)     Begin
(302)          If Found then                         {Overwrite previous data?}
(303)          Begin
(304)               WriteAt(4,21,Colors.TxtFg,Colors.TxtBg,'Overwrite previous data (Y)es/(N)o ? ');
(305)               Sound(1000);
(306)               Delay(200);
(307)               Nosound;
(308)               Repeat
(309)                    Verify := Upcase(GetkeyInput);
(310)               Until (Verify = 'Y') or (Verify = 'N');
(311)               If (Verify = 'Y') then
```

Bylaag B.2
NODEUTIL.PAS

```
(312)                Begin
(313)                     Reset(NodeFile);
(314)                     Seek(NodeFile,FoundPtr);
(315)                     Write(NodeFile,NodeRec);
(316)                     Close(NodeFile);
(317)                End;
(318)           End
(319)           Else
(320)           Begin                                {Overwrite previous data}
(321)                WriteAt(4,21,Colors.TxtFg,Colors.TxtBg,'Updating node id database .........');
(322)                If (Not NodeDbaseExist(NodeName)) then Rewrite(NodeFile)
(323)                Else Reset(NodeFile);
(324)                Seek(NodeFile,Filesize(NodeFile));
(325)                Write(NodeFile,NodeRec);
(326)                Close(NodeFile);
(327)                VerifySort;
(328)           End;
(329)      End;
(330) End;

(331) {************************}
(332) {View 8255 pin config data}
(333) {************************}
(334) Procedure ViewPinConfig(NodeName : String);
(335) Var
(336)   I    : Byte;
(337)   Wait : Char;
(338) Begin
(339)      If (Not NodeDbaseExist(NodeName)) then ErrorMsg('DATABASE DOESN''T EXIST YET !!')
(340)      Else
(341)      Begin
(342)           Reset(NodeFile);
(343)           DrawBox(2,4,79,22,15,0);
(344)           WriteAt(3,5,Colors.HlBg,Colors.HlFg,
(345)                   ' 8255 PIN     DESCRIPTION                                PIN TYPE      MEM ADDRESS  ');
(346)           WriteAt(3,6,Colors.HlBg,Colors.HlFg,
(347)                   ' On times     T1        T2        T3        T4        T5                    ');
(348)           WriteAt(3,7,Colors.HlBg,Colors.HlFg,
(349)                   ' Off times    T1        T2        T3        T4        T5                    ');
(350)           WriteAt(3,21,Colors.HlBg,Colors.HlFg,
(351)                   ' File -> '+NodeFileName+'        <Esc> = Exit        PgDn/PgUp        Home/End  ');
(352)           Window(3,8,78,20);
(353)           While (Not Eof(NodeFile)) do
(354)           Begin
(355)                Read(NodeFile,NodeRec);
(356)                With NodeRec do
(357)                Begin
(358)                     SetTxtColor(Colors.TopicFg,Colors.TopicBg);
```

Bylaag B.2
NODEUTIL.PAS

```
(359)                   Write(Port,'     ',Name,'     ');
(360)                   If (Ptype = output) then Write('OUTPUT        ')
(361)                   Else Write('INPUT         ');
(362)                   Writeln(MemLoc);
(363)                   Write('         ');
(364)                   For I := 1 to 5 do Write('     ',times[1,i]);
(365)                   Writeln;
(366)                   Write('         ');
(367)                   For I := 1 to 5 do Write('     ',times[2,i]);
(368)                   Writeln;
(369)                   If ((Filepos(NodeFile) mod 4) = 0) or Eof(NodeFile) then
(370)                   Repeat
(371)                        If Eof(NodeFile) then WriteAt(1,WhereY,LightRed,Black,'<< End Of File >>');
(372)                        Wait := GetkeyInput;
(373)                        Case Wait of
(374)                             HomeKey : If (Filepos(NodeFile) < 5) then Wait := SoundError
(375)                                       Else
(376)                                       Begin
(377)                                            Clrscr;
(378)                                            Seek(NodeFile,0);
(379)                                       End;
(380)                             EndKey  : If Not Eof(NodeFile) then
(381)                                       Begin
(382)                                            Clrscr;
(383)                                            If ((Filesize(Nodefile) mod 4) = 0) then
(384)                                              Seek(NodeFile,Filesize(NodeFile)-4)
(385)                                            Else
(386)                                            Seek(NodeFile,Filesize(NodeFile)-(Filesize(Nodefile) mod 4));
(387)                                       End
(388)                                       Else Wait := SoundError;
(389)                             PgDn    : If Eof(NodeFile) then Wait := SoundError
(390)                                       Else Clrscr;
(391)                             PgUp    : If (Filepos(NodeFile) < 5) then Wait := SoundError
(392)                                       Else
(393)                                       Begin
(394)                                            Clrscr;
(395)                                            If ((Filepos(NodeFile) mod 4) = 0) then
(396)                                            Seek(NodeFile,Filepos(NodeFile)-8)
(397)                                            Else
(398)                                            Seek(NodeFile,Filepos(NodeFile)-(4+(Filepos(NodeFile) mod 4)));
(399)                                       End;
(400)                             Esc     : Seek(NodeFile,Filesize(NodeFile));
(401)                        End; {Case Wait}
(402)                   Until (Wait = Esc) or (Wait = HomeKey) or (Wait = Endkey) or
(403)                         (Wait = PgDn) or (Wait = PgUp);
(404)              End;
(405)         End;
(406)         Close(NodeFile);
```

Bylaag B.2
NODEUTIL.PAS

```
(407)     End;
(408) End;

(409) {*****************************************}
(410) {Print page header on each page of printout}
(411) {*****************************************}
(412) Procedure PrintConfigHeader(NodeName : String; Continue : Boolean);
(413) Begin
(414)     If Continue then                          {Continuing from previous page?}
(415)     Begin
(416)          Writeln(Lst,'Continue on next page .....'#12);
(417)          Writeln(Lst,'..... Continued from previous page');
(418)     End;
(419)     Writeln(Lst,'+-------------------------------------------------------------------------------+');
(420)     Writeln(Lst,'|                  LISTING OF NODE 8255 PIN CONFIGURATION                       |');
(421)     Writeln(Lst,'+-------------------------------------------------------------------------------+');
(422)     Writeln(Lst);
(423)     Writeln(Lst,'NODE FOR WHICH REPORT IS REQUESTED ->',Nodes[NodesB,NodesD]);
(424)     Writeln(Lst);
(425)     Writeln(Lst,'+----------+----------------------------------+----------+-------------+');
(426)     Writeln(Lst,'| 8255 PIN  | DESCRIPTION                      | PIN TYPE  | MEM ADDRESS |');
(427)     Writeln(Lst,'| ON TIMES  | T1        | T2        | T3        | T4        | T5          |');
(428)     Writeln(Lst,'| OFF TIMES | T1        | T2        | T3        | T4        | T5          |');
(429)     Writeln(Lst,'+-------------------------------------------------------------------------------+');
(430) End;

(431) {*********************************}
(432) {Print 8255 pin data to list device}
(433) {*********************************}
(434) Procedure PrintConfig(NodeName : String);
(435) Begin
(436)     If (Not NodeDbaseExist(NodeName)) then ErrorMsg('DATABASE DOESN''T EXIST YET !!')
(437)     Else
(438)     If PrnPresent then
(439)     Begin
(440)          PrintConfigHeader(NodeName,False);
(441)          Window(1,1,80,25);
(442)          WriteCenter(24,LightRed,Black,'Generating and printing report ...');
(443)          Reset(NodeFile);
(444)          While Not Eof(NodeFile) do
(445)          Begin
(446)               Read(Nodefile,NodeRec);
(447)               If (Not Eof(NodeFile)) then
(448)                  If ((Filepos(NodeFile) mod 13) = 0) then PrintConfigHeader(NodeName,True);
(449)               With NodeRec do
(450)               Begin
(451)                    Write(Lst,'|',Port,'  | ',Name,'    | ');
(452)                    If (Ptype = Output) then Write(Lst,'OUTPUT    | ')
```

Bylaag B.2
NODEUTIL.PAS

```
(453)                    Else Write(Lst,'INPUT     | ');
(454)                    Writeln(Lst,MemLoc,'        |');
(455)                    Writeln(Lst,'|          | ',Times[1,1],'     | ',Times[1,2],'     | ',
(456)                              Times[1,3],'     | ',Times[1,4],'     | ',Times[1,5],'       |');
(457)                    Writeln(Lst,'|          | ',Times[2,1],'     | ',Times[2,2],'     | ',
(458)                              Times[2,3],'     | ',Times[2,4],'     | ',Times[2,5],'       |');
(459)                    Writeln(Lst,'=============================================================================');
(460)               End;
(461)          End;
(462)          Writeln(Lst,'<<< End of report >>>'#12);
(463)          Close(NodeFile);
(464)     End
(465)     Else ErrorMsg('PRINTER DEVICE NOT READY OR DISCONNECTED !!');
(466) End;

(467) {*********************************************************}
(468) {Delete 8255 pin data from database - only 1 record at a time}
(469) {*********************************************************}
(470) Procedure DeleteRecord(Fpos : LongInt; ScrnPos : Byte);
(471) Var
(472)   BupFile : File of NodePortConfig;
(473) Begin
(474)     Window(6,7,78,20);
(475)     WriteAt(2,ScrnPos,Colors.TxtFg,Colors.TxtBg,'Deleting record ......          ');
(476)     Assign(BupFile,'NODEBUP.FIL');
(477)     Rewrite(BupFile);
(478)     Seek(NodeFile,FPos);
(479)     Read(NodeFile,NodeRec);
(480)     Seek(NodeFile,Filepos(NodeFile)-1);
(481)     NodeRec.Index := '#';
(482)     Write(NodeFile,NodeRec);
(483)     Close(NodeFile);
(484)     Reset(NodeFile);
(485)     While Not Eof(NodeFile) do
(486)     Begin
(487)          Read(NodeFile,NodeRec);
(488)          If (NodeRec.Index <> '#') then Write(BupFile,NodeRec);
(489)     End;
(490)     Close(NodeFile);
(491)     Erase(NodeFile);
(492)     Close(BupFile);
(493)     Rename(BupFile,NodeFileName);
(494)     Assign(NodeFile,NodeFileName);
(495)     Reset(NodeFile);
(496)     Seek(NodeFile,Filesize(NodeFile));
(497) End;
```

Bylaag B.2

NODEUTIL.PAS

```
(498) {***************************}
(499) {Choose 8255 pin to be erased}
(500) {***************************}
(501) Procedure ChooseRec;
(502) Var
(503)   Select    : Boolean;
(504)   CurPos    : LongInt;
(505)   I         : Byte;
(506)   Sel,Ch1   : Char;
(507)   Verify    : Char;
(508)   SelArr      : Array[1..12] of NodePortConfig;
(509)   DispPoint,ArrPoint,Ypl,Ypos : Byte;
(510) Begin
(511)     Reset(NodeFile);
(512)     Drawbox(2,4,79,22,15,0);
(513)     WriteAt(3,5,Colors.HlBg,Colors.HlFg,'    8255 PIN     DESCRIPTION                          PIN TYPE     MEM
   ADDRESS  ');
(514)     WriteAt(3,21,Colors.HlBg,Colors.HlFg,'    PgUp/PgDn    Home/End    CurUp/CurDown     [Esc]=Quit
   [Enter]=Select ');
(515)     HideCursor(14,0);
(516)     Select := False;
(517)     WriteAt(5,4,Colors.WinFg,Colors.WinBg,'Ñ');
(518)     For I := 5 to 21 do WriteAt(5,I,Colors.WinFg,Colors.WinBg,'³');
(519)     WriteAt(5,22,Colors.WinFg,Colors.WinBg,'Ï');
(520)     WriteAt(3,7,White,Black,'=□');
(521)     Window(6,7,78,20);
(522)     Ypos := 1;
(523)     Ypl := 0;
(524)     ArrPoint := 0;
(525)     Sel := NulKey;

(526)     While (Not Eof(NodeFile)) and (Sel <> Esc) and (Sel <> Enter) do
(527)     Begin
(528)          Read(NodeFile,NodeRec);
(529)          Inc(ArrPoint);
(530)          Ypl := ArrPoint;
(531)          SelArr[ArrPoint] := NodeRec;
(532)          With SelArr[ArrPoint] do
(533)          Begin
(534)               WriteAt(1,Ypl,Colors.TopicFg,Colors.TopicBg,Port);
(535)               WriteAt(15,Ypl,Colors.TopicFg,Colors.TopicBg,Name);
(536)               If (Ptype = Output) then WriteAt(48,Ypl,Colors.TopicFg,Colors.TopicBg,'OUTPUT')
(537)               Else WriteAt(48,Ypl,Colors.TopicFg,Colors.TopicBg,'INPUT');
(538)               WriteAt(61,Ypl,Colors.TopicFg,Colors.TopicBg,MemLoc);
(539)          End;
(540)          If Eof(NodeFile) then WriteAt(1,Ypl+2,LightRed,Black,'<< End of File >>');

(541)          If (ArrPoint = 12) or Eof(NodeFile) then
```

Bylaag B.2
NODEUTIL.PAS

```
(542)           Begin
(543)                Window(3,7,5,20);
(544)                GotoXY(1,Ypos);
(545)                Repeat
(546)                      Sel := GetkeyInput;

(547)                      Case Sel of
(548)                           Esc,MouseEsc : Seek(NodeFile,Filesize(NodeFile));

(549)                           CurDown,MouseDown : Begin
(550)                                                    WriteAt(1,WhereY,Colors.TxtFg,Colors.TxtBg,'  ');
(551)                                                    If (WhereY+1) = (ArrPoint+1) then
(552)                                                        WriteAt(1,1,White,Black,'=□')
(553)                                                    Else WriteAt(1,WhereY+1,White,Black,'=□');
(554)                                               End;

(555)                           CurUp,MouseUp : Begin
(556)                                                  WriteAt(1,WhereY,Colors.TxtFg,Colors.TxtBg,'  ');
(557)                                                  If ((WhereY-1)=0)  then WriteAt(1,ArrPoint,White,Black,'=□')
(558)                                                  Else WriteAt(1,WhereY-1,White,Black,'=□');
(559)                                             End;

(560)                           Enter,MouseEnter : Begin
(561)                                                   DispPoint := WhereY;
(562)                                                   CurPos := (Filepos(NodeFile)-(ArrPoint+1))+DispPoint;
(563)                                                   Window(6,7,78,20);
(564)                                                   WriteAt(2,ArrPoint+1,LightRed+128,0,'Delete record (Y/Esc) ?
   ...');
(565)
(566)                                                   Repeat
(567)                                                        Verify := Upcase(GetkeyInput);
(568)                                                   Until (Verify = 'Y') or (Verify = Esc);
(569)                                                   If (Verify = Esc) then
(570)                                                   Begin
(571)                                                        WriteAt(2,ArrPoint+1,
                                                                 Colors.TxtFg,Colors.TxtBg,'
   ');
(572)
(573)                                                        Sel := Nulkey;
(574)                                                        Window(3,7,5,20);
(575)                                                        GotoXY(1,DispPoint);
                                                        End;

(576)                                              End;
(577)                           Homekey : Begin
(578)                                          WriteAt(1,WhereY,Colors.TxtFg,Colors.TxtBg,'  ');
(579)                                          WriteAt(1,1,White,Black,'=□');
(580)                                          Window(6,7,78,20);
(581)                                          Clrscr;
(582)                                          ArrPoint := 0;
```

Bylaag B.2
NODEUTIL.PAS

```
(583)                                   Ypos := 1;
(584)                                   Seek(NodeFile,0);
(585)                           End;

(586)                    Endkey : If (FileSize(NodeFile) > 12) then
(587)                             Begin

(588)                                 ArrPoint := 0;
(589)                                 WriteAt(1,WhereY,Colors.TxtFg,Colors.TxtBg,'  ');
(590)                                 Window(6,7,78,20);
(591)                                 Clrscr;
(592)                                 If ((Filesize(Nodefile) mod 12) = 0) then
(593)                                    Seek(NodeFile,Filesize(NodeFile)-12)
(594)                                 Else
(595)                                 Seek(NodeFile,Filesize(NodeFile)-(Filesize(Nodefile) mod 12));
(596)                                 Window(3,7,5,20);
(597)                                 Ypos := (FileSize(NodeFile) mod 12);
(598)                                 If (Ypos = 0) then Ypos := 12;
(599)                                 WriteAt(1,Ypos,White,Black,'=□');
(600)                                 Window(6,7,78,20);
(601)                           End
(602)                           Else
(603)                           Begin
(604)                                 WriteAt(1,WhereY,Colors.TxtFg,Colors.TxtBg,'  ');
(605)                                 If (ArrPoint = 1) then WriteAt(1,1,White,Black,'=□')
(606)                                 Else WriteAt(1,ArrPoint,White,Black,'=□');
(607)                                 Ypos := WhereY;
(608)                                 Sel := Nulkey;
(609)                           End;

(610)                    PgDn : If (Not Eof(NodeFile)) then
(611)                           Begin
(612)                                 WriteAt(1,WhereY,Colors.TxtFg,Colors.TxtBg,'  ');
(613)                                 WriteAt(1,1,White,Black,'=□');
(614)                                 ArrPoint := 0;
(615)                                 Window(6,7,78,20);
(616)                                 Clrscr;
(617)                                 Ypos := 1;
(618)                           End
(619)                           Else Sel := Nulkey;

(620)                    PgUp : If (Filepos(NodeFile) > 13) then
(621)                           Begin
(622)                                 WriteAt(1,WhereY,Colors.TxtFg,Colors.TxtBg,'  ');
(623)                                 WriteAt(1,1,White,Black,'=□');
(624)                                 Ypos := 1;
(625)                                 Window(6,7,78,20);
(626)                                 Clrscr;
```

Bylaag B.2
NODEUTIL.PAS

```
(627)                                    Seek(NodeFile,Filepos(NodeFile)-(12+ArrPoint));
(628)                                    ArrPoint := 0;
(629)                               End
(630)                               Else Sel := Nulkey;
(631)                          End; {Case}
(632)               Until (Sel = Enter) or (Sel = MouseEnter) or
(633)                     (Sel = Esc) or (Sel = MouseEsc) or
(634)                     (Sel = Homekey) or (Sel = Endkey) or
(635)                     (Sel = PgUp) or (Sel = PgDn);

(636)               If ((Sel = Enter) or (Sel = MouseEnter)) and (Verify = 'Y')
(637)               then DeleteRecord(CurPos,ArrPoint+1);;
(638)          End;
(639)     End;

(640)     If (FileSize(NodeFile) > 0) then Close(NodeFile)
(641)     Else
(642)     Begin
(643)          Close(NodeFile);
(644)          Erase(NodeFile);
(645)     End;
(646) End;

(647) {************************************}
(648) {Does database exist for pin deletion ?}
(649) {************************************}
(650) Procedure DeletePinConfig(NodeName : String);
(651) Begin
(652)     If NodeDbaseExist(NodeName) then ChooseRec
(653)     Else ErrorMsg('DATABASE DOESN''T EXIST YET !!');
(654) End;

(655) {*****************************************************}
(656) {Get commandline parameters to determine desired procedure}
(657) {*****************************************************}
(658) Procedure DoNodeMenu;
(659) Var
(660)   I          : Byte;
(661)   NodeChosen : Boolean;
(662) Begin
(663)     Drawbox(20,10,45,21,15,0);
(664)     WriteAt(22,20,Colors.TxtFg,Colors.TxtBg,'Please select node ...');
(665)     For I := 1 to 8 do WriteAt(22,I+10,Colors.TopicFg,Colors.TopicBg,Nodes[0,I-1]);
(666)     For I := 1 to 8 do WriteAt(34,I+10,Colors.TopicFg,Colors.TopicBg,Nodes[1,I-1]);
(667)     NodeChosen := NodeChoose;

(668)     If NodeChosen then
(669)        If (Paramstr(2) = '/C') then ConfigurePins(Nodes[NodesB,NodesD])
```

Bylaag B.2
NODEUTIL.PAS

```
(627)                                        Seek(NodeFile,Filepos(NodeFile)-(12+ArrPoint));
(628)                                        ArrPoint := 0;
(629)                                   End
(630)                                   Else Sel := Nulkey;
(631)                              End; {Case}
(632)                   Until (Sel = Enter) or (Sel = MouseEnter) or
(633)                         (Sel = Esc) or (Sel = MouseEsc) or
(634)                         (Sel = Homekey) or (Sel = Endkey) or
(635)                         (Sel = PgUp) or (Sel = PgDn);

(636)                   If ((Sel = Enter) or (Sel = MouseEnter)) and (Verify = 'Y')
(637)                   then DeleteRecord(CurPos,ArrPoint+1);;
(638)              End;
(639)      End;

(640)      If (FileSize(NodeFile) > 0) then Close(NodeFile)
(641)      Else
(642)      Begin
(643)           Close(NodeFile);
(644)           Erase(NodeFile);
(645)      End;
(646)  End;

(647)  {************************************}
(648)  {Does database exist for pin deletion ?}
(649)  {************************************}
(650)  Procedure DeletePinConfig(NodeName : String);
(651)  Begin
(652)      If NodeDbaseExist(NodeName) then ChooseRec
(653)      Else ErrorMsg('DATABASE DOESN''T EXIST YET !!');
(654)  End;

(655)  {*****************************************************}
(656)  {Get commandline parameters to determine desired procedure}
(657)  {*****************************************************}
(658)  Procedure DoNodeMenu;
(659)  Var
(660)    I          : Byte;
(661)    NodeChosen : Boolean;
(662)  Begin
(663)      Drawbox(20,10,45,21,15,0);
(664)      WriteAt(22,20,Colors.TxtFg,Colors.TxtBg,'Please select node ...');
(665)      For I := 1 to 8 do WriteAt(22,I+10,Colors.TopicFg,Colors.TopicBg,Nodes[0,I-1]);
(666)      For I := 1 to 8 do WriteAt(34,I+10,Colors.TopicFg,Colors.TopicBg,Nodes[1,I-1]);
(667)      NodeChosen := NodeChoose;

(668)      If NodeChosen then
(669)         If (Paramstr(2) = '/C') then ConfigurePins(Nodes[NodesB,NodesD])
```

Bylaag B.2

NODEUTIL.PAS

```
(670)         Else
(671)         If (Paramstr(2) = '/V') then ViewPinConfig(Nodes[NodesB,NodesD])
(672)         Else
(673)         If (Paramstr(2) = '/P') then PrintConfig(Nodes[NodesB,NodesD])
(674)         Else
(675)         If (Paramstr(2) = '/D') then DeletePinConfig(Nodes[NodesB,NodesD])
(676)         Else
(677)         If (Paramstr(2) = '/X') then XtractInfo(Nodes[NodesB,NodesD],'');
(678) End;


(679) Begin
(680)     If ((Paramcount = 0) or (Paramstr(1) <> 'NODE8031')) THEN
(681)     Begin
(682)          Writeln('Error : This Program Cannot Run On It''s Own !');
(683)          Writeln('Run The MAINPROG Program !');
(684)     End
(685)     Else DoNodeMenu;
(686) End.
```

Bylaag B.3

RES_UTIL.PAS

```
(1)   {*********************************************************}
(2)   {PROGRAM:  RESULT UTILITIES MANAGER                        }
(3)   {PURPOSE:  Performs all result database related functions }
(4)   {*********************************************************}

(5)   {$M 4096,0,16384);                          {STACK SIZE = 4K; HEAP SIZE = 16K, MIN = 0K}
(6)   Program xResUtil;
(7)   Uses Crt,GetKey,ScrnUtil,ComnDecl,Printer; {Link to these units}
(8)   Const
(9)       Sfile = 'SORTFILE.FIL';                 {Sortfile name}

(10)  Type
(11)     DtaPointr = ^SortRec;                    {Sort record memory pointer}

(12)     SortRec = Record                         {Sort record structure}
(13)                     S1    : String[9];      {String to sort with}
(14)                     Fpos  : LongInt;        {Position in database}
(15)                     Next  : DtaPointr;      {Pointer to next record for linked list}
(16)               End;


(17)  Var
(18)    StartItem,                               {Linked list pointers}
(19)    NewItem,
(20)    OldItem,
(21)    CurItem ,
(22)    NextItem     : DtaPointr;
(23)    Sf           : File of XtractRecord;     {Sort file}

(24)    Found        : Boolean;                  {Found record ?}
(25)    FoundPtr     : LongInt;                  {Filepos of record found}


(26)  {*****************************}
(27)  {Exchange records in linked list}
(28)  {*****************************}
(29)  Procedure Exchange(RecA,RecB : DtaPointr);
(30)  Var
(31)    TempS1  : String[9];
(32)    TempPos : Integer;
(33)  Begin
(34)      With RecA^ do                          {Replace RecA's data with RecB's}
(35)      Begin
(36)           TempS1  := S1;
(37)           TempPos := Fpos;
(38)           S1      := RecB^.S1;
(39)           Fpos    := RecB^.Fpos;
(40)      End;
```

Bylaag B.3
RES_UTIL.PAS

```
(41)        With RecB^ do                          {Replace RecB's data with RecA's}
(42)        Begin
(43)             S1   := TempS1;
(44)             Fpos := TempPos;
(45)        End;
(46)   End;

(47)   {************************************}
(48)   {Sort result database using linked list}
(49)   {************************************}
(50)   Procedure SortToMemory(Fsize : Integer);
(51)   Var
(52)     I,J : Integer;
(53)   Begin
(54)        DispMessage(True,13,Colors.TxtFg,Colors.TxtBg,'Sorting result database for '+ResFileName);
(55)        Reset(ResFile);
(56)        If (MemAvail < 100) then ErrorMsg('Not enough memory to perform SORT !!!')
(57)        Else
(58)        Begin
(59)             Read(ResFile,ResRec);
(60)             New(StartItem);
(61)             With StartItem^ do
(62)             Begin
(63)                  Next := Nil;
(64)                  StartItem^.S1 := ResRec.Port;
(65)                  Fpos := Filepos(ResFile)-1;
(66)             End;
(67)             OldItem := StartItem;
(68)             While (Not Eof(ResFile)) do
(69)             Begin
(70)                  Read(ResFile,ResRec);
(71)                  If (MemAvail < 100) then
(72)                  Begin
(73)                       ErrorMsg('Not enough memory to perform SORT !!!');
(74)                       Close(ResFile);
(75)                       Exit;
(76)                  End
(77)                  Else
(78)                  Begin
(79)                       New(NewItem);
(80)                       NewItem^.Next := OldItem^.Next;
(81)                       OldItem^.Next := NewItem;
(82)                       NewItem^.S1 := ResRec.Port;
(83)                       NewItem^.Fpos := Filepos(ResFile)-1;
(84)                       OldItem := NewItem;
(85)                  End;
(86)             End;
(87)             Close(ResFile);
```

Bylaag B.3

RES_UTIL.PAS

```
(88)            For I := 1 to Fsize do
(89)            Begin
(90)                 CurItem := StartItem;
(91)                 NextItem := CurItem^.Next;
(92)                 While (NextItem <> Nil) do
(93)                 Begin
(94)                      If (NextItem^.S1 < CurItem^.S1) then Exchange(NextItem,CurItem);
(95)                         CurItem := CurItem^.Next;
(96)                         NextItem := CurItem^.Next;
(97)                 End;
(98)            End;

(99)            CurItem := StartItem;
(100)           Assign(Sf,Sfile);
(101)           Rewrite(Sf);
(102)           Reset(ResFile);
(103)           While (CurItem <> Nil) do
(104)           Begin
(105)                Seek(ResFile,CurItem^.Fpos);
(106)                Read(ResFile,ResRec);
(107)                Write(Sf,ResRec);
(108)                CurItem := CurItem^.Next;
(109)           End;
(110)           Close(ResFile);
(111)           Erase(ResFile);
(112)           Close(Sf);
(113)           Rename(Sf,ResFileName);

(114)           Release(StartItem);
(115)      End;
(116) End;

(117) {*********************************}
(118) {Can sorting being done on database?}
(119) {*********************************}
(120) Procedure VerifySort;
(121) Var
(122)   Fsize : Integer;
(123) Begin
(124)      Assign(ResFile,ResFileName);
(125)      Reset(ResFile);
(126)      Fsize := Filesize(ResFile);
(127)      Close(ResFile);
(128)                                          {If file has more than one record -> SORT!}
(129)      If (Fsize > 1) then SortToMemory(Fsize);
(130) End;
```

Bylaag B.3

RES_UTIL.PAS

```
(131) {*****************************}
(132) {View results in result database}
(133) {*****************************}
(134) Procedure ViewResults;                          {View results in node database}
(135) Var
(136)   I     : Byte;
(137)   Wait  : Char;
(138) Begin
(139)     Assign(ResFile,ResFileName);
(140)     Reset(ResFile);
(141)     DrawBox(2,4,79,22,15,0);
(142)     WriteAt(3,5,Colors.HlBg,Colors.HlFg,
(143)             ' 8255 PIN     DESCRIPTION                          TYPE   DATE       TIME   VAL ');
(144)     WriteAt(3,21,Colors.HlBg,Colors.HlFg,
(145)             ' File -> '+ParamStr(3)+'        <Esc> = Exit        PgDn/PgUp        Home/End  ');
(146)     Window(3,6,78,20);
(147)     While (Not Eof(ResFile)) do             {Display results on screen}
(148)     Begin
(149)          Read(ResFile,ResRec);
(150)          With ResRec do
(151)          Begin
(152)               SetTxtColor(Colors.TopicFg,Colors.TopicBg);
(153)               Write(Port,'     ',Name,'    ');
(154)               If (Ptype = Op) then Write('O/P    ')
(155)               Else
(156)               If (Ptype = Ip) then Write('I/P    ')
(157)               Else
(158)               If (Ptype = Undef) then Write('UNDEF  ');
(159)               Writeln(Date,'   ',Time,'   ',Result);
(160)               If ((Filepos(ResFile) mod 14) = 0) or Eof(ResFile) then
(161)               Repeat
(162)                    If Eof(ResFile) then WriteAt(1,WhereY,LightRed,Black,'<< End Of File >>');
(163)                  Wait := GetkeyInput;    {Wait for key to be pressed}
(164)                  Case Wait of
(165)                       HomeKey : If (Filepos(ResFile) < 15) then Wait := SoundError
(166)                                 Else
(167)                                 Begin
(168)                                      Clrscr;
(169)                                      Seek(ResFile,0);
(170)                                 End;
(171)                       EndKey  : If Not Eof(ResFile) then
(172)                                 Begin
(173)                                      Clrscr;
(174)                                      If (Filesize(Resfile) mod 14) = 0 then
(175)                                         Seek(ResFile,Filesize(ResFile)-14)
(176)                                      Else
(177)                                      Seek(ResFile,Filesize(ResFile)-(Filesize(Resfile) mod 14));
(178)                                 End
```

Bylaag B.3
RES_UTIL.PAS

```
(179)                                   Else Wait := SoundError;
(180)                          PgDn     : If Eof(ResFile) then Wait := SoundError
(181)                                     Else Clrscr;
(182)                          PgUp     : If (Filepos(ResFile) < 15) then Wait := SoundError
(183)                                     Else
(184)                                     Begin
(185)                                          Clrscr;
(186)                                          If ((Filepos(ResFile) mod 14) = 0) then
(187)                                          Seek(ResFile,Filepos(ResFile)-28)
(188)                                          Else
(189)                                          Seek(ResFile,Filepos(ResFile)-(14+(Filepos(ResFile) mod 14)));
(190)                                     End;
(191)                          Esc      : Seek(ResFile,Filesize(ResFile));
(192)                     End; {Case Wait}
(193)                Until (Wait = Esc) or (Wait = HomeKey) or (Wait = Endkey) or
(194)                      (Wait = PgDn) or (Wait = PgUp);
(195)           End;
(196)      End;
(197)      Close(ResFile);
(198) End;

(199) {***************************************}
(200) {Print page header on each page of printout}
(201) {***************************************}
(202) Procedure PrintHeader(Continue : Boolean);
(203) Begin
(204)      If Continue then                         {Continuation from previous pages?}
(205)      Begin
(206)           Writeln(Lst,'+-------------------------------------------------------------------------------+');
(207)           Writeln(Lst,'Continue on next page .....'#12);
(208)           Writeln(Lst,'..... Continued from previous page');
(209)      End;
(210)      Writeln(Lst,'+-------------------------------------------------------------------------------+');
(211)      Writeln(Lst,'|                  RESULT LISTING OF FILE : ',ResFileName,'                     |');
(212)      Writeln(Lst,'+-------------------------------------------------------------------------------+');
(213)      Writeln(Lst);
(214)      Writeln(Lst,'+-------------------------------------------------------------------------------+');
(215)      Writeln(Lst,'| 8255 PIN | DESCRIPTION                    | TYPE | RES DATE / TIME | RES |');
(216)      Writeln(Lst,'+----------+--------------------------------+------+-----------------+-----+');
(217) End;

(218) {***********************************}
(219) {Print results of node on list device}
(220) {***********************************}
(221) Procedure PrintResults;
(222) Begin
(223)      If (Not PrnPresent) then ErrorMsg('PRINTER DEVICE NOT READY OR DISCONNECTED !!')
(224)      Else
```

Bylaag B.3

RES_UTIL.PAS

```
(225)      Begin
(226)           WriteCenter(24,Red,Black,'Generating and printing report ...');
(227)           PrintHeader(False);
(228)           Assign(Resfile,ResFileName);
(229)           Reset(ResFile);
(230)           While Not Eof(ResFile) do
(231)           Begin
(232)                Read(ResFile,ResRec);
(233)                                          {Print max 48 records per page}
(234)                If (Filepos(ResFile) mod 49) = 0 then PrintHeader(True);
(235)                With ResRec do
(236)                Begin
(237)                     Write(Lst,'|',Port,' | ',Name,' | ');
(238)                     If (Ptype = Op) then Write(Lst,'O/P  | ')
(239)                     Else
(240)                     If (Ptype = Ip) then Write(Lst,'I/P  | ')
(241)                     Else
(242)                     If (Ptype = Undef) then Write(Lst,'UNDEF| ');
(243)                     Writeln(Lst,Date,'  ',Time,' |  ',Result,'  |');
(244)                End;
(245)           End;
(246)           Writeln(Lst,'+-------------------------------------------------------------------------------+');
(247)           Writeln(Lst,'<<< End of report >>>'#12);
(248)      End;
(249) End;

(250) {*****************************************************************}
(251) {Analize command line paramaters to determine procedure to execute}
(252) {*****************************************************************}
(253) Procedure ExecResultFunctions;
(254) Begin
(255)      ResFileName := ParamStr(3);           {Get result file name}
(256)      If (Paramstr(2) = '/V') then ViewResults
(257)      Else
(258)      If (Paramstr(2) = '/P') then PrintResults
(259)      Else
(260)      If (Paramstr(2) = '/S') then VerifySort;

(261) End;

(262) Begin
(263)      If ((Paramcount = 0) or (Paramstr(1) <> 'NODE8031')) THEN
(264)      Begin
(265)           Writeln('Error : This Program Cannot Run On It''s Own !');
(266)           Writeln('Run The MAINPROG Program !');
(267)      End
(268)      Else ExecResultFunctions;
(269) End.
```

Bylaag B.4

COMNDECL.PAS

```
(1)   {*********************************************************}
(2)   {UNIT:    COMMON DECLARATIONS                             }
(3)   {PURPOSE: All common declarations to be done in this unit.}
(4)   {*********************************************************}

(5)   UNIT ComnDecl;
(6)   INTERFACE
(7)   Uses Crt,Dos,ScrnUtil,Getkey,Com_Unit;      {Link to these units =}
(8)                                               {PASCAL Crt unit,}
(9)                                               {PASCAL Dos unit,}
(10)                                              {Screen utility unit,}
(11)                                              {Get key unit,}
(12)                                              {Communication unit.}

(13)  Const
(14)      SetupFileName = 'GENSETUP.FIL';         {Setup file name}
(15)      NaddrFileName = 'NODEADDR.CFG';         {Node address file name}


(16)      MaxLen = 30;                            {Max menu item string length}

(17)                                              {MAIN MENU topics}
(18)      MainMenu    : Array[0..1] of String[MaxLen] =
(19)                    (' Utilities',' Setup');
(20)                                              {UTILITY MENU topics}
(21)      UtilMenu    : Array[0..2] of String[MaxLen] =
(22)                    (' Extract Results',' Dbase Management',' Program Node');
(23)                                              {SETUP MENU topics}
(24)      SetupMenu   : Array[0..2] of String[MaxLen] =
(25)                    (' General Options',' Node Addresses',' Node Port Pins');
(26)                                              {NODE PORT PINS MENU topics}
(27)      NodeIdMenu  : Array[0..3] of String[MaxLen] =
(28)                    (' View',' Edit',' Print',' Delete');
(29)                                              {GENERAL SETUP MENU topics}
(30)      GenSetupMenu : Array[0..1] of String[MaxLen] =
(31)                     (' View',' Edit');
(32)                                              {NODE ADDRESSES MENU topics}
(33)      NodeAddrMenu : Array[0..1] of String[MaxLen] =
(34)                    (' View',' Edit');
(35)                                              {Node Menu selection strings}
(36)      Nodes      : Array[0..1,0..7] of String[8] =
(37)                   ((' Node 01',' Node 02',' Node 03',' Node 04',
(38)                     ' Node 05',' Node 06',' Node 07',' Node 08'),
(39)                    (' Node 09',' Node 10',' Node 11',' Node 12',
(40)                     ' Node 13',' Node 14',' Node 15',' Node 16'));
(41)                                              {DBASE MANAGEMENT MENU topics}
(42)      DbaseMenu    : Array[0..2] of String[MaxLen] =
(43)                    (' View',' Print',' Sort');
```

Bylaag B.4

COMNDECL.PAS

```
(44)  Type
(45)     Str2 = String[2];                          {String of 2 characters}

(46)     SetupRecord = Record                       {Setup structure}
(47)                      ComPort  : Byte;          {Comm port used}
(48)                      BaudRate : Word;          {Baud rate - 1200 fixed for this appl}
(49)                      Poll,                     {Polling time interval}
(50)                      Hold     : Byte;          {Result expiration interval}
(51)                End;

(52)     NodeAddrRecord = Record                    {Node address structure}
                                              (53)      2char strings}
(54)                           NodeArr : Array[0..1,0..7] of String[2];
(55)                      End;

(56)     NodePortConfig = Record                    {8255 PPI pin structure}
(57)                           Index    : Char; {Index for delete status}
                                            (58)      port pin number}
(59)                           Port     : String[9];
(60)                                              {Id name of pin}
(61)                           Name     : String[30];
(62)                                              {Input or output pin ?}
(63)                           Ptype    : (Input,Output);
(64)                                              {Memory location in node RAM}
(65)                           MemLoc   : String[4];
(66)                                              {On/Off times of pin - max 5 each}
(67)                           Times    : Array[1..2,1..5] of
(68)                                      String[5];
(69)                      End;

(70)     XtractRecord = Record                     {Node result structure}
(71)                          Index    : Char;    {Index for delete status}
                                                (72)      pin number}
(73)                          Port     : String[9];
(74)                                              {Id name of pin}
(75)                          Name     : String[30];
(76)                                              {Input/Output/Undefined pin ?}
(77)                          Ptype    : (Ip,Op,Undef);
(78)                                              {Date of result request}
(79)                          Date     : String[8];
(80)                                              {Time of result request}
(81)                          Time     : String[5];
(82)                          Result   : Byte;    {Result of pin 0 or 1}
(83)                    End;
```

Bylaag B.4
COMNDECL.PAS

```
(84)  Var
(85)    SetupFile     : File of SetupRecord;       {Typed file for setup record}
(86)    SetupRec      : SetupRecord;               {Variable to structure of setup}
(87)    NaddrFile     : File of NodeAddrRecord;    {Typed file for node addresses}
(88)    NaddrRec      : NodeAddrRecord;            {Variable to structure of address structure}
(89)    NodeFile      : File of NodePortConfig;    {Typed file for 8255 pin config}
(90)    NodeRec       : NodePortConfig;            {Variable to structure of 8255 pin config}
(91)    ResFile       : File of XtractRecord;      {Typed file for results from node}
(92)    ResRec        : XtractRecord;              {Variable to result structure}

(93)    NodeFileName,                              {Current node file loaded}
(94)    ResFileName   : String;                    {Current result file loaded}
(95)    NodesA,                                    {Two dimensional pointers to}
(96)    NodesB,                                    {node menu selection strings.}
(97)    NodesC,
(98)    NodesD        : ShortInt;
(99)    Regs          : Registers;                 {DOS registers variable}
(100)   RegES,RegBX   : Word;                      {ES & BX register values}


(101) Function DosSafe : Boolean;                   {DOS currently busy serving other function calls?}
(102) Function PrnPresent : Boolean;                {Printer present/online?}
(103) Function SetupFileExist : Boolean;            {General setup done yet?}
(104) Function NodeAddrFileExist : Boolean;         {Node addresses specified yet?}
(105)                                               {Specified comm port existing?}
(106) Function ComPortExist(Cport : Byte) : Boolean;
(107) Function NodeChoose : Boolean;                {Node been chosen?}
(108)                                               {Node pin config database exists?}
(109) Function NodeDbaseExist(NodeName : String) : Boolean;
(110)                                               {Result file for node exists?}
(111) Function ResFileExist(NodeName : String) : Boolean;
(112) Function ConvStr2(N : Integer) : Str2;        {Convert integer values to 2char strings}
(113) Function SystemDate : String;                 {Return current system date}
(114) Function SystemTime : String;                 {Return current system time}
(115) Function HexStrToByte(HexStr : Str2) : Byte;{Convert HEX address string to BYTE data type}
(116)                                               {Extract results from node}
(117) Procedure XtractInfo(NodeName, PollId : String);


(118) IMPLEMENTATION

(119) {*********************************}
(120) {DOS busy with other function calls?}
(121) {*********************************}
(122) Function DosSafe : Boolean;
(123) Begin
(124)     DosSafe := (Mem[RegES:RegBX]=0);          {INDOS flag = true?}
(125) End;
```

Bylaag B.4

COMNDECL.PAS

```
(126) {****************}
(127) {Printer present?}
(128) {****************}
(129) Function PrnPresent : Boolean;
(130) Var
(131)   Result : Byte;
(132) Begin
(133)     With Reg do
(134)     Begin
(135)          Ah := $02;
(136)          Dx := 0;
(137)          Intr($17,Reg);
(138)          Result := Ah;
(139)   PrnPresent := (Result = 144);    {Printer present/online?}
(140)     End;
(141) End;

(142) {***********************}
(143) {General setup done yet?}
(144) {***********************}
(145) Function SetupFileExist : Boolean;
(146) Begin
(147)     Assign(SetupFile,SetupFileName);
(148)     {$i-} Reset(SetupFile); {$i+}
(149)     If (IoResult <> 0) then SetupFileExist := False
(150)     Else
(151)     Begin
(152)          Read(SetupFile,SetupRec);
(153)          Close(SetupFile);
(154)          SetupFileExist := True;
(155)     End;
(156) End;

(157) {*************************}
(158) {Node addresses specified?}
(159) {*************************}
(160) Function NodeAddrFileExist : Boolean;
(161) Begin
(162)     Assign(NaddrFile,NaddrFileName);
(163)     {$i-} Reset(NaddrFile); {$i+}
(164)     If (IoResult <> 0) then NodeAddrFileExist := False
(165)     Else
(166)     Begin
(167)          Read(NaddrFile,NaddrRec);
(168)          Close(NaddrFile);
(169)          NodeAddrFileExist := True;
(170)     End;
(171) End;
```

Bylaag B.4

COMNDECL.PAS

```
(172) {**************************}
(173) {Specified comm port exists?}
(174) {**************************}
(175) Function ComPortExist(Cport : Byte) : Boolean;
(176) Begin
(177)    ComPortExist := ((Port[PortParam[Cport].Base+MCR] and $E0) = 0);
(178) End;

(179) {****************************}
(180) {Choose node from menu strings}
(181) {****************************}
(182) Function NodeChoose : Boolean;
(183) Var
(184)   Fin,NoChoice : Boolean;
(185)   NodeCh       : Char;
(186)   NodeStr      : String[2];
(187)   Error        : Integer;

(188)   Procedure NodeHilight;                    {Do menu hilighting for topics}
(189)   Begin
(190)        Window((12*NodesA)+22,11+NodesC,(12*NodesA)+22+Length(Nodes[NodesA,NodesC]),11+NodesC);
(191)        SetTxtColor(Colors.TopicFg,Colors.TopicBg);
(192)        Clrscr;
(193)        Write(Nodes[NodesA,NodesC]);
(194)        Window((12*NodesB)+22,11+NodesD,(12*NodesB)+22+Length(Nodes[NodesB,NodesD]),11+NodesD);
(195)        SetTxtColor(Colors.HlBg,Colors.HlFg);
(196)        Clrscr;
(197)        Write(Nodes[NodesB,NodesD]);
(198)        SetTxtColor(Colors.TopicFg,Colors.TopicBg);
(199)   End; {of procedure nodehilight}

(200) Begin                                       {Select node from menu}
(201)    NodesA := 0; NodesB := 0;
(202)    NodesC := 0; NodesD := 0;

(203)    Fin := False;
(204)    NoChoice := False;
(205)    NodeHilight;
(206)    Repeat
(207)          NodesA := NodesB; NodesC := NodesD;
(208)          NodeCh := GetkeyInput;
(209)          Case NodeCh of
(210)               CurRight,MouseRight : Inc(NodesB);
(211)               CurLeft,MouseLeft   : Dec(NodesB);
(212)               CurDown,MouseDown   : Inc(NodesD);
(213)               CurUp,MouseUp       : Dec(NodesD);
(214)               Esc,MouseEsc        : Begin
(215)                                       Fin := True;
```

Bylaag B.4
COMNDECL.PAS

```
(216)                                          NoChoice := True;
(217)                                     End;
(218)            End; {Case}

(219)            If (NodesB = 2) then NodesB := 0;
(220)            If (NodesB = -1) then NodesB := 1;
(221)            If (NodesD = 8) then NodesD := 0;
(222)            If (NodesD = -1) then NodesD := 7;
(223)            If (Not Fin) then NodeHilight;
(224)       Until (NodeCh = Enter) or (NodeCh = MouseEnter) or (Fin);
(225)       NodeChoose := (Not Fin);                {Node chosen?}
(226)  End;

(227)  {************************}
(228)  {Pin config done for node? }
(229)  {************************}
(230)  Function NodeDbaseExist(NodeName : String) : Boolean;
(231)  Begin
(232)       Delete(NodeName,1,1);
(233)       NodeName[5] := ' ';
(234)       NodeFileName := NodeName+'.CFG';       {Filename:=? eg. "NODE_01.CFG"}
(235)       Assign(NodeFile,NodeFileName);
(236)       {$i-} Reset(NodeFile); {$i+}
(237)       If (IoResult = 0) then
(238)       Begin
(239)            NodeDbaseExist := True;
(240)            Close(NodeFile);
(241)       End
(242)       Else NodeDbaseExist := False;
(243)  End;

(244)  {****************************}
(245)  {Result database exist for node?}
(246)  {****************************}
(247)  Function ResFileExist(NodeName : String) : Boolean;
(248)  Begin
(249)       Delete(NodeName,1,1);
(250)       NodeName[5] := ' ';
(251)       ResFileName := NodeName+'.RES';        {Filename:=? eg. "NODE_01.RES"}
(252)       Assign(ResFile,ResFileName);
(253)       {$i-} Reset(ResFile); {$i+}
(254)       If (IoResult = 0) then
(255)       Begin
(256)            ResFileExist := True;
(257)            Close(ResFile);
(258)       End
(259)       Else ResFileExist := False;
(260)  End;
```

Bylaag B.4

COMNDECL.PAS

```
(261) {*****************************************}
(262) {Convert integer values to 2character strings}
(263) {*****************************************}
(264) Function ConvStr2(N : Integer) : Str2;
(265) Var
(266)   S : Str2;
(267) Begin
(268)     Str(N:2,S);
(269)     If (N < 10) then S[1] := '0';          {String:=? eg. "02"}
(270)     ConvStr2 := S;
(271) End;

(272) {************************************************}
(273) {Current system date? return format = eg. "92/01/01"}
(274) {************************************************}
(275) Function SystemDate : String;
(276) Var
(277)   Year,Month,Day,Dow : Word;
(278)   YearStr            : String[4];
(279) Begin
(280)     GetDate(Year,Month,Day,Dow);
(281)     Str(Year,YearStr);
(282)     Delete(YearStr,1,2);
(283)     SystemDate := Concat(YearStr,'/',ConvStr2(Month),'/',ConvStr2(Day));
(284) End;

(285) {************************************************}
(286) {Current system time? return format = eg. "10:32:15.01"}
(287) {************************************************}
(288) Function SystemTime : String;
(289) Var
(290)   Hour,Min,Sec,Sec100 : Word;
(291) Begin
(292)     GetTime(Hour,Min,Sec,Sec100);
(293)     SystemTime := Concat(ConvStr2(Hour),':',ConvStr2(Min),':',ConvStr2(Sec),'.',ConvStr2(Sec100));
(294) End;

(295) {*************************************}
(296) {Convert HEX addr string to BYTE datatype}
(297) {*************************************}
(298) Function HexStrToByte(HexStr : Str2) : Byte;
(299) Var
(300)   I,J        : Byte;
(301)   Pos1,Pos2  : Byte;                    {Positional results in string}
(302) Begin
(303)     For J := 2 downto 1 do
(304)     Begin
(305)          Case HexStr[J] of
```

Bylaag B.4

COMNDECL.PAS

```
(306)           '0'..'9' : I := Ord(HexStr[J])-48;
(307)           'A'..'F' : I := Ord(HexStr[J])-55;
(308)           End;
(309)           If (J = 2) then Pos2 := I;
(310)           If (J = 1) then Pos1 := (16 * I);
(311)      End;
(312)      HexStrToByte := Pos1 + Pos2;
(313) End;

(314) {**********************}
(315) {Get results from nodes}
(316) {**********************}
(317) Procedure XtractInfo(NodeName, PollId : String);
(318) Var
(319)   Cport,                                    {Comm port?}
(320)   SecCount,                                 {Seconds to acknowledge}
(321)   Address  : Byte;                          {Node address}
(322)   BaudRate : Word;                          {Baud rate?}
(323)   Error    : Integer;
(324)   PrevSec,                                  {Second strings to determine}
(325)   CurrSec  : String[2];                     {acknowledge timeout}
(326)   DelayEnd,                                 {End of acknowledge delay?}
(327)   Found    : Boolean;                       {Node found?}
(328)   Wait     : Char;
(329)   Rec No,                                   {No of records programmed in node}
(330)   DtaPtr,                                   {Data buffer offset pointer}
(331)   I,J      : Byte;
(332)   CurrTime : String[5];
(333) Begin
(334)      If NodeDbaseExist(NodeName) then
(335)      Begin
(336)           If (PollId <> '/PNX') then        {Polling nodes for results?}
(337)           Begin                             {Parameters passed through command line}
(338)                DispMessage(True,13,Colors.HlBg,Colors.HlFg,'Initializing com port
   '+ParamStr(3)+','+ParamStr(4)+',N,1');
(339)                Val(ParamStr(3),Cport,Error);
(340)                Val(ParamStr(4),BaudRate,Error);
(341)                InitPort(Cport,BaudRate);
(342)           End;

(343)           Drawbox(10,10,70,20,Colors.HlFg,Colors.TxtBg);
(344)           WriteCenter(10,Colors.TxtFg,Colors.TxtBg,' Extracting data for'+NodeName+' ');
(345)           Window(11,11,69,19);
(346)           SetDtrHigh(True);

(347)           If NodeAddrFileExist then
(348)           Begin
(349)                Reset(NaddrFile);
```

Bylaag B.4
COMNDECL.PAS

```
(350)               Read(NaddrFile,NaddrRec);
(351)               Close(NaddrFile);
(352)          End;

(353)          {Convert hex string to byte value and add address id bit}
(354)          Address := HexStrToByte(NaddrRec.NodeArr[NodesB,NodesD]) or $80;

(355)          Write('Addressing node');
(356)          TxString(#32+Char(Address));
(357)          Acknowledge := False;

(358)          {give node sufficient time to acknowledge - max 10 seconds}
(359)          PrevSec := Copy(SystemTime,7,2);
(360)          SecCount := 0;
(361)          DelayEnd := False;
(362)          While (Not Acknowledge) and (Not DelayEnd) do
(363)          Begin
(364)               CurrSec := Copy(SystemTime,7,2);
(365)               If (CurrSec <> PrevSec) and (Not Acknowledge) then
(366)               Begin
(367)                    Write('.');
(368)                    Inc(SecCount);
(369)                    PrevSec := CurrSec;
(370)               End;
(371)               If (SecCount = 10) then DelayEnd := True;
(372)          End;

(373)          Delay(1000);
(374)          Writeln;

(375)          If Acknowledge then
(376)          Begin
(377)               Acknowledge := False;
(378)               Writeln('Transmitting result request command ...');
(379)               TxString(#32+Char(Enq));
(380)               While (Not Acknowledge) do;  {ensure node sends all it's messages before capturing any data}
(381)               Acknowledge := False;
(382)               Capt := True;                {read data from node into stream? = true}
(383)               While (Not Acknowledge) do;
(384)               Capt := False;
(385)               Acknowledge := False;
(386)               Repeat
(387)                    Wait := Readkey;
(388)               Until (Wait = Enter);
(389)               CurrTime := Copy(SystemTime,1,5);
(390)               TxString(#32+(Char(Eot)));
(391)          End
(392)          Else
```

Bylaag B.4
COMNDECL.PAS

```
(393)           Begin
(394)                Writeln('No acknowledgement from node => Check addresses !!!');
(395)                Delay(1000);
(396)           End;

(397)           SetDtrHigh(False);
(398)           RestoreValues;

(399)           If ResFileExist(NodeName) then Reset(ResFile)
(400)           Else Rewrite(ResFile);
(401)           Seek(ResFile,FileSize(ResFile));

(402)           {*********************}
(403)           {Process data received}
(404)           {*********************}

(405)           Val(CaptLine[1]+CaptLine[2],Rec_No,Error);
(406)           DtaPtr := 3;
(407)           For I := 1 to Rec_No do
(408)           Begin
(409)                With ResRec do
(410)                Begin
(411)                     Found := False;
(412)                     Index := '*';
(413)                     Port := ' Port ';
(414)                     For J := 0 to 2 do Port := Port+CaptLine[DtaPtr+J];
(415)                     Reset(NodeFile);
(416)                     While (Not Eof(NodeFile)) and (Not Found) do
(417)                     Begin
(418)                          Read(NodeFile,NodeRec);
(419)                          If (NodeRec.Port = Port) then
(420)                          Begin
(421)                               Found := True;
(422)                               Name := NodeRec.Name;
(423)                               If (NodeRec.Ptype = Output) then Ptype := Op
(424)                               Else Ptype := Ip;
(425)                          End;
(426)                     End;
(427)                     Close(NodeFile);
(428)                     If (Not Found) then
(429)                     Begin
(430)                          Ptype := Undef;
(431)                          Name := 'PORT PIN IS UNDEFINED !!!     ';
(432)                     End;
(433)                     Date := SystemDate;
(434)                     Time := CurrTime;
(435)                     Val(CaptLine[DtaPtr+3],Result,Error);
(436)                End;
```

Bylaag B.4
COMNDECL.PAS

```
(437)                Inc(DtaPtr,4);
(438)                Write(ResFile,ResRec);
(439)           End;

(440)           If (FileSize(ResFile) > 0) then Close(ResFile)
(441)           Else
(442)           Begin
(443)                Close(ResFile);
(444)                Erase(ResFile);
(445)           End;

(446)      End
(447)      Else ErrorMsg('DATABASE DOESN''T EXIST YET !!');
(448) End;


(449) Begin
(450)      Regs.AH := $34;                        {Get memory pointer of INDOS flag}
(451)      Intr($21,Regs);
(452)      RegES := Regs.ES;                      {Save pointer in ES:BX registers}
(453)      RegBX := Regs.BX;
(454) End.
```

Bylaag B.5

COM_UNIT.PAS

```
(1)   {*****************************************************}
(2)   {UNIT:    COMMUNICATIONS UNIT                         }
(3)   {PURPOSE: To be the serial interface for communicating}
(4)   {         with the microcontroller nodes.             }
(5)   {*****************************************************}

(6)   UNIT Com_Unit;

(7)   INTERFACE

(8)   Uses Crt,Dos;
(9)   Const                                    {Offset of comm registers from base address}
(10)      THR = 0;                             {Transmit holding register offset}
(11)      IER = 1;                             {Interrupt enable register offset}
(12)      IIR = 2;                             {Interrupt identification register offset}
(13)      LCR = 3;                             {Line control register offset}
(14)      MCR = 4;                             {Modem control register offset}
(15)      LSR = 5;                             {Line status register offset}
(16)      MSR = 6;                             {Modem status register offset}

(17)      PortParam : Array[1..4] of           {Communication port structure}
(18)                  Record
(19)                        Base : Word;       {Base address of port}
(20)                        Int  : Byte;       {Interrupt assigned to port}
(21)                  End
(22)                  = ((Base:$03F8; Int:$0C),{Com 1, int 0CH}
(23)                     (Base:$02F8; Int:$0B),{Com 2, int 0BH}
(24)                     (Base:$03E8; Int:$0C),{Com 3, int 0CH}
(25)                     (Base:$02E8; Int:$0B));{Com 4, int 0BH}

(26)                                           {Communication id constants}
(27)      Prg = $03;                           {Master wants to program node}
(28)      Eot = $04;                           {End of transmission}
(29)      Enq = $05;                           {Enquiry - get results from node}
(30)      Ack = $06;                           {Acknowledgement from node}

(31)      MaxBuff = 500;                       {max data buffer size in characters}

(32)  Var
(33)    OldIntVect  : Pointer;                 {Saved interrupt vector for interrupt handler}
(34)    OldLCR      : Byte;                    {Saved LCR value}
(35)    OldIER      : Byte;                    {Saved IER value}
(36)    OldMCR      : Byte;                    {Saved MCR value}
(37)    Reg         : Registers;               {Variable to DOS register type}
(38)    ComPort     : Byte;                    {Current comm port}
(39)    CurCh,
(40)    PrevCh      : Byte;                    {Previous character read from comm port}
(41)    Capt        : Boolean;                 {Need to read data into buffer?}
```

Bylaag B.5

COM_UNIT.PAS

```
(42)     CaptLine     : Array[1..MaxBuff] of Char; {Data buffer}
(43)     DtaPoint     : Word;                      {Data buffer offset pointer}
(44)     Acknowledge  : Boolean;                   {Acknowledgement flag}
(45)     TxDelay      : Byte;                      {Transmit delay variable for different baud rates}

(46)                                               {Predefined procedures for unit}
(47)                                               {Initialise comm port}
(48)   Procedure InitPort(CPort : Byte; BaudRate : Word);
(49)   Procedure SetDtrHigh(High : Boolean);       {Set DTR high/low}
(50)   Procedure TxString(S : String);            {Transmit textstring}
(51)   Procedure RestoreValues;                   {Restore saved values on exit}

(52)   IMPLEMENTATION

(53)   {************************************}
(54)   {Serial interrupt handler to receive data}
(55)   {************************************}
(56)   Procedure IntHandler; Interrupt;
(57)   Begin
(58)       CurCh := Port[PortParam[ComPort].Base];{Read character from comm port}
(59)       If ((CurCh <> 13) and (CurCh <> 10))
(60)          and (PrevCh = 13) then Writeln;     {Give CR & LF}
(61)       If (CurCh <> 0) then
(62)       Begin                                   {Aknowledgement from node ?}
(63)            If (CurCh = Ack) then Acknowledge := True;
(64)                                               {Write character to screen if printable}
(65)            If Not(CurCh in [0..31]) then Write(Char(CurCh));

(66)            If (Capt) then                     {Read data into buffer if needed}
(67)            Begin
(68)                 Inc(DtaPoint);                {Offset + 1}
(69)                                               {Put serial character into buffer}
(70)                 CaptLine[DtaPoint] := Char(CurCh);
(71)            End;
(72)            PrevCh := CurCh;                   {Prepare to read next character from comm port}
(73)       End;
(74)       Port[$20] := $20;                       {Reset "READ" detection}
(75)   End;

(76)   {***************}
(77)   {Init serial port}
(78)   {***************}
(79)   Procedure InitPort(CPort : Byte; BaudRate : Word);
(80)   Var
(81)     Temp,Temp2 : Byte;
(82)   Begin                                        {Clear data buffer}
(83)       For DtaPoint := 1 to MaxBuff do CaptLine[DtaPoint] := ' ';
```

Bylaag B.5
COM_UNIT.PAS

```
(84)       DtaPoint := 0;                          {Data buffer offset = 0}
(85)       ComPort := Cport;                       {Get value of port to init}
(86)       TxDelay := (((4800 div BaudRate)*6)-5);{Calculate delay between char's for baud rate}
(87)       With Reg do
(88)       Begin
(89)             Ah := 0;
(90)             Dx := (ComPort-1);
(91)             Case BaudRate of
(92)                  1200 : Al := $83;
(93)                  2400 : Al := $A3;          { Set values for baudrate }
(94)                  4800 : Al := $C3;
(95)             End;
(96)             Intr($14,Reg);                  {Init comm port 8 databits,}
(97)                                             {               No parity, }
(98)                                             {               1 stop bit.}
(99)       End;

(100)      With PortParam[ComPort] do
(101)      Begin
(102)           CurCh := 0; PrevCh := 0;          {Characters read = 0}
(103)           GetIntvec(Int,OldIntVect);        {Save old Com Int Value }
(104)           SetIntvec(Int,@IntHandler);       {Set Com Int to the INThandler}
(105)           Inline($FA);                      {Disable interrupts}
(106)           Temp  := Port[Base+LSR];
(107)           Temp  := Port[Base];              {Clear the values of}
(108)           Temp  := Port[$21];               {  specific addr's  }
(109)           Temp2 := (1 Shl (Int-8));
(110)                                             { Set value for EOI signal }
(111)           Port[$21] := (Temp and (Not Temp2));
(112)           OldLCR := Port[Base+LCR];         {Save current register values}
(113)           OldIER := Port[Base+IER];
(114)           OldMCR := Port[Base+MCR];
(115)           Port[Base+IER] := $01;            {Enable RXdata Interrupt Detection}
(116)           Port[Base+MCR] := (OldMCR or $08);
(117)           Inline($FB);                      {Enable interrupts}
(118)           Port[$20] := $20;                 {Issue this statement after a READ operation also}
(119)           Capt := False;                    {to reset the "read detection"                    }
(120)      End;
(121) End;

(122) {******************}
(123) {Set DTR to high/low}
(124) {******************}
(125) Procedure SetDtrHigh(High : Boolean);
(126) Begin
(127)      If High then Port[PortParam[ComPort].Base+MCR] := $0B
(128)      Else Port[PortParam[ComPort].Base+MCR] := $08;
(129) End;
```

Bylaag B.5

COM_UNIT.PAS

```
(130) {*************************************}
(131) {Transmit a single character on comm port}
(132) {*************************************}
(133) Procedure TxChar(Ch : Char);
(134) Var
(135)   TxReady : Boolean;                        {Ready for transmission?}
(136) Begin
(137)     TxReady := False;
(138)     With PortParam[ComPort] do
(139)     Begin                                   {Wait until ready for transmission}
(140)          While (Not TxReady) do TxReady := ((Port[LSR+Base] and $20)=$20);
(141)          Port[Base] := Ord(Ch);             {Transmit character}
(142)     End;
(143) End;

(144) {*******************************}
(145) {Transmit text string on comm port}
(146) {*******************************}
(147) Procedure TxString(S : String);
(148) Var
(149)   I : Byte;
(150) Begin
(151)     For I := 1 to Length(S) do
(152)     Begin
(153)          TxChar(S[I]);                      {Transmit current character in string}
(154)          Delay(TxDelay);                    {Wait for USART to send next character}
(155)     End;
(156) End;

(157) {*************************************}
(158) {Restore register values to original state}
(159) {*************************************}
(160) Procedure RestoreValues;
(161) Begin
(162)      With PortParam[ComPort] do
(163)      Begin
(164)           SetIntvec(Int,OldIntVect);
(165)           Port[Base+LCR] := OldLCR;
(166)           Port[Base+IER] := OldIER;
(167)           Port[Base+MCR] := OldMCR;
(168)    End;
(169) End;

(170) Begin
(171)     Capt := False;                          {No need to read data to buffer at startup}
(172)     Acknowledge := False;                   {No acknowledgement}
(173) End.
```

Bylaag B.6

EDITLN.PAS

```
(1)   {*****************************************}
(2)   {UNIT:    FULL LINE EDITOR                 }
(3)   {PURPOSE: To act as a full line editor with  }
(4)   {         HOME,END, etc. function integrated.}
(5)   {*****************************************}

(6)   UNIT EditLn;
(7)   INTERFACE
(8)   Uses Crt,Getkey;

(9)   Type
(10)   CharSet = Set of Char;
(11)                                          {SET of characters}
(12)  Var
(13)    EditFin,                              {Done with editing?}
(14)    EditSave : Boolean;                   {Save after editing?}


(15)  {*********************************************}
(16)  {Edit line specified- paramaters as follows:    }
(17)  {S = string to be edited.                       }
(18)  {X/Y = x/y-coordinates on screen.               }
(19)  {Len = max length of string S.                  }
(20)  {LegalChars = legal characters for S.           }
(21)  {Term = characters to indicate end of editing.  }
(22)  {*********************************************}
(23)  Procedure EditLine(Var S : String; X, Y, Len : Byte;
(24)                     LegalChars, Term : Charset);

(25)  IMPLEMENTATION
(26)  {$V-}

(27)  {*************************************}
(28)  {Edit string S at X,Y for LEN characters}
(29)  {*************************************}
(30)  Procedure EditLine(Var S : String; X, Y, Len : Byte;
(31)                     LegalChars, Term : Charset);
(32)  Var
(33)   P,I              : Byte;
(34)   Ch               : Char;
(35)   First            : Boolean;
(36)  Begin
(37)      First := True;
(38)      EditFin := False;
(39)      EditSave := False;
(40)      GotoXY(X,Y);
(41)      Write(S);
```

Bylaag B.6

EDITLN.PAS

```
(42)        P := 0;
(43)        Repeat
(44)        GotoXY(X+P,Y);
(45)        Ch := Upcase(GetkeyInput);
(46)        If Not (Upcase(Ch) in Term) then
(47)        Case Ch of
(48)             #32..#126 : If (P < Len) and (Ch in LegalChars) then
(49)                         Begin
(50)                              If First then
(51)                              Begin
(52)                                   Write(' ':Len);
(53)                                   Delete(S,P+1,Len);
(54)                                   GotoXY(X+P,Y);
(55)                              End;
(56)                              If Length(S) = Len then
(57)                              Delete(S,Len,1);
(58)                              P := Succ(P);
(59)                              Insert(Ch,S,P);
(60)                              Write(Copy(S,P,Len));
(61)                         End
(62)                         Else
(63)                         Begin
(64)                              Sound(500);
(65)                              Delay(200);
(66)                              Nosound;
(67)                         End;

(68)             ^S, CurLeft  : If (P > 0) then  P := Pred(P);

(69)             ^D, CurRight : If (P < Length(S)) then P := Succ(P);

(70)             ^A, HomeKey  : P := 0;

(71)             ^F, EndKey   : P := Length(S);

(72)             ^G, DelKey   : If (P < Length(S)) then
(73)                            Begin
(74)                                 Delete(S,P+1,1);
(75)                                 Write(Copy(S,P+1,Len),' ');
(76)                            End;
(77)                    BS    : If (P > 0) then
(78)                            Begin
(79)                                 Delete(S,P,1);
(80)                                 Write(^H,Copy(S,P,Len),' ');
(81)                                 P := Pred(P);
(82)                            End;
```

Bylaag B.6
EDITLN.PAS

```
(83)                    ^Y    : Begin
(84)                                  Write(' ':Len);
(85)                                  Delete(S,P+1,Len);
(86)                            End;
(87)       End;  {of case}
(88)       First := false;
(89)       Until UpCase(Ch) in Term;
(90)       P := Length(S);
(91)       For I := (P+1) to Len do Insert(' ',S,I);
(92)       GotoXY(X+P,Y);
(93)       Write('' :Len-P);
(94)       If (Ch = Esc) then EditFin := True;
(95)       If (Ch = F2) then EditSave := True;
(96)  End;

(97)  End.
```

Bylaag B.7

GETKEY.PAS

```
(1)   {*******************************************************}
(2)   {UNIT:    GETKEY PRESSED OR MOUSE ACTION                 }
(3)   {PURPOSE: To get the key pressed on the keyboard or      }
(4)   {         the mouse action performed & return the result.}
(5)   {*******************************************************}

(6)   UNIT GetKey;
(7)   INTERFACE
(8)   Uses Dos,Crt;
(9)   Const                                          {Key definitions}
(10)      MouseUp      = #128;
(11)      MouseDown    = #129;
(12)      MouseLeft    = #130;
(13)      MouseRight   = #131;
(14)      MouseEsc     = #132;
(15)      MouseEnter   = #133;
(16)      Bs           = #8;                       {Backspace}
(17)      Enter        = #13;                      {Enter key}
(18)      Esc          = #27;                      {ESC key}
(19)      CurDown      = #208;                     {Down arrow}
(20)      CurUp        = #200;                     {Up arrow}
(21)      CurRight     = #205;                     {Right arrow}
(22)      CurLeft      = #203;                     {Left arrow}
(23)      HomeKey      = #199;
(24)      EndKey       = #207;
(25)      DelKey       = #211;
(26)      PgDn         = #209;
(27)      PgUp         = #201;
(28)      Nulkey       = #0;
(29)      F2           = #188;                     {Save key}

(30)  Type
(31)     Button = (NoB,LeftB,RightB,BothB);       {Mouse buttons pressed}

(32)  Var
(33)    Reg       : Registers;                     {DOS register variable}
(34)    MouseInst : Boolean;                       {Mouse installed?}
(35)    Horiz_Sen : Integer;                       {Horizontal sensitivity}

(36)  Function  SoundError : Char;                 {Beep and return NULL key}
(37)  Function  Interrupt_Loaded : Boolean;        {Mouse interrupt handler loaded?}
(38)  Function  Mouse_Installed : Boolean;         {Is mouse present?}
(39)  Procedure Show_Mouse_Cursor;                 {Show mouse cursor - use if needed}
(40)  Procedure Hide_Mouse_Cursor;                 {Hide mouse cursor}
(41)                                               {Get mouse action}
(42)  Procedure Get_Mouse_Action(Var But : button;
(43)                             Var Hor,Ver: Integer);
```

Bylaag B.7

GETKEY.PAS

```
(44)
(45)   Procedure Move_Mouse(Hor,Ver: Integer);                {Move mouse position}
(46)   Function  GetKeyInput : Char;                          {Which key to return?}

(47)   IMPLEMENTATION

(48)   {*******************************}
(49)   {Beep on error and return char #0}
(50)   {*******************************}
(51)   Function SoundError : Char;
(52)   Begin
(53)       Sound(1000);
(54)       Delay(200);
(55)       Nosound;
(56)       SoundError := Nulkey;
(57)   End;

(58)   {************************}
(59)   {Mouse int handler active?}
(60)   {************************}
(61)   Function Interrupt_loaded : Boolean;
(62)   Begin
(63)       Reg.Ax := 0;
(64)       Intr($33,Reg);
(65)       Interrupt_Loaded :=  (Reg.Ax <> 0);
(66)   End;

(67)   {************************}
(68)   {Mouse present and active?}
(69)   {************************}
(70)   Function Mouse_Installed:Boolean;
(71)   Begin
(72)      If Memw[$0000:$00CC] = 0 then
(73)         Mouse_Installed := False              {Don't call interrupt If vector is zero}
(74)      Else
(75)         Mouse_Installed := Interrupt_loaded;
(76)   End;

(77)   {*****************************}
(78)   {Make mouse text cursor visible}
(79)   {*****************************}
(80)   Procedure Show_Mouse_Cursor;
(81)   Begin
(82)      Reg.Ax := 1;
(83)      Intr($33,Reg);
(84)   End;
```

Bylaag B.7

GETKEY.PAS

```
(85)  {*******************************}
(86)  {Take mouse text cursor off screen}
(87)  {*******************************}
(88)  Procedure Hide_Mouse_Cursor;
(89)  Begin
(90)     Reg.Ax := 2;
(91)     Intr($33,Reg);
(92)  End;

(93)  {*********************************************}
(94)  {Return mouse action in enumerated type BUTTON}
(95)  {*********************************************}
(96)  Procedure Get_Mouse_Action(Var But: Button;
(97)                           Var Hor,Ver: Integer);
(98)  Begin
(99)     with Reg do
(100)    Begin
(101)       Ax := 3;
(102)       Intr($33,Reg);
(103)       Hor := Cx div 8;
(104)       Ver := Dx div 8;
(105)       {$B+}                                {Complete boolean evaluation}
(106)       If ((Bx and $1) <> $1)  and  ((Bx and $2) <> $2) then
(107)       Begin
(108)          But := NoB;
(109)          exit;
(110)       End;
(111)       If ((Bx and $1) = $1)  and   ((Bx and $2) = $2) then
(112)          But := BothB
(113)       Else
(114)       Begin
(115)          If (Bx and $1) = $1 then
(116)             But := LeftB
(117)          Else
(118)             But := RightB;
(119)       End;
(120)       {$B-}
(121)    End; {with}
(122) End;

(123) {*********************}
(124) {Move mouse to Hor,Ver}
(125) {*********************}
(126) Procedure Move_Mouse(Hor,Ver: Integer);
(127) Begin
(128)    Reg.Ax := 4;
(129)    Reg.Cx := Pred(Hor*8);
(130)    Reg.Dx := Pred(ver*8);
```

Bylaag B.7

GETKEY.PAS

```
(131)     Intr($33,Reg);
(132)  End;

(133)  {***************}
(134)  {Mouse released? }
(135)  {***************}
(136)  Function Mouse_Released(Button : Integer) : Boolean;
(137)  Begin
(138)     Reg.Ax := 6;
(139)     Reg.Bx := Button;
(140)     Intr($33,Reg);
(141)     Mouse_Released := (Reg.BX > 0);
(142)  End;

(143)  {***************}
(144)  {Mouse pressed? }
(145)  {***************}
(146)  Function Mouse_Pressed(Button:Integer):Boolean;
(147)  Begin
(148)     Reg.Ax := 5;
(149)     Reg.Bx := Button;
(150)     Intr($33,Reg);
(151)     Mouse_Pressed := (Reg.BX > 0);
(152)  End;

(153)  {***************************}
(154)  {Get key/mouse button pressed}
(155)  {***************************}
(156)  Function GetKeyInput : Char;
(157)  Const
(158)  H = 40;
(159)  V = 13;
(160)  Var
(161)   Action, Finish  : Boolean;
(162)   Hor, Ver        : Integer;
(163)   B               : Button;
(164)   Ch              : Char;

(165)  {*****************************************}
(166)  {Keypressed function using INT 21H         }
(167)  {NOTE: THIS IS NEEDED SO THAT INT28H CAN OCCUR}
(168)  {      PASCAL USES INT16H NORMALLY AND IT DOES}
(169)  {      NOT ALLOW INT28H TO OCCUR.          }
(170)  {*****************************************}
(171)  Function  Keypressed : Boolean;
(172)  Begin
(173)      Reg.AH := $0b;
(174)      MsDos(Reg);
```

Bylaag B.7

GETKEY.PAS

```
(175)      Keypressed := (Reg.AL = $FF);
(176)  End;

(177)  {*****************************************}
(178)  {Readkey function using INT 21H            }
(179)  {NOTE: THIS IS NEEDED SO THAT INT28H CAN OCCUR}
(180)  {      PASCAL USES INT16H NORMALLY AND IT DOES}
(181)  {      NOT ALLOW INT28H TO OCCUR.          }
(182)  {*****************************************}
(183)  Function Readkey : Char;
(184)  Begin
(185)      Reg.AH := $07;
(186)      MsDos(Reg);
(187)      Readkey := Char(Reg.AL);
(188)  End;

(189)  Begin
(190)     Finish := False; Action := False;
(191)     B := NoB;
(192)     If (MouseInst) then Move_Mouse(H,V);     {Logically put mouse in middle of screen}
(193)     Repeat                                   {keep checking Mouse for activity until keypressed}
(194)          If (MouseInst) then
(195)          Begin
(196)              Get_Mouse_Action(B,Hor,Ver);
(197)              Case B of
(198)              LeftB : Begin
(199)                          Ch := MouseEnter;
(200)                          Finish := True;
(201)                          Delay(200);
(202)                          Repeat
(203)                          Until Mouse_Pressed(0) = False; {absorb}
(204)                      End;
(205)              RightB: Begin
(206)                          Ch := MouseEsc;
(207)                          Finish := True;
(208)                          Delay(200);
(209)                          Repeat
(210)                          Until Mouse_Pressed(1) = False; {absorb}
(211)                      End;
(212)              End; {case}

(213)               If (Ver - V) > 1 then
(214)              Begin
(215)                  Ch := MouseDown;
(216)                  Finish := True;
(217)              End
(218)              Else
(219)                 If (V - Ver) > 1 then
```

Bylaag B.7
GETKEY.PAS

```
(220)                Begin
(221)                    Ch := MouseUp;
(222)                    Finish := True;
(223)                End
(224)                Else
(225)                   If (Hor - H) > Horiz_Sen then
(226)                   Begin
(227)                       Ch := MouseRight;
(228)                       Finish := True;
(229)                   End
(230)                   Else
(231)                      If (H - Hor) > Horiz_Sen then
(232)                      Begin
(233)                          Ch := MouseLeft;
(234)                          Finish := True;
(235)                      End;
(236)          End;
(237)          If Keypressed or Finish then Action := True;
(238)    Until Action;

(239)    While (Not Finish) do                    {Determine key pressed}
(240)    Begin
(241)        Finish := True;
(242)        Ch := ReadKey;
(243)        If (Ch = Nulkey) then
(244)        Begin
(245)            Ch := ReadKey;
(246)            Case Ord(Ch) of
(247)            15,
(248)            16..25,
(249)            30..38,
(250)            44..50,
(251)            59..68,
(252)            71..73,
(253)            75,77,
(254)            79..127 : Ch := Chr(Ord(Ch) + 128);
(255)            128..140: Ch := Chr(Ord(Ch) + 6);
(256)            Else      Finish := False;
(257)            End;  {case}
(258)        End;
(259)    End;
(260)    GetKeyInput := Ch;                       {Return key pressed or mouse action}
(261) End;

(262) Begin
(263)    MouseInst := Mouse_Installed;            {Is mouse installed?}
(264)    If MouseInst then Horiz_Sen := 3;        {Set mouse to be less sensitive}
(265) End.
```

Bylaag B.7
GETKEY.PAS

```
(220)                Begin
(221)                    Ch := MouseUp;
(222)                    Finish := True;
(223)                End
(224)                Else
(225)                   If (Hor - H) > Horiz_Sen then
(226)                   Begin
(227)                       Ch := MouseRight;
(228)                       Finish := True;
(229)                   End
(230)                   Else
(231)                      If (H - Hor) > Horiz_Sen then
(232)                      Begin
(233)                          Ch := MouseLeft;
(234)                          Finish := True;
(235)                      End;
(236)          End;
(237)          If Keypressed or Finish then Action := True;
(238)     Until Action;

(239)     While (Not Finish) do                    {Determine key pressed}
(240)     Begin
(241)         Finish := True;
(242)         Ch := ReadKey;
(243)         If (Ch = Nulkey) then
(244)         Begin
(245)             Ch := ReadKey;
(246)             Case Ord(Ch) of
(247)             15,
(248)             16..25,
(249)             30..38,
(250)             44..50,
(251)             59..68,
(252)             71..73,
(253)             75,77,
(254)             79..127 : Ch := Chr(Ord(Ch) + 128);
(255)             128..140: Ch := Chr(Ord(Ch) + 6);
(256)             Else      Finish := False;
(257)             End;  {case}
(258)         End;
(259)     End;
(260)     GetKeyInput := Ch;                       {Return key pressed or mouse action}
(261) End;

(262) Begin
(263)     MouseInst := Mouse_Installed;            {Is mouse installed?}
(264)     If MouseInst then Horiz_Sen := 3;        {Set mouse to be less sensitive}
(265) End.
```

Bylaag B.8

SCRNUTIL.PAS

```
(1)   {*************************************}
(2)   {UNIT:    SCREEN UTILITIES              }
(3)   {PURPOSE: Screen utilities and functions }
(4)   {         to be used by program segments.}
(5)   {*************************************}

(6)   UNIT ScrnUtil;
(7)   INTERFACE
(8)   Uses Crt,Dos;

(9)   Const
(10)      MaxScrn = 10;                          {Max of 10 screens to be saved}

(11)  Type
(12)     Str80 = String[80];                     {String of 80 characters}

(13)     ColorRecord = Record                    {Screen colour structure}
(14)                         TxtFg,              {Text foreground colour}
(15)                         TxtBg,              {Text background colour}
(16)                         WinFg,              {Window foreground colour}
(17)                         WinBg,              {Window background colour}
(18)                         TopicFg,            {Menu topic foreground colour}
(19)                         TopicBg,            {Menu topic background colour}
(20)                         HlFg,               {Menu highlighted topic foreground colour}
(21)                         HlBg    : Byte;     {Menu highlighted topic background colour}
(22)                   End;

(23)     ScrnArray = Array[0..3999] of Byte;     {4000 bytes reserved to save text screens - }
(24)                                             {= 25 rows * 80 columns -> }
                                          (25)       for character on screen, 2000 for attribute}


(26)  Var
(27)    ScreenAddr,                              {Base address of screen}
(28)    ScrnTxtAttr : Word;                      {Current text attribute in use}
(29)                                             {Max 10 screens can be saved during program execution}
(30)    ScrnPtr     : Array[1..MaxScrn] of ScrnArray;
(31)    ScrnNo,                                  {Screen number that is saved?}
(32)    Ccol,Crow   : Byte;                      {X,Y coordinates of active screen}

(33)    Reg         : Registers;                 {DOS registers variable}
(34)    Colors      : ColorRecord;               {Colour structure variable}


(35)  Procedure ColorSettings;                    {Set initial screen colours}
(36)  Procedure SaveScreen;                       {Save screen to memory buffers}
(37)  Procedure RestoreScreen;                    {Restore saved screen from memory}
```

Bylaag B.8

SCRNUTIL.PAS

```
(38)  Procedure HideCursor(Top,Bot : Byte);        {Take cursor off screen}
(39)  Procedure RestCursor;                        {Put cursor back on screen}
(40)  Procedure SetEnterResident;                  {Write CR to keyboard buffer}
(41)  Procedure SetTxtColor(FgCol,BgCol : Byte);   {Set text color - foreground, background}
(42)                                               {Write text string to specific location}
(43)                                               {on screen with specific colours}
(44)  Procedure WriteAt(Xcor,Ycor,FgCol,BgCol : Byte; S : Str80);
(45)                                               {Center text on screen}
(46)  Procedure WriteCenter(Y,FgCol,BgCol : Byte; S : Str80);
(47)  Procedure ClearWindow(X1,Y1,X2,Y2 : Byte);   {Clear active window specified}
(48)                                               {Draw a box on the screen}
(49)  Procedure DrawBox(X1,Y1,X2,Y2,FgCol,BgCol : Byte);
(50)  Procedure ErrorMsg(Msg : String);            {Display an error message}
(51)                                               {Display any message with a delay or not}
(52)  Procedure DispMessage(xDelay : Boolean; Y,FgCol,BgCol : Byte; Msg : String);


(53)  IMPLEMENTATION


(54)  {************************************}
(55)  {Init screen colours at program startup}
(56)  {************************************}
(57)  Procedure ColorSettings;

(58)     Function ColorScreen : Boolean;           {Determine if a colour screen is used}
(59)     Begin                                     {Determine screen base address}
(60)          If (Mem[0000:1040] and 48) <> 48 then ScreenAddr := $B800
(61)          Else ScreenAddr := $B000;
(62)          ColorScreen := (ScreenAddr = $B800);
(63)     End;

(64)  Begin
(65)      If ColorScreen then                      {Set screen colours for colour screen}
(66)      With Colors do
(67)      Begin
(68)           TxtFg   := Yellow;
(69)           TxtBg   := Black;
(70)           WinFg   := Cyan;
(71)           WinBg   := Black;
(72)           TopicFg := White;
(73)           TopicBg := Black;
(74)           HlFg    := Cyan;
(75)           HlBg    := LightGreen;
(76)      End
(77)      Else
(78)      With Colors do                           {Set screen colours for monochrome screen}
(79)      Begin
```

Bylaag B.8
SCRNUTIL.PAS

```
(80)            TxtFg := White;
(81)            TxtBg := Black;
(82)            WinFg := White;
(83)            WinBg := Black;
(84)            TopicFg := White;
(85)            TopicBg := Black;
(86)            HlBg    := Black;
(87)            HlFg    := White;
(88)        End;
(89)    End;

(90)    {************************************************}
(91)    {Save current screen to memory pointed to by ScrnNo}
(92)    {************************************************}
(93)    Procedure SaveScreen;
(94)    Begin
(95)        If (ScrnNo >= MaxScrn) then  ErrorMsg('TOO MANY SCREENS SAVED ALREADY !!')
(96)        Else
(97)        Begin
(98)            Inc(ScrnNo);
(99)            ScrnTxtAttr := TextAttr;          {Save current text attribute}
(100)           Ccol := WhereX;                   {Save x & y coordinates}
(101)           Crow := WhereY;
(102)                                             {Save screen to memory}
(103)           Move(Mem[ScreenAddr:0000],ScrnPtr[ScrnNo],4000);
(104)       End;
(105)   End;

(106)   {**************************************}
(107)   {Restore previous saved screen from memory}
(108)   {**************************************}
(109)   Procedure RestoreScreen;
(110)   Begin
(111)       Move(ScrnPtr[ScrnNo],Mem[ScreenAddr:0000],4000);
(112)       Dec(ScrnNo);                          {Screen number - 1}
(113)       Window(1,1,80,25);
(114)       GotoXY(Ccol,Crow);                    {Restore X & Y coordinates}
(115)       TextAttr := ScrnTxtAttr;              {Restore text attribute}
(116)   End;

(117)   {**********************}
(118)   {Take cursor off screen}
(119)   {**********************}
(120)   Procedure HideCursor(Top,Bot : Byte);
(121)   Begin
(122)       With Reg do
(123)           Begin
(124)               AX := $0100;
```

Bylaag B.8
SCRNUTIL.PAS

```
(125)                CX := ((Top Shl 8) + Bot);
(126)                Intr($10,Reg);                {Hide cursor}
(127)             End;
(128) End;

(129) {************************}
(130) {Put cursor back on screen}
(131) {************************}
(132) Procedure RestCursor;
(133) Begin
(134)     With Reg do
(135)     Begin
(136)          If LastMode = Mono then
(137)             Begin
(138)           AH := 1;
(139)                 CH := 11;
(140)                 CL := 12;
(141)             End
(142)   Else
(143)      Begin
(144)           AH := 1;
(145)                 CH := 6;
(146)                 CL := 7;
(147)      End;
(148)    End;
(149)    Intr($10,Reg);                            {Restore cursor}
(150) End;

(151) {***************************}
(152) {Write CR to keyboard buffer}
(153) {***************************}
(154) Procedure SetEnterResident;
(155) Begin
(156)     With Reg do
(157)         Begin
(158)              Ah := $05;
(159)              Ch := 13;
(160)              Cl := 13;
(161)              Intr($16,Reg);
(162)         End;
(163) End;

(164) {*************************************************************************}
(165) {Set text colour to FgCol = foreground colour, BgCol = background colour}
(166) {*************************************************************************}
(167) Procedure SetTxtColor(FgCol,BgCol : Byte);
(168) Begin
(169)     TextColor(FgCol);
```

Bylaag B.8

SCRNUTIL.PAS

```
(170)     TextBackground(BgCol);
(171) End;

(172) {*************************************}
(173) {Write text string to screen at Xcor,Ycor}
(174) {*************************************}
(175) Procedure WriteAt(Xcor,Ycor,FgCol,BgCol : Byte; S : Str80);
(176) Begin
(177)     GotoXY(Xcor,Ycor);
(178)     SetTxtColor(FgCol,BgCol);
(179)     Write(S);
(180)     SetTxtColor(15,0);
(181) End;

(182) {*********************************}
(183) {Center text string on screen at row Y}
(184) {*********************************}
(185) Procedure WriteCenter(Y,FgCol,BgCol : Byte; S : Str80);
(186) Var
(187)   Xcor : Byte;
(188) Begin
(189)     Xcor := (80 - Length(S)) div 2;
(190)     WriteAt(Xcor,Y,FgCol,BgCol,S);
(191) End;

(192) {*****************************************}
(193) {Clear active window specified by x1,y1,x2,y2}
(194) {*****************************************}
(195) Procedure ClearWindow(X1,Y1,X2,Y2 : Byte);
(196) Var
(197)   ActX1,ActY1,ActX2,ActY2 : Byte;
(198) Begin
(199)     ActX1 := Lo(WindMin)+1;              {Save actvice screen coordinates}
(200)     ActY1 := Hi(WindMin)+1;
(201)     ActX2 := Lo(WindMax)+1;
(202)     ActY2 := Hi(WindMax)+1;
(203)     SetTxtColor(15,0);
(204)     Window(X1,Y1,X2,Y2);
(205)     Clrscr;
(206)     Window(ActX1,ActY1,ActX2,ActY2);     {Restore active screen coordinates}
(207) End;
```

Bylaag B.8

SCRNUTIL.PAS

```
(208) {*************************************************}
(209) {Draw a framed box on the screen                  }
(210) {x1,y1,x2,y2 = screen coordinates                 }
(211) {FgCol = foreground colour, BgCol = background colour}
(212) {*************************************************}
(213) Procedure DrawBox(X1,Y1,X2,Y2,FgCol,BgCol : Byte);
(214) Const
(215)     UlCor = 'Õ';                          {Upper left corner character}
(216)     UrCor = ' ';                          {Upper right corner character}
(217)     LlCor = 'Ô';                          {Lower left corner character}
(218)     LrCor = '¾';                          {Lower right corner character}
(219)     HorLn = 'Í';                          {Horizontal line character}
(220)     VerLn = '³';                          {Vertical line character}
(221) Var
(222)   A : Byte;
(223) Begin                                      {Draw framed box}
(224)     ClearWindow(X1,Y1,X2,Y2);
(225)     Window(1,1,80,25);
(226)     WriteAt(X1,Y1,FgCol,BgCol,UlCor);
(227)     WriteAt(X2,Y1,FgCol,BgCol,UrCor);
(228)     WriteAt(X1,Y2,FgCol,BgCol,LlCor);
(229)     WriteAt(X2,Y2,FgCol,BgCol,LrCor);
(230)     For A := (X1+1) to (X2-1) do
(231)     Begin
(232)          WriteAt(A,Y1,FgCol,BgCol,HorLn);
(233)          WriteAt(A,Y2,FgCol,BgCol,HorLn);
(234)     End;
(235)     For A := (Y1+1) to (Y2-1) do
(236)     Begin
(237)          WriteAt(X1,A,FgCol,BgCol,VerLn);
(238)          WriteAt(X2,A,FgCol,BgCol,VerLn);
(239)     End;
(240) End;

(241) {*****************************}
(242) {Display error message on screen}
(243) {*****************************}
(244) Procedure ErrorMsg(Msg : String);
(245) Var
(246)   ActX1,ActY1,ActX2,ActY2,I : Byte;
(247)   Wait                      : Char;
(248) Begin
(249)     ActX1 := Lo(WindMin)+1;             {Save current screen attributes}
(250)     ActY1 := Hi(WindMin)+1;
(251)     ActX2 := Lo(WindMax)+1;
(252)     ActY2 := Hi(WindMax)+1;
(253)     SaveScreen;                         {Save current screen contents}
(254)     Window(1,1,80,25);
```

Bylaag B.8

SCRNUTIL.PAS

```
(255)      WriteCenter(24,LightRed,Black,Msg);     {Display error message}
(256)      Sound(1000);                            {Beep}
(257)      Delay(400);
(258)      Nosound;
(259)      WriteCenter(25,White,Black,' press any key to continue ');
(260)      Wait := Readkey;                        {Wait for key pressed}
(261)      RestoreScreen;                          {Restore saved screen}
(262)      Window(ActX1,ActY1,ActX2,ActY2);
(263)  End;

(264)  {*********************************************************************}
(265)  {Display any message on screen - Y = row on screen, FgCol,BGcol = colours}
(266)  {*********************************************************************}
(267)  Procedure DispMessage(xDelay : Boolean; Y,FgCol,BgCol : Byte; Msg : String);
(268)  Var
(269)    ActX1,ActY1,ActX2,ActY2,Xcor : Byte;
(270)  Begin
(271)      ActX1 := Lo(WindMin)+1;
(272)      ActY1 := Hi(WindMin)+1;
(273)      ActX2 := Lo(WindMax)+1;
(274)      ActY2 := Hi(WindMax)+1;
(275)      Window(1,1,80,25);
(276)      Xcor := ((80-Length(Msg)) div 2) - 1;  {Center text string on screen}
(277)      Drawbox(Xcor,Y-1,Xcor+Length(Msg)+1,Y+1,FgCol,BgCol);
(278)      WriteCenter(Y,FgCol,BgCol,Msg);
(279)      If Xdelay then
(280)      Begin
(281)           Delay(1000);
(282)           ClearWindow(Xcor,Y-1,Xcor+Length(Msg)+1,Y+1);
(283)      End;
(284)      Window(ActX1,ActY1,ActX2,ActY2);
(285)  End;

(286)  Begin
(287)      ColorSettings;                          {Do initial colour settings}
(288)      ScrnNo := 0;                            {No of screens saved = 0}
(289)  End.
```

Bylaag C

8031NODE.C

```
(1)   /*********************************************************/
(2)   /* Program: 8031node controller                          */
(3)   /*                                                       */
(4)   /* Pupose:  To be the BIOS program for the 8031 node and */
(5)   /*          to perform serial I/O and database management */
(6)   /*          for data to/from master computer.            */
(7)   /*                                                       */
(8)   /* Author:  Wickus de Koker                              */
(9)   /*********************************************************/

(10)  /* PROGRAM DECLARATIONS */
(11)                                          /* linker directives */
(12)  #pragma CODE DEBUG SYMBOLS              /* include debug & symbol information in listfile generation */
(13)  #pragma ROM (LARGE)                     /* large memory model for ROM */
(14)  #pragma PAGELENGTH (66)                 /* default to 66 lines @page for printing list file */
(15)  #pragma OPTIMIZE(5)                     /* optimize completely (levels 1 to 4 incl) */

(16)                                          /* prototypes needed for program */
(17)  #include <reg51.h>                      /* special funtion register 8051 series */
(18)  #include <string.h>                     /* string and memory function library */
(19)  #include <stdio.h>                      /* standard ANSI i/o library */
(20)  #include <stdlib.h>                     /* standard library */
(21)  #include <ctype.h>                      /* character function library */

(22)                                          /* data communication id words */
(23)  #define  ctrlword  0x89                /* 8255 control word */
(24)  #define  prg       0x03                /* master want to program node */
(25)  #define  eot       0x04                /* end of transmission id */
(26)  #define  enq       0x05                /* result enquiry */
(27)  #define  ack       0x06                /* acknowledgement signal */
(28)  #define  lf        0x0a                /* line feed character */
(29)  #define  cr        0x0d                /* carriage return character */
(30)  #define  msg       0x02

(31)  #define  max_data  1310                /* maximum data buffer size = 24 records + 3 extra bytes */

(32)  #define  false 0                       /* boolean equates for false and true */
(33)  #define  true  1

(34)  #define XBYTE ((unsigned char *) 0x20000L)  /* prototype for absolute address access */

(35)  #define porta   XBYTE [0xc000]         /* 8255_porta = byte at $c000 */
(36)  #define portb   XBYTE [0xc100]         /* 8255_portb = byte at $c100 */
(37)  #define portc   XBYTE [0xc200]         /* 8255_portc = byte at $c200 */
(38)  #define ctrlreg XBYTE [0xc300]         /* 8255_ctrlreg = byte at $c300 */
```

Bylaag C

8031NODE.C

```
(39)  enum ip_op  {input,output};                /* enumerated type to indicate pin type */
(40)  enum on_off {on,off};                      /* enumerated type to indicate pin status */

(41)  struct node_config {
(42)            unsigned char port[4];          /* 8255 port pin number  */
(43)            enum ip_op ptype;               /* input or output pin ?  */
(44)                                             /* 1->5=on/6->10=off times for pin */
(45)            unsigned char times[10][6];
(46)            enum on_off status;             /* is pin turned on or off?  */
(47)          };


(48)                                             /* node_arr = 24 records of structure node_config */
(49)  xdata struct node_config node_arr[24];


(50)  sbit de            = P1^5;                 /* de = boolean at p1.5 */
(51)  sbit re            = P1^6;                 /* re = boolean at p1.6 */
(52)  sbit ppi_reset_pin = P1^7;                 /* 8255 reset = boolean at p1.7 */
(53)  sbit led           = P3^4;                 /* led = boolean at p3.4 */


(54)  char xdata data_buffer[max_data];          /* data_buffer to store received data */

(55)  unsigned int xdata hour,                   /* internal clock - hour_counter */
(56)                     min,                    /* internal clock - minute_counter */
(57)                     sec,                    /* internal clock - second_counter */
(58)                     rec_no,                 /* number of records in database received */
(59)                     prev_sec;


(60)  int  xdata micro_sec;                      /* internal clock - micro_second_counter */


(61)  char xdata unique_addr,                    /* unique identification address of node */
(62)            ser_in;                          /* serial_in data variable */

(63)  int  xdata data_ptr;                       /* data_pointer to data_buffer */

(64)  bit  idata capture,                        /* node need to capture reveived data */
(65)             rxd_flag,                       /* this node has been addressed by master computer */
(66)             address_flag,                   /* node address correctly received */
(67)             addr_correct,
(68)             prog_flag,                      /* master wants to program node */
(69)             result_flag,                    /* master requests results from node */
(70)             no_more_requests;               /* end of data transmission */
(71)  char xdata port_value[3];                  /* structure to get value to send to 8255 I/O ports */
```

Bylaag C

8031NODE.C

```
(72)  /* INTERRUPT HANDLERS */

(73)  timer0() interrupt 1 using 1              /* timer0 interrupt handler - using resiterbank 1 */
(74)  {
(75)    ++micro_sec;                            /* micro_sec = 333.33æ * 11.0592 MHz = 3686  */
(76)    if (micro_sec == 3686)                  /* one second later ? */
(77)    {                                       /* yes ! ... */
(78)       micro_sec = 0;                       /* reset micro_second counter */
(79)       ++sec;                               /* seconds + 1 */
(80)       if (sec == 60)                       /* one minute later ? */
(81)       {                                    /* yes ! ... */
(82)          sec = 0;                          /* reset second_counter */
(83)          ++min;                            /* minutes + 1 */
(84)          if (min == 60)                    /* one hour later ? */
(85)          {                                 /* yes ! ... */
(86)             min = 0;                       /* reset minute_counter */
(87)             ++hour;                        /* hours + 1*/
(88)             if (hour == 24) hour = 0;
(89)          }
(90)       }
(91)    }
(92)    TF0 = false;                            /* resert timer0 interrupt flag */
(93)  }

(94)  serial() interrupt 4 using 2              /* serial interrupt handler - using registerbank 2 */
(95)  {
(96)    if (RI)                                 /* receive interrupt occurred ? */
(97)       {                                    /* yes ! ... */
(98)          RI = false;                       /* clear rxd interrupt flag */
(99)          ser_in = SBUF;                    /* save received character in variable "ser_in" */
(100)                                           /* if end of transmission => no more requests */
(101)         if (ser_in == eot) no_more_requests = true;
(102)                                           /* node address by master ? */
(103)         if ((ser_in & 0x80) == 0x80) rxd_flag = true;
(104)                                           /* yes? then node must receive data */
(105)         if (capture)                      /* dump serial data to data buffer */
(106)         {
(107)            if (ser_in != eot)             /* yes!. if not end_of_transmission then .. */
(108)            {                              /* put next character in databuffer */
(109)               data_buffer[data_ptr] = ser_in;
(110)               ++data_ptr;                /* data_pointer + 1 */
(111)            }
(112)         }
(113)      }
(114) }
```

Bylaag C

8031NODE.C

```
(115) /* PROGRAM FUNCTIONS */

(116) #pragma OPTIMIZE(4)                          /* convert integer values to its ASCII string equivalent */
(117) char xdata *int_to_str(int int_val) reentrant
(118) {
(119)   char xdata string[5],                      /* string to save reversed ASCII values */
(120)              string1[5];                     /* ASCII string to be returned */

(121)   int  xdata qut,                            /* quotient of division */
(122)              rem,                            /* remainder of division */
(123)              pointr,                         /* pointer to offset in conversion string */
(124)              temp;                           /* temp value to hold qut after each div cycle */

(125)                                              /* init variables */
(126)   qut = 1;
(127)   rem = 0;
(128)   pointr = 0;
(129)   temp = int_val;

(130)   while (qut != 0)                           /* divide while qut <> 0 */
(131)   {
(132)      qut = (temp/10);                        /* get quotient */
(133)      rem = (temp%10);                        /* get remainder */
(134)      temp = qut;                             /* store qut in temp */

(135)      string[pointr] = rem+0x30;              /* get ASCII by adding 30hex */
(136)      ++pointr;                               /* inc pointer position in string */
(137)   }

(138)   rem = 0;                                   /* reverse ASCII in string to normal seq in string1 */
(139)   for (temp = (pointr-1); temp >= 0; temp--)
(140)   {
(141)      string1[rem] = string[temp];            /* temp used as pointer in reversed string */
(142)      rem++;                                  /* rem used as normal counter */
(143)   }
(144)   string1[rem] = 0x00;                       /* insert null terminator to end of string */
(145)   return(string1);                           /* return pointer to ASCII string */
(146) }
(147) #pragma OPTIMIZE(5)

(148) void reset_misc_variables(void)              /* initialize variables at program startup */
(149) {
(150)   char xdata i,j,k;

(151)   hour      = 0;
(152)   min       = 0;
(153)   sec       = 0;
(154)   micro_sec = 0;
```

Bylaag C

8031NODE.C

```
(155)    rec_no     = 0;
(156)    data_ptr   = 0;
(157)    ser_in     = 0;
(158)    prev_sec   = 0;

(159)    capture          = false;              /* no data to capture yet  */
(160)    rxd_flag         = false;              /* no data received yet  */
(161)    address_flag     = false;              /* node not addressed by main computer,  */
(162)    addr_correct     = false;              /* no address polled to,  */
(163)    prog_flag        = false;              /* not programmed by main computer,  */
(164)    result_flag      = false;              /* not polled for results by main computer and   */
(165)    no_more_requests = false;              /* no requests from main computer yet.  */

(166)    for (i=0; i<=2; i++) port_value[i] = 0;  /* clear port values for 8255 I/O pins */

(167)    for (i=0; i<=23; i++)                  /* reset database structure values */
(168)    {
(169)       for (j=0; j<=2; j++) node_arr[i].port[j] = 0x00;
(170)       node_arr[i].port[3] = 0x0U;
(171)       node_arr[i].ptype = output;

(172)       for (j=0; j<=9; j++)
(173)           {
(174)           for (k=0; k<=4; k++) node_arr[i].times[j][k] = 0x00;
(175)           node_arr[i].times[j][5] = 0x00;
(176)           }
(177)       node_arr[i].status = off;
(178)    }
(179) }

(180) void get_unique_addr(void)               /* determine unique node id and save it */
(181) {
(182)    unique_addr = (P1 & 0x1f);            /* use only 5 LSBits */
(183) }

(184) void init_serial_interrupt(void)         /* initialize serial interface */
(185) {
(186)    re   = false;                         /* enable RS485 receiver */
(187)    de   = false;                         /* disable RS485 transmitter */
(188)    TMOD = (TMOD | 0x20);                 /* TIMER1 = 8 bit auto reload timer  */
(189)    SCON = 0x50;                          /* serial mode 1, REN=1  */
(190)    TH1  = -48;                           /* -24=2400 baud ... -48=1200bd  */
(191)    PCON = 0x80;                          /* if SCON is set to 1  */
(192)    TR1  = true;                          /* enable timer 1  */
(193)    TI   = true;                          /* kick-start transmitter  */
(194) }
```

Bylaag C

8031NODE.C

```
(195) void init_timer0_interrupt(void)          /* initialize timer0 interrupt */
(196) {
(197)   TR0  = true;                            /* enable timer 0  */
(198)   TMOD = (TMOD | 0x02);                   /* timer 0 = 8 bit auto reload timer  */
(199)   TH0  = 6;                               /* interrupt every 250 micro seconds @11.0592 MHz */
(200) }

(201) void init_8255ppi(void)
(202) {
(203)   ctrlreg = ctrlword;                     /* initialize 8255PPI for 2 simple ouput port & 1 input port */
(204) }

(205) void enable_interrupts(void) reentrant
(206) {
(207)   IE = 0x92;                              /* allow interrupts (serial & timer0) to occur */
(208) }

(209) void tx_char(char ch) reentrant           /* transmit character on RS232 port */
(210) {
(211)   char xdata i;

(212)   while (!TI);                           /* wait till txd interrupt occurs */
(213)   TI = false;                            /* clear transmit interrupt flag */
(214)   SBUF = ch;                             /* transmit character */
(215)   for (i=1; i<=200; i++);                /* wait for USART to stabilize */
(216) }

(217) void tx_character(char chr_to_tx) reentrant/* transmit single characters on RS232 port */
(218) {
(219)   char xdata i;

(220)   IE = 0x90;                             /* disable "real time" clock interruption */
(221)   re = true;                             /* disable receiver of RS485 chip (SN75175) */
(222)   de = true;                             /* enable RS485 transmitter */
(223)   for (i=1; i<=200; i++);                /* wait for USART to stabilize */
(224)   tx_char(chr_to_tx);                    /* transmit character */
(225)   for (i=1; i<=200; i++);                /* wait for USART to stabilize */
(226)   re = false;                            /* enable RS485 receiver */
(227)   de = false;                            /* disable RS485 transmitter */
(228)   enable_interrupts();                   /* allow serial & timer0 int to occur */
(229) }
(230)                                       /* transmit strings on RS232 port */
(231) void tx_string(char xdata *str_to_tx) reentrant
(232) {
(233)   int xdata i;

(234)   IE = 0x90;                             /* prevent timer0 interruption during data transmission */
(235)   re = true;                             /* disable receiver of RS485 chip (SN75175) */
```

Bylaag C

8031NODE.C

```
(236)    de = true;                                  /* enable RS485 transmitter */
(237)    str_to_tx[strlen(str_to_tx)] = 0x00;        /* ensure string is terminated by null terminator */
(238)    for (i=1; i<=200; i++);                     /* wait for USART to stabilize */
(239)    for (i=0; i<=(strlen(str_to_tx)-1); i++) tx_char(str_to_tx[i]);
(240)    for (i=1; i<=200; i++);                     /* wait for USART to stabilize */
(241)    re = false;                                 /* enable RS485 receiver */
(242)    de = false;                                 /* disable RS485 transmitter */
(243)    enable_interrupts();                        /* enable interrupts to occur */
(244) }

(245) void tx_lf_cr(void) reentrant                  /* transmit a line feed & carriage return character */
(246) {
(247)    tx_character(lf);
(248)    tx_character(cr);
(249) }

(250) char *current_time(void) reentrant             /* return the current system time as a string of 8 characters */
(251) {
(252)    char xdata time_string[9] = "00:00:00",
(253)               *temp_string;


(254)    temp_string = int_to_str(hour);
(255)    if (hour < 10) time_string[1] = temp_string[0];
(256)    else
(257)    {
(258)       time_string[0] = temp_string[0];
(259)       time_string[1] = temp_string[1];
(260)    }

(261)    temp_string = int_to_str(min);
(262)    if (min < 10) time_string[4] = temp_string[0];
(263)    else
(264)    {
(265)       time_string[3] = temp_string[0];
(266)       time_string[4] = temp_string[1];
(267)    }

(268)    temp_string = int_to_str(sec);
(269)    if (sec < 10) time_string[7] = temp_string[0];
(270)    else
(271)    {
(272)       time_string[6] = temp_string[0];
(273)       time_string[7] = temp_string[1];
(274)    }

(275)    return(time_string);
(276) }
```

Bylaag C

8031NODE.C

```
(277) void update_8255_status(void)                /* update the I/O ports' status of the 8255 PPI */
(278) {                                            /* array to determine bit set & to be set */
(279)   char code  set_array[8] = {0x01,0x02,0x04,0x08,0x10,0x20,0x40,0x80},
(280)                                              /* array to clear bit if needed */
(281)              clr_array[8] = {0xfe,0xfd,0xfb,0xf7,0xef,0xdf,0xbf,0x7f};

(282)   char xdata i, j,                           /* misc pointers */
(283)              time_now[6],                    /* current time comparison string */
(284)              arr_point,                      /* pointers to port value array structure */
(285)              bit_point,
(286)              *pres_time;

(287)   int  xdata strcmp_result;                  /* result of string comparison */


(288)   pres_time = current_time();
(289)   for (i=0; i<=4; i++) time_now[i] = pres_time[i];
(290)   time_now[5] = 0x00;                        /* null terminator insertion */

(291)   for (i=0; i<=(rec_no-1); i++)
(292)   {
(293)    if (node_arr[i].ptype == output)          /* output pin */
(294)      {
(295)         for (j=0; j<=4; j++)
(296)         {
(297)            strcmp_result = strcmp(node_arr[i].times[j],"**:**");
(298)            if (strcmp_result != 0)
(299)            {
(300)               strcmp_result = strcmp(time_now,node_arr[i].times[j]);
(301)               if (strcmp_result == 0)
(302)               {
(303)                  if (node_arr[i].status == off)
(304)                  {
(305)                        node_arr[i].status = on;
(306)                                              /* 'A'=65dec => pointer=(65-65)=0, etc.*/
(307)                        arr_point = node_arr[i].port[0] - 65;
(308)                                              /* '0'=48dec => pointer=(48-48)=0, etc.*/
(309)                     bit_point = node_arr[i].port[2] - 48;
(310)                                              /* set bit of pin to be turned on */
(311)                     port_value[arr_point] = (port_value[arr_point] || set_array[bit_point]);

(312)                  }
(313)               }
(314)            }
(315)         }

(316)         for (j=5; j<=9; j++)
(317)         {
```

Bylaag C

8031NODE.C

```
(318)             strcmp_result = strcmp(node_arr[i].times[j],"**:**");
(319)             if (strcmp_result != 0)
(320)             {
(321)                strcmp_result = strcmp(time_now,node_arr[i].times[j]);
(322)                if (strcmp_result == 0)
(323)                {
(324)                   if (node_arr[i].status == on)
(325)                   {
(326)                         node_arr[i].status = off;
(327)                                             /* 'A'=65dec => pointer=(65-65)=0, etc.*/
(328)                         arr_point = node_arr[i].port[0] - 65;
(329)                                             /* '0'=48dec => pointer=(48-48)=0, etc.*/
(330)                      bit_point = node_arr[i].port[2] - 48;
(331)                                             /* clear bit of pin to be turned off */
(332)                      port_value[arr_point] = (port_value[arr_point] && clr_array[bit_point]);
(333)                   }
(334)                }
(335)             }
(336)          }
(337)       }
(338)       else                                  /* input pin  */
(339)       for (j=0; j<=4; j++)
(340)       {
(341)          strcmp_result = strcmp(node_arr[i].times[j],"**:**");
(342)          if (strcmp_result != 0)
(343)          {
(344)             strcmp_result = strcmp(time_now,node_arr[i].times[j]);
(345)             if (strcmp_result == 0)
(346)              {
(347)                arr_point = 2;               /* provision only for portc as input port  */
(348)                                             /* read portC into port_value structure */
(349)                port_value[arr_point] = portc;
(350)                                             /* get bit to be set  */
(351)                                             /* '0'=48dec => pointer=(48-48)=0, etc.*/
(352)                bit_point = node_arr[i].port[2] - 48;
(353)                                             /* determine if pin is on or off & update pin status */
(354)                if ((port_value[arr_point] && set_array[bit_point]) == set_array[bit_point]) node_arr[i].status
   = on;
(355)                else node_arr[i].status = off;
(356)             }
(357)          }
(358)       }
(359)    }
(360)    porta = port_value[0];                   /* update portA's status from port_value structure */
(361)    portb = port_value[1];                   /* update portB's status from port_value structure */
(362) }
```

Bylaag C
8031NODE.C

```
(363)                                                   /* capture the data from the master in the data buffer */
(364)  void accept_programming_data(void) reentrant
(365)  {
(366)    char xdata sync_time[9] = "00:00:00",
(367)               temp_str[3]  = "00",          /* temp string for general use */
(368)               *time_str;                    /* current time pointer */

(369)    int  xdata i,j,k,l;


(370)    capture = true;                          /* dump incomming serial data to data buffer */
(371)    data_ptr = 0;                            /* data buffer offset set to 0 */
(372)    rec_no   = 0;                            /* no of records read from master */
(373)    led = false;                             /* light LED to indicate node is busy */

(374)    for (i=0; i<=23; i++)                    /* clear data structure of records in memory */
(375)    {
(376)       node_arr[i].port[0] = 0x00;
(377)       node_arr[i].port[1] = 0x00;
(378)       node_arr[i].port[2] = 0x00;
(379)       node_arr[i].status  = off;
(380)    }
(381)                                             /* clear data buffer */
(382)    for (i=0;i<=(max_data-1);i++) data_buffer[i] = 0x00;

(383)    while (! no_more_requests);              /* wait for eot */

(384)    capture = false;                         /* no need to dump serial data to data buffer anymore */
(385)    for (i=1;i<=200;i++);

(386)    data_buffer[data_ptr] = 0x00;            /* null terminator insertion */

(387)    j = 0;                                   /* get sync time from master from data buffer */
(388)    for (i=(data_ptr-8); i<=(data_ptr-1); i++)
(389)    {
(390)       sync_time[j] = data_buffer[i];
(391)       ++j;
(392)    }
(393)    sync_time[8] = 0x00;                     /* null terminator insertion */

(394)    temp_str[2] = 0x00;                      /* null terminator insertion */

(395)    temp_str[0] = sync_time[0];
(396)    temp_str[1] = sync_time[1];
(397)    hour = atoi(temp_str);                   /* set hour counter to sync hour */
(398)    temp_str[0] = sync_time[3];
(399)    temp_str[1] = sync_time[4];
(400)    min = atoi(temp_str);                    /* set min counter to sync min */
```

Bylaag C

8031NODE.C

```
(401)    temp_str[0] = sync_time[6];
(402)    temp_str[1] = sync_time[7];
(403)    sec = atoi(temp_str);                    /* set sec counter to sync sec */

(404)    time_str = current_time();               /* read back time sync'd */
(405)    tx_string("Time sync'd ");
(406)    tx_string(time_str);
(407)    tx_lf_cr();

(408)    tx_string("Processing received data. Please be patient...");
(409)    tx_lf_cr();

(410)    temp_str[0] = data_buffer[0];
(411)    temp_str[1] = data_buffer[1];
(412)    rec_no = atoi(temp_str);                 /* get no of records sent by master */

(413)    j = 2;
(414)    l = 6;
(415)                                             /* read port config to memory structure provided */
(416)    for (i=0; i<=(rec_no-1); i++)
(417)    {
(418)       node_arr[i].port[0] = data_buffer[j];
(419)       node_arr[i].port[1] = data_buffer[j+1];
(420)       node_arr[i].port[2] = data_buffer[j+2];

(421)       if (data_buffer[j+3] == 'O') node_arr[i].ptype = output;
(422)       else node_arr[i].ptype = input;

(423)       for (k=0; k<=9; k++)
(424)       {
(425)          node_arr[i].times[k][0] = data_buffer[l];
(426)          node_arr[i].times[k][1] = data_buffer[l+1];
(427)          node_arr[i].times[k][2] = data_buffer[l+2];
(428)          node_arr[i].times[k][3] = data_buffer[l+3];
(429)          node_arr[i].times[k][4] = data_buffer[l+4];
(430)          node_arr[i].times[k][5] = 0x00;
(431)          l += 5;
(432)       }

(433)     j += 54;
(434)       l += 4;
(435)    }
(436)    tx_string("Done !  Press any key to exit.");
(437) }
```

Bylaag C

8031NODE.C

```
(438) void send_results_to_master(void) reentrant/* send the current status results of the node to teh master */
(439) {
(440)   int  xdata i, j;

(441)   char xdata dummy[] = "0",
(442)              *rec_str;
(443)                                          /* clear data buffer */

(444)   for (i=0;i<=(max_data-1);i++) data_buffer[i] = 0x00;

(445)   tx_string("Processing results ...");
(446)   tx_lf_cr();                            /* transmit line feed & carriage return */
(447)   tx_character(ack);                     /* acknowledge that node sent intro messages */
(448)   if (rec_no == 0)
(449)   {
(450)      tx_string("No records in memory !!!");
(451)      tx_lf_cr();
(452)   }
(453)   else                                   /* put result data in data buffer */
(454)   {
(455)      rec_str = int_to_str(rec_no);

(456)      if (rec_no < 10)
(457)      {
(458)         data_buffer[0] = '0';
(459)         data_buffer[1] = rec_str[0];
(460)      }
(461)      else
(462)      {
(463)         data_buffer[0] = rec_str[0];
(464)         data_buffer[1] = rec_str[1];
(465)      }

(466)      data_ptr = 2;

(467)      for (i=0; i<=(rec_no-1); i++)
(468)      {
(469)         data_buffer[data_ptr] = node_arr[i].port[0];
(470)         ++data_ptr;
(471)         data_buffer[data_ptr] = node_arr[i].port[1];
(472)         ++data_ptr;
(473)         data_buffer[data_ptr] = node_arr[i].port[2];
(474)         ++data_ptr;

(475)         if (node_arr[i].status == on) data_buffer[data_ptr] = '1';
(476)       else data_buffer[data_ptr] = '0';
(477)         ++data_ptr;
(478)      }
```

Bylaag C

8031NODE.C

```
(479)       data_buffer[data_ptr] = 0x00;
(480)                                       /* send results to master */
(481)       for (i=0; i<=(data_ptr-1); i++) tx_character(data_buffer[i]);
(482)       tx_lf_cr();                     /* transmit line feed & carriage return */
(483)    }
(484)    tx_character(ack);                 /* send end of transfer id */
(485)    tx_string("Press <Enter> to exit.");
(486)    while (! no_more_requests);        /* wait for end of transmission char */
(487) }

(488) void stay_in_endless_loop(void) reentrant  /* re-entrant function to stay in loop */
(489) {
(490)    char code acknow_msg[] = "..... Acknowledged at ";
(491)    char xdata j,
(492)               *time;

(493)    update_8255_status();              /* update 8255PPI I/O ports' status */

(494)    if (!led) led = true;              /* flash LED */
(495)    else led = false;

(496)    for (j=1; j<=200; j++);            /* give short delay */

(497)    if (rxd_flag)                      /* master address id correct */
(498)    {                                  /* master addressed nodes for any reason ?*/
(499)       address_flag = ((ser_in & 0x80) == 0x80);
(500)       if (address_flag)
(501)       {                               /* this node's address sent by master ? */
(502)          if ((ser_in & 0x1f) == unique_addr) addr_correct = true;
(503)          address_flag = false;
(504)       }
(505)       rxd_flag = false;
(506)    }

(507)    if (addr_correct)                  /* this node was addressed by master = true */
(508)    {
(509)       addr_correct = false;
(510)       time = current_time();          /* read time of acknowledgement */
(511)       for (j=1; j<=200; j++);         /* wait for USART to stabilize  */
(512)       tx_character(ack);              /* acknowledge node presence */
(513)       tx_string(acknow_msg);
(514)       tx_string(time);

(515)       no_more_requests = false;
(516)       while (!no_more_requests)       /* wait for master to send required id's until <eot> */
(517)       {
(518)          switch(ser_in)
(519)          {                            /* master wants to program node */
```

Bylaag C

8031NODE.C

```
(520)             case prg: accept_programming_data();
(521)                       break;                  /* master requests results from node */
(522)             case enq: send_results_to_master();
(523)                       break;
(524)          }

(525)       }
(526)    }
(527)    stay_in_endless_loop();
(528) }

(529) /* START OF MAIN PROGRAM */
(530) /* NOTE: MAIN PROGRAM STARTED HERE SO THERE IS NO NEED TO REDECLARE FUNCTIONS USED GLOBALLY */

(531) void main(void)
(532) {
(533)    ppi_reset_pin = false;                     /* end of 8255 PPI reset cycle */
(534)    reset_misc_variables();                    /* init misc variables */
(535)    get_unique_addr();                         /* determine unique address id of node */

(536)    init_serial_interrupt();                   /* init serial interface */
(537)    init_timer0_interrupt();                   /* init internal "real time" clock */
(538)    init_8255ppi();                            /* init 8255 PPI for parallel I/O */
(539)    enable_interrupts();                       /* enable serial and timer0 interrupts */
(540)    stay_in_endless_loop();                    /* remain in endless program loop */
(541) }
```

LITERATUURLYS

ARON, J.D. 1974. *The Program Development Process, Part1.* Reading, Massachusetts. Addison-Wesley Publishing Company.

DETTMAN T. 1989. *DOS Programmer's Reference, 2nd Ed.* Carmel, Indiana. Que Corporation.

ELEKTOR. 1983. *Data Sheet Book.* Canterbury. Elektor Publishers Ltd.

FREEMAN, R.L. 1985. *Reference Manual for Telecommunications Engineering.* New York. John Wiley & Sons, Inc.

HOROWITZ, P. & HILL, W. 1980. *The Art of Electronics, 2nd Ed.* Cambridge. Cambridge University Press.

HELD, G. 1991. *The Complete Modem Reference.* New York. John Wiley & Sons, Inc.

INTEL. 1991a. *8-Bit Embedded Controllers.* Mt. Prospect, I.L. Intel Corporation.

INTEL. 1991b. *Embedded Applications.* Mt. Prospect, I.L. Intel Corporation.

INTEL. 1985. *iAPX 86/88,186/188 User's Manual - Programmer's Reference.* Santa Clara, C.A. Intel Corporation.

KATZIN, E. 1985. *How to Write a Really Good User's Manual.* New York. Van Nostrand Reingold Co., Inc.

LITERATUURLYS

ARON, J.D. 1974. *The Program Development Process, Part1.* Reading, Massachusetts. Addison-Wesley Publishing Company.

DETTMAN T. 1989. *DOS Programmer's Reference, 2nd Ed.* Carmel, Indiana. Que Corporation.

ELEKTOR. 1983. *Data Sheet Book.* Canterbury. Elektor Publishers Ltd.

FREEMAN, R.L. 1985. *Reference Manual for Telecommunications Engineering.* New York. John Wiley & Sons, Inc.

HOROWITZ, P. & HILL, W. 1980. *The Art of Electronics, 2nd Ed.* Cambridge. Cambridge University Press.

HELD, G. 1991. *The Complete Modem Reference.* New York. John Wiley & Sons, Inc.

INTEL. 1991a. *8-Bit Embedded Controllers.* Mt. Prospect, I.L. Intel Corporation.

INTEL. 1991b. *Embedded Applications.* Mt. Prospect, I.L. Intel Corporation.

INTEL. 1985. *iAPX 86/88,186/188 User's Manual - Programmer's Reference.* Santa Clara, C.A. Intel Corporation.

KATZIN, E. 1985. *How to Write a Really Good User's Manual.* New York. Van Nostrand Reingold Co., Inc.

PASAHOW, E.J. 1981. *Microcomputer Interfacing.* New York. McGraw-Hill, Inc.

RETTKE, M. 1990. *Practical Data Communications.* Watsonville. McGraw-Hill, Inc.

SCHILDT, H. 1990. *Turbo C/C++ : The Complete Reference.* Berkerley. McGraw-Hill, Inc.

SCHWEBER, W.L. 1988. *Data Communications.* Singapore. McGraw-Hill, Inc.

SINNEMA, W. & McGOVERN, T. 1986. *Digital, Analog, and Data Communication, 2nd Ed.* Englewood Cliffs. Prentice-Hall International, Inc.

SOMMERVILLE, I. 1982. *Software Engineering.* London. Addison-Wesley Publishing Company.

TUGAL, D.A. & TUGAL, O. 1989. *Data Transmission.* New York. McGraw-Hill, Inc.

UFFENBECK, J. 1987. *The 8086/8088 Family: Design, Programming, and Interfacing.* Englewood Cliffs. Prentice-Hall International, Inc.

VAN TASSEL, D. 1978. *Program Style, Design, Efficiency, Debugging, and Testing, 2nd Ed.* Englewood Cliffs, N.J. Prentice-Hall, Inc.

VERBURG, W.P. 1989. *Principles of Digital Systems and 8 Bit Microprocessors.* Pretoria. Dirk Robin Publishers.

ZAKS, R. & LESEA, A. 1977. *Microprocessor Interfacing Techniques, 3rd Ed.* Berkeley. Sybex, Inc.

ADDISIONELE BRONNE
(GERAADPLEEG, MAAR NIE NA VERWYS NIE)

AVOCET. 1986. *AVSIM51 8051 Family User's Manual.* Rockport, Maine. Avocet Systems, Inc.

BORLAND. 1990. *Turbo Pascal : Library Reference.* Scotts Valley, CA. Borland International, Inc.

BORLAND. 1990. *Turbo Pascal : Programmer's Guide.* Scotts Valley, CA. Borland International, Inc.

BORLAND. 1990. *Turbo Pascal : Turbo Vision Guide.* Scotts Valley, CA. Borland International, Inc.

BORLAND. 1990. *Turbo Pascal : User's Guide.* Scotts Valley, CA. Borland International, Inc.

BORLAND. 1988,1991. *Turbo Debugger Version 2.5 : User's Guide.* Scotts Valley, CA. Borland International, Inc.

BROOKES C.H.P., GROUSE P.J., JEFFERY D.R, LAWRENCE M.J. 1982. *Information Systems Design.* Sydney, Australia. Prentice-Hall Int.

DUTTON, F. 1988. *Turbo Pascal Toolbox.* San Francisco. Sybex, Inc.

HALL, D.V. 1986. *Microprocessors and Interfacing - Programming and Hardware.* McGraw-Hill. Singapore.

HARDY, J.K. 1986. *Electronic Communications Technology.* Prentice-Hall Int. Inc. Englewood Cliffs.

HORDESKI, M. 1991. *Microcomputer LANs, 2nd Ed.* Blue Ridge Summit, PA. TAB Professional and Reference Books.

INTEL. 1986. *MCS-51 Macro Assembler User's Guide for DOS Systems, Development Solutions.* Santa Clara, C.A. Intel Corporation.

JORDAAN G.D. 1988. *Development of a Programmable Array Logic Programmer Using a Home Computer.* Ongepubliseerde MDip TECH verhandeling. Technikon O.V.S. Bloemfontein.

OHLSEN, C. & STOKER, G. 1989. *Turbo Pascal Advanced Techniques.* Carmel, Indiana. Que Corporation.

SWAN T. 1991. *Mastering Turbo Pascal 6, 4th Ed.* Carmel, Indiana. Hayden Books.

SYCK, G. 1991. *The Waite Group's Turbo Assembler Bible.* Carmel, Indiana. Sams, a Division of Macmillan Computer Publishing.

WILLIAMS, A.B. 1984. *Designer's Handbook of Integrated Circuits.* New York. McGraw-Hill.