SISÄLLYSLUETTELO

LYHENNELUETTELO

BIOS	Basic Input Output System
CASE	Computer Aided Software Engineering
CPU	Central Processing Unit
CMOS	Complementary Metal Oxide Semiconductor
CSU	Current Signal Output Unit
DFD	Data Flow Diagram
EPROM	Erasable Programmable Read Only Memory
FIFO	First In First out
I/O	Input/Output
IRQ	Interrupt Request
ISR	Interrupt Service Routine
ISA	Industry Standard Architecture
MMU	Microprocessor and Memory Unit
PC	Personal Computer
PIC	Programmable Interrupt Controller
PPU	Pulse Processing Unit
RAM	Random Access Memory
ROM	Read Only Memory
RTSA	Real-Time Structured Analysis
SPU	Serial Parallel Interface Unit
SSEM	Systems Software Engineering Method
STD	State Transition Diagram
UART	Universal Asynchronous Receiver/Transmitter
WIU	Wiring Interface Unit

1. JOHDANTO

1.1 Sulautettu reaaliaikainen järjestelmä

Sulautetulla järjestelmällä tarkoitetaan sähkömekaaniseen laitteeseen integroitua ohjaavaa tietokonejärjestelmää, joka ohjaa laitteen sähköisiä ja mekaanisia toimintoja liityntä- ja ohjauselektroniikan avulla (/6/). Sulautettuja järjestelmiä on nykyisin käytössä kaikkialta. Hyvinkin yksinkertaisista laitteista ja kodinkoneista löytyy jonkinlainen mikroprosessori tai -kontrolleri, joka ohjaa niiden toimintaa tai muilla tavoin käsittelee tietoa. Tyypillisiä sulautettujen järjestelmien sovellusalueita ovat muun muassa matkapuhelimet, hissit, TV:t ja erilaiset prosessinohjausjärjestelmät. Sulautettujen järjestelmien suunnittelu ja toteutus ovat taloudellisesti merkittäviä aloja, koska suurin osa kaikista valmistetuista prosessoreista on sulautetuissa järjestelmissä ja valtaosa ohjelmointityöstä tehdään niiden ohjelmoimiseksi.

Sulautetut järjestelmät ovat usein myös reaaliaikaisia. Tämä on niin yleistä, että joissakin yhteyksissä nimityksiä reaaliaikainen järjestelmä ja sulautettu järjestelmä käytetään toistensä synonyymeinä. Tietokonejärjestelmä on reaaliaikainen, jos sen edellytetään vastaanottavan dataa, lähettävän dataa tai olevan vuorovaikutuksessa ympäristöön, joka on täsmällinen ajan suhteen (/9/).

1.2 Työn tavoite

Tämä työ liittyy pääasiassa sementtitehtailla ja teollisuusmineraalilaitoksilla käytettävän hihna-analysaattorin kehitykseen. Hihna-analysaattori on laite, jolla pystytään mittaamaan eri alkuaineiden pitoisuuksia liikkuvalla kuljetinhihnalla siirrettävästä materiaalista. Analysaattorin mittaukset perustuvat röntgenfluoresenssi-ilmiön hyödyntämiseen (röntgensäteiden käyttöön) mittauksissa. Analysaattori pystyy mittaamaan useita eri alkuaineita

samanaikaisesti. Analysaattorijärjestelmä koostuu varsinaisesta analysaattorista ja siihen liitetystä tietokoneesta ja valvomo-ohjelmistosta.

Työn tavoitteena on korvata analysaattorin käytössä oleva, vanha keskusyksikkö uudella kaupallisella standardin mukaisella prosessorikortilla, ja tehdä uuteen keskusyksikköön ohjelma. Tähän työhön kuuluu analysaattorin uuden keskusyksikön ohjelman ensimmäisen vaiheen toteutus. Ensimmäisessä vaiheessa uuden ohjelman tulee toimia kuten vanhan keskusyksikön ohjelman ja lisäksi tavoitteena on pyrkiä löytää ratkaisu, jossa uudet keskusyksiköt voivat toimia vanhoissa jo toimitetuissa laitteissa varaosina. Analysaattorin kehitysprojektin myöhemmässä vaiheessa sen toimintaan lisätään uusia ominaisuuksia ja analysaattorin keskusyksikön ohjelmaan uusia piirteitä.

1.3 Työn sisältö

Työn alussa käydään läpi reaaliaikaisen ja sulautetun järjestelmän suunnittelua ja erilaisia toteutustapoja. Luvuissa 2-3 esitetään erilaisissa sulautetuissa järjestelmissä käytettyjä vaihtoehtoja ja käydään läpi niiden etuja ja haittoja. Tämän jälkeen esitellään kehitettävä hihna-analysaattori. Analysaattorin ja sen yksiköiden toiminnasta käydään tarkemmin läpi työn kannalta merkittävät osuudet. Työn loppuosassa käsitellään toteutettua järjestelmää. Aluksi esitellään uuden keskusyksikön toimintavaatimukset. Lisäksi tutustutaan kehitystyön eri vaiheisiin, ratkaisuihin ja lopulliseen toteutustapaan.

2. SULAUTETUN JÄRJESTELMÄN SUUNNITTELU

2.1 Sulautetun järjestelmän suunnittelun vaatimuksia

Sulautettujen järjestelmien suunnittelu ja toteutus on usein monimutkaisempaa ja ongelmallisempaa kuin pelkkien ohjelmistoprojektien vastaavat toimenpiteet. Ohjelmiston tulee toimia yhteydessä laitteistoon ja koko järjestelmän taas vuorovaikutuksessa ympäristön kanssa. Tässä luvussa käsitellään pääasiassa ohjelmiston suunnittelua ja laitteiston suunnittelu jätetään vähemmälle käsittelylle.

Sulautetuille järjestelmille asetetaan usein vaatimuksia turvallisuuden, luotettavuuden, ylläpidettävyyden, käyttökelpoisuuden, huollettavuuden ja käyttöliittymän suhteen. Järjestelmät ovat yleensä tiukasti kytkettyjä fyysiseen prosessiin. Tämä tekee sulautetuista järjestelmistä reaaliaikaisia, jolloin järjestelmällä voi olla hyvin tiukat suorituskyky- ja ajoitusvaatimukset.

Sulautettujen järjestelmien elinkaari on yleensä paljon pidempi kuin pelkkien ohjelmistotuotteiden tai laitteistopuolella esimerkiksi toimistomikrojen. Pitkästä elinkaaresta seuraa myös runsas ylläpitotarve, joka pitää ottaa huomioon jo järjestelmän suunnitteluvaiheessa.

Laitteiston ja ohjelmiston yhteistyö merkitsee ylimääräisiä vaiheita ja lisätyötä järjestelmän toteutusprojektissa. Sulautetun järjestelmän resurssien, kustannusten ja aikataulun arvioiminen on vaikeaa ja siihen sisältyy helpommiin riskejä kuin pelkissä ohjelmistoprojekteissa. Sulautetun järjestelmän suunnittelussa on yleensä otettava kantaa jo hyvin varhaisessa vaiheessa ohjelmiston ja laitteiston tehtävien jakoon.

Sulautettujen järjestelmien ohjelmisto-osuuden merkitys on kasvanut yhä tärkeämmäksi. Niillä pystytään joustavasti toteuttamaan asiakaskohtaisia

ominaisuuksia tuotteisiin ja saavuttamaan näin kilpailuetua. Ohjelmistomuutoksia on usein helpompaa toteuttaa kuin muutoksia laitteistoon. Entistä monipuolisemmat sulautetut järjestelmät vaativat toimiakseen laajempia ja monimutkaisempia ohjelmistoja.

Sulautettuja järjestelmiä on hyvin erilaisia ja eri tasoisia. Sulautettu ohjelmisto voi toimia laajassa hajautetussa rinnakkaisessa moniprosessorijärjestelmässä tai yhdessä ainoassa yksinkertaisessa mikro-ohjaimessa. Järjestelmien ja niiden ohjelmistojen sovellusalue on hyvin laaja ja monipuolinen.

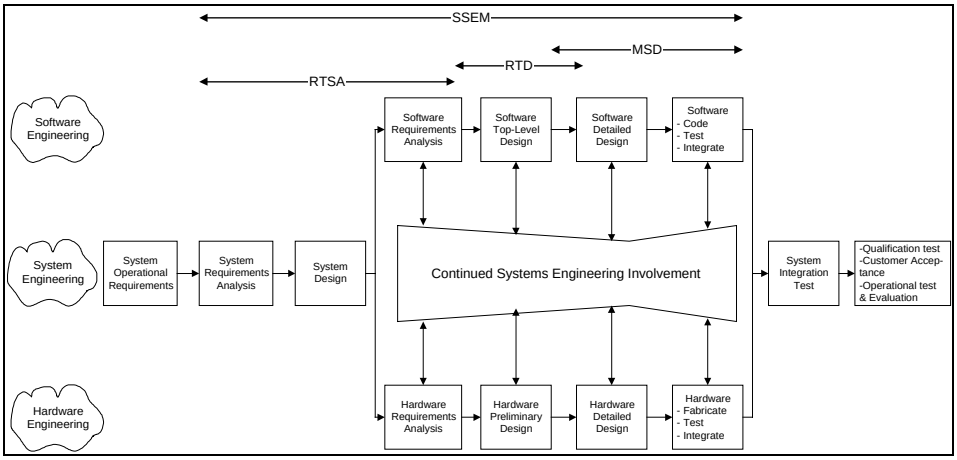
Muunmuassa edellämainittujen syiden johdosta sulautetujen järjestelmien ohjelmistoille ei ole olemassa yhtä määrittely- ja suunnittelumenetelmää, jota voisi käyttää kaikien sulautettujen järjestelmien toteutuksissa. Kunkin sovelluksen erityispiirteet voivat vaatia menetelmiltä jotain erityistä ominaisuutta tai tukea.

2.2 Ohjelmistokehityksen vaihejako

Useimmat nykyiset ohjelmistojen kehitysmenetelmät perustuvat vesiputousmalliin. Perinteisessä vesiputousmallissa ohjelmiston kehitys on jaettu eri peräkkäisiin vaiheisiin. Vesiputousmalli alkaa järjestelmän vaatimusmäärittelystä, jota seuraavat ohjelmiston määrittely, suunnittelu, koodaus, testaus ja ylläpito (/14/). Vesiputousmallin ja sen johdannaisten lisäksi käytetään lukuisia muita vaihejakomalleja. Tällaisiä malleja ovat muunmuassa protoilumallit ja spiraalimalli. Näille kaikille on tyypillistä se, että tuoteesta tai sen osasta tuotetaan jonkinlainen versio, josta saadaan palautetta ja käyttökokemusta. Koko prosessia toistetaan peräkkäin ja lopulta saadaan lopullinen tuote.

Perinteisen vesiputousmallin ongelmat johtuvat pääasiassa sen jäykästä peräkkäisestä vaihejaosta. Todellisuudessa projektit eivät etene lineaarisesti

vaiheittain, vaan eri vaiheisiin joudutaan palaamaan myöhemmin uudelleen ja kutakin vaihetta toistetaan useamman kerran. Vesiputousmallissa oletetaan, että järjestelmän vaatimusmäärittelyt olisivat täydelliset heti alussa, mikä ei useinkaan ole todellisuudessa realistista. Virheiden löytyminen ja erilaisten muutosten tarpeellisuuden havaitseminen tapahtuu yleensä vasta hyvin myöhäisessä vaiheessa.



Kuva 1. Sulautetun järjestelmän kehitysprosessin vaihejako (/8/)

Sulautetun järjestelmän kehitysprosessin vaiheissa on mukana myös laitteiston kehitys. Kuvassa 1 esitetään yksinkertainen vaihejako sulautetun järjestelmän kehitykseen. Järjestelmän kehitys jakautuu järjestelmänsuunnitteluvaiheen jälkeen ohjelmisto- ja laitteistoprojekteihin.

Ohjelmiston kehitys muodostaa oman kokonaisuutensa josta käytetään lyhennystä SSEM (Systems Software Engineering Method). Järjestelmän ohjelmistokehitys on vielä jaettu eri vaiheita sisältäviin komponentteihin, jotka on nimetty seuraavasti:

- Real-Time Structured Analysis (RTSA)
- Real-Time Design (RTD)
- Modular Structured Design (MSD)

Sulautetun järjestelmän ohjelmiston kehitysvaiheet (/8/):

- Järjestelmän vaatimusmäärittely (System Requirements Analysis) on asiakastarpeiden siirtämistä järjestelmäsuunnittelun vaatimuksisksi ja vaatimusten huomioonottamista järjestelmäsuunnittelun komponenteissa.

- Järjestelmäsuunnittelu (System Design) on koko järjestelmän arkkitehtuurin suunnittelua siten, että se täyttää järjestelmälle asetetut vaatimukset. Kokonaisjärjestelmä pitää sisällään laitteiston, ohjelmiston, ihmiset ja niiden välisen kommunikoinnin. Tässä vaiheessa määritellään eri osien työnjako sekä rajapinnat.

- Ohjelmiston vaatimusmäärittely (Software Requirements Analysis) on ohjelmistokomponenttien toiminnan yksityiskohtaista määrittelyä sekä laitteiston ja ohjelmiston välisten rajapintojen määrittelyä.

- Ohjelmiston arkkitehtuurin suunnittelu (Software Top-Level Design) määrittelee ohjelmiston kokonaisarkkitehtuurin sisäiset ja ulkoiset rajapinnat. Lisäksi määritellään ohjelmiston sisäiset ja ulkoiset rajapinnat.

- Ohjelmiston yksityiskohtaisessa suunnittelussa (Software Detailed Design) määritellään ja eritellään ohjelmistokomponenttien sisäinen rakenne pseudokoodin avulla.

- Ohjelmiston toteutus-, testaus- ja kokoamisvaiheessa (Software Code/Test/Integrate) ohjelmisto toteutetaan. Ohjelmistokomponentit voidaan toteuttaa ja testata erikseen, jonka jälkeen ne liitetään yhteen.

2.3 Kuvaustekniikoita

Ohjelmiston määrittelyn ja suunnittelun apuna on käytettävissä lukuisia kuvaus-tekniikoita. Yksinkertaisin tapa kuvata asioita on käyttää normaalia kieltä,

erilaisia taulukoita ja matemaattisia kaavoja. Näiden tapojen lisäksi on olemassa erityisiä kuvaustekniikoita kuten tila- ja tietovirtakaaviot. Menetelmät ovat tapoja soveltaa kuvaustekniikointa käytännössä. Yhdistelemällä erilaisia kuvaustekniikoita ja ohjeistamalla niiden käyttö saadaan erilaisia menetelmiä, kuten esimerkiksi SA-menetelmä (/5/).

Seuraavassa on esitetty joitakin ohjelmistokehityksen eri vaiheiden tekniikoita luettelonomaisesti (/8/). Tekniikoista on käytetty niiden englanninkielisiä nimiä, koska läheskään kaikille tekniikoille ei ole vakiintunut suomenkielistä vastinetta. Samoja tekniikoita voidaan joissakin tapauksissa käyttää useissa eri vaiheissa.

Järjestelmän vaatimusmäärittelyssä käytettyjä kuvaustekniikoita:

- Data context diagram (DCD)
- Data flow diagrams (DFD)
- Data dictionary
- Control context diagram (CCD)
- Control flow diagram (CFD)
- Response time specification (RTS)
- Requirements traceability matrix (RTM)
- Event-action diagram (EAD)

Järjestelmäsuunnittelussa käytettyjä kuvaustekniikoita:

- Architecture context diagram (ACD)
- Architecture flow diagrams (AFD)
- Architecture interconnect diagrams (AID)
- Architecture dictionary
- Response time allocation (RTA)
- Requirements traceability matrix (RTM)

Ohjelmiston vaatimusmäärittelyssä käytettyjä kuvaustekniikoita:

- Enhanced flow diagram (EFD)
- Data context diagram (DCD)
- Data flow diagrams (DFD)
- Data dictionary
- Control context diagram (CCD)
- Control flow diagram (CFD)
- Process activation tables (PAT)
- State transition diagrams/tables (STD/STT)
- Response time specification (RTS)
- Requirements traceability matrix (RTM)

Ohjelmiston arkkitehtuurin suunnittelussa käytettyjä kuvaustekniikoita:

- Flattened flow diagram (FFD)
- Task communication graph (TCG)
- Software architecture diagram (SAD)
- Structure charts
- Requirements traceability matrix (RTM)
- Program design language (PDL)

Ohjelmiston yksityiskohtaisessa suunnittelussa käytettyjä kuvaustekniikoita:

- Software architecture diagram (SAD)
- Structure charts
- Requirements traceability matrix (RTM)
- Program design language (PDL)

2.4 Oliomenetelmät

Olio ajattelu ja olio-ohjelmointikielet ovat edistäneet oliomenetelmien käyttöä ohjelmistojen määrittelyssä ja suunnittelussa. Oliomenetelmät jaetaan kahteen eri luokkaan, oliokeskeisiin (object-oriented) ja oliopohjaisiin (object-based) menetelmiin. Oliokeskeinen menetelmä sisältää periytymisen ja oliopohjainen

menetelmä ei sisällä sitä (/6/). Oliokeskeisille menetelmille on hyvin vaikeaa kuvata mitään yleistä esitystä. Ne voidaan kuitenkin jakaa karkeasti perinteiseen määrittelyyn, suunnitteluun ja toteutukseen.

Oliokeskeisessä määrittelyssä kuvataan tyypillisesti tietomalli, tilamalli ja prosessimalli. Tietomalli on ohjelmiston staattisten ominaisuuksien kuvaus, jossa kuvataan oliot, niiden attribuutit ja olioiden väliset suhteet. Tilamallissa kuvataan tilakäyttäytyminen ja olioiden välinen kommunikointi. Olion tilakäyttäytyminen kuvataan tilasiirtymäkaaviolla ja niiden välinen kommunikointi olioiden kommunikointimallilla. Prosessimallissa kuvataan toimintoihin liittyvä prosessointi ja kuvaustapana käytetään tilavuokaavioita.

Suunnittelu voidaa jakaa järjestelmäsuunnitteluun ja oliosuunnitteluun. Järjestelmäsuunnittelussa tehdään järjestelmää koskevat korkean tason suunnittelupäätökset. Oliosuunnittelussa tehdään varsinainen oliokeskeinen suunnittelu laajentamalla, syventämällä ja lisämällä määrittelyssä kuvattuja piirteitä. Suunnittelussa määritellään muunmuassa olioiden attribuuttien esitystapa ja jaetaan ohjelmisto erilaisiin osiin, moduuleihin ja paketteihin.

Toteutusvaiheessa suunnitteluvaiheessa määritellyt ja suunnitellut luokat sekä niiden väliset kommunikaatiot koodataan olio-ohjelmointia tukevalla kielellä. Toteutusvaihe on yleensä hyvin mekaaninen. Tämä vaihe voi olla osaksi myös automaattinen, mikäli määrittelyssä ja suunnitteluvaiheissa on käytetty työkaluja, jotka mahdollistavat automaattisen koodin generoinnin.

Kaikilla oliomenetelmillä olioiden tunnistaminen on vaativa ja tärkeä vaihe ohjelmiston kehityksessä. Menetelmät perustuvat olioiden tunnistamiseen ongelmakentästä sekä niihin liittyvien toimintojen ja olioiden välisen kommunikoinnin kuvaamiseen. Oliomenetelmiä on hyvin monia ja ne painottavat eri ominaisuuksia. Seuraavassa on lyhyesti esitelty muutama menetelmä ja niiden pääpiirteet. Menetelmien nimeämisessä on käytetty

menetelmiin liittyvien julkaisujen tai kirjojen tekijöiden nimiä, ellei menetelmälle ole vakiintunut jotain muuta nimeä (/6/).

- Booch

  Menetelmä pohjautuu ohjelmistokehityksen prosessimalliin, niin sanottuun spiraalimalliin. Menetelmä on iteratiivinen menetelmä, jossa suunnitelma tarkentuu vaihe vaiheelta. Menetelmässä ei ole kuvattu varsinaisen määrittelyvaiheen tehtäviä, vaikka se sisältääkin tiettyjä määrittelyvaiheen kuvaustapoja. Se on pyritty tekemään hyvin puhtaasti oliokeskeiseksi.

- Colbert

  Menetelmä kattaa sekä määrittely- että suunnitteluvaiheen. Oliokommunikointikaavio sisältää hierarkisen kuvauksen olioiden välisestä kommunikoinnista. Käyttäytymiskuvauksissa käytetään muunmuassa tilasiirtymäkaavioita.

- HOOD (Hierarchical Object-Oriented Design)

  Menetelmä tukee suunnittelu- ja toteutusvaiheita ja on tarkoitetu reaaliaikajärjestelmien suunnitteluun. Menetelmä tukee oliorakenteen esittämistä hierarkisesti, jolloin ohjelmiston arkkitehtuuria voidaan suunnitella asteittain tarkentaen. Menetelmän soveltaminen edellyttää ainakin laajojen järjestelmien yhteydessä jonkin oliokeskeisen määrittelymenetelmän käyttöä.

- ROOM (Real-time Object-Oriented Modeling)

  Menetelmä on reaaliaikajärjestelmien kehitykseen soveltuva oliokeskeinen menetelmä. Se soveltuu hyvin hajautettujen järjestelmien kehitykseen. Menetelmässä järjestelmä kuvataan käyttäen kolmea eri tyyppistä oliota (acctor, protocol ja data). Menetelmä perustuu kerrosrakenteeseen. Kerrokset tarjoavat palveluita muiden kerrosten käytettäväksi.

Oliomenetelmien käytön tueksi on kehitelty useita kaupallisia työvälineitä. Työvälineet ovat keskenään erilaisia ja ne tukevat eri oliomenetelmiä. Yleensä ne sisältävät oliokeskeisen kuvausmenetelmän ja osittaisen automaattisen koodin generoinnin.

2.5 Sulautetun järjestelmän suunnittelun ominaisuuksia

Kuten edellä mainittiin sulautetut järjestelmät poikkeavat toisistaan hyvin paljon ominaisuuksiltaan ja käyttökohteiltaan. Järjestelmän suunnittelun ja toteutuksen lähestymistapa riippuu paljon myös järjestelmän luonteesta. Koko projektin lähestymistapa on hyvin erilainen riippuen onko toteutettava järjestelmä kontrollipainoitteinen, tietopainoitteinen vai laskentapainoittenen.

Sulautettujen järjestelmien määrittelyssä ja suunnittelussa voidaan käyttää myös eri menetelmiä projektin eri vaiheissa. Järjestelmän suunnittelu voidaan tehdä käyttäen perinteisiä menetelmiä, mutta suunnittelussa ja toteutuksessa käytetään oliomenetelmiä. Kussakin vaiheessa pyritään käyttämään menetelmää, josta on eniten hyötyä ja joka antaa parhaan tuloksen. Erilaisia menetelmiä käytetään myös melko vapaasti soveltaen. Tuotekehitystä tekevillä yrityksillä on usein oma ohjeistuksensa järjestelmän kehityksen eri vaiheista. Oman ohjeistuksen avulla voidaan kehitystyö tehdä yritykselle ja kullekin tuotteelle parhaimmalla ja soveliaimmalla tavalla.

Sulautettujen järjestelmien laatuvaatimukset ovat usein paljon korkeammat kuin pelkillä ohjelmistoprojekteilla. Sulautetut järjestelmät voivat vaikuttaa ihmisten terveyteen ja hyvinvointiin suoraan tai välillisesti. Ihmishenki saattaa riippua suoraan järjestelmän toiminnasta, kuten esimerkiksi sairaalalaitteissa tai lentokoneteollisuudessa. Tällöin on selvää, että järjestelmien laatuun ja toimivuuteen tulee kiinnittää erityistä huomiota. Laatuvaatimusten johdosta koko toteutetun prosessin tulee noudattaa vaadittuja standardeja ja menetelmiä sekä sen tulee olla dokumentoitu riittävän tarkasti.

Yhä monimutkaisemmista järjestelmistä ja niiden komponenteista on tullut tarve kehittää ja käytää uusia määrittely- ja suunnittelumenetelmiä. Lisääntyneitä vaatimuksia ei ole pystytty saavuttamaan ja toteuttamaan perinteisillä menetelmillä. Uusia menetelmiä on tullut ja tulee koko ajan lisää sekä laitteisto- että ohjelmistokehityksen puolelle. Yhtenä tällaisenä uutena menetelmänä voidaan mainita Petri-verkkojen käyttöä sulautettujen järjestelmien mallinnuksessa. Monimutkaisten järjestelmien kuvaamiseen on kehitelty myös erilaisia matemaattisia menetelmiä (/4/, /20/).

3. REAALIAIKAISEN SULAUTETUN JÄRJESTELMÄN TOTEUTUS

3.1 Laitteiston valinta ja toteutus

Yksi merkittävin sulautetun järjestelmän suunnittelun ja toteutuksen päätöksistä on, millä tasolla järjestelmän laitteisto suunnitellaan ja toteutetaan itse. Toteutettava järjestelmä tai sen toimintaympäristö voi asettaa rajat toteutustasolle. Järjestelmän toteutuksen aikataulu ja kustannustaso vaikuttavat myös valintoihin.

Järjestelmän toteutusasteet voidaan jaoitella seuraavasti (/9/):

1. Käytetään valmista järjestelmää.
2. Hankitaan järjestelmän laitteisto valmiina ja tehdään ohjelmisto itse.
3. Integroidaan järjestelmä valmiista piirilevy-yksiköistä.
4. Suunnitellaan laitteisto käyttäen valmiita integroituja piirejä ja komponentteja.
5. Suunnitellaan käytettävät integroidut piirit itse. Joissakin tapauksissa suunnitellaan kaikki komponentit transistoritasolta alkaen.

Usein osa sulautetun järjestelmän toiminnoista on mahdollista toteuttaa sekä laitteistolla että ohjelmallisesti. Yleensä laitteiston suunnittelussa ja varsinaisessa toteutuksessa joudutaan menemään matalammalle tasolle, mikäli toimintoja aiotaan toteuttaa enemmän laitteistolla kuin ohjelmistolla.

3.1.1 Valmiiden kaupallisten korttien käyttäminen

Useassa tapauksessa sulautettu järjestelmä tai jokin sen osa on mahdollista toteuttaa valmiilla kaupallisilla korttitason yksiköillä. Markkinoilla on saatavilla hyvin monenlaisia prosessorikortteja, joissa on erityyppisiä ja suorityskykyisiä

prosessorivaihtoehtoja. Järjestelmästä riippuen erilaisia I/O- ja muita yksiköitä löytyy kuhunkin arkkitehtuuriin vaihteleva määrä.

Valmiiden piirilevy-yksiköiden käytöllä on mahdollista saavuttaa suuria ajallisia säästöjä järjestelmän toteutuksessa. Tuotekehityksen nopeus ja järjestelmien markkinoille saantiaika ovat nykyisin hyvin merkittäviä taloudellisia ja kilpailullisia tekijöitä monissa tapauksissa. Nopeudella voidaan ratkaista tuotteiden markkinaosuudet ja niiden menestymismahdollisuudet.

Valmiita yksiköitä käyttämällä niiden suunnittelu ja valmistusaika erilaisine prototyyppivaiheineen jää pois. Järjestelmä tai sen osa saadaan valmiiksi hyvin nopeasti ja päästään testaamaan esimerkiksi ohjelmistoa heti sen kehityksen alkuvaiheessa. Järjestelmän testausaika lyhenee, koska valmiit piirilevy-yksiköt ovat toimivia ja ne on jo testattu valmistajan toimesta.

Järjestelmän suunnittelussa ja toteutuksessa säästetty aika merkitsee myös yleensä suoraan toteutusvaiheen kustannussäästöjä. Suunnitelun ja toteutuksen lyhentyminen ja yksinkertaistuminen pienentää niihin tarvittavaa taloudellista panosta ja muita resursseja. Etenkin laitteiston suunnittelussa ja toteutuksessa saatavat säästöt voivat olla huomattavia, koska niiden toteuttaminen on yleensä kallista ja paljon resursseja vaativaa.

Toteutuksen vaatima aika, työmäärä ja kustannukset on helpompi arvioida käytettäessä valmiita korttitason yksiköitä verrattuna itse suunniteltuihin yksikköihin. Näin saadaan toteutusvaiheen riskejä pienennettyä, koska alemman tason toteutukseen liittyy aina helpommin yllätyksiä ja virhearviointeja.

Valmiiden kaupallisten yksiköiden käyttöä helpottavat usein valmistajalta saatavat valmiit ohjelmistokomponentit. Kaupallisiin kortteihin löytyy ohjelmakirjastoja ja ajureita useille eri ohjelmointikielille ja käyttöjärjestelmille. Ohjelmoinnissa voidaan usein käyttää korkeamman tason ohjelmointikieliä

asemblerin sijasta. Ohjelmointityö helpottuu, jonka seurauksena säästetään jälleen ajassa ja kustannuksissa.

3.1.2 Alemman tason suunnittelu ja toteutus

Joihinkin sovelluluksiin voi olla mahdotonta löytää sopivaa valmista kaupallista ratkaisua. Tämä voi johtua esimerkiksi järjestelmän erikoisuuden, tavanomaisuudesta poikkeavien suorituskykyvaatimusten, järjestelmän koko vaatimusten tai toimintaympäristön olosuhteiden johdosta. Tällöin on selvää, että järjestelmä joudutaan itse suunnittelemaan ja toteuttamaan alemmalta tasolta.

Valmiit kaupalliset piirilevy-yksiköt on pyritty suunnittelemaan siten, että ne ovat mahdollisimman yleiskäyttöisiä. Valmistajat pyrkivät varmistamaan niille yleiskäyttöisyydellä mahdollisimman suuret markkinat. Tämän johdosta valmiilla yksiköillä on usein paljon tarpeettomia toimintoja, ominaisuuksia ja komponentteja, joilla ei ole mitään käyttöä toteutettavassa järjestelmässä. Itse suunnitellulla järjestelmällä saadaan toteutettua kokoonpano, jossa ei ole kuin järjestelmän tarpeelliset osat ja ominaisuudet. Tällöin on mahdollista säästää kustannuksissa, koska tarpeettomia ominaisuuksia ei toteuteta ja niiden vaatimia komponentteja ei tarvitse hankkia. Kustannuksia pystytään alentamaan myös mitoittamalla laitteiston eri osien tehokkuus sopivaksi, jolloin vältytään turhan tehokkaiden komponenttien aiheuttamilta ylimääräisiltä kustannuksilta.

Sulautetut järjestelmät asettavat useissa tapauksissa laitteiston ominaisuuksille melko tiukkoja vaatimuksia, joihin valmiiden ratkaisujen löytäminen voi olla mahdotonta tai vaikeaa. Itse suunniteltu järjestelmä tai osa voidaan mitoittaa ja toteuttaa esimerkiksi koon ja muodon suhteen sopivaksi. Mekaaninen muoto on näin mahdollista optimoida sovellukselle parhaiten soveltuvaksi. Virrankulutus ja sille asetetut vaatimukset on toinen esimerkki, jonka suhteen järjestelmän vaatimukset voivat johtaa valmiiden ratkaisujen hylkäämiseen.

Vaikka itse suunnitellun järjestelmän suunnittelu- ja toteutuskustannukset ovat alkuvaiheessa suuremmat, tilanne voi kääntyä tuotantovaiheessa alemman tason toteutukselle edullisemmaksi. Järjestelmän tai sen yksiköiden valmistuskustannukset tulevat sitä merkityksellisimmiksi mitä suurempia määriä järjestelmää valmistetaan. Mahdollisimman taloudellisen järjestelmän tai sen osan suunnitteluun kannattaa uhrata enemmän resursseja, mikäli niitä tuotetaan hyvin suuria määriä. Laajassa massatuotannossa pienetkin säästöt tulevat merkittäviksi.

Joissakin tapauksissa valmiita kaupallisia ratkaisuja käytetään järjestelmän suunnitteluvaiheessa. Järjestelmä toteutetaan aluksi kaupallisia kortteja käyttäen. Näin saadaan tuote nopeasti toteutettua ja testattua. Tuotteesta pystytään toteuttamaan prototyyppi, jonka avulla sen käytännön toiminta voidaan todeta ja seurauksena esimerkiksi ohjelmistokehitys helpottuu. Varsinaisen tuotannon alettua, tai kun on varmistuttu tuotteen menekistä, laitteisto voidaan korvata itsesuunnitellulla ratkaisulla.

Valmiiden yksiköiden käytöllä voidaan joutua sitoutumaan niiden valmistajaan, mikäli muita korvaavia toimittajia ei löydy, tai niitä on vain hyvin rajoitettu määrä. Kaupallisen yksikön valmistajan toimenpiteet aiheuttavat suuren riskin koko toteutetulle järjestelmälle, jossa sitä käytetään. Yksikön tuotanto saattaa loppua tai sen toimintaa saatetaan muuttaa ilman, että sulautetun järjestelmän toteuttajalla on mitään vaikutusmahdollisuuksia asiaan. Mikäli tällöin täysin vastaavaa tuotetta ei löydy tilalle, joudutaan kaupallisella yksiköllä totetutettu sulautettu järjestelmä suunnittelemaan ja toteuttamaan ainakin osittain uudelleen. Joistakin tilanteista saatetaan selvitä pelkillä ohjelmistomuutoksilla, mutta on myös mahdollista, että lisäksi muuhun laitteistoon joudutaan tekemään muutoksia.

Yksiköiden ja komponenttien saatavuuteen on jollakin tavalla varauduttava koko sulautetun järjestelmän elinkaaren ajalle, ellei haluta ottaa riskiä järjestelmän uudelleen suunnittelusta ja toteutuksesta mahdollisen komponenttipulan sattuessa. Pahimmassa tapauksessa varastoon on hankittava riittävä määrä järjestelmässä käytettyjä ulkopuolelta hankittuja yksiköitä. Alemman tason toteutuksen etuna on, että toiminnaltaan vastaavia komponentteja saa yleensä useammasta lähteestä. Varastoitavien komponenttien arvo on myös pienempi kuin valmiiden yksiköiden, mikäli joudutaan tilanteeseen, jossa komponenttaja tai yksiköitä ostetaan varastoon.

Alemman tason suunnitteluun ja toteutukseen voi olla myös muita kuin teknisiä perusteita. Yritykselle voi olla imagollisesti tai statuksellisesti tärkeää, että se valmistaa omat tuotteensa. Tuotteiden muotoilun tai muiden ominaisuuksien voidaan haluta poikeavat kilpailijoiden ratkaisuista. Joissakin tapauksissa voidaan alemman tason toteutus tehdä liikesalaisuuksien takia.

3.1.3 PC-pohjaiset sulautetut järjestelmät

PC-arkkitehtuuriin perustuvat järjestelmät ovat nousseet yhdeksi suosituimmaksi kortti-pohjaiseksi ratkaisuksi sulautetuissa järjestelmissä. Tämä on hiukan ristiriitaista, ottaen huomioon sulautettujen järjestelmien luonteen ja toimintaympäristön erilaisuuden verrattuna PC-laitteiden alkuperäiseen käyttötarkoitukseen ja ympäristöön. Toisin kuin toimistokäytössä olevien mikrojen, tulee varsinkin teollisuuskäytössä olevien sulautettujen järjestelmien pystyä toimimaan vaikeissa ympäristöolosuhteissa keskeytyksettä, virheettömästi ja ilman käyttäjän toimenpiteitä (/10/).

Monet edellä mainitut kaupallisten korttien käytön edut ja haitat pätevät korostettuina PC-pohjaisten järjestelmien käytössä sulautetuissa järjestelmissä. Toimistokäyttöön suunniteltu arkkitehtuuri ei sovellu ongelmitta sulautetuissa järjestelmissä käytettäväksi, eikä sitä sellaisenaan ole mahdollista käyttää

läheskään kaikissa sovelluksissa. PC-pohjaisten järjestelmien puutteista huolimatta niillä on monia merkittäviä etuja, joiden takia niiden käyttö sulautetuissa järjestelmissä on perusteltua ja kannattavaa.

PC-arkkitehtuurin suosioon sulautetuissa järjestelmissä johtaneita syitä ovat mm. seuraavat (/18/):

- PC-laitteita on kaikkialla.
- Koetaan, että PC-pohjaiset ratkaisut on helpompia käyttää kuin niiden vaihtoehdot.
- Toimistomikrot tunnetaan hyvin, mikä tekee PC-arkkitehtuurista tunnetun.
- PC-pohjaiset laitteet ovat edullisia.
- Tarjolla on runsaasti edullisia ja korkeatasoisia ohjelmistokehitystyökaluja.
- Saatavilla on suuri määrä erilaisia PC:hen yhteensopivia tuotteita.
- Markkinoille tulee jatkuvasti entistä tehokkaampia laitteita.
- Laaja valikoima näyttöjä ja erilaisia syöttölaitteita.
- Suunnittelijoiden ei tarvitse olla sulautettujen järjestelmien asiantuntijoita.
- Saatavilla suuri määrä ajureita ja ohjelmakirjastoja.
- PCMCIA-kortit soveltuvat sulautettuihin järjestelmiin.

Edullisuus on yksi tärkeimmistä syistä PC-pohjaisten tuotteiden suosioon sulautetuissa järjestelmissä. Edullisuus ei rajoitu pelkästään keskusyksiköihin ja muuhun laitteistoon, vaan myös käyttöjärjestelmät, valmiit sovellukset ja ohjelmointityökalut ovat hyvin kilpailukykyisiä muihin valmiisiin vaihtoehtoihin verrattuna. Järjestelmän laitteiston lisäksi myös suunnittelu ja testaus tulevat usein edulliseksi. Järjestelmän ohjelmistokehitystä ja testausta on mahdollista suorittaa hyvin pitkälle tavallisilla toimistomikroilla, koska ne perustuvat samaan arkkitehtuuriin kuin toteutettava järjestelmä.

Aikaisemmin mainittu tuotekehityksen nopeus pätee myös PC-arkkitehtuurilla toteutettuun järjestelmään. PC-pohjaisten yksiköiden käytöllä on mahdollista saada tuote hyvin nopeasti markkinoille. Erilaisten yksiköiden määrä on niin

laaja, että pelkästään niitä käyttämällä on mahdollista toteuttaa hyvin monenlaisia ja monimutkaisia järjestelmiä. Normaalien keskusyksiköiden ja I/O-laitteiden lisäksi saatavilla on esimerkiksi erilaisia anturiliitäntä-, kenttäväylä- ja verkkokortteja.

Järjestelmän tuotekehityksen nopeutta lisäävät myös monipuolinen ja korkeatasoinen ohjelmistovalikoima. Saatavilla on laaja työkaluohjelmien valikoima helpottamaan ja nopeuttamaan työskentelyä. PC-pohjaisiin laitteisiin on paljon erilaisia ohjelmakirjastoja ja valmiita ohjelmakomponentteja, joita voi käyttää hyväksi omaa ohjelmistoa tehtäessä. Myös käyttöjärjestelmien puolella on saatavana suuri määrä eritasoisia ja ominaisuuksiltaan erilaisia vaihtoehtoja.

Järjestelmän laitteiden ja työkalujen ominaisuuksien tuen merkitys kasvaa käyttöliittymää tehdessä, joka on usein hyvin työläs vaihe toteuttaa. Sulautettujen järjestelmien käyttöliittymän merkitys on kasvanut, koska nykyisin käsitellään yhä suurempia tietomääriä ja laitteistoissa on entistä monipuolisempia toimintoja. Ohjelmistojen lisäksi PC-pohjaisten järjestelmien yhtenä etuna vastaaviin muihin kaupallisiin järjestelmiin on laitteistopuolella erilaisten käyttöliittymien toteutusmahdollisuuksien runsaus. Käyttöliittymän toteutus ja testaus helpottuu myös käytännössä, kun se pystytään tekemään samoja työkaluja käyttäen ja samanlaiseen arkkitehtuuriin perustuvalle laitteistolle kuin suunnitteluvaiheessa käytössä olevat tietokoneet.

PC-pohjaisen laitteiston ja siinä toimivien ohjelmistojen tunnettuvuus ja “yksinkertaisuus” mahdollistavat tuotekehityksen tekemisen myös henkilöiltä, joilla ei ole varsinaista kokemusta sulautettujen järjestelmien suunnittelusta ja toteutuksesta. Tämä on mahdollista erityisesti silloin, kun järjestelmältä ei vaadita jotakin sulautetun järjestelmän erikoista ominaisuutta kuten esimerkiksi kovaa reaaliaikaisuutta tai hyvin laiteläheisiä toimintoja. Tällöin järjestelmä voidaan toteuttaa henkilöstöllä, joka ei koostu sulautetun järjestelmän toteutuksen asiantuntijoista.

PC-komponenttien ja -laitteiden normaali elinkaari on usein huomattavasti lyhyempi kuin sulautettujen järjestelmien elinkaari. Kuten aikaisemmin mainittiin tämä aiheuttaa ongelmia, aivan kuin muitakin valmiita kaupallisia komponentteja käytettäessä. Riskit ovat piiritasolla suuremmat, koska eri valmistajien PC-arkkitehtuurin piirit eivät ole keskenään täysin yhteensopivia. Eräät valmistajat ovat helpottaneet sulautettujen järjestelmien toteutusta lupaamalla valmistaa tiettyjä piirisarjoja ainakin edeltäkäsin ilmoitetun ajanjakson. Prosessorikortitasolla tämä ongelma ei ole niin suuri, mikäli ohjelmia ei ole tehty suoraan laitteistoa ohjaamaan, vaan käytetään esimerkiksi BIOS:in palveluita erottamaan ohjelmisto laitteistosta. Näin prosessoreiden kasvavat tehot on mahdollista saada käyttöön yksinkertaisella laitteistopäivityksellä.

Toimistokäyttöön suunniteltu tavallinen emolevy ja lisäkorttirakenne ovat huonoja mekaanisten ominaisuuksiensa takia useissa sulautettujen järjestelmien tapauksissa. Sulautettuja järjestelmiä varten onkin kehitetty uusia versioita, jotka soveltuvat niihin perinteisiä ratkaisuja paremmin. Uudet versiot ovat toiminnoiltaa yhteensopivia vanhojen järjestelmien kanssa, vaikka niiden mekaaniset ratkaisut poikkeavat perinteisestä ATX-emolevystä. Toiminnallisella yhteensopivuudella saadaan käyttöön PC-arkkitehtuurin tarjoamat edut.

Lähinnä perinteistä emolevyratkaisua on passiivinen ISA-väylä, joka perustuu täysin signaaleiltaan ja liittimiltään ISA-määrittelyihin. Prosessori oheispiireineen on omana korttinaan kehikossa olevassa vaylässä. Tällä tavoin päästään mekaniikaltaan parempaan, joustavampaan ja tilaa säästävämpään ratkaisuun. Korttien kiinnitystapa ja kehikon muotoilu on mahdollista toteuttaa sitten, että järjestelmän tärinäkestävyys ja jäähdytysominaisuudet paranevat.

Toinen ISA-väylään pohjautuva ratkaisu on PC/104-kortit. Ne noudattavat täysin ISA-väylän signaaleita, mutta ovat liittimiltään erilaisia. PC/104-korteissa väylä menee korttien läpi ja ne voidaan pinota toistensa päälle ilman erillistä väyläkorttia. PC/104-moduulit ovat kooltaan vain 3.6 x 3.8 x 0.6 tuumaa, joten niistä voidaan muodostaa pienikokoinen ja hyvin tärinää sietävä kokonaisuus. Yhdellä PC/104-CPU-kortilla on yleensä prosessorin, muistin ja normaaleiden oheispiirien lisäksi näppäimistö-, sarja- ja rinnakkaisliitännät (/1/). PC104-kortteihin pohjautuvat ratkaisut ovat saaneet jalansijaa sovelluksissa, joissa VME tai vastaavat isot väyläratkaisut ovat liian isoja tai liian hintavia (/13/).

Erilaiset PC-pohjaiset väylättömät eli yhden kortin tietokoneet ovat myös suosittuja vaihtoehtoja sulautettuihin järjestelmiin. Näissä yhdelle kortille on integroitu koko tietokone ja sen liitynnät, aina verkkokorttia ja näytönohjainta myöten. Usein näistä väylättömistä yhden kortin tietokoneista löytyy kuitenkin jonkinlainen laajennusväylävaihtoehto, kuten esimerkiksi PC/104-väylä. Kaikista pisimmälle integroinnissa on menty yhden sirun tietokoneissa, joissa yhdelle mikropiirille on pyritty integroimaan melkein koko PC-emolevy toimintoineen. Näillä ratkaisuilla on mahdollista muodostaa hyvin pieni ja kompakti PC-yhteensopiva järjestelmä.

3.2 Käyttöjärjestelmän valinta

Sulautetut järjestelmät asettavat käyttöjärjestelmälle usein suuria vaatimuksia. Toisaalta yksinkertaisimmissa sulautetuissa järjestelmissä ei varsinaista järjestelmätasoa ole lainkaan, vaan kaikki ohjelmat ovat sovellusohjelmia. Esimerkiksi mikrokontrollerissa oleva ohjelmisto voi toimia ikuisessa silmukassa, jossa se tutkii syötevirtoja ja säätää niiden avulla ulostulojaan.

Käyttöjärjestelmän käytöllä usein yksinkertaistetaan sovellusohjelman tekemistä ja ylläpitoa. Käyttöjärjestelmä tarjoaa sovellusohjelmalle useita palveluita ja

näin sulautetun järjestelmän toteutuksessa ei tarvitse huolehtia näistä perustoiminnoista. Toisaalta käyttöjärjestelmään perustuva ratkaisu voi olla huonompi kuin kokonaan itse toteutettu järjestelmä, koska käyttöjärjestelmän tarjoaman palvelut tai niiden ominaisuudet eivät välttämättä ole riittävät. Käyttöjärjestelmän käytöstä joutuu yleensä maksamaan lisenssimaksuja käyttöjärjestelmän valmistajalle ja näin sen käyttö voi tulla kalliimmaksi kuin oman pienen ja yksinkertaisen käyttöjärjestelmän toteutus.

Valittu laitteisto vaikuttaa mahdollisiin käyttöjärjestelmävaihtoehtoihin ja päinvastoin. Joihinkin järjestelmiin on saatavilla vain rajoitettu määrä käyttöjärjestelmiä. Vastaavasti laitteiston valinnassa voidaan joutua rajoittumaan tiettyihin ratkaisuihin, jos on pakko tai halutaan käyttää jotain tiettyä käyttöjärjestelmää.

Käyttöjärjestelmä ja sen ominaisuudet korostuvat mitä monimutkaisemmasta järjestelmästä on kyse. Sulautettu järjestelmä voi vaatia moniajo-ominaisuuksia ja reaaliaikaisuutta. Se voi olla hajautettu moniprosessorijärjestelmä, jossa eri yksiköt ovat yhteydessä toisiinsa.

3.2.1 Reaaliaikaiset käyttöjärjestelmät

Vasteaikavaatimusten mukaan reaaliaikajärjestelmät voidaan jakaa ns. koviin ja pehmeisiin reaaliaikajärjestelmiin. Pehmeissä reaaliaikajärjestelmissä ei ole tiukkoja vasteaikavaatimuksia, vaan ne voidaan ylittää. Kovissa reaaliaikajärjestelmissä vasteaikavaatimukset ovat kriittisiä, ja niitä ei voida ylittää. Vasteajan ylittäminen voi johtaa vakaviin seurauksiin tai ainakin tuloksen tai toiminnon hyödyttömyyteen. Kovien reaaliaikajärjestelmien vasteaikavaatimus ei välttämättä ole nopeus vaan vasteajan täsmällisyys, jonka täytyy olla aina sama.

Reaaliaikaisen moniajokäyttöjärjestelmän yhtenä päätarkoituksena on tehtävien priorisointi ja suoritusjärjestyksen määrääminen siten, että tiukkaa reaaliaikaisuutta edellyttävät tehtävät tulevat suoritetuksi määräajassa. Samalla järjestelmän on pystyttävä antamaan resursseja muillekin tehtäville.

Reaaliaikaisia käyttöjärjestelmiä on saatavilla useita vaihtoehtoja eri valmistajilta. Nämä reaaliaikakäyttöjärjestelmät tarjoavat yleensä peruspalveluita prosessien hallintaan, synkronointiin, kommunikointiin ja poissulkemiseen. Lisäksi saattaa olla tarjolla apuneuvoja oheispiirien ohjaamiseen ja ohjelmien testaukseen.

3.2.2 Muut käyttöjärjestelmät

Normaaleissa toimistomikroissa käytettävät käyttöjärjestelmät ovat myös saavuttaneet asemaa sulautetuissa järjestelmissä reaaliaikakäyttöjärjestelmien ja erillisten sulautetuille järjestelmille suunniteltujen käyttöjärjestelmien lisäksi. Perinteiset DOS-, Windows ja Windows NT-käyttöjärjestelmät ovat laajalti käytössä sulautetuissa järjestelmissä sellaisenaan tai paremmin sulautettuihin järjestelmiin soveltuvina muunnelmina.

Näiden toimistokäyttöön tarkoitettujen käyttöjärjestelmien käytössä sulautetuissa järjestelmissä on useita ongelmia. Esimerkiksi DOS on hyvin alkeellinen ja ei-reaaliaikainen käyttöjärjestelmä. Sen ominaisuudet ovat hyvin puutteellisia ja käyttövarmuus ei ole kovinkaan korkea. DOS tarjoaa vain peruspalvelut kuten tiedostojärjestelmän. Se ei sisällä mitään muistinsuojausmekanismia, jotka estäisivät yhden ohjelman sotkemasta toisen ohjelman muistialuetta. Lisäksi DOS vaatii sellaisia järjestelmäresursseja, joita sulautetuissa järjestelmissä ei useinkaan ole (/17/).

DOS-käyttöjärjestelmän etuna on kuitenkin edullisuus ja sen tukema sulautetuissa järjestelmissä suosittu ISA-arkkitehtuuriin perustuva laitteisto.

DOS-käyttöjärjestelmä on laajalti tunnettu ja siihen on saatavana erittäin suuri määrä ohjelmointikieliä ja -työkaluja sekä valmiita ohjelmakirjastoja.

Alunperin toimistokäyttöön tarkoitettujen käyttöjärjestelmien puutteiden korjaamiseksi on kehitelty useita muunnelmia käyttöjärjestelmistä, jotka ovat alkuperäisten kanssa yhteensopivia. Muunnelmien avulla käyttöjärjestelmistä on tehty sulautettuihin järjestelmiin soveltuvampia ja säilytetty kuitenkin niiden alkuperäiset edut. DOS-käyttöjärjestelmästä on esimerkiksi olemassa useita versiota, jotka pystyvät toimimaan ROM-muistissa levyaseman sijasta. Tällöin leyaseman tilalla järjestelmässä voidaan käyttää niihin paremmin soveltuvia vaihtoehtoja kuten EPROM- tai flash ROM-muistia.

DOS- ja Window NT-käyttöjärjestelmistä on saatavilla myös useita versioita, joissa on tiukka tosiaikainen ydin, ja jotka kykenevät syrjäyttävään moniajoon. Ne sisältävät käyttöjärjestelmän normaalit palvelut ja ohjelmointirajapinnan. Muunnelmat toimivat sellaisenaan ISA-yhteensopivissa laitteissa ja kykenevät ajamaan normaaleita käyttöjärjestelmän ohjelmia.

3.3 Ohjelmiston kehitystyökalut

Työkaluilla tarkoitetaan ohjelmistopohjaisia apuvälineitä, jotka helpottavat tai edesauttavat jotain ohjelmistotyön vaihetta (/5/). Varsinaisten ohjelmointityökalujen lisäksi ohjelmiston kehitystyökalut pitävät sisällään kaikenlaiset apuvälineet projektinhalintatyökaluista aina testaus- ja dokumentointityökaluihin. Kehitystyökaluista käytetään usein nimitystä CASE -välineet tai -työkalut.

Työkaluja valitessa tulisi kiinnittää huomiota siihen, ettei työkaluja hankita vain niiden itsensä takia vaan, että ne todellakin tehostavat ja helpottavat työtä. Työkalujen tulee olla sovitettavissa kunkin projektin tai käyttäjien työmetodeihin eikä päinvastoin (/20/). Sovitettavuuteen ja työkalujen

mahdollisimman laajaan soveltuvuuteen pyritään työkaluohjelmien muunneltavuudella eli konfiguroitavuudella. Työn tehokkuuden lisäämisen lisäksi kehitystyökaluilla saavutetaan myös monia muita tärkeitä etuja.

Järjestelmien ja ohjelmistojen laajentuminen ja monimutkaistuminen ovat lisänneet erilaisten työkalujen tarvetta. Laajojen projektien hallinta ja suunnittelu eivät enää onnistu ilman projektinhallintatyökaluja. Nopeat tuotekehityssyklit vaativat, että projektista on koko ajan oltava ajantasalla olevaa tietoa. Mahdollisiin ongelmiin ja viivästymisiin on päästävä puuttumaan ennakoivasti tai välittömästi niiden ilmaantuessa. Projektinhallintatyökaluihin liittyy usein myös kuluseurantaa, joka mahdollistaa erilaiset kannattavuus- ja kustannuslaskelmat. Näitä tietoja voidaan käyttää kyseisen projektin lisäksi myös tulevia projekteja suunnitellessa ja arvioitaessa.

Useat työkalut tuottavat dokumentointia ainakin osittain automaattisesti. Tämä parantaa järjestelmän kokonaisdokumentointia ja edesauttaa dokumenttien laatimista projektin eri vaiheissa. Ohjelmiston ja koko järjestelmän laadun ja dokumentoinnin parantuminen ovat hyvin merkittäviä tekijöitä, joita työkalujen avulla pyritään saavuttamaan. Työkaluohjelmien tuottamat tulokset ja niillä tehdyt muutokset menevät useissa tapauksissa talteen jonkinlaiseen tietokantaan. Näin kehitysprosessista muodostuu historia, jota voidaan tarkastella ja analysoida myöhemmin. Lisäksi työkalujen tietokantaan tallentamia tuloksia on mahdollista käyttää uudelleen hyväksi muissa projekteissa.

Ohjelmiston suunnittelua on mahdollista helpottaa työkalujen avulla. Esimerkiksi erilaisia ohjelmiston määrittely- ja kuvausmenetelmiä varten on niitä tukevia ohjelmistoja. Ohjelmiston suunnittelu ja kehitys tapahtuu vaiheittain usein erillisinä prosesseina. Työkaluohjelmille olisi edullista, että ne pystyisivät käyttämään edellisessä vaiheessa tuotettua tulosta hyväkseen.

Nykyisin useat työkaluohjelmat toimivat verkkoympäristössä sekä mahdollistavat että tukevat rinnakkais- ja tiimityöskentelyä. Näistä ominaisuuksista on hyötyä varsinkin suuremmissa projekteissa, joissa työskentelee suuri määrä ihmisiä. Erilaiset projektit ja niiden osat on voitu jakaa useaan osaan, joita voidaan hallita ja toteuttaa useankin toimittajan toimesta.

Versionhallintatyökaluilla helpotetaan kehitysprosessin lisäksi järjestelmän ylläpitoa ja huoltoa. Tieto toimitetuista versioista helpottaa päivitysten ja havaittujen korjausten tekemistä. Tämä on eräs tapa, jonka avulla työkaluohjelmilla pystytään parantamaan myös asiakaspalvelua. Tuotetiedonhallintatyökaluilla pystytään hallitsemaan erilaisia tuotteita, jolloin on mahdollista käyttää hyväksi esimerkiksi samoja komponentteja eri tuotteissa.

Tavallisia “toimisto-ohjelmia” käytetään laajasti kehitystyökaluina, erityisesti kehitystyökaluiksi suunniteltujen ohjelmien ohella. Normaaleilla piirto- ja tekstinkäsittelyohjelmilla voidaan tehdä monia erityisten kehitystyökalujen toimintoja. Nämä normaalit toimistokäytöön suunnitellut ohjelmat ovat usein paljon edullisempia kuin varsinaiset kehitystyökalut.

Eräät laitevalmistajat tarjoavat omia kehitystyökaluja valmistamilleen laitteille tai komponenteille. Toisaalta on nähtävissä, että kehitystyökalujen valmistaminen on siirtynyt enemmän kolmansille osapuolille, joiden ohjelmistot eivät ole sitoutuneet yhteen laitevalmistajaan. Näiden lisäksi suurilla tuotekehitystä tekevillä yksiköillä voi olla omat menetelmänsä ja itse kehitetyt työkaluohjelmat, jotka on räätälöity sopimaan heidän toimintatapoihinsa.

Ohjelmistojen ja laitteistojen testaamista varten on saatavana erilaisia testaustyökaluja. Simulointi- ja mallinnusohjelmilla voidaan testauksen lisäksi käyttää suunnittelun apuna. Esimerkiksi käyttöliittymiä ja niiden toimintaa on mahdollista hioa ja testata etukäteen mallintamalla ja simuloimalla niiden toiminta ennen varsinaista toteutusta. On mahdollista tehdä erilaisia

käytettävyystestejä ja saada käyttäjien kommentteja tuotteesta ennenkuin varsinaista ohjelmistoa ja laitteistoa on alettu edes toteuttamaan.

Sulautetuissa järjestelmissä tarvitaan ohjelmistojen ja laitteistojen suunnittelun lisäksi myös mekaniikan ja muidenkin tuoteteknologioiden tiivistä yhteissuunnittelua. Työkaluohjelmien avulla on mahdollista tuoda ja integroida järjestelmän-, ohjelmiston-, laitteiston- ja elektroniikansuunnitelu yhä lähemmäksi toisiaan. Järjestelmän toteutuksessa on etua mitä aikaisemmassa vaiheessa ja mitä tiiviimmin eri osa-alueet saadaan sidottua toisiinsa.

Sulautettujen järjestelmien suuri erilaisuus aikaansaa myös hyvin kirjavan kehitystyökalujen käytön. Esimerkiksi joidenkin yksinkertaisten mikrokontrolleriin perustuvien tuotteiden ainoana kehitystyökaluna voi olla lähinnä ohjelmistoympäristö. Toisaalta laajojen hajautettujen järjestelmien toteutuksessa voidaan ja usein joudutaan käyttämään apuna monimutkaisempia ja laajimpia kehitystyökaluja.

Sulautetuissa järjestelmissä nykyisin yleisimmin käytetyt ohjelmiston kehitystyökalut ovat (/16/):

- C-ohjelmointi ympäristö
- Projektinhallintatyökalut
- Dokumentointityökalut
- Ohjelmien versionhallinta ja tuotteenhallintatyökalut
- Erilaiset CASE-välineet ohjelmiston suunnitteluun ja kuvaukseen

Sulautettujen järjestelmien ohjelmistoa on usein hyvin hankalaa tai mahdotonta kehittää ennenkuin laitteisto on valmis. Varsinkin erilaisten ajurien ja laitteläheisten osuuksien tekeminen ja testaaminen ilman laitteistoa on vaikeaa. Tuotteet tulisi kuitenkin saada markkinoille mahdollisimman nopeasti, eikä olisi varaa odottaa tuotekehityksen loppuvaiheessa ohjelmiston valmistumista. Tämän johdosta järjestelmän ohjelmiston kehitys pitäisi pystyä aloittamaan ja

toteuttamaan samanaikaisesti järjestelmän laitteiston kehityksen kanssa. Tämän ongelman ratkaisuksi sulautettujen järjestelmien kehitystyötä varten on tarjolla erityisiä yhteissuunnitteluohjelmistoja, joiden avulla järjestelmän laitteisto ja sen toiminta voidaan mallintaa ohjelmiston avulla.

Yhteissuunnitteluohjelmistoissa laitteisto ja sen prosessori mallinnetaan erityisen laitteistokuvauskielen avulla. Mallinnettu järjestelmä toimii kuten todellinen laitteisto, reaaliaikaisuutta lukuunottamatta. Mallinnetussa järjestelmässä voidaan ajaa ja testata todellista järjestelmää varten kehitettyä ohjelmistoa ennenkuin varsinainen laitteisto on valmis.

Ohjelmointityökalu on usein kokonainen kehityspaketti, joka sisältää varsinaisen kääntäjän lisäksi editorin, linkittimen, erilaisia ohjelmointikirjastoja sekä debuggerin. Ohjelmiston toteutukseen käytettäväksi valittavan ohjelmointityökalun valintaan vaikuttaa usein ohjelmoijien omien mieltymysten lisäksi moni muu seikka. Valittu laitteistoratkaisu ja käyttöjärjestelmä voivat rajoittaa käytettävissä olevia vaihtoehtoja, tai jopa pakottaa käyttämään jotain tiettyä ohjelmointikieltä tai tietyn toimittajan ohjelmistoa.

Ohjelmointityökalulta ja -kieleltä vaaditaan varsinkin sulautetuissa järjestelmissä, että ne tuottavat tehokasta käännettyä ohjelmaa sen lisäksi, että niitä on työskennellessä tehokasta käyttää. Kaikkein aikakriittisimpien sovelluksien, tai niiden osien, toteutuksessa joudutaan usein käyttämään assembler-kieltä korkeamman tason kielen tilalla, vaikka järjestelmä olisikin muuten toteutettu korkeamman tason kielellä. Nykyiset ohjelmointityökalut tosin tuottavat niin tehokasta ohjelmakoodia, ettei ainakaan kokemattomampi ohjelmoija saa asembler-kielellä aikaiseksi yhtään sen tehokkaampaa koodia. Sulautetuissa järjestelmissä C-kieli onkin ollut niin suosittu, että siitä on muodostunut lähes standardi. Se tuottaa tehokasta koodia, mutta piilottaa laitteiston monimutkaisuuden rakenteidensa taakse. Oliomenetelmien suosion

lisääntyminen on lisännyt myös C++-kielen suosiota sulautetuissa järjestelmissä.

4. JÄRJESTELMÄN TESTAUS

Testaus on hyvin merkittävä osa järjestelmän ja ohjelmiston kehityksessä. Testaukseen tarvittava aika ja resurssit aliarvioidaan hyvin helposti. Yleensä ohjelmistoprojektin testaukseen ja siihen liittyviin korjauksiin kuluu helposti puoletkin projektin resursseista. Sulautettujen järjestelmien luonteesta johtuen niiden testaukseen kuluu resursseja vielä merkittävästi enemmän. Esimerkiksi joidenkin kriittisten, hyvin monimutkaisten tai vaativassa ympäristössä toimivien sovellutusten ja järjestelmien testaukseen voi joutua panostamaan helposti montakin kertaa enemmän kuin koko muuhun ohjelmistokehitykseen yhteensä.

Testauksen tavoitteena on varmistaa järjestelmän toimivuus ja luotettavuus. Ohjelmiston testauksen yhteydessä testaus määritellään suunnitelmalliseksi virheiden etsimiseksi ohjelmaa tai sen osaa suorittamalla (/5/). Varsinaisten ohjelmiston toimintavirheiden lisäksi etsitään, ettei järjestelmän toiminnalla ole haitallisia sivuvaikutuksia eli, ettei ohjelmisto tai järjestelmä tee jotain sellaista mitä sen ei pitäisi tehdä. Testauksella etsitään myös järjestelmän suunnittelussa tapahtuneita virheitä. Ohjelma tai järjestelmä voivat toimia täysin virheettömästi ja suunnitelmien mukaisesti, mutta ne eivät välttämättä vastaa niille asetettuja vaatimuksia.

Hyvän ja tehokkaan testauksen tavoitteet voidaan määritellä seuraavasti (/11/):

- Testauksen tavoitteena on löytää ohjelmasta virhe.
- Hyvä testitapahtuma on sellainen, jossa suurella todennäköisyydellä löydetään joku vielä havaitsematon virhe.
- Onnistunut testi paljastaa aikaisemmin havaitsemattoman virheen.

Testauksella ei koskaan saavuteta täyttä varmuutta järjestelmän toiminnasta, eikä sillä pystytä takaamaan kaikissa tilanteissa toimivaa järjestelmää. Tämä johtuu siitä, että jo yksinkertaisella ohjelmalla ja järjestelmällä on äärettömän

monta mahdollista tilaa tai suorituspolkua, eikä kaikkia näitä ole mahdollista testata ja käydä lävitse. On hyväksyttävä tosiasiaksi, että testauksella voi ainoastaan löytää uusia vikoja, mutta ei osoittaa järjestelmää virheettömäksi. Testauksen määrä on aina kompromissi järjestelmän luotettavuudesta saavutetun varmuuden ja testaukseen käytettyjen resurssien välillä.

Testauksen lähtökohtana tulee aina olla testaussuunnitelma, joka sisältää testitapaukset. Testatussuunnitelmaa tehtäessä on pyrittävä määrittelemään testauksen kohde mahdollisimman yksiselitteisesti, jolloin testin suunnittelu onnistuu helpommin. Varsinaiseen testaukseen siirrytään vasta, kun testitapausten lähtötiedot ja lopputulokset on määritelty. Olennainen osa testisuunnitelmaa on määritellä testauksen lopettamiskriteerit.

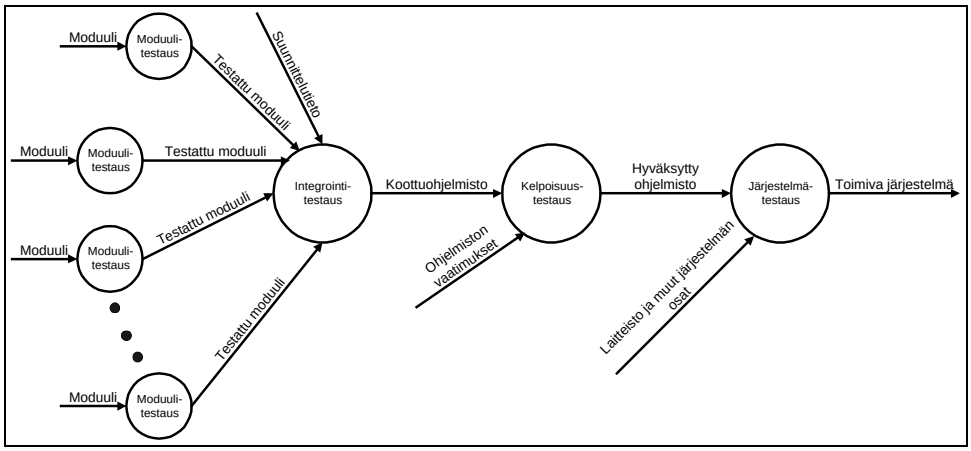
Testauksen vaiheista pitäisi tehdä muistiinpanot ja testauksesta tulisi aina syntyä dokumentti. Muistiin tulisi merkitä ainakin havaitut virheet, virheiden kuvaukset, miten ja milloin virhe havaittiin ja miten se korjattiin. Sen lisäksi, että virheiden kirjaamisella saadaan yksi järjestelmän merkittävistä dokumenteista, voidaan sen avulla päätellä, milloin testausta on tehty riittävästi ja milloin se voidaan lopettaa.

Eräs testaustapa on erilaiset katselmointitilaisuudet. Katselmoimalla voidaan löytää virheitä jo suunnitteluvaiheessa. Alkuvaiheiden katselmoineilla pystytään karsimaan suunniteluvirheiden lisäksi erilaisia määrittelyvaiheen väärinkäsityksiä. Katselmointien käyttö ei rajoitu vain toteutuksen alkuvaiheeseen vaan niitä voidaan käyttää koko kehitysprosessin ajan.

4.1 Testauksen vaiheet

Testaus voidaan jakaa vaiheisiin, joissa pienemmistä yksiköistä siirrytään aina suurempiin kokonaisuuksiin. Aluksi kannattaakin testata pienemmät yksiköt, koska mitä myöhemmin virhe löydetään sitä kalliimmaksi ja vaikeammaksi sen

korjaaminen tulee. Testauksen vaiheet ovat moduulitestaus, integrointitestaus, kelpoisuustestaus ja järjestelmätestaus (/14/).



Kuva 2. Testaamisen vaiheet (/14/)

Yksikkö- eli moduulitestauksessa testataan yksittäisen ohjelmamoduulin toiminta. Moduulista testataan yleensä sen liitännät muuhun ohjelmistoon, sisäiset tietorakenteet, erilaiset toimintatilat eli toimintapolut ja moduulin virheen käsittely. Moduulit toimivat harvoin yksin. Tämän johdosta moduulitestausta varten voidaan joutua tekemään erillisiä testiapuohjelmia tai ajureita, joilla simuloidaan muuta ohjelmistoa ja laitteistoa.

Integrointitestauksessa painopiste on moduulien välisten rajapintojen toimivuuden testauksessa. Integrointitestauksessa kootaan yhteen moduuleita tai moduuliryhmiä ja testataan niiden toimivuutta yhdessä. Lopulta peräkkäisten yhdistämisten jälkeen kaikki moduulit muodostavat kokonaisen toteutettavan järjestelmän. Moduulien yhdistäminen voi tapahtua ylhäältä alaspäin integroimalla (top-down menetelmä), alhaalta ylöspäin integroimalla (down-top menetelmä) tai näiden menetelmien yhdistelmällä.

Ohjelmistolle on asetettu tietyt vaatimukset sen suunnitteluvaiheessa. Kelpoisuustestauksessa varmistetaan, että integrointitestauksen lopputuloksena saatu toimiva ohjelma täyttää sille asetetut vaatimukset. Nämä ohjelmistolle

asetetut vaatimukset pitäisi olla kirjattu vaatimusmäärittelyyn, jossa on kirjattu ylös myös ohjelmiston hyväksymiskriteerit.

Järjestelmätestauksessa testataan koko lopullinen järjestelmää, johon sisältyy laitteisto ja ohjelmisto. Järjestelmätestauksessa testataan myös muut kuin järjestelmän toiminnalliset ominaisuudet. Tällaisia testejä ovat esimerkiksi luotettavuustestit, asennustestit, kuormitustestit ja käytettävyystestit.

4.2 Sulautetun järjestelmän testatuksen ominaispiirteitä

Sulautettujen järjestelmien testaukseen liittyy monia ominaisuuksia ja ongelmia, jotka ovat monimutkaisempia tai joita ei edes ole pelkän ohjelmiston testauksessa.

Moduulitestausvaiheessa ohjelmamoduuleita voi olla hyvin vaikea testata realistisesti, jos ne lopullisessa järjestelmässä ohjaavat tai ovat vuorovaikutuksessa laitteiston kanssa. Moduulitestaus suoritetaan kehitysjärjestelmässä, johon voi olla hyvin vaikea toteuttaa realistisesti kaikkia kohdejärjestelmän rajapintoja ja niihin liittyvää dynamiikkaa. Moduulitestauksia ja muita testauksia voi olla hankalaa suorittaa, jos kehityksessä käytetty tietokone on erilainen kuin varsinainen kohdejärjestelmän keskusyksikkö. Kehitysjärjestelmä on yleensä kaupallinen tietokone tai järjestelmä, kun taas kohde on sulautettu järjestelmä. Se voi olla erikoisprosessori, joka on suunniteltu vain kehitettävää järjestelmää varten. Järjestelmien erilaisuudesta johtuen testaus voidaan joutua jakamaan kahteen osaan. Kehitysjärjestelmässä testataan eräänlaista kehitysversiota ohjelmasta, koska siinä ei pystytä testaamaan varsinaista lopullista ohjelmaa. Järjestelmään toimitettavan ohjelman testaus voidaan suorittaa vasta lopullisessa järjestelmässä (/19/).

Sulautettujen järjestelmien laitteistojen ominaispiirteet voivat vaikeuttaa ohjelmiston testaamista lopullisessa järjestelmässä. Testausvaiheessa ohjelmaan useasti lisätään testausta mahdollistavia ja helpottavia piirteitä. Sulautetun järjestelmän laitteisto voi olla niin rajoittunut, ettei ohjelmaan ole mahdollista tehdä kaikkia testauksen kannalta tarpeellisia lisäyksiä. Esimerkiksi järjestelmän muistin rajoittuneisuus voi estää erilaisten tapahtumien ja tilojen tallentamisen myöhempää tarkastelua varten.

Sulautetun järjestelmän liitynnät ympäristöön vaikeuttavat sen testausta. Järjestelmän ympäristöllä on usein oma dynamiikkansa, ja sitä ei voi useinkaan palauttaa virheen sattuessa sitä edeltäneeseen tilaan. Näin virheiden toistaminen ja niiden ymmärtäminen sekä havaitseminen vaikeutuvat merkittävästi. Ympäristön tilasta johtuvia virhetilanteita on myös hyvin vaikea löytää lukuisten mahdollisten tilojen takia.

Reaaliaikaisen järjestelmän testaus on moninverroin hankalampaan kuin normaalien järjestelmien. Reaaliaikajärjestelmän toiminta saattaa riippua ajoituksista. Järjestelmän tilojen ajoituksia on hyvin vaikea saada testattua tai edes tuottaa erilaisia ajoituskombinaatioita. Järjestelmän sisäinen tila voi vaikuttaa sen toimintaan reaaliaikaisuutta vaativissa kohdissa. Myös erilaiset testijärjestelmät ja ohjelmaan tehdyt lisäykset saattavat muuttaa järjestelmän toimintaa niin paljon, ettei se vastaa enää todellista järjestelmää.

Sulautetuissa järjestelmissä joudutaan ohjelmiston lisäksi testaamaan myös laitteistoa ja näitä molempia yhdessä. Laitteiston testaukseen on omat järjestelmänsä ja mittalaitteensa. Yleiskäyttöisiä testauslaitteistoja ei ole kuitenkaan aina mahdollista käyttää lopullisen tuotteen testaukseen. Tällöin voidaan joutua valmistamaan ja kehittämään omat testilaitteistot, tai valmistamaan järjestelmästä erillinen versio testausta varten. Varsinkin herkkien järjestelmien testauksen ongelmaa lisää se, että mitta- tai testauslaitteistot saattavat vaikuttaa itse testattavan järjestelmän toimintaan.

Testausta on pyritty automatisoimaan sulautettujen järjestelmien testauksen parantamiseksi. Testauksen automatisointi tapahtuu erillisellä ohjelmalla, joka suorittaa siihen ohjelmoituja testejä testattavalle ohjelmalle. Testaus tapahtuu ohjelmiston rajapinnan kautta, joka simuloidaan niinkutsutulla rajapintasimulaattorilla. Testauksen automatisoinnilla pyritään parantamaan testauksen laatua, nopeuttamaan testauksen läpimenoaikaa ja pienentämään testauksen kustannuksia (/3/).

5. ALKUPERÄINEN HIHNA-ANALYSAATTORI

5.1 Analysaattorijärjestelmä

Kehitettävä QuarCon hihna-analysaattori on on-line eli jatkuvatoiminen röntgenflueresenssianalysaattori. Se mittaa alapuolellaan liikkuvalta kuljetinhihnalta siirrettävän materiaalin alkuainepitoisuuksia reaaliajassa. Analysaattori antaa mittauksen aikana hihnalla kulkeneen materiaalin keskimääräiset pitoisuudet jokaisen mittauksen jälkeen. Yhden mittauksen kestoksi voidaan valita sekunneista useampaan minuuttiin, ja mittauksen aikana voi kuljetinhihnalla kulkea mitattavaa materiaalia muutamasta tonnista aina kymmeniin tonneihin asti.

Varsinaisen analysaattorin mittapään eli proben pääosat ovat röntgenputki, ilmaisin eli detektori, ja väylällä yhteenliitetyt elektroniikkakortit. Analysaattori on liitetty kenttäliityntäosaan, jota kutsutaan nimellä PES (Probe Electronic Set). PES-yksikössä on muunmuassa analysaattorin paikalliset kytkimet ja toiminnan merkkivalot, liityntäyksikkö ulkoisille signaaleille, virtalähde sekä röntgenputken toiminnan vaatima korkeajännitegeneraattori.

Kokonaisjärjestelmä koostuu varsinaisesta analysaattorista ja siihen liitetystä tietokoneesta ja valvomo-ohjelmistosta. Tietokoneessa voidaan laskea erilaisia aika- ja tonnipainotteisia keskiarvoja, kerätä mittaushistoriaa ja kasatuloksia sekä tulostaa erilaisia raportteja. Analysaattorijärjestelmän tuloksia voidaan siirtää laitoksen muihin järjestelmiin, tai prosessin säätö- ja toimilaitteita on mahdollista ohjata suoraan järjestelmästä. Yhteen tietokoneeseen ja valvomo-ohjelmaan voidaan liittää yksi tai useampia analysaattoreita.

5.2 Mittaus

Hihna-analysaattori säteilyttää mitattavaa ainetta röngensäteillä. Röntgensäteiden lähteenä käytetään laitteessa olevaa röntgenputkea. Röntgenputkesta lähtevä primäärisäteily virittää mitattavan materiaalin alkuaineiden atomit. Atomien viritys purkautuu ja muodostaa mitattavissa olevaa säteilyä. Virityksen purkautumisesta muodostunut säteilyn energia on kullekin alkuaineelle tyypillinen, ja sitä kutsutaan alkuaineen karakteristiseksi säteilyksi eli fluoeresenssisäteilyksi.

Säteilykvantit havaitaan ilmaisimella eli detektorilla, jossa niiden energia mitataan. Mitatusta säteilystä muodostuu spektri, josta nähdään paljonko tietyn energian omaavia säteilykvantteja tulee detektorille tiettynä ajanjaksona. Havaittavan spektrin muoto ja sen huippujen korkeus riippuvat mitattavasta aineesta ja siinä olevien alkuaineiden pitoisuuksista.

Detektori on verrannollisuuslaskuri ja se laskee montako tietyn energian omaavaa säteilykvanttia sille tulee mittausaikana. Detektorin havaitsemia pulsseja vahvistetaan esivahvistimessa, jonka jälkeen ne lähetetään jatkokäsittelyyn analogisina signaaleina.

Detektorin lähettävät analogiset signaalit muutetaan tämän jälkeen digitaalisiksi. Mitatut pulssit jaoitellaan energiansa perusteella mikrokanaviin. Tämä tapahtuu antamalla kullekin pulssille lukuarvo väliltä 0-255 - pulssin energian mukaan. Mittauksen aikana tulleet pulssit eli mikrokanavanumerot kerätään ja lasketaan yhteen kullekin mikrokanavalle tulleet pulssit.

Mikrokanavien muodostamasta taulukosta saadaan mittauksen aikana kerätty spektri. Tietystä alkuaineesta tulleet pulssit kertyvät omalle mikrokanava-alueelleen. Näille alueille pystytään määrittelemään ylä- ja alarajat ja niitä kutsutaan kanaviksi. Kukin kanava edustaa yhdeltä alkuaineelta tai tietyltä

sironnalta tulleita säteilykvantteja. Laite pystyy näinollen mittaamaan useita eri alkuaineita samanaikaisesti.

Mittauksen loputtua lasketaan kullekin kanavalle, eli mikrokanava-alueelle, kertyneet pulssit yhteen. Jakamalla kanavan pulssit mittausajalla, saadaan kanavan pulssitaajuus (pulssia/sekuntti) eli intensiteetti. Analysaattori voi tämän jälkeen lähettää intensiteetit suoraan ulkopuoliselle järjestelmälle, tai suorittaa itse alkuaineiden pitoisuuslaskennat, ja lähettää valmiit tulokset. Mittaustulokset voidaan välittää ulkopuoliseen järjestelmään suoraan sarjaväylän kautta tai muuntamalla ne analogisiksi virtaviesteiksi.

5.3 Analysaattorin mittapään yksiköiden toiminta

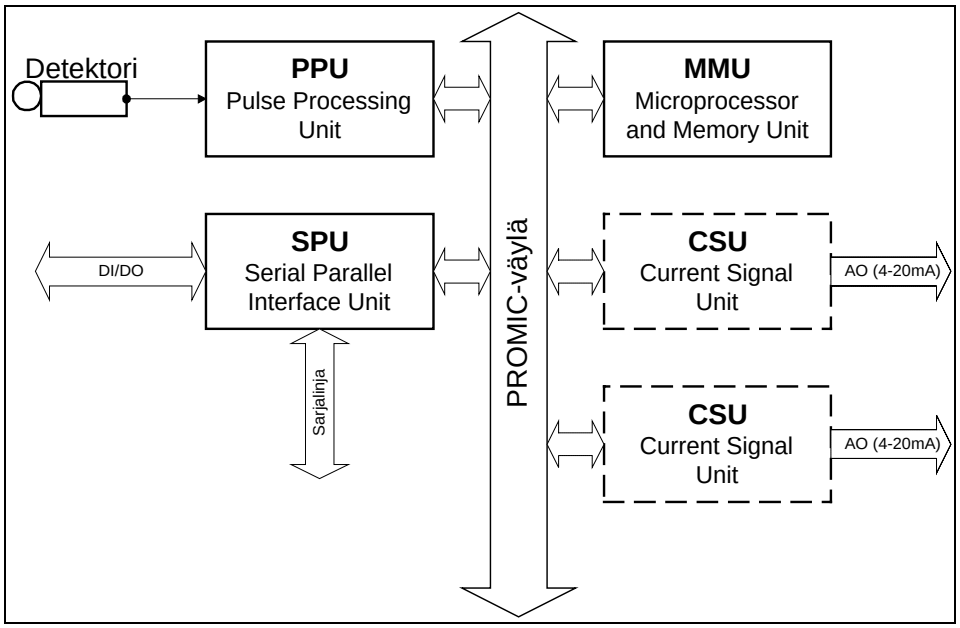
Analysaattorin mittapäässä on röngenputken ja detektorin lisäksi analysaattorin elektroniikkakortit. Analysaattorin elektroniikkakortit on suunniteltu pääosin 80-luvulla Outokumpu Oy:ssä ja ne on liitetty toisiinsa passiivisella väylällä. Samoja kortteja käytetään ja on käytetty myös muissa erilaisissa analysaattoreissa.

Kullakin kortilla on joukko registereitä, joiden kautta niiden toimintaa voidaan ohjata tai siirtää tietoa niiden ja keskusyksikön välillä. Korttien registereiden alkuosoite voidaan asettaa halutuksi kullakin kortilla olevalla erillisellä siltausyksiköllä. Joillakin korteilla on registereiden lisäksi keskeytyslinjoja, joiden avulla ne saavat lähetettyä keskeytyspyyntöjä keskusyksikölle.

5.3.1 PROMIC väylä

Analysaattorin elektroniikkakortit ovat kytketty toisiinsa PROMIC-väylällä. PROMIC-väylä on Outokumpu Oy:n 70-luvun lopulla analysaattoreihinsa kehittämä väylä. Väylän osoite- ja dataväylä perustuvat sellaisenaan Motorollan 6800 prosessoriperheen signalointiin. Väylällä on muunmuassa 16 bitin

osoiteväylä, 8 bitin dataväylä, 11 kappaletta kontrollisignaaleja ja 8 keskeytyssignaalia.



Kuva 3. Lokokuva analysaattorin mittapään elektroniikka yksiköistä

5.3.2 Microprocessor and Memory Unit (MMU)

MMU-kortti on analysaattorin, olemassa oleva, vanha, keskusyksikkö. Se hoitaa kaikki analysaattorin sisäiset ohjaukset ja laskennat. MMU-kortti on varustettu Motorollan M6800P mikroprosessorilla, 32k EPROM ja 8k CMOS-RAM muistilla. MMU-kortti on yhteydessä muihin kortteihin lukemalla ja kirjoittamalla niiden rekistereitä PROMIC-väylän kautta.

MMU-kortin ohjelma on kirjoitettu Asembler kielellä pääosin 80-luvun aikana. Viimeisimmät muutokset ja korjaukset ohjelmaan on tehty 90-luvun alussa. Tämän työn toteutusvaiheessa MMU-kortin ohjelmasta ei ollut käytännössä enää jäljellä muita dokumentteja kuin lähdekoodi. Analysaattorin ohjelman sisältäviä EPROM-muisteja kopioidaan uusia MMU-kortteja varten, eikä ohjelmaan ole käytännössä mahdollista tehdä muutoksia tai korjauksia.

Tämän työn kohteena on juuri korvaavan ratkaisun löytäminen MMU-kortille ja ohjelman tekeminen uuteen toteutukseen.

5.3.3 Pulse Processing Unit (PPU)

PPU-kortti on yhteydessä detektoriin ja saa sen havaitsemat säteilykvantit analogisina signaaleina. MMU-kortti hoitaa mittauksen käynnistämisen ja pysäyttämisen kirjoittamalla PPU-kortille mittauksen ohjausta varten varattuun registeriin. Mittauksen ollessa käynnissä PPU-kortti suodattaa ja muuntaa analogiset mittauspulssit digitaalisiksi arvoiksi, antamalla kullekin säteilykvantille sen energian perusteella mikrokanava-arvon.

PPU-kortilla on 16 tavun FIFO (First In First Out) muisti, johon mittaustieto väliaikaisesti tallennetaan. Kortti ilmoittaa keskeytyssignaalilla MMU-kortille, kun sen FIFO muisti on puoliksi täynnä. Keskytyssignaalin havaittuaan MMU-kortti lukee PPU-kortin FIFOn tyhjäksi, ja lisää havaitut mikrokanava-arvot, eli mitatut energiakvantit, ylläpitämäänsä mittauksen aikaiseen spektriin.

Kunkin mittaussignaalin käsittely detektorissa ja PPU-kortilla aiheuttaa “kuollutta-aikaa”. Kuollut aika alkaa, kun energiakvantti absorboituu detektoriin ja loppuu, kun järjestelmä on valmis vastaanottamaan seuraavan pulssin. Tämän ajanjakson aikana saapuneita uusia mittaussignaaleja ei havaita, vaan ne menetetään. Kuolleen ajan takia analysaattori ei pysty havaitsemaan kaikkia saapuvia energiakvantteja, ja näin mitattavasta aineesta kerätty kokonaispulssimäärä on pienempi kuin todellinen pulssimäärä.

Mittausaikana kertyneen kuolleen ajan aiheuttama virhe kompensoidaan kasvattamalla mittausaikaa. Analysaattorin yhden mittauksen aika on aina todellisuudessa hieman pidempi, kuin sille parameterillä määrätty “tehollinen” mittausaika. Kuolleen ajan kompensoimiseksi MMU-korttin ohjelma ei käytä reaaliaikakelloa analysaattorin mittausajan määrittämiseksi, vaan laskee

mittausajan PPU-korttin muodostamista keskeytyssignaaleista. Periaatteessa PPU-kortti generoi keskeytyssignaalin yhden sekunnin välein. Todellisuudessa tämä aika on hieman pidempi, koska PPU-kortti lisää siihen kyseisen ajanjakson aikana syntyneen kuolleen ajan. MMU-kortti kerää ennen yksittäisen mittauksen lopettamista näitä keskeytyssignaaleita yhtä monta, kuin sille on sekunneissa määritelty mittausaikaa. Näin toteutunut mittaus sisältää yhteensä annetun tehollisen mittausajan, jonka aikana kaikki saapuvat energiakvantit havaitaan.

Analysaattorin toimintaan kuuluu sen antamien tulosten pitäminen keskenään mahdollisimman vertailukelpoisina pitkälläkin aikajaksolla. MMU-korttin ohjelma voi muuttaa, PPU-kortin kautta, detektorin korkeajännitettä analysaattorissa tapahtuvien muutosten kompensoimiseksi. Ilman kompensointia analysaattorin antamat tulokset eivät pitkällä tähtäimellä säilyisi stabiileina, esimerkiksi detektorissa ja elektroniikkakorteilla tapahtuvien muutosten sekä niiden ikääntymisen takia.

Analysaattorissa tapahtuneiden muutosten havaitsemiseksi sillä mitataan säännöllisin väliajoin tunnettua vakionäytettä. Tätä toimenpidettä kutsutaan referenssimittaukseksi ja tunnettua näytettä referenssilevyksi. MMU-kortin ohjelma vertaa referenssimittauksen tulosta aikaisemmin asetettuun alkutilanteeseen, ja säätää detektorin jännitteen niin, että analysaattorin tulokset ovat mahdollisimman lähellä alkuperäistä tilannetta. Detektorin korkeajännitettä säädetään kirjoittamalla PPU-kortin rekisteriin niinsanottu “gain” arvo, joka voi saada kokonaislukuarvon väliltä 0-255.

PPU-kortti säilyy sellaisenaan myös uudessa toteutuksessa. Kortin toiminta liittyy oleellisesti analysaattorin mittaustapahtumaan ja kiinteästi detektorin toimintaan sekä sen ohjaukseen. Sen muuttaminen tai korvaaminen ei ole ajankohtaista myöskään analysaattoriin myöhemmin toteutettavissa jatkokehityksessä. Koko analysaattori joudutaan suunnittelemaan uudestaan,

mikäli PPU-kortti korvataan jossakin vaiheessa uudella yksiköllä. Tällöin on järkevää luopua kokonaan myös nykyisestä PROMIC-väylästä.

5.3.4 Serial Parallel Interface Unit (SPU)

SPU-kortin kautta tapahtuu kaikki tiedonsiirto analysaattorin ja ulkomaailman välillä. Kortilla on kaksi sarjaväylää, joiden sarjaliikenne voidaan valita erillisellä yksiköllä joko RS-232 tai RS-422 muotoon. Kumpaakin sarjaväylää varten SPU kortilla on neljä rekisteriä ja kaksi keskeytyslinjaa. Kehitettävässä QuarCon hihna-analysaattorissa ei ole koskaan käytössä kuin toinen sarjaväylistä.

Sarjaliikenteen lisäksi kortilla on yksitoista digitaalista tuloa ja kahdeksan lähtöä. Kolmen sisääntulon signaalit voidaan yhdistää keskeytyslinjoihin, jolloin niiden aktivoituminen voisi käynnistää keskeytysohjelman. QuarCon analysaattorissa näitä sisääntuloja ei käytetä keskeyttävinä, vaan ne ovat samassa asemassa muiden kahdeksan digitaalisen sisääntulon kanssa. Digitaalisten sisääntulojen tila voidaan lukea ja lähtöjen tila kirjoittaa, käyttäen SPU-kortilla olevia rekistereitä.

Digitaalisia lähtöjä käytetään esimerkiksi analysaattorin paikallisten merkkivalojen ja referenssimittauksen ohjauksessa. Digitaalisien tulojen avulla analysaattoria voidaan ohjata paikallisilla kytkimillä, ja sen ohjelmalle voidaan välittää erilaisia tila- ja hälytyssignaaleja. Analysaattorin digitaaliset I/O-signaalit kytketään SPU-kortilta PES-yksikössä olevaan WIU-korttiin. WIU-kortilla on toteutettu signaalien galvaaninen erottaminen.

SPU-korti oli tarkoitus säilyttää käytössä ainakin tähän työhön kuuluvassa analysaattorin kehityksen ensimmäisessä vaiheessa. QuarCon analysaattorissa käytössä olevat digitaaliset I/O-signaalit tuli hoitaa nykyisen SPU-WIU yhdistelmän avulla. Analysaattorin kommunikointi pystyttäisiin hoitamaan

SPU-kortin kautta, tai sitten käyttämällä uuden keskusyksikön tarjoamia ratkaisuja.

5.3.5 Current Signal Output Unit (CSU)

Analysaattorin tulokset voidaan halutessa välittää analogiasignaaleina CSU-kortin kautta. Kortti muuntaa MMU-yksikön sille kirjoittaman digitaalisen signaalin analogiseksi. CSU-kortilla on neljä kappaletta galvaanisesti erotettuja 4-20mA lähtöjä. Näiden lisäksi kortilla on kaksi muuta analogialähtöä ja kuusi digitaalilähtöä, mutta niitä ei käytetä QuarCon analysaattorissa. Kortteja voi olla kerralla käytössä yksi tai kaksi kappaletta.

Toteutettavan järjestelmän tuli pystyä käyttämään CSU-kortteja, koska niitä on käytössä monissa vanhoissa järjestelmissä. Tulevaisuudessa analogialähtöihin voidaan käyttää CSU-kortteja, tai ne voidaan mahdollisesti korvata uuden keskusyksikön tarjoamilla uusilla ratkaisuilla.

6. TEKNINEN RATKAISU

6.1 Uuden keskusyksikön toimintavaatimukset

Tähän työhön kuului hihna-analysaattorin kehityksen ensimmäisen vaiheen toteutus. Ensimmäisessä vaiheessa vaihdettiin analysaattoriin uusi keskusyksikkö ja toteutettiin keskusyksikölle perusohjelma. Myöhemmin toteutettavassa kehityksessä analysaattorin ominaisuuksia lisätään uuden sukupolven laitteita vastaavaksi, ja keskusyksikön ohjelman toimintaa monipuolistetaan.

Ensimmäiseksi toteutetun ohjelman tuli toimia kuten vanha MMU-kortin ohjelma, ilman mitään uusia ominaisuuksia. Tämä vaihe haluttiin toteuttaa, koska eräät MMU-kortin vanhassa ohjelmassa olevat virheet tuli saada korjatuiksi. Näitä virheitä ei käytännössä ollut mahdollista korjata vanhan MMU-kortin ohjelmaan. Kortin ohjelmassa olevien virheiden lisäksi joidenkin siinä käytettyjen vanhojen komponentien saatavuus alkoi olla epävarmaa. Ei ollut enää varmuutta kuinka kauan MMU-kortista pystyttäisiin valmistamaan uusia sarjoja.

Vaiheittaisella toteutuksella haluttiin varmistaa vanhojen järjestelmien varaosasaatavuus keskusyksikön suhteen. Ensimmäisessä vaiheessa toteutettavaa uutta keskusyksikköä oli tarkoitus pystyä käyttämään tarvittaessa MMU-kortin varaosana jo asennetuissa analysaattoreissa. Vanhoihin analysaattorijärjestelmiin ei tarvitsisi tehdä mitään toiminnallisia muutoksia, koska uusi ensimmäisen vaiheen ohjelma toimisi samoin kuin vanha MMU-kortin ohjelma. Uudessa toteutuksessa tuli pyrkiä tekniseen ratkaisuun, joka mahdollistaisi uuden keskusyksikön vaihtamisen vanhan MMU-kortin tilalle aikaisemmin asennettuihin analysaattoreihin.

Analysaattorin keskusyksikön uudistamisen ensimmäisessä vaiheen ratkaisussa tuli kuitenkin ottaa huomioon myöhempien vaiheiden asettamat vaatimukset, vaikka niiden toteutus ei tähän työhön kuulunutkaan. Ensimmäisen vaiheen ratkaisun täytyi mahdollistaa myöhemmin järjestelmään toteutettavat uudet ominaisuudet.

Samaa vanhaa ohjelmistoa, MMU-korttia ja muita elektroniikkakortteja käytetään myös toisessa analysaattorissa. Uutta keskusyksikköä on tarkoitus käyttää myöhemmin myös tässä IMACON analysaattorissa. Tämän johdosta uuden keskusyksikön tuli mahdollistaa myös tämän toisen analysaattorin vaatimat erilaiset toiminnot, jotka toteutetaan uudelle keskusyksikölle myöhemmin.

Mittausteknisesti IMACON analysaattori toimii aivan samoin kuin QuarCon hihna-analysaattorikin. IMACON analysaattoria käytetään hienojakoisen aineen mittaamiseen, joka virtaa tai on pysäytettynä analysaattorin etuseinään sijoitettuun mittakennoon. Näytteenkäsittely ja useampien näytelinjojen mittaaminen tuovat mukanaan joitakin vaatimuksia, joita ei ole tarvinut ottaa huomioon hihna-analysaattorin toteutuksessa.

Analysaattorin toiminnalle asetettuja myöhemmin toteutettavia ominaisuuksia:

- Mahdollisuus käyttää yhdessä analysaattorissa kahta detektoria ja PPU-korttia samanaikaisesti. Ratkaisulla voidaan parantaa mittaustarkkuutta tai mitata kahta erillistä energia-aluetta samanaikaisesti.

  Kahden PPU-kortin käyttö edellyttää keskusyksiköltä kaksinkertaista nopeutta keskeytysten käsittelyssä ja mittaustulosten lukemisessa. Lisäksi kummankin PPU-kortin tulokset tulee käsitellä ja laskea erikseen.

- Mahdollisuus käyttää kahta eri röntgenputken jännitettä mittauksessa.

Keskusyksikön pitää pystyä ohjaamaan ja tarkkailemaan röntgenputken jännitettä.

- Vapaammin aseteltavat laskentamallit ja mahdollisuus käyttää useampaa erilaista laskentayhtälöä (“multimodel”) alkuainepitoisuuksien laskemisessa.

  Laskennassa pitää pystyä käyttämään intensiteettikanavia vapaammin ja normaaleja laskutoimituksia (jako-, kerto-, yhteen- ja vähennyslaskua) näiden välillä. Käyttäjän pitää pystyä määräämään ja asettamaan nämä yhtälöt ja ehdot analysaattorin käyttöliittymän kautta.

- Laskennan on otettava huomioon mikäli kuljetinhihnalla ei ole tarpeeksi materiaalia mittauksen aikana.

  Mittauksen aikana pystyttävä tarkkailemaan hihnavaa’an arvoa. Mittaustulosten laskentoja korjataan, tai ne saatetaan hylätä riippuen hihnavaa’an tuloksista.

- Lämpötilakorjauksen laskeminen analyyseihin.

  Mittauksen aikana tarkkaillaan lämpötila-anturia ja analysaattorin tuloksia korjataan lämpötilalla.

- Säilytettävä tietty määrä vanhoja analyysituloksia.

  Analysaattorin on säilytettävä lyhyt historia analyysituloksista. Lisäksi pieni määrä spektrejä ja referenssimittaustuloksia tulee säilyttää analysaattorin muistissa.

- Automaattinen referenssimittaus.

  Analysaattori tekee automaattisesti referenssimittauksen. Analysaattorin pitää pystyä tarkkailemaan, että kuljetinhihna ei ole käynnissä ja ettei kuljetinhihnalla ole materiaalia. Tämän jälkeen käynnistetään referenssilevyn moottori siirtämään referenssilevy

referenssimittausasentoon ja tehdään tarvittava määrä referenssimittauksia.

Edellä luetellut myöhemmin toteutettavat ominaisuudet vaikuttivat uuden toteutuksen laajennettavuuteen. Useimmat ominaisuudet lisäävät analysaattorin käyttämien digitaalisten sekä analogisten lähtöjen ja tulojen määrää. Analysaattorin uuden keskusyksikön tulee olla myös nopeudeltaan riittävä suoriutuakseen uusista ominaisuuksista.

Myöhemmässä vaiheessa analysaattoriin joudutaan mahdollisesti tekemään myös uusi komentokieli tai käyttöliittymä, koska uudet ominaisuudet lisäävät analysaattorin parametroitavia asioita ja parametrien määrää melkoisesti. Joidenkin uusien ominaisuuksien ohjaus ja parametrointi voi olla vanhalla komentorakenteella vaikeaa tai mahdotonta toteuttaa ja hankalaa käyttää.

6.2 Uuden keskusyksikön valinta

Kuten alussa mainittiin, työn lähtökohtana oli pyrkiä korvaamaan vanha MMU-kortti standardilla kaupallisella keskusyksiköllä. Kaupallisen kortin käyttäminen oli hyvin perusteltua tässä tapauksessa, koska keskusyksiköiden menekki vuositasolla on maksimissaan vain muutama kymmenen kappaletta. Näin pienen sarjan ollessa kyseessä oman laitteiston suunnittelu, valmistus sekä ylläpito tulisivat olemaan huomattavasti kalliimpia ja kiinteitä kustannuksia lisääviä, kuin valmiin kaupallisen vaihtoehdon käyttäminen. Lisäksi yrityksen teknologiastrategiana on keskittyä sovellusosaamiseen ja välttää resurssien sitomista esimerkiksi laitteiston ylläpitoon, joka on mahdollista toteuttaa valmiilla komponenteilla.

6.2.1 PC-pohjaiseen arkkitehtuuriin päätyminen

Valmiiden kaupallisten keskusyksiköiden vertailussa valittiin jo melko aikaisessa vaiheessa PC-pohjainen prosessoriratkaisu. Erilaisten PC-pohjaisten laajennusyksiköiden suuri määrä varmisti sen, että myöhemmin halutut ominaisuudet voidaan toteuttaa kaupallisilla komponenteilla. Laajasta valikoimasta pystytään valitsemaan sopivat moduulit myöhemmin päätettävälle ratkaisulle eikä toteutuksessa jouduta tekemään kompromisseja tarjonnan rajoittuneisuuden takia.

Uuden keskusyksikön tuli olla mahdollisimman hyvin tunnettu ja yksinkertainen. Tavoitteeksi oli asetettu, että analysaattorin keskusyksikön jatkokehitys ja ylläpito olisivat yksinkertaisia tehdä. Vanhasta MMU-kortista oli muodostunut eräänlainen “mustalaatikko”, jonka kehitys ei enää ollut taloudellisesti kannattavaa, koska järjestelmän alkuperäisiä kehittäjiä ei ollut käytettävissä. PC-arkkitehtuurin tunnettavuudella haluttiin varmistaa, että järjestelmän myöhempi kehitystyö olisi mahdollisimman helppoa myös ilman järjestelmän alkuperäisiä toteuttajia. Näin uusien henkilöiden olisi helpompi jatkaa kehitystyötä, ja aloituskynnys työn jatkamiselle olisi alhaisempi.

Merkittävänä osatekijänä PC-pohjaisen ratkaisun valintaan vaikuttivat saatavissa olevat kehitystyökalut ja kehitysympäristö. Laajan ja laadukkaan valikoiman lisäksi myös kehitystyökalut tukivat tunnettavuutensa johdosta järjestelmän myöhempää avointa kehitystä. Lisäksi saatavilla olevat kääntäjät ja muut kehitystyökalut olivat halpoja verrattuna monien muiden laitteistoratkaisujen vastaaviin ohjelmiin.

Suurimpana riskinä ja haittatekijänä PC-arkkitehtuurin käytölle pidettiin tuotteiden lyhyttä elinkaarta verrattuna analysaattorin suunniteltuun elinkaareen. Tähän ongelmaan päätettiin varautua valitsemalla käyttöön vaihtoehto, jolla olisi useita korvaavia toimittajia. PC-korttien yhdeksi eduksi katsottiin juuri

niiden hyvin laaja valmistajaryhmä. Tämän lisäksi uusi järjestelmä pyrittiin toteuttamaan mahdollisimman laitteistoriippumattomaksi.

6.2.2 PC/104-kortin valinta

Analysaattorin toimintaympäristö asetti valittavalle keskusyksikölle ympäristön sietovaatimuksia. Analysaattorit asennetaan kuljetinhihnan päälle ja usein hyvin lähelle suuria murskaimia, jonka johdosta niihin kohdistuu voimakkuudeltaan ja taajuudeltaan vaihtelevaa tärinää. Tämän lisäksi analysaattorin toimintaympäristön lämpötilavaihtelut voivat olla hyvin suuret. Analysaattoreita asennetaan ympäri maapalloa, jonka lisäksi yhden analysaattorin ympäristön lämpötila voi vaihdella yli viisikymmentäkin astetta.

Ympäristöolosuhteiden sietokyky oli yksi tärkeimmistä syistä siihen että, päädyttiin valitsemaan PC/104-CPU uudeksi analysaattorin keskusyksiköksi. Normaaleihin PC-kortteihin verrattuna PC/104-kortien ympäristöolosuhteiden kesto on parempi. Tämä johtuu muunmuassa siitä, että ne on alunperinkin suunniteltu käytettäväksi vaikeammissa olosuhteissa. Tärinänsietokyky on erikoisen väylän ansiosta merkittävästi parempi kuin passiiviväylän korteilla. Tämän lisäksi korttien komponenttien kiinnitys on tehty paremmin kuin normaaleilla PC-korteilla.

Kuten edellä mainittiin PC/104-väylä on ISA-standardin muunnos. Aluperin PC/104-kortteja valmisti Ampro Computers -niminen yhtiö vuodesta 1987 alkaen. Muodollinen määrittely korteille julkaistiin vuonna 1992. Korttien väylä muodostuu niiden laidassa olevasta kaksiosaisesta 40- ja 64-nastan piikkirimaliittimestä, joiden lukumäärästä korttityyppi saa nimensä. Väylällä on kaikki 96 ISA-väylän signaalia ja se on arkkitehtuuriltaan, laitteistoltaan ja ohjelmistoltaan täysin yhteensopiva ISA-väylän kanssa. Ylimääräiset kuusi liitintä on kytketty nollapotentiaaliin, mikä lisää korttin häiriösietoisuutta. Kortin toisella puolella liittimet ovat uros- ja toisella naaras-puoleisia. Erilaisia

kortteja voidaan pinota päällekäin muodostaen näin PC-yhteensopiva tietokone. PC/104-korttien toinen yleinen käyttötapa on käyttää niitä muiden komponenttien tapaan piirilevyllä, jossa on PC/104-väylä (/1/, /12/).

Yhtenä uuden keskusyksikön vaatimuksena oli, että sillä toteutettu ratkaisu olisi käyttökelpoinen varaosa vanhoissa analysaattoreissa. Tämä asetti rajoituksia valittavan toteutuksen koolle. PC/104-korttien pieni koko oli etuna suunniteltaessa ratkaisua, joka mahdollistaisi uuden toteutuksen käytön myös vanhoissa järjestelmissä.

Uuden keskusyksikön elinkaarta ajatellen PC/104-korttin valinta oli myös turvallisempi kuin jonkin muun PC-pohjaisen ratkaisun. PC/104-kortit on alunpitäenkin suunniteltu sulautettuihin järjestelmiin, ja näin niiden saatavuus on varmistettu toimittajien taholta normaaleja PC-kortteja paremmin ja pidemmällä aikajänteellä. Lisäksi standardi mahdollistaa muiden valmistajien tuotteiden käytön, jolloin ei jouduta sitoutumaan yhteen valmistajaan, kuten jossakin valmistajakohtaisessa ratkaisussa.

6.3 Väyläsovitin

6.3.1 Väyläsovituksen periaate

Uuden keskusyksikön toimintaedellytyksenä oli pystyä liittymään, ja toimimaan yhdessä vanhojen PROMIC-väylän elektroniikkakorttien kanssa, koska analysaattorin muita yksiköitä ei ollut tarkoitus uusia. Tämän johdosta jouduttiin liittämään toisiinsa kaksi erilaista väylää. Kahden erilaisen väylän ja tarvittavien signaalien liittämiseksi suunniteltiin ja toteutettiin erillinen väyläsovitin.

Väyläsovittimen tarkoituksena on muuttaa kahden eri väylän sähköiset signaalit ja ajoitukset toisilleen yhteensopiviksi. Sen avulla uusi PC/104-väyläinen

keskusyksikkö pystyy lukemaan ja kirjoittamaan PROMIC-väylän korttien muistialueita. Väyläsovitin tarjoaa läpinäkyvän yhteyden, jonka kautta PROMIC-kortien registerit näkyvät keskusyksikön muistiavaruuden osana.

6.3.2 Väyläsovittimen toiminta

Uusi keskusyksikkö käyttää PROMIC-väylän vanhojen korttien registereitä oman I/O-avaruuden ikkunansa kautta. Sovitinkortti mahdollistaa 128 peräkkäisen I/O-registerin käytämisen siihen kytkettyyn PROMIC-väylään sijoitetuilta korteilta. PC/104-väylän I/O-osoitukset noudattavat ISA-väylän käytäntöä perustuen 10-bittisiin osoitteisiin ja yhden kilon osoiteavaruuteen. Käytetyistä kymmenestä osoitesignaalista on SA0-SA6 kytketty suoraan PROMIC-väylän osoitteisiin A0-A6. Loput PROMIC-väylän osoitesignaaleista A7-A15 on aseteltavissa sovitinkortin siltauksilla haluttuun asentoon.

PROMIC-väylän kellosignaali tehdään sovitinkortilla olevalla kellogeneraattorilla. Järjestelyn seurauksena PC/104-väylän jaksot ajoittuvat sykronoimattomasti PROMIC-väylän perustahtiin verrattuna. Tämä aiheuttaa pientä vaihtelevaa viivettä luku- ja kirjoitusjaksojen alkupuolella, kun joudutaan odottamaan, että PROMIC-väylän kellotuksessa esiintyy seuraava sovelias jakson aloitustilanne. Kaikki sovitinkortista aiheuttuvat viiveet PC/104-kortilla hoidetaan käyttäen sen wait-tiloja.

PROMIC-väylän keskeytyssignaalit liitetään sovitinkortin kautta PC/104-kortin keskeytyssignaaleihin. PROMIC-väylän keskeytykset joudutaan invertoimaan sovitinkortilla, koska ne ovat 0-aktiivisia ja PC/104 väylän signaalit ovat 1-aktiivisia. Joustavan käytön ja tulevaisuuden jatkokehityksestä johtuen, PROMIC-väylän kahdeksaa keskeytystä ei ole kytketty kiinteästi tiettyihin PC/104-kortin keskeytyksiin vaan keskeytysten fyysistä kytkemistä varten sovitinkortilla on erillinen siltauslohko. Siltauslohkolla tarvittavat PROMIC-

väylän keskeytykset voidaan kytkeä vapaasti PC/104 kortilta valittuihin IRQ5, IRQ7, IRQ9-IRQ12, IRQ14 ja IRQ15 keskeytystuloihin.

PC/104 keskusyksikkö tarvitsee 5 voltin käyttöjännitteen toimiakseen. Sovitinkortilla on mahdollista valita siltauksin, ottaako PC/104-kortit käyttöjännitteensä PROMIC-väylästä vai erillisestä virtalähteestä. Tähän ratkaisuun päädyttiin, koska tulevaisuudessa PROMIC-väylän virtalähteen teho ei välttämättä riitä kaikille myöhemmin mahdollisesti käytteenotettaville PC/104-väylän korteille.

6.3.3 Sovitinkortin toteutus

Sovitinkortin varsinainen suunnittelu ei kuulunut tähän työhön. Kortin toiminta- ja vaatimusmäärittelyjen jälkeen varsinainen suunnittelutyö teetettiin ulkopuolisena alihankintana. Tämä suunnittelutyö sisälsi muunmuassa tarvittavat väyläajoituksien tarkistusmittaukset ja varsinaisen piirikaavion toteutuksen.

Sovitinkortin toiminta varmistettiin aluksi erillisellä protokytkennällä, joka oli toteutettu “räppäämällä”. Tällä tavoin haluttiin säästää aikaa ja mahdollisesti yksi protopiirilevyn valmistuskierros. Sovitinkortin sähköisen suunnittelun jälkeen pystyttiin nopeasti valmistamaan kytkentä, jonka avulla uusi PC/104 pohjainen keskusyksikkö saatiin liitettyä PROMIC-väylään. Protokytkennällä testattiin varsinainen väyläsovittimen toiminta ja uuden keskusyksikön ohjelman testaus oli myös mahdollista todellisessa ympäristössä. Logiikka-analysaattorin avulla varmistettiin signaalien ja väylien ajoitukset.

Varsinainen protokortti valmistettiin sen jälkeen, kun protokytkennällä oli varmistettu sovitinkortin toiminta. Kortti mitoitettiin entisten PROMIC-korttien muotoiseksi. Uusi keskusyksikkö sijoitettiin sovitinkortilla olevaan PC/104

liittimeen. Sovitinkortti ja siinä oleva uusi keskusyksikkö laitettiin vanhojen analysaattorikorttien kanssa samaan alkuperäiseen PROMIC-kehikkoon.

6.4 Keskusyksikön muistiratkaisu

Uudelle keskusyksikkölle tarvittiin pysyvää muistia käyttöjärjestelmää, sovellusohjelmaa ja ohjelman parametreja varten. Haitumattomalle muistille tuli pystyä kirjoittamaan myös ohjelman suorituksen aikana, koska käyttäjällä on oltava mahdollista muuttaa analysaattorin parametreja milloin tahansa. Lisäksi tulevaisuudessa analysaattorin toimintaan oli tarkoitus lisätä historiankeruuta.

Tavallisen mekaanisen levyaseman käytön sulki pois analysaattorin toimintaympäristössä mahdollisesti oleva tärinä. Mekaanisen levyaseman toimintavarmuuden ja kestävyyden ei katsottu riittävän analysaattorin toimintaympäristöön. Haitumattoman muistin valinnassa päädyttiin flash-muistin käyttöön. Siinä yhdistyi uudelleenkirjoitettavuus ja ympäristöolosuhteiden sietokyky.

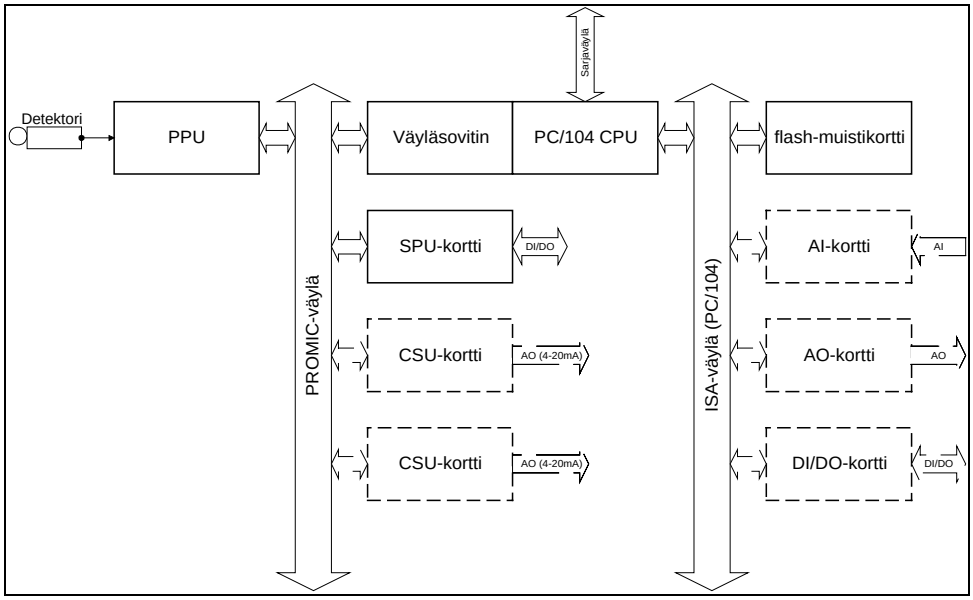
Toteutukseen valitulla PC/104-kortilla oli suoraan kanta flash-muistipiiriä varten. Tämän käyttö ei kuitenkaan ollut mahdollista, koska kyseiseen kantaan oli saatavilla työn toteutusvaiheessa vain kerralla kokonaan uudelleen kirjoitettavia flash-piirejä. Tämän johdosta päädyttiin valitsemaan PC/104-väylään liitettävä flash-muistikortti, joka emuloi kiintolevyä. Se toimii ja näkyy ohjelmistolle aivan kuten normaali kiintolevy.

Flash-muistikortia oli mahdollista kirjoittaa osissa ja käyttöjärjestelmän käynnistys oli mahdollinen kortilta. Tulevaisuudessa varauduttiin siirtymään PC/104 kortin kannalle asetettavaan piirin, kun piirin ominaisuudet kehittyisivät erillistä flash-muistikorttia vastaaviksi. Ratkaisun etuna olisi, että sovitinkortilla oleva PC/104-korteista muodostunut pino tulisi yhtä yksikköä matalammaksi.

Flash-muistikortin ohjelmointiin eli sille kirjoittamiseen tarvitaan 12 voltin jännite. Tällaistä jännitettä ei ollut saatavilla analysaattorin virtalähteestä. Jännite muodostettiin sovitinkortilla PROMIC-väylän 15V jännitteestä.

6.5 Laitteistoratkaisu

Analysaattorin toteutuksessa päädyttiin laitteistoratkaisuun, jonka lohkokaavio on kuvassa 4. Tähän työhön kuuluva minimitoteutus muodostuu vanhasta PROMIC-väylän kehikosta, jossa on vanhojen PPU-, SPU- ja mahdolisten CSU-korttien lisäksi uusi sovitinkortti. Sovitinkortin päälle on asetettu PC/104-väylän keskusyksikkö ja sen päälle flash-muistikortti. Sovitinkortti ja PC/104-yksiköt muodostavat kokonaisuuden, joka vie PROMIC-kehikosta kahden normaalipaksuisen kortin tilan.



Kuva 4. Lohkokaavio analysaattorin uudesta toteutuksesta

Uudessa järjestelmässä PPU-kortti toimii täysin vanhalla tavalla hoitaen detektorilta tulleiden signaalien muuntamisen mikrokanavanumeroiksi, ja kompensoimalla mittauksessa esiintyvän “kuolleen ajan”. SPU-kortin kautta hoidetaan kaikki aikaisemmat digitaaliset I/O-signaalit entiseen tapaan. CSU-

kortteja on mahdollista käyttää silloin, kun analysaattorin halutaan antavan ulos tuloksia myös analogisessa muodossa.

Sarjaliikenne päätettiin siirtää kulkemaan SPU-kortin sijasta PC/104-keskusyksikön oman sarjaportin kautta. PC/104-kortin olisi pitänyt palvella useampia PROMIC-väylän keskeytyksiä, mikäli sarjaliikenne olisi hoidettu entiseen tapaan SPU-kortin kautta. Sarjaliikenteen käsittelemisen katsottiin tulevan yksinkertaisemmaksi, kun se tehdään uuden keskusyksikön sarjaportin kautta.

Analysaattorin kommunikoinnin jatkokehitys tulevaisuudessa tuli mahdolliseksi valitun sarjaliikenteen toteutuksen ansiosta. Tulevaisuudessa ei tarvitse tyytyä SPU-kortin tarjoamiin ominaisuuksiin vaan koko kommunikointi voidaan tarvittaessa toteuttaa täysin uudella tavalla. Sarjaliikenteen uuden toteutuksen johdosta analysaattoriin täytyy lisätä modeemi tai linjamuunnin, joka muuntaa PC/104-kortin sarjaportin RS-232 signaalin paremmin häiriöitä sietävään ja pidemmän matkaa siirrettävään muotoon.

PC/104-väylän puolelle on jatkossa mahdollista lisätä erilaisia I/O-kortteja, jotka mahdollistavat analysaattoriin myöhemmin toteutettavat ominaisuudet. Tällöin on myös mahdollista korvata entinen SPU-kortti digitaalisten I/O-korttien avulla. SPU-kortillahan ei ole enää muuta käyttöä, kun analysaattorin kommunikointi menee suoraan uuden PC/104-kortin sarjaporttiin.

SPU-kortin korvaamiselta tulevaisuudessa vältytään myös WIU-kortin käytöstä ja saadaan näin kustannussäästöjä. WIU-yksikö on alunperin suunniteltu käytettäväksi useissa erilaisissa anylysaattorreissa. Yksiköllä on useita toimintoja, joilla ei ole enää käyttöä ja sen valmistuskustannukset ovat korkeat käyttötarkoitukseen verrattuna. Toisaalta WIU-yksikön poistaminen aiheuttaa lisäkustannuksia, koska analysaattorin ulkopuolisten signaalien galvaaninen

erotus on hoidettava jollain muulla tavoin. Tätä varten on kuitenkin olemassa erilaisia kaupallisia ratkaisuja, jotka ovat halvempia kuin SPU-WIU yhdistelmä.

CSU-kortit on mahdollista korvata PC/104-analogia korteilla. PC/104-analogia kortit ovat huomattavasti edullisempia kuin CSU-kortit. Lisäksi saatavilla on paremmalla digitaali-analogiamuunnoksella varustettuja kortteja, kuin mihin vanhat CSU-kortit pystyvät. Korvaavien korttien käytöllä päästään parantamaan analysaattorin analogisesti välitettyjen tulosten tarkkuutta.

PC/104-pohjaisten I/O-korttien käyttö sisältää riskin, koska käyttöön valittavien korttien ohjaus ja käsittely joudutaan toteuttamaan uuden keskusyksikön ohjelmassa. Ohjelman toimintaa joudutaan muuttamaan, mikäli jotkut kortin ominaisuudet muuttuvat, tai se joudutaan korvaamaan toisella kortilla alkuperäisen yksikön valmistuksen loputtua. I/O-kortin liittimet ja kaapelit voidaan joutua myös uusimaan korttia vaihdettaessa. Vanhojen SPU- ja CSU-korttien etuna on niiden muutumattomuus. Niitä käyttämällä ohjelma ja liitännät voidaan pitää muuttumattomana.

Uuden keskusyksikön ohjelman jatkokehityksen yhdeksi tavoitteeksi asetettiin erottaa PC/104 I/O-kortit mahdollisimman hyvin varsinaisesta analysaattorin ohjelman rungosta. Jokaista käytettäväksi valittua korttia varten tulisi tehdä erillinen, mahdollisimman pieni kortin laiteohjain osuus. Tällöin laitteiston vaihtaminen ei aiheuttaisi kuin laiteohjaimen uudelleen kirjoittamisen tai päivittämisen.

Valittu laitteistoratkaisu mahdollistaa myös täysin uusien ominaisuuksien kehittämisen analysaattorille. PC-/104-väylään voidaan helposti lisätä näytönohjain, jolloin analysaattorille on mahdollista saada paikallisnäyttö. Paikallisnäyttöä voitaisiin käyttää tulosten näyttämisen lisäksi analysaattorin huoltotoimien ja kalibroinnin apuna. Uuteen keskusyksiköön on mahdollista

lisätä näppäimistö, jolloin analysaattorin paikallinen konfigurointi ja päivittäminen onnistuisivat helposti.

Keskusyksikön mahdollisia ohjelmistopäivityksiä varten tarjoutuu useita vaihtoehtoja. Ne voitaisiin hoitaa vaihtamalla käytössä oleva flash-muistikortti uuden ohjelmaversion sisältämään korttiin. PC/104-väylän pakkaan voitaisiin väliaikaisesti liittää PCMCIA-liitynnän omaava kortti, josta uusi ohjelma voitaisiin kopioida. Valmistajan tarjoamilla lisäohjelmistoilla voidaan sarjaväylää käyttää myös uuden ohjelmiston lataamiseen kortille.

7. OHJELMISTO

7.1 Toteutuksessa käytettyjä menetelmiä

Ohjelmiston määrittelyn ja suunnittelun apuna ei käytetty mitään erityisiä työkaluohjelmia vaan ne tehtiin käyttäen normaaleita tekstinkäsittely- ja piirustusohjelmia. Määrittely- ja suunnitteluvaiheessa ei käytetty täysin kirjaimellisesti mitään varsinaista menetelmää. SA/RT menetelmää käytettiin hyvin vapaasti omalla tavalla soveltaen. Tilakaaviot (STD) ja tietovuokaaviot (DFD) piirrettiin noudattamatta orjallisesti tiettyä kuvaustekniikkaa ja niistä kirjoitettua materiaalia.

Uuden keskusyksikön ohjelman toteutuskieleksi valittiin C-kieli. C-kieli oli tuttu entuudestaan työn toteuttajalle, ja sillä on nopeampi tehdä kehitystyötä kuin esimerkiksi assemblerilla. C-kielen kääntäjä tuottaa toiminnaltaan riittävän nopean ohjelman. Lisäksi sillä oli asembleria helpompi toteuttaa ohjelma, jota on jatkossakin mahdollisuus vaivattomasti jatkokehittää.

7.2 DOS käyttöjärjestelmä

Käyttöjärjestelmävalinnassa päädyttiin perinteiseen DOS-käyttöjärjestelmään, koska analysaattorille asetetut toimintavaatimukset eivät asettaneet käyttöjärjestelmälle mitään erityisiä vaatimuksia. DOS-käyttöjärjestelmää ja BIOSin palveluita käyttämällä ohjelmisto erotettiin laitteistosta niin, ettei uusien PC/104-CPU versioiden myöhemmälle käyttöönotolle tulisi ongelmia.

DOS-käyttöjärjestelmän valintaa puolsivat monet jo aikaisemmin mainitut syyt. Se on halpa, yksinkertainen ja tunnettu käyttöjärjestelmä. DOS-käyttöjärjestelmään oli saatavilla halpoja ja hyviä C-kehitysympäristöjä. DOS-käyttöjärjestelmän käytön laajuuden takia sillä katsottiin olevan jatkuvuutta,

joka koettiin hyvin tärkeäsi ominaisuudeksi valittaessa käyttöjärjestelmää uudelle keskusyksikölle.

Valitun käyttöjärjestelmän johdosta ohjelmiston kehitys oli helppo aloittaa ennenkuin sovitinkortti saatiin suunnitelluksi ja toteutetuksi. Ohjelmiston runko pystyttiin tekemään ja hyvin pitkälle moduulitestaamaan tavallisella toimistomikrolla. Järjestelmän sarjakommunikointi pystyttiin kehittämään ja testaamaan kytkemällä kaksi mikroa yhteen sarjalinjalla, ja ajamalla toisessa mikrossa analysaattorin kommunikointia hoitavaa ohjelmaosuutta.

7.3 Reaaliaikaisuus keskeytyksillä

Analysaattorin reaaliaikaisten toimintojen toteutuksessa käytettiin keskeytyspalveluita. Niiden avulla varmistettiin, että analysaattorin aikakriittiset toiminnot tuli suoritettua oikealla hetkellä ja riittävällä nopeudella. Sovitinkortin kautta uudelle keskusyksikölle tuli PROMIC-väylällä käytetyt keskeytykset, joista käyttöönotettiin vain PPU-kortin muodostamat kaksi keskeytyslinjaa. Näiden keskeytyslinjojen käsittelyn lisäksi analysaattorin sarjakommunikoinnin käsittelystä tuli yksi toteutettava keskeytyspalvelu.

PC-laitteissa laitteistokeskeytykset hoidetaan Intellin 8259 tai sen kanssa yhteensopivalla ohjelmoitavalla keskeytysohjainpiirillä (Programmable Interrupt Controller, PIC). Ulkoiset laitteet tekevät keskeytyspiirille keskeytyspyyntönsä oman keskeytyslinjansa kautta. Keskeytysohjain hoitaa keskeytysten priorisoinnin, niiden sallimisen ja keskeytyspyynnön välityksen prosessorille. Prosessorille välitetään keskeytyspyynnön identifikaatio, jonka perusteella prosessori laskee keskeytysvektorin sijainnin. Keskeytysvektorista löytyy varsinaisen keskeytysohjelman (“interrupt service routine”, ISR) alkuosoite. Prosessori suorittaa keskeytysohjelman, ja palaa tämän jälkeen jatkamaan varsinaista ohjelmaansa (/2/, /7/).

Laitteistokeskeytyksiä voidaan käyttää sovellusohjelmissa hyväksi kirjoittamalla oma keskeytysohjelma, joka korvaa varsinaisen keskeytysohjelman. Keskeytysohjelma voidaan kirjoittaa asemblerilla tai korkeamman tason kielellä. Keskeytysohjelma on periaatteessa aivan tavallinen ohjelmamoduuli. Keskeytysvektoriin sijoitetaan alkuperäisen keskeytysohjelman alkuosoitteen sijasta toteutetun keskeytyspalvelun osoite. Laitteistokeskeytyksen tullessa prosessori suorittaa alkuperäisen keskeytysohjelman sijasta toteutetun keskeytysohjelman.

7.3.1 PPU-kortin keskeytysohjelmat

Uuden keskusyksikön ohjelmaan toteutetuista kolmesta laitteistokeskeytyksestä kaksi tulee PPU-kortilta. Näiden lisäksi tulee myöhemmin tehdä kaksi uutta keskeytysohjelmaa, mikäli järjestelmään liitetään toinen detektori ja PPU-kortti.

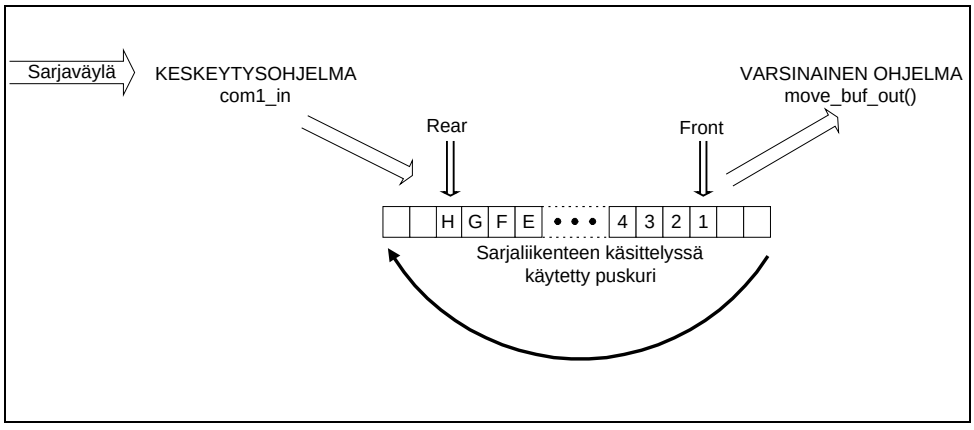
Analysaattorin todellinen mittausaika saadaan laskemalla PPU-yksikön keskeytyssignaaleita, kuten luvussa *5.3.3 Pulse Processing Unit* (PPU) kerrotaan. Tätä varten kirjoitettiin keskeytysohjelma, joka ko. keskeytyksen saapuessa lisäsi kesken olevaa mittausaikaa yhdellä, ja vertasi onko mittausaika jo täynnä. Mikäli mittausaika on tullut täyteen, lopetetaan mittaus ja välitetään varsinaiselle ohjelmalle tieto mittauksen valmistumisesta. Tämän jälkeen varsinainen ohjelma voi tehdä kaikki mittauksen jälkeiset toimenpiteensä.

PPU-kortin FIFOn tyhjennystarpeen keskeytyspyyntö toteutettiin omalla keskeytysohjelmallaan. Keskeytyspyynnön saapuessa keskeytysohjelma lukee PPU-kortilta 16 mikrokanava-arvoa ja lisää pulssit niiden arvojen mukaisille paikoille spektritaulukkoon. Lopuksi keskeytyspalvelu nollaa PPU-kortin keskeytyksen.

7.3.2 Sarjaliikenteen keskeytysohjelma

Sarjaliikenteen käsitteleminen suoraan ohjelmoimalla olisi monimutkaista. Tämän takia sen hoitamisen helpottamiseksi on oma piirinsä, jota kutsutaan UART-piiriksi (Universal asynchronous receiver transmitter). PC-laitteissa käytetään eri versioita Intellin 8250 UART piiristä. Ohjelmoinnin kannalta piiri muodostuu kymmenestä registeristä ja keskeytyslinjasta, joiden avulla sarjaliikenne voidaan hoitaa.

Toteutettu sarjaliikenteen keskeytysohjelma suorittaa saapuvan liikenteen käsittelyn lisäksi virheiden käsittelyn ja lähtevän datan registerin hallinnan. UART-piiri ilmoittaa sarjaväylästä saapuneesta merkistä keskeytyssignaalilla ja keskeytyksen identioivan registerin arvolla, jonka jälkeen keskeytysohjelma siirtää saapuneen merkin puskuriin. Puskuri on ohjelmassa oleva yksiulotteinen taulukko, josta normaali ohjelma lukee keskeytyspalvelun sinne kirjoittamia merkkejä. Taulukkoa täytetään ensimmäistä tyhjää paikaa ilmoittavan osoitimen kohdalta, ja tyhjennetään viimeisintä käsittelemätöntä merkkiä osoittavan osoittimen kohdalta. Taulukko on periaatteessa “ympyrä”, koska sen täyttäminen aloitetaan uudelleen alusta aina, kun saavutetaan sen loppupää.



Kuva 5. Saapuvan sarjaliikenteen käsittely

7.4 Käyttöliittymä

QuarCon analysaattorin käyttöliittymänä on symbolinen komentokieli ja kommunikointi analysaattoriin tapahtuu sarjaliikenteellä. Käyttöliittymä on periytynyt aikaisemmista analysaattoreista, jolloin analysaattoriin oli liitetty vain pääte parametrointia varten. Analysaattorin komennot muodostuvat yhdestä tai useammasta ASCII-merkistä. Analysaattori lähettää tuloksiaan ja virheilmoituksiaan saman sarjaväylän kautta.

Uuden keskusyksikön vaatimuksena oli, että se käyttää samaa komentokieltä ja samanmuotoisia viestejä. Yksi järjestelmään kehitettyistä valvomo-ohjelmistoista käytti vanhaa analysaattorin komentokieltä, joten yhteensopivuuden takaamiseksi muutoksia tähän ei ollut mahdollista tehdä.

Analysaattorin käyttöliittymää ja komentokieltä on tarkoituksena kehittää ja muuttaa tulevaisuudessa. Nykyisin käytetty komentokieli ei ole käyttäjäystävällinen eikä enää teknisesti järkevä. Sillä ei ole mahdollista käyttää tai parametroida analysaattoriin kaavailtuja uusia ominaisuuksia.

Uuden käyttöliittymän ja komentokielen kehittäminen eivät kuuluneet tähän työhön. Valittu toteutustapa mahdollistaa jatkossa käyttöliittymän vapaan kehityksen. Sarjaliikenneratkaisu voidaan toteuttaa helposti uudella tavalla, koska SPU-kortin sijasta käytettäväksi valittiin PC/104-kortin sarjaportti. Sarjaliikenteen käyttäminen ei kuitenkaan ole enää ainoa vaihtoehto. Eräs mahdollisuus olisi käyttää PC/104-verkkokorttia, ja kytkeä valvomo-ohjelmisto ja analysaattori toisiinsa verkoyhteydellä.

Tulevaisuudessa analysaattoriin on mahdollista lisätä myös paikallisnäyttö ja liityntä. Tällainen ratkaisu olisikin monessa tilanteessa hyödyllinen, koska järjestelmän tietokone ja sen sisältämä valvomo-ohjelmisto voivat sijaita kilometrien päässä analysaattorista. Paikallisnäyttö olisi mahdollista toteuttaa käyttäen PC/104-näytönohjainta ja siihen liitettyä näyttöä. Tällaistä ratkaisua

käytettiin testattaessa uuden keskusyksikön ohjelmaa, jolloin erilaisia testitulostuksia ja tilatietoja näytettiin uuteen keskusyksikköön kiinnitetystä näytöstä.

8. YHTEENVETO

Tämän työn tavoitteena oli löytää uusi korvaava ratkaisu hihna-analysaattorin vanhalle keskusyksikölle. Lisäksi tavoitteena oli toteuttaa ensimmäisen vaiheen järjestelmä ja sen ohjelmointi. Toteutettavan ratkaisun tuli mahdollistaa hihna-analysaattorin jatkokehitys, johon liittyviä kaikkia ominaisuuksia ja vaatimuksia ei ollut vielä tämän työn toteutusvaiheessa tiedossa.

Projekti myöhästyi alkuperäisestä aikataulustaan. Syynä tähän oli pääasiassa vanhan järjestelmän toiminnan selvittämisen vaikeudet. Vanhojen elektroniikkakorttien ja keskusyksikön yksityiskohtaisesta toiminnasta oli hyvin vaikea saada tietoa dokumentoinnin puutteiden ja sen vähyyden johdosta. Useita ominaisuuksia ja toimintoja jouduttiin selvittämään ja varmistamaan testaamalla laitteistoa käytännössä. Hyvin merkittävä osa tiedoista saatiin ainoastaan suullisena tietona järjestelmää tuntevilta henkilöitä.

Uuden järjestelmän ja keskusyksikön toteutus on jätetty protoasteelle, eikä sitä ole vielä otettu varsinaiseen käyttöön. Uuden keskusyksikön käyttöönotto siirtyi, koska vanhan keskusyksikön käytössä ilmenneitä ongelmia ratkaistiin siirtämällä analysaattorin toimintaa ja laskentaa siihen kytketylle valvomotietokoneelle. Tämä toteutettiin samalla kun, valvomo-ohjelmistoa jouduttiin uusimaan muiden pakottavien syiden takia. Toimintojen siirrolla valvomo-ohjelmistoon saavutettiin monia samoja etuja, joita oli tarkoitus saada käyttöön uudella keskusyksiköllä. Varsinaisen analysaattorin toimintojen yksinkertaistamisella ja toimintojen siirtämisellä normaalille tietokoneelle parannettiin koko järjestelmän hallittavuutta ja helpotettiin ohjelmistokehitystä.

Tälle työlle asetetut tavoitteet saavutettiin, vaikka uuden keskusyksikön käyttöönotto siirtyikin alkuperäisestä suunnitelmasta. Valittu ja toteutettu ratkaisu on onnistunut, koska tulokseksi saatiin avoin järjestelmä. Toteutettu

ratkaisu mahdollistaa analysaattorin jatkokehityksen ja tulevien tarpeiden täyttämisen.

Merkittävänä saavutuksena voidaan pitää myös koko analysaattorijärjestelmän ja sen yksiköiden yksityiskohtaisen toiminnan selvittämistä. Näiden tietojen perusteella järjestelmän normaali käyttö ja ongelmatilanteiden selvittäminen helpottuivat aikaisempaan tilanteeseen verrattuna. Analysaattori ei ole enää samanlainen mustalaatikko kuin aikaisemmin.

Kokonaisuutena analysaattorin kehityksen ensimmäistä vaihetta voidaan pitää onnistuneena. Analysaattorin jatkokehitys on jo alkanut. Uusia ominaisuuksia ja toimintoja on joiltakin osin tarkoitus toteuttaa alkuperäisestä suunnitelmasta poikkeavalla tavalla. Työssä valittu ratkaisu mahdollistaa myös näiden toteutuksen ja näinollen on osoittautunut onnistuneeksi.

LÄHTEET

1. Ampro 1994 Product Catalog. Ampro Computers, esite. 1994

2. Biggerstaff, Ted J. System Software Tools. New Jersey, Prentice-Hall, 1986. ISBN 0-13-881756-1.

3. Eskelinen, Juhani. 1992. Sulautetun järjestelmän automatisoitu testaus. Embedded and Real-Time Systems, Sulautetut ja reaaliaikaiset järjestelmät, Espoo, 6.11.1992. Helsinki, The Finnish Artificial Intelligence Society. s. 32-36. ISBN 951-96190-8-9.

4. Esser, Ropert. An object oriented Petri net approach to embedded system design. For the degree of doctor of technical sciences. Zürich, Eidgenössische Technische Hochschule (ETH),1997. ISBN 3-7281-2416-8.

5. Haikala, Ilkka & Märijärvi Jukka. Ohjelmistotuotanto. Suomen Atk-kustannus Oy. 5. Painos. Espoo, Suomen Atk-kustannus Oy, 1998. ISBN 951-762-666-5.

6. Ihme, Tuomas & Pesonen, Pekka. Oliokeskeiset menetelmät sulautettujen ohjelmistojen kehittämisessä. Teknologian kehittämiskeskus (TEKES). Helsinki, TEKES, 1992. TEKESin julkaisusarja 33/92. ISSN 0782-5420. ISBN 951-47-1925-5.

7. Jourdain, Robert. Programmer's Problem Solver. 2nd ed. New York, Brandy Publishing, 1992. ISBN 0-13-720194-X.

8. Keller, Marilyn & Shumate, Ken. Software Specification and Design, A Disciplined -Approach for Real-Time Systems. New York, John Wiley & Sons Inc, 1992. ISBN 0-471-53296-7.

9. Lawrence, Peter D. & Konrad Mauch. Real-Time Microcomputer System Design An introduction. McGraw-Hill Book Company, 1987. ISBN 0-07-036731-0.

10. Lehrbaum, Rick. Using PC/104 Modules In Industrial Embedded-PC Applications. CONTROL ENGINEERING, 1993, August. Vol. 40. ISSN 0010-8049.

11. Mayers, G.J. The Art of Software Testing. New York , John Wiley & Sons, 1979. ISBN 0-471-04328-1.

12. PC/104 Resource Guide. PC/104 Consortium, esite. 1995.

13. Peltoniemi, Jari. Sulautettua etsimässä. PROSESSORI, 1996. Vol. 12, No. 10, s. 39-42. ISSN 0357-4121.

14. Pressman, Roger S. Software Engineering A Practitioner's Approach. Second Edition. Singapore, McGraw-Hill Book Company, 1987. ISBN 0-07-100232-4

15. Ravn, Aders P. Design of Embedded Real-Time Computing Systems. For the degree of doctor of technical sciences. Lyngby, Technical University of Denmark, 1995. ISSN 0902-2821.

16. Seppänen Veikko & Kähkönen Anna-Maria & Oivo Markku & Perunka Harri & Isomursu Pekka & Pulli Petri. Strategic needs and future trends of embedded software. Teknologian kehittämiskeskus (TEKES). Sipoo,

Paino-Center Oy, 1996. Technology review 48/96. ISSN 0782-5420. ISBN 951-53-0744-9.

17. Shear David. Desktop DOS goes undercover to run embedded systems. EUROPEAN EDITION THE DESIGN MAGAZINE OF THE ELECTRONICS INDUSRY, 1994. Vol. 39, No. 16, s. 43-47. ISSN 0012-7515.

18. Shear David. Embedded system developers embrace the PC architecture. EUROPEAN EDITION THE DESIGN MAGAZINE OF THE ELECTRONICS INDUSRY, 1994. Vol. 39, No. 5, s. 39-44. ISSN 0012-7515.

19. Tervonen Ilkka, Validation of embedded software in three-level development process. Oulu, Oulun Yliopisto, Monistus ja kuvakeskus, 1988. Research papers Series A 11. ISBN 951-42-2683-6. ISSN 0784-638X.

20. Viskari, Juha. 1992. A Generic Software Engineering Environment for Supporting Software Development Process. Embedded and Real-Time Systems, Sulautetut ja reaaliaikaiset järjestelmät, Espoo, 6.11.1992. Helsinki, The Finnish Artificial Intelligence Society. s. 24-31. ISBN 951-96190-8-9.