Mikä on sulautettu järjestelmä?

- Sulautettu järjestelmä on prosessoripohjainen järjestelmä, joka on suunniteltu yhtä tai hyvin rajattua käyttötarkoitusta varten.
- Sulatetun järjestelmän erottaa yleiskäyttöisestä järjestelmästä se, että sulautetussa järjestelmässä ajetaan järjestelmän valmistajan määrittelemiä ohjelmia. Yleiskäyttöisessä järjestelmässä käyttäjä voi valita ohjelmistot ja niiden toimittajan.
- Sulautettujen järjestelmien ohjelmoinnissa korostuvat järjestelmän käyttötarkoitukseen liittyvät ominaisuudet
  - Sulautettu järjestelmä on (yleensä) tarkoitettu toimimaan itsenäisesti
    - Luotettavuus, vikasietoisuus ja vikatilanteista toipuminen
      - Järjestelmää ei voida aina käynnistää uudelleen milloin tahansa
      - Turvallisuus (esim. potilas, operaattori, huoltohenkilö, jne)
  - Järjestelmä on jatkuvassa yhteydessä ympäristöönsä
    - Anturit, toimilaitteet, tietoliikenneyhteydet, jne.
    - Voi olla osa suurempaa kokonaisuutta

Sulautetun järjestelmän ohjelmointi

- Pääosa sulautettujen järjestelmien ohjelmista kehitetään muussa kuin toimintaympäristössään (vrt. PC-ohjelmien kehitys)
  - Erillinen kehitysympäristö lisää testauksen haasteita
    - Kaikkia osajärjestelmiä ei välttämättä pysty simuloimaan, vaan testejä on ajettava myös kohdelaitteistolla
    - Testien toistettavuus ja testitapausten lisäämismahdollisuus huomioitava
- Sulautetun järjestelmän laitteiston suorituskyky mitoitetaan yleensä sovelluksen vaatimusten perusteella
  - Suorituskyky pitää huomioida koko kehitysprosessissa
  - Järjestelmän yksikkökustannukset pyritään minimoimaan → tehoa ei yleensä ole liikaa

Sulautettujen järjestelmien skaala on niin laaja, että on erittäin vaikea antaa yleispätevää kuvausta siitä millainen on sulautettu järjestelmä. On arvioitu, että maailmassa on tällä hetkellä enemmän sulautettuja tietokoneita kuin yleiskäyttöisiä. Sulautettujen tietokoneiden määrä kasvaa nopeammin kuin yleiskäyttöisten, joten on selvää että kaiken kattavaa määritelmää on vaikea esittää.

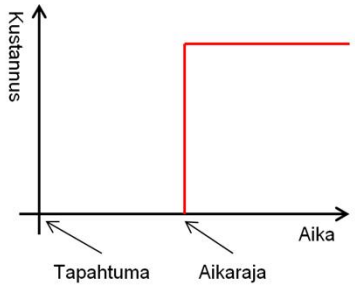
Sulautetun järjestelmän ohjelmointi

- Useimmilta sulatetuilta järjestelmiltä vaaditaan reaaliaikaisuutta
  - Aikarajoitteet voivat olla kovia (hard) tai pehmeitä (soft)
  - Aikarajoitteet koskevat järjestelmän vasteaikaa
    - Esimerkiksi reagointi tapahtumaan tai sanoman käsittelyaika
- Kova aikarajoitus (hard realtime)
  - Aikarajan ylitys on aina virheellinen toiminta
  - Aikarajaa ei saa ylittää missään tapauksessa
- Pehmeä aikarajoitus (soft realtime)
  - Aikarajan ylitys tarkoittaa puutteellista toimintaa (mutta ei välttämättä virhetilanne)
  - Aikarajavaatimukset ovat usein tilastollisia
    - Esimerkiksi keskimääräinen suoritusaika

Reaaliaikaisuus

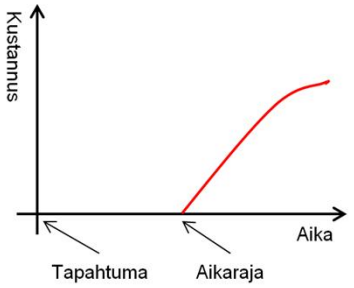
Kova reaaliaikaisuus

- Aikarajan ylitys aiheuttaa vaaraa tai mittavia taloudellisia vahinkoja



Pehmeä reaaliaikaisuus

- Aikarajan ylitys aiheuttaa lisäkustannuksia, mutta on hyväksyttävissä
- Liittyy yleensä palvelun laatuun (Quality of Service, QoS)



Ajoituksesta

- Sulautetut järjestelmät reagoivat useimmiten herätteisiin (events)
  - Viestit
  - Signaalit (ulkoiset)
  - Ajastimet
  - Integroidut oheislaitteet
    - Liittyvät usein viestien käsittelyyn (esim. UART, verkkokortti, jne)
- Herätteillä oma hierarkia
  - Protokollapinon eri kerrokset
- Saapumismalleja
  - Jaksollinen
    - Jakson pituudessa voi olla vaihtelua
  - Jaksoton
    - Esim. purskeinen

Herätteisiin reagointi voidaan toteuttaa monella eri tavalla. Yksinkertaisin tapa on kiertokysely eli pollaus, jossa käydään esim. jokainen oheislaite läpi vuorollaan ja tarvittaessa suoritetaan laitteen vaatimat toimenpiteet. Kiertokysely toimii kohtuullisesti kevyesti kuormitetussa järjestelmässä, mutta raskaalla kuormituksella viive kahden pollauksen välillä venyy helposti liian pitkäksi.

Tyypillisesti herätteet priorisoidaan jollain tavalla. Yksinkertaisimmillaan priorisointi voidaan hoitaa siirtämällä aikakriittinen käsittely keskeytysrutiineihin ja tekemällä vähemmän kriittiset asiat pääohjelman puolella.

Tapahtumien priorisointi ja ajoituksen varmistaminen on yksi sulautettujen järjestelmien hankalimmista asioista. Pelkästään huolellisella ohjelman kirjoittamisella ei voi taata oikeaa ajoitusta, vaan yleensä tueksi tarvitaan formaaleja menetelmiä tai mallinnusta. Hyvä ohjelmistoarkkitehtuuri ja huolellinen koodaus toki tekee elämän helpommaksi siinä vaiheessa kun järjestelmään (väistämättä) tulee muutoksia.

Ajoituksesta

- Toimintojen sisäiset suoritusajat ovat keskeisiä järjestelmän ajoitettavuuden kannalta
  - Kovia reaaliaikajärjestelmiä mallinnetaan usein pisimpien mahdollisten suoritusaikojen avulla (worst-case execution time)
  - Pehmeissä voi käyttää keskimääräisiä suoritusaikoja (average execution time)
    - Hajonta vaikuttaa myös palvelun laatuun
- Toiminnot voivat olla peräkkäisiä tai samanaikaisia
  - Peräkkäiset ja toisistaan riippumattomat toiminnot ovat ”helppoja”
  - Toisistaan riippuvat samanaikaiset toiminnot tarvitsevat keskinäistä synkronointia

Erityispiirteitä

- Osoittimet oheislaitteiden ja prosessorin ohjausrekistereihin
  - Prosessorin ja (integroitujen) oheislaitteiden ohjausrekistereitä lukemalla ja/tai kirjoittamalla saadaan aikaan halutut toiminnot
    - Luvun ja kirjoituksen järjestys ja oikea-aikaisuus on tärkeää
  - Ohjausrekisterit on yleensä sijoitettu muistiavaruuteen, joten niitä käsitellään ohjelmoinnissa samalla tavalla kuin tavallisia muistipaikkoja
  - Tarvitaan tietoa laitteiston toiminnasta
    - Ohjausrekisterien sijanti selviää muistikartasta, joka löytyy prosessorivalmistajan dokumentaatiosta
- Volatile lisämääre
  - Kertoo kääntäjälle, että muuttujan arvo voi muuttua milloin tahansa – myös silloin kun ohjelma ei käytä muuttujaa
  - Oheislaitteiden rekisterit määritellään volatile-tyyppisiksi tai käytetään accessor-funktioita

Volatile-lisämääre estää kääntäjää siirtämästä muuttujaa rekisteriin, jolloin jokainen viittaus muuttujaan menee suoraan muistiin. Kääntäjä ei myöskään optimoi viittausten lukumäärää esimerkiksi silmukan tapauksessa vaan jokainen viittaus myös toteutetaan. Muistiavauruuteen sijoitettujen IO-laitteiden rekisterit voivat vaihtaa tilaansa milloin tahansa kahden lukukerran välissä, joten ”turhien” lukemisten pois optimointi johtaa todennäköisesti virheelliseen toimintaan.

Nykyprosessoreissa käskyt suoritetaan liukuhihnalla ja prosessori saattaa jopa vaihtaa käskyjen suoritusjärjestystä, joten ohjausrekisterien käsittelyjärjestyksen takaaminen saattaa vaatia lisäkäskyjen tai vihjeiden antamista prosessorille. Vihjeet ovat käskyjä, jotka eivät varsinaisesti tee mitään, mutta niiden avulla prosessorille annetaan ohjeita käskyjen suorituksesta. Memory barrier-käsky kertoo prosessorille, että kaikki ennen sitä annetut muistiviittaukset on käsiteltävä loppuun ennen kuin sen jälkeen tulevia muistiviittauksia voidaan käsitellä.

Erityispiirteitä

- Bittioperaatiot
  - Loogisia operaatiota ja siirtoja tarvitaan laiteläheisessä ohjelmoinnissa, koska ohjausrekistereistä pitää pystyä lukemaan tai muokkaamaan yksittäisiä bittejä tai bittiryhmiä
- Laitteiston abstraktiotaso on matalampi
  - Ohjelmoija tarvitsee tietoa laitteiston toiminnasta
  - Käyttöjärjestelmät ovat yleistyneet varsinkin isommissa järjestelmissä, jolloin osa laitteistosta voidaan piilottaa rajapintojen taaksi
    - Laiteohjaimen kirjoittajan on kuitenkin ymmärrettävä laitteiston toimintaa
    - Sulautetut/reaaliaikakäyttöjärjestelmät ovat yleensä ”kevyempiä” kuin yleiskäyttöisten koneiden käyttöjärjestelmät
- Keskeytykset

Yksittäisten bittien tai bittiryhmien muokkaukseen käytettävät operaatiot tunnetaan myös nimillä ”shift and mask”. Operaatiossa bittiryhmän arvoa ajatellaan pieneksi kokonaisluvuksi, joka siirto-operaation avulla siirretään oikeaan kohtaan esim. 32-bittisessä luvussa. Maskien avulla muutokset voidaan kohdistaa koskemaan ainoastaan haluttuja bittejä.

Esimerkiksi muutetaan 32-bittisen luvun bittejä 13-15, siten että muut bitit eivät muutu. Muutettava luku on *SFR* ja arvo joka asetetaan on *val*.

```
SFR = (SFR & 0xFFFF1FFF) | (val << 13);
```

Tai

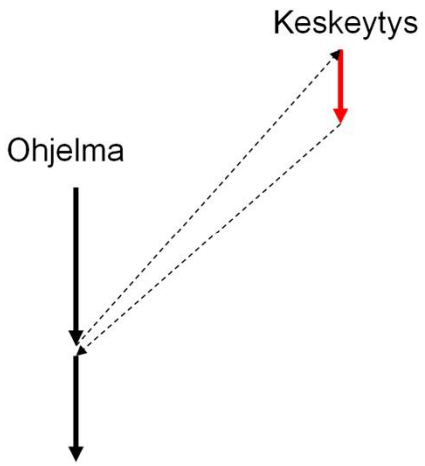
```
SFR = (SFR & ~0xE000) | (val << 13);
```

Tai

```
SFR &= ~0xE000;
SFR |= val << 13;
```

Keskeytykset

- Keskeytykset ovat prosessorin ominaisuus, joka mahdollistaa nopean reagoinnin ulkoisiin (tai sisäisiin) tapahtumiin
- Keskeytyksessä ajettavan ohjelman suoritus keskeytetään ja prosessori siirtyy suorittamaan ohjelmaa toisesta (ennaltamäärätystä) osoitteesta
- Keskeytys on ajettavan ohjelman kannalta ”läpinäkyvä” eli suoritettava ohjelma ei ajallista keskeytystä lukuunottamatta häirinny
- Prosessorissa on yleensä useita keskeytyslähteitä



Vaikka abstraktiotaso on matala, niin silti voidaan käyttää ”rajapintoja”, jotka helpottavat ohjelmien tai moduulien siirtämistä ympäristöstä toiseen. ARM on määritellyt Cortex Microcontroller Software Interface Standardin (CMSIS), joka on valmistajariippumaton laitteistorajapinta, jota käyttämällä ohjelmistojen siirtäminen eri valmistajien ARM-prosessoreiden välillä on helpompaa.

CMSIS on laaja määrittely, joka pitää sisällään myös sovellustason rajapintoja. Standardin ydinosiin kuuluu rajapinnat prosessorin ohjausrekistereiden ja keskeytysten hallintaan.

Keskeytykset

- Keskeytyksessä prosessori keskeyttää normaalin ohjelman suorituksen ja siirtyy suorittamaan keskeytyspalveluohjelmaa (Interrupt Service Routine, ISR) (keskeytyskäsittelijä, interrupt handler)
- Keskeytys on laitteiston pakottama "aliohjelmakutsu"
- Tavallisen aliohjelman kutsumiseen ohjelmoija voi varautua, mutta keskeytyksen ajankohtaa ei voi ennustaa, joten keskeytyspalveluohjelmalla on tiukemmat tallennusvaatimukset kuin aliohjelmalla
  - Kielestä riipumatta alimmalla tasolla kaikki ohjelmat ovat konekielisiä (välissä on yleensä kääntäjä tai tulkki)
    - Suoritettava ohjelma voi keskeytyä kesken lauseen suorituksen, koska lauseesta yleensä muodostuu useita konekielisiä käskyjä
  - Keskeytyspalveluohjelman suoritus ei saa sotkea normaalisti ajettavan ohjelman toimintaa
  - C-kielellä ohjelmoitaessa keskeytyskäsittelijän toteutuksessa on kääntäjä- ja prosessorikohtaisia eroja
    - Erot liittyvät tallennusvaatimusten toteuttamiseen

Ohjelmien ja funktioiden yhteensopivuuden varmistamiseksi ohjelmointikielten kääntäjät kääntävät kaikki funktiot kutsusopimuksen (procedure/function call standard) mukaisesti. Kutsusopimus pitää sisällään periaatteet prosessorin rekisterien yhteiskäytöstä, parametrien välityksestä ja tulosten palauttamisesta. Kutsusopimuksen avulla kääntäjä tietää mm. mitkä rekisterit voivat muuttua aliohjelmakutsun (=funktio) aikana ja osaa sen perusteella tallettaa esim. laskennan välitulokset funktiokutsujen välillä.

Keskeytyksen tapahtumahetkeä ei voi ennustaa, joten keskeytysrutiinin on käytännössä talletettava kaikki rekisterit, joita keskeytysrutiinissa käytetään, jotta se ei sotke keskeytetyn ohjelman toimintaa. Yleensä kääntäjälle voidaan lisämääreellä kertoa, että funktio on keskeytysrutiini, joilloin kääntäjä lisää rutiinin alkuun prosessorin tilan tallennuksen ja loppuun tilan palautuksen.

ARM Cortex M3:ssa keskeytysrutiinit voi kirjoittaa suoraan C:llä, mikäli kääntäjä noudattaa AAPCS:ää (ARM Architecture Procedure Call Standard), sillä prosessori tallettaa automaattisesti AAPCS:n mukaiset rekisterit. Jos käytössä on skeduleri/käyttöjärjestelmä, joka mahdollistaa kontekstin vaihdon, niin silloin kontekstin vaihdossa on pakko käyttää jonkun verran assemblyä.

Keskeytykset

- Keskeytyksessä prosessori keskeyttää normaalin ohjelman suorituksen ja siirtyy suorittamaan keskeytyspalveluohjelmaa (Interrupt Service Routine, ISR) (keskeytyskäsittelijä, interrupt handler)
  - Keskeytyskäsittelijän pitää selvittää mikä aiheutti keskeytyksen ja reagoida tapahtumaan
    - Keskeytyksen aiheuttanut tilanne pitää usein kuitata jollain tavalla
      - Erillisellä signaalilla
      - Väylällä
      - Joissakin tilanteissa ei tarvita kuittausta ollenkaan
  - Keskeytyskäsittelijästä pitää pystyä palaamaan takaisin jatkamaan normaalia ohjelman suoritusta samasta kohdasta missä suoritus keskeytettiin
    - Käytetään joko kääntäjän apua tai
    - Kirjoitetaan assemblyllä “wrapper”, joka hoitaa prosessorin tilan tallennuksesta ne osat, joita ei pysty C:llä tekemään

Keskeytyksen tapahtuessa prosessorin tila tallennetaan pinomuistiin, koska pinoon voidaan viitata tietämättä sen absoluuttista osoitetta, mikä mahdollistaa joustavan toiminnan, koska tallennuspaikkaa ei tarvitse tietää ennalta.

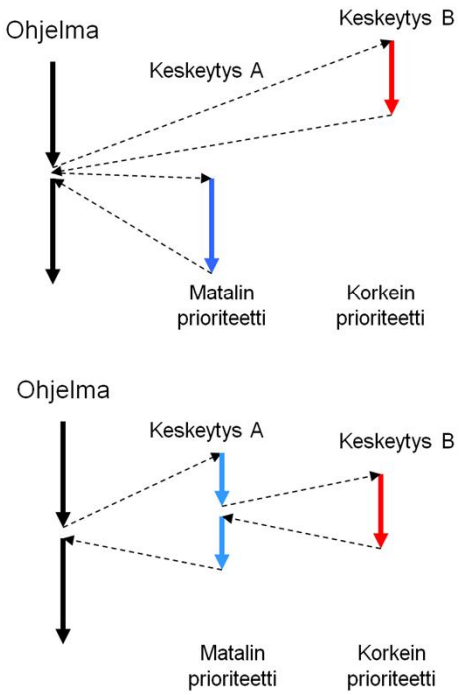
C:ssä paikalliset muuttujat varataan pinosta, jolloin ylimääräistä pinomuistia tarvitaan prosessorin tilan tallennuksen vaatima määrä plus paikallisten muuttujien käyttämä tila plus mahdollisesti muiden kutsuttujen funktioiden käyttämä tila. Periaatteessa keskeytysrutiinista voi kutsua mitä tahansa funktiota, mutta yleensä esim. C:n standardikirjastofunktioita ei ole suunniteltu kutsuttavaksi keskytysrutiinista. Tyypillisesti keskeytysrutiini pyritään pitämään mahdollisimman lyhyenä ja helposti analysoitavana.

Keskeytykset

- Prosessorin tila talletetaan yleensä pinomuistiin
  - Pitää pystyä ennakoimaan kuinka paljon pinomuistia keskeytykset pahimmassa tapauksessa vaativat
- Keskeytyskäsittelijän pitää selvittää mikä aiheutti keskeytyksen
  - Vektorointi = keskeytyksen tyyppi määrittää suoritettavan keskeytyspalveluohjelman. Jokaista erityyppistä keskeytystä varten on keskeytysvektori, johon voidaan tallettaa keskeytyspalveluohjelman osoite. Tällöin keskeytyskäsittelijällä on valmiiksi tieto keskeytyksen aiheuttajasta
    - Esim. Intel PC-prosessorit, AVR, ARM Cortex-sarja
  - Vakiokeskeytys = kaikilla keskeytyksillä on yksi yhteinen keskeytyspalveluohjelman osoite. Keskeytyspalveluohjelman pitää ensimmäiseksi selvittää minkä tyyppinen keskeytys on kyseessä. Tieto usein esim. erikoisrekisterissä
    - Esim. MIPS, ARM7

KESKEYTYKSET

- Keskeytyksillä on yleensä prioriteetit, joka määrää missä järjestyksessä keskeytykset suoritetaan
  - Korkeamman prioriteetin keskeytys suoritetaan ennen matalan prioriteetin keskeytystä
  - Korkeamman prioriteetin keskeytys voi keskeyttää alemman prioriteetin keskeytyksen
    - Jos järjestelmä sallii keskeytysrutiinin keskeyttämisen, niin suunnittelussa vaaditaan erityistä huolellisuutta, että pino pysyy sille varatulla alueella!
- Prioriteetit valitaan kiireellisyysjärjestyksen perusteella



Jos järjestelmä sallii sisäkkäiset keskeytykset, niin keskeytyksille pitää varata pinosta tilaa niin paljon että kaikkien mahdollisesti sisäkkäin olevat keskeytysten käyttämä tila mahtuu pinoon yhtä aikaa.

Pinon kulutusta voidaan helposti seurata esim. alustamalla pinolle varattu muistialue tiettyyn arvoon ja tutkimalla miten suuri osa arvoista on muuttunut pinon käytön seurauksena. Pinon kulutuksen jatkuva seuranta ohjelmallisesti kuluttaa resursseja. Osa kääntäjistä tukee pinon kulutuksen ohjelmallista seurantaa. Seurannan ollessa käytössä kääntäjä lisää funktiokutsujen yhteyteen pinon ylivuototarkistukset. Jos kääntäjän tukea ei ole se voidaan toteuttaa käyttöjärjestelmän yhteyteen tai yksikertaisesti lisätä tarkistuskutsut sopiviin paikkoihin.

Joissakin prosessoreissa on mahdollista käyttää erillistä keskeytyspinoa, jolloin pinon käytön analysointi on helpompaa.

Prosessorissa, jossa on muistinhallintayksikkö (MMU tai MPU) voidaan pinon käyttöä ja ylivuotoja valvoa laitteistotasolla.

KESKEYTYSVASTE

- Keskeytysvaste on aika, joka kuluu keskeytyssignaalin asettumisesta keskeytyspalveluohjelman käynnistymiseen
  - Prosessorin oman keskeytysvasteen lisäksi muiden keskeytysten prioriteetit ja keskeytyskäsittelijöiden pituus vaikuttaa keskeytysvasteeseen
  - Keskeytyskäsittelijän pitää olla mahdollisimman nopea, jotta ehditään reagoida ajoissa myös odottaviin keskeytyksiin
- Keskeytykset voidaan tarvittaessa estää, jolloin keskeytys jää odottamaan siihen asti että keskeytykset taas sallitaan
  - Käytetään takaamaan kriittisen sektion (critical section) suoritus keskeytyksettä
  - Keskeytyksiä estettäessä pitää miettiä kuinka pitkäksi ajaksi keskeytykset estetään, koska estäminen pidentää vasteaikaa
  - Esimerkiksi jokaisen AD-muunnoksen jälkeen tulee keskeytys ja se on estetty niin pitkäksi ajaksi, että seuraava muunnos on tehty, niin järjestelmä ei enää ole oikea-aikainen

Keskeytysvaste

- Yleensä estetään saman (ja alemman) prioriteettitason keskeytykset keskeytyskäsittelijän suorituksen ajaksi
  - Pahimman mahdollisen tapauksen vaste
- Kaikkia keskeytyksiä ei voi estää
  - NMI (Non Maskable Interrupt) on keskeytys, jota ei voida estää
  - NMI:tä käytetään vain poikkeustapauksissa (normaali laiteohjain ei tarvitse NMI:tä)

Keskeytysvaste muodostuu prosessorin keskeytysvasteesta ja sekä ajasta, jonka keskeytykset ovat estettynä. Käyttöjärjestelmäkutsuissa on usein kriittisiä sektioita, joiden yhteydessä keskeytykset on estetty.

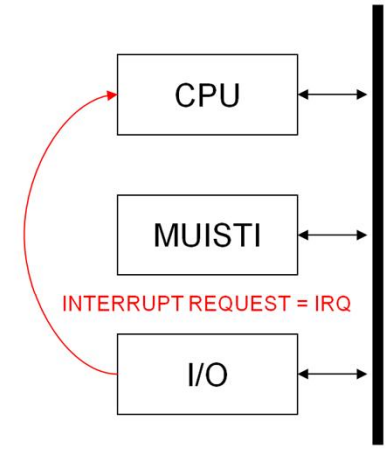
Vaikka järjestelmä itsessään olisi pehmeä reaaliaikajärjestelmä, niin keskeytystasolla toiminta pyritään toteuttamaan kovan vaatimustason mukaan ja jos järjestelmän suorituskyky ei riitä, niin tietojen ”hukkaaminen” on yleensä turvallisempaa tehdä ylemmillä protokollatasoilla.

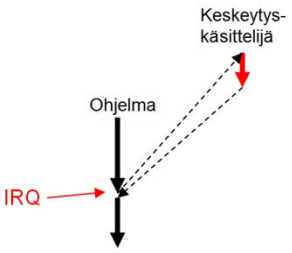
Keskeytykset ja suojaustilat

- Jos prosessorissa on vähintään kaksi tilaa (käyttäjän tila ja etuoikeutettu tila), niin keskeytykset suoritetaan etuoikeutetussa tilassa
  - Käyttäjätilassa ohjelmalla on rajoitetut oikeudet esim. prosessori voi estää IO-laitteiden käsittelyn, muistinhallinnan tai prosessorin tilan vaihtamisen jne.
  - Etuoikeutetussa tilassa kaikki prosessorin resurssit ovat käytettävissä
  - Käyttöjärjestelmät hyödyntävät jakoa etuoikeutettuun ja käyttäjätilaan
    - Käyttöjärjestelmän funktioiden kutsuminen voidaan toteuttaa ohjelmakeskeytyksillä (software interrupt), jolloin käyttöjärjestelmän funktion kutsuminen siirtää prosessorin etuoikeutettuun tilaan

Keskeytysohjattu IO

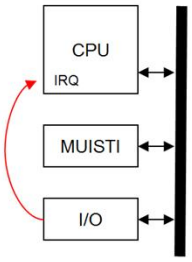
- IO-laite on kytketty prosessorin keskeytyslinjaan
- Kun laite on valmis tiedonsiirtoon se asettaa keskeytyssignaalin
- Prosessori keskeyttää ohjelman suorituksen ja ajaa keskeytyspalveluohjelman
- Keskeytyspalveluohjelma suorittaa IO-tapahtumaan liittyvät toimenpiteet, esim. kopioi vastaanotetun datan IO-laitteen puskurista ja kuittaa keskeytyksen käsitellyksi
  - Jos data vaatii laajempaa käsittelyä se tehdään pääohjelman puolella

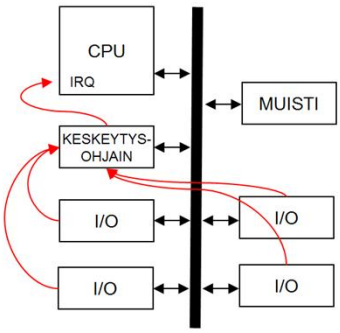




Keskeytysohjattu IO

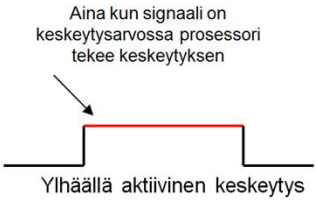
- Keskeytysohjain vastaanottaa IO-laitteiden keskeytykset ja välittää ne edelleen prosessorille
  - Voi muuntaa keskeytyssignaaleja toisen tyyppisiksi
  - Priorisoi keskeytykset
  - Pitää kirjaa aktiivisista keskeytyksistä
  - Joissakin ohjaimissa mahdollisuus tallettaa keskeytysvektori ohjaimeen
  - Mikrokontrollereissa keskeytysohjain yleensä integroitu prosessoriin

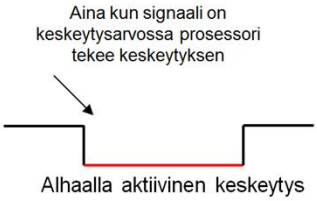




Keskeytyssignaalityypit

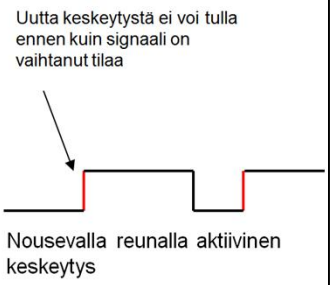
- Tasoherkkä
  - Keskeytyssignaalin arvo indikoi keskeytyksen
  - Keskeytys on aktiivinen niin kauan kuin signaalilla on ko. arvo (ylhäällä tai alhaalla aktiivinen)
  - Keskeytysrutiinin pitää aikaansaada keskeytyssignaalin poistuminen, jotta samaa keskeytystä ei tulkita useaksi peräkkäiseksi keskeytykseksi





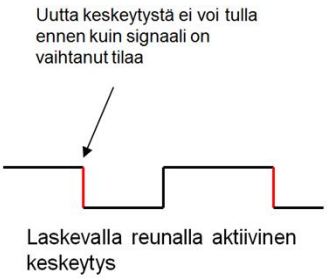
Keskeytyssignaalityypit

- Reunaherkkä
  - Keskeytyssignaalin muutos indikoi keskeytyksen (nouseva tai laskeva reuna)
  - Keskeytyspyyntö liittyy muutokseen: uutta ei voi tulla ennen kuin signaali on vaihtanut tilaa
  - Ei riskiä saman keskeytyksen tulkitsemiseksi useaksi keskeytykseksi



Nousevalla reunalla aktiivinen keskeytys

- Ulkoisilla laitteilla keskeytyssignaalin tyyppi pitää kertoa keskeytysohjaimelle
- Integroiduilla laitteilla riittää keskeytysten salliminen



Laskevalla reunalla aktiivinen keskeytys

Tiedonvälitys keskeytysrutiinista

- Keskeytysrutiinien ja muun ohjelman välillä kommunikointi tapahtuu käyttämällä yhteisiä muistialueita
  - Yksinkertaisimmillaan keskeytysrutiini asettaa lipun (globaalin muuttujan) ja pääohjelma tekee oman osuutensa havaittuaan lipun asettuneen
    - Järkevää yleensä vain hyvin pienissä järjestelmissä
  - Tyypillisesti käytetään jonkinlaista jonotusta
    - Esim. vastaanotetut merkit menevät puskuriin, josta ne voidaan lukea myöhemmin
  - Käyttöjärjestelmän tapauksessa muistialueet/tietorakenteet on piilotettu käyttäjän näkyvistä ja kommunikointi tapahtuu käyttöjärjestelmän kutsujen avulla
- Yhteisen muistialueen käsittelyn pitää olla atominen operaatio
  - Pääohjelman käsitellessä yhteistä muistialuetta keskeytysrutiini ei saa muuttaa sitä samaan aikaan → kriittinen sektio

Jos ohjelma kirjoitetaan suoraan prosessorille, niin keskeystysvasteen kannalta järkevintä on estää ainoastaan ne keskeytyket, jotka käsittelevät kyseistä muistialuetta. Eri liitäntälaitteilla on yleensä oman jononsa, joten yhden laitteen jonon käsittelyn ajaksi ei tarvitse estää kuin kyseisen liitäntälaitteen keskeytykset.

Käyttöjärjestelmään käytettäessä kommunikointiin käytetään käyttöjärjestelmän palveluja esimerkiksi semaforeja tai jonoja, jolloin tarve keskeytysten estämiseen vähenee huomattavasti.

Keskeytykset

- Perusperiaate
- Kääntäjätuki
- Ohjelman ja keskeytysrutiinin kommunikointi
- Kriittinen sektio

Muistinhallinta

- Monissa sulautetuissa järjestelmissä muistin määrä on pieni verrattuna yleiskäyttöisiin laitteisiin, joka tietenkin on huomioitava ohjelmissa
- Dynaaminen muistinvaraus (malloc) ei välttämättä ole käytössä
  - Pirstoutumisen hallinta on tärkeää, jos malloc on käytössä
  - Jos ohjelmaa ei voida sulkea välillä pirstoutumisongelma korostuu
  - Dynaaminen muistinvaraus voi johtaa ennalta-arvaamattomiin viiveisiin tai virhetilanteisiin
    - Kaikki muisti on varattu
      - Joko odotetaan tai yritetään toipua virheestä (esim. Hylätään vastaanotettu data)
  - Kompaktointi voi aiheuttaa viiveitä

Standardikirjastot

- C:n standardikirjastot ovat yleensä melkoisia muistisyöppöjä ja niissä on riippuvuuksia toimintoihin, joita sulautetussa järjestelmässä ei välttämättä ole toteutettu
  - Tiedostot (myös stdio)
  - Malloc
- Muistirajoitteisissa järjestelmissä standardikirjastojen käyttö voi kasvattaa muistivaatimuksia huomattavasti
  - Erilliset sulautettuihin järjestelmiin suunnitellut kirjastot (esim. Redlib)