Bilag 1 – teknisk gennemgang af GOPA

Indholdsfortegnelse

Til brugeren af de færdige programmer

Her er en beskrivelse, der fokuserer på fordelene ved de færdige programmer.
Det, der skiller sig mest ud, er full-motion grafikken. Betegnelsen fullmotion bruges ofte om høje opdateringsfrekvenser f.eks. 30 billeder (hz) i sekundet. GOPA bruger enten 20 eller 30 hz, men af årsager, der ikke er fundet nogen forklaring på, får skærmsynkroniseringen objekter i bevægelse til at fremstå skarpere end f.eks. rigtige tv-billeder. En ny TV standard med tilsvarende egenskaber ville formentlig komme til at hedde noget i retning af "crystal clear motion".

Pilothjerner kan udnytte pilot-interface

Procesoperatører skiller sig noget ud fra gennemsnitsbefolkningen. De er i høj grad i stand til at modstå stresspåvirkninger uden at tabe overblikket. Præcis de samme egenskaber man bliver testet for, når man søger ind som jagerpilot. En almindelig passager, der kommer for skade at kigge ud i cockpittet, bliver forskrækket over de mange instrumenter i hurtig opdatering: Hvordan kan piloten dog overskue alt dette?
Det er dels en træningssag, dels har piloten allerede ved optagelsesprøven vist, at vedkommende har pilothjerne. Den massive strøm af information virker ikke stressende - underbevidstheden holder øje med, hvordan alle mulige målepunkter udvikler sig, selvom de er langt fra alarmgrænserne - og det virker beroligende på piloten, så længe alt ser ud som det plejer.

Det er sandsynligt, at operatører der fungerer godt i en højtydende procesindustris ofte kaotiske virkelighed, har pilothjerner i samme omfang som rigtige piloter.
Giv dem et pilot-interface - de kan nemt overskue det, og bliver mere pro-aktive - bedre til at tage problemerne i opløbet, og får mere fra hånden, end hvis de havde haft et traditionelt Windows-interface med 2-3 billeder i sekundet.

Om fremstilling af færdige programmer

Dette afsnit omhandler selve produktionen af færdige programmer ved hjælp af GOPA-systemets værktøjsfaciliteter. Altså hvad applikations-programmøren kommer ud for.

Der er vel tale om en variant inden for den objektorienterede storfamilie, hvor det er tilstræbt at begrænse den tid, nybegynderen skal bruge på at lære systemet at kende.

Her er den kendte 80/20 regel for megen menneskelig aktivitet udnyttet: At det er de første 20% af indsatsen, der giver de 80% af nytteværdien. Det er kun specialisten, der gider tage de sidste 80% af indsatsen med, og derved indkassere de sidste 20% af nytteværdien.

I forhold til hvad der findes inden for objektorienteret programmering, er en række af de mere avancerede ting valgt fra, mens der til gengæld er gjort ekstra meget ud af, at forenkle de mekanismer, der så er beholdt.

En af de vigtigste er multi-programmeringen. At en og samme kode kan eksekvere i multiple versioner, hvor hver version har egne data, egne tråde osv. Hver version vil i overbliksbilledet fremstå som en komplet mini-PLC – dog med indikation af, at den deler kode med andre mini-PLC'er.

En "grafisk linker" giver så forskellige versioner af alle slags PLC-er – multi såvel som single - adgang til at snakke sammen. Mini-PLC'erne fremstår som små kasser, der kan forbindes med streger, der viser, hvem der kan snakke med hvem. Det andet vigtige objekt-tema der er beholdt, er "Interfacet" mellem objekter, der også er lagt stor vægt på at gøre letforståeligt.

Der findes i dokumentationen et eksempel – kaffeautomaten – der er egnet til at vise, hvor produktivitetsfremmende denne enkle form for multiprogrammering kan være.

Den typiske applikations-programmør behøver ikke have nogen edb-relateret uddannelse, og ikke have den fjerneste anelse om hvad "objektorienteret" betyder. Det koncept brugeren skal forholde sig til, er tæt på at findes i den fysiske virkelighed. Der findes nemlig mange fabrikker, hvor et moderat antal halvstore fysiske PLC'ere snakker sammen via kabler. Det man simulerer med GOPA er sammenlignet med dette, blot flere og mindre PLC'ere. - plus en 100% sikker kommunikation (ingen fysiske kabler) og ingen af de, fra datakommunikation, kendte problematikker som "handshake", og hvad der f.eks. sker, hvis et par af PLC'erne bliver ramt af strømafbrydelse, mens de øvrige arbejder videre.

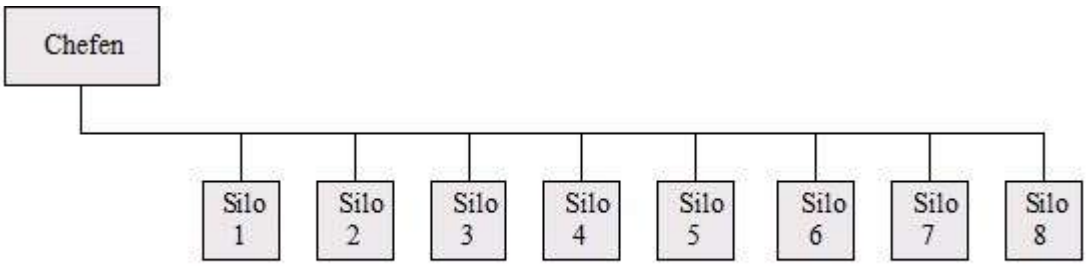
I en GOPA-applikation tilstræbes en udveksling af letforståelige meddelelser mellem de små PLC'ere, så flest mulige af de involverede interessenter (fra bygherre til operatører) får en fuld forståelse af programmets virkemåde.
Man kan forestille sig de enkelte PLC'ere som personer, der har taget opstilling ud for de maskindele, de hver især har ansvaret for. Man opfinder nu nogle "spilleregler", herunder mulige meddelelser, disse personer kan sende til hinanden. Når det lykkes allerbedst, har de involverede interessenter en oplevelse af i fællesskab at have opfundet en ny selskabsleg med spilleregler og mulige "meddelelser".

Eksempler på selskabslege

Logikken i enhver styring kan udvikles og verificeres som en slags selskabsleg. Et antal personer, der hver især spiller rollen som mini-PLC'er, bliver enige om hvem der skal kunne sende hvilke beskeder til hvem, og hvilke "tilstande" de enkelte mini-PLC'ere (aktører) kan befinde sig i. Det er ikke uden grund, at det første multi-trådede objektsprog kom til at hedde "Actor" - altså skuespiller.

Meget simpel selskabsleg

Det mindste eksempel fra det virkelige liv er, når en enkelt ud af en gruppe ensartede maskindele skal startes på grundlag af en talværdi – f.eks. et silonummer.
Her lægger chefen silonummeret ud i et register, som alle siloerne lytter til. Den silo, der genkender nummeret som sit eget, vil holde sin relæudgang tændt.
Hver silo har sin egen mini-PLC, som virkelig ikke har anden funktion end at holde øje med om et silonummer i et interface-register matcher dens eget:



Stregen, der forbinder PLC-erne i oversigtsbilledet, får dem til at deles om et fælles interface-register.

Princippet er meget generelt – ansvaret delegeres nedefter. Chefen udsteder kommandoer til sine underordnede – det er så de underordnede selv, der finder ud af om de skal reagere på disse kommandoer.

I dette tilfælde lyder spillereglen ganske enkelt:

1) Hvis du genkender nummeret i det fælles interfaceregister som dit eget silonummer, skal du tænde din udgang (som starter sneglen) – ellers skal du slukke den.

Og det indre af silo-PLC'en indeholder ikke andet end dette:

Et lidt større eksempel

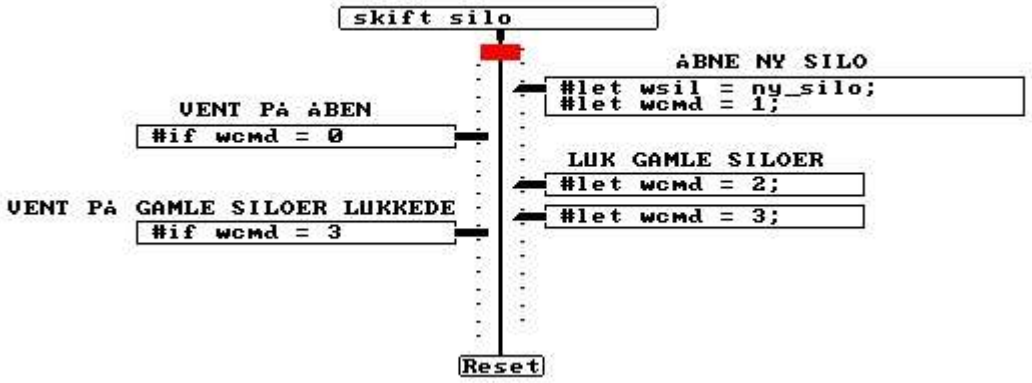
Her har vi en mini-PLC, der kan styre et redlerspjæld, der sidder side om side med andre redlerspjæld under en fælles redler. Hvert spjæld fører ned til en silo. Denne gang dækker den grafiske forbindelse ikke over et enkelt, men over to forskellige interface-registre; kommando og silonummer. Spjældet supporterer følgende kommandoer fra kommandoregisteret:

1) Åbn spjæld hvis silonummer genkendes som ens eget. Når spjældet er åbent, kvitteres kommandoen, så chefen kan se, at opgaven er løst
2) Luk spjæld hvis silonummer ikke genkendes som ens eget
3) Nulstil kommando, hvis vi endnu ikke er færdige med at lukke spjældet fra kommando 2

Fidusen ved kommando 3 er, at man kan komme ud for, at kommando 2 får flere forskellige spjæld til at lukke sig på samme tid. Chefen har brug for at vide, hvornår alle er færdige. Chefen putter derfor et 3-tal i kommandoregisteret i hvert eneste scan. Så længe der er blot et enkelt spjæld, der endnu ikke er færdigt med at lukke sig, vil spjældet nulstille registeret, og chefen kan derved se, at alle endnu ikke er færdige. Førstkommende scan, hvor der ikke længere er spjæld, der mangler at gøre sig færdige, vil chefen genfinde det 3-tal, den selv puttede i registeret i forrige scan.

Dette viser et eksempel på den meget sikre kommunikation man får, når alle tråde arbejder strengt sekventielt. Når man stiller et spørgsmål, er man 100% sikker på, at alle andre tråde har haft mulighed for at reagere på dette spørgsmål, før man næste gang selv kommer til fadet. Det erstatter endeløse rækker af and eller or-gates – eller hvis ladder-diagram – endeløse rækker af serie eller parallelforbindelser.

Sekvenseren er som skræddersyet til den slags gruppe-forespørgsler, chefen typisk foretager sig - er alle færdige?



Her vises indholdet af chef-PLC'en. Vi har valgt at lade de grafiske interfaceregistre begynde med w. Først sender vi kommando om at få åbnet den nye silo (wcmd=1) – derefter venter vi på at den er åben (if wcmd=0) – så beder vi om at få lukket alle andre siloer (wcmd=2) – og endelig har vi den specielle gruppe-forespørgsel.

I starten af hvert scan spørges om 3-tallet har overlevet mødet med siloerne. Så længe det ikke er tilfældet, bliver murstenen liggende og trykker stiften ind i højre side. Derved skrives et nyt 3-tal, som siloerne så evt. kan nulstille.
Ved førstkommende scan, hvor siloerne ikke har overskrevet 3-tallet, falder murstenen ned til næste step.

Dette er et eksempel på, hvor lidt "overhead" der er brug for i kommunikationen mellem PLC'erne. Handshake er aldrig nødvendigt. Når chefen f.eks. skal udsende kommando 2, er det tilstrækkeligt at lade 2-tallet være synligt i en enkelt scantid. Alle siloer vil med sikkerhed opfatte det.

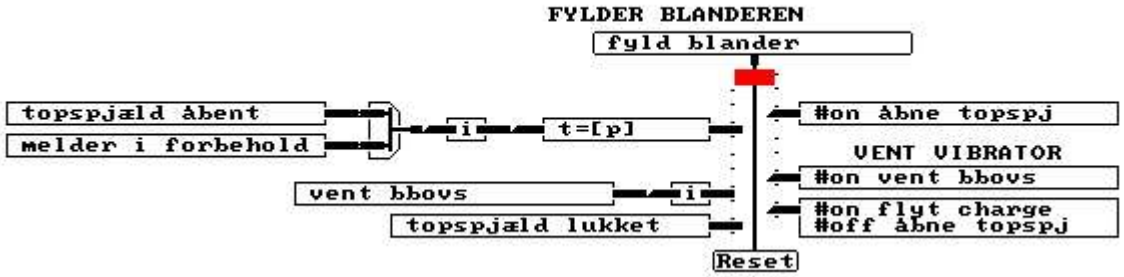
Generelt om spilleregler

Disse eksempler er valgt, fordi de viser nogle produktivitets-fremmende ting, der er specielle for multiobjekter – dvs mini-PLC'ere, hvor alle eksemplarer har samme programindhold, men forskellige data. Man kan komme ud for mere "eksotiske" ting, som f.eks. at kædereaktioner af kommandoer bruges til at "finde vej" gennem transportsystemer, men faktisk vil langt over halvdelen af multiobjekterne i et typisk projekt være banale opgaver som at forsyne maskindele med manuelle betjeningsfaciliteter – herunder SCADA support, alarmer osv. Disse importeres oftest fra den værktøjskasse, superbrugeren har lavet.
Meget af koden i et typisk projekt vil være single-objekter, der er lavet specielt til opgaven – men der er alligevel grundregler, der går igen:

1) Der vil ofte være en chef, der udsender kommandoer til en gruppe underordnede
2) Chefens kommandoer vil ofte være af generel art, så det overlades i stort omfang til de underordnede selv at beskæftige sig med detaljerne – herunder om det overhovedet er relevant for den enkelte at reagere på kommandoen

Når der ikke er tale om gruppe-kommunikation, anvendes almindeligvis ikke tal-registre. Her anvendes i stedet de mere selvforklarende on/off bit, der udtrykkes med hele sætninger.

Noget i stil med dette:



Langt det meste kode er lige ud ad landevejen. Ovenstående sekvenser løser flg. opgave. Den startes ved at et fremmed job (kunden) siger ”#on fyld blander”. Dette får den røde mursten til at falde og igangsætte flg. aktiviteter:

1) Åbne blanderens topspjæld
2) Vente på åbne melding samtidig med tommelding fra melder i forbeholder i den tid, (p) der er specificeret i parametertabellen
3) Start et hjælpejob ”vent bbovs” som kører med en vibrator i en bestemt tid, hvorefter hjælpejobbet slukker sig selv.
4) Luk topspjæld (og send info om skærmopdatering ”flyt charge”) og vent på lukket melding.
5) Endelig slukker sekvenseren sig selv (reset) hvilket fortæller kunden, at opgaven nu er løst – dvs blanderen er fyldt.

For en ordens skyld, prøver vi lige at indordne systemet i en klassisk datalogisk sammenhæng.

Soft-PLC’en set med datalogisk/psykologiske briller

Skal det beskrives på godt dansk, kan det kaldes ”connectionism by force”. Programmøren tvinges så at sige til at arbejde ”konnektionistisk”. Det skal her forstås som den retning inden for objektorienteret programmering, der ligger stik modsat den ”procedurale”. I stedet for at have nogle få store objekter til at indkapsle en kompleks funktionalitet, har man mange – nogen gange rigtig mange – små objekter, hvis indbyrdes kommunikationsveje så vidt muligt eksponeres grafisk. Konnektionisme skal her kun forstås som variant inden for objektorienteret programmering – ikke i den rå betydning, der kendes fra f.eks. ”neural-net” og ”cell automation”.

Tvangen ligger primært i, at mini-PLC’erne er meget små. De kan hver især ikke indeholde ret meget kode, så man er ganske enkelt tvunget til at arbejde delvist konnektionistisk. Her bruges et objekt netop ikke til at opnå en ”you dont need to know” effekt – altså beskytte programmøren mod viden – men tværtimod til at få så meget som muligt af problemløsningen til at blive eksponeret på et overordnet niveau – nemlig i kommunikationen mellem de små PLC’ere – så flest mulige af de involverede interessenter opnår en fælles forståelse af, hvordan programmet virker.

Hvem der kan snakke med hvem, afbildes så vidt muligt grafisk. Synlighed af data har 4 niveauer:

1)Lokale – kun synlige inden for PLC-ens rammer
2)Wire – kan grafisk forbindes til bestemte andre PLC’ere, som så også får adgang til disse data

3)Global – synlig for alle
4)Fysisk in/out.- data, der udveksles med ydre fysiske enheder, relæ-boxe, analog-boxe mv

Ved multiobjekter er både lokal-data, wire-data og fysisk in/out specifikke for hvert eksemplar (version), mens global-data er fælles for alle versionerne. Global-data bruges derfor - sammen med multiobjekter - mest til broadcast-lignende formål.

Ved procesanlæg er det indlysende grafisk at arrangere de små PLC'ere, så de afspejler maskinernes indbyrdes relationer. Meget falder automatisk på plads i og med at informationsstrømme så naturligt kommer til at følge materialestrømmen fra maskine til maskine. Hver maskine har sin egen mini-PLC (i udgangspunktet).

Mere komplekse problemløsninger opløses først i flere simplere del-løsninger (mini-PLC'ere), hvis funktionsbeskrivelser skal være forholdsvist enkle at udtrykke sprogligt. Deres behov for indbyrdes udveksling af data dækkes ved at give dem nogle multi-benede stikforbindelser i overbliksbilledet, hvor stikbenenes indbyrdes afstande matcher stikbenene hos de mini-PLCer, de er beregnede til at kunne samarbejde med.

Det perfekte objekt-interface

Det perfekte interface er det multibenede stik!
Et simpelt eksempel på et multibenet stik, er et el-stik til stikkontakten. Her er der 2 (eller 3) ben på stikket, hvis indbyrdes afstand skal passe med stikkontaktens huller.

Hvor den fysiske virkeligheds multibenede interface skæmmes af den fysiske skade, der opstår, når den utålmodige bruger forsøger at mase to elektronik-stik sammen, der ikke passer til hinanden, sker dette ikke i den digitale version.
Her er det multibenede stik virkelig det perfekte interface. Afstandene mellem benene på et stik kan af objektdesigneren justeres helt ned på pixelniveau, og giver masser af kombinationer ligesom hakkerne i nøglen til en dørlås. Hvis alle benene passer sammen, er de to objekter kompatible, og kan garanteres at ville fungere sammen. Også udsnit af stik følger denne regel. Hvis et objekt med f.eks. 2 ben i stikket, matcher to af benene hos et objekt med f.eks. 4 ben i stikket, - ja så skal der være 100% funktionel kompatibilitet, hvis de to objekter forbindes med netop de to stikben. For superbrugeren, der designer standardobjekter til brug for en "værktøjskasse", skal gælde det dogme, at hvis det er muligt for normal-brugeren at forbinde stikket til et andet standardobjekt fra samme værktøjskasse, skal de to objekter kunne "forstå hinanden" - de skal være 100% enige om spillereglerne.

Den lidt mere ambitiøse normal-bruger kan tillade sig at se stort på superbrugerens garantier, og selv påtage sig ansvaret for at sammenkoble to standardobjekter, hvor benafstandene ikke passer sammen. Man trækker bare objekterne et par centimeter fra hinanden, og forbinder dem med fleksible "ledninger" i stedet for at lade dem være forbundet direkte.
Dette fortæller en fremmed programmør, at vi her har et standardobjekt, der bruges på en lidt anden måde end oprindeligt tiltænkt. Allerede på overbliks-billede niveau.

Legetøjs-sprog søger at dæmpe angst

Selve programsproget, der virker inde i de små mini-PLC'ere, bygger på mekaniske objekter i stedet for elektriske, og skal give mindelser om legetøj til større børn – det kan godt virke lidt barnligt.
Hvis man kan få pustet liv i barndommens legelyst, har man en nyttig modgift mod den "frygt for at miste overblikket" som – paradoksalt nok – er den hyppigste årsag til, at man faktisk mister overblikket.

En programmør, der er under voldsomt tidspres for at løse et problem, vil måske smile tappert og kalde det ”udfordringer”, men indvendigt er der frygt, og denne frygt øger risikoen for at stirre sig blind på ikke-relevante detaljer, og ikke kunne få øje på det indlysende.

Måske ville angstdæmpende medicin virke lige så godt, men GOPA kræver ikke recept fra lægen, og en angst der dæmpes af legelyst må vel – alt andet lige – gøre ”patienten” mere vågen og observant, end en angst der dæmpes medicinsk.

Selve det at bruge mekaniske objekter – sekvensere og gates – i stedet for elektriske, som de fleste PLC-systemer gør, har til formål at øge blodtilstrømningen til den del af hjernen, der beskæftiger sig med mekaniske problemer. Da den ydre virkelighed netop er mekanisk, øges chancen for at programmøren, så ofte som muligt, tænker på netop den mekaniske virkelighed, programmet skal styre. En hyppig kilde til dårlig programmering er, at programmøren primært fokuserer på den ydre virkelighed ved projektstart, og så efterfølgende bliver så forelsket i sin egen kode, at den ydre virkelighed ender med at blive en distraktion, snarere end den bliver formålet med det hele.
Det er sundt at blive tvunget til at løse en mekanisk opgave med mekaniske værktøjer.

Applikationsprogrammørens uddannelse

Der opereres med 2 slags brugere. Normalbrugeren og superbrugeren. Normalbrugeren kan uddannes på 2x4 timer fordelt på 2 dage – mens superbrugeren skal bruge ca. en uge, og – modsat normalbrugeren - i forvejen skal have et vist kendskab til PC’ere på brugerniveau (filhåndtering o. lign.). De to betegnelser afspejler et ideal, som endnu ikke er nået. På nuværende tidspunkt er det sådan, at normalbrugeren kun formodes at kunne løse meget små styringsopgaver helt på egen hånd. For at kunne løse mellemstore og store styringsopgaver skal der være en ”ansvarshavende superbruger” på projektet. I forhold til denne kan et antal ”normalbrugere” så fungere som medhjælpere.

Når en større antal mindre tilretninger er gennemført på værktøjssiden (typisk sikkerhedsforanstaltninger), vil normalbrugeren også kunne gennemføre store projekter på egen hånd. Så vil superbrugeren være noget man kun behøver ringe til i ekstreme tilfælde – reparation af ødelagte filer o. lign.

Dog kan normalbrugerens produktivitet allerede nu, blive særdeles høj, hvis superbrugeren har leveret en værktøjskasse af – typisk branche-specifikke - præfabrikerede standard-objekter, hvor stikforbindelserne overholder ”plug-n-play” reglen: Hvis du kan lave ”plug” (stikket passer) skal PLC’en levere ”play” (fungere meningsfuldt).

Fællesnævneren er at udnytte de virtuelle fordele

Der er 3 niveauer, hvor vi kan sige, at vi udnytter, at det virtuelle er bedre end det tilsvarende fysiske:

1. Selve soft-PLC tanken – at man i én og samme computer simulerer mange forskellige mindre computere – er velkendt. Man slipper for at skulle ligge inde med mange forskellige reservedele, og kan når som helst udskifte ”dumme” in/out enheder med andre fabrikater, fordi de ikke indeholder nogen viden. Al viden er koncentreret i soft-PLC’en.
2. Hvis man skulle opnå GOPAs fordele i forhold til problemformulering, genbrug og fejlretning, kunne man i princippet godt montere et stort antal meget små PLC’ere rundt omkring i fabrikken. Men det ville gøre kommunikationen vanskeligere – der ville være behov for handshake m.m. - det er der ikke i GOPAs virtuelle verden, der arbejder strengt synkront og deterministisk. Og i GOPAs verden kommer man heller ikke ud for det bøvl, det vil give, hvis f.eks. 1 eller 2 PLC’ere bliver ramt af en strømafbrydelse, mens de øvrige arbejder videre. I en virtuel verden er der aldrig strømafbrydelser.

3. Den mest gennemprøvede måde at kontrollere om to apparater er funktionelt kompatible på – se om deres multibenede stik passer sammen – er uproblematisk i en virtuel verden. I den fysiske verden ses det ikke så sjældent, at folk ødelægger elektronik-stik ved at mase dem sammen.

Info til kerne-programmøren

Realtidskerner har en tendens til at ville gøre prioritet proces-afhængigt i stedet for situations-afhængigt. I GOPA har alle tråde potentielt adgang til højeste prioritet – men programsproget opfordrer programmøren til at udnytte ”GOPA's første lov”:

”Når en tidskritisk begivenhed netop har fundet sted, er der dejligt længe til, at samme slags begivenhed igen kan finde sted.”

Man kan tænke på en radioantenne. Når en bølgetop netop har passeret, er der længst mulig tid til næste bølgetop vil passere. Denne synsvinkel er med til at gøre begrebet ”båndbredde” veldefineret.

Superfrivillig tidsdeling slår både interrupt og DMA

Alle tidskritiske opgaver i det synkront-deterministiske GOPA-system følger i princippet denne skitse:

1. Man forbereder sig på at reagere på en tidskritisk begivenhed – dette må godt tage lidt tid.
2. Man stiller sig i venteposition – venter på at den tidskritiske begivenhed indtræffer.
3. Den tidskritiske begivenhed er netop indtruffet, og de mest tidskritiske dele af håndteringen løses i dette step.
4. Der ryddes op efter den tidskritiske begivenhed – det må gerne tage lidt tid, da der jo netop er længe til, at samme tidskritiske begivenhed igen vil indtræffe.

Denne evige cyklus supporter step-sekvenseren på simpel vis. De to faser som gerne må bruge tid: 1 og 4 strækker sig over flere step, mens overgangen 2-3 klares i et enkelt step (step = at CPU'en fastholdes til det er overstået).
Selv et worst-case for mange realtidskerner – at samtlige tidskritiske begivenheder i hele systemet indtræffer i selvsamme nano-sekund – er intet problem for GOPA. Alle de mange programmer gennemløb simultant den tidskritiske, men kortvarige fase 2-3 overgang – men nu gælder det jo netop for alle de tidskritiske begivenheder, at der er længe til de igen vil kunne indtræffe, så der er rigelig tid til at gennemløbe fase 4 og 1 - og igen stille sig parat ved fase-2 – klar til at høre ”start-pistolen” næste gang.

Et klassisk interrupt-drevent operativsystem ville bruge 20-50 gange så lang tid på at løse et sådant 1-nanosekund sammentræf af tidskritiske begivenheder. Opbygningen er umiddelbart ikke så forskellig fra et klassisk system. Den tidskritiske overgang 2-3 svarer til den hurtige interrupthandler – de øvrige faser svarer til hvad operativsystemets højt prioriterede tråde normalt tager sig af. Men der vil være en masse indbyrdes sniksnak (handshakes) og selve hardware-interruptsne introducerer i sig selv en betydelig belastning.

GOPA's anden lov siger:

”Uanset hvilken kombination af logiske begivenheder realtidskerne-designeren har stillet til rådighed som kriterie for at blive kaldt op ved høj prioritet, er der altid én bestemt der mangler: Nemlig den, man lige står og har brug for”.

Det er bedre at give applikationsprogrammøren selv direkte adgang til den høje prioritet.

Og så for fuldstændighedens skyld GOPA's tredje lov:

”CPU-tid er ikke noget man tiltvinger sig adgang til. Det er noget man tilbyder andre ved selv at holde sig tilbage.”

Det er GOPA's tredje lov i kombination med ”pop-only” stakken, der har givet navnet ”Super-frivillig tidsdeling”.

Afgørende er selvfølgelig, at det skal være billigt at switche ned gennem de tusindvis af tråde, der det meste af tiden bare står og poller efter, om der er kunder i butikken.

Pop-only stakken

Hvad der er ”quick and dirty” og hvad der er ”pænt”, er i virkeligheden kun et spørgsmål om, hvordan programværktøjer med tilhørende debuggere er udformet. Kunsten er, at gøre de teknikker, der på hardware-niveau giver størst datakraft pr. transistor, attraktive og letforståelige for programmøren. Man må ikke glemme at udnytte det forhold, at menneskehjernen (endnu) er mere fleksibel end computerne.

Vi kan bruge processorens stack som en superhurtig tråd-skifter. Instruktionen RETurn, som normalt er beregnet til at returnere fra et subrutine-kald, bruges i stedet til at skifte tråd med. Vi bruger den således ikke sammen med CALL instruktionen.

Jamen hvordan laver i så jeres serielle funktionskald? - kunne man spørge.

Svaret er, at de er afskaffet. Det serielle kald med tilhørende behov for at beskytte en ”kontekst” er det største benspænd af alle i den nødvendige konvertering fra enkelt-CPU til multi-CPU systemer. Noget der er i et overraskende skæbnefællesskab med det tilsyneladende modsatte: At få én CPU til at løse mange forskellige opgaver effektivt og med forudsigelige responstider.

Historisk var pop-only stakken en enkelt-CPU realisering af, hvad der i det oprindelige multi-CPU koncept hed ”skrive-stafetten”. Det var noget med klynger af CPU’er, der kun en ad gangen havde retten til at skrive, mens alle andre lyttede med (snooping).
CPU’erne samarbejdede ikke via delt ram, men brugte et (i princippet ubegrænset skalerbart) bredt word, som én CPU af gangen skrev til (skrive stafetten) mens samtlige andre snoopede – og kunsten var at få fordelt opgaverne fase-rigtigt rundt i cyklussen.

Det oprindelige koncept, der aldrig kom længere end til skrivebordet, handlede netop om vigtigheden af determinisme, hvis et stort antal CPU-er skulle samarbejde effektivt om løsningen af en fælles opgave. Ellers løber man alt for nemt ind i problemet med at ”taxien ankom til stationen lige 2 minutter efter at toget var kørt sin vej”.

Nyheder og fremtidsmuligheder

For at kunne påbegynde markedsføring med noget, der er lettere at kommunikere ud i medierne end blot ”fullmotion brugerinterface”, som man næsten skal have prøvet for helt at forstå det, er der lavet noget vi kalder for ”Ultimativ Sporbarhed”.

Sporbarheds-film ved fjernstyring

Applikationerne har lige nu en fjernstyrings-facilitet, der over internettet kan oppebære mere end 95% af full-motion kvaliteten. Al proces og betjening optages som en slags videoklip, der efterfølgende vil kunne afspilles. Kompressionen er overordentlig effektiv – mindre end 1MB pr. times spilletid selv ved 30 billeder i sekundet. Faciliteten bruges lige nu dels til, at man kan fjernstyre, hvor filmklippet bruges som ”ultimativ sporbarhed” - den der fjernbetjenede kan dokumentere alle handlinger ned til mindste detalje. Dels bruges faciliteten til at verificere ændringer i anlægget. Der optages filmklip af typiske kørsler før ændringen. Disse filmklip kan efterfølgende sammenlignes med tilsvarende kørsler optaget efter ændringen. Det er en overbevisende måde at dokumentere på, om en investering har kunnet betale sig.

Kontinuerlig filmoptagelse

Med ganske få udviklingstimer (100-150) kan bestående facilitet, der er specifik for fjernstyring, videreudvikles til ”Kontinuerlig Ultimativ Sporbarhed”.
Her sker der en konstant filmoptagelse af alle aktiviteter i 100% fullmotion – lokalt og helt uafhængigt af internettet.
Jeg vil tro, at vi er lysår foran resten af feltet, når vi kan tilbyde løsninger, hvor der er et ”virtuelt overvågningskamera” der optager al betjening og al proces med 20 eller 30 frames i sekundet og komprimerer det ned til 1MB/time. Selv en lille harddisk kan rumme mange års døgndrift produktion.
En hvilken som helst fejlsituation ude i anlægget burde efterfølgende kunne udredes ned til mindste detalje, da filmindspilningen følger, ikke blot grafikken, men også soft-PLC’en synkront.

Sikkerhed mod hackere

Det eneste man under fjernstyring transmitterer til fabriks-PC’en, er info om musens bevægelser og indtastninger fra tastaturet. Da det ikke lokalt er muligt at kompromittere programmet via betjening af keyboard eller mus, er det heller ikke muligt for fjern-operatøren.
Lokal operatøren har mulighed for at afslutte programmet via keyboardet, men så ankommer man til en DOS-lignende kommandoprompt, hvor al anden in/out aktivitet end keyboard og skærm er nulstillet. Systemet kan kun kompromitteres ved at modificere de filer, der indeholder den nye programversion. Disse indlæses efter telefonisk samråd med operatøren, hvor længde, dato og evt. tjeksum for pakke-filen mundtligt kan verificeres i telefonen.

Oplagte muligheder for videreudvikling

Som nyttig spin-off effekt af ”Kontinuerlig Ultimativ Sporbarhed” kan der laves en fejl-forebyggende automatisk overvågning. Det er en speciel facilitet ved data-kompression, der udnytter, at maskindele der fungerer godt, vil give jævne signaler, der er lette at komprimere. Optakten til en fejl vil vise mere ujævne signaler (det samme som operatøren holder øje med via fullmotion grafikken), som kompressoren opdager ved at data for netop dette målepunkt bliver vanskeligere at komprimere. Det vil således kunne producere forebyggende vedligeholdelses-alarmer på formen ”prøv lige at holde øje med xxxx”.

Langsigtede perspektiver

GOPA's grundlæggende princip (frivillig tidsdeling baseret på pop-only stak) byggede oprindeligt på et multi-processor koncept, der så i praksis er udført i en enkelt-processor udgave. Alle processor-arkitekturer kan udnyttes, men dem der har ”Harris-arkitektur” f.eks. Intel og ARM er specielt velegnede. Det bygger på en erkendelse af, at determinisme og synkronitet er vigtige egenskaber – på samme måde som det er godt at have lyskurve i en trafikeret storby. I alleryderste konsekvens vil GOPA-kon-

ceptet kunne siges, at være den arkitektur, der giver mest datakraft pr. transistor, hvis forudsigelige responstider er et must.
Eller – for nu at gå helt tilbage til det oprindelige multiprocessor-koncept – det koncept der giver bedst adgang til den determinisme, der er forudsætningen for, at et stort antal CPU'ere kan samarbejde effektivt uden at "gå i vejen for hinanden" og uden at komme ud for, at toget desværre kørte sin vej 2 minutter før taxien ankom til stationen.