



CHALMERS

Mjukvaruutveckling av FreeRTOS-baserade inbyggda system

På hårdvara och molnet

Examensarbete inom Data- och Informationsteknik

LUDVIG SVENSSON
MICHAEL DANG

Data- och Informationsteknik

CHALMERS TEKNISKA HÖGSKOLA
Göteborg, Sverige 2022

EXAMENSARBETE 2022

Mjukvaruutveckling av FreeRTOS- baserade inbyggda system

På hårdvara och molnet

LUDVIG SVENSSON
MICHAEL DANG



CHALMERS

Data- och Informationsteknik
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2022

Mjukvaruutveckling av FreeRTOS-
baserade inbyggda system
På hårdvara och molnet
LUDVIG SVENSSON
MICHAEL DANG

© Ludvig Svensson, 2022.

© Michael Dang, 2022.

Handledare: Jonas Duregård, Institutionen för data- och informationsteknik

Examinator: Lars Svensson, Institutionen för data- och informationsteknik

Examensarbete 2022

Data- och Informationsteknik

Chalmers University of Technology

SE-412 96 Gothenburg

Typeset in L^AT_EX
Gothenburg, Sweden 2022

Förord

Den här rapporten är ett examensarbete som utfördes vid institutionen för Data- och informationsteknik på Chalmers Tekniska Högskola i samarbete med företaget Diadrom. Vi vill tacka Henrik Fagrell och Jonas Hellberg från Diadrom som har stöttat oss under arbetet med deras tekniska expertis. Vi vill också tacka vår handledare Jonas Duregård på Chalmers för handledningen under arbetet.

Michael Dang & Ludvig Svensson, Göteborg, juni 2022

Sammanfattning

Mjukvaruutveckling mot inbyggda system har historiskt varit präglad av begränsad tillgång till fysisk hårdvara. Som ett resultat kan utvecklingen och testerna som görs mot fysisk hårdvara orsaka konflikter och tidskrävande problem mellan teammedlemmar. Diadrom strävar efter att effektivisera utvecklings- och distributionsprocessen genom att utforska möjligheterna att simulera inbyggda system i molnet, så att teammedlemmarna inte begränsas av den fysiska hårdvaran.

Syftet med examensarbetet är att undersöka möjligheterna att simulera hårdvara baserad på FreeRTOS applikationer i molnet. Dessutom: hur kan mjukvaruutveckling av en inbyggda applikationer utformas för att underlätta virtualisering i molnet men samtidigt fungera mot den fysiska hårdvaran?

Undersökningen började med att definiera problemet och sedan gå vidare till att utforma ett lösningskoncept. Innan man kom fram till en lösning behövdes en stor mängd efterforskning och testning göras. Huvudfokusområdet låg kring Amazon AWS och ARM AVH. Efter avslutad efterforskning och testning kunde en prototyp implementeras.

Den praktiska bidraget var ett proof of concept som kördes på hårdvara och AWS. Samtidigt är det teoretiska bidraget ett tillägg till kunskapen om simulering av inbyggd mjukvara som kommer att vara användbar under den kommande framtiden. Sammanfattningsvis har resultaten visat indikationer på att det är möjligt att skapa en simulerad miljö för utveckling av inbyggd applikationsprogramvara som kan representera en fysisk miljö inom en snar framtid

Abstract

Software development on embedded devices is historically constrained by scarce access to target hardware. As a result, the development and testing done on real hardware could cause conflicts and time-consuming problems between team members. Diadrom aims to speed up the development and deployment process by exploring the possibilities to simulate embedded devices in the cloud, so that the team members are not limited by the hardware device.

The purpose of this project is to examine the possibilities of simulating hardware based on FreeRTOS embedded applications in the cloud. Additionally, how can software development of an embedded application be designed to facilitate virtualization in the cloud yet work on the target hardware device?

The examination started off by defining the problem and then moving on to design a solution/concept. Before coming up with a solution, a considerable amount of research and testing had to be done. The main environment of investigation were around Amazon AWS and ARM AVH. Upon finishing the research and testing, a prototype could be implemented.

The practical contribution effort was a proof of concept running on hardware and AWS meanwhile the theoretical contribution is an addition to the knowledge of simulation of embedded software that will be useful in decades to come. In conclusion, the results have shown indications that it is possible to create a simulated environment for embedded application software development that could represent a physical environment in the near future.

Keywords: embedded devices, target hardware, simulating hardware, FreeRTOS, AWS, AVH.

Akronymer

Nedan är en lista på akronymer som använts inom rapporten:

API	Applikationsprogrammeringsgränssnitt
AVH	Arm Virtual Hardware
AWS	Amazon Web Services
BSD	Berkley sockets
GPIO	General-purpose input/output
I/O	Input/Output
SSH	Secure Shell
UART	Universal Asynchronous Receiver/Transmitter
USART	Universal synchronous and asynchronous receiver-transmitter
VHT	Virtual Hardware Target
CI	Continuous integration
CD	Continuous delivery
TCP	Transmission Control Protocol
MQTT	MQ Telemetry Transport
HTTP	Hypertext Transfer Protocol
LoRaWAN	Long Range Wide Area Network

Innehåll

Lista av akronymer	ix
Figurer	xiii
1 Inledning	1
1.1 Bakgrund	1
1.2 Syfte	1
1.3 Avgränsningar	1
1.4 Precisering av frågeställningen	1
2 Teknisk bakgrund	3
2.1 ARM Virtual Hardware (AVH)	3
2.2 FRDM K64F	3
2.3 ARM Cortex-M4	4
2.4 FreeRTOS	4
3 Metod	5
3.1 Definiera problemet	5
3.2 Identifiering av lösning	5
3.3 Implementation av prototyp	6
4 Resultat	11
4.1 Integrering av hårdvara	11
4.2 Simulering av mikroprocessor	11
4.3 Exekveringsprestanda i ticks	12
5 Diskussion	15
6 Slutsats	17
6.1 Simulatorn och target	17
6.2 Proof of concept	18
Källor	19
A Kod för att grafiskt visa värdena från bilagorna B-C	I
B Ticks per iteration på simulatoren	III

C	Ticks per iteration på target	V
D	Main program testkod	VII

Figurer

3.1	En översikt över några funktioner som inkluderas i VHT	6
3.2	Överblick på programmet PuTTY	7
3.3	Exempel på en körning i simulatorn	8
3.4	En överblick på Keil MDKs Pack Installer	9
3.5	Strukturen för funktionerna i standard I/O C run-time biblioteket . .	9
4.1	Jämförelse mellan det fysiska kortet och det simulerade kortet i hur många ticks det tagit att genomföra testet	13

1

Inledning

1.1 Bakgrund

Diadrom utvecklar mjukvarulösningar för olika sorter av inbyggda system där man vanligtvis enbart har en eller två uppsättningar av hårdvaran. Därmed kan det uppstå en åtkomstkonflikt när flera medarbetare vill använda samma hårdvara. Därför vill företaget undersöka potentialen att simulera delar av hårdvaran i molnet för att underlätta utvecklingen av projekt.

1.2 Syfte

Syftet med uppdraget är att undersöka ifall det är möjligt att simulera hårdvara baserad på FreeRTOS i molnet samt hur låg fördröjning man kan uppnå mot tidskritiska system. Målet med uppdraget är att få fram en virtuell kopia av hårdvaran, reda ut vad för system som kan implementeras samt få fram mätbara värden på den fördröjning som sker och se om den är acceptabel mot tidskritiska system.

1.3 Avgränsningar

Projektet kommer inte att fokusera på att utveckla fysisk hårdvara eller att optimera implementering mot molnet utan snarare se till hur stor del man kan simulera en mikroprocessor.

Simuleringen av hårdvaran kommer att ske genom AWS där det finns stöd för API med FreeRTOS samt för hårdvaran som kommer att användas under projektet.

Eftersom fokus för detta uppdrag är att undersöka hur väl man kan simulera hårdvara i molnet och inte utveckling av mjukvara kommer projektet att vara begränsat till lösningar som redan existerar på marknaden.

1.4 Precisering av frågeställningen

Hur kan ett inbyggt system simuleras i en cloud-miljö med syfte att uppnå utvecklingseffektivitet och minska behovet av fysiska instanser av hårdvara?

1. Inledning

Man kan förutsätta att den simulerade versionen kommer bete sig annorlunda än den fysiska. Vad gränserna går för vad som kan betraktas som tillräckligt lika är troligen beroende på den individuella produktens krav. Vi kommer att undersöka skillnader i aspekter såsom i process scheduling, exekveringshastighet, latency och minnesanvändning.

2

Teknisk bakgrund

I detta kapitel beskrivs de nödvändiga begrepp och teknologier som har använts under projektet.

2.1 ARM Virtual Hardware (AVH)

ARM Virtual Hardware är en del av ARMs modelleringsteknologi som tillhandhåller modeller av Arm-baserade processorer, system samt tredjeparts-hårdvara för bland annat applikationsutvecklare. Målet med teknologin är att kunna utveckla och testa mjukvara innan man har fått tillgång till den riktiga hårdvaran [1]. Förutom simulering av hårdvara finns integrering utav CI/CD omgivningar för att testa och verifiera kod.

ARM Virtual Hardware Developer Resource (AVHDR) som är en del av AVH ger utvecklaren tillgång till drivrutiner för gränssnitt som gör att man kan koppla den simulerade processorn mot virtuell utrustning med hjälp av python script. Användningar för dessa funktioner kan kopplas allt från ljudprocessering, maskininläring till IoT applikationer [2].

2.2 FRDM K64F

FRDM K64F är ett utvecklingskort för Kinetis® platformen utvecklad av NXP semiconductors. Kortet drivs av en mikrokontroller MK64FN1M0VLL12 vilket är baserad på en arkitektur från en ARM Cortex-M4 mikrokontroller [3].

Formfaktorn för i/o portarna på kortet är kompatibel med formfaktorn av en Arduino UNO Rev3 samtidigt som det finns en ethernet port samt två stycken mikro-USB portar där det går att driva kortet genom 5V [3].

För att debugga på kortet finns det en inbyggd adapter som är kopplad till en av mikro-USB portarna. Debuggern heter OpenSDAv2 och den medför att man kan debugga kortet genom datorn, flasha kortet samt hålla en seriell kommunikation mellan dator och kortet [4].

2.3 ARM Cortex-M4

ARM Cortex-M4 är en processor som är designad av ARM och släpptes året 2010 och använder sig av en 32-bit arkitektur. Den används inom mikrokontrollers för allt från system som kräver låg energiförbrukning och låg minneskapacitet till mer krävande system [5]. Processorn är konstruerad för att vara energisnål och har därmed ett antal vilofunktioner för att minska energikonsumtionen [6].

2.4 FreeRTOS

FreeRTOS är ett realtidsoperativssystem som är designat för små mikroprocessorer. Det distribueras under en gratis licens distribuerad av MIT där själva kärnan och ett antal bibliotek inkluderas. Exempel på funktioner som finns att ta del av är TCP, MQTT, HTTP och loRaWAN vilket är olika typer av nätverksprotokoll. Funktioner som finns som standard inom FreeRTOS kärnan är threads och funktioner för att få ut mätvärden som till exempel ticks, vilket är en intern konfigurerbar klocka som FreeRTOS använder sig av. [7].

3

Metod

3.1 Definiera problemet

Mjukvaruutveckling för inbyggda system sker vanligen med hårdvara som fysiskt finns tillgänglig. Behovet av hårdvaran gör att utvecklingen kan sinkas vid tillfällen där man inte har tillgång till denna. Vid tillfällen där något av de kort eller korten som man har tillgång till under utvecklingen skulle skadas skapas det en flaskhals. Det kan medföra lång väntetid tills man kan fortsätta att utveckla sin kod med hårdvaran.

Liknande problem uppstår när man arbetar med prototyper av inbyggda system, då är det vanligt att man enbart producerar ett lågt antal prototyper då det uppstår en stor kostnad att producera i små serier. Det medför att om det är ett stort team som jobbar med att utveckla mot prototypen så kommer inte alltid de som vill testa sina funktioner ha tillgång till prototypen.

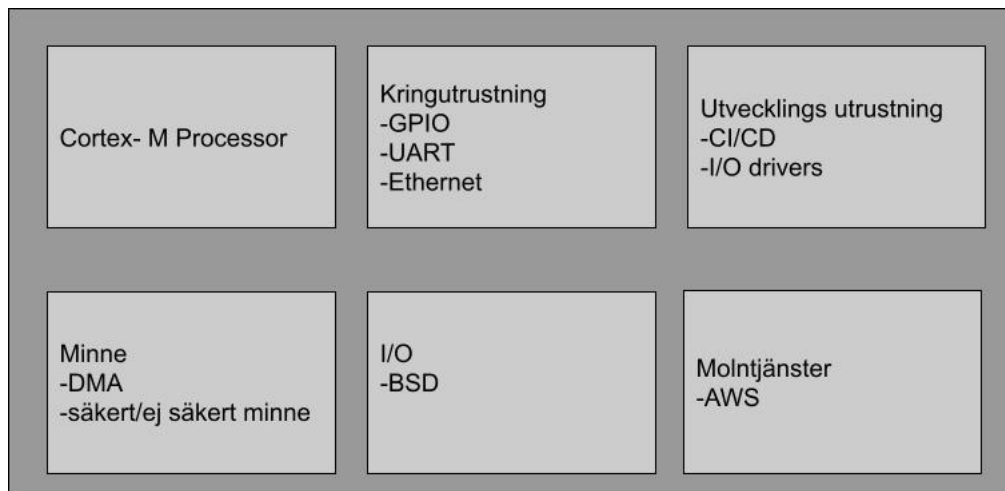
3.2 Identifiering av lösning

För att minska behovet av specifik fysisk hårdvara, vore det önskvärt med något annat alternativ som fungerar liknande eller bättre att utveckla mot och som inte drar ner på utvecklingstiden.

Ett alternativ till att ha tillgång till fysisk hårdvara kan vara att simulera hårdvaran. Detta kan ske på olika nivåer. Olika nivåer på implementation krävs för olika typer av utveckling mot hårdvaran. Det kan vara från att testa hur exekveringshastigheten är på mikroprocessorn till att implementera kontakt med I/O.

Vid implementering av en simulator för en mikroprocessor är det viktigt att man inte enbart får veta att koden kan köras på simulatören utan att man även får fram inställningar som gör att man sedan direkt kan köra samma kod på den fysiska hårdvaran utan att göra några ändringar. Därför kommer en simulator som heter ARM VHT att användas för att undersöka till vilken detaljnivå som man kan implementera en mikroprocessor samt hur prestandan skiljer sig mot en fysisk motsvarighet.

Enligt Arm VHT är det möjligt att simulera alla deras Cortex-M processorer vilket inkluderar simulering där man använder sig av "riktigt" simulerat minne, I/O portar som BSD sockets samt kringutrustning som GPIO [2].



Figur 3.1: En översikt över några funktioner som inkluderas i VHT

Som man ser i Figur 3.1 ska det existera lösningar för att testa och utveckla kod mot simulerade portar och kommunicera mellan det simulerade kortet och datorn som simuleringen körs på.

Förutom simulering av mikroprocessorn körs koden inom ett lättviktigt realtidsooperativsystem som är designat för att köras på mindre system som mikrokontroller. Operativsystemet som implementeras baseras på FreeRTOS där det finns funktioner för att köra program i trådar och få ut data för att läsa av hur lång tid en funktion eller program har tagit [8]. Så för att få ut data för skillnader och likheter mellan kort och simulatoren kommer all kod som testas köras med FreeRTOS.

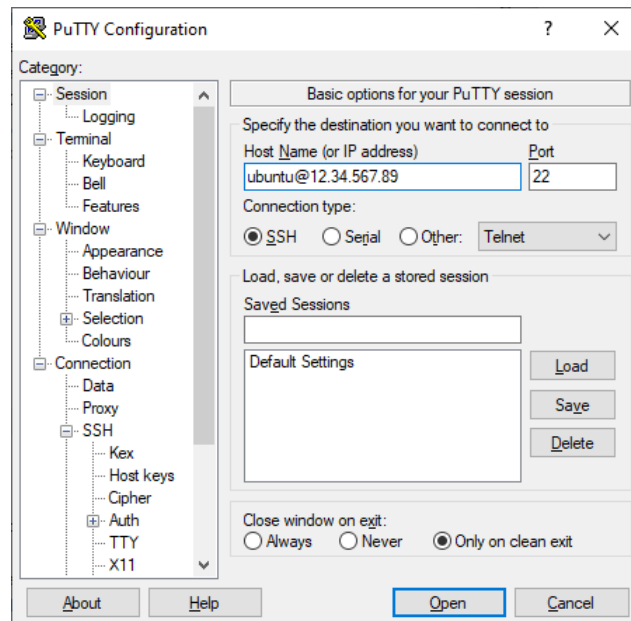
3.3 Implementation av prototyp

Innan man börjar med implementationen så behöver man ta reda på vad för processorkitektur hårdvaran använder sig av. För detta arbete användes utvecklingskortet FRDM-K64F som är baserat på en ARM Cortex-M4 processor. ARM har en produkt som kallas för Arm Virtual Hardware (AVH) där de tillhandahåller simulerade ARM-processorer, även kallade ARM Virtual Hardware Targets [2].

För att utnyttja dessa simulerade processorer i molnet krävs det att man använder Amazons tjänst AWS Marketplace då de är de enda som stödjer denna ARM-produkt för tillfället. Genom AWS Marketplace startar man en instans som innehåller produkten AVH, vilket är en virtuell server som ligger i AWS Cloud.

Det finns ett fåtal olika sätt att ansluta sig till servern beroende på vilken typ av operativsystem man använder. För detta projekt används kommunikationsprotokollet Secure Shell Protocol (SSH) med hjälp av programmet PuTTY för att på ett säkert sätt ansluta sig till den virtuella servern. Innan man startar en instans krävs

det att man skapar ett så kallat nyckelpar i AWS. Nyckelparet består av två delar, en offentlig nyckel och en privat nyckel. Dessa två nycklar används tillsammans som ens egna säkerhetsuppgifter för att visa instansen att man har rättigheter till att ansluta. Den offentliga nyckeln sparas i själva instansen medan man själv får lägga undan sin privata nyckel, då det är denna man använder sig av för att ansluta sig genom SSH med hjälp av PuTTY och instansens IP-adress.



Figur 3.2: Överblick på programmet PuTTY

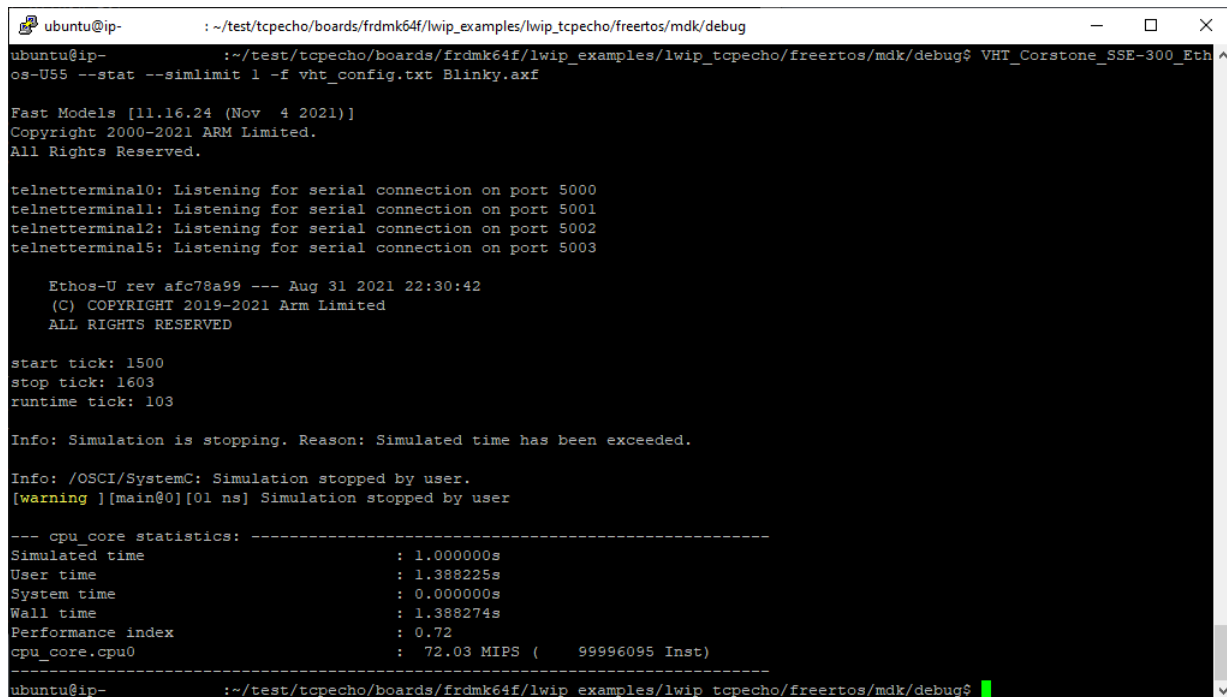
I Figur 3.2 kan man se hur programmet PuTTY användes för att ansluta till den virtuella servern. Under kategorin SSH på vänstra sidan kan man finna underkategorin Auth vilket ger en alternativt att importera den privata nyckeln för att autentisera.

Operativsystemet Ubuntu vilket är baserat på Linuxkärnan körs på den virtuella servern. För att köra en applikation mot en simulerad processor så behöver man namnet på den simulerade modellen och applikationen som ska köras. Applikationen väljs med hjälp av kommandoradsalternativet `-a` följt av namnet på den. Ett exempel på hur man kör en applikation mot en simulerad Cortex-M4 är som följande: `VHT_MPS2_Cortex-M4 -a Blinky.axf`. Som man kan se är filen av typen `axf` vilket krävs för att den simulerade modellen skall kunna köra applikationen. Denna filtyp kan man få ut från Arm Keil MDK vilket är ett program för att skapa applikationer. Kompilatorn i programmet skapar en exekverbar fil i formatet ELF som får filtillägget `axf`.

För att föra över en applikation till den virtuella servern så användes Github, genom att kopiera repository till servern. Med hjälp av Github funktioner som push och pull är det möjligt att snabbt få uppdaterade versioner av applikationerna. Figur 3.3 visar hur det kan se ut efter en körning i simulatorn. Med kommandoradsalternativet `-stat` får man ut diverse statistiker bland annat om hur lång tid körningen

3. Metod

tog för simulatören och i realtid.



```
ubuntu@ip- : ~/test/tcpecho/boards/frdmk64f/lwip_examples/lwip_tcpecho/freertos/mdk/debug
ubuntu@ip- :~/test/tcpecho/boards/frdmk64f/lwip_examples/lwip_tcpecho/freertos/mdk/debug$ VHT_Corstone_SSE-300_Eth
os-U55 --stat --simlimit 1 -f vht_config.txt Blinky.axf

Fast Models [11.16.24 (Nov 4 2021)]
Copyright 2000-2021 ARM Limited.
All Rights Reserved.

telnetterminal0: Listening for serial connection on port 5000
telnetterminal1: Listening for serial connection on port 5001
telnetterminal2: Listening for serial connection on port 5002
telnetterminal5: Listening for serial connection on port 5003

Ethos-U rev afc78a99 --- Aug 31 2021 22:30:42
(C) COPYRIGHT 2019-2021 Arm Limited
ALL RIGHTS RESERVED

start tick: 1500
stop tick: 1603
runtime tick: 103

Info: Simulation is stopping. Reason: Simulated time has been exceeded.

Info: /OSCI/SystemC: Simulation stopped by user.
[warning ][main@0][01 ns] Simulation stopped by user

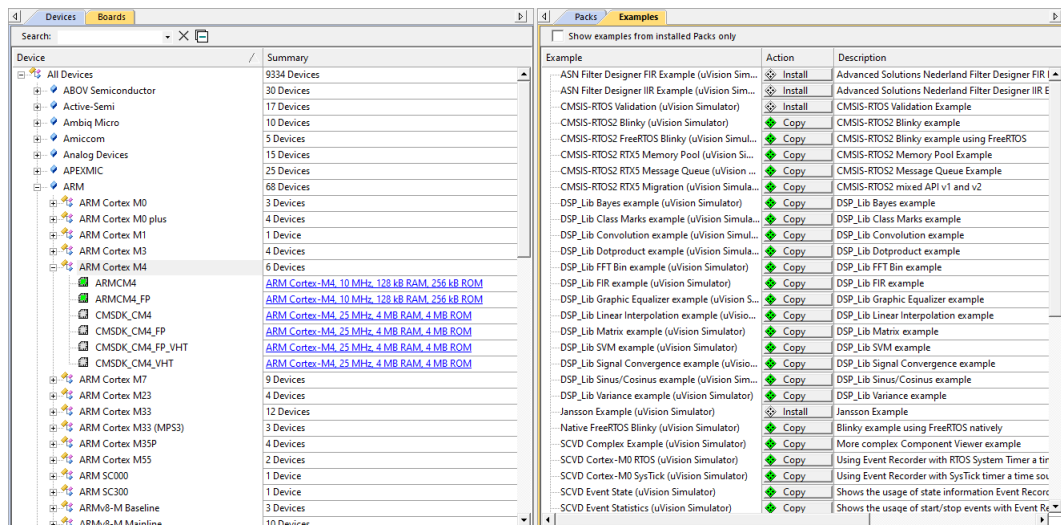
--- cpu_core statistics: -----
Simulated time      : 1.000000s
User time           : 1.388225s
System time         : 0.000000s
Wall time           : 1.388274s
Performance index   : 0.72
cpu_core.cpu0       : 72.03 MIPS ( 99996095 Inst)
-----

ubuntu@ip- :~/test/tcpecho/boards/frdmk64f/lwip_examples/lwip_tcpecho/freertos/mdk/debug$
```

Figure 3.3: Exempel på en körning i simulatören

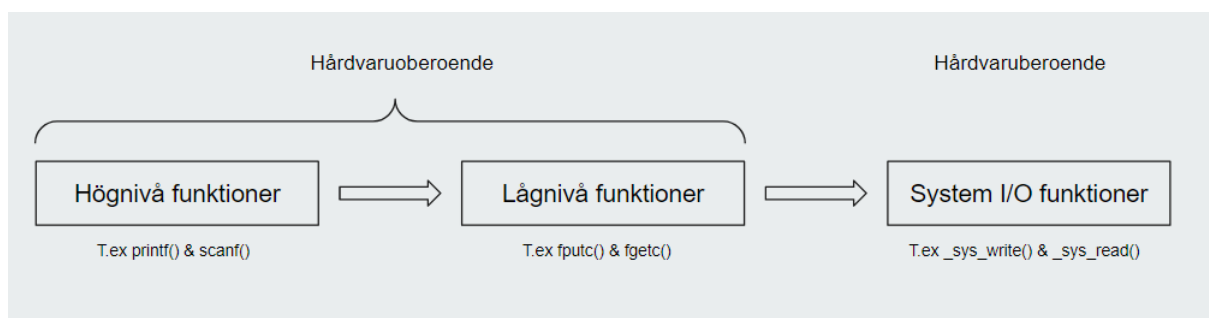
Som man kan se i början av Figur 3.3 så kördes applikationen mot en simulerad modell med namnet VHT_Corstone_SSE-300_Ethos-U55 och denna modell är baserad på en Cortex-M55 processor. Anledningen till att testerna i simulatören kördes mot en Cortex-M55 processor istället för Cortex-M4 som den fysiska hårdvaran använder sig av var svårigheter med att implementera funktionen för att kunna printa ut resultat samtidigt som FreeRTOS var integrerat.

Det som testades först var att ladda ner ett färdigt exempel som hade FreeRTOS integrerat på Cortex-M4 processorn från Keil MDKs så kallad Pack Installer. I Figur 3.4 kan man se Pack Installer fönstret och dess flera olika mjukvarupaket som erbjuds, såsom bibliotek och drivrutiner som kan användas för att vidareutveckla. Problemet som uppstod med FreeRTOS-exemplet var att man inte kunde utnyttja sig av `printf()` funktionen för att direkt skriva till kommandotolken i simulatören. Detta är på grund av att output i simulatören skickas via USART till telnetterminalen. Telnet är en typ av internetprotokoll som tillåter en användare att ansluta sig till en annan dator virtuellt genom en virtuell terminal [9]. Som man kan se i Figur 3.3 så lyssnar simulatören på telnet portarna 5000 till 5003 vilket man kan använda sig av för att kommunicera med simulatören.



Figur 3.4: En överblick på Keil MDKs Pack Installer

För att utföra input/output använder oftast programkoden I/O bibliotek funktioner som till exempel `printf()` och `scanf()`. I Figur 3.5 kan man se strukturen för funktionerna i biblioteket. Högnivå funktionerna utför input/output genom att kalla på lågnivå funktionerna som i tur kallar på system I/O funktionerna. Hög- och lågnivå funktionerna är icke beroende av hårdvara vilket system I/O funktionerna är, därav använder de första två funktionerna sig av system I/O för att kunna interagera med hårdvaran.



Figur 3.5: Strukturen för funktionerna i standard I/O C run-time biblioteket

ARM erbjuder en funktion som heter I/O Retargeting som ger möjligheten att ändra om I/O funktioner i standard C run-time biblioteket [10]. En möjlig lösning till det tidigare problemet är att modifiera outputen så att den blir USART istället för den vanliga STDOUT men detta gick inte för exemplet mot Cortex-M4 processorn. Varför detta inte gick var på grund av att exemplet för just Cortex-M4 processorn inte hade tillgång till alla nödvändiga drivrutiner och API:er. Dessa drivrutiner och API:er krävs för att kunna komma åt konfigurationsfiler för Cortex-M4 processorn för att sedan kunna genomföra retargeting.

För att lösa problemet hittades ett exempel med FreeRTOS integrerat mot en simulerad Cortex-M55 modell som lyckades att printa direkt mot telnet-terminalen

med hjälp av `printf()` funktionen. Detta exempel modifierades för att sedan köra testen som kommer att presenteras i resultatdelen av rapporten. Anledningen till att exemplet behövde ha FreeRTOS integrerat är att testerna utgår från att jämföra och mäta antalet ticks det tar för att köra testerna i både simulatorn och det fysiska kortet. Ticks i detta fallet innebär system tick, vilket är en tidsenhet som timers och fördröjningar är baserade på i realtidsoperativsystem [11].

4

Resultat

Tabellen 4.1 innehåller en sammanfattning på de tester som genomförts på både kortet och simulatören där den resterande delen av kapitlet ger mer insyn på hur dessa har genomförts.

Testtyp	Resultat av test
Integrering	Fick ut skrivningar men ej yttre kopplingar
Simulering av mikroprocessor	Fungerar motsvarande arkitektur
Testa exekveringshastighet i form av ticks	Fått ut resultat från både kort och simulatören

Tabell 4.1: Tabell med sammanfattning av test och resultat

4.1 Integrering av hårdvara

När integreringen av hårdvaran mot simuleringen genomförs fokuseras det på två punkter som ligger till grund för att de andra testen ska gå att genomföra.

- Simulering av processor från samma arkitektur
- Simulering av I/O

När man senare jämför hårdvaran mot den simulerade miljön är det viktigt att man har en simulering av arkitekturen som är så lik som möjligt mot den fysiska hårdvaran, för att få ett beteende som kan jämföras över de två plattformarna.

Som man kan se i Figur 3.1 så finns det funktioner för att simulera säkert minne vilket är minne som går att komma åt och osäkert minne som inte går att komma åt, BSD sockets, GPIO samt mera [2]. Det ger möjlighet att kommunicera med den simulerade mikroprocessorn genom den lokala servern som den ligger på.

4.2 Simulering av mikroprocessor

När det fysiska mikroprocessorn skulle simuleras visade det sig att ett antal funktioner som man kan se i Figur 3.1 ej blivit implementerade i Cortex-M4 modellen. Därmed går det inte att få ut information från simulatören som exempel utskrivningar av data från programmet.

Resultatet av att information ej gick att få ut från den simulerade Cortex-M4 mikroprocessorn simulerades istället en Cortex-M55 mikroprocessor [12]. Med få ändringar på inställningar av vilket kort man ska kompilera för går det att köra samma grundprogram på både det simulerade M55 och fysiska M4 mikropocessorn.

4.3 Exekveringsprestanda i ticks

Detta test går ut på att ta reda på hur den simulerade miljön skiljer sig från target när det kommer till hur lång tid det tar att exekvera ett testfall. Resultatet från testet kommer att ge ut hur många ticks som exekveringen tog att genomföra.

Koden för testen som körs i detta kapitel börjar med att skapa en array som innehåller tio slumpmässiga tal från 0 till 99, därefter sorteras denna array från minsta till högsta värdet. Denna biten av kod ligger i en for-loop som ger en möjligheten till att skala upp testen till önskad storlek för att sedan mäta ticksen. I detta test så körs loopen tio gånger där man efter varje loop ökar antalet med tio ytterligare slumpmässiga tal vilket ger en både större och fler arrayer att sortera. Ticksen mäts genom att ta fram det aktuella antalet ticks precis innan arraysen skapas samt precis efter att arraysen har blivit sorterade, sedan tar man skillnaden mellan dessa två värden för att få fram den totala körningen av testet i antalet ticks.

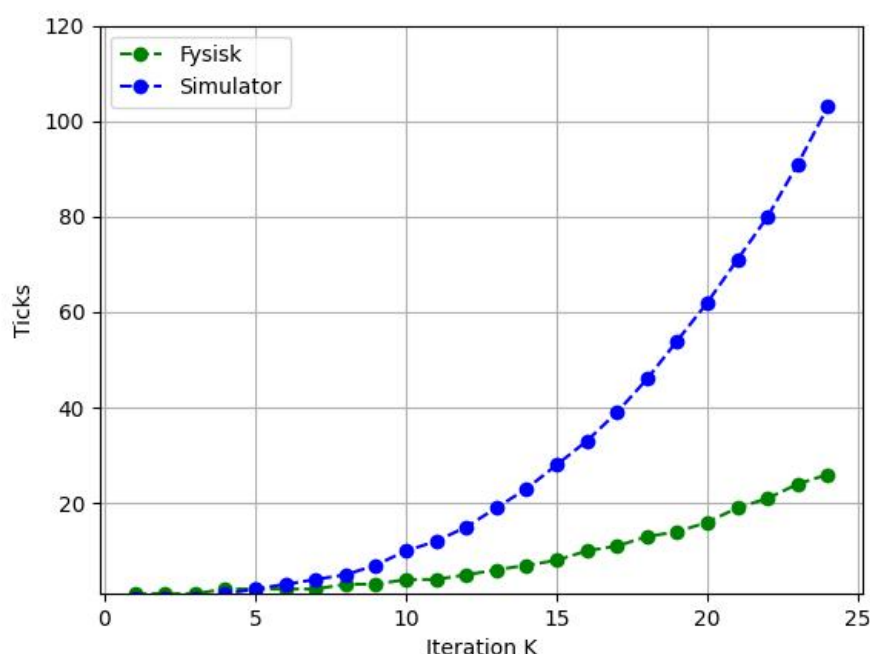
```
1  long starttick = osKernelGetTickCount();
2  printf("start tick: %ld \n", starttick);
3  int sizeArray = 10;
4
5  for(int k = 0; k < 25; k++) {
6      int randomArray[sizeArray];
7      for(int i = 0; i < sizeArray; i++) {
8          randomArray[i] = rand()%100;
9      }
10     for (int i = 0; i < sizeArray; ++i){
11         for (int j = i + 1; j < sizeArray; ++j){
12             if (randomArray[i] > randomArray[j]){
13                 int a = randomArray[i];
14                 randomArray[i] = randomArray[j];
15                 randomArray[j] = a;
16             }
17         }
18     }
19
20     sizeArray = sizeArray + 10;
21 }
22
23 long stoptick = osKernelGetTickCount();
24 printf("stop tick: %ld \n", stoptick);
```

```

25
26     long runtick = stoptick - starttick;
27     printf("runtime tick: %ld \n" , runtick);
28 }
29 }

```

Exekvering av koden genomfördes vid 25 iterationer där varje iteration k ökade storleken på arrayen som sorteras. I grafen som kan avläsas i Figur 4.1 representeras den blåa linjen den simulerade mikroprocessorn och den gröna linjen representerar det fysiska kortet. Ett lägre värde på y axeln är något som man vill eftersträva då det representerar en snabbare exekvering.



Figur 4.1: Jämförelse mellan det fysiska kortet och det simulerade kortet i hur många ticks det tagit att genomföra testet

Vid analys av datan under Figur 4.1 syns det att de första iterationerna av testet går ungefär lika snabbt. Där den blåa grafen sedan ökar med en större takt än den gröna grafen. När testet har körts 25 gånger får man ut att vid det största k värdet 25 har simulatören ticks på 103 och max värde på det fysiska kortet är 26 ticks.

Testet visar att med den prestandan med nuvarande VHT modell är det möjligt att utveckla mot en simulerad processor med minimala eller inga ändringar i koden. Prestandaskillnaderna som uppstår under testet beror med stor sannolikhet på att modellen som användes under testet hade en klockfrekvens som är ca 3.8 ggr långsammare.

5

Diskussion

Som beskrevs i rapporten är simuleringsverktyget som användes under arbetet fortfarande inte helt utvecklat och fortfarande under beta. Detta har medfört att mycket tid gick åt för att undersöka vilka funktioner som fanns tillgängliga för Cortex-M4 processorn under projektet.

När FreeRTOS integrerades mot simulatören och den fysiska hårdvaran uppkom det en del problem att importera paketet mot både kortet och simulatören. Även fast det existerade portning mot FreeRTOS i Keil studio så fungerade det inte direkt och krävde en del tid att lösa ut de portningsfel som uppstod.

Som fortsättning på arbetet går det att undersöka hur man skulle gå vidare med att implementera egna funktioner för att få bättre kommunikation ut och in från den simulerade processorn. Exempel på detta hade varit att försöka implementera kommunikation genom UART eller CAN från servern mot simulatören för att förbättra typen av utveckling som går att köra mot simulatören och inte kräver att man har tillgång till hårdvaran.

Med fortsatt arbete hade man kunnat implementera GitHub Actions arbetsflödet vilket är ett sätt för att automatisera tester under tiden man mjukvaruutvecklar. Med hjälp av Github Actions hade man kunnat sätta upp arbetsflödet så att varje gång ny kod pushas till ens repository så kör simulatören testet automatiskt och skickar tillbaka resultat. På detta vis behöver man inte manuellt köra varje kommando som krävs för att genomföra testet varje gång man har uppdaterat koden.

6

Slutsats

6.1 Simulatore och target

Vid jämförelse av prestandan mellan den fysiska och simulerade mikroprocessorn visar testet att det fysiska kortet klarar av att köra testet på ett mindre antal ticks. Den stora skillnaden i prestanda som man kan se i Figur 4.1 när iterationerna ökar går till stor del att förklara genom att det fysiska mikroprocessorn har en klocka som är ca 4 ggr så snabb som den simulerade.

Eftersom testet använde sig av en annan modell blev resultatet mer av en riktlinje i hur prestanda kan skilja sig mellan det fysiska mikroprocessorn och den simulerade. Resultatet speglar någorlunda att om man får upp klockfrekvenen på simulatore så kan man få ut ett likvärdigt resultat när det kommer till utveckling på molnet.

Även fast simulatore är långsammare räknat i ticks så körs det snabbare i användarens tid. Detta gör att utvecklings-effektiviteten kan öka då man kan köra genom programmet på kortare tid än på det fysiska kortet.

Det kom fram under arbetets gång att funktioner som beskrevs i material för simulatore inte fungerade som de skulle eller inte var implementerade för alla modeller som fanns att välja mellan. Detta kan ha att göra med att simulatore fortfarande är relativt ny och fortfarande i beta, vilket gör det förstäeligt att alla funktioner inte är implementerade än.

Frågeställningen i början av projektet beskrev att minnesanvändning skulle undersökas. När simulatore implementerades existerade det underlag på hemsidan för VHT där det fanns en möjlighet att på simulatore köra en debugger för att få ut information från processorn. Detta fungerade emellertid inte och därmed blev resultatet ett ickeresultat.

Hantering utav latens undersöktes inte när simulatore testades, eftersom ingen kontakt kunde uppnås med simulatore medan den körde och därmed inte få ut mätningar på hur processorn skulle reagerat om den fick en insignal.

6.2 Proof of concept

För stunden så är detta ett proof of concept där man kan utveckla mot en simulerad miljö som ska motsvara en fysisk miljö. Med lite mer utveckling anser vi att ett liknande upplägg i framtiden kan bidra till snabbare utveckling av inbyggda system, men att det fortfarande har en bit att gå när det kommer till att implementera I/O och snabbare processormodeller som motsvarar de som man kan få tag på. Det teoretiska bidraget till nuvarande existerande kunskaper om ämnet kommer att vara väl användbara i framtiden.

Litteratur

- [1] ARM, *Virtual Hardware – Software Development Without Hardware*, 2022. URL: <https://www.arm.com/products/development-tools/simulation/virtual-hardware> (hämtad 2022-06-07).
- [2] ARM, *Overview: Introduction*, 2022. URL: <https://arm-software.github.io/AVH/main/overview/html/index.html> (hämtad 2022-06-01).
- [3] NXP, *FRDM-K64F Platform/Freedom Development Board/Kinetis MCUs / NXP Semiconductors*. URL: <https://www.nxp.com/design/development-boards/freedom-development-boards/mcu-boards/freedom-development-platform-for-kinetis-k64-k63-and-k24-mcus:FRDM-K64F> (hämtad 2022-04-05).
- [4] H. Anthony. (2014). “OpenSDAv2 - NXP Community”, URL: <https://community.nxp.com/t5/Kinetis-Microcontrollers/OpenSDAv2/tap/1121569> (hämtad 2022-06-07).
- [5] J. Yiu, *The Definitive Guide to ARM CORTEX-M3 and CORTEX-M4 Processors (Third Edition)*, Third Edition. Oxford: Newnes, 2014. URL: <https://www.sciencedirect.com/science/article/pii/B9780124080829000014> (hämtad 2022-06-06).
- [6] H. P B, S. R. Anireddy, J. F T och V. R, “Introduction to ARM processors & its types and Overview to Cortex M series with deep explanation of each of the processors in this Family”, i *2022 International Conference on Computer Communication and Informatics (ICCCI)*, 2022, s. 1–8. URL: <https://ieeexplore.ieee.org/document/9740768/> (hämtad 2022-06-06).
- [7] FreeRTOS, *RTOS - Free professionally developed and robust real time operating system for small embedded systems development*. URL: <https://www.freertos.org/RTOS.html> (hämtad 2022-06-07).
- [8] FreeRTOS, *FreeRTOS - Market leading RTOS (Real Time Operating System) for embedded systems with Internet of Things extensions*. URL: <https://www.freertos.org/> (hämtad 2022-04-05).
- [9] IBM. (2022). “Telnet Protocol - IBM Documentation”, URL: <https://www.ibm.com/docs/en/aix/7.1?topic=protocols-telnet-protocol> (hämtad 2022-06-08).
- [10] Keil, *I/O Retargeting: I/O Retargeting*. URL: <https://www.keil.com/pack/doc/compiler/RetargetIO/html/index.html> (hämtad 2022-06-08).

- [11] Keil. (2022). “Systick Timer (SYSTICK)”, URL: https://www.keil.com/pack/doc/cmsis/Core/html/group__SysTick__gr.html (hämtad 2022-06-08).
- [12] ARM. (2022). “Cortex-M55 coremark”, URL: <https://developer.arm.com/Processors/Cortex-M55> (hämtad 2022-05-30).

A

Kod för att grafiskt visa värdena från bilagorna B-C

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3 from numpy import genfromtxt, maximum
4
5 Target = genfromtxt('Ticks_target.txt', delimiter=",",
6                     skip_header = 1)
7 Simulator = genfromtxt('Ticks_simulator.txt', delimiter=",",
8                       skip_header = 1)
9
10 time = list(range(1,25))
11
12 plt.ylim((0,110))
13
14 plt.plot(time , Target , "--bo",label= "Fysisk" , color = '
15           green' )
16 plt.plot(time , Simulator , "--bo",label= "Simulator" ,
17           color ='blue' )
18 plt.legend()
19 plt.xlabel('Iteration K')
20 plt.ylabel('Ticks')
21 plt.grid(True)
22 plt.show()
```


B

Ticks per iteration på simulatorn

Iteration	Ticks
1	0
2	0
3	0
4	0
5	1
6	2
7	3
8	4
9	5
10	7
11	10
12	12
13	15
14	19
15	23
16	28
17	33
18	39
19	46
20	54
21	62
22	71
23	80
24	91
25	103

C

Ticks per iteration på target

Iteration	Ticks
1	1
2	1
3	1
4	1
5	2
6	2
7	2
8	2
9	3
10	3
11	4
12	4
13	5
14	6
15	7
16	8
17	10
18	11
19	13
20	14
21	16
22	19
23	21
24	24
25	26

D

Main program testkod

```
1  /*
2  * Copyright (c) 2013 - 2015, Freescale Semiconductor, Inc.
3  * Copyright 2016-2017 NXP
4  * All rights reserved.
5  *
6  * SPDX-License-Identifier: BSD-3-Clause
7  */
8
9  #include "fsl_device_registers.h"
10 #include "fsl_debug_console.h"
11 #include "pin_mux.h"
12 #include "clock_config.h"
13 #include "board.h"
14 #include "cmsis_os.h"
15 #include "cmsis_os2.h"
16 #include "MK64F12.h"
17 #include <stdio.h>
18 #include "RTE_Components.h"
19 #include CMSIS_device_header
20
21 osThreadId_t tid_phaseA;          /* Thread id of
22     thread: phase_a              */
23
24
25 osThreadId_t tid_clock;          /* Thread id of
26     thread: clock                */
27
28
29
30
31 struct phases_t {
32     int_fast8_t phaseA;
33 } g_phases;
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2580
2581
2582
2583
2584
2585
2586
2587
2588
2589
2590
2591
2592
2593
2594
2595
2596
2597
2598
2599
2600
2601
2602
2603
2604
2605
2606
2607
2608
2609
2610
2611
2612
2613
2614
2615
2616
2617
2618
2619
2620
2621
2622
2623
2624
2625
2626
2627
2628
2629
2630
2631
2632
2633
2634
2635
2636

```

```
33 void signal_func (osThreadId_t tid) {
34     osThreadFlagsSet(tid_clock, 0x0100);          /* set signal to
        clock thread */
35     osDelay(500);                                  /* delay 500ms
        */
36     osThreadFlagsSet(tid_clock, 0x0100);          /* set signal to
        clock thread */
37     osDelay(500);                                  /* delay 500ms
        */
38     osThreadFlagsSet(tid, 0x0001);                /* set signal to
        thread 'thread' */
39     osDelay(500);                                  /* delay 500ms
        */
40 }
41
42
43 /*


---


44 *          Thread 1 'phaseA ': Phase A output
45 *


---


        */
46 void phaseA (void *argument) {
47     for (int i = 0 ; i<1 ; i++) {
48         osThreadFlagsWait(0x0001, osFlagsWaitAny ,osWaitForever)
            ; /* wait for an event flag 0x0001 */
49         g_phases.phaseA = 1;
50         signal_func(tid_phaseA);
            /* call common
            signal function */
51         g_phases.phaseA = 0;
52
53
54         long starttick = osKernelGetTickCount();
55         PRINTF("start tick: %d \n", starttick);
56         int sizeArray = 10;
57
58
59
60         for(int k = 0; k < 10 ; k++) {
61             int randomArray[sizeArray];
62             for(int i = 0; i < sizeArray; i++) {
63                 randomArray[i] = rand()%100;
64
65             }
66
```

```

67
68
69     for (int i = 0; i < sizeArray; ++i){
70         for (int j = i + 1; j < sizeArray; ++j){
71             if (randomArray[i] > randomArray[j]){
72                 int a = randomArray[i];
73                 randomArray[i] = randomArray[j];
74                 randomArray[j] = a;
75             }
76         }
77     }
78
79
80     sizeArray = sizeArray + 10;
81 }
82
83
84 long stoptick = osKernelGetTickCount();
85 PRINTF("stop tick: %d \n", stoptick);
86
87 long runtick = stoptick - starttick;
88 PRINTF("runtime tick: %d \n" , runtick);
89
90 }
91 }
92
93
94
95
96
97
98
99
100 /*
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2580
2581
2582
2583
2584
2585
2586
2587
2588
2589
2590
2591
2592
2593
2594
2595
2596
2597
2598
2599
2600
2601
2602
2603
2604
2605
2606
2607
2608
2609
2610
2611
2612
2613
2614
2615
2616
2617
2618
2619
2620
2621
2622
2623
2624
2625
2626
2627
2628
2629
2630
2631
2632
2633
2634
2635
2636
2637
2638
2639
2640
2641
2642
2643
2644
2645
2646
2647
2648
2649
2650
2651
2652
2653
2654
2655
2656
2657
2658
2659
2660
2661
2662
2663
2664
2665
2666
2667
2668
2669
2670
2671
2672
2673
2674
2675
2676
2677
2678
2679
2680
2681
2682
2683
2684
2685
2686
2687
2688
2689
2690
2691
2692
26
```

```
107 | }
108 | }
109 |
110 | /*
111 |      Main: Initialize and start the application
112 | */
113 | void app_main (void *argument) {
114 |     tid_phaseA = osThreadNew(phaseA, NULL, NULL);
115 |
116 |
117 |     tid_clock = osThreadNew(clock, NULL, NULL);
118 |
119 |     osThreadFlagsSet(tid_phaseA, 0x0001); /* set signal to
120 |         phaseA thread */
121 |
122 |     osDelay(osWaitForever);
123 |     while(1);
124 | }
125 | /*
126 |      Main: Initialize and start the RTOS2 Kernel
127 | */
128 | int main (void) {
129 |
130 |     //char ch;
131 |
132 |     /* Init board hardware. */
133 |     osKernelInitialize();
134 |     BOARD_InitBootPins();
135 |     BOARD_InitBootClocks();
136 |     BOARD_InitDebugConsole();
137 |     SystemCoreClockUpdate();
138 |     PRINTF("new run \n");
139 |
140 |
141 |     // System Initialization
142 |
143 |
144 |         // Initialize CMSIS-RTOS
```

```
145  osThreadNew(app_main, NULL, NULL);    // Create
      application main thread
146  if (osKernelGetState() == osKernelReady) {
147      osKernelStart();                  // Start thread
      execution
148  }
149
150  while(1);
151 }
```



CHALMERS