TMT 2013:25

Ethernet POWERLINK drivrutin för Proview

SIXTEN ISAKSSON
JACK ENGSTRÖM HOFF

Examensarbete inom
ELEKTROTEKNIK
Inriktning
Högskoleingenjör, 15 hp
Södertälje, Sverige 2013

Ethernet POWERLINK drivrutin för Proview

av

SIXTEN ISAKSSON
JACK ENGSTRÖM HOFF

Examensarbete TMT 2013:25
KTH Industriell teknik och management
Tillämpad maskinteknik
Mariekällgatan 3, 151 81 Södertälje

KTH Industriell teknik
och management

Examensarbete TMT 2013:25

Ethernet POWERLINK drivrutin för Proview

Sixten Isaksson
Jack Engström Hoff

Godkänt	Examinator KTH	Handledare KTH
2013-06-20	Christer Albinsson	Lars Johansson
	Uppdragsgivare	Företagskontakt/handledare
	SSAB	Daniel Claeson

Sammanfattning

För att ytterligare stärka sina positioner och vidareutveckla sitt egentillverkade styrsystem Proview önskade SSAB att implementera databussen Ethernet Powerlink. Powerlink kan enkelt beskrivas som CANopen över Ethernet. Powerlink har öppen källkod, realtidsprestanda och körs över Ethernets snabba och billiga hårdvarugränssnitt vilket är bidragande faktorer till att dess popularitet växer - speciellt i Asien. Målet med detta arbete var att implementera ett gränssnitt mellan den öppna Ethernet Powerlink stacken och Proview. Då även Proview har öppen källkod samt att dess utvecklingsteam uppmuntrar användare till att utveckla drivrutiner för nya I/O-enheter fanns redan mycket förarbete gjort som underlättade implementeringen. Detta tillsammans med regelbunden kontakt med Proviews utvecklare och en grundlig litteraturgenomgång av Powerlink var nyckeln till ett lyckat genomförande. Examensarbetet dokumenterar genomförandet av gränssnittsimplementeringen mellan Powerlinkstacken och Proview samt hur en generisk slav kan skapas och fungera tillfredsställande i ett Powerlinknätverk. Tester för att pröva stabiliteten utfördes även i större Powerlinknätverk med goda resultat. För att förbättra prestandan rekommenderar författarna att Powerlink i framtiden implementeras på så sätt att det kan laddas in i operativsystemets kärnutrymme. För att få Powerlink enhetlig med resterande databussar i Proview och göra det mer användarvänligt bör en egen nodkonfigurator utvecklas och låta den ingå i Proview.

Nyckelord

openPOWERLINK, Proview, openCAN, openCONFIGURATOR, Ethernet

KTH Industrial Engineering and Management

Bachelor of Science Thesis TMT 2013:25

Ethernet POWERLINK driver for Proview

Sixten Isaksson
Jack Engström Hoff

Approved	Examiner KTH	Supervisor KTH
2013-06-20	Christer Albinsson	Lars Johansson
	Commissioner	Contact person at company
	SSAB	Daniel Claeson

Abstract

A desire by SSAB to further strengthen their positions and add to the development of SSAB:s control system Proview was to implement the Ethernet Powerlink protocol in Proview. Powerlink can in plain terms be described as openCAN over Ethernet. Powerlink is an open source real-time protocol that runs over Ethernets fast and inexpensive hardware interface. The aim of this project was to implement an interface between the Ethernet Powerlink stack and Proview. Since Proview is an open source system and its development team encourages users to develop drivers for the new I/O devices much of the groundwork was already done which facilitated implementation. This preliminary work was crucial, together with regular contact with the developers of Proview and a thorough literature review of Powerlink, for a successful implementation. This thesis documents the implementation of the interface between the Powerlink stack and Proview. It also covers how a generic slave can be created, configured and function satisfactorily in a Powerlink network. Tests to examine the stability was also performed in larger Powerlink networks with good results. To increase performance and precision of Powerlink the authors recommend that Powerlink, in the future, will be implemented as a kernelmodule. The development of a configurator for the nodes that is included in Proview is also recommended to get the Powerlink protocol consistent with the rest of the I/O protocols in Proview and to make it more user friendly.

Key-words

openPOWERLINK, Proview, openCAN, openCONFIGURATOR, Ethernet

Contents

Chapter 1

Introduktion

SSAB använder idag styrsystemet Proview vilket också har skapats och utvecklats av SSAB. Systemet har gränssnitt till flera olika kommunikationsbussar bland annat Profibus, Modbus och UDP. Ett gränssnitt till Ethernet Powerlink saknas dock i nuläget. Ethernet Powerlink är ett Ethernet baserat realtidsprotokoll med helt öppen källkod vars popularitet växer[17]. Powerlink är ett kommunikationsprotokoll som enkelt skulle kunna beskrivas som CANopen över Ethernet. Detta gör att Powerlink kan förväntas inneha samma funktionssäkerhet som CAN-protokollet tillsammans med hastigheten hos Ethernet[21]. Ethernet-baserad kommunikation har också visat sig kunna vara mycket kostnadseffektiv. Bland annat eftersom infrastrukturen för kommunikation ofta redan finns tillgänglig[18] samt att hårdvarugränssnitten är både billiga och vanliga i de flesta datoriserade utrustningar [19].

1.1 Syfte

Kina är idag ett av de enskilt största producenterna och konsumenterna av stål[6], vilket gör att intresset är stort för SSAB att stärka sina positioner där. SSAB Swedish Steel (China) Co., Ltd är ett dotterbolag i SSAB som grundades 2006, och nyligen öppnade även SSAB ett nytt utveckling- och forskningscenter i Kunshan, Kina[7]. Syftet med detta är att ta nya marknadsandelar i Kina och Asien samt att kunna erbjuda sin spetskompetens[8]. En förutsättning för att kunna göra detta möjligt är att vara i takt med den tekniska utvecklingen. Powerlink utsågs nyligen till att vara en standard i Kina av Standardization Administration of China (SAC) [20]. Med detta som bakgrund är det ett önskemål från SSAB därför att kunna ha ett gränssnitt mellan Proview och en Ethernet-baserad realtidsbuss som Ethernet Powerlink.

1.2 Mål

Målet med detta arbete är att implementera ett gränssnitt mellan den öppna Ethernet Powerlink stacken och Proview, på så sätt att minst en generisk slav kan skapas och fungera tillfredsställande i ett Powerlinknätverk.

1.3 Avgränsning

Någon djupare analys av realtidsprestationen hos Ethernet Powerlink kommer ej att genomföras. En egen tillverkad nodkonfigurator kommer ej att ingå i projektet.

1.4 Frågeställning

För att lyckas med gränssnittsimplementationen måste följande frågor besvaras.

- Hur konfigureras Powerlink stacken?
- Hur initieras och startas Powerlink stacken?
- Hur konfigureras noderna?
- Hur integreras Powerlink stacken i Proview?

1.5 Metod

Inledningsvis kommer en litteraturgenomgång genomföras. Materialet för detta består framför allt av den dokumentation som tillhandahålls av Ethernet POWERLINK Standardization Group (EPSG). När tillräckligt med information och kunskap införskaffats påbörjas själva genomförandet. Till en början görs försök med kommunikation mellan persondatorer i ett mindre nätverk. Detta följs av kommunikationstester med utvecklings-kitet Spartan-6 FPGA LX9 från Xilinx. Slutligen kommer tester utföras på frekvensomriktare utrustade med Powerlink gränssnitt från ABB. För att sedan kunna integrera Powerlink i Proview kommer redan befintliga implementationer av databussgränssnitt att studeras. Stöd kommer finnas tillgängligt i form av kontinuerlig kontakt med utvecklare av Proview. Debbugning av applikationer och nätverk kommer ske med hjälp av GNU Project Debugger (GDB) respektive Wireshark.

Chapter 2

Bakgrund

Detta kapitel kommer att ge läsaren nödvändig bakgrundsfakta av SSAB:s styrsystem Proview, CANopen och slutligen Ethernet Powerlink.

2.1 Proview

Proview är förmodligen det första styrsystem i världen som har öppen källkod och är helt gratis[9]. Systemet är utvecklat av SSAB och är mycket kraftfullt och används för bland annat styrning, reglering, datainsamling, kommunikation, och övervakning. Systemet är mycket modernt och har god prestanda och kan tävla med de stora systemtillverkarna som till exempel ABB och Siemens. Proview är ett så kallat soft-PLC vilket innebär att man installerar mjukvara i en vanlig persondator så att den kan agera PLC. Begränsningarna består därför till största del av operativsystemets kapacitet och hårdvarans prestanda. Detta gör att det inte finns några gränser för hur många I/O enheter systemet kan kommunicera med, eller hur många PID-loopar och PLC program som kan köras samtidigt. Systemet körs i Linuxmiljö[13]. In och utsignaler kommer in i systemet via någon form av buss och något kommunikations protokoll. Några exempel är Profibus, Modbus, UDP, TCP, PSS9000, Profinet och det finns många fler. Konfigureringen av systemet görs helt grafiskt, Proview är ett så kallat distribuerat system vilket gör att styrsystemet kan bestå av flera datorer anslutna i ett nätverk. Ett typiskt system består ofta av en styrsystemsdator och ett antal operatörsstationer. Det är även lätt att konfigurera operatörsstationer med så kallade HMI-system som kan övervaka ett stort antal styrsystem från flera anläggningar. Programmering kan göras helt objektorienterat. Programmering på traditionellt vis med enkla funktionsblock är möjligt. Komplexa objekt och funktioner kan lätt skapas. Systemet har stöd för flera högnivåspråk som till exempel C, C++, Java och FORTRAN[9].

2.1.1 I/O konfigurering

Eftersom Proview har öppen källkod, körs i Linux och har stöd för högnivåspråk är det möjligt att implementera nya typer av I/O-enheter. Nya I/O-system implementeras genom att skapa klasser i det aktuella projektet eller genom att skapa klasserna direkt i Proviews bassystem [11]. I/O-enheterna kan sedan konfigureras genom att skapa instanser av klasser som finns definierade i bassystemet eller i det aktuella projektets klassvolym. Objekten delas upp i två olika träd, ett som representerar en logisk modell av det fysiska systemet (anläggningshierarkin) och ett som representerar noder (nodhierarkin) [10]. Anläggningshierarkin beskriver de olika funktionerna och processerna som finns i anläggningen. Under denna kategori lägger man PLC kod och instanser av olika objekt som används av PLC programmen. En del objekt (signalobjekt) kan kopplas till objekt i den andra hierarkin (kanalobjekt), exempelvis så kopplar man ett digitalt ingångsobjekt, di-objekt, i anläggningshierarkin till ett digital ingångskanal, ChanDi-objekt, i nodhierarkin [11]. Detta för att ge PLC programmen tillgång till processdata. Processdata kan vara både in- och utdata. Indata kommer från olika typer av sensorer, givare eller andra datorsystem. Utdata är data som ska ut från systemet för att till exempel styra motorer och ställdon. Eftersom givare, sensorer och aktuatorer kan lämna/motta många olika typer av data så finns det flera signalobjekt att välja mellan för anläggningshierarkin, några exempel är digitala in/utgångar samt analoga in/utgångar det finns motsvarande kanalobjekt att lägga i nodhierarkin [12]. Nodhierarkin återspeglar hårdvaran med I/O-system och systemprocesser. I/O-systemen är i sin tur uppbyggda som hierarkier av klasser med fyra nivåer: agent-, rack-, kort- och kanalobjekt. I ett distribuerat I/O-system kan agentnivån ses som en masterenhet, racknivåerna som slavar, kortnivåerna I/O-moduler och slutligen kanalnivåerna olika I/O-kanaler. Dessa objekt konfigureras med parametrar som behövs för de olika I/O-systemen. Exempelvis kan ett Powerlink master objekt konfigureras med det nätverksgränssnitt som stacken ska använda sig av.

2.2 Ethernet Powerlink

2.2.1 CANopen

Controller Area Network (CAN) utvecklades till en början framför allt för användning inom bilindustrin. Där blev den oerhört populär och är idag en av de mest överlägsna fältbussarna i avseende på låga kostnader, förmågan att fungera i påfrestande miljöer, hög tillförlitlighet när det gäller realtidsegenskaper, utmärkt feldetektering och hantering samt lättanvänt[16]. På grund av dess framgångar inom bilindustrin har det även växt fram en stor marknad med billiga mikrokontrollers försedda med CAN-gränssnitt[14]. CAN-protokollet har på så sätt flyttats utanför bilindustrin och har i och med det skapat ett behov av ett öppet och standardiserad högnivåprotokoll som erbjuder samma pålitliga dataöverföring som CAN. Ett flertal högnivå CAN protokoll har således dykt upp, där CANopen är en av de absolut mest

populära[14]. I Open Systems Interconnection (OSI) modellen tillhör CANopen det sjunde lagret, det vill säga applikationsskiktet. Målet med CANopen är att möjliggöra kommunikation och samverkan mellan olika typer av enheter genom ett koncept som kallas profilering[16]. En analogi skulle kunna vara att betrakta protokollet som ett språk exempelvis svenska. Då utgör det fysiska skiktet, exempelvis CAN eller Ethernet, pennan och pappret. Datalinkskiktet kan ses som det svenska alfabetet. Applikations skiktet motsvarar en svensk ord- och grammatikbok medan CANopens enhetsprofil tillhandahåller svenska fraser. Enhetsprofilen utgör kärnan i CANopen och det är denna som garanterar samverkan mellan olika typer av CANopen enheter. Den mest fundamentala delen av enhetsprofilen är dess objektlexikon. Denna innehåller dataobjekt, kommunikationsobjekt och olika kommandon. CANopen ger full access till enheters objektlexikon så att användaren både kan läsa och skriva i detta. Objektlexikonet består 65536 adressplatser som vardera kan hålla 256 parametrar. Adressplatserna är sedan uppdelade i specifika intervall som följer den standard som CAN in Automatio (CiA) har definierat[16]. CiA är en internationell organisation bestående av användare och tillverkare som tillsammans främjar och utvecklar de översta lagren i CAN-baserade protokoll. Kommunikationen kan delas upp i fyra olika typer. Service data object (SDO) och process data object (PDO) används för att överföra data. SDO meddelande kan användas för att överföra data samt konfigurera noderna men har större overhead än PDO meddelanden. PDO meddelandena är ett effektivare sätt att överföra data men dessa meddelanden måste bli definierade under initieringen. Network management (NMT) meddelandena har högst prioritet och används för att koordinera enheterna i nätverket samt övervaka statusen hos noderna. Utöver dessa meddelande finns även meddelanden med fördefinierade format som används för felrapportering, synkronisering och timing.

2.2.2 Ethernet

Ethernet utvecklades från början av Xerox PARC under 70-talet [2]. Den kommersiella introduktionen och standardiseringen kom under 80-talet som ISO standard IEEE 802.3. Ehternet är idag helt dominerade inom höghastighetskommunikation[3] och hårdvarugränssnittet finns i de flesta datoriserade utrustningarna. I begynnelsen av Ethernet låg överföringshastigheten på 10 Mbit/s men är idag uppe i 100 Gbit/s. I OSI modellen befinner sig Ethernet i de två lägsta lagren - det fysiska skiktet och datalänkskiktet[1]. På grund av detta kan det fysiska mediet variera. Tidigare var olika typer av koaxialkabel vanligast men idag är partvinnad kabel och fiberoptik helt klart mest förekommande. Fördelen med de partvinnade kablar är att de är relativt billiga i jämförelse med fiberoptisk kabel. Fiberoptisk kabel är dock helt överlägset när det gäller möjlighet till långa avstånd mellan noderna[5]. För att överföra datapaketen över Ethernet inkapslas dessa i Ethernetramar. I huvudet på ramen finns 32-bitars MAC adresser till källan och destinationen. Ramen avslutas med en 32-bitars cyclic redundancy check (CRC) för att upptäcka eventuellt korrupt data[4]. CSMA/CD tillämpas inom Ethernet för att upptäcka och hantera

kollisioner[4]. Om en kollision upptäcks väntar noden en slumpartad tid innan den åter försöker sända igen. Detta gör att Ethernet kommunikation inte skulle lämpa sig för realtidsapplikationer då dessa måste vara deterministiska. Moderna Ethernet baserade nät använder sig dock inte av CSMA/CD längre[4]. Trots att dessa nät är helt kollisionsfria på grund av introduktionen av full-duplex funktioner och switchar är dessa fortfarande bakåtkompatibla och har således stöd för CSMA/CD.

2.2.3 Ethernet Powerlink

Eftersom CANopen befinner sig i applikationsskiktet är möjligt att använda andra fysiska gränssnitt än CAN. Detta anammade Bernecker & Rainer (B&R) när de utvecklade Ethernet Powerlink som idag administreras och förvaltas av EPL Standardization Group (EPSG) [22]. Ett av huvudmålen med Powerlink var att vara plattformsoberoende vilket gör att nuvarande implementationer kan köras på bland annat Linux, Windows och VxWorks[25]. Powerlink är baserat på Ethernet standarden IEEE 802.3 tillsammans med CANopen[23]. Anslutningen mellan nätverksenheterna kan ske med nätverkshubbar eller switchar, nätverkshubbar är dock att föredra då dessa har mindre fördröjningar och ett mer förutsägbart beteende[25]. Powerlink protokollet är baserat på principerna av ett master-slav system på ett delat Ethernet segment som kallas Slot Communication Network Management (SCNM)[22]. I Powerlink protokollet finns två typer av noder definierade - Managing Node (MN) och Controlled Node (CN) [22]. MN-noden är unik och representerar mastern i en master-slav relation. CN-noderna utgör således slavarna och i ett och samma Powerlink nätverk kan 240 stycken slavar implementeras[17]. Masternoden säkerställer åtkomst för realtidskritiska cykliska data för att sedan låta icke realtidskritisk kommunikation passera endast i tidsluckor avsedda för detta ändamål. Powerlink cykeln startar genom att MN-noden skickar ut en Start of Cycle (SoC) ram som ett multicast meddelande. Detta meddelande innehåller inga data utan är endast till för att synkronisera alla Powerlink enheter[25]. Direkt efter det att SoC meddelandet har skickats börjar MN-noden skicka Poll Re-quest (PReq) meddelande till samtliga noder i nätverket. CN noderna är enbart tillåtna att sända data när de har fått en direkt begäran genom ett PReq meddelande. Vardera CN nod svarar med ett Poll Response (PRes) meddelande som MN noden väntar på. Detta gör att kollisioner undviks trots användandet av nätverkshubbar[24]. Då PRes ramen skickas som ett multicast meddelande tillåter detta direkt kommunikation mellan CN noderna utan inverkan av MN noden[25]. Detta minskar tiden avsevärt för dataöverföring mellan noderna. Efter den synkrona fasen börjar den asynkrona fasen. Denna fas är dedikerad till all kommunikation som inte är realtidskritisk exempelvis diagnostiska data, ARP eller TCP/IP[23]. CN noderna kan signalera i PRes meddelande till MN noden att denna vill sända asynkrona data. MN noden avgör sedan vilka enheter som får sända asynkrona data och meddelar detta i Start of Asynchronous (SoA) ramen [25]. SoA ramen markerar också början på den asynkrona fasen till samtliga enheter på nätverket. Enligt protokollet tillåts dock endast en nod ett asynkront meddelande under vardera cykel. Enligt

Powerlink standarden finns två typer av asynkrona meddelanden: Powerlink ASnd (Asynchronous Send) och Legacy Ethernet meddelanden. Efter den asynkrona fasen påbörjas den inaktiva perioden då MN noden inväntar nästa cykelstart. Denna schemaläggning garanterar det deterministiska beteende som ett hårt realtidsprotokoll kräver [23]. Då Powerlink kommunikationen är starkt beroende av exakt timing är dess mest kritiska parameter SoC-jitter, det vill säga oönskade tidsavvikelser mellan SoC meddelanden. Storleken på SoC-jittret är främst beroende av operativsystemet och dess nätverksstack[25]. Hög upplösning hos operativsystemets interna klockor är nödvändigt för att minimera jittret. Fördröjningar som skapas av nätverksdrivern då ett paket tas emot och skickas ut påverkar också SoC-jittret. Figur 2.1 nedan visar en typisk Powerlink cykel. Powerlink kan implementeras som

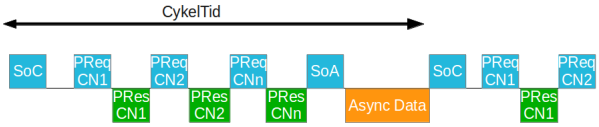


Figure 2.1. Schematisk bild av Powerlink cykeln. Cyekltiden mäts från SoC till Soc.

en del av operativsystemets kärnutrymme eller som en del av användarutrymmet. För maximal prestanda och minimering av jittter rekommenderas att Powerlink körs i kärnutrymmet[25]. Att köra Powerlink i användarutrymmet har dock andra fördelar - bland annat behövs inte specifika Ethernetkontroller, och det kräver mindre underhåll och är lättare att felsöka. Powerlinks kommunikationsprofil är i övrigt nästintill direkt anpassad efter CANopen. PDO-meddelande används för dataöverföring och SDO-meddelanden för att konfigurera noder. PDO-meddelandena utbyts under den synkrona fasen eftersom händelseutlösta PDO skulle påverka de hårda realtidskraven [23]. För att överensstämma med IEEE 802.3 standarden har varje Powerlinkenhet en unik MAC address. Varje Powerlinkenhet tilldelas också ett nod-ID. MN-noden har nod-ID 240 och övriga slavar tilldelas ett nod ID < 240. Om det är nödvändigt för en Powerlinkenhet att använda sig av TCP/IP tilldelas denna en klass C IP adress där host delen motsvarar enhetens nod-ID[23].

Chapter 3

Genomförande

Detta kapitel kommer att behandla gränssnittsimplementationen mellan Powerlink och Proveiw. Specifikt innebär detta en genomgång av konfigurering och initiering av Powerlink stacken samt implementeringen i Proveiw.

3.1 Nodkonfigurering

MN-noden i ett Powerlink nätverk har möjlighet att konfigurera samtliga noder genom SDO meddelanden. För att detta skall vara möjligt krävs dock att MN noden har tillgång till en binärfil med alla instruktioner. I detta projekt användes openCONFIGURATOR, som är ett öppet konfigureringsverktyg tillverkat av Kalycito Infotech[26], för att skapa den binära filen.

3.1.1 openCONFIGURATOR

Enhetsprofilen för vardera nod beskrivs i XML Device Descriptions (XDD) filer. Dessa tillhandahålls av återförsäljaren för produkten eller kan tillverkas manuellt för skräddarsydda applikationer. Standardenhetsprofiler för ett flertal olika enheter finns beskrivna av CiA[16] bland annat för generiska IO moduler, servon och stegmotorer. I openCONFIGURATOR kan en hierarki sättas upp med MN-noden och underliggande CN-noder. Till noderna kan färdiga XDD-filer importeras, men annars används en fördefinierad standard. Under varje nod kan ytterligare index skapas och definieras. Indexen är uppdelad i specifika områden efter vilken typ av parameter indexet innehåller. Tabell 3.1 beskriver hur uppdelningen av index är definierad efter parametertyp i ett objektlexikon.

Intervall	Parameter beskrivning
0001H - 009FH	Definitioner av datatyper
00A0H - 0FFFH	Reserverade för framtida bruk
1000H - 1FFFH	Utrymme för kommunikationsprofil. Bland annat parametrar för timing och identifiering av enhet.
2000H - 5FFFH	Utrymme för produktspecifika parametrar definierade av tillverkaren.
6000H - 9FFH	Utrymme för standardiserad enhetsprofil. Kan användas enligt en CiA enhetsprofil standard.
A000H - FFFFH	Reserverade för framtida bruk

Table 3.1. Indexbeskrivning för objektlexikon

Under dessa adressplatser kan ytterligare 256 subindex skapas och definieras. Varje subindex innehåller sedan i sin tur ett variabelnamn, objekttyp, datatyp samt andra attribut som anger om läsning och skrivning är tillåten. De datatyper som kan användas sträcker sig från boolska variabler till 64 bitars heltal. Objekttyperna kan vara exempelvis variabler, arrayer eller structar. Det räcker dock inte med att definiera parametrar i objektlexikonet för att konfigureringen av noder och dataöverföring skall fungera. De data som skall skickas i PDO-meddelandena måste kartläggas på förhand. Det finns två typer av PDO-meddelanden, Transmit Process Data Objects (TPDO) och Receive Process Data Objects (RPDO), som används för att sända data respektive mottaga data. De processvariabler som skall sändas kartläggs under vardera nods TPDO och måste innehålla index och subindex för variabeln, offset samt storlek. På motsvarande sätt kartläggs de processvariabler som skall mottagas under RPDO. När projektet sedan byggs har openCONFIGURATOR stöd för att automatiskt generera PDO-kartläggningen för MN-noden. När konfiguration byggt projektet skapas en binärfil av filformatet CDC som sedan kan användas vid initieringen av Powerlink stacken. Konfiguration ger inte bara möjlighet att definiera och möjliggöra överföring av processvariabler utan har också stöd för att konfigurera alla andra parametrar som exempelvis olika tidsparametrar, watchdogs och nod-ID.

3.2 Powerlinkstacken

Följande sektion beskriver hur Powerlink stacken initieras, startas och avslutas. Här ges också en beskrivning av hur den asynkrona och synkrona dataöverföringen möjliggörs i applikationen.

3.2.1 Initiering

Initieringen av Powerlink är förhållandevis enkel då Powerlinkbiblioteket tillhandahåller en stor mängd färdigskrivna klasser och funktioner. Första steget i initieringen av Powerlink är att skapa en instans av klassen EplApiInitParam. Denna klass har ett antal attribut och metoder som måste definieras för att initieringen skall fungera. De attribut som är obligatoriska att definiera samt beskrivning av attributet visas i tabell 3.2 nedan.

Parameter	Beskrivning
uiNodeId	Lokalt nod-ID, 1-239
dwIpAddress	Lokal IP-adress
abMacAddress	Lokal MAC adress, om MAC adressen sätts till noll används värdet som finns förvarat i enhetens EEPROM.
pszDevName	Namn på valt nätverkskort
dwCycleLen	Cykeltid i μs, läggs i det lokala objektlexikonet på plats 0x1006
fAsyncOnly	Huruvida noden enbart medverkar i asynkron kommunikation eller både synkron/asynkron.
uiIsochrTxMaxPayload	Maximal datamängd att sända i den synkrona trafiken, läggs i det lokala objektlexikonet på plats 0x1F98/1
uiIsochrRxMaxPayload	Maximal datamängd att mottaga i den synkrona trafiken, läggs i det lokala objektlexikonet på plats 0x1F98/2
dwAsndMaxLatency	Maximal fördröjning hos asynkrona dataramar, läggs i det lokala objektlexikonet på plats 0x1F98/3
dwPresMaxLatency	Maximal fördröjning hos PRes meddelanden, läggs i det lokala objektlexikonet på plats 0x1F98/6
pfnCbEvent	Funktionspekare till applikationens asynkrona callback funktion
pfnCbSync	Funktionspekare till applikationens synkrona callback funktion

Table 3.2. Parameterbeskrivning hos de parametrar som är obligatoriska att definiera

EplApiInitParam innehåller många fler parametrar som kan definieras, bland annat serienummer och enhetstyp. Om parametrar tilldelas det maximala värdet

för datatypen ignoreras parametervärdet och motsvarande förinställda värde i objektlexikonet används istället. Efter det att attributen hos EplApiInitParam har definierats används metoden EplApiInitialize för att initiera stacken. EplApiInitialize tar instansen av EplApiInitParam som argument.

3.2.2 Start

Innan starten av stacken sker måste länkningen av den binära CDC-filen göras, samt viss initiering för att möjliggöra överföring av processdata. Funktionen EplApiSetCdcFilename tar ett argument av datatypen string som i sin tur enbart innehåller en sökväg till CDC-filen. För att kunna överföra processdata används en processbild (Process Image) det vill säga ett minnesblock som kan uppdateras cykliskt. Minnesblocket allokeras genom ett funktionsanrop till EplApiProcessImageAlloc som tar argumentet för storleken på in- och utdata. Ett funktionsanrop till EplApiProcessImageSetup för länkning av processvariabler är också nödvändig innan starten. Slutligen startas stacken genom att anropa funktionen EplApiExecNmtCommand med argumentet kEplNmtEventSwReset. Argumentet kEplNmtEventSwReset är en enum och innebär en mjuk omstart av stacken.

3.2.3 Callback funktioner

Som nämnts tidigare innehåller initieringen referenser till två typer av callbackfunktioner - en för synkron och en för asynkron dataöverföring. Funktionen för asynkron överföring anropas då nätverks hanterarens (NMT) eller noders status förändras, åtkomst ges till objektlexikonet, SDO meddelanden avslutas samt om olika typer av fel på Powerlinkstacken uppstår. NMT statusen i Powerlink representeras av olika enums av typen tEplNmtState. Vissa tillstånd måste hanteras av applikationen eller Powerlink stacken men många kräver inte någon åtgärd alls. I funktionen för synkron dataöverföring kan läsning och skrivning till CN-noder utföras. Genom att kalla på funktionen EplApiProcessImageExchange med adressen för ut- och indata som argumentet kan ett säkert utbyte av processdata utföras.

3.2.4 Avslut

För att avsluta stacken på ett säkert sätt anropas EplApiExecNmtCommand med argumentet kEplNmtEventSwitchOff. Detta argument stänger NMT så att Powerlinkstacken säkert kan avslutas. Innan avslutet är det dock viktigt att frigöra det minnesblocket som tidigare allokerades genom EplApiProcessImageAlloc. För att frigöra minnet anropas funktionen EplApiProcessImageFree. Slutligen anropas funktionen EplApiShutdown för att avsluta och stänga ned Powerlinkstacken.

3.3 Proview IO

För att implementera ett nytt I/O-system i ett Proviewprojekt skapas klasser i dess klassvolym. Om det är ett I/O-system som kan komma att användas i flera system sker implementationen med fördel i Proviews bassystem och kommer då bli tillgängligt för alla som använder Proview. Detta för att skapa en utvecklingsmiljö där återandvändbarhet förespråkas. Powerlink implementationen ingår i bassystemet. Proviews utvecklingsteam uppmuntrar användare till att utveckla interface för nya I/O-enheter, i Proview finns därför redan en mängd färdigskrivna klasser och metoder som underlättar implementeringen. Det finns bland annat basklasser för agent-, rack- och kortobjekten som redan innehåller de vanligaste attributen för respektive objekt. De subklasser som sedan skapas ärver alla de attribut och metoder som basklasserna innehåller. För initiering, läsning, skrivning och avslut registreras metoder för vardera objekt. Powerlink är ett så kallat distribuerat I/O-system, därför representerar agentnivån MN-noden, racknivån CN-noder, kortnivån I/O-moduler och slutligen kanalnivån olika I/O. Denna sektion kommer beskriva vilka klasser som är nödvändiga att skapa, och vad dessa innehåller för att implementationen av Powerlink skall vara möjlig i Proview. Vidare kommer också en förklaring på hur processen körs igång samt hur dataöverföringen möjliggörs.

3.3.1 Initiering

Vid uppstart av ett Proviewsystem så startar flera processer som hanterar olika delar i systemet, ett exempel är rt_errh som tar emot meddelanden från andra processer. Processen rt_powerlink är den process som hanterar Powerlinkstacken. För att möjliggöra initiering och utbyte av data med PLC processen kopplar processen upp sig mot Proviews realtidsdatabas. Om inga Powerlinkmastrar hittas av rt_powerlink så avslutas processen, annars körs initieringen. Agentobjekten har en initieringsfunktion registrerad hos sig som anropas av rt_powerlink. Funktionen heter IoAgentInit, den kommer i sin tur anropa en funktion för initieringen och start av Powerlink stacken på det sätt som är beskrivet i kapitel 3.2. Rack- och kortobjekten behöver således ingen egen initieringsfunktion, då all initiering hanteras av agentobjektet. Under initieringen görs också viss förberedelse för dataöverföringen. Agentobjektet innehåller två pekare för in- och utdataareor. Allokeringen av minne till dessa minnesareor sker dynamiskt. För att beräkna storleken för minnesallokeringen måste först agentobjektets slavhierarki traverseras för att undersöka antalet underliggande kanalobjekt samt dess storlekar. När beräkningen av storleken på in- och utdataareorna görs räknas även offset ut för varje slav, detta för att processen ska veta vilka delar av areorna som tillhör vilken slav. På samma sätt räknas även offset ut för varje kanalobjekt i varje slav. När detta är gjort har en länk skapats mellan Powerlinks processbild och Proviews kanalobjekt. Proview har ett befintligt system för att hantera meddelanden från olika processer, implementationen av Powerlinkstacken är kopplad till detta system för att ge en bra bild av vad som händer i stacken.

3.3.2 Läsning och skrivning

Agentobjektet har en funktion för läsning och en för skrivning av processdata registrerat i sin definition, funktionerna IoAgentRead respektive IoAgentWrite. Powerlink stacken är uppbyggd så att MN-noden sköter all läsning och skrivning, och det är därför inte nödvändigt att implementera rack- och kortobjektens motsvarande metoder. Agentobjektet kopierar istället Powerlinks processbild och distribuerar data till sina underliggande objekt. När funktionen AppCbSync anropas så anropar den i sin tur funktionen EplApiProcessImageExchange för att överföra Powerlinks processbild till de minnesareor som agentobjektets in- och utdatapekare pekar på. Callback-funktionen AppCbSync tillhör rt_powerlink processen och har en egen tråd som går parallellt med tråden som sköter dataöverföringen mellan Proview och Powerlinkstacken. Efter det att initieringen är gjord så startar en sekvens med funktionsanrop, först anropas IoAgentRead för alla Powerlink agentobjekt som då använder sin indata pekare för att datasätta sina underliggande slavobjekts kanalobjekt. I slutet av sekvensen anropas sedan IoAgentWrite som använder kanalobjektens data för att skriva till det minne som agentobjektets utdata pekare pekar på. Dessa data blir då en del av Powerlinks processbild. Denna sekvens upprepas sedan om och om igen med en periodtid som bestäms i agentobjektet. PLC-programmets process kan då fritt använda sig av processdata vid exekveringen av den faktiska PLC-koden.

3.3.3 Avslut

När Proview avslutar IO-systemet anropas metoden IoAgentClose. Denna metod stänger ned Powerlink stacken och frigör det minne som allokerats på det sätt som är beskrivet i sektion 3.2.4.

Chapter 4

Tester och Resultat

4.1 Generiska CN-noder

Ett flertal testfall med stegrande svårighetsgrad konstruerades för att testa systemet. De utformades så att de skulle kunna testa stabiliteten i nätverkskommunikationen under olika situationer. Svårighetsgraden ökade med ökande antal CN-noder, ökande antal parametrar och varierande parametertyper. Slutligen lämnades systemet igång i 72 timmar med ett nätverk bestående av en MN-nod och fyra CN-noder som alla skickade och mottog olika antal och typer av processdata. I Proview skrevs ett PLC program som simulerade processdata att styra ut till de olika CN-noderna. Nätverkskommunikationen visade sig vara stabil och fungerade tillfredsställande utan några omstartande noder eller upphakningar i systemet. Systemet var även stabilt när noder kopplades på och av från bussen. Nätverket beskrivs i figur 4.1.

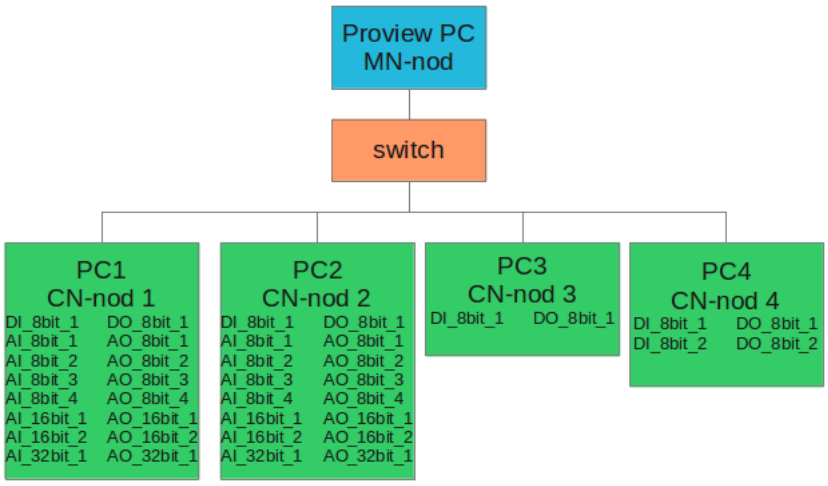


Figure 4.1. Beskrivning av testnätverk samt parametertyper för vardera CN-nod.

Chapter 5

Diskussion och Slutsats

Då den nuvarande implementationen är tillräcklig för att uppfylla de grundkrav som ställdes från SSAB:s sida finns fortfarande flera saker som kan korrigeras och förbättras.

Initieringen och uppstarten av stacken körs nu i en egen process på ett sätt som inte helt följer standarden i Proview. När försök gjordes att följa standarden avslutades applikationen abrupt. Applikation felsöktes då med GDB som visade att problemet härrörde från en deadlock-situation. Det vill säga en situation där två eller flera konkurrerande uppgifter tvingas att samtidigt vänta ut varandra, vilket då får konsekvensen att ingen uppgift utförs. Deadlock-situationen kan ha sitt ursprung i de flertal trådar som skapas under initieringen för att bland annat ta hand om dataöverföring, Ethernettrafik och databufferten. Flera av dessa trådar använder lås för delade resurser i form av semaforer och mutex. Det är vanligt att dessa kan ge upphov till deadlock-situationer. Eftersom det är känt att applikationer med multipla trådar är notoriskt svåra att felsöka[27] är det inte helt säkert att det verkligen rörde sig om ett deadlock. Det kunde snarare vara så att timingen påverkades av felsökningen och på så sätt skapade en deadlock-situation.

Ytterligare ett problem som uppdagades under projektet tog lång tid att lösa. Det visade sig att nätverk med flera slavnoder kan bli instabila under vissa omständigheter. Situationen kunde uppstå efter det att ett nätverk satts upp i en viss nod-ID ordning som sedan kastades om. Detta kunde leda till att en av noderna började omstartas. Av vad som går att utläsa från felmeddelandena i masternoden härrör detta fel från att ett tröskelvärde överskridits på felaktiga PRes meddelanden från den berörda slaven. Detta skulle förklara varför slaven omstartas men ger dock ingen förklaring till varför felet uppstår från början. Flera teorier har föreslagits för att sedan förkastas. Till en början misstänktes att orsaken hade att göra med vilken typ av nätverkskomponent som styrde datatrafiken mellan noderna. Då Powerlinknätverk använder sig av både MAC adresser och nod-ID för att identifiera slavar blev den första tanken att problemet hade sitt urspung i att MAC-addresser sparats i switchens CAM-tabell (Content Addressable Memory). Att byta ut switchen mot en nätverkshubb visade sig dock bara försämra stabiliteten. Nästa

tanke var att problemet orsakades av en konflikt mellan noder som kördes på eller hade konfigurerats med 64- respektive 32-bitars system. Problemet uppstod när tre eller färre slavar användes och alltid då en av noderna som ingick i nätverket kördes på ett 64-bitars operativsystem och denna inte hade högst nod-ID. Felet kunde inte provoceras fram då 64-bitars noden hade högst nod-ID oavsett hur de resterande slavarnas nod-ID kastade om. Detta visade sig dock inte stämma när ytterligare en nod introducerades i nätverket. Då började det misstänkas att problemet hade att göra med överföringsparametrar som hastighet, duplexläge eller flödeskontroll alternativt val av nätverkskort som i sin tur gav upphov till felaktig timing. Det visade sig dock tillslut att det var inställningar på PollResponse-timeout värdet i slavnoderna som var för korta. Detta värde avgör hur länge masternoden väntar på på svar från slavnoden.

För att konfigurera noderna används i nuläget den externa nodkonfiguratorn openCONFIGURATOR. Anledningen till detta är konfiguratorn producerar en binär CDC fil med instruktioner som sedan Powerlinkstacken använder sig av vid initieringen. Att använda en extern programvara innebär tyvärr merarbete i mer än en bemärkelse. Dels eftersom att nodhirarkin först måste skapas i konfiguratorn för att sedan återigen skapas i Proview. I/O kanalerna i openCONFIGURATOR ordnas också automatiskt efter 32bitars anpassning. Detta innebär att det i Proview kan bli nödvändigt att manuellt införa utfyllnadsvariabler mellan kanalobjekten för att överföringen av processdata skall fungera. I openCONFIGURATOR måste mappningen av processdata ske manuellt vilket är tidskrävande och lätt en källa till fel. Att använda en extern konfigurator innebär också att användaren måste lära sig ett nytt grafiskt gränssnitt vilken kan vara både tidskrävande och mödosamt. Det bör därför läggas ned tid och resurser för att utveckla en intern konfigurator med samma möjlighet som i openCONFIGURATOR men som också har automatisk mappning av processvariabler. Eftersom att openCONFIGURATOR har öppen källkod skulle mycket förarbetet redan vara utfört, men trots detta skulle det säkerligen krävas en lika stor arbetsinsatts som detta projekt tog i anspråk.

I detta projekt implementerades Powerlink som en del av operativsystemets användarutrymme. Det var fördelaktigt under arbetsfasen då en hel del tester och felsökning utfördes samt många typer av slavar användes med olika nätverkskort. Powerlink bör dock, i framtiden, implementeras som en del av kärnutrymmet. Detta kräver emellertid att specifika Ethernet kontroller används men ger då i gengäld maximal prestanda, timing och minimering av jitter.

Bibliography

[1] Digital Equipment Corporation, Intel Corporation and Xerox Corporation (1980). *The Ethernet, A Local Area Network. Data Link Layer and Physical Layer Specifications, Version 1.0*, Xerox Corporation.

[2] Nationalencyklopedin., *Ethernet*, http://www.ne.se/lang/ethernet

[3] U, Von Burg, M Kenny (2003). *Sponsors, Communities, and Standards: Ethernet vs. Token Ring in the Local Area Networking Business*, Industry and Innovation, 351-375.

[4] C E. Spurgeon(2000). *Ethernet: The Definitive Guide*, O'Reilly Media, Sebastopol

[5] U, Von Burg, M Kenny (2001). *The Triumph of Ethernet: Technological Communities and the Battle for the LAN Standard*, Stanford Business Books, Palo Alto.

[6] Nationalencyklopedin., *Fakta om stål*, http://www.ne.se.focus.lib.kth.se/lang/stål

[7] SSAB., *Hållbara produkter*, http://www.ssab.com/en/Investor–Media/Sustainability/33/

[8] JJ, Löwgren. (2013). *SSAB Global Business Development: A Study of the International Marketing Expansion Model for HWP in China*, (Student paper). Mälardalens högskola.

[9] Proview Team (2013), *Fakta om Proview*, http://www.proview.se/

[10] SSAB. (2009). *Getting Started Guide*, SSAB, Oxelösund.

[11] SSAB. (2010). *Guide to I/O System*, SSAB, Oxelösund.

[12] SSAB. (2011). *Designer's Guide*, SSAB, Oxelösund.

[13] SSAB. (2011). *Developer's Guide*, SSAB, Oxelösund.

[14] O Pfeiffer, A Ayre, C Keydel. (2008). *Embedded networking with CAN and CANopen*, RTC Books, San Clemente.

[15] W Voss. (2005). *A Comprehensible Guide to Controller Area Network*, Copperhill Technologies Corporation,Greenfield Massachusetts.

[16] M Farsi, K Ratcliff, M Barbosa. (1999). *An introduction to CANopen*, Computing and Control Engineering Journal, 161-168.

[17] L. Seno, S. Vitturi, C. Zunino. (2009). *Real Time Ethernet Networks Evaluation Using Performance Indicators*, Emerging Technologies and Factory Automation,1-8.

[18] L. JA,Maestro, P Reviriego. (2010). *Energy Efficiency in Industrial Ethernet: The Case of Powerlink*, IEEE transactions on industrial electronics, 2896-2903.

[19] JD Decotignie. (2005). *Ethernet-Based Real-Time and Industrial Communications*, Proceedings of the IEEE, 1102-1117.

[20] A Quek. (2012). *POWERLINK Awarded National Standard in China*, Control Engineering Asia.

[21] G Cena, L Seno, A Valenzano and S Vitturi. (2010). *Performance analysis of Ethernet Powerlink networks for distributed control and automation systems*, Computer Standards and Interfaces, 566-572.

[22] M Felser. (2005). *Real-Time Ethernet?Industry Prospective*, Proceedings of the IEEE, 1118-1129.

[23] SYS TEC electronic. (2008). *Introduction into openPOWERLINK*, SYS TEC electronic GmbH, Greiz.

[24] J Baumgartner, S Schoenegger.(2010). *POWERLINK and Real-Time Linux: A Perfect Match for Highest Performance in Real Applications*, In Twelfth Real-Time Linux Workshop, Nairobi.

[25] W Wallner, J Baumgartner.(2011). *openPOWERLINK in Linux Userspace: Implementation and Performance Evaluation of the Real-Time Ethernet Protocol Stack in Linux Userspace*, Bernecker & Rainer Industrie-Elektronik, Eggelsberg.

[26] Kalycito Powerlink Team.(2009). *UM DemoProject openCONFIGURATOR For openCONFIGURATOR*, Kalycito Infotech, Coimbatore.

[27] ED Berger, T Yang, T Liu,G Novark.(2009). *Grace: safe multithreaded programming for C/C++*, ACM Sigplan Notices, 81-96.

Appendix A

Klasshjälp & Klasseditor

Detta är en grafisk representation av det verktyg som används för att skapa klasser, följt av den kod som genereras när klassvolymen byggs och den klasshjälp som finns tillgänglig.

Type $TypeHier
Class $ClassHier
Pl_Module $ClassDef
RtBody $ObjBodyDef
Description $Attribute
Specification $Attribute
DataSheet $Attribute
Process $Attribute
ThreadObject $Attribute
ErrorCount $Attribute
ErrorSoftLimit $Attribute
ErrorHardLimit $Attribute
InputAreaOffset $Attribute
InputAreaSize $Attribute
OutputAreaOffset $Attribute
OutputAreaSize $Attribute
IoMethods $RtMethod
IoCardInit $Method
IoCardClose $Method
IoCardRead $Method
IoCardWrite $Method
Template Pl_Module
Pl_CN $ClassDef
RtBody $ObjBodyDef
Description $Attribute
Specification $Attribute
DataSheet $Attribute
Process $Attribute
ThreadObject $Attribute
ErrorCount $Attribute
ErrorSoftLimit $Attribute
ErrorHardLimit $Attribute
ByteOrdering $Attribute
InputAreaOffset $Attribute
InputAreaSize $Attribute
OutputAreaOffset $Attribute
OutputAreaSize $Attribute
Template Pl_CN
Pl_MN $ClassDef
RtBody $ObjBodyDef
Description $Attribute
CDCfile $Attribute
Device $Attribute
NodeId $Attribute
Process $Attribute
ThreadObject $Attribute
EplStatus $Attribute
IoMethods $RtMethod
IoAgentInit $Method
IoAgentClose $Method
IoAgentRead $Method
IoAgentWrite $Method
Template Pl_MN

```
1  /*    Proview V4.8.6 pwr_cvolpltestclasses.h */
2
3  /*    Generated by co_convert. */
4  /*    Do not edit this file. */
5
6  #ifndef pwr_cvolpltestclasses_h
7  #define pwr_cvolpltestclasses_h
8
9  #ifndef pwr_class_h
10 #include "pwr_class.h"
11 #endif
12
13
14 /*_* Enum: EplStatusEnum
15     @Aref EplStatusEnum EplStatusEnum
16 */
17
18 typedef pwr_tEnum pwr_tEplStatusEnum;
19
20 typedef enum {
21   pwr_eEplStatusEnum_EplSuccessful     = 0,
22   pwr_eEplStatusEnum_EplIllegalInstance = 1,
23   pwr_eEplStatusEnum_EplInvalidInstanceParam = 2,
24   pwr_eEplStatusEnum_EplNoFreeInstance = 3,
25   pwr_eEplStatusEnum_EplWrongSignature = 4,
26   pwr_eEplStatusEnum_EplInvalidOperation = 5,
27   pwr_eEplStatusEnum_EplInvalidNodeId  = 7,
28   pwr_eEplStatusEnum_EplNoResource     = 8,
29   pwr_eEplStatusEnum_EplShutdown       = 9,
30   pwr_eEplStatusEnum_EplReject         = 10,
31   pwr_eEplStatusEnum_EplRetry          = 11,
32   pwr_eEplStatusEnum_EplInvalidEvent   = 12,
33   pwr_eEplStatusEnum_EplEdrvNoFreeTxDesc = 17,
34   pwr_eEplStatusEnum_EplEdrvInvalidCycleLen = 18,
35   pwr_eEplStatusEnum_EplEdrvInitError  = 19,
36   pwr_eEplStatusEnum_EplEdrvNoFreeBufEntry = 20,
37   pwr_eEplStatusEnum_EplEdrvBufNotExisting = 21,
38   pwr_eEplStatusEnum_EplEdrvInvalidRxBuf = 22,
39   pwr_eEplStatusEnum_EplEdrvInvalidParam = 28,
40   pwr_eEplStatusEnum_EplEdrvNextTxListNotEmpty = 29,
41   pwr_eEplStatusEnum_EplEdrvCurTxListEmpty = 30,
42   pwr_eEplStatusEnum_EplEdrvTxListNotFinishedYet = 31,
43   pwr_eEplStatusEnum_EplDllOutOfMemory = 33,
44   pwr_eEplStatusEnum_EplDllIllegalHdl  = 34,
45   pwr_eEplStatusEnum_EplDllCbAsyncRegistered = 35,
46   pwr_eEplStatusEnum_EplDllAsyncSyncReqFull = 36,
47   pwr_eEplStatusEnum_EplDllAsyncTxBufferEmpty = 37,
48   pwr_eEplStatusEnum_EplDllAsyncTxBufferFull = 38,
49   pwr_eEplStatusEnum_EplDllNoNodeInfo  = 39,
50   pwr_eEplStatusEnum_EplDllInvalidParam = 40,
51   pwr_eEplStatusEnum_EplDllInvalidAsndServiceId = 41,
52   pwr_eEplStatusEnum_EplDllTxBufNotReady = 46,
53   pwr_eEplStatusEnum_EplDllTxFrameInvalid = 47,
54   pwr_eEplStatusEnum_EplObdIllegalPart = 48,
55   pwr_eEplStatusEnum_EplObdIndexNotExist = 49,
56   pwr_eEplStatusEnum_EplObdSubindexNotExist = 50,
57   pwr_eEplStatusEnum_EplObdReadViolation = 51,
58   pwr_eEplStatusEnum_EplObdWriteViolation = 52,
```

```
59    pwr_eEplStatusEnum_EplObdAccessViolation = 53,
60    pwr_eEplStatusEnum_EplObdUnknownObjectType = 54,
61    pwr_eEplStatusEnum_EplObdVarEntryNotExist = 55,
62    pwr_eEplStatusEnum_EplObdValueTooLow = 56,
63    pwr_eEplStatusEnum_EplObdValueTooHigh = 57,
64    pwr_eEplStatusEnum_EplObdValueLengthError = 58,
65    pwr_eEplStatusEnum_EplObdErrnoSet    = 59,
66    pwr_eEplStatusEnum_EplObdInvalidDcf  = 60,
67    pwr_eEplStatusEnum_EplObdOutOfMemory = 61,
68    pwr_eEplStatusEnum_EplObdNoConfigData = 62,
69    pwr_eEplStatusEnum_EplNmtUnknownCommand = 64,
70    pwr_eEplStatusEnum_EplNmtInvalidFramePointer = 65,
71    pwr_eEplStatusEnum_EplNmtInvalidEvent = 66,
72    pwr_eEplStatusEnum_EplNmtInvalidState = 67,
73    pwr_eEplStatusEnum_EplNmtInvalidParam = 68,
74    pwr_eEplStatusEnum_EplNmtSyncReqRejected = 69,
75    pwr_eEplStatusEnum_EplSdoUdpMissCb   = 80,
76    pwr_eEplStatusEnum_EplSdoUdpNoSocket = 81,
77    pwr_eEplStatusEnum_EplSdoUdpSocketError = 82,
78    pwr_eEplStatusEnum_EplSdoUdpThreadError = 83,
79    pwr_eEplStatusEnum_EplSdoUdpNoFreeHandle = 84,
80    pwr_eEplStatusEnum_EplSdoUdpSendError = 85,
81    pwr_eEplStatusEnum_EplSdoUdpInvalidHdl = 86,
82    pwr_eEplStatusEnum_EplSdoSeqMissCb   = 96,
83    pwr_eEplStatusEnum_EplSdoSeqNoFreeHandle = 97,
84    pwr_eEplStatusEnum_EplSdoSeqInvalidHdl = 98,
85    pwr_eEplStatusEnum_EplSdoSeqUnsupportedProt = 99,
86    pwr_eEplStatusEnum_EplSdoSeqNoFreeHistory = 100,
87    pwr_eEplStatusEnum_EplSdoSeqFrameSizeError = 101,
88    pwr_eEplStatusEnum_EplSdoSeqRequestAckNeeded = 102,
89    pwr_eEplStatusEnum_EplSdoSeqInvalidFrame = 103,
90    pwr_eEplStatusEnum_EplSdoSeqConnectionBusy = 104,
91    pwr_eEplStatusEnum_EplSdoSeqInvalidEvent = 105,
92    pwr_eEplStatusEnum_EplSdoComUnsupportedProt = 112,
93    pwr_eEplStatusEnum_EplSdoComNoFreeHandle = 113,
94    pwr_eEplStatusEnum_EplSdoComInvalidServiceType = 114,
95    pwr_eEplStatusEnum_EplSdoComInvalidHandle = 115,
96    pwr_eEplStatusEnum_EplSdoComInvalidSendType = 116,
97    pwr_eEplStatusEnum_EplSdoComNotResponsible = 117,
98    pwr_eEplStatusEnum_EplSdoComHandleExists = 118,
99    pwr_eEplStatusEnum_EplSdoComHandleBusy = 119,
100   pwr_eEplStatusEnum_EplSdoComInvalidParam = 120,
101   pwr_eEplStatusEnum_EplEventUnknownSink = 128,
102   pwr_eEplStatusEnum_EplEventPostError = 129,
103   pwr_eEplStatusEnum_EplEventReadError = 130,
104   pwr_eEplStatusEnum_EplEventWrongSize = 131,
105   pwr_eEplStatusEnum_EplTimerInvalidHandle = 144,
106   pwr_eEplStatusEnum_EplTimerNoTimerCreated = 145,
107   pwr_eEplStatusEnum_EplTimerThreadError = 146,
108   pwr_eEplStatusEnum_EplSdoAsndInvalidNodeId = 160,
109   pwr_eEplStatusEnum_EplSdoAsndNoFreeHandle = 161,
110   pwr_eEplStatusEnum_EplSdoAsndInvalidHandle = 162,
111   pwr_eEplStatusEnum_EplPdoNotExist    = 176,
112   pwr_eEplStatusEnum_EplPdoLengthExceeded = 177,
113   pwr_eEplStatusEnum_EplPdoGranularityMismatch = 178,
114   pwr_eEplStatusEnum_EplPdoInitError   = 179,
115   pwr_eEplStatusEnum_EplPdoConfWhileEnabled = 183,
116   pwr_eEplStatusEnum_EplPdoErrorMapp   = 184,
```

```
117     pwr_eEplStatusEnum_EplPdoVarNotFound = 185,
118     pwr_eEplStatusEnum_EplPdoVarNotMappable = 186,
119     pwr_eEplStatusEnum_EplPdoSizeMismatch = 188,
120     pwr_eEplStatusEnum_EplPdoTooManyTxPdos = 189,
121     pwr_eEplStatusEnum_EplPdoInvalidObjIndex = 190,
122     pwr_eEplStatusEnum_EplPdoTooManyPdos = 191,
123     pwr_eEplStatusEnum_EplCfmConfigError = 192,
124     pwr_eEplStatusEnum_EplCfmSdocTimeOutError = 193,
125     pwr_eEplStatusEnum_EplCfmInvalidDcf  = 194,
126     pwr_eEplStatusEnum_EplCfmUnsupportedDcf = 195,
127     pwr_eEplStatusEnum_EplCfmConfigWithErrors = 196,
128     pwr_eEplStatusEnum_EplCfmNoFreeConfig = 197,
129     pwr_eEplStatusEnum_EplCfmNoConfigData = 198,
130     pwr_eEplStatusEnum_EplCfmUnsuppDatatypeDcf = 199,
131     pwr_eEplStatusEnum_EplApiTaskDeferred = 320,
132     pwr_eEplStatusEnum_EplApiInvalidParam = 322,
133     pwr_eEplStatusEnum_EplApiNoObdInitRam = 323,
134     pwr_eEplStatusEnum_EplApiSdoBusyIntern = 324,
135     pwr_eEplStatusEnum_EplApiPIAlreadyAllocated = 325,
136     pwr_eEplStatusEnum_EplApiPIOutOfMemory = 326,
137     pwr_eEplStatusEnum_EplApiPISizeExceeded = 327,
138     pwr_eEplStatusEnum_EplApiPINotAllocated = 328,
139     pwr_eEplStatusEnum_EplApiPIJobQueueFull = 329,
140     pwr_eEplStatusEnum_EplApiPIJobQueueEmpty = 330,
141     pwr_eEplStatusEnum_EplApiPIInvalidJobSize = 331,
142     pwr_eEplStatusEnum_EplApiPIInvalidPIPointer = 332,
143     pwr_eEplStatusEnum_EplApiPINonBlockingNotSupp = 333,
144 } pwr_eEplStatusEnum;
145
146
147 #ifndef pwr_cClass_Pl_Module
148 #define pwr_cClass_Pl_Module 1662779416UL
149
150 /*_* Class: Pl_Module
151     Body:  RtBody
152     @Aref Pl_Module pwr_sClass_Pl_Module
153 */
154
155 typedef struct {
156     pwr_tString80                       Description pwr_dAlignLW;
157     pwr_tString80                       Specification pwr_dAlignW;
158     pwr_tURL                            DataSheet pwr_dAlignW;
159     pwr_tIoProcessMask                  Process pwr_dAlignW;
160     pwr_tObjid                          ThreadObject pwr_dAlignW;
161     pwr_tUInt16                         ErrorCount pwr_dAlignW;
162     pwr_tUInt16                         ErrorSoftLimit pwr_dAlignW;
163     pwr_tUInt16                         ErrorHardLimit pwr_dAlignW;
164     pwr_tUInt32                         InputAreaOffset pwr_dAlignW;
165     pwr_tUInt32                         InputAreaSize pwr_dAlignW;
166     pwr_tUInt32                         OutputAreaOffset pwr_dAlignW;
167     pwr_tUInt32                         OutputAreaSize pwr_dAlignW;
168 } pwr_sClass_Pl_Module;
169
170 #endif
171
172
173 #ifndef pwr_cClass_Pl_CN
174 #define pwr_cClass_Pl_CN 1662779408UL
```

```
175
176 /*_* Class: Pl_CN
177     Body:  RtBody
178     @Aref Pl_CN pwr_sClass_Pl_CN
179 */
180
181 typedef struct {
182   pwr_tString80                         Description pwr_dAlignLW;
183   pwr_tString80                         Specification pwr_dAlignW;
184   pwr_tURL                              DataSheet pwr_dAlignW;
185   pwr_tIoProcessMask                    Process pwr_dAlignW;
186   pwr_tObjid                            ThreadObject pwr_dAlignW;
187   pwr_tUInt16                           ErrorCount pwr_dAlignW;
188   pwr_tUInt16                           ErrorSoftLimit pwr_dAlignW;
189   pwr_tUInt16                           ErrorHardLimit pwr_dAlignW;
190   pwr_tByteOrderingEnum                 ByteOrdering pwr_dAlignW;
191   pwr_tUInt32                           InputAreaOffset pwr_dAlignW;
192   pwr_tUInt32                           InputAreaSize pwr_dAlignW;
193   pwr_tUInt32                           OutputAreaOffset pwr_dAlignW;
194   pwr_tUInt32                           OutputAreaSize pwr_dAlignW;
195 } pwr_sClass_Pl_CN;
196
197 #endif
198
199
200 #ifndef pwr_cClass_Pl_MN
201 #define pwr_cClass_Pl_MN 1662779400UL
202
203 /*_* Class: Pl_MN
204     Body:  RtBody
205     @Aref Pl_MN pwr_sClass_Pl_MN
206 */
207
208 typedef struct {
209   pwr_tString80                         Description pwr_dAlignLW;
210   pwr_tString256                        CDCfile pwr_dAlignW;
211   pwr_tString80                         Device pwr_dAlignW;
212   pwr_tUInt16                           NodeId pwr_dAlignW;
213   pwr_tIoProcessMask                    Process pwr_dAlignW;
214   pwr_tObjid                            ThreadObject pwr_dAlignW;
215   pwr_tEplStatusEnum                    EplStatus pwr_dAlignW;
216 } pwr_sClass_Pl_MN;
217
218 #endif
219
220 #endif
221
```

Class CVolPlTest:Pl_MN

Pl_MN

Author **JackE SixtenI**
Version **1.0**

Description

Before the use of a Powerlink agent object a configuration file must be created (cdc-file).

The prefered software for building the cdc-file is openCONFIGURATOR. When using openCONFIGURATOR you must provide the device description files (.xdd) for all the slaves that will be used in the Powerlink network.

The mapping of the in- and outputareas will be made with help of openCONFIGURATOR. All communication parameters such as cycle time will be configured using openCONFIGURATOR.

When the configuration is complete you build the cdc-file and transfer it to your project. Prefered location of the cdc-file is $pwrp_exe of the current project. Then all you have to do is to specify the path in the CDCfile attribute.

When building the cdc-file you will also get a file named xap.h, this file contains two structs that represents the in- and outputareas. You use this as a map when you add channel objects to the module object, when done this file should coincide with the channel objects you put under the module objects.

The NodeId is always 240 for a Powerlinkmaster. The address range 1-239 is used for slaves, you specify the address of a slave in the openCONFIGURATOR.

A basic setup would be: first put a Pl_MN object in the node hierarchy and under the Pl_MN object you put a Pl_CN object and under the Pl_CN object you put a Pl_Module object and under the Pl_Module object you put channel objects, with the chan-in objects first and the chan-out objects last.

Example:

```
Pl_MN
 -Pl_CN
  -Pl_Module
   -ChanAi
   -ChanAi
   -ChanAi
   -ChanIo
   -ChanAo
 -Pl_CN
  -Pl_Module
   -ChanAi
   -ChanAo
```

RtBody

Description

String80 $Attribute
Used to add a short description of the master

CDCfile

String256 $Attribute
This attribute should contain a path to a cdc-file.
Example: /home/pwrp/mnobd.cdc

Example: /home/pwrp/mnobd.cdc

Device

String80 $Attribute

The network device used by the Powerlinkstack
Example: eth0

NodeId

UInt16 $Attribute Flags[Noedit]

The NodeId of the master, should always be 240.

Process

IoProcessMask $Attribute

The Proview process the agent object is associated with

ThreadObject

Objid $Attribute

EplStatus

EplStatusEnum $Attribute Flags[State|Noedit]

The current status of the Powerlink stack

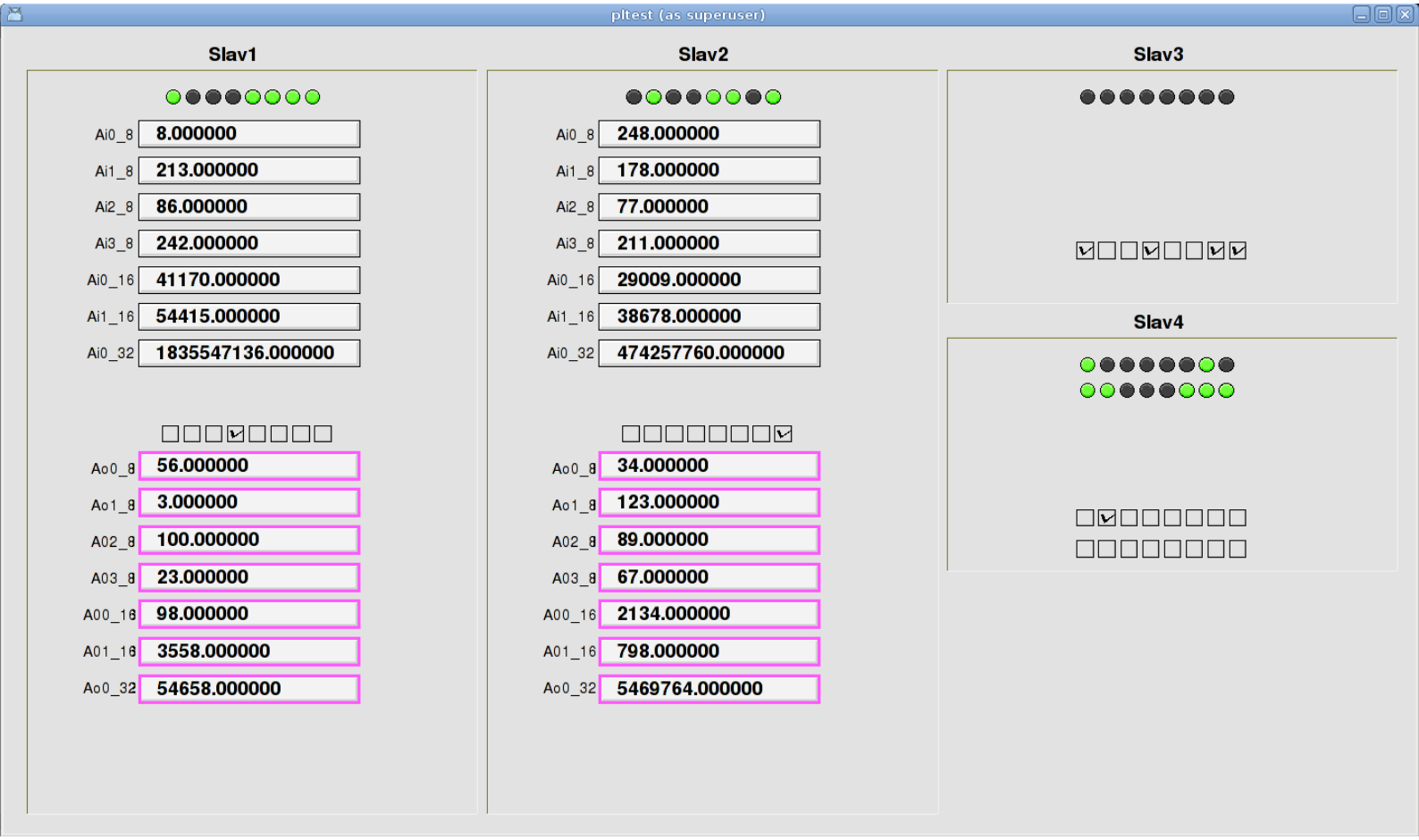
Appendix B

Anläggnings- & nodhierarki

Detta är en grafisk representation av ett Proviewprojekt. Första bilden visar anläggningshierarkin följt av nodhierarkin. Sista bilden visar en operatörsbild.

POWERLINK	$PlantHier	
Slav1	$PlantHier	
Slav2	$PlantHier	
Slav3	$PlantHier	
Styrning	PlcPgm	
DigitalIn0	Di	
DigitalIn1	Di	
DigitalIn2	Di	
DigitalIn3	Di	
DigitalIn4	Di	
DigitalIn5	Di	
DigitalIn6	Di	
DigitalIn7	Di	
DigitalOut0	Do	
DigitalOut1	Do	
DigitalOut2	Do	
DigitalOut3	Do	
DigitalOut4	Do	
DigitalOut5	Do	
DigitalOut6	Do	
DigitalOut7	Do	
Slav4	$PlantHier	
Styrning	PlcPgm	
DigitalIn0	Di	
DigitalIn1	Di	
DigitalIn2	Di	
DigitalIn3	Di	
DigitalIn4	Di	
DigitalIn5	Di	
DigitalIn6	Di	
DigitalIn7	Di	
DigitalIn8	Di	
DigitalIn9	Di	
DigitalIn10	Di	
DigitalIn11	Di	
DigitalIn12	Di	
DigitalIn13	Di	
DigitalIn14	Di	
DigitalIn15	Di	
Padd1	Di	
DigitalOut0	Do	
DigitalOut1	Do	
DigitalOut2	Do	
DigitalOut3	Do	
DigitalOut4	Do	
DigitalOut5	Do	
DigitalOut6	Do	
DigitalOut7	Do	
DigitalOut8	Do	
DigitalOut9	Do	
DigitalOut10	Do	
DigitalOut11	Do	
DigitalOut12	Do	
DigitalOut13	Do	
DigitalOut14	Do	
DigitalOut15	Do	
Padd2	Do	
ABB	$LibHier	

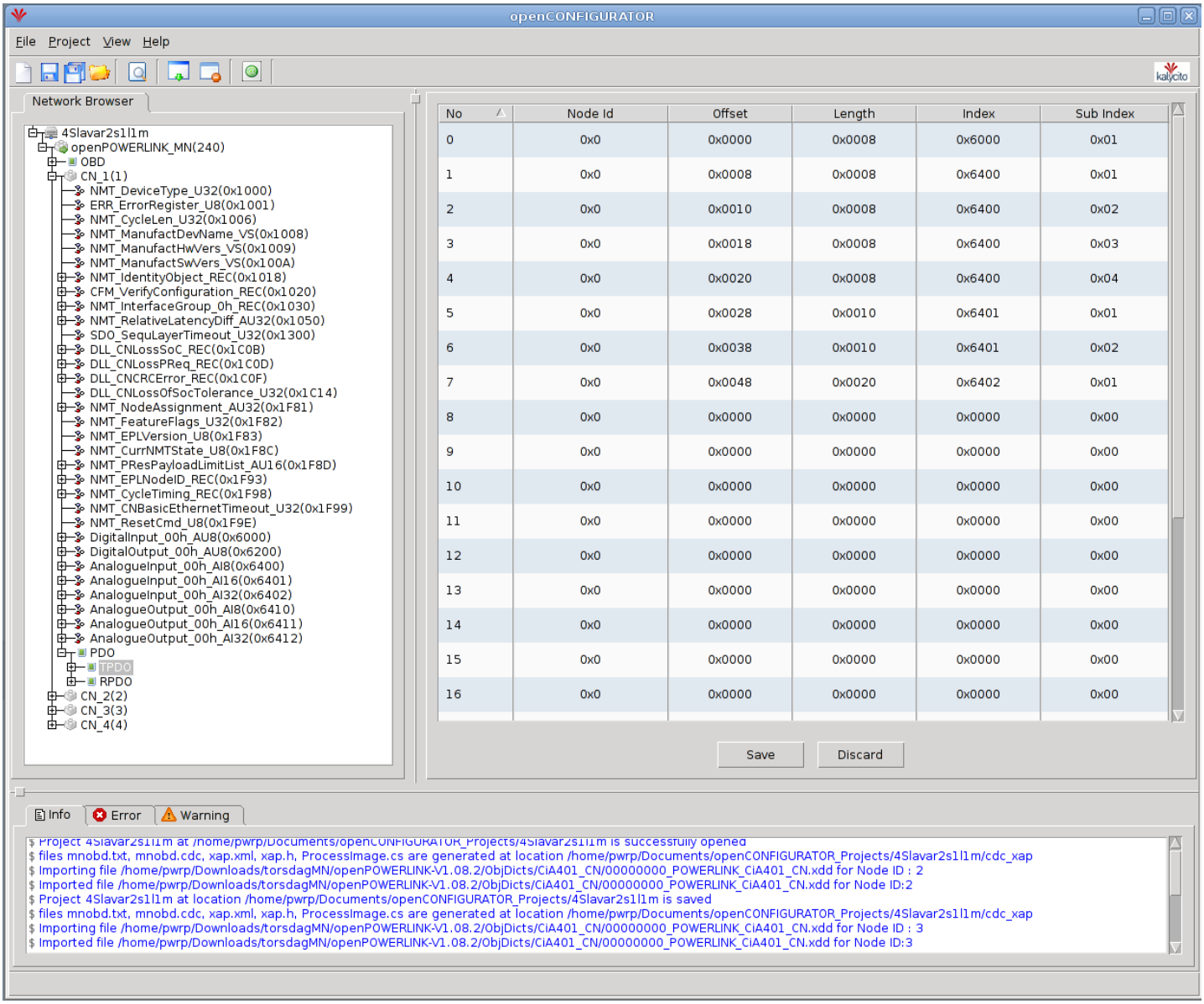
Nodes	$NodeHier			
PlTest	$Node			
Security	$Security			
MessageHandler	MessageHandler			
IOHandler	IOHandler			
Backup	Backup_Conf			
Op	OpPlace			
Maintenance	OpPlace			
OpDefault	OpPlace			
Plc	PlcProcess			
WebHandler	WebHandler			
WebBrowser	WebBrowserConfig			
StatusServer	StatusServerConfig			
Master	Pl_MN	Första Powerlink MN		
Slav1	Pl_CN			
Card	Pl_Module			
Slav2	Pl_CN			
Slav3	Pl_CN			
Card	Pl_Module			
DI_0	ChanDi			
DI_1	ChanDi			
DI_2	ChanDi			
DI_3	ChanDi			
DI_4	ChanDi			
DI_5	ChanDi			
DI_6	ChanDi			
DI_7	ChanDi			
DO_0	ChanDo			
DO_1	ChanDo			
DO_2	ChanDo			
DO_3	ChanDo			
DO_4	ChanDo			
DO_5	ChanDo			
DO_6	ChanDo			
DO_7	ChanDo			
Slav4	Pl_CN			
POWERLINK	$PlantHier			
pwrNode	$NodeHier			


pltest (as superuser)
Slav1
Ai0_8 8.000000
Ai1_8 213.000000
Ai2_8 86.000000
Ai3_8 242.000000
Ai0_16 41170.000000
Ai1_16 54415.000000
Ai0_32 1835547136.000000
Ao0_8 56.000000
Ao1_8 3.000000
A02_8 100.000000
A03_8 23.000000
A00_16 98.000000
A01_16 3558.000000
Ao0_32 54658.000000
Slav2
Ai0_8 248.000000
Ai1_8 178.000000
Ai2_8 77.000000
Ai3_8 211.000000
Ai0_16 29009.000000
Ai1_16 38678.000000
Ai0_32 474257760.000000
Ao0_8 34.000000
Ao1_8 123.000000
A02_8 89.000000
A03_8 67.000000
A00_16 2134.000000
A01_16 798.000000
Ao0_32 5469764.000000
Slav3
Slav4

Appendix C

openCONFIGURATOR

Nodkonfiguration som visar variabelmappning i openCONFIGURATOR. Bild två visar en autogenererad h-fil som representerar variablerna position i processbilden.


openCONFIGURATOR
File Project View Help
Network Browser
4Slavar2s1l1m
openPOWERLINK_MN(240)
OBD
CN_1(1)
NMT_DeviceType_U32(0x1000)
ERR_ErrorRegister_U8(0x1001)
NMT_CycleLen_U32(0x1006)
NMT_ManufactDevName_VS(0x1008)
NMT_ManufactHwVers_VS(0x1009)
NMT_ManufactSwVers_VS(0x100A)
NMT_IdentityObject_REC(0x1018)
CFM_VerifyConfiguration_REC(0x1020)
NMT_InterfaceGroup_0h_REC(0x1030)
NMT_RelativeLatencyDiff_AU32(0x1050)
SDO_SequLayerTimeout_U32(0x1300)
DLL_CNLossSoC_REC(0x1C0B)
DLL_CNLossPReq_REC(0x1C0D)
DLL_CNCRCError_REC(0x1C0F)
DLL_CNLossOfSocTolerance_U32(0x1C14)
NMT_NodeAssignment_AU32(0x1F81)
NMT_FeatureFlags_U32(0x1F82)
NMT_EPLVersion_U8(0x1F83)
NMT_CurrNMTState_U8(0x1F8C)
NMT_PResPayloadLimitList_AU16(0x1F8D)
NMT_EPLNodeID_REC(0x1F93)
NMT_CycleTiming_REC(0x1F98)
NMT_CNBasicEthernetTimeout_U32(0x1F99)
NMT_ResetCmd_U8(0x1F9E)
DigitalInput_00h_AU8(0x6000)
DigitalOutput_00h_AU8(0x6200)
AnalogueInput_00h_AI8(0x6400)
AnalogueInput_00h_AI16(0x6401)
AnalogueInput_00h_AI32(0x6402)
AnalogueOutput_00h_AI8(0x6410)
AnalogueOutput_00h_AI16(0x6411)
AnalogueOutput_00h_AI32(0x6412)
PDO
TPDO
RPDO
CN_2(2)
CN_3(3)
CN_4(4)
No
Node Id
Offset
Length
Index
Sub Index
0 0x0 0x0000 0x0008 0x6000 0x01
1 0x0 0x0008 0x0008 0x6400 0x01
2 0x0 0x0010 0x0008 0x6400 0x02
3 0x0 0x0018 0x0008 0x6400 0x03
4 0x0 0x0020 0x0008 0x6400 0x04
5 0x0 0x0028 0x0010 0x6401 0x01
6 0x0 0x0038 0x0010 0x6401 0x02
7 0x0 0x0048 0x0020 0x6402 0x01
8 0x0 0x0000 0x0000 0x0000 0x00
9 0x0 0x0000 0x0000 0x0000 0x00
10 0x0 0x0000 0x0000 0x0000 0x00
11 0x0 0x0000 0x0000 0x0000 0x00
12 0x0 0x0000 0x0000 0x0000 0x00
13 0x0 0x0000 0x0000 0x0000 0x00
14 0x0 0x0000 0x0000 0x0000 0x00
15 0x0 0x0000 0x0000 0x0000 0x00
16 0x0 0x0000 0x0000 0x0000 0x00
Save
Discard
Info
Error
Warning
$ files mnobd.txt, mnobd.cdc, xap.xml, xap.h, ProcessImage.cs are generated at location /home/pwrp/Documents/openCONFIGURATOR_Projects/4Slavar2s1l1m/cdc_xap
$ Importing file /home/pwrp/Downloads/torsdagMN/openPOWERLINK-V1.08.2/ObjDicts/CiA401_CN/00000000_POWERLINK_CiA401_CN.xdd for Node ID : 2
$ Imported file /home/pwrp/Downloads/torsdagMN/openPOWERLINK-V1.08.2/ObjDicts/CiA401_CN/00000000_POWERLINK_CiA401_CN.xdd for Node ID:2
$ Project 4Slavar2s1l1m at location /home/pwrp/Documents/openCONFIGURATOR_Projects/4Slavar2s1l1m is saved
$ files mnobd.txt, mnobd.cdc, xap.xml, xap.h, ProcessImage.cs are generated at location /home/pwrp/Documents/openCONFIGURATOR_Projects/4Slavar2s1l1m/cdc_xap
$ Importing file /home/pwrp/Downloads/torsdagMN/openPOWERLINK-V1.08.2/ObjDicts/CiA401_CN/00000000_POWERLINK_CiA401_CN.xdd for Node ID : 3
$ Imported file /home/pwrp/Downloads/torsdagMN/openPOWERLINK-V1.08.2/ObjDicts/CiA401_CN/00000000_POWERLINK_CiA401_CN.xdd for Node ID:3

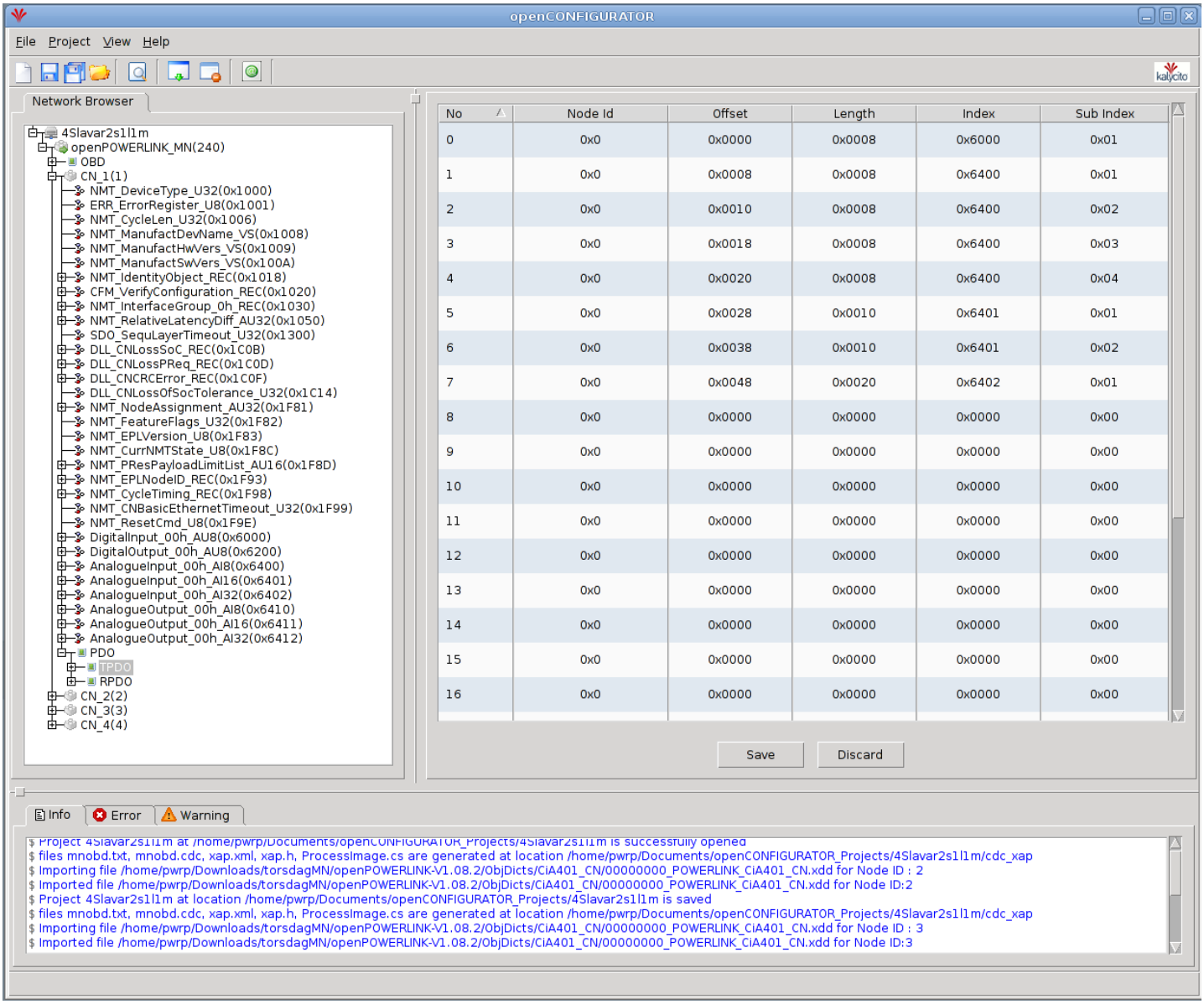
```
1  /* This file was autogenerated by openCONFIGURATOR-1.2.2 on 17-Jun-2013 14:09:1
   1 */
2  #ifndef XAP_h
3  #define XAP_h
4
5
6  # define COMPUTED_PI_OUT_SIZE 36
7  // The inputarea
8  typedef struct
9  {
10     unsigned CN1_M00_DigitalInput_00h_AU8_DigitalInput:8;
11     unsigned CN1_M00_AnalogueInput_00h_AI8_AnalogueInput01:8;
12     unsigned CN1_M00_AnalogueInput_00h_AI8_AnalogueInput02:8;
13     unsigned CN1_M00_AnalogueInput_00h_AI8_AnalogueInput03:8;
14     unsigned CN1_M00_AnalogueInput_00h_AI8_AnalogueInput04:8;
15     unsigned PADDING_VAR_1:8;
16     unsigned CN1_M01_AnalogueInput_00h_AI16_AnalogueInput01:16;
17     unsigned CN1_M01_AnalogueInput_00h_AI16_AnalogueInput02:16;
18     unsigned PADDING_VAR_2:16;
19     unsigned CN1_M02_AnalogueInput_00h_AI32_AnalogueInput:32;
20     unsigned CN2_M00_DigitalInput_00h_AU8_DigitalInput:8;
21     unsigned CN2_M00_AnalogueInput_00h_AI8_AnalogueInput01:8;
22     unsigned CN2_M00_AnalogueInput_00h_AI8_AnalogueInput02:8;
23     unsigned CN2_M00_AnalogueInput_00h_AI8_AnalogueInput03:8;
24     unsigned CN2_M00_AnalogueInput_00h_AI8_AnalogueInput04:8;
25     unsigned PADDING_VAR_3:8;
26     unsigned CN2_M01_AnalogueInput_00h_AI16_AnalogueInput01:16;
27     unsigned CN2_M01_AnalogueInput_00h_AI16_AnalogueInput02:16;
28     unsigned PADDING_VAR_4:16;
29     unsigned CN2_M02_AnalogueInput_00h_AI32_AnalogueInput:32;
30     unsigned CN3_M00_DigitalInput_00h_AU8_DigitalInput:8;
31     unsigned CN4_M00_DigitalInput_00h_AU8_DigitalInput01:8;
32     unsigned CN4_M00_DigitalInput_00h_AU8_DigitalInput02:8;
33     unsigned PADDING_VAR_5:8;
34 } PI_OUT;
35
36 # define COMPUTED_PI_IN_SIZE 36
37 // The outputarea
38 typedef struct
39 {
40     unsigned CN1_M00_DigitalOutput_00h_AU8_DigitalOutput:8;
41     unsigned CN1_M10_AnalogueOutput_00h_AI8_AnalogueOutput01:8;
42     unsigned CN1_M10_AnalogueOutput_00h_AI8_AnalogueOutput02:8;
43     unsigned CN1_M10_AnalogueOutput_00h_AI8_AnalogueOutput03:8;
44     unsigned CN1_M10_AnalogueOutput_00h_AI8_AnalogueOutput04:8;
45     unsigned PADDING_VAR_1:8;
46     unsigned CN1_M11_AnalogueOutput_00h_AI16_AnalogueOutput01:16;
47     unsigned CN1_M11_AnalogueOutput_00h_AI16_AnalogueOutput02:16;
48     unsigned PADDING_VAR_2:16;
49     unsigned CN1_M12_AnalogueOutput_00h_AI32_AnalogueOutput:32;
50     unsigned CN2_M00_DigitalOutput_00h_AU8_DigitalOutput:8;
51     unsigned CN2_M10_AnalogueOutput_00h_AI8_AnalogueOutput01:8;
52     unsigned CN2_M10_AnalogueOutput_00h_AI8_AnalogueOutput02:8;
53     unsigned CN2_M10_AnalogueOutput_00h_AI8_AnalogueOutput03:8;
54     unsigned CN2_M10_AnalogueOutput_00h_AI8_AnalogueOutput04:8;
55     unsigned PADDING_VAR_3:8;
56     unsigned CN2_M11_AnalogueOutput_00h_AI16_AnalogueOutput01:16;
57     unsigned CN2_M11_AnalogueOutput_00h_AI16_AnalogueOutput02:16;
```

```
58      unsigned PADDING_VAR_4:16;
59      unsigned CN2_M12_AnalogueOutput_00h_AI32_AnalogueOutput:32;
60      unsigned CN3_M00_DigitalOutput_00h_AU8_DigitalOutput:8;
61      unsigned CN4_M00_DigitalOutput_00h_AU8_DigitalOutput:8;
62      unsigned PADDING_VAR_5:16;
63  } PI_IN;
64
65  #endif
66
```

Appendix D

openCONFIGURATOR

Nodkonfiguration som visar variabelmappning i openCONFIGURATOR. Bild två visar en autogenererad h-fil som representerar variablerna position i processbilden.


openCONFIGURATOR
File Project View Help
Network Browser
4Slavar2s1l1m
openPOWERLINK_MN(240)
OBD
CN_1(1)
NMT_DeviceType_U32(0x1000)
ERR_ErrorRegister_U8(0x1001)
NMT_CycleLen_U32(0x1006)
NMT_ManufactDevName_VS(0x1008)
NMT_ManufactHwVers_VS(0x1009)
NMT_ManufactSwVers_VS(0x100A)
NMT_IdentityObject_REC(0x1018)
CFM_VerifyConfiguration_REC(0x1020)
NMT_InterfaceGroup_0h_REC(0x1030)
NMT_RelativeLatencyDiff_AU32(0x1050)
SDO_SequLayerTimeout_U32(0x1300)
DLL_CNLossSoC_REC(0x1C0B)
DLL_CNLossPReq_REC(0x1C0D)
DLL_CNCRCError_REC(0x1C0F)
DLL_CNLossOfSocTolerance_U32(0x1C14)
NMT_NodeAssignment_AU32(0x1F81)
NMT_FeatureFlags_U32(0x1F82)
NMT_EPLVersion_U8(0x1F83)
NMT_CurrNMTState_U8(0x1F8C)
NMT_PResPayloadLimitList_AU16(0x1F8D)
NMT_EPLNodeID_REC(0x1F93)
NMT_CycleTiming_REC(0x1F98)
NMT_CNBasicEthernetTimeout_U32(0x1F99)
NMT_ResetCmd_U8(0x1F9E)
DigitalInput_00h_AU8(0x6000)
DigitalOutput_00h_AU8(0x6200)
AnalogueInput_00h_AI8(0x6400)
AnalogueInput_00h_AI16(0x6401)
AnalogueInput_00h_AI32(0x6402)
AnalogueOutput_00h_AI8(0x6410)
AnalogueOutput_00h_AI16(0x6411)
AnalogueOutput_00h_AI32(0x6412)
PDO
TPDO
RPDO
CN_2(2)
CN_3(3)
CN_4(4)
No
Node Id
Offset
Length
Index
Sub Index
0 0x0 0x0000 0x0008 0x6000 0x01
1 0x0 0x0008 0x0008 0x6400 0x01
2 0x0 0x0010 0x0008 0x6400 0x02
3 0x0 0x0018 0x0008 0x6400 0x03
4 0x0 0x0020 0x0008 0x6400 0x04
5 0x0 0x0028 0x0010 0x6401 0x01
6 0x0 0x0038 0x0010 0x6401 0x02
7 0x0 0x0048 0x0020 0x6402 0x01
8 0x0 0x0000 0x0000 0x0000 0x00
9 0x0 0x0000 0x0000 0x0000 0x00
10 0x0 0x0000 0x0000 0x0000 0x00
11 0x0 0x0000 0x0000 0x0000 0x00
12 0x0 0x0000 0x0000 0x0000 0x00
13 0x0 0x0000 0x0000 0x0000 0x00
14 0x0 0x0000 0x0000 0x0000 0x00
15 0x0 0x0000 0x0000 0x0000 0x00
16 0x0 0x0000 0x0000 0x0000 0x00
Save
Discard
Info
Error
Warning
$ files mnobd.txt, mnobd.cdc, xap.xml, xap.h, ProcessImage.cs are generated at location /home/pwrp/Documents/openCONFIGURATOR_Projects/4Slavar2s1l1m/cdc_xap
$ Importing file /home/pwrp/Downloads/torsdagMN/openPOWERLINK-V1.08.2/ObjDicts/CiA401_CN/00000000_POWERLINK_CiA401_CN.xdd for Node ID : 2
$ Imported file /home/pwrp/Downloads/torsdagMN/openPOWERLINK-V1.08.2/ObjDicts/CiA401_CN/00000000_POWERLINK_CiA401_CN.xdd for Node ID:2
$ Project 4Slavar2s1l1m at location /home/pwrp/Documents/openCONFIGURATOR_Projects/4Slavar2s1l1m is saved
$ files mnobd.txt, mnobd.cdc, xap.xml, xap.h, ProcessImage.cs are generated at location /home/pwrp/Documents/openCONFIGURATOR_Projects/4Slavar2s1l1m/cdc_xap
$ Importing file /home/pwrp/Downloads/torsdagMN/openPOWERLINK-V1.08.2/ObjDicts/CiA401_CN/00000000_POWERLINK_CiA401_CN.xdd for Node ID : 3
$ Imported file /home/pwrp/Downloads/torsdagMN/openPOWERLINK-V1.08.2/ObjDicts/CiA401_CN/00000000_POWERLINK_CiA401_CN.xdd for Node ID:3

```
1  /* This file was autogenerated by openCONFIGURATOR-1.2.2 on 17-Jun-2013 14:09:1
   1 */
2  #ifndef XAP_h
3  #define XAP_h
4
5
6  # define COMPUTED_PI_OUT_SIZE 36
7  // The inputarea
8  typedef struct
9  {
10     unsigned CN1_M00_DigitalInput_00h_AU8_DigitalInput:8;
11     unsigned CN1_M00_AnalogueInput_00h_AI8_AnalogueInput01:8;
12     unsigned CN1_M00_AnalogueInput_00h_AI8_AnalogueInput02:8;
13     unsigned CN1_M00_AnalogueInput_00h_AI8_AnalogueInput03:8;
14     unsigned CN1_M00_AnalogueInput_00h_AI8_AnalogueInput04:8;
15     unsigned PADDING_VAR_1:8;
16     unsigned CN1_M01_AnalogueInput_00h_AI16_AnalogueInput01:16;
17     unsigned CN1_M01_AnalogueInput_00h_AI16_AnalogueInput02:16;
18     unsigned PADDING_VAR_2:16;
19     unsigned CN1_M02_AnalogueInput_00h_AI32_AnalogueInput:32;
20     unsigned CN2_M00_DigitalInput_00h_AU8_DigitalInput:8;
21     unsigned CN2_M00_AnalogueInput_00h_AI8_AnalogueInput01:8;
22     unsigned CN2_M00_AnalogueInput_00h_AI8_AnalogueInput02:8;
23     unsigned CN2_M00_AnalogueInput_00h_AI8_AnalogueInput03:8;
24     unsigned CN2_M00_AnalogueInput_00h_AI8_AnalogueInput04:8;
25     unsigned PADDING_VAR_3:8;
26     unsigned CN2_M01_AnalogueInput_00h_AI16_AnalogueInput01:16;
27     unsigned CN2_M01_AnalogueInput_00h_AI16_AnalogueInput02:16;
28     unsigned PADDING_VAR_4:16;
29     unsigned CN2_M02_AnalogueInput_00h_AI32_AnalogueInput:32;
30     unsigned CN3_M00_DigitalInput_00h_AU8_DigitalInput:8;
31     unsigned CN4_M00_DigitalInput_00h_AU8_DigitalInput01:8;
32     unsigned CN4_M00_DigitalInput_00h_AU8_DigitalInput02:8;
33     unsigned PADDING_VAR_5:8;
34 } PI_OUT;
35
36 # define COMPUTED_PI_IN_SIZE 36
37 // The outputarea
38 typedef struct
39 {
40     unsigned CN1_M00_DigitalOutput_00h_AU8_DigitalOutput:8;
41     unsigned CN1_M10_AnalogueOutput_00h_AI8_AnalogueOutput01:8;
42     unsigned CN1_M10_AnalogueOutput_00h_AI8_AnalogueOutput02:8;
43     unsigned CN1_M10_AnalogueOutput_00h_AI8_AnalogueOutput03:8;
44     unsigned CN1_M10_AnalogueOutput_00h_AI8_AnalogueOutput04:8;
45     unsigned PADDING_VAR_1:8;
46     unsigned CN1_M11_AnalogueOutput_00h_AI16_AnalogueOutput01:16;
47     unsigned CN1_M11_AnalogueOutput_00h_AI16_AnalogueOutput02:16;
48     unsigned PADDING_VAR_2:16;
49     unsigned CN1_M12_AnalogueOutput_00h_AI32_AnalogueOutput:32;
50     unsigned CN2_M00_DigitalOutput_00h_AU8_DigitalOutput:8;
51     unsigned CN2_M10_AnalogueOutput_00h_AI8_AnalogueOutput01:8;
52     unsigned CN2_M10_AnalogueOutput_00h_AI8_AnalogueOutput02:8;
53     unsigned CN2_M10_AnalogueOutput_00h_AI8_AnalogueOutput03:8;
54     unsigned CN2_M10_AnalogueOutput_00h_AI8_AnalogueOutput04:8;
55     unsigned PADDING_VAR_3:8;
56     unsigned CN2_M11_AnalogueOutput_00h_AI16_AnalogueOutput01:16;
57     unsigned CN2_M11_AnalogueOutput_00h_AI16_AnalogueOutput02:16;
```

```
58      unsigned PADDING_VAR_4:16;
59      unsigned CN2_M12_AnalogueOutput_00h_AI32_AnalogueOutput:32;
60      unsigned CN3_M00_DigitalOutput_00h_AU8_DigitalOutput:8;
61      unsigned CN4_M00_DigitalOutput_00h_AU8_DigitalOutput:8;
62      unsigned PADDING_VAR_5:16;
63  } PI_IN;
64
65  #endif
66
```

Appendix E

Källkod

Denna bilaga innehåller källkoden för Powerlinkimplementationen.

- rt_io_user.h importerar de registrerade metoderna till Proviews ramverk
- rt_io_m_pl.h definierar de sammansatta datatyper som är specifika för varje klass
- ra_io.c emulerar Proviews sätt att initiera och starta I/O system
- ra_io_m_pl_mn.c agentobjektets funktioner för initiering, läsning, skrivning och avslut av Powerlinkstacken
- ra_io_m_pl_module.c modulobjektets funktioner för initiering, läsning, skrivning och avslut av Powerlinkstacken

```
 1 /*
 2  * Proview   Open Source Process Control.
 3  * Copyright (C) 2005-2012 SSAB EMEA AB.
 4  *
 5  * This file is part of Proview.
 6  *
 7  * This program is free software; you can redistribute it and/or
 8  * modify it under the terms of the GNU General Public License as
 9  * published by the Free Software Foundation, either version 2 of
10  * the License, or (at your option) any later version.
11  *
12  * This program is distributed in the hope that it will be useful
13  * but WITHOUT ANY WARRANTY; without even the implied warranty of
14  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
15  * GNU General Public License for more details.
16  *
17  * You should have received a copy of the GNU General Public License
18  * along with Proview. If not, see <http://www.gnu.org/licenses/>
19  *
20  * Linking Proview statically or dynamically with other modules is
21  * making a combined work based on Proview. Thus, the terms and
22  * conditions of the GNU General Public License cover the whole
23  * combination.
24  *
25  * In addition, as a special exception, the copyright holders of
26  * Proview give you permission to, from the build function in the
27  * Proview Configurator, combine Proview with modules generated by the
28  * Proview PLC Editor to a PLC program, regardless of the license
29  * terms of these modules. You may copy and distribute the resulting
30  * combined work under the terms of your choice, provided that every
31  * copy of the combined work is accompanied by a complete copy of
32  * the source code of Proview (the version used to produce the
33  * combined work), being distributed under the terms of the GNU
34  * General Public License plus this exception.
35  **/
36
37 #include "pwr.h"
38 #include "rt_io_base.h"
39
40 // Import methods registered in the classeditor
41 pwr_dImport pwr_BindIoUserMethods(Pl_Module);
42 pwr_dImport pwr_BindIoUserMethods(Pl_MN);
43
44 pwr_BindIoUserClasses(User) = {
45   pwr_BindIoUserClass(Pl_Module),
46   pwr_BindIoUserClass(Pl_MN),
47   pwr_NullClass
48 };
49
50
```

```
/*
 * Proview   Open Source Process Control.
 * Copyright (C) 2005-2012 SSAB EMEA AB.
 *
 * This file is part of Proview.
 *
 * This program is free software; you can redistribute it and/or
 * modify it under the terms of the GNU General Public License as
 * published by the Free Software Foundation, either version 2 of
 * the License, or (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with Proview. If not, see <http://www.gnu.org/licenses/>
 *
 * Linking Proview statically or dynamically with other modules is
 * making a combined work based on Proview. Thus, the terms and
 * conditions of the GNU General Public License cover the whole
 * combination.
 *
 * In addition, as a special exception, the copyright holders of
 * Proview give you permission to, from the build function in the
 * Proview Configurator, combine Proview with modules generated by the
 * Proview PLC Editor to a PLC program, regardless of the license
 * terms of these modules. You may copy and distribute the resulting
 * combined work under the terms of your choice, provided that every
 * copy of the combined work is accompanied by a complete copy of
 * the source code of Proview (the version used to produce the
 * combined work), being distributed under the terms of the GNU
 * General Public License plus this exception.
 **/


 #ifndef rt_io_pl_h
#define rt_io_pl_h

typedef struct {
  unsigned long channel;
  unsigned long board;
  unsigned int diag_cnt;
  unsigned int diag_interval;
  unsigned int dev_init;
  unsigned int dev_init_cnt;
  unsigned int dev_init_limit;
  int softlimit_logged;
  int input_area_size;
  int output_area_size;
  void *input_area;
  void *output_area;
} io_sLocalPl_MN;


typedef struct {
  int byte_ordering;
```

```
59    int float_representation;
60  } io_sLocalPl_CN;
61
62  #endif
63
```

```
 1 /*
 2  * Proview   Open Source Process Control.
 3  * Copyright (C) 2005-2012 SSAB EMEA AB.
 4  *
 5  * This file is part of Proview.
 6  *
 7  * This program is free software; you can redistribute it and/or
 8  * modify it under the terms of the GNU General Public License as
 9  * published by the Free Software Foundation, either version 2 of
10  * the License, or (at your option) any later version.
11  *
12  * This program is distributed in the hope that it will be useful
13  * but WITHOUT ANY WARRANTY; without even the implied warranty of
14  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
15  * GNU General Public License for more details.
16  *
17  * You should have received a copy of the GNU General Public License
18  * along with Proview. If not, see <http://www.gnu.org/licenses/>
19  *
20  * Linking Proview statically or dynamically with other modules is
21  * making a combined work based on Proview. Thus, the terms and
22  * conditions of the GNU General Public License cover the whole
23  * combination.
24  *
25  * In addition, as a special exception, the copyright holders of
26  * Proview give you permission to, from the build function in the
27  * Proview Configurator, combine Proview with modules generated by the
28  * Proview PLC Editor to a PLC program, regardless of the license
29  * terms of these modules. You may copy and distribute the resulting
30  * combined work under the terms of your choice, provided that every
31  * copy of the combined work is accompanied by a complete copy of
32  * the source code of Proview (the version used to produce the
33  * combined work), being distributed under the terms of the GNU
34  * General Public License plus this exception.
35  **/
36
37
38 #include "pwr.h"
39 #include "rt_gdh.h"
40 #include "rt_io_base.h"
41 #include "rt_errh.h"
42
43
44 int main()
45 {
46
47     pwr_tStatus sts;
48     io_tCtx io_ctx;
49     float ctime = 1;
50
51     // Make connection to error handler
52     errh_Init("pwr_io", errh_eAnix_appl1);
53
54     // Make connection to realtime database
55     sts = gdh_Init("ra_io");
56     if (EVEN(sts)) {
57         errh_Fatal("ra_io aborted\n%m", sts);
58         exit(sts);
```

```
59      }
60
61      // Create context and call init functions of all agent,
62      // rack and cardobjects
63      sts = io_init(io_mProcess_User, pwr_cNObjid, &io_ctx, 1, ctime);
64      if ( EVEN(sts)) {
65          errh_Fatal("ra_io aborted\n%m", sts);
66          exit(sts);
67      }
68
69
70      // Call IoAgentRead() IoAgentWrite() IoCardRead() IoCardWrite()
71      // IoModuleRead() IoModuleWrite() forever
72      for (;;) {
73
74          sts = io_read(io_ctx);
75          sts = io_write(io_ctx);
76          EplTgtMilliSleep(100);
77      }
78  }
79
```

```
/*
 * Proview   $Id$
 * Copyright (C) 2005 SSAB Oxelösund.
 *
 * This program is free software; you can redistribute it and/or
 * modify it under the terms of the GNU General Public License as
 * published by the Free Software Foundation, either version 2 of
 * the License, or (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with the program, if not, write to the Free Software
 * Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.
 */

 /*****************************************************************************

  (c) SYSTEC electronic GmbH, D-07973 Greiz, August-Bebel-Str. 29
      www.systec-electronic.com

  (c) Bernecker + Rainer Industrie-Elektronik Ges.m.b.H.
      B&R Strasse 1, A-5142 Eggelsberg
      www.br-automation.com

  License:

    Redistribution and use in source and binary forms, with or without
    modification, are permitted provided that the following conditions
    are met:

    1. Redistributions of source code must retain the above copyright
       notice, this list of conditions and the following disclaimer.

    2. Redistributions in binary form must reproduce the above copyright
       notice, this list of conditions and the following disclaimer in the
       documentation and/or other materials provided with the distribution.

    3. Neither the name of the copyright holders nor the names of its
       contributors may be used to endorse or promote products derived
       from this software without prior written permission. For written
       permission, please contact info@systec-electronic.com.

    THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
    "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
    LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS
    FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE
    COPYRIGHT HOLDERS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT,
    INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING,
    BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
    LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER
    CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
    LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN
    ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
    POSSIBILITY OF SUCH DAMAGE.
```

```
59
60      Severability Clause:
61
62          If a provision of this License is or becomes illegal, invalid or
63          unenforceable in any jurisdiction, that shall not affect:
64          1. the validity or enforceability in that jurisdiction of any other
65             provision of this License; or
66          2. the validity or enforceability in other jurisdictions of that or
67             any other provision of this License.
68
69  ****************************************************************************/
70  #include "pwr.h"
71  #include "pwr_basecomponentclasses.h"
72  #include "rt_io_base.h"
73  #include "rt_io_agent_init.h"
74  #include "rt_io_agent_close.h"
75  #include "rt_io_msg.h"
76
77  #include "pwr_cvolpltestclasses.h"
78
79  #include "rt_io_m_pl.h"
80
81
82  #include "Epl.h"
83
84  #include <stdio.h>
85  #include <unistd.h>
86  #include <sys/types.h>
87  #include <sys/socket.h>
88  #include <sys/select.h>
89  #include <sys/ioctl.h>
90  #include <netinet/in.h>
91  #include <net/if.h>
92  #include <string.h>
93  #include <termios.h>
94  #include <pthread.h>
95  #include <sys/syscall.h>
96  #include <sys/resource.h>
97  #include <errno.h>
98
99  #include <sys/stat.h>
100 #include <fcntl.h>
101 #include <signal.h>
102 #include <time.h>
103 #include <stdarg.h>
104
105
106 #include <pcap.h>
107
108 #include "EplTgtConio.h"
109
110
111 /****************************************************************************/
112 /*                                                                          */
113 /*                                                                          */
114 /*          G L O B A L   D E F I N I T I O N S                             */
115 /*                                                                          */
116 /*                                                                          */
```

```
117 /****************************************************************************/
118
119 //---------------------------------------------------------------------------
120 // const defines
121 //---------------------------------------------------------------------------
122 #define IP_ADDR     0xc0a86401          // 192.168.100.1
123 #define SUBNET_MASK 0xFFFFFF00          // 255.255.255.0 "
124
125 //---------------------------------------------------------------------------
126 // module global vars
127 //---------------------------------------------------------------------------
128 CONST BYTE abMacAddr[] = {0x00, 0x00, 0x00, 0x00, 0x00, 0x00};
129 static unsigned int uiCycleLen_g = 0;
130 static unsigned int uiCurCycleLen_g = 0;
131 tEplKernel EplRet = kEplSuccessful;
132
133
134 // process image stuff
135 static void *AppProcessImageIn_g;
136 static void *AppProcessImageOut_g;
137 static tEplApiProcessImageCopyJob AppProcessImageCopyJob_g;
138
139
140 /*---------------------------------------------------------------------------
    */
141
142 //---------------------------------------------------------------------------
143 // local function prototypes
144 //---------------------------------------------------------------------------
145
146 // This function is the entry point for your object dictionary. It is defined
147 // in OBJDICT.C by define EPL_OBD_INIT_RAM_NAME. Use this function name to def
    ine
148 // this function prototype here. If you want to use more than one Epl
149 // instances then the function name of each object dictionary has to differ.
150 tEplKernel PUBLIC  EplObdInitRam (tEplObdInitParam MEM* pInitParam_p);
151 tEplKernel PUBLIC AppCbEvent(
152     tEplApiEventType        EventType_p,    // IN: event type (enum)
153     tEplApiEventArg*        pEventArg_p,    // IN: event argument (union)
154     void GENERIC*           pUserArg_p);
155 tEplKernel PUBLIC AppCbSync(void);
156 tEplKernel PUBLIC AppInit(void);
157
158
159 /*---------------------------------------------------------------------------
    *\
160    Init method for the Powerlink module
161 \*---------------------------------------------------------------------------
    */
162 static pwr_tStatus IoAgentInit (io_tCtx ctx, io_sAgent *ap) {
163     io_sLocalPl_MN *local;
164     int sts;
165     pwr_sClass_Pl_MN *op = (pwr_sClass_Pl_MN *)ap->op;
166
167     local = (io_sLocalPl_MN *) calloc( 1, sizeof(io_sLocalPl_MN));
168     ap->Local = local;
169
170
```

```
171     static tEplApiInitParam EplApiInitParam;
172     char* pszCdcFilename_g = op->CDCfile;
173     char* sHostname = malloc(1023);
174 
175     gethostname(sHostname, 1023);
176 
177     // Init the I/O area
178     unsigned int input_area_offset = 0;
179     unsigned int input_area_chansize = 0;
180     unsigned int output_area_offset = 0;
181     unsigned int output_area_chansize = 0;
182     io_sRack *rp;
183     io_sCard *cp;
184 
185     for ( rp = ap->racklist; rp; rp = rp->next) {
186         rp->Local =  calloc( 1, sizeof(io_sLocalPl_CN));
187         rp->MethodDisabled = 1;
188 
189         // Show device offset and size
190         if ( rp->Class == pwr_cClass_Pl_CN && rp->op) {
191             ((pwr_sClass_Pl_CN *)rp->op)->InputAreaOffset = input_area_offset
    + input_area_chansize;
192             ((pwr_sClass_Pl_CN *)rp->op)->OutputAreaOffset = output_area_offse
    t + output_area_chansize;
193         }
194 
195         // Get byte ordering
196         pwr_tAName name;
197         pwr_tEnum byte_ordering;
198 
199         strcpy( name, rp->Name);
200         strcat( name, ".ByteOrdering");
201         sts = gdh_GetObjectInfo( name, &byte_ordering, sizeof(byte_ordering));
202         if ( ODD(sts))
203             ((io_sLocalPl_CN *)rp->Local)->byte_ordering = byte_ordering;
204         else
205             ((io_sLocalPl_CN *)rp->Local)->byte_ordering =
206         pwr_eByteOrderingEnum_LittleEndian;
207 
208         for ( cp = rp->cardlist; cp; cp = cp->next) {
209 
210             cp->MethodDisabled = 1;
211 
212             // Show module offset and size
213             if ( cp->Class == pwr_cClass_Pl_Module && cp->op) {
214                 ((pwr_sClass_Pl_Module *)cp->op)->InputAreaOffset = input_area
    _offset + input_area_chansize;
215                 ((pwr_sClass_Pl_Module *)cp->op)->OutputAreaOffset = output_ar
    ea_offset + output_area_chansize;
216             }
217 
218             io_bus_card_init( ctx, cp, &input_area_offset, &input_area_chansiz
    e, &output_area_offset, &output_area_chansize, byte_ordering);
219 
220             // Show module offset and size
221             if ( cp->Class == pwr_cClass_Pl_Module && cp->op) {
222                 ((pwr_sClass_Pl_Module *)cp->op)->InputAreaSize = input_area_o
    ffset + input_area_chansize - ((pwr_sClass_Pl_Module *)cp->op)->InputAreaOffse
```

```
;
223                 ((pwr_sClass_Pl_Module *)cp->op)->OutputAreaSize = output_area
    _offset + output_area_chansize - ((pwr_sClass_Pl_Module *)cp->op)->OutputAreaO
    ffset;
224             }
225         }
226
227         // Show device offset and size
228         if ( rp->Class == pwr_cClass_Pl_CN && rp->op) {
229             ((pwr_sClass_Pl_CN *)rp->op)->InputAreaSize = input_area_offset +
    input_area_chansize - ((pwr_sClass_Pl_CN *)rp->op)->InputAreaOffset;
230             ((pwr_sClass_Pl_CN *)rp->op)->OutputAreaSize = output_area_offset
    + output_area_chansize - ((pwr_sClass_Pl_CN *)rp->op)->OutputAreaOffset;
231         }
232     }
233
234     // This is the calculated in- and outputarea size
235     local->input_area_size = input_area_offset + input_area_chansize;
236     local->output_area_size = output_area_offset + output_area_chansize;
237
238
239     // Initialize target specific stuff
240     EplTgtInit();
241
242     EPL_MEMSET(&EplApiInitParam, 0, sizeof (EplApiInitParam));
243     EplApiInitParam.m_uiSizeOfStruct = sizeof (EplApiInitParam);
244
245     // Get devicename from attribute in agent
246     EplApiInitParam.m_HwParam.m_pszDevName = op->Device;
247
248     // Get nodeid from attribute in agent
249     EplApiInitParam.m_uiNodeId = op->NodeId;
250     EplApiInitParam.m_dwIpAddress = (0xFFFFFF00 & IP_ADDR) | EplApiInitParam.m
    _uiNodeId;
251
252     // write 00:00:00:00:00:00 to MAC address, so that the driver uses the rea
    l hardware address
253     EPL_MEMCPY(EplApiInitParam.m_abMacAddress, abMacAddr, sizeof (EplApiInitPa
    ram.m_abMacAddress));
254
255     EplApiInitParam.m_fAsyncOnly = FALSE;
256
257     EplApiInitParam.m_dwFeatureFlags            = -1;
258     // required for error detection
259     EplApiInitParam.m_dwCycleLen                = uiCycleLen_g;
260     // const
261     EplApiInitParam.m_uiIsochrTxMaxPayload      = 256;
262     // const
263     EplApiInitParam.m_uiIsochrRxMaxPayload      = 256;
264     // const; only required for IdentRes
265     EplApiInitParam.m_dwPresMaxLatency          = 50000;
266     // required for initialisation (+28 bytes)
267     EplApiInitParam.m_uiPreqActPayloadLimit     = 36;
268     // required for initialisation of Pres frame (+28 bytes)
269     EplApiInitParam.m_uiPresActPayloadLimit     = 36;
270     // const; only required for IdentRes
271     EplApiInitParam.m_dwAsndMaxLatency          = 150000;
272     // required for error detection
```

```
273     EplApiInitParam.m_uiMultiplCycleCnt         = 0;
274     // required to set up max frame size
275     EplApiInitParam.m_uiAsyncMtu                = 1500;
276     // required for sync
277     EplApiInitParam.m_uiPrescaler               = 2;
278     EplApiInitParam.m_dwLossOfFrameTolerance    = 500000;
279     EplApiInitParam.m_dwAsyncSlotTimeout        = 3000000;
280     EplApiInitParam.m_dwWaitSocPreq             = 150000;
281     // NMT_DeviceType_U32
282     EplApiInitParam.m_dwDeviceType              = -1;
283     // NMT_IdentityObject_REC.VendorId_U32
284     EplApiInitParam.m_dwVendorId                = -1;
285     // NMT_IdentityObject_REC.ProductCode_U32
286     EplApiInitParam.m_dwProductCode             = -1;
287     // NMT_IdentityObject_REC.RevisionNo_U32
288     EplApiInitParam.m_dwRevisionNumber          = -1;
289     // NMT_IdentityObject_REC.SerialNo_U32
290     EplApiInitParam.m_dwSerialNumber            = -1;
291
292     EplApiInitParam.m_dwSubnetMask              = SUBNET_MASK;
293     EplApiInitParam.m_dwDefaultGateway          = 0;
294     EPL_MEMCPY(EplApiInitParam.m_sHostname, sHostname, sizeof(EplApiInitParam.
    m_sHostname));
295     EplApiInitParam.m_uiSyncNodeId              = EPL_C_ADR_SYNC_ON_SOA;
296     EplApiInitParam.m_fSyncOnPrcNode            = FALSE;
297
298     // set callback functions
299     EplApiInitParam.m_pfnCbEvent = AppCbEvent;
300
301     EplApiInitParam.m_pfnObdInitRam = EplObdInitRam;
302     EplApiInitParam.m_pfnCbSync  = AppCbSync;
303
304
305     // initialize POWERLINK stack
306     EplRet = EplApiInitialize(&EplApiInitParam);
307     if(EplRet != kEplSuccessful)
308     {
309         errh_Error("EplApiInitialize() failed (Error:0x%x!\n", EplRet);
310         goto Exit;
311     }
312
313
314     EplRet = EplApiSetCdcFilename(pszCdcFilename_g);
315     if(EplRet != kEplSuccessful)
316     {
317         goto Exit;
318     }
319
320     // Allocate memory for the in- and outputareas
321     AppProcessImageIn_g = malloc(local->output_area_size);
322     AppProcessImageOut_g = malloc(local->input_area_size);
323
324
325     // Save pointer to in- and outputareas in THIS agent object
326     local->input_area = AppProcessImageOut_g;
327     local->output_area = AppProcessImageIn_g;
328
329     AppProcessImageCopyJob_g.m_fNonBlocking = FALSE;
```

```
330     AppProcessImageCopyJob_g.m_uiPriority = 0;
331     AppProcessImageCopyJob_g.m_In.m_pPart = AppProcessImageIn_g;
332     AppProcessImageCopyJob_g.m_In.m_uiOffset = 0;
333     AppProcessImageCopyJob_g.m_In.m_uiSize = local->output_area_size;
334     AppProcessImageCopyJob_g.m_Out.m_pPart = AppProcessImageOut_g;
335     AppProcessImageCopyJob_g.m_Out.m_uiOffset = 0;
336     AppProcessImageCopyJob_g.m_Out.m_uiSize = local->input_area_size;
337
338     EplRet = EplApiProcessImageAlloc(local->output_area_size, local->input_are
    a_size, 2, 2);
339     if (EplRet != kEplSuccessful)
340     {
341         goto Exit;
342     }
343
344     EplRet = EplApiProcessImageSetup();
345     if (EplRet != kEplSuccessful)
346     {
347         goto Exit;
348     }
349
350     // start processing
351     EplRet = EplApiExecNmtCommand(kEplNmtEventSwReset);
352     if (EplRet != kEplSuccessful)
353     {
354         IoAgentClose(NULL, NULL);
355         goto Exit;
356     }
357
358     errh_Success ("Powerlink init successfull");
359     return IO__SUCCESS;
360
361
362 Exit:
363     errh_Error("IoCardInit: returns 0x%X\n", EplRet);
364     op->EplStatus = EplRet;
365     return IO__SUCCESS;
366
367 }
368
369
370 //============================================================================//
371 //                                                                            //
372 //          P R I V A T E   F U N C T I O N S                                 //
373 //                                                                            //
374 //============================================================================//
375
376
377 //------------------------------------------------------------------------------
378 //
379 // Function:    AppCbEvent
380 //
381 // Description: event callback function called by EPL API layer within
382 //              user part (low priority).
383 //
384 // Parameters:  EventType_p     = event type
385 //              pEventArg_p     = pointer to union, which describes
386 //                                the event in detail
```

```
387 //              pUserArg_p      = user specific argument
388 //
389 // Returns:     tEplKernel      = error code,
390 //                                kEplSuccessful = no error
391 //                                kEplReject = reject further processing
392 //                                otherwise = post error event to API layer
393 //
394 // State:
395 //
396 //---------------------------------------------------------------------------
397 tEplKernel PUBLIC AppCbEvent
398 (
399     tEplApiEventType        EventType_p,    // IN: event type (enum)
400     tEplApiEventArg*        pEventArg_p,    // IN: event argument (union)
401     void GENERIC*           pUserArg_p    //__attribute((unused))
402 )
403 {
404
405     UINT                uiVarLen;
406
407     UNUSED_PARAMETER(pUserArg_p);
408
409     // check if NMT_GS_OFF is reached
410     switch (EventType_p)
411     {
412         case kEplApiEventNmtStateChange:
413         {
414             switch (pEventArg_p->m_NmtStateChange.m_NewNmtState)
415             {
416                 case kEplNmtGsOff:
417                 {   // NMT state machine was shut down,
418                     // because of user signal (CTRL-C) or critical EPL stack e
    rror
419                     // -> also shut down EplApiProcess() and main()
420                     EplRet = kEplShutdown;
421
422                     errh_Fatal("Event:kEplNmtGsOff originating event = 0x%X (%
    s)\n", pEventArg_p->m_NmtStateChange.m_NmtEvent,
423                                EplGetNmtEventStr(pEventArg_p->m_NmtStateChange.m
    _NmtEvent));
424
425                     break;
426                 }
427
428                 case kEplNmtGsResetCommunication:
429                 {
430                     // continue
431                 }
432
433                 case kEplNmtGsResetConfiguration:
434                 {
435                     if (uiCycleLen_g != 0)
436                     {
437                         EplRet = EplApiWriteLocalObject(0x1006, 0x00, &uiCycle
    Len_g, sizeof (uiCycleLen_g));
438                         uiCurCycleLen_g = uiCycleLen_g;
439                     }
440                     else
```

```
441                     {
442                         uiVarLen = sizeof(uiCurCycleLen_g);
443                         EplApiReadLocalObject(0x1006, 0x00, &uiCurCycleLen_g,
    &uiVarLen);
444                     }
445                     // continue
446                 }
447
448                 case kEplNmtMsPreOperational1:
449                 {
450                     errh_Info("AppCbEvent(0x%X) originating event = 0x%X (%s)\
    n", pEventArg_p->m_NmtStateChange.m_NewNmtState, pEventArg_p->m_NmtStateChange
    .m_NmtEvent, EplGetNmtEventStr(pEventArg_p->m_NmtStateChange.m_NmtEvent));
451
452                     // continue
453                 }
454
455                 case kEplNmtGsInitialising:
456                 case kEplNmtGsResetApplication:
457                 case kEplNmtMsNotActive:
458                 case kEplNmtCsNotActive:
459                 case kEplNmtCsPreOperational1:
460                 {
461                     break;
462                 }
463
464                 case kEplNmtCsOperational:
465                 case kEplNmtMsOperational:
466                 {
467                     break;
468                 }
469
470                 default:
471                 {
472                     break;
473                 }
474             }
475
476             break;
477         }
478
479         case kEplApiEventCriticalError:
480         case kEplApiEventWarning:
481         {   // error or warning occurred within the stack or the application
482             // on error the API layer stops the NMT state machine
483
484             errh_Error( "%s(Err/Warn): Source = %s (%02X) EplError = %s (0x%03
    X)\n",
485                         __func__,
486                         EplGetEventSourceStr(pEventArg_p->m_InternalError.m_Ev
    entSource),
487                         pEventArg_p->m_InternalError.m_EventSource,
488                         EplGetEplKernelStr(pEventArg_p->m_InternalError.m_EplE
    rror),
489                         pEventArg_p->m_InternalError.m_EplError);
490
491             // check additional argument
492             switch (pEventArg_p->m_InternalError.m_EventSource)
```

```
493             {
494                 case kEplEventSourceEventk:
495                 case kEplEventSourceEventu:
496                 {   // error occurred within event processing
497                     // either in kernel or in user part
498
499                     errh_Error(" OrgSource = %s %02X\n",  EplGetEventSourceStr
    (pEventArg_p->m_InternalError.m_Arg.m_EventSource),
500                                                           pEventArg_p->m_Interna
    lError.m_Arg.m_EventSource);
501
502                     break;
503                 }
504
505                 case kEplEventSourceDllk:
506                 {   // error occurred within the data link layer (e.g. interru
    pt processing)
507                     // the DWORD argument contains the DLL state and the NMT e
    vent
508
509                     errh_Error(" val = %X\n", pEventArg_p->m_InternalError.m_A
    rg.m_dwArg);
510
511                     break;
512                 }
513
514                 default:
515                 {
516                     break;
517                 }
518             }
519             break;
520         }
521
522         case kEplApiEventHistoryEntry:
523         {   // new history entry
524
525             errh_Info("%s(HistoryEntry): Type=0x%04X Code=0x%04X (0x%02X %02X
    %02X %02X %02X %02X %02X %02X)\n",
526                     __func__,
527                     pEventArg_p->m_ErrHistoryEntry.m_wEntryType,
528                     pEventArg_p->m_ErrHistoryEntry.m_wErrorCode,
529                     (WORD) pEventArg_p->m_ErrHistoryEntry.m_abAddInfo[0],
530                     (WORD) pEventArg_p->m_ErrHistoryEntry.m_abAddInfo[1],
531                     (WORD) pEventArg_p->m_ErrHistoryEntry.m_abAddInfo[2],
532                     (WORD) pEventArg_p->m_ErrHistoryEntry.m_abAddInfo[3],
533                     (WORD) pEventArg_p->m_ErrHistoryEntry.m_abAddInfo[4],
534                     (WORD) pEventArg_p->m_ErrHistoryEntry.m_abAddInfo[5],
535                     (WORD) pEventArg_p->m_ErrHistoryEntry.m_abAddInfo[6],
536                     (WORD) pEventArg_p->m_ErrHistoryEntry.m_abAddInfo[7]);
537
538             break;
539         }
540
541 #if (((EPL_MODULE_INTEGRATION) & (EPL_MODULE_NMT_MN)) != 0)
542         case kEplApiEventNode:
543         {
544             // check additional argument
```

```
545             switch (pEventArg_p->m_Node.m_NodeEvent)
546             {
547                 case kEplNmtNodeEventCheckConf:
548                 {
549                     errh_Info("%s(Node=0x%X, CheckConf)\n", __func__, pEventAr
    g_p->m_Node.m_uiNodeId);
550
551                     break;
552                 }
553
554                 case kEplNmtNodeEventUpdateConf:
555                 {
556                     errh_Info("%s(Node=0x%X, UpdateConf)\n", __func__, pEventA
    rg_p->m_Node.m_uiNodeId);
557                     break;
558                 }
559
560                 case kEplNmtNodeEventNmtState:
561                 {
562                     errh_Info("%s(Node=0x%X, NmtState=%s)\n", __func__, pEvent
    Arg_p->m_Node.m_uiNodeId, EplGetNmtStateStr(pEventArg_p->m_Node.m_NmtState));
563
564                     break;
565                 }
566
567                 case kEplNmtNodeEventError:
568                 {
569                     errh_Error("%s (Node=0x%X): Error = %s (0x%.4X)\n", __func
    __,
570                                pEventArg_p->m_Node.m_uiNodeId,
571                                EplGetEmergErrCodeStr(pEventArg_p->m_Node.m_wError
    Code),
572                                pEventArg_p->m_Node.m_wErrorCode);
573
574                     break;
575                 }
576
577                 case kEplNmtNodeEventFound:
578                 {
579                     errh_Info("%s(Node=0x%X, Found)\n", __func__, pEventArg_p-
    >m_Node.m_uiNodeId);
580
581                     break;
582                 }
583
584                 default:
585                 {
586                     break;
587                 }
588             }
589             break;
590         }
591
592 #endif
593
594 #if (((EPL_MODULE_INTEGRATION) & (EPL_MODULE_CFM)) != 0)
595         case kEplApiEventCfmProgress:
596         {
```

```
597             errh_Info("%s(Node=0x%X, CFM-Progress: Object 0x%X/%u, ", __func__
    , pEventArg_p->m_CfmProgress.m_uiNodeId, pEventArg_p->m_CfmProgress.m_uiObject
    Index, pEventArg_p->m_CfmProgress.m_uiObjectSubIndex);
598             errh_Info("%lu/%lu Bytes", (ULONG) pEventArg_p->m_CfmProgress.m_dw
    BytesDownloaded, (ULONG) pEventArg_p->m_CfmProgress.m_dwTotalNumberOfBytes);
599             if ((pEventArg_p->m_CfmProgress.m_dwSdoAbortCode != 0)
600                 || (pEventArg_p->m_CfmProgress.m_EplError != kEplSuccessful))
601             {
602                 errh_Error(" -> SDO Abort=0x%lX, Error=0x%X)\n", (unsigned lon
    g) pEventArg_p->m_CfmProgress.m_dwSdoAbortCode,
603                                                                  pEventArg_p->m_C
    fmProgress.m_EplError);
604             }
605             else
606             {
607
608             }
609             break;
610         }
611
612         case kEplApiEventCfmResult:
613         {
614             switch (pEventArg_p->m_CfmResult.m_NodeCommand)
615             {
616                 case kEplNmtNodeCommandConfOk:
617                 {
618                     errh_Info("%s(Node=0x%X, ConfOk)\n", __func__, pEventArg_p
    ->m_CfmResult.m_uiNodeId);
619                     break;
620                 }
621
622                 case kEplNmtNodeCommandConfErr:
623                 {
624                     errh_Info("%s(Node=0x%X, ConfErr)\n", __func__, pEventArg_
    p->m_CfmResult.m_uiNodeId);
625                     break;
626                 }
627
628                 case kEplNmtNodeCommandConfReset:
629                 {
630                     errh_Info("%s(Node=0x%X, ConfReset)\n", __func__, pEventAr
    g_p->m_CfmResult.m_uiNodeId);
631                     break;
632                 }
633
634                 case kEplNmtNodeCommandConfRestored:
635                 {
636                     errh_Info("%s(Node=0x%X, ConfRestored)\n", __func__, pEven
    tArg_p->m_CfmResult.m_uiNodeId);
637                     break;
638                 }
639
640                 default:
641                 {
642                     errh_Info("%s(Node=0x%X, CfmResult=0x%X)\n", __func__, pEv
    entArg_p->m_CfmResult.m_uiNodeId, pEventArg_p->m_CfmResult.m_NodeCommand);
643                     break;
644                 }
```

```
645             }
646             break;
647         }
648 #endif
649
650         default:
651             break;
652     }
653
654
655     return EplRet;
656 }
657
658
659 //------------------------------------------------------------------------
660 //
661 // Function:    AppCbSync
662 //
663 // Description: sync event callback function called by event module within
664 //              kernel part (high priority).
665 //              This function sets the outputs, reads the inputs and runs
666 //              the control loop.
667 //
668 // Parameters:  void
669 //
670 // Returns:     tEplKernel      = error code,
671 //                                kEplSuccessful = no error
672 //                                otherwise = post error event to API layer
673 //
674 // State:
675 //
676 //------------------------------------------------------------------------
677 tEplKernel PUBLIC AppCbSync(void)
678 {
679
680
681     EplRet = EplApiProcessImageExchange(&AppProcessImageCopyJob_g);
682
683
684     if (EplRet != kEplSuccessful)
685     {
686         return EplRet;
687     }
688
689
690     return EplRet;
691 }
692
693
694 /*------------------------------------------------------------------------
    *\
695    Close method for the Powerlink master
696 \*------------------------------------------------------------------------
    */
697 static pwr_tStatus IoAgentClose (
698   io_tCtx    ctx,
699   io_sAgent    *ap
700 )
```

```
701 {
702     pwr_sClass_Pl_MN *op = (pwr_sClass_Pl_MN *)ap->op;
703 
704     // halt the NMT state machine
705     // so the processing of POWERLINK frames stops
706     EplRet = EplApiExecNmtCommand(kEplNmtEventSwitchOff);
707 
708     // delete process image
709     EplRet = EplApiProcessImageFree();
710     // delete instance for all modules
711     EplRet = EplApiShutdown();
712 
713     printf("IoAgentInit(): returns 0x%X\n", EplRet);
714 
715     op->EplStatus = EplRet;
716     return IO__SUCCESS;
717 
718 }
719 /*-------------------------------------------------------------------------
    *\
720    Read method for the Powerlink master
721 \*-------------------------------------------------------------------------
    */
722 static pwr_tStatus IoAgentRead( io_tCtx ctx, io_sAgent *ap)
723 {
724     io_sLocalPl_MN *local = (io_sLocalPl_MN *)ap->Local;
725     pwr_sClass_Pl_MN *op = (pwr_sClass_Pl_MN *)ap->op;
726     io_sRack *rp;
727     io_sCard *cp;
728 
729 
730     for ( rp = ap->racklist; rp; rp = rp->next) {
731         for ( cp = rp->cardlist; cp; cp = cp->next) {
732             io_bus_card_read( ctx, rp, cp, local->input_area, 0, ((io_sLocalPl
    _CN *)rp->Local)->byte_ordering, pwr_eFloatRepEnum_FloatIEEE);
733         }
734     }
735 
736 
737     op->EplStatus = EplRet;
738     return IO__SUCCESS;
739 }
740 /*-------------------------------------------------------------------------
    *\
741    Write method for the Powerlink master
742 \*-------------------------------------------------------------------------
    */
743 static pwr_tStatus IoAgentWrite( io_tCtx ctx, io_sAgent *ap)
744 {
745     io_sLocalPl_MN *local = (io_sLocalPl_MN *)ap->Local;
746     pwr_sClass_Pl_MN *op = (pwr_sClass_Pl_MN *)ap->op;
747     io_sRack *rp;
748     io_sCard *cp;
749 
750     for ( rp = ap->racklist; rp; rp = rp->next) {
751         for ( cp = rp->cardlist; cp; cp = cp->next) {
752             io_bus_card_write( ctx, cp, local->output_area, ((io_sLocalPl_CN *
    )rp->Local)->byte_ordering, pwr_eFloatRepEnum_FloatIEEE);
```

```
753       }
754     }
755
756     op->EplStatus = EplRet;
757     return IO__SUCCESS;
758
759 }
760
761 /*-----------------------------------------------------------------------------
    *\
762   Every method to be exported to the workbench should be registred here.
763 \*-----------------------------------------------------------------------------
    */
764
765 pwr_dExport pwr_BindIoUserMethods(Pl_MN) = {
766   pwr_BindIoMethod(IoAgentInit),
767   pwr_BindIoMethod(IoAgentClose),
768   pwr_BindIoMethod(IoAgentRead),
769   pwr_BindIoMethod(IoAgentWrite),
770   pwr_NullMethod
771 };
772
```

```
 1 /*
 2  * Proview   $Id$
 3  * Copyright (C) 2005 SSAB Oxelösund.
 4  *
 5  * This program is free software; you can redistribute it and/or
 6  * modify it under the terms of the GNU General Public License as
 7  * published by the Free Software Foundation, either version 2 of
 8  * the License, or (at your option) any later version.
 9  *
10  * This program is distributed in the hope that it will be useful
11  * but WITHOUT ANY WARRANTY; without even the implied warranty of
12  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
13  * GNU General Public License for more details.
14  *
15  * You should have received a copy of the GNU General Public License
16  * along with the program, if not, write to the Free Software
17  * Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.
18  */
19 #include "pwr.h"
20 #include "pwr_basecomponentclasses.h"
21 #include "rt_io_base.h"
22 #include "rt_io_card_init.h"
23 #include "rt_io_card_close.h"
24 #include "rt_io_card_read.h"
25 #include "rt_io_card_write.h"
26 #include "rt_io_msg.h"
27
28 #include "pwr_cvolpltestclasses.h"
29
30
31 /*-----------------------------------------------------------------------------
    *\
32    Init method for the Powerlink module
33 \*-----------------------------------------------------------------------------
    */
34 static pwr_tStatus IoCardInit (
35   io_tCtx    ctx,
36   io_sAgent    *ap,
37   io_sRack    *rp,
38   io_sCard    *cp
39 )
40 {
41
42     return IO__SUCCESS;
43
44 }
45
46
47 /*-----------------------------------------------------------------------------
    *\
48    Read method for the Powerlink module
49 \*-----------------------------------------------------------------------------
    */
50 static pwr_tStatus IoCardRead (
51   io_tCtx    ctx,
52   io_sAgent    *ap,
53   io_sRack    *rp,
54   io_sCard    *cp
```

```
55 )
56 {
57
58     return IO__SUCCESS;
59 }
60
61
62 /*-------------------------------------------------------------------------
   *\
63    Write method for the Powerlink module
64 \*-------------------------------------------------------------------------
   */
65 static pwr_tStatus IoCardWrite (
66   io_tCtx    ctx,
67   io_sAgent    *ap,
68   io_sRack    *rp,
69   io_sCard    *cp
70 )
71 {
72
73
74     return IO__SUCCESS;
75
76 }
77
78 /*-------------------------------------------------------------------------
   *\
79    Close method for the Powerlink module
80 \*-------------------------------------------------------------------------
   */
81 static pwr_tStatus IoCardClose (
82   io_tCtx    ctx,
83   io_sAgent    *ap,
84   io_sRack    *rp,
85   io_sCard    *cp
86 )
87 {
88
89     return IO__SUCCESS;
90
91 }
92
93
94 /*-------------------------------------------------------------------------
   *\
95    Every method to be exported to the workbench should be registred here.
96 \*-------------------------------------------------------------------------
   */
97 pwr_dExport pwr_BindIoUserMethods(Pl_Module) = {
98   pwr_BindIoMethod(IoCardInit),
99   pwr_BindIoMethod(IoCardRead),
100  pwr_BindIoMethod(IoCardWrite),
101  pwr_BindIoMethod(IoCardClose),
102  pwr_NullMethod
103 };
104
```

Appendix F

Kompilering

För att bygga projektet används följande make-fil och opt-fil för länkning.

```
1  pltest_top : pltest
2
3  include $(pwr_exe)/pwrp_rules.mk
4
5  pltest_modules = $(pwrp_obj)/ra_io_m_pl_module.o \
6          $(pwrp_obj)/ra_io_m_pl_mn.o \
7                  $(pwrp_obj)/rt_io_user.o \
8          $(pwrp_obj)/ra_io.o \
9          $(pwrp_exe)/ra_io
10
11 # Main rule
12 pltest : $(pltest_modules)
13     @ echo "****** pltest modules built ******"
14
15
16 # Modules
17 $(pwrp_obj)/ra_io_m_pl_module.o : \
18 $(pwrp_appl)/ra_io_m_pl_module.c \
19       $(pwrp_inc)/pwr_cvolpltestclasses.h
20
21 $(pwrp_obj)/ra_io_m_pl_mn.o : \
22 $(pwrp_appl)/ra_io_m_pl_mn.c \
23     $(pwrp_inc)/pwr_cvolpltestclasses.h
24     gcc $(source) -g -c -o $(target) -Wall -DOS_LINUX -DOS_POSIX -I/home/pwrp/D
   ownloads/openPOWERLINK-V1.08.2/Include -I/home/pwrp/Downloads/openPOWERLINK-V1.
   08.2/SharedBuff -I/home/pwrp/Downloads/openPOWERLINK-V1.08.2/EplStack -I/home/p
   wrp/Downloads/openPOWERLINK-V1.08.2/Edrv -I/home/pwrp/Downloads/openPOWERLINK-V
   1.08.2/ObjDicts/CiA302-4_MN -I/home/pwrp/Downloads/openPOWERLINK-V1.08.2/Exampl
   es/X86/Generic/powerlink_user_lib -I$(pwrp_inc) -I$(pwr_inc)
25
26 $(pwrp_obj)/rt_io_user.o : $(pwrp_appl)/rt_io_user.c
27
28 $(pwrp_obj)/ra_io.o : $(pwrp_appl)/ra_io.c
29     gcc $(source) -g -c -o $(target) -Wall -DOS_LINUX -DOS_POSIX -DCONFIG_POWER
   LINK_USERSTACK -I/home/pwrp/Downloads/openPOWERLINK-V1.08.2/Include -I/home/pwr
   p/Downloads/openPOWERLINK-V1.08.2/SharedBuff -I /home/pwrp/Downloads/openPOWERL
   INK-V1.08.2/EplStack -I /home/pwrp/Downloads/openPOWERLINK-V1.08.2/Edrv -I/home
   /pwrp/Downloads/openPOWERLINK-V1.08.2/ObjDicts/CiA302-4_MN -I/home/pwrp/Downloa
   ds/openPOWERLINK-V1.08.2/Examples/X86/Generic/powerlink_user_lib -I$(pwrp_inc)
   -I$(pwr_inc)
30
31
32 $(pwrp_exe)/ra_io : $(pwrp_obj)/ra_io.o
33     g++ -o $(target) $(source) $(pwrp_obj)/rt_io_user.o $(pwrp_obj)/ra_io_m_pl_
   module.o $(pwrp_obj)/ra_io_m_pl_mn.o -L$(pwr_lib) -lpwr_rt -lpwr_cifx_dummy -lp
   wr_usb_dummy -lpwr_usbio_dummy -lpwr_pnak_dummy -lpwr_nodave_dummy -lpwr_rt -lp
   wr_co -lpwr_msg_dummy -lpcap -L /home/pwrp/Downloads/openPOWERLINK-V1.08.2/buil
   d/Examples/X86/Generic/powerlink_user_lib -lpowerlink /home/pwrp/Downloads/open
   POWERLINK-V1.08.2/build/Examples/X86/Generic/demo_mn_console/CMakeFiles/demo_mn
   _console.dir/__/__/__/__/ObjDicts/CiA302-4_MN/EplApiProcessImageSetup.c.o /home
   /pwrp/Downloads/openPOWERLINK-V1.08.2/build/Examples/X86/Generic/demo_mn_consol
   e/CMakeFiles/demo_mn_console.dir/__/__/__/__/EplStack/EplTgtConio.c.o -lpthread
    -lrt
34
```

```
1 $pwrp_obj/rt_io_user.o $pwrp_obj/ra_io_m_pl_module.o $pwrp_obj/ra_io_m_pl_mn.o -
  lpwr_rt -lpwr_cifx_dummy -lpwr_usb_dummy -lpwr_usbio_dummy -lpwr_pnak_dummy -lpw
  r_nodave_dummy -lpcap -L /home/pwrp/Downloads/openPOWERLINK-V1.08.2/build/Exampl
  es/X86/Generic/powerlink_user_lib -l powerlink /home/pwrp/Downloads/openPOWERLIN
  K-V1.08.2/build/Examples/X86/Generic/demo_mn_console/CMakeFiles/demo_mn_console.
  dir/__/__/__/__/ObjDicts/CiA302-4_MN/EplApiProcessImageSetup.c.o /home/pwrp/Down
  loads/openPOWERLINK-V1.08.2/build/Examples/X86/Generic/demo_mn_console/CMakeFile
  s/demo_mn_console.dir/__/__/__/__/EplStack/EplTgtConio.c.o
2
```