MŰEGYETEM 1782

DIPLOMATERV-FELADAT

Hegyi Balázs (F0LR8E)

szigorló villamosmérnök hallgató részére

AUTOSAR hálózatmenedzsment demonstrációs projekt fejlesztése

A modern gépjárművek biztonságtechnikai és kényelmi funkcióinak megvalósításában, környezetvédelmi jellemzőinek javításában stb. egyre jelentősebb szerepet kapnak a számítástechnikai megoldások. Ma egy prémium személyautó gyártójának közel száz elektronikus vezérlőegységből (ECU) és számos fedélzeti kommunikációs sínből kell kialakítani egy megbízhatóan működő elosztott rendszert, amely komoly algoritmus- és kommunikációtervezési, illetve munkaszervezési kihívást jelent. Az így adódó komplexitás uralására alakultak ki különféle szabványok, pl.: a megbízható kommunikáció biztosítására a CAN és FlexRay sínek, a valós idejű feladatok futtatására az OSEK operációs rendszer vagy a futási idejű monitorozást támogató XCP protokollcsalád.

A vezető autógyártók által 2002-ben életre hívott AUTOSAR konzorcium célja az, hogy ezen szakterületi szabványokra építve specifikáljon egy (i) *alapvető szolgáltatásstruktúrát,* amely eltakarja a hardver sajátosságait és támogatja az alkalmazási szoftver hordozhatóságát (base software stack, BSW), (ii) egy *modellezési nyelvet* az ECU-kon futó alkalmazási szoftver szabványos leírására (software component template), és (iii) az alkalmazások és BSW-k ECU-n belüli és ECU-k közti *transzparens kommunikációját* lehetővé tevő elosztott runtime szolgáltatást (RTE). Egy mai autóban megtalálható összes elektronikai egység működtetése már jelentős energiaszükséglettel bír, így a távlati célokban megjelennek olyan feladatok is, melyekkel az autó elektromos fogyasztása csökkenthető a passzív ECU-k kikapcsolásával. Ennek a problémának a megoldására szolgál az AUTOSAR szabványba foglalt hálózatmenedzsment funkcionalitás.

A jelölt feladata az AUTOSAR base software stackjére épülő, hálózatmenedzsment funkciók megfelelő működését demonstráló projekt elkészítése, amely a következő részfeladatokból épül fel:

- *Szoftver integráció: a cégnél rendelkezésre álló AUTOSAR szabványnak megfelelő BSW modulokból állítson össze olyan ECU-n is futtatható szoftvert, amely tartalmaz hálózatmenedzsment funkcionalitást is.*
- *Integráció tesztelése: készítsen olyan szoftver komponenseket, amelyekkel bizonyítható, hogy az összeállított rendszer hálózatmenedzsment funkciói megfelelően működnek. Ehhez implementáljon ECU-n futtatható PN master és PN slave applikációkat.*
- *Virtuális hálózat létrehozása: készítsen olyan PC oldali szoftvert, amely Fieldbus GW segítségével képes (i) egy virtuális részhálózatot szimulálni, (ii) értelmezni a buszról érkező hálózatmenedzsment üzeneteket, valamint (iii) futtatni az elkészített PN master és slave applikációkat.*

Tanszéki konzulens: Dr. Sujbert László docens
Külső konzulens: Dr. Pintér Gergely (ThyssenKrupp Presta Hungary Kft.)

Budapest, 2014. szeptember 25.

……
Dr. Jobbágy Ákos
tanszékvezető

Budapesti Műszaki és Gazdaságtudományi Egyetem
Villamosmérnöki és Informatikai Kar
Méréstechnika és Információs Rendszerek Tanszék

1117 Budapest, Magyar Tudósok krt. 2. I. ép. I.E.444.
Telefon: 463-2057, Fax: 463-4112
http://www.mit.bme.hu • e-mail: mitadm@mit.bme.hu

Budapesti Műszaki és Gazdaságtudományi Egyetem
Villamosmérnöki és Informatikai Kar

Hegyi Balázs

AUTOSAR hálózatmenedzsment demonstrációs projekt fejlesztése

KONZULENSEK

Dr. Pintér Gergely
(ThyssenKrupp Presta Hungary Kft.)

Dr. Sujbert László

BUDAPEST, 2015

HALLGATÓI NYILATKOZAT

Alulírott **Hegyi Balázs**, szigorló hallgató kijelentem, hogy ezt a diplomatervet meg nem engedett segítség nélkül, saját magam készítettem, csak a megadott forrásokat (szakirodalom, eszközök stb.) használtam fel. Minden olyan részt, melyet szó szerint, vagy azonos értelemben, de átfogalmazva más forrásból átvettem, egyértelműen, a forrás megadásával megjelöltem.

Hozzájárulok, hogy a jelen munkám alapadatait (szerző(k), cím, angol és magyar nyelvű tartalmi kivonat, készítés éve, konzulens(ek) neve) a BME VIK nyilvánosan hozzáférhető elektronikus formában, a munka teljes szövegét pedig az egyetem belső hálózatán keresztül (vagy hitelesített felhasználók számára) közzétegye. Kijelentem, hogy a benyújtott munka és annak elektronikus verziója megegyezik. Dékáni engedéllyel titkosított diplomatervek esetén a dolgozat szövege csak 3 év eltelte után válik hozzáférhetővé.

Kelt: Budapest, 2015. 05. 24.

...…….

Hegyi Balázs

Tartalomjegyzék

Kivonat

Napjaink gépjárműiben az elektromos vezérlőegységek (ECU) számának folyamatos növekedése tapasztalható. Az esetenként százas nagyságrendbe eső ECU-kból felépített komplex rendszer megtervezése egy igen komoly mérnöki kihívás. Ezt a feladatot elősegítendő a 2000-es évek elején megszületett az AUTOSAR szabványrendszer, amely különböző eszközökkel (modellezési nyelv, szoftverarchitektúra) segíti a jármű elektronikus rendszerének megtervezését.

A jármű elektronikus egységeinek bővülésével az elektromos teljesítmény felvétele növekszik, és ennek hatására a *gépjármű üzemanyag fogyasztása is megemelkedik*. Ez a nem kívánt hatás a *passzív elektronikus egységek kikapcsolásával csökkenthető*. A hagyományos megoldásokban az elektronikus egységek a jármű „gyújtás” (ignition) jelére kapcsolnak be illetve ki. Azonban ez a funkció a kommunikációs buszok segítségével is megoldható, amikor is a *hálózati aktivitás* kapcsolja be a buszra kapcsolódó ECU-kat. Mivel ritkán adódik lehetőség egy teljes kommunikációs busz kikapcsolására, a kommunikációs rendszer elemeit a végrehajtott funkcionalitás szerint úgynevezett logikai részhálózatokba (Partial Network) szokás csoportosítani. Ezzel igény szerint lehetőség nyílik a jármű egyes használaton kívüli funkcióinak kikapcsolására. A részhálózatok kezeléséhez vagy *szelektív ébresztés* megvalósulásához, olyan fizikai buszillesztőket kell alkalmazni, amelyek képesek arra, hogy csak *bizonyos üzenetek* hatására váltsanak ki ébresztést az ECU-ban, ezt a CAN kommunikációs protokoll is szabványosan támogatja (ISO 11898-6).

A megvalósított feladat célja egy részhálózatokat támogató rendszerbe illeszkedő, AUTOSAR szoftverarchitektúrával rendelkező vezérlőegység szoftveres vonatkozásainak megvalósítása, illetve a hálózatmenedzsment funkcionalitás helyes működésének ellenőrzése. A feladathoz tartozik még egy PC oldali teszteszköz is, amely egy átjárón keresztül kapcsolódik a kommunikációs buszhoz, és képes szimulálni, illetve megjeleníteni a busz hálózatmenedzsment üzeneteit. Ez a teszteszköz egy Eclipse alapú fejlesztőkörnyezetbe integrálva valósul meg, és felhasználja az AUTOSAR szabványnak megfelelő rendszermodelleket is.

Abstract

The number of electronic control units (ECU) in automobiles is continuously increasing. Designing complex system consisting of hundreds of ECUs is a challenging engineering task. The AUTOSAR standard established in the early 2000s, aims at supporting this design process by various facilities, like a modelling language and software architecture.

The growing number of electronic units results in higher electric power, and *fuel consumption*. This undesired effect can be reduced by switching off inactive units. In traditional topologies electric units are powered on and off according to the ignition signal. However this function can be realized by communication networks, where devices are turned on and off by *bus messages* respectively.

Since in most cases entire communication buses cannot be shut down, ECUs are organized into so called *partial networks* according to their functionalities. This refined structure enables selectively switching off certain unused services in the vehicle. To manage the partial networks, and to realize *selective wake-up*, we have to use such hardware bus interfaces (transceivers), which are able to power up the ECUs on *certain messages*. This mechanism is supported by CAN communication protocol standard (ISO 11898-6).

The goal of the work presented here is realising the software layer of a control unit in AUTOSAR architecture, supporting partial networks. The proper operation of network management is verified, and a PC side test tool is implemented. This application connects to the communication bus by a gateway, displays and simulates the network management messages. The test tool is realised in Eclipse environment, by means of AUTOSAR system models.

1. Bevezetés

Napjaink autóiban az elektronikus vezérlőegységek (Electronic Control Unit, ECU) egyre nagyobb számban terjednek el. Ezek az ECU-k felelősek az autó szinte minden elektromos rendszerének megfelelő működéséért, így tehát a legtöbb komfort szolgáltatásért, a motorvezérlésért, a műszerfal kezelésért, egyéb biztonsági rendszer szabályozásért, mint például a menetstabilizátor és az elektromos kormányrásegítő. Mivel ilyen nagyszámú, és különböző szolgáltatásokat nyújtó ECU-k találhatóak az autókban, ezért az autógyártóknak kézenfekvőnek tűnt szabványosítani ezeknek az elektronikus vezérlőegységeknek a szoftver architektúráit. Az így megalkotott szabvány neve az AUTOSAR.

A diplomatervemet a ThyssenKrupp Presta Hungary Kft.-nél készítettem el. Ez a cég elektromos kormányrásegítők fejlesztésével és gyártásával foglakozik. Mivel a cég is szeretne az autógyártók előírásainak megfelelni, így elengedhetetlen, hogy AUTOSAR kompatibilis ECU-kat fejlesszen.

A járművekben az ECU-k száma folyamatosan növekszik, ez magával vonzza a jármű elektromos hálózatának növekvő teljesítményigényét, amely közvetve a jármű fogyasztásának növekedésével jár együtt. Ezen okok miatt egyre növekvő szerepkör társul a hálózatmenedzsment funkcionalitáshoz, mivel ez felel többek között a jármű energiamenedzsmentjéért is, azaz a passzív elektronikus egységek kikapcsolásáért. Egy szemléletes példa lehet a tolatóradar, amelyre csak hátramenetben van szükség, vagyis a normál közlekedés során ez az egység kikapcsolható. Ugyanakkor a tolatóradar is ugyanarra a kommunikációs buszra kapcsolódik, mint a menetstabilizátor szenzorai, vagy fényszóróvezérlés, amelyekre menet közben szükség van.

A cégnél végzett eddigi munkám, és önálló laboratóriumi feladatom keretében részt vettem az AUTOSAR hálózatmenedzsment moduljainak megvalósításában. Az itt bemutatott diplomaterv feladatom egy, az ezen modulokra épülő demonstrációs és tesztelést támogató struktúra kialakítása, amely a cégnél korábban fejlesztett AUTOSAR Network Management Stack helyes működését igazolja. A szabványban definiált hálózatmenedzsment bizonyos funkciói csak pár éve jelentek meg, így ezekről még csekély tapasztalattal rendelkezünk, így a demonstrációs feladat a helyes működés igazolásán felül-, tapasztalatok megszerzésével is együtt jár.

2. Feladat bemutatása

Az elkészített demonstrációs projekt célja, hogy a saját készítésű hálózatmenedzsment funkciók megfelelő működését bemutassa és az esetleges hibákat a felszínre hozza. Azért is jelentős ennek a funkciónak a kipróbálása, mert korábban még nem volt jellemző a hálózatmenedzsment használata, különösképpen a részhálózat menedzsment, így a cégnél viszonylag kevés tapasztalati tudás halmozódott fel ezzel a funkcionalitással kapcsolatban. Tehát az így elkészített demonstrációs projekt a funkcionalitás megfelelő működésének igazolásán túl további tapasztalatok megszerzését is elősegíti.

Az elkészített demonstrációs alkalmazásnak be kell mutatnia, hogy a hálózatmenedzsment funkciók megfelelően működnek egy NM-Slave ECU-n valamint, NM-Master ECU-n is. Ezen felül a demonstrációs projektnek egy PC oldali vonatkozása is jelentkezik, ahol a PC-n futó szoftver tölti be a hálózatot koordináló eszköz szerepét.

A hálózatmenedzsment funkcionalitás teszteléséhez egyrészről szükséges a hálózatmenedzsment blokkot is tartalmazó *platform integrálása*. Ezzel természetesen még nincs teljes mértékben megvalósítva a teljes hálózatmenedzsment funkcionalitás, hiszen ez a blokk önmagukban csak egy szolgáltatást nyújtanak a hálózat kezelésére, azonban nem rendelkeznek önálló logikával. A rendszer logikai részei a BSW funkcionalitásán kívül az alkalmazások között helyezkedik el. Tehát a feladat második része olyan *szoftverkomponensek készítése*, melyekkel végső soron tesztelhető a BSW hálózatmenedzsment funkcionalitása. A hálózatmenedzsment funkcionalitás tesztelése során érdemesebb egy részhálózatokat is tartalmazó rendszert összeállítani, hiszen ez magában foglalja a hagyományos hálózatmenedzsmentet is.

A gyakorlatban, a részhálózat menedzsmentben az ECU-knak három különböző szerepkörét különböztetjük meg, melyek hierarchikusan egymásra épülnek. A legalsó szinten található a *Partion Network Slave* (PnS), amely csak fogadja a hálózatról érkező NM - üzeneteket és alkalmazkodik az abban kért hálózati állapothoz, illetve bizonyos esetekben ő maga is kérheti, hogy maradjon még ébren a hálózat. A PnS feletti szinten a *Partial Network Domain Master* (PnDM) található, amely a gateway ECU-k szoftvere, tehát feladata kiértékelni, hogy az egyik hálózatról érkező kéréseket egy másik hálózatra szükséges-e továbbítania vagy sem. A legfelső szinten a *Partial Network Master* (PnM)

található, amely gyakorlatilag a teljes jármű összes hálózata felett rendelkezik koordinációs jogokkal. Ebben az ECU-ban található a teljes jármű hálózatmenedzsmentjének a logikája.

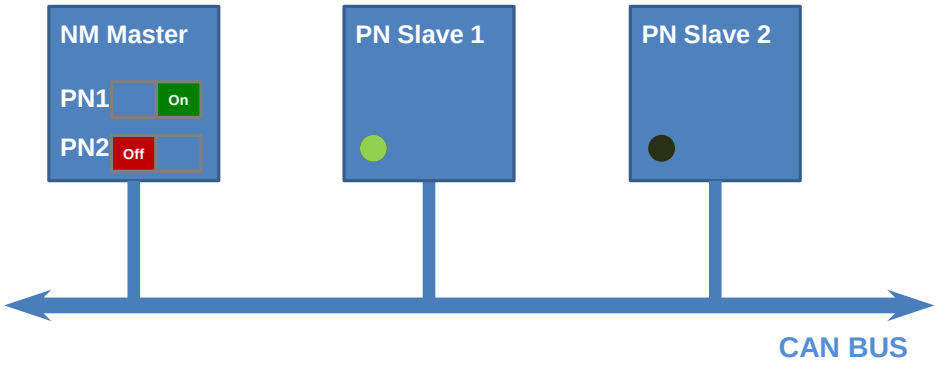
Ez a bevezetőben szereplő példánál maradva, azt jelentené, hogy a tolatóradar egy slave egységnek felel meg, amely csak akkor kapcsol be, ha a vezető a váltót hátramenetbe kapcsolja. Míg a master egység egy központi ECU, amely feldolgozva a váltóvezérlő üzenetét, bekapcsolja a parkolást segítő rendszereket, ezzel együtt a tolatóradart.

Ahogyan ezekből kiderül, minimum egy master és egy slave ECU-nak kell szerepelnie a hálózatban, így elengedhetetlenül szükséges mindkét egység megvalósítása.

2.1. Beágyazott szoftver (SWC) feladata

A feladat megvalósításához szükséges egy NM-Master és egy NM-Slave alkalmazás implementálása. A demonstrációs projektben a beágyazott szoftver első megközelítésben egy – a saját ECU-nkon is megtalálható mikrokontrollerrel ellátott – fejlesztői panelre illetve panelekre kerül letöltésre. Cél, hogy az NM-Master kódot futtató fejlesztői panelen kapcsolók segítségével kiválasztható legyen a bekapcsolni kívánt részhálózat, és a master eszköz ennek megfelelő „NM-Message”-eket küldjön a buszon. Ebben az esetben csakis annak a slave eszköznek szabad bekapcsolnia, amely a kiválasztott részhálózatba tartozik, míg a többi slave egységnek továbbra is alvó állapotba kell maradnia.

Az a slave egység, amelyik már felébredt a buszon érkező kérésre, a fejlesztői panelon rendelkezésre álló ledekkel jelezheti aktív állapotát, ezen felül a slave eszköz üzeneteket is küld, amellyel a kommunikációs busz forgalmát ellenőrizve bizonyítható az eszközök helyes működése, vagyis, hogy az eszköznek sikerült bekapcsolnia. Az üzenetekben elküldhet egy saját azonosítóját is, így könnyedén azonosíthatóak a bekapcsolt ECU-k.



2.1. ábra, Hálózatmenedzsment demó projekt

2.2. PC oldali szoftver

A PC oldali szoftverrel lényegében a master eszközt lehet lecserélni és így tesztelni a slave eszközök helyes működését a hálózatmenedzsment szempontjából. Ezzel egy olyan eszközhöz jutunk hozzá, amellyel már nem csak a fejlesztői panelen futó hálózatmenedzsment blokk működése vizsgálható, hanem valódi ECU-n is kipróbálhatóvá válik a funkcionalitás helyes működése. A PC-vel kiegészített tesztelés esetén válik jelentőssé az, hogy az ECU-n futó NM-Slave alkalmazás éber állapotban bizonyos üzeneteket küld, hiszen ezeket a PC-n feldolgozva ellenőrizhető az eszközök bekapcsolt állapota.

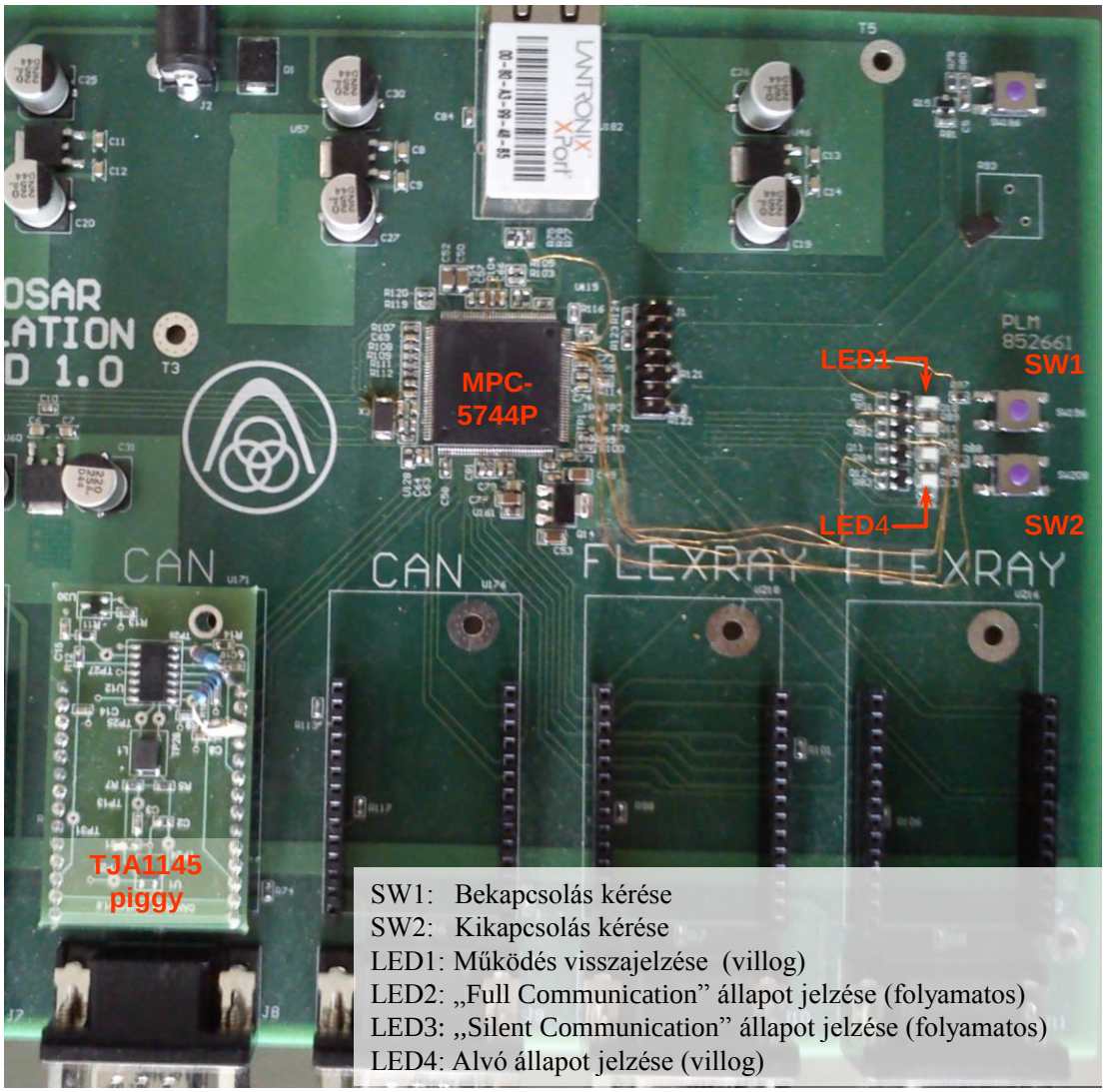
Az elkészített PC-s programmal azonban nem csak egy tesztkörnyezetet kapunk, hanem a valódi fejlesztés folyamán felhasználhatjuk a részhálózatokba illeszkedő ECU-k ébren tartására. Ilyenkor ugyanis mindenképp szükségesek az ECU-nak az „NM-Message”-ek jelenléte a kommunikációs buszon, hiszen ezek hatására marad ébren.

A korábbi években a cégnél elkészült egy gateway eszköz, amelynek segítségével Etherneten keresztül PC-vel lehet kapcsolódni különböző autóipari kommunikációs buszokra. A cél erre az eszközre épülő, hálózatmenedzsment üzenetek küldését és fogadását valamint azok tartalmának feldolgozását elvégezni képes teszteszköz megvalósítása.

3. Hardver

3.1. Fejlesztői panel bemutatása

A demonstrációs projekt egy, a cégnél tervezett fejlesztői panel felhasználásával valósult meg. A hardver tartalmaz egy autóipari felhasználásra tervezett mikrokontrollert és a hozzá kapcsolódó kommunikációs buszok jelszintjéhez illesztő transceivereket. A fejlesztői panel kialakítása olyan, hogy a transceiverek modulárisan cserélhetőek legyenek (úgynevezett piggy boardokra vannak építve), így a fejlesztői panellel tesztelhetőek különböző transceiverek működése. Ez két okból jelentős, egyrészről, amikor egy már használatban lévő transceivert egy hasonló funkcionalitású, de alacsonyabb árfekvésű termékkel akarják helyettesíteni, akkor könnyebben tesztelhető az új alkatrész elvárt működése. Vagy a másik ok lehet egy új, bővebb funkciókkal rendelkező transceiver kipróbálása, mint ebben a projektben is, a részhálózatokat is kezelni képes TJA1145. A teljes fejlesztői panelon két mikrovezérlő is található, de ebből csak a Freescale által gyártott MPC5744P típusú kontroller került felhasználásra. Ehhez a kontrollerhez két CAN és két FlexRay kommunikációs hálózat kapcsolható. Az egyes piggy boardok - ezek a modulárisan cserélhető transceiverek a normál kommunikációs vezetékeken felül, még SPI-porton is összeköttetésben állnak a mikrokontrollerrel. A továbbiakban még két nyomógomb és négy LED szolgáltatja a felhasználói interfészét a panelon, illetve egy Ethernet - soros port átalakítón keresztül nyílik lehetőség további kommunikációra a mikrovezérlővel, azonban ez jelen projektben nem használtam.



3.1. ábra, Fejlesztői panel

3.2. Mikrokontroller bemutatása

Ahogyan az előzőekben már említettem, jelen projektben a Freescale MPC5744P típusú mikrovezérlőjét használtam. Ez egy kifejezetten autóipari alkalmazások számára kifejlesztett kontroller, amely alkalmas biztonságkritikus feladatok elvégzésére, amit többek között a lock step működésű kétmagos processzor, valamint az integrált MPU biztosít. A megbízható működésen felül az autóipari felhasználhatóságát a perifériák között megtalálható három CAN, két LIN és a Dual-channel FlexRay kommunikációs kontrollerek is biztosítják. Továbbá a mikrovezérlő több futási módot is támogat, amelyekhez egyedileg megadható, hogy a processzor melyik perifériákat érjen el, illetve egyedi órajel frekvencia állítható be. Ez a funkcionalitás a demó projektben is felhasználásra kerül [21].

3.3. Transceiver

A hálózatmenedzsment működését demonstráló projektben az NXP TJA1145-ös CAN fizikai buszillesztő került alkalmazásra. Ez a transceiver kifejezetten autóipari felhasználásra készült és integrálva támogatja az ISO-11898-6 CAN „Partial Networking”-et, amely nagyban megfeleltethető az AUTOSAR szabványban definiált részhálózatok kezelésének. Tehát ez egy „intelligens” transceiver, amely rendelkezik a hagyományos transceiverek minden tulajdonságával és ezen felül képes a kommunikációs buszon megjelenő keretek dekódolására, illetve értelmezésére. Ez pontosabban megfogalmazva azt jelenti, hogy a transceiver felprogramozható a buszról érkező speciális üzenet detektálására – ezek az úgynevezett „wake up frame”-ek (WUF) –, amelynek hatására ébreszteni tudja az ECU-t. Ebben a szakaszban röviden bemutatásra kerül ennek a speciális transceivernek a működése.

Ahogy az a hardverrendszer leírásakor is látható volt, ez a transceiver a hagyományos társaival ellentétben, nemcsak a CAN kommunikációs vonalon keresztül áll összeköttetésben a mikrokontrollerrel, hanem SPI-on keresztül is, ezzel lehetőséget teremtve a transceiver felprogramozására. Még egy eltérést találhatunk a hagyományos buszillesztő áramkörökhöz képest, mégpedig az IC INH kimenetét, amely az ECU feszültségszabályozójának vezérlésére szolgál, ezáltal lehetőség van a teljes ECU kikapcsolására [20].

3.3.1. Transceiver állapotai

Természetesen egy ilyen transceiver a különböző funkcionalitások végrehajtása érdekében különböző állapotokkal rendelkezik. A következőkben ezek az állapotok és állapotváltozások kerülnek bemutatásra.

Kikapcsolt (Off): Ez a transceiver alapállapota, ide csak akkor kerülhet a transceiver, ha az energiaellátás nem megfelelő, vagyis az akkumulátor feszültsége túl alacsony.

Készenléti (Standby): Ha az akkumulátor feszültségszintje megfelelő, ebbe az állapotba kerül a transceiver. Bár ebben az állapotban az INH láb aktív – tehát az ECU bekapcsolt állapotban van –, ilyenkor nem lehetséges a CAN üzenetek küldése és fogadása. Abban az esetben, ha a CAN ébresztési funkcionalitás engedélyezett, akkor figyeli a buszt és teljes kommunikációs módba vált a WUF detektálásakor, vagy a szimpla buszforgalom hatására.

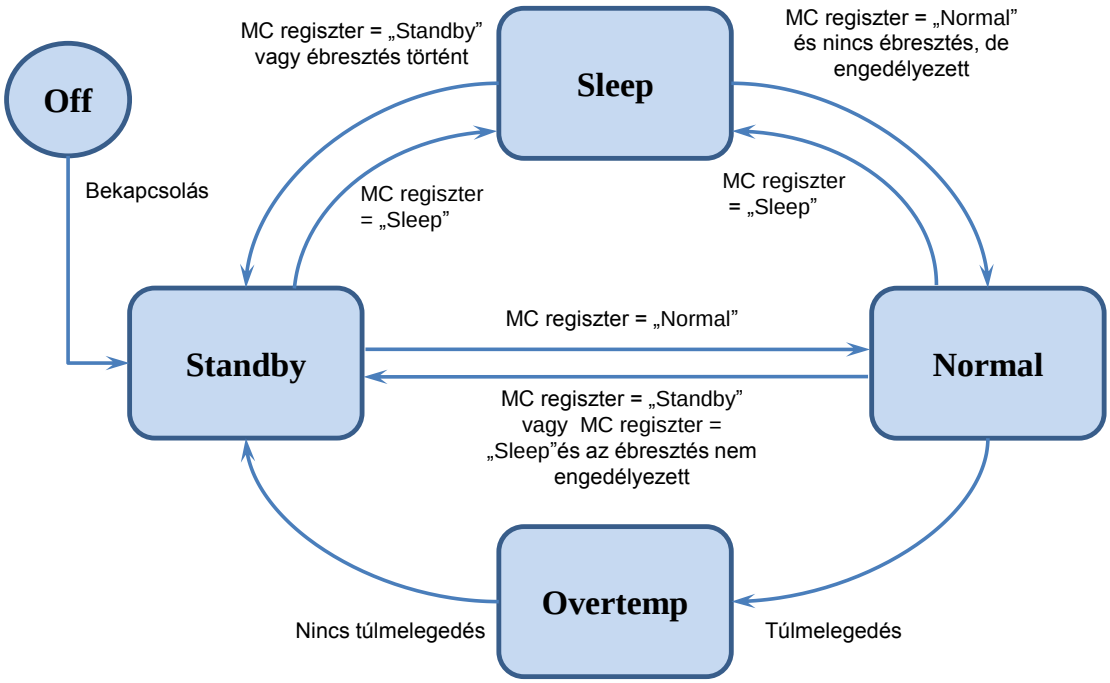
Alvó (Sleep): Ez a legenergiatakarékosabb módja a transceivernek és ilyenkor az INH láb is nagyimpedanciás, ami alkalmazástól függően az ECU kikapcsolt állapotát jelentheti. Ebben az állapotban is működik az ébresztési események figyelése. Ebben az üzemmódban a transceiver áramfelvétele 40μA-ig csökkenthető.

Normál (Normal): Ebben az állapotban a transceiver minden funkcionalitása elérhető, azaz nincs semmilyen korlátozás a kommunikációban.

Túlmelegedett (Overtemp): Ez az ötödik állapot kevésbé releváns, csupán a teljesség igénye miatt kerül megemlítésre. Ebbe az állapotba lép a transceiver, amennyiben valamilyen oknál fogva a chip egy küszöbhőmérséklet fölé emelkedik. Ilyenkor a transceiver lekapcsolódik a CAN buszról és letiltja az SPI kommunikációt is. Addig marad ebben az állapotban, amíg egy biztonságos hőmérséklet alá nem hűl az IC, majd alvó állapotba vált.

A következő táblázat összefoglalva tartalmazza a transceiver különböző állapotaiban elérhető funkcionalitásait, míg az azt követő ábrán az állapotátmenetek láthatóak.

	Működési mód				
	Kikapcsolt (off)	Készenléti (standby)	Alvó (sleep)	Normál (normal)	Túlmelegedett (overtemp)
SPI	nincs	elérhető	elérhető	elérhető	nincs
CAN	nincs	offline	offline	elérhető	nincs
INH	kikapcsolt	bekapcsolt	kikapcsolt	bekapcsolt	kikapcsolt



3.2. ábra, TJA1145 állapotgépe

3.3.2. A transceiver konfigurációja

Ahogy arról már korábban szó esett, a transceiver SPI-on keresztül konfigurálható, és ebben a szakaszban erről kívánok egy rövid áttekintést nyújtani.

A transceiver bájtos szervezésű, ez azt jelenti, hogy a memóriaregiszterei bájtonként érhetőek el, valamint a címzésük is bájtos. Tehát ezt a transceiver is sok külső perifériához hasonlóan egy olyan SPI üzenettel lehet kezelni, amelynek első bájtja tartalmazza a memóriarekesz címét illetve a vezérlőbitet – a 7. bit írást vagy olvasást jelöl – és a további bájtok pedig a sorra következő rekeszek tartalmát hordozzák.

A transceiver legfontosabb regiszterei között az első helyen található – a „System Control Register”, melyből kiolvasható a transceiver aktuális állapota, illetve beállíthatjuk az általunk elérni kívánt állapotot. Amennyiben szeretnénk, hogy a transceiver támogassa a részhálózatok kezelését, úgy első lépésként a „CAN Control Register”-ben kell ezt engedélyeznünk. Ennek a regiszternek a párja a „Transceiver Status Regiszter”, amelyből a transceiver aktuális állapotáról szerezhetünk információt. A részhálózatok támogatásához továbbá meg kell adni a CAN kommunikáció sebességét a „Data Rate Register”-ben, illetve a CAN keretek típusát - normál 11 bites vagy kiterjesztett 29 bites – és az adatcsomag hosszát – DLC mező – a „Frame Control” Register”-be. Ezen felül szükséges még megadni WUF-ek CAN azonosítóját és a hozzá

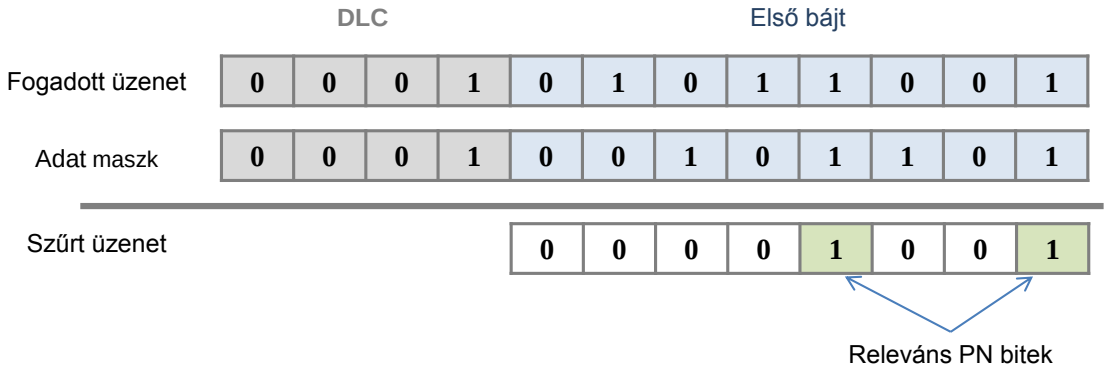
tartozó maszkot 4-4 bájton az „ID” és a „Mask” regiszterekben illetve az adatmező maszkot – max 8 bájt – a „Data Mask Register”-ekben.

A transceiver konfiguráció rövid áttekintését követően, a wake-up frame detektálási módját mutatom be. A wake-up frame semmiben sem különbözik a többi CAN üzenettől, tehát ugyanúgy van azonosító (ID), hossz (DLC), adat és ellenőrző összeg (CRC) mezője. A „wake-up frame”-ek számára kijelölhető a CAN üzenetek egy tartománya (CAN-ID-k csoportja) is. Összefoglalva a „wake-up frame”-ek esetében az azonosító mező lehet egyedi vagy egy csoport. Csoport megadása esetén a „Mask” regiszterben 1-esnek állított bitek számítanak közömbösnek, így az érvényes azonosítók meghatározásának menete a következő:

Azonosító (0x164)	0	0	1	0	1	1	0	0	1	0	0
Maszk (0x007)	0	0	0	0	0	0	0	0	1	1	1
Érvényes azonosító (0x160 – 0x167)	0	0	1	0	1	1	0	0	X	X	X

3.3. ábra, Érvényes CAN-ID számítása

Azonban egy WUF fogadása önmagában nem jelenti egy ébresztési esemény bekövetkeztét – kivéve, ha a DLC értéke 0 –, ahhoz az is szükséges, hogy az üzenet tartalma megfelelő legyen. Ez akkor következik be, amikor a fogadott üzenetben van olyan bit, ami a „Data Mask” mezőben is be lett állítva. Releváns „wake-up frame” detektálásának a menete következő ábrán látható.



3.4. ábra, Érvényes WUF detektálásának számítása

4. AUTOSAR elméleti áttekintés

4.1. AUTOSAR bemutatása

Az AUTOSAR egy mozaikszava az angol „AUTomotive Open System ARchitecture” kifejezésnek, mely magyarra „autóipari nyílt rendszer architektúraként” fordítható. Ez egy 2002-ben alapított konzorcium, melynek alapító tagjai között a BMW, a Bosch, a Continental, a DaimlerChrysler és a Volkswagen is megtalálható. Későbbiekben több autógyártó cég és autóipari beszállító is – többek között a ThyssenKrupp is - csatlakozott ehhez a szervezethez. Az autón belüli elektronikus egységek gyarapodásával együtt, fokozatosan növekedett az igény arra, hogy valamilyen módon ezek a komplex rendszerek absztrakt módon tervezhetőek legyenek. Az AUTOSAR szabvány erre a feladatra kínál egy megoldást, ugyanis a szabvány gyűjtemény meghatároz egy *fejlesztési módszertant*, egy *modellező nyelvet*, – amellyel ez a komplex rendszer leírható – és egy *szoftverarchitektúrát*, aminek segítségével megvalósítható a modellezett rendszer. Mivel a vezérlőegységek különböző gyártóktól kerülnek ki, így a szabványosított architektúra használatával az ECU-k kompatibilitása biztosított. A szabvány mind az ECU gyártók, mind az autógyártók számára gazdaságosabb a régi rendszernél, ezt tükrözi az alapító cégek tevékenységi köre is. Egy AUTOSAR szabványnak megfelelő ECU minőséget jelent az autógyártóknak, valamint garanciát a különböző beszállítók közötti együttműködésre. A szabványos szoftverkomponensek gyorsabbá és hatékonyabbá tehetik a fejlesztést a beszállító cégek számára és megkönnyítheti a különböző ECU változatok fejlesztését. Ugyanakkor a szabványnak hátrányai is vannak. Egyes esetekben a szabványnak való megfelelés miatt jelentősen megnövekedhet a szoftver mérete, valamint normál használatban is megnövekszik a processzor terhelése, ezen felül esetenként felmerülhetnek olyan igények, amelyek nem összeegyeztethetőek a szabvánnyal. Az AUTOSAR szabvány az újabb igényekhez igazodva, valamint az egyes hibákat orvosolva folyamatosan fejlődik. Jelenleg a 4.2.1.-es verzió a legújabb változata, azonban a diplomatervben felhasznált modulok a 4.0.3-as változatnak felelnek meg [1].

4.2. AUTOSAR modellező nyelv

Mivel a mai modern gépjárművek szinte minden részegységében megtalálható az elektronika, így az autógyártóknak a jármű mechanikus részein felül, annak szoftverrendszerét is meg kell tervezniük. Számukra ebben nyújt segítséget az AUTOSAR *modellezési nyelv*, amellyel felépíthetik absztrakt módon a jármű szoftverrendszerét, vagyis a teljes részletesség kidolgozása nélkül elegendő, ha meghatározzák a főbb funkcionalitásokat és a közöttük felmerülő kapcsolatokat. Az így létrehozott modell a „Software Composition”.

Egy következő lépésben az autógyártók a „System Description”-t készítik el, amelynek tervezése a szabványban definiált „System Template” alapján történik. Ez a modell már pontosan definiálja, hogy milyen ECU-k vannak a rendszerben és azokhoz mely szoftver kompozíciókat tartoznak, illetve ez a dokumentum rendelkezik az ECU-k közötti kommunikációról is, valamint tartalmazza az egyes kompozíciók közötti kommunikációs elemek (szignálok és PDU-k) leírását is.

Az egyes beszállítók általában a teljes rendszert leíró modelleknek csak a számukra releváns részét kapják meg, ennek a modellnek a neve az ECU Extract. Ez valójában az ECU-n futtatott szoftverkompozíciót tartalmazza, pontosabban annak is csak a portjait, mivel ezt a szoftverkompozíciót a szoftverkomponensekkel már a beszállítók tervezik meg. Mint látható, ezen a ponton mutatkozik meg a gyártók számára az AUTOSAR szabvány egyik előnye, hogy jól definiált komponensekből építik fel a teljes rendszert, így az egyes komponenseket bármikor kicserélhetik egy másik olyan komponensre, amely teljesíti annak előírásait, és megfelelően csatlakozik annak portjaira.

Azonban a modell vezérelt rendszertervezés itt még nem ért véget, ugyanis a beszállítók a saját kompozíciójukat további komponensekre bontják, amelyek egymással – ugyanúgy, mint más beszállítók szoftverkomponenseivel – a „Virtual Function Bus”-on keresztül kommunikálnak. Illetve ugyancsak modellezetten zajlik az ECU-n belüli rendszer konfigurációja is. Ennek a modellnek a neve „ECU Configuration”, amely tartalmazza az egyes *Basic Software modulok* (BSW) és az *Runtime Environment* (RTE) konfigurációját, amelyek a későbbiekben még kifejtésre kerülnek.

A beszállítóknak ezen a ponton mutatkozik meg az AUTOSAR szabvány előnye, mivel egyrészről a legtöbb autóipari elektronikai alkatrész beszállító a saját terméküket az AUTOSAR szabványnak megfelelő driverrel együtt árulják, így ezek is nagyban csereszabatosak (ugyanazokat az API-kat tartalmazzák), másfelől ez a modellezett szoftverarchitektúra elősegíti az újrafelhasználhatóságot.

Végezetül az „ECU Configuration”-ből kerül generálásra többek között az egyes BSW modulok konfigurációja, néhány modul illetve az RTE dinamikus kódrészlete, és esetlegesen az alkalmazások váza is [14][16][17][18].

4.3. AUTOSAR szoftverarchitektúra

Az AUTOSAR szoftverarchitektúra alapvetően három rétegre bontható:

- Alkalmazás réteg (Application Layer)
- Futtató környezet (Runtime Environment, RTE)
- Alapvető szoftver (Basic Software, BSW)

4.3.1. Alkalmazás réteg

Az alkalmazás réteg tartalmazza az applikációkat, vagyis az alkalmazás logikát tartalmazó szoftver komponenseket. Ez a réteg teljesen hardverfüggetlen, tehát hordozható az egyes AUTOSAR kompatibilis rendszerek között. Az AUTOSAR szoftverarchitektúrának megfelelő rendszerben az ECU lényegi feladatát végrehajtó kódot az egyes *szoftverkomponensekben* találhatjuk. A szoftverkomponenseknek három típusát különböztethetjük meg, ezek a paraméter komponens, ami valamilyen konfigurációt tartalmaz; a kompozíciók, ami több komponenst fog össze, illetve az atomi szoftverkomponensek, amelyek forráskódot tartalmaznak [16][17][18][19].

Minden szoftverkomponens és kompozíció rendelkezik *portokkal*, amelyen keresztül kapcsolatot teremt más szoftverkomponensekkel. Az AUTOSAR 4.0-ban a portoknak irányuk szerint két típusát különböztetjük meg. Az egyik típus a „*provided*” port, amelyen keresztül az adott komponens valamilyen szolgáltatást nyújt más szoftverkomponenseknek, míg a portok másik típusa a „*required*” port, amin keresztül az adott szoftverkomponens egy szolgáltatást vár el más szoftverkomponenstől.

Az egyes portokon való kommunikációt a portokhoz tartozó *interfészek* írják le. Az interfészeknek szintén több típusa létezik. A két legfontosabb interfész típus a „Sender - Receiver ” és „Client - Server ”.

A „Sender - Receiver” interfész a szoftverkomponensek *adatfolyam alapú kommunikációjának* megvalósítására használható. Amennyiben egy ilyen típusú interfészt használunk, akkor a futtatókörnyezet biztosítja számunkra, hogy az elküldött egy-egy adatelemet konzisztensen kapja meg a fogadó szoftverkomponens.

A „Client - Server” interfész *(távoli) eljáráshívásra* használható. Ez tulajdonképpen azt jelenti, hogy az egyik szoftverkomponensben megvalósított és a „provided” portján kiajánlott függvényét meghívhatjuk egy másik

szoftverkomponensből, amelynek létezik olyan ”requested” portja, amely ezt a függvényt reprezentálja. Ennél az interfésztípusnál a futtatókörnyezet azt biztosítja, hogy egy szerverhez több kliens kapcsolódhasson. A „Client – Server ” típusú kommunikációnak két változata létezik, a szinkron, amikor a hívó szoftverkomponens blokkolódik a függvény lefutásának végéig, míg a másik, aszinkron esetben a hívó nem blokkolódik, csak elindítja a kért eljárást és az eredményre egy későbbi lekérdezéssel tehet szert.

Az előző két típusú interfészen kívül a szabvány még a következő ritkábban használt interfésztípusokat definiálja:

- „Parameter” interfész, amely nagyban hasonlít a „Sender - Receiver” interfészhez, de ez csak olvasható és a különböző szoftverkomponensek konfigurációjának megadására szolgál.
- „Non Volatile” interfész, amely feladata elfedni a szoftverkomponensek elől a nem felejtő memóriát, mely szintén a „Sender - Receiver ”interfészhez hasonlít.
- „Mode - Switch” interfész, amely az AUTOSAR 4.0-ás változatától kezdve létezik. Ez az interfész szolgál a vezérlőegységen belüli különböző módok vezérlésére, mint például a kommunikációs mód váltásáért. Az ilyen interfészeknek létezik egy meghatározott állapottere (Mode Declaration Group) és csak ezen az állapottéren belüli állapotok kérhetőek rajta. Ez a kommunikációs mód, példánál maradva a teljes kommunikációt „Full Communication”, a csendes vagy hallgatózó módot „Silent Communication” és a kikapcsolt kommunikációt „No Communication” jelenti.
- „Trigger” interfész, amelyen keresztül az egyes interfészeknek aszinkron jelzések adhatóak.

4.3.2. Futtató környezet

A futtató környezet egy generált szoftverréteg, melyen keresztül a különböző felhasználói szoftvermodulok kapcsolódnak egymáshoz. Ezt az RTE rétegben megvalósított úgynevezett „Virtuális Funkció Busz” (Virtual Functional Bus, VFB) biztosítja. A VFB-hez minden szoftverkomponens egy szabványos protokollon keresztül kapcsolódik. A VFB létrehozása egy elvi meggondolás, miszerint a szoftverkomponenseknek – függetlenül attól, hogy fizikailag melyik ECU-n futnak – úgy kell látniuk, mintha ugyanahhoz a logikai buszhoz lennének kapcsolódva. Az RTE szerepe ezt biztosítani a megfelelő BSW komponensek felhasználásával [1][2][13].

4.3.3. Alapvető szoftver

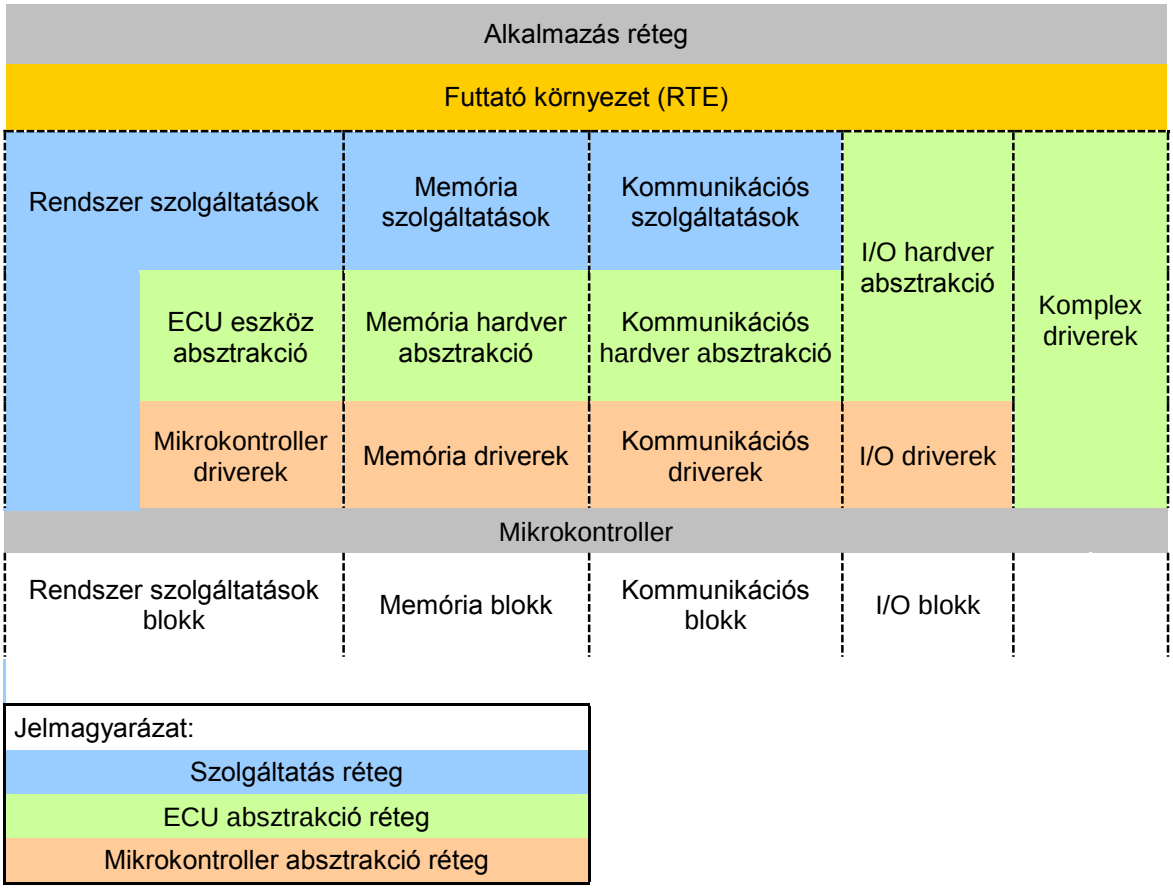
A Basic Software (BSW) egy szabványosított szoftver réteg, amely tartalmazza azokat a szoftver modulokat, amelyek szükségesek a szoftverkomponensek futtatásához. Ez a szoftverréteg a futtató környezeti réteg alatt helyezkedik el és olyan szolgáltatások szabványosított gyűjteménye, amelyre minden ECU-ban szükség van. A BSW réteg alatt már közvetlenül a mikrokontroller található. A BSW réteg moduljait két, eltérő elgondolás alapján különböző csoportokra lehet osztani, amit a 4.1. ábra szemléltet [1][2][13].

- Szolgáltatások: Ez azokat a BSW modulokat foglalja magában, amelyekkel a szoftverkomponensek interakcióba léphetnek. Ilyen modul többek között a memóriamenedzsment (NVRAM), a diagnosztikai protokoll (DCM), valamint még a kommunikációs egység (COM) és az operációs rendszer (OS).
- ECU Absztrakció: Ez a modul eltakarja a felsőbb szoftvermodulok elől a hardver specifikus tulajdonságokat. Például a kommunikációs kontrollerek számát, típusát, stb.
- Mikrokontroller absztrakció: Ehhez a funkcióhoz tartozó modul feladata elfedni a felsőbb szoftverkomponensek elől a mikrovezérlő specifikus funkciókat. Többek között tartalmazza a következő mikrovezérlő perifériák kezelését: Digitális ki/bemenetek (DIO), Analóg digitális konverterek (ADC), Watchdog Timer (WDT) stb.

Míg az előző felosztás a BSW horizontális tagolódását jelenti, addig a BSW modulok vertikális csoportosítása a modulok funkcionalitás szerint felosztását jelenti. A szabvány ezekre a csoportokra a „Stack” kifejezést használja. A legfontosabb ilyen blokkok (Stack-ek) a következőek:

- Rendszerszolgáltatások blokk: A rendszerszolgáltatásokat nyújtó modulok közé tartoznak a különböző felügyeleti rendszereket (WDT) kezelő modulok, a különböző diagnosztikai modulok (DEM, DET), valamint ebbe a blokkba tartozik az operációs rendszer (OS) is.
- Memória blokk: Ebbe a blokkba tartozik az összes memóriakezelésért felelős modul. A memóriablokk legfelsőbb rétegében található az NVRAM Manager, amely a teljes memóriahasználat koordinálásáért felelős. Az AUTOSAR szabvány támogatja az EEPROM és/vagy FLASH EEPROM használatát, továbbá felkészült más egyedi memóriák illesztésére.
- Kommunikációs blokk: Ebben a csoportban találhatóak a kommunikációval kapcsolatos szoftvermodulok. Az AUTOSAR szabvány támogatja az autóiparban legelterjedtebb kommunikációs protokollokat: CAN, LIN, FlexRay, Ethernet, és lehetővé teszi más protokollok hozzáadását a kommunikációs blokkhoz.
- I/O blokk: Ebben a blokkban találhatóak a mikrokontroller ki és bemeneti perifériáit kezelő szoftvermodulok. Ez a blokk főképpen különböző drivereket tartalmaz, mint például a digitális ki és bemenetek, az analóg digitális konverterek (ADC), és a pulzus szélesség modulátorok (PWM) driverjei.
- Komplex eszköz driver (meghajtó) (CDD): Ez a modul közvetlen hozzáférést nyújt az egyes hardveres erőforráshoz, ami erőforrás kritikus alkalmazások esetén nagy segítséget nyújthat.

Bár külön-külön blokkokba sorolhatóak az egyes modulok, azonban ezeknek a funkcióknak a megvalósulásai sok esetben túlnyúlnak az adott blokk határain és a teljes funkcionalitásukat más blokkokkal együtt érik el.

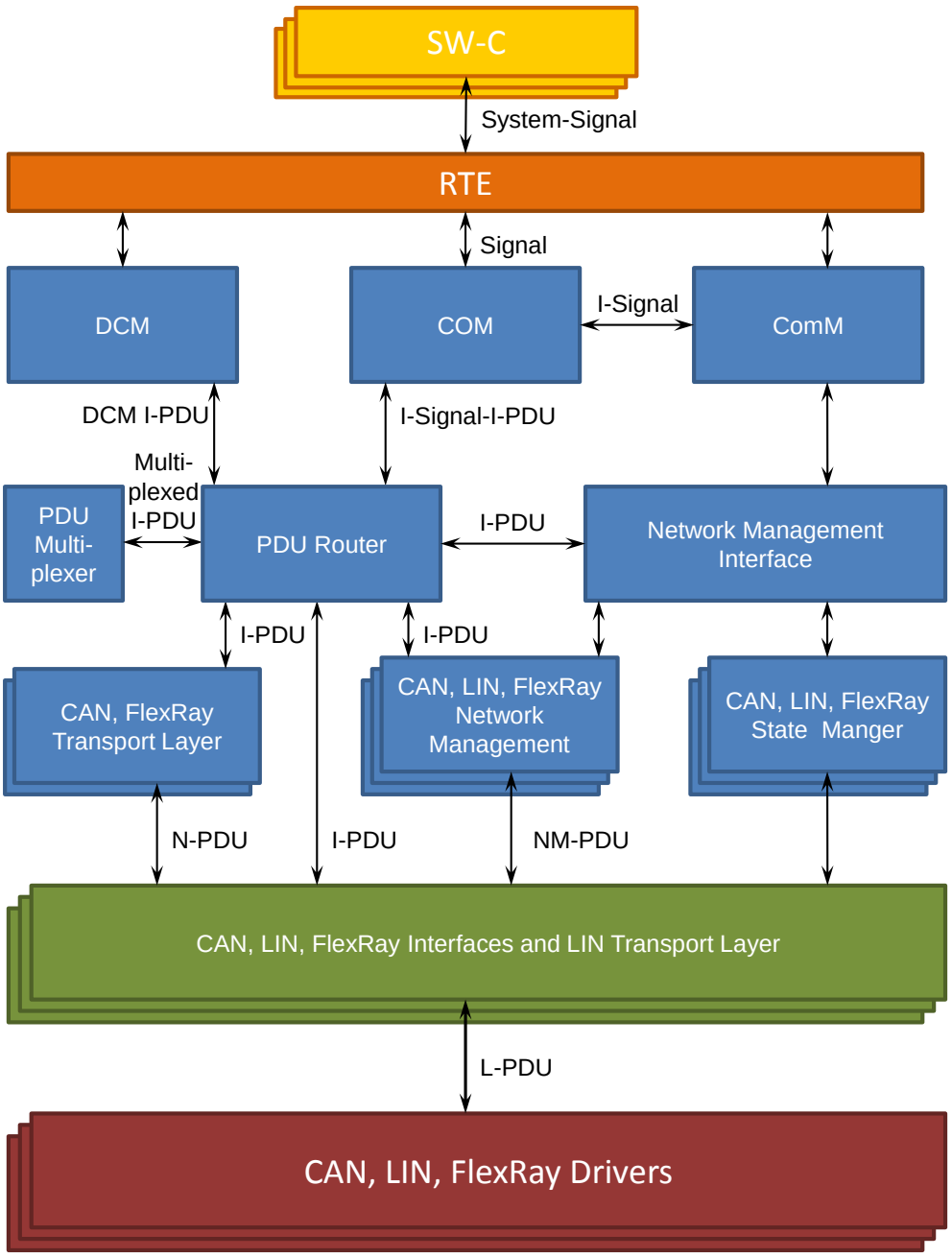


4.1. ábra, AUTOSAR szoftverarchitektúra

4.4. Kommunikációs blokk

A kommunikációs blokk, vagy ahogyan a szabványban szerepel „Communication Stack”, az alapvető szoftver réteg (BSW) része. Ebben az egységben található minden hálózati funkció ellátásához szükséges szoftvermodul. Ezek a szoftvermodulok jobbára az ISO/OSI modellnek megfelelően épülnek fel.

A 4.0.3-as AUTOSAR szabvány tartalmazza az autóipar legfontosabb kommunikációs protokolljait CAN, FlexRay és LIN, valamint az elterjedt Ethernet hálózati protokollt. Mindegyik kommunikációs blokk hasonló felépítéssel rendelkezik. Ezekről a 4.2. ábra nyújt átfogó képet. A hálózatmenedzsment blokk is a kommunikációs blokk része, bemutatására később kerül sor.

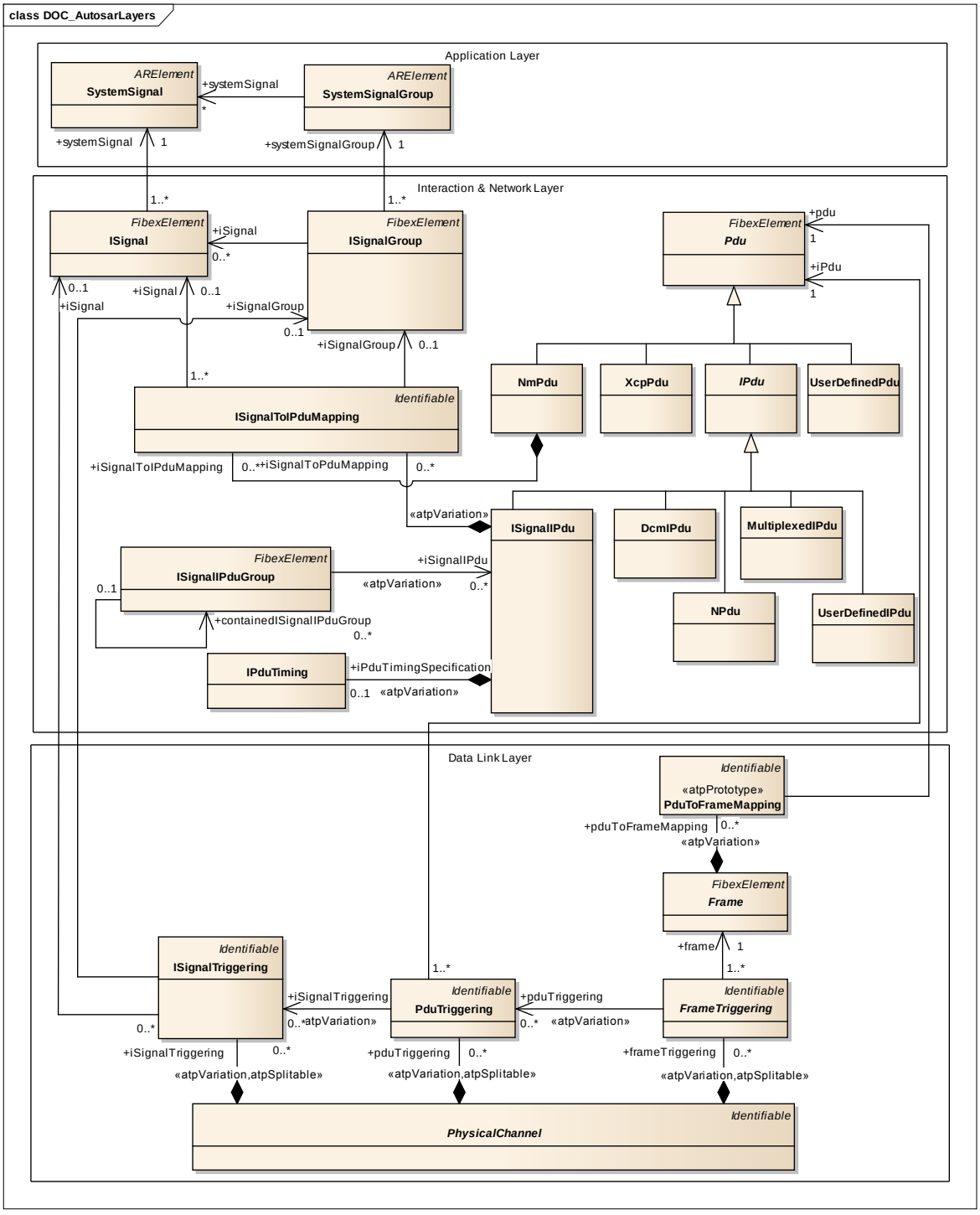


Prefixumok jelentése	
I-	Interaction Layer
N-	Network Layer
L-	Data Link Layer
NM-	Network Management

4.2. ábra, AUTOSAR Kommunikációs blokk

4.5. Adatcsomagok

Ebben a fejezetben röviden szeretném bemutatni az AUTOSAR szoftverarchitektúrában előforduló különböző adatcsomagokat és azok kapcsolatát. Ezen adatcsomagok rétegzetten egymásra épülnek, melynek részletes leírása a „System Template”-ben olvasható.



4.3. ábra, AUTOSAR kommunikációs rétegek[16]

A legfelsőbb szinten az egyes szoftverkomponensek egymással úgynevezett „System Signal”-okon keresztül kommunikálnak. A komponensek portjain megjelenő interfészekhez ezek a szignálok vannak hozzárendelve és a teljes rendszerben értelmezendőek, vagyis a VFB feladata ezeknek a szignáloknak a felhasználásával összekötni a megfelelő komponenseket. A „System Signal”-okból álló csoportokat „System Signal Group”-oknak nevezzük.

Az alkalmazás réteg alatt található interakciós és hálózati rétegben a felülről érkező adatelem neve „I-Signal” (**I**nteraction Layer Signal). Az „I-Signal”-ok már az adott ECU-n belül relevánsak csupán. Minden „System Signal”-hoz tartozik egy „I-Signal”, amelyeket a COM modul „I-Signal-I-PDU”-kba csomagol. Egy „I-Signal-I-PDU” több „I-Signal”-t is tartalmazhat. Az „I-Signal”-ok továbbá hozzá lehetnek rendelve a PDU-k egy speciális fajtájához, ezek az NM-PDU-k (**N**etwork **M**anagement PDU). Ez a PDU-típus köti össze a ComM modult az egyes hálózatmenedzsment modulokkal (az „I-Signal” tartalmazza a PN információt).

Az interakciós és hálózati réteg tartalmaz még szimpla „I-PDU”-kat (Interaction Layer PDU), amelyek az interakciós rétegben található modulok közötti kommunikációnak az adategységeit tartalmazzák, ezen felül ebben a rétegben még beszélhetünk N-PDU-król (**N**etwork Layer PDU), amely a transzport protokoll és interfész modulok közötti adategységeket jelenti. Ezeken felül még találkozhatunk néhány speciálisabb PDU típussal, mint a DCM-I-PDU, XCP-PDU, Multiplexed-I-PDU valamit UserDefined-PDU.

Az interakciós és hálózati réteg alatt található adatkapcsolati rétegben pedig L-PDU-król (Data **L**ink Layer PDU) beszélhetünk. Ezek az adatcsomagok kötik össze az egyes interfészeket a hozzájuk tartozó megfelelő driverrel. Az adatkapcsolati réteg MAC alrétegében pedig frame-ek formájában kerülnek továbbításra az adatcsomagok [14][16].

5. Hálózatmenedzsment fejlődése

5.1. OSEK hálózatmenedzsment

Az AUTOSAR hálózatmenedzsment moduljainak - több AUTOSAR modulhoz hasonlóan - az OSEK szabványban rögzített hálózatmenedzsment specifikációk szolgáltatják az alapját, ennek a szabványnak a továbbgondolásával, valamint az új igényeket kielégítő feladatok megfogalmazásával. Az OSEK szabványban definiált hálózatmenedzsment azzal a céllal született meg, hogy egy szabványosított eszközkészletet nyújtson a kommunikáció biztonságának és megbízhatóságának növelése érdekében. Tehát a szabványban csak az eszközkészlet került rögzítésre, nem pedig a hálózatmenedzsment logikai lépései. A szabványban definiált funkcionalitás többek között a kommunikációs busz állapotának nyomon követése, valamint koordinálása, illetve a hálózat többi eszközének az észlelése. Az OSEK szabványban a hálózatmenedzsmentnek két fajtáját különböztették meg. Az egyik a "Direct Network Management", amelyben a hálózatmenedzsment dedikált üzenetekkel valósul meg, míg a másik változat az "Indirect Network Management", ahol a hálózatmenedzsment, nem egy dedikált üzenetben valósul meg, ez azt jelenti, hogy a különböző ECU-kon futó alkalmazások, egymás közötti kommunikációval valósítják meg a hálózatmenedzsmentet [22].

5.2. AUTOSAR hálózatmenedzsment

Az AUTOSAR szabványba az OSEK szabványból csak a "Direct Network Management" funkcionalitás lett átvéve és továbbdolgozva. A kezdetekben (AUTOSAR 2.0) a hálózatmenedzsmentben csak a Communication Mode Manager (ComM) és egy Generic Network Management (GNM) modulok voltak megtalálhatóak. Későbbiekben a ComM modulból a kommunikációs kontrollert és transceivert vezérlő funkciók kiváltak és létrehozták a <Bus>Sm modulokat, míg a GNM pedig szétvált egyrészt a NM modulra, amely egy interfészt nyújt a ComM-nek, valamint a hálózat kikapcsolásának logikáját tartalmazza, másrészt a GNM modulból váltak ki a <Bus>Nm modulok. Ez a szétválás az AUTOSAR 3.0-ás változatában történt meg.

5.3. Részhálózatok megjelenése

Az AUTOSAR 3.2/4.0 - bizonyos tekintetben párhuzamos szabványverziók - változataiban jelenik meg először a “Partial Networking”, azaz a részhálózatok kezelése. Ezt a funkciót úgy kell elképzelni, hogy a teljes kommunikációs hálózat összes hálózati eszközéből tetszőleges halmazok kiválaszthatóak (PNC – Partial Network Cluster) és az azonos halmazba tartozó eszközök közösen koordinálhatóak, függetlenül attól, hogy fizikailag melyik kommunikációs buszra kapcsolódnak.

Az AUTOSAR 3.2/4.0-ás szabványból a részhálózatok menedzselésének megértése nehézkes, sok esetben tartalmaz nem teljesen specifikált funkciókat, illetve egymásnak ellentmondó követelményeket. De a szabvány fejlődésével ennek a funkciónak a specifikációja is folyamatosan fejlődik és az AUTOSAR 4.2-es (jelenleg legfrissebb) változata már egész jól definiálja ennek a funkciónak a működését. A részhálózatok kezelése alapvetően már elő volt készítve a hálózatmenedzsment üzenetekben definiált “Felhasználói adatmező (User Data Field)” biztosításával, azonban ennek a funkciónak a gyakorlati használatához szükségeses volt az “intelligens” transceiverek megjelenése.

A 3.2/4.0 AUTOSAR szabvány megjelenése előtt, a hálózatmenedzsmenttel csak egy teljes hálózat és ezzel együtt a buszra kapcsolódó összes ECU kikapcsolása volt lehetséges. Ez a gyakorlatban azt jelentette, hogy a járműben menet közben legfeljebb csak az alhálózatok kikapcsolására volt lehetőség, és nagyobb buszok kikapcsolására maximum a jármű parkoló állása során nyílt lehetőség. Azonban a járművekben egyre jelentősebb mennyiségű elektronikai egység jelenik meg, amely magával vonzza a jármű növekvő elektronikus teljesítmény igényét, amely növekvő üzemanyag fogyasztást is jelenthet, illetve az elektromos autóknál pedig az amúgy is korlátozott elsődleges energiaforrás használatát. A részhálózat menedzsment ezzel szemben azt jelenti, hogy nemcsak egy teljes busz és a hozzá tartozó összes elektronikus egységgel kapcsolható ki, hanem a hálózat tetszőleges ECU-ja lekapcsolható. Ez csakis a buszra kapcsolódó intelligens transceiverek segítségével valósítható meg.

A hagyományos transceiverek csak a hálózatra illesztik az ECU-t, esetleg a hálózaton megjelenő bármilyen aktivitás hatására ébreszti a processzort. Ezzel szemben az intelligens változatuk az alapvető funkcionalitáson felül képesek értelmezni a

hálózati üzeneteket, ezzel lehetőség nyílik, hogy egy előre beállított bitsorozatra vagy üzenetre adjanak ébresztés jelzést a processzornak.

Ezzel a megoldással egy ECU alvó állapotban maradhat, úgy is, hogy a buszon közben folyamatosan zajlik a kommunikáció, és csupán akkor ébred fel, ha rá valóban szükség van. Így egy ECU fogyasztása akár egészen a transceiver fogyasztásáig csökkenthető.

A részhálózatok menedzsmentje természetesen nem azt jelenti, hogy a jármű összes elektronikai egységének aktivitását egyenként szabályozzuk, hanem - mint azt a funkcionalitás neve is mutatja - logikai részhálózatokat lehet definiálni, és az ezekbe tartozó elektronikus egységeket lehet egységesen koordinálni. Ezzel lehetőség nyílik arra, hogy a funkcionálisan összetartozó ECU-k a jármű bármely fizikai hálózatára kapcsolódjanak, mégis, ha nincs szükség az adott funkcióra, akkor az adott funkcióhoz tartozó eszközök kikapcsolhatóak legyenek.

6. Network Management

A hálózat menedzsment funkciók a gyakorlatban úgynevezett hálózat menedzsment üzenetek (Network Management Message) segítségével valósulnak meg. Ezekre az üzenetekre a szabvány még hivatkozik röviden NM-Message vagy NM-PDU néven is. Az időosztásos buszok esetén - mint a FlexRay vagy a LIN - ezek az üzenetek periodikusak, és azok tartalma alapján döntik el a hálózatra csatlakozó eszközök, hogy elindíthatják-e a kikapcsolási folyamatokat vagy sem. Az eseményvezérelt hálózatokon - mint a CAN vagy Ethernet - a Network Management Message-ek jelenléte a buszon jelzi a node-ok számára a busz szükségességét. Amennyiben ilyen NM-Message egy bizonyos ideig nem érkezik a buszról, akkor kezdik meg a kikapcsolási folyamatokat.

Az eddigieket összefoglalva a Network Manager modulok felelősek az egyes buszok kikapcsolási lehetőségének vizsgálatáért és a kikapcsolási folyamat kezdeményezéséért. Ezen műveletek eléggé összetettek, melyekben az egyes Network Manager-ek különböző szerepeket tölthetnek be. Ezért a teljes hálózatmenedzsment funkciók megértéséhez, és ezeknek a szerepeknek megismeréséhez a következő alapfogalmak ismerete szükséges.

A hálózatok a szabvány szerint különböző klaszterekbe (cluster) sorolhatóak. Ezek a klaszterek rendszer szinten definiáltak. Egy node akár több különböző kommunikáció buszhoz is csatlakozhat, és ezek közül akár több busz is tartozhat azonos klaszterbe.

A Network Management funkciók mindig az azonos klaszterbe tartozó hálózatokat érinti. Tehát, ha egy ECU több klaszterbe tartozó buszokhoz csatlakozik, akkor az ECU-n futó Network Manager-ek egyszerre csak az azonos klaszterbe tartozó buszokat kapcsolhatnak ki. Ezekre a klaszterekre azért van szükség, mert egy ECU lehet akár átjáró (gateway) is, ilyenkor az egyik hálózatról a másikra továbbít üzeneteket. Ebben az esetben csak akkor kapcsolhatóak le az egyes buszok, ha egyik sem szükséges tovább, különben előfordulhat olyan konfliktus, hogy az egyik busz lekapcsol, míg a másik még adatot küldene.

Mivel az azonos klaszterbe különböző busztípusok tartozhatnak, ezért az AUTOSAR szabvány lehetőséget nyújt a szinkronizált kikapcsolásra. Ilyen szituáció

akkor alakulhat ki, ha azonos klaszterbe például CAN és FlexRay hálózat is tartozik. Ilyenkor a FlexRay hálózathoz szinkronizáltan kell kikapcsolni a CAN buszokat is. A szinkronizált kikapcsolásra azért van szükség, hogy elkerülhető legyen az a konfliktus, hogy az egyik busz már kikapcsolt, míg egy másik buszon esetlegesen újra valamilyen üzenet érkezik. Az, hogy egy klaszter szinkronizált kikapcsolású-e vagy sem, az egyes ECU-kon belül konfigurálható, mint ahogyan az is, hogy melyik csatornához szükséges szinkronizálni az adott klaszterbe tartozó buszokat.

Az egyes node-ok a hálózaton lehetnek koordinátorok (NM-Coordinator). Csak ezek a node-ok rendelkezhetnek a buszok kikapcsolásáról. Ez a paraméter hálózatonként változhat, tehát előfordulhat, hogy egy node hálózat koordinátora az egyik hálózatnak, de egy másiknak nem.

Egy node az egyes csatornákon aktív vagy passzív szerepet tölthet be. Passzív esetben, a node az adott csatornán nem rendelkezhet a busz használatának szükségességéről, vagyis az aktív node-ok küldhetnek csak NM-Message-eket, vagy szavazhatnak azokban a busz ébrentartásáról. A passzív node-ok csak fogadják ezeket az üzeneteket, és teljesítik a kérést [13].

A következőkben röviden bemutatásra kerül az AUTOSAR szabványban definiált Network Management Message-ek. Általánosságban elmondható ezekről, hogy nagyban hasonlítanak egymásra, esetenként (főképpen a FlexRay esetén) előfordulhatnak kisebb eltérések. A Network Message-ek általában nyolc bájtból állnak, aminek első két bájtja a Control Bit Vektor (CBV), ami a hálózat vezérlésére szolgáló információt tartalmazza, és a Source Node Identifier (NID, amely az üzenetek küldő node egyedi azonosítóját tartalmazza. Ezek után még hat bájt felhasználói adatmező (User Data) található, amely a Partial Networking során kerül felhasználásra.

A CAN hálózaton a Network Management funkciók a buszon időközönként megjelenő, NM-Message-ekkel valósulnak meg. Alapesetben csak az aktívan koordináló node-ok küldhetnek ilyen üzeneteket. Az AUTOSAR szabvány szerint, amennyiben ilyen NM-Message nem jelenik meg a hálózaton egy előre definiált ideig, akkor egyik, a hálózathoz kapcsolódó node sem igényli a hálózatot. A CAN NM-Message-ban a CBV és Source Node Identifier bájtok, az első és második bájtpozícióban kell elhelyezkedniük, azonban a sorrendjük felcserélhető, esetenként elhagyhatóak. Ilyenkor User Data bájt kerül a helyükre [7][13].

A CAN NM-Message felépítése a következő:

	7. bit	**6. bit**	**5. bit**	**4. bit**	**3. bit**	**2. bit**	**1. bit**	**0. bit**
7. bájt	5. Felhasználói adat (User Data)							
6. bájt	4. Felhasználói adat (User Data)							
5. bájt	3. Felhasználói adat (User Data)							
4. bájt	2. Felhasználói adat (User Data)							
3. bájt	1. Felhasználói adat (User Data)							
2. bájt	0. Felhasználói adat (User Data)							
1. bájt	Hálózatvezérlő adatok (CBV)							
0. bájt	Küldő azonosító (Source Node Identifier)							

6.1. ábra, CAN NM-PDU felépítése

	7. bit	6. bit	5. bit	4. bit	3. bit	2. bit	1. bit	0. bit
CBV	Foglalt	Részhálózat információ (Partial Network Information)	Foglalt	Aktív Ébresztés (Active Wakeup)	Hálózati koordinált alvás (NM-Coordinator Sleep Ready)	Foglalt	Foglalt	Ismételt adás kérés (Repeat Message Request)

6.2. ábra, CAN Control Bit Vector

Bit 0: Ismételt adás kérés (Repeat Message Request)

0: Az ismételt üzenet adás nem kért

1: Ismételt üzenet adás kérés

Bit 3: Hálózati koordinált alvás bit (NM-Coordinator Sleep Bit)

0: Szinkronizált kikapcsolás indításának tiltása

1: Szinkronizált kikapcsolás indítása

Bit 4: Aktív ébresztés Bit (Active Wakeup Bit)

0: A node nem ébresztett a hálózat által (passzív ébresztés)

1: A node a hálózat által ébresztett (aktív ébresztés)

Bit 6: Részhálózati információ bit (Partial Network Information Bit) PNI

0: A NM-Message részhálózat információt nem tartalmaz

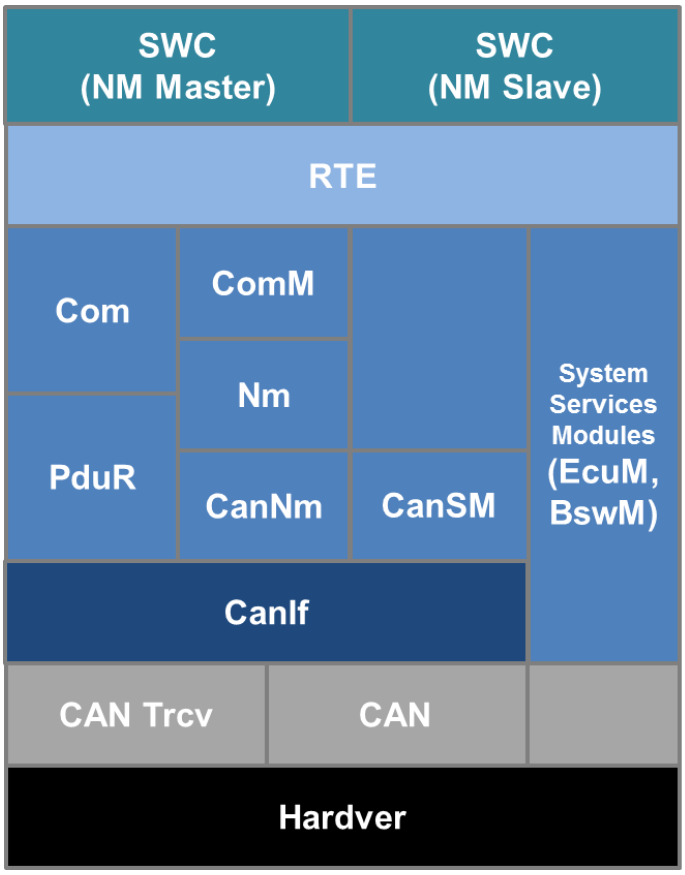
1: A NM-Message részhálózat információt tartalmaz

Bit 1, 2, 5, 7 nem használt a későbbi fejlesztésekre fenntartott.

7. Hálózatmenedzsment blokk felépítése

7.1. Az egyes modulok feladata

Az AUTOSAR szabványban definiált hálózat menedzsment funkcionalitást alapvetően három – a kommunikációs blokkban található – modul látja el, illetve ezeken a modulokon felül még alkalmazások is részt vesznek a hálózatmenedzsmentben. A BSW-ben található modulok látják el a hálózatmenedzsment alapvető funkcionalitását, pontosabban elérhetővé teszik a hálózatmenedzsment szolgáltatást, míg a szoftverkomponensek tartalmazzák azokat a logikai döntéseket, amelyek alapján az egyes hálózatok kikapcsolnak. A következő szakaszban először a *CAN hálózatmenedzsment* blokkjában található modulok működéséről lesz szó, majd röviden bemutatásra kerülnek azok a modulok is, amelyek a hálózatmenedzsment blokkal közvetlen kapcsolatban vannak.



7.1. ábra, Hálózatmenedzsment blokk és környezete

7.1.1. Communication Manager (ComM)

A ComM (Communication Manager) modul szolgáltatta interfészeken keresztül kérvényezhetik az alkalmazások (SWC) illetve BSW-k a kommunikáció´elindítását, illetve leállítását a kommunikációs csatornákon. Ezen az interfészen keresztül az egyes felhasználók jelezhetik, hogy igényt tartanak- a kommunikációs csatornára. Ezt az igényt a ComM modul a CanSM modul segítségével érvényesíti. Amennyiben egyetlen egy user (alkalmazás vagy BSW) sem tart igényt a továbbiakban a csatornára, valamint külső igény sincs a kommunikáció fenntartására, úgy a csatornán megindulhat a kikapcsolási folyamat.

Azt meg kell jegyezni, hogy a ComM modul az NM és CanNm modulok nélkül önálló szerepet tölt be a BSW-n belül, ugyanis ebben az esetben is ezen a modulon keresztül lehet a kommunikációs igényeket jelezni, illetve a ComM modul maga is tartalmaz egyszerű, önálló hálózatmenedzsment szolgáltatást. Ezek a hálózatmenedzsment szolgáltatásoknak két fajtája létezik. Az elsőben, ha valaki egyszer elindított egy kommunikációt egy csatornán, akkor az a csatorna "Full Communication" módba kerül és ott is marad az ECU kikapcsolásáig. Pontosabban megfogalmazva ezen a módon belüli "Requested" és "Ready Sleep" állapotok között változik a csatorna állapota. A második esetben már valódi hálózatmenedzsmentről beszélhetünk, itt ki is kapcsolnak a csatornák, amennyiben a csatorna egy adott ideig "tétlen" volt a "Ready Sleep" állapotban.

Ugyancsak a ComM modul szabványában találhatjuk a PNC állapotgépét, tehát a ComM modul a felelős a PNC-k állapotának a nyilvántartásáért, ezzel együtt, a PNC-hez tartozó csatornák állapotának vezérléséért is. Ha egy PNC "Requested" állapotban van, akkor a hozzá tartozó csatornáknak is "Full Communication" módban kell lenniük.

Összefoglalva a ComM modul feladata a különböző felhasználók - ezek lehetnek SWC-k, BSW-k (pl.: DCM) – kommunikációs igényeinek illetve a kommunikációs csatorna tiltásainak összegyűjtése és ezek alapján a csatorna állapotának megválasztása és prezentálása a CanSm modulnak, ezzel biztosítva a kommunikációt az adott csatornán, illetve a hálózatról érkező kommunikációs igények jelzése az alkalmazások felé [10].

7.1.2. Network Management Interface

A Network Management modul funkcionalitása két elkülönülő egységre bontható. Egyrészről az interfész szerepét tölti be a ComM és az alatta elhelyezkedő kommunikációs protokoll specifikus hálózatmenedzsment modulok között. Azaz a ComM modul kéréseit továbbítja a kommunikációs csatornához tartozó megfelelő hálózatmenedzsment modulnak – CAN esetén a CanNm-nek –, míg a hálózatmenedzsment modulok jelzéseit továbbítja a ComM modulnak.

Másrészt a Network Management modul a csatornák kikapcsolási folyamatait koordinálhatja, ezt a funkciót az átjáró (gateway) és a központi (central) ECU-k használják. Ezek az egységek több hálózathoz is csatlakoznak és azok között adatokat közvetítenek. A koordinátor funkciók szerepe a különböző hálózatok szinkronizált kikapcsolását megvalósítani.

A koordinátor algoritmusának képesnek kell lenni arra, hogy különböző busztopológiák esetén is megfelelően működjön, valamint az NM koordinátornak együtt kell működnie több másik NM koordinátorral ugyanabban a klaszterben. Egy busz maximum egy klaszterhez tartozhat. Az NM koordinátornak csak az ébren levő buszokat kell vezérelnie, azokat, amelyek már alvó állapotban vannak, csak monitoroznia szabad, de vezérelnie tilos [9][13].

7.1.3. CAN Network Management

Ebben a szakaszban a CAN Network Management modul kerül bemutatásra, mivel a demonstrációs projektnek alapvetően ezen a kommunikációs protokollon kell működnie, azonban a többi Network Management modul (FrNm, LinNm és UdpNm) működése is lényegében ugyanilyen, néhány apró eltéréssel.

A CAN Network Management modul a felelős a CAN hálózat menedzseléséért. A CAN hálózatmenedzsment funkciók alapja egy elosztott erőforrású stratégia, miszerint minden node felelős a busz alvó állapotának eléréséért. Minden node veszi a hálózatmenedzsment üzenteket és ezek alapján az egyes node-ok eldöntik, hogy szükséges vagy lekapcsolható a busz, illetve, hogy saját maga kikapcsolható-e, és amikor a busz összes node-ja lekapcsolt, akkor a busz is kikapcsolt állapotba kerül.

A CAN Network Management funkciók alapját képezi az elviekben periodikusan – gyakorlatban a buszterheltségtől függően – érkező Network Management üzenetek, amelyeket minden a hálózatra kapcsolódó node megkap. A küldő node a NM-Message-k küldésével jelzi az üzenetet fogadó node-ok számára, hogy a klaszterbe tartozó hálózatok szükségesek, ezért ébren kell tartani. Ha az összes node, amely az adott klaszterbe tartozik kész kikapcsolni a buszt, akkor az utolsó node is befejezi a NM-Message-k küldését. Végezetül, ha a node-ok észlelik, hogy a buszon nem érkezett üzenet egy meghatározott ideig, akkor a csomópontok belépnek „Bus Sleep Mode”-ba és kikapcsolják saját hálózati kontrollereiket, illetve transceivereiket.

Ha egy node a menedzselt hálózati klaszterben újra igényt tart a buszra, akkor felébresztheti azt, amennyiben újra NM-Message-eket kezd küldeni.

Összefoglalva a CAN Network Management modulnak a következő két specifikációs pontot kell teljesítenie:

- Minden node-nak a koordinált klaszterben periodikusan NM-Message-eket kell küldeni, ha igényli a hálózat használatát. Ha nincs szüksége a hálózatra, akkor le kell tiltania az ilyen üzenetek küldését.
- Amennyiben egy node nem vesz NM-Message-t a hálózatról egy előre definiált ideig, akkor a node-nak be kell lépni az alvó „Bus Sleep” módba [7][13].

7.2. Network Management blokk környezete

Ebben a szakaszban röviden bemutatására kerülnek azok a modulok, amelyekkel a hálózatmenedzsment modulok interakcióba lépnek, ezzel segítve a modulok működésének megértését.

7.2.1. Runtime Enviroment (RTE)

Az AUTOSAR szoftverarchitektúra bemutatásakor már szóba került ennek a szoftverrétegnek a funkcionalitása. Hálózatmenedzsment szempontból a feladata, hogy összekapcsolja a hálózatmenedzsment alkalmazásokat a hálózatmenedzsment blokkal, azon belül is a ComM modullal. Ugyancsak ezen a rétegen keresztül jelzi a ComM modul az alkalmazásoknak, ha a hálózat be- vagy kikapcsolt [2].

7.2.2. PDU Router (PduR)

A PDU Router feladata alapvetően az adatcsomagok (PDU-k) irányítása. Hálózatmenedzsment szempontjából akkor kerül fókuszba, amikor a NM Message-ben aktív a „User Data” mező. Ugyanez az adatmező használatos a részhálózatok állapotának jelzésére is. A „User Data” mezőnek az értékét a CanNm modul a PduR-en keresztül kapja meg vagy egy alkalmazástól vagy a ComM-től [5].

7.2.3. Communication (Com)

Az alkalmazások egymás között szignálokkal cserélnek adatokat, nem adatcsomagokkal, mint ahogyan az BSW között történik. Nagyvonalakban a Com feladata ezeknek a szignáloknak és a PDU-knak az összerendelése, bufferelése, küldése és fogadása. Ez a modul is akkor nyer jelentőséget a hálózatmenedzsmentben, amikor a „User Data” mező aktív. Ebben az esetben ebbe a modulba a „User Data” értéke az RTE-n keresztül az alkalmazásoktól érkezik, illetve részhálózatok kezelése esetén a részhálózatok állapotának értéke ComM-től érkezik szignálok formájában [6].

7.2.4. CAN Interface (CanIf)

A CAN Interface, elsősorban multiplexer szerepet játszik azzal, hogy egységes elérhetőséget nyújt tetszőlegesen sok transceiverhez és kontrollerhez. Tehát ezen a modulon keresztül küldhetők és fogadhatóak a CAN üzenetek, így a CAN hálózatmenedzsment üzenetek is, valamint állíthatóak a hálózati eszközök állapotai [3].

7.2.5. CAN State Manager (CanSm)

A CAN State Manager feladata a kommunikációs kontrollerek, illetve a transceiverek állapotának vezérlése és nyilvántartása. Ezen a modulon keresztül képes a ComM a kommunikációs kontrollert be- illetve kikapcsolni [4].

7.2.6. ECU State Manager (EcuM)

ECU State Manager az ECU állapotának kezeléséért felelős. Hálózatmenedzsment szempontjából az ECU hálózati ébresztése során nyer jelentőséget, ugyanis az ECU ébredése után ebben a modulban történik az ébredés okának kiértékelése. Tehát ha hálózati ébresztés volt, akkor az EcuM lekérdezi a transceivert, hogy a hálózati aktivitás miatt ébred-e fel az ECU és ebben az esetben ezt jelzi a ComM-nek, amely így képes bekapcsolni a megfelelő kommunikációs csatornát.

7.3. Network Management modulok együttes működése

Az előző szakaszban bemutatásra került egyenként az egyes hálózatmenedzsment modulok működése, valamint a hálózatmenedzsment blokkal közvetlen kapcsolatban lévő modulok szerepei. Ebben a szakaszban pedig a modulok együttműködéséről lesz szó, először csak általánosan, majd konkrét példákon keresztül kerül bemutatásra a hálózatmenedzsment blokk működése.

Ahogyan az már ismertetésre került, a ComM modul összefogja a különböző igényeket egy kommunikációs csatornán, legyen szó akár a SWC, BSW vagy PNC igényről. Miután ezek közül bármilyen kérés beérkezik a ComM modulba, a csatorna állapotgépe "Full Communication" módba vált, majd ezek után a ComM modul kéri az CanSM modult, hogy a kommunikációs kontrollert kapcsolja be, illetve jelzi a NM modulnak, hogy a csatorna aktív állapotba került. Amennyiben az ECU aktív koordinátor, akkor az NM modul eltárolja, hogy a csatornára egy felsőbb kérés érkezett, majd a jelzést továbbítja a CanNm modulnak, amely ezek után a csatornához tartozó állapotgépét kommunikációs módba váltja. Ha a CanNm modul aktív szerepet tölt be, akkor periodikusan NM-Message-eket küld a csatornán, illetve figyeli a csatornáról érkező NM-Message-eket. Amikor már sem az ECU-nak, sem a hálózat többi szereplőjének sincs szüksége a hálózatra, úgy megindulhat a kikapcsolási folyamat. Ennek egyik indikátora CAN hálózatok esetén, hogy a hálózaton megszűnik a NM-Message-ek jelenléte. Ezt a CanNm modul detektálja és jelzi az NM modulnak. Amennyiben az NM modul aktív koordinátor, úgy ebben az esetben elindítja a kikapcsolási algoritmust, ellenkező esetben ennek a jelzésnek a hatására még nem történik semmi. Egy bizonyos idő eltelte után azonban a CanNm állapotgépe átvált „Prepare Bus Sleep" állapotba és ennek a jelzésnek a hatására a ComM-ben a csatorna állapotgépe is „Silent" állapotba vált. Ekkor már nincs kommunikáció a buszon, csupán egy "védelmi" időt biztosít, amíg a hálózatra csatlakozó eszközöknek lehetőségük van jelezni, hogy mégse kapcsoljon ki a hálózat. Ennek az időnek a letelte után a csatorna állapotgépe mind a CanNm mind pedig a ComM modulba „Sleep", illetve „No Communication" állapotba kerül és ezzel együtt a ComM jelzi a CanSm modulnak, hogy kikapcsolhatja a kommunikációs kontrollert. Illetve a BswM-nek is jelzésre kerül, hogy a csatorna kikapcsolt, így ha a megfelelő feltételek teljesülnek, akkor a teljes ECU is kikapcsolhat, vagy alvó állapotba kerülhet.

7.4. Hálózatmenedzsment példák

7.4.1. NM-Slave ECU

Talán ez az üzemmód, ami a legjellemzőbb a járműben megtalálható ECU-kra. Ilyenkor ugyanis az ECU-k felébrednek a buszon érkező üzenetekre és aktiválják kommunikációs csatornáikat és ebben az állapotban maradnak mindaddig, amíg a hálózat aktív. Ennek a menete, hogy a hálózati aktivitást észleli a transceiver majd a mikrokontrollernek jelzi ezt az eseményt, minek hatására az felébred az alvó állapotból. Az ébresztést okozó eseményt az EcuM kezeli le, majd jelzi a ComM-nek, hogy az adott kommunikációs csatornán ébresztés történt. Ennek hatására a ComM csatorna állapotgépe „Full Communication" módba, azon belül is „Ready Sleep" állapotba kerül, ami azt jelenti, hogy a node kommunikálhat a csatornán, amíg ebben az állapotban van, de nem tartja ébren a hálózatot. Ilyenkor a ComM a CanSM modul segítségével bekapcsolja az ECU kommunikációs kontrollerét, és innentől kezdve az ECU képes a hálózati kommunikációra. A kontroller bekapcsolásával egy időben a ComM jelzi a CanNm-nek, hogy az adott kommunikációs csatorna be lett kapcsolva. NM-Slave ECU esetén a NM modul csak interfész funkcionalitást valósít meg.

Ezek után a CanNm-ben az állapotgép „Repeat Message" állapotba és onnan „Ready Sleep" állapotba vált. Tehát az ECU bármikor kész kikapcsolni, ha a hálózati kommunikáció befejeződik. Az állapotgép mindaddig ebben az állapotban marad, amíg a hálózaton érkeznek hálózatmenedzsment üzenetek az „NmTimeoutTime"-on belül. Ha a hálózaton nem érkezik a határidőn belül újabb NM-Message, úgy a CanNm állapotgépe "Prepare Sleep" állapotba vált és ezt jelzi a ComM számára is. A ComM csatorna állapotgépe ennek hatására "Silent" állapotba vált és ezt jelenti a CanSM modulnak is, amely pedig kéri a CanIf-től, hogy fejezze be az üzenetek küldését (Tx buffereket is ürítse). Ez után a pillanat után már az ECU nem képes új üzenetek küldésére, csak hallgatózhat a hálózatról. Végezetül egy bizonyos időkeret után a CanNm állapotgépe "Bus Sleep" állapotba vált –amennyiben ez idő alatt nem érkezett újabb hálózatmenedzsment üzenet – és ezt jelzi a ComM-nek is. A ComM modul csatorna állapotgépe is "No Pending Request" állapotba kerül és a CanSM modulnak jelezve ezt, az kikapcsolja a kommunikációs kontrollert is. Ezen felül a ComM jelzi a BswM-nek is, hogy a csatornák kikapcsoltak, így ha nincs egyéb feladata az ECU-nak, a processzor is kikapcsolhat, vagy alvó állapotba kerülhet.

7.4.2. NM-Master ECU

A második példában hálózatmenedzsment szempontjából sokkal kevésbé jellemző ECU típus kerül bemutatásra. Ez a vezérlő az, amely koordinátor szerepet tölt be hálózaton, tehát ez az egység képes ébreszteni a buszra kapcsolódó többi node-ot is, illetve kezdeményezni a kommunikációs hálózat kikapcsolását.

Az ébresztés menete viszonylag egyszerűen zajlik. Amikor egy alkalmazásnak kommunikálni kell egy másik ECU-n található alkalmazással, akkor ez egy ”User Request” formájában kerül jelzésre a ComM modulnak. A ComM-ben ennek hatására a „user”-hez tartozó csatorna állapotgépe “Network Requested” állapotba lép és ezzel egy időben a CanSM-en keresztül bekapcsolja az ECU kommunikációs kontrollerét és transceiverét, valamint jelzi a CanNm modulnak is, hogy a busz egy belső kérés hatására kapcsol be. Ilyenkor a NM modul csak továbbítja ezt a kérést. A CanNm modul állapotgépe ilyenkor “Repeat Message” állapotba kerül és periodikusan megkezdi a NM-Message-ek küldését, aminek a hatására a többi ECU is bekapcsol az előző szakaszban leírtak szerint. Amikor a hálózat többi résztvevője is bekapcsolt, akkor a többi NM-Aktív (de nem NM-Master) egység is elkezdi az NM-Message-ek küldését, ezzel jelezve, hogy a kommunikációs busz készen áll a használatra és így elkezdődhet a kommunikáció.

A kikapcsolás menete ennél egy kissé bonyolultabban zajlik. Először is az alkalmazás nem a kikapcsolást kéri, hanem csak azt jelzi, hogy már nincs szüksége a csatornára –mivel a hálózat többi egysége még kommunikálhat egymással -, amely a „User Release”-nek felel meg a ComM-ben. Ezek után a ComM csatorna állapotgépe “Ready Sleep” állapotba kerül, amit jelez az NM modulnak. Ebben a pillanatban az NM modul – ha aktív koordinátor – nem továbbítja ezt a jelzést a CanNm-nek, hanem abban feldolgozásra kerül, ami azt jelenti, hogy megvizsgálja az NM modul, hogy az azonos klaszterbe tartozó, összes csatorna készen áll-e az alvó állapotra. Amennyiben ez a szituáció áll fent, úgy megkezdődhet a klaszter kikapcsolása. Ha a klaszterben van idővezérelt kommunikációs csatorna, akkor a kikapcsolási folyamat elindításához egy szinkronizációs pont is szükséges. Ezután elindul a kikapcsolás késleltetése, ami azért szükséges, hogy ha alhálózatokat is tartalmaz a rendszer, ez idő alatt az alhálózatok kikapcsoljanak. Az időzítő lejárta után az NM modul elengedi a csatornákat és csak ezután válthat a CanNm modul állapotgépe “Ready Sleep” majd “Prepare Sleep” végül “Sleep” állapotba. A fő állapotok változását a CanNm modul jelzi a felette lévő

moduloknak is, aminek hatására a ComM modulban a csatorna állapotgépe is először “Silent Communication” majd “No Communication” módba vált, és ezeknek az állapotoknak megfelelő kontroller és transceiver állapotokat kéri a CanSm modultól.

7.4.3. Részhálózatokat tartalmazó rendszer

Ez a példa egy más aspektusát mutatja meg a hálózatmenedzsmentnek, amikor is részhálózatokat is tartalmaz a rendszer. Ebben az esetben a hálózat logikai részhálózatokra van felosztva. Ennek a funkcionalitásnak a megfelelő működéséhez elengedhetetlenül szükséges, hogy az ECU “intelligens” transceiveren keresztül kapcsolódjon a kommunikációs buszhoz. Ebben az esetben a hálózatmenedzsment lényegében ugyanúgy működik, mint az előzőekben leírt szituációkban, azzal a különbséggel, hogy a hálózatmenedzsment üzenetek tartalma alapján szelektálva vannak az üzenetek, így nem minden üzenet gyakorol ugyanolyan hatást a hálózatmenedzsment blokk moduljaira. Az intelligens transceiver azért szükséges, mert ez egy olyan eszköz, amelyet fel lehet programozni, hogy milyen üzenet vagy bitsorozat hatására küldjön ébresztésjelzést a mikrokontrollernek, így biztosítva azt, hogy egy ECU kikapcsolt vagy alvó állapotba maradjon, miközben a buszon valamilyen kommunikáció zajlik.

A CanNm modulban a különbség a hagyományos hálózatmenedzsment üzenetekhez képest az, hogy részhálózatok esetén le kell szűrnie, hogy a fogadott üzenet az ECU-nak megfelelő részhálózatnak szól-e, amit az üzenetben megtalálható megfelelő PNC bit jelöl. Ha a fogadott üzenetben be van állítva az ECU részhálózatának megfelelő PNC bit, akkor a CanNm a normál módon feldolgozza az üzenetet, egyébként eldobja. Mivel egy ECU több részhálózathoz is tartozhat, így a CanNm modulban külön időzítőt kell nyilvántartani az egyes részhálózatokra, amelyek lényegében ugyanolyan időzítőknek felelnek meg, mint a normál hálózatmenedzsment esetén. Az így nyilvántartott részhálózat állapotok minden változását jeleznie kell a ComM modulnak. Ez az üzenet, a CanNm $\rightarrow$ PduR $\rightarrow$ Com $\rightarrow$ ComM útvonalon kerül továbbításra. A ComM így nyilván tudja tartani az egyes részhálózatok állapotát és annak megfelelően koordinálja az egyes kommunikációs csatornákat.

Tehát, ha az ECU része egy részhálózatnak, akkor a ComM-ben a csatorna bekapcsolása esetén a PNC állapotgépe „PNC Requested” állapotba kerül. Ilyenkor a ComM beállítja a PNC-hez tartozó bitet az EIRA vektorban (ebben a vektorban vannak

tárolva a PNC belső, külső kéréseinek összessége), majd a vektor értékét szignál formájában átadja a Com-nak. Amikor a CanNm NM-Message-et küld, akkor elkéri ennek a vektornak az értékét a Com-tól a PduR-en keresztül, az ott található PNC biteket beállítja az NM-Message-ekben is. Ezen felül elindít egy időzítőt, aminek a lejártakor törölni kell a PNC bitet, amennyiben ez idő alatt nem érkezik olyan NM-Message a buszról, amiben a megfelelő PNC bit be van állítva. Összefoglalva a CanNm-ben találhatóak azok az időzítők, amelyeknek a lejártakor az egyes PNC-k kikapcsolása megindul, kivéve, ha valamilyen esemény újra nem indítja ezen időzítőket.

Amikor lejár egy PNC-hez tartozó időzítő, akkor azt a CanNm modul a PduR-en és Com-on keresztül jelzi a ComM-nek, aminek hatására a ComM-ben a PNC állapotgép először „PNC Ready Sleep” állapotba – amikor a lokális ECU-nak nincs szüksége a PNC-re -, majd „PNC Prepare Sleep” állapotba – ekkor már a hálózat többi eszközének sincs szüksége a PNC-re – kerül. Végezetül a ComM-ben a PNC állapotgépe „PNC No Communication” állapotba kerül egy bizonyos idő után. Eközben a csatorna állapotgépe a már korábban bemutatottak szerint kapcsol ki.

8. Integráció

Ebben a fejezetben a beágyazott oldali szoftver kerül bemutatásra. Az eddigiekben már szó esett az AUTOSAR szoftverarchitektúráról, aminek megfelelően épül fel az összeintegrált rendszer is, azonban ebben a fejezetben a megvalósított feladat gyakorlati oldaláról lesz szó. Először a beágyazott szoftverprojekt felépítésének rövid bemutatására kerül sor, majd ezután kifejtére kerül az egyes elemek megvalósítása egymásra épülésük szerint, a driver rétegtől felfelé haladva.

A beágyazott szoftverprojekt a következő forrásokból áll össze:

- Linker script
- Startup kód
- AUTOSAR szabványos BSW modulok
- BSW konfigurációk
- Stub kód
- Integrációs kód
- RTE
- Szoftverkomponensek

8.1. Linker script és startup kód

A feladat megvalósításához a cégnél már jól bevált beágyazott projektstruktúra került felhasználásra, amely magába foglalja a projekt összefordításához szükséges linker scripteket és startup kódot is, így ezek megvalósításával már nem kellett foglalkozni. A startup kódban megtörténik a mikrokontroller elsődleges inicializálása is.

8.2. AUTOSAR szabványos BSW modulok

Az elméleti áttekintés során már szó esett arról, hogy az egyes BSW moduloknak milyen szerep tulajdonítható a hálózatmenedzsment működése során, illetve a következőkben pedig részletesebben kifejtésre kerül ezeknek a moduloknak a konfigurációja. Összefoglalásként a projektben felhasznált modulok a következőek: McuDriver, PortDriver, DioDriver, CanDriver, CanTrcvDriver, CanIf, CanSM, CanNm, PduR, ComM, Com, EcuM, Os és Det, melyek később részletesen kifejtésre kerülnek.

8.3. BSW modulok konfigurációja

A BSW modulok – hogy a különböző alkalmazásokhoz igazodjanak – számos konfigurálási lehetőséget rejtenek magukban, így nem triviálisan egyszerű az egyes modulok felkonfigurálása, úgy, hogy azok egymással helyesen együttműködjenek. Ebben a szakaszban ennek a konfigurációnak az összeállításáról lesz szó. A konfiguráció az AUTOSAR Architect nevű modellező szoftverben történik, amellyel a kitöltött konfigurációs modellekből a konfigurációs forrásfájlok generálhatóak. Első lépésben létre kell hozni egy új „EcuValueCollection”-t, ami tartalmazza majd az azonos konfigurációhoz tartozó modulkonfigurációkat.

8.3.1. MCU Driver

Az Mcu Driver teljes nevén a „Microcontroller Unit Driver” felelős a mikrokontroller alapvető szolgáltatásainak kezeléséért, így például a különböző órajelek beállításáért. Tehát itt kell megadni a processzor, a perifériák órajel forrásait, illetve azok frekvenciáit (előosztók konfigurációit). Ezen felül ebben a modulban lehet felkonfigurálni, hogy a mikrovezérlő az alapvető futási módokon kívül milyen más módokat használhat. Az MPC5744P típusú mikrokontrollerben a „Halt” és „Stop” módon felül 7 különböző futási mód választható, amelyekhez külön órajelforrás és sebesség választható. A projektben az egyszerűség kedvéért az indulási módon kívül még két futási mód lett beállítva, az egyik a „RUN0”-ás mód, ahol a processzor a külső kristályról 180 MHz-en jár, míg a másik mód, az energiatakarékos módot szimuláló „RUN1”-es mód, ahol a processzornak a belső 16 MHz-es oszcillátorról szolgáltatja az órajelet.

8.3.2. Port Driver

Ez a modulban, mint ahogyan a nevéből is adódik a mikrokontroller portjainak a kezeléséért felelős. A konfigurációban definiálható az egyes portok iránya és funkciója, ugyanis az MPC5744P típusú mikrokontrollerben sok modern mikrovezérlőhöz hasonlóan az egyes fizikai kivezetéseihez több különböző funkció is konfigurálható. Ez a modul a demonstrációs projektben csak egyszer, az induláskor van kihasználva, hogy a megfelelő módon inicializálja a mikrokontroller lábait.

8.3.3. DIO Driver

Ezen a driveren keresztül érhetőek el a GPIO-nak konfigurált lábak, így tehát a ledek, a gombok és a „piggy”t vezérlő lábak, azaz a „piggy reset” és a „piggy sleep”, illetve a transceiver ébresztést jelző vonala a „wakeup”. Ennek a modulnak a konfigurációja csupán a digitális I/O lábak definiálásából áll.

8.3.4. CAN Driver

A CAN Driver a mikrovezérlő belső CAN kommunikációs kontrollerének driverét jelenti. Ennek a modulnak a konfigurációja során lehetőségünk van felkonfigurálni a mikrovezérlő mindhárom CAN perifériáját. Itt kell meghatározni többek között a CAN kommunikáció sebességét, illetve a hardveres bufferelést is. De ami a projekt szempontjából igazán fontos, hogy itt kell felsorolni az úgynevezett „CAN Hardware Object”-eket, amik megfelelnek a kommunikációs buszon megjelenő „CAN frame”-eknek. Minden „CAN Hardware Object”-nek van egy meghatározott iránya, tehát, hogy küldendő vagy fogadott üzenetről van szó, ezen felül itt kell megadni a „CAN frame ID”-t, illetve a hozzá tartozó maszkot – ezzel lehetőség van több „CAN frame”-et hozzárendelni ugyanahhoz a „Hardware Object”-hez.

Továbbá lehetőség van kiterjesztett vagy sztenderd CAN-ID használatának megadására, emellett egy prioritás is megadható, amely a feldolgozás során kerül előtérbe.

A demó projekt a következő 4 „CAN frame”-et használja:

Frame	CAN-ID	Leírás
RxNmFrame	40h	Más ECU hálózatmenedzsment üzenete
TxNmFrame	41h	Az ECU által küldött hálózatmenedzsment üzentek
SlaveDataFrame	80h	Slave üzentek frame-je
MasterDataFrame	81h	Master üzenetek frame-je

8.3.5. CAN Transceiver Driver

A CanTrcv a felhasznált transceiver drivere, amelynek konfigurációjában meg kell adni – amennyiben a transceiver támogatja a részhálózatok kezelését – a kommunikációs csatorna sebességét, „wakeup frame”-hez tartozó CAN-ID-t és maszkot, valamint a „Partial Network Mask”-ot is. Ez a demonstrációs projektben kulcsszerepet játszik ugyanis a hagyományos transceiverrel nem lehetne megvalósítani a szelektív ébresztést.

8.3.6. CAN Interface

Mint ahogyan már korábban is szó esett arról, a CanIf elfedi a felette elhelyezkedő modulok elől a kommunikációs kontrollereket és transceivereket. Ez a gyakorlatban azt jelenti, hogy a CanIf-en belül történik az egyes PDU-k és „CAN Hardware Object”-ek összekötése, így ha több kommunikációs kontroller is használatos – például egy gateway esetén – akkor a CanIf feletti moduloknak egy PDU küldése során nem kell tudniuk arról, hogy melyik CAN controllert kell használni az üzenet elküldéséhez. A CanIf-en belül történik meg a küldendő CAN üzenetek bufferelése is. Tehát a CAN interfész felkonfigurálásakor létre kell hozni a küldéshez használható CAN buffereket, és ezeket hozzá kell rendelni mind PDU-khoz, mind pedig TX irányú „CAN Hardware Objektek”-hez. A fogadott üzenetek esetében pedig az RX irányú „CAN Hardware Objectek”-et kell PDU-khoz hozzárendelni. Továbbá mind RX mind TX esetben meg kell határozni az üzenetek hosszát, és azt a modult, amellyel a CanIf interakcióba kerül az adott PDU esetében. Ez lehet a CanTp, CanNm és PduR modul. A demó projektben ez a következők szerint néz ki.

CAN Hw Obj	Pdu	Felsőbb modul	Hossz (bájt)
RxNmHwObj	PduRxNmMsg	CanNm	8
TxNmHwObj	PduTxNmMsg	CanNm	8
RxDataHwObj	PduRxData	PduR	2
TxDataHwObj	PduTxData	PduR	2

8.3.7. CAN State Manager

A CAN State Manager felelős az egyes kommunikációs kontrollerek és hozzájuk tartozó transceiverek állapotainak kezeléséért. Itt nincs sok konfigurációs lehetőség, csupán azt lehet megadni, hogy az egyes módváltásoknak mennyi idő alatt kell megtörténniük hibajelentés előtt, illetve a CAN bus-off detektálás és újrakapcsolódási kísérletek száma és periódusideje határozható még meg.

8.3.8. CAN Network Manager

A CanNm modul konfigurációjában az egyes funkciók ki/bekapcsolásán felül, az időzítési paraméterek megadása a legjelentősebb, amiket a következő táblázat foglal össze.

Paraméter	Érték	Leírás
ImmediateNmCyleTime	0.2s	A NM üzenetek periódusideje az immediate transmission alatt (csak PN Master).
ImmediateNmTransmissions	5db	Az immediate transmisson alatt kiküldendő NM üzentek száma (csak PN Master).
MsgCycleOffset	1.0s	Az NM üzenetek küldésének késletetését meghatározó paraméter, a buszterheltség csökkentő mechanizmus használatakor van jelentősége.
MsgCycleTime	1.0s	NM üzenetek periódusideje (csak PN Master).
MsgReducedTime	-	NM üzenetek periódusideje, ha a buszterheltség csökkentő mechanizmus használatban van.
MsgTimeoutTime	0.5s	NM üzenetek hiba nélküli elküldésének maximális ideje (csak PN Master).
RemoteSleepIndTime	0s	Ha ennyi ideig, nem érkezik NM üzenetek, akkor a hálózat többi node-jának alvó állapotát jelenti. Aktív koordinátor ECU-ban van jelentőssége.
RepeatMessageTime	2.0s	A csatorna állapotgépe ennyi ideig marad „Repeat Message” állapotban
TimeoutTime	5.0s	Fogadott NM üzenetek határideje, ennyi ideig marad „Ready Sleep” állapotba a csatorna állapotgép, mielőtt „Prepare Sleep” állapotba lépne.
WaitBusSleepTime	5.0s	Ennyi ideig marad „Prepare Sleep” állapotban az állapotgép, mielőtt „Bus Sleep” állapotba váltana
PnResetTime	2.5s	Egy-egy PN bit törlésének határideje, miden új fogadott vagy küldött NM üzenet, amelyben ez a PN bit aktív, újraindítja az adott PN bithez tartozó időzítőt.

A fent megadott időzítési paramétereken felül, még meg kell határozni azt a PN maszkot, amely segítségével a modul szelektálni tudja a fogadott hálózatmenedzsment üzeneteket. Ez megegyezik a transceiver konfigurációjában megadott Partial Network Mask-kal.

8.3.9. Network Management Interface

A Network Management Interfész abban az esetben, ha az ECU nem tölt be „gateway" szerepet, nem rendelkezik különösebb funkcionalitással, így a konfigurációja is leegyszerűsödik. Igazi szerepe akkor mutatkozik meg, ha az ECU-nak koordinált kikapcsolást kell végeznie. Mivel a cégnél gyártott kormányrásegítő ECU-ja valószínűtlen, hogy ilyen szerepbe kerüljön, így elsősorban az elvégzett tesztek nem ennek a funkcionalitásnak az ellenőrzésére irányultak, így ez a funkcionalitás egyelőre nem lett kihasználva.

8.3.10. Communication Manager

A Communication Manager modul koordinálja ez egyes kommunikációs csatornákat és nyújt összeköttetést a felhasználók, „Partial Network Cluster"-ek és csatornák között. A demó projektben ez a modul is viszonylag egyszerű konfigurációval rendelkezik mivel, mind az NM-Master mind a NM-Slave ECU esetében csak egyetlen CAN csatorna áll rendelkezésre. Mindkét esetben egy PNC-hez tartozik az ECU. Azonban van pár különbség a master és a slave konfiguráció között. Míg a master esetén a csatorna a PNC-n kívül még egy felhasználóhoz is hozzákötött és a ComM teljes Network Management-et valósít meg, addig a slave alkalmazás esetén, mivel nem kérhetik a teljes kommunikációs módot a platformtól, így a ComM a csatornán passzív hálózatmenedzsment koordinálást hajt végre, és a csatorna nincs is a felhasználóval összerendelve.

8.3.11. PDU Router

A PDU Router akkor kerül igazán szerepbe, amikor egy modulból érkező PDU-kat több másik modulba kell továbbítani, különben csak a felsőbb és alsóbb modulok interfészeit köti össze. A demó projektben a Com-tól érkező PDU-kat vagy a CanIf-nek vagy pedig a CanNm-nek továbbítani kell, így a PDU Router használata is szükségessé válik. Ebben a modulban először is fel kell venni az összekötendő modulokat, majd pedig ki kell tölteni a „routing” táblát, amely meghatározza melyik PDU melyik modulnak és milyen PDU ID-val kell továbbítani.

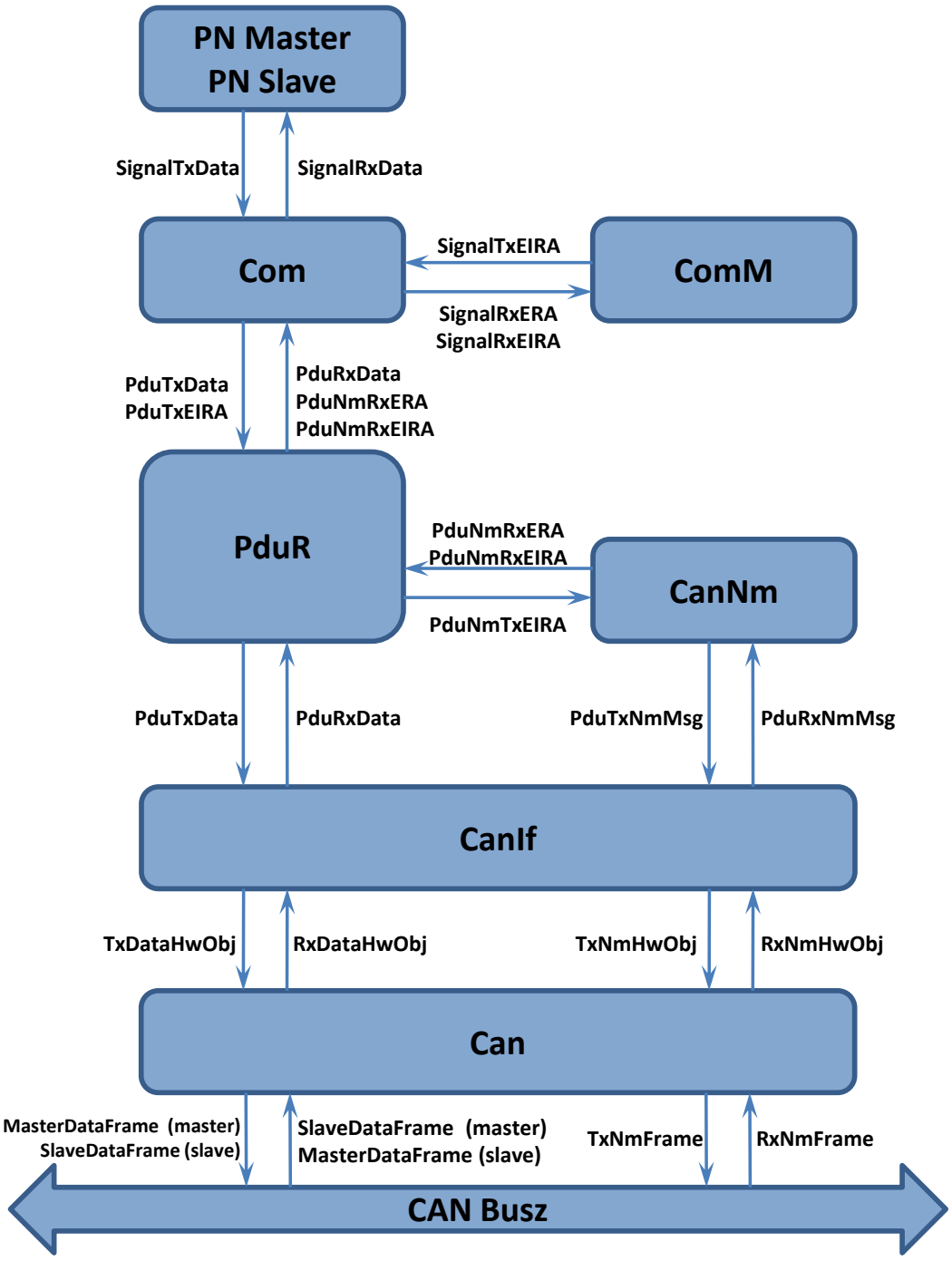
PDU Neve	Forrás modul	Cél modul	Forrás PDU ID	Cél PDU ID
PduNmTxEIRA	Com	CanNm	1	1
PduTxData	Com	CanIf	0	0
PduNmRxEIRA	CanNm	Com	2	2
PduNmRxERA	CanNm	Com	1	1
PduRxData	CanIf	Com	0	0

Mint látható, ebben az alkalmazásban – mivel a CanNm támogatja a nem szorosan indexelt PDU-ID-k használatát – megoldható volt, hogy minden PDU ugyanazzal az ID-val szerepeljen minden érintett modulban, ezzel támogatva az esetleges hibadetektálást.

8.3.12. Communication

A Communication modulban történik a szignálok PDU-khoz rendelése. A demó projektben a részhálózatok kezeléséhez szükséges információt a ComM modul szignálokon keresztül publikálja és várja, így vannak olyan Com szignálok, amelyekhez nem tartozik „I-Signal” és „System Signal” sem, csak BSW-n belül használt szignálok. Míg más Com szignálok a kommunikáció ellenőrzését végző szoftverkomponensek használnak, így ezeknek létezik párjuk (System signal) az alkalmazás rétegben is. Tehát ennek a modulnak a konfigurációja során a feladat a szignálok felkonfigurálása és PDU-khoz való hozzárendelése. Ezen felül a TX I-PDU-k esetében meg kell határozni, hogy a Com periodikusan küldje-e a PDU-kat, vagy csak akkor, amikor tartalma frissül, vagy esetleg majd az alsóbb modulok kérik el azok tartalmát (TriggerTransmit). Ebben a felhasználásban minden TX PDU csak akkor kerül küldésre, amikor azokba új adat kerül.

System-Signal	I-Signal	ComSignal	Pdu
SS_RxData	IS_RxData	SignalRxData	PduRxData
SS_TxData	IS_TxData	SignalTxData	PduTxData
-	-	SignalRxERA	PduRxERA
-	-	SignalRxEIRA	PduRxEIRA
-	-	SignalTxEIRA	PduTxEIRA



8.1. ábra, Adatcsomagok elnevezései a demó projektben

8.3.13. ECU State Manager

Az ECU State Manager – mint a neve is mutatja – felelős az ECU állapotának kezeléséért. Ebben a modulban meg kell határozni a használt BSW modulok konfigurációjának sorrendjét. Továbbá meg kell adni az MCU Driverben definiált futtatási módok közül, melyik módot kell használni a normál futás során, és melyiket Sleep módban. Valamint definiálni kell egy úgynevezett „Shutdown Target”-et is, amely azt az állapotot takarja, amelybe az ECU vált, ha sem felhasználói igény, sem pedig hálózati kérés nincs az ECU használatára. Ez a jelen projektben a „Sleep” állapot elérését, de lehetne még „Reset” és „Off” állapotok elérését is választani. Az előzőeken felül még meg kell adni az egyes „Wake Up Source” – kat, amelyek az ECU ébresztését okozhatják.

8.3.14. Operating System

Az operációs rendszer (OS) konfigurációjának egyik legfontosabb része a taszkok és az ütemezési tábla meghatározása. Az OS támogatná több partíció használatát is, amellyel a biztonságkritikus és a nem biztonság kritikus kódrészletek elkülönítése és a „memory protection unit” használatával a védelme megoldható. A demó projektben – mivel ez csak egy tesztprojekt és nem kerül valós alkalmazásba – elegendő egy partíció használata. Ugyancsak az OS-ben kell meghatározni azokat a közösen használt erőforrásokat, amelyeknek a védelmét az OS biztosítja. Az OS-ben a következő taszkok lettek létrehozva:

Taszk	Leírás
TaskBackground	Az EcuM ütemezett feladatai ebben a taszkban hajtódnak végre.
TaskComStack	Ebben a taszkban történik a kommunikációs blokk moduljaiban meghatározott ütemezett feladatok végrehajtása.
TaskStartPitTicker	Ezt a taszkot az OS automatikus indítja inicializálás után, a taszk pedig elindítja a mikrovezérlőnek azt az időzítőjét, amely az operációs rendszer ütemezési tábláját időzíti.
TaskEcuM	Ez a taszk egyszer fut le az ECU inicializálása során, ebben található az EcuM inicializáló kódrészletének azon része ami, az operációs rendszer elindulása után fut le.
Task250us	Ebben a taszkban történik az alkalmazások futtatása
TaskStartStop	Ebben a taszkban történik meg az RTE ütemezett feladatainak indítása, illetve leállítása kikapcsoláskor.

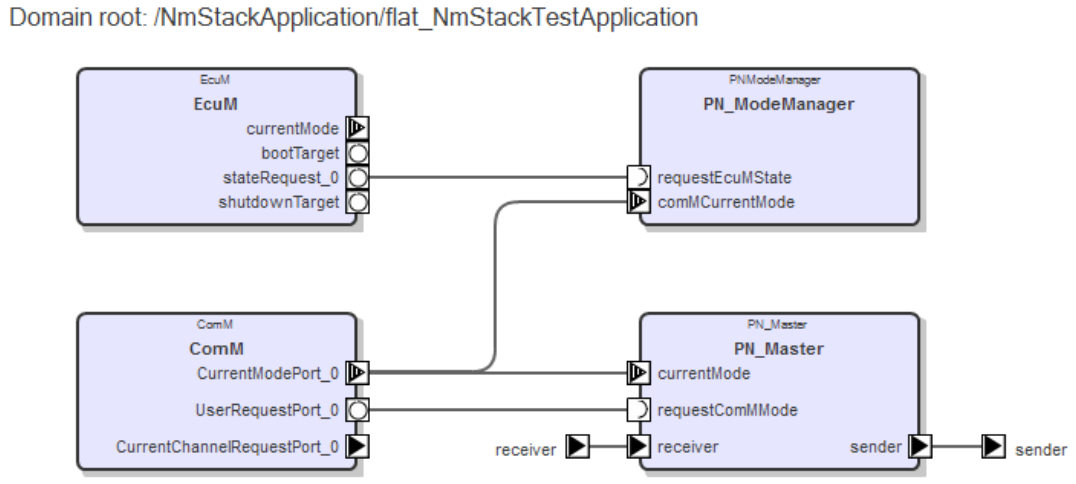
8.3.15. ECU Configuration

Az ECU Configuration csak a konfiguráció során megjelenő modell, ami az ECU általános – nem az egyes modulokhoz tartozó – konfigurációit tartalmazza. Itt kell meghatározni többek között, hogy milyen partíciók találhatóak a rendszerben és itt kell a partíciókhoz hozzákötni az egyes szoftverkomponenseket. Illetve ugyancsak ennek a modellnek a konfigurációjakor kell meghatározni a BSW-k közötti kommunikációra szolgáló PDU-kat is.

8.3.16. Runtime Enviroment

A Runtime Enviroment-ben történik meg a BSW legfelsőbb rétegének és az alkalmazás réteg komponenseinek összekötése. A platform legfelsőbb rétegbeli moduljainak szabvány szerint rendelkezniük kell modellezett portokkal, hogy interakcióba léphessenek az alkalmazás rétegben elhelyezkedő szoftverkomponensekkel, így néhány BSW modul úgynevezett „Service Component” is egyben. A demó projektben ezt a szerepet a ComM és EcuM modulok töltik be.

Az RTE konfiguráció generátora ezeknek az összeköttetéseknek a meghatározására a rendszerleíró modellt (System) használja fel, amelynek két fő eleme a „Software Composition és a System Mapping”. Az előbbi megfelel az ECU-t reprezentáló kompozíciónak, így definiálja az ECU portjait, illetve meghatározza a tartalmazott komponenseket és azok összeköttetéseit. Míg az utóbbi a „System Signal”-okat rendeli a portok interfészeihez.



8.2. ábra, Szoftverkomponensek összeköttetése

A modellből kinyert információkon felül, még az RTE-ben történik az egyes SWC-k futtatható entitásainak (Runnable Entity) taszkokhoz rendelése, vagyis az entitásokhoz tartozó kód a hozzákötött taszk futásakor fog végrehajtódni.

Összefoglalva az RTE felkonfigurálása után a konfiguráció generálásakor létrejön a SWC-k váza – az RTE az ezekben lévő függvényeket fogja meghívni –, amelyeket még ki kell tölteni a megfelelő funkcionalitással. Ezen felül létrejönnek még azoknak a taszkoknak a megvalósítása is, amelyekben történik az SWC-k funkcióinak kezelése is.

8.4. Szoftverkomponensek

A szoftverkomponensek létrehozása a korábban már említett modellező eszközben (AUTOSAR Architect) kezdődik, ugyanis minden szoftverkomponensnek létezik egy modellezett része, amelyen a portjai és az azokhoz tartozó interfészek vannak definiálva Ezen felül a modellben meg lehet adni eseményeket és ezekhez tartozó interakciókat, amiket az esemény bekövetkeztekor kell végrehajtania az alkalmazásnak. A felépített modell alapján az RTE kódgenerátorával már elkészíthető az alkalmazások váza, amelyet felhasználva implementálható az alkalmazásnak megfelelő funkcionalitás. A demó projektben három alkalmazás került megvalósításra.

8.4.1. PN Master

A PN Master alkalmazás (8.2. ábra) a hálózatmenedzsment szempontból aktív ECU bemutatásakor kerül felhasználásra. Feladata, hogy ha a ComM állatgépe „Full Communication” módba kerül, – erről jelzést kap az alkalmazás – akkor egy „User Request”-el maga is teljes üzemmódba tartsa a csatornát. Erre azért van szükség, mert a ComM-ben az állapotgép vagy egy felhasználói kérés vagy az EcuM által küldött „wakeup” jelzés hatására kerülhet „Full Communication” módba, azonban a második esetben egy idő után kikapcsol, ha nincs felhasználói igény. Ezt az igényt állítja be ezzel a kéréssel az alkalmazás. Ugyancsak ennek az alkalmazásnak a periodikus taszkjából történik a nyomógombok figyelése és azoknak megfelelő kommunikációs kérés végrehajtása. Ezzel tehát a gombokkal lehet szimulálni az ECU belső kommunikációs igényét. A másik funkcionalitás, amit ez az alkalmazás elvégez, hogy aktív kommunikáció esetén folyamatosan küldi a „Master üzeneteket” a buszra. A „Master üzenet” 6 bájtból áll, első bájtja 55h, utána két bájton egy belső számláló értéke és ezt követi a legutoljára fogadott „Slave üzenet” első három bájtja. A következő két táblázat a PN Master alkalmazás portjait és azon megvalósított interfészeket, valamint „akció listája” látható.

Port	Interfész	Leírás
currentMode (Required)	ComM_CurrentMode (Client - Server)	Ezen keresztül értesíti a ComM a kommunikációs csatorna állapotának változásáról.
requestMode (Required)	ComM_UserRequest (Client - Server)	Ezen keresztül kérheti az alkalmazás a kommunikációs csatorna be-, illetve kikapcsolását.
receiver (Required)	ReceiverInterface (Sender - Receiver)	A PN-Master saját interfésze, amelyen keresztül fogadja a slave egység üzeneteit.
sender (Provided)	SenderInterface (Sender - Receiver)	A PN-Master saját interfésze, amelyen keresztül küldi a master egység üzeneteit.

Események	Feladat
„Full” kommunikációs módba váltás	- A csatorna aktív állapotának biztosítás („FULL_COM” kérése) - Teljes kommunikációs mód jelzése (LED) - „Master üzenetek” küldésének engedélyezése
„Silent” kommunikációs módba váltás	- „Master üzentek” küldésének tiltása
„No” kommunikációs módba váltás	- Kikapcsolt kommunikációs mód jelzése (LED)
Periodikus ütemezés (250us)	- Aktív állapot jelzése (LED) - „Master üzenetek” periodikus küldése - Kommunikációs mód kérések kezelése (gombok)

8.4.2. PN Mode Manager

A PN Mode Manager alkalmazás (8.2. ábra) felelős azért, hogy az EcuM ne váltson addig sleep módba, amíg nem záródik le a kommunikációs blokk teljes kikapcsolási folyamata. Ez azért történhetne meg, mert az EcuM a kommunikációs csatornák állapotáról a ComM egyes csatorna állapotának periodikus lekérdezésével szerez információt. Azonban a CanSM-ben a kikapcsolási szekvencia csak akkor indul meg, amikor a ComM-ben a csatorna állapota „No Communication” módba lép. Ez azt jelenti, hogy az EcuM előbb érzékeli azt, hogy „sleep” állapotba válthat, mintsem, hogy a CanSM-ben megtörténne a kikapcsolási szekvencia. Ez az inkonzisztens állapot, egy olyan nem kívánt szituációhoz vezet, hogy a CanSM állapotgépében a „CANSM_BSM_S_PRE_NOCOM” állapotban a modul egy tranziens ”wake up”-ot idéz elő a transceivernél, amelyet az EcuM detektál és visszavált „normál” módba. Ezt megelőzendő az EcuM a „sleep” állapotba váltást csak akkor kezdheti meg, amikor a

CanSM-ben is befejeződött a kikapcsolás. Erről a CanSM a kikapcsolási szekvencia végén jelzést ad a ComM-nek erről, amely ezt a jelzést továbbítva jelez a szoftverkomponenseknek a kommunikációs csatorna állapotáról. A PN Mode Manager erre a jelzésre figyel, és ennek hatására engedi el az EcuM-ban a futás kérést. Ugyancsak a ComM-től érkező aktív csatorna állapot jelentése után kéri az EcuM-tól a RUN módot (amelybe amúgy is vált az EcuM az aktív ComM csatorna miatt). A következő két táblázat a PN Mode Manager alkalmazás portjait és azon megvalósított interfészeket, valamint „akció listáját” mutatja be.

Port	Interfész	Leírás
currentMode (Required)	ComM_CurrentMode (Client - Server)	Ezen keresztül értesíti a ComM a kommunikációs csatorna állapotának változásáról.
requestState (Required)	EcuM_StateRequest (Client - Server)	Ezen keresztül kérheti és engedheti le a módmenedzser a teljes futási módot az EcuM-ban

Események	Feladat
„Full” kommunikációs módba váltás	- Az ECU aktív állapotának biztosítás („RUN” állapot kérése)
„No” kommunikációs módba váltás	- Az ECU kikapcsolásának engedélyezése („RUN” állapot elengedése)

8.4.3. PN Slave

A PN Slave alkalmazás az hálózatmenedzsment szempontból passzív ECU bemutatásakor kerül felhasználásra. Mivel a passzív ECU-ban a ComM csatorna állapotgépéhez nem tartozik felhasználó, ezért ez az alkalmazás nincs is összeköttetésben a ComM modullal. A PN Slave alkalmazás a végletekig leegyszerűsített. Egyrészt tartalmaz egy periodikus futó kódrészletet, amely csak a futás visszaigazolását végzi. Míg egy másik esemény, amire figyel, a „Master üzenetek” fogadása. Ha egy új ilyen üzenet érkezik, akkor azt feldolgozza és visszaküld egy „Slave üzenetet”. A „Slave üzenet” szintén 6 bájtból áll, első bájt mind 0x55, ezt követi egy 2 bájtos üzenetszámláló, amely minden elküldött üzenet után növekszik eggyel, ezt követi a „Master üzenet” első bájtja és utána két bájton a „Master üzenetben” található számláló negáltja. A következő két táblázat a PN Slave alkalmazás portjait és azon megvalósított interfészeket, valamint „akció listáját” mutatja be.

Port	Interfész	Leírás
receiver (Required)	ReceiverInterface (Sender - Receiver)	A PN-Slave saját interfésze, amelyen keresztül fogadja a master egység üzeneteit.
sender (Provided)	SenderInterface (Sender - Receiver)	A PN-Slave saját interfésze, amelyen keresztül a válaszüzentet küldi a master egységnek

Események	Feladat
Fogadott adat jelzés	- A fogadott „Master üzenet ” feldolgozása és „Slave üzenet” elküldése
Periodikus ütemezés (250us)	- Aktív állapot jelzése (LED)

8.5. Integrációs kód

Habár az AUTOSAR szoftverarchitektúra szabványosított és definiálja a modulok egymással való kapcsolatát, azonban elkerülhetetlen az egyedi megoldások miatt a minimális projekt specifikus funkcionalitás hozzáadása. Ezek a kódrészletek szabványosan illeszkednek az architektúrához. A demó projektben az integrációs kódot két csoportba lehet sorolni, egyrészt a taszkok megvalósításai, másrészt az EcuM-hez kapcsolódó integrációs kódrészletek. Az EcuM szabványában a főbb állapotváltásoknál lehetőség van a modul működésének kiterjesztésére a következő függvények kitöltésével.

- EcuM_EnableWakeupSource(): Mielőtt az ECU alvó állapotba kerülne, meghívja ezt a függvényt. Itt történik meg a transceiver „wakeup” módjának bekapcsolása és a transceiver alvó állapotba küldése. Megjegyzés: Ez csak a demó projektben kerül ide, mivel itt csak altatva van az ECU, ha az ECU-t valóban ki kell kapcsolni, akkor az ide került kód kerülhet az EcuM_AL_SwitchOff()-ba vagy az OS ShutdownHook-jába, mivel a transceiver alvó állapotba váltása, a tápfeszültség kikapcsolását jelentheti.
- EcuM_DisableWakeupSource(): Itt kell letiltani ébredés után a transceivert, hogy nem jelezzen több „wakeup” eseményt. Ez igazából csak megszakítással jelzett „wakeup” esetén fontos.
- EcuM_CheckWakeup(): Ebben a függvényben történik a „wakeup” események lekérdezése. A demó projektben ez a transceiver regisztereinek olvasásával -

természetesen a driveren keresztül - történik. Ezt a függvényt periodikusan futtatja az EcuM, vagyis akkor használandó, amikor az ECU nem kikapcsol csak energiatakarékos módba vált.

- EcuM_SleepActivity(): Ez a függvény szintén periodikusan fut, és az „alvó” állapotban végzendő tevékenység forrása kerül bele. A demó projektben az alvó állapotot jelző led villogtatás került ide.
- EcuM_CheckValidation(): Ide szükség esetén a „wakeup” esemény érvényesítése kerülhet, például, ha a hálózatról történt ébresztés, akkor érvényesítés lehet egy helyesen fogadott üzenet. Amennyiben ez az érvényesítés egy határidőn belül nem történik meg, akkor az EcuM visszavált alvó állapotba.

Az itt felsoroltakon felül még számos állapotváltásnál kiegészíthető az EcuM, azonban a hálózatmenedzsment szempontjából ezek voltak relevánsak. Amiről még nem volt szó, de szintén ide tartozik, hogy az EcuM integrációs kódjából történik az egyes BSW modulok inicializálása is.

8.6. Stub modulok

Egy ilyen komplex szoftverarchitektúránál, mint amit az AUTOSAR szabvány definiál, néha előfordul, hogy néhány esetben minimális kiegészítéssel a szabványos architektúra bizonyos részei nem a szabványos elemekkel vannak ellátva. Mivel a diplomaterv feladatban megvalósított projekt nem egy végleges felhasználásba kerülő szoftver, így néhány modul elhagyásával egyszerűsíthető.

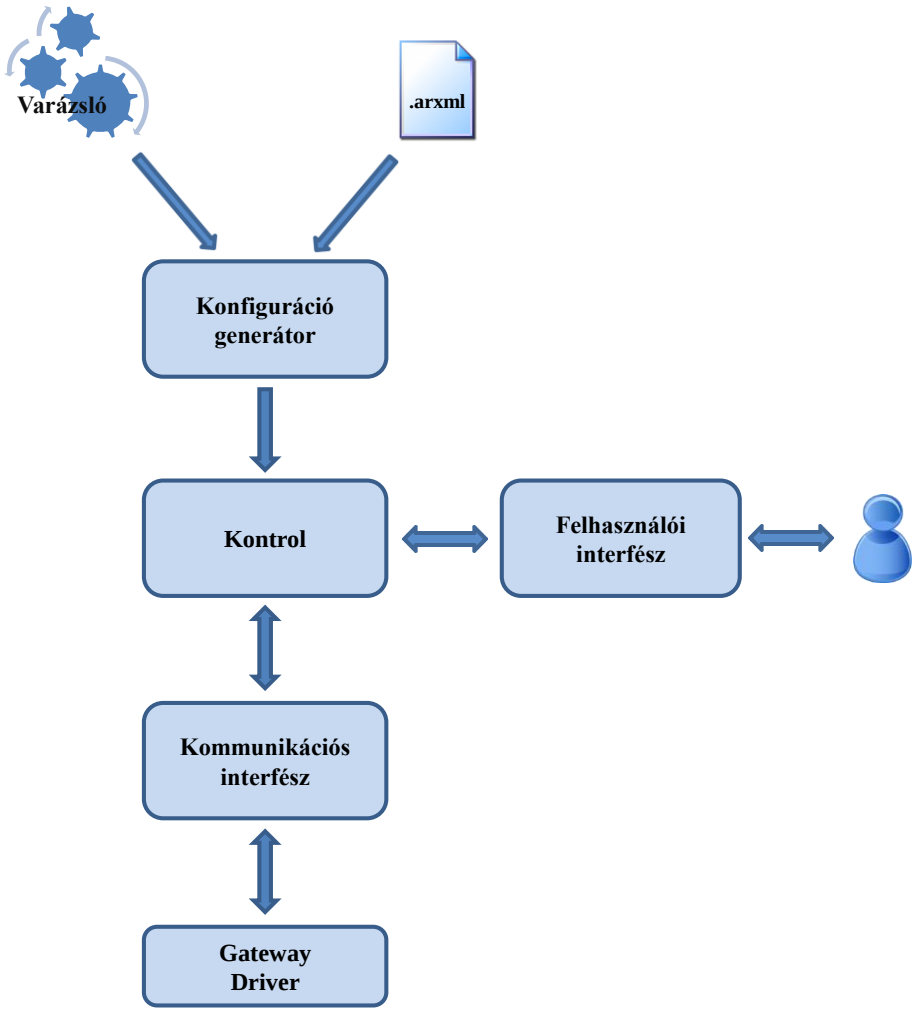
A projektben összesen két nagyobb jelentőségű stub modul lett létrehozva a BswM és az SPIDriver, amelyek a projekt létrehozásakor nem voltak megvalósítva illetve letesztelve. A BswM szerepe lenne a BSW modulok állapotainak kezelése. A demó projektben két feladatot lát el. Egyrészt a BswM-nek a ComM modul jelzi az egyes csatornák állapotát, így ez a modul végzi el a kommunikáció engedélyezését, ha a csatorna „Full Communication” módba kerül és tiltását, amikor „Silent Communication” módba lép. Ezen felül a Slave ECU-ban ez a modul látja el ugyanazt a feladatot, amit a Master ECU-ban a PN Mode Manager alkalmazás.

Az SPI Driver a végletekig le lett egyszerűsítve, a feladata a transceiver belső regisztereihez való hozzáférés biztosítása. Ennek megfelelően az SPI inicializálása a

mikrovezérlőnek csak a transceiverrel összekötött SPI modulját konfigurálja fel a transceiver kommunikációs módjának megfelelően. Illetve specializálva lett a transceiverrel való kommunikációra. Ez azért volt megtehető, mert a dolgozat készítésekor a TJA1145-ös transceiverhez sem állt rendelkezésre driver, azt is implementálni kellett, amelyen egy már meglévő transceiver driver kiegészítését jelentette.

9. PC oldali tesztkörnyezet

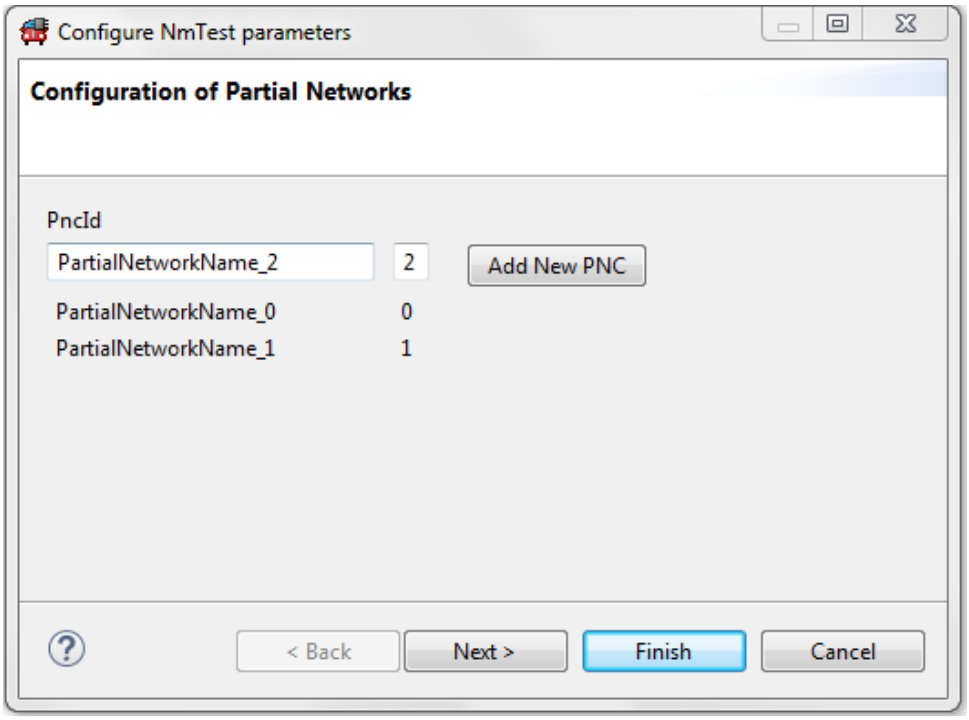
A diplomaterv utolsó nagy feladata egy PC oldali teszteszköz készítése, amellyel lehetőség nyílik a CAN buszon megjelenő hálózatmenedzsment üzenetek megjelenítésére, illetve szimulálására, ezzel tesztelve az ECU ébredését és alvását a hálózati kommunikációtól függően. A szoftver megtervezése során adott volt a PC-t a CAN busszal összekötő – a cégnél fejlesztett – „Fieldbus gateway”, illetve a korábbiakban már említett modellező eszköz. Mivel a gateway-hez a PC oldalon egy javaban íródott driver réteg is tartozik, illetve a modellező eszköz is Eclipse alapú, így az általam készített teszteszköz esetében is ez a programozási nyelv lett kiválasztva. Mivel a modellező eszköz Eclipse alapú, így lehetőség adódik az eszköz további bővítésére új plugin-ek hozzáadásával. A teszteszköz készítésekor is ezt a lehetőséget használtam ki. A tervezéskor az egyik fő szempont a modularitás volt, így alapvetően négy nagy blokkra lehet tagolni az elkészített szoftvert (9.1. ábra). A modularitással a cél az volt, hogy a szoftvert rugalmasan bele lehessen illeszteni különböző környezetbe. A kommunikációs interfész réteg használatával lehetőség adódik a különböző eszközök fölé helyezni a teszt eszközt, míg a saját konfiguráció használata pedig a konfigurálhatóság szabadságát adja.



9.1. ábra, Tesztkörnyezet felépítése

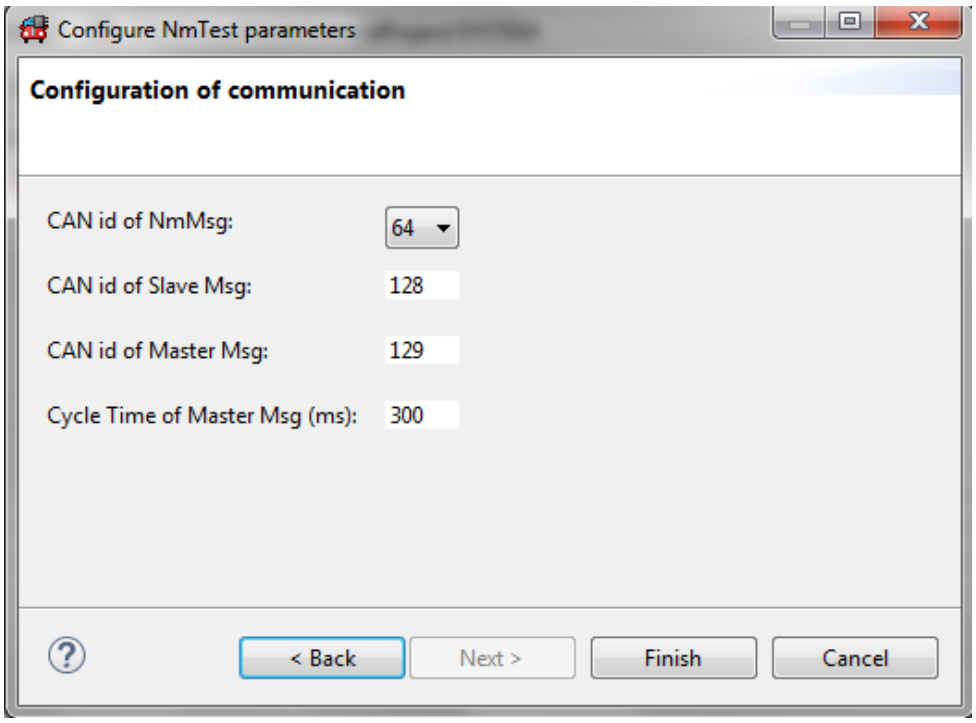
9.1. Konfiguráció generátor

A konfiguráció generátor feladata a teszteszköz által felhasznált konfiguráció feltöltése. Ez alapvetően két forrásból történik. Egyrészt a „ECU Extract” feldolgozása alapján, míg a másik forrás egy varázsló, amin keresztül közvetlenül megadhatóak bizonyos paraméterek. Erre két okból van szükség. Egyrészről a modellező eszköz nem teljes és a jelenleg még nem használt részleges hálózatkezeléshez tartozó modellelemeket nem kezeli, így nincs lehetőség ezek szerkesztésére és feldolgozására. Emiatt és a tesztelhetőség további kiterjesztése miatt is a részhálózatokat kézzel kell felvezetni. A teszt konfigurációjának generálása az „ECU Value Collection”-ön jobb klikkel a popup menüből érhető el. A modell feldolgozása után elindul a varázsló, amelynek első lapján lehet megadni a részhálózatokat (9.2. ábra).



9.2. ábra, Tesztkörnyezet varázslójának első lapja

A varázsló második lapján a hálózatmenedzsment CAN frame ID-ja választható ki a rendszer hálózatmenedzsment számára fenntartott frame-jei közül. Ezen felül itt adható meg még a Master és Slave üzenetek CAN frame-je és a Master üzenetek periódusideje is (9.3. ábra).



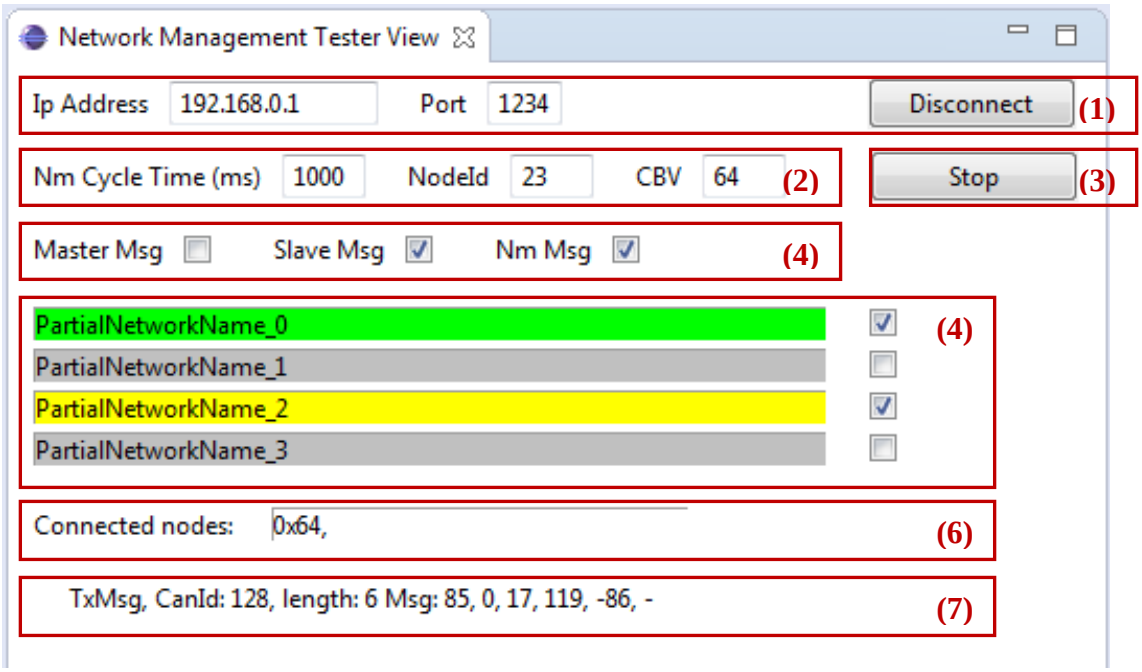
9.3. ábra, Tesztkörnyezet varázslójának második lapja

9.2. Control

A kontrol blokkban történik a teszt által küldendő és fogadott üzenetek feldolgozása. Az NM üzenetek és a Master üzenetek küldése is külön ütemezett taszkban történik, mind a kettő saját időzítővel rendelkezik és a START gombra indulnak. Ugyancsak ebben a blokkban állítja össze a hálózatmenedzsment a Master és a Slave üzeneteket. Minden üzenet csak akkor kerül valóban elküldésre, amikor az adott üzenet küldése engedélyezett.

9.3. Felhasználói interfész

Ez a teszteszköz felhasználói interfésze (9.4. ábra). A GUI-n keresztül beállítható a gateway IP címe és a használt port (1). Megadható továbbá a hálózatmenedzsment üzenet periódusideje, a NID és a CBV (2), valamint indítható és leállítható a funkcionalitás (3). Ezek alatt találhatóak az NM, Master és Slave üzenetek engedélyezése és tiltása (4). Ezek után láthatóak a felvett „partial network”-ök, illetve kapcsolhatóak be és ki ezek a részhálózatok (5). Ez alatt a kapcsolódott node-ok ID-ja a fogadott PN üzenetek NID mezőiből (6). A részhálózatok alatt pedig az utolsó néhány információs log látható (7).



9.4. ábra, Tesztkörnyezet megjelenítője

9.4. Kommunikációs interfész

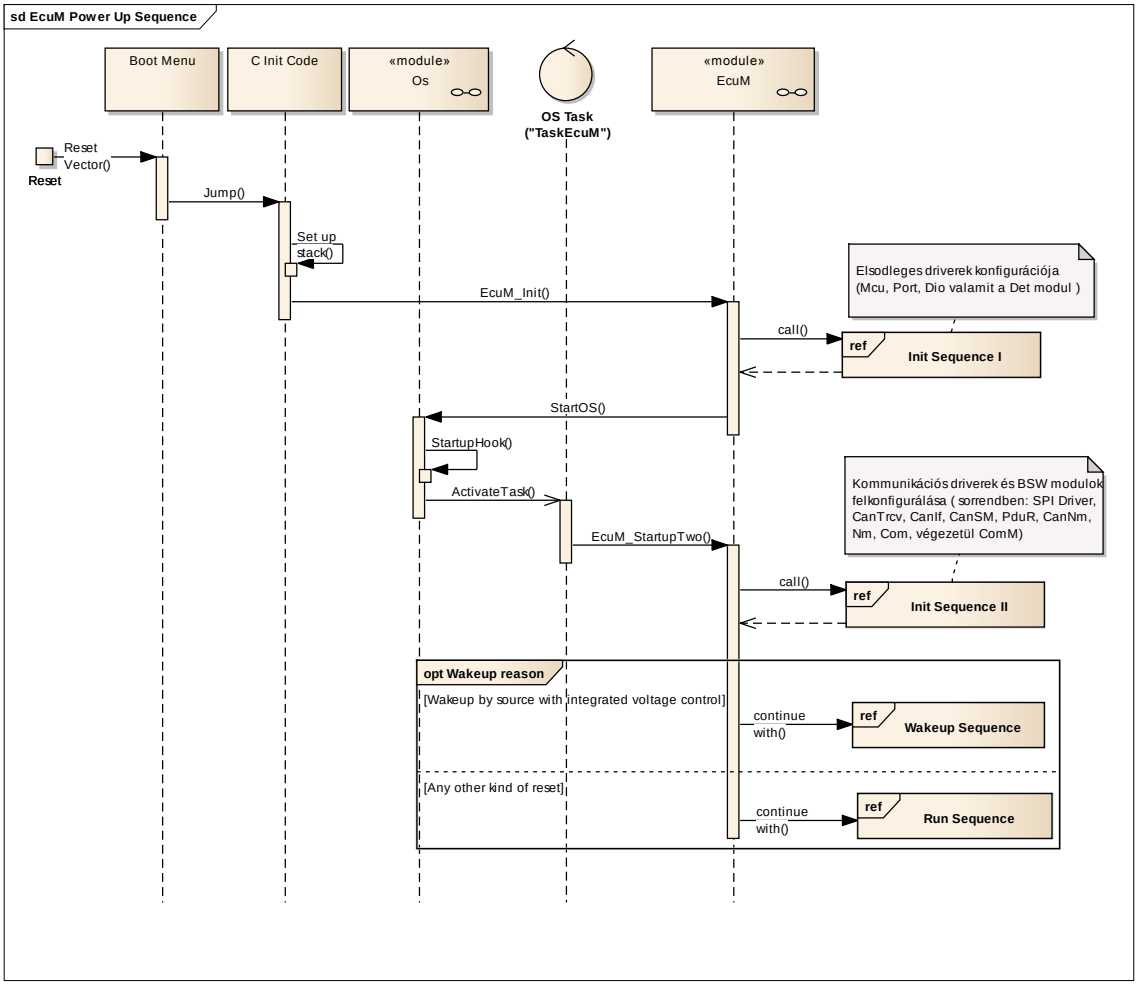
Ez a réteg azért lett létrehozva, hogy a tesztszoftver bármilyen szoftveres környezetbe rugalmasan beilleszthető legyen. Ezzel lehetősség nyílik arra, hogy a használt hardver (gateway) lecserélhető legyen más hardverre, de akár arra is esélyt ad, hogy a teszteszköz ne közvetlen a gateway driver-éhez kapcsolódjon, hanem egy komplexebb szoftver részét képezze (pl.: restbus szimulátor). A most használt gateway Etherneten keresztül kapcsolódik a számítógéphez, és saját driver rétegén keresztül használható. Az eszköz alapvetően három CAN kommunikációs csatornát tartalmaz, amelyek közül a teszteszköz fixen az egyes csatornát használja. A gateway használatához először létre kell hozni és fel kell konfigurálni RX és TX CAN Message objekteket. A későbbiek során csak ezeket lehet majd használni a kommunikációra. Ezen felül fel kell iratkozni listenerekre, amelyeken keresztül a gateway driver jelzést ad egy új üzenet fogadásáról vagy egy üzenet sikeres elküldéséről.

10. Rendszer működésének bemutatása

Ez a fejezet az AUTOSAR szabványban definiált szoftverarchitektúrának megfelelően összeállított rendszer működését mutatja be – a hálózatmenedzsment szempontjából releváns részletekre fókuszálva – olyan helyzetekben, mint az ECU első elindítása (start), az ECU alvó állapotba menetele (sleep), valamint ébresztése (wakeup).

10.1. Start

Az első szituáció, ami bemutatásra kerül az az ECU első bekapcsolása (10.1. ábra). Abban az esetben, ha az ECU-n PN-öt támogató transceiver található, a „power on reset” hatására „standby” állapotba kerül, és engedélyezi az ECU tápellátását. Ennek hatására az ECU-n található mikrovezérlő is bekapcsol, ahol a „startup” kód és esetleges boot program lefutása után, első lépésben az ECU State Manager inicializálása történik meg. Az EcuM felkonfigurálja az alapvető driver modulokat, mint például az Mcu, a Port, vagy a megszakítást vezérlő Icu driver. Ezután az EcuM modul indítja el az operációs rendszert is, amelynek indulása után a rendszer további indítása már OS támogatással történik. Tehát miután elindulnak a taszkok az OS-ben a „TaskEcuM” taszkban folytatódik az EcuM inicializálása. Itt több lépésben megtörténik a többi BSW modul felkonfigurálása kezdve a kommunikációs csatorna driverektől felfele haladva. Miután megtörtént a rendszer teljes felkonfigurálása, ezután következik a bekapcsolást kiváltó ok detektálása, mikor is az EcuM lekérdezi az összes „wakeup source”-ot, hogy melyiken történt „wakeup” esemény. Amennyiben a kommunikációs hálózati forgalomra ébredés támogatott, úgy ez együtt jár a kommunikációs kontroller vagy transceiver ébresztés jelzésének lekérdezésével. Esetünkben a TJA1145-ös transceiver kétfajta ébresztési eseményt is támogat. Az egyik a PO (power on) esemény, azaz amikor a transceiver megfelelő tápfeszültséget kap, míg a másik a releváns ébresztési esemény, amikor a hálózatról a megfelelő WUF-ot fogadja, és ennek hatására történik meg az ébresztés. Miután kiderült, milyen okból történt az ECU ébresztése, megfelelő módon kezelhető annak oka, majd elkezdődhet a normál működés.



10.1. ábra, Startup szekvencia[8]

10.2. Sleep

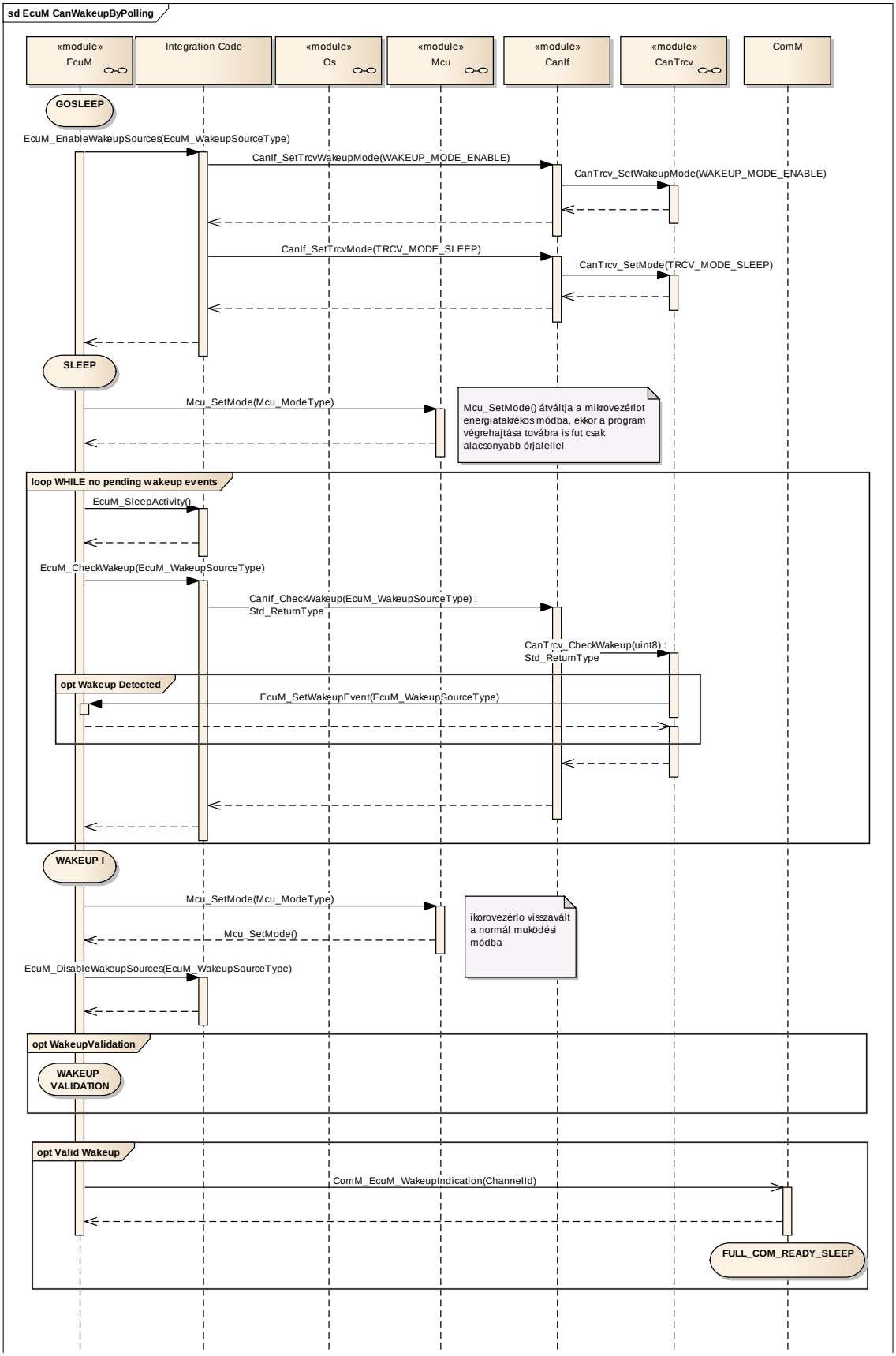
Ebben a szakaszban az előzőekhez, hasonlóan az ECU kikapcsolási folyamatairól (10.2. ábra) lesz szó. A kikapcsolást a bekapcsolás menetéhez hasonlóan az ECU State Manager modul végzi. Alapvetően a bekapcsolási szekvencia lezajlása után az EcuM-ban a cél a megfelelő „Shutdown target” elérés, amely lehet kikapcsolt, alvó állapot vagy újraindítás. De ameddig valamely szoftverkomponensnek az ECU bekapcsolt állapotára van szüksége, addig az EcuM is ébren tartja az eszközt. Ez a demó projekt esetében a PN Master ECU-nál egyrészt a ComM kommunikációs csatornák állapotától függ – tehát amíg van olyan csatorna, amelyiken kommunikáció zajlik, addig az ECU is ébren marad -, másrészt mind a PN Master, mind a PN Slave ECU-k esetében a rendszer éber állapotban marad, amíg valamelyik wakeup source aktív, jelen esetben a kommunikációs hálózat ébren tartja az ECU-t. Látható tehát, hogy az ECU mindaddig ébren marad, amíg az ECU a hálózat bármely másik szereplőjével kommunikációt

folytat. Azonban, ha a hálózati kapcsolatara már nincs szükség, akkor megkezdődhet az ECU kikapcsolása vagy alvó állapotba küldése.

Az EcuM először letiltja a kommunikációs csatornákat, majd kikapcsolás esetén elkezdődik az egyes modulok de-inicializálása, még alvó állapotba váltás előtt, az EcuM engedélyezi a wakeup source-okat, majd pedig elkezdődhet az alvó állapot. Az alvóállapot nem feltétlen jelenti azt, hogy a CPU HALT állapotba kerül, ugyanis az EcuM-ban lehetőség van kétféle „alvó állapot” elérésére. Az egyik ezek közül a szokványos alvó állapot, amikor is a CPU nem dolgozik és a mikrovezérlő egy megszakítás hatására ébred fel, míg a másik állapot egy energiatakarékos működést jelent. Ebben az esetben a CPU ugyanúgy aktív, csak esetleg alacsonyabb frekvenciáról jár, illetve bizonyos perifériák ki lehetnek kapcsolva. Ebből az állapotból a mikrovezérlő normál állapotba visszatérhet megszakítás hatására is, de lehetőség van arra is, hogy az EcuM folyamatos lekérdezéssel (pollingal) figyelje az egyes wakeup source-okat. A demonstrációs projektben ez az utóbbi változat került megvalósításra.

10.3. Wakeup

Az előző szakaszban látható volt, hogyan éri el az ECU a kikapcsolt, alvó vagy energiatakarékos módot. Ebben a részben pedig a bekapcsolás menete kerül röviden ismertetésre (10.2. ábra). Amikor egy „wakeup” esemény történik az az EcuM-ben kerül feldolgozásra (alapvetően vagy az EcuM figyelte lekérdezéssel a wakeup csatornákat, vagy a megszakításkezelésből kell meghívni az EcuM megfelelő függvényét). Ébredés után az EcuM először letiltja a „wakeup” forrásokat, ezután ha felkonfigurált, akkor ellenőrzi az ébresztés érvényességét. Ez a hálózati kommunikációra való ébredés esetén egy üzenet helyes fogadásával lehet egyenértékű. Ha nem sikerül határidőn belül érvényesíteni az ébresztést, úgy az EcuM visszaküldi alvó vagy energiatakarékos állapotba a mikrokontrollert. Sikeres ébresztés esetén pedig a ComM-et értesíti az ébredésről, amely hatására a hálózatmenedzsment blokkban megindul a passiv startup szekvencia, vagyis a kommunikációs csatorna „Full Com Ready Sleep” állapotba kerül. PN Slave ECU esetében ebben az állapotban marad, mindaddig, amíg a hálózatról érkezik ébrentartást kiváltó üzenet, míg PN Master esetében, a kommunikációs csatorna bekapcsolásának hatására, az ECU maga is kérheti a kommunikációs hálózat életben tartását, amit a hálózatmenedzsment üzenetek küldésével biztosít.



10.2. ábra, Sleep és Wakeup szekvencia[8]

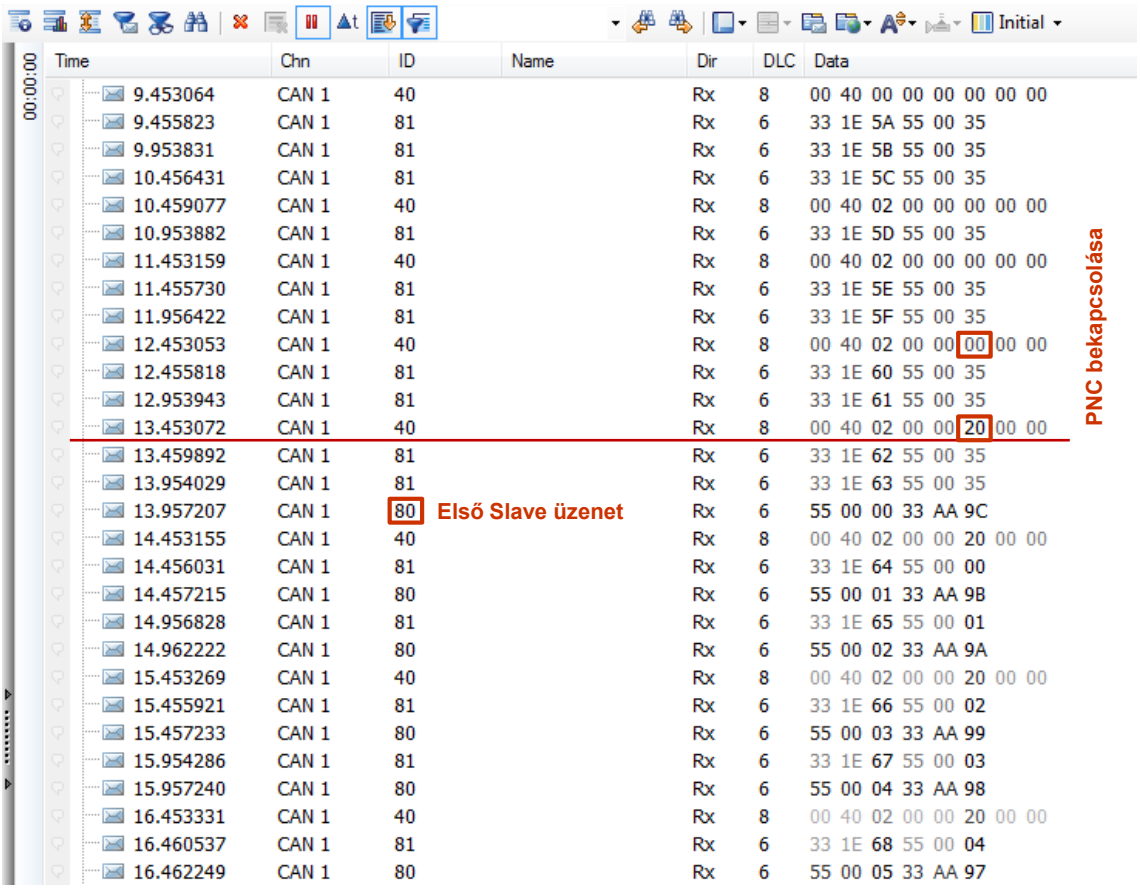
11. Demonstrációs projekt tesztelése

Az eddigi fejezetekben már esett szó az AUTOSAR hálózatmenedzsment elméleti működéséről és a demonstrációs projektben összeállított szoftver felépítéséről. Ebben a fejezetben pedig az elkészített feladat tesztelésének bemutatására kerül sor. Első lépésben a PN Slave ECU működése kerül bemutatásra, majd ezt követően a PN Master ECU megfelelő működéseinek ismertetésére kerül sor.

A helyes működés megfigyelésére egyrészről a fejlesztői panelon megtalálható ledeken keresztül, illetve a hálózati kommunikáció elemzésével van lehetőség. A hálózati kommunikáció megfigyelésére a Vector CANoe nevű szoftverével és egy Vector hálózati interfészen (VN7600) keresztül történt. A hálózatmenedzsment master eszközt a PC-s tesztkörnyezet szimulálta a tesztelés során, amely a már korábban említett Fiedbus-Gateway-en keresztül kapcsolódott a kommunikációs buszra.

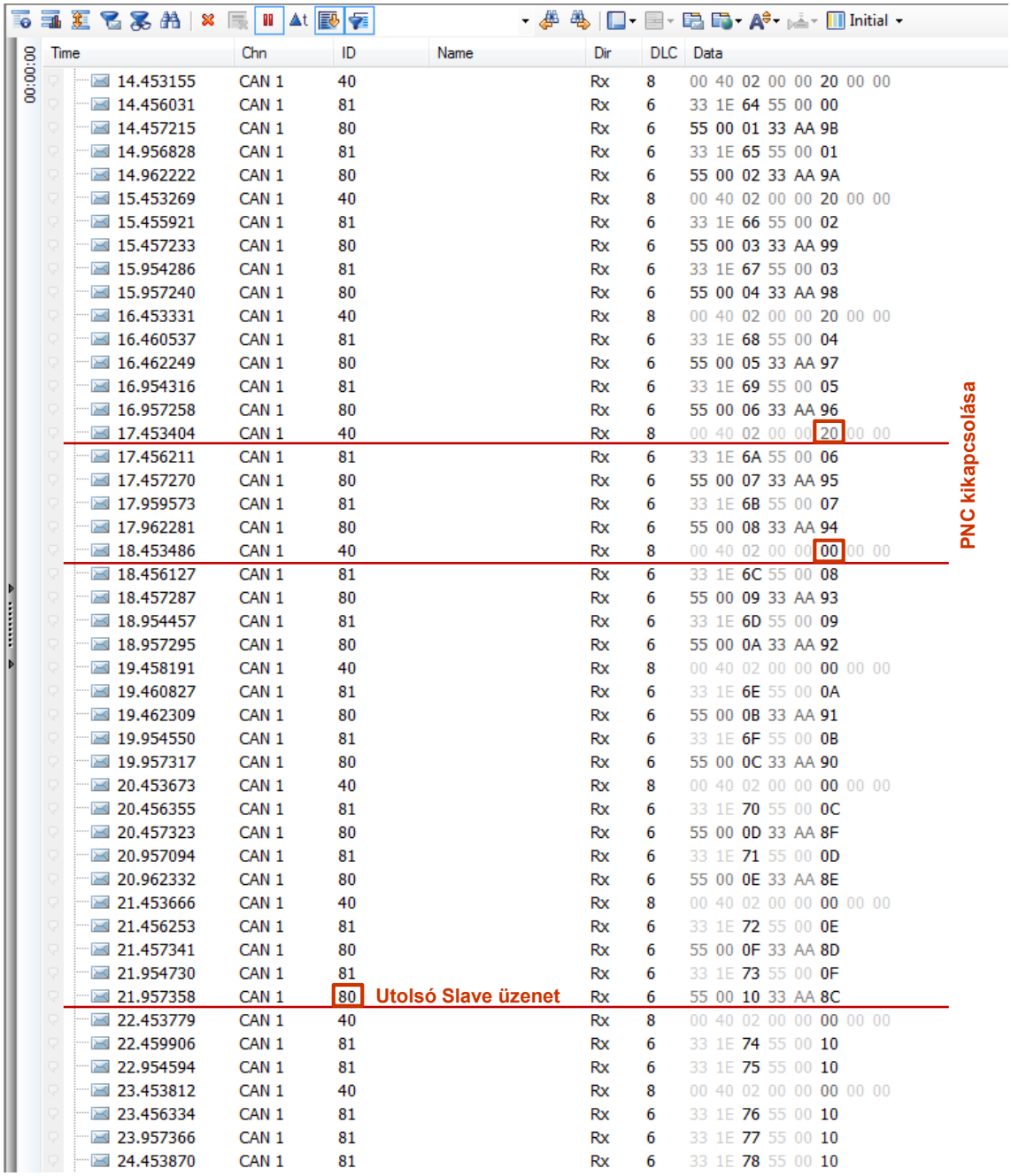
11.1. PN Slave

A PN Slave ECU tesztelése során (11.1. ábra) alapvetően azt kell megfigyelni, hogy az ECU bekapcsolás után – az inicializálások lezajlása után - alvó állapotba kerül és ott is marad, amíg a hálózaton nem érkezik olyan hálózat menedzsment üzenet, amelyben az ECU-t tartalmazó részhálózat bitje be van állítva. A PN Slave ECU a 0-s és a 29-es „partial network”-ökhöz lett hozzárendelve, ami azt jelenti, hogy a hálózatmenedzsment üzenet „User data” mezőjének – 2-7 bájt, mivel van az üzenetben CBV és NID bájt – 0. és 29. bitjének 1-be állításával kiváltható az ECU bekapcsolása. Tehát az elvárt viselkedés az, hogy amíg ezek a bitek nincsenek beállítva, az ECU ne válaszoljon a Master egység kéréseire, amelyek a 81h azonosítójú CAN üzenetben található. A PN-Slave ECU egészen addig nem válaszol a 80h-ás CAN üzenetben, amíg a hálózat menedzsment üzenetben csak a 1-es „partial network”-höz tartozó bit van beállítva, majd a 13s 453ms-ben küldött NM-Message-ben már beállításra kerül a 29-es PN-bit és a Slave ECU is bekapcsol és elkezd válaszolni a master üzeneteire.



11.1. ábra, NM Slave ECU bekapcsolása

A kikapcsolás meneténél (11.2. ábra) azt lehet megfigyelni, hogy amikor a hálózatmenedzsment üzenetben a PN Slave ECU részhálózata kikapcsolásra kerül – azaz a 29-es PN bit 0-ára vált – onnantól kezdve ez ECU még válaszol a master kéréseire, mindaddig, míg a ComM-ben az állapotgép „Silent” módba nem vált és ekkor a BswM letiltja a Com-ban az összes I-PDU küldését. Ez akkor történik meg, amikor a CanNm-ben található állapotgép „Ready Sleep” állapotból „Prepare Bus Sleep” állapotba nem vált, aminek feltétele a CanNmTimeoutTime elérése. Ez ebben a projekten 5ms, ami azt jelenti, hogy ha ennyi ideig nem érkezik érvényes hálózatmenedzsment üzenet – amiben van aktív releváns PN bit – akkor a hálózat kikapcsoltnak tekintett. Az ábrán is látható 17s 453ms-kor van az utolsó olyan NM üzenet a buszon, ami még ébren tartaná az ECU-t majd, a 22s 453ms-es NM üzenet után már nem is válaszol a master kéréseire. Azt azért még mindenképp meg kell jegyezni, hogy a valós felhasználásban nem lehetne a kikapcsolandó PN Slave ECU számára releváns üzenet a buszon, de itt ezeknek az üzenetek hatására lehet megfigyelni a busz forgalma alapján az ECU működését.

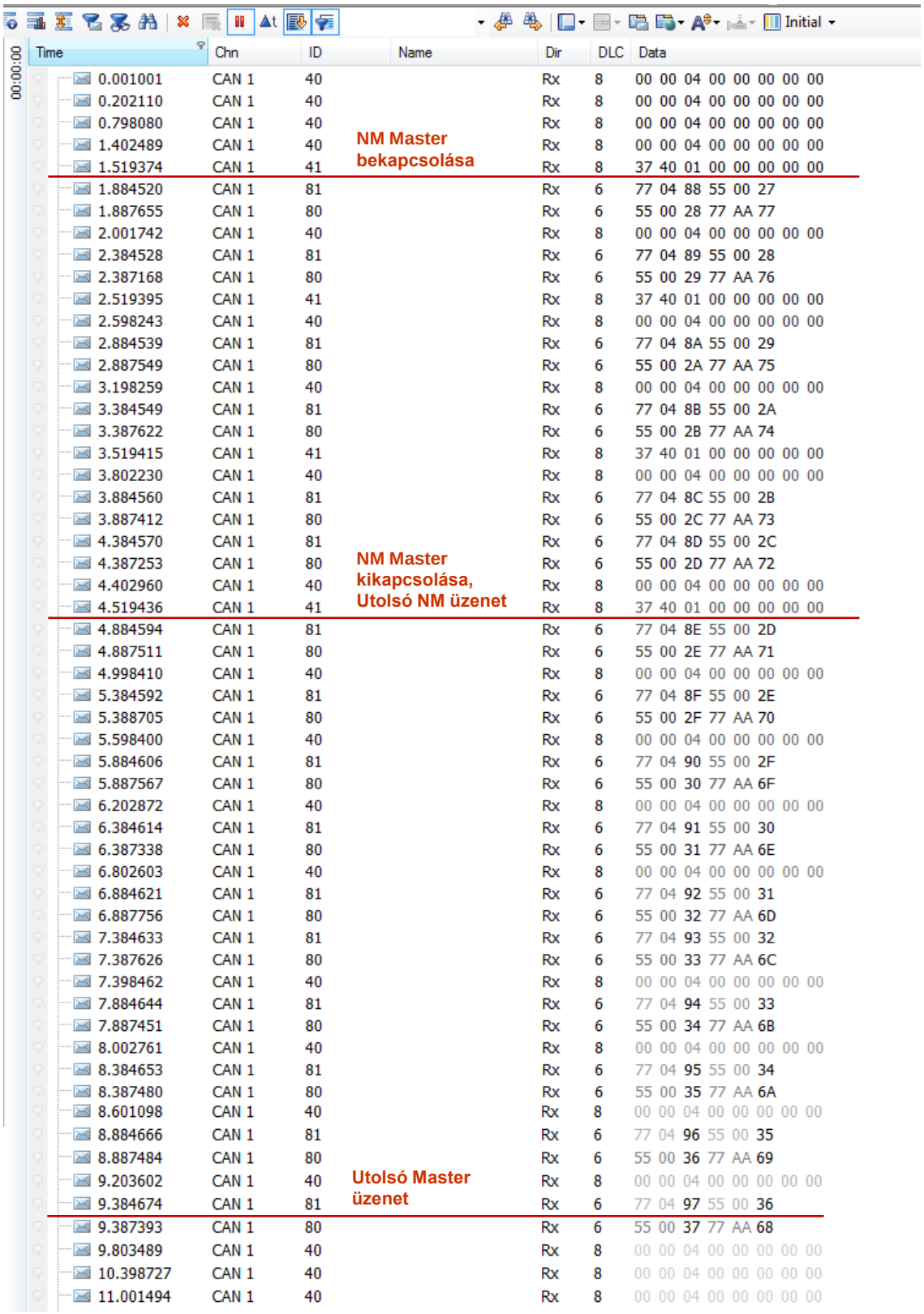


11.2. ábra, NM Slave ECU kikapcsolása

11.2. PN Master

A master alkalmazásba is a Slave-hez hasonlóan a hálózatmenedzsment üzenetek periódus ideje 1s, kivétel az „Immediate Restart” amikor 200ms. A PN Master ECU tesztelése során is a PN Slave-hez hasonlóan az ECU bekapcsolás után alvó állapotba kerül, amit a Sleep Activity-ből villogtatott LED jelez, illetve, hogy nem küld a buszra semmilyen üzenetet. Alvó állapotból az ECU-t vagy a megfelelő PN-bit beállításával a NM-Message-ben lehet bekapcsolni, vagy a panelon lévő SW1-es gomb megnyomásával. Amikor a gomb segítségével ébresztjük az ECU-t, látható, hogy a PN-Master ECU 1ms-ként küldi a NM üzentet a 41h-ás CAN frame-ben. Az ábrán az is látható, hogy bekapcsolás után elkezdi a Master üzenetek küldését (11.3. ábra).

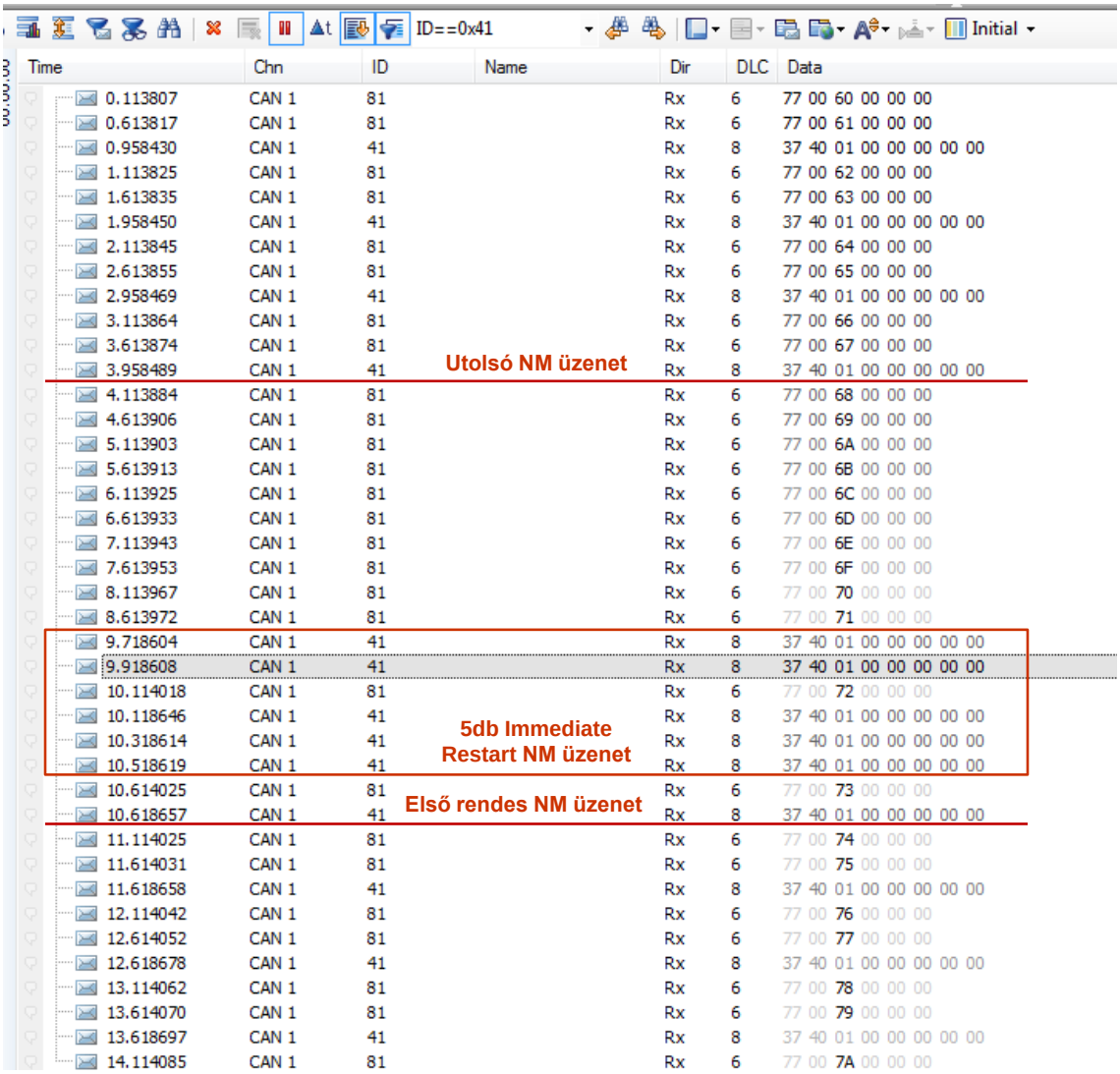
A második tesztelendő szituáció, a slave alkalmazáshoz hasonlóan, amikor a PN Master ECU kikapcsol. A kikapcsolást az SW2-es gomb megnyomásával lehet elindítani. A gomb megnyomásának hatására a PN master alkalmazás, „No Communication” állapotot kér a ComM-től, aminek hatására az egy „CanNm_NetworkRelease()” függvényhíváson keresztül (természetesen az NM modulon át) azonnal „Ready Sleep” állapotba állítja a modul állapotgépét, aminek hatására befejeződik a hálózatmenedzsment üzenetek küldése. Amíg a ComM állapotgépe „Full Com Ready Sleep” állapotban van, még küldhet üzeneteket, de amint belép „Silent” módba a BswM leállítja az üzenetek küldését, illetve a PN Master alkalmazás is leállítja az üzenetek küldését (11.3. ábra). Itt is látható, hogy az utolsó NM-Message 4s 519ms-kor kerül elküldésre és az utolsó Master üzenet pedig 9s 384ms-kor és ebben az esetben is a CanNmTimeoutTime 5ms.



Time	Chn	ID	Name	Dir	DLC	Data
0.001001	CAN 1	40		Rx	8	00 00 04 00 00 00 00 00
0.202110	CAN 1	40		Rx	8	00 00 04 00 00 00 00 00
0.798080	CAN 1	40		Rx	8	00 00 04 00 00 00 00 00
1.402489	CAN 1	40	NM Master bekapcsolása	Rx	8	00 00 04 00 00 00 00 00
1.519374	CAN 1	41		Rx	8	37 40 01 00 00 00 00 00
1.884520	CAN 1	81		Rx	6	77 04 88 55 00 27
1.887655	CAN 1	80		Rx	6	55 00 28 77 AA 77
2.001742	CAN 1	40		Rx	8	00 00 04 00 00 00 00 00
2.384528	CAN 1	81		Rx	6	77 04 89 55 00 28
2.387168	CAN 1	80		Rx	6	55 00 29 77 AA 76
2.519395	CAN 1	41		Rx	8	37 40 01 00 00 00 00 00
2.598243	CAN 1	40		Rx	8	00 00 04 00 00 00 00 00
2.884539	CAN 1	81		Rx	6	77 04 8A 55 00 29
2.887549	CAN 1	80		Rx	6	55 00 2A 77 AA 75
3.198259	CAN 1	40		Rx	8	00 00 04 00 00 00 00 00
3.384549	CAN 1	81		Rx	6	77 04 8B 55 00 2A
3.387622	CAN 1	80		Rx	6	55 00 2B 77 AA 74
3.519415	CAN 1	41		Rx	8	37 40 01 00 00 00 00 00
3.802230	CAN 1	40		Rx	8	00 00 04 00 00 00 00 00
3.884560	CAN 1	81		Rx	6	77 04 8C 55 00 2B
3.887412	CAN 1	80		Rx	6	55 00 2C 77 AA 73
4.384570	CAN 1	81		Rx	6	77 04 8D 55 00 2C
4.387253	CAN 1	80	NM Master kikapcsolása, Utolsó NM üzenet	Rx	6	55 00 2D 77 AA 72
4.402960	CAN 1	40		Rx	8	00 00 04 00 00 00 00 00
4.519436	CAN 1	41		Rx	8	37 40 01 00 00 00 00 00
4.884594	CAN 1	81		Rx	6	77 04 8E 55 00 2D
4.887511	CAN 1	80		Rx	6	55 00 2E 77 AA 71
4.998410	CAN 1	40		Rx	8	00 00 04 00 00 00 00 00
5.384592	CAN 1	81		Rx	6	77 04 8F 55 00 2E
5.388705	CAN 1	80		Rx	6	55 00 2F 77 AA 70
5.598400	CAN 1	40		Rx	8	00 00 04 00 00 00 00 00
5.884606	CAN 1	81		Rx	6	77 04 90 55 00 2F
5.887567	CAN 1	80		Rx	6	55 00 30 77 AA 6F
6.202872	CAN 1	40		Rx	8	00 00 04 00 00 00 00 00
6.384614	CAN 1	81		Rx	6	77 04 91 55 00 30
6.387338	CAN 1	80		Rx	6	55 00 31 77 AA 6E
6.802603	CAN 1	40		Rx	8	00 00 04 00 00 00 00 00
6.884621	CAN 1	81		Rx	6	77 04 92 55 00 31
6.887756	CAN 1	80		Rx	6	55 00 32 77 AA 6D
7.384633	CAN 1	81		Rx	6	77 04 93 55 00 32
7.387626	CAN 1	80		Rx	6	55 00 33 77 AA 6C
7.398462	CAN 1	40		Rx	8	00 00 04 00 00 00 00 00
7.884644	CAN 1	81		Rx	6	77 04 94 55 00 33
7.887451	CAN 1	80		Rx	6	55 00 34 77 AA 6B
8.002761	CAN 1	40		Rx	8	00 00 04 00 00 00 00 00
8.384653	CAN 1	81		Rx	6	77 04 95 55 00 34
8.387480	CAN 1	80		Rx	6	55 00 35 77 AA 6A
8.601098	CAN 1	40		Rx	8	00 00 04 00 00 00 00 00
8.884666	CAN 1	81		Rx	6	77 04 96 55 00 35
8.887484	CAN 1	80		Rx	6	55 00 36 77 AA 69
9.203602	CAN 1	40	Utolsó Master üzenet	Rx	8	00 00 04 00 00 00 00 00
9.384674	CAN 1	81		Rx	6	77 04 97 55 00 36
9.387393	CAN 1	80		Rx	6	55 00 37 77 AA 68
9.803489	CAN 1	40		Rx	8	00 00 04 00 00 00 00 00
10.398727	CAN 1	40		Rx	8	00 00 04 00 00 00 00 00
11.001494	CAN 1	40		Rx	8	00 00 04 00 00 00 00 00

11.3. ábra, NM Master be és kikapcsolása

A harmadik megfigyelendő eset, amikor a PN master már elkezdi a kikapcsolást a CanNm állapotgépe már „Prepare Bus Sleep” állapotba vált, de egy újabb kérés hatására ismét bekapcsol. Ekkor – mivel az „Immediate Restart” engedélyezett -, az aktív NM-nek kisebb periódusidővel kell küldenie a hálózatmenedzsment üzeneteket (11.4. ábra). Megfigyelhető, hogy a CanNm 3s 958ms-kor elküldi az utolsó NM üzentet. Ekkor a CanNm-ben az állapotgép „Ready Sleep”, majd „Prepare Bus Sleep” állapotba vált. Ez debugger-el figyelhető, és egy ebben az állapotban adott „User Request-el” a CanNm állapotgépe visszavihető „Repeat Message” állapotba, ahol teljesül az „Immediate Restart” vagyis ECU 5-db üzenetet küld 200ms-onként, majd az utolsó „Immediate” üzenetek után, az első hálózatmenedzsment üzenet 0.1ms-al kerül elküldésre, ami megegyezik a MsgCycleOffset idővel. Ezek után a további NM üzenetek a normális 1ms periódusidővel kerülnek küldésre.



11.4. ábra, NM Master, azonnali újraindítás

12. Összefoglalás

Ez a dolgozat a diplomatervezés két félévének záró anyagaként szolgál, melynek célja egyfelől egy átfogó kép nyújtása az AUTOSAR szabványról, azon belül is a hálózatmenedzsment funkcionalitásról, másfelől a diplomaterv feladat megvalósításának célja egy részhálózatokkal rendelkező rendszerbe illeszkedő ECU szoftver megalkotása volt. Ennek legfőbb célja, hogy a jövőbeni hálózatmenedzsmentet használó projektekben az itt megszerzett tudást felhasználva, elősegítse azok gördülékenyebb megvalósítását. Ez az elképzelés nem is nyúlik a távoli jövőbe, mivel a diplomaterv befejezésével egy időben, már aktuálissá vált az itt megszerzett tapasztalatok kamatoztatása.

Először a dolgozatban specifikálásra került a megvalósítandó feladat, majd ezután az olvasó a diplomaterv feladat hardveres oldalát ismerhette meg.

Ezt követően bemutatásra került az AUTOSAR szabvány, mind modellezési nyelv aspektusból, mind pedig a szabványban definiált szoftverarchitektúra szempontjából. Ezután az olvasó megismerhette a kommunikációs blokkon belül elhelyezkedő hálózatmenedzsment modulok szerepét, illetve együttműködésüket, mind egymással, mind az BSW többi moduljával. Ezek után pár példán keresztül bemutatásra került a hálózatmenedzsment működése.

Az elméleti áttekintés után a diplomatervezés feladat megvalósításának ismertetésére került sor, amely két nagyobb részfeladatra bontható. Egyrészről a beágyazott eszközre implementálandó alkalmazásokra, amelyekkel szemléletesen demonstrálható a hálózatmenedzsment funkciók működése. Másrészről egy olyan PC-s alkalmazásra, amellyel a hagyományos ECU-k hálózatmenedzsment funkciói is tesztelhetőek.

Végezetül néhány egyszerű tesztelési folyamat került bemutatásra, amellyel igazolható, hogy az összeállított rendszer valóban képes hálózatmenedzsmentet tartalmazó rendszerbe illeszkedni, tehát a feladatkiírásban kitűzött célok megvalósultak.

Azonban ez a demonstrációs feladat is tartalmaz néhány lehetőséget a továbbfejlesztésre. Egyrészről további komplexebb konfigurációkkal a hálózatmenedzsment extra funkciói is kipróbálhatóak, mint a busz terheltségének csökkentése, vagy a kikapcsolás koordinálása. Egy másik lehetőség lenne a PC oldali

teszteszköz továbbfejlesztése, aminek kapcsán lehetőség nyílik a jelenleg nem támogatott „Partial Networking”-et leíró modellek feldolgozásra, így a „Partial Network”-okhöz tartozó szignálok modellből való kinyerésére.

Rövidítések

Rövidítés	Kifejtés	Leírás
BSW	Basic Software	Az AUTOSAR szoftverarchitektúra legalsó rétege
CBV	Control Bit Vector	Hálózatmenedzsment üzenet hálózatvezérlő bitvektora
DLC	Data Length Code	CAN üzenetek hosszazonosítója
ECU	Electronic Control Unit	Elektronikus vezérlőegység
Gateway		Kommunikációs hálózatok között átjáró
NID	Node Identifier	Hálózatmenedzsment üzenetekben a küldő azonosítója
NM	Network Management	Hálózatmenedzsment
NM-Message	Network Management Message	Hálózatmenedzsment üzenet
PDU	Protokol Data Unit	BSW réteg kommunikációs adategysége
Piggy		Cserélhető kommunikációs busz szintillesztője
PN	Partial Network	Hálózatmenedzsment részhálózat
PNC	Partial Network Cluster	Részhálózatokat támogató ECU-k csoportja
RTE	Runtime Enviroment	Az AUTOSAR szoftverarchitektúra középső rétege
Signal		Alkalmazás réteg kommunikációs adategysége
SWC	Software Component	Az AUTOSAR szoftverarchitektúra legfelső rétegének eleme
VFB	Virtual Function Bus	Virtuális függvény busz, amely az SWC-ket köti össze
WUF	Wakeup Frame	CAN hálózat ébresztését jelző üzenet

Irodalomjegyzék

[1] AUTOSAR development partnership homepage:
http://www.autosar.org (2014 december)

[2] AUTOSAR development partnership: Layered Software Architecture 3.2.0, 2011.
http://www.autosar.org/fileadmin/files/releases/4-0/software-architecture/general/auxiliary/AUTOSAR_EXP_LayeredSoftwareArchitecture.pdf
(2014 december)

[3] AUTOSAR development partnership: Specification of CAN Interface 5.0.0, 2011.
http://www.autosar.org/fileadmin/files/releases/4-0/software-architecture/communication-stack/standard/AUTOSAR_SWS_CANInterface.pdf
(2014 december)

[4] AUTOSAR development partnership: Specification of CAN State Manager 2.2.0, 2011.
http://www.autosar.org/fileadmin/files/releases/4-0/software-architecture/communication-stack/standard/AUTOSAR_SWS_CANStateManager.pdf
(2014 december)

[5] AUTOSAR development partnership: Specification of PDU Router 3.2.0, 2011.
http://www.autosar.org/fileadmin/files/releases/4-0/software-architecture/communication-stack/standard/AUTOSAR_SWS_PDURouter.pdf
(2014 december)

[6] AUTOSAR development partnership: Specification of Communication 4.2.0, 2011.
http://www.autosar.org/fileadmin/files/releases/4-0/software-architecture/communication-stack/standard/AUTOSAR_SWS_COM.pdf
(2014 december)

[7] AUTOSAR development partnership: Specification of CAN Network Managemet 3.3.0, 2011.
http://www.autosar.org/fileadmin/files/releases/4-0/software-architecture/communication-stack/standard/AUTOSAR_SWS_CANNetworkManagement.pdf
(2014 december)

[8] AUTOSAR development partnership: Specification of ECU State Manager with fixed state machine 1.2.0, 2014
http://www.autosar.org/fileadmin/files/releases/4-0/software-architecture/system-services/standard/AUTOSAR_SWS_ECUStateManagerFixed.pdf
(2014 december)

[9] AUTOSAR development partnership: Specification of Network Management Interface 3.0.0, 2011.
http://www.autosar.org/fileadmin/files/releases/4-0/software-architecture/communication-stack/standard/AUTOSAR_SWS_NetworkManagementInterface.pdf
(2014 december)

[10] AUTOSAR development partnership: Specification of Communication Manager 4.0.0, 2011.
http://www.autosar.org/fileadmin/files/releases/4-0/software-architecture/communication-stack/standard/AUTOSAR_SWS_COMManager.pdf
(2014 december)

[11] Konkoly Balázs, AUTOSAR kompatibilis FlexRay stack integrálása autóipari vezérlőegység szoftverébe, szakdolgozat, BME, 2011.

[12] Boda Tamás, Autóipari vezérlőegységek energiamenedzsment tesztjének fejlesztése, szakdolgozat, BME, 2011.

[13] Hegyi Balázs, AUTOSAR hálózatmenedzsment modulok megvalósítása, szakdolgozat, BME, 2012.

[14] Faragó Dániel, Eclipse környezetbe integrált járműbusz vizualizáció, diplomaterv, BME, 2014.

[15] Sisak Gergely, Eclipse környezetbe integrált járműbusz szimuláció, diplomaterv, BME, 2014.

[16] AUTOSAR development partnership: System Template 4.2.0, 2011.
http://www.autosar.org/fileadmin/files/releases/4-0/methodology-templates/templates/standard/AUTOSAR_TPS_SystemTemplate.pdf
(2015 május)

[17] AUTOSAR development partnership: Software Component Template 4.2.0, 2011.
http://www.autosar.org/fileadmin/files/releases/4-0/methodology-templates/templates/standard/AUTOSAR_TPS_SoftwareComponentTemplate.pdf
(2015 május)

[18] AUTOSAR development partnership: Specification of ECU Configuration 3.2.0, 2011.
http://www.autosar.org/fileadmin/files/releases/4-0/methodology-templates/templates/standard/AUTOSAR_TPS_ECUConfiguration.pdf
(2015 május)

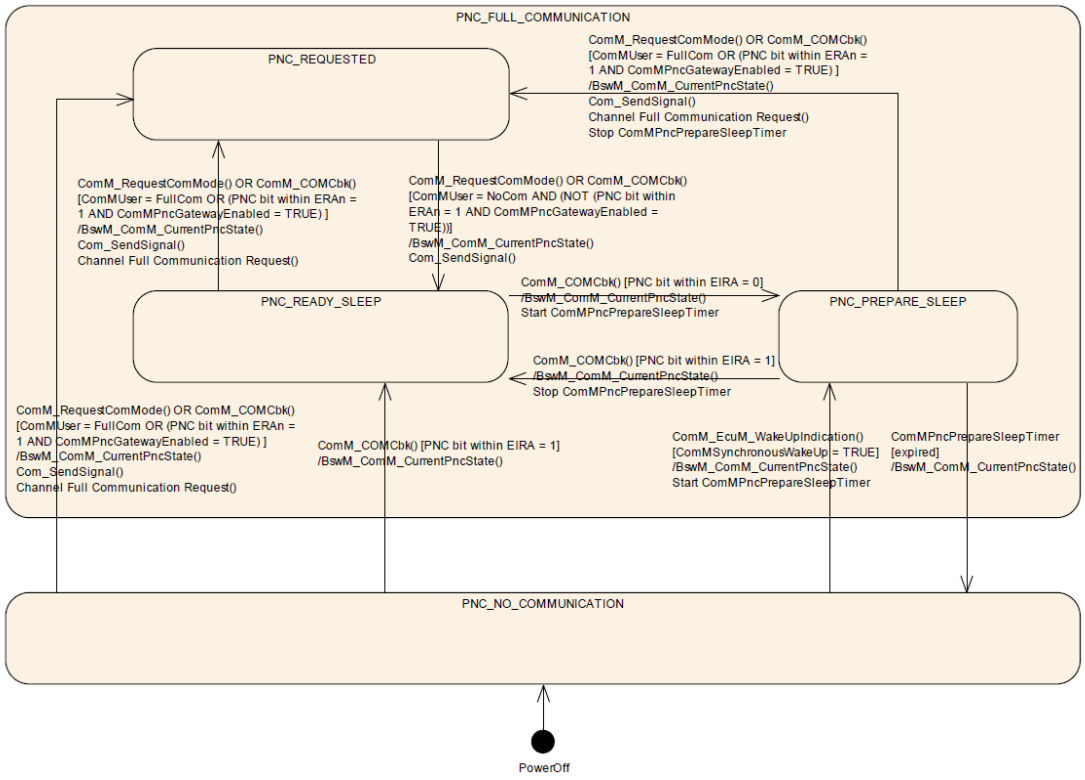
[19] AUTOSAR development partnership: Specification of RTE 3.2.0, 2011,
http://www.autosar.org/fileadmin/files/releases/4-0/software-architecture/rte/standard/AUTOSAR_SWS_RTE.pdf
(2015 május)

[20] NXP: TJA1145 High-speed CAN transceiver for partial networking Rev2, 2014. (adatlap)
http://www.nxp.com/documents/data_sheet/TJA1145.pdf
(2015 május)

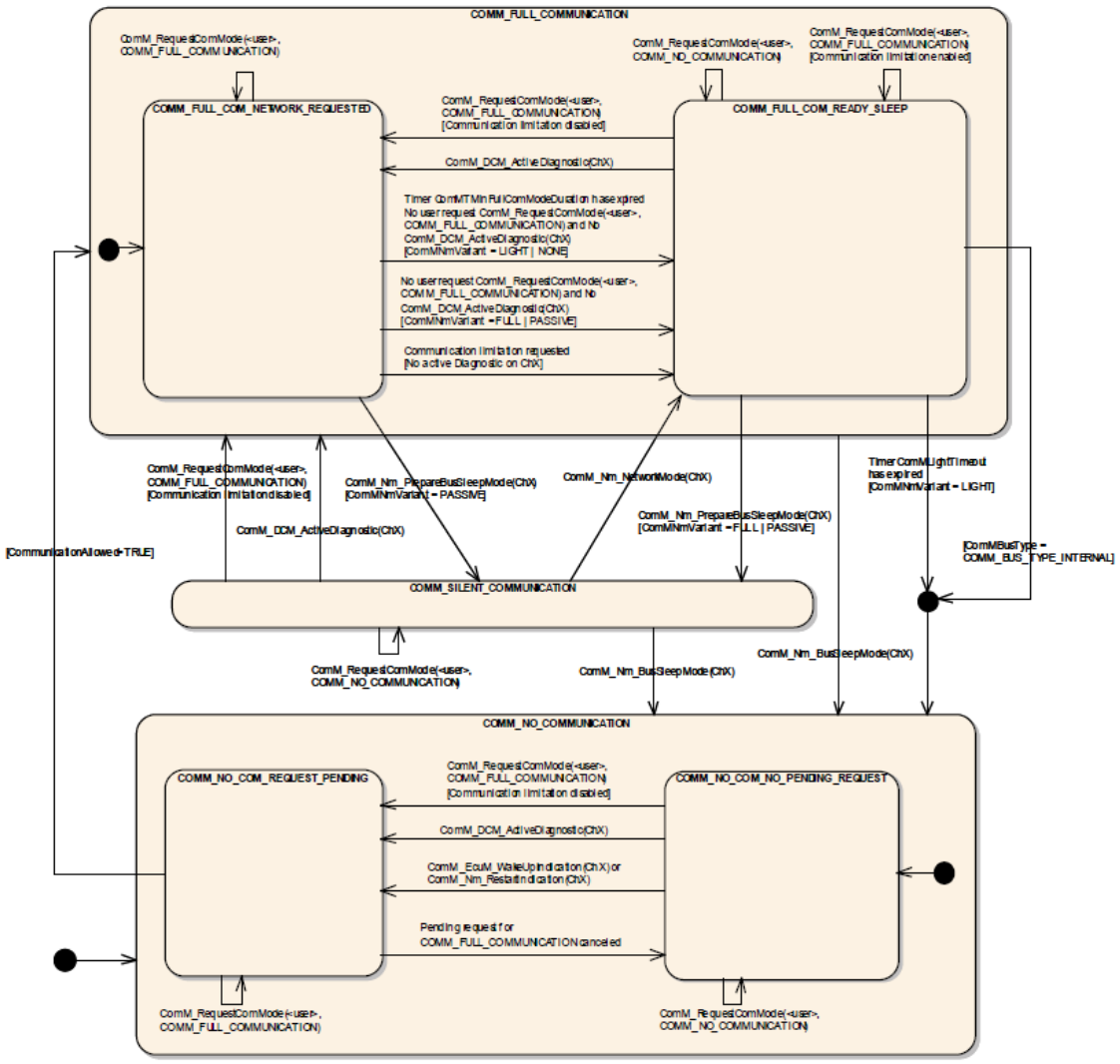
[21] Freescale: MPC5744P Data Sheet Rev3, 2014.
http://cache.freescale.com/files/32bit/doc/data_sheet/MPC5744P.pdf?pspll=1
(2015 május)

[22] OSEK/VDX, Network Management, Concept and Application Programming Interface 2.5.3, 2004
http://portal.osek-vdx.org/files/pdf/specs/nm253.pdf
(2015 május)

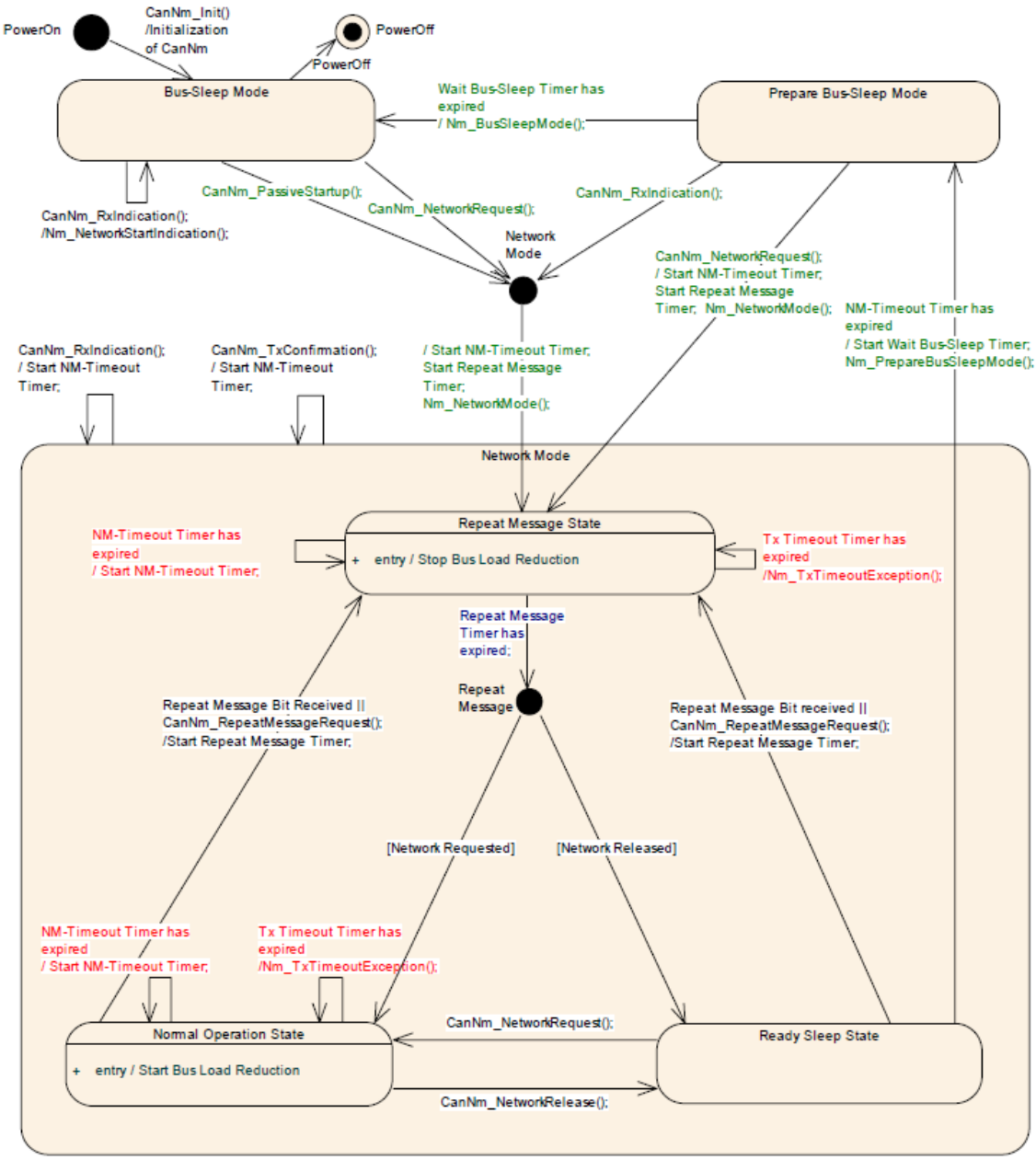
Függelék



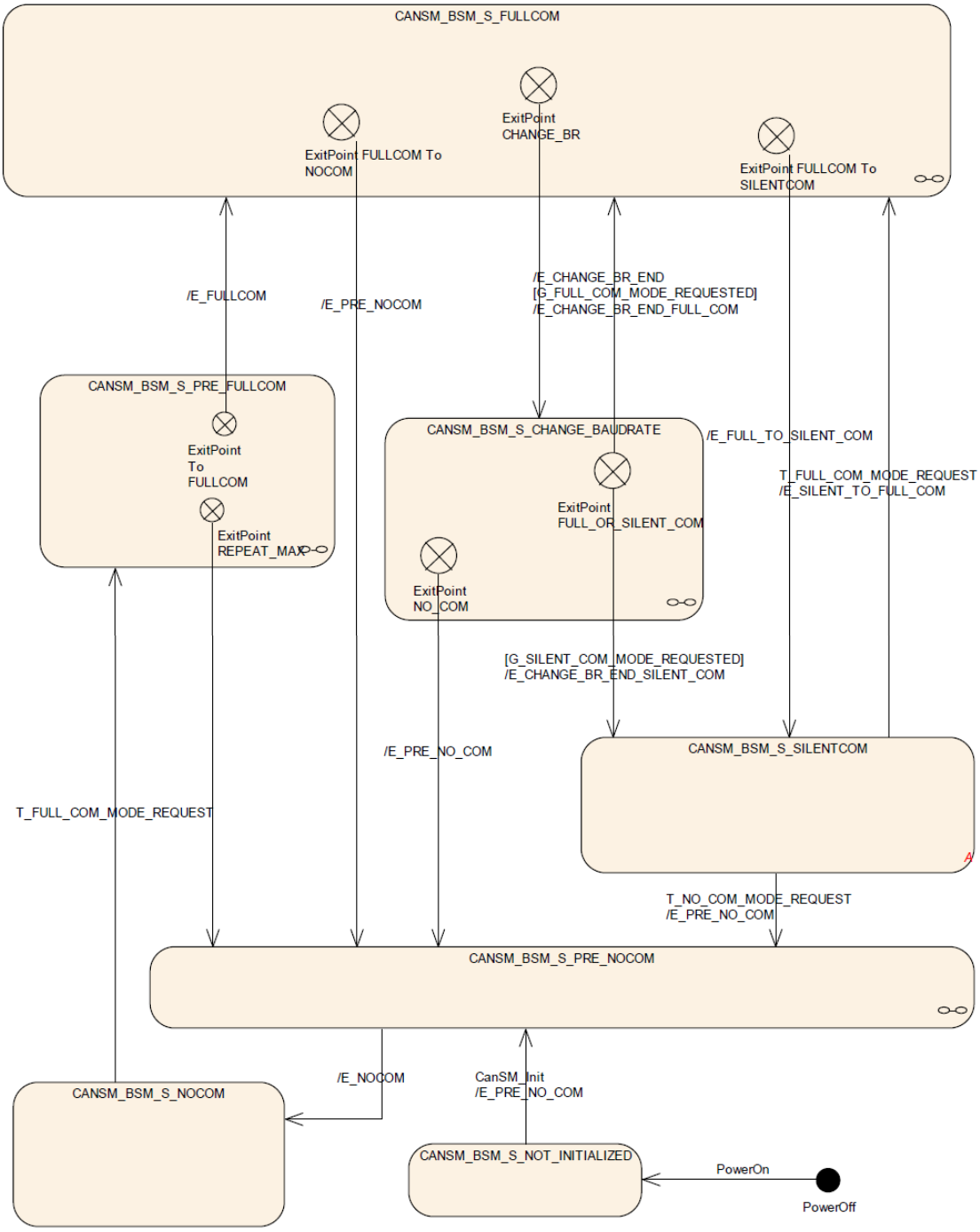
1. Függelék. PNC állapotgép a ComM modulban



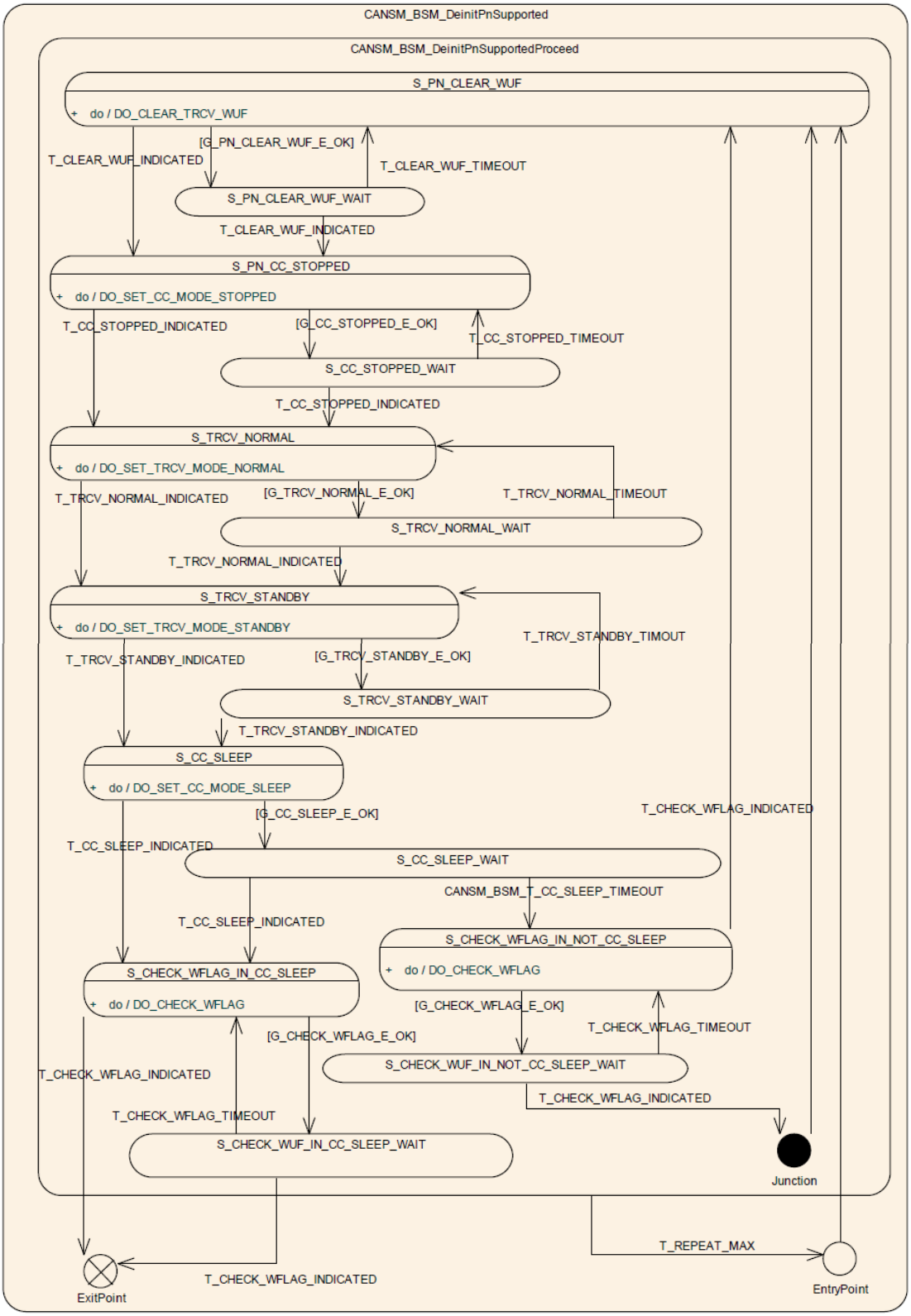
2. Függelék. Logikai csatorna állapotgépe a ComM modulban



3. Függelék. Kommunikációs busz állapotgépe a CanNm modulban



4. Függelék. Csatorna fő állapotgépe a CanSm modulban



5. Függelék. CanSm részhálózatokat támogató alállapotgépe