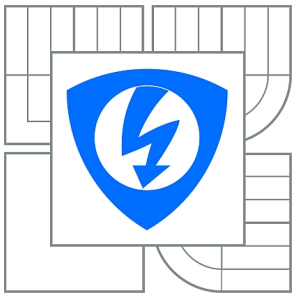
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

DEPARTMENT OF CONTROL AND INSTRUMENTATION

ADD-ON INSTRUKCE PRO SÍŤ AS-INTERFACE

ADD-ON INSTRUCTIONS FOR AS-INTERFACE NETWORK

DIPLOMOVÁ PRÁCE

MASTER'S THESIS

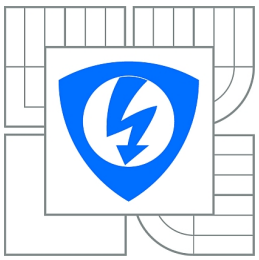
AUTOR PRÁCE — Bc. TOMÁŠ GREPL

AUTHOR

VEDOUCÍ PRÁCE — Ing. RADEK ŠTOHL, Ph.D.

SUPERVISOR

BRNO 2013



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav automatizace a měřicí techniky

Diplomová práce

magisterský navazující studijní obor
Kybernetika, automatizace a měření

Student:	Bc. Tomáš Grepl	***ID:***	119413
Ročník:	2	***Akademický rok:***	2012/2013

NÁZEV TÉMATU:

Add-on instrukce pro síť AS-Interface

POKYNY PRO VYPRACOVÁNÍ:

1. Navrhněte Add-on instrukce pro komunikaci na síti AS-Interface
2. Realizujte a ověřte navržené funkce na modelu Labyrintu.
3. Realizujte vizualizaci modelu pomocí prostředků Rockwell Automation.
4. Vytvořte ukázkovou laboratorní úlohu s využitím Add-on instrukcí.
5. Ověřte funkčnost.

DOPORUČENÁ LITERATURA:

Becker, R. a kol.: AS-Interface, řešení pro automatizaci. AS-International Association, 2004, 184 s. ISBN 80-214-2958-5

Logix5000 Controllers General Instructions (Reference Manual). Milwaukee: Rockwell Automation, Inc. 2008.

Logix5000 Controllers Add-on Instructions (Programming Manual). Milwaukee: Rockwell Automation, Inc. 2011.

Dle vlastního literárního průzkumu a doporučení vedoucího práce.

Termín zadání:	11.2.2013	***Termín odevzdání:***	20.5.2013

Vedoucí práce: Ing. Radek Štohl, Ph.D.
Konzultanti diplomové práce:

doc. Ing. Václav Jirsík, CSc.
Předseda oborové rady

UPOZORNĚNÍ:

ABSTRAKT

Diplomová práce se zabývá návrhem a tvorbou instrukcí pro průmyslovou sběrnici AS-Interface. Tyto instrukce byly realizovány jako add-on instrukce v prostředí ControlLogix5000. Teoretická část práce uvádí čtenáře do problematiky průmyslového řízení, seznamuje se sběrnicí AS-Interface, add-on instrukcí a modelem labyrintu. V praktické části je předložen návrh add-on instrukce a objasněny její funkce. Dále je v diplomové práci uvedena ukázková instrukce a následně seznam všech realizovaných instrukcí. V závěrečných kapitolách jsou popsány vizualizace a laboratorní úloha.

KLÍČOVÁ SLOVA

AS-Interface, Add-on instrukce, ControlLogix5000, PLC, labyrint

ABSTRACT

This master's thesis deal with design and creating of the instructions for industry bus AS-Interface. These instructions were implemented in the software ControlLogix5000 like add-on instructions. In the teoretical part there are presented the issues of industrial control, AS-Interface bus, add-on instruction and model of the labyrinth. The practical part of the master's thesis is focused to design of the add-on instruction and work with instruction and its characteristics followed by the description of the sample add-on instruction and the list of all implemented instructions. The final two chapters include visualization and laboratory exercise.

KEYWORDS

AS-Interface, Add-on instruction, ControlLogix5000, PLC, Model of labyrinth

GREPL, Tomáš *Add-on instrukce pro síť AS-Interface*: diplomová práce. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav automatizace a měřicí techniky, 2013. 100 s. Vedoucí práce byl Ing. Radek Štohl, Ph.D.

PROHLÁŠENÍ

Prohlašuji, že svou diplomovou práci na téma „Add-on instrukce pro síť AS-Interface“ jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno ....

(podpis autora)

PODĚKOVÁNÍ

Chtěl bych poděkovat rodičům za podporu při tvorbě práce. Dále mé snoubence Báře Oravčíkové, která mě podporovala a pomáhala mi s jazykovou korekturou. A největší část díků patří vedoucímu diplomové práce panu Ing. Radku Štohlovi, Ph.D. za odborné vedení, náhled nad řešenou problematikou, ochotu a trpělivost s mou osobou.

Brno ....

(podpis autora)

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

ÚVOD

Diplomová práce se zabývá průmyslovou sběrnicí AS-Interface. Úkolem práce je vytvořit obslužné instrukce k této sběrnici, které budou realizované jako add-on instrukce. Zároveň pomocí instrukcí vytvořit vizualizaci, laboratorní úlohu a ověřit jejich funkčnost. Práce je realizována a odzkoušena na modelu labyrintu.

Sběrnice AS-Interface je průmyslová sběrnice, která pracuje na nejnižší úrovni automatizace. Spojuje snímače a aktuátory s nadřazenými automatickými systémy. Slovo, které by sběrnici mohlo charakterizovat, je úspora. Vznik AS-Interface je spojen s možností propojit jediným vodičem prvky různých výrobců. S tím související zjednodušení kabeláže, která obsahuje pouze dva vodiče uložené v jednom kabelu. Paket, který je vysílán na sběrnici, se skládá z několika málo bitů. I zařízení na sítí zvané slavy jsou co nejlevnější a nejjednodušší. Sběrnice nepotřebuje zakončovat své vedení a umožňuj kromě datové komunikace i distribuci napájení jedním kabelem. S podobnými myšlenkami jsem se pokusil realizovat instrukce pro komunikaci na síti. A to zvláště, aby byly jednoduché, přehledné, krátké, univerzální, aby nebyly vázány na jiné instrukce a rovněž aby je bylo možno používat i na jiném než zde použitém zařízení.

V rámci diplomové práce byly vytvořeny add-on instrukce, které řeší komunikaci, diagnostiku a různá nastavení na síti AS-Interface. Realizovány byly na PLC firmy Rockwell Automation a AS-I masteru Bihl+Wiedemann BWU1488, ke kterému byla připojena síť z modelu labyrintu. Instrukce a program byly vytvořeny v prostředí ControlLogix5000 v jazyce LAD a vizualizace v programu Factory Talk View Studio.

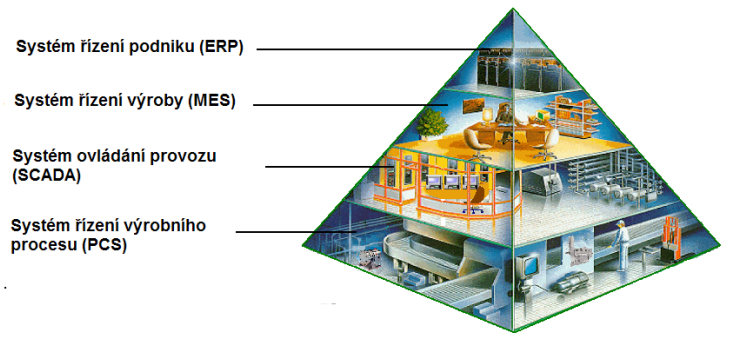
V úvodu teoretické části je čtenář seznámen s problematikou průmyslového řízení, jeho členěním a informacemi o programování a principu činnosti PLC. Následuje část, která se věnuje síti AS-Interface, ve které je popsána komunikace na síti, princip činnosti a začlenění mezi ostatní průmyslové sběrnice. Třetí kapitola se věnuje add-on instrukcím, nabízí jejich charakteristiku a možnosti využití. V závěrečné kapitole teoretické části je představen hardware využitý při zpracovávání praktické části diplomové práce, použitý maser a popsán model labyrintu.

Praktická část diplomové práce se v úvodu věnuje hardwarové konfiguraci PLC, návrhu add-on instrukce, jejímu programování, vlastnostem a práci s ní. Poté je představen jeden konkrétní příklad realizované add-on instrukce pro síť AS-I a uveden seznam všech realizovaných instrukcí. Další kapitola popisuje fungování vizualizaci a programu, který vizualizace obsluhuje. V poslední kapitole je uvedena vzorová laboratorní úloha, která zahrnuje zadání, teoretickou část a postup vypracování.

1 PROGRAMOVATELNÉ AUTOMATY A ŘÍZENÍ VÝROBY

Historie programovatelných automatů (PLC z anglických slov Programmable Logic Controlers) sahá do 60. let 20. století, kdy byla snaha nahradit efektivnějším způsobem reléovou a bezkontaktní logiku. Programovatelné automaty umožňují naprogramování logických rovnic, zatímco předcházející bezkontaktní nebo reléová logika řeší požadované logické rovnice pomocí fyzického zapojení. Jejich architektura je tedy navržena tak, aby zpracovávala binární informaci[1].

PLC je zařízení pro zpracování a vyhodnocení signálů z technologického procesu, které jsou přivedeny na jeho vstupy (poté se hovoří o vstupních signálech). Vyhodnocení probíhá na základě naprogramovaných sekvenčních logických a časových funkcích, díky nimž vydává na výstupy (výstupní signály) povely, kterými se ovládají motory, ventily, klapky, topná tělesa aj., nazývané aktuátory. Programovatelný automat poskytl v řídicím systému pružnost, snadnou rozšiřitelnost a modifikovatelnost, rychlou diagnostiku poruchy vstupních a výstupních signálů a zvýšenou spolehlivost řídicího systému. Jejich zavedením se do řízení technologických procesů a podniků mohla ustálit konfigurace řídicích informačních systémů výroby podniku, která je zobrazena na obrázku 1.1 [2]:



Obr. 1.1: Struktura řídicího a informačního systému podniku[2]

Tato struktura řízení podniku mohla vzniknout díky standardizaci PLC jako průmyslového prvku, který byl propojen sítí, která monitorovala chod celého systému a předávala tak informace mezi jednotlivými částmi řízení podniku.

Význam jednotlivých částí řídicího a informačního systému podniku[2]:

- **PCS**

  Je základní strukturou v řízení podniku, která zajišťuje fyzické řízení strojů. Patří sem snímače, aktuátory, PLC a sběrnice pro tuto vrstvu určené. Podrobnější popis sběrnic bude v následující kapitole.

- **SCADA**

  Tento systém umožňuje operátorovi nebo nadřazenému systému sledovat provoz technologie a také do něho zasahovat prostřednictvím ovládacích povelů pro řídicí systém. To se většinou děje pomocí vizualizace, která je grafickým znázorněním procesu s příslušnými daty a ovládacími prvky, a která běží na stolním počítači jako spuštěný software. Také se jí přezdívá okno do procesu.
  Systém SCADA může poskytovat tyto údaje:
  - Provozně - technické: měření provozních veličin, informace o správných krocích procesu
  - Poruchové: poruchy v procesu
  - Statistické: provozní hodiny, počty kusů, vyhodnocení chodu zařízení, aj.
  - Bilanční: informace o energetických a materiálových tocích
  - Logistické: informace o materiálových tocích v závislosti na sortimentu, stavy skladů apod.

- **MES - systém operativního řízení výroby**

  Jedná se o propojovací článek mezi řízením technologických procesů a ostatními informačními systémy (kam patří zejména ERP). Jeho úlohou je integrovat systémy do jednoho celopodnikového systému, čímž se dosáhne odstranění lidského faktoru z informačního řetězce a lze dosáhnout zvýšené produktivity výroby a pružnosti celého podniku.
  Funkční oblasti MES:
  - Krátkodobé rozvrhování: tvorba krátkodobých harmonogramů výroby, aby se zamezilo zbytečnému přestavování, čekání, minimalizování spotřeby energie apod.
  - Přidělování zdrojů a kapacit: vytváří předpoklad pro zahájení výroby a její pokrytí pracovní silou a stroji tak, aby bylo možno plnit plány výroby.
  - Dispečerské řízení: přidělování výrobních jednotek, zajišťování surovin, energie, sledování aktuálních stavů výrobního úseku a rozhodování se v případě poruchy nebo havárie, které mohou vzniknout v případě nedostatku surovin, energie apod.
  - Správa dokumentace: starost a evidence veškerých dokumentů.
  - Sledování toku materiálu a cesty produktu: sledování on-line podmínek a parametrů na zařízeních, kde byly produkty vyrobeny a uchovávání jejich historie.

  - Analýza výkonnosti: porovnání aktuálních dosahovaných výsledků s jejich krátkodobou historií a predikuje odhad ekonomických výstupů po uplynutí období (směna, týden...).
  - Sledování pracovníků: informace o výrobním personálu, náplni jeho práce. Společně s přidělováním zdrojů optimalizuje výběr pracovníků pro jednotlivé úkoly.
  - Řízení údržby: plánování a řízení údržby tak, aby byly stroje v provozuschopném stavu a předcházelo se neplánovaným odstávkám vlivem poruchy stroje.
  - Ovládání procesů: všechny řídicí, monitorovací a ovládací funkce technologického procesu.
  - Řízení jakosti: v reálném čase řeší analýzu dat snímaných z výrobního zařízení, aby šlo sledovat kvalitu vyráběného produktu.
  - Sběr dat: hlavní prvek systému MES. Řeší archivaci a přístup k datům pro všechny ostatní části MES.
- **ERP - informační systém řízení podniku**
  Dle typu informací by se podnik dal rozdělit na dvě části, dva samostatné světy, kancelářskou a provozní část. Informace pro kancelářské využití zpracovává systém ERP, který představuje obchodní část počítačového systému podniku. Pracuje s objednávkami, kvalitativními a technickými specifikacemi surovin a výrobků. Zpracovává požadavky na materiál, ceny a potřebné údaje k obchodním činnostem podniku. Velká část informací poskytovaných ze SCADA systému je zpracovávána v ERP, především statistické, bilanční a logistické informace.

1.1 Programovatelné automaty

Hardwarová struktura PLC vychází z požadavků na funkčnost, avšak základní strukturu tvoří napájecí zdroj, centrální procesor CPU a moduly vstupních a výstupních signálů 1.2. Pro malé aplikace se používají tzv. kompaktní PLC, ve kterých jsou všechny tři základní části sdruženy do jednoho pouzdra a nelze měnit jeho konfiguraci. Avšak toto řešení je možné jen pro velmi malou skupinu aplikací. Převážná většina aplikací používá modulární PLC, ve kterém lze využívat mnoho druhů CPU, vstupně/výstupních modulů a celé řady dalších rozšiřujících karet. Rozšiřující karty mohou sloužit jako komunikační, regulátory, frekvenční měniče a další, což je vidět právě v obrázku 1.2, kde jsou použity 2 jednotky CPU, s rozšiřujícími kartami pro komunikaci přes síť Ethernet a více druhů I/O karet. Tyto karty jsou v podobě zásuvných modulů, které se zasunují do tzv. racku, ve kterém je určena jejich pozice

a jsou spojeny vnitřní sběrnicí PLC. Výrobci PLC mají spojování řešeno různými způsoby, někteří používají lišty, jiní již výše zmíněný rack.[2]

Obr. 1.2: Základní sestava PLC[19].

1.1.1 Propojování programovatelných automatů v sítích

Programovatelné automaty lze v různých úrovních propojovat sítěmi, které usnadňují přenos dat od senzorů, povelů aktuátorům. Také přenos dat mezi jednotlivými programovatelnými automaty a nadřazenými systémy. Průmyslových sběrnic je mnoho typů [3]:

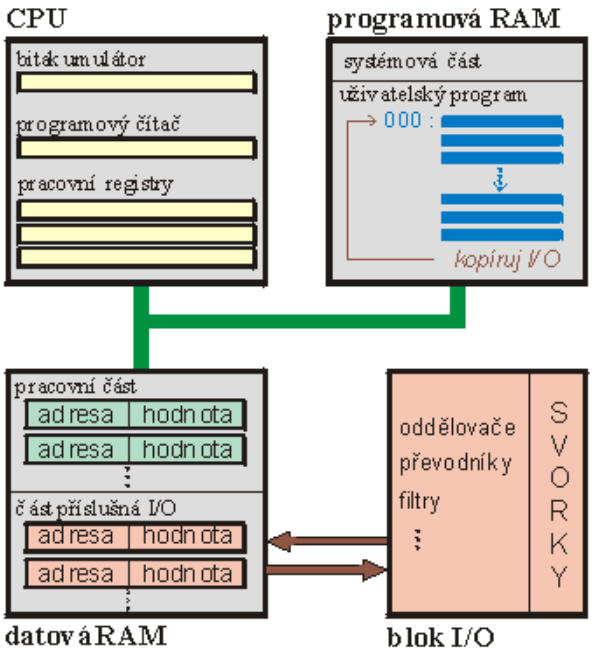
- Multi-Point Interface (MPI)
- Profibus
- Foundation Fieldbus
- Profinet
- CAN
- DeviceNet
- CANopen
- ControlNet
- Modbus (TCP/IP)
- Ethernet
- BacNet
- Point-to-point (PtP)
- AS-Interface

1.1.2 Programování PLC

Programovací jazyky PLC jsou značně jednoduché. Stále vycházejí ze svého původního poslání, kterým je možnost přejít z HW řešení boolovských rovnic (pomocí reléové bezkontaktní logiky) k řešení boolovských rovnic programem pro PLC. Z tohoto důvodu převažují tyto jazyky: [1]

- LAD - jazyk reléových schémat, také nazýván příčkový diagram
- FBD - jazyk funkčních bloků
- STL - strukturovaný text, který vychází z jazyka Assembler

1.1.3 Princip činnosti PLC



Obr. 1.3: Princip činnosti PLC[20]

Činnost programovatelných logických automatů je prováděna cyklickým řídicím programem, jinými slovy skenováním. Celý proces je zobrazen na obrázku 1.3. V centrální jednotce CPU jsou všechny operace prováděny mezi registry. Pro základní činnost jsou nejdůležitější první dva registry. Registr zvaný *programový čítač*, který slouží k uložení adresy instrukce v paměti, jenž se bude provádět v příštím kroku. Zabezpečuje tedy čtení programu z paměti provádění jednotlivých instrukcí za sebou. Registr *bitakumulátor* provádí logické operace. Na začátku každé logické operace je do něj uložen první operand, druhý operand se nachází v pracovním registru a výsledek operace je uložen opět do bitakumulátoru. Logická hodnota v bitakumulátoru určuje, zda se bude následující instrukce, která modifikuje obsah datové paměti a

tím i výstupy z automatu do procesu, provede (logická 1) a nebo neprovede (logická 0).

Jednotlivé příkazy, které jsou uloženy v programové části paměti, jsou vykonávány cyklicky. Datová paměť je rozdělena na dvě části. Jedna obsahuje přímo paměťové bloky, které jsou svázány se vstupy a výstupy. A druhá, která slouží k ukládání konstant, mezivýsledků výpočtů a jiných potřebných dat. Před zahájením cyklu se načtou hodnoty všech vstupů a zapíší se do příslušného segmentu datové paměti. V průběhu cyklu je obsah celé datové paměti modifikován a po ukončení cyklu se hodnoty výstupních proměnných přenesou z datové paměti na výstupy.[4]

2 SBĚRNICE AS-INTERFACE

První dvě písmena v názvu sběrnice AS znamenají Actuator (aktuátor) a Sensor (senzor), což vystihuje určení sběrnice. Často se pro její označení používá zkratka AS-i.

AS-interface je komunikační standard na nejnižší úrovni automatizace, tedy mezi snímači a akčními členy pro řízení technologických procesů. Tento systém umožňuje začlenit senzory a aktuátory od různých výrobců do sítě jedním propojovacím kabelem. Základem pro vytvoření této technologie byl požadavek na jednoduché, levné a transparentní řešení komunikace. Poznávacím znamením a také obchodní značkou se stal žlutý profilovaný kabel, který může přenášet společně data i napájení. Umožňuje tak připojení jednotlivých zařízení do sítě, ať jsou stanice umístěny kdekoliv na kabelu. Logo 2.1 AS-Interface je kromě profilovaného žlutého kabelu také jedním ze základních poznávacích prvku sběrnice[5].

Obr. 2.1: Logo AS-Interface[21]

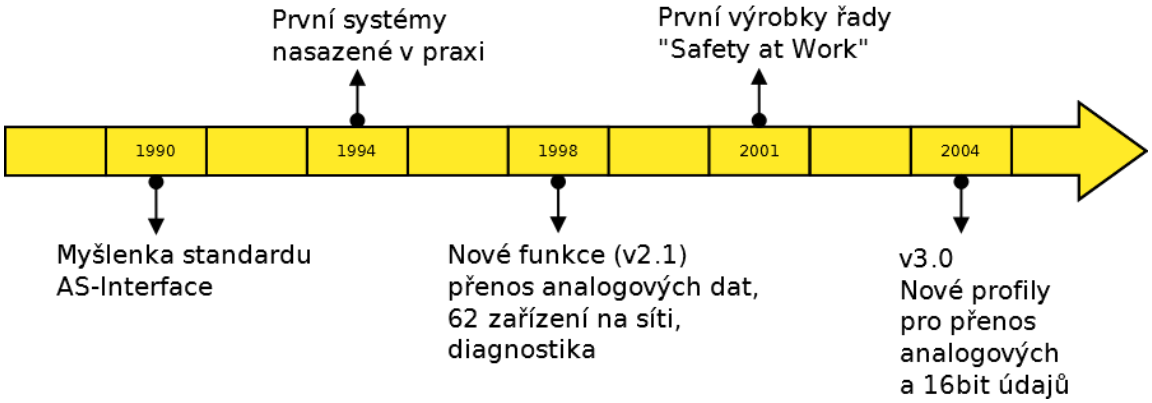
2.1 Vznik komunikačního standardu

Na základě prvotních myšlenek na řešení komunikace vznikly počátkem 90. let minulého století první požadavky na senzorovou síť, které byly navrhovány konsorciem jedenácti velkých výrobců automatizační techniky. Tito výrobci vycházeli z nápadu společné dvouvodičové digitální sériové sběrnice pro snímače a akční členy od různých výrobců, které spolu musejí vzájemně komunikovat. Možnost obecné topologie, společný přenos dat a napájení po jednom vedení, odolnost vůči elektromagnetickému rušení a krátká doba odezvy. Protože typický přenos byl od snímačů k akčním členům, postačovala délka datové zprávy pouhé tři až čtyři bity [6].

Na základě takovýchto požadavků vznikla první verze komunikačního standardu AS-Interface. Po prvotním uvedení do provozu vznikla v roce 1998 verze 2.1, která přinesla rozšíření zařízení na síti na počet 62 a jiné nové funkce. Postupem času byly přidány bezpečnostní prvky pro sběrnici podle normy $EN954-1$[1] (kategorie 4), což

[1]Dnes již není norma platná, nahrazuje ji norma EN $13849-1$

je vidět na časové ose 2.2. V roce 2004 vznikla další specifikace standardu - verze 3.0, která rozšířila přenos dat po sběrnici o 16-ti bitové informace, které se přenáší ve více cyklech sítě (případně v jednom cyklu sítě, za cenu snížení maximálního počtu slavů v síti).[8].



Obr. 2.2: Vývoj komunikačního standardu AS-I[7].

2.2 Specifikace AS-Interface

Typické vlastnosti pro sběrnici jsou [7]:

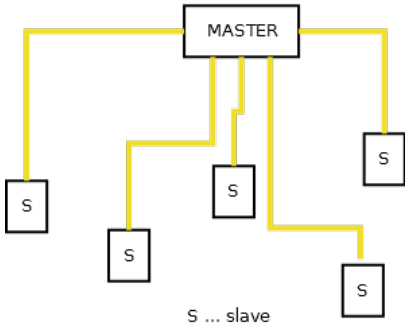
- Typ komunikace Master/Slave, pouze jeden master na síti.
- Komunikace probíhá cyklickým poolingem - master obvolává jednotlivé slavy v každém cyklu sítě. Každá zpráva obsahuje 4 datové bity.
- Topologie sítě - strom, hvězda, kruh, sběrnice.
- Médium - nekroucený nestíněný dvouvodičový kabel, který přenáší jak data, tak napájení. $U_{ss} = 24V$. Proudový odběr celé sítě maximálně $I_{max} = 8A$. Jedno zařízení typu slave odebírá max. $200mA$ dle $v1.0$, dle specifikace $v2.1$ max. $100mA$. V případě potřeby většího proudového odběru, je potřeba dovybavit síť napájením, které je také definováno standardem AS-I.
- Adresace slavů - adresa 0 až 31. Adresu přiřazuje master nebo projektant sítě ručním programátorem. Nový slave má nastavenou adresu 0. Od specifikace $v2.1$ je možnost mít připojeno až 62 slavů. Hodnoty adres $0 - 31$ zůstávají stejné, přičemž k adrese $1 - 31$ je přiřazeno písmeno 'A', 'B' nebo žádné a tím je rozšířen fyzický počet možných adres.
- Přenosová rychlost - $167kbs^{-1}$.
- Počet zařízení: 31 pro $v1.0$, 62 pro $v2.1$
- Cyklus sítě - závisí na počtu připojených zařízení, není delší než $< 4,7ms$ pro 31 zařízení, $< 10ms$ pro 62 zařízení.

- Délka sítě - $100m$ ($300m$ s opakovači). Se zakončovacími prvky dle $v3.0$ bylo dosaženo i dvojnásobné délky sítě.
- Počet I/O - slave má až 4 binární vstupy a až 4 binární výstupy. Celá síť může mít až 124 (128) binárních vstupů a 124 (128) binárních výstupů dle specifikace $v1.0$.
  Celkem 248 vstupů a 186 výstupů dle specifikace $v2.1$.
  Celkem 248 vstupů a 248 výstupů, standardizovaných zařízení 4I/4O a 8I/8O pro rozšířené adresování slavů (62 slavů) dle specifikace $v3.0$.
- Profily - existuje 15 standardních profilů (rozlišené podle ID kódu) dle specifikace $v1.0$
  Existuje 225 standardních profilů zařízení dle specifikace $v2.1$.
- Detekce chyb - způsob modulace a formát zprávy zajišťuje spolehlivou detekci chyb. V případě poškození zprávy je zajištěno její opakování.
- Master zajišťuje inicializaci sítě, identifikaci připojených zařízení, acyklické nastavení parametrů pro všechny slave, detekci chyb při přenosu dat, cyklické obnovování všech slave, hledání nových připojených zařízení, přidělování adresy zařízení, které byly z důvodu poruchy nahrazeny novým zařízením s adresou 0.

2.2.1 Topologie sítě

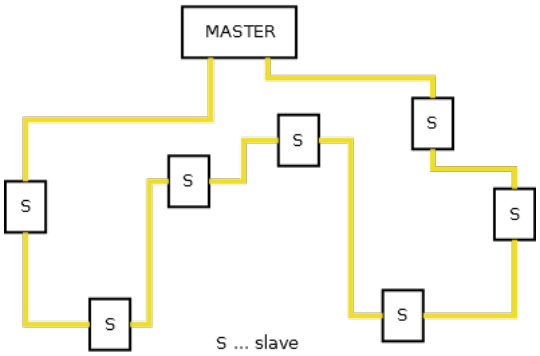
K připojení řídicích jednotek ke vstupům a výstupům na síti, to znamená k senzorům a aktuátorům v provozu, ovladačům a zobrazovačům je obecně potřebná elektrická síť. Termínem topologie se označuje fyzické uspořádání prvků při komunikaci. Toto topologické uspořádání pro sběrnici AS-Interface může být následující [6]:

- **Hvězda**
  Centrální prvek - master je spojen s každým účastníkem sítě dvouvodičovým spojením. Nevýhodou tohoto spojení je neúsporná kabeláž, neboť každý slave má jedno připojení k masteru 2.3.



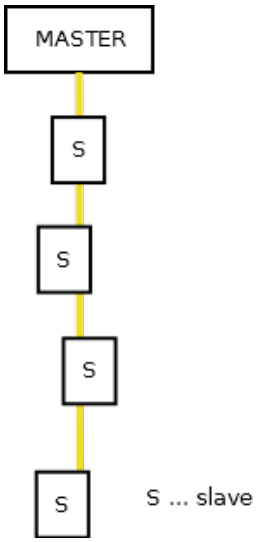
Obr. 2.3: Topologické uspořádání do hvězdy.

- **Kruh**
  Do masteru jsou připojeny pouze dva body. Těmi jsou začátek a konec propojovacího vodiče, přičemž nelze určit, který z nich je začátek a konec. Každý další prvek na sběrnici má také připojeny dvě větve. Po takovémto propojení všech prvků na síti vznikne kruhové uskupení prvků 2.4.



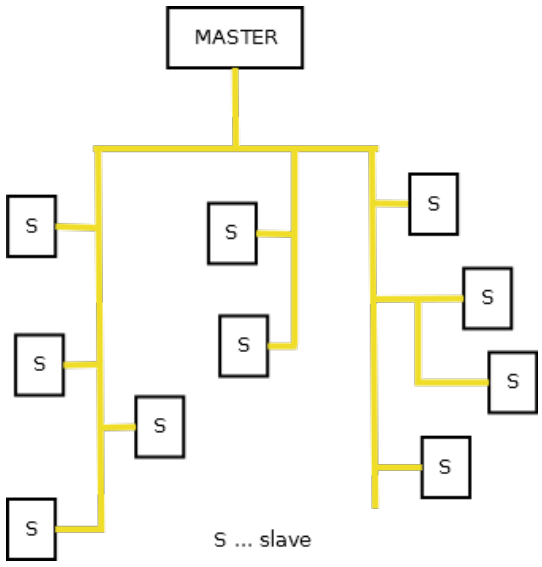
Obr. 2.4: Topologické uspořádání do kruhu.

- **Sběrnice**
  Toto uspořádání lze popsat jako lineární síťové propojení, kdy jsou všichni účastníci propojeni jedním vodičem. Z obecného hlediska se může jednat o nejúspornější variantu využití propojovacího vodiče, protože je ho potřeba na propojení všech prvků nejkratší vzdálenost 2.5. Ovšem konkrétní prostorová dispozice zařízení může z hlediska úspornosti kabeláže vyhovovat jiným topologickým uspořádáním.



Obr. 2.5: Topologické uspořádání sběrnice.

- **Strom**
  Jedná se o univerzální lineární topologii. Délka odboček není omezena a může být i nadále větvena. Celá struktura tak připomíná rozvětvenost stromu 2.6, po kterém dostala název. Výhodou této topologie je možnost jejího uspořádání dle prostorových možností stroje, zařízení. Lze jí pokrýt větší prostor než jinými lineárními strukturami.



Obr. 2.6: Topologické uspořádání stromu.

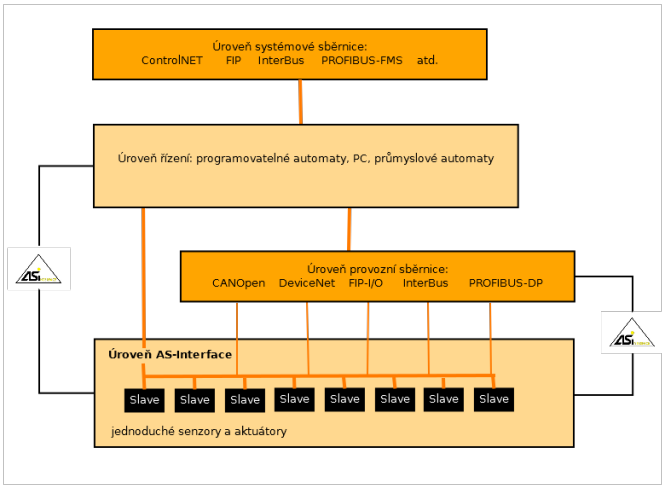
K problematice topologie sítě je nutné uvést, že lze předchozí varianty kombinovat a získat tak zcela jiné uspořádání prvků na sběrnici, které je kombinací výše zmíněných uspořádání. Dostáváme tak tzv. topologii smíšenou, která bude vyhovovat konkrétnímu problému, na který bude sběrnice nasazena a může být z hlediska úspornosti použití kabeláže nejvýhodnější.

2.3 Začlenění AS-Interface do síťových struktur

Začlenění sběrnice lze rozdělit na dva základní typy. Prvním je začlenění do řídicí jednotky, kde je začleněn master a slavy si vyměňují data přímo s řídicí jednotkou. Toto začlenění se používá nejčastěji. Je použito i při řešení této práce.

Druhým typem je začlenění pod nadřazený typ průmyslové sběrnice. Tím je master se slavy účastníkem vyšší průmyslové sběrnice a stává se tak mezisíťovým prvkem, gateway. Toto začlenění umožňuje i vyšší počet sítí AS-Interface pod nadřazenou sítí. Oba typy začlenění jsou znázorněny na obrázku 2.7. Na trhu existují

brány/gateway pro všechny důležité průmyslové sběrnice. Tím není AS-I konkurenčním typem sběrnice, ale doplňujícím doplňkem [9].



Obr. 2.7: Začlenění sběrnice AS-I pod nadřazené řídicí systémy [9].

2.4 Systém komunikace AS-Interface

Obdobně jako ostatní průmyslové sběrnice, lze i AS-Interface popsat ISO/OSI modelem. Na sběrnici jsou realizovány pouze tři ze sedmi vrstev, neboť:

- není podporována smíšená struktura pro alternativní spojení komunikace -> je vynechána vrstva 3
- síťová transportní rozhraní nejsou implementována -> je vynechána vrstva 4
- neprovádí se šifrování ani interpretace dat -> je vynechána vrstva 6

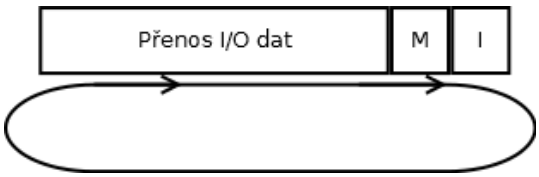
Fyzická vrstva se orientuje na elektrické a mechanické spojení. Přenáší tok informací mezi účastníky na síti.

Linková vrstva je nad touto fyzickou vrstvou a nese odpovědnost za spolehlivý přenos dat. Udává strukturu rámců.

Aplikační vrstva má na starosti povely, obsah dat, posloupnost celého cykli AS-I a chování účastníků (např. při připojení slavu během provozu sítě) [5].

Systém přenosu dat je pomocí dvojvodičového vedení, které propojuje slavy navzájem s masterem. Do tohoto systému patří také zdroj napájení a obvod oddělení dat pro oddělení datových signálů od napájení. Z toho vyplývá, že dvojvodičové vedení přenáší data spolu s rozvodem napájecího napětí. Napájet slavy s vyšší spotřebou lze řešit dalším nestíněným nekrouceným kabelem, pro který se používá černá barva [6].

Komunikace na síti probíhá v opakujících se cyklech 2.8 (cyclic pooling). Ty se skládají z přenosu dat mezi masterem a slavem, části managementu - M a hledání nových zařízení - I. Každá část má pevně danou délku a tím i čas, po který trvá. Master obvolává jednotlivá zařízení na síti a v daném okamžiku je aktivní vždy jen jedno spojení mezi řídicí stanicí a jednou z podřízených stanic. Tím je možné kombinovat v jediném cyklu přenos dat se zasláním parametrů vybrané stanici bez prodloužení periody cyklu sítě. Nebo opakovat výzvu stanici, která neodpovídá. Vedle toho je délka periody cyklu určena počtem slavů na síti, takže lze dosáhnout krátké periody cyklu sítě, kolem 1ms. Při přenosové rychlosti 167 kb/s je čistá přenosová rychlost dat 53,3 kb/s.[9]



Obr. 2.8: Cyklus sítě AS-I [7].

Rámec masteru 2.9 se skládá ze start bitu, řídicího bitu CB, adresy volaného slavu A délky 5 bitů, přenášené informace I také o délce 5 bitů, paritním bitem PB a koncovým bitem.

Obr. 2.9: Rámec masteru na síti AS-I [7].

Odpověď slavu2.10, jeho rámec, obsahuje pouze start bit, 4 bitovou informaci I, paritní bit a koncový bit [7].

Obr. 2.10: Rámec slavu na síti AS-I [7].

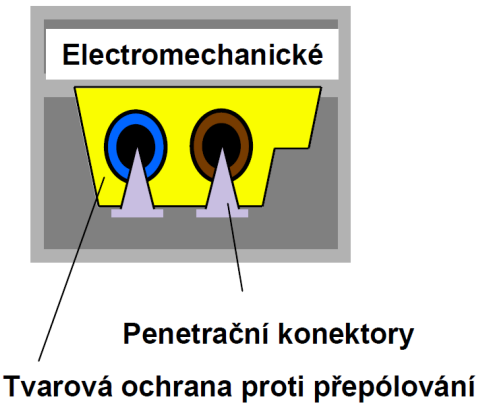
2.4.1 Modulace signálu

Při přenosu je použita modulace na principu AMP, která poskytuje dostatečné zabezpečení při použití nestíněného a nekrouceného kabelu. Tento způsob přenosu je vhodný pro sériový přenos dat v základním frekvenčním pásmu. Vysílaná bitová posloupnost je nejprve kódována kódem Manchester. Z takto získané posloupnosti je

generován proudový signál, který reaguje na každou změnu úrovně napětí v kódu Manchester. Vytvořený proudový signál se důsledkem indukčnosti transformuje na napětí. Při každém nárůstu proudu vznikne ve vedení záporný a při každém poklesu proudu kladný impuls napětí. Tím mohou stanice generovat napěťové impulzy s větší amplitudou než je úroveň napájecího zdroje. Díky tomu odpadá potřeba indukčnostních prvků v řízených stanicích a lze jejich elektroniku miniaturizovat. Přijímač napěťové signály ze sběrnice snímá a rekonstruuje ve sled bitů. Synchronizace přijímače probíhá příhcodem prvního negativního impulsu, který se interpretuje jako spouštěcí - START bit.[6]

2.4.2 Kabel sběrnice AS-Interface

Jako přenosové médium je definován nestíněný, nekroucený dvouvodičový kabel. Vyrábí se jako speciální plochý kabel AS-Interface se speciálními vlastnostmi pro montáž. Tento kabel představuje elektromechanický základ sběrnice. Byl vyvinut pro jednoduchou montáž zařízení pomocí prořezávací techniky. Plocha vodičů je v něm pevně dána, tudíž s touto montáží nevznikají žádné potíže. Díky této technice se nemusí kabel krátit. Pokud je nutné slave z něj odstranit, stačí ho oddělat a díky penetrační žluté izolaci se kabel zatáhne sám zpět. Tento profilovaný kabel lze nahradit i jiným dvouvodičovým nestíněným kabelem [7].



Obr. 2.11: Profilovaný kabel pro síť AS-I [7].

3 ADD-ON INSTUKCE

Kapitola se zabývá obecným popisem a teorií add-on instrukcí. Konkrétní informace týkající se implementace jsou uvedeny dále v práci v kapitole věnující se návrhu add-on instrukce.

3.1 Popis Add-on instrukce

Add-on instrukce slouží k zapouzdření běžně používaných funkcí a logických sekvencí. Jejich použitím uživatel vytváří v podstatě novou funkci pro běžně používanou logiku. Avšak nejsou moc vhodné pro navrhování algoritmů na vyšší hierarchické úrovni. Programy rutiny je vhodnější situovat do jiných úrovní aplikace.

Výhody použití add-on instrukcí:

- Opakovaně používaný kód - pro často používanou logickou posloupnost, sekvenci příkazů je možné umístění do add-on instrukce a vícenásobné použití je pohodlnější. Instrukci lze v jednom nebo i ve více projektech implementovat mnohonásobně.
- Srozumitelnější prostředí - umístění programové struktury do jedné zapouzdřené funkce, která má své vstupy a výstupy, což zpřehledňuje kód programu.
- Export/import - možnost add-on instrukci exportovat do jediného souboru $*.L5X$, což umožňuje vícenásobné použití a jednoduché předávní mezi jednotlivými projekty.
- Ochrana duševního vlastnictví autora - umístění vlastního kódu, na který lze použít Source Protection, což zabrání šíření algoritmu autora.
- Sledování verzí, historie změn - při tvorbě add-on instrukcí lze vytvářet dílčí revize instrukce a tím lze opět otevřít předchozí verze. Dále lze instrukci takzvaně podepsat, což vygeneruje jedinečný identifikátor a může zabránit úpravě logiky instrukce [11].

Porovnání vlastností Main rutiny, podrutiny a add-on instrukce je uvedeno v příloze 1 A.

3.1.1 Velikost add-on instrukce

Add-on instrukce může obsahovat pouze jednodušší algoritmy programu, protože již není možné ji ze samotné podstaty dále větvit. Obsahuje jednu primární logickou rutinu, která definuje její chování. Délka použitého kódu v ní omezena není (jako jiné běžné rutiny). Ovšem celkový součet vstupních, výstupních a lokálních proměnných musí být menší než 512. Neexistuje omezení pro počet vstupně-výstupních

proměnných. Omezení celkové velikosti instrukce (logika, proměnné) je 2 MB. Velikost datových typů a proměnných je zobrazeno ve spodní části okna, ve kterém se proměnné dané add-on instrukce definují [11].

3.1.2 Editace za běhu programu, ladění

Možnosti ladění závisí na verzi firmwaru. Při realizaci práce nebyla možná editace za běhu programu. Add-on instrukce mohla být modifikována pouze offline. Nešlo také měnit hodnotu proměnných při zapnutém režimu online s PLC. Ladění programu nebylo možné, protože add-on instrukce neumožňuje v run režimu PLC náhled na aktuální hodnoty proměnných, či neoznačuje prvky programu, které se nacházejí ve stavu *true* zelenou barvou tak, jako jiné druhy podprogramů [11].

3.1.3 Vnoření

Add-on instrukce mohou volat jiné add-on instrukce. To umožňuje vytváření modulární struktury a zjednodušení programu. Úroveň vnoření je možná do 7 úrovní hloubky.

3.1.4 Nepoužitelné příkazy v instrukci

Většina příkazů prostředí *RSLogix*5000, lze při programování add-on instrukce používat. Některé příkazy však použitelné nejsou. Jsou to tyto [11]:

- BRK - break
- EOT - End of Transition
- EVENT - Event Task Trigger
- FOR - cyklus For
- IOT - Immediate Output
- JSR - Jump to Subroutine
- JXR - Jump to External Routine
- MAOC - Motion Arm Output Cam
- PATT - Attach to Equipment Phase
- PCLF - Equipment Phase Clear Failure
- PCMD - Equipment Phase Command
- PDET - Detach from Equipment Phase
- POVR - Equipment Phase Override Command
- RET - Return
- SBR - Subroutine
- SFP - SFC Pause
- SFR - SFC Reset

4 POUŽITÝ HARDWARE

4.1 AS-I Master BWU 1488

Master je vybaven diagnostikou sběrnice a hlášením chyb. Uvedení do provozu, ladění a nastavení sítě může být provedeno pomocí tlačítek na přední straně masteru. Další parametry jsou uvedeny v tabulce 4.1.

Tab. 4.1: Parametry masteru BWU 1488 [**?**].

Typ nadřazeného systému	ControlLogix AS-I
Stupeň krytí	IP20
Počet AS-I sítí	2
Diagnostický interface	RS232

Obr. 4.1: MAster BWU 1488 [27].

4.2 Přípravek labyrint

Přípravek labyrint je zmenšený model. Je na něm použito několik AS-Interface slavů, ke kterým jsou připojeny digitální vstupy a výstupy. Přípravek neobsahuje žádné analogové ani bezpečnostní součásti. Ke slavům jsou připojeny snímače a zobrazovače, kterými jsou LED diody a světelné závory. Celkový popis přípravku je v citované bakalářské práci Petra Zbranka, který ho zkonstruoval [12].

4.2.1 Slave AC5213

Jedná se o aktivní ClassicLine modul, který lze připojit AS-I plochým kabelem. Obsahuje externí adresovací zdířku a čtyři zdířky pro periferie 4.2. AS-I parametry slavu jsou zobrazeny v tabulce 4.2 [13].

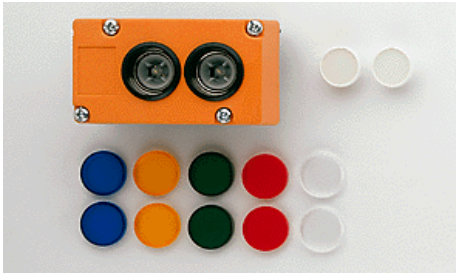
Obr. 4.2: Slave ifm AC5213 [22]

Tab. 4.2: AS-I parametry slavu AC5213

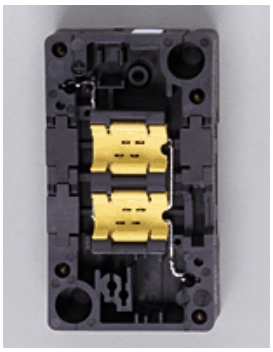
Rozšíření adres	AS-I 2.1
AS-I profil	$S-8.0.E$
I/O konfigurace	8_{HEX}
ID kód	$0.E_{HEX}$

4.2.2 Slave AC2086 a AC5000

Kombinovaný slave, který se skládá z horního dílu AC2086 4.3 a dolního dílu AC5000 4.4. Horní díl je aktivní část s vestavěnými LED indikátory. Umožňuje použití jako A i B slave a lze ho upravovat barevnými sklíčky[15]. Spodní část lze umístit na lištu či panel a umožňuje rychlý způsob montáže na AS-I plochý kabel [14].



Obr. 4.3: Slave AC2086 [24]



Obr. 4.4: Slave AC5000 [23]

Tab. 4.3: AS-I parametry slavu AC2086

Rozšíření adres	AS-I 2.1, 3.0
AS-I profil	$S-B.A.E$
I/O konfigurace	B_{HEX}
ID kód	$A.E_{HEX}$

4.2.3 Slave AC5210

Slave obsahující dvojnásobný počet přípojných bodů oproti slavu AC5213 4.5. Slave AC5210 také umožňuje montáž na AS-I kabel i rozšířenou adresaci. AS-I parametry zobrazuje tabulka 4.4 [16].

Tab. 4.4: AS-I parametry slavu AC5210[16].

Rozšíření adres	AS-I 2.1, 3.0
AS-I profil	$S-0.A.E$
I/O konfigurace	0_{HEX}
ID kód	$A.E_{HEX}$

Obr. 4.5: Slave ifm AC5210 [25]

4.2.4 Napájecí zdroj Siemens AS-i Power Supply 3A

Napájecí zdroj malých rozměrů, odolný proti přetížení a zkratu. Zdroj má integrovanou diagnostickou paměť přetížení a detekci zkratů. Svorkovnice jsou vyjímatelné [17]. Parametry zdroje jsou uvedeny v tabulce 4.5.

Tab. 4.5: Parametry zdroje Siemens AS-i Power Supply 3A [17].

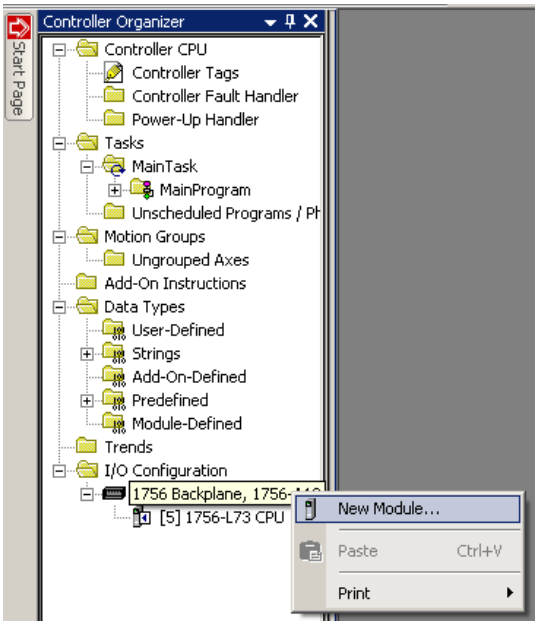
Jmenovité napětí	30 V
Proud	3A
Stupeň krytí	IP20
Svorkovnice	odnímatelná

5 NÁVRH A TVORBA ADD-ON INSTRUKCE

5.1 Vytvoření projektu a hardwarové konfigurace v RSLogix5000

Před započetím práce s PLC je potřeba v příslušném softwaru výrobce PLC vytvořit hardwarovou konfiguraci. Vypracovaný program totiž spolupracuje s hardwarem, bez kterého nemůže fungovat. Bez správné hardwarové konfigurace je snaha o vytváření programu téměř zbytečná.

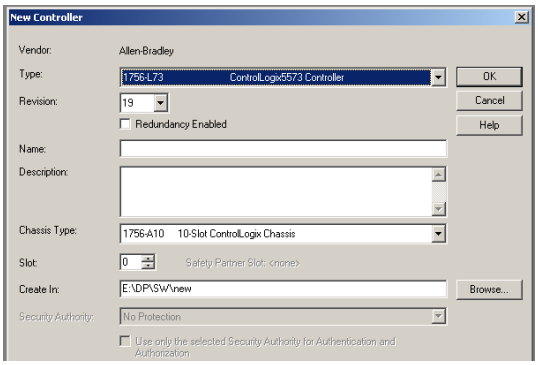
Spustíme software *RSLogix5000*. V úvodní obrazovce můžeme kliknout na Create new project. Nebo lze využít tlačítka New v horní nástrojové liště, které uživatelé znají i z jiných softwarových nástrojů. Otevře se projekt, který má v levé části *Controller Organizer*, což je hlavní rozhraní pro práci. Hardwarová konfigurace se tvoří v položce *I/O Configuration*, ve které se do lišty *Backplane* vkládají hardwarové karty. Kliknutím pravým tlačítkem na *backplane* vložíme nový modul 5.1.



Obr. 5.1: Přidání hardwarové karty

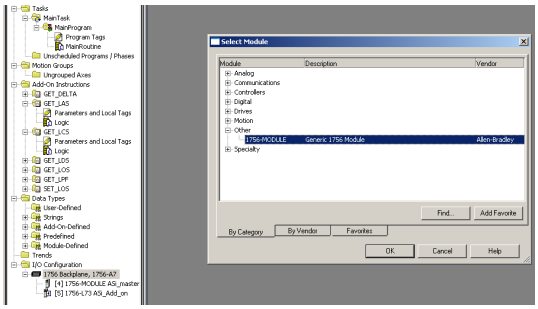
Jak bylo uvedeno v teoretické kapitole, jádrem PLC je jeho procesor. Prvně vložíme procesor (*1756-L73 CPU ControlLOgix 5573 Controller*) 5.2. V položce *Revision* musíme uvést revizi, na které procesor funguje. Tu zjistíme v programu *RSLinx*, ve kterém najdeme v síti naše PLC, zde rozklikneme lištu a pravým tlačítkem ve volbě *Properties* zjistíme požadovanou revizi. Pro náš případ je to hodnota

19. Dále musíme zadat název procesoru, respektive celého programu, který se bude zobrazovat na displeji CPU. V položce *Chasis type* vybereme sedmi slotové tělo a zadáme fyzické umístění procesoru v něm (položka *slot*). Je nutné dbát na správnou indexaci. V položce *Create in* zadáme cestu, kam chceme projekt v počítači uložit.



Obr. 5.2: Konfigurace procesoru

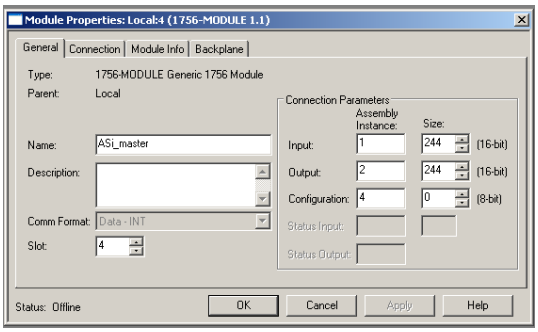
AS-I master se přidá obdobně, avšak jako typ zařízení se vybere *Other -> Generic 1756 module*5.3.



Obr. 5.3: Přidání masteru

V následujícím okně je potřeba nakonfigurovat AS-I master a to zadáním názvu a hlavně správnými *Connection Parameters*, jak je vidět na obrázku 5.4. Také je nutné přepnout *Comm format* na INT. Dále se musí zadat správný slot. Tímto krokem se do projektu přidají proměnné a datové typy pro obsluhu sběrnice. Tyto proměnné jsou globálního charakteru. Proměnné se jmenují:

- *Local:4:I* datového typu $AB : 1756\_MODULE\_INT\_400Bytes : I : 0$ - pro čtení ze sběrnice
- *Local:4:O* datového typu $AB : 1756\_MODULE\_INT\_400Bytes : O : 0$ - pro zápis na sběrnici
- *Local:4:C* datového typu $AB : 1756\_MODULE\_INT\_400Bytes : L : 0$ - popisující stavy sběrnice



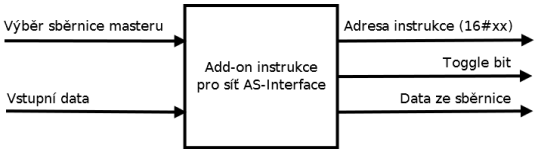
Obr. 5.4: Konfigurace masteru

Takto vytvořené hardwarová konfigurace se musí nahrát do PLC. To provedeme tlačítkem, které je zdůrazněno na obrázku 5.5. Po vyhledání správné cesty k našemu procesoru zvolíme tlačítko "Download", které nahraje HW konfiguraci do PLC. Pokud použijeme volbu "Upload", nahrajeme do počítače stávající program v PLC a vše si přemažeme.

Obr. 5.5: Nahrání HW konfigurace do PLC

5.2 Návrh add-on instrukce

Pro síť AS-Interface existuje celá řada instrukcí, které mají za úkol obsluhovat sběrnici, zajišťovat diagnostiku, nebo přenos analogových, či acyklických dat. Každá z těchto instrukcí má svou jedinečnou číselnou hodnotu, která ji reprezentuje. Tato hodnota se odešle na sběrnici, AS-I master rozpozná instrukci a provede požadované kroky. Obecným popisem lze takovouto instrukci popsat blokem 5.6.



Obr. 5.6: Vstupně/výstupní popis instrukce pro síť AS-I

Vstupy:

- K AS-I Masteru BWU 1488 lze připojit dvě sítě. Proto se pro každou instrukci musí vybrat jedna z nich (0 nebo 1).

- Některé instrukce AS-I vyžadují vstupní data, většinou to jsou adresy slavů nebo data, které se na sběrnici odesílají.

Výstupy:

- Adresa požadované instrukce, kterou vrací master pro kontrolu.
- Toggle bit - přepínací bit, který mění svou hodnotu (buď 0 nebo 1) při každém použití instrukce.
- Výstupní data ze sběrnice (některé příkazy je neobsahují)

5.2.1 Logika add-on instrukce

Logika navrhované instrukce je velice jednoduchá. Při požadavku na její spuštění se vybere výstup masteru (vybere sběrnici), přepne toggle bit, při požadavku na odeslání dat odešle data a instrukci na sběrnici. Pokud mají být výstupem instrukce data, vyčítá je ze sběrnice a následně vyčítá vrácenou adresu instrukce a toggle bit.

Při návrhu bylo dbáno na maximální přehlednost instrukce a tedy i její jednoduchost a úspornost, co se týká použitých proměnných i použitých bloků v programu. Také byl brán ohled na možnost použití jednotlivých instrukcí samostatně, proto na sebe nejsou žádným způsobem vázány.

Formát zadávaných dat na sběrnici je zobrazen v obrázku 5.7

command request								
byte	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	command							
2	T	O	circuit					
3	request parameter byte 1							
…	…							
36	request parameter byte 34							

Obr. 5.7: Interface požadavku na sběrnici a jeho struktura [11]

Bit **T** ve struktuře je toggle bit, který slouží ke kontrole provedení instrukce.

Circuit vybírá síť AS-I masteru. Pro výběr první sítě musí být **circuit**= 0, pro výběr druhé sítě musí být **circuit**= 1.

Bit **O** vybírá, jaký způsobem se přistupuje ke sběrnici. Je možné používat dva různé způsoby:

- $O = 0$ pro schéma firmy Bihl+Wiedemann, nejnižší bit 2^0 je také nejnižším bitem logické posloupnosti

- $O = 1$ pro schéma firmy Siemens, kdy je nejvyšší bit 2^7 nejnižším bitem logické posloupnosti (tedy obráceně). Sekvence bitů je inverzní.

Ve všech add-on isntrukcích je použito $O = 0$, schéma firmy Bihl+Wiedemann.

Parameter byte n je n-tý parametr příkazu. Počet těchto dat je různý pro různé instrukce. Některé instrukce data dokonce neobsahují. Maximální délka takto předávaných dat je 34 bytů.

V případě, že je příkaz příliš krátký nebo neúplný, nebo není dodržen potřebný počet parametrů, bude jeho provedení automaticky zamítnuto. Celková fyzická délka příkazu je 36 bytů.

Odpověď ze sběrnice má podobnou strukturu jako požadavek. První dva bajty jsou podobné, avšak délka dat může být rozličná oproti požadavku 5.7 :

command response								
byte	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	command (mirrored)							
2	T	result						
3	response byte 1							
…	…							
36	response byte 34							

Obr. 5.8: Interface odpovědi ze sběrnice a jeho struktura [11]

Command (mirrored) je hodnota právě zadaného příkazu na sběrnici, která je zrcadlena zpět.

Bit **T** je výstup toggle bitu, kterým lze kontrolovat, zda proběhla odezva ze sběrnice.

Result je potvrzení příkazu v sedmi nejnižších bitech 2. bytu. Tabulka ukazuje, co jednotlivá potvrzení ze sběrnice znamenají 5.1.

Tab. 5.1: Hodnoty kódu result

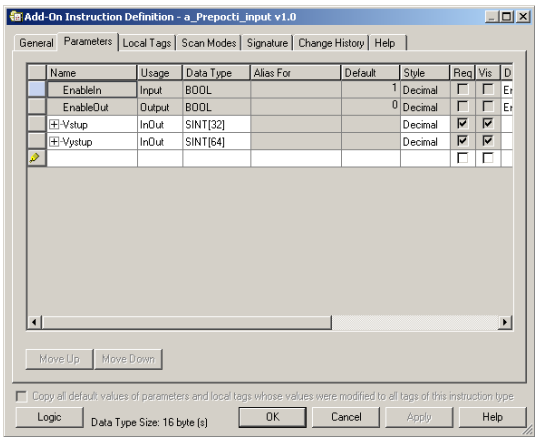
Název	Hodnota	Popis
OK	00_{16}	provedeno bez chyby
HI_NG	11_{16}	obecná chyba
HI_OPCODE	12_{16}	špatná hodnota v příkazu
HI_LENGTH	13_{16}	délka příkazu je příliš krátká
HI_ACCESS	14_{16}	nejsou přístupová práva
EC_NG	21_{16}	obecná chyba
EC_SND	22_{16}	slave (zdrojová adresa) není detekován
*EC_SD*0	23_{16}	slave 0 detekován
*EC_SD*2	24_{16}	slave (cílová adresa) není detekován
EC_DE	25_{16}	delete error
EC_SE	26_{16}	set error
EC_AT	27_{16}	dočasná adresa
EC_ET	28_{16}	dočasné xID1
EC_RE	29_{16}	chyba čtení xID1

5.2.2 Programování

Add-on instrukci lze programovat klasickými jazyky používanými při programování PLC (LAD, STL, FB). Při realizaci diplomové práce byl používán jazyk Ladder diagram.

V programu ConrolLogix 5000 v panelu *Controller Organizer* 5.1 lze najít položku *Add-on Instructions* a pravým tlačítkem myši lze vložit *New Add-on Instruction*. Vyplní se název instrukce podobně jako v hardwarové konfiguraci, dále verze revize, případně popis instrukce. Po potvrzení tohoto okna lze již s instrukcí pracovat, měnit její vstupně/výstupní parametry na záložce *Parameters* nebo vytvářet lokální proměnné na záložce *Local Tags*. Další záložky obsahují sledování historie, možnost podepsat instrukci, zabezpečit ji před čtením nebo vytvořit k instrukci nápovědu 5.9.

Add-on instrukce obsahuje dvě základní části - *Logic*, ve které se tvoří program a *Parameters and Local Tags*, což je seznam proměnných, stejně jako jiné rutiny. Následná tvorba logiky programu je stejná jako při programování běžné rutiny. Speciální je práce s proměnnými, které jsou popsány v následující části.



Obr. 5.9: Vstupně/výstupní popis instrukce pro síť AS-I

5.2.3 Proměnné a datové typy

Proměnné lze zadávat klasicky v nabídce *Program Tags* nebo po opětovném zvolením Add-on instrukce v záložce *Parameters* a *Local Tags*. Tvoří se stejně jako v běžném programu, ale přibilo zde několik možností:

- **Usage** - použití proměnné ve vztahu k instrukci. Musí být nastavena jedna z voleb:
  - In - proměnná je vstupní parametr
  - Out - proměnná je výstupní parametr
  - InOut - proměnná je vstupní i výstupní parametr, může předávat pole hodnot
  - Local - proměnná je pouze lokální, nejsou viditelné mimo instrukci. Jedinou možností přístupu mimo instrukci je Alias. Jako jediná může být vytvořena jako pole.
- **Visibility** - viditelnost proměnné
- **External access** - definuje úroveň přístupu pro externí zařízení.
  - Input - Read/Write, Read Only, None
  - Output - Read/Write, Read Only, None
  - Local - Read/Write, Read Only, None
  - EnableIn - Read Only
  - EnableOut - Read Only
  - InOut - N/A
- **Constant** - zadání konstanty (za pomocí volby *Default*)

EnableIn / EnableOut

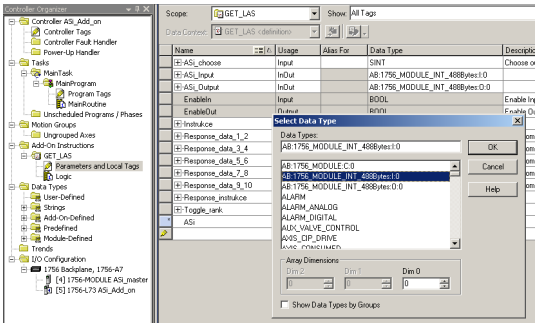
Tyto dva parametry se objevují ve výchozím nastavení instrukce a jsou s ní pevně svázány. Pokud chceme provádět logiku vytvořené instrukce, proměnná EnableIn musí být v 1. Obecně platí, že na EnableIn se v logice instrukce nelze odkazovat. Dalo by se říci, že tento parametr je vnitřním spuštěním instrukce.

Parametr EnableOut je ve výchozím nastavení závislý na EnableIn a udává informaci o proběhnutí instrukce. Avšak může být změněn uživatelem v logice instrukce.

Pokud je parametr EnableIn v 0 - false, není vykonávána logika instrukce a parametr EnableOut je také 0.

Datové typy

Datové typy jsou stejné jako v běžném programu. Jak je již uvedeno výše, vstupní a výstupní proměnné nemohou být pole, ty mohou být pouze lokální. Tato skutečnost vyplývá i z podstaty add-on instrukce, která nemá být velkým podprogramem, ale má řešit dílčí funkce. Kromě klasických datových typů lze samozřejmě používat datové typy modulů připojených k PLC (v tomto případě práce s AS Interface viz 5.10).



Obr. 5.10: Proměnná ze sběrnice AS Interface

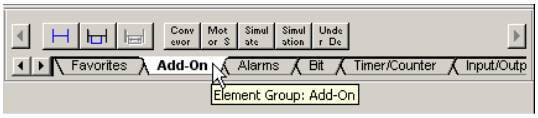
Nejčastěji používané datové typy (v návaznosti na řešení práce):

- $BOOL$ - Q bitový datový typ nabývající hodnot 0 a 1
- $SINT$ - 8 bitový celočíselný datový typ
- INT - 16 bitový datový typ
- $DINT$ - 32 bitový datový typ
- AB : 1756_$MODULE$_INT_400$Bytes$: I : 0 - datové pole sběrnice AS Interface - vstup do PLC, tedy výstup/stavy sběrnice
- AB : 1756_$MODULE$_INT_400$Bytes$: O : 0 - datové pole sběrnice AS Interface - vstup na sběrnici, tedy výstup z programu, PLC.

Nevýhodou alokování proměnných je, že i přesto, že když je vytvořena proměnná datového typu SINT nebo BOOL (8 bitů nebo jen 1 bit), v paměti PLC bude zabírat plných 32 bitů. Při použití Add-On instrukce si procesor kratší datové typy převádí na 32 bitové, tedy používání přímo datových typů DINT A REAL (které jsou 32 bitové) urychluje běh celé instrukce a šetří paměť.

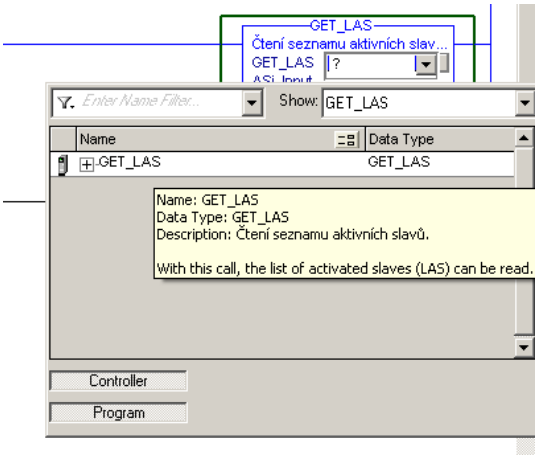
5.3 Použití add-on isntrukce

Pokud je add-on instrukce naprogramována, může se použít v programu. Najdeme ji mezi ostatními funkcemi softwaru v záložce Add-on5.11.



Obr. 5.11: Vložení instrukce do programu.

Aby mohla instrukce fungovat, musíme vytvořit řídicí proměnnou datového typu dané instrukce a přiřadit ji k ní 5.12. Dále je potřeba vložit do instrukce proměnné z AS-I sítě a dále přiřadit vstupně-výstupním proměnným příslušné hodnoty. Instrukce se řídí logickým sledem programu, ve kterém je použita. Tedy pokud je na příčce programu před jejím vstupem úroveň *true*, je instrukce vykonávána.

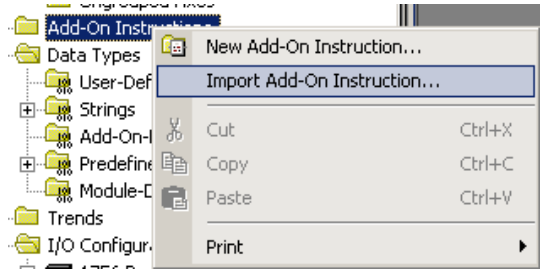


Obr. 5.12: Přiřazení řídicí proměnné.

5.4 Export / Import add-on instrukce

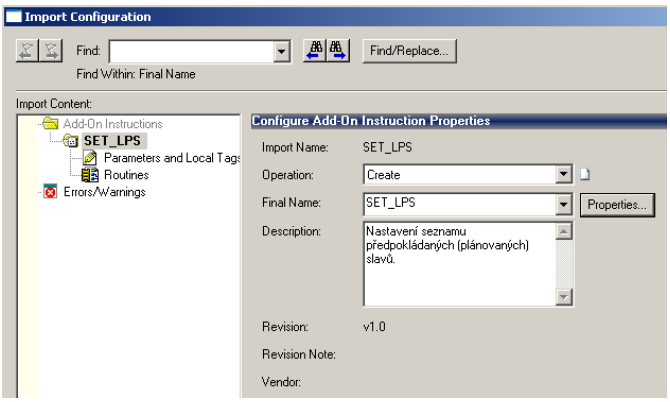
Pro možnost použití instrukce v jiných projektech lze využít jejího exportu. Ten se provede výběrem pravým tlačítkem myši na instrukci a následným uložením.

Vyexportuje se jediný soubor s příponou $*.L5X$. V novém projektu pak místo tvorby nové instrukce zvolíme import 5.13 a vybereme příslušnou instrukci.



Obr. 5.13: Importování add-on instrukce.

Po volbě importování instrukce se objeví dialogové okno, které nastavuje možnosti importu 5.14.



Obr. 5.14: Konfigurace importu.

5.5 Ukázková add-on instrukce

Jako ukázková add-on instrukce byla vybrána instrukce *READ_CDI*, která čte parametry slavu - ID, I/O, xID1 a xID2. Vstupní data instrukce je adresa slavu, ze kterého chceme parametry číst. Výstupní data jsou čtyři parametry.

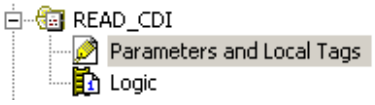
5.5.1 Proměnné

Instrukce *READ_CDI* používá tyto proměnné:

- **ASi_choose** (Input) - výběr sítě masteru (SINT)
- **ASi_Input** (InOut) - čtení ze sběrnice (AB:1756_MODULE_INT_488Bytes:I:0)
- **ASi_Output** (InOut) - zápis na sběrnici (AB:1756_MODULE_INT_488Bytes:O:0)

- **Bit_B** (Input) - bit, který říká zda je adresa požadovaného slavu v normální nebo rozšířené adresaci (BOOL)
- **EnableIn** (Input) - vnitřní proměnná add-on instrukce (BOOL)
- **EnableOut** (Output) - vnitřní proměnná add-on instrukce (BOOL)
- **Instruction** (Local) - vnitřní proměnná add-on instrukce, která obsahuje adresu instrukce, výběr sítě masteru a její 7. bit se používá jako toggle bit (SINT[2])
- **Instruction_Request** (Local) - vnitřní proměnná add-on instrukce, do které se na správné místo uloží adresa příslušného slavu s B bitem a odešlě se ve správném formátu na sběrnici (SINT[2])
- **Master_Response** (Local) - pomocná vnitřní proměnná, která slouží k uložení promaskované hodnotě ze sběrnice a k její další distribuci (INT)
- **Response_data** (InOut) - pole výstupních dat, každá položka pole obsahuje jeden parametr (ID, I/O, xID1, xID2). Jejich pořadí je popsáno v komentáři jednotlivých položek pole (SINT[4])
- **Response_instruction** (Output) - vrácená adresa instrukce ze sběrnice, přepnutý toggle bit a případné potvrzení-chybová hláška (INT)
- **Slave_address** (Input) - vstupní proměnná, která udává adresu slavu, jehož parametry se budou číst
- **Toggle_rank** (Local) - pomocná proměnná pro přepínání příček programu (INT)

Tyto proměnné lze zobrazit buď v okně *Parameters and Local Tags* - viz obrázek 5.15. Proměnné lze zobrazit také poklepáním na hlavičku add-on instrukce. V jednotlivých kartách *Parameters* a *Local Tags* si je zobrazit zvlášť pro vstupně-výstupní proměnné a pro lokální proměnné. Náhledy těchto karet jsou uvedeny v příloze C.1 a C.2.



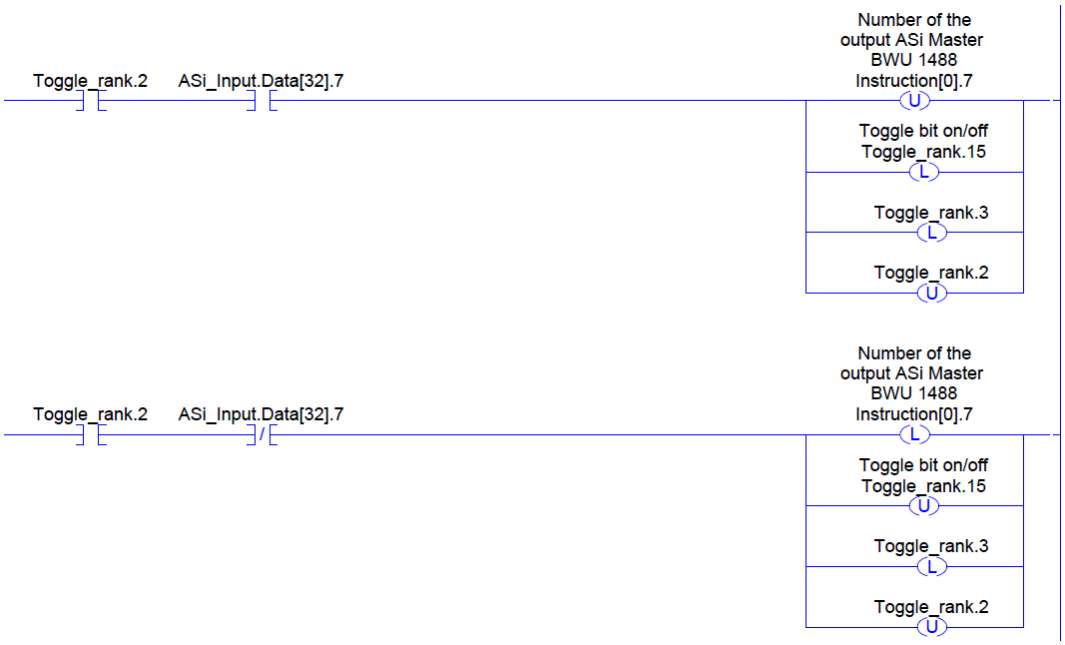
Obr. 5.15: Zobrazení proměnných add-on isntrukce

5.5.2 Logika

Prvním krokem, který add-on instrukce provede je nastavení její adresy. To je provedeno na 1. příčce logiky programu uložením hodnoty 28_{HEX} do proměnné *Instruction[1]*. Na druhé příčce je do nulového prvku této proměnné uložené nastavení sítě masteru. Tím je ukončena inicializační část instrukce. Ladder diagram pro tyto

příčky je uveden v příloze C.3. Opačné pořadí prvků v proměnné *Instruction* je zvoleno proto, že proměnná sběrnice má rozměr INT, ale vyžaduje prvně zapsání adresy a až potom sítě masteru. Tedy hexadecimální číslo zapsané do proměnné sběrnice musí vypadat takto: 16#2801. To je zajištěno opačným pořadím prvků v proměnné (na rozdíl od zobrazení interfacu instrukce 5.7).

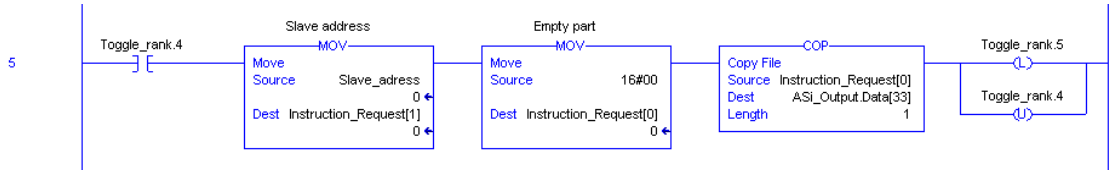
Další částí programu jsou dvě příčky, které zajišťují přepínání toggle bitu. To se děje pro kontrolu, zda byla instrukce vykonána. Část tohoto programu je na obrázku5.16.



Obr. 5.16: Přepínání toggle bitu

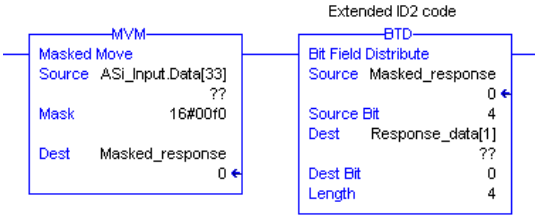
Následuje nakopírování proměnné Instruction na proměnnou sběrnice AS-Interface, nastavení Bitu B do proměnné *Slave_address* a promaskování proměnné. To zajišťuje, že bude adresa ve správném rozsahu C.4.

Po promaskování adresy, se stejným postupem jako se skládala instrukce, musí správně poskládat i adresa. To je zajištěno malým SINTovým polem o dvou prvcích, kdy v prvku s indexem 0 jsou nuly a v prvku s indexem 1 je adresa slavu. Takto poskládaný požadavek je zkopírován na příslušnou proměnnou sběrnice 5.17.



Obr. 5.17: Správně odeslaná adresa slavu na sběrnici.

Tímto skončil proces zápisu dat na sběrnici a lze přistoupit k vyčítání dat ze sběrnice. Instrukce a data se zapisovaly do proměnné *ASi_Output.data*. Čtení ze sběrnice probíhá z proměnné *ASi_Input.data*. Nejprve je ze sběrnice přečtena kontrolní adresa instrukce, toggle bit a odpověď/potvrzení. To je uloženo do proměnné Response_instruction, která má 16 bitů. Parametry ze sběrnice jsou uloženy v proměnné *ASi_Input.data* za touto odpovědí, tedy na indexech 33 a 34. Vyčítání probíhá promaskovaným přesunem do pomocné proměnné *Masked_response*, ze které se funkcí *Bit distributed* - bitovým přesunem nakopíruje do příslušného prvku výstupního pole *Response_data* 5.18. Celá část tohoto programu, kdy se čtou data ze sběrnice je zobrazena v příloze C.5. Na závěr celé logiky se nuluje nultý bit proměnné *Toggle_rank*, aby mohla být instrukce spuštěna znovu.



Obr. 5.18: Uložení promaskovaného parametru na příslušný prvek pole.

5.6 Seznam realizovaných add-on instrukcí

Podrobný seznam realizovaných instrukcí s obrázky interfacu na sběrnici je v příloze D.

5.6.1 Přenos 16 bitových dat

- **Read** 1 16−**bit Slave in.Data** (RD_7X_IN)
  Čtení čtyř 16-ti bitových kanálů na vstupu jednoho modulu podle profilů ($S-7.3$, $S-7.4$, $S-7.5$, $S-7.A.8$, $S.A.9$, $S-7.A.A$).
- **Write** 1 16−**bit Slave out.Data** (WR_7X_OUT)
  Zápis čtyř 16-ti bitových kanálů na výstup jednoho modulu v závislosti na profilu ($S-7.3, S-7.4, S-7.5, S-7.A.8, S.A.9, S-7.A.A$).
- **Read** 1 16−**bit Slave out.Data** (RD_7X_OUT)
  Čtení čtyř 16-ti bitových kanálů na výstupu jednoho slavu dle profilů ($S-7.3, S-7.4, S-7.5, S-7.A.8, S.A.9, S-7.A.A$).

- **Read** 4 16−**bit Slave in.Data** ($RD\_7X\_IN\_X$)
  Čtení čtyř 16-ti bitových kanálů na vstupech čtyř slavů, jejichž adresy jsou bezprostředně za sebou podle profilů ($S-7.3, S-7.4, S-7.5, S-7.A.8, S.A.9, S-7.A.A$).
- **Write** 4 7.3 **Slave out.Data**($WR\_7X\_OUT\_X$)
  Zápis čtyř 16-ti bitových kanálů na výstupy čtyř po sobě jdoucích slavů dle profilů ($S-7.3, S-7.4, S-7.5, S-7.A.8, S.A.9, S-7.A.A$).
- **Read** 4 7.3 **Slave out.Data** ($RD\_7X\_OUT\_X$)
  Čtení čtyř 16-ti bitových kanálů na výstupech čtyř slavů jejichž adresy jsou bezprostředně za sebou podle profilů ($S-7.3, S-7.4, S-7.5, S-7.A.8, S.A.9, S-7.A.A$).
- **Read** 16 **channels** 16−**bit Slave in.Data** ($OP\_RD\_16BIT\_IN\_CX$)
  Čtení 16-ti bitové informace vstupních dat na 16 kanálech s možností zadání počtu kanálu na jeden slave (maximum 16) dle profilu ($S-7.3, S-7.4, S-7.5, S-7.A.8, S.A.9, S-7.A.A$).
- **Write** 16 **channels** 16−**bit slave out.Data** ($OP\_WR\_16BIT\_OUT\_CX$)
  Zápis 16-ti bitové informace výstupních dat na 16 kanálů dle profilu ($S-7.3, S-7.4, S-7.5, S-7.A.8, S.A.9, S-7.A.A$) [10].

5.6.2 Příkazy pro profily $S-7.4/S-7.5$

- $WR\_74\_75\_PARAM$
  Zápis řetězce do slavu dle profilu $S-7.4$, případně jeho přenos dle profilu $S-7.5$. Ve slavu profilu $S-7.5$ musejí být data zapsána do bufferu stejnou formou jako na sběrnici AS-interface.
- $RD\_74\_75\_PARAM$
  Čtení řetězce dle profilu $S-7.4$ (nebo také podle profilu $S-7.5$).
- $RD\_74\_75\_ID$
  Čtení ID řetězce slavu dle profilu $S-7.4$ nebo 16 bitového slavu konfigurovaného dle profilu $S-7.5$. Pokud je řetězec delší než umožňuje rozhraní, je zapsán do vyrovnávací paměti, jejíž obsah může být čten po částech od indexu i. První bajt opět nese informaci o délce řetězce.
- RD_74_DIAG
  Čtení diagnostického řetězce slavu dle profilu $S-7.4$. Delší řetězec je uložen do bufferu. Obsah vyrovnávací paměti lze číst po částech od indexu i. První bajt určuje délku řetězce. Pokud je $i=0$ je řetězec právě čten ze slavu. Data mohou být čtena souvisle [10].

5.6.3 Acyklické příkazy

- $WRITE\_ACYCLIC\_TRANS$
  Funkce umožňuje různé acyklické přenosy pro slavy s profily $S-7.4$, $S-7.5$ a $Safetymonitory$. Přenos se provádí na pozadí a jeho výsledek může být čten pouze příkazem $READ\_ACYCLIC\_TRANS$. Tato funkce má být náhradou za funkce pro profily $S-7.4$ a $S-7.5$ - $WR\_74\_75\_PARAM$, $RD\_74\_75\_PARAM$, $RD\_74\_75\_ID$, RD_74_DIAG a $SafetyatWork$. Délka přenášených dat může být větší než velikost rozhraní. Data se nejprve zapíší do vyrovnávací paměti, dokud se přenos nespustí. n je délka sub-řetězce, který je zapsán do vyrovnávací paměti počínaje indexem i. Pokud $i=0$, spustí se přenos.

- $READ\_ACYCLIC\_TRANS$
  Čtení odpovědi na příkaz $WRITE\_ACYCLIC\_TRANS$ [10].

5.6.4 Diagnostika sítě AS-interface

- **Get List and Flags** (GET_LIST)
  Čtení diagnostických údajů sítě:
  seznam aktivních slavů = LAS
  seznam detekovaných slavů = LDS
  seznam naprojektovaných slavů = LPS
  příznaky (flagy) dle AS-interface specifikace
- **Get Flags** (GET_FLAGS)
  Čtení flagů (identifikátorů, příznaků).
  Pok = flag je nastaven (je roven 1), pokud slave nesignalizuje periferní chybu
  $S0$ = flag nastaven, když existuje slave s adresou 0
  AAs = flag nastaven, pokud může být vykonávána automatická adresace (tzn. slavy nejsou nekorektně připojeny)
  AAv = flag nastaven, když může být vykonávána automatická adresace a právě jeden slave je mimo provoz
  CA = příznak nastaven v konfiguračním režimu a obnoven v chráněném režimu
  NA = příznak nastaven, pokud je master v normálním stavu
  APF = příznak nastaven při nízké úrovni napětí na kabelu AS-interface
  Cok = příznak nastaven, pokud odpovídá požadovaná a aktuální konfigurace

AAe = flag indikuje, zda je automatická adresace povolena (= 1) nebo zakázána (= 0) uživatelem

OL = flag nastaven, pokud by má nastat změna do offline režimu nebo pokud je tento režim brzy dosažitelný

DX = když je flag nastaven, výměna dat mezi masterem a slavem je právě ve fázi výměny dat. Pokud bit není nastaven, výměna dat není k dispozici. Přečtené ID zprávy se přenášejí na slave. Bit je nastaven, pokud master vstoupí do offline režimu.

- **Get Delta List** (GET_DELTA)
  Delta list obsahuje seznam adres slavů s konfiguračními chybami.
- **Get list of corrupted Slaves** (GET_LCS)
  Seznam poškozených slavů.
- **Get list of activated Slaves** (GET_LAS)
  Získá seznam aktivních slavů
- **Get list of detected Slaves** (GET_LDS)
  Čtení seznamu detekovaných slavů
- **Get list of peripheral Faults** (GET_LPF)
  Čtení seznamu periferních poruch (LPF). Tento seznam je aktualizován cyklicky masterem. Pokud slave signalizuje chybu připojených periferií (např. přerušený drát) mohou být nalezeny v popisu slavu.
- **Get list of offline Slaves** (GET_LOS)
  Seznam slavů v offline režimu, když se vyskytuje konfigurační chyba. Uživatel může zvolit reakci na konfigurační chybu. Master může AS-interface při konfigurační chybě u důležitého slavu vypnout. Méně důležité slavy mohou hostiteli odeslat chybu (síť v tomto případě nebude v režimu offline).
- **Set list of offline Slaves** (SET_LOS)
  Příkazem je definován seznam slavů v režimu offline (u kterých se vyskytla konfigurační chyba).
- **Get transm.err.counters** (GET_TECA)
  Čtení čítače chyb všech slavů s číselnou adresou nebo s A adresou. Čtením této informace se pokaždé čítač chyb restartuje. Maximální hodnota čítače chyb je 254. Hodnota 255 způsobí, že čítač přeteče. Za účelem získání skutečného počtu transkripčních (počet špatných telegramů s daným modulem slave) chyb je potřeba číslo vynásobit hodnotou 2.
- **Get transm.err.counters** (GET_TECB)
  Čtení čítače chyb B - slavů. Ostatní chování je stejné, jako u příkazu GET_TECA.
- **Get transm.err.counters** (GET_TEC_X)
  Čtení chyb od zadané adresy určitého slavu do zadaného počtu adres (počítadlo adres se inkrementuje dle zadané hodnoty). Ostatní chování je stejné, jako

u příkazu GET_TECA [10].

5.6.5 Konfigurace AS-I masteru

- **Set Operation mode (**SET_OP_MODE**)**
  Příkaz přepíná mezi Configuration mode (projektový módem) a Protected mode (chráněným režimem). AS-I slave může být zapsán do Seznamu naprojektovaných slavů (List projected slaves - LPS) jen tehdy, pokud je aktivní Protected mode, čímž jsou jejich předpokládané a aktuální konfigurace aktivovány. Jinak řečeno, slave je aktivován, pokud detekovaná konfigurace I/O a ID jsou shodné s již nakonfigurovanými hodnotami.
  V projektovém režimu jsou všechny detekované slavy (kromě slavu "0") aktivovány. To platí i pro slavy, které mají rozdílnou hodnotu I/O a ID s aktuální konfigurací.
  Bit Operation Mode je trvale uložen a je zachován i po restartu. Při změně z Configuration mode na Protected mode provádí master warm restart (změna na offline následovaná změnou na on-line mode). Pokud je na sběrnici slave s adresou 0 a je zapsán v seznamu detekovaných slavů LDS, master nemůže provést změnu z Configuration mode na Protected mode.
  0 - protected mode
  1 - configuration mode
- **Store actual configuration** ($STORE\_CDI$)
  Ulož aktuální konfiguraci - při této výzvě jsou všechna konfigurační (skutečná) data (I/O konfigurace, ID kód, rozšířený ID1 a ID2 kód) všech slavů na sběrnici trvale uložena v EEPROM jako (očekávaná) konfigurační data. Seznam aktivních slavů (LAS) je přidán k seznamu stálých slavů (LPS).
- **Read actual configuration** ($READ\_CDI$)
  Čtení aktuální konfigurace - master získá (přečte) konfigurační data 1 slavu (I/O konfigurace, ID kód, ID1, ID2) jehož adresa je zadána jako vstupní parametr příkazu. Konfigurační data jsou specifikována výrobcem slavu. Přepnutí mezi typy adresací provádí tzv. B bit ($B = 0$ pro jednoduchou adresaci a A-slavy, $B = 1$ pro B-slavy).
- **Set permanent configuration** (SET_PCD)
  Nastavení trvalé konfigurace - funkce nastaví následující konfiguraci do slavu (I/O konfigurace, ID kód, ID1, ID2).
- **Get extended permanent configuration** (GET_PCD)
  Získání rozšířené stálé konfigurace - příkaz čte konfigurační údaje slavu uloženou v EEPROM paměti masteru (I/O konfigurace, ID kód, rozšířený ID1,2 kód).

- **Set list of projected slaves** (SET_LPS)
  Nastavení seznamu předpokládaných (plánovaných) slavů - tímto příkazem je seznam nakonfigurovaných slavů předán k trvalému uložení do paměti EEPROM masteru.

- **Get list of projected slaves** (GET_LPS)
  Získání seznamu předpokládaných (plánovaných) slavů - tímto příkazem je seznam předpokládaných slavů přečten z masteru.
- **Store actual parameters** ($STORE\_PI$)
  Uložení aktuálních parametrů - nakonfigurované parametry uložené v paměti EEPROM masteru jsou přepsány současnými, aktuálními parametry (I/O, ID, xID1, xID2) a trvale uloženy.
- **Write parameter** ($WRITE\_P$)
  Zapiš parametr P - hodnota parametru je předána na slave, jehož adresa je vstupem funkce.
- **Read parameter** ($READ\_PI$)
  Čtení parametru PI - vrací aktuální (skutečnou) hodnotu parametru ve slavu zaslané masterem. Tato hodnota nesmí být zaměňována s parametrem echo příkazu ($WRITE\_P$). Tento příkaz nelze použít pro přímé čtení parametru ze slavu.
- **Set permanent parameter** (SET_PP)
  Nastav trvalý parametr - nastaví hodnotu trvalého parametru zadanému slavu. Hodnota je trvale uložena v paměti EEPROM gateway. Parametr se přenáší pouze v případě, že je slave aktivní po zapnutí napájení masteru.
- **Get permanent parameter** (GET_PP)
  Získej trvalý parametr - master čte specifický parametr slavu uloženou v paměti EEPROM masteru.
- **Set auto address enable** (SET_AAE)
  Nastavení automatické adresace - tato volba může povolit nebo zakázat Automatické programování adres. Bit $AUTO_ADDR_ENABLE$ je uložen trvale, je zachován i po restartu masteru.
- **Change slave address (**$SLAVE\_ADDR$**)**
  Změna adresy slavu - používá se hlavně k přidání nového slavu s výchozí adresou 0 na sběrnici. Jako parametry příkazu se předává stará adresa slavu (ve většině případů adresa 0) a nová adresa (s B bitem, který určuje změnu adresy z A-salvu na B-slave).
- **Write AS-i slave extended ID1** ($WRITE\_XID1$)
  Zápis rozšířeného ID1 kódu [10].

5.6.6 Ostatní příkazy

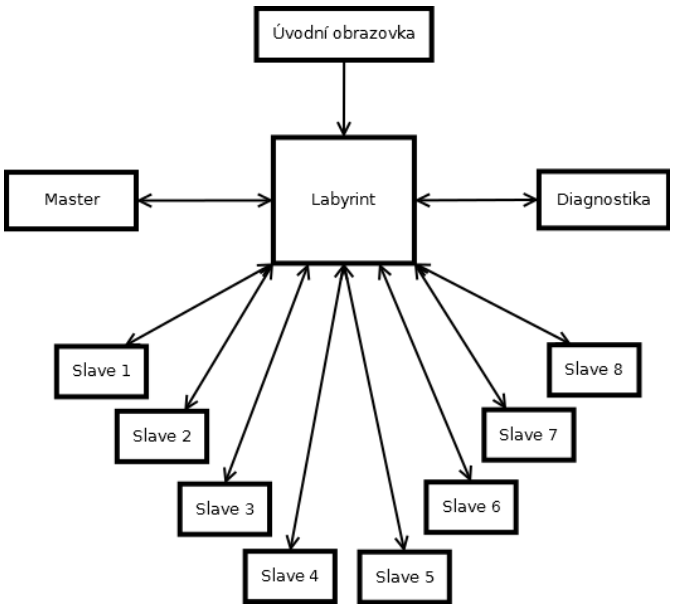
- $IDLE$
  Přepnutí do klidového stavu.
- **Read input data image** ($READ\_IDI$)
  Čtení vstupních dat všech slavů na sběrnici mimo cyklickou výměnu dat. Tento příkaz přenáší i hodnoty kontrolních flagů.
- **Write output data image** ($WRITE\_ODI$)
  Zápis výstupních dat všech slavů mimo cyklickou výměnu dat.
- **Read output data image** ($READ\_ODI$)
  Čtení výstupních dat všech slavů mimo cyklickou výměnu dat.
- **Set offline mode** ($SET\_OFFLINE$)
  Tento příkaz přepíná mezi online a offline režimem. Online je normální provozní stav pro master.
- **Release data exchange** (SET_DATA_EX)
  Bitem $Data\_Exchange\_Active$ se povoluje výměna dat na sběrnici.
- **BUTTONS** ($BUTTONS$)
  Příkazem se aktivuje/deaktivuje používání tlačítek [10].

6 VIZUALIZACE

6.1 Grafická část

Vizualizace byla vytvořena v prostředí Factory Talk View Studio. Jejím úkolem je zobrazování a ovládání základních funkčních vlastností vybraných instrukcí na sběrnici v návaznosti na model labyrintu. Vybrané instrukce jsou z převážné části diagnostické. Dále instrukce ovládající master a čtení parametrů jednotlivých slavů. Konkrétně se jedná o tyto instrukce:

- *GET_DELTA*
- *GET_FLAGS*
- *GET_LAS*
- *GET_LCS*
- *GET_LDS*
- *GET_LPF*
- *GET_LOS*
- *SET_LOS*
- *GET_LPS*
- *SET_LPS*
- *SET_OP_MODE*
- *SET_AAE*
- *READ_CDI*



Obr. 6.1: Struktura obrazovek vizualizace

Vizualizace se skládá z několika obrazovek (ve Factory Talk takzvaný *Displays*), které na sebe navazují. Struktura těchto obrazovek, a jejich návaznost na sebe v celé vizualizaci, je znázorněna na obrázku 6.1. Přičemž každá z obrazovek obsahuje tlačítko pro ukončení celé vizualizace. Pokud chce uživatel provést nějakou změnu, musí ji vždy potvrdit tlačítkem *Proveď*, jinak se změna/příkaz neprovede. Každá z obrazovek, na které lze vykonávat nějaké funkce (obrazovky všech slavů, masteru a diagnostiky) obsahují indikátor toggle bitu, který zastává kontrolu vykonání funkce na sběrnici. Indikátor po potvrzení instrukce problikne, což znamená, že se na sběrnici změnil kontrolní toggle bit, který právě program přečetl a vše proběhlo správně.

Úvodní obrazovka E.1 popisuje název práce a slouží jako informační panel. Jádrem celé vizualizace je obrazovka labyrintu E.2, ze které se přistupuje k jednotlivým funkčním prvkům a na kterou se vždy uživatel vrací.

Přejít k obslužným funkcím masteru lze kliknutím na něj. Otevře se okno E.4, ve kterém je možnost měnit pomocí funkce SET_OP_MODE režim masteru z projektového na chráněný a naopak. Mód, ve kterém se master nachází je po jeho použití zobrazen na přepínacím tlačítku. Lze také zapínat a vypínat automatickou adresaci pomocí funkce SET_AAE. Veškeré přepnutí se musejí potvrzovat příslušným tlačítkem.

Stejně jako se přistupuje k obrazovce masteru, tak se přistupuje i k obrazovkám jednotlivým slavům E.5 - kliknutím na příslušný slave. Každý slave má vlastní obrazovku. Otevře se okno, ve kterém lze zjistit parametry příslušného slavu - ID, I/O, xID1 a xID2.

K přístupu na obrazovku diagnostiky E.3 slouží tlačítko k tomu určené. Na této obrazovce lze vybrat ze seznamu diagnostických instrukcí. V seznamu se lze pohybovat šipkami a potvrzení výběru instrukce proběhne klávesou Enter. Aby se vybraná a potvrzená instrukce vykonala musí se opět vybráno tlačítko *Proveď*. Kontrolu vykonání zastupuje toggle bit.

Při získání seznamu flagů se seznam zobrazí v levé části obrazovky a zobrazí se pouze flagy, které jsou v tento okamžik v logické jedničce. Flagy v logické nule nejsou zobrazeny vůbec. Při použití ostatních funkcí, které čtou data ze sběrnice (obecně GET_*) se jejich výstup zobrazí ve spodní části okna. Logická jednička se zobrazí obrázkem slavu na příslušné adrese a logická nula přeškrtnutým obrázkem. Při použití funkce, které zapisují požadavek na sběrnici (SET_*) lze jednotlivé slavy vybrat myší a tím je nastavit do příslušného stavu. Příkladem může být nastavení seznamu offline slavů tak, že jsou označeny myší. Výběr proběhne tak, že slavy online mají obrázek slavu a slavy offline přeškrtnutý znak. Čtením seznamu offline slavů se lze přesvědčit o správném nastavení slavů. Pro nastavení seznamu projektovaných slavů (SET_LPS) musí být master přepnutý do projektového módu.

Obr. 6.2: Bit na adrese v 1

Obr. 6.3: Bit na adrese v 0

Jednotlivé obrazovky vizualizace jsou uvedeny v příloze E.

6.2 Program pro vizualizaci

Samotné add-on instrukce nelze použít bez programu, ve kterém by byly volány, proto bylo nutné vytvořit program, který bude vizualizace ovládat.

Program obsahuje mnoho proměnných, avšak nejdůležitější pro řízení programu jsou tyto6.1:

Tab. 6.1: Řídicí a důležité proměnné vizualizace

Název proměnné	Datový typ	Popis
Toggle_rank	DINT	Přepínání příčet diagnostických instrukcí
Making	BOOL	Spuštění diagnostické add-on instrukce
ASi_choose	SINT	Výběr sítě masteru
Aux_next_rung	SINT	Pomocná proměnná, přepnutí na další příčku
Master_*	BOOL	Řídicí proměnné instrukcí pro master
Slave_*	BOOL	Řídic proměnné instrukcí pro jednotlivé slavy
Toggle_bit	BOOL	Toggle bit

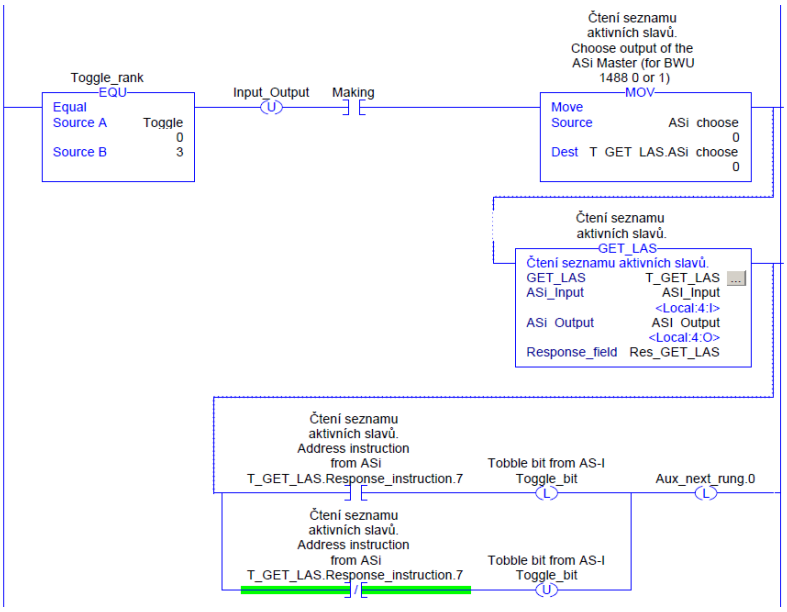
Instrukce pro sběrnici AS-I v programu se dají rozdělit do tří skupin:

1. pro master
2. pro slavy
3. diagnostické instrukce

Obecně lze chování programu popsat vývojovým diagramem uvedeným v příloze F.1. Pro každou skupinu výše zmiňovaných instrukcí se liší rozhodovací logika.

6.2.1 Diagnostické instrukce

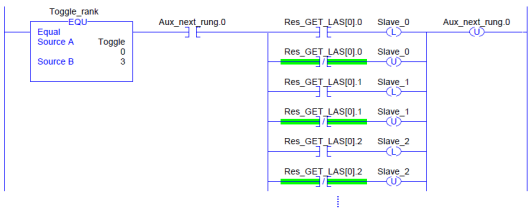
Pro ukázku bude sloužit instrukce *GET_LAS* - čtení seznamu aktivních slavů. Na obrázku je vidět způsob vybrání této příčky. Ve vizualizaci E.2 se vybere patřičná instrukce. Protože prvek seznamu ve vizualizaci neumí k proměnné *Toggle_rank* přistupovat binárně, uloží do něho příslušnou dekadickou hodnotu. Ta se porovnává na začátku příčky instrukce funkcí *EQU*, aby byla vybrána příslušná příčka.



Obr. 6.4: Diagnostická instrukce rung_1

Nastavení nebo resetování proměnné *Input_Output* slouží k nastavení viditelnosti zobrazovacího prvku slavů ve vizualizaci. Proměnná *Making* je spojena s tlačítkem *Proveď* ve vizualizaci a spouští vybranou add-on instrukci. Před spuštěním instrukce tlačítkem musí být vybraná síť masteru (0/1). Po vykonání instrukce se vyčítá ze sběrnice toggle bit, který nastavuje globální proměnnou *Toggle_bit*, protože každá instrukce provádí jeho přepínání a vyčítání samostatně. Posledním prvkem na příčce je přepnutí se na druhou příčku *Aux_next_rung.0* pro nastavení proměnných pro vizualizaci.

Druhá příčka 6.5 instrukce provádí nastavování jednotlivých bitů, aby mohly být navázány ve vizualizaci. To se musí dělat proto, že Factory Talk neumí přistupovat k jednotlivým bitům proměnných (jako např. SINT). Pokud chceme něco ovládat bitově, musí být pro každou položku zvláštní jednobitová proměnná. Natavování jednotlivých bitů dále pokračuje ve směru naznačených teček až pro Slave_8.

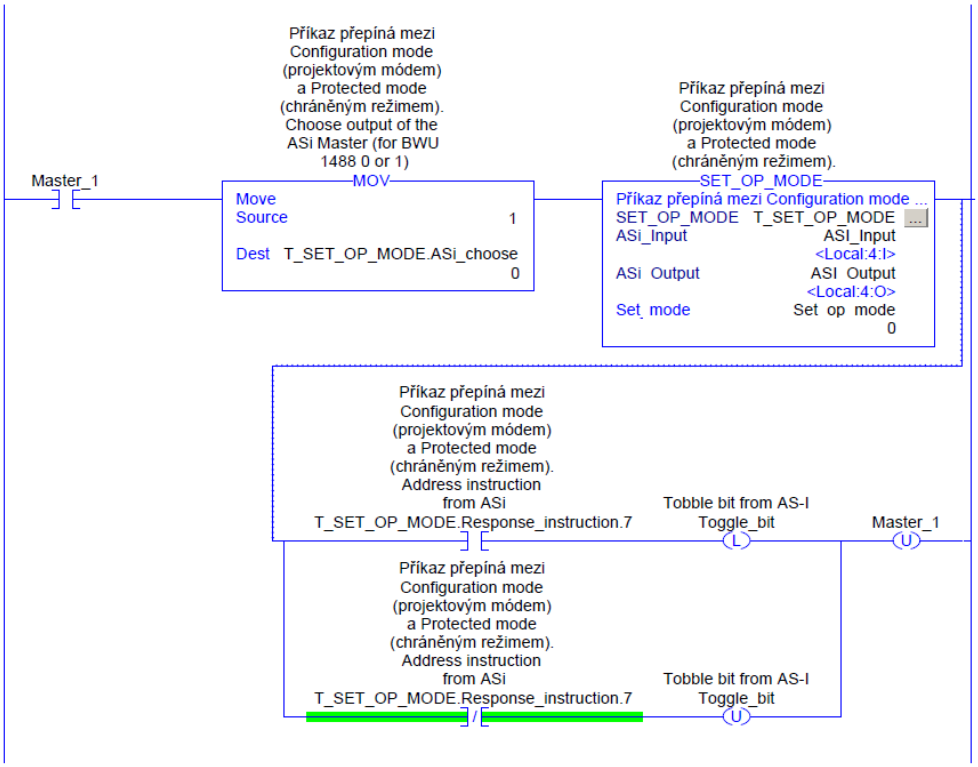


Obr. 6.5: Diagnostická instrukce rung_2

Jednotlivé instrukce se liší volanou add-on instrukcí, případně počtem nastavovaných bitů. Instrukce zapisující na sběrnici se liší pořadím příček a nastavováním bitových proměnných.

6.2.2 Instrukce pro master

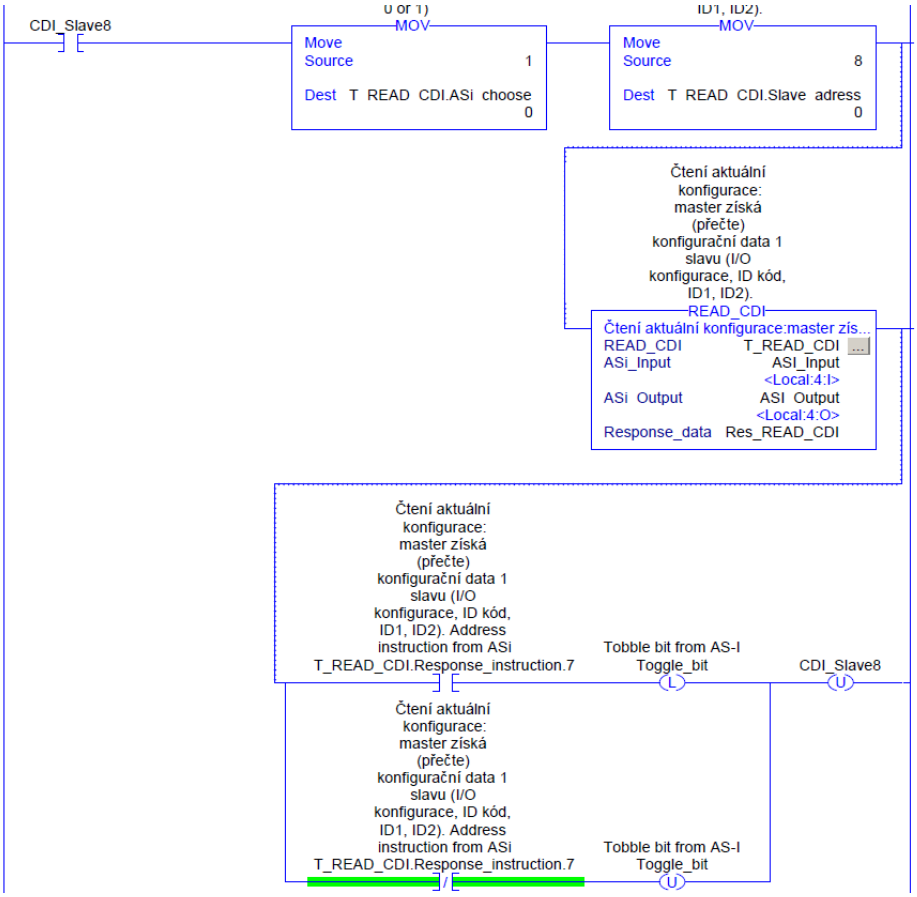
Instrukce pro master je jednodušší oproti diagnostické instrukci 6.6, protože odpadá výběr ze seznamu více instrukcí a tím i funkce *EQU* a nastavování viditelnosti bitem *Input_Output*. Příčku jednotlivé instrukce spouští přímo tlačítko *Proved* v okně vizualizace.



Obr. 6.6: Instrukce pro master

6.2.3 Instrukce pro slave

Tyto instrukce jsou podobné instrukcím pro master. Obsahují ovšem adresu příslušného slavu, která je v programu zapsána funkcí *MOV*, která nastaví adresu slavu do instrukce na požadovanou hodnotu v proměnné *T_READ_CDI.Salve_address*. Spuštění se provede opět tlačítkem *Proveď* v obrazovce příslušného slavu.



Obr. 6.7: Instrukce pro slave

7 NÁVRH LABORATORNÍ ÚLOHY

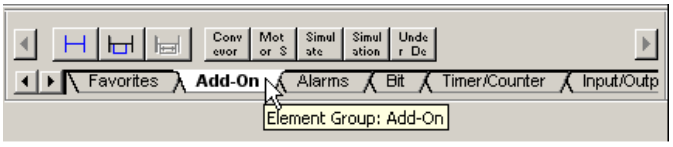
7.1 Zadání

1. Vytvořte hardwarovou konfiguraci pro PLC s připojeným modelem labyrintu.
2. Pomocí dokumentace k AS-Interface masteru zjistěte adresy připojených slavů
3. Implementujte add-on instrukce obsluhující sběrnici AS-I do projektu
4. Ověřte počet aktivních slavů a jejich adresy, zjistěte příznaky sítě a co znamenají.
5. Pro každý slave na síti zjistěte jeho ID, I/O, xID1 a xID2 kód.

7.2 Úvod

AS-Interface je sběrnice na nejnižší úrovni automatizace. Spojuje senzory a aktuátory od různých výrobců do sítě jediným kabelem. Takto spojené nejnižší prvky automatizace lze propojovat do vyšších úrovní řízení (PLC, Gateway na vyšší typ sběrnice). Toto propojení je zpravidla přes žlutý profilovaný kabel, podle kterého lze sběrnici bezpečně rozpoznat. Ke kabelu se zařízení připojují prořezávací technikou, kamkoliv po jeho délce. Sběrnice může vytvářet různé topologické struktury.

Komunikace probíhá pomocí masteru, umístěného v šasi PLC, se slavy na sběrnici respektive gateway, která zastává funkci masteru a PLC, přičemž umožňuje připojení k vyššímu typu průmyslové sběrnice. Komunikace je cyklická. Master pravidelně obvolává všechny slavy připojené ke sběrnici. Každý slave má svou adresu, I/O popis , ID kód a extended ID1 a xID2. Tímto popisem je identifikován a master identifikuje způsob, jakým způsobem k němu přistupovat. Adresu slavu uděluje master automaticky (pokud je zapnutá automatická adresace) nebo ji může zadat programátor softwarově nebo pomocí ručního programátoru adres. Každý rámec, který se na sběrnici objeví, má pevnou délku a lze určit přesnou dobu trvání cyklu sběrnice. Dobu trvání jednoho cyklu určuje počet zařízení na síti (max.62 slavů).



Obr. 7.1: Výběr add-on instrukcí v prostředí ControlLogix5000

Add-on instrukce jsou druhem subrutiny v prostředí ControlLogix5000, která zapouzdřuje často používanou logiku a zpřehledňují celý program. Hotová add-on

instrukce se používá stejně jako ostatní funkce prostředí ControlLogix5000 a lze ji najít v horníma panelu 7.1. Samotnou add-on instrukce lze exportovat do souboru $*.L5X$ a importovat do jiných projektů.

Instrukce pro síť AS-Interface umožňují práci se sběrnicí od její diagnostiky, čítání chyb, konfigurace slavů, konfigurace masteru, přes výměny dat mimo cyklus sítě, až po předávání 16 bitových nebo i delších bloků dat. Každá instrukce má svoji pevnou *adresu instrukce*, kterou je reprezentována. Struktura požadavku a odpovědi sběrnice viz obrázek 7.2. 7.2.

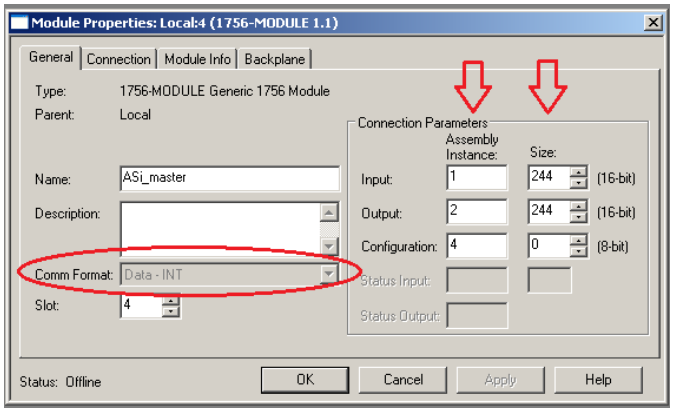
command request								
byte	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	command							
2	T	O	circuit					
3	request parameter byte 1							
...	...							
36	request parameter byte 34							

command response								
byte	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	command (mirrored)							
2	T	result						
3	response byte 1							
...	...							
36	response byte 34							

Obr. 7.2: Požadavek a odpověď ze sběrnice AS-I

7.3 Postup

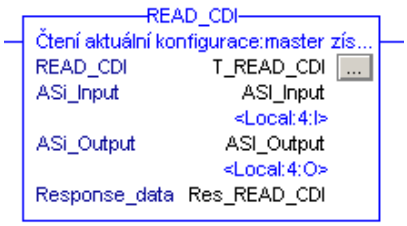
1. Do HW konfigurace přidejte procesor, dle jeho typu a pořadí v PLC. Master AS-Interface se vkládá jako Other 1756-MODULE. Connection Parameters nastave dle obrázku 7.3. Dále nastavte i Common format na Data-INT. Ostatní postup při vkládání Masteru je totožný jako při vkládání jakéhokoli jiného hardwaru.



Obr. 7.3: Nastavení parametrů masteru v HW konfiguraci

2. HW konfiguraci nahrejte do PLC.

3. Vyhledejte datasheet k AS-I masteru. Podle návodu v datasheetu jej přepněte do projektového módu. Na jeho displeji se budou postupně zobrazovat adresy všech připojených slavů k síti. Před přepnutím do projektového (konfiguračního) módu je nutné přepnout procesor z režimu *Remote* do režimu *Program* pomocí klíče. Nezapomeňte přepnout master i procesor do předchozích stavů.
4. Importujte add-on instrukce do vytvořené hardwarové konfigurace. Pro čtení aktivních slavů použijte instrukci *GET_LAS*. Pro získání flagů použijte instrukci *GET_FLAGS*. Import se provádí v panelu *Controller Organizer* v nabídce *Add-on Instructions*. Přidané instrukce můžete vložit do programu stejně jako běžné funkce prostředí. Aby instrukce fungovala, vytvořte k ní řídicí proměnnou datového typu dané instrukce. Pro získání informací ze sběrnice do add-on instrukce musíte připojit i proměnné, které obsluhují sběrnici (vytvořily se jako globální proměnné při tvorbě HW konfigurace masteru). Výsledek by měl vypadat tato 7.4. Každá instrukce je opatřena nápovědou, případně se lze podívat do jejího zdrojového kódu a zjistit tak velikost výstupního pole, nebo co znamenají jednotlivé proměnné. Podrobný popis instrukcí lze najít na internetu v dokumentu *AS-I 3.0 Command Interface Description of the commands* firmy Bihl+Wiedemann.



Obr. 7.4: Add-on instrukce s proměnnými.

5. Instrukce *READ_CDI* dokáže číst parametry jednotlivých slavů. Vložte ji stejně jako předchozí instrukce, ovšem přibude jeden vstupní parametr - adresa slavu, ze kterého budete parametry číst.

8 ZÁVĚR

V předložené diplomové práci byl čtenář seznámen s průmyslovým řízením provozu a sběrnicí AS-Interface. Pro tuto síť existují obslužné instrukce, které byly realizovány. Naprogramovány byly jako add-on instrukce v prostředí ControlLogix5000 a testovány na modelu labyrintu. Práce popisuje řízení průmyslové výroby, samotnou sběrnici a AS-I master s modelem labyrintu, na kterém byly jednotlivé instrukce testovány.

Ze znalostí teoretické části a seznamu všech instrukcí pro sběrnici AS-Interface bylo nejprve zjištěno, jak se s instrukcemi na sběrnici pracuje a jaké jsou jejich možnosti a nároky související s jejich implementací. Z těchto poznatků byl postupně navržen obecný model instrukce tak, aby byla jednoduše implementovatelná, byla přehledná a úsporná při využívání paměťového prostoru procesoru. Zároveň také, aby každá instrukce byla nezávislá a šla samostatně používat v programech případně i na jiných laboratorních přípravcích a modelech a ne pouze na předloženém modelu labyrintu.

Podle obecného modelu instrukce byla následně uskutečněna konkrétní realizace instrukcí pro sběrnici AS-Interface. V jazyku LAD byly implementovány jednotlivé instrukce, jejichž seznam je uveden v textu diplomové práce v kapitole 5.6. Celkový počet instrukcí je 48. Instrukce byly implementovány tak, aby byla vstupní/výstupní data snadno reprezentovatelná. V popisu jednotlivých instrukcí se instrukce drží bajtové reprezentace jednotlivých prvků pole, což bylo pro přehlednost dodrženo. U 16 bitových instrukcí, u kterých se pracuje s dvou bajtovými prvky, jsem volil přehlednější variantu reprezentace pomocí dvoubajtových proměnných. Všechna vstupní a výstupní pole jsou přehledně okomentována, aby nemohlo dojít k záměně dat a jejich špatné interpretaci. Instrukce jsou vyexportovány do samostatných souborů a připravené k použití.

Add-on instrukce byly ověřeny při tvorbě vizualizace, přičemž program pro vizualizaci obsahuje všechny vytvořené instrukce. Na modelu labyrintu jsou použity pouze binární slavy, proto nemohlo dojít k ověření funkčnosti pro 16-ti bitové instrukce, instrukce pro profily $S7.4$ a $S7.5$ a acyklické příkazy. Funkčnost těchto 16-ti bitových instrukcí byla ověřena pouze z části. A to tak, že byly ověřeny části programu pracující s daty - s poli (které se odesílají/přijímají ze sběrnice), které fungovaly.

Program pro vizualizaci a samotná vizualizace demonstrují diagnostiku sítě a základní funkční požadavky instrukcí pro jejich použití s ohledem na navrhovanou laboratorní úlohu. Vizualizace byla navržena na míru modelu labyrintu, z čehož vychází její grafická podoba. Ovládání vizualizace je navrženo tak, aby bylo intuitivní. Z důvodu velkého počtu add-on instrukcí, počtu použitých proměnných a délky

programu jsou reporty programu přiloženy jako elektronická příloha.

V rámci diplomové práce byla navržena laboratorní úloha, která pracuje s add-on instrukcemi a se sběrnicí AS-Interface. Úloha bere ohled na neznalost práce studentů s add-on instrukcemi a je navržena tak, aby ji studenti zvládli a stihli vypracovat během jednoho cvičení v laboratoři.

Rozšíření práce je možné směrem k přenosu 16-ti bitových dat. Zvláště pak rozšíření o problematiku acyklických přenosů 16-ti bitových dat, které jsem neměl možnost otestovat. Také by bylo možné práci rozšířit o speciální instrukce pracující s konkrétní gateway nebo o instrukce pracující se safety monitorem.

LITERATURA

[1] ZEZULKA, František. *Prostředky průmyslové automatizace.* Brno: VUTIUM Brno, 2004, 146 s.

[2] PÁSEK, Jan. *Programovatelné automaty v řízení technologických procesů.* Brno, 2007, 138 s.

[3] ZEZULKA, František, Petr FIEDLER, Petr VAŇOUS a Petr CACH. *Průmyslové komunikační sítě.* ÚAMT FEI VUT v Brně, 2000, 72 s.

[4] KADLEC, Karel a Miloš KMÍNEK. *Programovatelné logické automaty.* Měřicí a řídicí technika [online]. únor 2005 [cit. 2013-05-02]. Dostupné z URL: <http://uprt.vscht.cz/kminekm/mrt/F5/F5-ram.htm>.

[5] BECKER, Rolf, Bernhard MÜLLER, Andreas SCHIFF, Tilman SCHINKE a Heinz WALKER. *AS-Interface Řešení pro automatizaci: příručka, technika, funkce, aplikace.* Germany, Schweinfurt: AS-International Association, 2004, 184 s. ISBN 80-214-2958-5.

[6] ZEZULKA, F., HYNCICA, O. *Průmyslová komunikační síť AS-interface.* Průmyslová komunikační síť AS-interface. Automa [online]. 2004, č. 04 [cit. 2013-04-10]. Dostupné z URL: <http://www.odbornecasopisy.cz/index.php?id_document=32276>.

[7] AS-Interface Česká republika. *AS-Interface = jednoduché systémové řešení: Představení technologie AS-Interface.* [online]. 2006, s. 42 [cit. 2013-05-03]. Dostupné z URL: <http://as-interface.cz/Seminar/Zaklady_ASI_V3_0.pdf>.

[8] AS-Interface Česká republika. *Základní informace o sběrnici AS-I.* [online]. [cit.2013-05-03]. Dostupné z URL: <http://www.as-interface.cz/AS-i_zaklad.html>.

[9] ŠTOHL, R., Petr FIEDLER, Robert DUBJEL *Bezpečnostní systém AS-Interface Safety at Work (část 1).* [online]. [cit.2013-05-17]. Dostupné z URL: <file://localhost/C:/Users/Tom/Dropbox/DP/materialy/články%20-%20net/Automa%20_%20Bezpečnostní%20systém%20AS-Interface%20Safety%20at%20Work%20(část%201).mht>.

[10] Bihl + Wiedemann *AS-i 3.0 Command Interface.* Description of the commands. [online]. [cit. 2013-05-07]. Dostupné z URL: <http://www.trelectronic.se/downloads/bw/docs/2206/command_interface_description_of_commands.pdf>.

[11] *Logix5000 Controllers Add-On Instructions* [online]. [cit. 2013-05-7]. Dostupné z URL: <http://literature.rockwellautomation.com/idc/groups/literature/documents/pm/1756-pm010_-en-p.pdf>.

[12] ZBRANEK, P. *Model labyrintu.* Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2011. 50s. Vedoucí bakalářské práce byl Ing. RADEK ŠTOHL, Ph.D.

[13] *IFM Electronic.* AC5213 [online]. [cit. 2013-05-05]. Dostupné z URL: <http://www.ifm.com/products/cz/ds/AC5213.htm>.

[14] *IFM Electronic.* AC5000 [online]. [cit. 2013-05-05]. Dostupné z URL: <http://www.ifm.com/products/cz/ds/AC5000.htm>.

[15] *IFM Electronic.* AC2088 [online]. [cit. 2013-05-05]. Dostupné z URL: <http://www.ifm.com/products/cz/ds/AC2086.htm>.

[16] *IFM Electronic.* AC5210 [online]. [cit. 2013-05-05]. Dostupné z URL: <http://www.ifm.com/products/cz/ds/AC5210.htm>.

[17] *AS-Interface* Siemens AS-i Power Supply 3A (DC 30V) [online]. [cit. 2013-05-05]. Dostupné z URL: <http://www.as-interface.net/as-international/members/si/products/as-i-power-supply-3a-(dc-24v)>.

[18] *AS-i 3.0 Master/Scanner for Allen-Bradley ControlLogix* [online]. [cit. 2013-05-18]. Dostupné z URL: <http://www.bihl-wiedemann.de/fileadmin/catalog/Selektoren/Gateways/1611_1488/1488/en/1488_1611_data_sheet.pdf>.

Obrázky dostupné z internetu

[19] *PLC firmy Allen-Bradley* [cit. 2013-05-01]. Dostupný z URL: <http://epub1.rockwellautomation.com/images/web-proof-large/GL/48407.jpg>.

[20] *Princip činnosti PLC* [cit. 2013-05-02]. Dostupný z URL: <http://uprt.vscht.cz/kminekm/mrt/F5/F5k53-o54.gif>.

[21] *Logo AS-i* [cit. 2013-05-02]. Dostupný z URL: <http://www.anybus.de/images/logo_as-interface.gif>.

[22] *Slave ifm AC5213* [cit. 2013-05-05]. Dostupný z URL: <http://www.ifm.com/tedo/foto/asi_0202.gif>.

[23] *Slave ifm AC5000* [cit. 2013-05-05]. Dostupný z URL: <http://www.ifm.com/tedo/foto/asi_0086.gif>.

[24] *Slave ifm AC2086* [cit. 2013-05-05]. Dostupný z URL: <http://www.ifm.com/tedo/foto/asi_0055.gif>.

[25] *Slave ifm AC5210* [cit. 2013-05-05]. Dostupný z URL: <http://www.ifm.com/tedo/foto/asi_0200.gif>.

[26] *AS-i Power Supply 3A* [cit. 2013-05-05]. Dostupný z URL: <http://cache.automation.siemens.com/dnl/DA/DAzNTk0OQAA_3RX95010BA00_MLFB/P_NSA0_XX_00526i.jpg>.

[27] *Master BWU 1488* [cit. 2013-05-18]. Dostupný z URL: <http://www.bihl-wiedemann.de/typo3temp/pics/1488_1611_8c4036196d.jpg>.

SEZNAM SYMBOLŮ, VELIČIN A ZKRATEK

PLC Programmable Logic Controlers

ERP Enterprise Resource Planning

MES Manufacturing Execution System

SCADA Supervisory Control And Data Acquisition

PCS Proces Control System

CPU Central Processing Unit

I/O Input/Output

TCP/IP Transmission Control Protocol / Internet Protocol

HW Hardware

LAD Ladder Diagram

FBD Function Block Diagram

STL Statement List

ASi Actuator/Sensor Interface

AMP Alternating Pulse Modulation

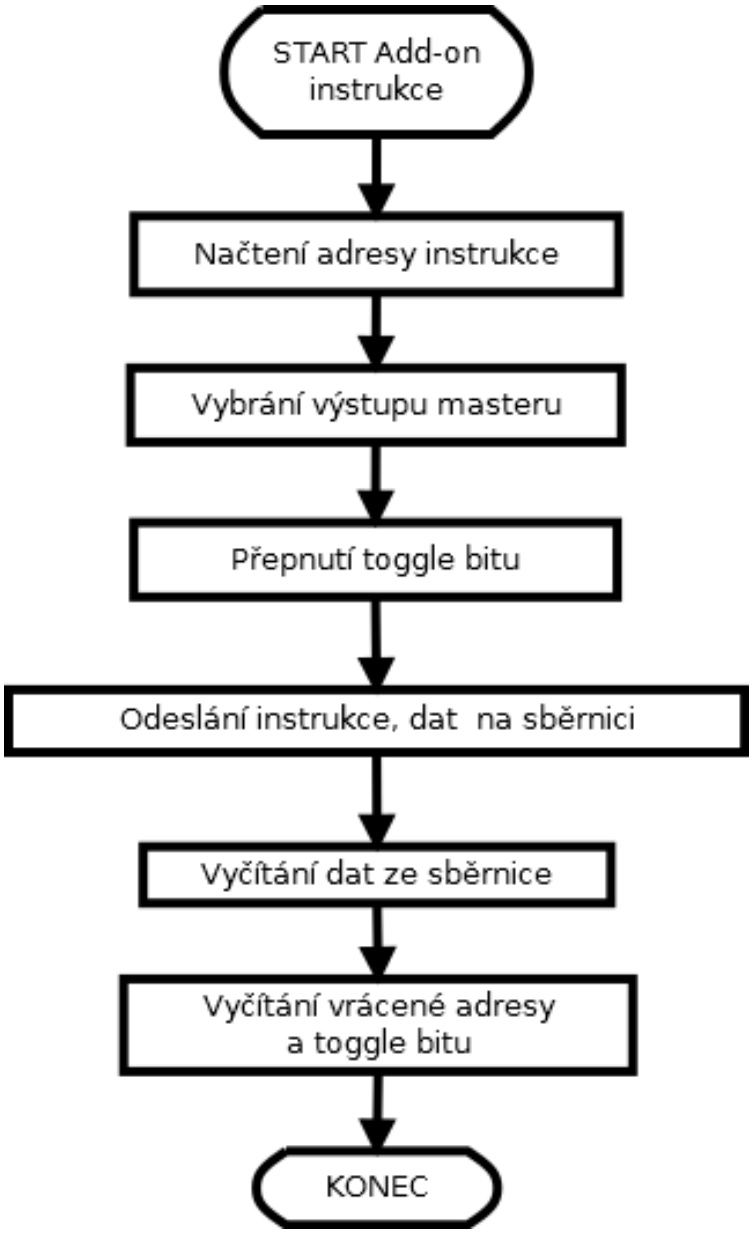
SEZNAM PŘÍLOH

A POROVNÁNÍ ÚROVNÍ PROGRAMOVÁNÍ V CONTROL LOGIX5000

Aspekt	Main Routine	Subrutine	Add-on instrukce
Přístupnost	N/A	V rámci programu (lze více kopií, jedna pro každý program)	Kdekoliv (jedna kopie pro celý projekt
Parametry	N/A	Předání hodnotou	Předání hodnotou přes vstupní a výstupní parametr nebo odkazem přes InOut
Číselný parametr	N/A	Bez převodu (přetypování), který musí provést uživatel	Automatický datový typ pro vstupní a výstupní parametry
Parametry Datových typů	N/A	Jednoduchý, pole, struktura	Jednoduchý pro kterýkoliv parametr, pole a struktura musí být InOut
Kontrola parametrů	N/A	Musí řešit uživatel	Ověření správného typu argumentu podle stanoveného parametru
Zapouzdření dat	N/A	všechny data přístupné	Lokální data jsou oddělený (přístupné pouze konkrétní instrukci)
Monitoring, ladění	Online kód i data	Smíšená data z vícenásobného volání subrutiny	Volání jednotlivých datových instancí, nemožnost ladění (či náhledu) kódu online
Ochrana	Uzamčení, možný náhled	Uzamčení, možný náhled	Uzamčení, možný náhled & Uzamčení, možný náhled
Dokumentace	Rutina, příčka, textbox	Rutina, příčka, textbox	Instrukce, informace o revizi, autor, rozšířená nápověda, textbox
Provedení logiky	Nejrychlejší	JSR / SBR / RTN přidají hlavičky, všechna data jsou kopírována	InOut parametr předán formou odkazu (rychlejší než kopírování dat)
Využití paměti	Největší	Velmi kompaktní	Kompaktní volání vyžaduje více paměti než volání subrutiny. Všechny odkazy potřebují další slovo.
Editace	Online / offline	Online / offline	Online/offline & Pouze offline
Import / export	Celá rutina, včetně proměnných, *.L5X	Celá rutina, včetně proměnných, *.L5X	Celá rutina, včetně proměnných, *.L5X

Obr. A.1: Porovnání úrovní programování v softwaru ControlLogix5000[11]

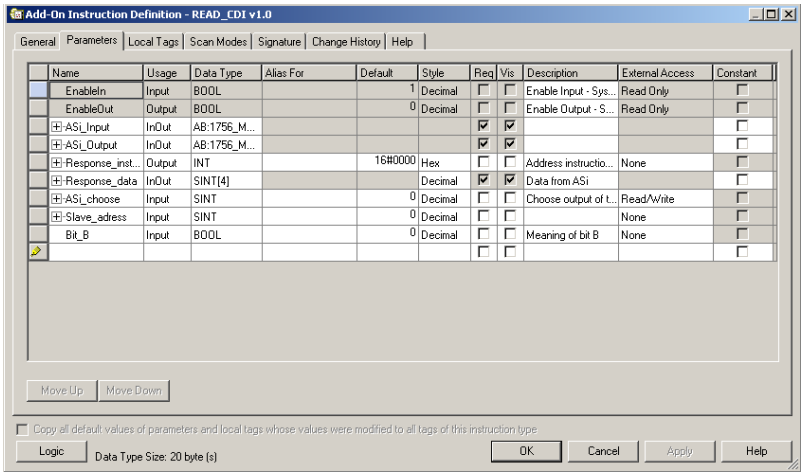
B VÝVOJOVÝ DIAGRAM ADD-ON INSTRUKCE



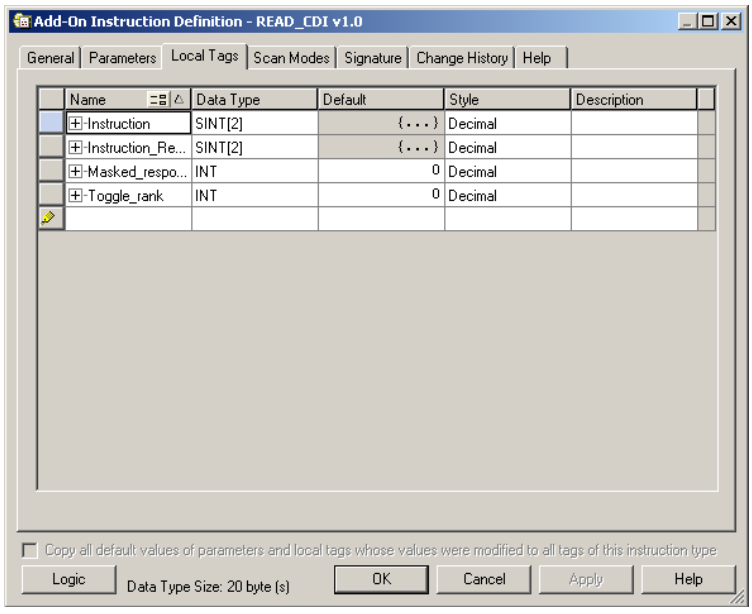
Obr. B.1: Vývojový diagram add-on instrukce.

C UKÁZKY LOGIKY INSTRUKCE *READ_CDI*

C.1 Proměnné

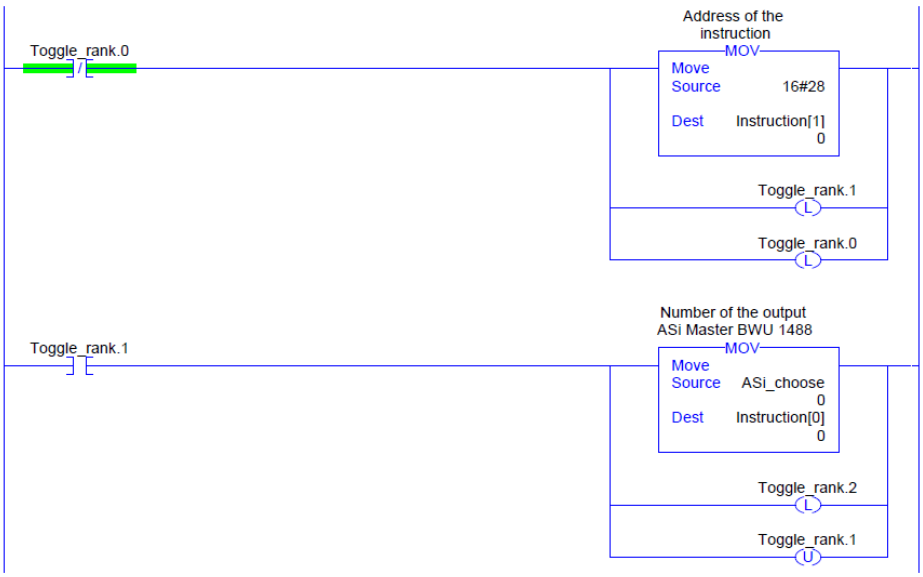


Obr. C.1: Vstupně - výstupní proměnné add-on instrukce *READ_CDI*

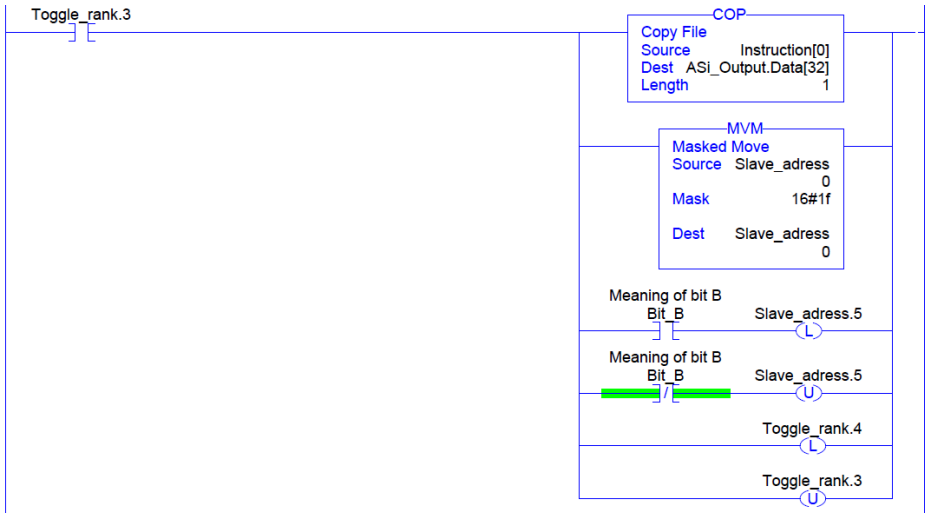


Obr. C.2: Lokální proměnné add-on instrukce *READ_CDI*

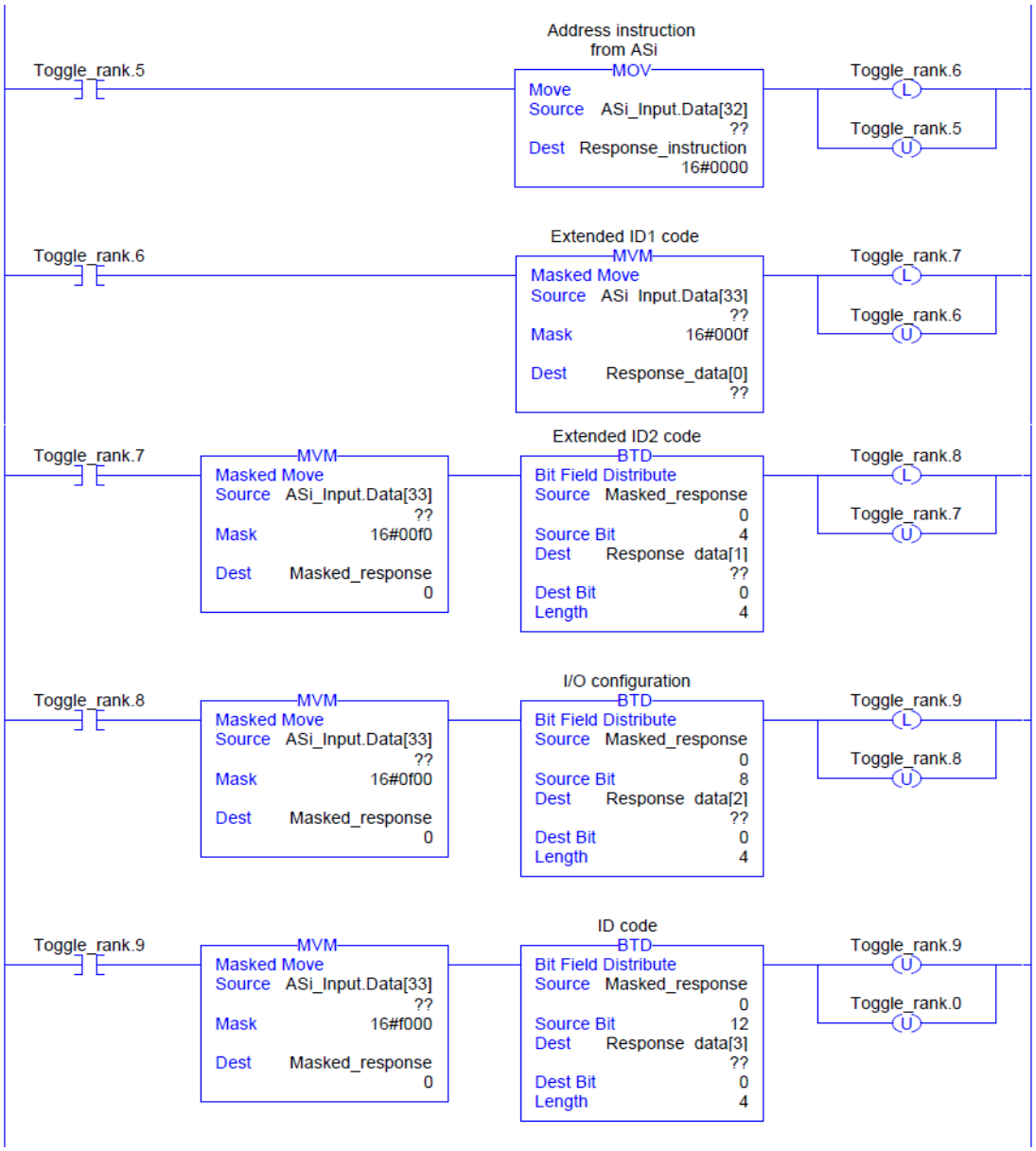
C.2 Logika instrukce



Obr. C.3: První dvě příčky logiky add-on instrukce



Obr. C.4: Promaskování adresy a nakopírování instrukce do proměnné sběrnice



Obr. C.5: Vyčítání dat ze sběrnice.

D PODROBNÝ SEZNAM REALIZOVANÝCH ADD-ON INSTRUKCÍ

D.0.1 Přenos 16 bitových dat

- **Read** 1 16−**bit Slave in.Data** (RD_7X_IN)
  Čtení čtyř 16-ti bitových kanálů na vstupu jednoho modulu podle profilů ($S-7.3$, $S-7.4$, $S-7.5$, $S-7.A.8$, $S.A.9$, $S-7.A.A$).

Request								
byte	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	50_{16}							
2	T	–	circuit					
3	–		0	slave address				

Response								
byte	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	50_{16}							
2	T	result						
3	channel 1, high byte							
...	...							
10	channel 4, low byte							

Obr. D.1: Vstupní a výstupní interface instrukce RD_7X_IN

- **Write** 1 16−**bit Slave out.Data** (WR_7X_OUT)
  Zápis čtyř 16-ti bitových kanálů na výstup jednoho modulu v závislosti na profilu ($S-7.3, S-7.4, S-7.5, S-7.A.8, S.A.9, S-7.A.A$).

Request								
byte	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	51_{16}							
2	T	–	circuit					
3	–		0	slave address				
4	channel 1, high byte							
...	...							
11	channel 4, low byte							

Response								
byte	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	51_{16}							
2	T	result						

Obr. D.2: Vstupní a výstupní interface instrukce WR_7X_OUT

- **Read** 1 16−**bit Slave out.Data** (RD_7X_OUT)
  Čtení čtyř 16-ti bitových kanálů na výstupu jednoho slavu dle profilů ($S-7.3, S-7.4, S-7.5, S-7.A.8, S.A.9, S-7.A.A$).

Request								
byte	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	52_{16}							
2	T	–	circuit					
3	–		0	slave address				

Response								
byte	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	52_{16}							
2	T	result						
3	channel 1, high byte							
…	…							
10	channel 4, low byte							

Obr. D.3: Vstupní a výstupní interface instrukce RD_7X_OUT

- **Read** 4 16−**bit Slave in.Data** ($RD\_7X\_IN\_X$)
  Čtení čtyř 16-ti bitových kanálů na vstupech čtyř slavů, jejichž adresy jsou bezprostředně za sebou podle profilů ($S-7.3, S-7.4, S-7.5, S-7.A.8, S.A.9, S-7.A.A$).

Request								
byte	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	53_{16}							
2	T	–	circuit					
3	–		0	1st slave address				

Response								
byte	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	53_{16}							
2	T	result						
3	1st slave, channel 1, high byte							
…	…							
34	4th slave, channel 4, low byte							

Obr. D.4: Vstupní a výstupní interface instrukce $RD\_7X\_IN\_X$

- **Write** 4 7.3 **Slave out.Data**($WR\_7X\_OUT\_X$)
  Zápis čtyř 16-ti bitových kanálů na výstupy čtyř po sobě jdoucích slavů dle profilů ($S-7.3, S-7.4, S-7.5, S-7.A.8, S.A.9, S-7.A.A$).

Request								
byte	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	54_{16}							
2	T	–	circuit					
3	–		0	1st slave address				
4	1st slave, channel 1, high byte							
…	…							
35	4th slave, channel 4, low byte							

Response								
byte	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	54_{16}							
2	T	result						

Obr. D.5: Vstupní a výstupní interface instrukce $WR\_7X\_OUT\_X$

- **Read** 4 7.3 **Slave out.Data** ($RD\_7X\_OUT\_X$)
  Čtení čtyř 16-ti bitových kanálů na výstupech čtyř slavů jejichž adresy jsou bezprostředně za sebou podle profilů ($S-7.3, S-7.4, S-7.5, S-7.A.8, S.A.9, S-7.A.A$).

Request								
byte	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	55_{16}							
2	T	–	circuit					
3	–		0	1st slave address				

Response								
byte	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	55_{16}							
2	T	result						
3	1st slave, channel 1, high byte							
…	…							
34	4th slave, channel 4, low byte							

Obr. D.6: Vstupní a výstupní interface instrukce $RD\_7X\_OUT\_X$

- **Read** 16 **channels** 16**−bit Slave in.Data** ($OP\_RD\_16BIT\_IN\_CX$)
  Čtení 16-ti bitové informace vstupních dat na 16 kanálech s možností zadání počtu kanálu na jeden slave (maximum 16) dle profilu ($S-7.3, S-7.4, S-7.5, S-7.A.8, S.A.9, S-7.A.A$).

Request								
Byte	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	$4C_{16}$							
2	T	–	circuit					
3	1. slave							
4	1. channel							

Response								
Byte	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	$4C_{16}$							
2	T	result						
3	1. slave, channel 1, high byte							
4	1. slave, channel 1, low byte							
…	…							
33	16. channel, high byte							
34	16. channel, low byte							

Obr. D.7: Vstupní a výstupní interface instrukce $OP\_RD\_16BIT\_IN\_CX$

- **Write 16 channels 16−bit slave out.Data** ($OP\_WR\_16BIT\_OUT\_CX$)
  Zápis 16-ti bitové informace výstupních dat na 16 kanálů dle profilu ($S-7.3, S-7.4, S-7.5, S-7.A.8, S.A.9, S-7.A.A$) [10].

Request								
Byte	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	$4D_{16}$							
2	T	circuit						
3	1. slave							
4	1. channel							
5	1. slave, 1. channel, high byte							
6	1. slave, 1. channel, low byte							
…	…							
35	16. channel, high byte							
36	16. channel, low byte							

Response								
Byte	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	$4D_{16}$							
2	T	result						

Obr. D.8: Vstupní a výstupní interface instrukce $OP\_WR\_16BIT\_IN\_CX$

D.0.2 Příkazy pro profily $S-7.4/S-7.5$

- $WR\_74\_75\_PARAM$
  Zápis řetězce do slavu dle profilu $S-7.4$, případně jeho přenos dle profilu $S-7.5$. Ve slavu profilu $S-7.5$ musejí být data zapsána do bufferu stejnou formou jako na sběrnici AS-interface. Vzhledem k tomu, že řetězec může být delší než dovoluje příkaz, je zapsán po částech do bufferu poté je odeslán do

slavu. n udává délku řetězce, který by měl být zapsán do bufferu (vyrovnávací paměti) od indexu i dál. Pokud je $i = 0$ je řetězec převeden na slave.

Request								
byte	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	$5A_{16}$							
2	T	–	circuit					
3	slave address							
4	i							
5	n							
6	buffer byte i							
…	…							
n+5	buffer byte i+n-1							

Response								
byte	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	$5A_{16}$							
2	T	results						

Obr. D.9: Vstupní a výstupní interface instrukce $WR\_74\_75\_PARAM$

- $RD\_74\_75\_PARAM$
  Čtení řetězce dle profilu $S - 7.4$ (nebo také podle profilu $S - 7.5$). Je - li profil $S - 7.5$ jsou data v bufferu následujícím způsobem:
  FF_h00_h: přenos je stále aktivní
  FF_hxx_h: přenos skončil chybou
  Pokud první byte není FF_h slave právě odpovídá. Pokud je řetězec delší, než se vleze do paketu, je zapsán do bufferu. První bajt v bufferu je délka řetězce.

Request								
byte	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	$5B_{16}$							
2	T	–	circuit					
3	slave address							
4	i							

Response								
byte	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	$5B_{16}$							
2	T	result						
3	buffer byte i							
…	…							
n+2	buffer byte i+n-1							

Obr. D.10: Vstupní a výstupní interface instrukce $RD\_74\_75\_PARAM$

- *RD_74_75_ID*
  Čtení ID řetězce slavu dle profilu $S - 7.4$ nebo 16 bitového slavu konfigurovaného dle profilu $S - 7.5$. Pokud je řetězec delší než umožňuje rozhraní, je zapsán do vyrovnávací paměti, jejíž obsah může být čten po částech od indexu i. První bajt opět nese informaci o délce řetězce.

Request								
byte	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	$5C_{16}$							
2	T	–	circuit					
3	slave address							
4	i							

Response								
byte	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	$5C_{16}$							
2	T	result						
3	buffer byte i							
…	…							
n+2	buffer byte i+n-1							

Obr. D.11: Vstupní a výstupní interface instrukce *RD_74_75_ID*

- *RD_74_DIAG*
  Čtení diagnostického řetězce slavu dle profilu $S - 7.4$. Delší řetězec je uložen do bufferu. OBsah vyrovnávací paměti lze číst po částech od indexu i. První bajt určuje délku řetězce. Pokud je $i = 0$ je řetězec právě čten ze slavu. Data mohou být čtena souvisle [10].

Request								
byte	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	$5D_{16}$							
2	T	–	circuit					
3	slave address							
4	i							

Response								
byte	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	$5D_{16}$							
2	T	result						
3	buffer byte i							
…	…							
n+2	buffer byte i+n-1							

Obr. D.12: Vstupní a výstupní interface instrukce *RD_74_DIAG*

D.0.3 Acyklické příkazy

- $WRITE\_ACYCLIC\_TRANS$
  Funkce umožňuje různé acyklické přenosy pro slavy s profily $S-7.4$, $S-7.5$ a $Safetymonitory$. Přenos se provádí na pozadí a jeho výsledek může být čten pouze příkazem $READ\_ACYCLIC\_TRANS$. Tato funkce má být náhradou za funkce pro profily $S-7.4$ a $S-7.5$ - $WR\_74\_75\_PARAM$, $RD\_74\_75\_PARAM$, $RD\_74\_75\_ID$, RD_74_DIAG a $SafetyatWork$. Délka přenášených dat může být větší než velikost rozhraní, data se nejprve zapíší do vyrovnávací paměti, dokud se přenos nespustí. n je délka sub-řetězce, který je zapsán do vyrovnávací paměti počínaje indexem i. Pokud $i=0$, spustí se přenos.
  V příkazu $WRITE\_ACYCLIC\_TRANS$ se uvádí číselná hodnota příkazu, který se má vykonat D.1:

Tab. D.1: Jednotlivé příkazy instrukce $WRITE\_ACYCLIC\_TRANS$.

Příkaz	Popis
1	čtení řetězce $S-7.4\,ID$
2	čtení řetězce $S-7.4\,diag$
3	čtení řetězce $S-7.4\,param$
4	zápis řetězce $S-7.4\,param$
5	Přenos $S-7.5$
6	Cyklické čtení 16 bitové konfigurace slave $S-7.5$
7	Čtení diagnostiky safety monitoru (roztříděno dle OSSD)
8	Čtení diagnostiky safety monitoru (netříděno)
9	Není definováné / rezervováno
10	Čtení aktuální diagnostiky Safety monitoru
11	Čtení historie vypínání (shutdown)
12	Čtení aktuální diagnostiky alokovaných zařízení Safety monitoru
13	Čtení historie vypínání alokovaných zařízení Safety monitoru
14	Čtení diagnostiky safety monitoru
15	Stav bezpečnosti (safety)
16	Čtení indexu identifikátoru zařízení (identifikátor číst jako prostý text)

Request								
Byte	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	$4E_{16}$							
2	T	circuit						
3	slave address							
4	buffer index (i) high							
5	buffer index (i) low							
6	command[1]							
7	number of (n)							
8	data 0							
...	...							
n+7	data n-1							

Response								
Byte	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	$4E_{16}$							
2	return							

Obr. D.13: Vstupní a výstupní interface instrukce *WRITE_ACYCLIC_TRANS*

- *READ_ACYCLIC_TRANS*
  Čtení odpovědi na příkaz *WRITE_ACYCLIC_TRANS* [10].

Request								
Byte	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	$4F_{16}$							
2	T	circuit						
3	slave address							
4	buffer index (i) high							
5	buffer index (i) low							

Response								
Byte	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	$4F_{16}$							
2	T	response						
3	data i							
...	...							
m[1]	data i+(m-2)							

Obr. D.14: Vstupní a výstupní interface instrukce *READ_ACYCLIC_TRANS*

D.0.4 Diagnostika sítě AS-interface

- **Get List and Flags** (*GET_LIST*)
  Čtení diagnostických údajů sítě:
  seznam aktivních slavů = LAS
  seznam detekovaných slavů = LDS
  seznam naprojektovaných slavů = LPS

příznaky (flagy) dle AS-interface specifikace:

Pok = Periferie _ OK

$S0 = LDS.0$

AAs = povolení automatické adresace

AAv = automatická adresace k dispozici (je možná)

CA = konfigurace aktivní

NA = normální provoz aktivní

APF = chyba napájení AS-interface

OF = offline připraven

Cok = konfigurace OK

AAe = automatická adresace povolena

OL = offline

DX = datová výměna aktivní

Request								
byte	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	30_{16}							
2	T	O	circuit					

Response (if O ≡ 0)								
byte	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	30_{16}							
2	T	result						
3	7A	6A	5A	4A	3A	2A	1A	0A
…	LAS							
10	31B	30B	29B	28B	27B	26B	25B	24B
11	7A	6A	5A	4A	3A	2A	1A	0A
…	LDS							
18	31B	30B	29B	28B	27B	26B	25B	24B
19	7A	6A	5A	4A	3A	2A	1A	0A
…	LPS							
26	31B	30B	29B	28B	27B	26B	25B	24B
27	–							Pok
28	OR	APF	NA	CA	AAv	AAs	S0	Cok
29	–					AAe	OL	DX

Obr. D.15: Vstupní a výstupní interface instrukce *GET_LIST*

- **Get Flags (*GET_FLAGS*)**

Čtení flagů (identifikátorů, příznaků).

Pok = flag je nastaven (je roven 1), pokud slave nesignalizuje periferní chybu

$S0$ = flag nastaven, když existuje slave s adresou 0

AAs = flag nastaven, když může být vykonávána automatická adresace (tzn.slavy nejsou nekorektně připojeny)

AAv = flag nastaven, když může být vykonávána automatická adresace

a právě jeden slave je mimo provoz

CA = příznak nastaven v konfiguračním režimu a obnoven v chráněném režimu

NA = příznak nastaven, pokud je master v normálním stavu

APF = příznak nastaven, při nízké úrovni napětí na kabelu AS-interface

Cok = příznak nastaven, pokud odpovídá požadovaná a aktuální konfigurace

AAe = flag indikuje, zda je automatická adresace povolena (= 1) nebo zakázána (= 0) uživatelem

OL = flag nastaven, pokud by měla být změna do offline režimu nebo pokud je tento režim brzy dosažitelný

DX = když je flag nastaven, výměna dat mezi masterem a slavem je právě ve fázi výměny dat. Pokud bit není nastaven, výměna dat není k dispozici. Přečtené ID zprávy se přenášejí na slave. Bit je nastaven, pokud master vstoupí do offline režimu.

Request

byte	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	47_{16}							
2	T	–	circuit					

Response

byte	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	47_{16}							
2	T	response						
3								Pok
4	OR	APF	NA	CA	AAv	AAs	S0	Cok
5	–					AAe	OL	DX

Obr. D.16: Vstupní a výstupní interface instrukce GET_FLAGS

- **Get Delta List** (GET_DELTA)
  Delta list obsahuje seznam adres slavů s konfiguračními chybami.

Request								
byte	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	57_{16}							
2	T	0	circuit					

Response (if O ≡ 0)								
byte	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	57_{16}							
2	T	result						
3	7A	6A	5A	4A	3A	2A	1A	–
…	…							
10	31B	30B	29B	28B	27B	26B	25B	24B

Obr. D.17: Vstupní a výstupní interface instrukce *GET_DELTA*

- **Get list of corrupted Slaves** (*GET_LCS*)
  Seznam poškozených slavů.

Request								
byte	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	60_{16}							
2	T	O	circuit					

Response (if O ≡ 0)								
byte	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	60_{16}							
2	T	result						
3	7A	6A	5A	4A	3A	2A	1A	0A
…	…							
10	31B	30B	29B	28B	27B	26B	25B	24B

Obr. D.18: Vstupní a výstupní interface instrukce *GET_LCS*

- **Get list of activated Slaves** (*GET_LAS*)
  Získá seznam aktivních slavů

Request								
byte	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	45_{16}							
2	T	O	circuit					

Response (if O ≡ 0)								
byte	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	45_{16}							
2	T	result						
3	7A	6A	5A	4A	3A	2A	1A	0A
…	…							
10	31B	30B	29B	28B	27B	26B	25B	24B

Obr. D.19: Vstupní a výstupní interface instrukce *GET_LAS*

- **Get list of detected Slaves** (*GET_LDS*)
  Čtení seznamu detekovaných slavů

Request								
byte	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	46_{16}							
2	T	O	circuit					

Response (if O ≡ 0)								
byte	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	46_{16}							
2	T	result						
3	7A	6A	5A	4A	3A	2A	1A	0A
…	…							
10	31B	30B	29B	28B	27B	26B	25B	24B

Obr. D.20: Vstupní a výstupní interface instrukce *GET_LDS*

- **Get list of peripheral Faults** (*GET_LPF*)
  Čtení seznamu periferních poruch (LPF). Tento seznam je aktualizován cyklicky masterem. Pokud slave signalizuje chybu připojených periferií (např. přerušený drát) mohou být nalezeny v popisu slavu.

Request								
byte	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	$3E_{16}$							
2	T	O	circuit					

Response (if O ≡ 0)								
byte	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	$3E_{16}$							
2	T	result						
3	7A	6A	5A	4A	3A	2A	1A	0A
…	…							
10	31B	30B	29B	28B	27B	26B	25B	24B

Obr. D.21: Vstupní a výstupní interface instrukce *GET_LPF*

- **Get list of offline Slaves** (*GET_LOS*)
  Seznam slavů v offline režimu, když se vyskytuje konfigurační chyba. Uživatel může zvolit reakci na konfigurační chybu. Master může AS-interface při konfigurační chybě u důležitého slavu vypnout. Méně důležité slavy mohou hostiteli odeslat chybu (síť v tomto případě nebude v režimu offline).

Request								
byte	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	61_{16}							
2	T	O	circuit					

Response (if O ≡ 0)								
byte	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	61_{16}							
2	T	result						
3	7A	6A	5A	4A	3A	2A	1A	0A
…	…							
10	31B	30B	29B	28B	27B	26B	25B	24B

Obr. D.22: Vstupní a výstupní interface instrukce *GET_LOS*

- **Set list of offline Slaves** (SET_LOS)
  Příkazem je definován seznam slavů v režimu offline (u kterých se vyskytla konfigurační chyba).

Request (if O ≡ 0)								
byte	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	62_{16}							
2	T	O	circuit					
3	7A	6A	5A	4A	3A	2A	1A	0A
...	...							
10	31B	30B	29B	28B	27B	26B	25B	24B

Response								
byte	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	62_{16}							
2	T	result						

Obr. D.23: Vstupní a výstupní interface instrukce SET_LOS

- **Get transm.err.counters** (GET_TECA)
  Čtení čítače chyb všech slavů s číselnou adresou nebo s A adresou. Čtením této informace se pokaždé čítač chyb restartuje. Maximální hodnota čítače chyb je 254, hodnota 255 způsobí, že čítač přeteče. Za účelem získání skutečného počtu transkripčních (počet špatných telegramů s daným modulem slave) chyb je potřeba číslo vynásobit hodnotou 2.

Request								
byte	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	63_{16}							
2	T	–	circuit					

Response								
byte	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	63_{16}							
2	T	result						
3	APF							
4	slave 1A							
...	...							
34	slave 31A							

Obr. D.24: Vstupní a výstupní interface instrukce GET_TECA

- **Get transm.err.counters** (GET_TECB)
  Čtení čítače chyb B - slavů. Ostatní chování je stejné, jako u příkazu GET_TECA.

Request								
byte	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	64_{16}							
2	T	–	circuit					

Response								
byte	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	64_{16}							
2	T	result						
3	APF							
4	slave 1B							
...	...							
34	slave 31B							

Obr. D.25: Vstupní a výstupní interface instrukce GET_TECB

- **Get transm.err.counters** (GET_TEC_X)
  Čtení chyb od zadané adresy určitého slavu do zadaného počtu adres (počítadlo adres se inkrementuje dle zadané hodnoty). Ostatní chování je stejné, jako u příkazu GET_TECA [10].

Request								
byte	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	66_{16}							
2	T	–	circuit					
3	1. slave address							
4	number of counters							

Response								
byte	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	66_{16}							
2	T	result						
3	counter 1							
...	...							
n	counter n - 2							

Obr. D.26: Vstupní a výstupní interface instrukce GET_TECX

D.0.5 Konfigurace AS-I masteru

- **Set Operation mode** (SET_OP_MODE)
  Příkaz přepíná mezi Configuration mode (projektový módem) a Protected mode (chráněným režimem). AS-I slave může být zapsán do Seznamu naprojektovaných slavů (List projected slaves - LPS) jen tehdy, pokud je aktivní Protected mode, čímž jsou jejich předpokládané a aktuální konfigurace aktivovány. Jinak řečeno, slave je aktivován, pokud detekovaná konfigurace I/O a ID jsou shodné s již nakonfigurovanými hodnotami.
  V projektovém režimu jsou všechny detekované slavy (kromě slavu "0") aktivovány. To platí i pro slavy, které mají rozdílnou hodnotu I/O a ID s aktuální konfigurací.

Bit Operation Mode je trvale uložen a je zachován i po restartu. Při změně z Configuration mode na Protected mode provádí master warm restart (změna na offline následovaná změnou na on-line mode). Pokud je na sběrnici slave s adresou 0 a je zapsán v seznamu detekovaných slavů LDS, master nemůže provést změnu z Configuration mode na Protected mode.
0 - protected mode
1 - configuration mode

Request								
byte	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	$0C_{16}$							
2	T	–	circuit					
3	operation mode							

Response								
byte	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	$0C_{16}$							
2	T	result						

Obr. D.27: Vstupní a výstupní interface instrukce *SET_OP_MODE*

- **Store actual configuration** (*STORE_CDI*)
  Ulož aktuální konfiguraci - při této výzvě jsou všechna konfigurační (skutečná) data (I/O konfigurace, ID kód, rozšířený ID1 a ID2 kód) všech slavů na sběrnici trvale uložena v EEPROM jako (očekávaná) konfigurační data. Seznam aktivních slavů (LAS) je přidán k seznamu stálých slavů (LPS).
  Při provádění tohoto příkazu se master přepne do offline fáze a poté se vrátí k normálnímu režimu (warm restart). Tento příkaz může být spuštěn pouze v konfiguračním režimu.

Request								
byte	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	07_{16}							
2	T	–	circuit					

Response								
byte	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	07_{16}							
2	T	result						

Obr. D.28: Vstupní a výstupní interface instrukce *STORE_CDI*

- **Read actual configuration** (*READ_CDI*)
  Čtení aktuální konfigurace - master získá (přečte) konfigurační data 1 slavu (I/O konfigurace, ID kód, ID1, ID2) jehož adresa je zadána jako vstupní parametr příkazu. Konfigurační data jsou specifikovány výrobcem slavu. Přepnutí mezi typy adresací provádí tzv. B bit ($B = 0$ pro jednoduchou adresaci a A-slavy, $B = 1$ pro B-slavy).

Request								
byte	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	28_{16}							
2	T	–	circuit					
3	–		B	slave address				

Response								
byte	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	28_{16}							
2	T	result						
3	xID2				xID1			
4	ID				I0			

Obr. D.29: Vstupní a výstupní interface instrukce *READ_CDI*

- **Set permanent configuration** (*SET_PCD*)
  Nastavení trvalé konfigurace - funkce nastaví následující konfiguraci do slavu (I/O konfigurace, ID kód, ID1, ID2), jeho adresa je předána jako vstupní parametr funkce a B bit vybírá typ adresace. Konfigurační data jsou trvale umístěna v paměti EEPROM v masteru a jsou použita jako očekávaná konfigurace v Protected mode.
  Pokud slave nepodporuje rozšířený ID kód 1 nebo 2, musí být nahrazen hodnotou F_{HEX}. Při provádění tohoto příkazu provede master warm restart.

Request								
byte	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	25_{16}							
2	T	–	circuit					
3	–		B	slave address				
4	xID2				xID1			
5	ID				I0			

Response								
byte	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	25_{16}							
2	T	result						

Obr. D.30: Vstupní a výstupní interface instrukce *SET_PCD*

- **Get extended permanent configuration** (*GET_PCD*)
  Získání rozšířené stále konfigurace - příkaz čte konfigurační údaje slavu uloženou v EEPROM paměti masteru (I/O konfigurace, ID kód, rozšířený ID1,2 kód). Adresa slavu a typ adresace jsou předány s voláním funkce.

Request								
byte	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	26_{16}							
2	T	–	circuit					
3	–		B	slave address				

Response								
byte	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	26_{16}							
2	T	result						
3	xID2				xID1			
4	ID				I0			

Obr. D.31: Vstupní a výstupní interface instrukce *GET_PCD*

- **Set list of projected slaves** (*SET_LPS*)
  Nastavení seznamu předpokládaných (plánovaných) slavů - tímto příkazem je seznam nakonfigurovaných slavů předán k trvalému uložení do paměti EEPROM masteru.
  Při provádění tohoto příkazu provede master warm restart. Příkaz může být použit pouze v konfiguračním módu.

Request (if O ≡ 0)								
byte	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	29_{16}							
2	T	0	circuit					
3	00_{16}							
4	7A	6A	5A	4A	3A	2A	1A	–
…	…							
11	31B	30B	29B	28B	27B	26B	25B	24B

Response								
byte	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	29_{16}							
2	T	result						

Obr. D.32: Vstupní a výstupní interface instrukce *SET_LPS*

- **Get list of projected slaves** (*GET_LPS*)
  Získání seznamu předpokládaných (plánovaných) slavů - tímto příkazem je seznam předpokládaných slavů přečten z masteru.

Request								
byte	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	44_{16}							
2	T	O	circuit					

Response (if O ≡ 0)								
byte	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	44_{16}							
2	T	result						
3	7A	6A	5A	4A	3A	2A	1A	0A
…	…							
10	31B	30B	29B	28B	27B	26B	25B	24B

Obr. D.33: Vstupní a výstupní interface instrukce *GET_LPS*

- **Store actual parameters** (*STORE_PI*)
  Uložení aktuálních parametrů - nakonfigurované parametry uložené v paměti EEPROM masteru jsou přepsány současnými, aktuálními parametry (I/O, ID, xID1, xID2) a trvale uloženy.

Request								
byte	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	04_{16}							
2	T	–	circuit					

Response								
byte	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	04_{16}							
2	T	result						

Obr. D.34: Vstupní a výstupní interface instrukce *STORE_PI*

- **Write parameter** (*WRITE_P*)
  Zapiš parametr - hodnota parametru je předána na slave, jehož adresa je vstupem funkce, tak jako samotný parametr. Parametr je v masteru uložen pouze dočasně a není uložen v jeho EEPROM paměti. Slave přenáší v odpovědi aktuální hodnotu parametru (parametr echo). Ta se může lišit od hodnoty, který byla právě zapsána dle specifikace masteru.

Request								
byte	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	02_{16}							
2	T	–	circuit					
3	–		B	slave address				
4	–			parameter				

Response								
byte	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	02_{16}							
2	T	result						
3	–			slave response				

Obr. D.35: Vstupní a výstupní interface instrukce *WRITE_P*

- **Read parameter** (*READ_PI*)
  Čtení parametru - vrací aktuální (skutečnou) hodnotu parametru ve slavu zaslané masterem. Tato hodnota nesmí být zaměňována s parametrem echo příkazu (*WRITE_P*). Tento příkaz nelze použít pro přímé čtení parametru ze slavu.

Request								
byte	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	03_{16}							
2	T	–	circuit					
3	–		B	slave address				

Response								
byte	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	03_{16}							
2	T	result						
3	–				PI			

Obr. D.36: Vstupní a výstupní interface instrukce *READ_PI*

- **Set permanent parameter** (*SET_PP*)
  Nastav trvalý parametr - nastaví hodnotu trvalého parametru zadanému slavu. Hodnota je trvale uložena v paměti EEPROM gateway. Parametr se přenáší pouze v případě, že je slave aktivní po zapnutí napájení masteru.

Request								
byte	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	43_{16}							
2	T	–	circuit					
3	–		B	slave address				
4	–				PP			

Response								
byte	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	43_{16}							
2	T	result						

Obr. D.37: Vstupní a výstupní interface instrukce *SET_PP*

- **Get permanent parameter** (*GET_PP*)
  Získej trvalý parametr - master čte specifický parametr slavu uloženou v paměti EEPROM masteru.

Request								
byte	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	01_{16}							
2	T	–	circuit					
3	–		B	slave address				

Response								
byte	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	01_{16}							
2	T	result						
3	–				PP			

Obr. D.38: Vstupní a výstupní interface instrukce *GET_PP*

- **Set auto address enable** (*SET_AAE*)
  Nastavení automatické adresace - tato volba může povolit nebo zakázat Automatické programování adres. Bit $AUTO_ADDR_ENABLE$ je uložen trvale, je zachován i po restartu masteru.

Request								
byte	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	$0B_{16}$							
2	T	–	circuit					
3	Auto_Address_Enable							

Response								
byte	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	$0B_{16}$							
2	T	result						

Obr. D.39: Vstupní a výstupní interface instrukce SET_AAE

- **Change slave address** ($SLAVE\_ADDR$)
  Změna adresy slavu - používá se hlavně k přidání nového slavu s výchozí adresou 0 na sběrnici. Jako parametry příkazu se předává stará adresa slavu (ve většině případů adresa 0) a nová adresa (s B bitem, který určuje změnu adresy z A-salvu na B-slave). Tato změna může být provedena pouze tehdy, jsou-li splněny následující podmínky:
  1. Musí existovat slave s původní adresou.
  2. Pokud stará adresa slavu není rovna 0, nelze připojit současně slave s adresou 0.
  3. Nová adresa musí být v platném rozsahu $(1 - 31/Anebo1 - 31B)$.
  4. Nesmí existovat slave, který již má plánovanou novou adresu.

  Když se změní adresa slavu, slave není resetován. Jeho výstupní data jsou zachována dokud nejsou na novou adresu přijata data nová.

Request								
byte	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	$0D_{16}$							
2	T	–	circuit					
3	–		B	source address				
4	–		B	target address				

Response								
byte	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	$0D_{16}$							
2	T	result						

Obr. D.40: Vstupní a výstupní interface instrukce $SLAVE\_ADDR$

- **Write AS-i slave extended ID1** ($WRITE\_XID1$)
  Zápis rozšířeného ID1 kódu - na slave s adresou 0 je zapsán přímo přes ASI kabel xID1. Příkaz je určen pro diagnostické účely a není nutné, aby byl master v normálním provozním módu. Master předává $xID1$ bez kontroly správnosti [10].

Request								
byte	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	$3F_{16}$							
2	T	–	circuit					
3	–				xID1			

Response								
Byte	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	$3F_{16}$							
2	T	result						

Obr. D.41: Vstupní a výstupní interface instrukce $WRITE\_XID1$

D.0.6 Ostatní příkazy

- $IDLE$
  Přepnutí do klidového stavu.

Request								
byte	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	00_{16}							
2	T	–	circuit					

Response								
byte	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	00_{16}							
2	T	result						

Obr. D.42: Vstupní a výstupní interface instrukce $IDLE$

- **Read input data image** ($READ\_IDI$)
  Čtení vstupních dat všech slavů na sběrnici mimo cyklickou výměnu dat. Tento příkaz přenáší i hodnoty kontrolních flagů.

Request								
byte	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	41_{16}							
2	T	–	circuit					

Response								
byte	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	41_{16}							
2	T	result						
3	–							Pok
4	OR	APF	NA	CA	AAv	AAs	s0	Cok
5	–				slave 1A			
6	slave 2A				slave 3A			
...	...							
36	slave 30B				slave 31B			

Obr. D.43: Vstupní a výstupní interface instrukce $READ\_IDI$

- **Write output data image** ($WRITE\_ODI$)
  Zápis výstupních dat všech slavů mimo cyklickou výměnu dat.

Request								
byte	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	42_{16}							
2	T	–	circuit					
3	–				slave 1A			
4	slave 2A				slave 3A			
…	…							
34	slave 30B				slave 31B			
Response								
byte	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	42_{16}							
2	T	result						

Obr. D.44: Vstupní a výstupní interface instrukce $WRITE\_ODI$

- **Read output data image** ($READ\_ODI$)
  Čtení výstupních dat všech slavů mimo cyklickou výměnu dat.

Request								
byte	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	56_{16}							
2	T	–	circuit					
Response								
byte	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	56_{16}							
2	T	result						
3	–				slave 1A			
	slave 2A				slave 3A			
…	…							
34	slave 30B				slave 31B			

Obr. D.45: Vstupní a výstupní interface instrukce $READ\_ODI$

- **Set offline mode** ($SET\_OFFLINE$)
  Tento příkaz přepíná mezi online a offline režimem. Online je normální provozní stav pro master. Následující úkony jsou prováděny cyklicky:

  Během fáze výměny dat je pole výstupních dat všech slavů přeneseno pro všechny slavy v LAS. Pokud byl přenos bez chyb, slavy předloží hodnoty svých vstupů masteru.

  Následuje fáze, ve které master vyhledává slavy a nově nalezené přidá do LDS a LAS.

  Ve fázi řízení (management) jsou uživatelovy příkazy zapsány a provedeny.

  V režimu offline přijímá master pouze výzvy od uživatele (věci, které potřebují okamžité vyřešení, např. adresace slavů, je odmítnuta s chybou). V tuto chvíli neexistuje žádná cyklická výměna dat se slavy.

  V režimu offline je sběrnice v "bezpečném stavu". OFFLINE bit (= 1) není nikde uložen a po restartu je master opět v online módu.

  pozn. Přepnutí do offline režimu je zapsáno v 3 bajtu ($log1$). Master se změní do režimu online, když je do třetího bajtu zapsána logická 0.

Request								
byte	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	$0A_{16}$							
2	T	–	circuit					
3	Off-Line							

Response								
byte	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	$0A_{16}$							
2	T	result						

Obr. D.46: Vstupní a výstupní interface instrukce *SET_OFFLINE*

- **Release data exchange** (*SET_DATA_EX*)
  Bitem *Data_Exchange_Active* se povoluje výměna dat na sběrnici.

Request								
byte	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	48_{16}							
2	T	–	circuit					
3	Data_Exchange_Active							

Response								
byte	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	48_{16}							
2	T	result						

Obr. D.47: Vstupní a výstupní interface instrukce *SET_DATA_EX*

- **BUTTONS** (*BUTTONS*)
  Příkazem se aktivuje/deaktivuje používání tlačítek [10].

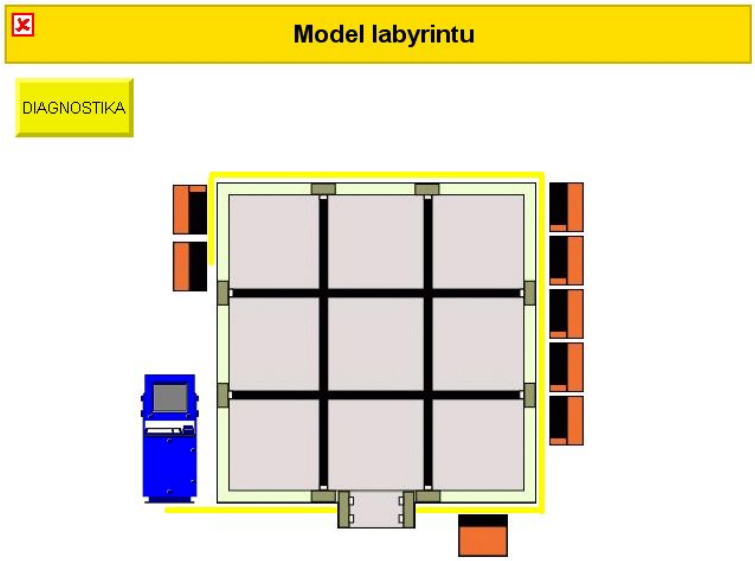
Request								
byte	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	75_{16}							
2	T	–	circuit					
3	Buttons disabled							

Response								
byte	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	75_{16}							
2	T	result						

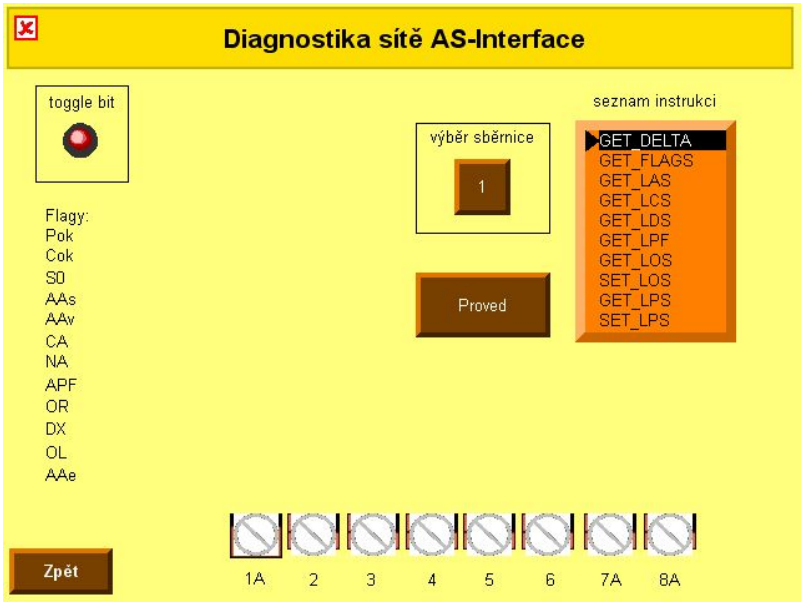
Obr. D.48: Vstupní a výstupní interface instrukce *BUTTONS*

E OBRAZOVKY VIZUALIZACE

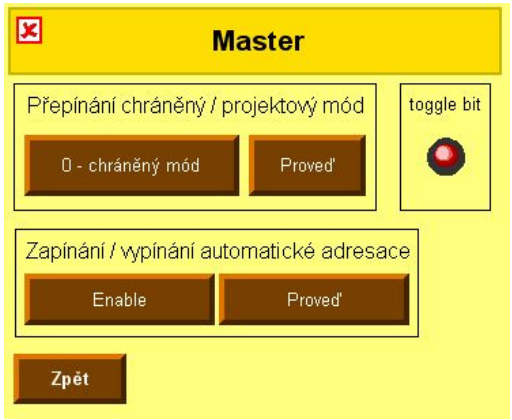
Obr. E.1: Úvodní obrazovka vizualizace.



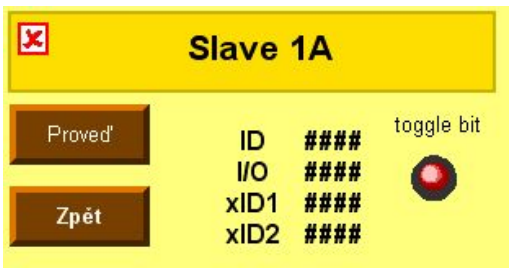
Obr. E.2: Obrazovka labyrintu



Obr. E.3: Obrazovka diagnostiky

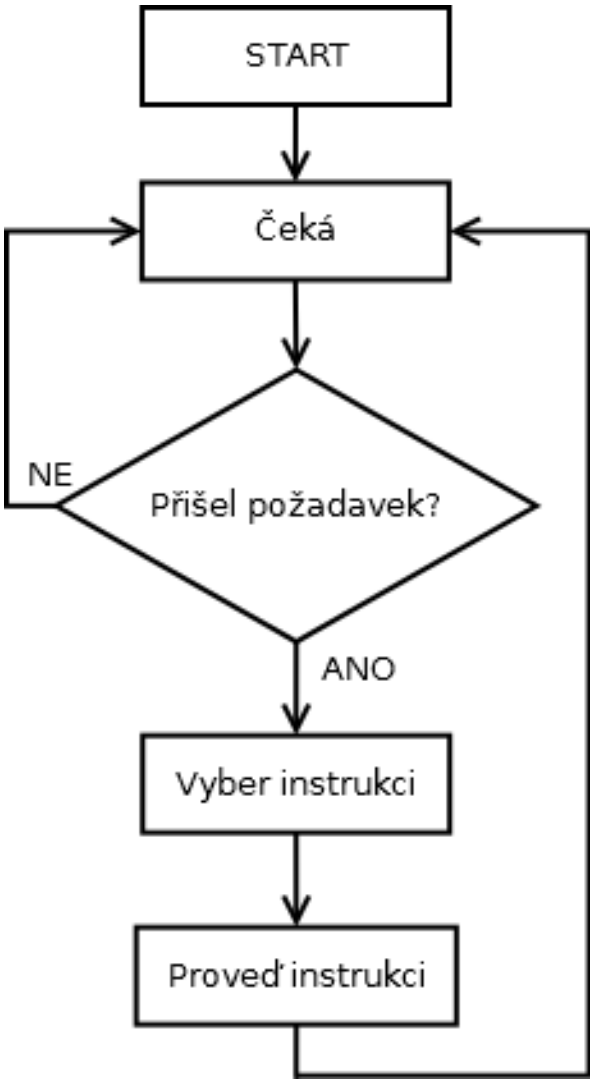


Obr. E.4: Obrazovka masteru



Obr. E.5: Obrazovka slavu

F VÝVOJOVÝ DIAGRAM PROGRAMU VIZUALIZACE



Obr. F.1: Vývojový diagram programu pro vizualizaci

G SEZNAM PŘÍLOH NA CD

Přiložené CD obsahuje tyto soubory:

1. Diplomovou práci v elektronické verzi
2. 48 vyexportovaných add-on instrukcí
3. Zdrojový kód programu vizualizace
4. Zdrojový soubor vizualizace
5. Reporty z programů a vizualizace
6. Datasheety k používanému hardwaru
7. Seznam instrukcí pro síť AS-Interface od firmy Bihl+Wiedemann