České učení technické v Praze
Fakulta strojní
Ústav přístrojové a řídící techniky

Řízení dopravníku v systému automatického rozvrhování výroby

Diplomová práce

Bc. Vojtěch Outlý

Magisterský program: Průmysl 4.0
Magisterský obor: Bez oborový
Vedoucí práce: Mgr. Ing. Jakub Jura, Ph.D.

Praha, červen 2019

Zadání

Vedoucí práce:
Mgr. Ing. Jakub Jura, Ph.D.
Ústav přístrojové a řídící techniky
Fakulta strojní
České vysoké učení technické v Praze
Technická 2
160 00 Praha 6
Česká republika

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně, s tím, že její výsledky mohou být dále použity podle uvážení vedoucího diplomové práce jako jejího spoluautora. Souhlasím také s případnou publikací výsledků diplomové práce nebo její podstatné části, pokud bude uveden jako její spoluautor.

V Praze 2019

.....
Bc.Vojtěch Outlý

Abstract

Abstrakt

Jako praktická část diplomové práce je vytvořeno řízení pro inteligentní dopravníkový systém, tedy automatizované ovládaní linky i vozíků a jejich monitorování v systému. Práce dokumentuje současný stav dopravníkového systému Montrac. Jedná se o jednokolejní automatizační linku, po které se pohybují vozíky s materiálem. Je zde popsána mechanická, hardwarová i komunikační část této linky. V práci je vysvětlen princip a postup konfigurace komunikace mezi zařízeními, jako jsou senzory, řídící jednotka a PLC, včetně ovládání linky. Linka je řízena softwarovým PLC Simatic S7-1500 od firmy Siemens a program je napsán v jazyce ST. Celé řízení je vytvořeno v vývojovém prostředí TiaPortal.

Klíčová slova: Automatizační linka, Softwarové PLC, Siemens, Montratec, Tia Portal, Jednokolejní dopravníkový systém

Abstract The practical part of this diploma thesis develops the intelligent system for automated control and monitoring of the carriage movements and system. This diploma thesis documents the conveyor system Montrac. It is a one rail automated line on which material transporting carriages operate. The work describes mechanical, hardware and communication aspects of the line. It also explains the theory and practice of configurating communications between various devices of the system, such as sensors, control unit and PLC. The program to control the line is written in the language ST, in a software environment PLC Simatic S7-1500 by Siemens. The overall management is created in development environment TiaPortal.

Keywords: Automation Line, Software PLC, Siemens, Montratec, Tia Portal, Singletrack Transport System

Poděkování

Rád bych touto cestou vyjádřil poděkování Mgr. Ing. Jakubovi Jurovi, Ph.D. za jeho cenné rady, doporučení a trpělivost při vedení mé diplomové práce. Taktéž bych chtěl poděkovat Ing. Pavelu Burgetovi, Ph.D. za vstřícnost, ochotu a pomoc při získání potřebných informací a podkladů. Rád bych poděkoval také své rodině, přítelkyni a všem přátelům, kteří mě při vytváření této práce podpořili, a bez jejichž pomoci by nebylo možné práci dokončit.

Seznam tabulek

Seznam obrázků

Seznam zkratek

CAN Controller Area Network. 5, 8–10, 14, 19

CPU Central Processing Unit. 26

DB Data Block. 33

ERP Enterprise Resource Planning. 31

FB Function Block. 34

FBD Function Block Diagram. 26

FC Function. 34

FRAM Ferroelectric Random Access Memory. 11

HMI Human Machine Interface. 25, 30, 80

IL Instruction List. 26, 27

IoT Internet of Things. 31

IPC Industrial Computer. 28

IRM Intelligent Routing Modul. 5, 8–10, 19

ISM Intelligent Shuttle Modul. 11, 16

LD Ladder Diagram. 26

MDAC Montrac Data Acquisition and Control. 7, 12, 16, 17, 22, 23

MES Manufacturing Execution System. 31

OB Organization Block. 33

OPC UA Open Platform Communications Unified Architecture. xiii, 32, 76

PA Programovatelný automat. 25

PAC Programmable Automation Controller. 25

PLC Programmable Logic Controller. 25, 26

PLM Product Lifecycle Management. 31

PN/IE PROFINET/Industrial Ethernet. 28, 34

RAM Random Access Memory. 18

SCADA Dispečerské řízení a sběr dat. 80

SFC Sequential Function Chart. 26

SPS Speicherprogrammierbare Steuerung. 25

SSH Secure Shell Connection. 23, 65

ST Structured Text. 26, 27, 35

TC2-IRM TracControl 2 IRM. 5, 7, 12

TC2-Unit TracControl 2 Unit. vii, 5, 7–10, 13, 17, 19, 20, 23, 37, 58, 61

TCP/IP Transmission Control Protocol/Internet Protocol. 12, 15

TiaPortal Totally Integrated Automation Portal. 30

UDP User Datagram Protocol. xii, 7, 12, 15

Obsah

Kapitola 1

Úvod

Automatizace a robotizace je jedním z nejvíce se rozvíjejících odvětví v průmyslu. Stále více se firmy snaží monitorovat své výrobky, aby se mohla výroba optimalizovat a tím se i zrychlit. Výrobky často mají QR či čárový kód, přes který se výrobek identifikuje v systému, a různá čidla nebo kamery mohou tato data přenášet do řídícího sytému. Poté jde lehce sledovat polohu daného výrobku a optimalizovat průběh výroby. Tyto automatizované linky často spolupracují s roboty a dochází ke vzniku celého systému.

Hlavním tématem této diplomové práce je vytvoření řízení dopravníkové linky programovatelným automatem, včetně popisu mechanického stavu linky. Systém se nachází v budově CIIRC v místnosti Tesbedu pro Průmysl 4.0. Jedná se o jednokolejný dopravníkový systém, po kterém se pohybují vozíky. Toto pracoviště spolupracuje s dalšími čtyřmi roboty. Linka musí být schopná od vyslání vozíku dostat vozík k určenému stanovišti bez operátorské pomoci. Linka je řízena pomocí softwarového PLC S7-1500 Software Controlle přes průmyslový počítač CPU1515SP IPC, obojí od firmy Siemens.

V teoretické části je jako první představen dopravníkový systém a následně popsána jeho hardwarová část, což zahrnuje senzoriku, řídící jednotku a vozíky. Dalším bodem je popis komunikace mezi jednotlivými komponenty, která je zásadní pro správné fungování linky. Následně je popsán komunikační protokol MDAC vyvinutý firmou Montrac, která linku vyrobila. Tento protokol slouží k posílání příkazů do vozíku a modulů. Také je popsáno webové rozhraní WebGui, přes které je možno linku částečné ovládat a konfigurovat.

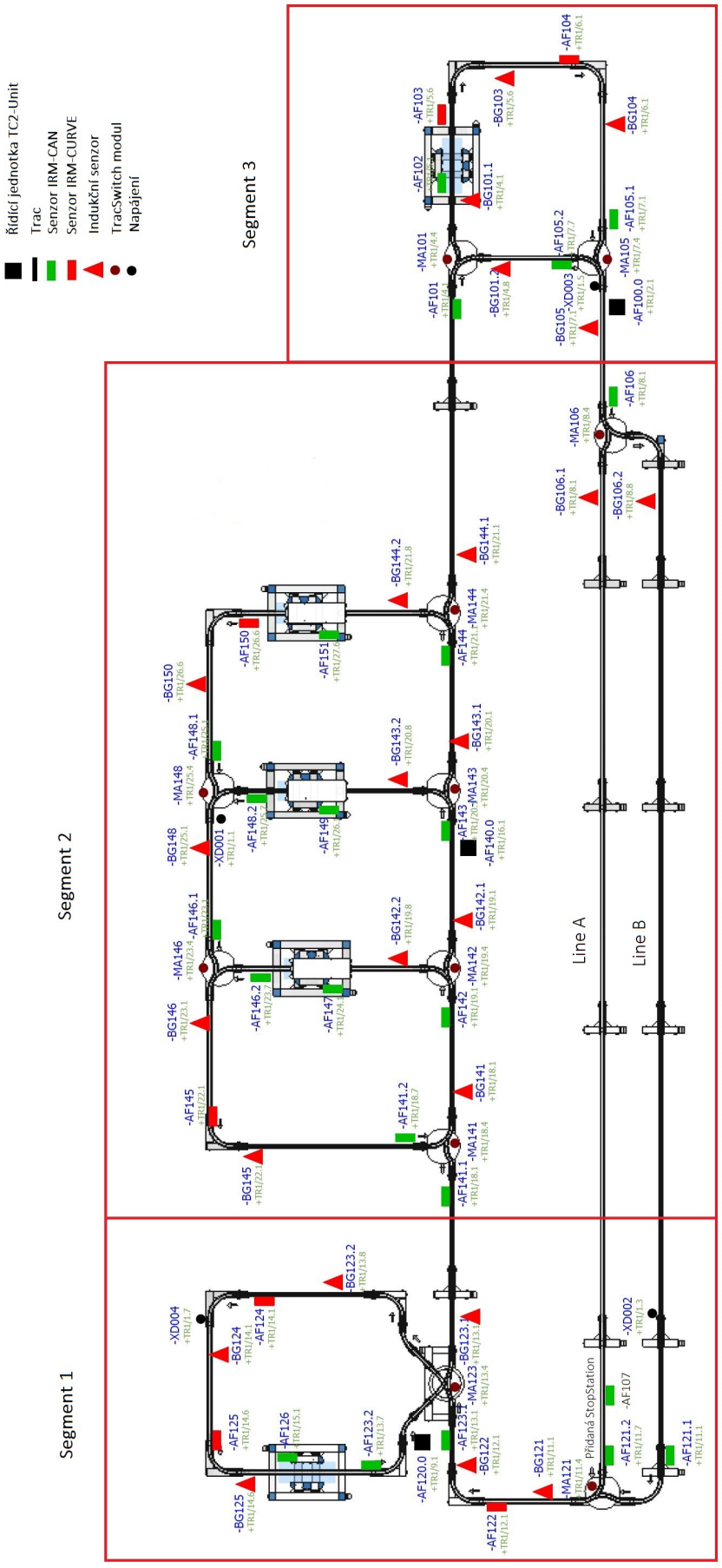
V další kapitole diplomové práce je stručně seznámeno s programovatelnými automaty a obecnými principy jejich činnosti, využitím a s vývojovým prostředím TiaPortal.

Hlavní bod diplomové práce, tedy praktická část, slouží k popisu ovládání linky. Jsou zde popsány všechny mnou vytvořené funkční a datové bloky. Tato část slouží i jako návod pro ovládání linky a případné přidávání komponentů či vozíků. První část popisuje připojení PLC a vytvořených funkční bloků. Dále je popsáno samotného ovládání linky pomocí datových bloků. Posledním bodem je popis lokálního operačního panelu.

Kapitola 2

Představení dopravníkového systému

V této kapitole jsou popsány všechny mechanické prvky, ze kterých je linka postavena, včetně vozíků pohybujících se po lince. Dále je vysvětlena komunikace mezi PLC řídící jednotkou a ostatními prvky, které jsou obsaženy v dopravníkovém systému. Také je zde popis speciálně vyvinutého komunikačního protokolu Montrac Data Acquisition and Control, pomocí kterého lze ovládat vozíky a moduly na lince. V neposlední řadě je zde také popis hardwarové konfigurace a popis softwarového PLC řídící linku. Je vysvětleno fungování samotného programovatelného automatu. Na obrázku 2.1 je vidět půdorys celé linky, v rámci této kapitoly je postupně popsán.



Obrázek 2.1: Půdorys celého dopravníkového systému (upraveno z [1])

2.1 Mechanická část dopravníku

Linka je složena ze systému vodících prvků (Trac), znázorněno na obrázku 2.2, po kterém jezdí vozíky. Koleje jsou vyrobeny z bezbarvého eloxovaného hliníkového profilu tyče. V této koleji jsou zabudovány vodiče, přes které linka získává energii. Tyto vodiče se mohou nacházet vlevo nebo vpravo na koleji, aby se co nejlépe mohly umisťovat řídící jednotky a senzory. Dopravníkový systém je tvořen z několika modulů, slouží k tomu, aby linka měla požadovanou funkci a tvar. Tyto moduly jsou podrobně popsány v kapitole 2.1.3. Dále se linka skládá z komponentů TracLink, což je spojovací prvek mezi dvěma sekcemi Trac nebo mezi moduly a také slouží jako podpěrný bod. Tento prvek zajišťuje elektrické připojení vodivých kolejnic a umožňuje tepelnou dilataci. Kolejnice mohou měnit směr pomocí prvků TracCurve, které mají různý poloměr. Tato zakřivení mohou mít různý poloměr. Celý dopravníkový systém je postaven na hliníkových rámech, které jsou přibližně jeden metr nad zemí. Jedná se o extrudovaný profil z hliníkové tyče dle DIN 17615. Celému dopravníkovému systému je dodávána do kolejnic elektrická energie z rozvaděče TracSupply proudem 24 V DC, který je vybavený hlavním vypínačem, osvětleným spínačem pro zapnutí napájení a spínačem pro vynutí napájení. Tato linka splňuje antistatické požadavky a směrnice v souladu s normami EN61000-6-2: 2005 (Elektromagnetická kompatibilita) a EN61000-6-4: 2011 (Elektromagnetická kompatibilita). Podél celé linky je také umístěno 5 stop tlačítek, které vypnou přívod energie do linky a způsobí její okamžité zastavení. Na fotografii níže 2.3 je zdokumentováno rozložení dopravníkového systému spolu s kolaborativními roboty. [2]

Obrázek 2.2: Fotka prvku Trac [2]

Obrázek 2.3: Fotka dopravníkového systému (vlastní tvorba)

2.1.1 Použitá senzorika

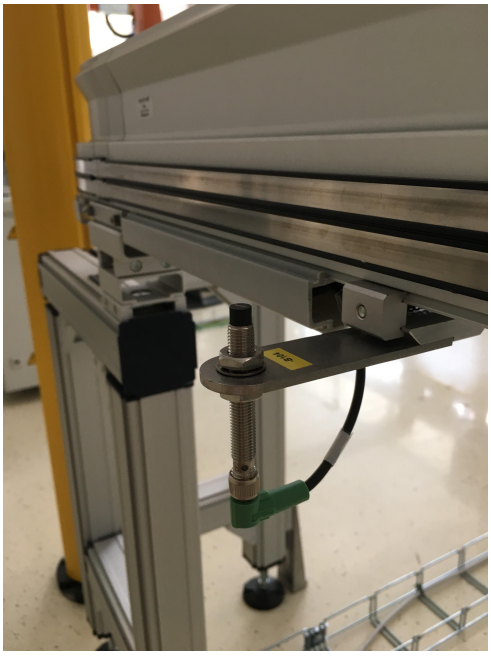
Pro správné fungování linky je důležité detekovat přítomnost vozíku s výrobkem na lince a to před každou zatáčkou a modulem tak, aby mohla být předána informace o přítomnosti vozíku řídící jednotce a také nedocházelo ke kolizím vozíků v zatáčkách nebo v modulech. Tyto funkce v lince vykonávají dva typy snímačů - bezkontaktní indukční snímač a optoelektronický snímač od firmy Montrac.

TracControl 2 IRM (TC2-IRM) znázorněný na obrázku 2.4 je optoelektronické čidlo s programovatelnými ovládacími prvky. Každé TracControl 2 čidlo je připevněno na vodící systém tak, aby byla umožněna vzájemná komunikace s vozíkem přes infračervené rozhraní. Čidlo TracControl 2 se v systému vyskytuje ve variantě komunikující s řídící jednotkou a zprostředkovávající výměnu dat mezi vozíkem a řídící jednotkou (označeno jako TC2-Unit-IRM-CAN, ve schématu 2.1 zelený obdélník) nebo ve variantě umístěné před TracCurve prvky, která zabraňuje průjezdu vozíku (a případné kolizi vozíků) v případě sepnutého indukčního čidla Sign-off umístěného za TracCurve (označováno jako TC2-Unit-IRM-CURVE), ve schématu 2.1 červený obdélník). Druhá varianta tedy komunikuje pouze se Sign-off indukčními čidly. Každé toto čidlo musí být uvedeno do webového rozhraní WebGUI a poté se mu v tomto rozhraní musí přiřadit dané funkce (tedy typ modulu, popsáno v kapitole 2.1.3), čímž se také určí, o kterou ze dvou variant se jedná. Čidlo je napájeno přímo z vodící linky a to 24V DC. Čidla komunikují z řídící jednotkou přes sběrnici CAN. [2]

Obrázek 2.4: Ukázka optoelektronického čidla IRM [2]

Bezkontaktní indukční snímač nebo-li Sign-off-sensor znázorněný na obrázku 2.5 (ve schématu 2.1 červený trojúhelník) je vložen za každý prvek TracCurve nebo jiný modul. V případě průjezdu vozíku Sign-off senzor detekuje kovový materiál a vysílá digitální signál do PLC (rozsvícené diody na senzoru) a signál se vypne po přejetí vozíku. Tento signál tedy potvrzuje uvolnění trasy pro následující vozík, který může pokračovat dále (vjezd do zatáčky či modulu) do vypnutí indukčního snímače (dioda nesvítí). Toto čidlo je napájeno přímo z vodící linky a to 24V DC. [3]

Spíše to bude tak, že indukční senzor posílá digitální signál do PLC a navíc to indikuje i diodou (pro kontrolu, ladění, diagnostiku).

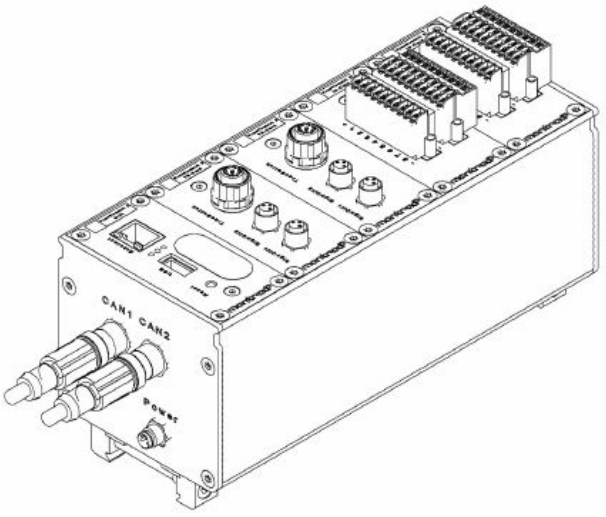


Obrázek 2.5: Ukázka optoelektronického čidla Sing-off (vlastní tvorba)

2.1.2 Řídící jednotka - TracControl 2 Unit

Celá linka je řízena pomocí centrální řídící jednotky systému TracControl 2 Unit (TC2-Unit) (znázorněna na obrázku 2.6, ve schématu 2.1 černý čtverec). K této řídící jednotce musí být připojeny všechny senzory TC2-IRM (maximální možný počet připojených senzorů na jednotku je 20). TC2-Unit komunikuje se senzory pomocí CAN sběrnice a pomocí ethernetové připojení jednotka komunikuje s PLC nebo průmyslovým počítačem. Pro čtení a ovládání TracControl 2 Unit je vyvinut specifický protokol Montrac Data Acquisition and Control (MDAC) založený na UDP protokolu, pro který je hlavním požadavkem ethernetové spojení. Řídící jednotka má také vestavění operační systém Linux. Protokol MDAC je popsán níže. Jednotka TC2-Unit má k dispozici několik různých slotů kare pro vložení přepínačů nebo digitálních vstupních / výstupních karet. Její detailnější specifikace se můžeme dočíst v [4].

Na této lince jsou použity tři řídící jednotky TracControl 2 Unit. Celá linka je tedy rozdělena do tří segmentů (červeně naznačeno ve schématu 2.1), kdy každý ze segmentů může být řízen nezávisle. Jak čidla komunikují s jednotlivými jednotkami je podrobněji popsáno v praktické části. Každá z jednotek má svoji unikátní IP adresu a také svoje unikátní NodeID a ModulID, přes které se s jednotkou dá komunikovat. Každá z těchto jednotek musí mít také svůj unikátní komunikační Local Port a Remote Port pro komunikaci s MDAC rozhraním. Local port je lokální adresa komponenty TCON, sloužící pro komunikace s PLC. Každá řídící jednotka musí mít nastavenou jinou hodnotu. Remote port je port dané řídící jednotky a portu protokolu MDAC. Tyto porty musí být rozdílné, pokud jedno PLC komunikuje s více TC2-Unit jednotkami. Tyto hodnoty je možné snadno změnit ve webovém rozhraní WebGUI.



Obrázek 2.6: Ukázka optoelektronického čidla IRM [2]

2.1.3 Moduly dopravníkového systému

Moduly dopravníkového systému jsou funkční seskupení jednotlivých komponent (Trac, Sign-off senzorů, TC2-Unit-IRM-CAN) a udávají funkční a strukturní vlastnosti linky, např. výhybky, křižovatky a StopStation, kde vozík může zastavit, ale existují i další moduly jako je například Lift (výtah). Všechny dosud dostupné moduly jsou ukázané v dokumentu [2]. V nadcházející kapitole je vysvětlen obecný princip fungování modulů a dále jsou popsány jednotlivé typy modulů použité v rámci této linky (TracSwitch divide, TracSwitch collect, TracSwitch arena a StopStation modul).

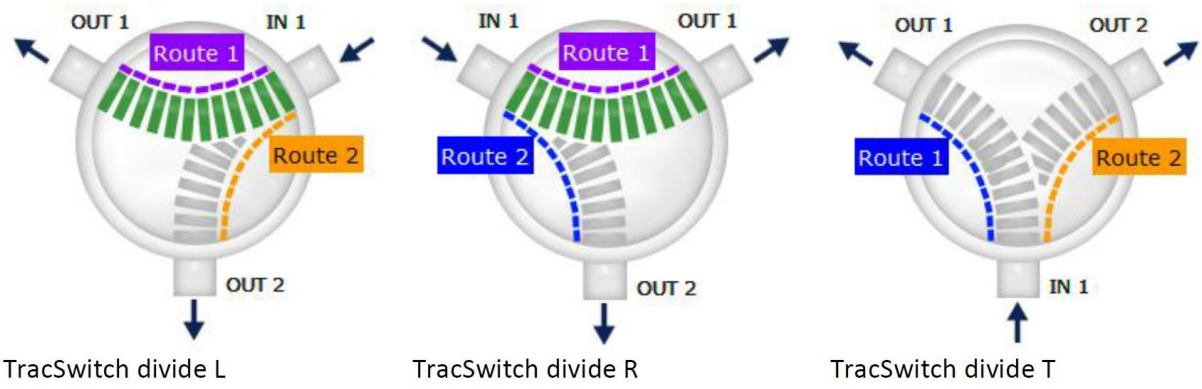
Pro zajištění směrování vozíku požadovaným směrem při průjezdem modulu je zásadní, aby nastavení modulu odpovídalo informacím o daném vozíku. Získání těchto informací umožňuje TC2-Unit-IRM-CAN čidlo umístěné před každým modulem a je tak zásadní pro správnou konfiguraci modulů (např. natočení výhybky). Zároveň se za každým modulem nachází čidlo Sing-off, aby nevznikla kolize vozíků. Moduly jsou mezi sebou propojeny kolejnicovým systémem.

Vozík, který přijíždí na daný modul, oznámí svou přítomnost prostřednictvím TC2-Unit-IRM-CAN čidla do řídící jednotky TC2-Unit. Řídící jednotka TC2-Unit poté přenáší tuto událost přes rozhraní TC2-Unit funkcí Shuttle Arrival a rozhoduje se o dalším postupu vozíku na základě přijatých parametrů vozíku, což jsou Shuttle ID a Target Address. Pokud je vozík na daném modulu, je jeho pozice oznámena proměnou Shuttle_Available. Následující proces se liší v závislosti na modulu a jeho konfiguraci. Např. přítomnost vozíku na modulu StopStation je signalizována funkcí Shuttle Processing (hodnota 1 znamená, že vozík komunikuje s jednotkou a předávají si mezi sebou data, v případě nulové hodnoty již došlo k přenosu), vjezd vozíku do modulu TracSwitch je signalizován funkcí Shuttle Transfer. Výjezd vozíku z jakéhokoliv modulu je poté hlášen funkcí Shuttle Departure. Pokud vozík z nějakého důvodu nemůže nebo nesmí vyjet z daného modulu, například je sepnutý indukční senzor, je na modulu hlášeno funkcí Shuttle Waiting (vozík čeká na modulu, dokud neobdrží příkaz k pohybu). Součástí všech modulů je také váleček pro mechanické zastavení vozíku. Tyto proměnné a princip fungovaní je pro všechny moduly stejný. Moduly obsahují tabulky tras - Chaos Table a Control Table, které rozhodují, kam má daný vozík jet (podrobně popsáno v kapitole 2.3.3).

Každý modul je ovládán elektricky a je napájen přímo z vodící kolejnice. Každý modul má různé in-links, tedy varianty tras příjezdu vozíku a out-links, varianty možných směrů výjezdu z modulu. Každý modul je možné lehce spravovat ve webovém rozhraní WebGUI, které je popsáno níže. Konfigurace nově přidaného modulu přes toho webové rozhraní je nutná pro jeho fungování.

TracSwitch divide modul

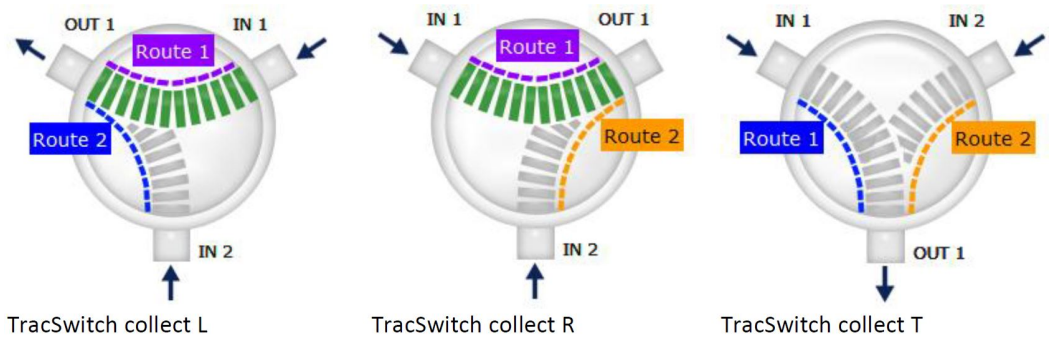
TracSwitch divide modul nebol-li distribuční přepínač je definován tím, že má pouze jednu vstupní cestu (in-link) a dvě výstupní cesty (out-links). Modul tedy komunikuje přes jeden TC2-Unit-IRM-CAN senzor a dva senzory Sing-off. Na základě informace z vozíku je směr výhybky nastaven pomocí Control a Chaos Table. . Varianty nastavení modulu jsou znázorněny na obrázku 2.7. [5]



Obrázek 2.7: Ukázka modulu TracSwitch divide [5]

TracSwitch collect modul

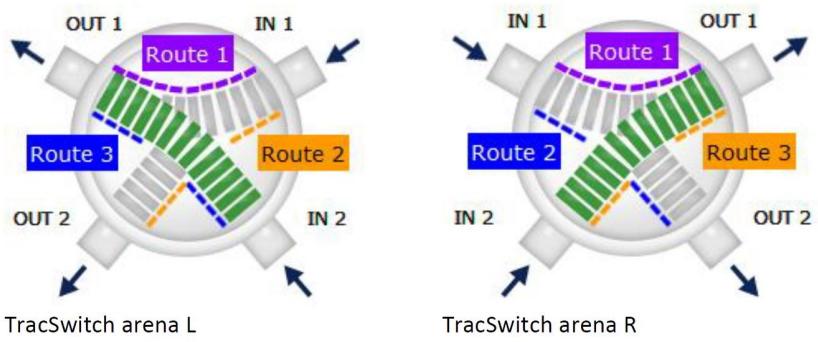
TracSwitch collect modul nebo-li spojovací switch je definován tím, že má dvě vstupní cesty (in-links) a pouze jednu výstupní cestu (out-links). Tento modul má na obou in-links TC2-Unit-IRM-CAN čidlo a u výjezdové cesty jeden Sing-off senzor. Jako standardní trasa pro obě in-links je nastavena jediná možná out-links a Control a Chaos Table tedy nejsou u tohoto modulu používány . Varianty nastavení modulu jsou znázorněny na obrázku 2.8. [5]



Obrázek 2.8: Ukázka modulu TracSwitch collect [5].

TracSwitch arena modul

Nejkomplexnějším modulem použitým v této lince je modul TracSwitch arena. Tento modul obsahuje dvě vstupní cesty (in-links) a dvě výstupní cesty (out-links). Ze směru IN 1 existují dvě varianty výstupních cest, musí se tedy použít Control nebo Chaos Table. Při vjezdu do modulu z cesty IN 2 je pouze jedna možná trasa výjezdu vozíku. Celkem tedy vozík může projet modulem třemi různými způsoby. Tento modul má TC2-Unit-IRM-CAN senzory na obou in-links a dva Sing-off senzory na výjezdových cestách. Varianty nastavení modulu jsou znázorněny na obrázku 2.9.[5]



Obrázek 2.9: Ukázka modulu TracSwitch arena [5].

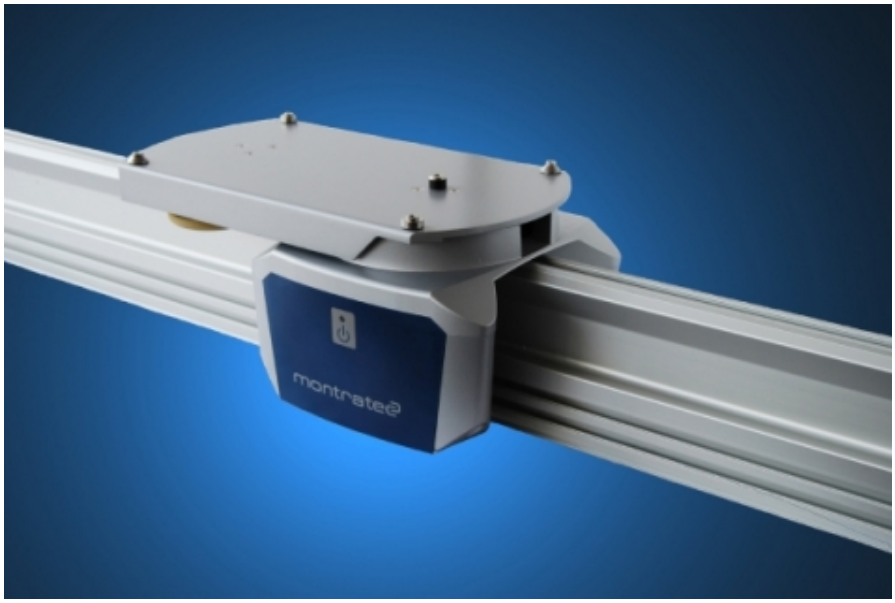
StopStation

Posledním modulem, který linka obsahuje je tzv. StopStation nebo-li místo zastavení. Tyto moduly jsou využívány k předání nákladu nebo informací vozíku a jeho setrvání v oblasti modulu do dalšího příkazu, nedochází k jeho samovolnému výjezdu. Tento modul se skládá z jednoho TC2-Unit-IRM-CAN čidla, které je umístěno vně StopStation. Tento modul má virtuální trasu, což znamená, že za modulem se nenachází žádné Sing-off čidlo, což je nutné zaznamenat při vkládání modulu do systému ve webovém rozhraní WebGui. K ovládání StopStation se používají Control a Chaos Tables (viz kapitola 2.3.3), vozík může pokračovat ihned po zastavení nebo může být zastaven, dokud nezíská signál. Tento modul obsahuje také píst sloužící k uzamčení pracovní desky na vozíku, vysunutí pístu znehybní desku na vozíku a usnadní např. snímaní desky kamerou. Tento modul můžeme vidět na obrázku 2.10. [5]

Obrázek 2.10: Ukázka modulu StopStation [2]

2.1.4 Vozíky

Vozíky slouží k manipulaci s materiálem a jsou hlavním prvkem na lince. Vzhledem ke svému speciálnímu tvaru jezdí na jednokolejní vodící dráze, kde osa vozíku je na ose vodící dráhy. Jsou poháněny bezúdržbovým nízkonapěťovým elektromotorem, umístěným v kyvadlové nápravě vozíku a je napájený přímo z linky. Vozík lze zapnout a vypnout tlačítkem a pomocí vypnutí přívodu energie je možné jej vypnout a vyjmout z dráhy. V našem případě jsou k dispozici vozíky pouze s jednou hnací nápravou, ale vyrábějí se i s dvěma hnacími nápravami. Vozík obsahuje infračervený port, přes který komunikuje se senzorem TC2-IRM. Maximální rychlost vozíku činí 55 metrů za minutu a minimální rychlost je c metry za minutu. Maximální hmotnost nákladu vozíku je 30 kg a výrobek by neměl být větší než 130 mm. Každý vozík má senzorový systém, nacházející se na přídi každého vozíku a umožňuje detekci dalšího vozíku nebo překážky před vozíkem a následné zastavení. Největší vzdálenost, kterou senzor dokáže zachytit je 450 mm, vozíky potom zastaví přibližně 100 mm od překážky. Vozík obsahuje Intelligent Shuttle Modul (ISM), což je pevná paměť vozíku ve formě 8 bajtového hodnoty. Když je vozík vybaven systémem TracControl 2, zahrnuje 4096-bytový FRAM, do které lze zapsat libovolné údaje [6]. FRAM je nevolatilní paměť s přímým přístupem, která dokáže uchovat data i po vypnutí napájení [7]. Vozík můžeme vidět na obrázku . Každý vozík může obsahovat pracovní desku a je reprezentován svoji Shuttle ID a vlastní Target Address, viz kapitola 2.3.



Obrázek 2.11: Ukázka vhledu vozíku [6]

2.1.5 Rychlost vozíku

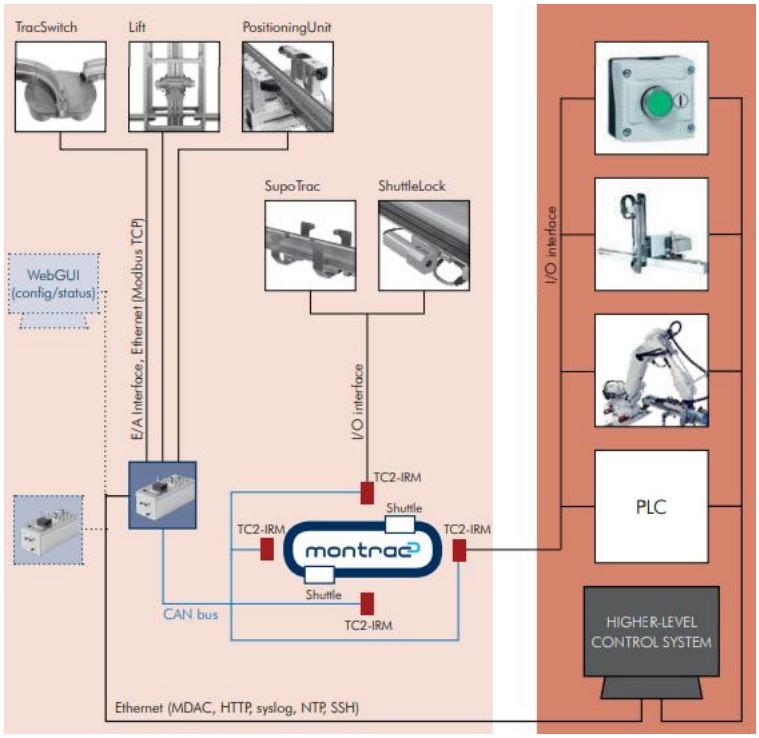
Rychlost vozíku se dá nastavit dvěma způsoby, a to nastavením rychlosti vozíku u senzorů TC2-IRM pomocí webového rozhraní WebGui (viz kapitola 2.3.6) nebo pomocí válečků, které se připevňují na Trac. Ve webové rozhraní se dají nastavit vozíku čtyři možné rychlosti: Rychlost AB je nejpomalejší, B je prostření rychlost, C je nejrychlejší a Default je nastaveno na rychlost B. Touto rychlostí vozík opustí čidlo IRM po vzájemné komunikaci. Také se dá řídit podle válečků, umístěné na lince. Pokud je váleček umístěn kolmo k lince, rychlost se po přejetí vozíku zvýší (pokud je kámen na vyšší drážce) nebo sníží (kámen je na nižší drážce). Pokud je kámen umístěn rovnoměrně s linkou, vozík se po přejezdu zastaví. Rychlost se bohužel nedá nastavit posláním příkazu do daného senzoru s PLC. Také jde zadat vozíku rychlost pomocí zapsání do registru vozíku viz [5].

2.2 Komunikace

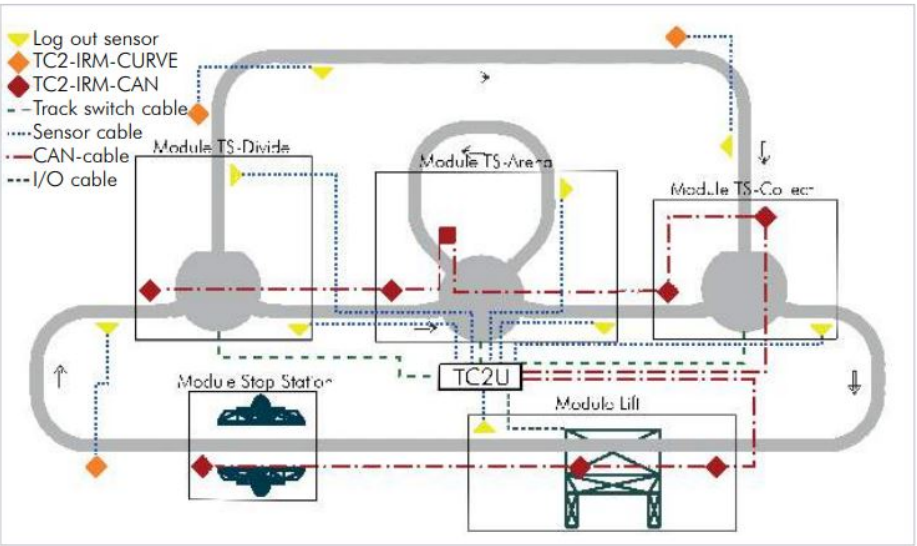
Moduly (čidla IRM) jsou připojené k řídící jednotce pomocí komunikační sběrnice CAN. Řídící jednotka je připojena s PLC pomocí ethernetové sběrnice Transmission Control Protocol/Internet Protocol (TCP/IP) a funguje na bázi UDP protokolu, který komunikuje přes speciálně vyvinutý komunikační protokol MDAC. Všechny protokoly jsou popsány níže. Sing-off senzor je spojen pouze pomocí I/O kabelu a to s čidlem IRM. Pomocí řídící jednotky TracControl 2 mohou být všechny kabely snadno a rychle připojeny k příslušným komponentům [2]. Podrobná specifikace komunikace UDP včetně funkčního rozsahu protokolu předávaného prostřednictvím protokolu MDAC je popsána níže. Celá

komunikace mezi jednotlivými komponenty je vidět přehledně na obrázku 2.12 nebo na obrázku 2.13.

Všechny protokoly jednotlivých modulů jsou odeslané přes syslog. Vestavěný operační systém Linux na TC2-Unit obsahuje BusyBox tím pádem i syslog, což je způsob, jak mohou síťová zařízení posílat zprávy o událostech na logovací server, známý jako Syslog server. Syslog je standard pro záznam programových zpráv a umožňuje oddělit software generující zprávy od systému, který je ukládá a softwaru, jenž poskytuje reporty a analýzy. Jedná se protokol typu klient/server. Tyto zprávy mohou být posílány pomocí User Datagram Protocol (UDP) nebo Transmission Control Protocol (TCP).[8] BusyBox je svobodný software, který poskytuje řadu unixových nástrojů v jediném spustitelném souboru. Například se dají používat příkazy jako ping [9]. Tímto způsobem jsou logovací zprávy ze všech modulů a TC2-Unit přenášeny přes rozhraní Ethernet a mohou být shromažďovány v serveru syslog. Pro komunikaci s jednotlivými čidly nebo řídící jednotkou se také dá použít klient PuTTY. Jak pracovat s tím klientem je popsáno v praktické části viz kapitola 3.4.3.



Obrázek 2.12: Ukázka komunikace mezi danými komponenty. [2]



Obrázek 2.13: Ukázka komunikace mezi danými komponenty. [2]

2.2.1 CAN sběrnice

Controller Area Network (CAN) sběrnice je sériový komunikační protokol umožnující distribuované řízení systému v reálném čase s vysokou mírou zabezpečení proti chybám. Jedná se o multi-master sběrnici. To znamená, že každý uzel může být master a řídit chování jiných uzlů. Tento protokol je definován nornou ISO 11898. Ta popisuje fyzickou vrstvu protokolu. Po sběrnici probíhá komunikace mezi dvěma uzly pomocí zpráv (datová zpráva a žádost o data), a management sítě (signalizace chyb, pozastavení komunikace) je zajištěn pomocí dvou speciálních zpráv (chybové zprávy a zprávy o přetížení). Zprávy vysílané po sběrnici protokolem CAN neobsahují žádnou informaci o cílovém uzlu, kterému jsou určeny, a jsou přijímány všemi ostatními uzly připojenými ke sběrnici. Sběrnice CAN je rozdělena do tří vrstev: CAN vrstvy objektů, CAN transportní vrstva a fyzická vrstva. Vrstva objektů a transportní vrstva zahrnuje veškeré služby a funkce poskytované v rámci linkové vrstvy, tak jak je definována referenčním modelem ISO/OSI, který se používá jako příklad pro komunikaci v počítačových a telekomunikačních sítích pomocí sedmivrstvého modelu, jednotlivé vrstvy jsou na sobě nezávislé jednotlivé vrstvy jsou popsané v další kapitole. Vrstva objektů je odpovědná za nalezení zprávy, která má být vyslána. Rozhodnutí, které přijaté zprávy od transportní vrstvy mají být použité a poskytování rozhraní aplikační vrstvě související s hardwarem. Transportní vrstva je především přenosový protokol. Například řízení rámců a signalizace chyb. Úkolem fyzické vrstvy je vlastní přenos jednotlivých bitů mezi jednotlivými uzly s respektováním všech elektrických vlastností. [10] [11]

2.2.2 Komunikační protokol TCP/IP

Transmission Control Protocol/Internet Protocol (TCP/IP)Jedná se o jeden z nejvíce používaných síťových protokolů. Komunikace probíhá v několika vrstvách, kde každá vrstva má svůj úkol podle referenčního modelu ISO/OSI. Referenční model ISO/OSI je sedmivrstvý. První čtyři vrstvy modelu (fyzická, linková, síťová a transportní vrstva) jsou vrstvy zaměřené na vlastní komunikační síť a zbylé vrstvy (relační, prezentační a aplikační) jsou orientovány na jejich aplikaci. Za zmínku v této práci stojí síťová a transportní vrstva. Úkolem transportní vrstvy je spolehlivý přenos dat v dané kvalitě a přidávání zdrojových dat do hlaviček každého paketu. Úkolem síťové vrstvy je zajištění síťové adresace a správné nasměrování dat. Protokol přijímá datové segmenty a přidává do nich svoji hlavičku a poté odešle data na cílové adresy. IP protokol ale nekontroluje, zda data, která odeslal došla v pořádku, kontrolní funkci zde plní protokol TCP (transmission control protocol). TCP/IP je založen na normě IEEE 802.3. [12]

2.2.3 User Datagram Protocol (UDP) protocol

UDP protokol (z anglického User Datagram Service) je protokol transportní vrstvy definovaný pro použití s protokolem IP adresy.

Služby protokolu UDP poskytuje minimální, nespolehlivý přenosí a žádné záruky doručení či zamezení duplikace datagramů, ale zato prokazuje nejvyšší možnou snahu pro doručení datagramu (anglicky best effort service). Datagram je základní jednotka informací, které je přepravována v počítačové síti. Není zaručeno ani dodržení pořadí doručení datagramů. Protokol je tedy jednodušší a snižuje režii pro přenos dat. Tato vlastnost je u některých aplikací vítaná, zejména při přenosu hlasu a videa, kde ztráta jednoho z datagramů příliš neovlivní výsledném kvalitu a opětovně odeslaný datagram by byl vzhledem ke zbytku přenosu neaktuální. Konfigurace síťového rozhraní je statická (statická IP adresa), v každém případě je přesná konfigurace IP realizována ve vztahu k celkovému projektu. Dva porty slouží k identifikaci koncových bodů ve zdrojovém a cílovém počítači. Proto se protokol UDP nepoužívá pro odeslání důležitých dat, např. databázové informace nebo webové stránky.[13] V porovnání s ostatními transportními protokoly, zejména TCP, je UDP nespojovaný protokol, to znamená, že nevytváří spojení mezi koncovými stanicemi před odesláním samotných dat. To opět snižuje režii přenosu a snižuje dobu odezvy (anglicky latency). Díky těmto vlastnostem poskytuje UDP velmi efektivní a rychlou komunikaci, což může vyhovovat některým aplikacím. [14]

2.2.4 PuTTY

PuTTY je emulátor softwarového terminálu pro Windows a Linux. Poskytuje textové uživatelské rozhraní ke vzdálenému ovládávaní počítačů používající kterýkoli z podporovaných protokol, včetně SSH a Telnet. [15] V našem případě se používá pro připojení k řídící jednotce a poté pro připojení na čidla IRM.

2.3 Protokol Montrac Data Acquisition and Control

Montrac Data Acquisition and Control (MDAC) je specifický komunikační protokol vyvinutý firmou Montrac pro komunikace mezi PLC a řídící jednotkou. Protokol umožňuje řídit a monitorovat celý systém linky. V této kapitole je vysvětleno jak tento protokol funguje a jak se sním pracuje. To znamená, ovládání vozíku a modulu na lince. Toto rozhraní je implementováno na základě UDP komunikace.

2.3.1 Shuttle ID a Target Address

Jedním z hlavních prvku v systému je vozík. každý vozík má svoje Shuttle ID a Target Address. Shuttle ID je unikátní označení vozíku, podle kterého je vozík reprezentován v systému. Target Address (někdy Target ID) je označení cíle což je StopStation, kam má vozík jet. Tyto dvě hodnoty jsou uloženy v Intelligent Shuttle Modul (ISM), což je pevná paměť vozíku ve formě 8 bajtového hodnoty. Tato hodnota se skládá ze čtyřbajtového ShuttleID a čtyřbajtové Target Address. Tyto hodnoty musí být nenulové a ShuttleID by také mělo být jedinečné. Rozložení bajtů můžeme vidět v tabulce 2.1. Tyto hodnoty mají tvar IP adresy například Shuttle ID 0.0.0.1). Tato hodnota se také zapisuje pomocí datového typu DWORD (Shuttle ID 16#0000_0001). Jak změnit ShuttleID je vysvětleno v praktické části.

Byte-Offset	0	1	2	3	4	5	6	7
Contents	Shuttle ID				Target address			

Tabulka 2.1: Rozloženi Shuttle ID a Target Address. [5]

2.3.2 Modul ID a Node ID

Každá řídící jednotka TC2-Unit má svoje unikátní Node ID číslo. Jedná se o jedinečné identifikační číslo v komunikačním uzlu, podle kterého lze poznat k jaké jednotce jsme připojeni. Modul ID je unikátní číslo každého IRM čidla, které je připojen k dané TC2-Unit řídící jednotce. Toto číslo se zadává do funkční bloku, pokud se připojujeme k řídící jednotce. Také podle Modul ID najdeme čidlo ve výkresové dokumentaci. Tato čísla je možno měnit v rozhraní Web GUI viz kapitola 2.3.6.

2.3.3 Control a Chaos Table

Pokud přijede vozík na čidlo, která má dva out-links (vybrat mezi Route 1 nebo Route 2), modul musí rozhodnout kam vozík pošle v případě StopStation se vozík zastaví. K tomu slouží Control Table a Chaos Table. Pokud moduly nemají žádné větvící prvky (pouze jeden out-links) jako je TracSwitch collect, tak jsou Control a Chaos Table vypnuty. Do těchto tabulek se zapisuji hodnoty ShuttleID nebo Target Address, podle který se rozhoduje, jak se daný modul bude chovat. Po příjezdu vozíku na čidlo IRM se prohledají tyto tabulky a podle toho se rozhodne jakým směrem pojede vozík dále. Tyto tabulky jsou prohledávány shora dolů. Jako prioritu má Control Table před Chaos Table. Pokud se v tabulkách nenajde vhodná hodnota, tak se modul nastavený podle výchozího nastavení a vozík jede dále.

V základním nastavení jsou tabulky prázdné a vozíky projíždějí StopStation a modulu se natáčení podle základního nastavení. Proto je nutné tyto tabulky vyplnit. Příklady příkazů jsou v tabulce 2.4.

V nastavení se mohou nastavit tři provozní režimy. První provozní režim 0 je Chaos + MDAC, tento režim prohledává jak Chaos Table, tak Control Table. V provozním režimu 1 MDAC jsou vyhodnocovány pouze Control Table tabulky. A v provozním režimu 2 Chaos jsou vyhodnocovány pouze tabulky Chaos Table.

Chaos Table

Chaos Table může mít 16 položek po 16 bajtech. Každá položka se skládá z 8 bajtového Shuttle ID a 8 bajtového Target Address. Můžeme si tedy zvolit rozsah ShuttleID - Shuttle ID a rozsah Target Address - Target Address. Tabulku je možno napsat jak pro Route 1 tak pro Route 2. Pokud přijede vozík se svým ShuttleID a Target Address a jedna s těchto hodnot bude v rozsahu v tabulce, vozík se tou cestou vydá. Tato tabulka se po provedení příkazu nevymaže. Různé typy rozsahů můžeme vidět v tabulce 2.2.

	Single ID / Range StartID								Range EndID							
Byte-Offset	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Single Shuttle	S	S	S	S	0	0	0	0	0	0	0	0	0	0	0	0
Shuttle Range	S1	S1	S1	S1	0	0	0	0	S2	S2	S2	S2	0	0	0	0
Single Target	0	0	0	0	T	T	T	T	0	0	0	0	0	0	0	0
Target Range	0	0	0	0	T1	T1	T1	T1	0	0	0	0	T2	T2	T2	T2

S = Shuttle ID, S1 = Shuttle Range Start ID, S2 = Shuttle Range End ID
T = Target Address, T1 = Target Range Start ID, T2 = Target Range End ID

Tabulka 2.2: Rozsah Chaos Table. [5]

Control Table

Control Table může obsahovat až 32 položek po 8 bajtech. Do této tabulky je možno zapisovat pouze jedno Shuttle ID, ke kterému je přiřazena pouze jedna Target Address. Tabulku lze zapsat pro Route 1 tak i pro Route 2. Pokud vozík přijede na čidlo IRM a této tabulce se najde shoda s ShuttleID nebo Target Address, vozík se vydá tou cestou. Po vykonání příkazu se tento jeden příkaz smaže. Control Table je uložena pouze v RAM paměti. Zápis do tabulky je vidět v tabulce 2.3.

	Shuttle ID / Target Address							
Byte-Offset	0	1	2	3	4	5	6	7
Single Shuttle	S	S	S	S	0	0	0	0
Single Target	0	0	0	0	T	T	T	T

S = Shuttle ID, T = Target Address

Tabulka 2.3: Rozsah Control Table. [5]

Příklady zápasu do Chaos Table:

V tabulce 2.4 je popsáno, kdy daný vozík zastaví ve StopStation kdy pojede dále.

Shuttle ID	Target ID	Popis
0.0.0.0 - 0.0.0.0	100.0.0.2 - 100.0.0.2	Vozík s Target ID zastavý, ostaní vozíky jedou dále
0.0.0.0 - 0.0.0.0	0.0.0.0 - 0.0.0.0	Všechny vozíky projedou
0.0.0.2 - 0.0.0.2	0.0.0.0 - 0.0.0.0	Vozík s Shuttle ID zastavý, ostaní vozíky jedou dále
0.0.0.2 - 0.0.0.2	100.0.0.2 - 100.0.0.2	Řídí se podle Shuttle ID

Tabulka 2.4: Příklady zápisu do Chaos Table (vlastní tvorba)

2.3.4 Funkce protokolu MDAC

Ke čtení a zapisovaní dat z řídících jednotek TC2-Unit a čidel TC2-Unit-IRM-CAN se používají funkce. Výjimkou je jen příkaz 70, sloužící k navázání spojení s řídící jednotkou, tento příkaz je popsán níže. V této kapitole jsou ukázány funkce, použité v této diplomové práci. Funkce se zapisují ve tvaru, který můžeme vidět v tabulce 2.5 a odpověď přijde ve tvaru 2.6.

Tyto funkce se musejí zapisovat do datového bloku TC2U_Prtc_SENDRECV (který je popsán v kapitole 3.3.8) a poté odeslat do řídící jednotky. Funkce, které jsou použity se automaticky posílají do tohoto datového bloku. Pokud bychom chtěli použít jinou funkci, tak postup je popsán ve zmíněné kapitole.

Celá funkce má velikost 256 bytu. Požadavek tvoří NodeID a ModulID, tím se vybere řídící jednotka a čidlo IRM, do kterého příkaz pošleme, popřípadě z jakého čidla dostáváme data. TransactionID je pouze informativní hodnota, kolikátý příkaz se provedl. Po každém příkazu se k této hodnotě přičte jedna. Na další pozici je vidět, zda byl daný příkaz proveden (1 - příkaz se provedl, 0 - příkaz se neprovedl). Na pozici 10. a 11. bytu je unikátní číslo funkce, která se má provést. Na pozici 14. a 15. bytu je zapsána velikost dat, posílána do jednotky nebo posílaná z jednotky. Na pozici od 16. do 256. bytu jsou zapsána data, posílaná do čidla, popř. jaká data dostáváme z čidla. Jednotka podle těchto informací rozhodne, co nám má poslat zpět. Všechny použité funkce v této diplomové práci jsou vidět v tabulce 2.7. Tento protokol má více funkčních příkazů, které se dají použít. Všechny příkazy a další funkce jde dohledat v [5].

Request																
Offset	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	as from byte 16
Contents	NodeID		ModuleID		TransactionID				0		3000		0		2		RouteNo

Tabulka 2.5: Posláni funkce do jednotky TC2-Unit. [5]

Answer																	
Offset	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	as from byte 16
Contents	NodeID		ModuleID		TransactionID				1		3000		0		256		Control table

Tabulka 2.6: Odpověď TC2-Unit funkce na příkaz. [5]

Funkce			
Název	Hodnota	Poznámka	Data Len
Clear_Shuttle_List	3002	Vymaže celou Control Table	2
Add_Shuttle_List	3003	Přidá novou hodnotu do Control Table	10
Del_Shuttle_List	3004	Vymaže první řádek Control Table	10
Get_Def_Shuttle_List	3100	Vypíše Chaos Table	2
Set_Def_Shuttle_List	3101	Vymaže a zapíše novou Chaos Table	258
Set_Host_Address	70	Připojení na řídící jednotku	Data

Tabulka 2.7: Tabulka použitých funkcí (vlastní tvorba)

Data do funkcí od 16 bitu:

Tyto data se zapisuji do **TC2U_Prtc_SENDRECV -> Send -> ST_Data**

Clear_Shuttle_List:

Na pozici bitu 2 se zapíše číslo cesty, kterou chci vymazat (1 - Route 1, 2 - Route 2).

Add_Shuttle_List:

Na pozici bitu 2 se zapíše číslo cesty, kterou chci vymazat (1 - Route 1, 2 - Route 2). Na další pozici se zapíše od bitu 3 - 6 ShuttleID a na pozici 7 - 10 se zapíše TargetID. Jde přidat i více příkazů na jednou, vždy ve stejném tvaru jako předchozí příkaz, ale musí se vždy napsat správná velikost Data Len, podle toho kolik posílám bitů.

Del_Shuttle_List:

Na pozici bitu 2 se zapíše číslo cesty, kterou chci vymazat (1 - Route 1, 2 - Route 2). A tím se vymaže jeden řádek.

Get_Def_Shuttle_List:

Na pozici bitu 2 se zapíše číslo cesty, kterou chci vymazat (1 - Route 1, 2 - Route 2). Pouze vypíše Chaos Table.

Set_Def_Shuttle_List:

Na pozici bitu 2 se zapíše číslo cesty, kterou chci vymazat (1 - Route 1, 2 - Route 2). Na další pozici se zapíše od bitu 3 - 6 ShuttleID a na pozici 7 - 10 se zapíše TargetID od bitu 11 - 14 ShuttleID a na pozici 15 - 18 se zapíše TargetID. Jde přidat i více příkazů na jednou, vždy ve stejném tvaru jako předchozí příkaz, ale musí se vždy napsat správná velikost Data Len, podle toho kolik posílám bitů. Pouze pro jednu cestu.

Zápis funkce přes TiaPortal:

Tyto funkce se zapisují do datového bloku **TC2U_Prtc_SENDRECV** (viz kapitola 3.16) a do proměnné **Send**. Proměnná ST_Header se vyplní podle hlavičky, která je výše. Do proměnné **ST_Data**, se napíšou data, které chceme poslat. Poté se tento příkaz může poslat do řídící jednotky, tento příkaz se dá poslat pouze přes funkční blok ve složce Function_Block. V této složce jsou funkční blok pro každou řídící jednotku (Connection_Block_TC2-Unit_151). V těchto blocích se dá odeslat tento příkaz do řídící jednotky pomocí proměnné **Send_UDT_StopStation**. V proměnné **Recv**, jsou data, které dostaneme jako odpověď.

Příkazy, které se pouze přijímají:

Pokud chceme komunikovat s řídící jednotkou, musíme se s ní nejdříve spojit. Toto se provede funkcí 70. Funkce poskytuje cílovou adresu asynchronních zpráv uzlu (NodeID). Žádost musí být vždy odeslána přímo na komunikační uzel (ModulID = 0). Musíme tedy použít NodeID pro připojení s danou řídící jednotkou. IP adresa Host-ID je 0 a tímto trvale ukládá. U všech ostatních identifikátorů Host-ID je adresa vyřazena, jakmile do zadaného časového limitu neprojde další ping, nebo uzel není restartován. Tato funkce se

automaticky provede při připojování řídící jednotky a musí se počkat, než se tento příkaz provede správně.

Pomocí protokolu MDAC dostáváme také informace o vozíku. Jedná se o funkce, které jsou pouze přijímací a nemůžeme je používat jako zapisovací funkce. Jejich vlastnost a tvar odpovědi, najdeme v tabulce 2.8.

Funkce			
Název	Hodnota	Poznámka	Data Len
Shuttle_Arrival	3200	Ohlášení příjezdu vozíku na StopStation	2
Shuttle_Departure	3201	Oznámení odjezdu vozíku se StopStation	10
Shuttle_Transfer	3202	Oznámení že touto stanici projíždí vozík	10
Shuttle_Processing	3203	Zpracování dat vozíku na modulu	2
Shuttle_Waiting	3204	Ohlášení, že vozík čeká na povolení jet	258

Tabulka 2.8: Tabulka odpovídacích funkcí (vlastní tvorba)

2.3.5 Chybové hlášky protokolu MDAC

V tabulce 2.9 je vidět chybové hlášky protokolu MDAC, které mohou vzniknout špatným zapsáním funkce nebo provedením funkce. Tyto chybové hlášky jsou vidět u každé řídící jednotky a to v datovém bloku **Connection_TC2-Units** v proměnné **os_MDAC_HeaderResult**

Result code	Value	Description
OK	0	The request was processed without error
Not_OK	1	General error when processing the request.
Not_supported	2	This function is not supported by node or module.
Mod_not_available	3	The addressed module is not available.
List_Size_not_OK	4	The length of the list does not concur with the specified length.
Param_not_OK	5	The request contains an invalid parameter.

Tabulka 2.9: Rozloženi Shuttle ID a Target Address. [5]

2.3.6 Webové rozhraní Web-GUI

Pro konfiguraci dopravníkového systému je vytvořeno webové rozhraní WebGUI, který je znázorněný na obrázku 2.14. Toto rozhraní jde jednoduše otevřít v jakémkoli webovém prohlížeči pomocí zapsání IP adresy vybrané řídící jednotky (IP adresy, použité pro řídící jednotky jsou 10.35.91.151, 10.35.91.152 a 10.35.91.153).

Pokud chceme toto rozhraní spravovat, musí se provést přihlášení pomocí hesla (hesla jsou níže), tyto hesla jsou poté v tomto rozhraní změnit. Toto rozhraní je navrženo pro dotykovou obrazovku, ale lze jej ovládat také lehce pomocí klávesnice a myši. V tomto rozhraní jde konfigurovat řídící jednotku TC2-Unit a všechny moduly, respektive jejich čidla IRM, připojena na danou řídící jednotku. Pro přístup k TC2-Unit je k dispozici Secure Shell Connection (SSH), aby bylo možné provést specifická nastavení linky nebo provádět aktualizaci softwaru. SSH je bezpečná šifrovací komunikace mezi dvěma počítači. Používá se např. pro ovládání počítače na dálku (v našem případě řídící jednotky). [16] Vzhledem k tomu, že TC2-Unit může být integrována do sítě, jde snadno realizovat vzdálenou údržbu. Pro spojení s řídící jednotkou TC2-Unit se také dá použít klient PuTTY, který je popsán v praktické části.

Hlavní obrazovka zobrazuje seznam všech nakonfigurovaných modulů, připojených k dané řídící jednotce. Pokud se přidává nový modul, musí se propojit přes kabel s řídící jednotkou a poté je třeba v tomto rozhraní vybrat daný modul a nakonfigurovat jej. Moduly jsou rozdělené do dvou kategorií, a to na moduly vyžadující zásuvnou kartu na řídící jednotce TC2-Unit (např. TracSwitch modul) a virtuální moduly, které tuto kartu nepotřebují (např. StopStation).

Konfigurovat daný modul a danou jednotku je možné až po přihlášení. V nastavení řídící jednotky lze editovat např. IP adresa dané jednotky, NodeID, kontrolovat připojení daného modulu, probliknout daný modul pro určení jeho lokalizace nebo nastavení serveru MDAC a počtu slotů na dané řídící jednotce. Po deaktivaci modulu je možno měnit jeho základní výjezdovou cestu, rychlost vozíku po projetí modulu, vzhled modulu, jeho in-links a out-links a rychlostní odezvu. Editovat jde také tabulky pro Control Table a Chaos Table, viz kapitola 2.3.3. Jak konfigurovat toto webové rozhraní lze najít v dokumentu [4].

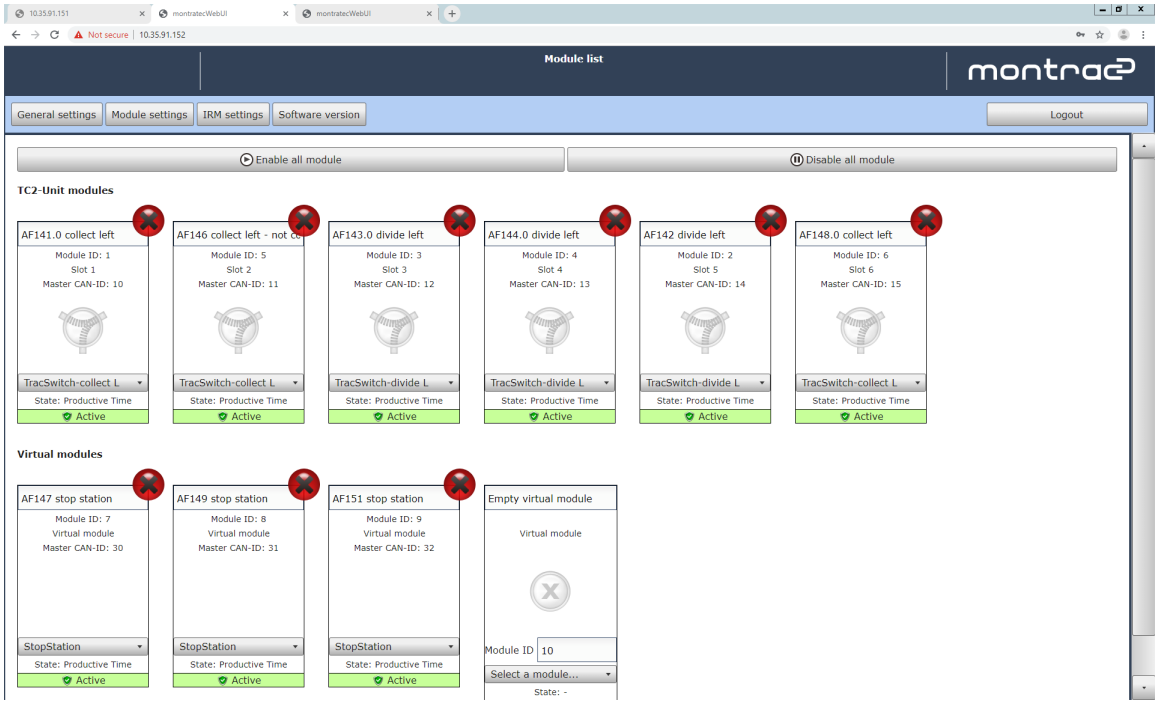
Popis ikon ve webové prostředí:

- **General settings** - Zde se dá konfigurovat řídící jednotku (NodeID, počet připojených slotů, heslo, IP adresa, Local Port, MDAC Port, Reboot TC2)
- **Modul settings** - Zde jsou všechny moduly připojený na jednotku. Jdou odebírat nebo přidávat

- **IRM settings** - Zde se dá kontrolovat zda jsou čidla IRM správně nakonfigurovaná a zda mají správně přiřazený modul
- **Software version** - aktuální verze softwaru

Přihlašovací údaje do WebGui:

- Řídící jednotka 10.35.91.151
  - Heslo: bez hesla
- Řídící jednotka 10.35.91.152
  - Heslo: CIIRC
- Řídící jednotka 10.35.91.153
  - Heslo: CIIRC



Obrázek 2.14: Ukázka webového prostředí WebGui (vlastní tvorba)

2.4 Hardwarová konfigurace

V této kapitole je popsána hardwarové konfigurace, která řídí chod celé linky. Primární řídící prvek celého dopravníkového systému je softwarově programovatelný automat S7-1500 Software Controller, spuštěný na průmyslovém počítači CPU 1515SP IPC, obojí od firmy Siemens. Dále jsou použity tři pneumatické terminály CPX Rev 20 od firmy Festo, které slouží k zamykání StopStation. Je zde popsáno, co je to PLC, softwarové PLC a průmyslový počítač a jaké programovací jazyky je možné použít pro programování PLC.

2.4.1 Programovatelný logický automat

Programmable Logic Controller (PLC) je do češtiny překládáno jako programovatelný logický automat, česká zkratka Programovatelný automat (PA). Někdy se také můžeme setkat s německým názvem Speicherprogrammierbare Steuerung (SPS). Někteří autoři také používají označení Programmable Automation Controller (PAC) jako označení pro moderní programovatelný automat. V poslední době se tyto automaty označují jako hybridní zařízení, která kombinují vlastnosti programovatelných automatů a průmyslových počítačů. Moderní PLC mohou např. komunikovat s frekvenčními měniči, distribuovanými vstupně/výstupními moduly, průmyslovými počítači, pohony nebo inteligentními senzory. Ke komunikaci s obsluhou se používají Human Machine Interface (HMI) nebo SCADA systémy, které slouží pro vizualizaci a ovládání daného stroje či linky. Pro komunikace mezi PLC a linkou či strojem se používají různé komunikační protokoly, jako jsou CAN sběrnice, MODBUS nebo PROFITNE a další.

Jedná se o řídící programovatelný systém určený pro řízení průmyslových a technologických procesů. Automaty využívají logické řízení, bitové operace, časovače, čítače, zpracování analogových a binárních signálů a matematické funkce. Program je ve většině případů vykonáván cyklicky, ale u některých automatů lze zvolit i sekvenční provádění programu. V jednom cyklu PLC načte vstupní data v podobě signálů z linky a jejich hodnota se uloží do registru. Následně je proveden uživatelský program na základě načtených vstupních dat v registrech a vnitřních datových proměnných. V této části uživatelský program pracuje pouze s obrazem vstupů uložených v registrech. Výsledek uživatelského programu je uložen do výstupních registrů. Posledním krokem je nastavení výstupu programovatelného automatu. Při vytváření uživatelského programu se vyžívají organizační bloky, které jsou vykonávány cyklicky, při startu PLC a při přerušení. [17]

Programovatelné automaty se používají v průmyslu z mnoha důvodů, především kvůli jejich vysoké spolehlivosti, snížení nákladů k fyzické realizaci a údržbě výrobního zařízení, vestavěné diagnostice a univerzálnosti. Další velkou výhodou je, že program může být velmi přehledně napsaný a PLC může být řízeno v reálném čase. Nevýhodou progra-

movatelných automatů jsou hlavně nižší programátorský komfort a nižší výpočetní výkon oproti průmyslovým počítačům. Další velkou nevýhodu je nemožnost upravovat hardware samotného základního modulu PLC a není tedy možné upravovat vlastní paměť zařízení, operační paměť ani Central Processing Unit (CPU). Proto je nutné přesně vybrat programovatelný automat odpovídající výkonem dané aplikaci. Hardwarové programovatelné automaty už obsahují CPU, ale lze k nim přidávat další moduly (např. PLC Siemens SIMATIC S7-1500) nebo se vyrábějí jako modulární. Dnes mezi hlavní výrobce programovatelných automatů patří Siemens, Beckhoff, Rockwell-Allen Bradley, ABB, Teco a další výrobci. [17]

Programovací jazyky pro PLC

Programovatelný automat lze naprogramovat v různých programovacích jazycích jako jsou Instruction List (IL), Structured Text (ST), Function Block Diagram (FBD), Ladder Diagram (LD) a Sequential Function Chart (SFC). Všechny tyto programovací jazyky jsou definované a sjednocené pomocí mezinárodní normy IEC EN 61131-3. Tato norma definuje syntax, sémantiku programovacích jazyků a architekturu PLC systémů. [18]

Programovací jazyky můžeme rozdělit dle způsobu programování na grafické a textové jazyky. U grafických jazyků se program vytváří pomocí spojování jednotlivých bloků, které představuji jednotlivé funkce. Mezi tyto jazyky patří LD, FBD a SFC. Textové programovací jazyky jsou tvořeny jen textem. Mezi textové jazyky patří IL a ST. Mezi tři nejpoužívanější jazyky patří jazyky LD, FBD a ST. [17] [19]

- **Ladder Diagram (LD)** nebo-li reléová schémata je grafický zápis programu. Tento způsob zápisu vychází z reléové logiky, kdy jsou propojována vzájemná relé. Svojí strukturou je LD ideální pro rychlé a přehledné zpracování velkého množství logických signálů a jejich čítání a časování.

- **Function Block Diagram (FBD)** nebo-li funkční bloky jsou grafickým programováním za použití bloků, pomocí nichž jsou tvořené různé funkce. Zápis je poměrně přehledný, logické operace mají podobu hradel. FBD je vhodný pro realizaci řídících sekcí programu a zpracování vstupních a výstupních signálů.

- **Sequential Function Chart (SFC)** nebo-li sekvenční je grafický zápis programu programu pomocí bloků, které mohou představovat konkrétní (pod)programy zapsané ve výše uvedených jazycích. Na základě splněných podmínek je rozvětvena struktura programu a program je vykonáván po krocích. Největší význam má při realizaci sekvenční logiky.

- **Structured Text (ST)** nebo-li strukturovaný text je programovací jazyk vyšší úrovně, podobně jako např. Pascal nebo C. Zápis je tvořen postupností instrukcí. Tento jazyk se používá při práci s daty, řetězci a databázemi a pro naprogramování složitých výpočetních algoritmů. Je tak vhodný pro zpracování analogových signálů. Důležitá je znalost příkazů a přesná syntaxe zápisu.

- **Instruction List (IL)** nebo-li instrukční list je nejzákladnějším zápisem programu. Nejvíce se podobá programování v Assembleru. Program se skládá z posloupností operací. Program tedy není náročný na výpočetní techniku, ale jsou velmi nepřehledné. Tento jazyk se už téměř nepoužívá.

Kromě těchto standardních jazyků definované normou nabízí někteří výrobci programovatelných automatů také vlastní jazyky nebo možnost programovat zařízení pomocí programovacích jazyků známých ze standardních osobních počítačů. Firma Siemens např. nabízí možnost k programování využívat grafický jazyk GRAPH. Firma Beckhoff nabízí programování pomocí jazyka C nebo jazyka C++.[20]

2.4.2 Softwarový programovatelný automat

V dnešní době se kromě čistě hardwarových verzí programovatelných automatů využívají i jejich softwarové verze, které jsou založeny na emulaci programovatelného automatu do počítače. Tyto softwarové verze lze většinou používat pouze na průmyslových počítačích, které vyrábí výrobce daného softwarového programovatelného automatu. Tyto automaty jsou určeny především pro aplikace, kde je potřeba řízení provádět v reálném čase, komunikovat s dalšími zařízeními, realizovat vizualizaci a zpracování velkého množství dat. Hlavní výhodou softwarové automatu oproti klasické verzi je, že na tomto automatu může být souběžně spuštěn uživatelský PLC program, vizualizační software a další standardní aplikace jako Matlab, Simulink, Microsoft Excel a další softwary. Softwarové verze programovatelných automatů většinou obsahují softwarový modul reálného času, který zajišťuje vykonávání PLC programu v reálném čase a tedy s minimální reakční dobou. Dále software obsahuje samotný softwarový automat a vizualizační software, který emuluje uživatelské rozhraní. Softwarová verze PLC se z pohledu programátora chová a programuje stejně jako v klasické hardwarové verzi a splňuje tedy normu IEC-1131.[18] Mezi hlavní výrobce těchto softwarových automatů patří například firma Siemens nebo Beckhoff. [17]

2.4.3 Průmyslový počítač

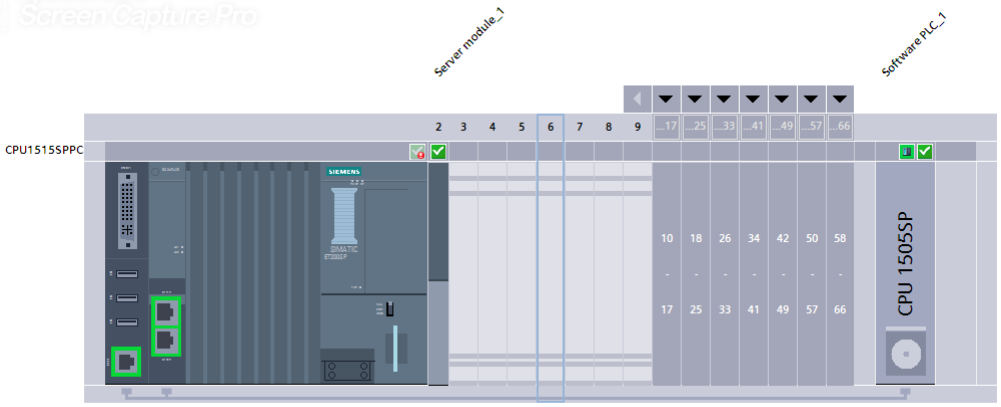
Industrial Computer (IPC) znamená v překladu průmyslový osobní počítač. Jedná se prakticky o standardní osobní počítač, ale s ohledem na jeho průmyslové použití musí tento počítač fungovat bez poruchy i v horších podmínkách, které mohou v průmyslových provozech panovat. Proto jsou průmyslové počítače stavěny tak, aby odolaly všem možným nepříznivým vlivům (např. otřesy, vibrace, elektromagnetické rušení, tepelné výkyvy a další extrémní vlivy), což se odráží i na dané konstrukci IPC. Proto jsou často tyto počítače dražší. IPC jsou většinou postaveny na 32-bitové PC platformě x86. Velkou výhodou IPC oproti PLC je možnost skloubit řízení s vizualizací a externí uživatelské aplikace na jedno zařízení. IPC se většinou používají pro aplikace jako jsou měření, kontrola probíhajících technologických procesů, vizualizační úkony (informační terminály) nebo sběr a zpracování dat z jednotlivých dílčích zařízení typu PLC.

V IPC se také využívají další přídavná zařízení, např. redundantní napájení, integrovaná diagnostika nebo vstupní a výstupní karty. Průmyslové počítače mají pouze omezený počet slotů pro připojení dalších karet a proto IPC často využívají distribuované moduly, které jsou připojeny k IPC pomocí komunikačního rozhraní. Na samotném IPC je obvykle spuštěn operační systém v reálném čase. Nejčastěji jsou používány operační systémy Linux nebo Windows. Tyto operační systémy jsou opatřeny jádry reálného času, minimalizující odezvu řídící aplikace. Na IPC lze nainstalovat softwarové PLC, SCADA systémy a další aplikace.

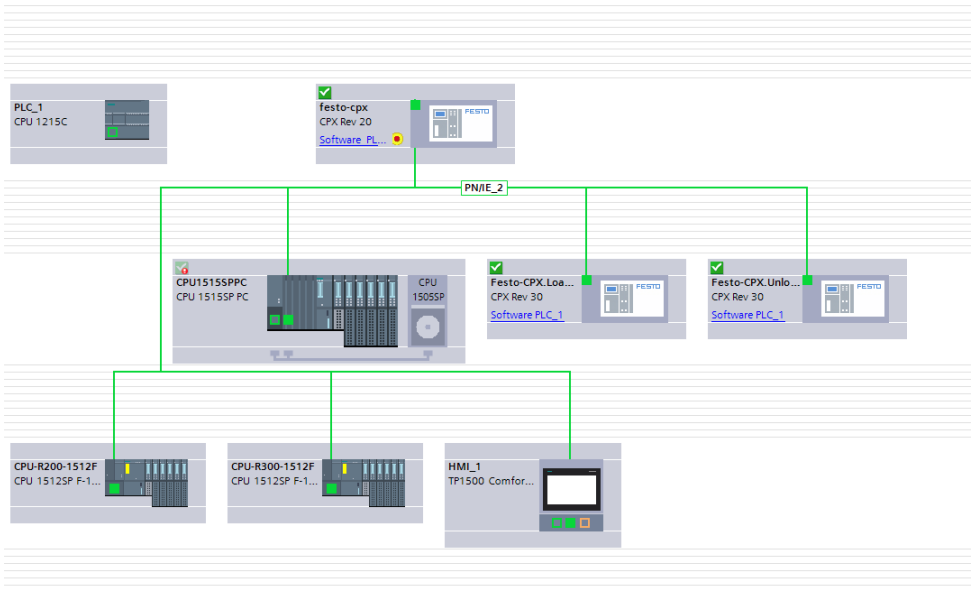
IPC se vyrábí ve variantách Rack PC, Box PC a Panel PC. Rack PC jsou určeny pro instalaci do rozvaděčových skříní. K tomuto typu IPC je nutno připojit externí monitor. Box PC jsou vyráběny formou klasické volně stojící počítačové skříně nebo formou krabice obdobného tvaru jako u programovatelných automatů. Tento typ není vybaven monitorem. Posledním typem IPC jsou Panel PC, které jsou vybaveny monitorem. Tento monitor je dnes často vyráběn jako dotykový. [17]

2.5 Softwarové PLC použité pro řízení linky

Primárním řídícím prvkem celého dopravníkového systému je softwarový programovatelný automat S7-1500 Software Controller. Softwarový programovatelný automat běží na CPU 1515SP IPC od firmy Siemens a obsahuje ještě Server module_1. Toto IPC a softwarové PLC je svým výkonem dostačující pro řízení této linky. Komunikace s ostatními komponentami (řídící jednotka TC2-Unit) probíhá přes rozhraní PROFINET/Industrial Ethernet (PN/IE) komunikaci, která je založena na UDP protokolu.



Obrázek 2.15: CPU na kterém běží softwarové PLC S7-1500 (vlastní tvorba)



Obrázek 2.16: Ukázka celé řídící konfigurace dopravníkového sytému (vlastní tvorba)

S7-1500 Software Controller je softwarové PLC, které vyvinula firma Siemens. Toto softwarové PLC musí být nainstalováno na IPC od stejného výrobce a nefunguje na běžném osobním počítači. Tento programovatelný automat běží na operačním systému Windows. S7-1500 Software Controller patří k nejnovější rodině PLC Siemens SIMATIC S7-1500 a doplňuje hardwarovou řadu programovatelných automatů, proto má toto softwarové PLC stejné vlastnosti a pracuje se s ním stejně jako s klasickým hardwarovým PLC. Jeho celou specifikaci lze vyčíst z tohoto manuálu [21]. Podobu tohoto PLC pomocí TiaPortalu můžete vidět na obrázku 2.15 a celou strukturu hardwarové sestavy na obrázku 2.16. Toto softwarové PLC je programuje z pohledu programátora obdobně jako klasické

hardwarové PLC rodiny S7-1500 a zařízení podporuje klasické programovací jazyky podle normy IEC EN 61131-3, které jsou popsány v kapitole Programovatelný logický automat 2.4.1. PLC může s dalšími zařízeními komunikovat pomocí komunikačního rozhraní PROFIBUS nebo PROFINET, v závislosti na typu rozhraní fyzicky umístěném na IPC. Pomocí těchto komunikačních rozhraní může softwarové PLC využívat externí zařízení jako distribuované vstupy nebo výstupy, hardwarová PLC, další IPC a další zařízení. Přes tyto komunikační rozhraní je možné dané PLC naprogramovat, sledovat nebo diagnostikovat pomocí vývojovým prostředím TiaPortal, STEP 7, WinCC nebo dalšími inženýrskými programy. Celý modul se nazývá ET 200SP Open Controller a jeho podrobná specifikace je obsažena v tomto manuálu [22].

Obrázek 2.17: Ukázka použitého Průmyslového počítače (vlastní tvorba)

2.6 Vývojové prostředí TiaPortal

Vývojové prostředí Totally Integrated Automation Portal (TiaPortal) se používá pro přehledné řízení PLC. Toto prostředí zahrnuje všechny nástroje potřebné k projektování a konfiguraci PLC systémů a uživatelských panelů Human Machine Interface (HMI) pomocí SIMATIC WinCC včetně rozvržení komunikace celých distribuovaných systémů nebo STEP 7. [23] Software je orientován na uživatelské rozhraní a zároveň na funkčnost, která zdržuje funkce při práci s automatizační technikou. U nástroje Step 7 si lze vybrat ze dvou variant: Step 7 Professional je software pro konfiguraci a programování všech Simatic kontrolérů, poskytuje možnost online diagnostiky pro celý projekt a obsahuje funkce pro PID regulaci. Umožňuje plánování sítě zařízení včetně nastavení komunikace. Druhou možností je Step 7 Basic, což je omezená verze Step 7 Professional a lze jej použít pouze pro kontroléry řady S7-1200. V této práci je použita verze Simatic Step 7 Professional V15.

2.7 Další prvky v konfiguraci

Dalšími prvky, zapojenými v tomto projektu jsou dvě PLC, CPU-R200 a R300 a slouží především jako záloha v případě poruchy hlavního IPC. V současné době se tedy nepoužívají. Dalším připojenými prvky jsou modulární elektrické terminály CPX Rev 20 od firmy Festo, které řídí pneumatické písty sloužící k zamčení desky na jednotlivých StopStation. Specifikace tohoto terminálu jsou obsaženy v [24].

2.7.1 Manufacturing Execution System

Manufacturing Execution System (MES), neboli Výrobní informační systém spojuje, monitoruje a řídí komplexní výrobní systémy a datové toky. Za účelem zefektivnění realizace výrobních operací a zlepšení produkce MES monitoruje a shromažďuje přesná data v reálném čase o celém životním cyklu výroby, počínaje vydáním zakázky až do fáze dodání hotového produktu. MES je často integrován společně s Enterprise Resource Planning (ERP). Existuje několik společností, které dodávají MES systém, mezi ně patří například IQMS, ABB nebo Schneider Electric SE. [25]

2.7.2 Enterprise Resource Planning

Enterprise Resource Planning (ERP) nebo-li plánování podnikových zdrojů je modulární softwarový systém navržený pro integraci hlavních funkčních oblastí podnikových procesů do jednotného systému. ERP zahrnuje často oblasti jako jsou finance a účetnictví, HR, výroba a správa materiálů, řízení vztahu se zákazníkem a řízení dodavatelského řetězce. ERP také dokáže sdílet a spolupracovat s dalšími softwary. Nové systémy často používají výkonné analytické funkce, strojové učení a průmyslové Internet of Things (IoT) nebo datové uložení na cloudovém úložišti. Výhodami ERP systému jsou efektivnější automatizace sběru dat, sdílení dat a zlepšování řízení dodavatelského řetězce. Mezi dodavatele ERP systému patří SAP, Oracle nebo NetSuite Inc. [26]

2.7.3 Product Lifecycle Management

Product Lifecycle Management (PLM) řízení životního cyklu výrobku. Jedná se o co nejefektivnější řízení produktu po celou jeho dobu životního cyklu, od první myšlenky, jeho výroby a uvedení na trh až po jeho likvidaci. PLM spravuje nejen všechny jednotlivé produkty, ale i produktové portfolio, což je sběr všech produktů společnosti. Cílem PLM je zvýšit výnos z produktu, snížit náklady související s produktem a maximalizovat hodnotu současných a budoucích produktů pro zákazníky. Tento systém spolupracuje s dalšími systémy jakou jsou ERP. [27]

2.7.4 Open Platform Communications Unified Architecture (OPC UA)

Linka se dá ovládat také přes Open Platform Communications Unified Architecture (OPC UA) protokol. Jedná se komunikační standard vyvinutý OPC Fundation. Jedná se o bezpečný protokol pro výměnu dat v průmyslové automatizaci a dalších průmyslových odvětvích. Je nezávislý na platformě a zajišťuje tedy plynulý tok informací mezi zařízeními od více dodavatelů. OPC UA navazuje na protokol OPC. Na tomto protokolu může fungovat více operačních systému, jako je Windows, Linux nebo Android. [28]

Kapitola 3

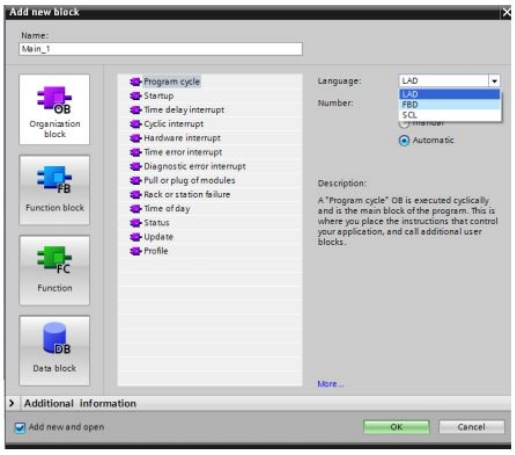
Praktická část

V rámci praktické části jsem se nejdříve seznámil s celým dopravníkovým systémem, tedy s hardwarovou, softwarovou i komunikační částí. Dále jsem se naučil pracovat s vývojovým prostředím TiaPortal V15 a vytvořit ukázkový program demonstrující použití softwarového programovatelného automatu S7-1500 od firmy Siemens. Jeho IP adresa je **10.35.91.21**. Úkolem bylo naprogramovat úlohu posílání vozíků na určená stanoviště (StopStations) a monitorování dopravníkového systému (aktuální pozice vozíku v systému). Celý systém lze ovládat dvěma způsoby, a to přes webové rozhraní WebGUI, přes které se i konfigurují modulu, ale ovládání je pouze částečné (vysvětleno v kapitole 2.3.6) nebo přes vývojové prostředí TiaPortal pomocí datových bloků, popřípadě funkčních bloků. V této kapitole je popsána softwarová část celého projektu, kterou jsem vytvořil. Pro každý modul je vytvořený funkční blok (vycházel jsem s funkčních bloků, které byly použity pro minulý projekt, ale nefungovaly) a k tomu je vytvořený datový blok, přes který se ovládá. Nejdříve jsou popsány všechny funkční bloky a poté ovládání celé linky přes datové bloky. Také je zde popsáné navázání pneumatických stanici, aby se automaticky spustili, když vozík dorazí do StopStation.

3.1 Project Tree v TiaPortalu

Vývojové prostředí poskytuje čtyři základní programovací bloky, viz obrázek 3.1. Organization Block (OB) funguje jako rozhraní mezi operačním systémem a uživatelským programem. Tento blok obsahuje především cyklicky vykonávané programy, programy vyvolané různým přerušením (např. Hardwarem nebo Cyclic interrupt) nebo spouštěcí chování a ošetření chyb. Bloky jsou rozděleny podle funkce a mají označení např. OB1. Data Block (DB) je blok, pomocí kterého je možné ukládat data do různých paměťových oblastí, jako jsou například obrazy vstupů a výstupů nebo bitová paměť. Používají se pro ukládání dat a výsledků. Bloky mohou být globální nebo nebo pouze instanční, tedy

přiřazeny pouze k jednotlivému Function Block (FB). Function Block (FB) bloky mají vždy přiřazen svůj datový blok. Data a parametry jsou v instanci uloženy i po ukončení funkčního bloku. Specifický funkční blok je možné volat i víckrát v jenom cyklu a pokaždé může uložit data do jiného datového bloku, proto mají trvalou paměť. Funkce (Function (FC)) jsou rychle vykonané funkční bloky na základě vstupních parametrů. Data ukládají do globální paměti a používají se k vytvoření znovu použitelných operací, jako jsou matematické výpočty a technické funkce. Pro správu proměnných slouží nástroj PLC tags (použité pro ovládání pístu ze stanice Festo). Takzvané tagy jsou proměnné, které se v programovém kódu používají se symbolickým adresováním, při programování se pak uživatel nemusí odkazovat přímo na fyzickou adresu, ale jen na název tagu, což je mnohem přehlednější.[29]

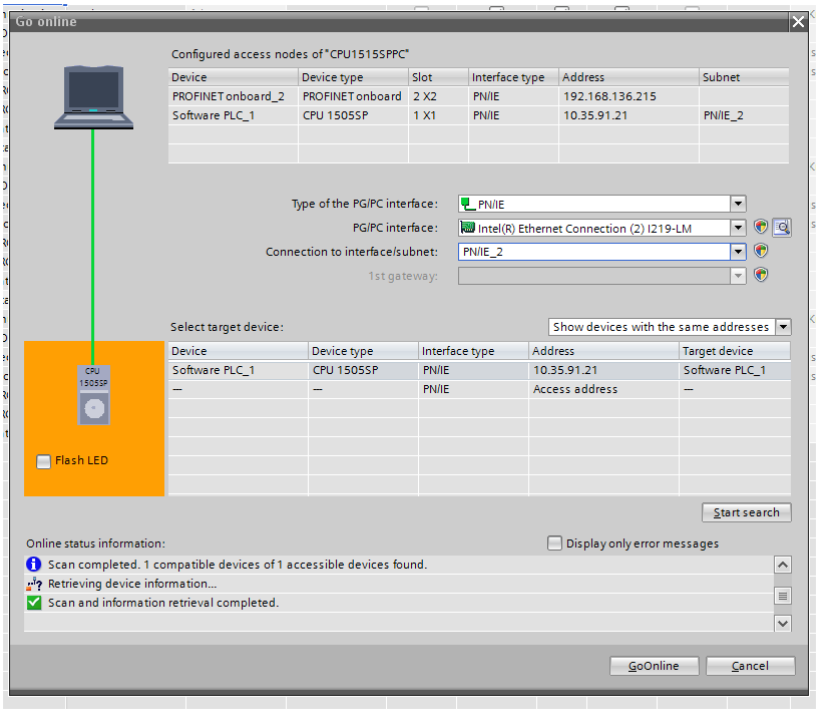


Obrázek 3.1: Okno s výběrem programovacích bloků (vlastní tvorba)

3.2 Připojení PLC

Jako první krok se musí otevřít projekt **Montrac_Linka_Funkční** ve vývojovém prostředí TiaPortal. Ověřit či připojit PLC je možné pomocí tlačítka Go Online nebo Accessible devices umístěné v horním panelu nástrojů, pomocí něhož vyvoláme okno, kde nastavíme příslušný typ rozehraní (PN/IE) a síťovou kartu, přes kterou se připojuji k jednotlivým zařízením. Stisknutím tlačítka "Start search"se provede skenování zařízení, připojených k daném rozhraní. Po dokončení skenování se v tabulce zobrazí všechny dostupná zařízení včetně jejich příslušných IP adres. Je také možné provést test spojení pomocné blikání LED na panelu PLC nebo displeje HMI panelu.

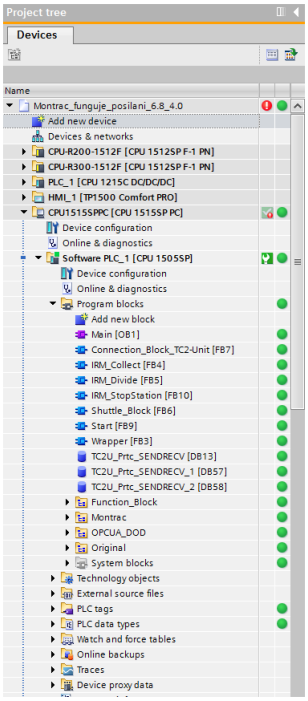
Softwarové PLC, které se používá pro řízení dopravníkového sytému má název Software PLC_1 Device type je CPU 1505SP, Interface type je PROFINET/Industrial Ethernet (PN/IE) jeho IP adresa je **10.35.91.21**. Ukázku toho připojení můžeme vidět na obrázku 3.2. Hardwarová konfigurace je vyobrazena v 2.16.



Obrázek 3.2: Ukázka připojení softwarového PLC (vlastní tvorba)

3.3 Funkční bloky

Ve vytvořeném projektu v Project Tree (lze vidět na obrázku 3.3) nalezneme složku CPU15152PPC [CPU 15152P PC]. Jedná se o průmyslový počítač, kterým je řízeno softwarové PLC. V podsložce **Software PLC_1 [CPU 1505SP]** se nachází celý program, řídící dopravníkový systém. Jako hlavní programovací jazyk pro celý projekt jsem zvolil jazyk Structured Text (ST), protože se jedná o složitější úlohu a proto bude kód v tomto jazyce nejpřehlednější. Tyto bloky slouží pro řízení linky a dají se ovládat pomocí datových bloků, které jsou popsány níže. Částečně jsem vycházel s funkčních bloků, které byly použity pro řízení minulého stavu linky.



Obrázek 3.3: Ukázka Project Tree funkčních bloku, které mohu použít (vlastní tvorba)

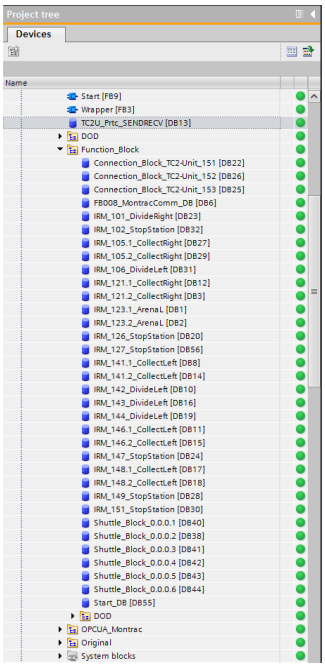
Ve složce **"Program Block"**, se nacházejí všechny funkční bloky, které jsem vytvořil pro ovládání linky. Každý funkční blok má přiřazený svůj datový blok, přes který se dá ovládat. Jednotlivé funkční i datové bloky jsou popsány v další kapitole. V automaticky vygenerovaném organizačním bloku OB1 jsou aktuálně používané funkční bloky. Tento blok je hlavním blokem v PLC a je volán cyklicky v systému CPU.

Ve složce **"Montrac"** se nacházejí datové bloky, kterými lze řídit dopravníkový systém. V této složce je také funkční blok FB008_MontracComm[FB8]. Tento funkční blok obsahuje všechny použité bloky v celém projektu a jeho rozhraní je v LD jazyce. Všechny tyto bloky jsou aktivní.

Složka **"OPCUA_DOD"** byla vytvořena pro ovládání nadřazeným systémem, který komunikuje přes rozhraní OPC. Nachází se v ní datové bloky pro jednotlivé vozíky a jsou přizpůsobené pro snazší ovládání nadřazeného systému.

Ve složce **"Function_Block"** jsou všechny aktivní a použité funkční bloky. Přes tuto složku jde rychle najít daný blok a podívat se, zda funguje správně. Všechny funkční bloky lze vidět na obrázku 3.4.

Ve složce **"Original"** jsou funkční bloky, které byly v souboru, než jsem na lince začal pracovat.



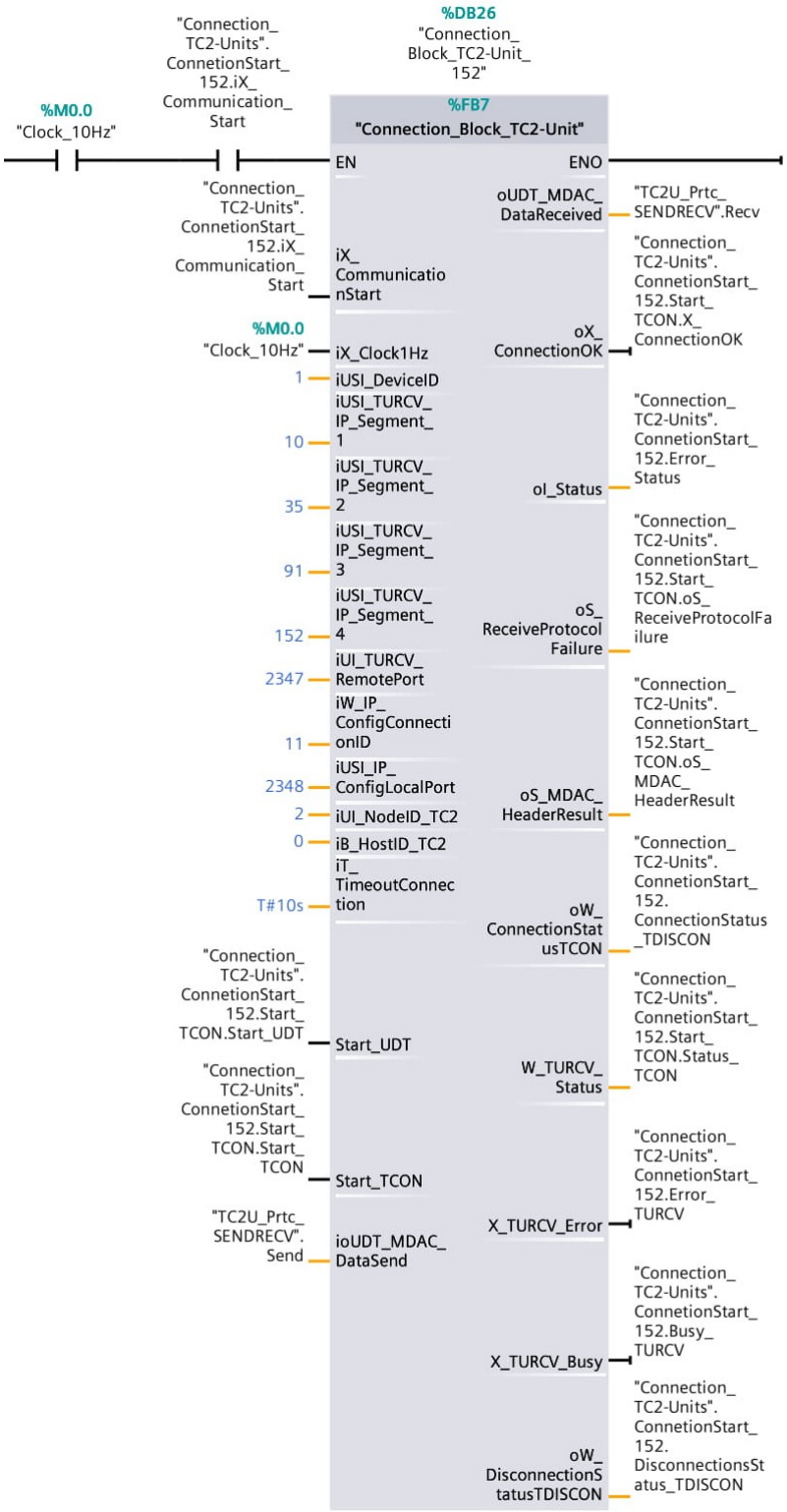
Obrázek 3.4: Ukázka Project Tree se všemi použitými funkčními bloky (vlastní tvorba)

3.3.1 Connection_Block_TC2-Unit

Tento funkční blok slouží pro připojení a komunikaci mezi PLC a jedné řídící jednotky TC2-Unit. Přes tento funkční blok se přijímají a odesílají data do řídící jednotky, která dále komunikuje s moduly. Dopravníkový systém používá tři tyto řídící jednotky (pro každý segment jedna jednotka), takže se musely použít tři tyto funkční bloky. Na obrázku 3.5 lze vidět, jak tento blok vypadá se všemi vstupy a výstupy. Tabulka 3.1 detailně popisuje význam jednotlivých vstupů a výstupů a také jejich datový typ. Každý blok je navázaný na datový blok, přes který je řízen. V případě tohoto funkčního bloku se jedná o datový typ Connection_TC2-Units [DB9]. Funkční blok funguje na protokol UDP. Proto jsou pro komunikaci použity bloky TUSEND a TURECV (tyto bloky slouží pro komunikace mezi CPU pomocí protokolu PROFINET viz [30]). Každá řídící jednotka je popsána níže.

Local Port a Remote Port

Pokud jednotka PLC komunikuje s více řídicími jednotkami TC2-Unit, tak Local port musí být u každé jednotky jiný. Toto jde jednoduše nastavit ve WebGui.



Obrázek 3.5: Vstupně/výstupní rozhraní bloku Connection Block (vlastní tvorba)

Connection_Block_TC2-Unit		
Input		
Název	Datový typ	Funkce
iX_ComunicationStart	Bool	Start připojení mezi CPU a řídící jednotkou TC2-Unit
iX_Clock1Hz	Bool	Časové přijímání dat ze řídící jednotky TC2-Unit
iUSI_DeviceID	USInt	Identifikace pro lokální PN/IE interface
iUSI_TURCV_IP_Segment_1	USInt	IP adresa TC2-Unit, první segment
iUSI_TURCV_IP_Segment_2	USInt	IP adresa TC2-Unit, druhý segment
iUSI_TURCV_IP_Segment_3	USInt	IP adresa TC2-Unit, třetí segment
iUSI_TURCV_IP_Segment_4	USInt	IP adresa TC2-Unit, čtvrtý segment
iUI_TURCV_RemotePort	Uint	TC2U - Local MDAC port. Tento port může být pro všechny jednotky stejný. Lze nastavit ve WebGui.
iW_IP_ConfigConnectionID	Word	Referenční ID pro datové bloky TSEND a TRECV
iUSI_IP_ConfigLocalPort	Uint	Lokální adresa komponenty TCON. Pro každý datový blok, musí být nastavená jiná hodnota. Lze nastavit ve WebGui
iUI_NodeID_TC2	Uint	Node ID řídící jednotky TC2-Unit. Lze nastavit ve WebGui.
iB_HostID_TC2	Byte	Host ID podřadného systému. Nastavené u každé jednotky na hodnotu 0.
iT_TimeoutConnection	Time	Timeout host u řídící jednotky TC2-Unit
Start_UDT	Bool	Povolení posílání dat přes UDT
Start_TCON	Bool	Zahájení připojení pro bloky TCON.
InOut		
ioUDT_MDAC_DataSend	UDT_TC2U_BlockA_MDAC_PrtcData132	Stkruktura MDAC protokolu
Output		
oUDT_MDAC_DataReceived	UDT_TC2U_BlockA_MDAC_PrtcData132	Struktura obdržených dat ze řídící jednotky TC2-Unit
oX_ConnectionOK	Bool	UDP ConnectionOK
oI_Status	Int	Error status (Header - Result)
oS_ReceiveProtocolFailure	String	Recv prtc
oS_MDAC_HeaderResult	String	Error state (Header Result)
oW_ConnectionStatusTCON	Word	TCON status
W_TURCV_Status	Word	TURCV status
X_TURCV_Error	Bool	TURCV error
X_TURCV_Busy	Bool	TURCV busy
oW_DisconnectionStatusTD	Word	TDISCON status

Tabulka 3.1: Tabulka s datovými typy funkčního bloku Connection Block TC2-Unit (vlastní tvorba)

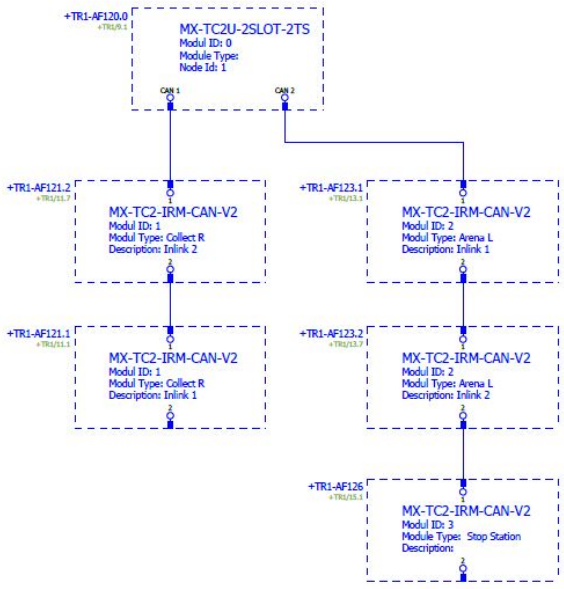
oI_Status:

- 1 - 5 - Header Resutl viz kapitola 2.3.5
- - 50 - Timeout Senden
- -100 - No Connection

Informace o řídicích jednotkách

První řídící jednota TC2-Unit:

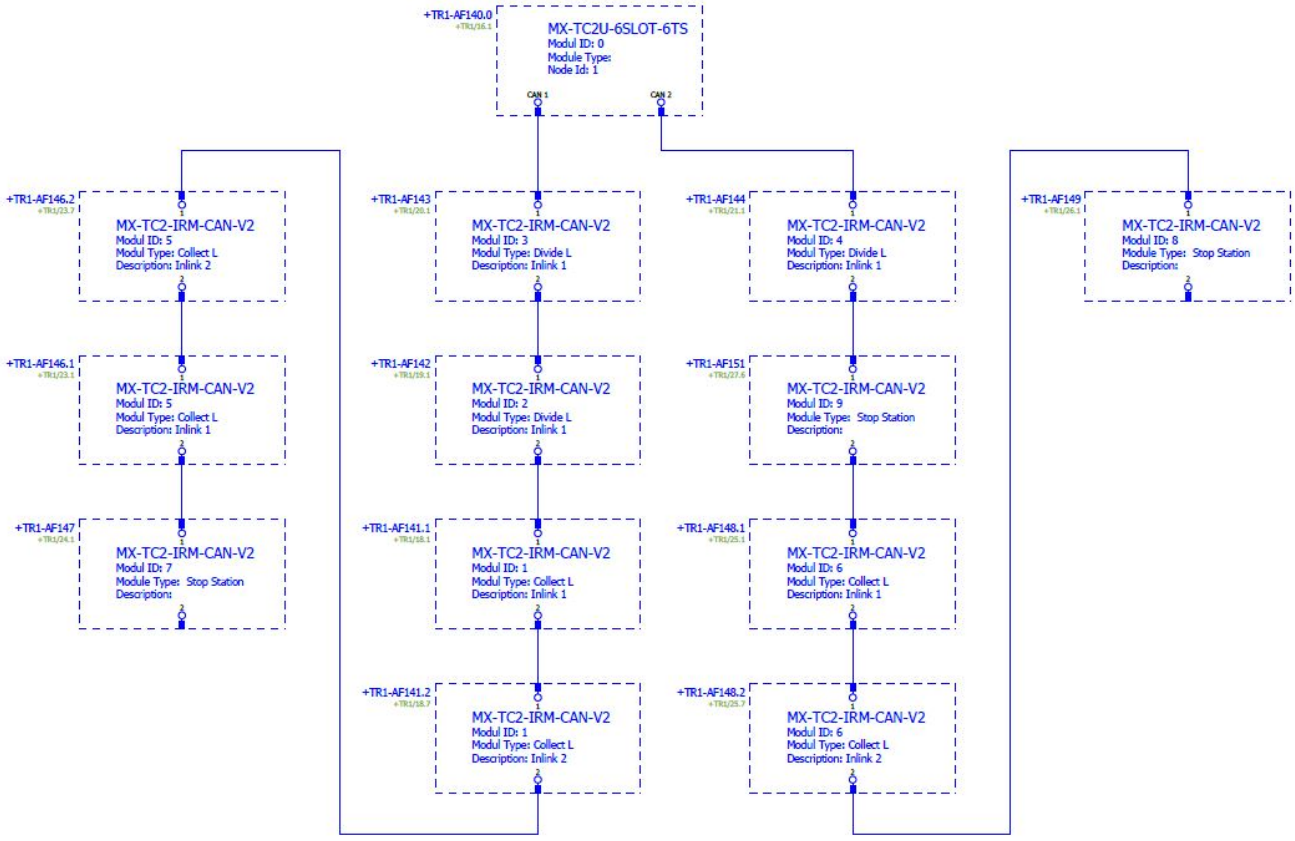
- Označení na výkrese je -AF120.0
- IP adrese této jednotce je 10.35.91.151
- Node ID = 1
- Local Port = 2345
- Remote Port = 2346
- Subnet mask je 255.255.255.0
- Připojené moduly lze vidět na obrázku 3.6
- K tato jednotka také řídí přidanou StopStation -AF107.0
- Datový blok TC2U_Prtc_SENDRECV



Obrázek 3.6: Připojené moduly k řídící jednotce -AF120.0. [1]

Druhá řídící jednota TC2-Unit:

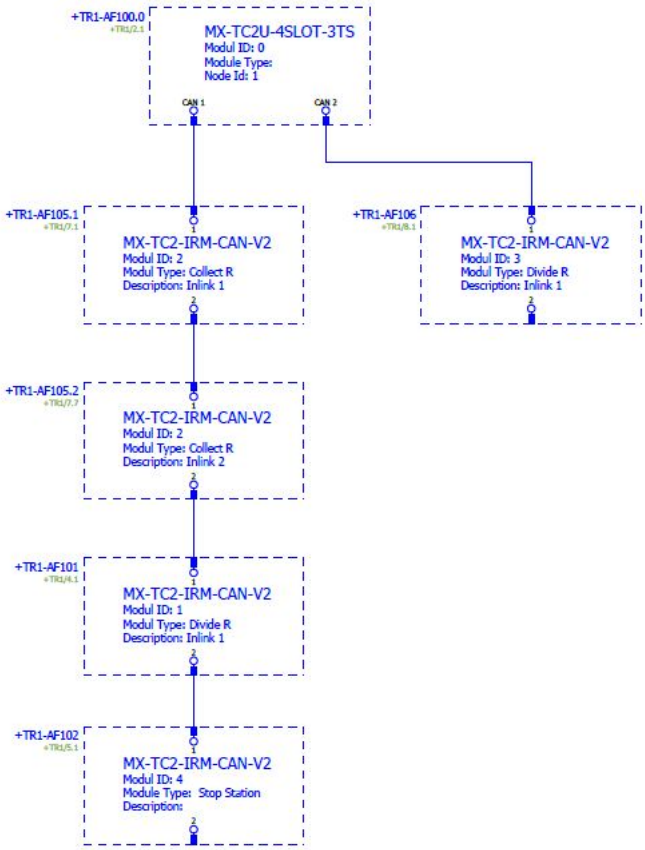
- Označení na výkrese je -AF140.0
- IP adrese této jednotce je 10.35.91.152
- Node ID = 2
- Local Port = 2347
- Remote Port = 2348
- Subnet mask je 255.255.255.0
- Připojené moduly lze vidět na obrázku 3.7
- Datový blok TC2U_Prtc_SENDRECV_1



Obrázek 3.7: Připojené moduly k řídící jednotce -AF140.0. [1]

Třetí řídící jednota TC2-Unit:

- Označení na výkrese je -AF100.0
- IP adrese této jednotce je 10.35.91.153
- Node ID = 3
- Local Port = 2346
- Remote Port = 2352
- Subnet mask je 255.255.255.0
- Připojené moduly lze vidět na obrázku 3.8
- Datový blok TC2U_Prtc_SENDRECV_2



Obrázek 3.8: Připojené moduly k řídící jednotce -AF100.0. [1]

Poznámka:

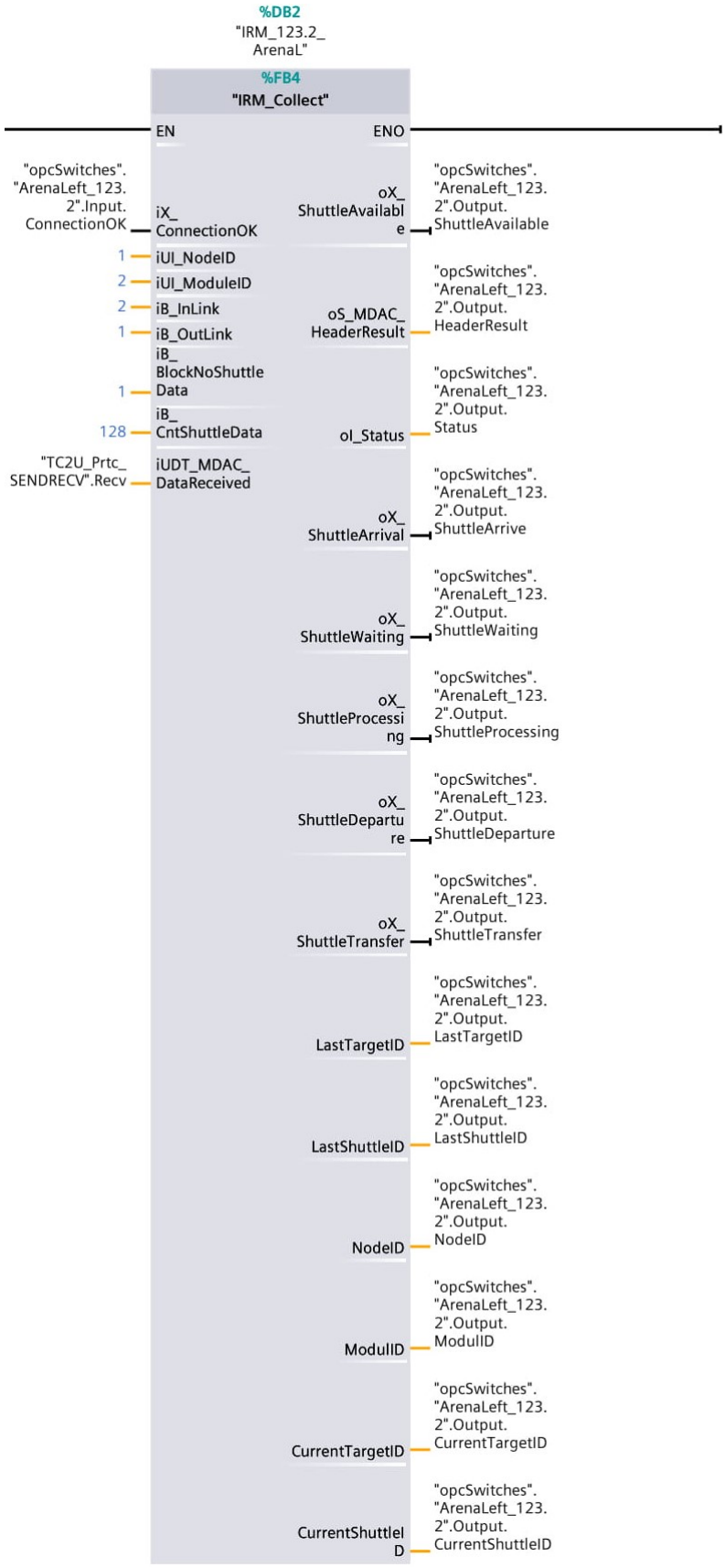
- Pokud by se řídící jednotky často odpojovali, lze v kódu u proměnné **T_Ping** zvýšit hodnotu
  - Pokud vypadne proud, tak vypadne spojení jakmile tato hodnota dosáhne svého maxima
- Proměnná **T_Pollen** slouží k opětovnému připojení řídících jednotek

Přidání funkčního bloku Connection_Block_TC2-Unit

- Musí se provést úpravy v kódu funkčních bloků IRM_Divide, IRM_StopStation, Shuttle_Block, Wrapper a Start
- Pouze přidání řádku, aby se dané funkce posílal také do tohoto bloku (řídící jednotky)

3.3.2 IRM_Collect

Tento funkční blok slouží ke čtení dat z konkrétního modulu TracSwitch collect. Pro každý tento modul musí být přidán jeden tento blok a správně nakonfigurován. Na obrázku 3.9 lze vidět, jak tento blok vypadá se všemi vstupy a výstupy. Tabulka 3.2 detailně popisuje význam jednotlivých vstupů a výstupů a také jejich datový typ. Celkově je použito deset těchto bloků, pro každý modul se musí použít dvě čidla, to znamená, že na licence je pět těchto bloků.



Obrázek 3.9: Vstupně/výstupní rozhraní bloku IRM Collect (vlastní tvorba)

IRM_Collect		
Input		
Název	Datový typ	Funkce
iX_ComunicationOK	Bool	Status UDP připojení je OK
iUI_NodeID	Uint	Node ID řídící jednotky TC2-Unit. Podle jednotky, ke které je modul připojený.
iUI_ModuleID	Uint	Modul ID. Každý modul má své jedinečné číslo.
iB_InLink	Byte	In Link číslo. Podle toho kolik daný modul má vstupů na koleji.
iB_OutLink	Byte	Out Link číslo. Podle toho kolik daný modul má výstupů na koleji.
iB_BlockNoShuttleData	Byte	Block number (read/write) 1 - 32
iB_CntShuttleData	Byte	
iUDT_MDAC_DataReceived	UDT_TC2U_BlockA_MDAC_PrtcData132	Struktura obdržených dat ze řídící jednotky TC2-Unit
Output		
oX_ShuttleAvailable	Bool	Vozík je dospný na této stanici
oS_MDAC_HeadResult	String	Error state (Header Result)
oI_Status	Int	Error state (Header - Result)
oX_ShuttleArrival	Bool	Vozík dorazil na modul
oX_ShuttleWaiting	Bool	Vozík čeká na tomto modulu na příkaz
oX_ShuttleProcessing	Bool	Vozík zpracováva příkaz, který je na tomto modulu
oX_ShuttleDeparture	Bool	Vozík opouští daný modul
oX_ShuttleTransfer	Bool	Vozík komunikuje s daným modulem
LastTargetID	Dword	Poslední Target ID, který bylo na této stanici
LastShuttleID	Dword	Poslední Shuttle ID, který bylo na této stanici
NodeID	Int	Node ID řídící jednotky TC2-Unit. Podle jednotky, ke které je modul připojený.
ModulID	Int	Modul ID. Každý modul má své jedinečné číslo.
CurrentTargetID	Dword	Aktuální Target ID na modulu
CurrentShuttleID	Dword	Aktuální Shuttle ID na modulu

Tabulka 3.2: Tabulka s datovými typy funkčního bloku IRM Collect (vlastní tvorba)

oI_Status:

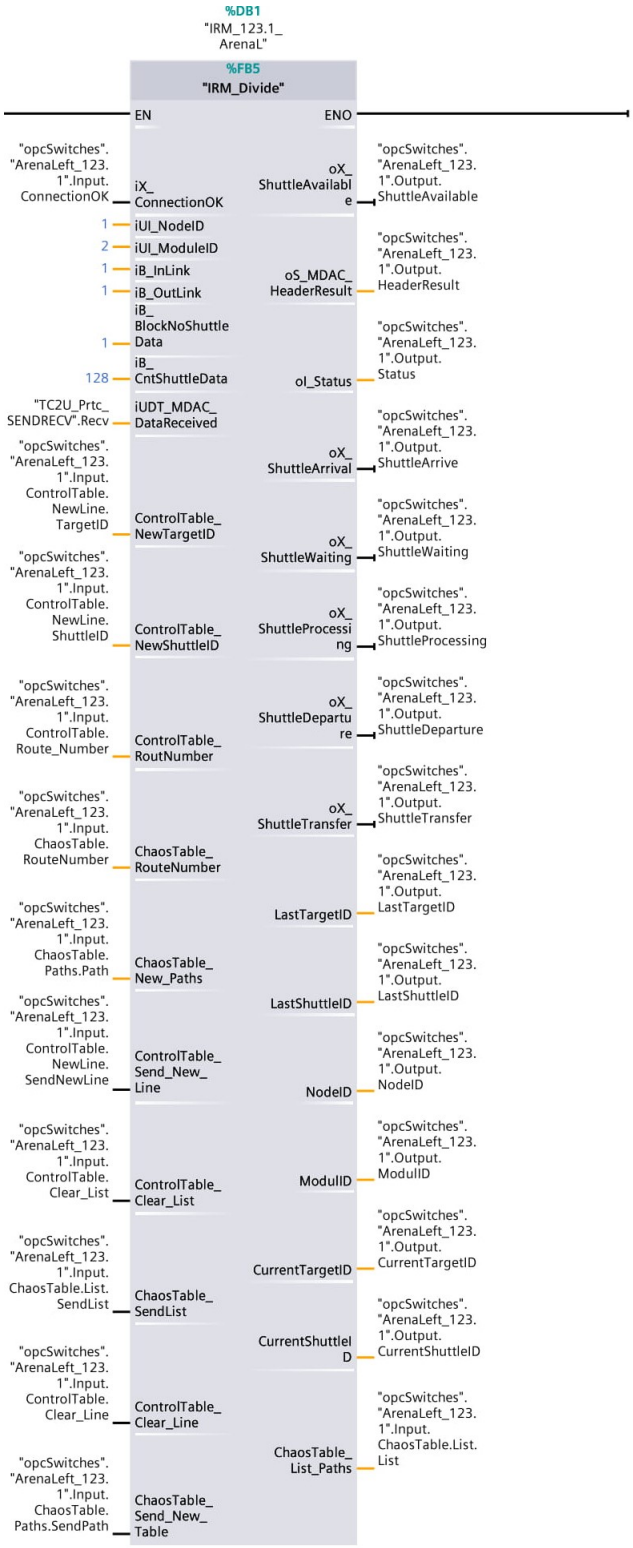
- 1 - 5 - Header Resutl viz kapitola 2.3.5
- -100 - No Connection

Přidání funkčního bloku IRM_Collect

- Do funkčního bloku **Shuttle_Block** se musí tento blok přidat do kódu pod Inicializace
- Pouze přidat řádek, že existuje další modul

3.3.3 IRM_Divide

Tento funkční blok slouží ke čtení a posílání dat z konkrétního modulu TracSwitch divide. Pro každý tento modul musí být přidán jeden tento blok a správně nakonfigurován. Na obrázku 3.10 lze vidět jak tento blok vypadá se všemi vstupy a výstupy. Tabulka 3.11 detailně popisuje význam jednotlivých vstupů a výstupů a také jejich datový typ. Datový typ **TracSwitch divide** má stejné funkce jako datový blok opcStopStation, až na jednu výjimku, že s tohoto bloku nelze posílat vozík do nové StopStation. Celkově je použito sedm těchto bloků.



Obrázek 3.10: Vstupně/výstupní rozhraní bloku IRM Divide (vlastní tvorba)

IRM_Divide		
Input		
Název	Datový typ	Funkce
iX_ComunicationOK	Bool	Status UDP připojení je OK
iUI_NodeID	Uint	Node ID řídící jednotky TC2-Unit. Podle jednotky, ke které je modul připojený.
iUI_ModuleID	Uint	Modul ID. Každý modul má své jedinečné číslo.
iB_InLink	Byte	In Link číslo. Podle toho kolik daný modul má vstupů na koleji.
iB_OutLink	Byte	Out Link číslo. Podle toho kolik daný modul má výstupů na koleji.
iB_BlockNoShuttleData	Byte	Block number (read/write) 1 - 32
iB_CntShuttleData	Byte	
iUDT_MDAC_DataReceived	UDT_TC2U_BlockA_MDAC_PrtcData132	Struktura obdržených dat ze řídící jednotky TC2-Unit
ControlTable_NewTargetID	Dword	Target ID, které chceme poslat do Control Table
ControlTable_NewShuttleID	Dword	ShuttleID, které chceme poslat do Control Table
ControlTable_SendPath	Bool	Pošle nový řádek do Control Table.
ControlTable_RoutNumber	Byte	Route Number, do jaké cesty chceme daný příkaz poslat. Když je jedna cesta tak 1.
ControlTable_ClearLine	Bool	Vymaže první řádek v zadané Control Table
ControlTable_ClearList	Bool	Vymaže celou tabulku v Conrol Table
ChaosTable_Send_New_Table	Bool	Pošle novou tabulku do Chaos Table
ChaosTable_RouteNumber	Byte	Route Number, do jaké cesty chceme daný příkaz poslat. Když je jedna cesta tak 1.
ChaosTable_SendList	Bool	Vypíše tabulku z Chaos Table
ChaosTable_NewPaths		Nová tabulka co se má zapsat do modulu. Až 5 řádků
FromTargetID	Dword	Rozmezí od jakého TargetID
UntilTargetID	Dword	Rozmezí do jakého TargetID
FromShuttleID	Dword	Rozmezí od jakého TargetID
UntilShuttleID	Dword	Rozmezí do jakého ShuttleID
InOut		
ioUDT_MDAC_DataSend	UDT_TC2U_BlockA_MDAC_PrtcData132	Stkruktura MDAC protokolu
Output		
oX_ShuttleAvailable	Bool	Vozík je dospný na této stanici
oS_MDAC_HeadResult	String	Error state (Header Result)
oI_Status	Int	Error state (Header - Result)
oX_ShuttleArrival	Bool	Vozík dorazil na modul
oX_ShuttleWaiting	Bool	Vozík čeká na tomto modulu na příkaz
oX_ShuttleProcessing	Bool	Vozík zpracovává příkaz, který je na tomto modulu
oX_ShuttleDeparture	Bool	Vozík opouští daný modul
oX_ShuttleTransfer	Bool	Vozík komunikuje s daným modulem
LastTargetID	Dword	Poslední Target ID, který bylo na této stanici
LastShuttleID	Dword	Poslední Shuttle ID, který bylo na této stanici
NodeID	Int	Node ID řídící jednotky TC2-Unit. Podle jednotky, ke které je modul připojený.
ModulID	Int	Modul ID. Každý modul má své jedinečné číslo.
CurrentTargetID	Dword	Aktuální Target ID na modulu
CurrentShuttleID	Dword	Aktuální Shuttle ID na modulu
ChaosTable_List		Zde se po příkazu vypíše celou Chaos Table

Obrázek 3.11: Tabulka s datovými typy funkčního bloku IRM Divide (vlastní tvorba)

oI_Status:

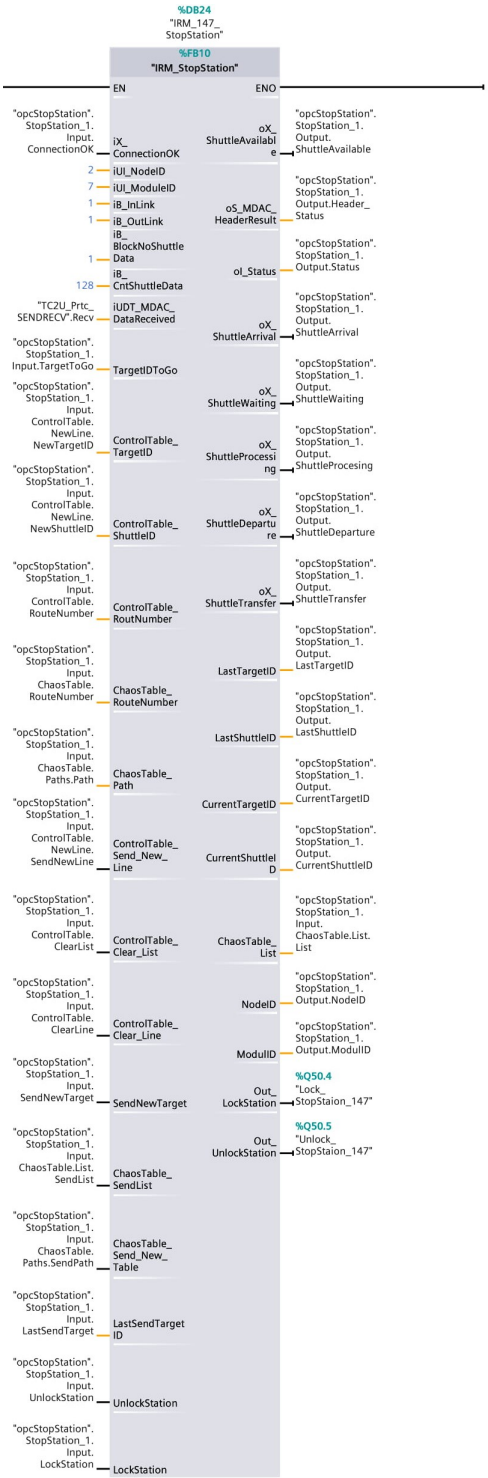
- 1 - 5 - Header Resutl viz kapitola 2.3.5
- -100 - No Connection

Přidání funkčního bloku IRM_Divide

- Do funkčního bloku **Shuttle_Block** se musí tento blok přidat do kódu pod Inicializace
- Pouze přidat řádek, že existuje další modul

3.3.4 IRM_StopStaion

Tento funkční blok slouží ke čtení a posílání dat z konkrétního modulu StopStation. Pro každý tento modul musí být přidán jeden tento blok a správně nakonfigurován. Na obrázku 3.12 lze vidět jak tento blok vypadá se všemi vstupy a výstupy. Tabulka 3.3 detailně popisuje význam jednotlivých vstupů a výstupů a také jejich datový typ. Tento bloku slouží k posílání vozíků do nové StopStation. Přes tento blok, můžeme také ovládat pístky, které slouží k zamykání pracovních desek na vozíku a monitorování StopStation.



Obrázek 3.12: Vstupně/výstupní rozhraní bloku IRM StopStation (vlastní tvorba)

IRM_StopStation		
Input		
Název	Datový typ	Funkce
iX_ComunicationOK	Bool	Status UDP připojení je OK
iUI_NodeID	Uint	Node ID řídící jednotky TC2-Unit. Podle jednotky, ke které je modul připojený.
iUI_ModuleID	Uint	Modul ID. Každý modul má své jedinečné číslo.
iB_InLink	Byte	In Link číslo. Podle toho kolik daný modul má vstupů na koleji.
iB_OutLink	Byte	Out Link číslo. Podle toho kolik daný modul má výstupů na koleji.
iB_BlockNoShuttleData	Byte	Block number (read/write) 1 - 32
iB_CntShuttleData	Byte	
iUDT_MDAC_DataReceived	UDT_TC2U_BlockA_MDAC_PrtcData132	Struktura obdržených dat ze řídící jednotky TC2-Unit
TargetIDToGo	Dword	Nové Target ID, které se má do vozíku poslat.
LastSendTargetID	Dword	Pošle nové Target ID do vozíku. Pokud je vozík dostupný
Manual_Lock_Station	Bool	Pokud není vozík dostupný, používá se tato proměná pro ovládání pístu
Unlock_Station	Bool	Pokud je vozík dosupný, používá se tato proměná pro ovládní pístu
ControlTable_NewTargetID	Dword	Target ID, které chceme poslat do Control Table
ControlTable_NewShuttleID	Dword	ShuttleID, které chceme poslat do Control Table
ControlTable_SendPath	Bool	Pošle nový řádek do Control Table.
ControlTable_RoutNumber	Byte	Route Number, do jaké cesty chceme daný příkaz poslat. Když je jedna cesta tak 1.
ControlTable_ClearLine	Bool	Vymaže první řádek v zadané Control Table
ControlTable_ClearList	Bool	Vymaže celou tabulku v Conrol Table
ChaosTable_Send_New_Table	Bool	Pošle novou tabulku do Chaos Table
ChaosTable_RouteNumber	Byte	Route Number, do jaké cesty chceme daný příkaz poslat. Když je jedna cesta tak 1.
ChaosTable_SendList	Bool	Vypíše tabulku z Chaos Table
ChaosTable_NewPaths		Nová tabulka co se má zapsat do modulu. Až 5 řádků
FromTargetID	Dword	Rozmezí od jakého TargetID
UntilTargetID	Dword	Rozmezí do jakého TargetID
FromShuttleID	Dword	Rozmezí od jakého TargetID
UntilShuttleID	Dword	Rozmezí do jakého ShuttleID
InOut		
ioUDT_MDAC_DataSend	UDT_TC2U_BlockA_MDAC_PrtcData132	Stkruktura MDAC protokolu
Output		
oX_ShuttleAvailable	Bool	Vozík je dospný na této stanici
oS_MDAC_HeadResult	String	Error state (Header Result)
oI_Status	Int	Error state (Header - Result)
oX_ShuttleArrival	Bool	Vozík dorazil na modul
oX_ShuttleWaiting	Bool	Vozík čeká na tomto modulu na příkaz
oX_ShuttleProcessing	Bool	Vozík zpracováva příkaz, který je na tomto modulu
oX_ShuttleDeparture	Bool	Vozík opouští daný modul
oX_ShuttleTransfer	Bool	Vozík komunikuje s daným modulem
LastTargetID	Dword	Poslední Target ID, který bylo na této stanici
LastShuttleID	Dword	Poslední Shuttle ID, který bylo na této stanici
NodeID	Int	Node ID řídící jednotky TC2-Unit. Podle jednotky, ke které je modul připojený.
ModulID	Int	Modul ID. Každý modul má své jedinečné číslo.
CurrentTargetID	Dword	Aktuální Target ID na modulu
CurrentShuttleID	Dword	Aktuální Shuttle ID na modulu
ChaosTable_List		Zde se po příkazu vypíše celou Chaos Table
Out_LockStation	Bool	Tag, kterým se spustí pneumatický píst
Out_UnLockStation	Bool	Tag, kterým se vypne pneumatický píst

Tabulka 3.3: Tabulka s datovými typy funkčního bloku IRM StopStation (vlastní tvorba)

oI_Status:

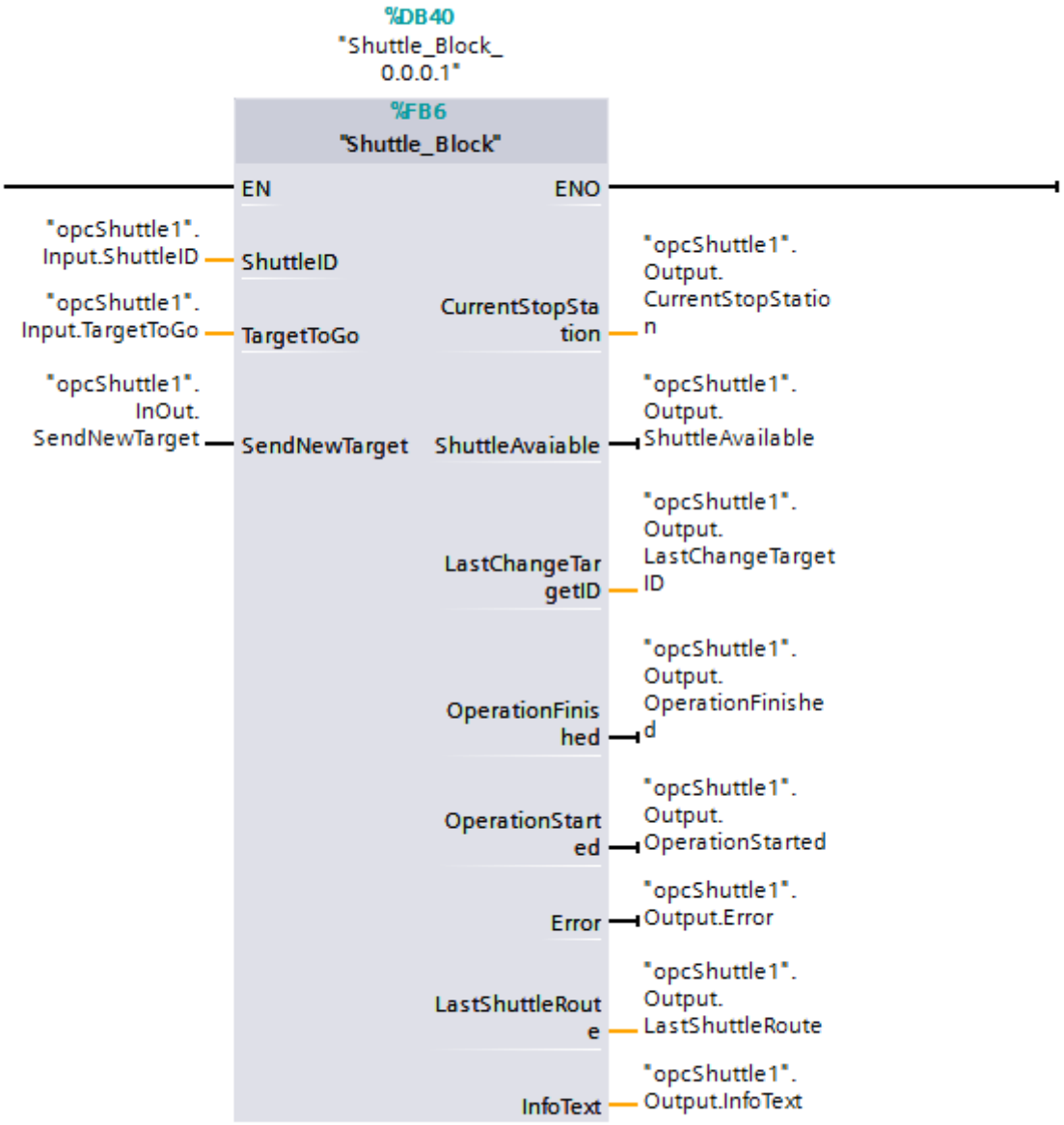
- 1 - 5 - Header Resutl viz kapitola 2.3.5
- -100 - No Connection

Přidání funkčního bloku IRM_StopStaion

- Musí se tato StopStation přidat do kódu funkčních bloků **Shuttle_Block**, **Start** a **Wrapper**
- Aby vozíky věděli, že existuje další StopStation a dal se monitorovat stav vozíku na této StopStation

3.3.5 Shuttle_Block

Tento funkční blok slouží ke čtení a ovládaní jednoho vozíku, tedy posílání vozíků na novou StopStation a monitorováváni stavu vozíku. Pro každý tento vozík musí být přidán jeden tento blok a správně nakonfigurován. Na obrázku 3.13 lze vidět jak tento blok vypadá se všemi vstupy a výstupy. Tabulka 3.4 detailně popisuje význam jednotlivých vstupů a výstupů a také jejich datový typ.



Obrázek 3.13: Vstupně/výstupní rozhraní bloku Shuttle Block (vlastní tvorba)

Shuttle_Block		
Input		
Název	Datový typ	Funkce
ShuttleID	Bool	Shuttle ID vozíku, který chceme řídit
TargetToGo	Dword	Target ID, které se má poslat do vozíku
InOut		
SendNewTarget	Bool	Pošle nové Target ID do vozíku. Pokud je vozík dostupný.
Output		
ShuttleAvailable	Bool	Vozík je dospný na této stanici
CurrentStopStation	Dword	Aktuální stanice na, které se vozík nachází
LastChangeTargetID	Dword	Poslední Target ID, které bylo do vozíku posláno
OperationFinished	Bool	Vozík dojel do stanice, která mu byla zadána
OperationStarted	Bool	Vozík vyje ze stanice a jede do určené StopStaiton
Error	Bool	Error
InfoText	String	ErrorText
LastShuttleRoute		Monitorování posledního projetého čidla
NodeID	UInt	Číslo řídicí stanice, se kterým komunikuje modul
ModulID	UInt	Číslo modulu, podle kterého se dá identifikovat
IRM_Name	String	Označení modul ve výkrese

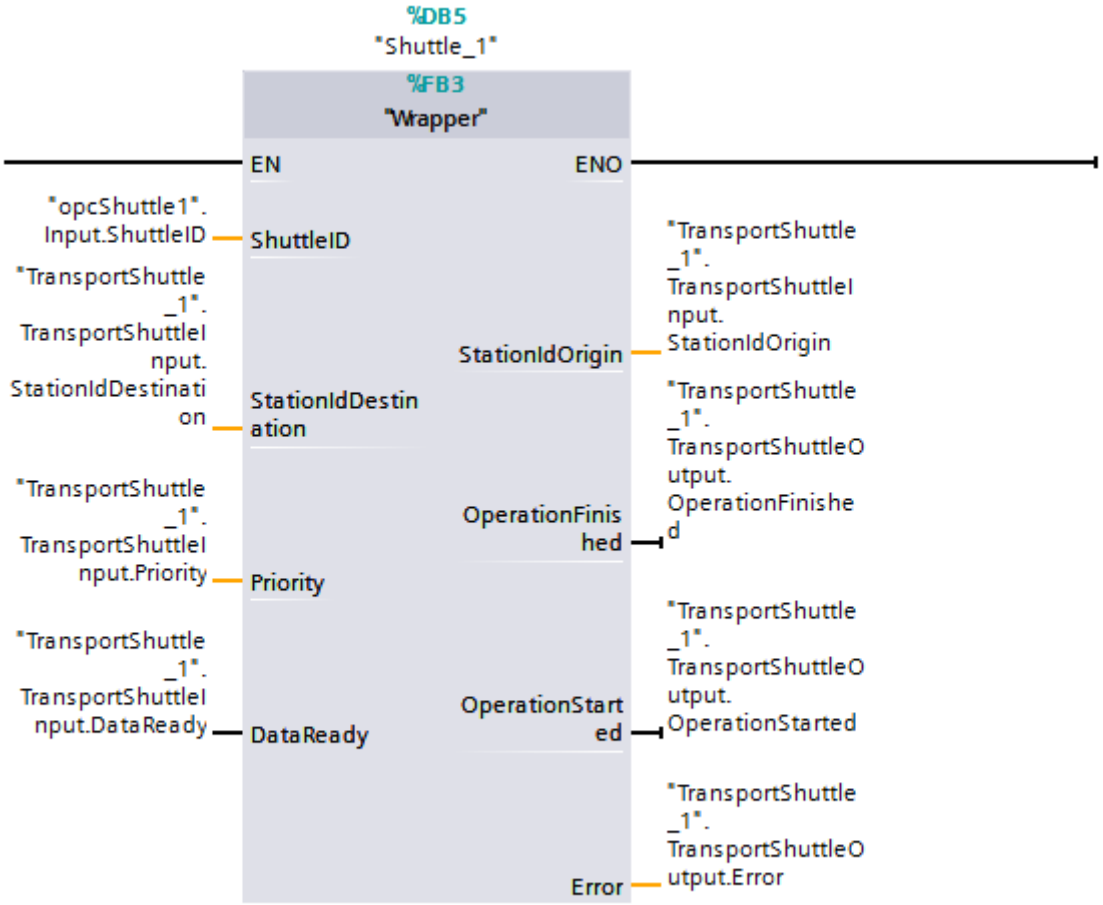
Tabulka 3.4: Tabulka s datovými typy funkčního bloku Shuttle Block (vlastní tvorba)

Přidání funkčního bloku Shuttle_Block

- Musí se provést přidání nové Shuttlu do kódu funkčních bloků **Start**
- Musí se přidat do Inicializace
- Také se musí přidat do další místo do **Shuttles_Arrival**

3.3.6 Wrapper

Tento funkční blok slouží ke čtení a ovládaní jednoho vozíku a byl vytvořen pro propojení s nadřazeným systém, tak aby se dal ovládat přes OPCUA server. Slouží pro posílání vozíků do nové StopStaion a jeho monitorování. Pro každý tento vozík musí být přidán jeden tento blok a správně nakonfigurován. Na obrázku 3.14 lze vidět jak tento blok vypadá se všemi vstupy a výstupy. Tabulka 3.5 detailně popisuje význam jednotlivých vstupů a výstupů a také jejich datový typ.



Obrázek 3.14: Vstupně/výstupní rozhraní bloku Wrapper (vlastní tvorba)

Wrapper		
Input		
Název	Datový typ	Funkce
ShuttleID	Bool	Shuttle ID vozíku, který chceme řídit
StationIdDestionation	INT	Target ID, které se má poslat do vozíku
Priority	Bool	Pouze přípava na další aplikaci
InOut		
DataReady	Bool	Pošle nové Target ID do vozíku. Pokud je vozík dostupný.
Output		
StationIdOrigin	INT	Vozík je dospný na této stanici
OperationFinished	Bool	Vozík dojel do stanice, která mu byla zadána
OperationStarted	Bool	Vozík vyje ze stanice a jede do určené StopStaiton
Error	DINT	Error

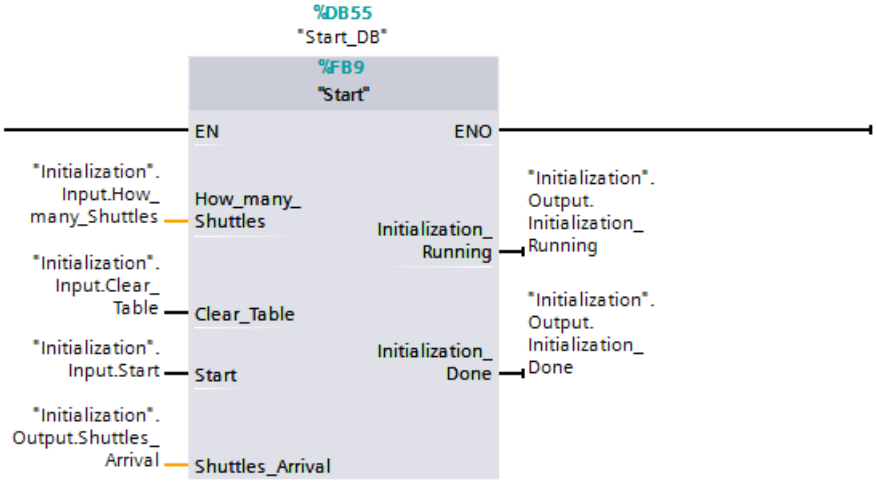
Tabulka 3.5: Tabulka s datovými typy funkčního bloku Wrapper (vlastní tvorba)

Přidání funkčního bloku Wrapper

- Nemusí se provádět žádná změna
- Inicializace je ale navázání na funkční blok **Shuttle_Block**

3.3.7 Start

Tento funkční blok slouží k inicializaci vozíků. Pokud nevíme kde se vozíky nacházejí, stačí použít tento blok a vozíky se seřadí na StopStation číslo 6 (-AF107). Na obrázku 3.15 lze vidět tak tento bloky vypadá se všemi vstupy a výstupy. Tabulka 3.6 detailně popisuje význam jednotlivých vstupů a výstupů a také jejich datový typ.



Obrázek 3.15: Vstupně/výstupní rozhraní bloku Start (vlastní tvorba)

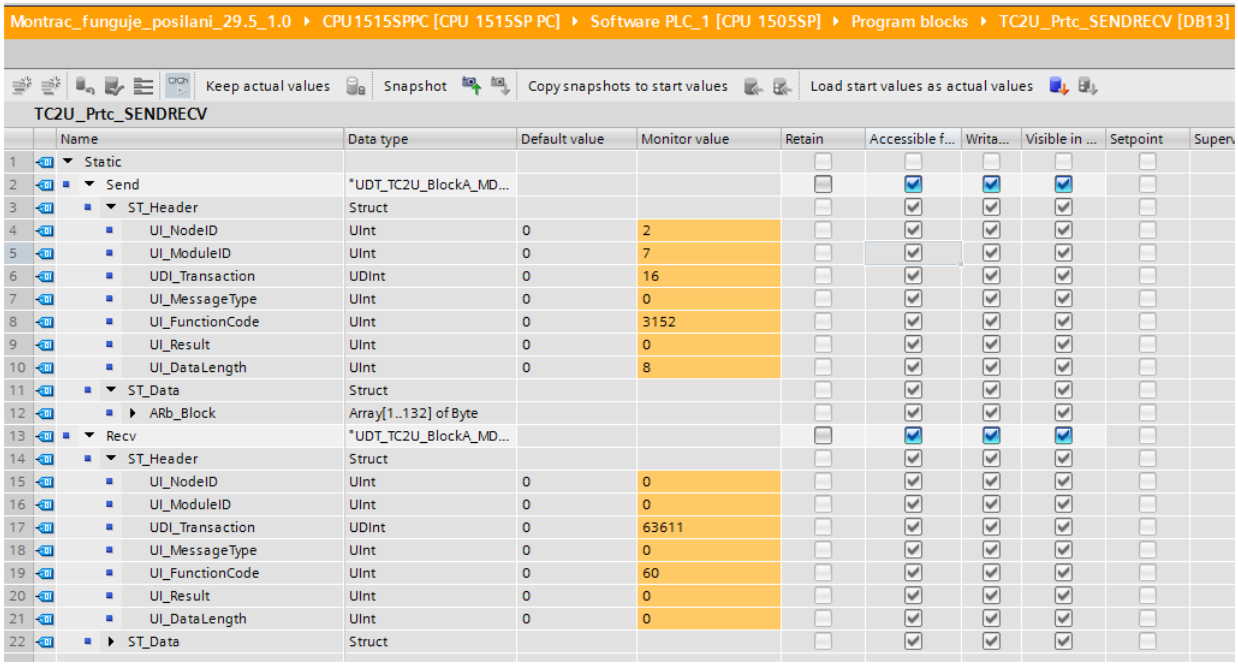
Start		
Input		
Název	Datový typ	Funkce
Start	Bool	Příkaz pro start poslání vozíků
How_many_Shuttles	INT	kolik vozíků se nachází aktuálně na stanici (Defual = 6)
Clear_Table	Bool	Vymaže data v tabulce Shuttles_Arrival
Output		
Initilization_Running	Bool	Inicalizace probíhá
Initilization_Done	Bool	Inicializace proběhla
Shuttles_Arrival	Array[1..6]ofDword	Tabulka, jak jsou vozíky seřazené na stanici

Tabulka 3.6: Tabulka s datovými typy funkčního bloku Start (vlastní tvorba)

3.3.8 TC2U_Prtc_SENDRECV

Přes tento datový blok jsou přijímána data z modulů a posílána data do řídící jednotky TC2-Unit. Tento datový blok se musí připojit na funkční bloky IRM_Collect, IRM_Divide a IRM_StopStaion, ale vždy musíme připojit ten datový blok, který komunikuje se správnou řídící jednotkou, jinak se příkazy nepošlou do správné řídící jednotky. Jeho zobrazení lze vidět na obrázku 3.16. Tento datový blok je pro každou řídící jednotku jedinečný. (v našem případě jsou tedy použity tři tyto bloky). V proměnné **Send** jsou data, která chceme posílat do řídící jednotky a následně do čidla. V proměnné **Recv** jsou data, která se dostávají ze všech čidel v systému. Jejich tvar je popsán v kapitole 2.3. Funkce se zapisují do toho bloku a poté se odesílají do řídící jednotky.

Pokud by se chtěla použít jiná funkce než, která se používá. Musíme vyplnit správně funkci v proměnné **Send** a poté jít do funkčního bloku dané řídící jednotky a přes proměnnou Send_UDT_StopStation poslat daný příkaz do řídící jednotky.

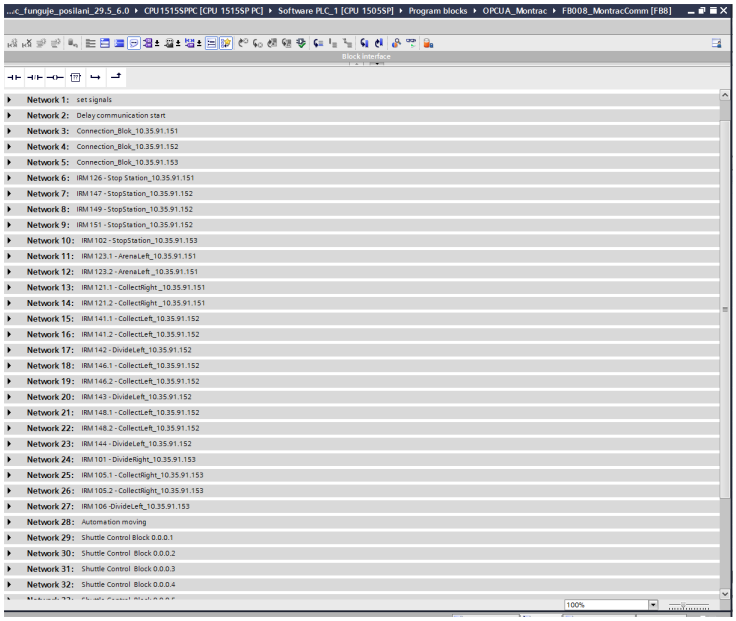


	Name	Data type	Default value	Monitor value	Retain	Accessible f...	Writa...	Visible in ...	Setpoint	Superv
1	Static									
2	Send	"UDT_TC2U_BlockA_MD...								
3	ST_Header	Struct								
4	UI_NodeID	UInt	0	2						
5	UI_ModuleID	UInt	0	7						
6	UDI_Transaction	UDInt	0	16						
7	UI_MessageType	UInt	0	0						
8	UI_FunctionCode	UInt	0	3152						
9	UI_Result	UInt	0	0						
10	UI_DataLength	UInt	0	8						
11	ST_Data	Struct								
12	ARb_Block	Array[1..132] of Byte								
13	Recv	"UDT_TC2U_BlockA_MD...								
14	ST_Header	Struct								
15	UI_NodeID	UInt	0	0						
16	UI_ModuleID	UInt	0	0						
17	UDI_Transaction	UDInt	0	63611						
18	UI_MessageType	UInt	0	0						
19	UI_FunctionCode	UInt	0	60						
20	UI_Result	UInt	0	0						
21	UI_DataLength	UInt	0	0						
22	ST_Data	Struct								

Obrázek 3.16: Ukázka Project Tree složky s datovými bloky (vlastní tvorba)

3.3.9 Shrnutí funkčních bloků

Celkově se v použilo 41 funkčních bloků pro ovládání celé linky. Všechny bloky můžeme konfigurovat ve funkčním bloku FB008_MontracComm. Do tohoto bloku se vždy přidá nový funkční blok a nakonfiguruje s pomocí příslušného datového bloku. Na obrázku 3.17 lze vidět všechny použité funkční bloky.



Obrázek 3.17: Všechny použité funkční bloky (vlastní tvorba)

3.3.10 Navázání pneumatických stanic

K této lince jsou připojeni tři modulární elektrické terminály CPX Rev 20 od firmy Festo, které řídí pneumatické písty sloužící k zamčení desky na jednotlivých StopStation. Každý terminál je každé řídící jednotky a podle toho řídí i StopStation, která řídí daná řídící jednotka. Tyto tagy se musí navázat na funkční blok, každé StopStation. Ovládání pístu, lze spravovat pouze přes datový blok StopStaion.

Jednotlivé Tagy, které řídí otvírání a zavíraní pístu u StopStation:

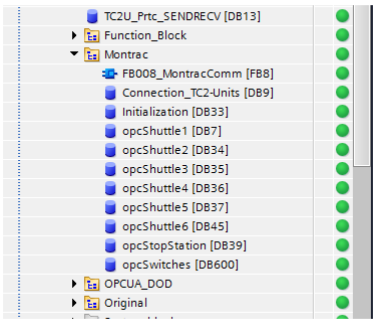
- LockStation_126 - %Q69.0
- UnlockStation_126 - %Q69.1
- LockStation_147 - %Q50.4
- UnlockStation_147 - %Q50.5
- LockStation_149 - %Q50.2
- UnlockStation_149 - %Q50.3
- LockStation_151 - %Q50.0
- UnlockStation_151 - %Q50.1
- LockStation_102 - %Q73.0
- UnlockStation_102 - %Q73.1

3.4 Ovládání linky

V této kapitole je podrobně vysvětleno jak se linka ovládá přes datové bloky, které jsem vytvořil. Datové bloky, použité k ovládání se nacházejí ve složce **Montrac**. Na obrázku 3.18, lze vidět jaké bloky se ve složce nachází.

Příkazy které se posílají do modulu musejí probíhat sekvenčně, pokud chceme použít datový blok StopStation, vždy se vykoná jeden příkaz a poté se musí poslat příkaz další. Pokud chceme poslat více příkazů na jednou, jde to mocí datových bloků **opcShuttle** nebo **TransportShuttle**, kdy se může poslat do dvou a více vozíků příkaz paralelně. Občas nastává situace, že vozík přijede do StopStation, ale čidlo zareaguje až po 20 vteřinách, toto je způsobeno tím, že StopStation komunikuje s více čidly na jednou. Příkazy tedy nejde posílat paralelně.

Pokud nějaké čidlo špatně komunikuje s vozíkem, je možné že váleček umístění před modulem nebo vně modulu se posunul a musí se upravit. Vozíky lze posílat do nové StopStation přes tři datové bloky (opcShuttle, opcStopStation, TransportShuttle), které jsou na sobě nezávislé. Musíme tedy příkaz poslat z jednoho datového bloku a ne posílat příkaz najednou z více bloků. Také je popsáno jak změnit ShuttleID, jak funguje ovládaní zamykání a odemykání StopStation pomocí pístu a posílání příkazy do Control a Chaos Table.



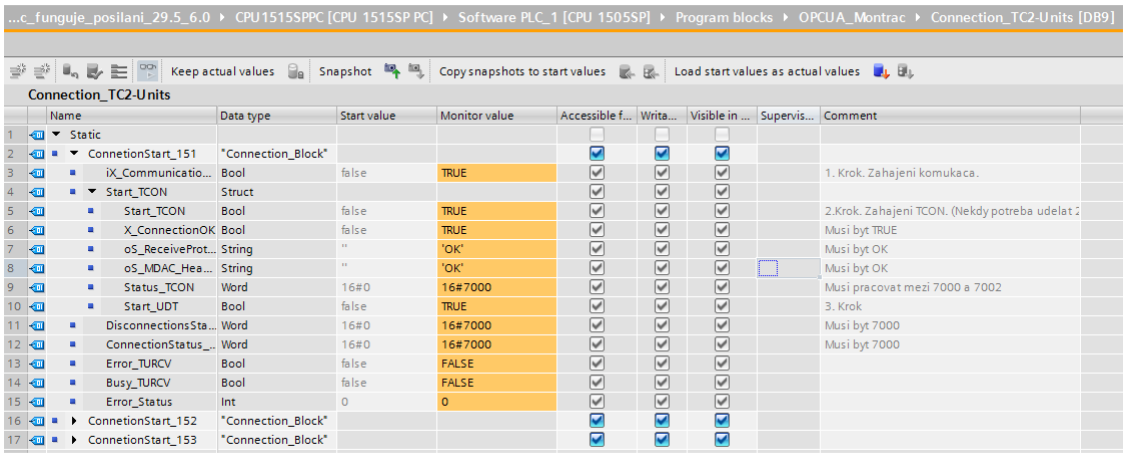
Obrázek 3.18: Ukázka Project Tree složky s datovými bloky (vlastní tvorba)

3.4.1 Popis fungovaní funkčních bloků

Hlavním funkčním blokem je blok Connection_TC2-Units, kterým se připojíme na řídící jednotku. Ostatní funkční bloky, musejí být připojená na řídící jednotku a to podle NodeID a podle ModulID (v kapitole 2.3). Pokud pošlu například příkaz do StopStation přes datový blok. Tak funkční kód se zapíše do datového bloku TC2U_Prtc_SENDRECV, kde se vyplní i data, které se mají poslat do vozíku. Poté se příkaz pošle do řídící jednotky a pomocí proměnné Send_UDT_StopStation se příkaz vykoná. Pokud by nastala chyba, tak v proměnné oS_ReceiveProtocolFaiuler nebo v oS_MDAC_HeaderResult lze vyčíst chybu protokolu MDAC.

3.4.2 Connection_TC2-Units

Tento datový blok slouží k navázání spojení s řídící jednotkou TracControl 2 Unit (TC2-Unit). V tomto bloku lze řídit a monitorovat všechny tři Connection Bloky, které ovládají tuto linku. Pokud by se přidávala další tato jednotka, může se jednoduše do tohoto bloku přidat další datový typ Connection_Block, ten je popsaný níže. Na obrázku 3.19 lze vidět jak vypadá tento datový blok. Každá jednotka má svůj unikátní název. Pro zjednodušení jsem zvolil názvy podle poslední číslo IP adresy, každé jednotky. Tedy názvy jsou ConnectionStart_151, ConnectionStart_152 a ConnectionStart_153.



	Name	Data type	Start value	Monitor value	Accessible f...	Writa...	Visible in ...	Supervis...	Comment
1	Static								
2	ConnetionStart_151	"Connection_Block"			☑	☑	☑		
3	iX_Communicatio...	Bool	false	TRUE	☑	☑	☑		1. Krok. Zahajeni komukaca.
4	Start_TCON	Struct			☑	☑	☑		
5	Start_TCON	Bool	false	TRUE	☑	☑	☑		2.Krok. Zahajeni TCON. (Nekdy potreba udelat 2
6	X_ConnectionOK	Bool	false	TRUE	☑	☑	☑		Musi byt TRUE
7	oS_ReceiveProt...	String	''	'OK'	☑	☑	☑		Musi byt OK
8	oS_MDAC_Hea...	String	''	'OK'	☑	☑	☑		Musi byt OK
9	Status_TCON	Word	16#0	16#7000	☑	☑	☑		Musi pracovat mezi 7000 a 7002
10	Start_UDT	Bool	false	TRUE	☑	☑	☑		3. Krok
11	DisconnectionsSta...	Word	16#0	16#7000	☑	☑	☑		Musi byt 7000
12	ConnectionStatus_...	Word	16#0	16#7000	☑	☑	☑		Musi byt 7000
13	Error_TURCV	Bool	false	FALSE	☑	☑	☑		
14	Busy_TURCV	Bool	false	FALSE	☑	☑	☑		
15	Error_Status	Int	0	0	☑	☑	☑		
16	ConnetionStart_152	"Connection_Block"			☑	☑	☑		
17	ConnetionStart_153	"Connection_Block"			☑	☑	☑		

Obrázek 3.19: Ukázka Project Tree složky s datovými bloky (vlastní tvorba)

Popis pro připojení řídící jednotky:

1. Jako první se musím u proměnné **iX_Communication_Start** = TRUE
2. Poté v pod složce **Start_TCON** se nachází proměnná **Start_TCON** = TRUE
3. Jako poslední krok se musí u proměnné **Start_UDT** = TRUE
4. Poté se u výstupních proměnných musí objevit **oS_ReciveProtocolFailure** = 'OK' , **oS_MDAC_HeaderResult** = 'OK' a u **X_ConnectionOK** = TRUE
5. Poté jsme se úspěšně připojili na řídí jednotku. Pokud se neobjeví chybová hláška.
   - Pokud se nedá připojit, lze zkusit u proměnné Start_TCON 2x změnit hodnotu s TRUE na FALSE a opět na TRUE
6. Toto se musí provést u každé jednotky, se kterou chceme komunikovat.

Poznámka k ostatním proměnným:

- **TURCV_Status** - Tato proměnná musí ukazovat hodnoty 16#7002 nebo 16#7000
- **TURCV_Error** - Hodnota musí být FALSE
- **TURCV_Busy** - Hodnota se mění mezi TRUE a FALSE, probíhá komunikace
- **DisconnectionStatusTDISCON** - Hodnota musí být 16#7000
- **ConnectionStatusTCON** - Hodnota musí být 16#7000
- **StatusTCON** - Hodnota musí být 0

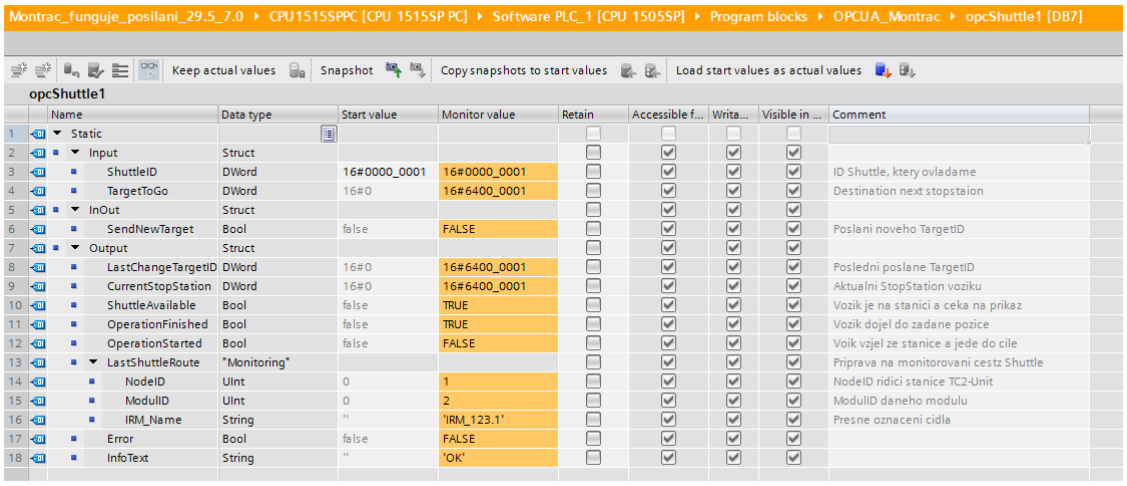
- **oS_MDAC_HeaderResult**
  - Status 'No Connection UDP', linka nemá proud
  - Status 'Reconnect in +...', linka se snaží sama připojit (Start_TCON = TRUE)
- **oS_ReciveProtocolFailure**
  - Status 'No Connection UDP', linka nemá proud
  - Status 'Reconnect in +...', linka se snaží sama připojit (Start_TCON = TRUE)

Pokud proměnné ukazují jiné hodnoty, tak nastala někde chyba. Nejlepší řešení je podívat do Helpu v TiaPortalu. V tomto Helpu si najít funkční blok TURECV nebo TUSEND, podívat se jaká nastala chyba a pokusit se tuto chybu vyřešit. Možnost je také restartovat celé PLC, pokud ani to nepomůže může být chyba ve špatně nakonfigurovaném funkčním bloku například špatně napsané Local nebo Remote port.

3.4.3 Datový blok opcShuttle

Tento datový blok slouží k monitorování a ovládání jednoho vozíku vybraného vozíku. Na této lince se nachází šest vozíků. Pro každý vozík je tedy vytvořen jeden datový blok. Každý datový blok má své unikátní název, který je opcShuttle_1 až opcShuttle_6. Pokud by se v budoucnu chtěl přidat další vozík, jednoduše se přidá tento datový blok s názvem Shuttle_Type s tohoto bloku připojit na příslušný funkční blok 3.3.5. ShuttleID jde měnit pouze pomocí přímého připojení na čidlo IRM přes rozhraní PuTTY, tento postup je popsán níže. Naopak TargetID jde měnit jednoduše pokud vozík je na modulu StopStation. Každému vozíku jsem dal unikátní číslo, tak aby bylo jednoduché je rozpoznat. Vozíky mají následující **ShuttleID**:

- 16#0000_0001
- 16#0000_0002
- 16#0000_0003
- 16#0000_0004
- 16#0000_0005
- 16#0000_0006



Montrac_funguje_posilani_29.5_7.0 ▸ CPU 1515SPPC [CPU 1515SP PC] ▸ Software PLC_1 [CPU 1505SP] ▸ Program blocks ▸ OPCUA_Montrac ▸ opcShuttle1 [DB7]

Keep actual values | Snapshot | Copy snapshots to start values | Load start values as actual values

opcShuttle1

	Name	Data type	Start value	Monitor value	Retain	Accessible f...	Writa...	Visible in ...	Comment
1	Static								
2	Input	Struct							
3	ShuttleID	DWord	16#0000_0001	16#0000_0001					ID Shuttle, ktery ovladame
4	TargetToGo	DWord	16#0	16#6400_0001					Destination next stopstaion
5	InOut	Struct							
6	SendNewTarget	Bool	false	FALSE					Poslani noveho TargetID
7	Output	Struct							
8	LastChangeTargetID	DWord	16#0	16#6400_0001					Posledni poslane TargetID
9	CurrentStopStation	DWord	16#0	16#6400_0001					Aktualni StopStation voziku
10	ShuttleAvailable	Bool	false	TRUE					Vozik je na stanici a ceka na prikaz
11	OperationFinished	Bool	false	TRUE					Vozik dojel do zadane pozice
12	OperationStarted	Bool	false	FALSE					Voik vzjel ze stanice a jede do cile
13	LastShuttleRoute	"Monitoring"							Priprava na monitorovani cestz Shuttle
14	NodeID	UInt	0	1					NodeID ridici stanice TC2-Unit
15	ModulID	UInt	0	2					ModulID daneho modulu
16	IRM_Name	String	''	'IRM_123.1'					Presne oznaceni cidla
17	Error	Bool	false	FALSE					
18	InfoText	String	''	'OK'					

Obrázek 3.20: Ukázka datové bloku opcShuttle1 (vlastní tvorba)

Postup pro ovládání daného vozíku:

1. Musíme zkontrolovat jaký vozík se snažíme řídit. Například **ShuttleID** = 16#0000_0001
2. Vozík musí být dostupný na StopStation. **ShuttleAvaiable** = TRUE
3. Zjistíme na jaká StopStation se vozík nachází. **CurrentStopStation** = 16#6400_0001
4. Pokud chceme vozík poslat na jinou StopStation, musíme mu zadat do jaké StopStation má jet. Například **TargetToGo** = 16#6400_0002
5. Poté už jen pošleme příkaz do vozíku. **SendNewTarget** = TRUE
6. Poté vozík opustí tuto StopStation a vyjede do nové.

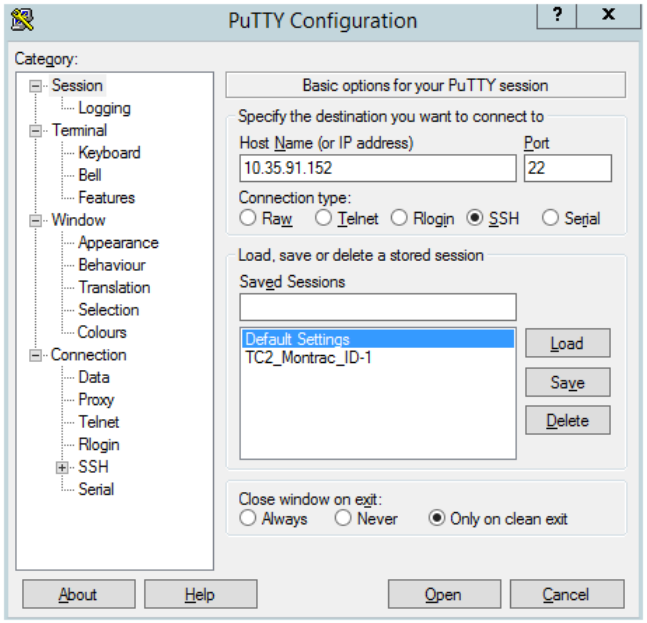
Poznámka k ostatním proměnným:

- **LastShuttleIDTargetID** - Poslední poslané TargetID do vozíku (16#6400_0002)
- **OperationFinished** - Vozík dorazil do nové StopStation (TRUE)
- **OperationStarted** - Vozík odjel ze StopStation a pohybuje se k nové (TRUE)
- **Error** - Chybová hláška (TRUE)
- **ErrorText** - Vysvětlení chybové hlášky

- **LastShuttleRoute** - Poslední modul, kterým vozík projel
  - **NodeID** - Číslo řídicí StopStation, se kterým komunikuje modul
  - **ModulID** - Číslo modulu, podle kterého se dá identifikovat
  - **IRM_Name** - Označení modul ve výkrese
- Pokud vozík dostane nové TargetID, které není přiházeno žádnému StopStation, tak vozík přijede do StopStaion_300.
- Vozík zastaví vždy ve StopStation pokud je nějaký příkaz v Chaos nebo Control Table a shoduje s ShuttleID nebo TargetID daného vozíku. Pokud tak není, vozík ihned vyjede dále.

Změna ShuttleID

Změna ShuttleID jde provádět pouze přes rozhraní PuTTY , což je klient protokolů SSH. Změna ShuttleID jde provést na jakémkoli moduly respektive čidlu IRM. Většinou se změna ShuttleID provádí na StopStation. V tomto rozhraní jde měnit i Target Address, rychlost vozíku jakou má vyjet z daného čidla, poslat vozík dozadu nebo dané čidlo monitorovat. Na obrázku 3.21 je vidět jak vypadá rozhraní PuTTY. Jako Host Name se musí zapsat IP adresa řídící jednotky, ke které je čidlo připojeno. Dále se musí vybrat **TC2_Montrac_ID-1** a kliknout na tlačítko Open. Poté už se zobrazí příkazové okno a postupuje se podle postupu níže.



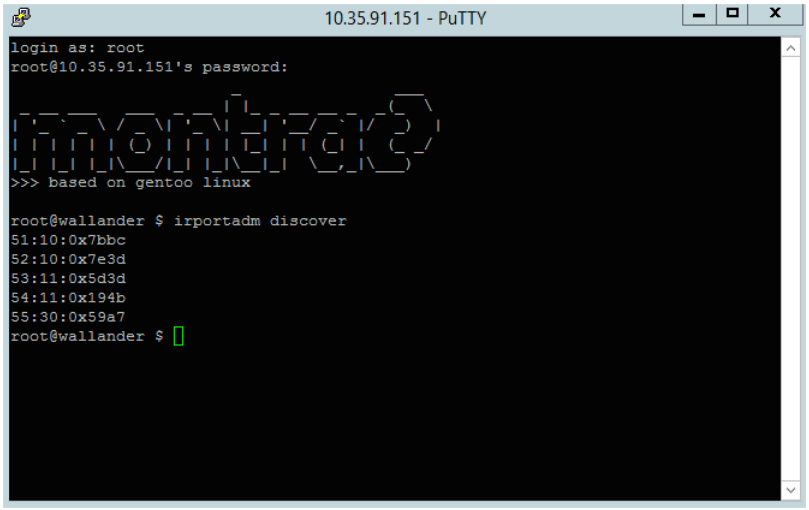
Obrázek 3.21: Ukázka rozhraní PuTTY (vlastní tvorba)

Postup pro změnu ShuttleID je následující:

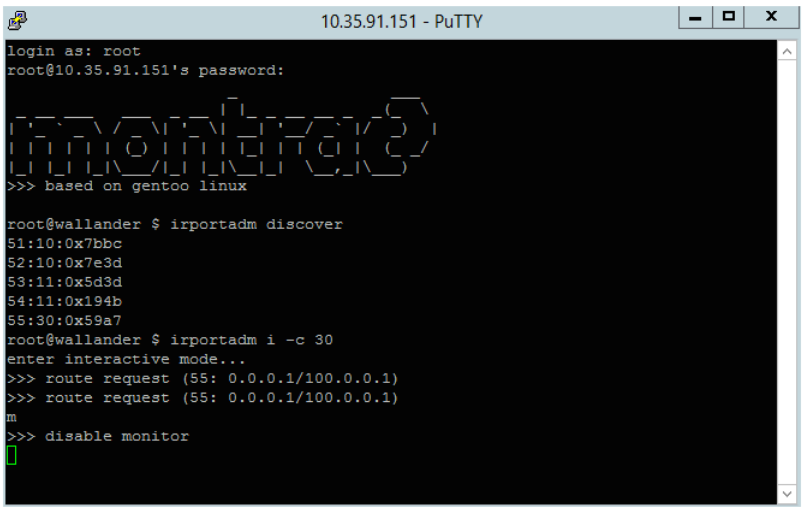
1. Ve webové rozhraní WebGui se musí deaktivovat modul, kde chceme provést změnu ShuttleID.
2. V tomto webové prostředí se také musíme podívat jaké označení IRM má daný modul (CAN adresa). Toto najdeme na stránce IRM settings
3. Poté se připojíme na SSH připojení přes rozhraní PuTTY. Na obrázku 3.22 lze vidět jako toto okno vypadá po přihlášení.
   - **Host Name**: IP adresa dané řídící jednotky. Například: 10.35.91.151
   - **Port**: 22
   - **Username**: root
   - **Password**: hellollo
4. Do příkazové řádky napíšeme příkaz "irportadm discover". Tento příkaz nám ukáže všechny moduly připojené na danou řídící jednotku.
   - Struktura vypsaných hodnot je : CAN adresa : Master ID : Random ID
5. Umístíme vozík před daný modul.
6. Vozík zprovozním a pustíme na modul.
7. Vozík by se měl zastavit na čidle IRM a čidlo by mělo zažít blikat oranžově.
8. Do příkazové řádky napíšeme příkaz "irportadm interactive -c Master ID". Na obrázku 3.23 vidíme co stane po provedení tohoto příkazu.
   - Master ID zjistíme s předchozího příkazu podle CAN adresy = IRM z WebGui
   - Po pár vteřinách by se objevit aktuální informace o vozíku, který se nachází na čidle. Jeho tvar vypadá například takto (50: 0.0.0.1 / 64.0.0.1) = (IRM: ShuttleID / TargetID)
9. Do příkazové řádky napíšeme příkaz "shid -c CAN Adres -s -t ShuttleID TargetID".
   - Tento příklad zapíše na zadaném čidle (CAN adresa) nové ShuttleID a TargetID
   - Příklad: "shid -c 55 -s -t 0.0.0.2 64.0.0.4"
10. Poté vozík upustí StopStation s novými hodnotami

Poznámka:

- Pokud napíšeme do příkazové řádky "m"vymneme a zapneme monitorování čidla
- Příkaz "h"nám vypíše nápovědu pro detailnější řízení
- Další informace se nacházejí v dokumentu [4]



Obrázek 3.22: Ukázka příkazu pro načtení připojených čidel IRM (vlastní tvorba)



Obrázek 3.23: Ukázka připojení na dané čidlo IRM (vlastní tvorba)

3.4.4 Datový blok opcStopStation

Tento datový blok slouží k ovládání a monitorování všech StopStation modulů. Pokud se do linky přidával další modul StopStation, stačí pomocí datového typu přidat další StopStation a navázat s funkčním blokem IRM_StopStation 3.3.4. Na obrázku 3.24 lze vidět jak tento datový blok vypadá. Aktuálně se v celém systému nachází šest StopStation modulů. Každému moduly jsem si přiřadil unikátní Target Address a přes WebGui 2.3.6 jsem si vytvořil fungování linky, aby vozík s daným Target Address zastavil na StopStation, která toto Target Address obsahuje. Tyto Target Address se vždy dají lehce změnit v Route Tables. Pokud vozík přijede do StopStation se správnou TargetID jako má StopStation, vozík se zastaví a čeká dokud se mu nepošle nové TargetID. Pokud vozík přijede s jinou TargetID, vozík odjede ihned pryč. Vozík odjede ihned ze stanic tehdy, pokud s Control nebo Chaos Table se neshoduje žádný příkaz (ShuttleID nebo TargetID) se kterými přijede vozík do StopStation. Na obrázku 3.24 lze vidět tento datový blok. StopStation je připojená pokud proměnná **ConnectionOK** = TRUE. Jednotlivé StopStation mají následující Target Address:

- StopStaion s označením AF126
  - **TargetID** = 16#6400_0001. Vozík s tímto TargetID přijede na tuto StopStation
  - **StopStation_100**. Toto je označení StopStation v datovém bloku
- StopStation s označením AF147
  - **TargetID** = 16#6400_0002. Vozík s tímto TargetID přijede na tuto StopStation
  - **StopStation_1**. Toto je označení StopStation v datovém bloku
- StopStation s označením AF149
  - **TargetID** = 16#6400_0003. Vozík s tímto TargetID přijede na tuto StopStation
  - **StopStation_2**. Toto je označení StopStation v datovém bloku
- StopStation s označením AF151
  - **TargetID** = 16#6400_0004. Vozík s tímto TargetID přijede na tuto StopStation
  - **StopStation_3**. Toto je označení StopStation v datovém bloku

- StopStation s označením AF102
  - **TargetID** = 16#6400_0005. Vozík s tímto TargetID přijede na tuto StopStation
  - **StopStation_200**. Toto je označení StopStation v datovém bloku
  - Na této stanici mohou být za sebou pouze dva vozíky, další vozík bude jezdit stále dokola
- StopStation s označením AF103
  - **TargetID** = 16#6400_0006. Vozík s tímto TargetID přijede na tuto StopStation
  - **StopStation_300**. Toto je označení StopStation v datovém bloku

Montrac_funguje_posilani_29.5_6.0 ▸ CPU 1515SPPC [CPU 1515SP PC] ▸ Software PLC_1 [CPU 1505SP] ▸ Program blocks ▸ OPCUA_Montrac ▸ opcStopStation [DB39]

Keep actual values | Snapshot | Copy snapshots to start values | Load start values as actual values

opcStopStation

	Name	Data type	Start value	Monitor value	Writa...	Comment
1	Static					
2	StopStation_100	"StopStaion_Block"				StopStation_126
3	StopStation_1	"StopStaion_Block"				StopStation_147
4	Input	Struct				
5	ConnectionOK	Bool	true	TRUE		Enable Connection
6	TargetToGo	DWord	16#0	16#0000_0000		Na jakou adresu ma voyik jet
7	SendNewTarget	Bool	false	FALSE		Poslani nove adresz do voyiku
8	LastSendTarget	DWord	16#0	16#0000_0000		Posledni poslana adresa voziku
9	LockStation	Bool	false	FALSE		StopStation je zamcena
10	UnlockStation	Bool	false	FALSE		StopStation je odemcena
11	ChaosTable	Struct				
12	RouteNumber	Byte	16#1	16#01		Cislo cesty do ktere chceme poslat prikaz. U StopStation vzdy 1
13	Paths	Struct				
14	SendPath	Bool	false	FALSE		Poslani novych cestu do Chaostable. Vsechny cestz se prepisou
15	Path	Array[1..5] of "Cha...				jake cesty se maji poslat do ChaosTable
16	List	Struct				
17	SendList	Bool	false	FALSE		Vypise aktualni ChaosTable
18	List	Array[1..5] of "Cha...				Tabulka ChaosTable
19	ControlTable	Struct				
20	RouteNumber	Byte	1	16#01		Cislo cesty do ktere chceme poslat prikaz. U StopStation vzdy 1
21	NewLine	Struct				Pridani noveho prikazu do ControlTable
22	SendNewLine	Bool	false	FALSE		
23	NewTargetID	DWord	16#0	16#0000_0000		Nove TargetID do ControlTable
24	NewShuttleID	DWord	16#0	16#0000_0000		Nove ShuttleID do ControlTable
25	ClearList	Bool	false	FALSE		Vymaze vsechny prikazy v ControlTable
26	ClearLine	Bool	false	FALSE		Vymaze prvni prikaz v ControlTable
27	Output	Struct				
28	LastShuttleID	DWord	16#0	16#0000_0002		Posledni Shuttle na stanici
29	LastTargetID	DWord	16#0	16#6400_0002		Posledni TagetID na stanici s ShuttleID
30	CurrentShuttleID	DWord	16#0	16#0000_0000		Aktualni vozik na stanici
31	CurrentTargetID	DWord	16#0	16#0000_0000		Aktualni targetID na stanici
32	ShuttleAvailable	Bool	false	FALSE		Vozik je dostupny
33	ShuttleArrival	Bool	false	FALSE		Vozik prijizdi do stanice
34	ShuttleWaiting	Bool	false	FALSE		Vozik ceka na povel
35	ShuttleProcesing	Bool	false	FALSE		Vozik zpracovava data
36	ShuttleTransfer	Bool	false	FALSE		Vozik komunikuje se stanici
37	ShuttleDeparture	Bool	false	FALSE		Vozik odjede ze stanice
38	NodeID	UInt	0	2		NodeID TC2-Unit
39	ModulID	UInt	0	7		ModulID tohoto modulu
40	Header_Status	String	''	'OK'		Status, zda je vse v poradku
41	Status	Int	0	0		Error status
42	StopStation_2	"StopStaion_Block"				StopStation_149
43	StopStation_3	"StopStaion_Block"				StopStation_151
44	StopStation_200	"StopStaion_Block"				StopStation_102

Obrázek 3.24: Ukázka datového bloku opcStopStation (vlastní tvorba)

Příjezd vozíku do StopStation:

Pokud vozík přijede do nové StopStation, měl by se ihned objevit v datovém bloku. Občas vozík přijede a musí se počkat než se čidlo zakomunikuje s řídící jednotkou a poté se až vozík ukáže že přijel.

Poslání vozíku do nové StopStation:

1. Vozík musí být na StopStation dostupný. **ShuttleAvaiable** = TRUE
2. Pokud chceme vozík poslat na jinou StopStaion, musíme říct do jaké nové StopStation má vozík jet. Například **TargetToGo** = 16#6400_0002
3. Poté už jen pošleme příkaz do vozíku. **SendNewTarget** = TRUE
4. Poté vozík opustí tuto StopStaion a vyjede do nové
5. V proměnné **LastSendTargetID** vidíme poslední poslané TargetID (16#6400_0002)

Ovládání proměnné Chaos Table:

Princip Chaos Table je vysvětlený v kapitole číslo 2.3.3.

- **RouteNumber** - Toto číslo musí být vždy 1, protože StopStation má pouze jednu out-links a jednu in-links
- **Paths:**
  1. Do proměnné **Path** se napíšou všechny příkazy, které chceme poslat do tabulky Chaos Table. Formát zápisu je popsán takto:
     - **FromTargetID** - Rozmezí od jakého TargetID, například 16#6400_0002
     - **UntilTargetID** - Rozmezí do jakého TargetID, například 16#6400_0003
     - **FromShuttleID** - Rozmezí od jakého ShuttleID, například 16#6400_0001
     - **UntilShuttleID** - Rozmezí do jakého ShuttleID, například 16#6400_0005
  2. Poté se příkaz pošle pomocí **SendPath** = TRUE. Hodnoty vždy přepíšou aktuální tabulku.
- **List:**
  1. Příkazem **SendList** = TRUE se provede vypsání aktuální Chaos Table.
  2. V proměnné **List** se tato tabulka vypíše.

Ovládání proměnné Control Table:

Princip Chaos Table je vysvětlený v kapitole číslo 2.3.3.

- **RouteNumber** - Toto číslo musí být vždy 1, protože StopStation má pouze jednu out-links a jednu in-links
- **NewTarget:**
  1. **NewTargetID** - TargetID, které chceme poslat do Tabulky Control Table
  2. **NewShuttleID** - ShuttleID, které chceme poslat do Tabulky Control Table
  3. Poté se příkaz pošle pomocí **Send** = TRUE. Do Control Table se zapíše příkaz
- Pokud chceme vymazat celou tabulku použijeme **ClearList** = TRUE.
- Pokud chceme vymazat pouze první řádek použijeme **ClearLine** = TRUE

Použití pneumatické pístu k upevnění desky:

- **LockStation** - Pokud se nachází v hodnotě TRUE. Pneumatický píst je vysunutý
- **TakeDesk** - Pokud se nachází v hodnotě TRUE. Pneumatický píst je zasunutý

Monitorování StopStation:

- **LastShuttleID** - Poslední vozík, který byl na této StopStation
- **LastTargetID** - Poslední TargetID, který byl na této StopStation
- **CurrentShuttleID** - Vozík, který se aktuálně nachází na StopStation
- **CurrentTargetID** - Vozík s aktuálním Target ID na StopStation
- **ShuttleAvailable** - Vozík je dostupný a může se mu poslat příkaz
- **ShuttleArrival** - Vozík přijíždí do StopStation
- **ShuttleWaiting** - Vozík čeká na příkaz
- **ShuttleProcessing** - Vozík zpracovává data ze StopStation
- **ShuttleTransfer** - Vozík komunikuje se StopStation
- **ShuttleDeparture** - Vozík opouští aktuální StopStation
- **NodeID** - Číslo řídící jednotky se kterou komunikuje tento modul

- **ModulID** - Jedinečné číslo tohoto moduly, podle kterého může komunikovat s řídící jednotkou
- **Header_Status** - Error status, zda je vše v pořádku. Pokud 'OK' vše je v pořádku
- **Status** - Error status, zda je vše v pořádku. Pokud 0 vše je v pořádku

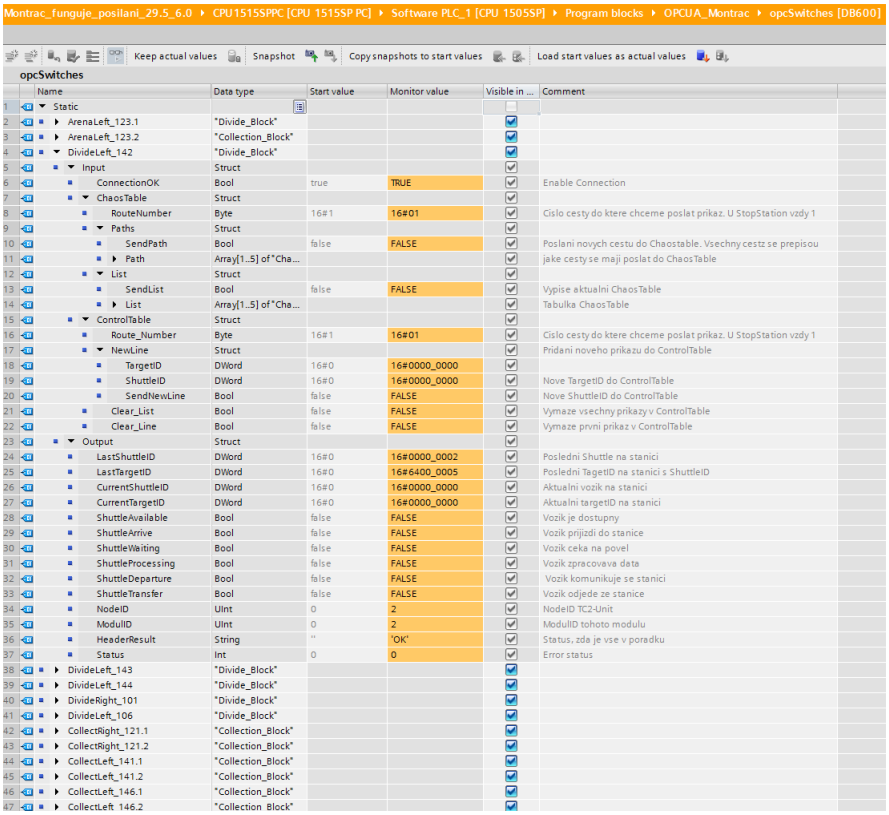
Poznámka:

- Pokud vozík špatně komunikuje s čidlem, je možné že váleček který je v modulu se uvolnit a musí se zkontrolovat
- Občas se vozík přijede do StopStation a chvíli trvá než se načte. Je to tím, že řídící jednotka pomaleji a musí se tedy počkat. Toto by se nemělo stávat pravidelně.

3.4.5 Datový blok opcSwitches

Datový blok slouží k ovládání a monitorování všech modulů TracSwitch divide, TracSwitch collect a TracSwitch arena. Pokud by se do linky přidával další modul , stačí přidat příslušný datový typ s názvem Collection_Block nebo Divide_Block (TracSwitch divide a TracSwitch arena mají stejný datový typ) podle toho o jaký modul se bude jednat a navázat s funkčním blokem IRM_Divide 3.3.3 nebo IRM_Collect 3.3.2. Pokud se přidává modul TracSwitch arena, záleží jaký datový typ se má použít podle toho zda do čidla můžeme zapisovat nějakou cestu nebo jen monitorovat. Na obrázku 3.25 lze vidět vypadá datový typ pro TracSwitch divide a na obrázku 3.26 lze vidět jak vypadá datový typ pro TracSwitch collect.

Modul je připojený pokud proměnná **ConnectionOK** = TRUE.



Obrázek 3.25: Ukázka datového bloku opcSwitches pro TracSwitch divide (vlastní tvorba)

Ovládání Chaos Table:

Princip Chaos Table je vysvětlený v kapitole číslo 2.3.3.

- **RouteNumber** - Cesta, kterou vozík pojede. U TracSwitch divide je možnost s vybrat RouteNumber = 1 nebo RouteNumber = 2. Ve WebGui lze zjisti detaily ohledně těchto modulů a jejich RouteNumber.
- **Paths:**
  1. Do proměnné **Path** se napíšou všechny příkazy, které chceme poslat do tabulky. Formát zápisu je popsán takto:
     - **FromTargetID** - Rozmezí od jakého TargetID, například 16#6400_0002.
     - **UntilTargetID** - Rozmezí od jakého TargetID, například 16#6400_0003.
     - **FromShuttleID** - Rozmezí od jakého ShuttleID, například 16#6400_0001.
     - **UntilShuttleID** - Rozmezí od jakého ShuttleID, například 16#6400_0005.
  2. Poté se příkaz pošle pomocí **SendPath** = TRUE. Hodnoty vždy přepíšou aktuální tabulku.

- **List:**
  1. Příkazem **SendList** = TRUE se provede vypsání aktuální Chaos Table.
  2. V proměnné **List** se tato tabulka vypíše.

Ovládání Control Table:

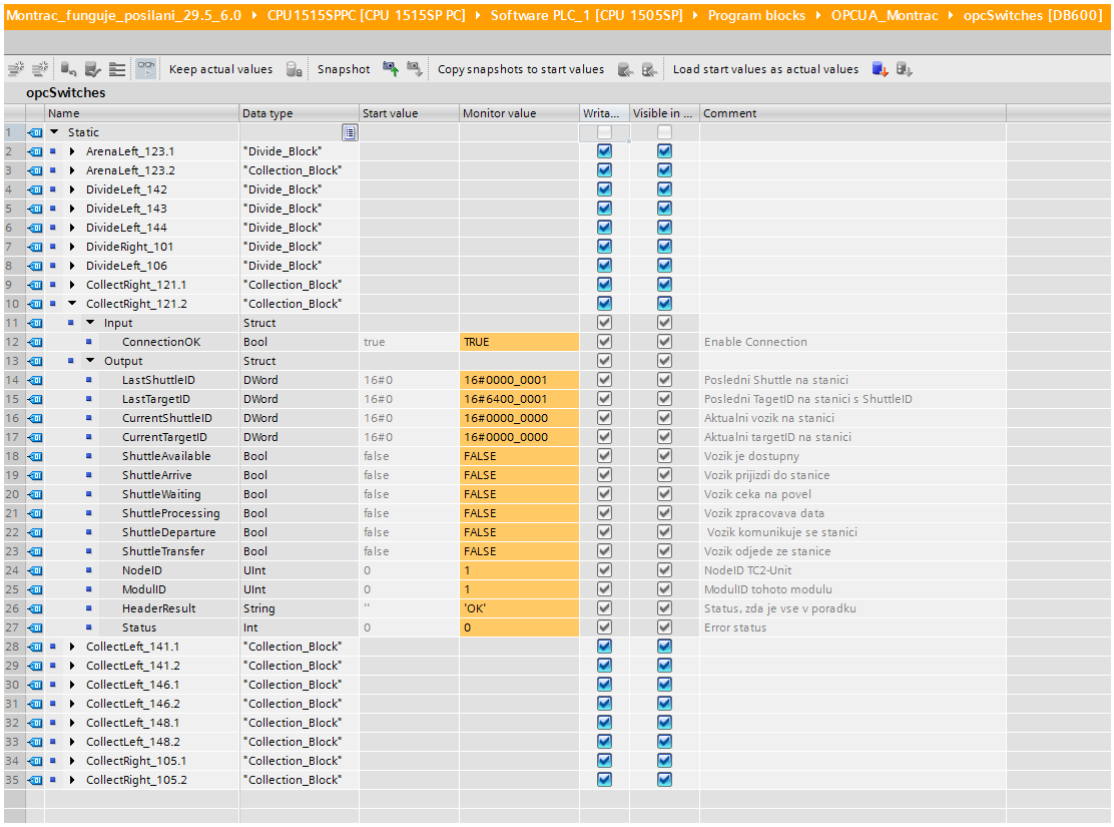
Princip Chaos Table je vysvětlený v kapitole číslo 2.3.3.

- **RouteNumber** - Cesta, kterou vozík pojede. U TracSwitch divide je možnost si vybrat RouteNumber = 1 nebo RouteNumber = 2. Ve WebGui lze zjisti detaily ohledně těchto modulů a jejich RouteNumber.
- **NewTarget:**
  1. **NewTargetID** - TargetID, které chceme poslat do Tabulky Control Table
  2. **NewShuttleID** - ShuttleID, které chceme poslat do Tabulky Control Table
  3. Poté se příkaz pošle pomocí **Send** = TRUE. Do Control Table se zapíše tento příkaz
- Pokud chceme vymazat celou tabulku použijeme proměnnou **ClearList** = TRUE.
- Pokud chceme vymazat pouze první řádek v Control Table použijeme proměnnou **ClearLine** = TRUE

Monitorování TracSwitch divide:

- **LastShuttleID** - Poslední vozík, který byl na této StopStation
- **LastTargetID** - Poslední TargetID, který byl na této StopStation
- **CurrentShuttleID** - Vozík, který se aktuálně nachází na StopStation
- **CurrentTargetID** - Vozík s aktuálním Target ID na StopStation
- **ShuttleAvailable** - Vozík je dostupný a může se mu poslat příkaz
- **ShuttleArrival** - Vozík přijíždí do StopStation
- **ShuttleWaiting** - Vozík čeká na příkaz
- **ShuttleProcessing** - Vozík zpracovává data ze StopStation
- **ShuttleTransfer** - Vozík komunikuje se StopStation

- **ShuttleDeparture** - Vozík opouští aktuální StopStation
- **NodeID** - Číslo řídící jednotky se kterou komunikuje tento modul
- **ModulID** - Jedinečné číslo tohoto moduly, podle kterého může komunikovat s řídící jednotkou
- **Header_Status** - Error status, zda je vše v pořádku. Pokud 'OK' vše je v pořádku
- **Status** - Error status, zda je vše v pořádku. Pokud 0 vše je v pořádku



	Name	Data type	Start value	Monitor value	Writa...	Visible in ...	Comment
1	Static						
2	ArenaLeft_123.1	"Divide_Block"					
3	ArenaLeft_123.2	"Collection_Block"					
4	DivideLeft_142	"Divide_Block"					
5	DivideLeft_143	"Divide_Block"					
6	DivideLeft_144	"Divide_Block"					
7	DivideRight_101	"Divide_Block"					
8	DivideLeft_106	"Divide_Block"					
9	CollectRight_121.1	"Collection_Block"					
10	CollectRight_121.2	"Collection_Block"					
11	Input	Struct					
12	ConnectionOK	Bool	true	TRUE			Enable Connection
13	Output	Struct					
14	LastShuttleID	DWord	16#0	16#0000_0001			Posledni Shuttle na stanici
15	LastTargetID	DWord	16#0	16#6400_0001			Posledni TagetID na stanici s ShuttleID
16	CurrentShuttleID	DWord	16#0	16#0000_0000			Aktualni vozik na stanici
17	CurrentTargetID	DWord	16#0	16#0000_0000			Aktualni targetID na stanici
18	ShuttleAvailable	Bool	false	FALSE			Vozik je dostupny
19	ShuttleArrive	Bool	false	FALSE			Vozik prijizdi do stanice
20	ShuttleWaiting	Bool	false	FALSE			Vozik ceka na povel
21	ShuttleProcessing	Bool	false	FALSE			Vozik zpracovava data
22	ShuttleDeparture	Bool	false	FALSE			Vozik komunikuje se stanici
23	ShuttleTransfer	Bool	false	FALSE			Vozik odjede ze stanice
24	NodeID	UInt	0	1			NodeID TC2-Unit
25	ModulID	UInt	0	1			ModulID tohoto modulu
26	HeaderResult	String	''	'OK'			Status, zda je vse v poradku
27	Status	Int	0	0			Error status
28	CollectLeft_141.1	"Collection_Block"					
29	CollectLeft_141.2	"Collection_Block"					
30	CollectLeft_146.1	"Collection_Block"					
31	CollectLeft_146.2	"Collection_Block"					
32	CollectLeft_148.1	"Collection_Block"					
33	CollectLeft_148.2	"Collection_Block"					
34	CollectRight_105.1	"Collection_Block"					
35	CollectRight_105.2	"Collection_Block"					

Obrázek 3.26: Ukázka datového bloku opcSwitches pro TracSwitch collect (vlastní tvorba)

Datový typ **TracSwitch collect** slouží pro monitorování vozíku. Vozík tímto modulem pouze projíždí a tudíž do něj nelze zapisovat žádný Control nebo Chaos Table ani posílat nové TargetID vozíku.

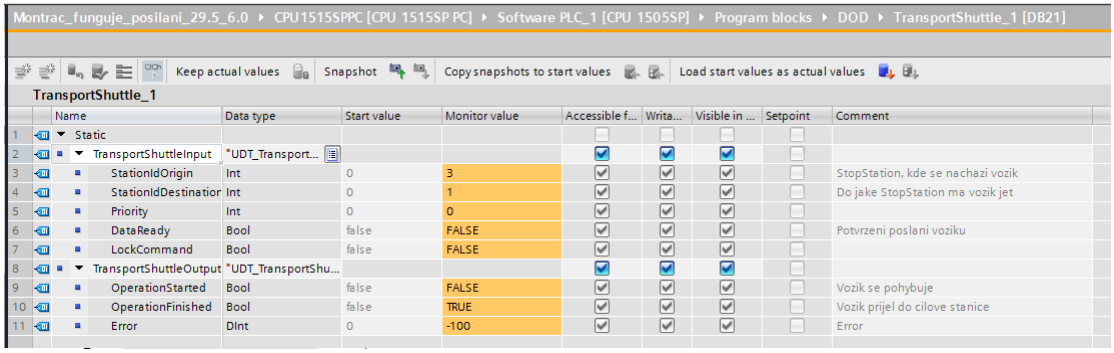
Monitorování TracSwitch collect:

- **LastShuttleID** - Poslední vozík, který byl na této StopStation
- **LastTargetID** - Poslední TargetID, který byl na této StopStation

- **CurrentShuttleID** - Vozík, který se aktuálně nachází na StopStation
- **CurrentTargetID** - Vozík s aktuálním Target ID na StopStation
- **ShuttleAvailable** - Vozík je dostupný a může se mu poslat příkaz
- **ShuttleArrival** - Vozík přijíždí do StopStation
- **ShuttleWaiting** - Vozík čeká na příkaz
- **ShuttleProcessing** - Vozík zpracovává data ze StopStation
- **ShuttleTransfer** - Vozík komunikuje se StopStation
- **ShuttleDeparture** - Vozík opouští aktuální StopStation
- **NodeID** - Číslo řídící jednotky se kterou komunikuje tento modul
- **ModulID** - Jedinečné číslo tohoto moduly, podle kterého může komunikovat s řídící jednotkou
- **Header_Status** - Error status, zda je vše v pořádku. Pokud 'OK' vše je v pořádku
- **Status** - Error status, zda je vše v pořádku. Pokud 0 vše je v pořádku

3.4.6 Datový blok TransportShuttle

Tento datový blok byl vytvořen především pro to, aby nadřazeny systém mohl ovládat vozíky na lince. Nadřazený protokol, který tento systému může ovládat používá protokol OPC UA . U tohoto blok se především museli změnit datový typy a to z formátu DWord na formát INT, aby se vozík přes protokol OPC lehce ovládal. Proto tento blok jsou vytvořeny dva datové typy ve složce OPCUA. Pro každý vozík se musí přidat vždy jeden tento datový blok. Jeho parametry jsou popsány níže. Na obrázku 3.27 lze tento blok vidět.



Obrázek 3.27: Datový blok, vytvoření pro ovládáni přes OPC server (vlastní tvorba)

- **StationIdOrigin** - Aktuální StopStation vybraného vozíku
- **StationIdDestination** - StopStation do které chceme vozík poslat
- **Priority** - Pouze příprava na další práci s touto proměnou
- **DataReady** - Poslání nové TargetID
- **LockCommadn** - Uzamykaní StopStation pomocí pneumatického pístu (Pouze ilustrativně, píst se ovládá přes StopStation)
- **OperationStarted** - Vozík je na cestě do nové StopStation
- **OperationFinished** - Vozík se nachází na StopStation
- **Error** - pokud je Error = - 100, znamená to, že vozík je nedostupný

Poznámka k ovládání

- StopStation mají označení jiné než je uvedené v bloku opcStopStation
  - StopStation_100 = 1
  - StopStation_1 = 2
  - StopStation_2 = 3
  - StopStation_3 = 4
  - StopStation_200 = 5
  - StopStation_300 = 6

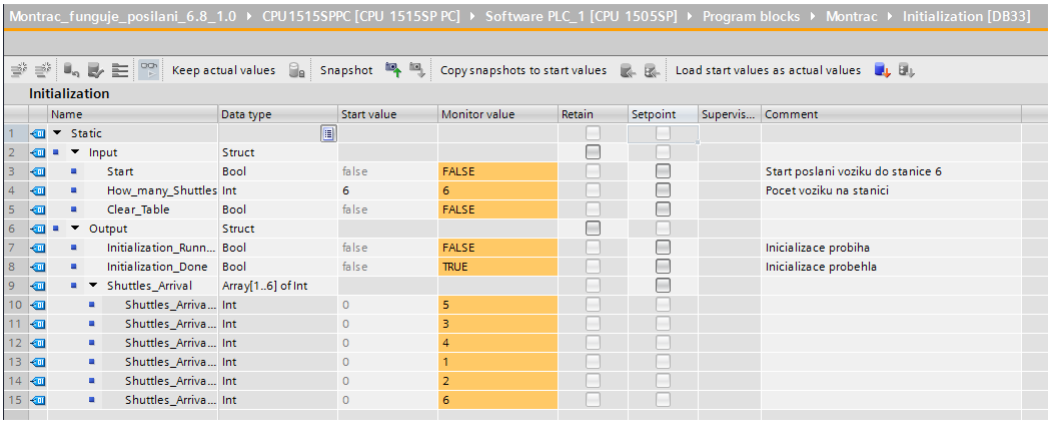
3.5 Inicializace a výpadek spojení

3.5.1 Inicializace

Při použití datového bloku **Initialization** se všechny vozíky automaticky seřadí na StopStation -AF107 (StopStation číslo 6). A v tabulce **Shuttles_Arrival** lze vidět, jak se vozíky za sebou seřadily. Tato tabulka se vždy vyprázdní při spuštění inicializace nebo se může ručně vyprázdnit pomocí proměnné **Clear_Table**. Tato inicializace probíhá po dobu 49 vteřin, což je dostatečná doba pro vyjetí všech vozíků ze StopStation. Na obrázku 3.28 je vidět jak tento blok vypadá.

Popis jednotlivých proměnných:

- Start - Spustí se příkaz inicializace a všechny vozíky se seřadí do StopStation
- **How_many_Shuttles** - Do této proměnné se zapíše počet vozíků na licence.
  - Například, pokud budou na lince pouze čtyři vozíky, hodnota musí být 4
  - Jako základní hodnota je nastavena 6, pro všechny vozíky
- **Clear_Table** - Vyprázdní tabulky **Shuttles_Arrival**
- **Initialization_Running** - Inicializace probíhá
- **Initialization_Done** - Inicializace proběhla
- **Shuttles_Arrival** - Tabulka, kde jsou vozíky seřazeny za sebou na StopStaion



	Name	Data type	Start value	Monitor value	Retain	Setpoint	Supervis...	Comment
1	Static							
2	Input	Struct						
3	Start	Bool	false	FALSE				Start poslani voziku do stanice 6
4	How_many_Shuttles	Int	6	6				Pocet voziku na stanici
5	Clear_Table	Bool	false	FALSE				
6	Output	Struct						
7	Initialization_Runn...	Bool	false	FALSE				Inicializace probiha
8	Initialization_Done	Bool	false	TRUE				Inicializace probehla
9	Shuttles_Arrival	Array[1..6] of Int						
10	Shuttles_Arriva...	Int	0	5				
11	Shuttles_Arriva...	Int	0	3				
12	Shuttles_Arriva...	Int	0	4				
13	Shuttles_Arriva...	Int	0	1				
14	Shuttles_Arriva...	Int	0	2				
15	Shuttles_Arriva...	Int	0	6				

Obrázek 3.28: Datový blok, pro inicializaci vozíků (vlastní tvorba)

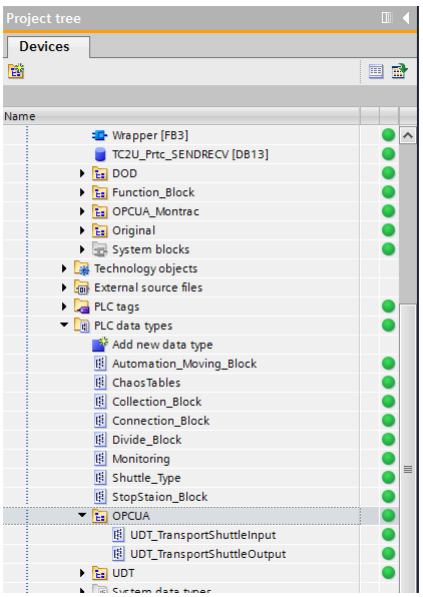
3.5.2 Výpadek spojení

Pokud se zmáčkne červené tlačítko a do linky přestane proudit napájení a řídící jednotky přestanou komunikovat s PLC. U všech StopStation se zasunou pneumatické písty (toto je navázáno na proměnou v datovém bloku Connection_TC2-Unit na Error -50), aby byl vozík volný . Po opětovném zapnutí, červeného tlačítka se vysune a zmáčkne se modré tlačítko (toto tlačítko zmáčknout až se zasunou písty), která jsou kolem linky, se automaticky po chvíli opět nahodí a začnou komunikovat. Pokaždé když se nahodí opět napájení vozíky se popojedou zpět a dopředu, pro inicializaci na modulech. Všechny modulu se vyzkoušejí tím, že se protočí výhybky na modulech.

Funkční bloky pro připojení s řídícími jednotky se po chvíli opět připojí (přibližně do 30 vteřin) a linka se dá ovládat. Pokud se nedá připojit, je nejlepší podívat se do datového bloku **Connection_TC2-Units** (viz kapitola 3.4.2) a lze zkusit u proměnné **Start_TCON** 2x změnit hodnotu s TRUE na FALSE a opět na TRUE. Pokud ani toto nepomůže, je nejlepší proměnou **iX_Communication_Start** nahodit do hodnoty FALSE a to samé i u **Start_TCON** a **Start_UDT**. Poté opět nahodit proměnou **iX_Communication_Start** do TRUE a jednotka by se měla připojit. Posledním možným krokem je restartovat PLC nebo nahrát do PLC celý software a tím se hodnoty hodí do základních hodnot.

3.6 Vytvořené datové typy

Pro ulehčení práce, jsem si vytvořil vlastní datové typy. Ty jsou použit u datových bloků. Jsou nalezení v Project Tree ve složce PLC Data Type.



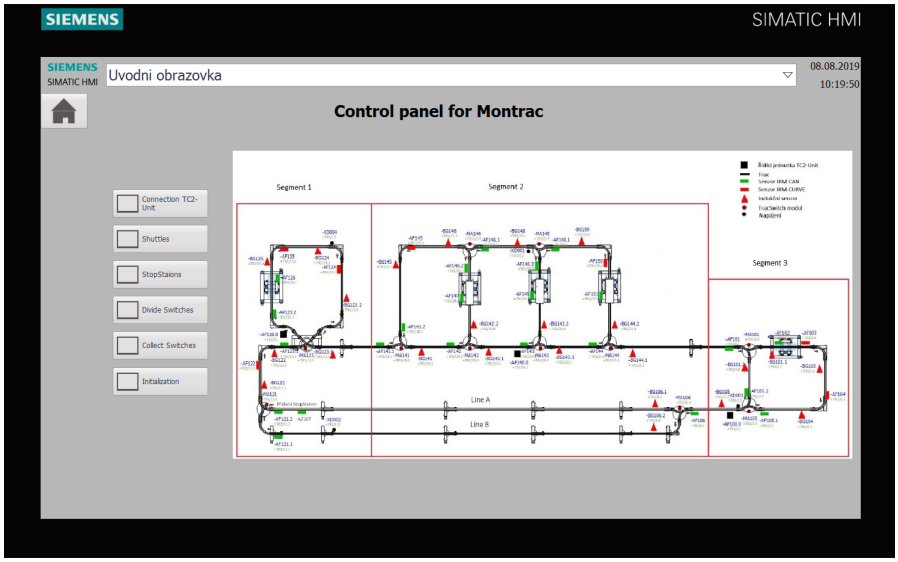
Obrázek 3.29: Datové typy, které jsou vytvořeny pro tento projekt (vlastní tvorba)

Kapitola 4

Lokální operátorský panel

Poslední částí mé diplomové práce bylo vytvořit operátorský panel Human Machine Interface (HMI) pro řízení linky. U linky žádný panel není, proto je tento panel pouze simulovaný. Pro demonstraci jsem si vybral panel T1500 Comfort PRO. Tento panel slouží pro přehlednější, rychlejší a lehčí ovládání linky. Celý panel je vytvořen ve vývojovém prostředí WinCC, který je zakomponovaný ve vývojovém prostředí TiaPortal. Human Machine Interface (HMI) představuje rozhraní mezi zařízením a člověkem (obsluha, operátor). HMI představuje prostředky pro zobrazeni a předání informace o stavu zařízení (stav, hodnoty) obsluze nebo operátorovi a zároveň poskytuje možnost ovládání stroje a zadávání hodnot. Často se používá i výraz Dispečerské řízení a sběr dat (SCADA) což je systém pro vizualizaci, ovládání a supervizi technologických provozů. Oba tyto názvy se používají často pospolu. HMI je obvykle propojen s centralizovanými databázemi SCADA. [31]

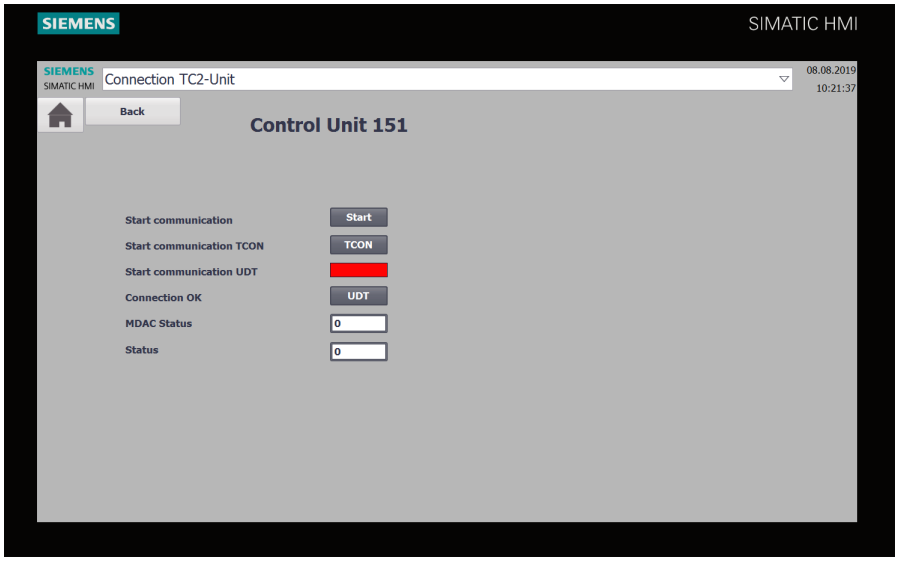
Na úvodní stránce lze vidět půdorys linky a také stránky na které je možno dále pokračovat. Jejich názvy jsem volil podle datový bloků zmíněných výše. Úvodní stránku lze vidět na obrázku 4.1. Konfigurace tohoto panelu je ve složce HMI_1.



Obrázek 4.1: Úvodní stránka operátorského panelu (vlastní tvorba)

4.0.1 Connection TC2-Unit

Tato stránka slouží k ovládání všechny tří řidičích jednotky. Jednotlivé proměnné jsou popsány v kapitole 3.4.2. Tuto stránku lze vidět na obrázku 4.2.

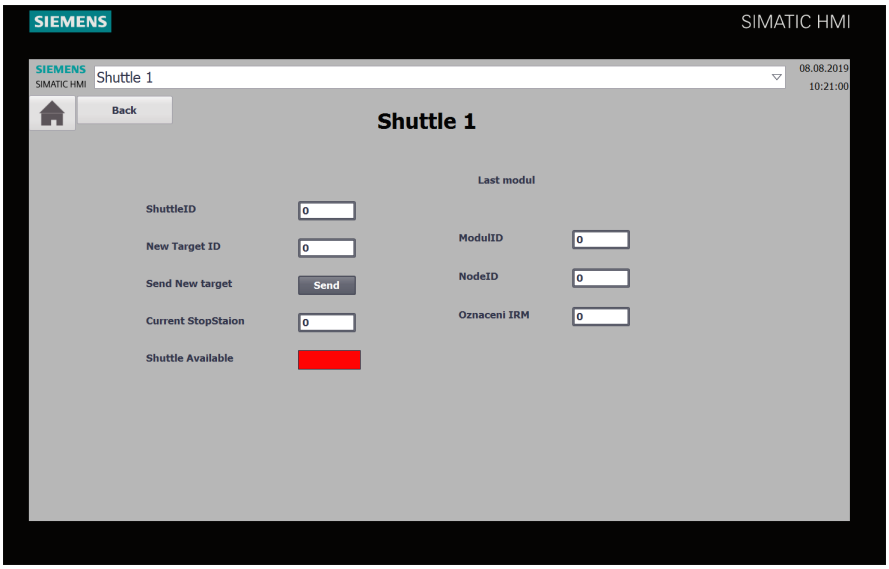


Obrázek 4.2: Stránka s ovládáním řídících jednotek (vlastní tvorba)

4.0.2 Shuttles

Tato stránka slouží k řízení a monitorování všechny šesti vozíků, které se pohybují na lince. Každý vozík má svoji stránku, kde jsou vidět jeho informace. Proměnné jsou brané

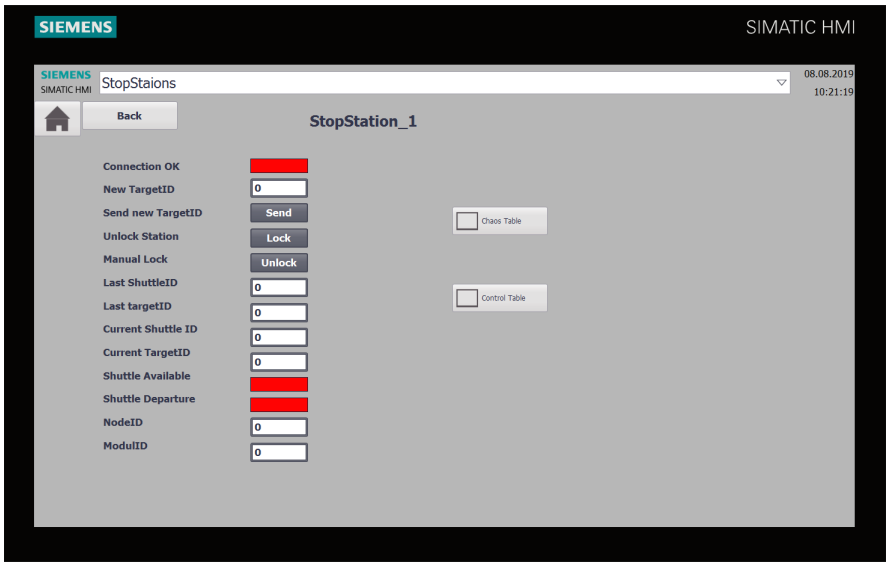
z datových bloků opcShuttle. Jednotlivé proměnné jsou popsány v kapitole 3.4.3. Tuto stránku lze vidět na obrázku 4.3.



Obrázek 4.3: Stránka s ovládáním jednotlivých vozíků (vlastní tvorba)

4.0.3 StopStations

Tato stránka slouží k ovládání všechen pět StopStation modulů, které se nacházejí na lince. Každý tento modul má svoji stránku, kde lze vidět komplexní informace o každé StopStation. Popis jednotlivých proměnných je popsán v kapitole výše viz 3.4.4. Tuto stránku lze vidět na obrázku 4.4.



Obrázek 4.4: Ovládání a monitorování StopStations (vlastní tvorba)

4.0.4 Divide Switches

Tato stránka slouží k zapisování a monitorová všechen Divide modulů. Každý tento modul má svoji stránku, kde lze vidět komplexní informace o každém tomto modulu. Popis jednotlivých proměnných je popsán v kapitole výše viz 3.4.5. Tuto stránku lze vidět na obrázku 4.5.

Obrázek 4.5: Stránka se vsemi Divide moduly (vlastní tvorba)

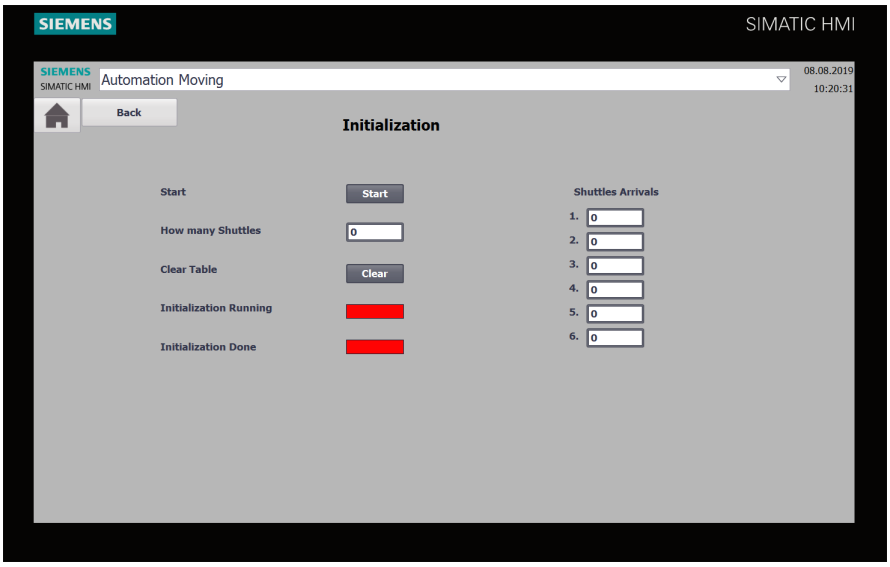
4.0.5 Collect Switches

Tato stránka slouží k zapisování a monitorová všechen Collect modulů. Každý tento modul má svoji stránku, kde lze vidět komplexní informace o každém modulu. Popis jednotlivých proměnných je popsán v kapitole výše viz 3.4.5. Tuto stránku lze vidět na obrázku 4.6.

Obrázek 4.6: Stránka se všemi Collect moduly (vlastní tvorba)

4.0.6 Inicializace

Tato stránka slouží k ovládání inicializace vozíků na lince, která je popsána výše viz kapitola 3.5.1. Tuto stránku lze vidět na obrázku 4.7.



Obrázek 4.7: Stránka pro odvládni inicializace vozíků (vlastní tvorba)

Kapitola 5

Závěr

Hlavním cílem diplomové práce bylo naprogramovat funkční řízení pro inteligentní dopravníkový systém s vozíky, který je jádrem modelu robotické montážní linky s vysokou úrovní variability výroby, automatickým inteligentním rozvrhováním a plánováním pomocí nadrženého decentralizačního systému. Tato linka má ukázat továrnu v duchu vize Industry 4.0.

V teoretické části jsem nejdříve popsal mechanické prvky celé linky včetně modulů, které fungují jako strukturní prvky propůjčující funkci jednotlivým úsekům linky. Dále jsem popsal použitou senzoriku, podstatnou pro správné fungování linky a také vozíky pohybující se po lince. Vysvětlena je také řídící jednotka důležitá pro komunikaci mezi linkou a PLC. Dále je v textu rozebrána komunikace mezi jednotlivými komponenty, která funguje přes sběrnici CAN a komunikační protokol UDP.

Pro ovládání linky je vytvořený komunikační prokol MDAC firmou Montrac, který jsem musel nastudovat. Poté jsem tento protokol popsal a použil ho při vytvoření programu na ovládání linky. V neposlední řadě jsem také popsal softwarové PLC běžící na průmyslovém počítači, přes které je celá linka ovládána. V poslední kapitole teoretické části jsou popsány termíny MES, ERP a PLM.

Hlavním bodem diplomové práce byla praktická část. Pracoval jsem ve vývojovém prostředí TiaPortal a vytvořil řízení pro tuto linku. Prvním krokem bylo vytvoření funkčních bloků a datových bloků pro každý modul přes které se dá linka ovládat. Vozíky se mohou posílat z jedné StopStation na druhou a každý vozík a modul jde monitorovat. Pokud je vozík ve StopStation, automaticky se zamyká píst. Také lze posílat příkazy do Control a Chaos Table, které slouží pro správnou volbu cesty vozíku z modulu. V neposlední řadě jsem vytvořil inicializaci vozíků, kdy se vozíky seřadí na jedné StopStation. Pokud dojde k výpadku proudu, otevřou se u všech StopStation písty, aby vozíky bylo možné odebrat. Po opětovném nahození proudu se řídící jednotky automaticky připojí.

Tato práce by měla také sloužit jako návod, jak s linkou pracovat a ovládat ji. Všechny bloky jsou podrobně popsány. V posledním bodě jsem vytvořil operátorský panel HMI.

V závěru bylo nutné celý systém otestovat a optimalizovat jeho fungování.

Bibliografie

[1] "Cvut transportsystem, Tento dokumente byl vytvořen k této lince.", 13.6.2018.

[2] (2017). Montrac intelligent in motion, Design guide 2017/2018, WWW: https://www.montratec.de/fileadmin/user_upload/downloads/public/default/montratec_Design_Guide_2018-12_ENG.pdf (cit. 23. 07. 2019).

[3] (2014). Bezkontaktní indukční snímače přiblížení - obecný popis, WWW: https://automatizace.hw.cz/komponenty-mereni-a-regulace/indukcni-snimace-priblizeni-obecny-popis.html (cit. 21. 07. 2019).

[4] "Schmid, assembly instruction montrac, Tento dokumente byl vytvořen k této lince.", 25.4.2016.

[5] "Mdac. montrac data acquisition and control. mdac-protocoll-spezifikation, Tento dokumente byl vytvořen k této lince.", 1.5.2018.

[6] "Assembly instructions montrac", WWW: https://www.yumpu.com/en/document/read/49790050/18-shuttle-3-montratec-ag (cit. 08. 05. 2019).

[7] (2014). Fram – vestavěná paměť s velmi nízkou spotřebou, WWW: http://www.ti.com/lit/ml/slat151/slat151.pdf (cit. 05. 05. 2019).

[8] (2009). Understanding syslog: Servers, messages & security, WWW: https://www.networkmanagementsoftware.com/what-is-syslog/ (cit. 08. 05. 2019).

[9] (2010). Definition - what does busybox mean?, WWW: https://www.techopedia.com/definition/27267/busybox (cit. 08. 05. 2019).

[10] K. Polák. (). Sběrnice CAN, WWW: http://www.elektrorevue.cz/clanky/03021/index.html (cit. 23. 07. 2019).

[11] (1999). CAN in automation (CiA): CAN knowledge, WWW: https://www.can-cia.org/can-knowledge/ (cit. 23. 07. 2019).

[12] "Systémy scada a nástroje pro sběr, vizualizaci a analýzu průmyslových dat", Bakalářská práce, ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE FAKULTA STROJNÍ, Praha, 2017.

[13] P. Goncharov, "Interaktivní simulátor sítí pro analýzu a vizualizaci protokolů", Magisterská práce, České vysoké učení technické v Praze, České vysoké učení technické v Praze. Výpočetní a informační centrum., 25.1.2018.

[14] "Udp a tcp komunikace na fitkitu", Bakalářská práce, VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ, Brno, 2010.

[15] (2010). Chapter 4: Configuring PuTTY | PuTTY user manual (putty-0.68-manual), WWW: https://www.ssh.com/ssh/putty/putty-manuals/0.68/Chapter4.html#config (cit. 23. 07. 2019).

[16] P. Krčmář. (). Útoky na SSH: Je ještě bezpečné?, Root.cz, WWW: https://www.root.cz/clanky/utoky-na-ssh-je-jeste-bezpecne/ (cit. 23. 07. 2019).

[17] "Softwarové plc s7-1500", Diplomová práce, Univerzita Pardubice, Univerzita Pardubice, 2017.

[18] (2005). Programování plc podle normy iec en 61131-3 – víc než jednotné jazyky, WWW: http://automa.cz/cz/casopis-clanky/programovani-plc-podle-normy-iec-en-61131-3-vic-nez-jednotne-jazyky-2005_02_30310_1237/ (cit. 14. 05. 2019).

[19] (2014). ELUC, WWW: https://eluc.kr-olomoucky.cz/verejne/lekce/976 (cit. 25. 07. 2019).

[20] (2014). Siemens. industry online support forum. difference lad stl scl. © siemens ag industry, WWW: https://support.industry.siemens.com/tf/ww/en/posts/difference-lad-stl-scl/108671/?page=0\&pageSize=10\#top (cit. 14. 05. 2019).

[21] (2018). Manuál, simatic s7-1500 software controller, WWW: https://cache.industry.siemens.com/dl/files/725/109740725/att_916430/v1/s71500_cpu150xs_manual_en-US_en-US.pdf?download=true (cit. 14. 05. 2019).

[22] (2017). Manuál, simatic et 200sp open controller, WWW: https://support.industry.siemens.com/cs/attachments/109248384/cpu_1515sp_pc_manual_en-US_en-US.pdf?download=true (cit. 14. 05. 2019).

[23] (2015). Siemens. totally integrated automation portal. © siemens ag industry, WWW: http://www1.siemens.cz/ad/current/index.php?ctxnh=2416f2e791\&ctxp=home (cit. 14. 05. 2019).

[24] "Modulární elektrické terminály cpx", WWW: https://www.festo.com/cat/en-gb_gb/data/doc_CS/PDF/CZ/CPX_CZ.PDF (cit. 23. 07. 2019).

[25] *What is manufacturing execution system (MES)? - Definition from WhatIs.com*, en. WWW: https://searcherp.techtarget.com/definition/manufacturing-execution-system-MES (cit. 22. 07. 2019).

[26] *What is ERP (enterprise resource planning)? - Definition from WhatIs.com*, en. WWW: https://searcherp.techtarget.com/definition/ERP-enterprise-resource-planning (cit. 22. 07. 2019).

[27] J. Stark, *Product lifecycle managementn2,* Cham [u.a.: Springer, 2016, OCLC: 952107918, ISBN: 978-3-319-24434-1 978-3-319-24436-5.

[28] (2008). Unified architecture, OPC Foundation, WWW: https://opcfoundation.org/about/opc-technologies/opc-ua/ (cit. 23. 07. 2019).

[29] "Řízení a automatizace procesů pomocí distribuovaných sítí systémů firmy siemens", Diplomová práce, Univerzita Palackého v Olomouci, Univerzita Palackého v Olomouci, 2016.

[30] (2014). How do you program the communication blocks FB63 "tsend", FB64 "trcv", FB65 "tcon"... - ID: 29737950 - industry support siemens, WWW: `https://support.industry.siemens.com/cs/document/29737950/how-do-you-program-the-communication-blocks-fb63-tsend-fb64-trcv-fb65-tcon-and-fb66-tdiscon-in-order-to-use-the-tcp-protocol-for-data-exchange-by-means-of-the-integrated-profinet-interface-of-an-s7-300-s7-400-cpu-?dti=0&lc=en-WW` (cit. 06. 08. 2019).

[31] T. Macaulay a B. L. Singer, *Cybersecurity for Industrial Control Systems: SCADA, DCS, PLC, HMI, and SIS*, English. 2016, OCLC: 991528139, ISBN: 978-1-4398-0198-7.