VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY
DEPARTMENT OF CONTROL AND INSTRUMENTATION

SW MODEL KŘIŽOVATKY ŘÍZENÝ PROGRAMOVATELNÝM AUTOMATEM
SW MODEL OF PLC CONTROLLED CROSSING

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **Jakub Simon**
AUTHOR

VEDOUCÍ PRÁCE — **prof. Ing. František Zezulka, CSc.**
SUPERVISOR

BRNO 2016

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

bakalářský studijní obor **Automatizační a měřicí technika**
Ústav automatizace a měřicí techniky

Student: Jakub Simon ***ID:*** 164396
Ročník: 3 ***Akademický rok:*** 2015/16

NÁZEV TÉMATU:

SW model křižovatky řízený programovatelným automatem

POKYNY PRO VYPRACOVÁNÍ:

1. Zpracujte rešerši SW modelů křižovatek
2. Proveďte rozbor postupu pro vytvoření SW modelu jednoúrovňové křižovatky s přechody pro chodce na vyžádání.
3. Vytvořte koncepci modelu a řídicího systému, včetně vizualizace v systému InTouch dle pokynů vedoucího práce.
4. Vytvořte algoritmy pro simulační programy v systému InTouch pro různé modely křižovatek a režimy jejich řízení.
5. Vypracujte řídicí programy pro simulované křižovatky v různých programovacích jazycích standardu IEC 61131-3.

DOPORUČENÁ LITERATURA:

[1] Simon J.: Semestrální práce: SW model řízení křižovatky, UAMT, FEKT, Brno, 2015

Termín zadání: 8.2.2016 ***Termín odevzdání:*** 23.5.2016

Vedoucí práce: prof. Ing. František Zezulka, CSc.
Konzultant bakalářské práce:

doc. Ing. Václav Jirsík, CSc., *předseda oborové rady*

Abstrakt

Cílem této práce je vytvořit softwarový model pro různé typy dopravních křižovatek řízených pomocí programovatelného automatu. Řídicí systém je vytvářen na PLC SIMATIC S7-300 od firmy SIEMENS. Programy systému jsou psány v různých programovacích jazycích, jmenovitě GRAPH, STL, LAD. K řízení je také vytvořena vizualizace pomocí SCADA systému InTouch od společnosti Wonderware. Vizualizace komunikuje s programovatelným automatem pomocí DAServeru. Na vizualizaci jsou jak vstupní, tak i výstupní prvky. Neslouží tedy pouze ke sledování chodu řídicího systému, ale i k jednoduchému ovládání uživatelem, který na systém pomocí vizualizace dohlíží. Je možné ho spouštět a zastavovat, přepínat mezi režimy řízení křižovatky a také aktivovat tlačítko pro chodce čekajícího na zelenou.

Klíčová slova

PLC, řídicí systém, InTouch, vizualizace, bakalářská práce, proměnné

Abstract

The objective of this thesis is to create a software model to control various types of crossroads using programmable logic controller. The control system is created on Siemens PLC SIMATIC S7-300. The programmes of system are written in different programming languages such as GRAPH, STL, LAD. The visualisation is also created to control by using of SCADA system InTouch from the company Wonderware. The visualisation comunicates with the PLC using a DAServer. The visualisation has both, input and output elements, so it is not intended for supervising the run of the control system only, but also for simple user‘s control, using a graphical user interface. It is possible to control the start and stop events of the system, and to switch between different crossroad's control modes and also to trigger the pedestrian's button.

Keywords

PLC, control system, InTouch, visualisation, bachelor’s thesis, variable

Bibliografická citace

SIMON, J. *SW model křižovatky řízený programovatelným automatem.* Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2016. 41 s. Vedoucí bakalářské práce prof. Ing. František Zezulka, CSc.

Poděkování

Děkuji vedoucímu své bakalářské práce prof. Ing. Františkovi Zezulkovi, CSc. za čas, který mi věnoval, a za cenné rady a zkušenosti při vedení bakalářské práce, které mi pomohly k jejímu dokončení. Poděkování dále patří Ing. Janu Páskovi, CSc. a Mgr. Ivanu Picekovi ze společnosti Pantek s.r.o. za užitečné rady při vytváření vizualizace v programu InTouch.

Prohlášení

„Prohlašuji, že svou bakalářskou práci na téma SW model křižovatky řízený programovatelným automatem jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené bakalářské práce dále prohlašuji, že v souvislosti s vytvořením této bakalářské práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

V Brně dne: **20. května 2016** ……

podpis autora

Obsah

Seznam obrázků

1 ÚVOD

Řízení procesů je jednou z nejrozšířenějších oblastí dnešní automatizační techniky. Řízeno může být prakticky cokoliv. Vždy je však zapotřebí si vybrat správné řídicí systémy pro konkrétní aplikaci, která má být řešena. Neboť se svými parametry, případně jinými požadavky na řešení, jako jsou bezpečnostní opatření nebo třeba cena, se mohou lišit. Řídicích systémů je mnoho, avšak jeden z nejběžněji používaných je programovatelný logický automat, známý také jako PLC. PLC mohou být spojeny pomocí komunikační sběrnice do jednoho řídicího systému s počítačem. Ten pak reprezentuje hlavní řídicí prvek systému, zatímco jednotlivé moduly PLC reprezentují výkonovou složku systému. Ta s počítačem komunikuje pomocí komunikační sběrnice. Právě programovatelný logický automat je použit v našem případě, kde se budou řídit modely jednoúrovňových dopravních křižovatek. Pro zjištění správného fungování vytvořeného řídicího systému, je zapotřebí si navrhnout zobrazení stavů procesu, tzv. vizualizaci řídicího systému. Ta slouží jako uživatelské rozhraní obsluze, která křižovatku řídí. Tento problém je řešen SCADA systémy, které nám zajišťují propojení vizualizace v nich vytvořené, s řídicím systémem, uloženým v PLC.

1.1 Cíl práce

Cílem práce je vytvořit softwarový model pro řízení jednoúrovňové křižovatky pomocí programovatelného automatu. K tomuto softwarovému modelu navrhnout okno pro vizualizaci, aby bylo následně možné uživateli sledovat správný chod řízení křižovatky. Tato bakalářská práce má v budoucnu sloužit jako podklad pro laboratorní úlohu pro studenty školy FEKT VUT v Brně v magisterském předmětu Automatizace procesů (MAUP) příp. může být využita i v bakalářských předmětech Programovatelné automaty (BPGA) či Prostředky průmyslové automatizace (BPPA). Studenti si na této úloze vyzkouší tvorbu řídícího programu v různých programovacích jazycích, a dále také tvorbu vizualizace ve SCADA systému InTouch od společnosti Wonderware. Budou si muset správně nastavit komunikaci mezi řídícím programem v PLC a vizualizací v InTouch pomocí DAServeru, aby pak následně mohli jako uživatelé sledovat správnost jejich řízení křižovatky.

1.2 Řešení modelů křižovatek

V oblasti bakalářských, příp. diplomových prací, se různé typy modelů pro řízení dopravních křižovatek vytvářejí opravdu různými způsoby. Touto úlohou se zabývají nejrůznější vysoké školy a university. Od ČVUT v Praze přes technické university v Plzni, Jihlavě a Zlíně až po VUT v Brně. Někde se vytváří fyzický model křižovatky,

na kterém se pak testuje jeho funkčnost jednoduchou řídicí úlohou. U těchto prací je však hlavní náplní práce právě konstrukce fyzického modelu křižovatky, nikoliv vytvoření řídicího systému. Jinde se zase řeší spíše nejrůznější algoritmy pro průjezdy vozidel různými typy křižovatek, nebo jejich modelováním. V této práci bude vytvářen model softwarový ve SCADA systému InTouch, řízený program v připojeném PLC.

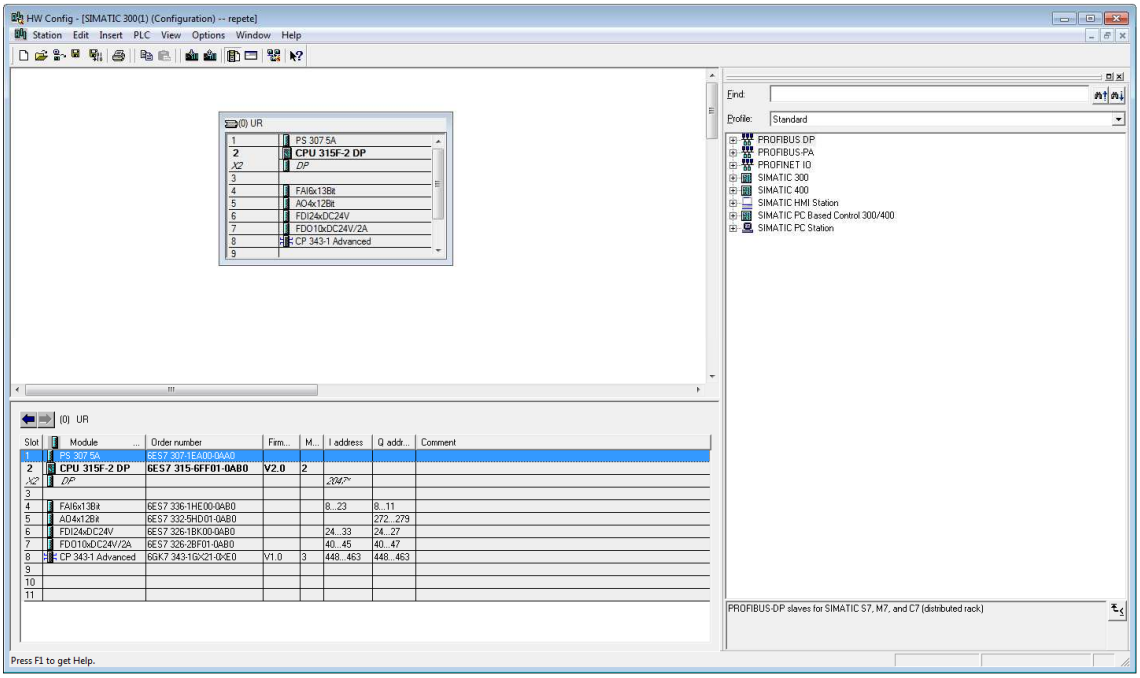
2 ŘÍDICÍ SYSTÉM

2.1 Hardwarová konfigurace PLC

Řídicí systém byl vytvořen na PLC typu SIMATIC S7-300 od společnosti SIMENS. Před samotným vytvářením řídicího systému bylo zapotřebí udělat hardwarovou konfigurace daného PLC, na kterém bude následně řídicí systém vytvořen. Systém je možné spustit i pomocí jiného PLC, nicméně při změně automatu je zapotřebí udělat vždy novou hardwarovou konfiguraci. Na ploše se spustí SIMATIC Manager, což je prostředí od společnosti SIEMENS, ve kterém bude řídicí systém vytvářen. Po spuštění prostředí SIMATIC Manager je nejprve zapotřebí založit nový projekt. Vytvoří se z nabídky *File -> New*. Zde se zadá jméno a vytvořený projekt se potvrdí. V levé části SIMATIC Manageru se projekt otevře i s již vloženou stanicí s názvem SIMATIC 300. Pokud tomu tak není, je zapotřebí si stanici do projektu přidat. Toho se docílí v nabídce *Insert -> Station*, výběrem stanice s názvem SIMATIC 300. Po rozkliknutí této části se zobrazí položka *Hardware.* Po dvojitém kliknutí na tuto položku se otevře hardwarová konfigurace PLC.

V hardwarové konfiguraci je vpravo hardwarový katalog, ze kterého se postupně vybírají jednotlivé moduly použitého PLC. Začíná se montážní lištou, která se nachází v nabídce *Rack -> Rail*. Tato položka se myší přetáhne do horní části okna hardwarové konfigurace, kde se otevře okno pro jednotlivé moduly stanice. Do tohoto okna se postupně vkládají jednotlivé moduly. Tyto moduly jsou vkládány zleva doprava tak, jak jsou skutečně umístěny v PLC. V dolním okně je zobrazena tabulka s počátečními adresami modulů. Při vkládání jednotlivých modulů do okna hardwarové konfigurace se musí dbát na to, aby se moduly vybrané v katalogu na pravé straně skutečně shodovaly s fyzickými moduly použitými v PLC. Při procházení katalogu se vždy po najetí kurzoru myši na příslušný modul dole pod katalogem zobrazí sériové číslo tohoto modulu. I toto číslo je potřeba zkontrolovat s příslušným fyzickým modulem. Po najití správného modulu se přetáhne do okna modulů v horní části. Při přetahování modulu myší se dovolený slot zbarví zeleně. Takto se do tohoto okna umístí všechny moduly. Po dokončení celé hardwarové konfigurace je potřeba tuto konfiguraci zkompilovat a následně uložit. To se provede v nabídce *PLC -> Save and compile*. Pokud proběhne kompilace bez hlášení chyb, je potřeba ještě hardwarovou konfiguraci nahrát do fyzického PLC. Opět v nabídce *PLC -> Download*. Po proběhnutí nahrávání

konfigurace do PLC vyskočí okno s cílovým procesorem. Zkontroluje se, zda se jedná o správný procesor a potvrdí se tlačítkem *OK*. Již je konfigurace skutečně dokončena a je možné ji zavřít.



Obr. 1 Hardwarová konfigurace PLC

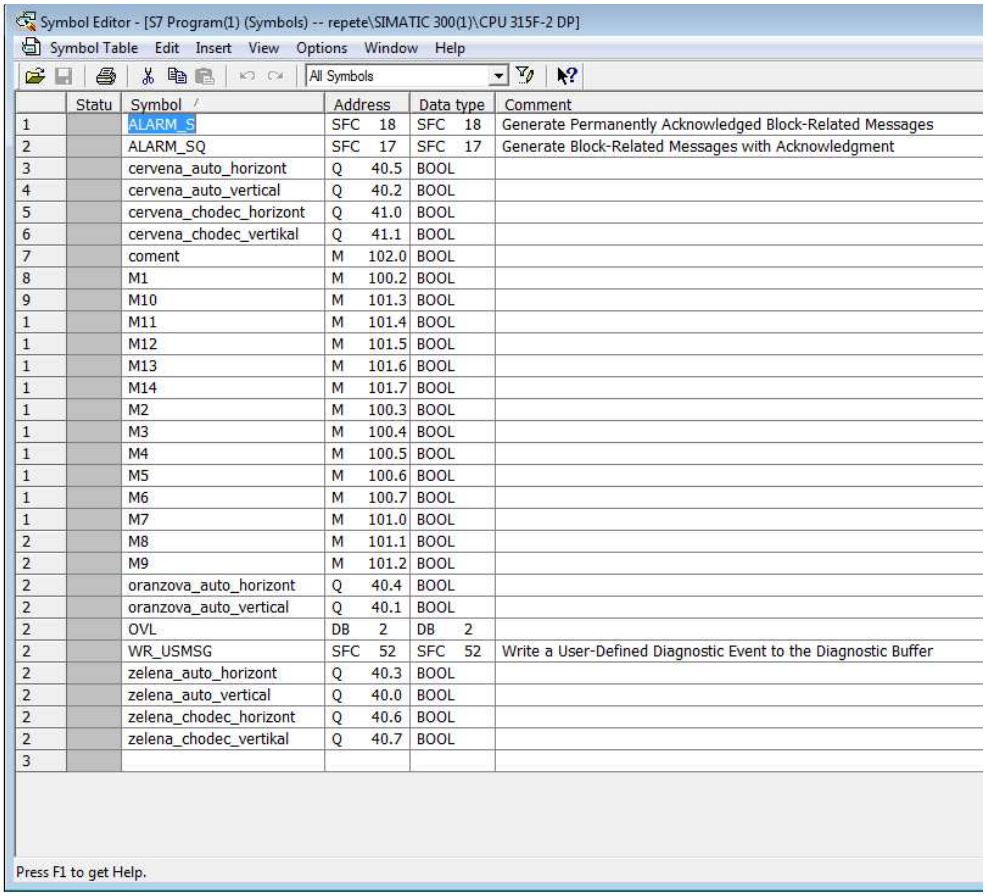
2.2 Vlastní řídicí systém

Po dokončení hardwarové konfigurace se v levé části okna klikne na položku *Blocks*. Po kliknutí na tuto položku se objeví ikonka *OB1*. Ta představuje hlavní organizační blok. V tomto bloku musí být uveden program, bude zde však pouze volán. V této práci je v organizačním bloku pokaždé zvolen jako programovací jazyk Ladder Diagram (dále Ladder). Samotným program bude vytvářen v bloku *FB1*, což je funkční blok. Tento blok se do programu vloží přes nabídku *Insert -> Blocks -> Function Block*. Zde se zvolí, který programovací jazyk bude využit. Pro názornost je v každém z typů křižovatky použit jiný programovací jazyk. Nicméně záleží na programátorovi, který programovací jazyk si zvolí.

2.2.1 Křižovatka typu „Kříž“

Pro vytváření řídicího systému pro tento typ křižovatky byl zvolen grafický programovací jazyk SIMATIC S7-GRAPH zvaný též i jako Grafcet. Tento sekvenční programovací jazyk funguje na principu stavového automatu, resp. představuje stavový diagram. Skládá se ze dvou základních prvků. První jsou kroky, v programu zvané jako *Step* a druhými jsou přechody zvané jako *Trans*. Jazyk funguje tak, že v každém kroku se provádí nějaká funkce příp. činnosti. V přechodech jsou uvedeny podmínky, při jejichž splnění dochází k přechodu z aktuálního kroku do kroku dalšího, kde se provádějí činnosti další. Následně při splnění podmínek dalšího přechodu program opět přejde do kroku následujícího. K hlavnímu programu je možné přidat alternativní cesty. Tyto alternativní cesty je možné využívat pro rozvětvení programu. Jsou ukončeny skokem *(Jump)*, který skočí na další aktuální krok. Je možné přeskakovat libovolně, tzn. že, není podmínkou skočit na další krok, ale na jakýkoliv zvolený krok. Těmito kroky je možné zajistit i cyklické opakování programu, kdy se z posledního kroku skočí zpět na krok první.

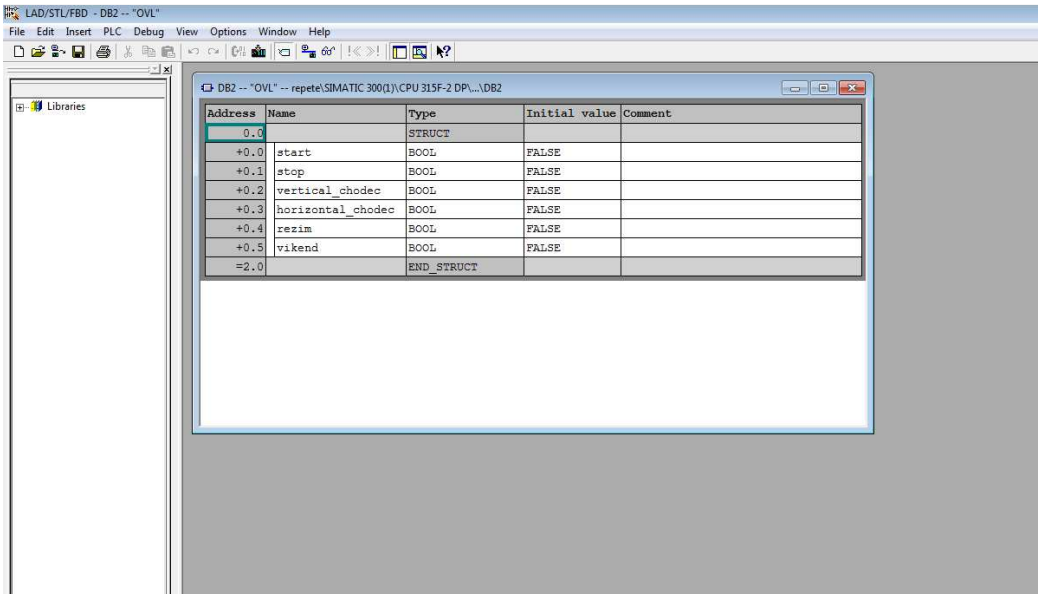
Nyní již ke konkrétnímu řešení vytváření programu. Nejprve se vytvoří tabulka proměnných (*Tag Table*), které se budou v programu používat. Tabulku se vytvoří z otevřeného bloku v nabídce *Options -> Symbol Table*. Do této tabulky se vkládají všechny proměnné, které budeme v programu používat i s jejich adresami, tzv. aliasy a také je zapotřebí uvést jejich datový typ. V našem případě používáme všechny proměnné jako logické, takže máme všude zvolen datový typ *BOOL*. Nicméně do tabulky je možné kdykoliv v průběhu vytváření programu přidávat další potřebné proměnné, případně ty nepotřebné mazat.



	Statu	Symbol	Address	Data type	Comment
1		ALARM_S	SFC 18	SFC 18	Generate Permanently Acknowledged Block-Related Messages
2		ALARM_SQ	SFC 17	SFC 17	Generate Block-Related Messages with Acknowledgment
3		cervena_auto_horizont	Q 40.5	BOOL	
4		cervena_auto_vertical	Q 40.2	BOOL	
5		cervena_chodec_horizont	Q 41.0	BOOL	
6		cervena_chodec_vertikal	Q 41.1	BOOL	
7		coment	M 102.0	BOOL	
8		M1	M 100.2	BOOL	
9		M10	M 101.3	BOOL	
1		M11	M 101.4	BOOL	
1		M12	M 101.5	BOOL	
1		M13	M 101.6	BOOL	
1		M14	M 101.7	BOOL	
1		M2	M 100.3	BOOL	
1		M3	M 100.4	BOOL	
1		M4	M 100.5	BOOL	
1		M5	M 100.6	BOOL	
1		M6	M 100.7	BOOL	
1		M7	M 101.0	BOOL	
2		M8	M 101.1	BOOL	
2		M9	M 101.2	BOOL	
2		oranzova_auto_horizont	Q 40.4	BOOL	
2		oranzova_auto_vertical	Q 40.1	BOOL	
2		OVL	DB 2	DB 2	
2		WR_USMSG	SFC 52	SFC 52	Write a User-Defined Diagnostic Event to the Diagnostic Buffer
2		zelena_auto_horizont	Q 40.3	BOOL	
2		zelena_auto_vertical	Q 40.0	BOOL	
2		zelena_chodec_horizont	Q 40.6	BOOL	
2		zelena_chodec_vertikal	Q 40.7	BOOL	
3					

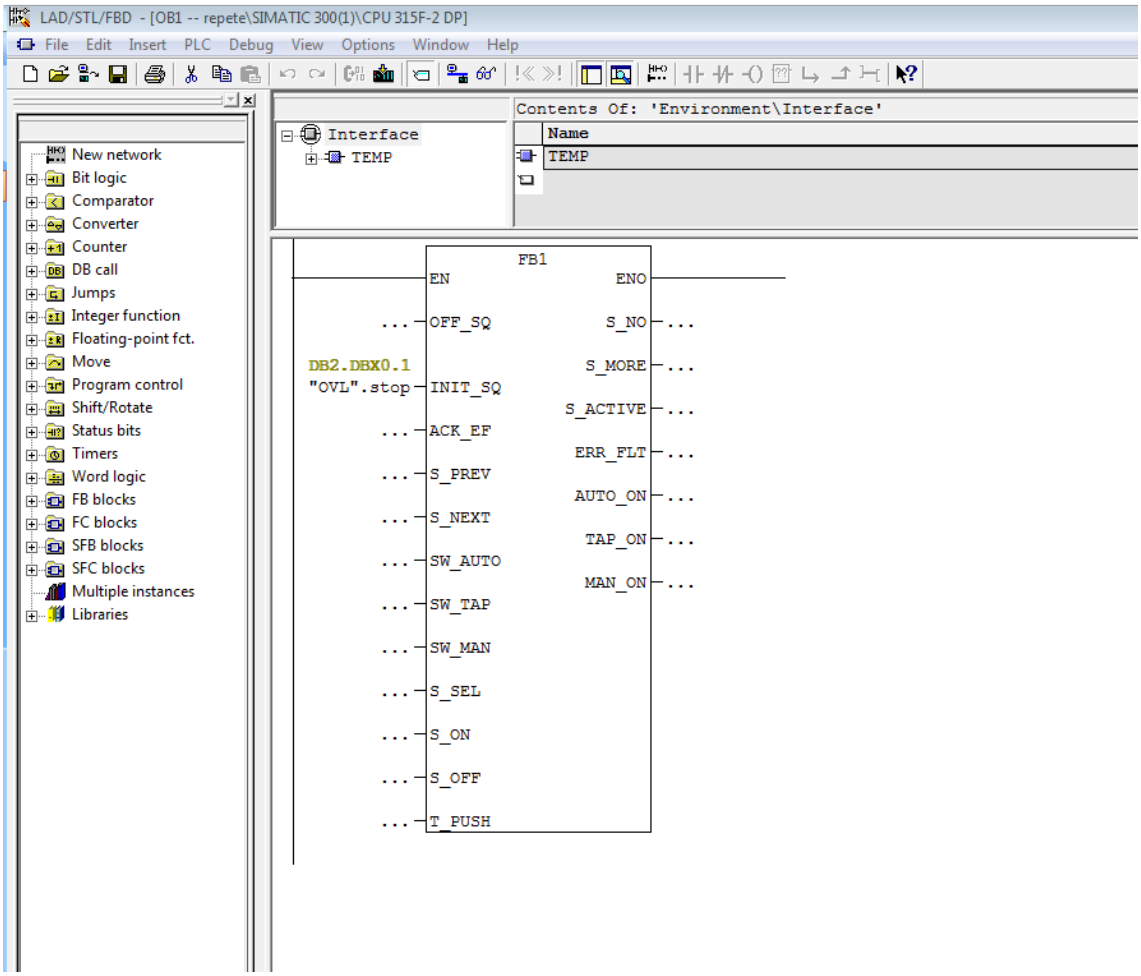
Obr. 2 Tabulka proměnných u křižovatky typu „Kříž“

Po dokončení tabulky proměnných se tabulka uloží a je možné ji zavřít. Jak je vidět z Obr. 2, v tabulce proměnných se vyskytuje proměnná *OVL* s aliasem a datovým typem *DB 2*. Jedná se o datový blok, ve kterém jsou zahrnuty všechny proměnné, které jsou použity na ovládání řídicího systému. Ovládací prvky jsou v novém datovém bloku kvůli pozdější adresaci ve SCADA systému InTouch, ve kterém je vytvořena vizualizace. Datový blok DB2 s jednotlivými ovládacími proměnnými je na Obr. 3.



Obr. 3 DB2 s adresací ovládacích proměnných u křižovatky typu „Kříž“

Dále se v hlavním organizačním bloku OB1 vytvoří volání funkčního bloku FB1, ve kterém bude později samotný program. Po otevření OB1 je po levé straně vidět lišta jednotlivých prvků, které lze do programu vložit. Tyto prvky se vkládají na jednotlivé řádky tzv. networky. Tyto řádky se v programu vykonávají současně, nikoliv postupně jeden po druhém. Nám však postačí pouze jeden. První řádek je v organizačním bloku vložen automaticky. Ze stromu vlevo se z nabídky *FB blocks* vybere blok *FB1* a přetáhne se na network. K jednotlivým pinům tohoto bloku se dále mohou přiřazovat proměnné. Zde však stačí pouze jedna. K pinu s názvem *INIT_SQ* se připojí proměnná *STOP*. Tato proměnná bude fungovat jako absolutní zastavení běžícího programu, který je následně volán z funkčního bloku FB1.



Obr. 4 Organizační blok OB1 u křižovatky typu „Kříž“

Nyní se může přejít k samotné tvorbě programu řídicího systému pro řízení křižovatky typu „Kříž“. Jak už bylo výše zmíněno, program se vytvářel ve funkčním bloku FB1 a je psán grafickým programovacím jazykem S7-GRAPH. Otevře se tedy funkční blok FB1. Po otevření je vidět již zobrazený první krok (*Step*) s přechodem (*Trans*). Do kroku jde normálně kliknout myši a poté si do něj napsat činnosti, které se mají vykonat. V případě přechodu se musí vybrat z levé boční lišty, která podmínka má být vložena (např. spínací kontakt, rozpínací kontakt). Následně se jen přidávají další kroky a přechody.

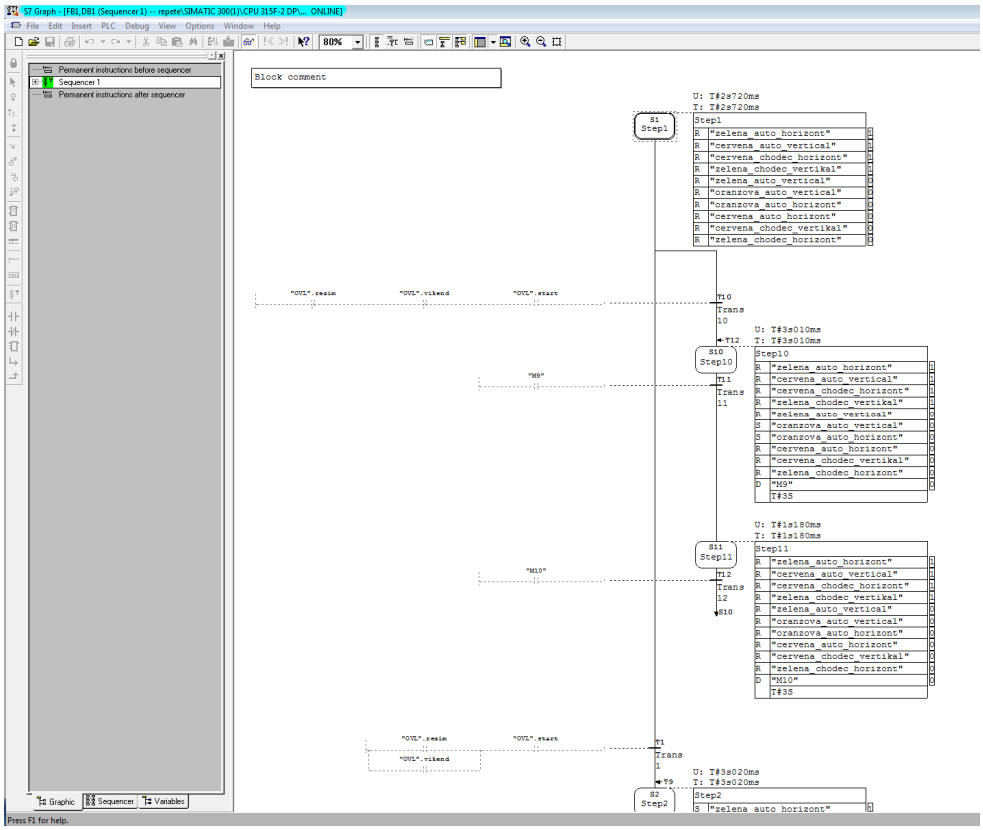
Pro řídicí systém na provoz křižovatky každý krok reprezentuje buď rozsvícená či zhasnutá světla semaforu, jak pro chodce, tak pro auta. Po uplynutí určité doby dojde k přechodu do kroku dalšího, kde jsou již rozsvícena světla jiná. Zjednodušeně si to lze představit tak, že v jednom kroku svítí pro auta červená, po uplynutí nějakého času pak oranžová, následně zelená a pak opět oranžová a zase červená, která již svítila v kroku

prvním. Došlo tedy ke skoku z posledního kroku, kdy svítila pro auta oranžová, na skok první, kde již opět svítí červená a celý cyklus se opět opakuje.

Přechody pro chodce jsou řešeny tak, že pokud čekající chodec na přechodu pro chodce nezmáčkne tlačítko určené jako požadavek pro chodce, tak i přes to, že v jeho směru mají auta červenou, tak chodci zelená na semaforu nepadne. Aby chodci padla zelená, musí zmáčknout tlačítko jako požadavek, a v následném cyklu chodu semaforů se mu rozsvítí zelená a chodec může přejít. Dále jsou v programu na řízení křižovatky implementovány jednotlivé režimy. Pokud není zvolen žádný režim, semafory blikají oranžově. Toto reprezentuje jakýsi pohotovostní režim. Takto semafory fungují např. v noci. Dále se pak rozlišuje režim víkendový a pracovní. Pracovní režim je navrhnut tak, že na křižovatce trvá zelená, příp. červená stejně dlouhou dobu. Je tak učiněno z důvodu velikého provozu ve všední dny. Tento pracovní režim reprezentuje proměnná *REZIM*. Zatímco u víkendového režimu trvá zelená pro auta na vodorovné silnici, která je brána jako hlavní, déle než zelená na silnici svislé, brané jako vedlejší. O víkendu je provoz menší, a proto, aby nedocházelo ke zbytečnému omezování aut na hlavní silnici i přesto, že po vedlejší žádná auta skoro nejezdí, tudíž nepotřebují mít zelenou tak dlouho. Celý řídicí systém křižovatky se spouští tlačítkem *START*, nicméně ještě před jeho spuštěním je zapotřebí zvolit si režim, ve kterém se vyžaduje, aby program fungoval. Přepínání za chodu semaforu není možné. Pro změnu režimu je tedy potřeba program na okamžik vypnout tlačítkem *STOP*, zvolit si jiný režim a následně program opět spustit tlačítkem *START*. Ukázku části řídicího systému psaného v programovacím jazyce SIMATIC S7-GRAPH v bloku FB1 můžeme vidět na Obr. 5.

Po každé změně programu je potřeba program opět uložit a nahrát do PLC (*download*), aby se změny projevily. Vývojový diagram celého řídicího systému křižovatky typu „Kříž“ je znázorněn na Obr. 7.

Pro zkoušení a odlaďování napsaného programu se využívá tzv. VAT tabulka. Vytvoříme si ji stejně jako funkční blok FB1 přes nabídku na horní liště *Insert -> Blocks -> Variable Table*. Do této tabulky si zapíšeme proměnné podobně jako do tabulky proměnných. Zde stačí však pouze ty, které chceme sledovat, případně s nimi manipulovat tzn. měnit jejich hodnoty. Při chodu programu tak můžeme sledovat, které proměnné jsou aktuálně aktivní a které ne. Nebo přepisovat hodnoty režimů z *true (log. 1)* na *false (log. 0)*, a tím měnit režim chodu semaforů. VAT tabulka je znázorněna na Obr. 6.



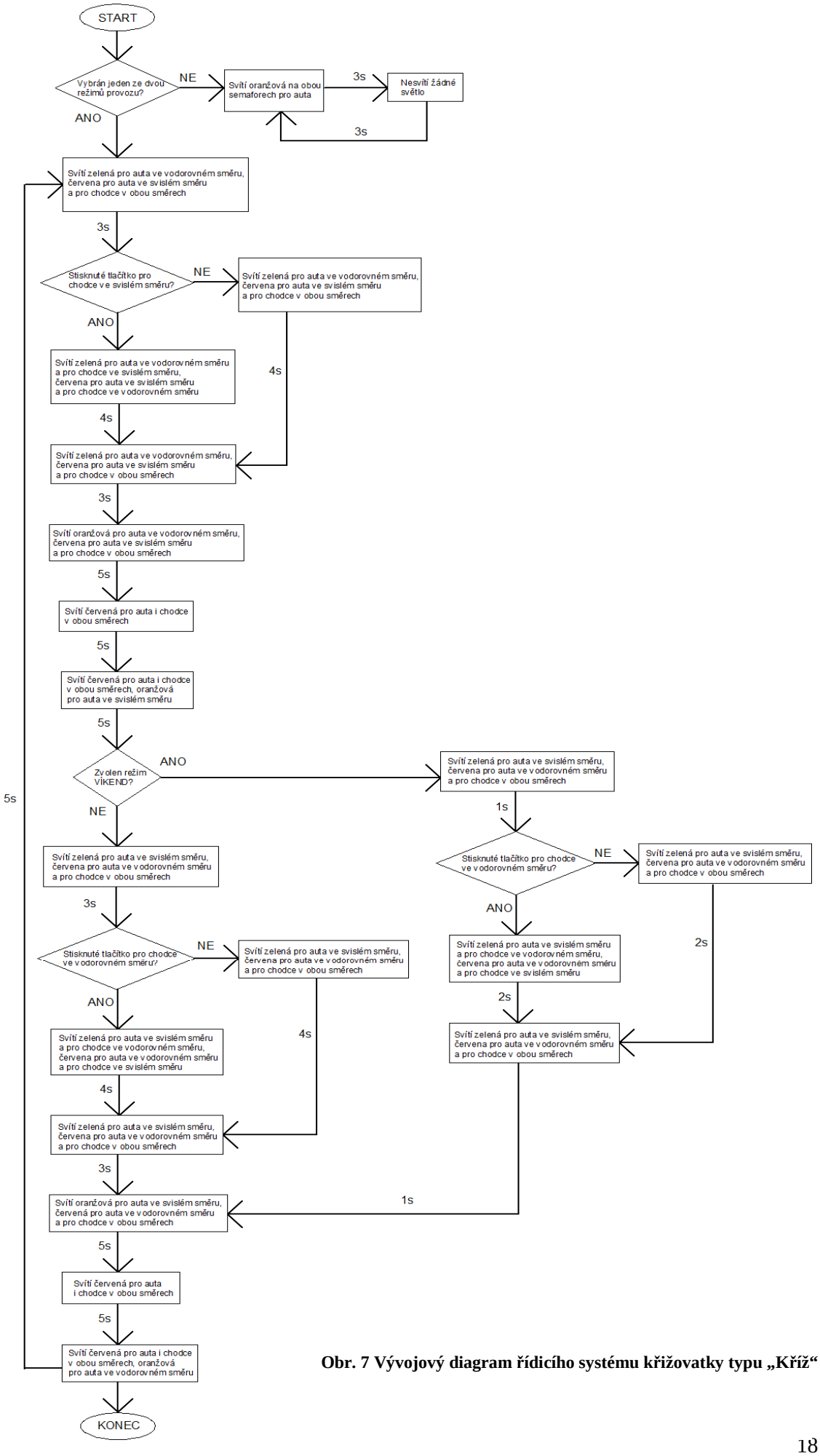
Obr. 5 Ukázka části programu v program. jazyce S7-GRAPH

Var - [VAT2 -- @repete\SIMATIC 300(1)\CPU 315F-2 DP\S7 Program(1) ONLINE]

Table Edit Insert PLC Variable View Options Window Help

	Address	Symbol	Display format	Status value	Modify value
1	Q 40.0	"zelena_auto_vertical"	BOOL	false	
2	Q 40.1	"oranzova_auto_vertical"	BOOL	false	
3	Q 40.2	"cervena_auto_vertical"	BOOL	true	
4	Q 40.3	"zelena_auto_horizont"	BOOL	false	
5	Q 40.4	"oranzova_auto_horizont"	BOOL	true	
6	Q 40.5	"cervena_auto_horizont"	BOOL	true	
7	Q 40.6	"zelena_chodec_horizont"	BOOL	false	
8	Q 40.7	"zelena_chodec_vertikal"	BOOL	false	
9	Q 41.0	"cervena_chodec_horizont"	BOOL	true	
10	Q 41.1	"cervena_chodec_vertikal"	BOOL	true	
11	M 100.2	"M1"	BOOL	false	
12	M 100.3	"M2"	BOOL	false	
13	M 100.4	"M3"	BOOL	false	
14	M 100.5	"M4"	BOOL	false	
15	M 100.6	"M5"	BOOL	false	
16	M 100.7	"M6"	BOOL	false	
17	M 101.0	"M7"	BOOL	false	
18	M 101.1	"M8"	BOOL	false	
19	DB2.DBX 0.0	"OVL".start	BOOL	false	
20	DB2.DBX 0.1	"OVL".stop	BOOL	false	
21	DB2.DBX 0.2	"OVL".vertical_chodec	BOOL	true	
22	DB2.DBX 0.3	"OVL".horizontal_chodec	BOOL	true	
23	DB2.DBX 0.4	"OVL".rezim	BOOL	false	
24	DB2.DBX 0.5	"OVL".vikend	BOOL	true	
25					

Obr. 6 VAT tabulka



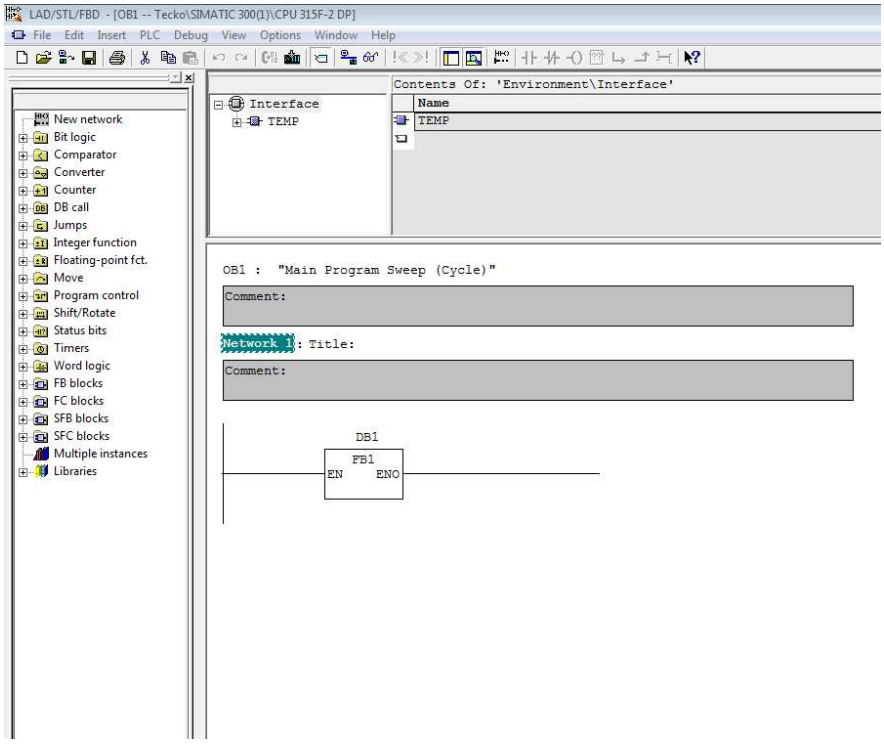
Obr. 7 Vývojový diagram řídicího systému křižovatky typu „Kříž“

2.2.2 Křižovatka ve tvaru písmene „T“ s odbočovacími pruhy

Nejprve je opět zapotřebí založit si nový projekt v prostředí SIMATIC Manageru a vytvořit si hardwarovou konfiguraci používaného PLC. Jelikož je použit stále stejný programovací logický automat, tak i hardwarová konfigurace bude úplně stejná, jako u předchozího typu křižovatky. Její ukázka byla znázorněna na Obr. 1. Po jejím dokončení je zapotřebí konfiguraci opět uložit a nahrát do PLC pomocí tlačítka *download*. Následně se již může přejít k vytváření programu.

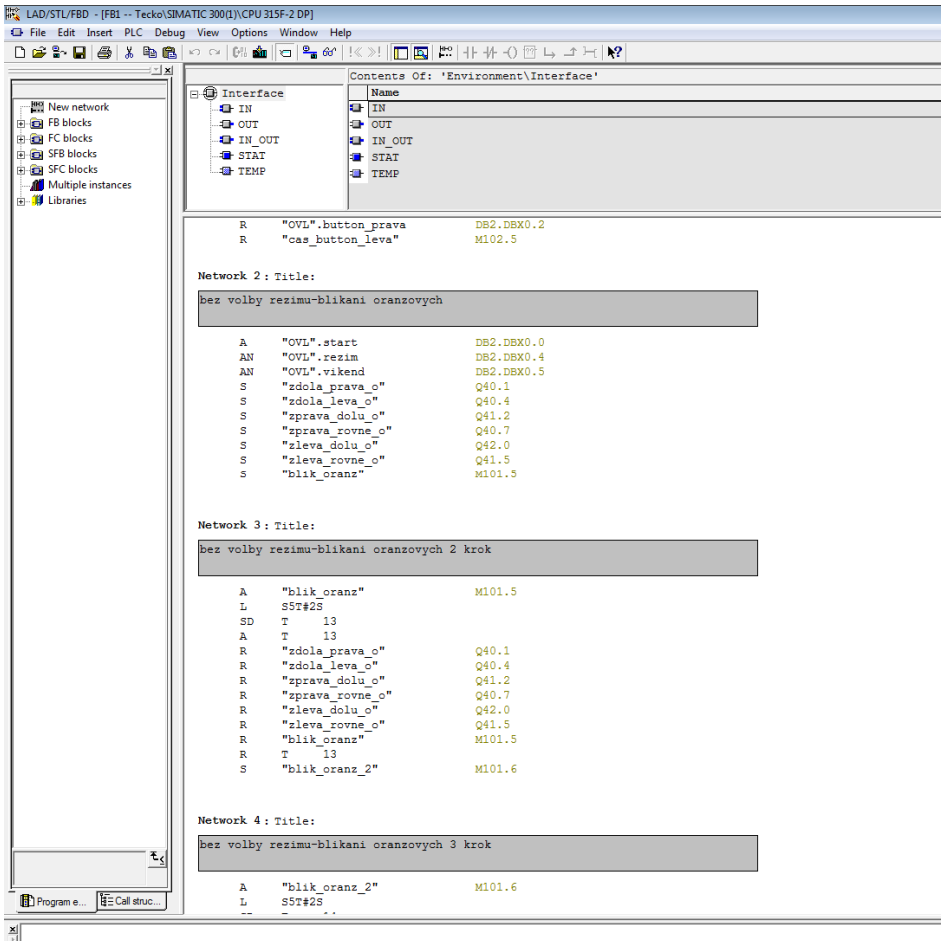
Zde je stejně jako v předchozím případě zapotřebí nejprve v organizačním bloku OB1 volat funkční blok FB1, ve kterém bude psán vlastní program řídicího systému. Volání funkčního bloku FB1 z hlavního organizačního bloku je ukázáno na Obr. 8.

Po založení bloku FB1 je nutno si opět zvolit programovací jazyk, ve kterém bude řídicí systém vytvářen. U tohoto typu křižovatky je zvolen programovací jazyk STL (Statement list). Tento programovací jazyk je vlastně seznam příkazů psaných pod sebou v jednotlivých částech programu (networks). Tyto části však nejsou sekvenční, jak tomu bylo u předchozího typu křižovatky, kde se mezi jednotlivými kroky přecházelo po splnění podmínky v následujícím přechodu.



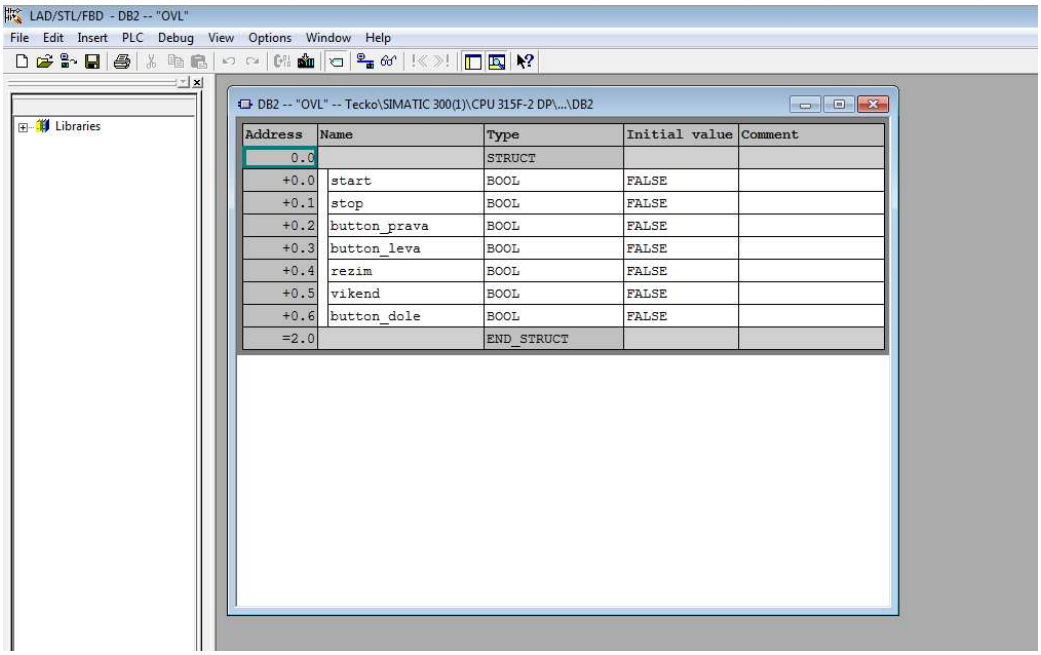
Obr. 8 Organizační blok OB1 u křižovatky ve tvaru „T“

Tyto tzv. networky se tedy provádí nezávisle na pořadí. Závisí pouze na splnění podmínek běžícího programu. Jako příklad poslouží třeba přecházení chodce přes přechod. Musí být splněna podmínka, že na přechodu pro chodce si daný chodec stiskl tlačítko jako požadavek pro přejití přes silnici, nicméně se ještě musí čekat na splnění další podmínky a to, že auta v tomto směru mají červenou. Stejně jako u programovacího jazyka Grafcet reprezentovaly podmínky v přechodu např. spínací a rozpínací kontakty, stejně tak je tomu i v jazyce STL. S tím rozdílem, že tyto kontakty nyní píšeme jako daný příkaz. Ukázku řídicího systému pro křižovatku ve tvaru písmene „T“ je znázorněna na Obr. 9. Jak je na této ukázce vidět, program je rozdělen do tří sloupců. V prvním sloupci je uveden příkaz (A, AN, O, ON,…), ve druhém název proměnné a ve třetím její alias. Písmeno *A* je příkaz pro přímou podmínku v sérii, *AN* pro podmínku v sérii negovanou. Písmeno *O* zase příkaz pro přímou podmínku paralelní a *ON* pro negovanou paralelní podmínku.



Obr. 9 Ukázka řídicího systému v jazyce STL

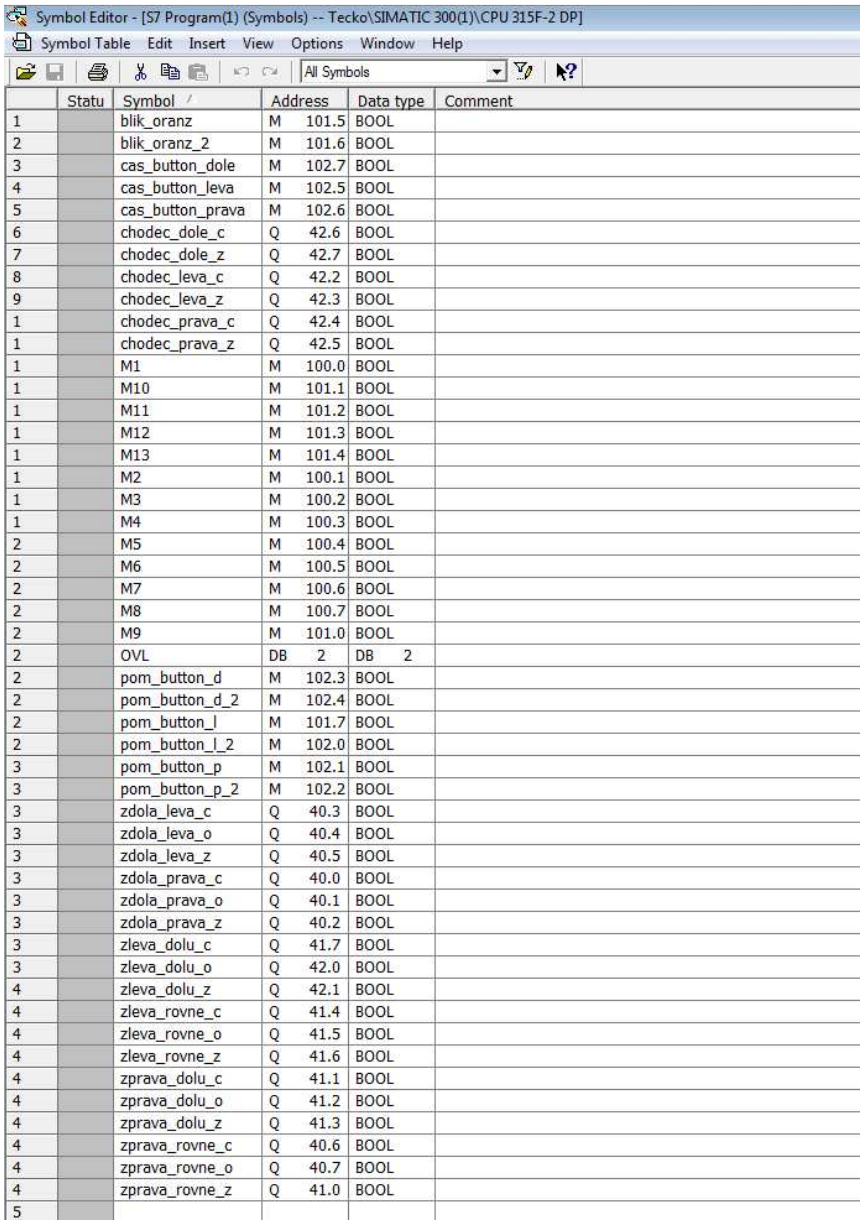
Stejně jako u předchozího typu křižovatky, jsou ovládací proměnné uvedeny v samostatném datovém bloku DB2 (viz Obr. 10). Vesměs jsou tyto ovládací proměnné stejné jako předtím, pouze je zde o jedno tlačítko pro chodce více.



Address	Name	Type	Initial value	Comment
0.0		STRUCT		
+0.0	start	BOOL	FALSE	
+0.1	stop	BOOL	FALSE	
+0.2	button_prava	BOOL	FALSE	
+0.3	button_leva	BOOL	FALSE	
+0.4	rezim	BOOL	FALSE	
+0.5	vikend	BOOL	FALSE	
+0.6	button_dole	BOOL	FALSE	
=2.0		END_STRUCT		

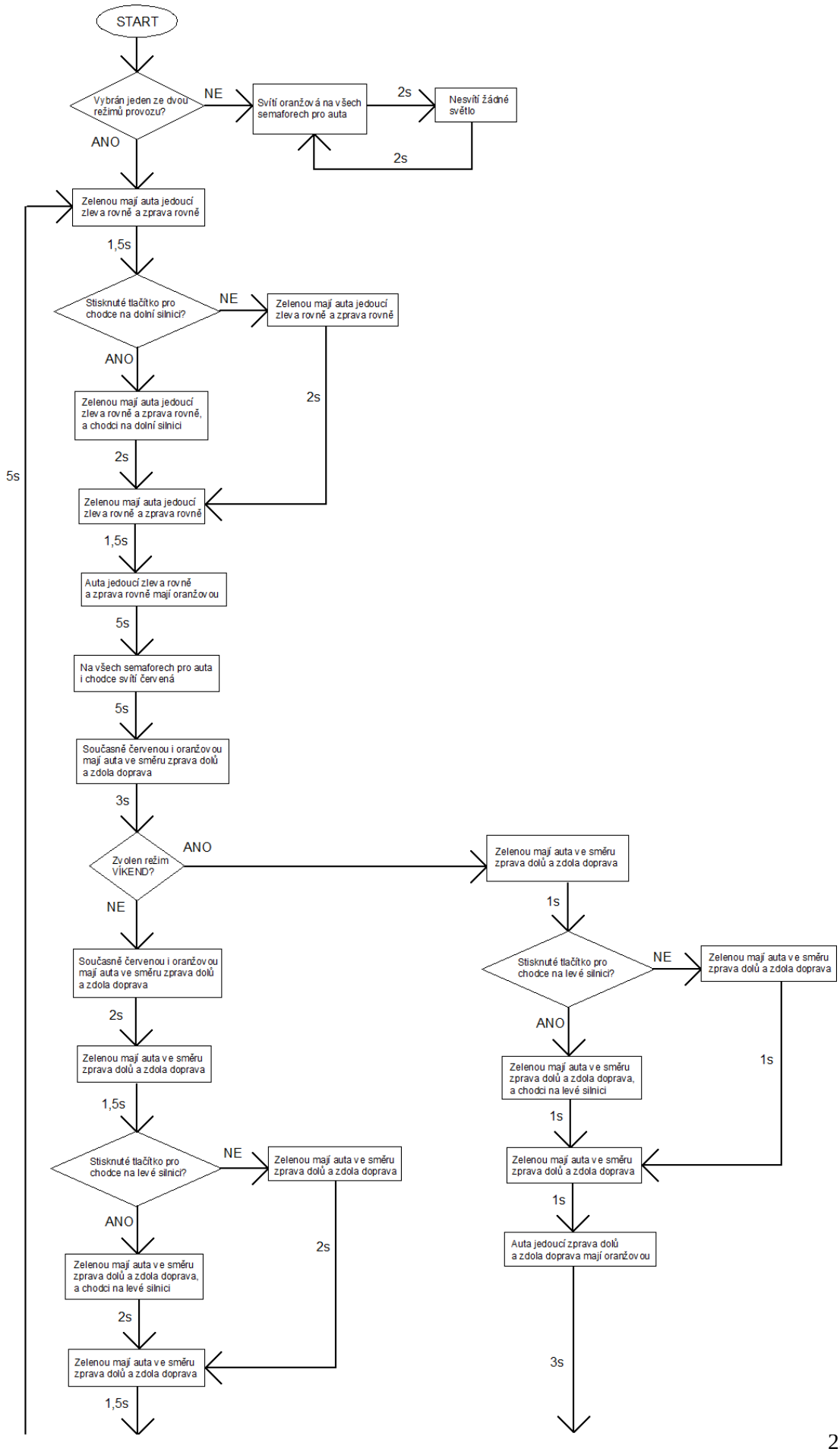
Obr. 10 DB2 s adresací ovládacích proměnných u křižovatky ve tvaru „T“

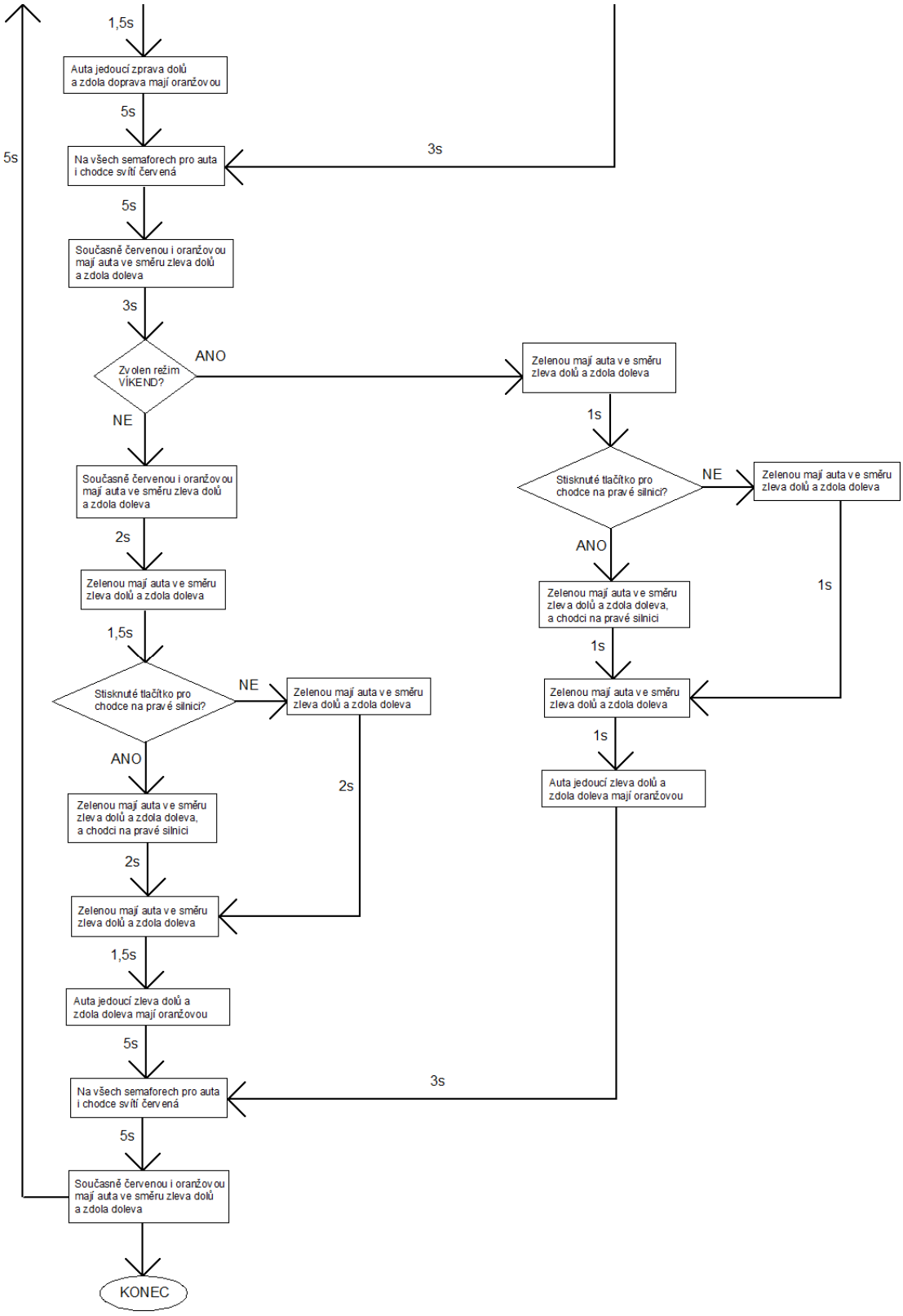
Všechny používané proměnné jsou uvedeny na Obr. 11 v tabulce proměnných. Jak je vidět, je zde uvedena spousta proměnných, jejichž adresa začíná písmenem *M*, tzv. merkery. Tyto merkery jsou vnitřní proměnné, které využívá sám programátor ke správné realizaci programu. Pomocné proměnné, jak se i jinak nazývají, se využívají převážně k tomu, aby nedocházelo ke splnění dvou různých podmínek v jiných částech programu. Ten by pak fungoval nesprávně. Na Obr. 12 je znázorněn vývojový diagram pro tento typ křižovatky.



	Statu	Symbol	Address	Data type	Comment
1		blik_oranz	M 101.5	BOOL	
2		blik_oranz_2	M 101.6	BOOL	
3		cas_button_dole	M 102.7	BOOL	
4		cas_button_leva	M 102.5	BOOL	
5		cas_button_prava	M 102.6	BOOL	
6		chodec_dole_c	Q 42.6	BOOL	
7		chodec_dole_z	Q 42.7	BOOL	
8		chodec_leva_c	Q 42.2	BOOL	
9		chodec_leva_z	Q 42.3	BOOL	
1		chodec_prava_c	Q 42.4	BOOL	
1		chodec_prava_z	Q 42.5	BOOL	
1		M1	M 100.0	BOOL	
1		M10	M 101.1	BOOL	
1		M11	M 101.2	BOOL	
1		M12	M 101.3	BOOL	
1		M13	M 101.4	BOOL	
1		M2	M 100.1	BOOL	
1		M3	M 100.2	BOOL	
1		M4	M 100.3	BOOL	
2		M5	M 100.4	BOOL	
2		M6	M 100.5	BOOL	
2		M7	M 100.6	BOOL	
2		M8	M 100.7	BOOL	
2		M9	M 101.0	BOOL	
2		OVL	DB 2	DB 2	
2		pom_button_d	M 102.3	BOOL	
2		pom_button_d_2	M 102.4	BOOL	
2		pom_button_l	M 101.7	BOOL	
2		pom_button_l_2	M 102.0	BOOL	
3		pom_button_p	M 102.1	BOOL	
3		pom_button_p_2	M 102.2	BOOL	
3		zdola_leva_c	Q 40.3	BOOL	
3		zdola_leva_o	Q 40.4	BOOL	
3		zdola_leva_z	Q 40.5	BOOL	
3		zdola_prava_c	Q 40.0	BOOL	
3		zdola_prava_o	Q 40.1	BOOL	
3		zdola_prava_z	Q 40.2	BOOL	
3		zleva_dolu_c	Q 41.7	BOOL	
3		zleva_dolu_o	Q 42.0	BOOL	
4		zleva_dolu_z	Q 42.1	BOOL	
4		zleva_rovne_c	Q 41.4	BOOL	
4		zleva_rovne_o	Q 41.5	BOOL	
4		zleva_rovne_z	Q 41.6	BOOL	
4		zprava_dolu_c	Q 41.1	BOOL	
4		zprava_dolu_o	Q 41.2	BOOL	
4		zprava_dolu_z	Q 41.3	BOOL	
4		zprava_rovne_c	Q 40.6	BOOL	
4		zprava_rovne_o	Q 40.7	BOOL	
4		zprava_rovne_z	Q 41.0	BOOL	
5					

Obr. 11 Tabulka všech proměnných v řídicím systému křižovatky ve tvaru „T“


START
Vybrán jeden ze dvou režimů provozu?
NE
Svítí oranžová na všech semaforech pro auta
2s
Nesvítí žádné světlo
2s
ANO
Zelenou mají auta jedoucí zleva rovně a zprava rovně
1,5s
Stisknuté tlačítko pro chodce na dolní silnici?
NE
Zelenou mají auta jedoucí zleva rovně a zprava rovně
ANO
Zelenou mají auta jedoucí zleva rovně a zprava rovně, a chodci na dolní silnici
2s
2s
5s
Zelenou mají auta jedoucí zleva rovně a zprava rovně
1,5s
Auta jedoucí zleva rovně a zprava rovně mají oranžovou
5s
Na všech semaforech pro auta i chodce svítí červená
5s
Současně červenou i oranžovou mají auta ve směru zprava dolů a zdola doprava
3s
Zvolen režim VÍKEND?
ANO
Zelenou mají auta ve směru zprava dolů a zdola doprava
1s
NE
Současně červenou i oranžovou mají auta ve směru zprava dolů a zdola doprava
2s
Zelenou mají auta ve směru zprava dolů a zdola doprava
1,5s
Stisknuté tlačítko pro chodce na levé silnici?
NE
Zelenou mají auta ve směru zprava dolů a zdola doprava
ANO
Zelenou mají auta ve směru zprava dolů a zdola doprava, a chodci na levé silnici
2s
2s
Zelenou mají auta ve směru zprava dolů a zdola doprava
1,5s
Stisknuté tlačítko pro chodce na levé silnici?
NE
Zelenou mají auta ve směru zprava dolů a zdola doprava
ANO
1s
Zelenou mají auta ve směru zprava dolů a zdola doprava, a chodci na levé silnici
1s
1s
Zelenou mají auta ve směru zprava dolů a zdola doprava
1s
Auta jedoucí zprava dolů a zdola doprava mají oranžovou
3s

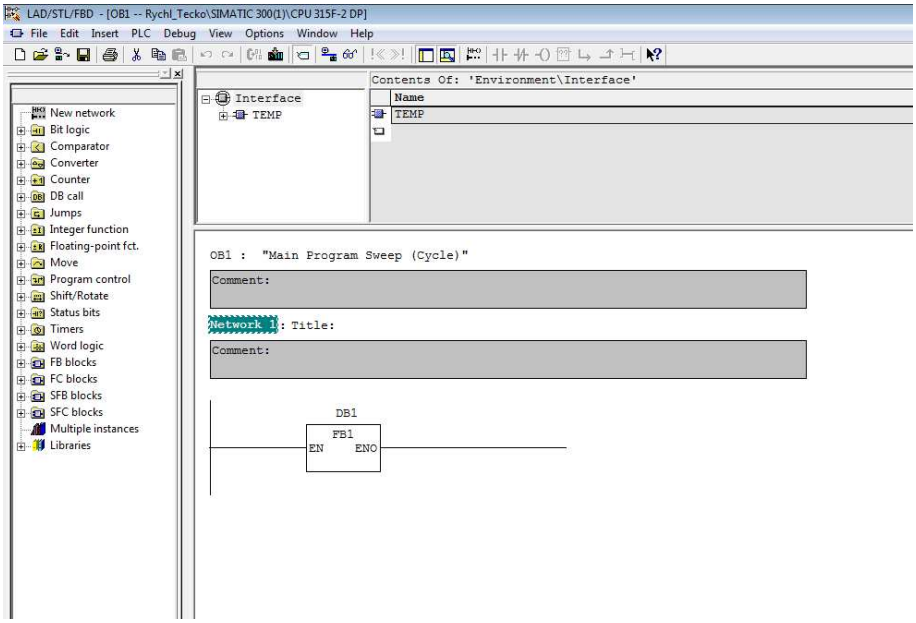


Obr. 12 Vývojový diagram řídicího systému křižovatky ve tvaru „T“

2.2.3 Rychlostní křižovatka ve tvaru písmene „T“ bez chodců

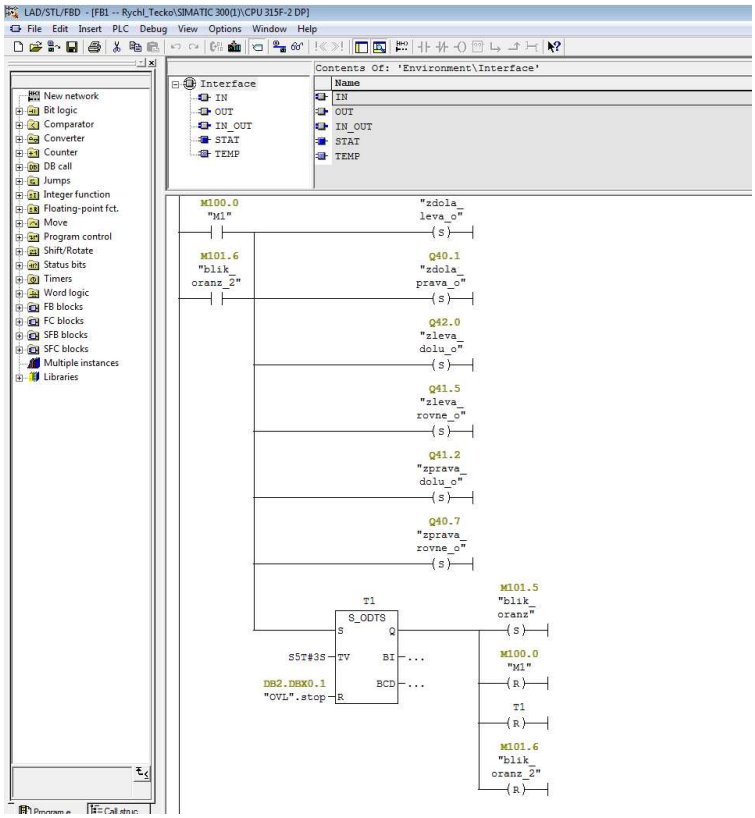
Pod pojmem „rychlostní křižovatka ve tvaru písmene T“ je myšlena křižovatka, která je obdobná té předchozí. Nicméně se odlišuje tím, že zde nejsou jenom odbočovací pruhy, ale zároveň i průběžné pruhy křižovatkou pro organizované odjíždění vozidel z křižovatky. Také zde nejsou implementovány žádné přechody pro chodce.

Opět jako u předchozích dvou typů křižovatek je zapotřebí založit si nový projekt a vytvořit si hardwarovou konfiguraci programovatelného automatu. I zde bude tato konfigurace stejná jako již dříve. Do organizačního bloku OB1 vložíme volání bloku funkčního, ve kterém bude vytvářen program řízení této křižovatky (viz Obr. 13).

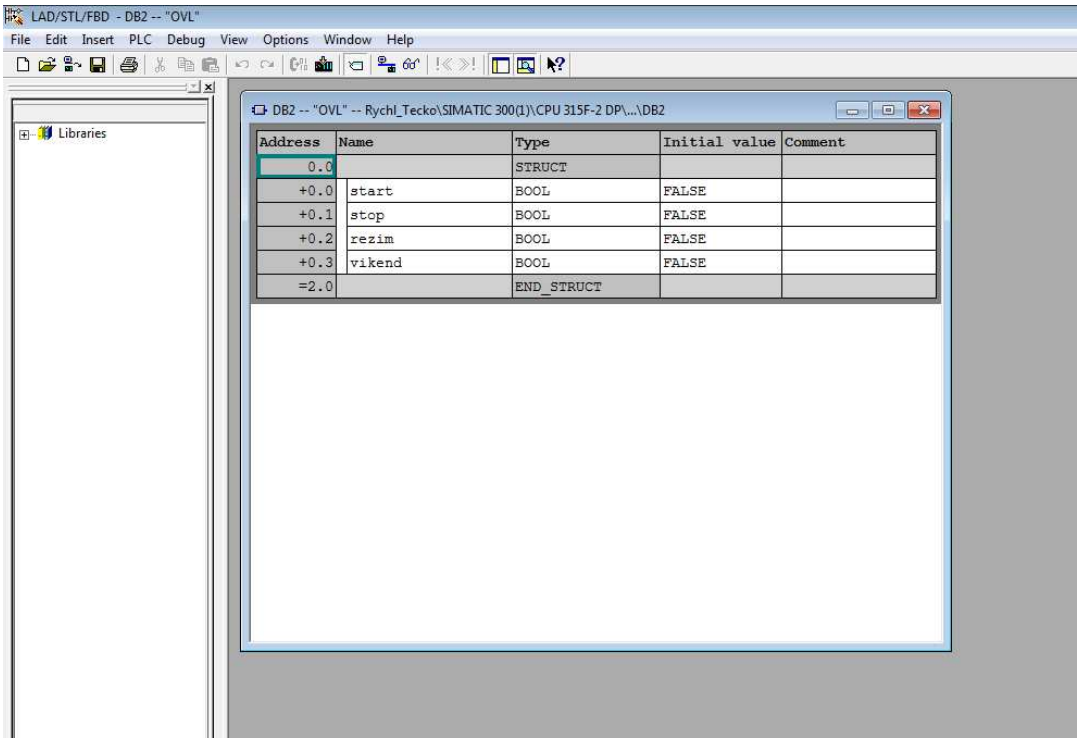


Obr. 13 Volání FB1 z OB1 u rychlostní křižovatky ve tvaru „T“

U tohoto typu křižovatky se program v bloku FB1 bude psát v programovacím jazyce zvaném Ladder diagram (dále Ladder). Ladder je grafický programovací jazyk pracující na principu reléových schémat. Opět je rozdělen na části (networky), kdy na začátku každého se udávají podmínky, v našem případě různými kombinacemi spínacích a rozpínacích kontaktů. Na konci každého networku je opět některá z událostí, které mají po splnění podmínky nastat. Tu představuje cívka relé, která nastaví či vypne (resetuje) nějaké světlo semaforu, příp. spustí časovač (timer). Po uběhnutí času v časovači je sepnut kontakt tohoto časovače v dalším networku a dochází k nastavení nebo resetování dalších světel semaforu. Ukázka části řídicího systému v programovacím jazyce Ladder diagram je na Obr. 14.



Obr. 14 Ukázka programu v jazyce Ladder diagram



Obr. 15 Datový blok DB2 s ovládacími proměnnými u rychlostní křižovatky ve tvaru „T“

Opět zde byl vytvořen datový blok DB2 pro adresaci proměnných sloužících k ovládání křižovatky. Jak je již z Obr. 15 patrné, nefigurují zde tlačítka pro požadavek chodce o přejití přes přechod. Na Obr. 16. v tabulce všech proměnných je vidět, že i zde je zapotřebí celkem velkého počtu vnitřních proměnných, i když ne tolik jako v předcházejícím případě. To je způsobeno tím, že se nevyskytují již zmiňované přechody pro chodce.

Celý algoritmus vyjadřující vytváření řídicího systému rychlostní křižovatky ve tvaru písmene „T“ bez přechodů pro chodce je znázorněn vývojovým diagramem tohoto problému na Obr. 17.

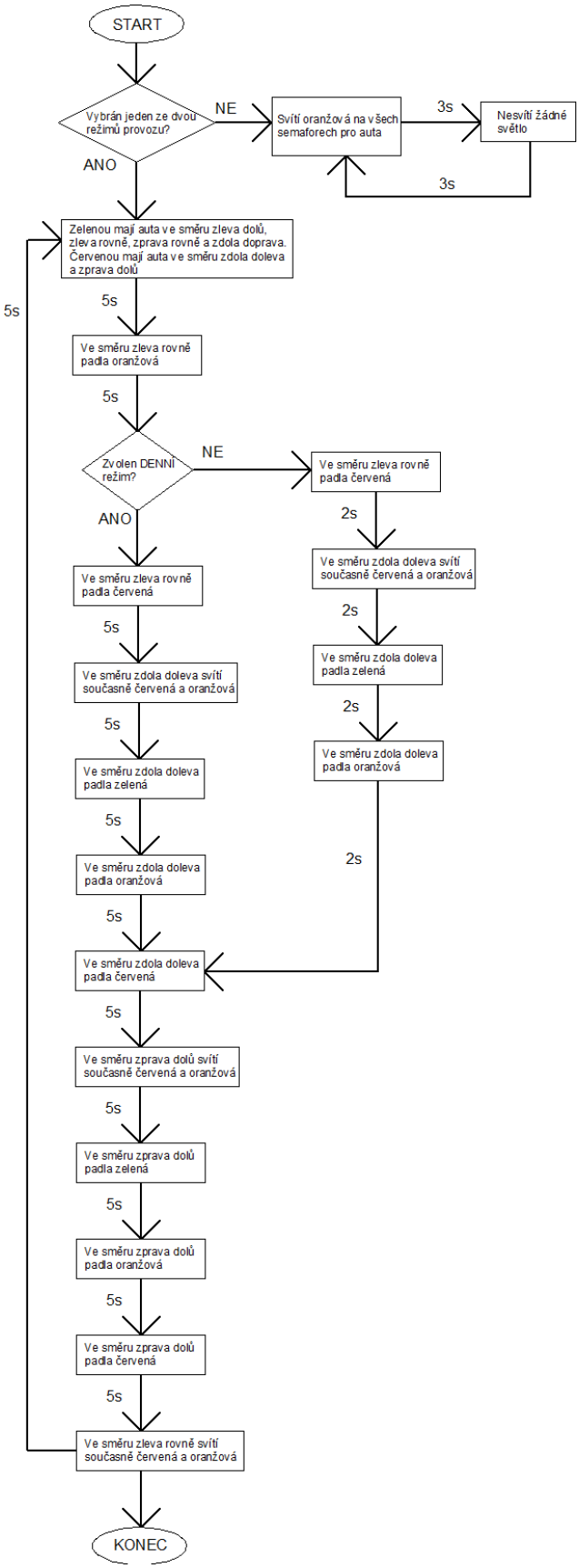
Symbol Editor - [S7 Program(1) (Symbols) -- Rychl_Tecko\SIMATIC 300(1)\CPU 315F-2 DP]

Symbol Table Edit Insert View Options Window Help

All Symbols

	Statu	Symbol	Address	Data type	Comment
1		blik_oranz	M 101.5	BOOL	
2		blik_oranz_2	M 101.6	BOOL	
3		M1	M 100.0	BOOL	
4		M10	M 101.1	BOOL	
5		M11	M 101.2	BOOL	
6		M12	M 101.3	BOOL	
7		M13	M 101.4	BOOL	
8		M2	M 100.1	BOOL	
9		M3	M 100.2	BOOL	
1		M4	M 100.3	BOOL	
1		M5	M 100.4	BOOL	
1		M6	M 100.5	BOOL	
1		M7	M 100.6	BOOL	
1		M8	M 100.7	BOOL	
1		M9	M 101.0	BOOL	
1		OVL	DB 2	DB 2	
1		zdola_leva_c	Q 40.3	BOOL	
1		zdola_leva_o	Q 40.4	BOOL	
1		zdola_leva_z	Q 40.5	BOOL	
2		zdola_prava_c	Q 40.0	BOOL	
2		zdola_prava_o	Q 40.1	BOOL	
2		zdola_prava_z	Q 40.2	BOOL	
2		zleva_dolu_c	Q 41.7	BOOL	
2		zleva_dolu_o	Q 42.0	BOOL	
2		zleva_dolu_z	Q 42.1	BOOL	
2		zleva_rovne_c	Q 41.4	BOOL	
2		zleva_rovne_o	Q 41.5	BOOL	
2		zleva_rovne_z	Q 41.6	BOOL	
2		zprava_dolu_c	Q 41.1	BOOL	
3		zprava_dolu_o	Q 41.2	BOOL	
3		zprava_dolu_z	Q 41.3	BOOL	
3		zprava_rovne_c	Q 40.6	BOOL	
3		zprava_rovne_o	Q 40.7	BOOL	
3		zprava_rovne_z	Q 41.0	BOOL	
3					

Obr. 16 Tabulka proměnných pro rychlostní křižovatku ve tvaru „T“



Obr. 17 Vývojový diagram řídicího systému rychlostní křižovatky ve tvaru „T“

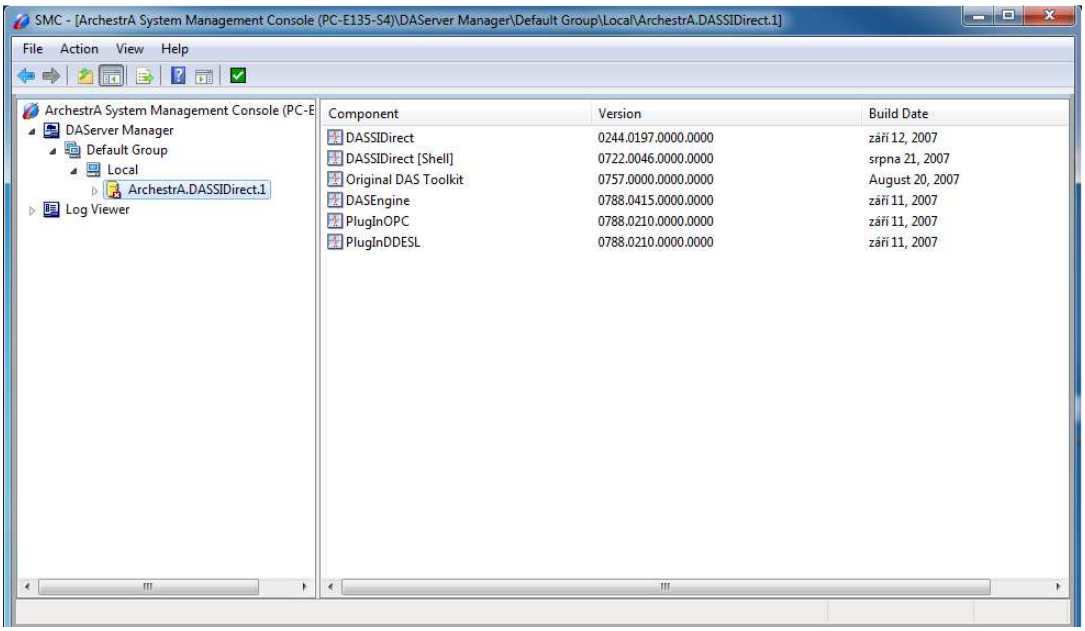
3 VIZUALIZACE

3.1 Nastavení komunikace

Pro vytvoření vizualizace je nejprve zapotřebí nastavit komunikaci mezi řídicím systémem v PLC a programem, ve kterém je vizualizace vytvořena. V tomto případě je pro vytvoření vizualizace vybrán SCADA systém InTouch od společnosti Wonderware. Jedná se o jeden z nejčastěji používaných softwarových nástrojů pro tvorbu vizualizace.

Samotnou komunikaci PLC zajišťuje komunikační procesor CP. Po otevření SIMATIC Manageru se v projektu, kde je vytvořen řídicí systém rozklikne hardwarová konfigurace. Zde by již měl být přidaný modul komunikačního procesoru CP z původní konfigurace. Nyní je zapotřebí mu nastavit správnou IP adresu. Dvakrát se klikne na komunikační procesor. Otevře se okno a v něm se zadává již zmiňovaná IP adresa komunikačního procesoru. Poté se opět hardwarová konfigurace uloží, zkompiluje a provede se download, aby se změněná konfigurace nahrála do PLC. Na fyzickém modulu komunikačního procesoru CP by měla svítit LED dioda u stavu *RUN*.

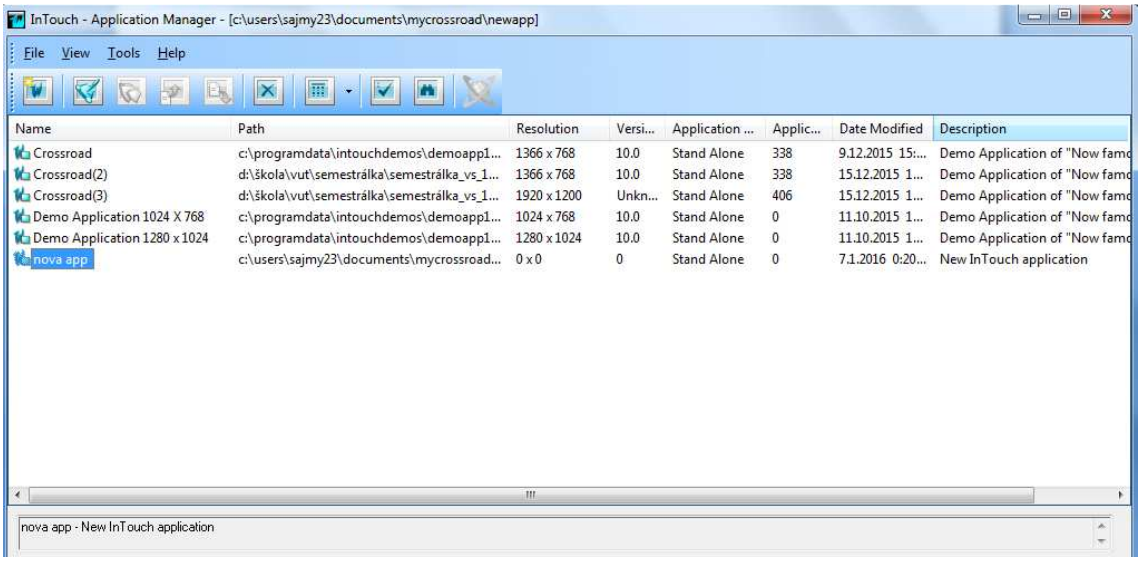
Nyní se přes tlačítko *START* na ploše PC vyhledá program System Management Console, také od společnosti Wonderware. Po otevření tohoto programu je vlevo ve sloupci položka DAServer Manager. Po rozklikání „stromu“ DAServeru se zobrazí ArchestrA.DASSIDirect.1. Tento server je potřeba aktivovat zelenou značkou na horní liště.



Obr. 18 Nastavení serveru v systému System Management Console

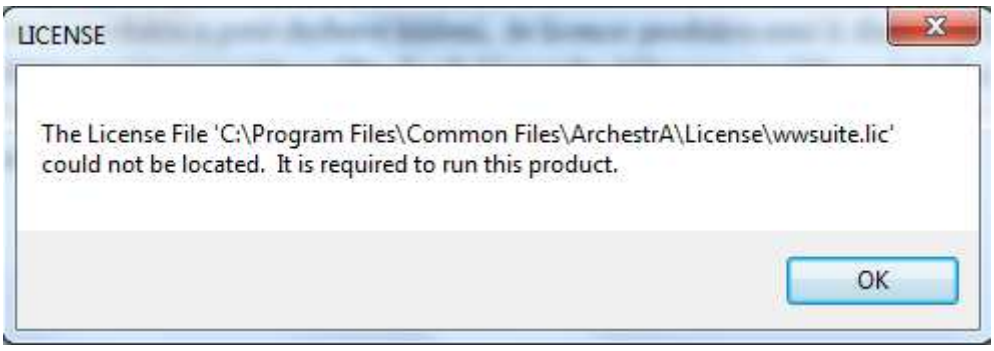
3.2 Vytvoření vizualizace

Po nastavení serveru pro komunikaci PLC s vizualizací v InTouch se se spustí program InTouch. Pokud není ikonka na uživatelské ploše, je možné si program vyhledat klasicky přes tlačítko *START*. Otevře se Application Manager, ve kterém jsou již uspořádány dříve používané projekty tzv. aplikace. Přes nabídku *File -> New* se vytvoří aplikace nová. Při vytváření nové aplikace se musí zadat umístění a název nové vytvářené aplikace.



Obr. 19 InTouch Application Manager

Po dvojitém kliknutí na tuto nově vytvořenou aplikaci se začne spouštět InTouch Window Maker. Při jeho spouštění však vyskočí několik okének s hlášením. Nejprve hlášení o licenci produktu a poté chybové hlášení, že licence produktu není k dispozici. Obě dvě hlášení lze vidět na Obr. 20 a 21. U prvního se klikne na *OK* a u druhého, chybového hlášení, na tlačítko *Ignorovat*. Program InTouch Window Maker se začne spouštět v Demo režimu.



Obr. 20 První hlášení při spouštění InTouch Window Maker

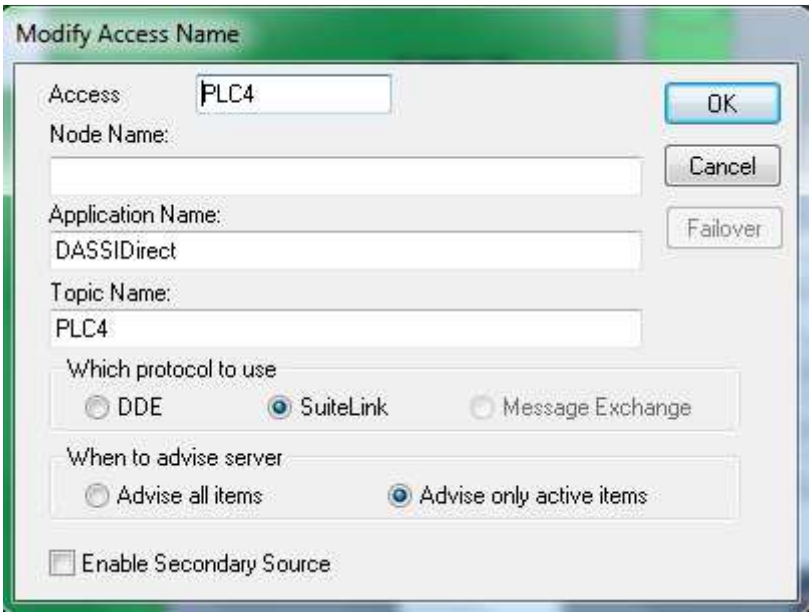
Obr. 21 Druhé chybové hlášení při spouštění InTouch Window Maker

InTouch Window Maker se dělí na dvě části. Na část vývojovou (Development) a část uživatelskou (Runtime). Ve vývojové části se vytváří vlastní vizualizace, zatímco v části uživatelské se již hotová vizualizace stává aktivní a může simulovat chod řídicího systému.

Program se implicitně spouští ve verzi vývojové. Nejprve je zapotřebí vytvořit si nové okno, ve kterém bude vizualizaci vytvářena. Nové okno se vytvoří přes nabídku *File -> New Window*, zvolí se název okna, barva jeho pozadí a potvrdí se *OK*. Pomocí jednoduchých nástrojů v liště na pravé straně okna se nechá nakreslit libovolná vizualizace, v našem případě křižovatku. Často se používají i složité nástroje, ke kterým se dostaneme přes ikonku *Wizards.*

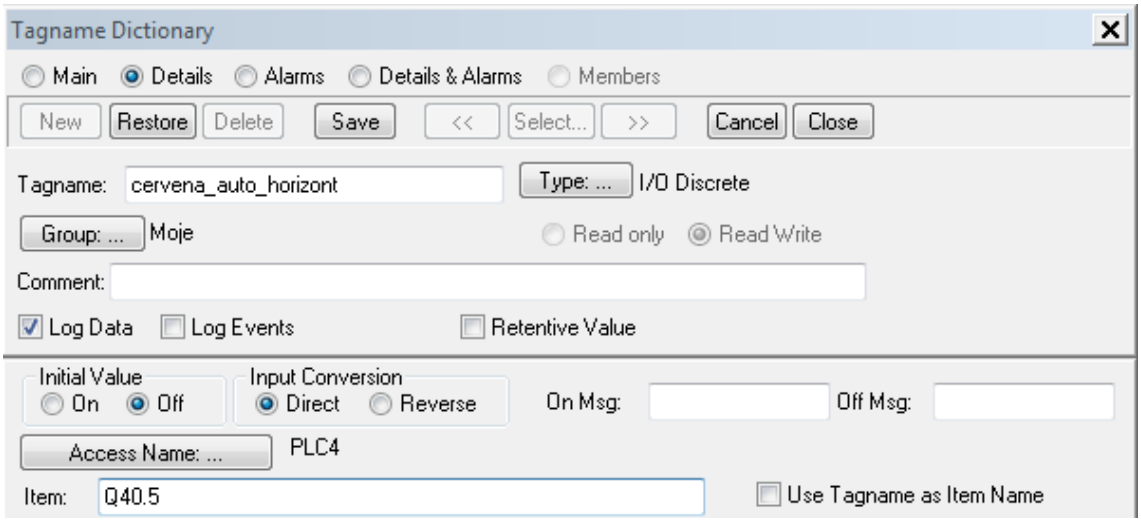
3.3 Propojení vizualizace s PLC, vytvoření proměnných

Ze začátku se musí vytvořit tzv. Access Name. Vyskytuje se v nabídce *Special -> -> Access Names*. Po otevření této nabídky se nový Access Name vytvoří tlačítkem *Add.* Správné nastavení je vidět na Obr. 22. Do položky *Access* se zadává název nově vytvořeného Access Name. Do položky *Application Name* je zadán *DASSIDirect*, což je server přes který vizualizace s PLC komunikuje, a do pole *Topic Name* se zadá číslo PLC, na kterém se v laboratoři pracuje. V tomto případě to je PLC4. V laboratoři jsou na PLC nalepeny nálepky s čísly PLC, takže lze jednoduše zjistit u jakého PLC se právě pracuje. Další dvě položky okna Access Names se nechají zaškrtnuty tak, jak je vidět na již zmíněném Obr. 22.



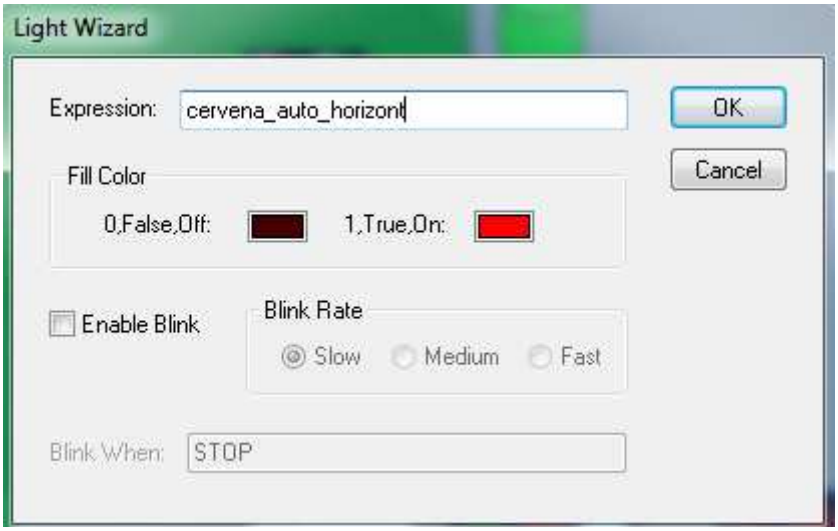
Obr. 22 Nastavení Access Name

Nyní se může přejít k vytváření proměnných. Proměnné se v programu InTouch vytvářejí pomocí tzv. slovníku proměnných. Nalezne se pod záložkou *Special -> -> Tagname Dictionary*. Po otevření slovníku proměnných se vytvoří nová proměnná tlačítkem *New*. Okno slovníku vidíme na Obr. 23. Zde se napíše název proměnné, a zvolí se její datový typ. Jelikož se v řídicím systému používají pouze logické proměnné, zvolí se zde typ *I/O Discrete*. Dále se musí proměnné nastavit Access Name, který byl vytvořen v předcházejícím kroku. Klikne se na tlačítko *Access Name* a z nabídky se vybere ten vytvořený. Do položky *Item* se uvádí adresa příslušné proměnné z řídicího systému PLC.



Obr. 23 Tagname Dictionary - vytváření proměnných

Stejným způsobem se vytvoří všechny proměnné, které budou potřeba ve vizualizaci. Nakonec, když je samotná vizualizace hotová a jsou vytvořené i všechny proměnné, je potřeba tyto jednotlivé proměnné přiřadit k vytvořeným objektům. Přiřazování proměnných k objektům je znázorněno na Obr. 24. V tomto případě jsou těmito objekty myšleny právě světla semaforů, příp. ovládací tlačítka.



Obr. 24 Přiřazení proměnné k červenému světlu semaforu

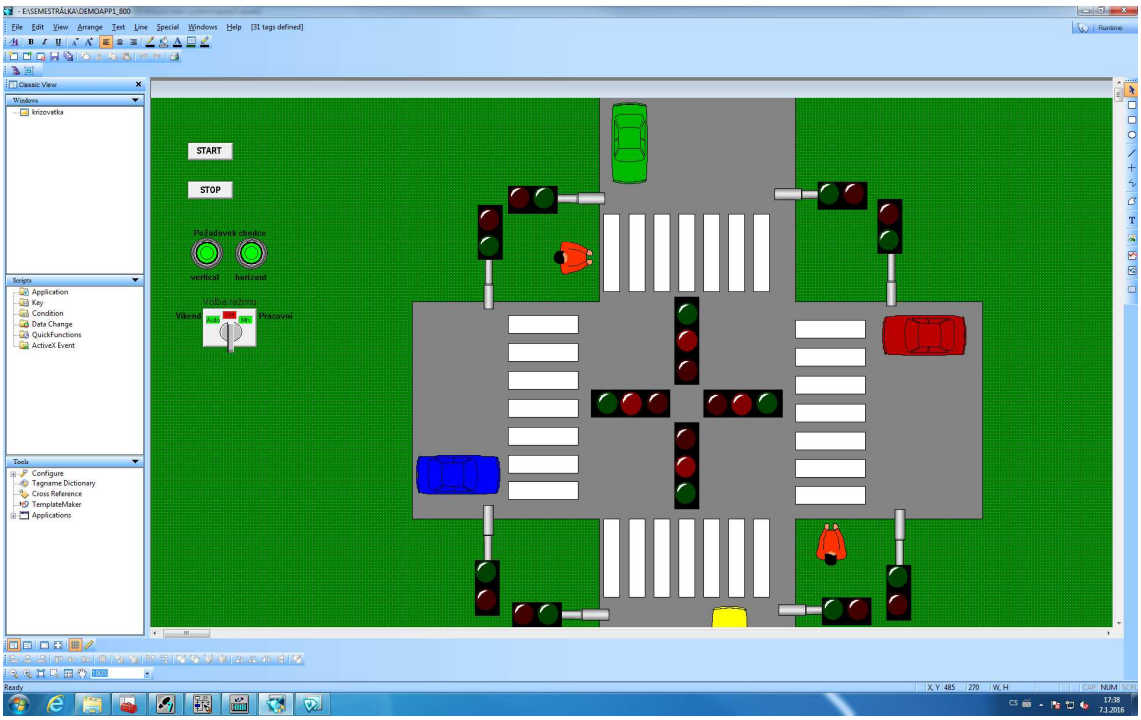
Na objekt se dvakrát poklepe myší a vyskočí okno znázorněné na Obr. 24 nebo jemu podobné. Záleží na objektu, kterému je proměnná přiřazována. Do pole *Expression* se napíše název proměnné, která má být danému objektu přiřazena. V případě zapomenutého názvu proměnné, lze do pole dvakrát kliknout myší a vyskočí okno slovníku proměnných, kde se již vybere proměnná, která je potřeba. Na Obr. 24 jsou také vidět dvě barevná okénka *Fill Color*, jedno *True* a druhé *False*. Zde je možné si zvolit barvu světla semaforu, pokud je aktivní a svítí (True), nebo pokud aktivní není a nesvítí (False).

3.4 Konečný návrh okna pro vizualizaci

3.4.1 Křižovatka typu „Kříž“

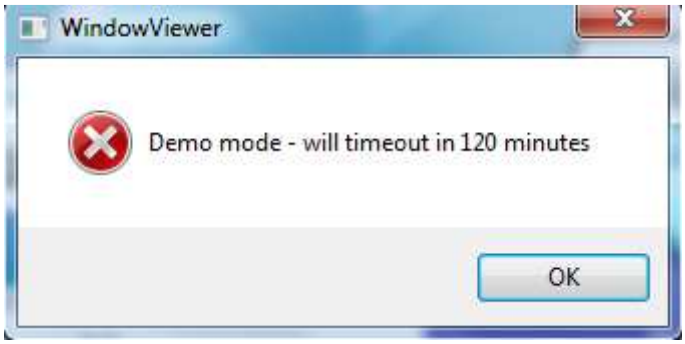
Na Obr. 25 je vidět návrh vizualizačního okna křižovatky typu „Kříž“ ve vývojovém prostředí (Development) programu InTouch Window Maker. Pokud je již všechno hotovo, jak řídicí systém, tak i vizualizace křižovatky, lze přepnout do uživatelského režimu a vyzkoušet, zda vše správně funguje. Jak řídicí systém, tak komunikace mezi ním a vizualizací křižovatky. Do uživatelského režimu se přepne vpravo nahoře kliknutím na tlačítko *Runtime*. Opět vyskočí několik chybových hlášení. První dvě jsou stejné jako při spouštění InTouch Window Maker (viz Obr. 20 a 21)

a další, které říká, že uživatelský režim může být v Demo režimu spuštěn pouze 120 min (viz Obr. 26). Finální vizualizace fungujícího řídicího systému pro řízení křižovatky je znázorněna na Obr. 27. Je na ní zobrazena křižovatka se semafory pro auta i pro chodce. V levém horním rohu se vyskytují ovládací prvky. Tlačítka *START* a *STOP* slouží pro spouštění nebo zastavení programu. Pod nimi jsou umístěna dvě kulatá tlačítka, která slouží jako požadavek chodce, který chce přejít přes přechod. Jedno pro vertikální směr a druhé pro směr horizontální. Posledním ovládacím prvkem je třístavový přepínač. Tím si uživatel volí režim řízení křižovatky. Pokud je přepínač ve stavu *OFF*, znamená to, že není zvolen režim žádný. V tom případě ne semaforech blikají oranžová světla. Další dva stavy reprezentují buď pracovní, případně víkendový režim.

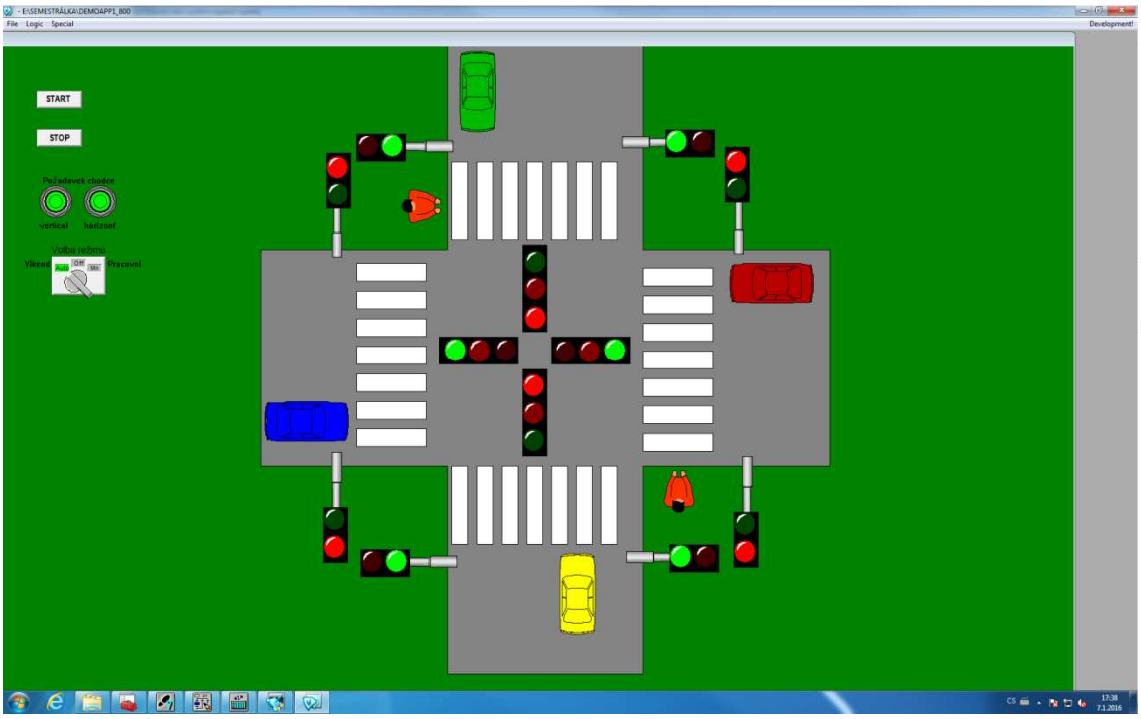


Obr. 25 Návrh vizualizace křižovatky typu „Kříž“ ve vývojovém prostředí

U tohoto typu křižovatky byla také vytvořena jakási simulace pohybujících se vozidel. Vozidla, která mají v danou chvíli zelenou, projedou křižovatkou a na jejím konci se zastaví. Po chvíli zmizí a v dalším cyklu se tato animace opakuje. Bohužel vozidla jezdí pouze rovným směrem. Vertikálně nebo horizontálně, dle toho, v jakém směru jedou. K vytvoření animace pohybujících se vozidel je zapotřebí si ve SCADA systému InTouch vytvořit další proměnné. Musí se jednat o analogové proměnné, zde je zvolen typ *INTEGER*. Vytvoření takovéto proměnné se provádí stejným způsobem, jako proměnné již zmíněné, ve slovníku proměnných (*Tagname dictionary*).

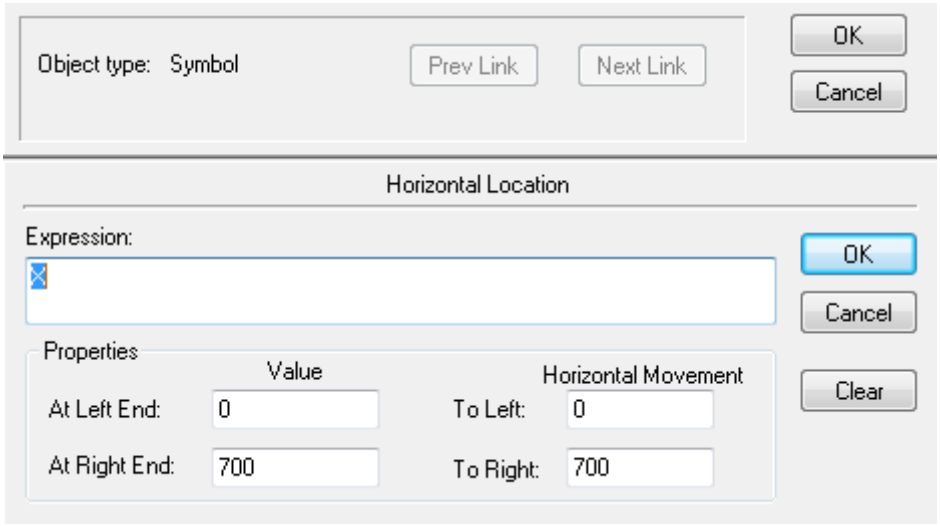


Obr. 26 Oznámení o chodu uživatelského režimu po dobu 120 min

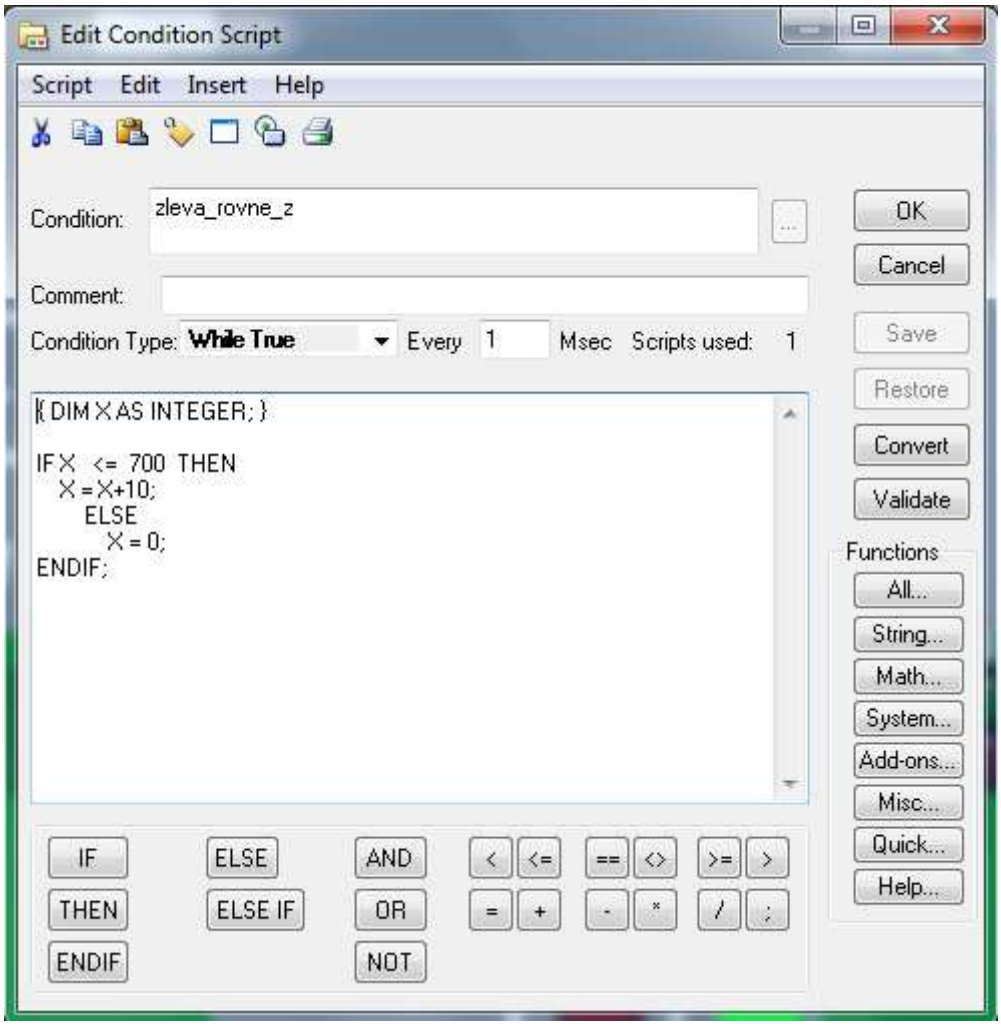


Obr. 27 Vizualizace systému na řízení křižovatky typu „Kříž“

Po vytvoření takovéto proměnné je potřeba tuto proměnnou přiřadit pohybujícímu se vozidlu. Po kliknutí na vozidlo se zvolí ve vybraném menu tlačítko *horizontal* příp. *vertical*, dle toho, jakým směrem se má vozidlo pohybovat. Po výběru tohoto směru se zobrazí další okno (viz Obr. 28). V něm se nastavují souřadnice, odkud a kam se má vozidlo pohybovat. Také se v něm musí vozidlu přiřadit vytvořená analogová proměnná. Po poklikání do položky *Expression* se otevře slovník proměnných a tam se analogová proměnná zvolí. Po tomto nastavení je potřeba ještě vytvořit *Condition Script*, který zajistí opakování pohybu vozidla. Tento skript se vytvoří v nabídce *Special -> Scripts -> Condition Scripts* příp. v okně *Scripts* v levé liště *Window Makeru*. Příklad skriptu je uveden na Obr. 29.

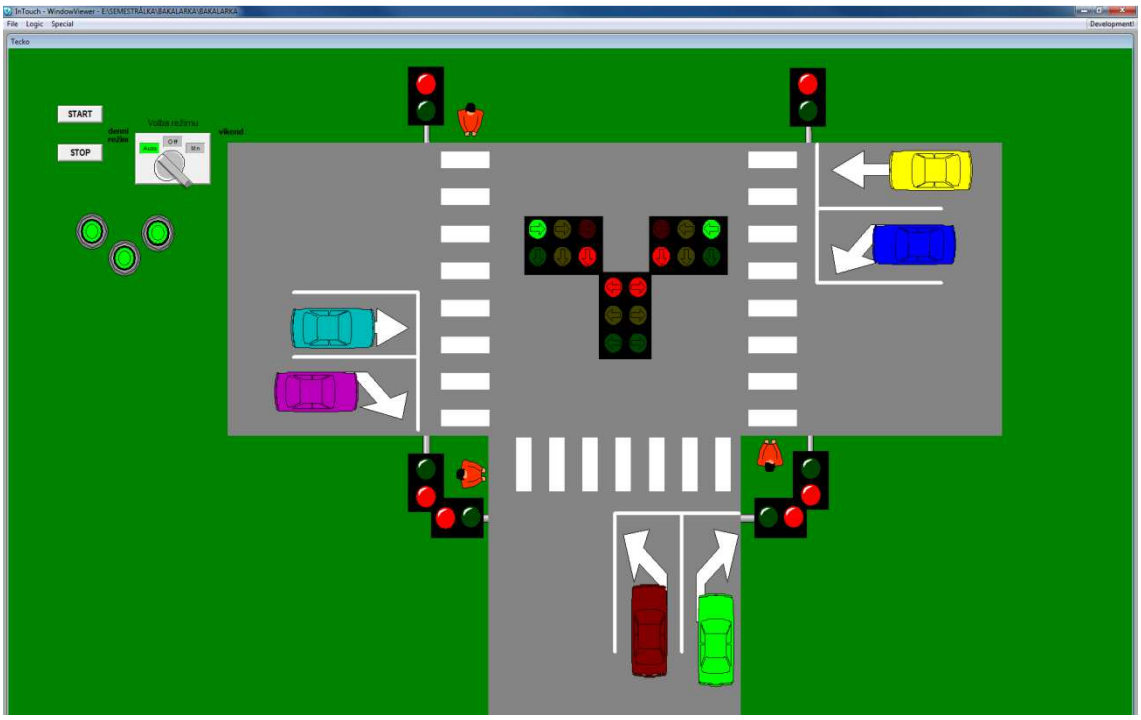


Obr. 28 Nastavení souřadnic pohybu vozidla



Obr. 29 Příklad skriptu pro pohyb vozidla

3.4.2 Křižovatka ve tvaru písmene „T“ s odbočovacími pruhy

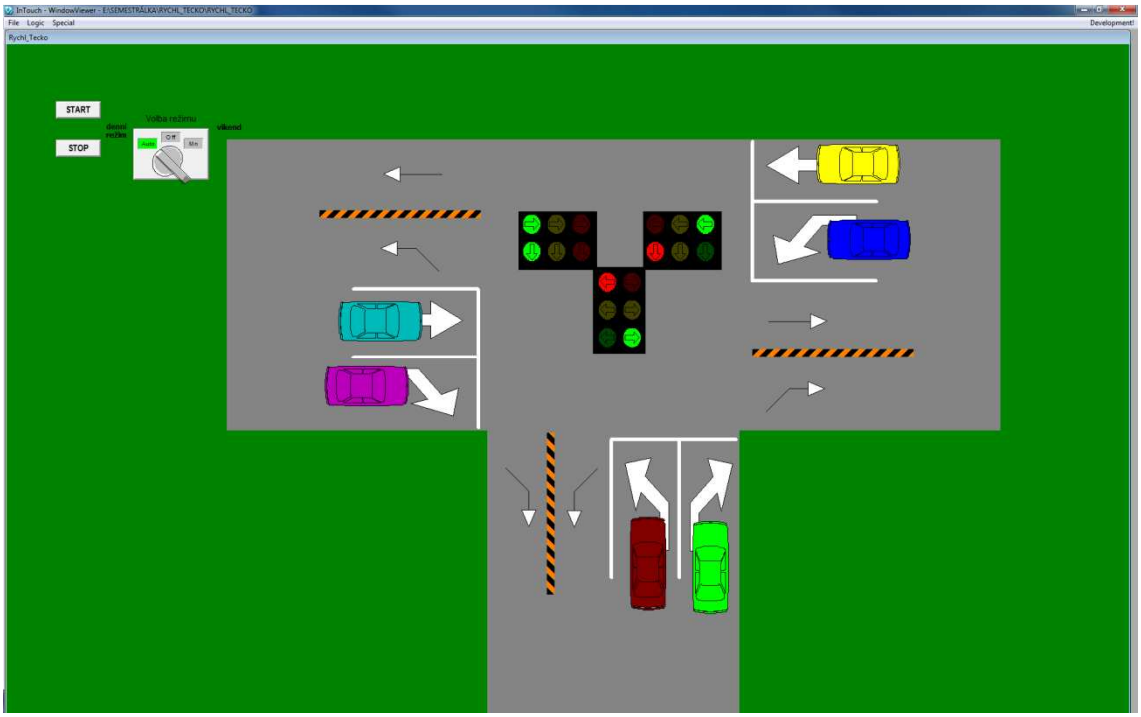


Obr. 30 Konečná vizualizace křižovatky ve tvaru písmene „T“

Na Obr. 30 je vidět finální návrh okna pro vizualizaci křižovatky ve tvaru písmene „T“ s odbočovacími pruhy. V levém horním rohu jsou opět jako u předchozího typu křižovatky ovládací prvky na její řízení. Tlačítka *START* a *STOP* ke spouštění a zastavení semaforů křižovatky. Dále pak třístavový přepínač na přepínání režimů, kde je možné přepínat mezi denním a víkendovým režimem. Pokud není zvolen žádný režim, na semaforech blikají oranžová světla, tak jak to bývá například v noci. Při přepínání režimů je vždy zapotřebí nejprve program ukončit tlačítkem *STOP*, přepnout režim a poté spustit tlačítkem *START*.

3.4.3 Rychlostní křižovatka ve tvaru písmene „T“ bez chodců

Konečná vizualizace tohoto typu křižovatky je ukázána na Obr. 31. Zde jsou jako ovládací prvky pouze tlačítka *START* a *STOP* na spouštění a zastavení semaforů a dále přepínač mezi režimy, které fungují stejně jako u předchozích dvou typů křižovatek.



Obr. 31 Konečná vizualizace rychlostní křižovatky ve tvaru písmene „T“

4 ZÁVĚR

Tato práce se zabývala vytvořením SW modelu pro řízení různých typů jednoúrovňové křižovatky. K řešení tohoto problému byl využit programovatelný automat SIMATIC S7-300 od společnosti SIEMENS. Nejprve se vytvořil řídicí systém pro řízení křižovatky ve vývojovém prostředí SIMATIC Manager, také od společnosti SIEMENS. Jako programovací jazyk byly postupně zvoleny grafický programovací jazyk SIMATIC S7-GRAPH. Tento jazyk byl zvolen proto, že funguje na principu stavového automatu, kdy se mezi jednotlivými kroky programu přechází po splnění podmínek uvedených v příslušném přechodu. Tento způsob se skvěle hodil k řešení řízení křižovatky, kde v jednotlivých krocích jsou rozsvícena či zhasnuta určitá světla semaforu a po uplynutí času, který spíná kontakt v následném přechodu, se přechází do kroku následujícího. V tomto následujícím kroku jsou již rozsvícena nebo zhasnuta světla jiná, tak aby byl zajištěn správný chod semaforů. Po takto uplynutém cyklu správného chodu semaforu je nakonec programu přidán skok zpět na první krok, a tím je zajištěno cyklické opakování. Jako další programovací jazyky byly zvoleny jazyky běžně používané v praxi. A to STL a Ladder diagram (LAD). Jelikož je tato práce určená jako podklad k laboratorní úloze pro studenty, je výběr několika různých programovacích jazyků zcela na místě. Poté, co byl dokončen řídicí systém, byla

následně vytvořena i vizualizace ve SCADA systému InTouch od společnosti Wonderware. Komunikace řídicího programu v PLC a vizualizací byla nastavena přes DAServer, který tuto komunikaci umožňuje. Po vytvoření vizualizace se proměnným, které se v systému používají, musel přiřadit správně vytvořený Acces Names a taky adresa z PLC. Tyto proměnné se pak přiřadily objektům vizualizace (např. světlům semaforu). Nyní se již mohlo vizualizaci přepnout z vývojového prostředí (Development) do prostředí uživatelského (Runtime). Program spustit a sledovat, zda vytvořený řídicí systém funguje ve vizualizaci správně.

Tato práce ukazuje způsob vytváření a testování řídicího systému. Vytvořením vizualizace je možné sledovat jeho chování a případně dolaďovat jeho chyby. Vizualizace dále slouží jako kontrola pro obsluhujícího pracovníka, který pomocí ní může na dálku sledovat správné chování řídicího systému případně tento řídicí systém ovládat, což je způsob daleko komfortnější, než kdyby musel neustále stát poblíž a sledovat zda se řídicí systém chová správně.

POUŽITÁ LITERATURA

[1] Ing. Jan Pásek, CSc.: Automatizace procesů, Laboratorní cvičení I. FEKT VUT v Brně, Brno

[2] Ing. Jan Pásek, CSc.: Programovatelné automaty v řízení technologických procesů. Brno, 30. 11. 2007

[3] SIMATIC System Manuals. Siemens AG, 2003 - 2006

SEZNAM PŘÍLOH