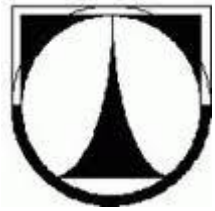
Technická univerzita v Liberci

Fakulta strojní



DIPLOMOVÁ PRÁCE

Simulace PLC kontroléru Yaskawa MP2300 v prostředí MS Excel

Simulation PLC controller Yaskawa MP2300 in MS Excel

Liberec 2011 **Marta Braierová**

Technická univerzita v Liberci

Fakulta strojní

Studijní obor: M2301 Strojní inženýrství – Automatizované systémy řízení ve strojírenství

Zaměření: Automatizace inženýrských prací

Katedra aplikované kybernetiky

Simulace PLC kontroléru Yaskawa MP2300 v prostředí MS Excel

Simulation PLC controller Yaskawa MP2300 in MS Excel

Marta Braierová

Vedoucí diplomové práce:	prof. Ing. Miroslav Olehla, CSc.
Konzultant diplomové práce:	Ing. Pavel Bureš
Počet stran:	69
Počet obrázků:	41
Počet tabulek:	9
Počet příloh:	7

ANOTACE

TECHNICKÁ UNIVERZITA V LIBERCI

Fakulta strojní

Katedra aplikované kybernetiky

Studijní obor:	M2301 Strojní inženýrství – Automatizované systémy řízení ve strojírenství
Studijní zaměření:	Automatizace inženýrských prací
Diplomant:	Marta Braierová
Téma práce:	Simulace PLC kontroléru Yaskawa MP2300 v prostředí MS Excel
Theme of work:	Simulation PLC controller Yaskawa MP2300 in MS Excel
Rok obhajoby DP:	2011
Vedoucí DP:	prof. Ing. Miroslav Olehla, CSc.
Konzultant:	Ing. Pavel Bureš, výzkumný pracovník oddělení Mechatronika VÚTS, a.s.

Resumé: Tato práce se zabývá tvorbou programu pro simulaci PLC kontroléru Yaskawa MP2300 v prostředí MS Excel použitím jazyka Visual Basic. Ve vytvořeném programu simulujeme logiku vazeb bitových proměnných, uživatelskou funkci sčítání a pohyb elektronické vačky. Sestrojením mustru obecné funkce lze následně naprogramovat libovolné uživatelské funkce pouhým doplněním příkazů do bloků. Přínosem této práce je v možném nahrazení v určitých případech drahých odlaďovačů jednoduchým simulačním Excel programem.

Abstract: This diploma thesis is concerned with creation of the program for PLC controller Yaskawa simulation using by language Visual Basic in MS Excel. In created program we are simulating logic of bit variable feedback, user functions for counts and move of electronic cam. After construction of general function sample there is possibility to programming any user function only by filling commands into blocks. Benefit of this work is substitution of expensive absorption traps by easy simulation Excel program.

Seznam obrázků a tabulek

Seznam obrázků:

Seznam tabulek:

Obsah

Použité zkratky:

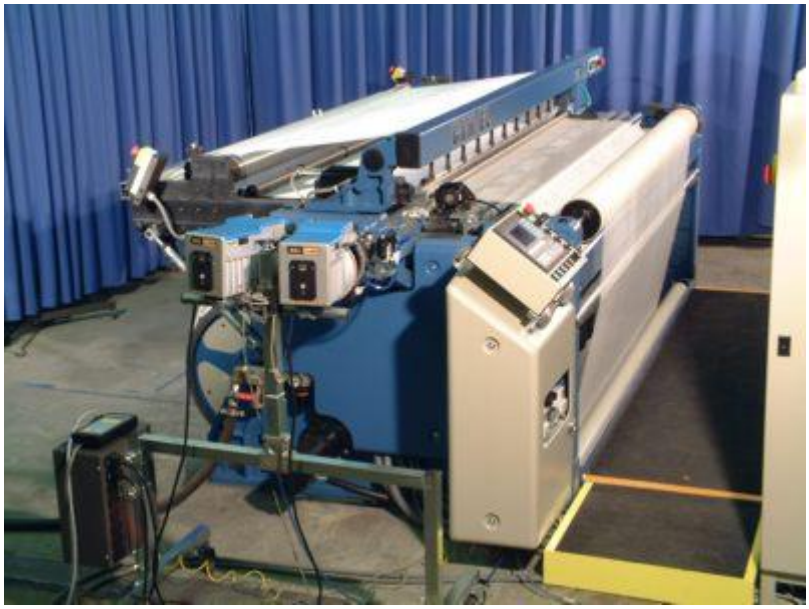
Str. 3	PLC	Programmable Logic Controller/Programovatelný logický automat
Str. 3	VÚTS	Výzkumný ústav textilních strojů
Str. 8	IPC	Indrustrial Personal Computer/Průmyslový počítač
Str. 8	CPU	Central Processing Unit/ Procesor
Str. 10	A/D	Analogový/Digitální převodník
Str. 10	RAM	Random-Access Memory/Paměť s přímým přístupem
Str. 11	PC	Personal Computer/Počítač
Str. 11	LD	Ladder Diagram/Reléové schéma
Str. 11	FBD	Function Block Diagram/Jazyk funkčních bloků
Str. 11	SFC	Sequential Function Chart/Jazyk sekvenčního programování
Str. 11	IL	Instruction List/Jazyk mnemokódů
Str. 11	ST	Structured Text/Strukturovaný text
Str. 14	I/O	Input/Output/Vstup/Výstup
Str. 15	LED	Elektroluminiscenční dioda
Str. 23	HMI	Human Machine Interface/Rozhraní člověk/stroj
Str. 24	MAC	Media-Access Control/Fyzická adresa
Str. 24	ROM	Read-Only Memory/Pamět pouze pro čtení
Str. 24	FPU	Floating-Point Unit/ Koprocesor
Str. 24	DWG	Drawing/Výkres
Str. 40	VBA	Visual Basic

Úvod

Kontrolér MP2300 je vyspělý automat, který prostřednictvím vývojového prostředí MPE720 umožňuje v jednom projektu programovat řídicí systémy v oblasti PLC a v oblasti souvislých pohybů os (Motion). Výzkumný ústav textilních strojů (dále jen VÚTS) Liberec na této hardwarové platformě vyvíjí aplikace elektronických vaček. PLC je programováno klasickým Ladder programem, který umožňuje použití rozličných číselných a bitových proměnných, struktur, uživatelských funkcí apod. Vyvíjené algoritmy uživatelských funkcí v aplikacích elektronických vaček je nutné testovat a odlaďovat na hardware Yaskawa, který je obtížně přenositelný, drahý a často vyžaduje práci ve ztížených provozních podmínkách. Simulace PLC v prostředí MS Excel by přinesla komfort při odlaďování logických struktur a algoritmů, zvláště uživatelských funkcí.

1. Výzkumný ústav textilních strojů

VÚTS byl založen roku 1951 za účelem výzkumu a vývoje nových textilních technologií. Za dobu svého působení na trhu obohatil světové textilní strojírenství o unikátní vynález tryskového způsobu tkaní, první patenty na bezvřetenové předení, technologii výroby netkaných textilií Arachne, křížem soukací automat Autosuk či rotační šablonový tiskací stroj v preserovém uspořádání Roša.



Obr. 1.1: Vzduchový tryskový tkací stroj

V dnešní době se VÚTS soustřeďuje na výzkum, vývoj, konstrukci a výrobu strojů a zařízení pro zpracovatelský průmysl v oblastech textilních strojů, balící, reprografické, polygrafické, potravinářské, sklářské, obráběcí a montážní techniky. VÚTS nabízí širokou škálu služeb, počínaje provedení informační analýzy přes konstrukci až po samotnou výrobu funkčních modelů, prototypů a jejich ověřením, zajištěním sériové výroby[7].

2. Obecný popis programovatelného logického automatu

Programovatelný logický automat neboli PLC (Programmable Logic Controller) je průmyslový počítač uzpůsobený pro řízení technologických procesů, strojů, automatizaci budov a jiné. Jedná se o jednoduché a lehce programovatelné jednotky s mnoha vstupy a výstupy pro snadné připojení senzorů, displejů, spínačů a tlačítek, motorů a různých dalších přístrojů a zařízení. Data jsou posílána přes drátové i bezdrátové komunikace. Funkce celého PLC i ovládání připojených prvků je řízeno uloženým programem, který lze snadno vytvořit pomocí výrobcem dodávaného grafického vývojového softwaru[1].

Obr. 2.1: Kontrolér MP2300

Počátky PLC spadají do druhé poloviny 60. let, kdy hlavním důvodem jejich vzniku bylo nahrazení pevných reléových obvodů v automobilovém průmyslu. Za autora prvního návrhu PLC stroje je považován Dick Morlay, podle něhož se začal vyrábět Modicon (MOdular DIgital CONtroller). PLC v současnosti řídí až 80% aplikací v průmyslu. Vedle tradičních aplikací ve strojírenství, jako jsou řízení strojů, mechanismů, linek a nejrůznějších výrobních technologií, v manipulační, dopravní a skladové technice, se dnes PLC hojně uplatňuje v energetice od regulace turbín, kompresorů až po předávací a výměníkové stanice, klimatizační jednotky, kotelny, ale i při regulaci spotřeby elektrické energie, řízení rozvoden a nově i v technice inteligentních budov. Nacházejí uplatnění také v zemědělství, potravinářství, v procesu měření, monitorování, sledování kvality a v mnohých dalších odvětvích. S vývojem PLC je snaha, o co možná největší miniaturizaci, čímž se výrazně snižuje prostorová náročnost[1][9]. Mezi hlavní požadavky na PLC patří robustnost, kompaktnost, spolehlivost, konfigurovatelnost a v neposlední řadě servis.

Pod pojmem robustnost máme na mysli vhodnou konstrukci pro způsob a místo využití. PLC by mělo být odolné proti mechanickému poškození, působení prachu, vlhkosti, nízkým a vysokým teplotám a vibracím.

Pro zajištění kompaktnosti se PLC instalují co nejblíže technologii a je snaha o co nejmenší rozměry.

PLC jsou používány v aplikacích s nepřetržitým, a také s velmi nebezpečným provozem, jako například v chemickém průmyslu, nebo v jaderných elektrárnách, kde je ohroženo zdraví lidí či dokonce jejich život, a proto jsou také kladeny velké nároky na spolehlivost. Bezpečnost zařízení je klasifikována dle norem IEC 61508 a IEC 61511, definující minimální požadavky na zařízení, kterými je hlavně rychlý servis a dostupnost náhradních dílů minimálně deset let po ukončení výroby.

2.1. Výhody použití PLC

Programovatelný logický automat lze s výhodou použít díky

- rychlé realizaci systému
- diagnostice
- snadnému a rychlému přizpůsobení programu při změně požadavků

Pod pojmem rychlá realizace systému máme na mysli nakoupení potřebných modulů programovatelného automatu pro funkci dané aplikace. Jednotlivé technické vybavení se již nemusí vyvíjet. Následně pak stačí napsat a odladit uživatelský program.

Velkou výhodou programovatelných automatů je jejich schopnost auto-diagnostiky. Často bývají opatřeny vnitřními diagnostickými funkcemi, které průběžně kontrolují činnost systému. Včas dokážou identifikovat závadu, lokalizovat ji a ošetřit.

Hlavní výhodou, kvůli které nahradily pevné reléové obvody je možnost snadné a jednoduché opravy stávajícího uživatelského programu při změně požadavků na aplikaci či při zadávání nových funkcí[1][9].

2.2. Základní rozdělení PLC

1) logické moduly/smart relay
2) mikroPLC
3) kompaktní PLC
4) modulární PLC
5) softPLC
6) PLC pro bezpečné řízení
7) PAC (Programmable Automation Controller)

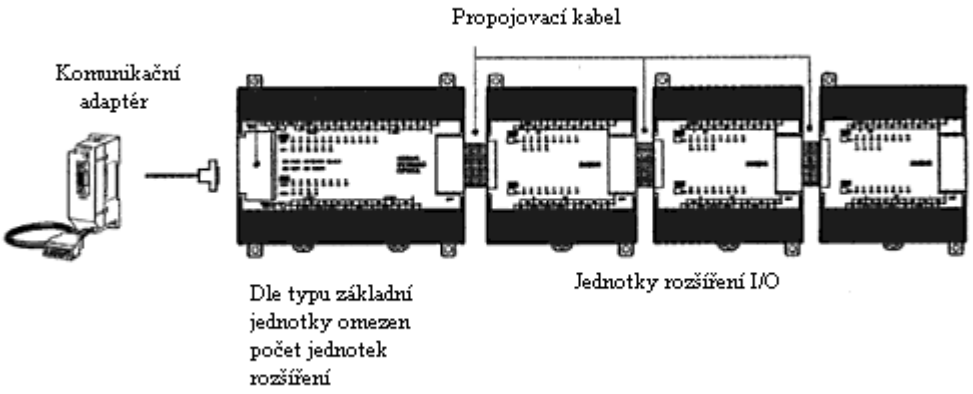
2.2.1. MikroPLC

MikroPLC je nejmenší, nejjednodušší a nejlevnější kompaktní PLC systém určený k řízení nejjednodušších automatizačních aplikací. Pro svoje malé rozměry, kompaktním provedení a nízkou cenou jsou velmi často využívané. MikroPLC má jasně daný počet vstupů a výstupů. Vedle digitálních vstupů a výstupů obsahují některé jednotky také analogové vstupy a výstupy. MikroPLC mají schopnost komunikace a většina z nich se může připojit i na síť. Některá novější mikroPLC se stávají modulárními po vzoru svých předchůdců, PLC. MikroPLC je plně vybaven instrukcemi jak pro integer, neboli celá čísla, tak pro floating-point čili desetinná čísla[1][9].

2.2.2. Kompaktní PLC

Základem kompaktních PLC je blok-modul, obvykle s pevně daným počtem vstupů a výstupů, někdy s rychlými čítači, analogovými vstupy a výstupy, modulem regulátoru a jinými. Má omezenou možnost variabilnosti konfigurace, která je dána použitím vsuvných modulů tzv. piggyback. Zásuvné moduly se nasouvají do patic na

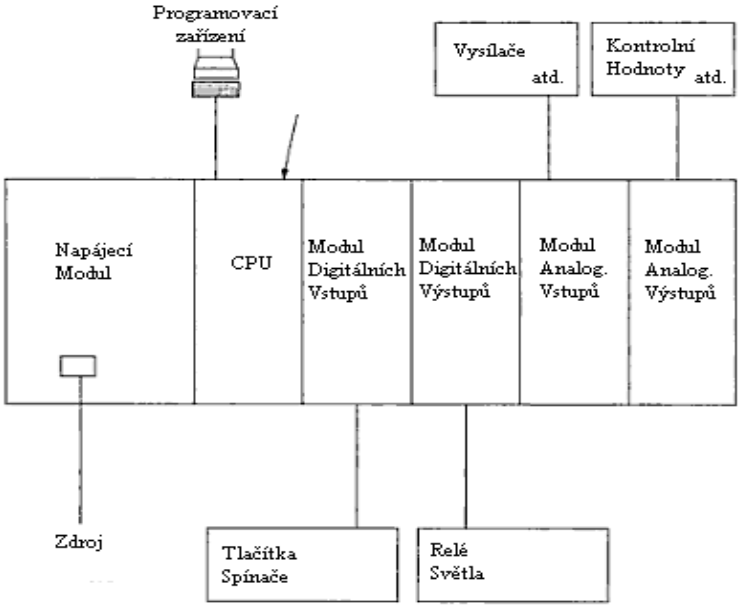
desce plošného spoje. Variabilitu lze také zvýšit použitím přídavných modulů. Kompaktní PLC je levný a používá se pro jednoduché aplikace. [1][5][9].



Obr. 2.2: Kompaktní PLC

2.2.3. Modulární PLC

Modulární PLC je cenově dražší než kompaktní PLC, je vhodný pro náročnější aplikace díky jeho vysoké volnosti konfigurace. Základem je rám, v jehož levé části je zdroj. V rámu je vedena interní sběrnice, na níž jsou konektory pro připojení modulů. Délka rámů závisí na jejich počtu. Vedle zdroje se zasouvá modul procesoru, může jich být i více, a pak následují moduly vstupu a výstupu. K základnímu rámu lze připojit rámy další. Uživatel si může sám sestavit a zapojit do rámu ty jednotky, které pro danou úlohu potřebuje. Může si nastavit tolik vstupů a výstupů kolik potřebuje, zapojit i potřebnou velikost paměti a jiné[1][9].



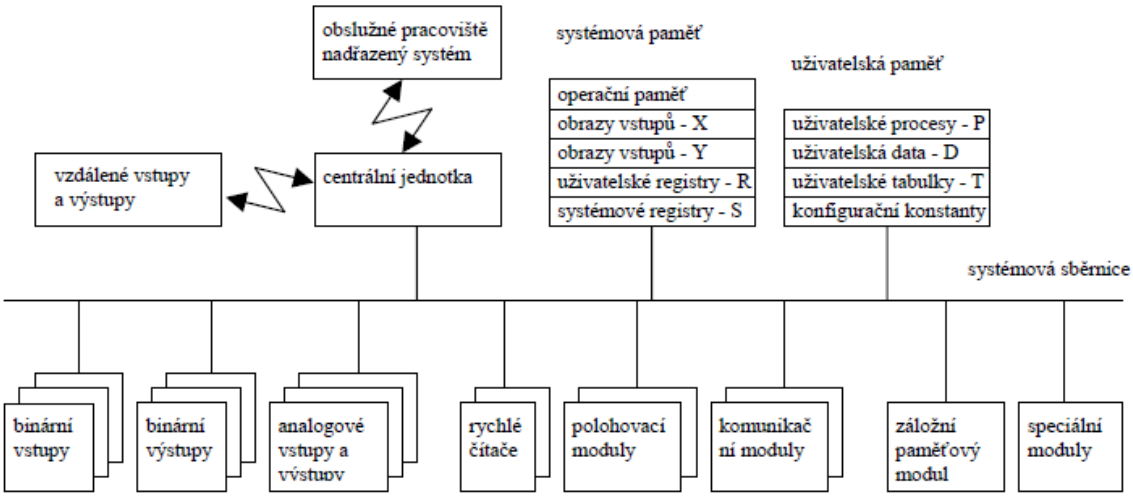
Obr. 2.3: Schéma modulárního PLC

2.2.4. Soft PLC

U Soft PLC se jedná o propojení PLC s průmyslovými počítači, IPC, které umožňují vykonávání logických funkcí sekvenčního řízení v reálním čase. Soft PLC lze s výhodou použít pro náročné řízení rozsáhlých aplikací, pro zpracování a archivaci velkých objemů dat, jinak se pro jeho velkou cenu příliš nepoužívá[6] [9].

2.3. Struktura PLC

Struktura PLC závisí na druhu aplikace, na který se má použít. Níže zobrazené blokové schéma je konfigurací pro náročnější systém.

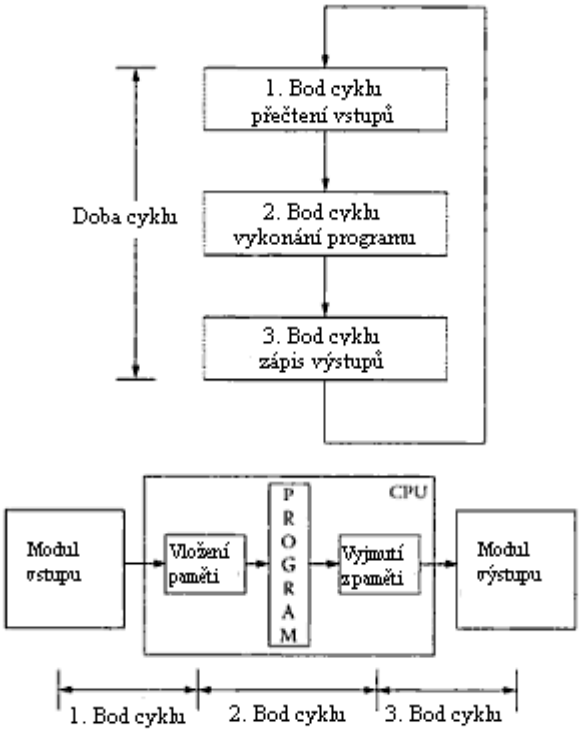


Obr. 2.4: Struktura PLC

2.4. Činnost PLC

Hlavní vlastností PLC, pro kterou se hojně používá a díky které si vydobylo své výsostné postavení v průmyslu, je jeho zpracovávání příkazů v cyklu. PLC je sekvenční zařízení, které plní jeden úkol za druhým. Existují čtyři hlavní body, které PLC vykonává[1] [9]:

1. Nejprve dojde k přečtení vstupů. PLC zkontroluje stav vstupů, zda jsou zapnuty či vypnuty a aktualizuje paměť s jejich současnou hodnotou.
2. Spustí se program. PLC provádí instrukce postupně jeden po druhém a výsledky ukládá do paměti pro jejich pozdější použití.
3. Napíše výstupy. Aktualizuje stav výstupů na základě získaných výsledků v průběhu provádění programu.
4. V části cyklu režie, která proběhne po vykonání třetího bodu, se provádí komunikace po sériových linkách a servisní služby, které zajišťují aktualizaci systémových registrů a časoměrných proměnných časovačů, nulování watchdog. Po vykonání režie se program znovu vrací a vykonává znova všechny příkazy od začátku v tak zvaném cyklu.

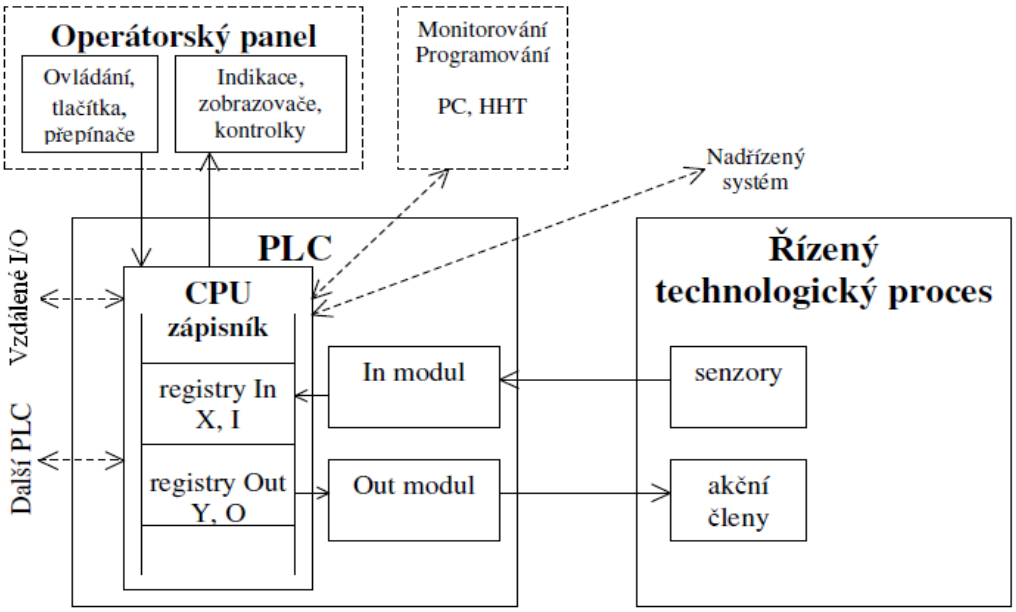


Obr. 2.5: Činnost PLC

2.4.1. Činnost PLC v technologickém procesu

Na obrázku níže je zobrazeno napojení PLC na řízený technologický proces. Do registrů vstupního modulu jsou průběžně zaznamenávány hodnoty ze senzorů. V případě získání hodnot ze senzorů v podobě analogových veličin, převedeme je ve vstupním modulu pomocí A/D převodníkem na digitální veličiny. V přesně daném časovém okamžiku, Input Scan, dochází k přenosu získaných veličin ze vstupního modulu do registrů zápisníkové paměti X,I. S těmito hodnotami program pracuje po celou dobu řešení v tzv. Program Scanu. Výsledky Program Scanu jsou pak průběžně ukládány do registrů zápisníkové paměti a to převážně do Y,O. Po skončení programu se přesunou hodnoty z výstupních registrů zápisníkové paměti do registru výstupního modulu, tomuto procesu se říká Output Scan. Následně se z výstupního modulu průběžně předávají akčním členům. Je-li potřeba, dochází ve výstupní jednotce k opětovnému převedení přes D/A převodník. "Výhodou tohoto řešení je, že vstupy ze senzorů jsou ovzorkovány v určitém časovém okamžiku a program řeší úlohu postupně. V každém okamžiku je tedy možné v průběhu programu přesně definovat aktuální hodnotu právě spočtených výstupů a výstupy posílané na akční členy jsou opět ovzorkovány v určitém časovém okamžiku." Další výhodou je v použití paměti RAM,

díky které jsou operace výrazně rychlejší, než kdyby PLC pracovalo přímo se vstupy a výstupy. K PLC se mohou připojit i ovládací prvky např. tlačítka, přepínače atd. a indikační prvky jako jsou kontrolky, sirény, zobrazovače, které mohou být spojeny do operátorského panelu. Také je tu možnost propojení PLC s PC. Toto propojení umožní programování a monitorování aplikace[9].



Obr. 2.6: Činnost PLC v technologickém procesu

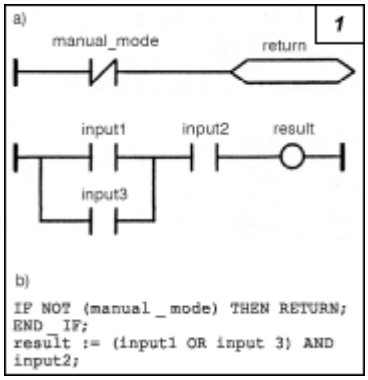
2.5. Programovací jazyky pro PLC

PLC má dvě formy programovacích jazyků, textové a grafické. Jedná se o specializované jazyky navržené pro snadnou, názornou a účinnou realizaci logických funkcí. Každý výrobce má svoje programovací prostředí. Jsou si často podobné, ale nejsou navzájem přenositelné. Mezi grafické jazyky patří reléové schéma LD, jazyk funkčních bloků FBD a také tam bývá někdy řazen jazyk sekvenčního programování SFC, který však není zařazen v normě IEC 61131-3. Mezi textové jazyky patří jazyk mnemokódů IL a strukturovaný text ST.

2.5.1. Ladder Diagram

Jazyk reléového schéma neboli Ladder Diagram je založen na grafické reprezentaci reléové logiky. Je také někdy označován jako jazyk kontaktních schémat. „Síť v jazyku LD je zleva i zprava ohraničena svislými čarami, které se nazývají levá a pravá napájecí sběrnice. Mezi nimi je tzv. příčka, která může být rozvětvena. Každý úsek příčky, vodorovný nebo svislý, může být ve stavu on nebo off. Do příček mohou

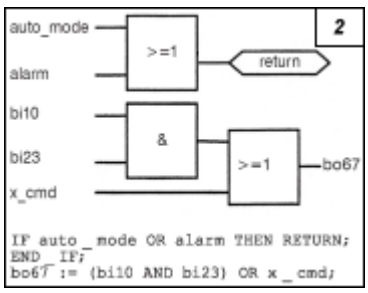
být včleněny kontakty (spínací, rozpínací apod.), cívky a dále funkce a funkční bloky." Jazyk kontaktních schémat se s výhodou používá u nejjednodušších logických operací a také je ho výhodné používat v případech, kdy s ním pracuje personál bez znalostí klasického programování. Velkou výhodou je rychlý servis. Při možnosti zvýraznění "vodivé cesty" je nalezení závady na stroji jednoduchou věcí. Vyskytují-li se v programu složitější instrukce, jako například aritmetické instrukce nebo logické s vektorovými operandy, pak použití tohoto jazyka postrádá svoji názornost[1][12] [13].



Obr. 2.7: Ukázka Ladder programu

2.5.2. Jazyk funkčních bloků

Jazyk funkčních bloků, Function Block Diagram, je dalším grafickým jazykem. „Vyjadřuje chování funkcí, funkčních bloků a programů jako soubor vzájemně provázaných grafických bloků podobně jako v elektronických obvodových diagramech. Jde o systém prvků, které zpracovávají signály. Často se zde používají standardní funkční bloky, například prvky pro detekci náběžné a sestupné hrany, čítače, časovače a komunikační bloky." Závisí pak na dané firmě, jaké další soubory bloků nabídnou ve svém programovacím prostředí, od spínacích hodin týdenních, ročních, až po generátory impulzů či komparátory. Je výhodný zejména pro personál zvyklý na kreslení logických schémat pro zařízení s integrovanými obvody[1][12][13].



Obr. 2.8: Ukázka jazyka funkčních bloků

2.5.3. Jazyk strukturovaného textu

Jazyk strukturovaného textu, Structured Text, je vyšší programovací jazyk navržený pro automatické procesy. Tento jazyk se převážně používá u složitých postupů, které nelze jednoduše vyjádřit pomocí grafických jazyků. Umožňuje úsporný a názorný zápis algoritmů. Je oblíben zejména u absolventů odborných škol. Základem je seznam strukturovaných příkazů. Každý příkaz končí středníkem. Základní typy příkazů strukturovaného textu jsou[1][12]:

- přiřazení příkazu (proměnná:=výraz)
- podprogram nebo volání funkce
- volání funkčního bloku
- příkazy výběru (IF,THEN,ELSE,CASE ...)
- příkazy opakování (FOR,WHILE,REPEAT ...)
- příkazy řízení (RETURN,EXIT ...)
- speciální příkazy pro spojení s jinými jazyky například SFC

2.5.4. Jazyk mnemokódů

Jazyk mnemokódů je nižší programovací jazyk. Používá se zejména pro menší aplikace nebo pro optimalizaci. Instrukční jazyk je seznam instrukcí. Každá instrukce musí začínat na novém řádku, musí obsahovat operátor, a když je potřeba pro speciální operace jednoho či více operandů, dělíme je čárkou. Instrukce se vždy vztahují k současnému výsledku. Má podobnosti v assembleru, například symbolické označení návěští pro cíle skoků a volání, symbolická jména pro číselné hodnoty, pro pojmenování vstupních, výstupních a vnitřních proměnných a jiných objektů programu a další[1][12].

```
Label: LD      in2                                          3
       ANA             (* result := ana (in2) *)
       MUL     2       (* result := 2*ana (in2) *)
       ST      temp    (* temp := result *)
       LD      in1
       ANA
       ADD     temp    (* result := 2*ana (in2) + ana (in1) *)
       MUL     2       (* 4*ana (in2) + 2*ana (in1) *)
       ST      temp    (* temp := result *)
       LD      in0
       ANA
       ADD     temp    (* 4*ana (in2) + 2*ana (in1) + ana(in0) *)
       ST      SUBPRO  (* return the current result *)
                       (* to the calling program *)
```

Obr. 2.9: Ukázka jazyka mnemokódů

3. Popis a charakteristika platformy Yaskawa MP2300

MP2300 je kompaktní kontrolér, který je složen z napájecího zdroje, CPU, I/O a komunikačních funkcí, a to vše se nachází v jednom systému. MP2300 je tvořen ze základního, tzv. Basic Modul, a přídavného volitelného modulu, tzv. Optional Modul. Základní modul vykonává řízení pohybů a sekvenční řízení, zatímco volitelný modul zajišťuje práci se vstupy a výstupy a komunikační funkce. MP2300S je menší než MP2300. Jedná se o malý kompaktní kontrolér s MECHATROLINK a s Ethernet portem. Rozdíl mezi MP2300 a MP2300S je, že k MP2300S lze připojit jeden, zatímco k MP2300 lze připojit až tři přídavné volitelné moduly, jinak MP2300S podporuje stejné funkce jako MP2300[3][4].

MP2300 nabízí následující vlastnosti:

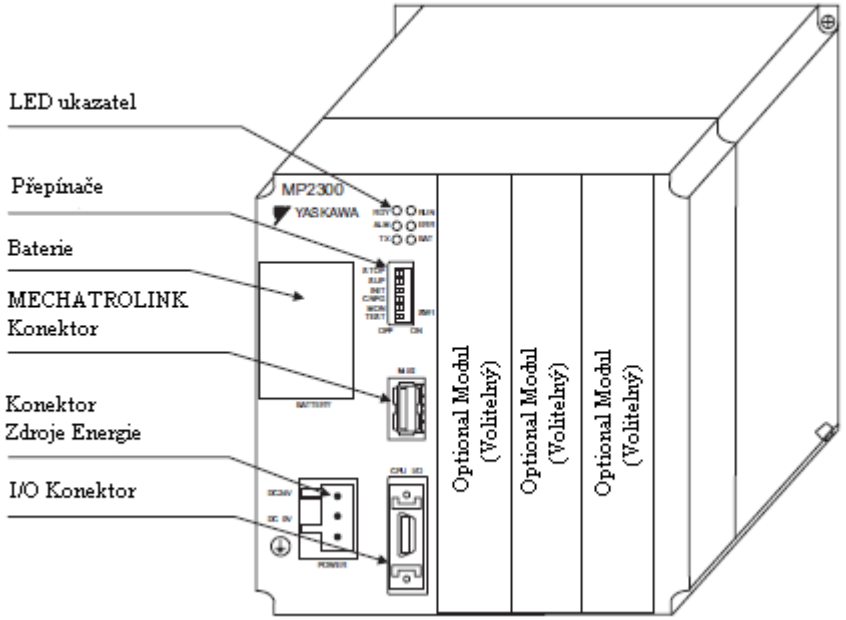
1. flexibilita

   K MP2300 lze přidávat volitelné moduly tak, aby uživatel získal optimální sestavu pro daný případ.
2. vysoký výkon
3. snadné ovládání

   Může se použít funkce samo-nastavení, která automaticky detekuje zařízení připojené k MECHATROLINK a nastaví potřebné parametry. Tím se čas seřízení značně zredukuje. MP2300 obsahuje aplikační program, který využívá v databance nashromážděné specifické znalosti z předchozího software k vylepšení následného systému[4].

3.1. Základní Modul (Basic)

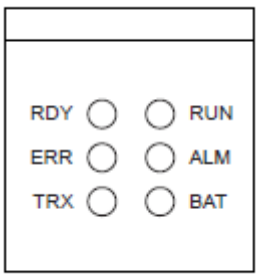
Vnější vzhled základního modulu:



Obr. 3.1: Základní modul

Indikátory:

LED ukazatelé neboli indikátory informují uživatele o provozním stavu základního modulu a podává informace o chybě.



Obr. 3.2: Indikátory základního modulu

RDY svítí zeleně, probíhá-li operace tak jak má.

RUN svítí zeleně během provádění uživatelského programu.

ALM bliká červeně, objeví-li se chyba.

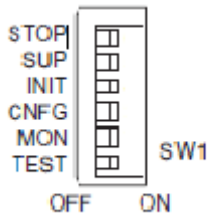
ERR bliká červeně, pokud dojde k poruše.

TX svítí během přenosu dat přes MECHATROLINK I/II.

BAT svítí červeně, dojde-li k poklesu energie.

Přepínače:

Přepínače nastavují operační podmínky pro základní modul, když je napájení zapnuté.



Obr. 3.3: Přepínače základního modulu

Jméno	Nastavení	Operační mód	Výchozí nastavení	Detaily
STOP	ON	Uživatelský program zastaven	OFF	Zastaví vykonávání uživatelského programu. Dostupný pouze v módu ON.
	OFF	Uživatelský program spuštěn		
SUP	ON	Využívá systém	OFF	Vždy nechat nastaveno na OFF.
	OFF	Normální operace		
INIT	ON	Vymazání paměti	OFF	Při nastavení ON dojde k vymazání paměti. Při nastavení OFF pak k vykonání programu, který je uložen ve flash paměti.
	OFF	Normální operace		
CNFG	ON	Mód konfigurace	OFF	Při nastavení ON dojde k vykonání samonastavení připojených zařízeních.
	OFF	Normální operace		
MON	ON	Využívá systém	OFF	Vždy nechat nastaveno na OFF.
	OFF	Normální operace		
TEST	ON	Využívá systém	OFF	Vždy nechat nastaveno na OFF.
	OFF	Normální operace		

Tab. 1: Tabulka významu zkratek přepínače základního modulu

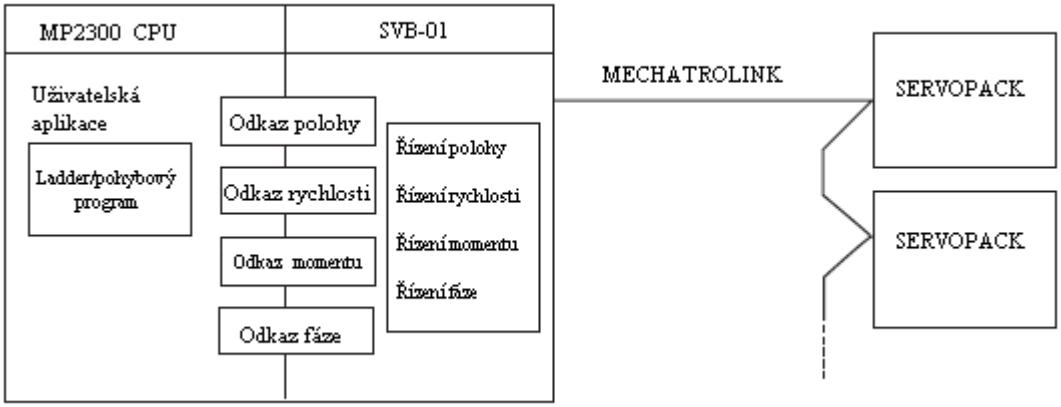
3.2. Volitelný Modul (Optional)

Volitelné moduly jsou:

1. Pohybový Modul
2. Vstupní/Výstupní Modul
3. Komunikační Modul

3.2.1. Pohybový Modul

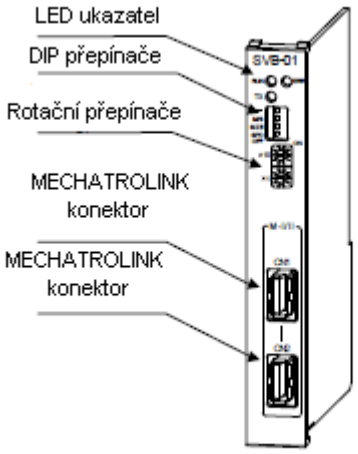
Ke kontroléru MP2300 se mohou připojit pohybové moduly SVB-01 a SVA-01. SVB-01 je kompatibilní s rozhraním MECHATROLINK-II. Použití MECHATROLINK umožňuje ovládání více os s redukovaným zapojením. MECHATROLINK-II reguluje polohu, otáčky, točivý moment, fáze a přesnou synchronizaci. K pohybovému modulu lze připojit až 21 podřízených stanic a kontrolovat až 16 os serva[3][4].



Obr. 3.4: Schéma propojení MP2300 s pohybovým modulem SVB-01

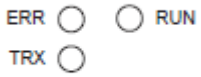
.

Vnější vzhled pohybového modulu SVB-01:



Obr. 3.5: Pohybový modul SVB-01

Indikátory:



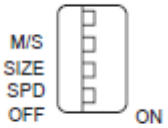
Obr. 3.6: Indikátory pohybového modulu SVB-01

ERR bliká červeně v případě poruchy.

RUN svítí zeleně, probíhá-li operace normálně.

TRX svítí zeleně, dochází-li k přenosu dat přes MECHATROLINK.

Přepínače:

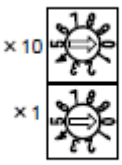


Obr. 3.7: Přepínače pohybového modulu SVB-01

Název	Status	Operační mód	Výchozí nastavení	Detaily
M/S	ON	Slave Mód	OFF	Slouží k přepnutí módu ovládající/ovládaný (Master/Slave)
	OFF	Master Mód		
SIZE	ON	17 Bytes	OFF	Volí počet odeslaných bytů
	OFF	32 Bytes		
SPD	ON	4 Mbps	OFF	Slouží k navolení přenosové rychlosti
	OFF	10Mbps		

Tab. 2: Tabulka významu zkratek přepínače pohybového modulu SVB-01

Rotační přepínače:



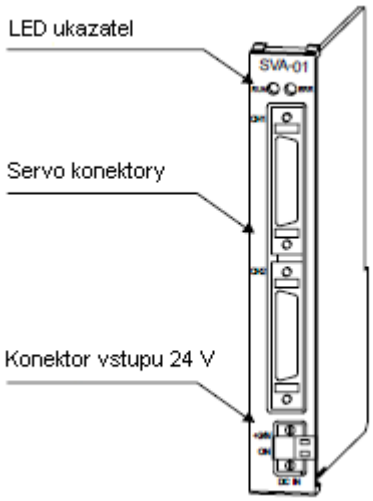
Obr. 3.8: Rotační přepínače pohybového modulu SVB-01

Označení	Rozsah	Operační mód	Výchozí nastavení	Detaily
x10	0 až 9	Lokální adresa stanice v módu slave	0	Nastavuje lokální adresu v módu slave (po desítkách)
x1	0 až 9	Lokální adresa stanice v módu slave	1	Nastavuje lokální adresu v módu slave (po jednotkách)

Tab. 3: Tabulka významu označení rotačního přepínače pohybového modulu SVB-01

SBA-01 je pohyb řídící modul s analogovými výstupy. Každý modul může řídit serva nebo převodníky až ve dvou osách. Má dva konektory pro připojení SERVOPACKU a externích vstupů/výstupů. Každý konektor poskytuje analogové výstupy pro odkaz rychlosti a momentu, analogové vstupy pro monitorování zpětnovazebné rychlosti a momentu, vstupy pulzní fáze A, B a C a pro všeobecné použití digitálních vstupů a výstupů[4].

Vnější vzhled pohybového modulu SBA-01:



Obr. 3.9: Pohybový modul SBA-01

Indikátory:

Obr. 3.10: Indikátory pohybového modulu SBA-01

ERR bliká červeně v případě poruchy.

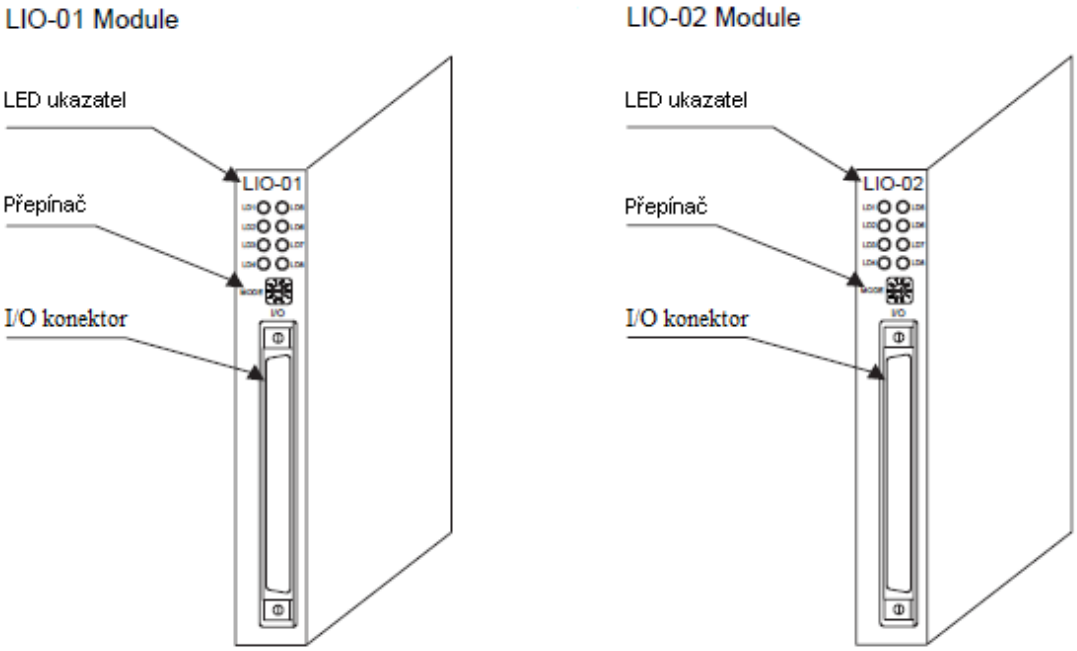
RUN svítí zeleně, probíhá-li operace normálně.

3.2.2. Vstupní/Výstupní Modul

Mezi Vstupní/Výstupní Moduly, které mohou být připojeny ke kontroléru MP2300 patří LIO-01, LIO-02, LIO-04, LIO-05, DO-01 a AI-01 modul. Blíže se seznámíme s moduly LIO-01 a LIO-02.

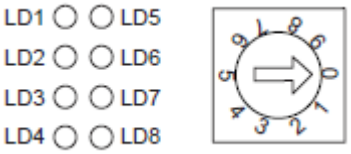
Moduly LIO-01 a LIO-02 mají digitální vstupy, výstupy a pulzní čítač. K dispozici je 16 digitálních vstupů a 16 digitálních výstupů pro digitální vstupní a výstupní funkce a také pulzní vstupní kanál pro pulzní čítač. Vstupní a výstupní hodnoty pro digitální vstupy, výstupy a pulzní čítač jsou obnoveny v pevně daném cyklu probíhajícím v každém vysokorychlostním a nízkorychlostním scanu. Rozdíl mezi moduly LIO-01 a LIO-02 je v použití typu digitálního výstupu. LIO-01 nabízí klesající digitální mód výstupu, který poskytuje zem, a LIO-02 zdrojový digitální mód výstupu, jenž dává zdroj napětí. Moduly LIO-04 a LIO-05 jsou vybaveny 32 digitálními vstupy a 32 digitálními výstupy. U modulu DO-01 můžete využít 64 digitálních vstupů[4].

Vnější vzhled Vstupního/Výstupního modulu LIO-01 a LIO-02 :



Obr. 3.11: Vstupní/Výstupní moduly

Indikátory a přepínač:



Obr. 3.12: Indikátory a přepínač

Přepínač	LED číslo	Stav, kdy svítí	LED číslo	Stav, kdy svítí
0 (základní stavový indikátor)	LD1	Normální stav (Chyba, když nesvítí).	LD5	Normální stav (Chyba, když nesvítí).
	LD2	Jeden ze vstupů DI_00/DI_07 zapnut.	LD6	Jeden ze vstupů DI_08/DI_15 zapnut.
	LD3	Jeden z výstupů DO_00/DO_07 zapnut.	LD7	Jeden z výstupů DO_08/DO_15 zapnut.
	LD4	Vstup pulzů A/B. Fáze A/B je zapnuta.	LD8	Vstup fáze Z. Fáze Z je zapnuta.
1 (DI vstupní indikátor 1)	LD1	Vstup DI_00 je zapnut.	LD5	Vstup DI_04 je zapnut.
	LD2	Vstup DI_01 je zapnut.	LD6	Vstup DI_05 je zapnut.
	LD3	Vstup DI_02 je zapnut.	LD7	Vstup DI_06 je zapnut.
	LD4	Vstup DI_03 je zapnut.	LD8	Vstup DI_07 je zapnut.
2 (DI vstupní indikátor 2)	LD1	Vstup DI_08 je zapnut.	LD5	Vstup DI_12 je zapnut.
	LD2	Vstup DI_09 je zapnut.	LD6	Vstup DI_13 je zapnut.
	LD3	Vstup DI_10 je zapnut.	LD7	Vstup DI_14 je zapnut.
	LD4	Vstup DI_11 je zapnut.	LD8	Vstup DI_15 je zapnut.
3 (DO výstupní indikátor 1)	LD1	Výstup DO_00 je zapnut.	LD5	Výstup DO_04 je zapnut.
	LD2	Výstup DO_01 je zapnut.	LD6	Výstup DO_05 je zapnut.
	LD3	Výstup DO_02 je zapnut.	LD7	Výstup DO_06 je zapnut.
	LD4	Výstup DO_03 je zapnut.	LD8	Výstup DO_07 je zapnut.
4 (DO výstupní indikátor 2)	LD1	Výstup DO_08 je zapnut.	LD5	Výstup DO_12 je zapnut.
	LD2	Výstup DO_09 je zapnut.	LD6	Výstup DO_13 je zapnut.
	LD3	Výstup DO_10 je zapnut.	LD7	Výstup DO_14 je zapnut.
	LD4	Výstup DO_11 je zapnut.	LD8	Výstup DO_15 je zapnut.
5 (PI vstupní indikátor)	LD1	Vstup pulzu A.	LD5	Detekce shody.
	LD2	Vstup pulzu B.	LD6	Fáze F přerušení.
	LD3	Vstup pulzu Z.	LD7	DI přerušení.
	LD4	-----	LD8	-----

Tab. 4: Tabulka významu kombinace přepínače a LED diody

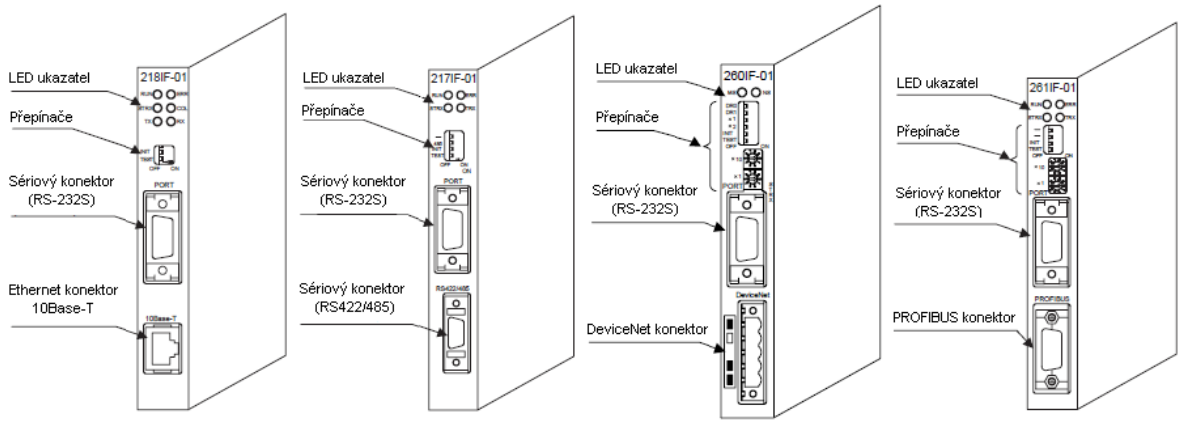
3.2.3. Komunikační Modul

Mezi Komunikační Moduly, které mohou být připojeny ke kontroléru MP2300 patří Moduly 218IF-01, 217IF-01, 260IF-01 a 261IF-01.

218IF-01 Modul má sériové komunikační rozhraní RS-232C a Ethernet rozhraní. Součástí Modulu 217IF-01 jsou pro změnu sériová rozhraní RS-232C a RS422/485, Modul 260IF-01 obsahuje port RS-232C a konektor DeviceNet a Modul

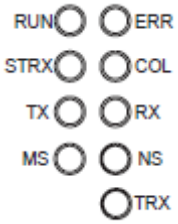
261IF-01 rozhraní RS-232C a PROFIBUS. Na všechny výše zmíněné Komunikační Moduly lze napojit běžné osobní počítače, zařízení HMI a kontroléry vyráběné jinými společnostmi[4].

Vnější vzhled komunikačních modulů 218IF-01, 217IF-01, 260IF-01 a 261IF-01:



Obr. 3.13: Komunikační moduly

Indikátory:



Obr. 3.14: Indikátory komunikačních modulů

ERR bliká červeně v případě poruchy.

RUN svítí zeleně, probíhá-li operace normálně.

STRX svítí zeleně v případě přenosu dat RS-232C portem.

COL svítí červeně při výskytu kolize v Ethernetu.

TX, RX svítí zeleně při vysílání a přijímání dat Ethernetem.

TRX svítí zeleně, vysílají-li se data přes port RS422/485, nebo se vysílají a přijímají pomocí PROFIBUS.

MS a NS indikátory mají dvě barvy pro upozornění svého stavu. Svítí-li indikátor zeleně, operace probíhá normálně.

MS Svítí-li červeně, modul ohlašuje chybu, nesvítí-li, napájení bylo odpojeno.

NS Svítí-li červeně sběrnice je buď vypnutá, nebo došlo ke zdvojení MAC adres. Bliká-li červeně, vyskytla se chyba v komunikaci. Nesvítí-li, napájení komunikace bylo odpojeno, zkontrolujte zdvojení MAC adres. Bliká-li zeleně připojení vstupů a výstupů je nestabilní.

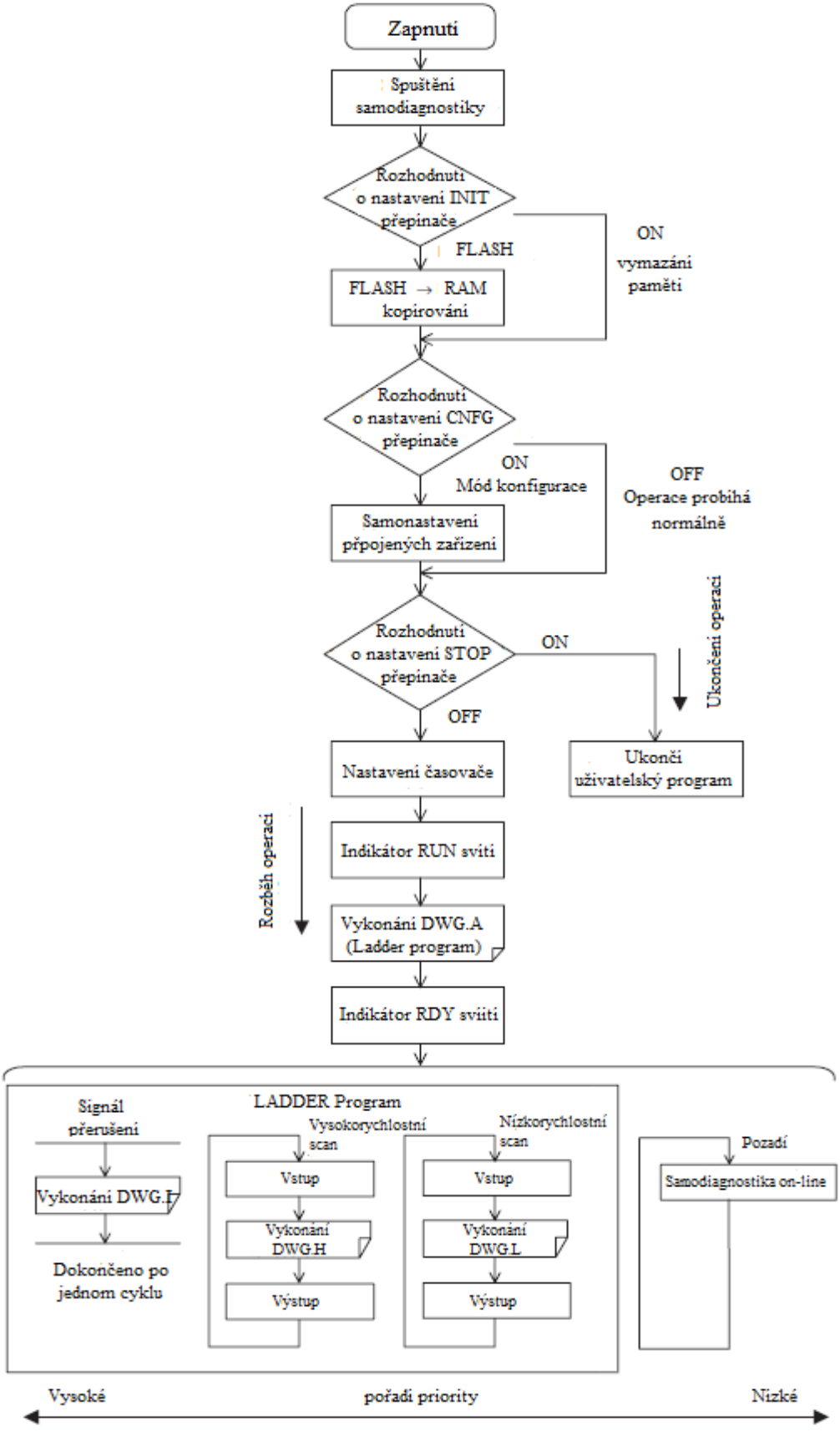
3.3. Posloupnost operací při startování PLC

Při spuštění PLC dojde k dané posloupnosti operací, provedení diagnostiky a rozsvícení příslušných indikátorů v závislosti na nastavení přepínače na základním modulu.

Po spuštění MP2300 proběhne diagnostika na následujících zařízeních

- Zápis a čtení paměti (RAM)
- Program systému (ROM)
- Funkce hlavního procesoru (CPU)
- Funkce koprocesoru určený pro operaci s čísly s plovoucí desetinnou čárkou (FPU)

Funkce autoregulace automaticky rozpozná připojené volitelné zařízení a vytvoří soubor definic. Během vykonávání automatického nastavování bliká zeleně RUN indikátor. Při nastavení přepínače STOP na hodnotu OFF, CPU spustí časovač a vykoná jako první scan DWG.A z Ladder programu. DWG.A se provede jen jednou, dokud není dokončen. Systém vstupů a výstupů se uskuteční z prvního scanu. MP2300 ukončí činnost řízení pohybů v případě, když je STOP na přepínači nastaven na ON. On-line diagnostika se provádí v programu systému, funkci hlavního procesoru a ve funkci FPU[4].



Obr. 3.15: Posloupnost operací při startování PLC

3.4. Uživatelský program

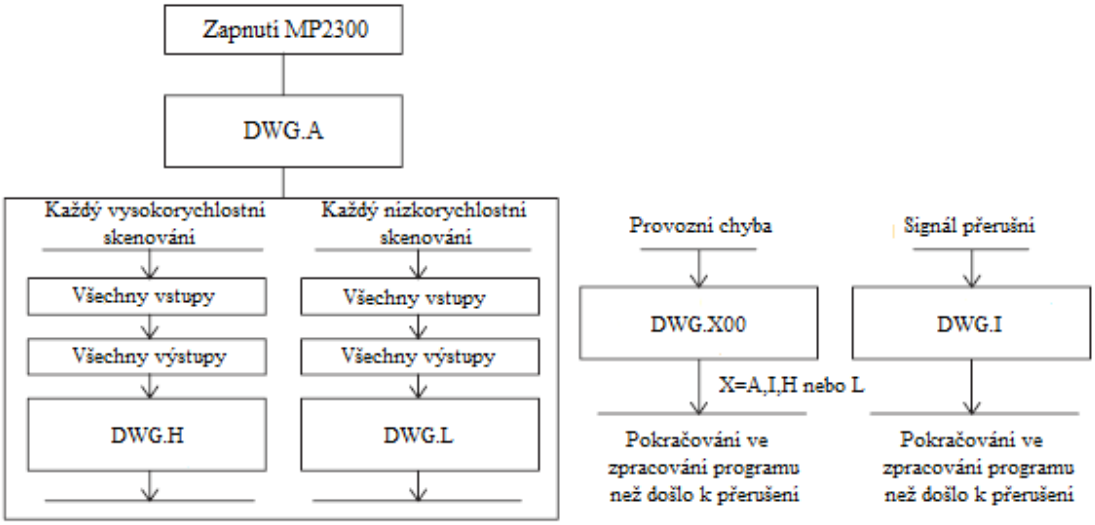
Uživatelský program, který slouží k řízení strojů pomocí MP2300 je ve formě Ladder programu. Základem Ladder programu jsou jednotlivé očíslované ladder výkresy tzv. ladder drawings. Ladder výkresy se dělí hierarchicky na rodičovské, tzv. Parent Drawings, podprogram k rodičovskému tzv. Child Drawings, dále podprogram k Parent Drawing a Child Drawings tzv. Grandchild Drawings a výkres provozní chyby při zpracování. Parent Drawings jsou při splněných podmínkách automaticky prováděny systémovým programem. Child Drawings jsou přístupné pomocí speciálního příkazu z Parent Drawings, zatímco Grandchild Drawings z Child Drawings. Výkresy provozní chyby při zpracování jsou automaticky prováděny systémem programu v případě, vyskytne-li se chyba při operaci[4].

Typ výkresu rodičovského	Funkce	Priorita	Podmínky uskutečnění	Max. počet výkresů
DWG.A	Proces spuštění.	1	Proces se uskuteční jen v případě spuštění MP2300.	64
DWG.I	Proces přerušení.	2	V případě výskytu vnějšího přerušovacího signálu.	64
DWG.H	Vysokorychlostní proces skenování	3	Při spuštění plánovaného cyklu (vykonán každý vysokorychlostní scan)	200
DWG.L	Nízkorychlostní proces skenování	4	Při spuštění plánovaného cyklu (vykonán každý nízkorychlostní scan)	500

Tab. 5: Popsání jednotlivých typů rodičovských výkresů

Výkresy se klasifikují podle jejich prvních písmen na základě účelu zpracování. Nejvyšší prioritu má výkres pod označením DWG.A. Maximální počet těchto výkresů je 64. Je vykonán pouze na začátku spuštění MP2300. O stupeň nižší prioritu má výkres DWG.I, proces přerušení. Podmínkou aktivace je signál vnějšího přerušení. Maximální počet DWG.I je stejný jako u DWG.A, a to 64. Nižší prioritu má DWG.H, neboli vysokorychlostní proces skenování, jenž se vykoná po spuštění cyklu. Možný počet těchto výkresů je 200. Stejnou podmínku pro vykonání má výkres s nejnižší prioritou DWG.L, neboli nízkorychlostní proces skenování s maximálním počtem výkresů až 500.

3.4.1. Pořadí priority výkresů

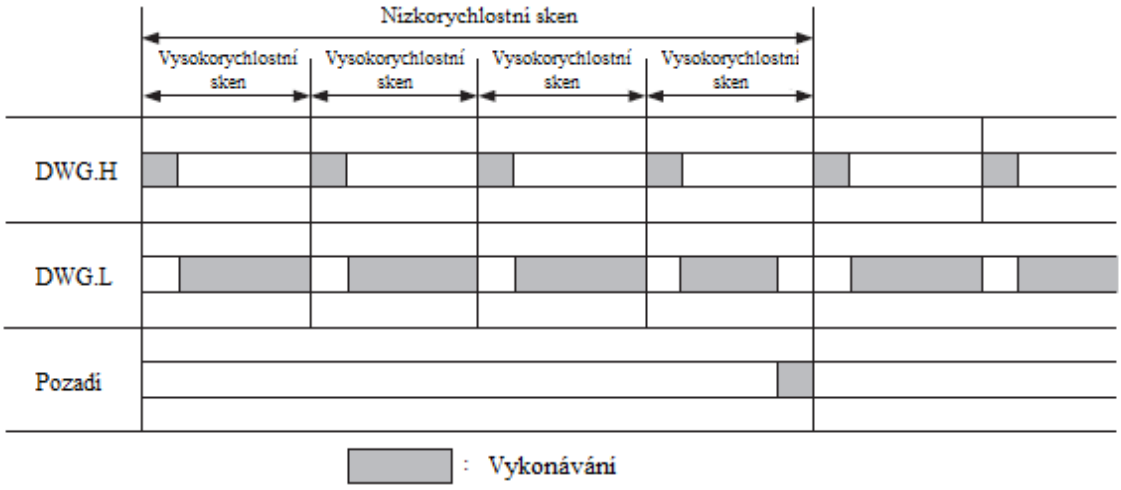


Obr. 3.16: Priorita výkresů

Obr. 3.16 znázorňuje pořadí priority ve vykonávání jednotlivých výkresů. Po zapnutí MP2300 se jako první vykoná výkres DWG.A, jenž se pak již dále nevykonává. Pak začne provádění jednotlivých příkazů z vysokorychlostního scanu a následně z nízkorychlostního scanu. Vyskytne-li se přerušovací signál či provozní chyba, prováděný scan se přeruší, vykoná se příslušný proces přerušení nebo proces při výskytu operační chyby a pak se pokračuje ve zpracovávání programu před přerušením.

3.4.2. Zpracování scanů

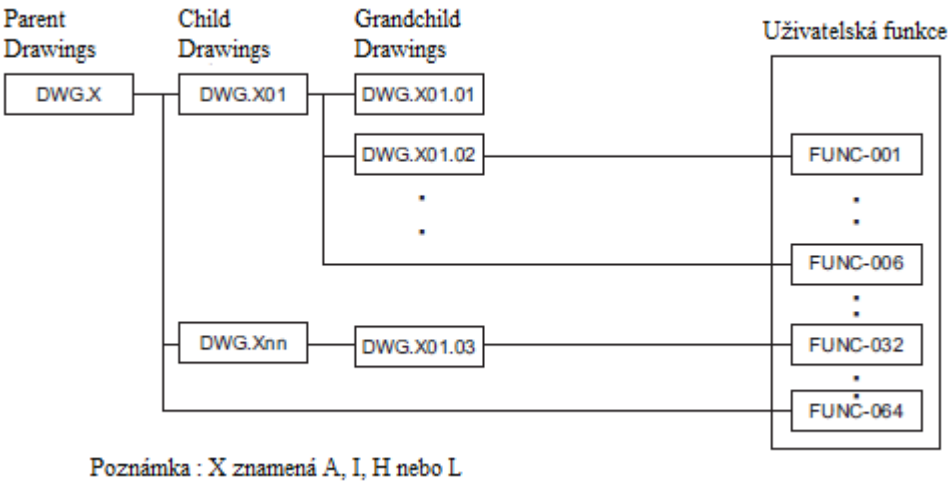
Zpracování jednotlivých scanů neprobíhá současně, zpracovávají se na základě priority a času sdílení. Z obrázku je patrno, že se nejprve vykoná vysokorychlostní scan a ve zbytku času se provede nízkorychlostní scan. Zpracování na pozadí se používá pro vykonání interního zpracování systému jako je například proces komunikace[4].



Obr. 3.17: Zpracování scanů

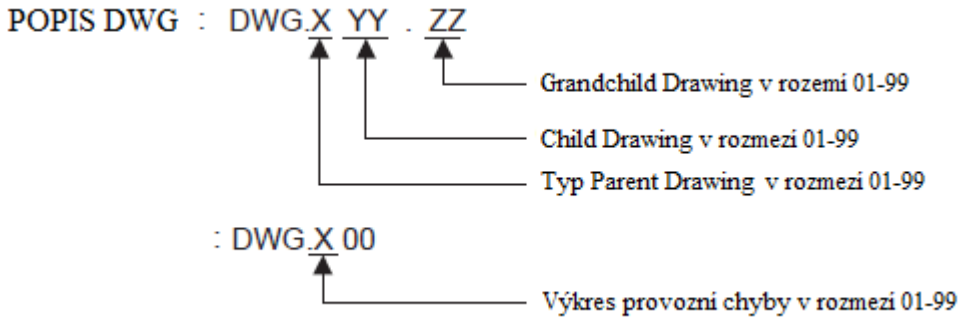
3.4.3. Hierarchie struktury výkresů

Každý program je složen z rodičovského výkresu, tzv. Parent Drawings, podprogramu k rodičovskému tzv. Child Drawings, dále podprogramu k Parent Drawing a Child Drawings tzv. Grandchild Drawings. Parent Drawings nemůže volat Child Drawings různých typů, tak jako Child Drawings nemůže volat Grandchild Drawings různých typů. Stejně tak Parent Drawing nemůže přímo volat Grandchild Drawings. Child Drawings jsou vždy volány Parents Drawings a Grandchild Drawings zase Child Drawings[4].



Obr. 3.18: Struktura výkresů

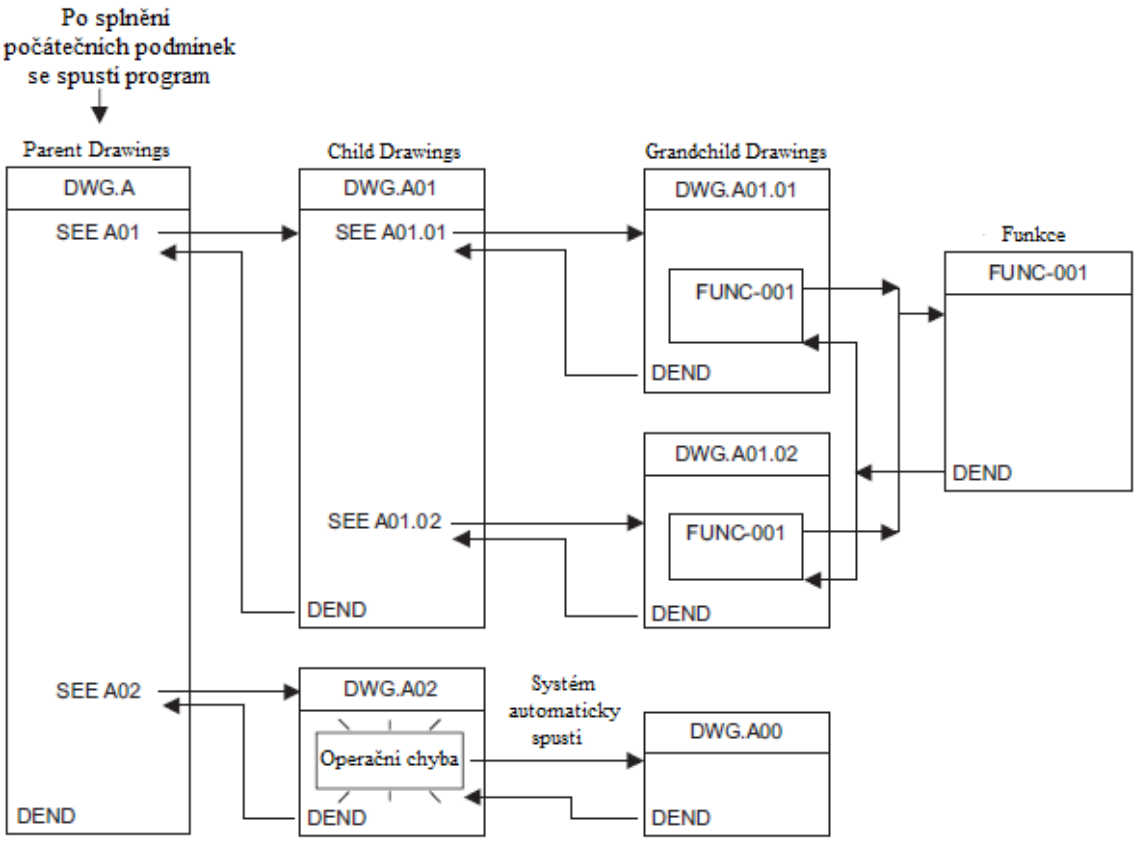
Typ a vzájemný vztah výkresů lze popsat dle následujícího principu:



Obr. 3.19: Princip názvů výkresů

3.4.4. Metoda zpracování výkresů

Metoda zpracování výkresů je založena na volání nižších úrovní výkresů z vyšších úrovní. Pro představu fungování této metody je níže na Obr. 3.20 zobrazen postup provedení zpracování výkresů, kde je jen pro příklad použit typ DWG.A. Funkce lze volat z jakékoliv kresby, může být volána i z jiných funkcí.



Obr. 3.20: Systém zpracování výkresů

3.5. Pohybový program

Pohybové programy se dělí na dva typy. Prvním typem je hlavní program a druhým typem je podprogram. Celkový počet pohybových programů může být vytvořeno nezávisle na výkresech až 256. Hlavní program je přístupný z DWG.H a je značen MPM, kde následuje číslo od 1 až 256, zatímco podprogram pouze z hlavního programu a je označen MPS 1 až 256. MP2300 může současně vykonávat až 16 pohybových programů. Existují dvě metody pro jejich specifikaci. První je přímá specifikace pohybového programu a druhá nepřímá, specifikovaná dle registračního čísla, kde je dané číslo programu uloženo. Skupina os související s danou operací může být zařazena jako jedna skupina pohybového programu a program pak může být vykonáván pro každou skupinu. To umožňuje jednomu MP2300 nezávisle kontrolovat více strojů použitím operačních skupin. Pohybové programy jsou vždy volány z parent, child, či grandchild DWG.H výkresů použitím příkazu MSEE v hierarchickém pořadí parent, child a grandchild v každém cyklu vysokorychlostního scanu, avšak pohybové programy nemohou být vykonány v jednom scanu. Pro tento případ je pohybový program řízen a kontrolován speciální systémovou funkcí. Ten samý pohybový program či podprogram může být volán pouze jednou v jednom scanu[4].

3.6. Funkce

Funkce jsou vykonány jejich voláním z každého typu výkresu použitím příkazu FSTART. Ta samá funkce může být volána v ten samý čas z odlišného typu výkresu či úrovně, taktéž je lze volat z jiných funkcí. Použití funkcí má přínos jednak v jednodušším vytváření komponent uživatelského programu a také v jeho snazším psaní a údržbě. Funkce mohou být dvojího typu. Standardní systémové funkce, které již v systému existují, a uživatel je nemůže změnit. Uživatelské funkce, které uživatel sám definuje[4].

Přehled systémových funkcí:

Typ	Název	Symbol	Obsah
Systémové funkce	Čítač	COUNTER	Přírůstkový/Sestupní čítač.
	První vstup/První výstup	FINFOUT	První vstup/výstup.
	Dráhová funkce	TRACE	Kontrola vykonání dat dráhy.
	Čtečka dat dráhy	DTRC-RD	Čte data z paměti pro data dráhy do uživatelské paměti.
	Čtení měniče trasování	ITRC-RD	Čte data z paměti měniče trasování do uživatelské paměti.
	Odesílač zpráv	MSG-SND	Odesílá zprávy na externí komunikační zařízení.
	Přijímač zpráv	MSG-RCV	Přijímá zprávy z externích komunikačních zařízení.

Tab. 6: Popis systémových funkcí

3.7. Registry

3.7.1. Registry výkresů

Registry slouží k ukládání dat. V Tab. 7 je výpis existujících typů registrů výkresů a jejich použití. Kde n značí číslo v decimální soustavě a h v hexadecimální soustavě. B, W, L, F a A jsou datové typy, kde B znázorňuje bit nabývající pouze dvou hodnot 0 nebo 1, I integer neboli celé číslo, L je double integer, jenž stejně jako integer znázorňuje celé číslo, ale má větší rozsah, F reálné číslo a A je adresa.

Typ	Název	Specifikace	Rozsah	Detaily	Charakteristiky
S	Systémový registr	SB, SW, SL, SFnnnnn (Sannnnn)	SW00000 až SW08191	Registry poskytované systémem. Registry SW00000 až SW00049 jsou při nastartování systému vynulovány.	Společné pro všechny výkresy
M	Datový registr	MB, MW, ML, MFnnnnn (MAnnnnn)	MW00000 až MW65534	Registry sdílené všemi výkresy. Používané například jako rozhraní mezi výkresy.	
I	Vstupní registr	IB, IW, IL, IFhhhh (IAhhhh)	IW0000 až IW13FFF	Registry využívané pro vstupní data.	
O	Výstupní registr	OB, OW, OL, OFhhhh (OAhhhh)	OW0000 až OW13FFF	Registry využívané pro výstupní data.	
C	Registr konstant	CB, CW, CL, CFnnnnn (CAnnnnn)	CW00000 až CW16383	Registry, které mohou být volány jen z programů.	
#	# registr	#B, #W, #L, #Fnnnnn (#Annnnn)	#W00000 až #W16383	Jen volané registry. Mohou být volané pouze příslušným výkresem. Rozsah použití nastavuje uživatel v MPE720.	Jedinečné pro každý výkres
D*	D registr	DB, DW, DL, DFnnnnn (DAnnnnn)	DW00000 až DW16383	Vnitřní registry jedinečné pro každý výkres. Mohou být použity jen odpovídajícím výkresem. Rozsah použití nastavuje uživatel.	

Tab. 7: Typy registrů výkresů a jejich popis

3.7.2. Registry funkcí

V Tab. 8 je seznam registrů, které mohou být použity s každou funkcí.

Typ	Název	Specifikace	Rozsah	Detaily	Charakteristiky
X	Registr vstupu funkce	XB, XW, XL, XFnnnnn	XW00000 až XW00016	Vstup do funkce: Vstup bitu: XB000000 až XB00000F Vstup integeru: XW00001 až XW00016 Vstup dvojitého integeru: XL00001 až XL00015	Jedinečný pro každou funkci.
Y	Registr výstupu funkce	YB, YW, YL, YFnnnnn	YW00000 až YW00016	Vstup do funkce: Vstup bitu: YB000000 až YB00000F Vstup integeru: YW00001 až YW00016 Vstup dvojitého integeru: YL00001 až YL00015	
Z	Vnitřní registr funkce	ZB, ZW, ZL, ZFnnnnn	ZW0000 až ZW00063	Vnitřní registr jedinečný pro každou funkci. Může být použit pro funkci vnitřního zpracování.	
A	Vnější registr funkce	AB, AW, AL, AFhhhh	AW0000 až AW32767	Vnější registry s vloženou hodnotou adresy jako základ adresy. Pro propojení s S, M, I, O, # a Dannnnn.	
#	# Registr	#B, #W, #L, #Fnnnnn (#Annnnn)	#W00000 až #W16383	Jen volané registry. Mohou být volané pouze příslušnou funkcí. Rozsah použití nastavuje uživatel v MPE720	
D	D registr	DB, DW, DL, DFnnnnn (DAnnnnn)	DW00000 až DW16383	Vnitřní registry jedinečné pro každou funkci. Mohou být volány jen odpovídající funkcí. Rozsah použití nastavuje uživatel v MPE720.	
S	Systémový registr	SB, SW, SL, SFnnnnn (SAnnnnn)	Stejně jako u registrů výkresů DWG. Tyto registry jsou sdílené jak funkcemi, tak výkresy.		
M	Datový registr	MB, MW, ML, MFnnnnn (MAnnnnn)			
I	Vstupní registr	IB, IW, IL, IFhhhh (OAhhhh)			
O	Výstupní registr	OB, OW, OL, OFhhhh (OAhhhh)			
C	Registr konstant	CB, CW, CL, CFhhhh (CAhhhh)			

Tab. 8: Typy registrů funkcí a jejich popis

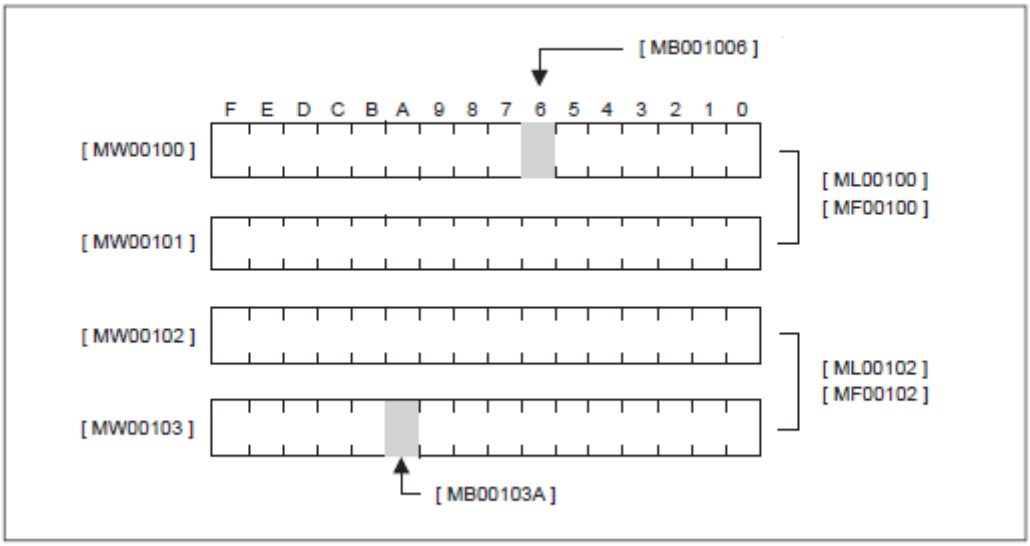
3.7.3. Datové typy a specifikace registrů

Do jednotlivých registrů se ukládají informace různého datového typu. V Tab. 9 jsou vypsány jednotlivé datové typy, se kterými se pracuje.

Typ	Datový typ	Číselná hodnota rozsahu	Poznámka
B	Bit	0, 1	Používané relé obvodem
W	Integer	od -32768 do 32767 od (8000H) do (7FFFH)	Používané pro operace s numerickými hodnotami. Hodnoty v () pro logické operace.
L	Long Integer	od -2147483648 do 2147483647 od (80000000H) do (7FFFFFFFH)	Používáme pro operace s numerickými hodnotami. Hodnoty v () pro logické operace.
F	Reálné číslo	+-(E1:OD175-E383:DO402+38)*0	Používané pro operace s numerickými hodnotami.
A	Adresa	od 0 až 32767	Použito pouze s uvedeným ukazatelem.

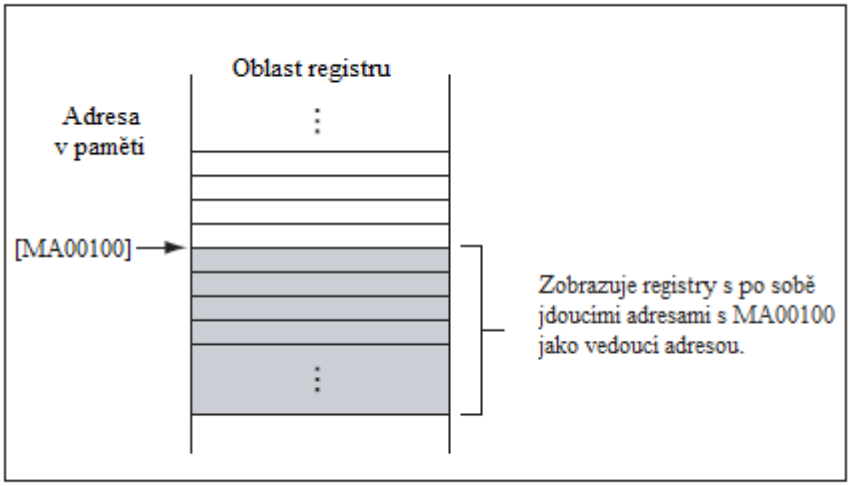
Tab. 9: Datové typy

Obr. 3.21 zobrazuje ukládání jednotlivých informací různých datových typů do registrů. Každé číslo registru značí jedno slovo. MB001006 značí, že informace bitu, která může nabývat jen velikosti 0 či 1, je umístěna na pozici šest v čísle registru 00100. MB00103A znamená, že informace bitu je umístěna na pozici A v čísle registru 00103. Do registrů MW00100 se ukládají informace datového typu integer, jenž slouží k operacím s numerickými hodnotami. ML00100 je registr pro long integer, jenž dosahuje dvojnásobné délky klasického integeru, tudíž jeho obor hodnot je větší. Stejně jako integer slouží k operaci s numerickými hodnotami. Registr MF00102 slouží k ukládání informací datového typu reálných čísel s pohyblivou řádovou čárkou. Při použití registru pro datový typ long integer či reálné číslo se dvě čísla registrů sloučí a je použito sudé číslo.



Obr. 3.21: Ukládání datových typů do registrů

Specifikace ukazatele a datový typ adresy:



Obr. 3.22: Ukládání adresy do registru

Obr. 3.22 zobrazuje způsob ukládání datového typu adresy do oblasti registru.

4. Logika simulace vývojového prostředí MP720 v MS Excel

Pro vytvoření programu na ovládání programovatelného automatu se ve firmách využívá speciální nástroj MP720. Před zavedením daného programu do provozu je dobré ho mimo odladit a zjistit tak případné chyby či kolapsy, které by se jinak projevily až za chodu. Pro odladění se využívají speciální nástroje, které jsou však

drahé. Cílem této práce je simulovat vývojové prostředí zmíněného MP720 v programu MS Excel, ve kterém se po naprogramování daného problému dá simulovat jeho průběh bez použití drahých odlaďovacích zařízení. Logikou dané simulace v MS Excel je propojení jednotlivých buněk sešitů, do kterých se nadefinují proměnné, jež by se ukládaly do vstupních registrů, s proměnnými deklarovanými pomocí Visual Basicu, který je nástrojem již zmíněného MS Excelu. Program dané proměnné zpracuje, dle obecné šablony používané ve VÚTS, a vrátí do jiných buněk, které mají zastupovat tak zvané výstupní registry. Jednotlivé listy mají znázorňovat samostatné programy a uživatelské funkce. Simulovat se zde také dají složitější cyklické operace. V našem případě jsme vedle zkušebních funkčních úloh typu sčítání simulovali rotaci vačkové hřídele, při čemž můžeme zadat různou úhlovou rychlost, měřítko zdvihu a periodu neboli počet pootočeních na jednu otáčku. Další velkou výhodou simulace pohybu v Excelu, je možnost krokování cyklu a tím podrobnější sledování pohybu, což je dobré pro představu fungování.

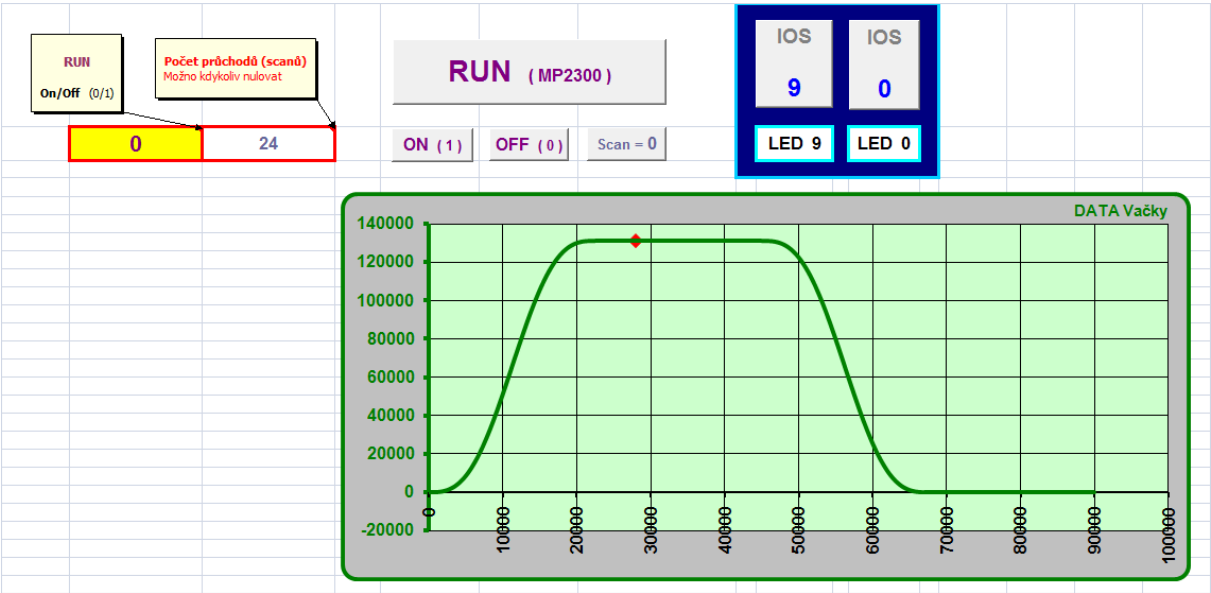
V praktické části diplomové práce se v prvním případě simulují logické vazby bitových proměnných, dále se simulují uživatelské funkce podle obecné šablony používané ve VÚTS a v neposlední řadě polohová data elektronické vačky v závislosti na zadané zdvihové závislosti pracovního členu.

4.1. Popsání jednotlivých listů Excelu

Při vytváření programu pro simulaci PLC kontroléru v Excelu jsme zavedli několik listů, jenž každý představuje určitou specifickou část.

4.1.1. List START

První list v Excelu je pojmenován „START“ a slouží k samotnému ovládání a simulaci daného pohybu. Zde můžeme nastavit, jestli chceme realizovat pouze jeden scan, nebo nechat provádět více scanů za sebou. Tím se nastavuje již zmíněné krokování. U skutečného kontroléru je délka scanu v řádu milisekund, což neumožňuje sledování jeho průběhu pouhým okem.



Obr. 4.1: List START

Tlačítko Scan = 0 umožňuje kdykoliv vynulovat počet scanů sčítajících se do červeně zvýrazněné tabulky. Tlačítka ON a OFF nastavují žlutě vybarvenou buňku buď na hodnotu jedna, nebo nula. Tato buňka pak udává, zda se budou provádět scany krokově nebo plynule. Dalším tlačítkem na listu „START“ je RUN. To je závislý na hodnotě buňky zadávané tlačítky ON/OFF. Při jeho použití mohou nastat různé varianty činnosti. Je-li ve zmíněné buňce aktivní nula a nesvítí ukazatel LED 9, zrealizuje se jeden scan, ale samotná vačka není v činnosti. Svítí-li ukazatel LED 9, zrealizuje se jeden scan a vačka se o danou hodnotu pootočí. Další variantou je, je-li ve žlutě zvýrazněné buňce přiřazena hodnota jedna a svítí ukazatel LED 9. Kontrolér je v činnosti, zvyšuje se počítadlo scanů a i samotná vačka je v činnosti. PLC se dá zastavit alternativně dvojklikem na libovolné buňce na libovolném místě. Je to stav, kdy program čeká na zadání hodnoty do buňky, a proto stojí. Toho se dá využít ke kontrole buněk, respektive registrů.

Zdrojový kód:
Pro tlačítko ON

```
'Po stisknutí tlačítka se spustí makro StartStopON() a nastaví buňku B2 na hodnotu 1.
    Public Sub StartStopON()
    Worksheets("START").Range("B2").Value = 1
    End Sub
```

Pro tlačítko OFF

```
'Stisknutím tlačítka se spustí makro StartStopOFF()a nastaví buňku B2 na hodnotu 0.
    Public Sub StartStopOFF()
    Worksheets("START").Range("B2").Value = 0
    End Sub
```

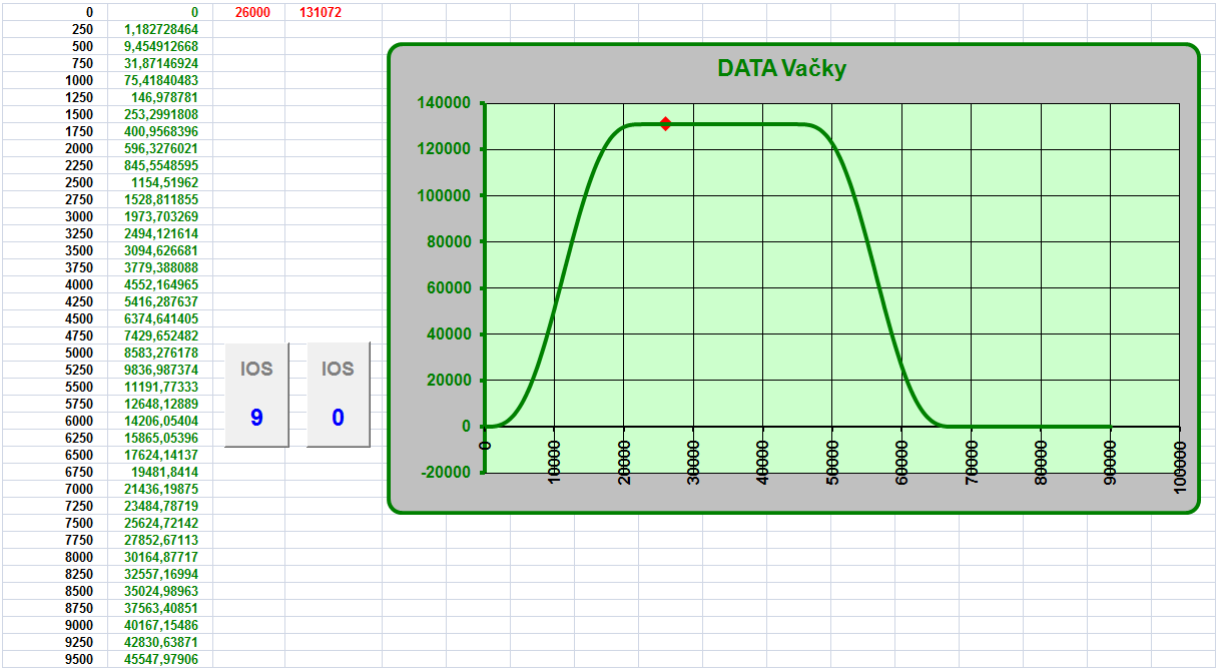
Pro tlačítko Scan

```
'Po stisknutí tlačítka se spustí makro PocetScanu0() a nastaví hodnotu buňky C2 na 0.
    Public Sub PocetScanu0()
    Worksheets("START").Range("C2").Value = 0
    End Sub
```

Při použití tlačítka RUN se spustí hlavní funkce pro simulaci vačkové hřídele. Tato část se rozebere později.

4.1.2. List DATA Vačky

Další list je pojmenován „DATA Vačky“. Smysl tohoto listu je ve vykreslování dat vačky a jejich zakreslení do grafu ve stavu, v jakém se v dané době vačka nachází. V levém sloupci na tomto listu je rozepsaná perioda na jednotlivé úseky a v pravém sloupci zelenými číslicemi jsou získaná data vačky v závislosti na periodě.



Obr. 4.2: List DATA Vačky

4.1.3. List M,C,IO,S

Třetí list v pořadí je nazván „M,C,IO,S“. Již podle názvu lze odhadnout, k čemu tento list slouží. Již v teoretické části bylo řečeno, že programovatelný automat MP2300 disponuje různými typy registrů. Na tomto listu jsou všechny společné registry pro funkce či výkresy vypsané ve sloupcích se značením, názvem, komentářem k čemu se daný registr využívá a v neposlední řadě hodnotou, pokud je zrovna v daném registru/buňce uložena. Zde probíhá monitorování činnosti elektronické vačky změnou hodnoty registru OL801C.

4.1.4. List D#_Main

Další list „D#_Main“ je samostatný program k uživatelské funkci sčítání. Je zde zakreslen tzv. vnitřní registr, který slouží jen pro tuto funkci. Do D registru ukládáme proměnné, v našem případě například Var_A, Var_B atd., které pak pošleme ke zpracování do již zmíněné uživatelské funkce a do vytvořeného programu, výsledky se pošlou zpět a uloží opět do D registrů.

D registr	Název proměnné	Hodnota proměnné	Komentář
DW00002	Var_A	111	proměnná 1. součtu, vstup do fceSOUCET
DW00004	Var_B	2	proměnná 1. součtu, vstup do fceSOUCET
DW00006	Var_C	3	proměnná 2. součtu, vstup do fceSOUCET
DW00008	Var_D	4	proměnná 2. součtu, vstup do fceSOUCET
DW00010	Var_AB	113	výsledek 1. součtu, výstup z fceSOUCET
DW00012	Var_CD	7	výsledek 2. součtu, výstup z fceSOUCET
DW00020	Var_X	113	posílá se adresa, vstup do fceSOUCET
DW00021	Var_Y	7	
DW00022	Var_XZ	120	

# registr	Název proměnné	Hodnota proměnné	Komentář

Obr. 4.3: List D#_Main

4.1.5. List D#01

List D#01 slouží k nastavování vstupních bitů a k zobrazení výstupních bitů získaných vyhodnocením z naprogramované logiky ve VBA. Ke změně bitů dochází ve žlutě podbarvených polí. Tyto pole jsou propojeny s proměnnými v proceduře H01(), se kterými daný program pracuje.

xxx

RUN

D registr	Název proměnné	Hodnota proměnné	Komentář
DB000001	Var_Bit1	0	
DB000002	Var_Bit2	0	
DB000003	Var_Bit3	0	
DB000004	Var_Bit4	1	
DB00005	Var_Bit5	0	
DB000006	Var_Bit6	0	
DB00000A	Var_BitA	1	Náběžná hrana
DW00010	Var_X	10	Int
DW00011	Var_Y	11	Int
DW00012	Var_Z	123	Sledovaná hodnota

# registr	Název proměnné	Hodnota proměnné	Komentář

Obr. 4.4: List D#01

4.1.6. List D#02

List označený „D#02“ je program sloužící ke spuštění mustru obecné funkce.

D registr	Název proměnné	Hodnota proměnné	Komentář
DB000000	Var_VstupOnOff	0	Vstup ON/OFF do MFCE (doplnit o volbu z HMI)
DL00002	Var_VystupWarning	123456	Výstup Warning
DL00004	Var_VystupAlarm	654321	Výstup Alarm
DW00006	Var_VystupRange	22	Výstup z oblasti odkud je vyslána hodnota

# registr	Název proměnné	Hodnota proměnné	Komentář

Obr. 4.5: List D#02

4.1.7. List D#03

List „D#03“ je program pro funkci spuštění elektronické vačky. Obsahuje vnitřní registr, kam se zadávají vstupní informace o elektronické vačce.

D registr	Název proměnné	Hodnota proměnné	Komentář
DB000009	Var_CamOnOff	0	Vstupní bit do fce CAM4 (bitový vstup ON-OFF)
DL00002	Var_VystupWarning	123456	Výstup Warning
DL00004	Var_VystupAlarm	654321	Výstup Alarm
DW00006	Var_VystupRange	22	Výstup z oblasti odkud je vyslána hodnota
DW00008	Var_RychlostMaster	2000	Zadávaná rychlost MASTERu [pulse/scan] (SpeedMaster)
DF00010	Var_Meritko	1	Měřítko zdvihu (Scale)
DL00012	Var_PeriodaMaster	90000	Perioda (cyklus) Master [pulse] (= 90000 [pulse])

# registr	Název proměnné	Hodnota proměnné	Komentář

Obr. 4.6: List D#03

4.1.8. Zbývající listy

Zbývající listy „CAM4“, „fceSOUCET“ znázorňují uživatelské funkce a „MFCE1“ je uživatelská funkce pro mustr obecné funkce, podle níž jsou uživatelské funkce sestaveny. Do listu zastupující uživatelskou funkci se pošle proměnná z listu samotného programu. Hodnota proměnné se uloží do předpřipravené proměnné vstupu, která má svůj ekvivalent ve VBA. To samé, ale opačným směrem se vykoná při vracení

výsledné proměnné do uživatelské funkce. Předpřipravené proměnné jsou propojeny s vytvořeným programem právě pomocí svých ekvivalentů ve VBA.

4.2. Mustr obecné funkce

Zdrojový kód:

```
Option Explicit
Public Sub MFCE1()
  'Deklarace lokálních proměnných definovaných v MFCE.
  'Definují se v Ladderu ve funkci MFCE podle listu „MFCE“.
  'Vstup.
      Dim Var_XB000000 As Range                'DB000000
  'Výstupy.
      Dim Var_Warning  As Range                'DL00018
      Dim Var_Alarm    As Range                'DL00020
      Dim Var_Range    As Range                'DW00022
  'Ostatní lokální proměnné.
      Dim Var_Work1 As Range                   'DB000341
  'Proměnná Var_Work2 je pouze deklarovaná, není nadefinovaná.
      Dim Var_Work2 As Range                   'DB000342
      Dim Var_Pom1_flag1scan As Range          'DB000343
      Dim Var_Pom2_flag1scan As Range          'DB000344
      Dim Var_flag1scan As Range               'DB000345
      Dim Var_Pom1_flag As Range               'DB000346
      Dim Var_Pom2_flag As Range               'DB000347
      Dim Var_flag As Range                    'DB000348
      Dim Var_1scanON_ON As Range              'DB00034A
      Dim Var_1scanON_OFF As Range             'DB00034B
      Dim Var_test_9 As Range                  'DB000359
      Dim Var_test_A As Range                  'DB00035A
      Dim Var_test_B As Range                  'DB00035B
      Dim Var_test_C As Range                  'DB00035C
      Dim Var_test_D As Range                  'DB00035D
      Dim Var_test_E As Range                  'DB00035E
      Dim Var_test_F As Range                  'DB00035F
```

```
'Pomocné proměnné, pomocí nichž se programuje VBA.
    Dim OnOff1 As Boolean
'Boolovská pomocná proměnná v této proceduře.
    Dim OnOff2 As Boolean
'Následuje deklarace lokálních proměnných. Musí dojít k propojení proměnných s příslušnými buňkami v Excelu. Toto se v MPE720 neprogramuje.
'Vstup.
    Set Var_XB000000 = Worksheets("MFCE1").Range("Q46")
'Výstupy.
    Set Var_Warning = Worksheets("MFCE1").Range("Q94")
    Set Var_Alarm = Worksheets("MFCE1").Range("Q96")
    Set Var_Range = Worksheets("MFCE1").Range("Q98")
'Ostatní lokální proměnné.
    Set Var_Work1 = Worksheets("MFCE1").Range("Q112")
'Nadefinovaná proměnná Var_Work2 není použita.
    Set Var_Work2 = Worksheets("MFCE1").Range("Q113")
    Set Var_Pom1_flag1scan = Worksheets("MFCE1").Range("Q114")
    Set Var_Pom2_flag1scan = Worksheets("MFCE1").Range("Q115")
    Set Var_flag1scan = Worksheets("MFCE1").Range("Q116")
    Set Var_Pom1_flag = Worksheets("MFCE1").Range("Q117")
    Set Var_Pom2_flag = Worksheets("MFCE1").Range("Q118")
    Set Var_flag = Worksheets("MFCE1").Range("Q119")
    Set Var_1scanON_ON = Worksheets("MFCE1").Range("Q121")
    Set Var_1scanON_OFF = Worksheets("MFCE1").Range("Q122")
    Set Var_test_9 = Worksheets("MFCE1").Range("Q137")
    Set Var_test_A = Worksheets("MFCE1").Range("Q138")
    Set Var_test_B = Worksheets("MFCE1").Range("Q139")
    Set Var_test_C = Worksheets("MFCE1").Range("Q140")
    Set Var_test_D = Worksheets("MFCE1").Range("Q141")
    Set Var_test_E = Worksheets("MFCE1").Range("Q142")
    Set Var_test_F = Worksheets("MFCE1").Range("Q143")
'Následuje oblast programování vlastní funkce, kde se využívá stejná logika jako v Ladderu.
'Tato část kódu je přeprogramována do Ladderu.
```

```
'Proměnná Var_XB000000 je základní vstup (ON/OFF), podle které se funkce MFCE rozděluje na základní oblasti 1 a 2.
'Když je vstup zapnut, do proměnné Var_XB000000 = 1 se uloží jedna. Spustí se první základní oblast.
'1.
        If  Var_XB000000 = 1 Then
                OnOff1 = Var_XB000000 And Not Var_Work1
                If    OnOff1 = True     Then
                        Var_1scanON_ON = 1
                Else
                        Var_1scanON_ON = 0
        Var_Work1 = Var_XB000000

'1.1 Vykonávání podmínek platných v prvním scanu činnosti FCE. Při prvním scanu se vstup ON přepisuje z OFF na ON.
        If  Var_1scanON_ON = 1 Then
                Var_test_9 = 1
'Var_test_9 je testovací proměnná. V Ladderu se jedná o testovací Rung.
        If  I_IL8002 = 0 Then
                Var_Pom1_flag1scan = 1
        Else
                Var_Pom1_flag1scan = 0
'Nastaví se varování.
        If  I_IL8004 = 0 Then
                Var_Pom2_flag1scan = 1
        Else
                Var_Pom2_flag1scan = 0
'Nastaví se alarm.
        If  Var_Pom1_flag1scan = 1 And Var_Pom2_flag1scan = 1 Then
                Var_flag1scan = 1
        Else
                Var_flag1scan = 0
'Konec prvního scanu 1.1.
        End If
```

```
'1.2 Součástí první základní oblasti je vykonávání podmínek platných v každém scanu činnosti FCE.
        Var_test_A = 1
'Testovací proměnná A. V Ladderu se jedná o testovací Rung.
        If  I_IL8002 = 0 Then
                Var_Pom1_flag = 1
        Else
                Var_Pom1_flag = 0
'Nastavení varování.
        If  I_IL8004 = 0 Then
                Var_Pom2_flag = 1
        Else Var_Pom2_flag = 0
'Nastavení alarmu.
        If  Var_Pom1_flag = 1 And Var_Pom2_flag = 1 Then
                Var_flag = 1
        Else
                Var_flag = 0
        If  Var_flag1scan = 1 And Var_flag = 1 Then
                Var_Warning = I_IL8002
                Var_Alarm = I_IL8004
                Var_Range = 13
        End If
'Konec 1.2.

'1.3
        If  Var_flag1scan = 1 And Var_flag = 1 Then
'F1
'Program proběhne v prvním scanu činnosti FCE, např. definice proměnných.
                If Var_1scanON_ON = 1 Then
                        Var_test_B = 1
'Testovací proměnná B. V Ladderu je to testovací Rung.
'Konec oblasti F1.
        End If
```

```
'F2.
'Průběh vlastní funkce. V následující oblasti bude program probíhat v každém scanu.
    Var_test_C = 1
'Testovací proměnná C. V Ladderu jde o testovací Rung.
'konec F2

'konec 1.3
    Else

'1.4.
'Tato část se vykoná, není-li FCE provedena.
    Var_test_D = 1
'Testovací proměnná D. V Ladderu jde o testovací Rung.
        Var_Warning = I_IL8002
        Var_Alarm = I_IL8004
        Var_Range = 14
'Natvrdo nastavíme proměnnou Var_flag1scan = 0, aby se nám funkce po vymazání alarmů nerozběhla.
    Var_flag1scan = 0
'konec 1.4
    End If

'konec 1
    Else

'2. Druhá základní oblast.
    OnOff2 = Not Var_XB000000 And Var_Work1
    If  OnOff2 = True Then
        Var_1scanON_OFF = 1
    Else
        Var_1scanON_OFF = 0
    Var_Work1 = Var_XB000000
```

```
'2.1. Oblast průběhu programu prvního scanu činnosti FCE po vstupu OFF. Po vstupu OFF se ON přepisuje na OFF.
    If  Var_1scanON_OFF = 1 Then
        Var_test_E = 1
'Testovací proměnná E. V Ladderu jde o testovací Rung.
'Konec oblasti 2.1.
    End If

'2.2.
'Možný prostor pro akce, které probíhají v případě vstupu OFF, např. nulování všech lokálních proměnných.
    Var_test_F = 1
'Testovací proměnná F. V Ladderu jde o testovací Rung.
    Var_test_9 = 0
    Var_test_A = 0
    Var_test_B = 0
    Var_test_C = 0
    Var_test_D = 0
    Var_Warning = 123456
    Var_Alarm = 654321
    Var_Range = 22
'Nulování lokálních proměnných.
    Var_Work1 = 0
    Var_Work2 = 0
    Var_Pom1_flag1scan = 0
    Var_Pom2_flag1scan = 0
    Var_flag1scan = 0
    Var_Pom1_flag = 0
    Var_Pom2_flag = 0
    Var_flag = 0
    Var_1scanON_ON = 0
    Var_1scanON_OFF = 0
'konec 2.2
'konec 2
```

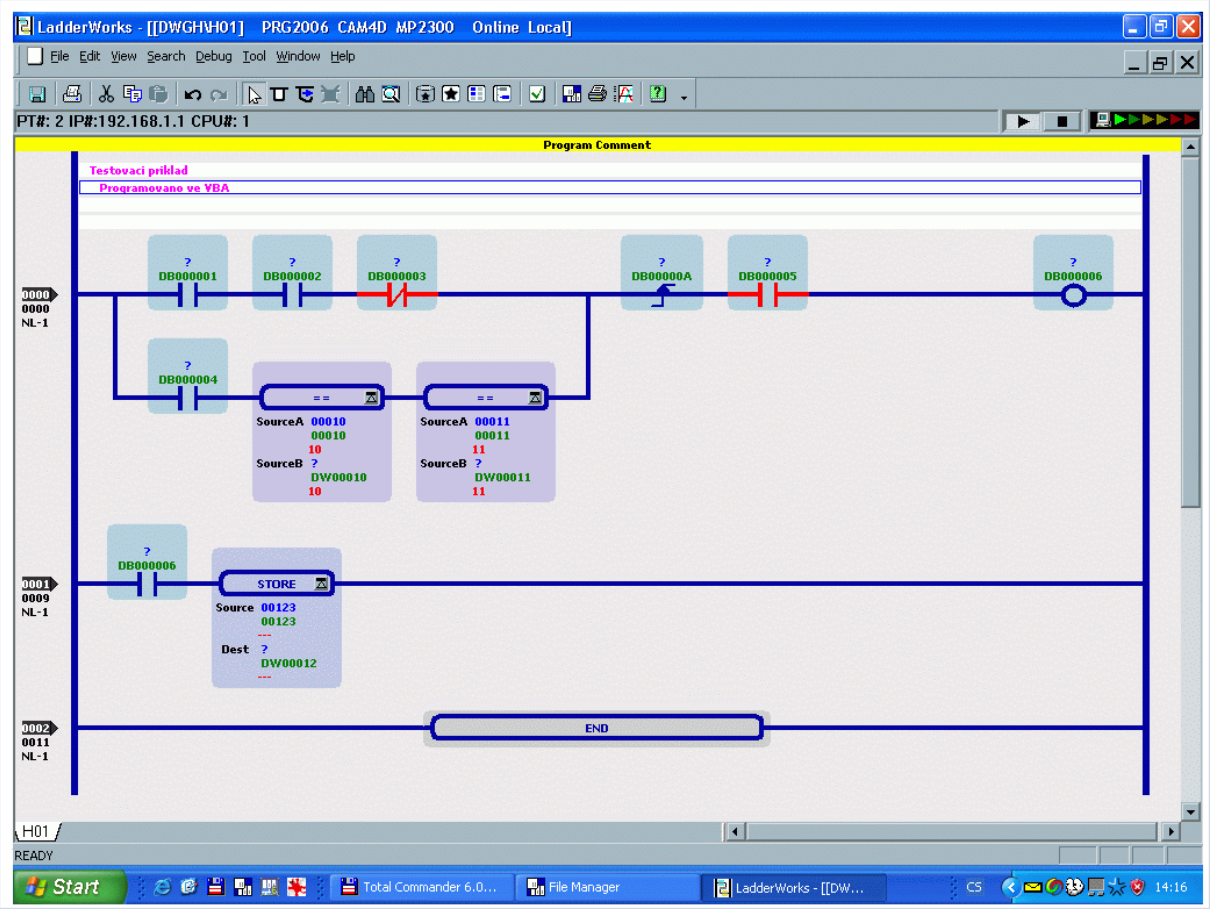
End If

End Sub

4.3. Simulace logických struktur PLC

Ladder diagram PLC je prostředek k programování komplexních řídicích systémů strojů. Často se jedná o komplikované logické struktury, které se bez speciálního hardware těžko testují či odlaďují. Speciální hardware je pro menší firmy finančně náročný. Simulace logických struktur v běžně používaném a dostupném programu Excel je vítanou finančně přijatelnou náhradou. Příkladem takového speciálního hardware pro MP2300 je MPE720.

Na obrázku níže je uveden skutečný zápis logiky v PLC systému kontroléru Yaskawa MP2300. Tato logika je pak převedena do programu ve VBA.



Obr. 4.7: Logika v MPE720

Zdrojový kód:

'Testovací procedura pro vyhodnocování bitových proměnných v logických vazbách.

Public Sub H01()

```
'Proběhne deklarace lokálních proměnných definovaných v H01.
    Dim Var_Bit1 As Range                    'DBW00001
    Dim Var_Bit2 As Range                    'DB000002
    Dim Var_Bit3 As Range                    'DB000003
    Dim Var_Bit4 As Range
    Dim Var_Bit5 As Range
    Dim Var_Bit6 As Range
    Dim Var_BitA As Range
    Dim Var_X    As Range
    Dim Var_Y    As Range
    Dim Var_Z    As Range
'Interní pomocné proměnné, které se v Ladderu nepoužívají.
    Dim OnOff  As Boolean
    Dim OnOff1 As Boolean
    Dim OnOff2 As Boolean
    Dim pom1   As Integer
    Dim pom2   As Integer
'Definice lokálních proměnných. V této části programu musí dojít k propojení
proměnných s příslušnými buňkami.
    Set Var_Bit1 = Worksheets("D#01").Range("G7")
    Set Var_Bit2 = Worksheets("D#01").Range("G8")
    Set Var_Bit3 = Worksheets("D#01").Range("G9")
    Set Var_Bit4 = Worksheets("D#01").Range("G10")
    Set Var_Bit5 = Worksheets("D#01").Range("G11")
    Set Var_Bit6 = Worksheets("D#01").Range("G12")
    Set Var_BitA = Worksheets("D#01").Range("G13")
    Set Var_X = Worksheets("D#01").Range("G14")
    Set Var_Y = Worksheets("D#01").Range("G15")
    Set Var_Z = Worksheets("D#01").Range("G16")
'Složené schéma s logikou zpracování bitů.
    OnOff1 = (Var_Bit1 And Var_Bit2 And Not Var_Bit3) Or _
                (Var_Bit4 And Var_X = 10 And Var_Y = 11)
    If  OnOff1 = True Then
        pom1 = 1
```

```
Else
        pom1 = 0
OnOff2 = pom1 And  Not Var_BitA
If  OnOff2 = True Then
        pom2 = 1
Else
        pom2 = 0
OnOff = pom2 And Var_Bit5
If  OnOff  =  True Then
        Var_Bit6 = 1
Else
        Var_Bit6 = 0
If  Var_Bit6  = 1 Then Var_Z = 123
Var_BitA = pom1   'Kopie pro příští scan (viz. ON/OFF-PLS)
End Sub
```

4.4. Simulace uživatelských funkcí podle obecné šablony používané ve VÚTS

Jako vzorová uživatelská funkce se zvolila Funkce sčítání, protože se jedná o funkci jednoduchou a tím pádem názornou.

4.4.1. Uživatelská funkce sčítání

Hodnoty, které chceme sčítat, zapíšeme do listu „D#_Main“ do buněk souvisejících a ležících vedle proměnných Var_A, Var_B, Var_C a Var_D. Ty se pošlou do vlastní funkce na listu „fceSOUCET“ a dále se přesunou do vstupních registrů, které využívá VBA. VBA data ze vstupních registrů zpracuje dle zdrojového kódu a pošle zpět do výstupních registrů na listu „fceSOUCET“. Z výstupních registrů se přesunou a pošlou zpět do listu vlastního programu „D#_Main“. Na listu „D#_Main“ se pro získání součtu sčítanců zmáčkne tlačítko RUN, které je napojeno na makro SpustitPLC.

Hlavní funkce
Zdrojový kód:

```
Option Explicit
  'Makro SpustitPLC() je hlavní spuštění procedura. Tato procedura volá proceduru H,
procedura H volá ostatní H01, H02 atd.
  'Spouštění simulačního programu PLC.
        Public Sub SpustitPLC()
        Dim i As Range
        Set i = Worksheets("START").Range("C2")
  'V této části procedura SpustitPLC volá proceduru H.
        Call H
  'Následuje zpracování PLC po scanech, je-li výsledek podmínky FALSE, proběhne
pouze jeden průchod. Smyslem je volání funkce sebe sama. V našem případě, je-li
podmínka TRUE, spustí se funkce SpustitPLC znovu za dvě sekundy.
        If  Worksheets("START").Range("B2").Value = 1 Then _
              Application.OnTime Now + TimeValue("00:00:02"), "SpustitPLC"
  'Níže je počitadlo scanů.
        i = i + 1
        End Sub
```

Call H volá ve zdrojovém kódu proceduru H. Procedura H deklaruje lokální proměnné použité pro funkci sčítání a propojuje proměnné s příslušnými buňkami. Volá procedury DefiniceEXTERN, fceSOUCET, H01, H02 and H03.

Procedura H()
Zdrojový kód:

```
Public Sub H()
  'Deklarace lokálních proměnných definovaných v H.
        Dim Var_A  As Range                    'DW00002
        Dim Var_B  As Range                    'DW00004
        Dim Var_C  As Range                    'DW00006
        Dim Var_D  As Range                    'DW00008
        Dim Var_AB As Range                    'DW00010
        Dim Var_CD As Range                     'DW00012
        Dim Var_X  As Range                    'DW00020
```

```
    Dim Var_Y  As Range                    'DW00021
    Dim Var_XY As Range                    'DW00022
'Voláním procedury DefiniceEXTERN se nadefinují externí a systémové proměnné.
    Call DefiniceEXTERN
'Následně proběhne definice lokálních proměnných a jejich propojení s buňkami.
    Set Var_A = Worksheets("D#_Main").Range("G8")
    Set Var_B = Worksheets("D#_Main").Range("G9")
    Set Var_C = Worksheets("D#_Main").Range("G10
    Set Var_D = Worksheets("D#_Main").Range("G11")
    Call fceSOUCET
    Call H01
    Call H02
    Call H03
    End Sub
```

Procedury H02, H03 a funkce fceSOUCET jsou popsány v následujících kapitolách 5.4.1.4. až 5.4.1.6. Procedura H01 byla již vysvětlena v kapitole 5.3. Simulace logických struktur PLC.

Funkce definice a deklarace externích a systémových proměnných
Zdrojový kód:

```
'Deklarace S registrů.
    Public S_AlwaysON As Range
'Deklarace M a C registrů neproběhne, žádných se v našem programu nevyužívá.
'Deklarace výstupních O registrů.
    Public O_OW8008 As Range
'Registr pro uložení pohybových příkazů.
    Public O_OL801C As Range
'Registr pro nastavení polohových referencí.
    Public O_OB04110 As Range
'Registr ledovky číslo 0 na IOS (nedefinovaný).
    Public O_OB04119 As Range
'Registr pro ledovku 9 na IOS pro list START.
'Deklarace vstupních I registrů.
    Public I_IB80001 As Range
'Registr pro uchování hodnoty je-li servo ON/OFF.
```

```
    Public I_IW8008 As Range
'Registr pro uložení pohybových příkazů.
    Public I_IL8002 As Range
'Upozornění.
    Public I_IL8004 As Range
'Alarm.
    Public I_IB04100 As Range
'Registr pro vstup stavu tlačítka 0 na IOS pro list START
    Public I_IB04109 As Range
'Registr pro vstup stavu tlačítka 9 na IOS pro list START
'Definice externích a systémových proměnných. Propojení proměnných s buňkami.

    Public Sub DefiniceEXTERN()
'Definice S registrů.
    Set S_AlwaysON = Worksheets("M,C,IO,S").Range("AE11")
'Definice výstupních O registrů.
    Set O_OW8008 = Worksheets("M,C,IO,S").Range("S119")
    Set O_OL801C = Worksheets("M,C,IO,S").Range("S153")
    Set O_OB04110 = Worksheets("M,C,IO,S").Range("S217")
    Set O_OB04119 = Worksheets("M,C,IO,S").Range("S226")
 'Definice vstupních I registrů.
    Set I_IB80001 = Worksheets("M,C,IO,S").Range("Y9")
    Set I_IL8002 = Worksheets("M,C,IO,S").Range("Y27")
    Set I_IL8004 = Worksheets("M,C,IO,S").Range("Y45")
    Set I_IW8008 = Worksheets("M,C,IO,S").Range("Y64")
    Set I_IB04100 = Worksheets("M,C,IO,S").Range("Y217")
    Set I_IB04109 = Worksheets("M,C,IO,S").Range("Y226")
    End Sub
```

Další částí, která je volaná z podprocedury hlavní procedury SpustiPLC() je samotná funkce fceSOUCET.

Funkce sčítání
Zdrojový kód:

```
Public Sub fceSOUCET()
```

```
'Deklarace lokálních proměnných definovaných ve vlastní funkci fceSOUCET.
    Dim Var_A  As Range                 'DW00001
    Dim Var_B  As Range                 'DW00002
    Dim Var_C  As Range                 'DW00003
    Dim Var_D  As Range                 'DW00004
    Dim Var_AB As Range                 'DW00018
    Dim Var_CD As Range                 'DW00019
'Deklarace vnějších funkčních proměnných. Hodnoty z vnějších funkčních proměnných jsou získávány přes zasílání adresy.
    Dim Var_X  As Range                 'AW00000
    Dim Var_Y  As Range                 'AW00001
    Dim Var_XY As Range                 'AW00002
'Po deklaraci musí následovat definice lokálních proměnných.
    Set Var_A = Worksheets("fceSOUCET").Range("Q62")
    Set Var_B = Worksheets("fceSOUCET").Range("Q63")
    Set Var_C = Worksheets("fceSOUCET").Range("Q64")
    Set Var_D = Worksheets("fceSOUCET").Range("Q65")
    Set Var_AB = Worksheets("fceSOUCET").Range("Q94")
    Set Var_CD = Worksheets("fceSOUCET").Range("Q95")
'Také je nutno definovat externí proměnné. Nesmíme však zapomenout, že se jedná o vnější proměnnou a tedy o adresu. Nadefinuje se vazba na posílanou adresu. Následující proměnné odvozené od této adresy se odvozují od definované buňky.
    Set Var_X = Worksheets("D#_Main").Range("G14").Cells(1, 1)
    Set Var_Y = Var_X.Cells(2, 1)
    Set Var_XY = Var_X.Cells(3, 1)

'Vlastní výpočet
    Var_AB = Var_A + Var_B
    Var_CD = Var_C + Var_D
'Práce s adresami
    Var_X = Var_AB
    Var_Y = Var_CD
    Var_XY = Var_X + Var_Y
    End Sub
```

Procedura H02()

Procedura H02 slouží k volání mustru obecné funkce. Viz kapitola 5.2.

Zdrojový kód:

```
Public Sub H02()
  'Nejprve proběhne deklarace lokálních proměnných definovaných v H02.
      Dim Var_VstupOnOff    As Range                    'DB000000
      Dim Var_VystupWarning As Range                    'DL00002
      Dim Var_VystupAlarm   As Range                    'DL00004
      Dim Var_VystupRange   As Range                    'DW00006
   'Následuje jejich definice. Musí dojít k propojení proměnných s příslušnými buňkami.
      Set Var_VstupOnOff = Worksheets("D#02").Range("G7")
      Set Var_VystupWarning = Worksheets("D#02").Range("G8")
      Set Var_VystupAlarm = Worksheets("D#02").Range("G9")
      Set Var_VystupRange = Worksheets("D#02").Range("G10")
      Call MFCE1
  'Po definici proběhne samotné volání mustru obecné funkce, jejíž kód je popsán již výše.
      End Sub
```

Zbývající procedura H03 souvisí až se simulací elektronické vačky. Pro úplnost a přehlednost ji popíšeme již zde.

Procedura H03()
Zdrojový kód:

```
Public Sub H03()
  'První co opět proběhne je deklarace lokálních proměnných.
      Dim Var_CamOnOff As Range                 'DB000009
      Dim Var_RychlostMaster As Range           'DW00008
      Dim Var_Meritko As Range                  'DF00010
      Dim Var_PeriodaMaster As Range            'DL00012
  'Interní pomocná proměnná.
      Dim OnOff  As Boolean
  'Opět následuje definice lokálních proměnných s propojením proměnných k příslušným buňkám.
      Set Var_CamOnOff = Worksheets("D#03").Range("G7")
```

```
        Set Var_RychlostMaster = Worksheets("D#03").Range("G12")
        Set Var_Meritko = Worksheets("D#03").Range("G13")
        Set Var_PeriodaMaster = Worksheets("D#03").Range("G14")
'Následuje logika pro zjištění, zda je vačka zapnuta.
        OnOff = (I_IB04109 Or Var_CamOnOff) And Not I_IB04100
        If  OnOff = True Then
                Var_CamOnOff = 1
         Else
                Var_CamOnOff = 0
'Definice výstupních O registrů ledovek.
'Simulace rozsvícení led diody 9 je znázorněna podbarvením buňky červeně.
        If  Var_CamOnOff = 1 Then
                O_OB04119 = 1
                Worksheets("START").Range("I2").Interior.ColorIndex = 3
'Je-li vačka zapnuta, rozsvítí se led dioda 9 červeně.
        Else
                O_OB04119 = 0
                Worksheets("START").Range("I2").Interior.ColorIndex = 2
        End If
'Volání vlastní vačkové funkce CAM4.
        Call CAM4
        End Sub
```

4.5. Simulace elektronické vačky

Simulací elektronické vačky máme na mysli simulaci polohových dat podle zadané zdvihové závislosti pracovního členu hřídele servomotoru. Ovládání elektronické vačky je na prvním listu Excelu pod názvem „START“. Fungování této stránky již bylo popsáno o pár stránek výše. Nejprve je potřeba zadat hodnotu 1 do proměnné Var_CamONOff, jenž má znázorňovat uvedení vačky do provozu. Program se spustí tlačítkem RUN, čímž se začne vykonávat procedura SpustitPLC(). Zdrojový kód této hlavní spouštěcí procedury je již popsán na straně 38. Zpracování jednotlivých procedur a podprocedur probíhá velmi podobně jako u funkce sčítání. Hlavní procedura SpustitPLC() volá podproceduru H(), která volá funkci DefiniceEXTERN, dále

fceSOUCET, H01, H02 a H03. Rozdíl mezi zpracováním předchozí funkce součet od zpracování pohybu vačky, je ve vyhodnocení navíc procedury CAM4().

4.5.1. Procedura elektronické vačky

Zdrojový kód:

Níže je popsán kód pro vyhodnocení a simulaci pohybu elektronické vačky. Základ této procedury je v obecné šabloně používané ve VÚTS.

```
Public Sub CAM4()
  'Deklarace lokálních proměnných definovaných v CAM4.
  'Vstup.
      Dim Var_XB000000 As Range                 'DB000000
  'Výstupy.
      Dim Var_Warning  As Range                 'DL00018
      Dim Var_Alarm    As Range                 'DL00020
      Dim Var_Range    As Range                 'DW00022
  'Ostatní lokální proměnné.
      Dim Var_Work1 As Range                    'DB000341
      Dim Var_Work2 As Range                    'DB000342
  'Proměnná Var_Work2 není použita.
      Dim Var_Pom1_flag1scan As Range           'DB000343
      Dim Var_Pom2_flag1scan As Range           'DB000344
      Dim Var_flag1scan As Range                'DB000345
      Dim Var_Pom1_flag As Range                'DB000346
      Dim Var_Pom2_flag As Range                'DB000347
      Dim Var_flag As Range                     'DB000348
      Dim Var_1scanON_ON As Range               'DB00034A
      Dim Var_1scanON_OFF As Range              'DB00034B
      Dim Var_test_9 As Range                   'DB000359
      Dim Var_test_A As Range                   'DB00035A
      Dim Var_test_B As Range                   'DB00035B
      Dim Var_test_C As Range                   'DB00035C
      Dim Var_test_D As Range                   'DB00035D
      Dim Var_test_E As Range                   'DB00035E
```

```
    Dim Var_test_F As Range                    'DB00035F
'Deklarace lokálních proměnných nad rámec MFCE1, slouží pro uložení rychlosti otáčení, měřítka zdvihu a periodu, počet úseků na jednu otáčku.
    Dim Var_Pom3_flag As Range                 'DB000350
    Dim Var_Pom8008 As Range                   'DB000351
    Dim Var_SpeedMaster As Range               'DW00001
    Dim Var_Scale As Range                     'DW00002
    Dim Var_CyklusMaster As Range              'DW00004
    Dim Var_VMaster As Range                   'DL00036
    Dim Var_Slave As Range                     'DL00038
'Pomocné proměnné, pomocí nichž se programuje ve VBA.
    Dim OnOff1 As Boolean
    Dim OnOff2 As Boolean
'prozatímní pomocná proměnná pro vykreslení grafu
    Static i As Integer
'Následně proběhne definice lokálních proměnných. Musí dojít k propojení proměnných s příslušnými buňkami. Toto se v MPE720 neprogramuje.
'Vstup.
    Set Var_XB000000 = Worksheets("CAM4").Range("Q46")
'Výstupy.
    Set Var_Warning = Worksheets("CAM4").Range("Q94")
    Set Var_Alarm = Worksheets("CAM4").Range("Q96")
    Set Var_Range = Worksheets("CAM4").Range("Q98")
'Ostatní lokální proměnné.
    Set Var_Work1 = Worksheets("CAM4").Range("Q112")
    Set Var_Work2 = Worksheets("CAM4").Range("Q113")
'Proměnná Var_Work2 není použita.
    Set Var_Pom1_flag1scan = Worksheets("CAM4").Range("Q114")
    Set Var_Pom2_flag1scan = Worksheets("CAM4").Range("Q115")
    Set Var_flag1scan = Worksheets("CAM4").Range("Q116")
    Set Var_Pom1_flag = Worksheets("CAM4").Range("Q117")
    Set Var_Pom2_flag = Worksheets("CAM4").Range("Q118")
    Set Var_flag = Worksheets("CAM4").Range("Q119")
    Set Var_1scanON_ON = Worksheets("CAM4").Range("Q121")
```

```
    Set Var_1scanON_OFF = Worksheets("CAM4").Range("Q122")
    Set Var_test_9 = Worksheets("CAM4").Range("Q137")
    Set Var_test_A = Worksheets("CAM4").Range("Q138")
    Set Var_test_B = Worksheets("CAM4").Range("Q139")
    Set Var_test_C = Worksheets("CAM4").Range("Q140")
    Set Var_test_D = Worksheets("CAM4").Range("Q141")
    Set Var_test_E = Worksheets("CAM4").Range("Q142")
    Set Var_test_F = Worksheets("CAM4").Range("Q143")
'Definice lokálních proměnných nad rámec CAM4.
    Set Var_Pom3_flag = Worksheets("CAM4").Range("Q128")
    Set Var_Pom8008 = Worksheets("CAM4").Range("Q129")
    Set Var_SpeedMaster = Worksheets("CAM4").Range("Q62")
    Set Var_Scale = Worksheets("CAM4").Range("Q63")
    Set Var_CyklusMaster = Worksheets("CAM4").Range("Q65")
    Set Var_VMaster = Worksheets("CAM4").Range("Q145")
    Set Var_Slave = Worksheets("CAM4").Range("Q146")
'Následuje oblast programování vlastní funkce, používá se stejná logika jako v Ladderu. Tato oblast se přeprogramovává do LADDERU.
'Proměnná Var_XB000000 je základní vstup informace ON/OFF, podle které se programování vlastní funkce rozděluje na dvě základní oblasti 1 a 2.
    If  Var_XB000000 = 1 Then
'1. Kód první základní oblasti se spustí, je-li proměnná Var_XB000000 velikosti 1.
    OnOff1 = Var_XB000000 And Not Var_Work1
    If  OnOff1 = True Then
        Var_1scanON_ON = 1
    Else
        Var_1scanON_ON = 0
    Var_Work1 = Var_XB000000
'1.1 Podmínky provedení platné v prvním scanu činnosti FCE, když vstup přepisuje vstup z ON na OFF.
    If  Var_1scanON_ON = 1 Then
        Var_test_9 = 1
'Var_test_9 je testovací proměnná. V Ladderu se jedná o testovací Rung.
    If  I_IW8008 = 0 Then
```

```
            Var_Pom8008 = 1
        Else
            Var_Pom8008 = 0
    'Proměnná registru pro příkazy pohybu.
        If Var_Pom8008 = 1 Then _
            Var_flag1scan = 1
        Else
            Var_flag1scan = 0
    'konec 1.1
        End If
    '1.2 Součástí první základní oblasti je vykonávání podmínek platných v každém scanu činnosti FCE. Výstup FCE jsou upozornění, alarm a výstup z oblasti, odkud je vyslána hodnota.
        Var_test_A = 1
    'Testovací proměnná A. V Ladderu se jedná o testovací Rung.
        If  I_IL8002 = 0 Then Var_Pom1_flag = 1 Else Var_Pom1_flag = 0
    'Upozornění.
        If  I_IL8004 = 0 Then Var_Pom2_flag = 1 Else Var_Pom2_flag = 0
    'Alarm.
        If  I_IB80001 = 1 Then Var_Pom3_flag = 1 Else Var_Pom3_flag = 0
    'Je-li servo zapnuto či vypnuto.
        If  Var_Pom1_flag = 1 And  Var_Pom2_flag = 1 And  Var_Pom3_flag = 1 Then
            Var_flag = 1 Else Var_flag = 0
        If  Var_flag1scan = 1 And Var_flag = 1 Then
            Var_Warning = I_IL8002
            Var_Alarm = I_IL8004
            Var_Range = 13
        End If
    'konec 1.2
    '1.3
        If  Var_flag1scan = 1 And Var_flag = 1 Then
    'F1 je oblast, kdy program proběhne v prvním scanu činnosti FCE, např. definice proměnných.
        If  Var_1scanON_ON = 1 Then
```

```
            Var_test_B = 1
'Testovací proměnná A. V Ladderu se jedná o testovací Rung.
'Konec oblasti F1.
        End If
'F2
'V oblast F2 probíhá řešení vlastní funkce.
        Var_test_C = 1
'Testovací proměnná C. V Ladderu se jedná o testovací Rung.
        Var_VMaster = Var_VMaster + Var_SpeedMaster
'Nezávisle proměnná překročí periodu nezávisle proměnné.
        If  Var_VMaster  > Var_CyklusMaster Then
            Var_VMaster = Var_VMaster - Var_CyklusMaster
            i = 0
        End If
'Výpočet závisle proměnné z tabulky vačky.
        Var_Slave =  FGN(CLng(Var_VMaster))
'Měřítkování.
        Var_Slave = Var_Slave * Var_Scale
'Příkazy pohybu.
        O_OW8008 = 4
        O_OL801C = CLng(Var_Slave)
'Pro vykreslení vypočtených hodnot.
        i = 1
'i=i+1 zanechává body => "kreslí čáru"
        Worksheets("DATA Vacky").Cells(i, 3) = Var_VMaster
        Worksheets("DATA Vacky").Cells(i, 4) = Var_Slave
'konec F2
'konec 1.3
        Else
'1.4
'Tato část se vykoná, není-li FCE provedena.
        Var_test_D = 1
'Testovací proměnná D. V Ladderu se jedná o testovací Rung.
        Var_Warning = I_IL8002
```

```
        Var_Alarm = I_IL8004
        Var_Range = 14
    'Natvrdo nastavíme proměnnou Var_flag1scan = 0, aby se nám funkce po vymazání alarmů nerozběhla.
        Var_flag1scan = 0
    'konec 1.4
        End If
    'konec 1
        Else
    '2
        OnOff2 = Not Var_XB000000 And Var_Work1
        If  OnOff2 = True Then
            Var_1scanON_OFF = 1
        Else
            Var_1scanON_OFF = 0
        Var_Work1 = Var_XB000000
    '2.1 Oblast průběhu programu prvního scanu činnosti FCE po vstupu OFF. Po vstupu OFF se ON přepisuje na OFF.
        If Var_1scanON_OFF = 1 Then
            Var_test_E = 1
    'Testovací proměnná E. V Ladderu se jedná o testovací Rung.
    'Žádné příkazy.
        O_OW8008 = 0
    'konec 2.1
        End If
    '2.2 Postor pro akce, které probíhají v případě vstupu OFF jako je například nulování všedh lokálních proměnných. Výstup funkce je upozornění, alarm a range rovno 22.
        Var_test_F = 1
    'Testovací proměnná F. V Ladderu se jedná o testovací Rung.
        Var_test_9 = 0
        Var_test_A = 0
        Var_test_B = 0
        Var_test_C = 0
        Var_test_D = 0
```

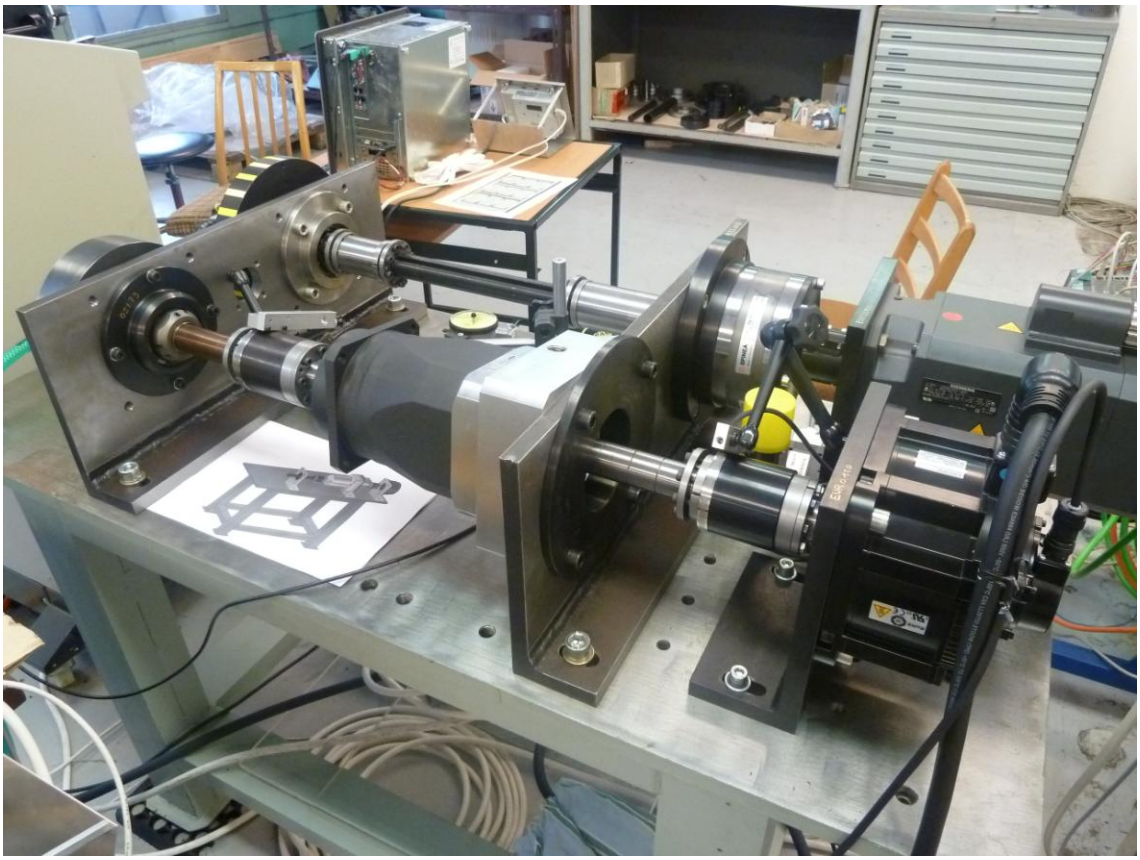
```
        Var_Warning = 123456
        Var_Alarm = 654321
        Var_Range = 22
    'Nulování lokálních proměnných.
        Var_Work1 = 0
        Var_Work2 = 0
        Var_Pom1_flag1scan = 0
        Var_Pom2_flag1scan = 0
        Var_flag1scan = 0
        Var_Pom1_flag = 0
        Var_Pom2_flag = 0
        Var_flag = 0
        Var_1scanON_ON = 0
        Var_1scanON_OFF = 0
        Var_VMaster = 0
        Var_Slave = 0
        O_OL801C = 0
    'Pomocná proměnná pro vykreslení grafu.
        i = 0
    'konec 2.2
    'konec 2
        End If
End Sub
```

5. Reálná sestava

Námi získané výsledky ze simulace PLC kontroléru Yaskawa v programu Excel můžeme ověřit na sestavené reálné soustavě vytvořené v laboratoři ve firmě VÚTS. Pomocí kontroléru se řídí připojená vačková hřídel.

Obr. 5.1: Reálná sestava kontroléru se SERVOPACKem

Kontrolérem v naší reálné soustavě je již zmíněný Yaskawa MP2300, avšak v jeho menší variantě MP2300s. K základnímu modulu je připojen volitelný modul LIO-02. Jedná se o vstupní/výstupní modul. K MP2300s je připojen SERVOPACKs SGDV-120D□1A, který ovládá žádané pohyby vačkové hřídele.



Obr.5.2: Řízená vačková hřídel

Závěr

Simulační program PLC kontroléru vytvořený v programu Excel je s výhodou používán pro ladění a kontrolu uživatelských funkcí. Přednost takto vytvořeného simulačního programu je hlavně v jeho názornosti, kdy můžeme simulovat průběh jednoho scanu, který v reálném kontroléru trvá řádově pouhé milisekundy, což má za následek možnost ladění složitých logických struktur, přičemž můžeme sledovat průběh hodnot proměnných po jednotlivých scanech PLC. Struktura programu je ve VBA předpřipravena do jednotlivých bloků, do kterých uživatel již pouze vpisuje funkční algoritmy. Tato předpřipravená struktura programu je nazvána mustrem obecné funkce. Výsledkem této práce je získání funkčního programu, který svojí schopností simulace, dokáže částečně nahradit speciální simulační program MPE720, nabízený za vysokou cenu profesionálními firmami. Simulace v Excelu sice postrádá určité speciální funkce, které jinak profesionální MPE720 nabízí, ale pro firmy, kterým pro jejich výrobu postačí základní přehled, je simulační program v Excelu postačující.

Použitá literatura

Knihy:

[1] Šmejkal, Ladislav; Martinásková Marie: PLC a automatizace. 1.díl, Základní pojmy, úvod do programování. Praha: BEN, 1999, ISBN 80-86056-58-9

[2] Walkenbach, John: Microsoft Excel 2000&2002 programování ve VBA. Computer Press Brno, 2004, ISBN 80-7226-547-4

[3] Yaskawa: Machine controller MP2000 SERIES. Yaskawa electric corporation, Publish in Japan September 2008.

[4] Yaskawa: Machine Controller MP2300 Basic Module USER'S MANUAL. Yaskawa electric corporation, 2003.

[5] Bolton, W: Programmable logic controllers. Elsevier Ltd., 2009, ISBN 978-1-85617-751-1

[6] Suh, Suk-Hwan; Kang, Seong-Kyoon; Chung, Dae-Hyuk; Stroud, Ian: Theory and Design of CNC Systems. Springer-Verlag London, 2008, ISBN 978-1-84800-335-4

Internetové zdroje:

[7] Charakteristika společnosti, VÚTS Liberec http://www.vuts.cz/cze/profil-firmy.html

[8] Šmejkal, Ladislav; Prokopová, Zuzana; Kovář, Josef: PLC-hardware-STR. PLC_hardware_STR.pdf

[9] Ing. Vondra, Zdeněk: Základy programování PLC.Vlašim, 2006. http://web.spsejecna.cz/projekt/PLC_zakl.pdf

[10] Doc. Ing. Kmínek, Miloš, CSc. http://uprt.vscht.cz/kminekm/mrt/F5/F5k53-PLC.htm

[11] Historie programovatelných automatů a jejich současné efektivní použití, AUTOMA, http://www.odbornecasopisy.cz/index.php?id_document=28831

[12] Programovací jazyky pro PLC, AUTOMA, http://www.automatizace.cz/article.php?a=142

[13] Programování PLC podle normy IEC EN 61131-3 – víc než jednotné jazyky, AUTOMA, http://www.odbornecasopisy.cz/index.php?id_document=30310

[14] Doc. Ing. Zezulka, František, CSc.; Ing. Bradáč, Zdeněk; Ing. Fiedler, Petr; Ing. Kučera, Pavel; Ing. Štohl, Radek: Programovatelné automaty. Brno, 2003.

Přílohy

Seznam příloh:

1. List M,C,IO,S.
2. List CAM4.
3. List fceSOUCET.
4. List MFCE1.
5. Uživatelská funkce mustru obecné funkce.
6. Uživatelská funkce sčítání.
7. Uživatelská funkce vačkové hřídele.