VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY
A KOMUNIKAČNÍCH TECHNOLOGIÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV MIKROELEKTRONIKY

DEPARTMENT OF MICROELECTRONICS

POROVNÁNÍ PROGRAMOVACÍCH JAZYKŮ PLC

COMPARISON OF PLC PROGRAMMIN LANGUAGES

BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE — Kamil Noga
AUTHOR

VEDOUCÍ PRÁCE — doc. Ing. Petr Fiedler, Ph.D.
SUPERVISOR

BRNO 2016

Bakalářská práce

bakalářský studijní obor
Mikroelektronika a technologie

Student:	Kamil Noga	***ID:***	147533
Ročník:	3	***Akademický rok:***	2015/2016

NÁZEV TÉMATU:

Porovnání programovacích jazyků PLC

POKYNY PRO VYPRACOVÁNÍ:

Seznamte se s programovacími jazyky používanými pro programování programovatelných automatů (PLC). Seznamte se s vývojovým prostředím pro systémy Heidenheim a Mitsubishi electric a s využitím těchto platforem vytvořte aplikace. Na základě praktických zkušeností porovnejte výhody a nevýhody jednotlivých jazyků a dále pak srovnejte použitá vývojová prostředí. Porovnejte jednotivé programovací jazyky, implementujte typické úlohy a zhodnoťte vhodnost jazyků pro dané typy úloh.

DOPORUČENÁ LITERATURA:

Termín zadání:	8. 2. 2016	***Termín odevzdání:***	2. 6. 2015

Vedoucí práce: doc. Ing. Petr Fiedler, Ph.D.

Konzultanti bakalářské práce:

doc. Ing. Jiří Háze, Ph.D.
Předseda oborové rady

UPOZORNĚNÍ:

ABSTRAKT

Tato práce se zabývá programovacími jazyky programovatelných automatů v systémech Heidenhain a Mitsubishi. Cílem této práce je porovnání jednotlivých systémů po stránce PLC programování a také porovnání diagnostických nástrojů pro programátora v jednotlivých systémech. Dále pak porovnáním vývojového prostředí.

ABSTRACT

This work is dealing with programmable logic controller programing in Heidenhain and Mitsubishi systems. Primary aim of this work is to compare individually PLC programing and comparing diagnostic tools for programmer in both systems. Then this work is dealing with programing environment.

KLÍČOVÁ SLOVA

PLC, Heidenhain iTNC530, Mitsubishi M700V/M70V, programovatelný automat, programovací jazyky, IEC 1131

KEYWORDS

PLC, Heidenhain iTNC530, Mitsubishi M700V/M70V , programmable automatic machine, programming languages, IEC 1131

Noga Kamil. Porovnání programovacích jazyků PLC. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií. Ústav mikroelektroniky 2016. 57 s., 0 s. příloh. Bakalářská práce. Vedoucí práce: doc. Ing. Petr Fiedler, Ph.D.

Prohlášení

Prohlašuji, že svou bakalářskou práci na téma „Porovnání programovacích jazyků“ 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 osobních a/nebo majetkových a jsem si plně vědom následků porušení stanovení § 11 a následujícího zákona č. 121/200 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších přepisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. Díl 4 Trestního zákoníku č. 40/2009 Sb.

V Brně dne……. ……..
Podpis autora

Poděkování

Děkuji vedoucímu bakalářské práce Doc. Ing. Petrovi Fiedlerovi, Ph.D. za účinnou metodickou, pedagogickou a odbornou pomoc a další cenné rady při zpracování mé bakalářské práce. Dále děkuji za odborné rady v oblasti systémů Heidenhain a Mitsubishi panu Ing. Petrovi Olšákovi, panu Luboši Prchlíkovi a panu Zbyňku Kostkovi. A v neposlední řadě děkuji za psychickou podporu své rodině a také panu Ing. Pavlu Rakovi.

V Brně dne……. ……..
Podpis autora

Obsah

1 Úvod

Tato práce se zabývá tvorbou základních PLC programů v systémech Heidenhain iTNC 530 a Mitsubishi M700V/M70V. Jedná se o dva významné výrobce elektroniky a měřících technologií s naprosto odlišným typem programování, pocházejících z rozdílných kontinentů světa. Tyto systémy se dnes používají po celém světě pro provoz přesných obráběcích strojů.

PLC program je základem každého tohoto stroje a ovládá veškeré periferie stroje. Ať už se jedná přímo o motory pohonů, vřetenový motor či periferie stroje jako jsou hydraulický agregát nebo čerpadla a chladiče provozních kapalin.

Cílem práce je porovnat složitost programů, pohodlnost programování, programovací software a v neposlední řadě výkon daného systému. V systému Heidenhain budou psány programy za použití softwaru PLC design a to textovou syntaxí. Naopak systém Mitsubishi bude programován softwarem GX Developer a to ladderovou syntaxí.

2 Stručná historie firmy Heidenhain

Firma byla založena roku 1889, Wilhelmem Heidenhainem, původně vyráběla šablony, štítky, stupnice a měřidla leptáním z kovů. Během 2. Světové války byla firma zničena a poté znovu založena synem původního majitele pod názvem Dr. Johannes Heidenhain. Dr. Johannes Heidenhain, byl žákem Otto Hahna, držitele Nobelovy ceny za chemii.

Firma pod jeho vedením dosáhla jednoho z velkých milníků. Přišli na způsob nanášení velice přesných a odolných struktur chromu kopírováním rastru na materiály jako je například sklo, pojmenované DIADUR. Dosahovali přesnosti až 0,015mm, a to byl základ pro výrobu přesných lineárních snímačů pro obráběcí stroje.

V roce 1968 vydali první obousměrný čítač pro 1 osu a v roce 1976 vyrobili první systém snímající polohu ve 3 osách stroje, TNC 110 a TNC 120.

V roce 1989 vyrobili lineární snímač měřící krok s přesností 1 nm a od roku 1995 používají do strojů synchronně-sériové rozhraní EnDat pro absolutní snímače polohy. Rok na to začali prodávat kompletní vybavení pro obráběcí stroje, což znamená systém TNC 410, motory, měniče atd.

V roce 2004 uvedli na trh Systém iTNC 530 se speciální HSCI architekturou. Tento systém je předmětem bakalářské práce [1]

3 Stručná historie firmy Mitsubishi

Mitsubishi Electric je přední japonská společnost pro výrobu automatů, řídicích systémů motorů a dalších s tímto odvětvím spojených zařízení. Zakladatelem Mitsubishi, společnosti, která se původně zabývala lodní přepravou založenou roku 1870, je muž jménem Yataro Iwasaki. Mitsubishi Electric Corporation bylo založeno roku 1921. Původní firma Mitsubishi Shipbuilding Co. (dnes Mitsubishi Heavy Industries, Ltd.) postavila továrnu ve městě Kobe a specializovala se na výrobu motorů pro oceánské lodě. Během prvního roku uvedla na trh hit, kterým byl elektrický ventilátor, vyráběný v masové produkci. Během deseti let získali několik kontraktů a jedním z nich byl kontrakt na rozvodnou stanici železniční tratě.

V roce 1930 začalo Mitsubishi Electric vyrábět a instalovat výtahy stejně jako generátory napětí. Mitsubishi Electric se během 60. let vypracoval na nejrychleji se rozvíjející a inovující společnost v Japonsku. Již na počátku 60. let začali otáčet pozornost k obrábění.

Během dalších uplynulých dvou desetiletích se Mitsubishi Electric stabilizovalo a zasáhlo do vývoje a výroby klimatizací, počítačů, automobilového průmyslu, satelitů, fotovoltaiky a nukleární energie. Firma Mitsubishi Electric stojí mimo jiné za zrodem první velkoplošné LED obrazovky, největší CRT televize či za nejrychlejším eskalátorem na světě.

V roce 1961 uvedla společnost na trh první systémy s názvem MELDAS 5230 a MELDAS 4200 s tranzistorovou interpolací. V roce 1981 již vyvinuli první CNC soustruh s interaktivním automatickým řízením a v roce 1986 vydali systém MELDAS 300 series CNC zahrnující 32 bitový procesor jako první na světě.[2]

Dalším milníkem je již světově uznávaný systém MELDAS 600 series. Hojně používaný systém s integrovaným připojením do sítě a inteligentními servomotory.

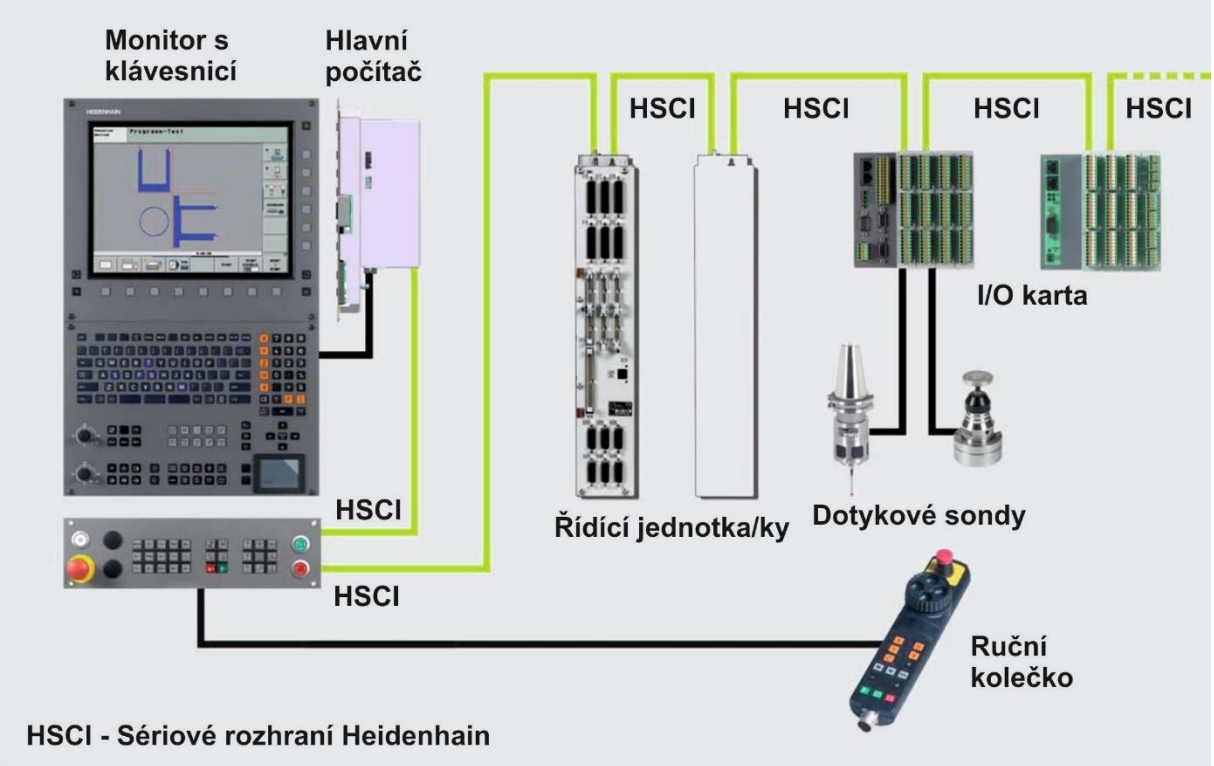
V roce 2003 začali prodávat systém Mitsubishi CNC 700 series, jež je obsahem této práce.

4 Architektura systému Heidenhain iTNC 530

Základním řídícím prvkem systému Heidenhain iTNC 530 je samozřejmě počítač. Obvykle se jedná o sestavu s procesorem Pentium 1,8 GHz, 1 GB RAM a základními periferiemi jako je například monitor (obvykle 17 palců), USB, síťová karta atd. Další periferií je klávesnice, které firma Heidenhain vyrábí na míru ke svým systémům a to se speciálními ovládacími tlačítky. Sestava má svůj vlastní pevný disk o velikosti původně 2 GB, dnes již podporuje větší kapacitu disků, případně lze v systému použít moderní SSD disky.

Ovládací část stroje se skládá z řídící jednotky „Controller Computer“ na které se nachází konektory pro řízení motorů, rotační enkodéry motorů, či lineární snímače polohy EnDAT. K této jednotce se připojí měniče napětí pro jednotlivé motory. Poslední a nepostradatelnou součástí je jednotka pro připojení vstupů a výstupů stroje. Od roku 2011 dodává firma Heidenhain systémy, ve kterých se všechny periferie propojují sběrnicí HSCI. Oproti sběrnici ProfiBusu, který zde působil dříve je toto připojení mnohem jednodušší, jedná se o síťové konektory a fungují na principu „PLUG&PLAY“. [3]

V hlavním počítači (viz obr. 4.1)se však skrývá ještě jedna speciální součástka. Jde o zařízení podobné flash kartě, na které je zaznamenána daná verze systému a příplatkové funkce. Je to tedy nejdražší část systému.



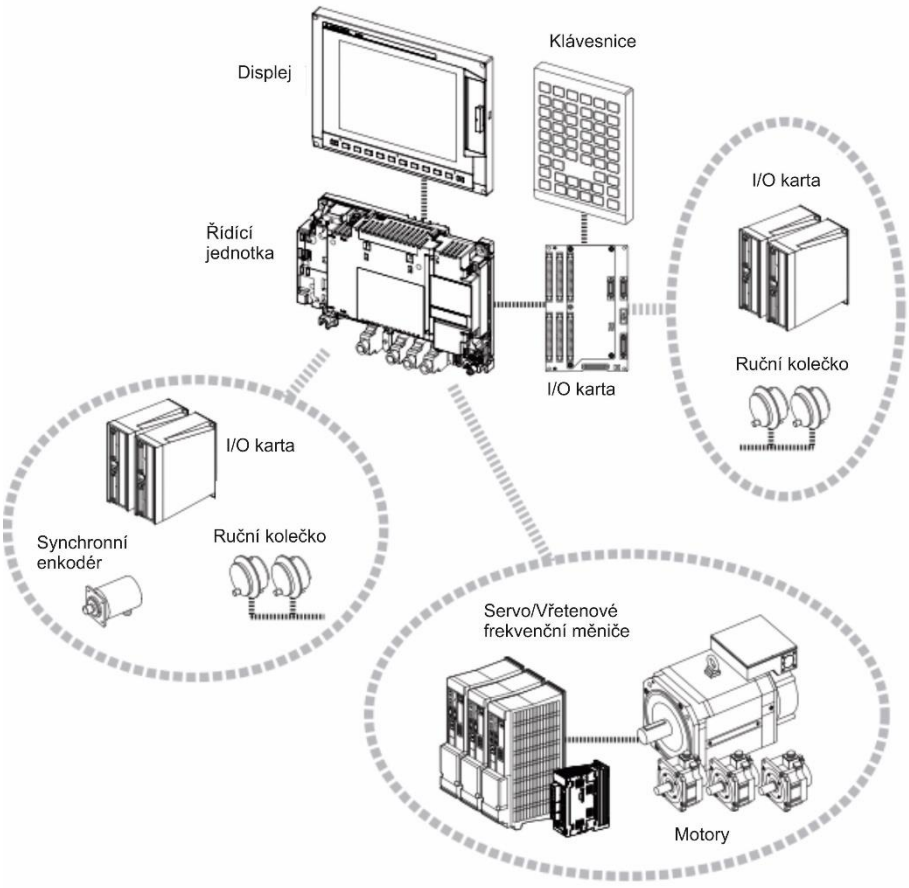
Obr. 4.1: Architektura systému Heidenhain [3]

5 Architektura systému Mitsubishi M70/M700

Principiální zapojení systému Mitsubishi M70/M700 je velmi podobné jako v předchozí kapitole. Základem je opět počítač, který se ukrývá za displejem stroje. K tomuto počítači se připojují periferie jako je klávesnice, vstupní/výstupní karta a zesilovače se servo motory. Tyto zesilovače jsou propojeny vysokorychlostními optickými propojkami. V tomto systému lze tedy plynule ovládat až 8 os.

Společnost Mitsubishi Electric si však sama vyvíjí a vyrábí procesory a další počítačové komponenty. Z tohoto důvodu nelze získat jejich přesné parametry, neboť jsou utajeny. Systém M700V se však pyšní speciálními parametry jako je rychlost zpracování NC programu, která je 168000 bloků za minutu. Zpracování PLC programu má v systému Mitsubishi svůj vlastní procesor a rychlost zpracován instrukcí je tedy 100 kroků za 1 µs. Mitsubishi udává výkon PLC ještě hodnotou nazývanou PCMIX, která se pohybuje okolo 17 (jedná se o průměrný počet instrukcí za 1 µs). [4]

Procesory v systému Mitsubishi pracují s 64 bitovou technologií RISC. Rozlišení posuvu ve všech ovládaných osách je až 1 nm a maximální počet ovládaných os je 16 a to na dvou kanálech. Maximální programovací kapacita je 2000 KB což je ekvivalentem cca 5120m děrné pásky.



Obr. 5.1: Architektura systému Mitsubishi [4]

6 Norma IEC ČSN EN 61 131-3 (IEC 1131-3)

Norma IEC ČSN EN 61 131-3 je jednou ze sedmi přijatých norem, zabývajících se programovatelnými řídícími jednotkami, neboli PLC. [5]

V této normě se pojednává o programovacích jazycích, které můžeme obecně zařadit do pěti skupin:

LD	-	**Ladder Diagram**	Schéma reléové logiky (žebřík)
FBD	-	**Function block Diagram**	Diagram funkčních bloků
IL	-	**Instruction List**	Instrukční sada
ST	-	**Structured text**	Strukturovaný text
SFC	-	**Sequential Function chart**	Diagram sekvenčních funkcí

Tato práce se bude zabývat především dvěma typy jazyků. V systému Heidenhain se programuje strukturovaným textem a v systému Mitsubishi se programuje jazykem Ladder. V případě systému Mitsubishi lze ještě Ladder Diagram přepnout do Instrukční sady, kterou se však tato práce nezabývá.

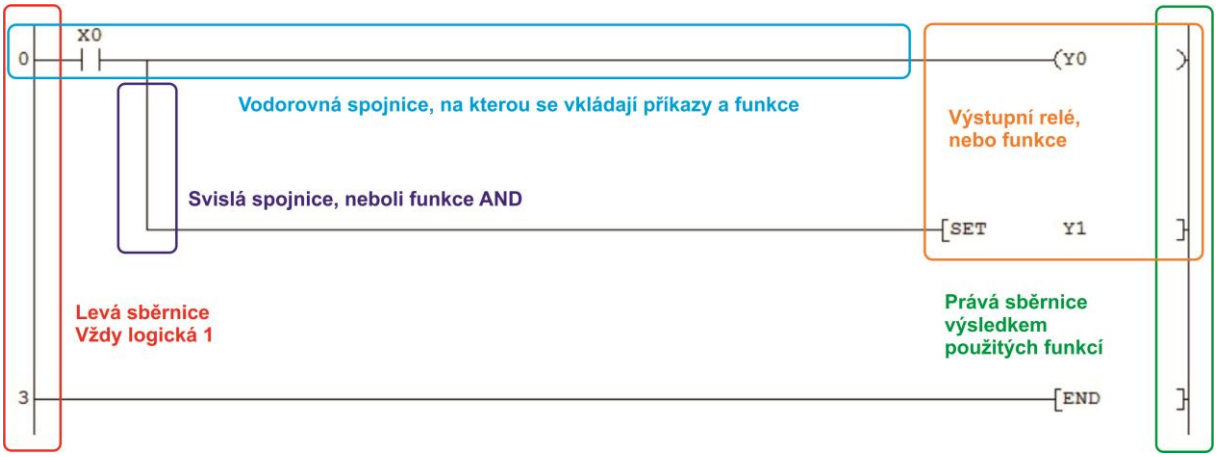
6.1 Structured Text

Strukturovaný text je typ vyššího programovacího jazyku. Vychází ze světoznámých jazyků jako například PASCAL, C a podobných. Tento jazyk je objektově orientován a obsahuje větvící funkce, jako jsou například funkce IF a funkce CASE. Vzhled takového stylu, konkrétně v systému Heidenhain je následující:

```
L               I_vstup             ; Načti

Funkce, příkaz  Název proměnné      ; Komentář
```

6.2 Ladder Diagram

Jedná se o grafický programovací jazyk, původem z USA. Označení „grafický“ vyplývá ze vzhledu programu. Tuto syntaxi si lze představit jako elektrické schéma, ve kterém je použito relátek jako výstupních prvků. Každé takové schéma má specifické rysy. Na pravé a levé straně se nachází linky, které je možné specifikovat jako napájecí sběrnice. Mezi tyto dvě linky jsou vloženy dané spínače a relátka, spojená příčnou čarou (spojnicí) a celé toto uskupení je možné označit jako řádek (viz obr. 6.1). Pak se v závislosti na stavu těchto zařízení přesouvá informace mezi jednotlivými zařízeními a řádky programu. Levá sběrnice má vždy hodnotu „1“, kdežto pravá má proměnlivý stav.



Obr. 6.1: Popis struktury Ladder Diagram, předloha ze systému Mitsubishi

Spojnici lze dále rozdělit na svislou a vodorovnou. Vodorovnou spojnicí se vytvoří mezi jednotlivými instrukcemi funkce logického součtu (OR), kdežto svislou spojnicí se vytvoří funkce logického součinu (AND). [6]

.

7 Úvod do programování v systému Heidenhain

Nyní lze přejít k programování v systému Heidenhain. Tento systém je rozdělen do tří hlavních částí.

- **TNC**
- **PLC**
- **SYS**

Z pohledu programátora jsou to vlastně tři pomyslná disková úložiště. Disk je z výroby rozdělen na tři partitiony. Část TNC slouží pro obsluhu. Obsluha stroje si do něj ukládá své obráběcí programy, obráběcí cykly, případně objekty pro hlídání kolizí atd.

Na dalším disku se nachází část PLC, je to část, která už podle názvu napovídá, že jde o jádro systému, ke kterému má programátor přístup a podle kterého celý stroj pracuje. Do tohoto systému se programátor dostane použitím 6 místného číselného hesla, které je z výroby pro všechny systémy od firmy Heidenhain stejné. Nicméně v případě nutnosti si jej programátor může změnit. Záměrně zde bylo řečeno slovo „programátor“ a to z důvodu, že obsluha má přístup do PLC stroje zakázán.

Poslední částí je SYS. Do této části se již za běžných okolností nedostane ani programátor. V této části se ukládají pro stroj životně důležité soubory a data. Do této sekce lze vstoupit pouze se speciálním heslem, které je vytvořeno algoritmem, který zná pouze firma Heidenhain. Tento algoritmus tedy vytvoří zvláštní heslo, které je funkční jen na daném stroji a pouze v předem určený datum.

Pro programátora je tedy zásadní část PLC. Zadáním hesla vstoupí do systému, kde se nachází několik důležitých funkcí:

7.1 TABLE

Zde je výpis veškerých vstupů/výstupů, proměnných, časovačů a čítačů. Tyto hodnoty zde mění hodnotu v reálném čase, což usnadňuje programování či diagnostiku problému.

7.2 I/O FORCE LIST

Již z názvu je patrné o jakou funkci se jedná. Do tohoto listu lze vložit kteroukoli proměnnou či vstup/výstup a natvrdo k ní přiřadit potřebnou hodnotu 1 nebo 0. Po přiřazení již nebude mít stroj možnost tuto hodnotu měnit (až do vypnutí funkce force listu).

7.3 TRACE IN-CODE

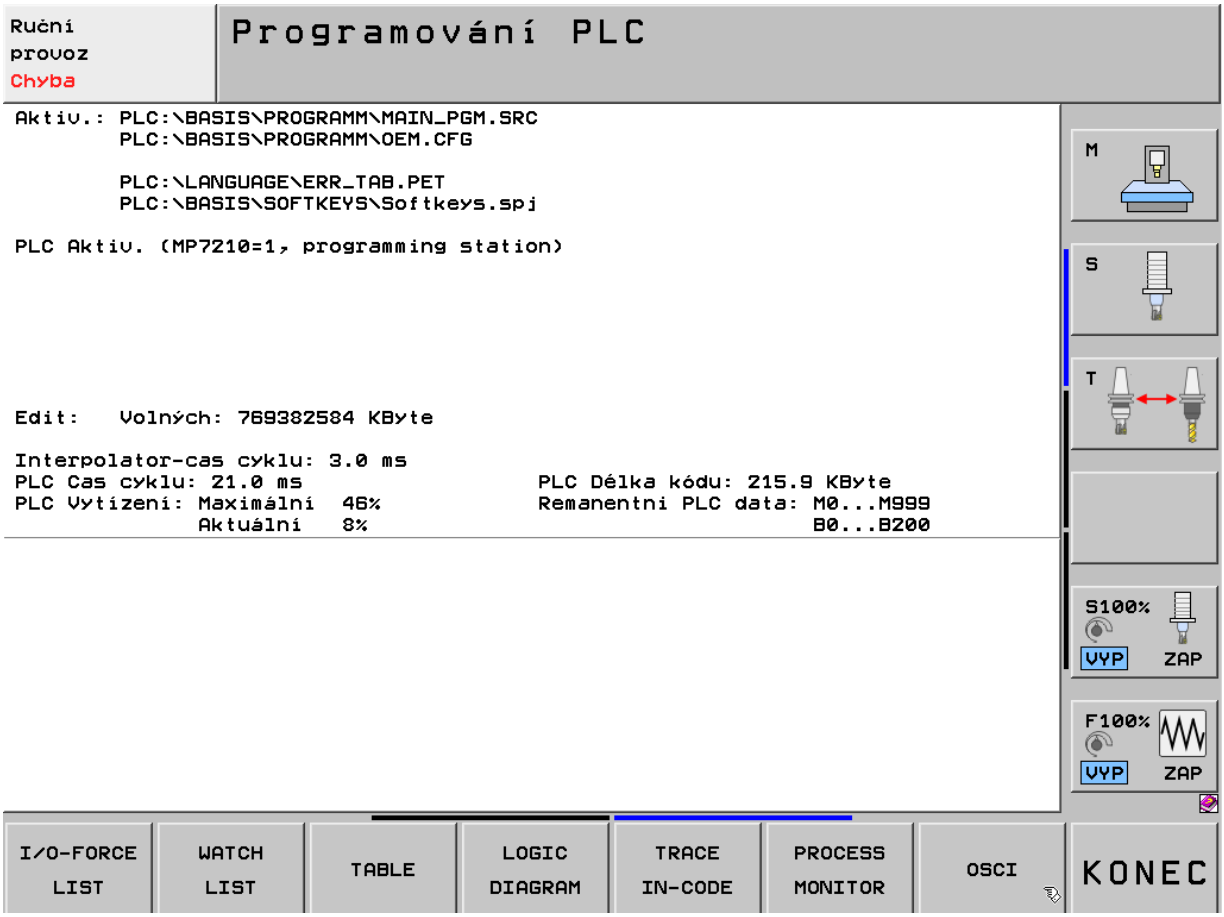
Jakmile je vytvořen nějaký program, je nutné jej vyzkoušet. Tato funkce dovoluje v reálném čase sledovat průběh programu. Jelikož PLC cyklus pracuje vždy od shora dolů, je zde dobře vidět jak se daný program chová.

7.4 COMPILE

Jelikož je možnost měnit program PLC na počítači, ale i přímo na stroji, nachází se zde funkce pro kompilaci zdrojového kódu. Tuto funkci je nutné použít po každé změně ve zdrojovém kódu, jinak stroj nebude možné uvézt do provozu. Dále tato funkce zkontroluje celý kód a vypíše chyby, kterých se programátor při psaní programu dopustil.

7.5 OSCI

Velikou výhodou systému Heidenhain je bezesporu integrovaný osciloskop. Tento osciloskop umožňuje sledovat chování programových funkcí, ale i nespočet elektrický veličin. Například se zde dá měřit signál z lineárního snímače, či proudy motorů atd.



Obr. 7.1: Menu Programování PLC

7.6 Popis obrazovky PLC

Kromě spodních softwarových tlačítek, tedy funkcí, které lze použít k diagnostice, se zde nachází také výpis aktuálního hlavního programu MAIN_PGM.src a konfiguračního souboru OEM.cfg. Dále je zde projekt softwarových kláves a aktuální soubor s error hláškami ERR_TAB.pet.

Ve střední části obrazovky lze spatřit délku cyklu, vytížení PLC a délku kódu. Zde již v KBytech a ne v délce děrného pásku jak je dnes ještě stále zvykem u jiných výrobců.

8 Popis proměnných v systému Heidenhain

Pro pochopení programování je důležité popsat alespoň základní proměnné a logické funkce. Proměnnými jsou myšleny všechny vstupní/výstupní a paměťové buňky. Lze je rozdělit na bitové proměnné a registry:

8.1 Bitové proměnné

Pod bitovými proměnnými si lze představit proměnné, které mohou nabývat pouze hodnoty logická 0 a 1. Někdy jinak řečeno true nebo false. Tyto proměnné jsou v systému Heidenhain popsány písmeny:

I - Input Jsou vstupní proměnné. Jedná se o signály z periferií stroje, například tlačítka anebo různé senzory jako jsou mikrospínače, indukční snímače atd.

O - Output Jsou výstupní proměnné, které naopak vydávají příkaz pro sepnutí či rozepnutí periferií stroje jako jsou například čerpadla, motory nebo signální osvětlení atd.

M - Marker Je bitová paměťová buňka. V systému Heidenhain můžou být markery rozděleny na dva typy a to globální a lokální. Globální markery je nutné zavádět do speciálního definičního souboru, ale za to je možné je použít ve všech podprogramech.

Lokální markery stačí zavádět pouze na začátku jednotlivých podprogramů, ale jejich platnost je omezena pouze na daný podprogram. Tuto hodnotu tedy nelze přenášet do jiných podprogramů.

T - Timer Jde o časovač. Aby to nebylo snadné, tak časovače jsou rozděleny na dva typy. První jsou pozůstatkem starých verzí systému Heidenhain a je jich k dispozici 48. V programu jsou značeny písmeny TS a TR. Jde o Timer start a Timer run. Jejich hodnota je nastavena ve strojních parametrech. Druhým typem časovačů jsou PLC timery, zavedené i s hodnotou v definičních souborech.

C - Counter Jako poslední jsou k dispozici countery, jinak řečeno čítače. Čítačů je 48, ale každý je dělen na 3 různé proměnné. První část v programu čítač aktivuje a udává, kolik cyklů bude čítač aktivní. Druhá část oznamuje, zda byl již dosažen žádaný počet impulsů a třetí čítač resetuje.

8.2 Registry

Jsou to proměnné, které nabývají libovolnou celočíselnou hodnotu. Systém pracuje i s desetinnými čísly, ty ale převádí na celá čísla (odstraní desetinnou čárku). K dispozici je typ Byte (**B**), Word (**W**) a Double (**D**). Přičemž byte je 8 bitová hodnota, word 16 bitová a double 32 bitová informace. Do proměnné typu Double lze tedy vložit hodnotu - 2 147 483 648 až 2 147 483 647.

Nakonec se zde nachází poslední proměnná, kterou je textový řetězec String (**S**), do které se vkládá text.

9 Základní logické funkce PLC v systému Heidenhain

V systému Heidenhain je nepřeberné množství funkcí, kterými se dá programovat. Z toho vyplývají obrovské možnosti pro programátory. V tomto odstavci budou popsány jen základní z nich. Za prvé by jich bylo obrovské množství a za druhé jsou to obvykle rozšířené základní funkce pro snadnější programování.

L - Load Je funkce načtení proměnné. Tedy pokud se v dané bitové proměnné nachází stav 1, je podmínka splněna.

A - And Neboli logický součin. Jsou-li dvě podmínky a obě pravdivé, je podmínka splněna.

OR - Or Neboli logický součet. Jsou-li dvě podmínky a alespoň jedna z nich pravdivá, je podmínka splněna.

S - Set Je příkazová funkce a nastavuje příslušnou bitovou proměnnou do hodnoty 1. Tato hodnota se v proměnné uchová i po přerušení podmínky. Proto ji je nutno pro návrat na do původního stavu resetovat.

R - Reset Je taktéž příkazová funkce, která navrací původní stav bitové proměnné.

= - Rovnítko Je funkce podobající se funkci Set, ale s tím rozdílem že bitová hodnota je platná pouze pokud platí podmínka, pokud podmínka přestane platit, proměnná se vrátí do původního stavu, a to i bez použití Reset funkce. V případě počítání s registry se toto znaménko používá pro získání výsledku.

N - Not Je funkce negace. Připojuje se například k předešlým funkcím, ze kterých tedy vyplývají příkazy jako např. LN (Load not), AN (And not) atd.

IF - If Je obecně známá podmínková funkce. V tomto případě, jsou-li tedy splněny všechny podmínky, provede nějakou operaci. Pokud však podmínky nejsou splněny, lze použít příkaz ELSE, který provede odlišnou operaci.

10 Základní PLC programy Heidenhain

Nejjednodušší cestou, jak vysvětlit programování je pomocí příkladu. Je zde uvedeno několik typů programů Heidenhain. Jsou to základní programy, které lze nalézt asi v každém obráběcím centru.

10.1 Tlačítkové spínání

V každém systému je zapotřebí spouštět vnější periferie stroje. Obsluha stroje má v tomto případě dvě možnosti. Sepnout periferii tlačítkem nebo M funkcí, což je softwarový spínač.

```
#define /s    ML_I_ tlacitko_zmacknuto    ; Definice lokální proměnné jako
M                                         Markeru

;----------------------------             ; Název podprogramu
LBL spínač
;----------------------------
L          I_tlacitko                     ; Pokud je na vstupu 1 (signál
                                          z tlačítka)
AN         ML_I_tlacitko_zmacknuto        ; A zároveň má marker ML hodnotu
                                          0
S          O_Svetlo                       ; Rozsviť světlo

LN         I_tlacitko                     ; Pokud vstupní hodnota 0
A          O_Svetlo                       ; A svítí světlo
S          ML_I_ tlacitko_zmacknuto       ; Nastav marker ML na hodnotu 1

L          I_tlacitko                     ; Pokud je na vstupu 1
A          ML_I_ tlacitko_zmacknuto       ; A zároveň je v markeru ML
                                          hodnota 1
R          O_Svetlo                       ; Zhasni světlo

LN         I_tlacitko                     ; Pokud je na vstupu 0
AN         O_Svetlo                       ; A nesvítí světlo
R          ML_I_tlacitko_zmacknuto        ; Resetuj marker ML do původní
                                          hodnoty

EM                                        ; Ukonči podprogram
```

Nechť je I_tlacitko vstupní signál z mechanického tlačítka. Úkolem programu je v závislosti na stisku tlačítka spínat výstupní relé, například světla. V tomto případě se jedná o jednoduchý spínač, nikoli přepínač. Proto je nutné do programu napsat pomocné markery pro vytvoření pulsu.

10.2 M funkce

Jak již bylo řečeno, jde o softwarový spínač. To znamená spínač, který je možné zahrnout do obráběcího programu a to bez zásahu obsluhy (není zapotřebí mačkat tlačítko). Tyto funkce se využívají pro spouštění otáček vřeten, pro zastavení nebo ukončení programu, či například pro spuštění chlazení atd. První asi dvě desítky M funkcí jsou stejné ve všech známějších systémech jako je Mitsubishi, Fanuc, Siemens, Heidenhain. Přehled některých standardních M funkcí:

M01	Programový stop
M03	Pravotočivé otáčky vřetene
M04	Levotočivé otáčky vřetene
M05	Vypnutí otáček vřetene
M07	Chlazení středem vřetene
M08	Oplachy třísek
M19	Orientace vřetene (přesná poloha vřetene například pro výměnu nástroje)

Program pro vytvoření takové m funkce může vypadat následovně:

```
;----------------------------
LBL  M funkce
;----------------------------

L           MG_M10_Zapni            ; Pokud má globální marker hodnotu 1
                                    (puls)
S           O_Svetlo                ; Rozsviť světlo

L           MG_M11_Vypni            ; Pokud má globální marker hodnotu 1
                                    (puls)
R           O_Svetlo                ; Zhasni světlo

EM
```

Zde je vidět, že programování M funkce je o mnoho snazší a to především z důvodu, že M funkce se obvykle vypínají jinou M funkcí. V tomto případě se tedy světlo rozsvítí funkcí M10 a zhasne funkcí M11. Na druhou stranu je zde použit globální marker MG, který je nutné zavést do definičního souboru a to proto, aby jeho hodnota byla dostupná ve všech programech. Tento definiční soubor obvykle nese název typu GLBTCMB.def a shromažďuje veškeré paměťové proměnné, které lze posléze sledovat ve funkci TABLE. Pro úplnost je nutno doplnit, že v definičním souboru GLBIO.def jsou stejným způsobem zavedeny všechny vstupy a výstupy stroje.

10.3 Timer (časovač)

Dalšími velmi užitečnými typy podprogramů jsou práce s časovači neboli Timery. Pokud je zapotřebí pracovat se zpožděným sepnutím, či kontrolovat, jestli se nějaká operace provede v daném čase a mnoho dalších funkcí, lze použít časovače. Jak bylo napsáno výše, timery jsou v systému Heidenhain rozděleny na dva druhy. Zde bude popsán první případ:

TS	Timer start	Startovací puls
TR	Timer run	Proměnná, která pracuje s časem nastaveným v parametrech

```
;-------------------------
LBL Timer
;-------------------------
L             I_tlacitko              ; Pokud je na vstupu puls logická 1
S             TS_spust_casovac        ; Odstartuj časovač

L             TS_spust_casovac        ; Pokud má časovač TS hodnotu 1
O             TR_odpocitavani         ; Anebo časovač TR má hodnotu 1
=             O_Svetlo                ; Rozsviť světlo

EM
```

V tomto případě bude světlo svítit po dobu, kterou máme nastavenou ve strojních parametrech systému (MP 4110.0 – 4110.47) + délka jednoho cyklu skenu PLC. To znamená, pokud je čas nastaven na 10 vteřin, bude přibližný čas sepnutého světla 10,021 s. To je dáno tím, že do podmínek byl přidán startovací Timer TS. Pokud by byl odebrán z podmínek, systém by světlo spustil až v následujícím cyklu ale na dobu přesně 10 s.

```
L             TR_odpočítávání         ; Pokud běží časovač TR
=             O_Světlo                ; Rozsviť světlo
```

Bohužel, v tom případě by se ale zbytek prvního skenu PLC provedl bez sepnutého výstupu a programátoři by mohli mít velký problém v podobě vzniklého varovného hlášení, které by navazovalo na daný výstup dále v programu.

Při práci s časovači se vyskytuje jeden drobný problém. Časovač, jak je popsáno ve vzorovém programu, je nutno spouštět pouze pulzem. V případě, že by vstupní hodnota byla úroveň, časovač by se v každém cyklu resetoval.

10.4 Counter (čítač)

Čítač je posledním zástupcem základních programů. V některých systémech je zapotřebí čítat impulsy a na základě toho spínat nějaký příslušný výstup. Využití této funkce však není moc časté.

V systému Heidenhain je k dispozici 48 čítačů. Každý čítač je rozdělen na tři různé proměnné a to:

CS	**Counter start**	Start čítače
CR	**Counter reset**	Změna bitu po doběhnutí čítače
CD	**Counter release**	Aktivace čítače

Program začíná proměnnou CS, neboli start a načtení hodnoty impulsů, která se nachází ve strojním parametru MP4120.0 – MP4120.47. Poté, pokud je aktivní proměnná CD, probíhá čítání impulsů v proměnné CR, která má hodnotu logická 1. Jakmile se dosáhne požadovaného počtu impulsů, proměnná CR se překlopí do hodnoty logická 0.

```
;-------------------------
LBL Counter
;-------------------------
L         I_Spousteci_impuls        ; Proběhne-li impuls
=         CS_citac_pulzu            ; Načti počet impulsů

L         I_MG_control_operational  ; Pokud je globální marker aktivní
=         CD_citac_pulzu            ; Čítač pracuje

LN        CR_citac_pulzu            ; Jakmile doběhne čítač
A         CD_citac_pulzu            ; A zároveň čítač pracuje
=         O_Svetlo                  ; Rozsviť světlo

EM
```

Tyto čítače opět jako proměnné typu Timer je nutné deklarovat jako globální proměnnou v definičním souboru GLB_TCMB.def.

V tuto chvíli je program téměř popsán, jen je nutno vysvětlit vstupní signál I_MG_Control_operational. Jedná se o předdefinovaný signál společností Heidenhain. Jeho hodnota nabývá logickou 1 v případě, že se na systému nachází řídící napětí. To znamená, že je systém aktivní. Pokud je systém vypnut, například nouzovým tlačítkem, hodnota této proměnné se překlopí do stavu logická 0.

11 Úvod do programování v systému Mitsubishi

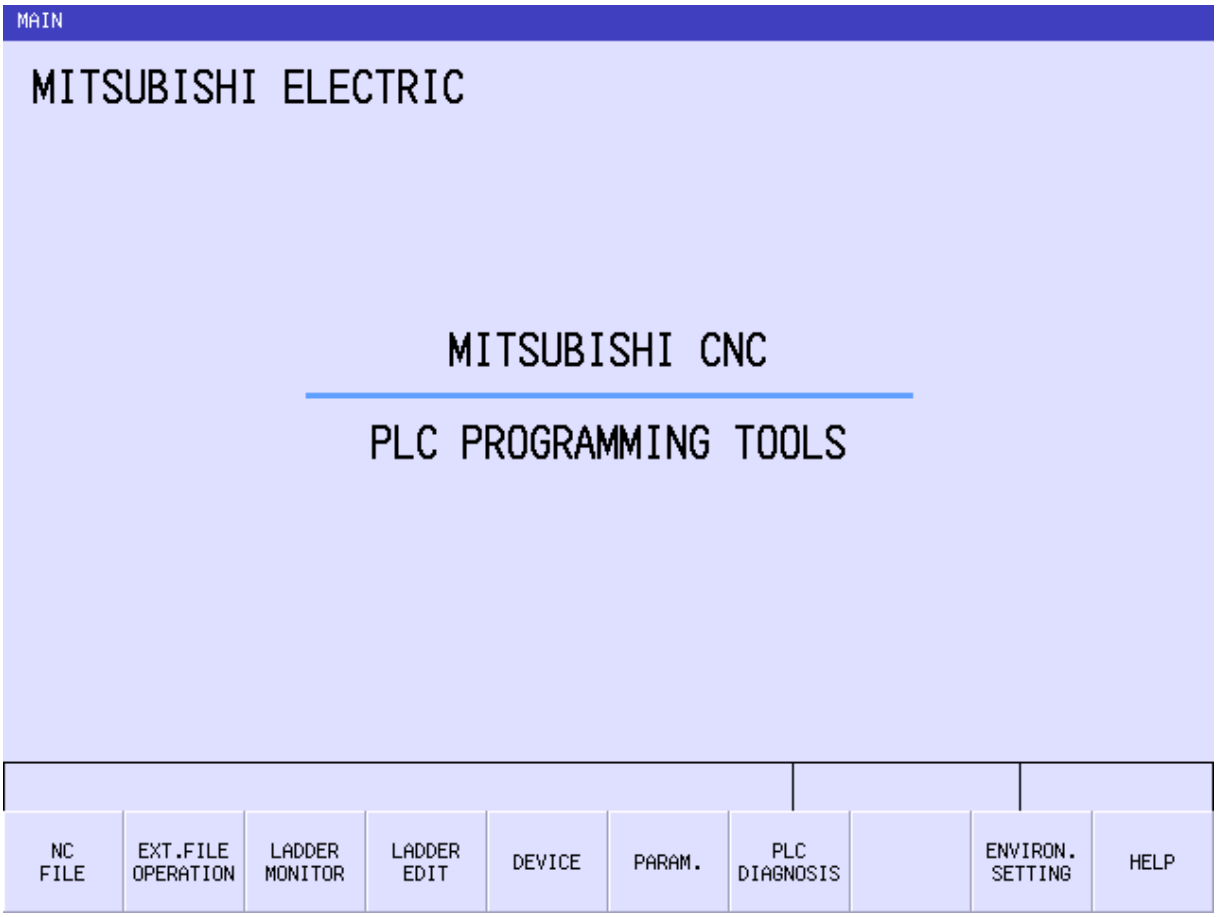
Systém Mitsubishi lze z pohledu programátora rozdělit na dvě základní části:

- PLC
- CNC

PLC je část, která je celá otevřená pro programátora. Zde se vkládají veškeré instrukce, volání podprogramů či výpočty.

Co se CNC části týče, jedná se o program, který je z výroby nahrán v každém systému. V tomto programu se nachází speciální funkce, registry a bitové proměnné. Díky těmto funkcím je programování o mnoho jednodušší. V podstatě je část programu hotová. Například není třeba programovat otáčky vřetene a jejich kontrolu. Vše je přichystáno, jen je zapotřebí vyplnit potřebné parametry vřetene a typ zesilovače.

CNC systém má také své přichystané vstupní a výstupní proměnné. Fungují zvláštním způsobem, a to tak, že výstup z části CNC je vlastně vstupem do PLC. To samé funguje zase naopak, tedy že výstup z části PLC je vstupem do CNC části (Například výstup Y188E provede orientaci vřetene a vstup X188E potvrzuje, že operace úspěšně proběhla). Je to sice krkolomnější představa, nicméně, není zapotřebí otrocky psát program, stačí použít jen danou proměnnou ze seznamu.



Obr. 11.1: Obrazovka PLC programování v systému Mitsubishi

Na obrázku (viz obr. 11.1) lze vidět několik základních funkcí PLC, některé z nich budou popsány:

11.1 NC FILE

Ikona sloužící pro otevření a načtení požadovaného programu. Program je pak možné sledovat, případně editovat.

11.2 LADDER MONITOR

Pod touto ikonou se schovává mód nazývaný monitor. Zde si programátor může v reálném čase prohlížet funkci PLC programu. Každá proměnná má zobrazenou svou aktuální hodnotu, a to dělá diagnostiku špatné funkce programu o mnoho snadnější. Nicméně, systémy Mitsubishi obvykle obsahují pouze 8 – 10 ti palcové monitory, na kterých je program celkem nepřehledný. Z tohoto důvodu je mnohem pohodlnější spustit stejnou funkci ve vývojovém prostředí GX Developer (viz kapitola č. 21).

11.3 LADDER EDIT

Zde se nachází prostředí umožňující provádět změny v programu. Ale jako v předešlém bodě je práce v tomto prostředí náročná a vhodná pouze pro minimální úpravy.

11.4 DEVICE

Jedná se o vyhledávací zařízení. Je zde možno procházet veškeré registry i bitové proměnné a v reálném čase sledovat jejich aktuální obsah, nebo tento obsah přepisovat.

11.5 PLC DIAGNOSIS

Leč je to honosný název, pod touto ikonou se skrývá pouze stav PLC. Při nahrávání jakéhokoli nového programu do systému je nutno PLC zastavit, nahrát do ROM systému a poté znovu nastartovat PLC. Z tohoto důvodu je zapotřebí tento stav PLC sledovat.

12 Popis proměnných v systému Mitsubishi

Proměnné v systému Mitsubishi lze rodělit stejně jako v systému Heidenhain. To znamená na bitové proměnné a registry.

12.1 Bitové proměnné

Opět se jedná o proměnné, které nabývají pouze hodnot 0 a 1.

X	**Vstupní proměnná**	Jde o vstupní proměnnou, pod kterou si lze představit signály vstupující do systému. Takže například tlačítka, senzory tlaku, a v případě Mitsubishi lze jako X dosadit výstupní signál z CNC.
Y	**Výstupní proměnná**	Opět se jedná o výstupní proměnnou, kterou si lze představit jako cívku relé.
M	**Temporary memory**	Jedná se o paměťovou buňku, která však ztrácí svou hodnotu, pokud je vypnuto napájení daného systému. Za předpokladu, že je zapotřebí tuto bitovou informaci uchovat i po vypnutí systému, dá se použít proměnná **L – Latch Relay**.
T	**Timer**	Jedná se o časovače, opět rozděleny na dvě skupiny. Časovače, které pracují s časem v parametrech anebo s časem nastaveným přímo v PLC..
C	**Counter**	Zde se jedná o čítač impulsů.

12.2 Registry

K - Constant Jednoduše přeloženo konstanta. Konstanta se v systému používá ve funkcích, jako pevné číslo. To znamená, že není označena žádným číslem a nedá se na ni odvolat z jiné části programu. Je použitelná pouze pro danou funkci a nabývá pouze hodnoty k ní na pevno přiřazené.

H - Hexadecimal constant Totéž jako K, liší se pouze hexadecimálním zápisem čísla.

D - Data Register Jedná se o datové registry pro programátory. S těmito registry pracuje programátor a používá je pro práci s 16 bitovými čísly. Zvláštností je tady možnost použít pro funkce také 32 bitový registr a to tak, že jsou spojeny dva registry dohromady a do programu je vepsán každý druhý (vždy první z dvojice).

R - File Register Opět se jedná o registry. Některé z těchto registrů však mají předem danou funkci z výroby. Řada z nich si uchová paměť i po vypnutí napájení (konkrétně R9800-R9899).

Pro pochopení nám popis těchto základních proměnných bude postačovat. Za zmínku však stojí si připomenout speciální proměnné. Mitsubishi PLC má sadu proměnných **SM – Special relay** a **SD – Special register**. Jedná se o proměnné, které mají z výroby stanovenou funkci a jsou zavedeny v CNC části systému. Programátor tyto funkce nemůže měnit, pouze s nimi pracovat. Například, pokud v systému dochází kapacita baterie, změní se v registru SD51 první bit z hodnoty logická 0 na 1, a to vede k zobrazení varovné hlášky LOW BATTERY (slabá systémová baterie).

13 Základní PLC programy Mitsubishi

Jelikož se v systému Mitsubishi programuje syntaxí nazývanou Ladder, neboli žebřík (brzy bude jasné proč toto značení), není zapotřebí popisovat logické funkce. Tyto funkce vyplynou přímo z programu. Oproti systému Heidenhain je to celkem příjemná změna, jelikož si programátor tak velké množství funkcí nemusí pamatovat. Ladder si lze představit jako schéma elektrického zapojení. Pro porovnání programovacích jazyků zde budou předvedeny stejné programy jako v systému Heidenhain.

13.1 Tlačítkové Spínání

I v systému Mitsubishi je nutné ovládat vnější periferie stroje. Zde je uveden příklad:

Obr. 13.1: Program pro tlačítkové spínání

Na obrázku (viz obr. 13.1) je vidět jednoduchá tlačítková logika označovaná jako přídržný kontakt.

Pár slov k funkci:

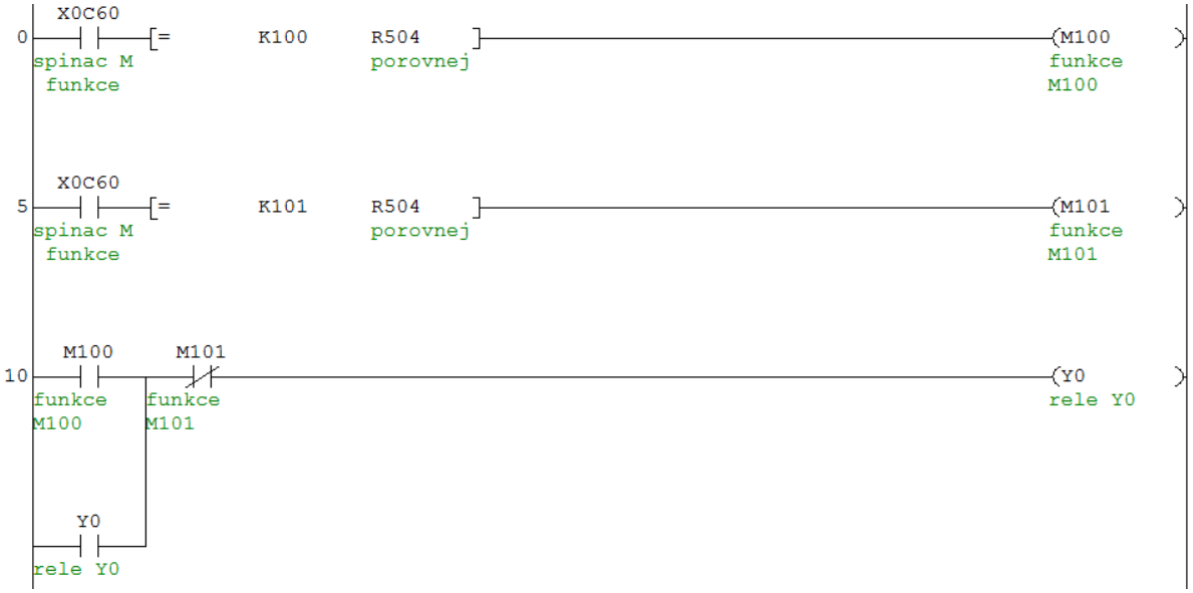
1. Pokud se stiskne tlačítko X000, sepne se výstup Y000 přes pomocný bit M0, jelikož je v klidovém stavu sepnutý.
2. Je sepnutý výstup Y000 a zároveň vstup X000, neboť je stále stisknuté tlačítko. Jakmile je tlačítko uvolněno, pomocný bit M0 se sepne přes Y000 a sepnutý kontakt X000. Proměnné Y000 a M0 jsou tedy ve stavu logická 1, a z toho plyne, že se sepnula libovolná periferie připojená na relé.
3. Nyní se stiskne X000 znovu, a tím se způsobí rozpojení přídržného kontaktu Y000 na prvním řádku, jelikož M0 je ve stavu 1, a tím je kontakt rozpojen. To samé platí pro X000. Dokud je, však stisknuté tlačítko je pomocný bit ve stavu log. 1.

4. Posledním krokem je uvolnění tlačítka, čímže se rozepne kontakt X000 a tudíž i výstup pro pomocný bit M0. Návrat na počáteční stav.

Leč to není úplně patrné, ale tento program je o mnoho jednodušší, než program v systému Heidenhain. Na první pohled si málokdo všimne, že zde není třeba žádné definice proměnných. V Mitsubishi si každý programátor může vzít jakoukoli paměťovou buňku z předem daného rozsahu (která ještě není použita jinde v programu) a začít s ní pracovat. To samé může udělat i se vstupy a výstupy, zde se však musí programátor řídit vstupy, které jsou elektricky napojeny do systému. To znamená, že nemůže použít vstupy X000 – X00F, pokud má do systému zavedeny vstupy X0010 – X 001F. Výběr z daného rozsahu je však libovolný a není třeba ho do programu zavádět, je dán výrobcem a počtem I/O karet.

13.2 M funkce

V systému Mitsubishi se taktéž jako základní softwarové spínače používají M funkce.
V systému Heidenhain se M funkce musela nejdříve zavézt do definičního souboru. Zde se tato funkce zavádí taktéž, ale velmi odlišně. V systému Mitsubishi M700/M70 je zaveden z výroby registr R504, se kterým se porovnává číslo funkce. V tomto případě bude mít funkce číslo 100. To je dáno konstantou K100. Znaménko rovnosti tady udává rovnost mezi konstantou K100 a registrem R504. Pokud je tato rovnost splněna, sepne se funkce M100.



Obr. 13.2: Program M funkce

Potom stačí přidat do programu libovolně funkci M100 jako spínač, který vydá puls a sepne relé Y0. Zde na řadu přichází opět přídržný kontakt Y0, který podrží sepnutý výstup i po skončení impulsu M100. Nakonec se zde nachází funkce M101, v podobě v klidovém stavu sepnutého kontaktu, která se při vyvolání funkce M101 rozpojí a tím přeruší přídržný obvod výstupu Y0.

Pro úplnost je nutné doplnit, že vstupní proměnná X0C60, je výrobcem udaný signál ze systému, který se sepne, pokud je v systému vydán příkaz pro M funkci. V registru R504 se nachází číslo požadované M funkce.

13.3 Timer (časovač)

I s časovači je práce v Mitsubishi o mnoho snadnější. Pokud nejsou použity parametrem nastavené časovače, vypadá program následovně. Funkce je stejná jako v programu pro systém Heidenhain. Pro správné pochopení tohoto programu je nutno dodat, že časovač v následujícím programu počítá, pouze pokud je sepnutá vstupní podmínka. Pokud by toto nebylo splněno, čítač se vynuluje. Za další je nutno vědět, že čítač po dobu počítání vykazuje stav logická 0. Do sepnutého stavu se přepne po doběhnutí času. Konkrétně počítá od 0 až do doby než dosáhne času K 100.

Obr. 13.3: Program časovač

Bližší pohled:

1. Pokud se sepne tlačítko X000, sepne časovač T0 a zároveň výstup Y000.
2. Zde se opět zapracuje jednoduchý přídržný kontakt. Pokud je sepnuto Y000 a čas běží, je sepnut jak časovač, tak výstup Y000.
3. Jakmile doběhne čas, rozpojí se podmínka T0 a přeruší se přídržný kontakt. To znamená, že se časovač vyresetuje a výstup Y000 se rozpojí.

Opět je zde méně práce se zaváděním proměnných. K časovači se taktéž nevztahuje žádný definiční obor. V tomto případě je definován pouze konstantou K100. A jelikož je tento program funkčně stejný, jako pro systém Heidenhain uvedený dříve (viz. Kapitola 10.3) je nutno objasnit, proč má konstanta hodnotu 100 místo 10.
Pokud je časovač ve tvaru jako na obrázku (viz obr. 13.3), je každá jednotka konstanty 100ms. Z toho plyne $T = 100 \times 100ms = 10s$.

Další možností je napsat časovač v tomto tvaru:

Obr. 13.4: Vysokorychlostní časovač

Jedná se o zrychlený časovač, ve kterém se každá jednotka konstanty rovná 10ms. Je to tzv. „High speed“ časovač. V tomto případě bude výsledný čas pouze 1s.

Tento způsob programování je dokonce tak propracovaný, že místo konstanty K lze zadat do proměnné T nějaký registr, např. D0. V tomto případě lze jeho hodnotu libovolně měnit různými operacemi s registry. Takže výsledný čas bude dán nějakým výpočtem.

13.4 Counter (čítač)

Zde je poslední ze základních programů v systému Mitsubishi. Je jím čítač impulsů, a jak je vidět na obrázku (viz obr. 13.5), je velmi jednoduchý.

Obr. 13.5: Program čítač

Na vstupu X000 se objevují impulsy. Každý tento impuls inkrementuje proměnnou C0. V tomto příkladu je potřebný počet impulsů pro sepnutí výstupního relé udán konstantou K, a to 10 impulzů.

V případě proběhnutí desátého impulsu sepne relé. Pokud je však zapotřebí začít počítat impulsy znovu, bude nutné zakomponovat do programu také reset proměnné C0. V případě tohoto programu totiž zůstane relé Y000 ve stavu sepnuto až do doby vypnutí systému.

14 Větvící příkazy v systému Heidenhain

Jak již bylo řečeno, systém Heidenhain se podobá programovacím jazykům jako je například PASCAL. Tím pádem se zde nalézají příkazy typu IF, CASE, WHILE, REPEAT . Nejčastěji užívané jsou bezpochyby příkazy IF a CASE, které budou dále vysvětleny.

V systému Heidenhain se s příkazem IF pracuje odlišně. Příkaz IF je rozdělen na pravdivý nebo nepravdivý. To je značeno buď „IFT (If True)“ anebo „IFF (If False)“. Program pak může vypadat následovně.

```
;-------------------------
LBL Tepelna ochrana svetla
;-------------------------
L           I_Teplotni_senzor         ; Načti proměnnou senzoru (Teplota v
                                      pořádku)
IFT                                   ; Pokud je vstupní proměnná logická 1
L           I_Spinac                  ; Načti spínací signál
S           O_Svetlo                  ; Rozsviť světlo
ELSE                                  ; Pokud je vstupní proměnná logická
L           I_Spinac                  ; Načti spínací signál
S           pn_error_prehrate_svetlo  ; Zobraz alarmové hlášení ,,přehřáté
                                      světlo“
ENDI                                  ; Konec příkazu IFT
EM                                    ; Konec programu
```

Tento program hlídá teplotu žárovky a v případě, že je teplota příliš vysoká, senzor se přepne na hodnotu logická 0. To znamená, že není splněna podmínka pro rozsvícení světla a na displeji se zobrazí alarmové hlášení „přehřáté světlo“. Proměnná pn_error bude popsána v kapitole Alarmová hlášení (viz kapitola 19).

V systému Heidenhain se ještě nachází speciální verze příkazu if. Tento příkaz if spouští vyhrazenou část programu, v závislosti na proměnné string v konfiguračních souborech.

```
#if CFG_funkce = "aktivní"          ; Pokud je string CFG_funkce = aktivní
L               I_input             ; Načti vstup
S               O_output            ; Sepni výstup
#endif                              ; Konec příkazu if
```

Tento if spouští určitý blok programu v závislosti na konfiguračním souboru. To se hodí například, pokud má programátor zpracované univerzální PLC a ke každému stroji pouze zapíná a vypíná dané opce či funkce, aniž by byla potřeba něco složitě programovat.

14.1 Funkce CASE

V případě příkazu CASE, je práce mnohem specifičtější. Používá se v případech, kdy po sobě následuje několik podprogramů a je nutné dodržet jejich pořadí. Například, nejčastější využití má tento cyklus v programech pro automatickou výměnu nástrojů, kde obsahuje až několik desítek podprogramů, které musí být vykonány, aby proběhla výměna jednoho nástroje.

Vzhled programu pro výměnu nástroje, velmi jednoduchou, s větvícím cyklem CASE může být následující:

```
;-------------------------
LBL Vymena nastroje
;-------------------------
CASE     BG_krok_vymeny_nastroje            ; Načti Byte pro příkaz CASE
CM       Start_vymeny                ;00    ; Jednotlivé podprogramy
CM       Cislo_nastroje              ;01
CM       Natoceni_zasobniku          ;02
CM       Orientace_vretene           ;03
CM       Najezd_os_do_pozice_vymeny  ;04
CM       Rameno_vymeniku_natocit     ;05
CM       Uvolneni_nastroje           ;06
CM       Vymena_nastroju             ;07
CM       Upnuti_nastroje             ;08
CM       Rameno_vymeniku_domu        ;09
CM       Aktualizuj_nastroje         ;10
ENDC                                        ; Konec cyklu CASE
```

V případě cyklu CASE se na začátku programu načte registr Byte, do kterého se zapisuje aktuální krok, neboli číslo podprogramu, který právě probíhá. Jednotlivé podprogramy se volají příkazem CM, neboli „call module“. Tento příkaz bude vysvětlen v kapitole podprogramy a skoky.

Programy jdou popořadě tak, jak jsou očíslovány, ale v případě potřeby lze vložit do tohoto příkazu nějaké přeskoky. To znamená, že za určitých podmínek přeskočí systém několik řádků a pak pokračuje dál.

Je nutné dodat, že se zde jedná o podprogramy, neboli moduly. To znamená, že pod každým názvem se skrývá podprogram se svým vlastním názvem (Label), strukturou a koncem programu (EM).

15 Podprogramy a skoky v systému Heidenhain

Při psaní programu nastávají chvíle, kdy je zapotřebí přeskočit určitou část programu v závislosti na nějaké vstupní veličině. Z tohoto důvodu se v systému nachází funkce JMP, neboli „jump“. Tato funkce přeskočí danou část programu na LABEL, který bude vepsán v definici funkce. Zbývá dodat, že je k dispozici funkce JPT (jump if true) a JPF (Jump if false). Program se skokem může vypadat následovně:

```
;-------------------------
LBL Cast 1
;-------------------------
L           I_podminka              ; Načti vstupní signál, pokud je
                                    logická 1
JPT         Cast 2                  ; Přeskoč na podprogram s názvem Cast
                                    2

L           I_tlacitko              ; Načti signál z tlačítka
R           ML_pomocny_marker       ; Resetuj lokální marker
EM                                  ; Konec podprogramu
;-------------------------
LBL Cast 2
;-------------------------
L           I_podminka              ; Načti signál z tlačítka
S           ML_pomocny_marker       ; Nastav lokální marker
EM                                  ; Konec podprogramu
```

Principielně lze říct, že funkce přeskoku funguje stejně jako funkce IF. Pokud je splněna vstupní podmínka, přeskočí sken programu všechny příkazy až po název podprogramu zadaného v hlavičce funkce JMP. V tomto případě přeskočí na podprogram „cast 2“. Pokud není vstupní podmínka splněna, proběhne sken prvního podprogramu a následně i druhého podprogramu.

Stejným způsobem funguje volání podprogramů. Pro tento případ se zde nachází funkce CM, neboli „call module“. Opět se dělí na příkazy CMT (call module if true) a CMF (call module if false). V případě volání podprogramu je však posloupnost následující.

```
;-------------------------
LBL Program
;-------------------------
L           I_podminka          ; Načti vstupní signál, pokud je logická 1
CMT         Cast 2              ; Přeskoč na podprogram s názvem Podprogram
EM                              ; Konec podprogramu
;-------------------------
LBL Podprogram
;-------------------------
L           I_podminka          ; Načti signál z tlačítka
S           ML_pomocny_marker   ; Nastav lokální marker
EM                              ; Konec podprogramu
```

Za těchto podmínek se může podprogram nacházet v kterémkoli souboru s příponou SRC, neboli v kterémkoli programu. Volání podprogramu se zde může zdát zbytečné, ale již v základním programu je od společnosti Heidenhain v databázi naprogramováno cca 400 tzv. modulů. Lépe řečeno, je zde asi 400 podprogramů, které mají z výroby svoji funkci a unikátní číslo, kterým je možné je volat.

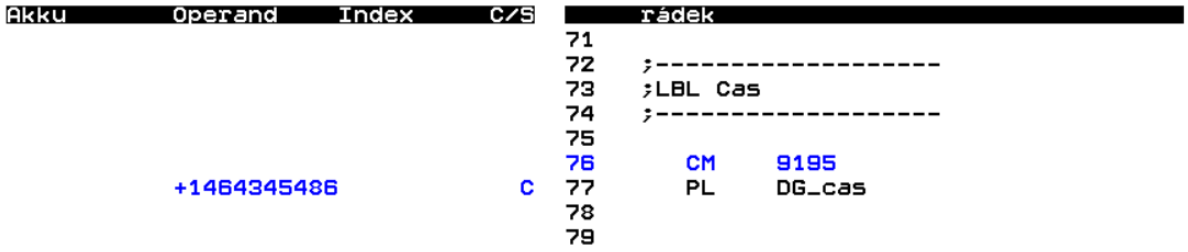
Jednoduchým příkladem je strojový čas. V systému Heidenhain lze pracovat se strojovým časem, a to za použití modulu s číslem 9195. Příklad by pak vypadal takto.

```
;-------------------------
LBL Cas
;-------------------------

CM                9195              ; Zavolej modul 9195
PL                DG_cas            ; Vlož výslednou hodnotu do registru
                                    DG_cas

EM
```

Základem je vždy příkaz pro zavolání modulu (CM). Poté se moduly dělí na ty, které výsledek takzvaně vytáhnou nebo vtlačí. To znamená, že jsou použity dva základní příkazy:

PS	Push	Vtlačit
PL	Pull	Vytáhnout

Příkaz PS je obecně použit, pokud je potřeba do modulu vtlačit nějakou vstupní hodnotu, zatímco příkaz PL vytáhne z modulu výsledek. Toto strukturované programování mnohonásobně zjednodušuje práci i program.



Obr. 15.1: Modul 9195 v režimu TRACE IN-CODE

Na obrázku (viz obr. 15.1) jde vidět program v režimu TRACE: Nachází se zde jedna zvláštnost, Tou je hodnota času v decimálním tvaru, jejíž obsah je +1464345486. Tato hodnota obsahuje počet sekund, které uplynuly od 1. ledna roku 1970. Toto je výchozí datum pro systém Heidenhain. Z tohoto čísla lze pak modulem, nebo vlastním výpočtem vytvořit například dnešní datum, či den v týdnu.

Pro aktuální datum se v systému nachází další modul 9055, který bude vysvětlen v další podkapitole.

15.1 Modul pro aktuální čas

V případě, že je zapotřebí pracovat s aktuálním časem, či datem, je nutno použít modul 9055. Tento podprogram vytvoří ze systémového času (viz kapitola 15) lokální čas. Program vypadá následovně:

```
#define /s    KL_cas_format       K+0      ; Definice lokálního markeru,
                                           udávajícího formát výsledného
                                           času (Rozsah 0 - 15)
;-------------------------
LBL Realny cas
;-------------------------

PS            DG_cas              ; Registr se systémovým časem (viz
                                  kapitola 15)
PS            K &SG_Real_cas      ; ,,STRING" výsledný registr
PS            KL_cas_format       ; Lokální marker s formátem času
CM            9055                ; Příkaz pro volání podprogramu

EM
```

Výsledkem tohoto podprogramu je aktuální datum nebo čas. Toto je dáno lokální proměnnou KL_cas_format. Modul 9055 nabízí 16 různých formátů času. Tyto formáty jsou vypsány v technickém manuálu iTNC 530.[3] V tomto případě je zvolený formát 0, což znamená časový formát DD.MM.YY. hh:mm:ss. Výsledný čas se zapisuje do globálního registru typu „STRING“. Registr poté vypadá následovně:

```
S 55""
S 56""
S 57"27.05.2016 16:20:42"
S 58""
S 59""
S 60""
S57 = SG_REAL_CAS
```

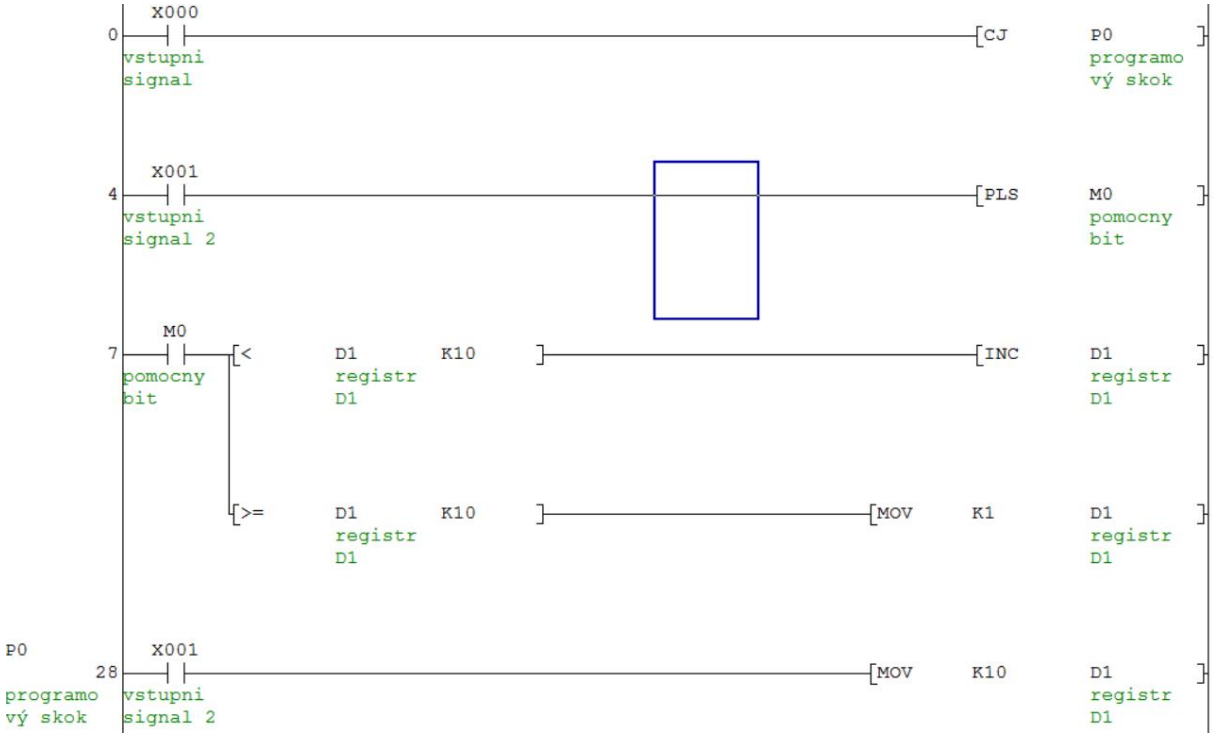
Obr. 15.2: Aktuální čas a datum v registru SG_REAL_CAS

Neboli do registru SG_REAL_CAS je vložen aktuální čas a datum. S tímto registrem je poté možné dále pracovat.

Nutno dodat, že i v případě globálních registrů typu STRING i DOUBLE, je zapotřebí definovat název a typ registru do definičního souboru s příponou .def, ale za normálních podmínek se číslo tohoto registru přiděluje automaticky kompilátorem (na obrázku má naprogramovaný registr hodnotu 57).

16 Podprogramy a skoky v systému Mitsubishi

Pro ukázku programového skoku byl převzat program softwarového zásobníku, který je podrobněji vysvětlen v kapitole 17.2, k němuž byl druhým vstupním signálem přidán programový skok.



Obr. 16.1: programový skok v systému Mitsubishi

Funkce programu bude následující. V případě, že se na vstupu X001 objeví logická 1, bude program přičítat +1 do registru až po hodnotu 9. Pokud se zároveň objeví logická 1 na vstupu X000, příkaz CJ, neboli „cycle jump“, přeskočí sken programu až na odrážku P0 (nachází se vždy vlevo od levé sběrnice). To znamená, že se do registru D1 přesune hodnota 10.

Takřka stejný princip platí pro volání podprogramů. Rozdíl je pouze v tom, že podprogramy nejsou závislé na pořadí skenování, ale můžou se nacházet na konci programu. Poté je stačí jen zavolat příkazem „CALL“ a číslem daného podprogramu.

Obr. 16.2: Volání podprogramu v systému Mitsubishi

V případě, že je vstupní signál logická 0, tak se do registru D1 zapisuje hodnota +11. Pokud se na vstupním signálu objeví logická 1, vyvolá se podprogram P1 a do registru D1 se přesune hodnota +10 a zároveň se vypne zápis hodnoty +11 (zajištění správného výsledku).

Funkce předvedených programů se může zdát zbytečná, nicméně cílem je porovnání a především jednoduchost programů pro pochopení funkce. Pak je na každém programátorovi jakým stylem bude programovat.

17 Práce s registry

V každém systému, a to i mimo Heidenhain a Mitsubishi, je nutno pracovat s registry. Tyto registry slouží k práci s čísly, která se skládají z několika bitů. Obvykle jsou to 4, 8, 16, 32 bitů. Z toho plyne, že například 8mi bitový registr obsahuje 8 bitů a každý z nich nabývá hodnotu logická jedna a logická nula. Tedy, 8mi bitový registr může nabývat 256 hodnot. Jak již bylo dříve popsáno, tyto registry se dělí na Byte (8 bitů), Word (16 bitů) a Double (32 bitů).
K práci s těmito registry se nejčastěji používají typické matematické znaky, z nichž jsou některé níže popsány:

- - **Sčítání**
+ - **Odčítání**
= - **Rovná se**
< - **Větší než**
> - **Menší než**
<> - **Nerovná se**
<= - **Větší nebo rovno**
>= - **Menší nebo rovno**

Z definice registru je nyní možné přijít na využití v PLC programu. Všechna čísla, která se ve stroji nachází v hexadecimálním nebo dekadickém tvaru, je nutné uchovávat v registrech. Kvůli rychlosti zpracování se pak vybírá správný typ registru. Pokud je zapotřebí pracovat s hodnotou např. 0 – 100, vybere si programátor Byte a ne Double.

Registry se používají například pro tabulky a čísla nástrojů, či pro uchování přesné softwarové pozice motorových enkodérů, ale také například pro uchování reálného času a data či napětí baterií a mnoha dalších veličin.

V následujících kapitolách bude znázorněno, jakým způsobem se pracuje s registry v systémech Heidenhain a Mitsubishi, což lze považovat za pokročilé programování.

17.1 Početní operace v systému Heidenhain

Pro začátek postačí jednoduchý příklad výpočtu, jako je například dělení registru BG_pocty.

```
;-------------------------
LBL Pocty, deleni (zacykleno)
;-------------------------
L            K+50               ; Načti konstantu s hodnotou 50
=            BG_pocty           ; Vlož konstantu do registru pocty

L            BG_pocty           ; Načti registr pocty
/            K+10               ; Vyděl obsah registru s hodnotou 10
=            BG_pocty           ; Výsledek vlož do registru pocty
```

V tomto případě je načten registr BG_pocty tzv. do zásobníku. Obvykle je obsah registru nějaká hodnota, která je výsledkem předchozích výpočtů, či bitovou hodnotou z A/D převodníku (například senzoru). Zde byla do registru nejdříve vložena decimální konstanta s hodnotou 50, poté byla konstanta opět načtena a vydělena decimální konstantou s hodnotou 10. Konečný výsledek byl nakonec opět zapsán do registru BG_pocty. Tento program se může zdát jako nesmyslný, jelikož při každém novém skenu je do něj opět vložena hodnota 50 a poté je na krátkou dobu vydělena na hodnotu 5. Jedná se o ukázku špatně navrženého programu.

Tento příklad je pouze ukázkový pro správné pochopení výpočetních příkazů. V tomto směru je vhodné připravit určitý program, který bude následovat v další podkapitole.

Zde je popsaný jednoúčelový příklad sčítání, který se nebude cyklicky přepisovat, jelikož zde program pracuje s pomocným lokálním registrem.

```
;-------------------------
LBL Pocty, scitani
;-------------------------
L            K+50               ; Načti konstantu s hodnotou 50
=            BL_pocty_pomoc     ; Vlož konstantu do registru pocty_pomoc

L            BG_pocty           ; Načti registr pocty
+            K+10               ; Přičti obsah registru s hodnotou 10
=            BG_pocty           ; Výsledek vlož do registru pocty
```

17.2 Program pro softwarové otáčení zásobníku v systému Heidenhain

Jednoduchým příkladem práce s registry je zásobník nástrojů. Jelikož má v dnešní době většina obráběcích center zásobník nástrojů a tudíž i automatickou výměnu, je nutné otáčení zásobníku a případnou výměnu naprogramovat do programu PLC.

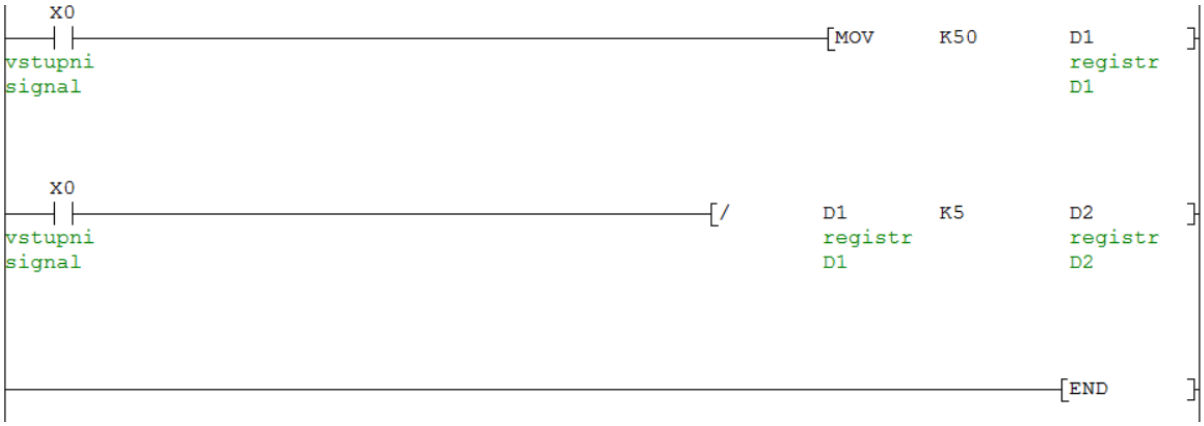
Pokud je zapotřebí otočit zásobníkem o jednu pozici (další nástroj), bude program vypadat následovně:

```
;-------------------------
LBL Zasobnik
;-------------------------
L        I_Zasobnik_vpravo   ; Vstupní proměnná
AN       ML_I_Pulse          ; Vytvoření pulsu
IFT                          ; Příkaz IF pokud je předchozí podmínka
                             splněna
L        k+10                ; Načti hodnotu +10
>        BG_Nastroj          ; Registr BG_Nastroj menší než +10
IFT                          ; Příkaz IF pokud je splněna předchozí
                             podmínka
L        K+1                 ; Načti hodnotu +1
+        BG_Nastroj          ; Přičti +1 k registruj BG_Nastroj
=        BG_Nastroj          ; Výsledek zapiš do registru BG_Nastroj
ELSE                         ; Příkaz ELSE
L        K+1                 ; Načti hodnotu +1
=        BG_Nastroj          ; Zapiš hodnou +1 do registru BG_Nastroj
ENDI                         ; Konec větve IF
ENDI                         ; Konec větve IF
EM
```

Tento program přičítá číslo nástroje po jednom až do deseti. V případě, že se dostane registr BG_nastroj na číslo 10, přeskočí zpět na nástroj číslo 1. V podstatě jde o cyklus FOR, ten však systém Heidenhain nepodporuje. Dále je nutno podotknout, že se zde jedná pouze o počítání. Do tohoto programu by bylo ještě nutno zakomponovat výstup na spínání motoru. Zde je to však závislé na hardwaru stroje. Například, v případě asynchronního motoru by zde byl výstup, který by sepnul otáčky motoru a poté vstup nějakého indukčního senzoru, který by udával posun o jeden nástroj.

17.3 Početní operace v systému Mitsubishi

V systému Mitsubishi vypadá konečný výpočet velmi podobně jako v předešlé kapitole, nicméně rozdíl v náročnosti je znatelný téměř okamžitě po začátku programování. Program vypadá následovně:



Obr. 17.1: Výpočetní program v systému Mitsubishi

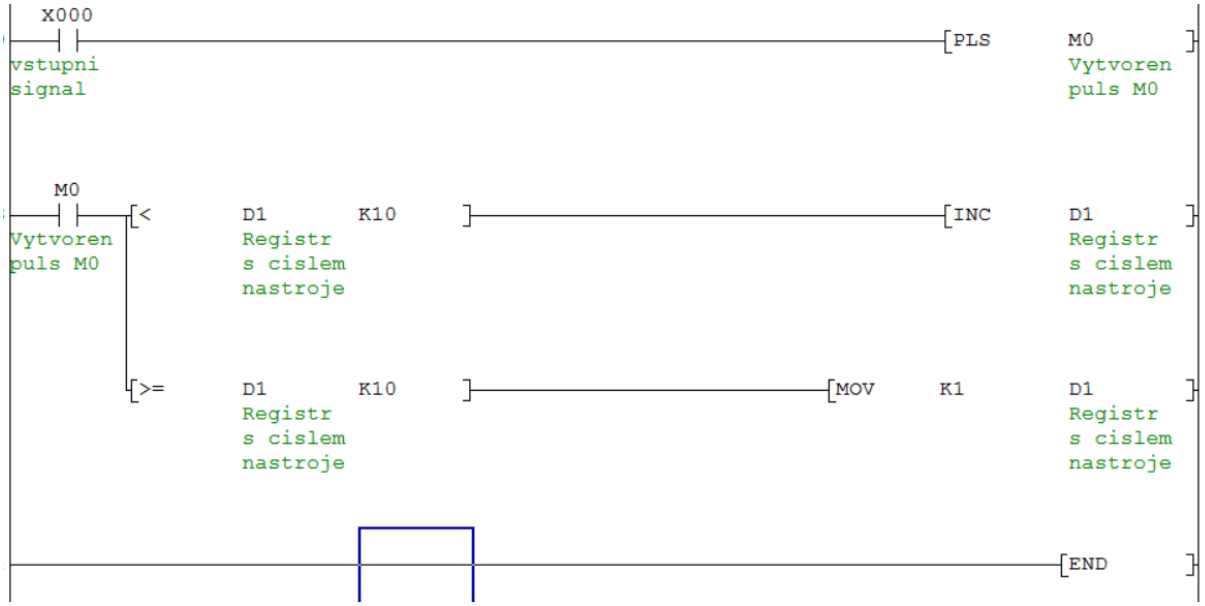
Zde je v závislosti na vstupním signálu X0 vložena decimální konstanta s hodnotou 50 do registru D1 a to příkazem MOV (jednoduchý příkaz pro přesun obsahu registru do dalšího registru). Poté je opět v závislosti na signál X0 hodnota z registru D1 vydělena konstantou s hodnotou 5, načež je výsledek zapsán do dalšího registru D2.

Nyní je nutno dodat, že programování v systému Mitsubishi je přímé. Z toho plyne, že není potřeba žádná definice registru, jako v systému Heidenhain, kde je nutno daný registr zapsat do definičního souboru aby s ním vůbec bylo možné pracovat.

Nakonec není nutno registr načítat do zásobníku tak, jako v systému Heidenhain. Jednoduše, jakmile se v registru nachází nějaká hodnota, je možné s ní okamžitě pracovat.

17.4 Program pro softwarové otáčení zásobníku v systému Mitsubishi

Zde je pro porovnání program pro ovládání otáčení softwarového zásobníku. Opět se jedná pouze o softwarový zásobník, neboť ke konečnému mechanickému otočení zásobníku je zapotřebí výstupu, který bude spouštět servo motor atd. Zde je výstup taktéž závislý na principu otáčení a snímání daného nástroje.



Obr. 17.2: Program pro softwarový zásobník

Co se týká funkce tohoto programu, jsou zde použity dva nové příkazy. Tedy příkaz PLS, který vytvoří pouze pulz nezávisle na délce vstupního signálu. Druhým použitým příkazem je INC, což je inkrement o hodnotu +1.

Nyní je zřetelné, že vstupní signál X000 vytvoří puls M0. Za tímto pulsem je v podstatě příkaz IF. V případě, že se v registru D1 nachází hodnota menší než 10, je registr inkrementován o hodnotu 1. Pokud se v registru objeví hodnota 10, anebo vyšší, je příkazem MOV do registru D1 vložena hodnota 1.

18 Speciální příkazy

Za předpokladu, že v kapitole o základních funkcích, byly vysvětleny základní příkazy, je nutné osvětlit speciální příkazy, kterými je možné usnadnit programování. Tyto příkazy se nacházejí v obou systémech.

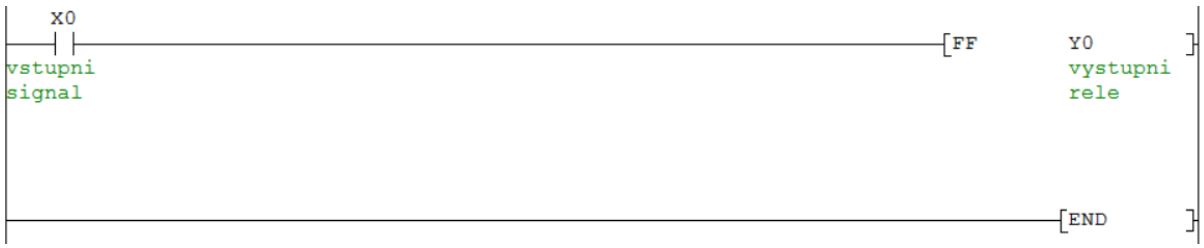
V kapitole 10.1 byl popsán program pro tlačítkové spínání. Tento příklad byl založen na vytvoření pulsu. Nicméně, na tak jednoduchou funkci se zdál program zbytečně složitý. Pro tento případ je však možné použít funkci XOR, neboli exkluzivní součet:

```
;-----------------------------------------------
LBL Tlacitkove spinani s funkci XOR
;-----------------------------------------------

L           I_tlacitko                ; Načti vstupní signál z tlačítka
AN          ML_I_tlacitko_zmacknuto   ; A zároveň marker s logickou
                                      hodnotou 0
XO          O_svetlo                  ; Exkluzivní součet hodnot
=           O_svetlo                  ; Spíná výstup se světlem

L           I_tlacitko                ; Načti vstupní signál z tlačítka
=           ML_I_tlacitko_zmacknuto   ; Nastav marker do hodnoty logická 1
EM
```

Pokud je splněna podmínka na prvních dvou řádcích a není sepnutý výstup (logická 0 a 1), znamená exkluzivní součet sepnutí výstupu. Sepnutím pomocného markeru (až po sepnutí výstupu) skončí platnost podmínky na prvních dvou řádcích, ale stále je sepnutý výstup. Tím pádem je dle exkluzivního součtu stále na výstupu logická 1. V případě vypnutí výstupu tlačítkem program funguje stejně, jen s opačnými hodnotami.

V tomto případě je program o polovinu kratší než v kapitole 10.1. Tím se minimalizuje chyba a zároveň se méně zatěžuje procesor.

Stejný příklad tlačítkového spínání je uveden i pro systém Mitsubishi a to v kapitole 13.1. I zde se nachází speciální funkce pro zjednodušení programu.



Obr. 18.1: Příkaz FLIP FLOP v systému Mitsubishi

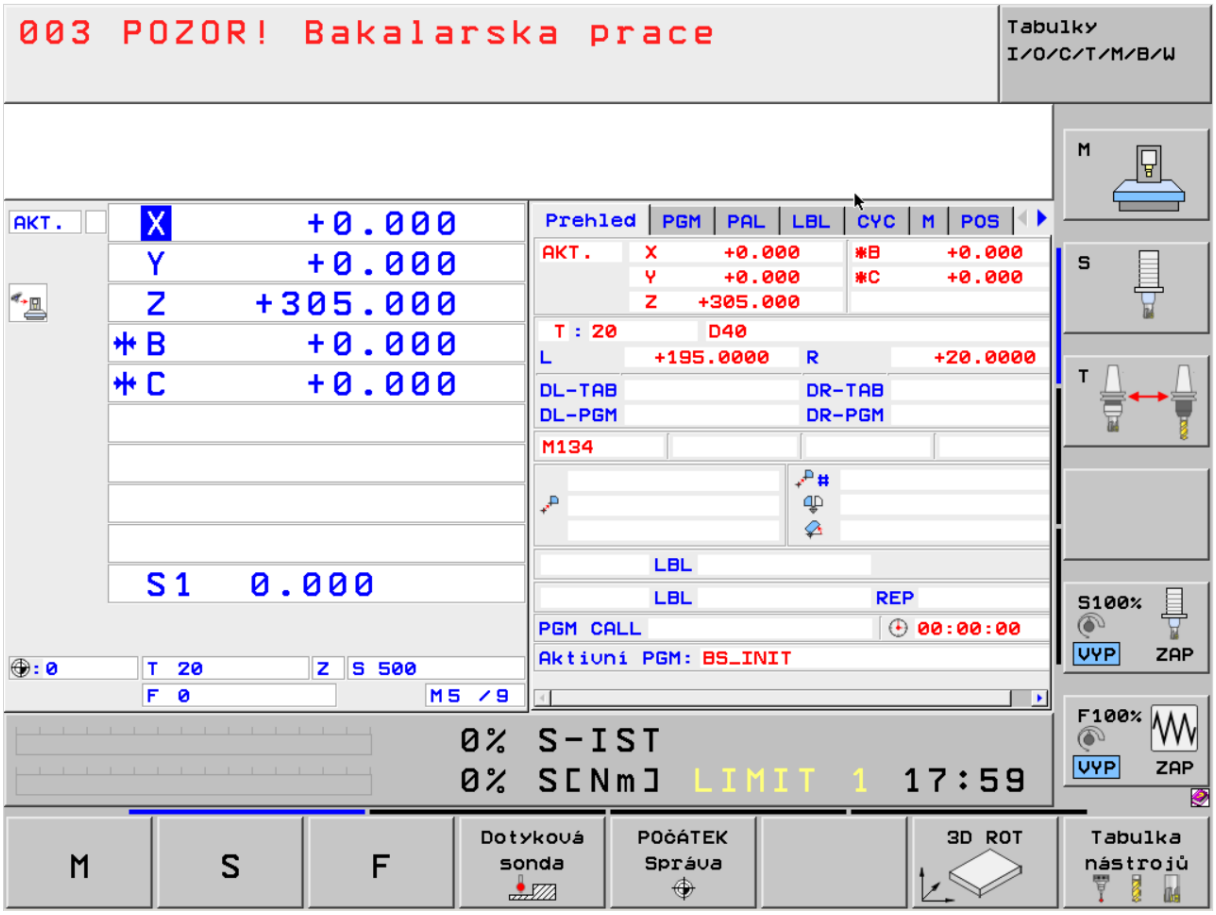
Zde se nachází funkce FF, neboli „flip flop“. Zjednodušeně to znamená, že v závislosti na stisku tlačítka X0 se bude přepínat výstupní relé Y0. Toto zjednodušení je oproti programu v kapitole 13.1 markantní a ušetřilo několik řádků programu.

19 Alarmová hlášení

Pro správný provoz musí mít stroj nějakou možnost komunikovat s obsluhou v případě, že se objeví nějaká závada. V tomto případě stroj zobrazí alarmové hlášení. Obvykle je toto hlášení zvýrazněno červenou barvou a může být rozděleno na několik druhů. Alarmové hlášení může upozorňovat na nějakou chybu obsluhy, či stroje. Dále pak lze nastavit toto hlášení tak aby při výskytu problému zastavilo obráběcí program, či úplně převedlo stroj do stavu „EMERGENCY“ (havarijní či nouzový stav), což je stav, ve kterém jsou veškeré periferie stroje nuceně odpojeny od elektrické sítě, všechny servopohony jsou zablokovány a vřetenový motor zabrzděn.

19.1 Alarmová hlášení v systému Heidenhain

V systému Heidenhain se alarmová hlášení zobrazují na horní liště tučným červeným písmem(viz obr. 19.1).



Obr. 19.1: Alarmové hlášení Heidenhain

Existuje několik desítek předdefinovaných alarmových hlášení přímo od výrobce systému a ty jsou neměnné. Dále se v systému nachází 2047 volných alarmových hlášení připravených pro programátora v souboru alarmových hlášení ERROR.tab. Tato alarmová hlášení se spouští bitovými markery od hodnoty 4800, a nebo speciálním modulem. Tedy pokud bude zapnut marker 4803, na displeji se zobrazí hlášení číslo 003(viz obr. 19.1). Výše zmíněná tabulka obsahuje parametry alarmových hlášení (viz obr. 19.2).

Soubor: ERR_TAB.PET >>

NR	ERROR	MARKER	RESET	NC-STOP	NC-CANCEL	F-STOP
0	#000 Motorschutz-Schalter	4800	0	1	0	0
1	#001 Temperatur Regler/Motoren	4801	0	1	0	0
2	#002 Hydraulik Druck	4802	0	0	0	0
3	#003 Pneumatik Druck	4803	0	0	0	0
4	#004 PW 210 Temperatur max	4804	0	1	0	0
5	#005 DA300 Druck	4805	0	0	0	0
6	#006 Versorgungsmodul Antriebe	4806	0	0	0	0
7	#007 Freigabe M07 für WZ inaktiv	4807	0	1	0	0
8	#008 Klemmbetrieb nicht erlaubt	4808	0	1	0	0
9	#009 Arbeitsraum geschlossen !	4809	0	0	0	0

Obr. 19.2: Tabulka alarmových hlášení Heidenhain

V této tabulce se kromě názvu alarmu dá nastavit několik dalších důležitých parametrů, jako jsou například:

MARKER Číslo markeru

NC-STOP Přerušení obráběcího programu

F-STOP Uvolnění posuvu

EMER. STOP Havarijní stav stroje

CE Povoleno smazání hlášení tlačítkem CE

V případě, že je vyplněna tabulka alarmových hlášení, už zbývá jen dané hlášení zobrazit na displeji. Jak bylo popsáno výše, zobrazení řídí marker. Ten je však nutno zavézt do definičního souboru GLB_TCMB.def. V tomto souboru je markeru přiřazen unikátní název typu „pn_error_popisek“. Marker je nyní zaveden a už zbývá poslední krok. Jelikož se alarm spouští v návaznosti na nějakou chybu, tedy obvykle na změnu nějaké vstupní nebo výstupní veličiny, je potřeba tento alarm doprogramovat do PLC programu jako například:

```
;-------------------------
LBL Hlídání hydraulického tlaku
;-------------------------

LN      I_hydraulicky_tlak          ; Pokud je vstup I v hodnotě log 0
S       pn_error_hydraulicky_tlak   ; Zobraz alarmové hlášení

EM
```

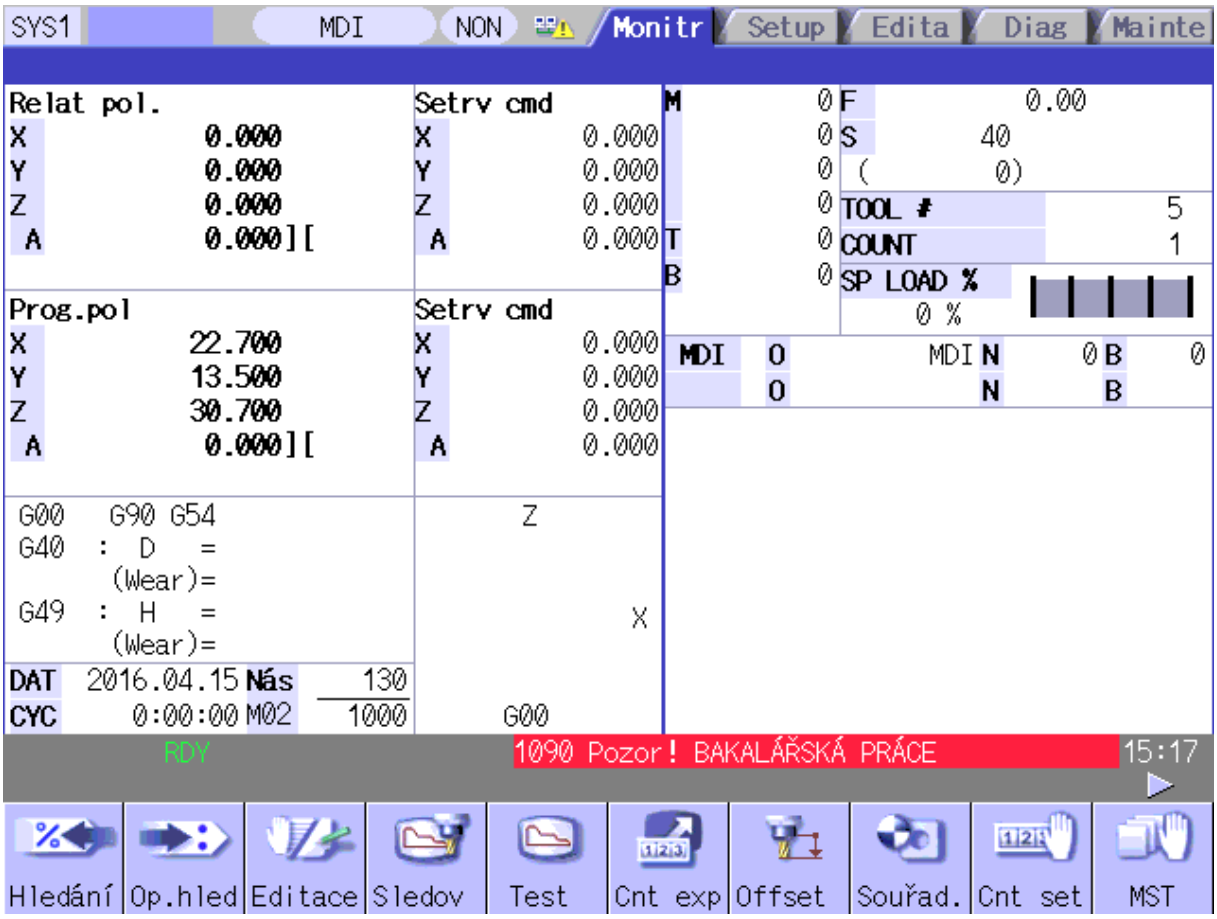
Tento jednoduchý podprogram hlídá díky tlakovému senzoru správný tlak hydraulického agregátu. Pokud je na hydraulickém agregátu správný tlak, senzor posílá do stroje signál +24V, neboli logická 1. Pokud tlak není v pořádku, senzor posílá do stroje signál 0V, neboli logická 0 a tím pádem se sepne marker M4802 dle tabulky ERROR.tab (viz obr. 11)

V tabulce ERROR.tab jsou všechny názvy alarmových hlášení v původním jazyce systému. Tedy německy nebo anglicky, dle verze systému. Toto je však pouze označení alarmového hlášení. Text, který se zobrazuje na displeji pochází ze souboru error.a a to proto, aby bylo možné systém přepínat mezi jednotlivými jazyky. Ve složce PLC:/LANGUAGE/CS se nachází česká lokalizace alarmových hlášení, neboli pouze překlad.

19.2 Alarmová hlášení v systému Mitsubishi

V systému Mitsubishi je s alarmovými hlášeními o poznání méně práce. Především, v systému Mitsubishi je alarmové hlášení, pouze oznámení. Nelze u něj tedy nastavit žádný parametr a jde jen o zobrazení textu na displeji. Samotné zablokování stroje, havarijní stav, se musí doprogramovat přímo do programu PLC dle potřeb programátora.

Hlášení se v systému Mitsubishi zobrazuje v pravém dolním rohu displeje a to bílým textem na červeném pásku (viz obr. 19.3).



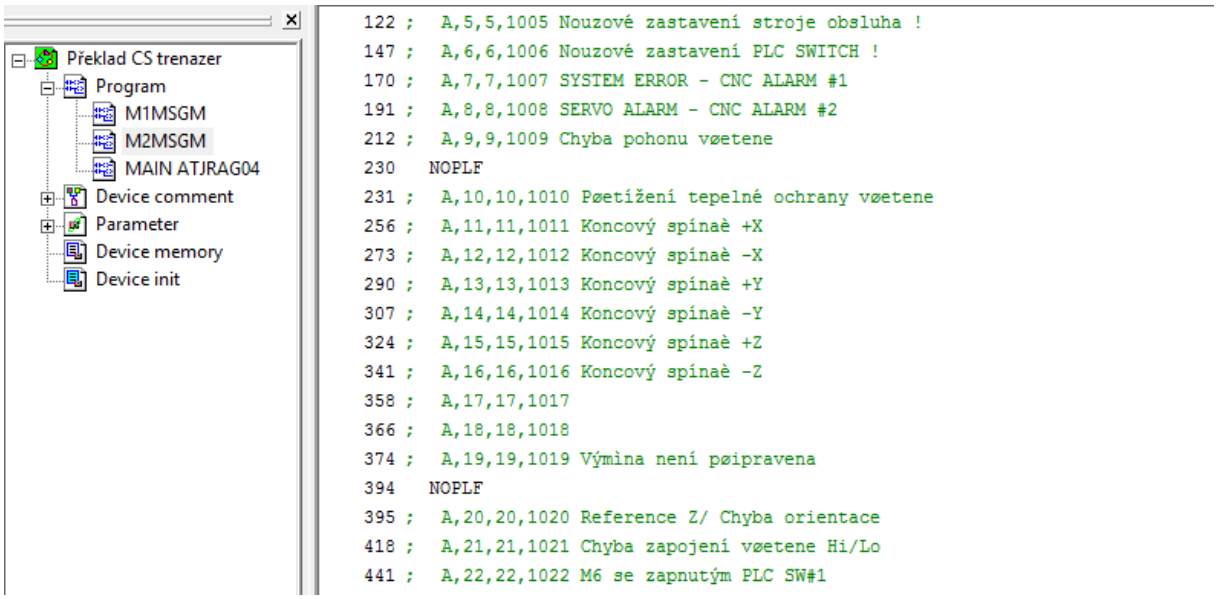
Obr. 19.3: Zobrazení alarmového hlášení v systému Mitsubishi

Programové spuštění tohoto hlášení pak vypadá následovně:

Obr. 19.4: Ovládání alarmového hlášení

Program zde funguje naprosto stejně jako ukázkový program v systému Heidenhain. Tedy, pokud je vstupní signál X0 logická 1, je hlášení neaktivní. Pokud je vstupní signál logická 0, je kontakt v klidovém stavu sepnutý a tudíž je aktivní hlášení F1. Nakonec je zde přidáno výrobcem definované tlačítko Reset (Y0C19), které přeruší přídržný kontakt a smaže alarmové hlášení z displeje.

Tento příkaz zobrazuje alarmové hlášení na displeji. Text, který se má zobrazit je načten z podprogramu M2MSGM v kořenovém adresáři PLC (viz obr. 19.5). V systému M70/M700 se v tomto adresáři nachází pouze hlavní program PLC a k němu soubory s textem pro alarmové hlášení. Těchto souborů se zde nachází tolik, kolik je zapotřebí jazyků, tzn. obvykle jsou zde dva, anglický a český. Text hlášení se z tohoto souboru vybírá podle čísla za příkazem F (F10 znamená alarmové hlášení č. 1010).



Obr. 19.5: Soubor textu alarmových hlášení, informačních hlášení a popisků

Zde lze pozorovat obrovskou výhodu oproti systému Heidenhain. Leč je zde méně možných nastavení alarmu, je programování podstatně snazší. Není zapotřebí žádná definice proměnných, překlady tabulek atd. Vytvoření alarmového hlášení trvá zlomek času než v systému Heidenhain.

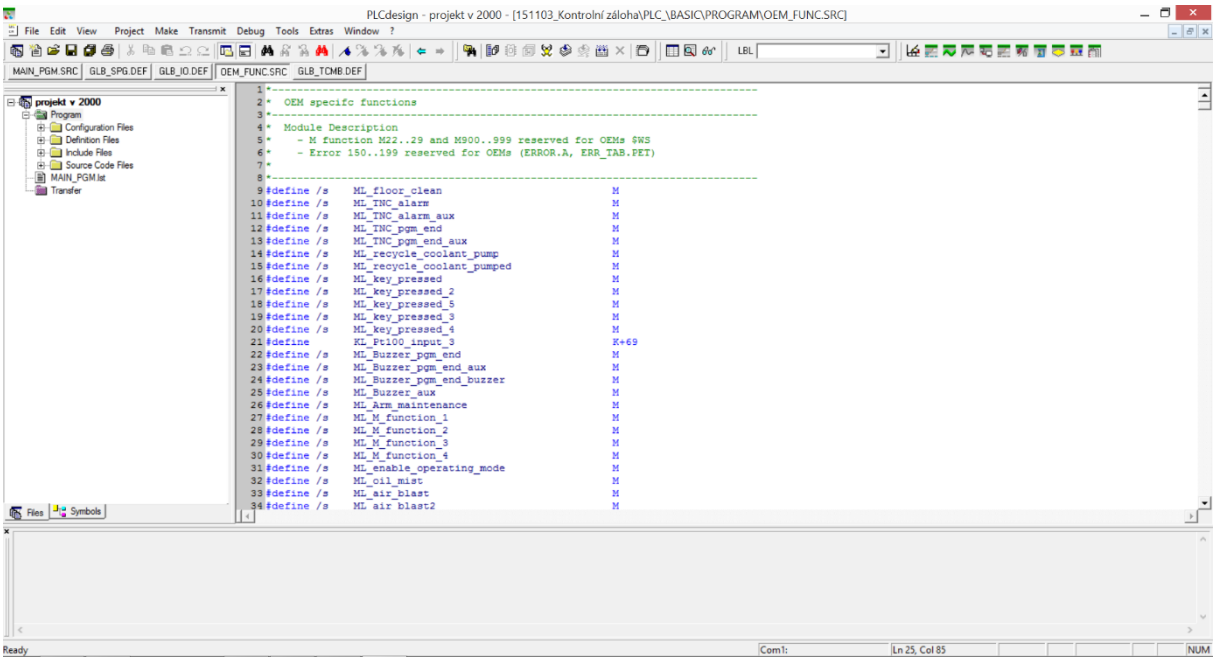
20 Vývojové Prostředí PLC Design pro Heidenhain

Systém Heidenhain nabízí své vlastní vývojové prostředí PLC programu. Toto prostředí má název PLC Design. Tento program je jedním z rozsáhlé sady programů Heidenhain. Kromě softwaru PLC design nabízí firma Heidenhain programy jako např.:

Teleservice	Program pro vzdálenou diagnostiku a přístup do systému
TNCremo	Program pro připojení počítače do systému přes konektor Ethernet nebo RS 232. Tento program umožňuje provádět komplexní zálohu systémových informací, PLC a parametrů.
Kinematics design	Program pro tvorbu kinematik stroje
IO Config	Program pro tvorbu struktury vstupů a výstupů (definiční soubory)
Config Design	Program pro tvorbu konfiguračních souborů
TNC Scope	Elektronický osciloskop

Tyto programy sice nejsou přímo určeny k programování, nicméně programátorovi usnadňují práci.

Firma Heidenhain při koupi nového systému dodává výrobci i základní program s názvem „BASIC“. V tomto programu jsou již hotové některé základní programy.



Obr. 20.1: Vývojové prostředí PLC Design

Program PLC design lze rozdělit do několika částí. Nahoře se nachází nástrojová lišta a pod ní základní funkční tlačítka pro ukládání, hledání, kompilaci a přepínání mezi přidruženými programy Heidenhain.

Na levé straně se nachází strom s PLC programem. Tento program je vždy rozdělen na 4 složky, přičemž všechny musí být vyplněny (musí obsahovat soubory ze stroje).

Source Code Files	Tato složka obsahuje veškeré programy a podprogramy, které systém skenuje. Všechny tyto programy mají příponu .src.
Definition Files	Tato složka obsahuje pouze definiční soubory. V těchto souborech se nachází seznamy vstupních a výstupních signálů, definice globálních proměnných typu Marker, Timer, Counter atd. vyznačuje se příponou .def.
Configuration Files	Zde se nachází konfigurační soubory s příponou .cfg.
Include Files	Speciální soubory s makry

Ve spodní části programu se nachází stavové okno. V tomto okně se zobrazuje průběh kompilace a případné chyby, kterých se programátor dopustil.

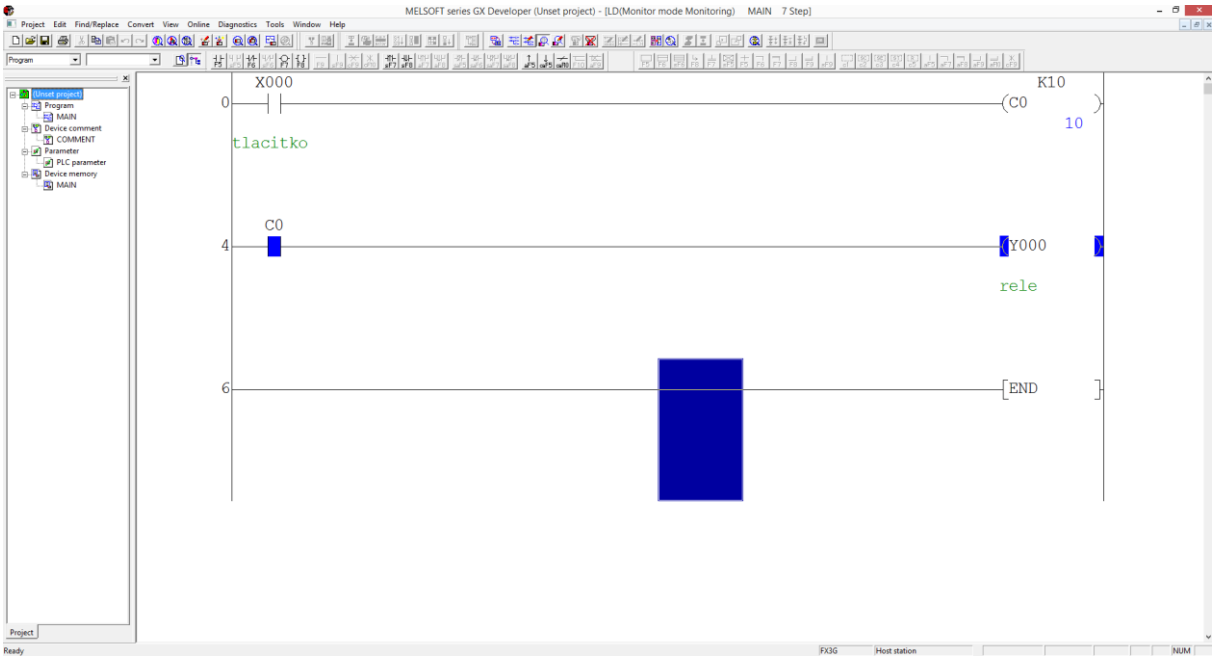
Oproti Mitsubishi je pro drobné úpravy program PLC Design téměř nepoužitelný. Nahrávání programů a definičních souborů je zbytečně zdlouhavé. Jednodušší programy je možné psát textovém prohlížeči.

Pokud se však jedná o složitější programování, objeví se nemalé výhody tohoto programu. Při programování se automaticky otevírají nabídky s jednotlivými proměnnými a případnými funkcemi, které je možno použít. Dokonce i vyhledávání jednotlivých instrukcí je velmi pohodlné, neboť program automaticky prohledá všechny programy.

Dále je možné psát jednodušší programy přímo na stroji. Není to sice pohodlné řešení, nicméně je velmi rychlé. Systém dokáže sám zkompilovat programy a vyhledat chyby a ve spojení funkcí TRACE činí programování přímo na stroji velmi efektivním.

21 Vývojové prostředí GX Developer pro Mitsubishi

Tak jako systém Heidenhain má systém Mitsubishi své vlastní programovací prostředí. Aplikaci zvanou GX Developer je možné použít pro programování všech typů PLC (řady Q a L, kompaktní PLC typu FXG atd.) od firmy Mitsubishi Electric Corporation. V tomto případě se pracuje pouze systémem CNC M70/M700.



Obr. 21.1: Vývojové prostředí GX Developer

V tomto rozvržení je vhodné popsat jednotlivé části tohoto programu. Jak bývá zvykem, nahoře se nachází nástrojová lišta a pod ní jednotlivá funkční tlačítka pro ukládání projektu, ale také pro jednotlivé programovací instrukce či přepínání mezi write módem a monitor módem.

Nalevo se nachází panel s programovou strukturou. Zde se nachází 4 základní ikony:

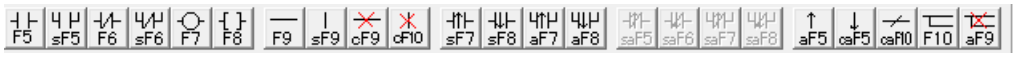
Program - Program, ve kterém se nachází hlavní program a veškeré podprogramy

Device comment - Zde se nachází seznam použitých komentářů a aliasů ke všem proměnným

Parameter - Ikona, která obsahuje základní parametry PLC

Device Memory - Zde jsou zapsány všechny aktuální hodnoty proměnných a registrů.

Hlavní programovací lišta s jednotlivými programovacími prostředky:



Obr. 21.2: Hlavní programovací lišta

1. F5 – spínací kontakt
2. F6 – Rozpínací kontakt
3. F7 – Cívka relé
4. F8 – Funkce
5. F9 – vodorovná spojovací čára
6. sF9 – svislá spojovací čára
7. cF9 a cF10 – mazací funkce spojovacích čar
8. sF7 – spínací kontakt reagující pouze na náběžnou hranu impulsu
9. sF8 – spínací kontakt reagující pouze na sestupnou hranu impulsu

Přepínač módu:


Obr. 21.3: Přepínač módu

1. První ikona je sledovací mód. V tomto módu lze jen vyhledávat zařízení
2. Druhá ikona je mód pro úpravy. V tomto módu se tvoří daný PLC program.
3. Třetí ikona je velmi důležitá. Jedná se o Monitor mód. V tomto režimu lze v reálném čase sledovat stav PLC. Bez této funkce se žádný programátor neobejde.
4. Poslední ikona v podstatě spojuje monitor a write mód. To znamená, že v tomto módu je možno provádět opravy a zároveň sledovat stav PLC, ale počet změn je omezen na 500.

Postup práce v tomto programu je velmi jednoduchý a intuitivní.

1. Připojení se k PLC systému přes ethernet, IP adresu se nastaví přes **Transfer setup**.
2. Vytvoření nového projektu a stažení PLC programu do počítače přes funkci **Read from PLC**
3. Vytvoření programu, či jen provedení úpravy ve stávajícím PLC programu.
4. Spuštění convert, funkce, která zkontroluje celý PLC program a vypočítá přesný počet použitých instrukcí.
5. Nahrání programu zpět do PLC funkcí **Write to PLC.¨**

Aplikace GX developer, však nabízí mnohem více funkcí. Nachází se zde několik vyhledávacích funkcí. Lze hledat jednotlivá zařízení či komentáře, ale také funkcí Cross reference list. Úkolem této funkce je prohledat veškeré podprogramy najednou a získat tím seznam daného zařízení v jednotlivých částech programu a přepínat mezi nimi.

Nachází se zde také funkce pro nucené nastavení proměnných. Z toho plyne, že jakoukoli proměnou lze na pevno nastavit a ověřit danou funkci programu.

Jako další se zde nachází funkce zvaná Ladder Logic Test. Jedná se v podstatě o funkci monitor mód, ale bez připojení PLC k počítači.

22 Konečný verdikt

Jelikož výrobce Mitsubishi Electric Corporation utajuje hardwarové specifikace svých systémů je problém porovnat přesný výkon obou systémů. Z tohoto důvodu byl proveden pokus o porovnání početního výkonu obou systémů na dříve uvedených programech. To znamená, že byl změřen čas skenu programu.

V systému Heidenhain je délka jednoho PLC skenu pevně stanovena na 21 ms. Poté lze pouze sledovat vytížení v procentuální hodnotě a to v celých číslech. V případě, že systém skenuje kompletní program na tříosém stroji, ve kterém se nachází tisíce řádků, se vytížení pohybuje v rozmezí 3 – 10 %. To je 0,7 – 2,1 ms.

V systému Mitsubishi čas skenu programu trvá cca 4ms (Odečteno z Aplikace GX Developer). Nicméně, tyto časy se nedají považovat za přesné měření, jsou to pouze orientační hodnoty. Z tohoto důvodu nelze přesně určit, který ze systému je výkonnější po stránce výpočetního výkonu.

Vývojová prostředí těchto systémů jsou si principielně velmi podobná. V obou programech je možné stáhnout daný program ze stroje, upravit a poté opět nahrát do stroje. Systém Heidenhain je však mnohem komerčnější, v Evropě rozšířenější a nabízí velký výběr podpůrných programů, kterými je možné vytvářet a diagnostikovat různé části stroje. Celý komplet těchto aplikací pak pomáhá výrobci stroje poskládat celý stroj dohromady, případně diagnostikovat problém. Tím je myšleno zavádění pohonů, čítačů polohy, kontrolérů a v neposlední řadě PLC.

V tomto směru je Mitsubishi mnohem náročnější. Všechny části systému je nutno zavádět do stroje manuálně. Z tohoto však plyne, že systém Mitsubishi má o mnoho více strojních parametrů, ve kterých má programátor obrovské možnosti nastavení, aniž by je musel programovat v PLC programu.

Na první pohled lze dojít k závěru, že programování v jazyku Ladder, který nabízí Mitsubishi je mnohem rychlejší a snadnější než v případě textového editoru který nabízí systém Heidenhain. Navíc pro drobné úpravy programu je software PLC design nepraktický a práce s ním je zdlouhavá. Proto je obvykle snazší provádět úpravy programu přímo na stroji. V programu GX Developer je na druhou stranu práce velmi rychlá a pohodlná a navíc lze sledovat funkci programu v reálném čase.

Systém Heidenhain je velmi podobný principu programování v jazyku C++. Konkrétně jde o definici proměnných, či o větvící příkaz typu CASE, REPEAT, IF atd. Tyto příkazy se v ladderu již z principu nepoužívají. Obecně lze ale říct, že PLC v systému Heidenhain je přehlednější. Díky názvům programů, podprogramů a možnosti použití modulů, je program úhledně strukturován, což jej dělá příjemnější pro čtení a následné opravy.

Pokud je potřeba rozdělit systémy podle aplikace, jedná se zde spíše o sympatie zákazníka. Výkonné CNC obráběcí centrum se dá vytvořit s oběma systémy a záleží pouze na tom, který systém si vybere zákazník.

Nicméně, se dá usoudit, že systém Heidenhain je již od počátku tvořen spíše pro kusovou výrobu dílců (která je v ČR velmi populární), neboť je graficky vlídnější pro uživatele a zároveň je obrábění o mnoho intuitivnější a snadnější.

Zatímco systém Heidenhain obsahuje cca stovky strojních parametrů, systém Mitsubishi obsahuje parametry v řádech tisíců. Již z tohoto vyplývá, že tento systém obsahuje mnohem větší možnosti nastavení a tedy je daleko vhodnější pro sériovou výrobu, neboť obsahuje více funkcí, ale zato je náročnější je přivést k dokonalosti. Tento systém je tedy nutné za pomoci parametrů a PLC programu odladit a tím je docíleno obrovského výrobního výkonu a především spolehlivosti všech komponentů stroje po řadu let.

Na druhou stranu, za předpokladu, že výběr systému bude závislý na programátorovi, bude výhodnější volit systém Mitsubishi. Tento výrobce se totiž specializuje na automatizaci výroby a tedy má v nabídce širokou škálu kompaktních PLC, či systémů pro robotiku. Obecně je tedy možné na systému Mitsubishi postavit automatizaci celé výrobní haly nebo závodu s bezobslužným automatickým provozem.

K tomu všemu je nutno dodat, že systém Heidenhain je v České republice velmi populární díky svým výhodám, na druhou stranu, jeho cena se za ty roky vyšplhala až na hranici 700 tisíc korun za kus. Zatímco systém Mitsubishi v plné verzi a s plnohodnotným 5ti osým obráběním vychází na polovinu.

23 Závěr

Cílem této práce je porovnání dvou naprosto odlišných systémů, konkrétně jejich programovacích jazyků, náročnosti programování a typu aplikace.

Jak bylo řečeno v předešlé kapitole (viz kapitola 22), oba systémy jsou velmi kvalitní a plně schopné ovládat obráběcí stroje. Systém Heidenhain iTNC 530 je však vhodnější na kusovou výrobu, kdežto systém Mitsubishi M70/M700 je vhodnější aplikovat na sériovou výrobu.

Za předpokladu, že si systém vybírá programátor, bude výsledný systém výsledkem sympatií programátora. Po shrnutí všech aspektů lze však říci, že systém Mitsubishi je obecně výkonnější, snazší na pochopení programovacího jazyka PLC, ale zároveň mnohem složitější na nastavení strojních parametrů a zároveň mnohem náročnější pro obsluhu stroje z důvodu práce s obráběcími cykly a obecně složitějšími funkcemi pro měření obrobků a nástrojů.

Systém Heidenhain lze však značit nálepkou „uživatelsky přívětivý“. Obráběcí cykly jsou velmi dobře zpracovány a obsahují intuitivní obrázky a stejně tak měření, s dotykovými sondami pro obrobky a nástroje je velmi hezky provedeno. Na druhou stranu je program PLC složitější na úpravy a celkové pochopení funkcí.

24 Seznam použité literatury

[1] Historie vývoje společnosti a mezníky ve vývoji produktů. Heidenhain. [online]. 2016 [cit. 2015-11-5]. Dostupné z: http://www.heidenhain.cz/cs_CZ/o-firme/historie/

[2] History of Mitsubishi Group. *Mitsubishi Electric Corporation.* [online]. [cit. 2015-10-20]. Dostupné z: http://www.mitsubishielectric.com/company/about/history/overview/group_history.html

[3] Technical Manual iTNC 530 HSCi. *DR. JOHANNES HEIDENHAIN GmbH.* Traunreut, 2014

[4] Setup Manual: M70V series. *Mitsubishi Electric Corporation,* Japan.Ver. IB1500958-E (ENG), 2014

[5] Programování PLC podle normy IEC EN 61131-3 – víc než jednotné jazyky. *Automa.* [online]. 2013 [cit. 2015-11-13]. Dostupné z: http://automa.cz/index.php?id_document=30310

[6] Programování PLC podle normy IEC 61 131-3 v prostředí Mosaic. *Výzkumné centrum pro strojírenskou výrobní techniku a technologii.* [online]. únor 2009 [cit. 2016-03-21]. Dostupné z: http://web.rcmt.cvut.cz/users/cerny/PLC_sup/TXV00321_%28v11%29_Programovani_PLC_TECOMAT_podle_IEC_61131-3.pdf

strojní parametry iTNC 530 HSCI(FS). *Heidenhain s.r.o.* Praha

[7] Instruction Manual M700V/M70V series. *Mitsubishi Electric Corporation* , Japan. Ver. IB1500922-G (ENG), 2014

[8] PLC Programming Manual: M700V/M70V/E70 series. *Mitsubishi Electric Corporation,* Japan. Ver. IB1500918-F (ENG), 2014

[9] Documentation PLC Basic Program iTNC 530. *DR. JOHANNES HEIDENHAIN GmbH.* Traunreut, 2014

[10] PLC Program Sample Manual: M700V/M70V series. *Mitsubishi Electric Corporation,* Japan. Ver. BNP-C8027-061B(ENG), 2010

[11] PLC interface Manual M700V/M70V/E70 series. *Mitsubishi Electric Corporation.* Japan. Ver. IB1500920-G (ENG), 2014

25 Použité zkratky

Zkratka	Anglický název	Český název
Alias	Alias	Podmíněný název instrukce
CM	Call module	Zavolej podprogram
CNC	Computer Numerical Control	Počítačová řídící jednotka
EM	End of module	Konec podprogramu (modulu)
EnDat		Sběrnice pro snímače polohy Heidenhain
FF	Flip Flop	Příkaz pro přepínání výstupní veličiny
HSCI	Heidenhain serial controller interface	Sériová sběrnice Heidenhain
IFT	If true	Pokračuje, pokud je tvrzení pravdivé
INC	Increment	Inkrementace
LBL	Label	Název podprogramu
LD	Ladder diagram	Žebříková syntaxe
LN	Load not	Načti logickou 0
MG	Marker global	Globální marker
ML	Marker local	Lokální marker
PCMIX		Počet instrukcí za 1 μs
PL	Pull	Vytáhni
PLC	Programable Logic Controller	Programovatelný logická jednotka
PLS	Pulse	Pulz
PS	Push	Vtlač
RISC	Reduced instruction set computing	Architektura procesorů s redukovanou instrukční sadou
RST	Reset	Reset
SG	String global	Globální string
ST	Structured text	Strukturovaný text
XOR	Exclussive or	Exkluzivní součet

26 Seznam obrázků