

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA STROJNÍHO INŽENÝRSTVÍ
FACULTY OF MECHANICAL ENGINEERING

ÚSTAV AUTOMATIZACE A INFORMATIKY
INSTITUTE OF AUTOMATION AND COMPUTER SCIENCE

ŘÍZENÍ DVOUOSÉHO POLOHOVACÍHO STOLU
CONTROL OF TWO-AXIS POSITIONING TABLE

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **Tomáš Mičulka**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. Pavel Houška, Ph.D.**
SUPERVISOR

BRNO 2017

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA STROJNÍHO INŽENÝRSTVÍ

Zadání bakalářské práce

Ústav:	Ústav automatizace a informatiky
Student:	**Tomáš Mičulka**
Studijní program:	Strojírenství
Studijní obor:	Aplikovaná informatika a řízení
Vedoucí práce:	**Ing. Pavel Houška, Ph.D.**
Akademický rok:	2016/17

Ředitel ústavu Vám v souladu se zákonem č.111/1998 o vysokých školách a se Studijním a zkušebním řádem VUT v Brně určuje následující téma bakalářské práce:

Řízení dvouosého polohovacího stolu

Stručná charakteristika problematiky úkolu:

Práce se zabývá návrhem a realizací řízení dvouosého polohovacího stolu. Polohovací stůl bude poháněn DC motory s inkrementálními enkodéry. Pro homing jsou osy vybaveny koncovými snímači. Řízení bude realizováno na PLC s motorovými modulem firmy Beckhoff.

Cíle bakalářské práce:

1. Seznamte se s programovacím prostředím TwinCAT 3;
2. Seznamte se s použitým modulem pro řízení dvou DC motorů;
3. Proveďte oživení jedné osy a změřte dosažitelné parametry jedné osy;
4. Realizujte řízení dvou os, ověřte polohování při pohybu po kružnici.

Seznam doporučené literatury:

SHELL, R. L. a HALL, E. L. Handbook of industrial automation. New York: M. Dekker, c2000, xii, 900 p. ISBN 0-8247-0373-1.

PAVELKA J., ČEŘOVSKÝ Z., JAVŮREK J.: Elektrické pohony, Nakladatelstvi ČVUT, Praha 2003.

Dokumentace PLC a modulů [online] Dostupná na http://www.beckhoff.com.

Termín odevzdání bakalářské práce je stanoven časovým plánem akademického roku 2016/17

V Brně, dne

L. S.

doc. Ing. Radomil Matoušek, Ph.D. ředitel ústavu	doc. Ing. Jaroslav Katolický, Ph.D. děkan fakulty

ABSTRAKT

Bakalářská práce se zabývá návrhem a realizací software pro řízení dvouosého polohovacího stolu. Řízení stolu je realizováno na PLC s kartou Beckhoff EL7342 pro řízení DC motorů. Software je realizováno v prostředí TwinCAT 3, s využitím knihoven TC_MC2 dle standardu PLCopen. Dále je prezentováno použití metod camming a gearing pro centrální řízení dvou motorů.

ABSTRACT

The bachelor thesis deals with design and implementation of software for control of two axis positioning table. Table control is implemeted in PLC with the Beckhoff EL 7642 card for DC motor control. Software is designed in the TwinCat3 enviroment, using TC_MC2 according to the PLCopen standard. The use of camming and gearing methods for central control of two motors is presented too.

KLÍČOVÁ SLOVA

Beckhoff, DC motor, PLC program, řízení dvouosého stolu, NC řízení, PLCopen, Motion Control

KEYWORDS

Beckhoff, DC motor, PLC program, control of two axes table, NC control, PLCopen, Motion Control


ai
ÚSTAV AUTOMATIZACE
A INFORMATIKY

BIBLIOGRAFICKÁ CITACE

MIČULKA, T. *Řízení dvouosého polohovacího stolu.* Brno: Vysoké učení technické v Brně, Fakulta strojního inženýrství, 2017. 49 s. Vedoucí bakalářské práce Ing. Pavel Houška, Ph.D.

PODĚKOVÁNÍ

Rád bych poděkoval svému vedoucímu práce Ing. Pavlu Houškovi Ph.D za cenné rady a připomínky, jeho ochotu, čas a odborné vedení při vytváření této práce.

ČESTNÉ PROHLÁŠENÍ

Prohlašuji, že tato práce je mým původním dílem, zpracoval jsem ji samostatně s použitím literatury uvedené v seznamu literatury a pod vedením Ing. Pavla Houšky Ph.D.

V Brně dne 20. 5. 2017 ……

Tomáš Mičulka

OBSAH

1 ÚVOD 15

2 PLCopen 17
2.1 Hlavní činnosti 17
2.1.1 Programovací jazyky: 17
2.1.2 Standardizace rozhraní 18

3 MOTION CONTROL 21
3.1 Základní stavební prvky 21
3.2 Stavový diagram 22
3.3 Zpracování chyb 25

4 FUNKČNÍ BLOKY 26
4.1 Základní informace 26
4.2 BufferMode 28
4.3 Data ‘AXIS_REF‘ 29
4.4 Reakce na hranu 29

5 SYNCHRONIZOVANÉ OVLÁDÁNÍ OS 31
5.1 Elektrické převody (‘GearIn‘) 31
5.2 Elektronické vačky (‘Camming‘) 31
5.2.1 CAM tabulka 32

6 OVLÁDNÍ VÍCE MOTORŮ 33
6.1 TwinCAT 3 (The Windows Control and Automation Technology) 33
6.2 Řídící modul pro DC motory - EL 7342 34
6.3 Fungování řídící jednotky 35
6.3.1 Parametry motorů 35
6.3.2 Nastavení os 36
6.3.3 Nastavení proporcionálního regulátoru 37
6.3.4 Výsledná přesnost motorů 39
6.4 Model stolu 39
6.4.1 Algoritmus ovládání 39
6.4.2 Reálný model 40
6.4.3 Testovaní funkcí modelu 41

7 ZÁVĚR 43

8 SEZNAM POUŽITÉ LITERATURY 45

9 SEZNAM ZKRATEK 47

10 SEZNAM PŘÍLOH 49

1 ÚVOD

Původním záměrem bylo navržení software pro řízení dvouosého polohovacího stolu, ale z důvodu odkladu se cíle pozměnily. Během této doby požadavek na vytvoření polohovacího stolu zanikl a nebylo možné navázat na původní zadání v plné míře. Téma centrálního řízení mě ale zaujalo natolik, abych dále pokračoval v tomto tématu a odzkoušel možnosti nabízené standardem MotionControl. Problematika centrálního řízení pohonných systémů je pro automatizaci a mechatroniku velice důležité téma. Znalosti principů řízení se využívají pro navrhování manipulátorů, dopravníků, nebo obráběcích zařízení.

Poznatky získané o standartu MotionControl a principech řízení jsou použity pro navržení algoritmu řízení v centrální jednotce. Pomocí programovacího prostředí Twincat3 je naprogramovaná aplikace řízení motorů ve které jsou použity knihovny TC_MC2 obsahující funkční bloky navržené pro práci s pohonnými soustavami. Následně je potřebné vhodně tyto bloky, pro spuštění a ovládání aktuátorů. Součástí programu jsou metody camming a gearing aplikované pro synchronizaci pohybu. Po rozchození, odzkoušení a odstranění chyb na samostatných osách je navržena aplikace pro synchronizaci os. Tato aplikace je následně odzkoušena na reálném modelu.

Model představuje zařízení k otestování pohybových funkcí motorů, na kterém můžeme jasně vidět případné nepřesnosti v ovládání. Aby bylo možné motory ovládat, bude celý pohonný systém řízen s použitím PLC spolu s kartou Beckhoff EL7342 určenou pro řízení dvou DC motorů. Počítač, na kterém poběží virtuální prostřední, komunikuje s řídící jednotkou a následně všemi komponenty přes protokol EtherCAT. Práce se také zabývá zkoumáním přesnosti řízení při použití různých motorů a parametrů ovlivňujících motory, enkodéry i regulátory.

Výsledná aplikace obsahuje několik testovacích funkcí, které se zaměřují na jednotlivé typy pohybů, které můžeme při pozičním polohování využít.

2 PLCopen

V dnešní době hraje v průmyslu, čím dál větší roli automatizace. Kvůli tomu vzniklo mnoho verzí nekompatibilních systémů a řešení, které postupem času zvyšovaly náklady zákazníků. Jejich úpravy, např. přidávání nových funkcí, nebo opravy byly komplikované a učení se nových systémů bylo matoucí. Všechny tyto problémy způsobují zpomalení vývoje a tvorby nových aplikaci. Aby se zvýšila kompatibilita systémů a řešila se unifikace programování PLC, vzniklo sdružení PLCopen, které následně definovalo standart IEC 61131. Z toho důvodu byla zapotřebí norma, která by zvýšila efektivitu během vývoje a zároveň zvýšila kvalitu aplikací [5].

2.1 Hlavní činnosti

PLCopen je organizace zabývající se řízením v průmyslu. Založila standardy pro řízení pohybu (Motion Control) knihovny, bezpečnost a XML specifikace. Výsledkem je zjednodušení návrhů řešení a nezávislost hardwaru na řídícím kódu. Seskupuje asociaci, které má za cíl řešit hlavní problémy a podporovat používání standardů v průmyslu. Aby byl zachován vývoj, sdružuje PLCopen návrhy od komunity, jejichž součástí jsou jak dodavatelé i vzdělávací instituce [5].

2.1.1 Programovací jazyky:

PLCopen se nejvíce zaměřuje na mezinárodní standard IEC 61131-3 [4], který spadá pod normu IEC 61131 [3]. Tato část se zabývá programovacími jazyky v automatizaci. Ty umožnují vývojářům s odlišnými schopnostmi a v rozdílných vývojových prostředích vytvářet různé části projektu: návrh, implementace, testování, instalaci a údržbu. Díky dodržování společné struktury můžou všechny části spolu fungovat [4,5].

V rámci standardu IEC 61131-3 je definováno pět typů jazyků. Každý je jednoznačně určen a není možno jejich strukturu upravovat. Na druhou stranu mají mezi sebou souvislosti, takže jsou částečně zaměnitelné. Dokonce některé vývojové prostředí přímo tuto možnost konverze z jednoho jazyky na druhy podporují (např. Siemens - STEP 7 – Micro/Win) [4,5].

Standard zahrnují SFC (Sequential Function Chart), LD (Ladder Diagram), IL (Instruction List), FBD Function Block Diagram), ST (Structured Text) [4,5].

Function Block Diagram připomíná zapojení funkčních bloků. Podrobné schéma nahrazuje jednoduchým blokovým schématem. Ladder Diagram neboli také jazyk kontaktních schémat pochází z USA a vychází z Ladder Logic. Jeho protějškem v Evropě je Instruction List, který má jasnou podobnost se syntaxí Assembleru. Poslední, Structured Text, využívající strukturu z jazyků Ada, Pascal, C. Ze všech pěti typů je nejvýkonnější co se komplexnosti týče a v případě komplexních řešení se používá jedině strukturovaný text. ST se také využívá pro psaní funkčních bloků, které mohou být následně použity ve FBD [4,5].

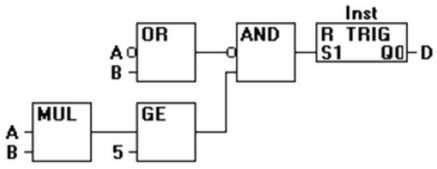
Instruction List

```
LD      17
ST      lint          (* comment *)
GE      5
JMPC    next
LD      idword
EQ      istruct.sdword
STN     test
next:
```

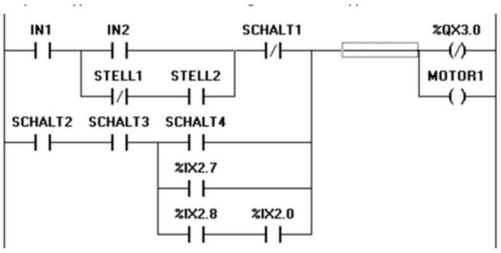
Structured Text

```
IF value < 7 THEN
    WHILE value < 8 DO
        value := value + 1;
    END_WHILE;
END IF;
```

Function Block Diagram



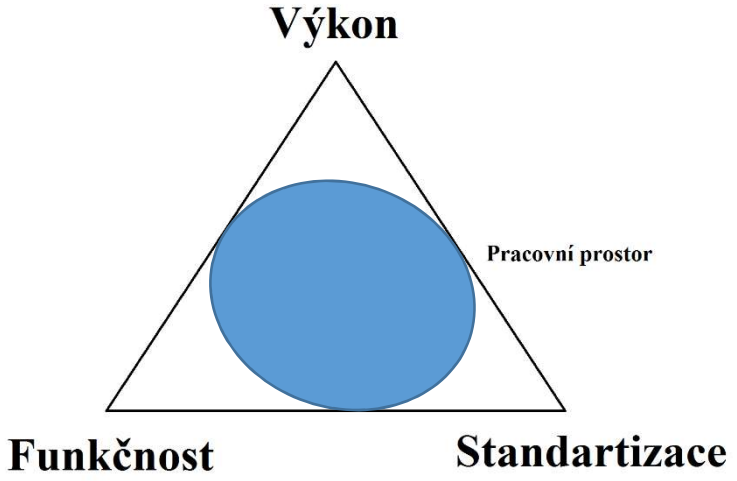
Ladder Diagram



Tab. 1: Přehled struktury čtyř programovacích jazyků [13]

2.1.2 Standardizace rozhraní

Standardizace rozhraní pro různé řešení napomáhá k menší závislosti na hardwaru. Zvyšuje znuvupoužitelnost aplikačního softwaru a zároveň se tím snižují náklady na vyškolení pracovníků a podporu softwaru. Proto uživatelé požadují, aby PLCopen pomáhalo řešit tento problém. Výsledkem tohoto požadavku je Motion Control Task Force, které definuje rozhraní programátora pomocí funkčních bloků [6,7].



Obr. 1: : Trojúhelník možností uživatele (převzato a upraveno z [5])

Trojúhelník na Obr. 2 ukazuje možnosti vývojáře. V ideálním případě se snažíme propojit všechny tři rohy:

- Výkon
- Funkčnost
- Standardizace

V praxi programátoři píší své programy úzce spojené s hardwarem pomoci dedikovaných funkcí s cílem získat maximální výkon, jaký jim prostředí dovoluje. To je omezuje v jejich možnostech vzhledem k použitému hardwaru a znuvupoužitelnosti řídícího softwaru [6, 7].

Druhý typ programátorů rád umožňuje velký rozsah funkcí, které jsou dostupné. To muže být užitečné pro vývoj, nebo správu, ale zřídka kdy vedou k vysokému výkonu

Třetí roh, standardizace, se zaměřuje především na opětovné použití v různých systémech od různých dodavatelů, včetně integrovaných, distribuovaných a síťových systému. Z důvodu obecného charakteru této definice, výkon na různých architekturách může být horší, než kdybychom použili hard coding (programování pro konkrétní účel/hardware). Od standardizace by se nemělo očekávat, že nabídne maximální výkon, ale může se těsně přiblížit maximální funkčnosti. To znamená, že spodní část trojúhelníku je velice krátká [5,6].

První specifikace byla vydána jako samostatná knihovna funkčních bloků pro řízení pohybu. To zahrnovalo funkčnost pohybu pro jednu osu i pro více os, řízení úloh a stavový diagram. Tato specifikace poskytuje uživateli standardní sadu příkazů a struktury nezávislé na poskytnutém hardwaru [6,7].

Tato struktura může být použita na mnoha platformách a architekturách. Díky tomu se muže rozhodnout o použití vhodné architektury v pozdější části vývojového cyklu. Výhody pro výrobce zařízení jsou mimo jiné, nižší náklady na podporu různých platforem, svobodu ve vývoji aplikací pomocí nezávislých metod, aniž by se omezila produktivita stroje. Kromě těchto výhod, se systém údržby stal jednodušší a vzdělávací doba je kratší [5,6].

Jednalo se o významný krok vpřed, který uživatelé postupně akceptovali, stejně jako dodavatelé. Po vydání 1. specifikace bylo jasné, že budou nutné další funkce. Zároveň poskytla základ pro sadu vzájemně souvisejících specifikací (kompletní norma IEC 61131) [5,6]:

Part 1 - PLCopen Function Blocks for Motion Control – Funkční bloky pro Motion Control
Part 2 - PLCopen Motion Control - Extensions, which in the new release 2.0 is merged with Part 1- Rozšíření, ve verzi 2.0 je spojeno s částí 1
Part 3 - PLCopen Motion Control - User Guidelines – Uživatelské pokyny
Part 4 - PLCopen Motion Control – Coordinated Motion – Koordinovaný pohyb
Part 5 - PLCopen Motion Control - Homing Extensions – Rozsšíření pro Homing
Part 6 -PLCopen Motion Control –Fluid Power Extensions – Rozšíření pro ovládání pomocí hydromechaniky

3 MOTION CONTROL

Na základě požadavků aplikace a specifikací projektu jsou inženýři nuceni si vybrat z široké škály řídícího hardwaru. V minulosti to vyžadovalo jedinečný software, který byl vytvořen pro každou aplikaci, i když funkce byly stejné. PLCopen Motion standard poskytuje normované knihovny, které jsou použitelné pro více hardwarových platforem. To snižuje náklady na vývoj, údržbu, podporu a zároveň odstraňuje zmatek [7].

Efektivní standardizace se provádí definováním knihoven pro často používané funkce, čímž se aplikace stávají škálovatelné napříč různými řídicími programy. Díky celistvosti jednotlivých bloků je Motion Control použitelný v různých rozmezích. Slouží jako základní vrstva pro ostatní definice v různých oblastech. Zároveň je otevřen pro stávající i nové technologie [7].

3.1 Základní stavební prvky

Nejdůležitějším prvkem aplikace je POU (Program Organization Unit), programová organizační jednotka. Norma rozlišuje 3 typy POU: funkce (Function), funkční blok (Function block) a program (Program) [8].

Každá programová jednotka POU se skládá ze tří částí. První částí je hlavička POU. Druhou částí je deklarace proměnných s přiřazením daného datového typu. V deklarační části se taky uvádí, které proměnné budou vstupní, výstupní, vstupně-výstupní a vnitřní. Třetí částí POU je vlastní programový kód. Pro psaní kódu využíváme jeden ze 4 programovacích jazyků (IL, ST, LD nebo FBD). Jazyk SFC neslouží přímo pro programování kódu, ale k rozvržení vazeb PLC programu. Jednotlivé části pak mohou být naprogramovány různými jazyky [8].

Funkce neobsahuje paměť, což je zřejmě největší odlišností od ostatních prvků. Při zavolání funkce se stejnými vstupními parametry dostaneme vždy stejný výstup [8].

Funkční blok oproti funkci navíc obsahuje paměť. Příkladem může být například čítač. Hodnota čítače je uložena ve vnitřní paměti funkčního bloku, která se uchová i do příštího zavolání [8].

Program představuje hlavní část aplikace. Obsahuje statickou paměť, stejně jako funkční blok. Navíc oproti funkčnímu bloku umožňuje definovat provázání proměnných na fyzické adresy. Z programu můžeme volat funkční blok i funkci a stejně tak z funkčního bloku je možno volat funkční blok a funkce. Je však zakázáno volání sebe sama, případně zprostředkované volání sebe sama přes jiný funkční blok. Jinými slovy není dovoleno rekurzivní chování algoritmu a je potřeba jej obejít například použitím cyklů. Z funkce je možno volat jen funkci (opět bez rekurze). Tím je dána jistá hierarchie, v níž je nejvýše program pod ním funkční blok a poslední je funkce [8].

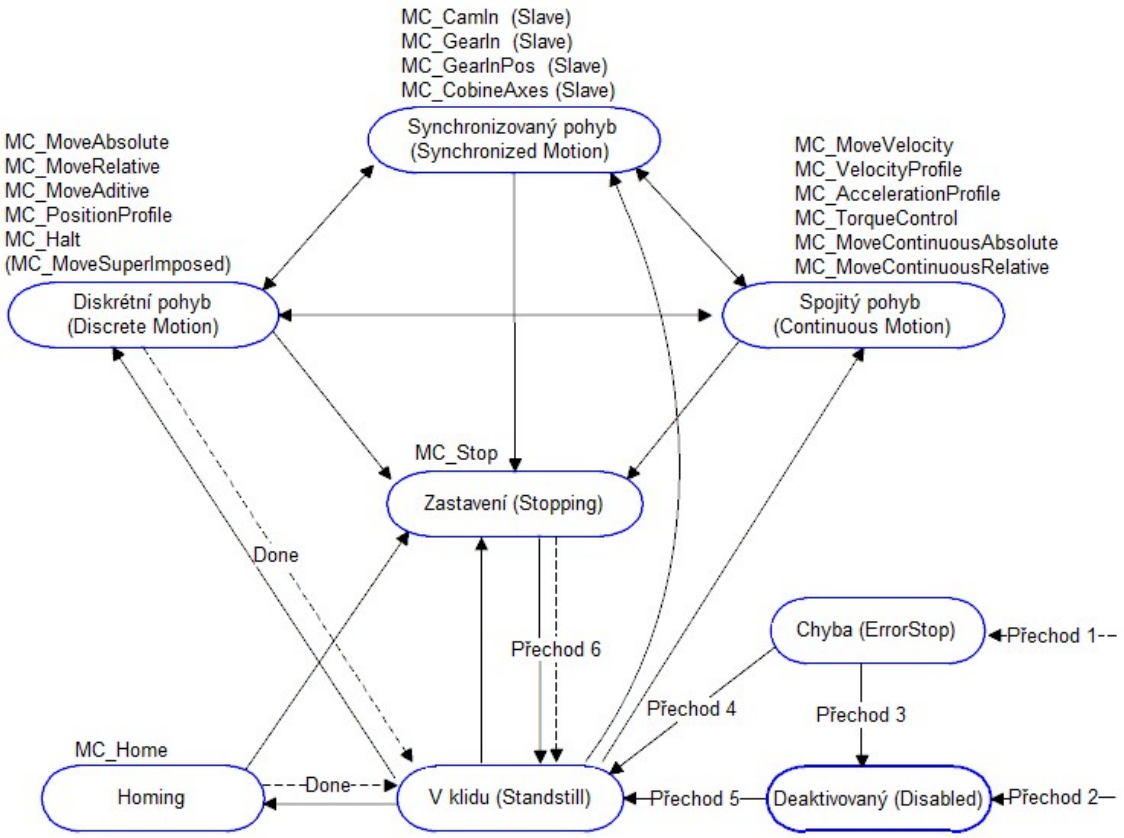
Funkční bloky řízení pohybu používají IEC 61131-3 jazyky. Zároveň je potřeba vzít v úvahu tyto faktory při tvorbě aplikace:

- Jednoduchost - snadnost použití, na programování, instalaci a údržbu
- Účinnost - v počtu funkčních bloků, vzhledem k efektivitě návrhu (a pochopení)
- Konzistence - dle normy IEC 61131-3
- Univerzálnost – nezávislý na Hardwaru
- Flexibilita - budoucí rozšíření / rozsah použití
- Úplnost - není povinné, ale dostatečně

3.2 Stavový diagram

Stavový diagram (State Diagram) je způsob grafického zápisu vývoje systému, který má konečný počet stavů. Takovým systémem může být konečný automat (stavový automat) či další podobné systémy, které vyjadřují stavy určitého objektu a přechody (přechodovou funkci) mezi nimi. Diagram poskytuje sadu elementů pro popis chování systému, který je vyjádřen průchodem stavy[6].

Následující diagram (Obr. 3) definuje chování os, když je současně aktivováno více funkčních bloků. Tato kombinace pohybových profilů je užitečná při programování více komplikovaných řešení, nebo ke zpracování výjimek v rámci programu. Základním pravidlem je přistupovat k pohybovým příkazům vždy sekvenčně, dokonce i za předpokladu že PLC má schopnost paralelního zpracování v reálném čase. Tyto příkazy pro osu jsou zobrazeny ve stavovém diagramu (Obr. 3)[6].



Obr. 2: Stavový diagram, znázorňují algoritmus fungování pohonů (Převzato a upraveno [6])

Přechod 1	Z jakéhokoliv stavu, pokud nastal error.
Přechod 2	Z jakéhokoliv stavu při zrušení povolení osy bez erroru.
Přechod 3	MC_Reset AND MC_Power.Status = FALSE
Přechod 4	MC_Reset AND MC_Power.Status = TRUE AND MC_Power.Enabl = TRUE
Přechod 5	MC_Power.Enable = TRUE AND MC_Power.Status = TRUE
Přechdo 6	MC_Stop.Done AND MC_Stop.Execute = FALSE

Tab. 2: Poznámky ke stavovému diagramu [6]

Osa je vždy v jednom z definovaných stavů (Tab. 1). Každý příkaz, který způsobí pohyb[2], změní stav osy. V důsledku toho se mění způsob, jakým je aktuální pohyb vypočítáván. Změna stavu se okamžitě projeví při vydání odpovídajícího příkazu pohybu [2].

Diagram je zaměřen na jednu osu. Funkční bloky pro více os (MC_CamIn, MC_GearIn, MC_Phasing) můžeme brát z pohledu stavového diagramu jako několik samostatných os v různých stavech. Například CAM-'master' muže být ve stavu 'ContinuousMotion' (plynulý pohyb). Odpovídající 'slave' je ve stavu 'SynchronizedMotion' (synchronizovaný pohyb). Připojení 'slave' osy k 'master' ose, nemá vliv na 'master' osu[6].

Šipky ve stavovém diagramu ukazují možné přechody mezi stavy. Stavové přechody způsobené kódem jsou znázorněny pomocí plných šipek. Přerušované šipky se používají pro přechody, které se vyskytnou, když je ukončen příkaz pro osu, nebo pro přechod související s funkcí systému (chyby, poruchy, přerušení). Pohybové příkazy, které mění stavy os, jsou uvedeny nad jednotlivými stavy. Tyto příkazy mohou být také použity, i když se už osa v daném stavu nachází[6].

Disabled	Stav 'Disabled' popisuje počáteční stav osy. V tomto stavu není osa ovlivněna FB. Napájení je vypnuté a není žádná chyba. V případě že voláme MC_Power s parametrem 'Enable=True', změní se stav na 'Standstill'. Pokud voláme ve FB MC_Power 'Enable=False' v jakémkoliv stavu kromě 'ErrorStop', změní se stav osy na 'Disabled' ať už přímo, nebo prostřednictvím jiného stavu. Všechny pohybové povely na ose jsou přerušeny.
ErrorStop	'ErrorStopErrorStop' má nejvyšší prioritu. Je použit v případě chyby. Osa může být povolená nebo zakázaná, nicméně pokud je pořád aktivní chyba, stav 'ErrorStop' zůstává. Záměrem 'ErrorStop'ErrorStop' stavu je skutečnost, že osa se zastaví hned, jak je to možné. Neexistuje žádný další příkaz, který by byl akceptován, dokud není proveden restart. Přechod na ‚ErrorStopErrorStop' odkazuje na chyby v ose a jejího ovládání. Nikoli z instance funkčního bloku. Tato chyba osy se muže odrazit i ve výstupním parametru 'Errors' funkčního bloku (FB).
StandStill	Napájení je povolené, nevyskytují se žádné chyby a neprovádí se žádný pohybový příkaz.

Poznámky ke specifickým FB

MC_Stop	Voláním FB MC_Stop změníme stav na 'Stopping' a pak následně do 'StandStill Stav 'Stopping' je aktivní do té doby, dokud je na vstupu MC_Stop 'Execute = True'.
MC_MoveSuperimposed	'MC_MoveSuperimposed' vydaný ve stavu 'Standstill' mění stav osy na 'Dicrete-Motion'. Použití v jiném stavu nic nezpůsobí.
MC_GearOut, MC_CamOut	Mění stav osy z 'SynchronizedMotion' do 'ContinuousMotion'. Použití těchto příkazů v jiných stavech způsobí chybu.

Tab. 3: Poznámky o stavech, které mohou nastat ve FB [6]

3.3 Zpracování chyb

Veškerý přístup k ovládání pohonu, nebo pohybu, je zprostředkován pomocí funkčních bloků. Interně jsou tyto FB schopny základní detekce chyb na vstupních datech. Na této definici je závislá i implementace. Například, pokud je nastavena ‘MaxVelocity‘ (maximální rychlost) na 6 000 a rychlost na vstupu je 10 000, systém zpomaluje nebo dojde k chybě. V případě, kdy je inteligentní pohon propojen přes síť, je pravděpodobně parametr ‘MaxVelocity‘ uložen v pohonu. FB musí dbát na to, že zpracovává chybu generovanou v motoru. S jinou implementací hodnota ‘MaxVelocity‘ může být uložena lokálně. V tomto případě bude FB generovat chybu lokálně [6].

Pokud máme více bloků, které mohou vysílat chybové zprávy existují dvě možnosti jak je zpracovávat. První možností je využití centralizovaného řešení, kde se všechny chyby vyhodnocují najednou To umožnuje zjednodušit programování FB, ale nelze s přesně určit kde přesně chyba vznikla. Naopak decentralizované zpracování dává možnost různých reakcí v závislosti na funkčním bloku, ze kterého chyba pochází, protože chyba se vyhodnocuje okamžitě u FB [6].

4 FUNKČNÍ BLOKY

Aby bylo možné FB propojovat, spouštět a celkově s nimi pracovat je potřeba aby měly vstupní, a výstupní parametry přes které je budeme ovládat. Zároveň musí být stanoveny vnitřní pravidla, které dodržují všechny FB spadající pod normu IEC 61131-3 [6].

4.1 Základní informace

Vstupní parametry
Spustit bez průběžné aktualizace (Execute without Continuous Update) - parametry jsou použity s náběžnou hranou na vstupu ‘Execute‘. K úpravě libovolného parametru je třeba změnit vstupní parametr a znovu aktivovat vstup ‘Execute‘[6].

Spustit s průběžnou aktualizací (Execute combined with ContinuousUpdate) - parametry jsou použity s náběžnou hranou na vstupu ‘Execute‘. Parametry mohou být průběžně aktualizovány, dokud je ‘ContinuousUpdate‘ aktivní.*Aktivace* (‘Enable‘) – parametry jsou použity s náběžnou hranou vstupu ‘Enable‘ a mohou být modifikovány nepřetržitě [6].

Vstupy přesahující meze použití
V případě, že funkční blok je řízen parametry, které vedou k porušení aplikačních limitů, jsou vstupy omezeny systémem nebo FB vygeneruje chybu. Následky této chyby pro osu jsou specifické podle aplikace, tudíž by měla být řešena programem aplikace [6].

Chybící vstupní parametry
Podle normy IEC 61131-3, jestliže některý parametr vstupů FB chybí, pak se použije hodnota z předchozího volání instance. Pokud jde o první volání, použijí se inicializační hodnoty[6].

Akcelerace, zpomalení, vůle
Jsou-li vstupy nastaveny na 0, výsledek je závislý na implementaci. Jedna z možností vede do chyby, druhá možnost končí v upozornění (přes dodavatele konkrétního vstupu), další vede do editoru. Jinou možností je převzít hodnoty z ‘AxisRef‘‘ nebo samotné jednotky. Poslední možností je použití maximální hodnoty. I když je nulová hodnota systémem přijata, je třeba tuto vlastnost požívat s opatrností. Obzvlášť pokud je cílená kompatibilita [6].

Exkluzivita výstupu
Při příkazu ‘‘Execute‘ se výstupy ‘Busy‘, ‘Done‘‘, ‘Error‘ a ‘CommandAborted‘ vzájemně vylučují. Pokud je ‘Execute = True‘, jeden z těchto výstupů musí být také ‘True‘. Pouze jeden z výstupů ‘Active’, ‘Error’, ‘Done’ a ‘CommandAborted‘ může být nastaven v jednu chvíli. . Výjimkou je funkce MC_Stop, kde lze nastavit současně položky ‘Active’ a ‘Done’.
Pokud je aktivní ‘Enable,‘, výstupy ‘Valid‘ a ‘Error‘ se vzájemně vylučují. Pouze jeden z nich muže být ve FB ‘True‘[6].

Stav Výstupu
Se sestupnou hranou na ‘Execute‘, výstupy ‘Busy’, ‘Done’, ‘Error’ a ‘CommandAborted’ jsou resetovány. Nicméně sestupná hrana nezastaví ani dokonce neovlivní uskutečnění aktuálního FB. Musí být zajištěno, že odpovídající výstupy budu aktivní alespoň po dobu jednoho cyklu, a to i v případě, že ‘Execute‘ je resetován před dokončením funkčního bloku.
Pokud je funkční blok znovu aktivován před tím než je dokončen, FB nevrátí žádnou odpověď typu ‘Done‘ nebo ‘CommandAborted‘ kvůli předchozí akci.
S povolením ‘Enable‘ jsou výstupy ‘Valid’, ‘Enabled’, ‘Busy’, ‘Error’ a ‘ErrorID’ resetovány se sestupnou hranou okamžitě, jak je to možné [6].

Chování výstupu ‘Done‘
Výstup ‘Done‘ se aktivuje po úspěšném dokončení příkazu. Pokud na jedné ose funguje více funkčních bloku, platí následující. Pokud je jeden pohyb přerušen jiným, aniž by dosáhl finální pozice, na prvním FB se ‘Done‘ nenastaví [6].

Chování výstupu ‘Busy‘
Při ‘Execute‘ může mít každý FB výstup ‘Busy‘, který informuje o tom že FB není dokončen a můžeme stále čekat nové výstupní hodnoty. ‘Busy‘ je nastaven spolu s náběžnou hranou ‘Execute‘ nebo ‘Enable‘. Je aktivní, dokud FB provádí jakoukoliv akci. Další možností, jak vynulovat výstup ‘Busy‘ je aktivace výstupu ‘Done’ nebo ‘Aborted‘. Dokud je ‘Busy‘ aktivní, výstupy se stále můžou měnit [6].

Výstup ‘Active‘
Tento výstup se převážně používá u předem načtených FB. Je nastaven ve chvíli, kdy FB převezme kontrolu nad osou. U režimu bez vyrovnávací paměti muže mít stejnou hodnotu jako ‘Busy‘. Na jedné ose může být více FB ‘Busy‘, ale jen jeden ‘Active‘. To neplatí u FB, které fungují paralelně, tedy kde může být aktivních více FB souvisejících s jednou osou [6].

Chování výstupu ‘CommandAborted‘
‘CommandAborted‘ je aktivován, jestliže je přikázaný pohyb přerušen jiným pohybem. Výstup se resetuje po vynulování ‘InVelocity‘ [6].

’Position‘ a ‘Distance‘

‚Position‘ (poloha) je hodnota definovaná v rámci souřadnicového systému. ‘Distance‘ (vzdálenost) je relativní měřítko související s technickými jednotkami. Vzdálenost je rozdíl mezi dvěma polohami [6].

Řešení chyb

Všechny bloky mohou mít dva výstupy, které řeší chyby ve FB. ‘Error‘ se stává aktivní ve chvíli, kdy nastane chyba. ‘ErrorID‘ je rozšiřující parametr, který napomáhá k určení chyby. Celkově mohou nastat tři typy chyb. První pochází z FB, ruha muže nastat během komunikace a poslední se vztahuje k pohonu [6].

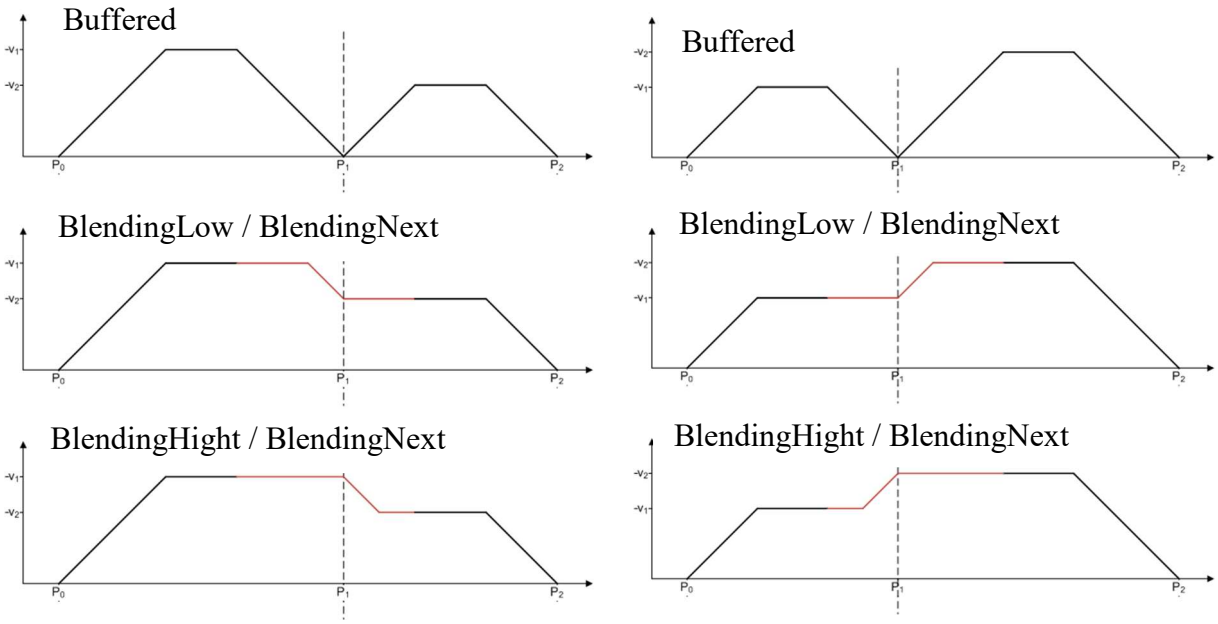
4.2 BufferMode

Některé FB mají na vstupu ‘BufferMode’, který slouží k určení, jak mají být zpracovávaný po sobě jdoucí příkazy z několika FB. Může definovat, zda jeden příkaz může přerušit druhý (režim bez fronty), nebo následující příkaz se provede až po ukončení předchozího (režim fronty). Ve frontě dále specifikuje přechody mezi jednotlivými příkazy. Toto nastavení je označeno jako Blending, které určuje rychlost os na přechodu. Další použití ‘BufferMode‘ u FB pro pohyb je pro aktualizaci parametrů. Protože není možné spustit stejný FB s rozdílnými parametry dokud je používán [14].

Pokud se nacházíme v režimu bez fronty, následující příkaz vede k ukončení probíhajícího příkazu. V tomto případě předchozí příkaz nastaví na výstupu ‘CommandAborded‘. Naopak ve frontě navazující program vyčká, dokud právě aktivní příkaz neukončí činnost [14].

‘BufferMode‘ nelze použit v kombinaci s FB pro spojitý pohyb (MC_MoveVelocity). V tomto případě příkaz ve frontě vede k okamžitému ukončení spojitého pohybu, jako je to v režimu bez fronty [14].

- ’Aborting‘ – Výchozí režim, nový příkaz se okamžitě spustí
- ‘Buffered‘ – Nový FB se spustí po dokončení předchozího
- ’BlendingLow‘ – Novy blok se spustí po dokončení předchozího, původní pohyb skončí s nižší rychlostí z obou bloků
- ‘BlendingPrevious‘ – Novy blok se spustí po dokončení přechozího,původní pohyb skončí se svojí koncovou rychlostí
- ‘BlendingNext‘ – Nový blok se spustí po dokončení předchozího, původní pohyb skončí s počáteční hodnotou nového bloku.
- ‘BlendingHight‘ – Nový blok se spustí po dokončení předchozího, původní pohyb skončí s vyšší rychlostí z obou bloků



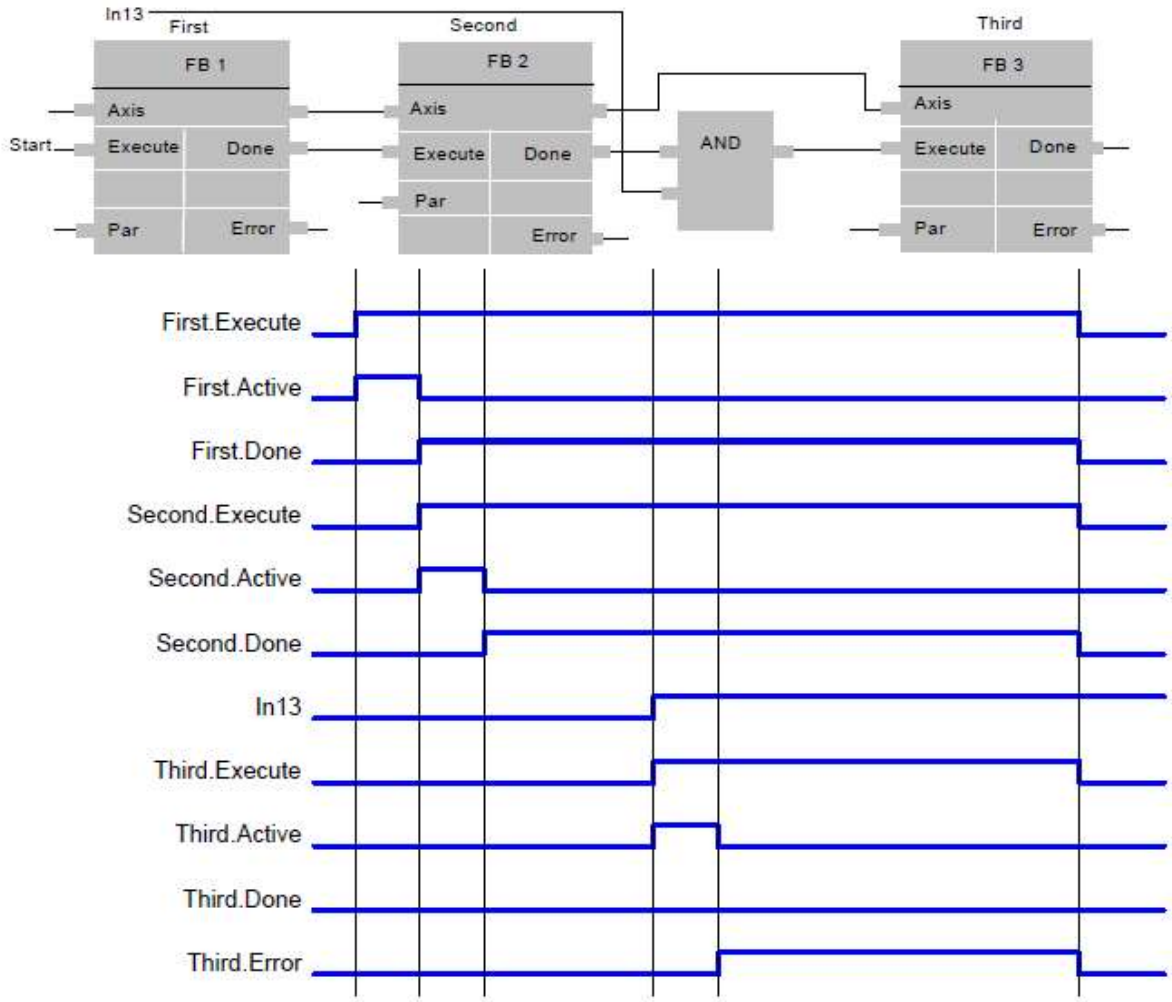
Obr. 3: Grafické znázornění režimů přednačtení pro ovládání rychlosti na ose (Převzato a upraveno[14])

4.3 Data ‘AXIS_REF‘

‘AXIS_REF‘ je struktura, která obsahuje informace o odpovídající ose. Používá se jako vstupní i výstupní proměnná ve všech FB, které mají na starosti pohyb. Obsah této struktury závisí na implementaci, může být i prázdný. Obnovovací frekvence struktury záleží na návrhu. Podle IEC 61131-3 je povoleno přepínat ‘AXIS_REF‘ pro aktivní FB, například z ‘Axis1‘ na ‘Axis2‘. Toto chování se však může lišit v různých platformách a není doporučováno [6].

4.4 Reakce na hranu

Vstup ‘Execute‘ v různých FB se vždy spouští náběžnou hranou. Důvodem je to, že spuštěním hranou mohou být zapsány nové hodnoty do předchozího příkazu. Výhodou této metody je přesná správa provádění pohybových příkazů. Kombinace různých FB je pak jednodušší jak v centralizovaných, tak i decentralizovaných modelech řízení os. Následující diagram (Obr. 5) jeden takový model ukazuje[6].



Obr. 4: Grafické znázornění reakcí na hranu ve funkčních blocích [6]

5 SYNCHRONIZOVANÉ OVLÁDÁNÍ OS

V automatizaci se vyskytují dvě možnosti řízení více os. Osy můžeme řídit nezávisle na sobě. Jednotlivé pohyby jsou řízeny samostatnými FB. Jedna z hlavních nevýhod je, že v případě poruchy jedné z os, druhá pokračuje dál podle programu. To může vést k poškození výrobku nebo celého stroje. Tento problém řeší synchronizace, časová, nebo poziční, kdy jedna osa (’Slave‘) je závislá na druhé (‚Master‘) a osy vzájemně o sobě ví. Pokud tedy nastane problém, jsou schopny ho řešit společně (zpomalení, zastavení,...).

5.1 Elektrické převody (‘GearIn‘)

Používají se ve vícerozměrném prostoru, seskupením více ‘Slave‘ os, s master osou. Toto je možné díky sadě FB s propojenými funkcemi koordinovaného pohybu, vyšší úrovně stavového diagramu, spojením stavových diagramů os do jedné skupiny. Všechny tyto funkce umožnují lepší správu os a plánování trajektorie [6].

‘Master‘ osa může být enkodér, ADC převodník, nebo trajektorie jiné osy. Na základě ‘master‘ se generuje poloha pro ‘Slave‘. Jakmile se pohne ‘Master‘, pohne se i ‘Slave‘. Pokud je ‘Master‘ zařízením zpětné vazby (enkodér, snímač momentu), ‘Slave‘ následuje zpětnou vazbu zařízení. Jestliže je ‘Master‘ trajektorie osy, ‘Slave‘ následuje generátor trajektorie ‘Master‘ osy. Pohne-li se master osa, ‘Slave‘ se nepohne, protože generátor ‘master‘ trajektorie není aktivní [6].

Výhodou elektrického převodu je flexibilita, možnost přidání vice převodových os, nebo možnost změny převodových poměrů podle potřeby dokonce i za běhu programu. Pohybový převod může být zadán relativně nebo absolutně ve formě zlomku. Pohyb ‘slave‘ osy se vypočítá jako násobek převodového poměru a pozice ‘Master‘ osy [6].

5.2 Elektronické vačky (‘Camming‘)

‘Camming‘ funguje podobně jako ‘GearIn‘. Pohybová vzdálenost ‘Slave‘ osy je také závislá na vzdálenosti ‘Master‘. Rozdíl je ve zpracování ‘Master‘/’Slave‘ poměrů pohybovou jednotkou. ‘Camming‘ vytváří flexibilnější profil použitím několika ‘Master‘/’Slave‘ poměrů, které pohybová jednotka zpracovává automaticky. Používá se pro aplikace nelineárních profilů pozic ‘Slave‘ os vzhledem k zařízení ‘Master‘ [6].

Vztahy mezi ‘Master‘ a ‘Slave‘ se ukládají do ‘Camming table‘, podle které se vykonává požadovaný pohyb. ‘Camming‘ můžeme rozdělit na dva hlavní typy:

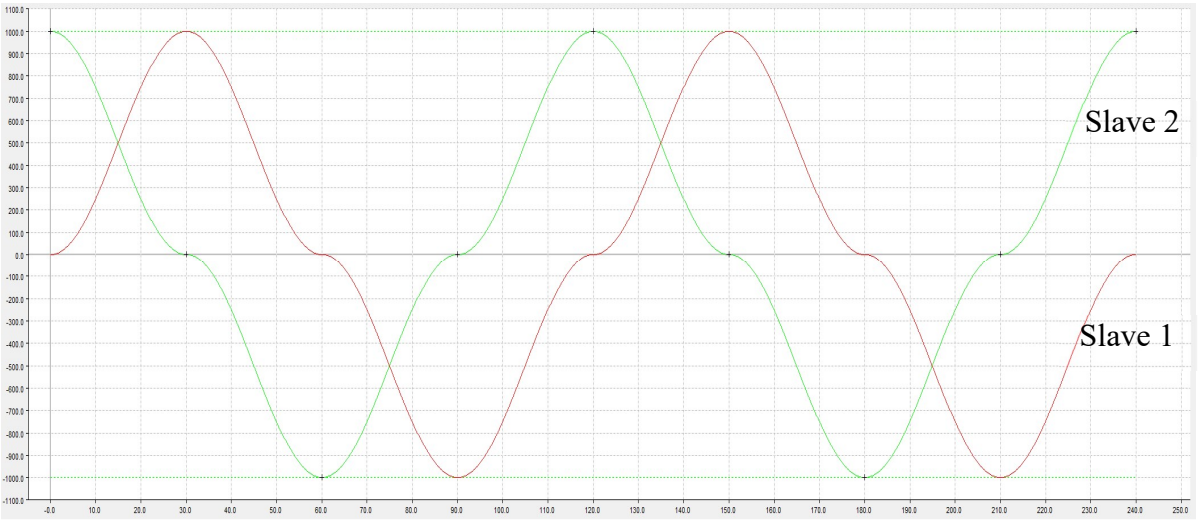
- Periodický (Periodic mode) – CAM profil se průběžně opakuje i za předpokladu že nesouhlasí s modulo. To znamená, pokud je u osy s modulo 360 stupňů použit CAM profil s modulo 90 stupňů, bude proveden čtyřikrát. V opačném případě je použit inverzní princip
- Neperiodický (Non-periodic mode) – CAM profil je spuštěn pouze jednou. Pokud je ‘master‘ pozice mimo profil CAM, osa ‘slave‘ zůstane v synchronizovaném pohybu, ale zůstane na poslední pozici.

‘Camming‘ může být složen z několika CAM tabulek, které mohou být postupně spouštěny. Například náběhová rampa, výrobní cyklus a doběh. Mezi jednotlivým spouštěním CAM křivek můžou být pauzy. Lze také použít vyrovnávací paměť, nebo výstup ‘EndOfProfile‘ pro spuštění dalšího profilu [6].

5.2.1 CAM tabulka

‘Camming‘ muže popisovat jedna tabulka, pro dvourozměrný prostor, kde je pozice ‘master‘ a ‘slave‘ popsána dohromady, nebo dvěma. Pro ‘master‘ a ‘slave‘ samostatně. Je možné za běhu tabulky měnit, nebo upravovat prvky v nich. Generování a vyplňování je prováděno externím nástrojem, který se liší podle dodavatele [6].

Těchto vlastností se využije pro vykreslení kružnice. Pro každou ‘slave‘ osu se vytvoří samostatná tabulka, která zastupuje osu/ motor v aplikaci. Obě tabulky jsou propojeny s virtuální ‘master‘ osou, podle které se přizpůsobují. Jelikož v tabulkách jde použit grafické znázornění, lže využít různé funkce. Hlavní myšlenkou je, aby se obě osy pohybovaly po sinusoidě, ale navzájem byly posunuty o 90 stupňů. Dokud bude aktivní ‘Master‘, osy budou stále kreslit danou kružnici.



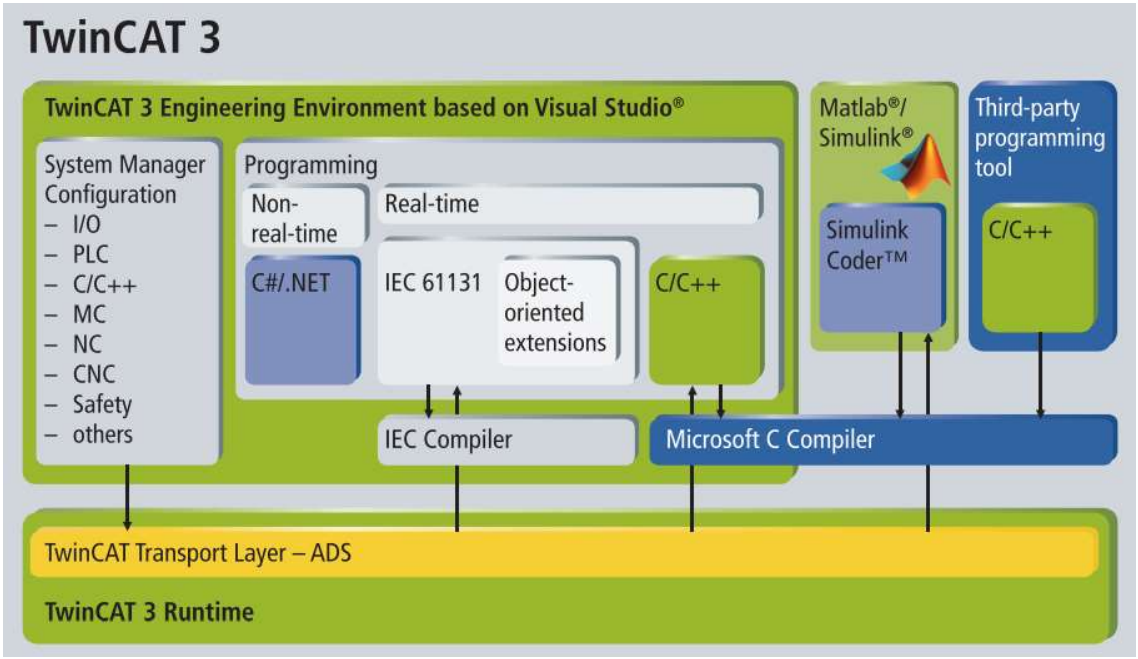
Obr. 5: Grafické znázornění CAM tabulek pro Slave 1 a Slave2

6 OVLÁDNÍ VÍCE MOTORŮ

Aby se mohli řídit všechny motory synchronizovaně a plynule, je využit centralizovaný systém řízení s použitím řídící jednotky od firmy Beckhoff. Díky modulárnímu systému je požitá karta určená přímo k řízení DC motorů. Proto nebylo nutné vytvářet další řídící obvody potřebné k ovládání. Zároveň pomocí zpětné vazby je možno detekovat chyby, poruchy, nepřesnosti a další události, které se následně řeší, ať už změnou rychlostí, trajektorie, nebo odpojením motoru.

6.1 TwinCAT 3 (The Windows Control and Automation Technology)

TwinCAT je vývojové prostředí pro celý systém od firmy Beckhoff. Lze v něm vytvářet software pro programovatelné automaty (PLC), i pro číslicové řízení (NC, CNC). Také nabízí real-time ovládání ve kterém mohou být jednotlivé moduly načteny, zakázány, nebo ovládány. Díky tomu že je prostředí zahrnuto do Microsft Visual Studia je možné aplikace navrhovat v klasických PLC jazycích (IEC 61131-3), vyšších jazycích jako C a C++, nebo Matlab/ Simulink. Tudíž programátor není omezován jazykem prostředí.



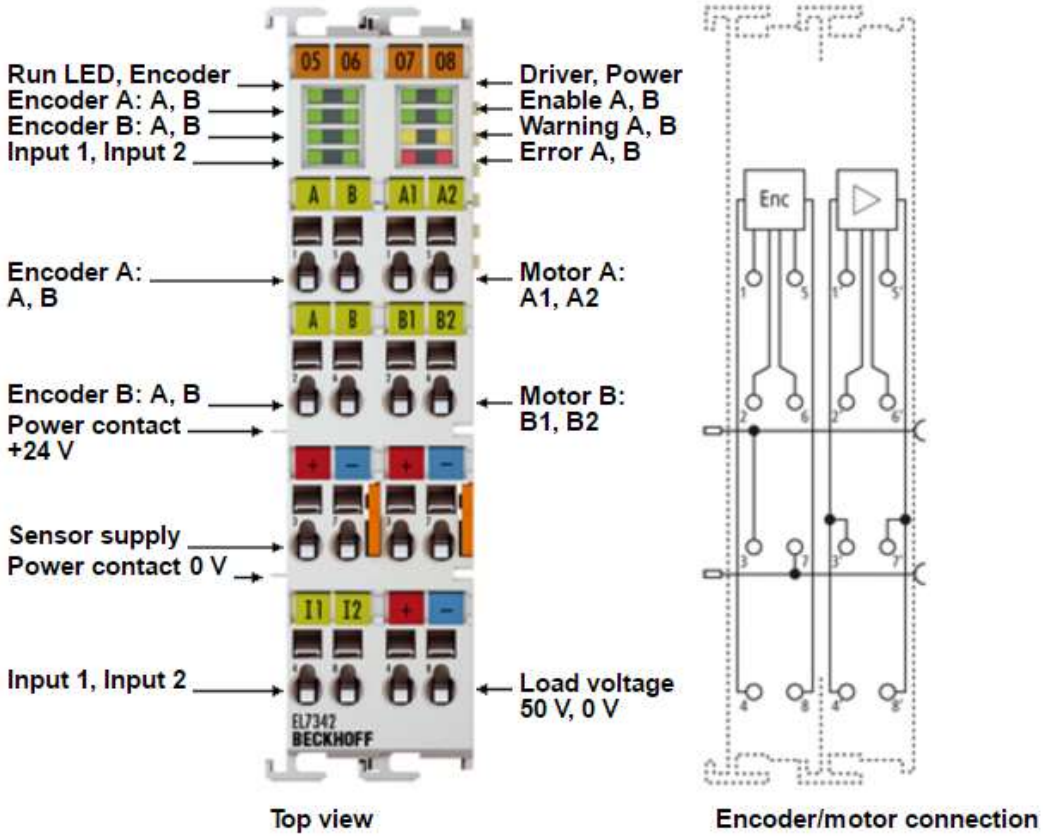
Obr. 6: Základní struktura vývojové prostředí TwinCat3 [11]

6.2 Řídící modul pro DC motory - EL 7342

Modul EL7342 umožňuje přímé řízení dvou DC motorů. Aby se mohly plně ovládat, karta disponuje vstupy pro připojení enkodérů umístěných na osách pohonů. Veškeré vstupy a výstupy jsou galvanicky odděleny a zároveň je výstupní část chráněna proti přetížení a zkratu. Ve vrchní části se nacházejí indikační diody, které nás informuji o stavu a provozu karty.

Počet ovládaných motorů	2
Počet kanálů	2 DC motory, 2 dig. vstupy, vstup pro enkodéry
Jmenovité zatěžovací napětí	8 …50 V DC
Maximální proud na výstupu	2 x 3.5 A
Frekvence PWM	32 kHz s fázovým posunem o 180°
Impulsní frekvence	max. 400,000 inkrementů/s (s čtyřnásobnou rozlišitelností)
Rozlišení ovládání	Max. 10-ti bitový proud, rychlostí 16 bitů
Elektrická izolace	500 V
Vstupní signál enkodéru	5…24 V DC, 5 mA

Tab. 4: *:* Základní parametry karty EL 7342 [12]



Obr. 7: Vyobrazení karty a její schématické zapojení (převzato z [12])

6.3 Fungování řídící jednotky

Informace z motorů i enkodéru jsou přeneseny do karty, ve které probíhá základní zpracování chyb. Karta dále komunikuje s PLC ve kterém se nachází řídící program. Software pak vyhodnocuje chyby, informace o poloze a další parametry. Aktuální polohu porovnává s požadovanou hodnotou vytvořenou generátorem trajektorie. Ten vytváří plán pohybu a každou periodu zasílá nové hodnoty závislé na omezení uživatele. Parametry zahrnují maximální rychlost, zrychlení, zpomalení, ryv (Jerk). Musíme brát v úvahu náběh a doběh na požadovanou rychlost., které souvisí přechody mezi rychlostmi a nepříjemným trháním motoru.
Celý tento proces kontroly, a plánování trasy spravuje řídící část, která vyhodnocuje příkazy od uživatele a předává je dále generátoru jako signály pro spuštění, nebo ukončení pohybu. V řídící části se také vyhodnocují všechny vstupy a výstupy, které uskutečňují inicializaci
Abychom jsme byli schopni vše správně ovládat, bez poškození motorů s co největší přesností je třeba hned v počátcích projektu nastavit v aplikaci parametry motoru a regulátorů.

6.3.1 Parametry motorů

Pro naše potřeby jsou použity DC motory Maxon 118798 a Faulhaber 024 CR, oba dva napájené 24V s enkodéry připojenými přímo na hřídeli. Aby při každém spuštění PLC byly nastavené parametry v CoE (CANopen over EtherCAT), který má na starosti správu parametrů zařízení. Je třeba využít StartUp, protože při vypnutí terminálu se CoE resetuje do továrního nastavení.[9]

Data	Maxon	Faulhaber
Nominální proud [mA]	2250	2000
Nominální napětí [V]	24	24
Přírůstek enkodéru (za 1 ot.)	4096	4096
Max. rychlost motoru [rpm]	6210	5500

Tab. 5: Parametry motorů potřebné k nastavení StartUp

6.3.2 Nastavení os

Při ovládání aktuátorů je snahou docílit co největší přesnosti. Zároveň je cílem minimalizovat překmit a s co nejrychlejším a nejplynulejším nájezdem do požadované hodnoty a pod zátěží by měly polevit, nebo se rozkmitat.

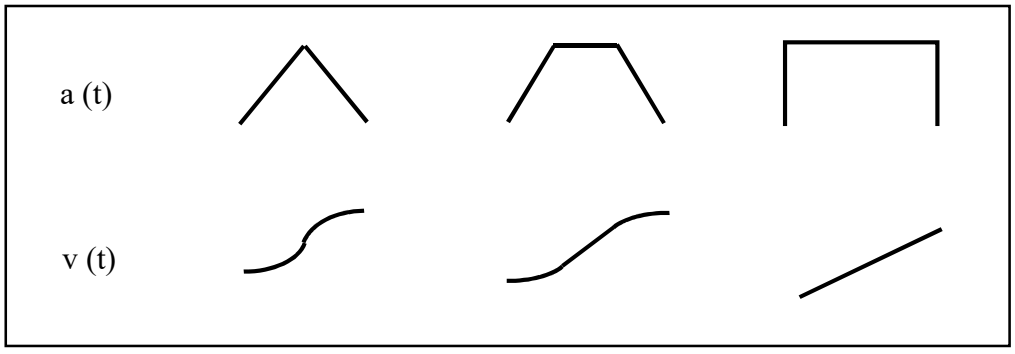
	Maxon	Faulhaber	Poznámky
Referenční rychlost [mm/s]	37260	33000	Parametry os
Maximální rychlost [mm/s]	20000	5500	
Maximální zrychlení [mm/s^2]	5000	2800	
Maximální zpomalení [mm/s^2]	5000	2800	
Zrychlení [mm/s^2]	4200	2700	
Zpomalení [mm/s^2]	4200	2700	
Ryv [mm/s^3] (Jerk)	1095	2250	
Monitorování plynulosti pohybu	False	False	
Měřítko (Scaling factor)	0.087890625	0.087890625	Parametry enkodéru
Proporciální faktor	42	38	Parametr regulátoru

Tab. 6: Parametry potřebné k nastavení os

Referenční rychlost je teoretická hodnota, která by zhruba měla odpovídat reálné hodnotě. Určuje jakou vzdálenost je motor schopný urazit za jednu sekundu. Hodnotu, kterou pak následně můžeme upravit v závislosti na situaci, vypočítáme z maximální rychlosti motoru.

(1) $v_{max} = (v_{max}\text{motoru}\ x\ 360^{\circ})\ /\ 60\ sec = (6210\ x\ 360^{\circ})\ /\ 60\ sec = 37260\ ot/sec$

Parametry zrychlení, zpomalení a ryv nastavují dynamiku motoru (Obr. 8). Nastavují jak rychle má motor najet na stanovenou rychlost, a tím zamezit nepříjemným trháním motoru.



Obr. 8: Typy charakteristik dynamiky motoru

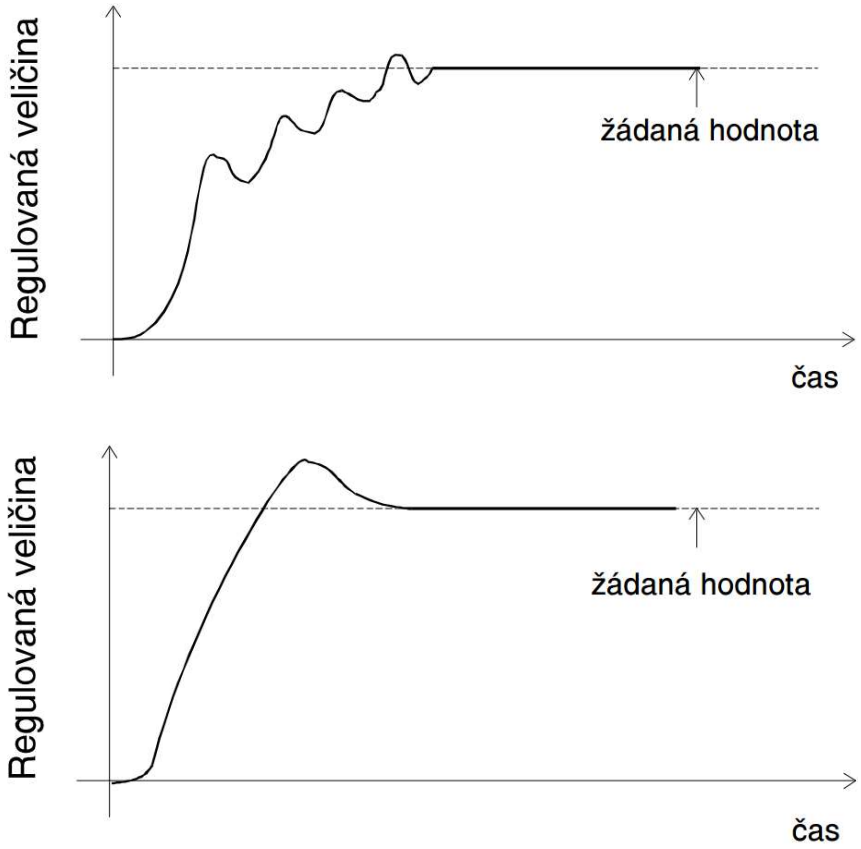
Parametr měřítko definuje o kolik mm se motor posune za jediný inkrement na enkodéru.

(2) $SF =$

$$vzdálenost\ na\ otáčku / přírustek\ enkodéru = 360^{\circ}/4096 = 0.087890625$$

6.3.3 Nastavení proporcionálního regulátoru

Pro tuto aplikaci vystačí proporcionální poziční regulátor, který má zesilovací účinek. Pro jeho použití je třeba určit jeho složku Kv. Čím je Kv faktor větší tím více P-regulátor reguluje. Bohužel při příliš vysokých hodnotách Kv se může regulovaná veličina rozkmitat a ztratit stabilitu (Obr. 9). Je také důležité zda je regulátor schopen udržet nastavenou hodnotu i po zatížení, popřípadě se vrátit po odstranění zatížení.



Obr. 9: Příklady chování proporcionální soustavy při počátečním náběhu na žádanou hodnotu (převzato a upraveno z [10])

Kv	Poloha 5000	Poloha 0	⌀ odchylka	Poznámka
1	5000,098	0,703	0,400	relativně měkký
5	5000,186	0,615	0,400	hodně měkký (dá se točit)
10	5000,010	0,703	0,356	hodně měkký, kmitá pod zátěží
15	5000,010	0,703	0,356	hodně měkký, velké kmitání pod zátěží
20	4999,834	0,703	0,435	tvrdší, pod zátěží kmitá
25	5000,010	0,615	0,313	hodně měkký, velké kmitání pod zátěží
30	4999,834	0,615	0,391	tvrdší pod zátěží kmitá
35	5000,010	0,264	0,137	relativně tvrdý, kmitá
40	4999,922	0,352	0,215	tvrdý, pod zátěží lehké kmitání
45	5000,010	0,264	0,137	tvrdý, s rostoucí zátěží roste kmitání
50	5000,098	0,703	0,400	tvrdý, s rostoucí zátěží roste kmitání
38	4999,922	0,352	0,215	tvrdý, skoro nekmitá

Tab. 7: Testování faktorů na motoru Maxon, kde můžeme vidět optimální hodnotu v rozsahu (35,45)

Kv	Poloha 5000	Poloha 0	⌀ odchylka	Poznámka
1	4999,922	0,615	0,347	relativně měkký
5	4999,922	0,352	0,215	hodně měkký (da se točit)
10	5001,328	0,352	0,840	hodně měkký, kmitá pod zátěží
15	4999,922	0,352	0,215	hodně měkký, velké kmitání pod zátěží
20	5000,010	0,352	0,181	tvrdší, pod zátěží kmitá
25	5000,010	0,352	0,181	Hodně měkký, velké kmitání pod zátěží
30	4999,922	0,352	0,215	tvrdší, pod zátěží kmitá
35	5000,010	0,352	0,181	relativně tvrdý, kmitá
40	4999,922	0,352	0,215	tvrdý, pod zátěží lehké kmitání
45	4999,922	0,352	0,215	tvrdý, s rostoucí zátěží roste kmitání
50	5000,010	0,352	0,181	tvrdý, s rostoucí zátěží roste kmitání
42	5000,010	0,352	0,181	tvrdý, lehké kmitání pod zátěží

Tab. 8: Testování faktorů na motoru Faulhaber, kde můžeme vidět optimální hodnotu v rozsahu (35,45)

6.3.4 Výsledná přesnost motorů

Pro zkontrolování správného nastavení regulátorů a přesnosti motorů, bylo provedeno několik testovacích měření. Ty proběhla ve formě opakovaného najetí na určitou polohu.

Faulhaber

Poloha [mm]	počet opakování		
	5	10	20
100	0,066	0,053	0,017
1 000	0,074	0,075	0,067
5 000	0,078	0,058	0,035
10 000	0,104	0,071	0,032

Maxon

Poloha [mm]	počet opakování		
	5	10	20
100	0,047	0,056	0,043
1000	0,055	0,068	0,029
5000	0,072	0,063	0,050
10000	0,066	0,071	0,032

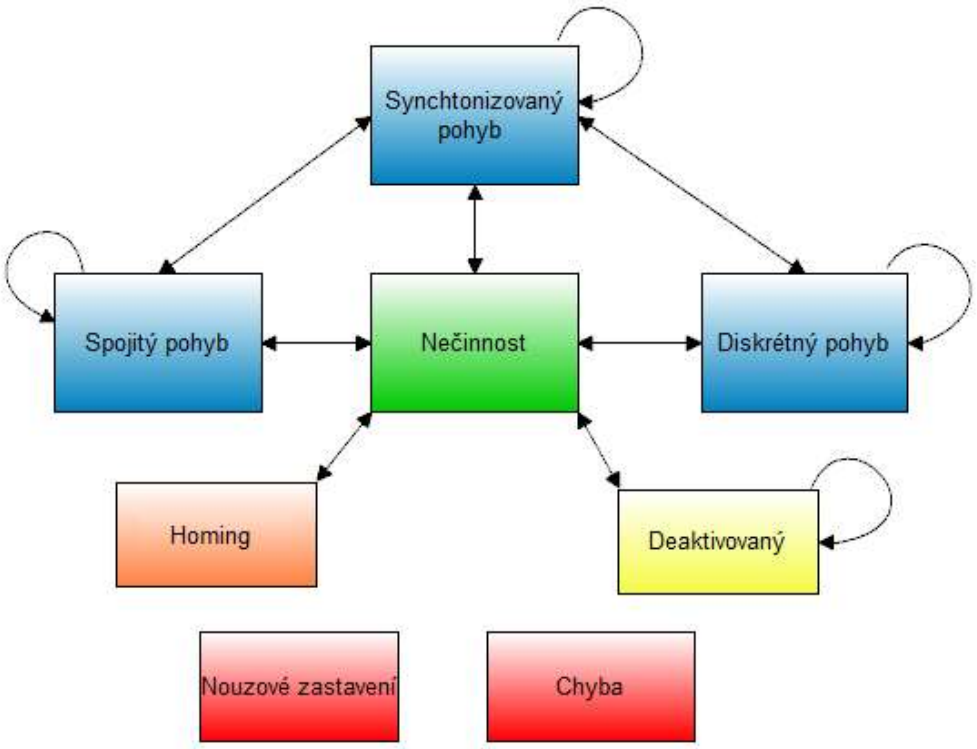
Tab. 9: Průměrná odchylka motorů po n-počtu opakování po najetí na určitou polohu

6.4 Model

V následující části je vysvětlený algoritmus ovládání a blíže se zaměříme na samotnou konstrukci modelu.

6.4.1 Algoritmus ovládání

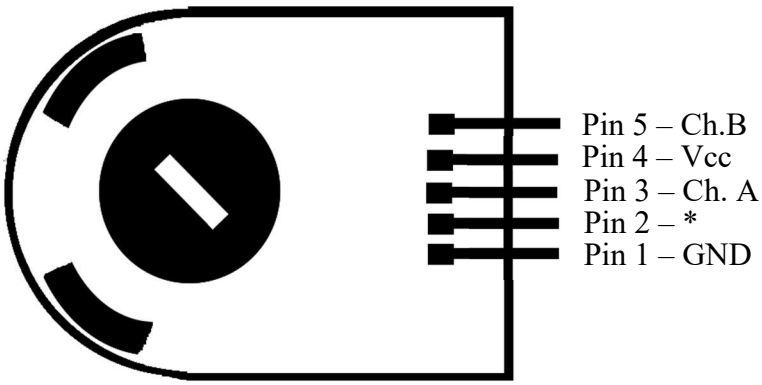
Základní konstrukce je převzata z PLCopen. Skládá se z několika částí Inicializace, Homing, Chyba, Pohybová, Nouzové zastavení. Na začátku každého cyklu se kontroluje, zda jsou motory aktivované a dá se s nimi pracovat. Pokud je nutná inicializace a aktivování motorů, následuje Homing proces. Tím se resetují polohy motorů a náslě je možné s motory přesně pracovat. Zároveň zanikne chyba vzniklá, během vypnutí zařízení kdy se s motory mohlo pohnout. V pohybové části se používají FB určené k nastavení polohy, rychlosti, nebo momentu. Pomocí těchto bloků lze řídit všechny motory. Z toho důvodu je zde vytvořeno další dělení, ve kterém jsou testovány jednotlivé typy pohybů, jako je posun po přímce na danou pozici, vykreslení kružnice, neustály pohyb a regulace rychlosti, nebo jemný posun na jednotlivých osách.



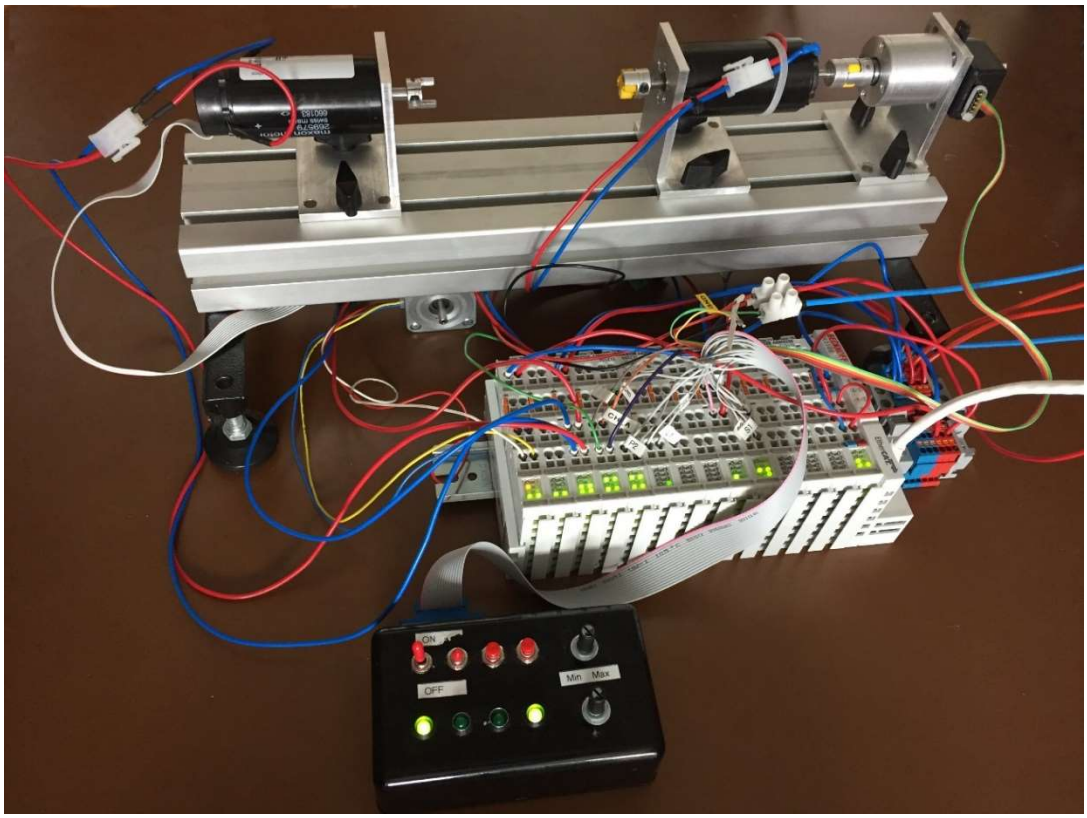
Obr. 10: Stavový diagram programu

6.4.2 Reálný model

V rámci otestování celého systému, je sestaven model. Pohonnou soustavu tvoří pohonné jednotky a centrální řídící jednotka, v našem případě počítač, na kterém běží virtuální PLC v prostředí TC3. V praxi lze využít jakékoliv jiné zařízení reálného času. Každá pohonná jednotka se skládá z motorů, enkodéru. Na základě přechozích testovaní, jsou použity dva motory Maxon s enkodéry Heds 5605 a Maxon MR. Oba pohony jsou umístěny naproti sobě, z důvodu jasné rozlišitelnosti synchronizace pohybu. Tato pohonná soustava je připojena k řídící kartě EL 7342. Dále je k zařízení připojen řídící panel, přes který aktivujeme, zastavujeme a řídíme modul.



Obr. 11: Výstupy enkodéru pro zapojení do karty



Obr. 12: Konstrukce a zapojení modelu

6.4.3 Testovaní funkcí modelu

Z důvodu otestování všech možností modulu jsou vytvořeny jednotlivé funkce zaměřující se vždy na jednu problematiku. Aby se předešlo poruše, nebo poškození motorů, lze mezi jednotlivými funkcemi přepínat jen v režimu Stop. V tomto modu přepínáme mezi :

- Jog (Posunutí) – osa 1
- Jog (posunutí) – osa 2
- MovAbs – pohyb os na dané pozice nezávisle na sobě, bez fronty
- MovAbs – pohyb osy1 na danou pozici, možností dvou rychlostí, testování BufferMode
- GearIn – odzkoušení synchronizace pohybu v režimu virtuální převodovky
- CamIn – odzkoušení synchronizace pohybu pomocí Cam tabulek

Posunutí je provedeno pomocí FB MC_Jog. Po zmáčknutí tlačítka se motor pohne o 10mm. Aby se motor stále otáčel po dobu stisknutí tlačítka, využíváme ve FB parametr Mode do kterého zapisujeme příkaz MC_JogMode_Continuous. Pokud se motor otáčí dostatečně dlouho, dosáhne maximální nastavené rychlosti, což je vhodné pro větší posuny. Jakmile je tlačítko uvolněno, motor se zastaví. Z důvodu nízkého počtu ovládacích prvků je posunutí rozděleno pro každou osu samostatně.

Absolutní pohyb je také spouštěn tlačítky. V obou modech je za sebe poskládáno několik FB pro pohyb. Ale v režimu bez fronty každé tlačítko spouští jinou osou. Aby motor mohl pokračovat v příkazech musí počkat na dokončení právě prováděného.

Naopak ve frontě spínače ovládají jednu osu. První tlačítko spouští pohyb osy, na který navazují další pohyby, protože mají nastavený BufferMode. Ty ale může přerušit FB aktivovaný druhým tlačítkem, na kterém je nastaveno přerušení v případě spuštění.

U synchronizovaného pohybu jsou využity obě metody. První metoda, camming, využívá tabulky popsané dříve (Obr. 6). Pro oba motory jsou nastaveny tabulky, které jsou řízené virtuální osou. Při použití elektrické převodovky je třeba si uvědomit směr pohybu motorů. Pokud se nastaví poměr 1:1. Motory se budou točit opačnými směry z důvodu protilehlého uložení. Aby rotovaly synchronizovaně, stejným směrem, je třeba použít záporný poměr. Opět je zde nastaveno několik FB s různými převody, aby bylo možné jasně rozlišit odlišnosti převodových poměrů.

7 ZÁVĚR

Tato práce využívá základní koncepce, rešerše a následné praktické části. V první části se nachází rešerše z oblasti řízení pohonných systémů, spolu se standardem MotionControl. Dále je popsán základní princip funkčních bloků a fungování modulu pro řízení DC motoru. Na konci teoretické čísti jsou rozebrány možnosti ovládaní pohonů. Pro synchronizované ovládání více motorů je potřeba využit jednu z metod, ať už gearing nebo CAM tabulky. Tyto znalosti jsou následně využity v praktické části pro naprogramování řízení dvou a více motorů. Samotné řešení řídicího systému se skládá ze softwarové a hardwarové části. Hlavní částí softwaru je řídící algoritmus, který vznikl odvozením od stavového digramu PLCopen. Součástí tohoto programu jsou také testovací funkce, pro odzkoušení jednotlivých možností řízení.

Hardware modelu se skládá ze dvou motorů od firmy Maxon ovládané modulem Beckhoff EL 7342. Celý systém řízení zastřešuje centrální řídící jednotka, na které běží aplikace řízení. Jako centrální řídící jednotka je zvolen počítač, na kterém je spuštěn virtuální PLC v prostředí TC3. Ta komunikuje s moduly a následně s pohony přes protokol EtherCAT.

Výsledkem této práce je software, určený pro ovládání pohonů. Model zvládá veškeré pohyby včetně synchronizovaného. Bohužel není možné odzkoušet polohování po kružnici. Vypracování práce spolu s aplikací bylo o rok odloženo a z tohoto důvodu nebylo možné uchovávat komponenty na realizaci. Tudíž ani nebylo možné navázat na zadání v plné míře.

Na práci lze navázat sestavením modelu schopného se pohybovat ve dvou kolmých osách, s následným rozšířením o další motor, třetí plnohodnotnou třetí osu. Celý model by tím tvořil zařízení, které můžeme vidět v praxi.

8 SEZNAM POUŽITÉ LITERATURY

[1] SHELL, R. L. a HALL, E. L. Handbook of industrial automation. New York: M. Dekker, c2000, xii, 900 p. ISBN 0-8247-0373-1.

[2] PAVELKA J., ČEŘOVSKÝ Z., JAVŮREK J.: Elektrické pohony, Nakladatelstvi ČVUT, Praha 2003..

[3] TC1 - Standards. *PLCopen* [online]. [cit. 2017-04-18]. Dostupné z: http://www.plcopen.org/pages/tc1_standards/

[4] IEC 61131-3: a standard programming resource. *Open Automation and Control Group (OACG)* [online]. [cit. 2017-04-18]. Dostupné z: http://www.oacg.co.uk/plcopen.pdf

[5] VOJÁČEK, Antonín. *Programovací režimy pro PLC dle IEC 61131-3 (CoDeSys)* [online]. 2011 [cit. 2017-04-18]. Dostupné z: http://automatizace.hw.cz/programovaci-rezimy-pro-plc-dle-iec-611313-codesys

[6] *Technical Specification PLCopen - Technical Committee 2 – Task Force: Function blocks for motion control*. Version 2.0, Published. PLCopen, 2011.

[7] *PLCopen: PLCopen, motion control, standards* [online]. 2007 [cit. 2017-04-18]. Dostupné z: http://www.plcopen.org/pages/tc2_motion_control/index.htm

[8] MARADA,Tomáš. *Přednáška VPL.4. Programovatelné automaty Časový chod programu, norma IEC-61131, jazyky* [online]. 2014 [cit. 2017-04-18]. Dostupný z: https://moodle.vutbr.cz/pluginfile.php/273178/mod_resource/content/2/VPL_prednaska_04.pdf

[9] *Dokumentation EL73x2: 2 channel DC motor output stage*. 3. 3. 2017. Dostupné také z: https://download.beckhoff.com/download/document/io/ethercat-terminals/el73x2en.pdf

[10] *PID základy regulace*. 2017. Dostupné také z: http://www.hotset.cz/uws_files/navody/pid_regulace_zaklady.pdf

[11] *TwinCAT 3 | Getting started*. 2016. Dostupné také z: https://download.beckhoff.com/download/document/catalog/TwinCAT_3_Booklet.pdf

[12] EL7342 | 2-channel DC motor output stage 50 V DC, 3.5 A. INn: *Beckhoff* [online]. [cit. 2017-05-11]. Dostupné z: https://www.beckhoff.com/english.asp?ethercat/el7342.htm

[13] Programming Languages. *Beckhoff Information System* [online]. [cit. 2017-05-22]. Dostupné z: https://infosys.beckhoff.com/english.php?content=./content/1033/tcplccontrol/html/tcplcctrl_languages.htm&id=

[14] MC_BufferMode. *Beckhoff Information* System [online]. [cit. 2017-05-22]. Dostupné z: https://infosys.beckhoff.com/english.php?content=./content/1033/tcplclib_tc2_mc2/27021597834370571.html&id=

9 SEZNAM ZKRATEK

AC (Alternating Current) –střídavý proud

ADC – většinou ve spojení s převodníkem, Převodník ze střídavého na stejnosměrný proud, většinou se používá pro vyhodnocovaní analogových vstupu číslicovým (digitálním) zařízením SFC (Sequential Function Chart) -

CAM - Computer Aided Manufacturing

CNC (Compuer numeric control) – počítačově čislicově řízení

CoE (CANopen over EtherCAT) – prostředí spravující parametry pro EtherCAT

DC (Direct Current) – Stejnosměrný proud

FBD (Function Block Diagram) – Zapojení Funkčních bloků

IL (Instruction List) – Seznam Instrukcí

LD (Ladder Diagram) – Jazyk Kontaktních schémat

Kv – Proporciální složka

MotionControl - část automatizace zahtnující systémy nebo podsystémy zapojené do řízených pohyblivých částí

NC (Numeric Conrtol) – číslicové ovládaní strojů

PLC –Programovatelný logický automat

POU (Program Oraganization unit) – Programová organizační jednotka, řídící program

ST (Structured Text) – Strukturovaný text

10 SEZNAM PŘÍLOH

cd