VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY

DEPARTMENT OF CONTROL AND INSTRUMENTATION

ŘÍZENÍ NOVÝCH TECHNOLOGIÍ PŘI SVÁŘENÍ OBALOVÝCH MATERIÁLŮ V POTRAVINÁŘSKÉM PRŮMYSLU

THE CONTROL OF NEW TECHNOLOGIES IN WELDING OF PACKAGING MATERIALS FOR THE FOOD INDUSTRY

DIPLOMOVÁ PRÁCE

MASTER'S THESIS

AUTOR PRÁCE — **Bc. David Marek**

AUTHOR

VEDOUCÍ PRÁCE — **Ing. Jan Pásek, CSc.**

SUPERVISOR

BRNO 2017

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Diplomová práce

magisterský navazující studijní obor **Kybernetika, automatizace a měření**
Ústav automatizace a měřicí techniky

Student: Bc. David Marek ***ID:*** 112583
Ročník: 2 ***Akademický rok:*** 2016/17

NÁZEV TÉMATU:

Řízení nových technologií při sváření obalových materiálů v potravinářském průmyslu

POKYNY PRO VYPRACOVÁNÍ:

1. Proveďte analýzu technologií používaných pro sváření obalových materiálů v potravinářském průmyslu a vyberte metodu, nejvhodnější pro balicí stroj, který se má inovovat
2. Pro zvolenou technologii sváření navrhněte její zasazení do stávajícího stroje a navrhněte a realizujte komunikaci potřebných nových zařízení do existujícího řídicího systému
3. Navrhněte a implementujte řídící algoritmus pro požadovanou svářecí funkci v balicím stroji.
4. Řídicí systém doplňte o vizualizaci (HMI) poskytující „chytré“ údaje o výrobě.

DOPORUČENÁ LITERATURA:

[1]Beckhoff Information System [online]. [cit. 2017-01-27]. Dostupné z: https://infosys.beckhoff.com/index_en.htm

[2]PLC Lib: Tc2_EtherCAT: Manuál TwinCAT 3 [online]. [cit. 2017-01-27]. Dostupné z: https://download.beckhoff.com/download/document/automation/twincat3/TwinCAT_3_PLC_Lib_Tc2_EtherCAT_EN.pdf

[3]TC3 TCP/IP: Manuál TwinCAT 3 [online]. [cit. 2017-01-27]. Dostupné z: https://download.beckhoff.com/download/document/automation/twincat3/TwinCAT_3_PLC_Lib_Tc2_EtherCAT_EN.pdf

Termín zadání: 6.2.2017 ***Termín odevzdání:*** 15.5.2017

Vedoucí práce: Ing. Jan Pásek, CSc.
Konzultant:

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

Abstrakt

Hlavním úkolem práce je praktické ověření ultrazvukové metody svařování pro tvorbu sáčků, u kterých je zvýšená pravděpodobnost zboží ve svaru. Většina práce se zabývá naprogramováním bloku pro celkovou obsluhu ultrazvukového generátoru. Součástí řešení je i tvorba vizualizace pro obsluhu generátoru. Poslední částí je navržení a realizace reporting systému, který pomůže zákazníkům získávat data o výrobě.

Klíčová slova

TwinCAT, PLC, ultrazvukové svařování, vizualizace, report, C#, EtherCAT, CoE, SSRS

Abstract

The main task of the thesis is the practical verification of the ultrasonic sealing method for producing bags with a higher probability of goods in the sealing area. Most of the work deals with programming a block for the overall operation of an ultrasonic generator. Part of the solution is creation of visualization of the generator user interface. The last part is designing and implementing a reporting system to help customers get production data.

Keywords

TwinCAT, PLC, ultrasonic welding, visualization, report, C#, EtherCAT, CoE, SSRS

Bibliografická citace:

MAREK, D. *Řízení nových technologií při sváření obalových materiálů v potravinářském průmyslu.* Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2017. 93 s. Vedoucí diplomové práce Ing. Jan Pásek, CSc..

Prohlášení

„Prohlašuji, že svou závěrečnou práci na téma Řízení nových technologií při sváření obalových materiálů v potravinářském průmyslu jsem vypracoval samostatně pod vedením vedoucího diplomové 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 závěrečné práce dále prohlašuji, že v souvislosti s vytvořením této závěrečné práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

V Brně dne **11. května 2017** ……

podpis autora

Poděkování

Děkuji vedoucímu diplomové práce Ing. Janu Páskovi, CSc. za účinnou metodickou, pedagogickou a odbornou pomoc a další cenné rady při zpracování mé diplomové práce. Následující poděkování patří kolegům, kteří svými radami přispěli k bližšímu pochopení dílčích problematik. Jmenovitě Ing. Miloši Vrbickému, Ing. Petru Korendovi, Miloši Machotkovi a Pavlu Zíkovi. Poděkování patří též rodině, která mi byla po celou dobu oporou.

V Brně dne **11. května 2017** ……

podpis autora

Obsah

Seznam obrázků

Seznam tabulek

ÚVOD

Tato práce vznikla jako reakce na problém svařování sáčků v potravinářství. Největším problémem je, že v přítomnosti zboží ve svaru (například prášek, olej, atd.), vzniká nedokonalý svar. Hlavním úkolem je praktické ověření přínosu ultrazvukové metody, která by měla tento problém řešit. Součástí úkolu je výběr vhodného ultrazvukového generátoru s ohledem na firemní řídící systém. Do ŘS naprogramovat funkční blok, díky kterému bude umožněno ovládání generátoru. Blok bude obsluhovat výměnu dat mezi PLC a generátorem. Pomocí vytvořené komunikace vznikne možnost kompletního nastavení a řízení svařovacího cyklu bez užití fyzických I/O. Další důležitou roli ve funkčním bloku hraje navržení vhodného rozhraní mezi blokem a PLC pro jednoduchou implementaci do stávajícího ŘS.

Druhým bodem bude navržení řídícího algoritmu pro splnění požadavků na běh stroje.

Další úkol spočívá v naprogramování části vizualizace, která bude následně implementována do současné vizualizace stroje. Požadavky na vizualizaci jsou takové, že musí obsahovat veškeré potřebné nástroje k nastavení, testování, diagnostice a obsluze generátoru.

Posledním bodem je vytvoření report systému, který rozšíří nabídku produktů společnosti. Reporting bude sbírat a vhodně interpretovat data strojů ve výrobě.

1 TEORETICKÁ ČÁST

1.1 Svařování termoplastů

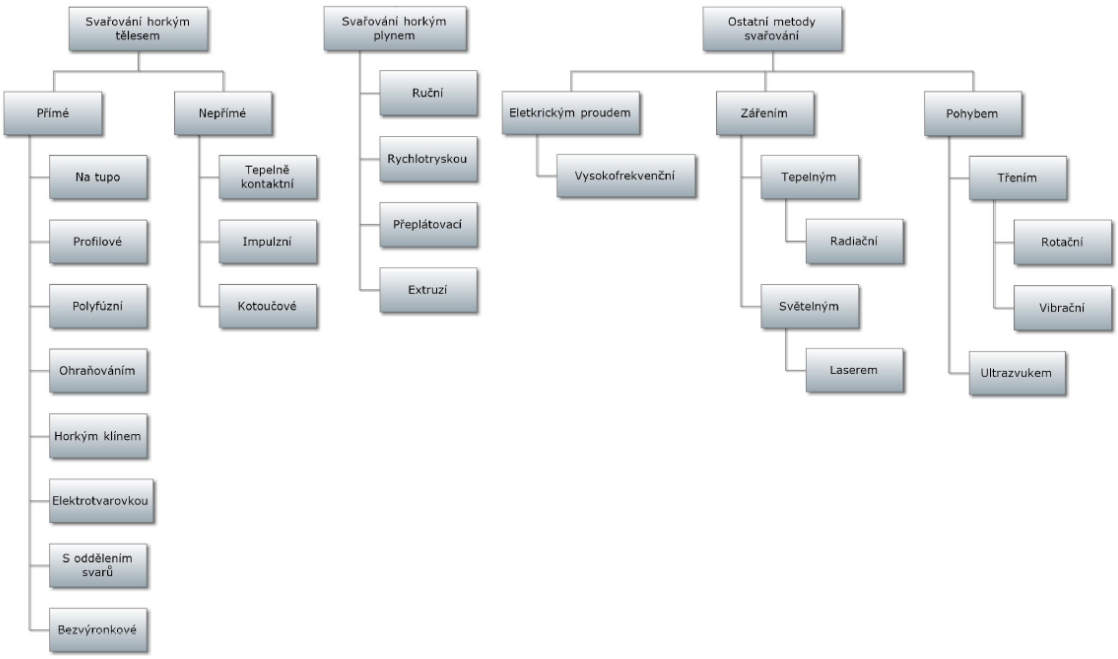
Svařování termoplastů je taková technologie, u které vzniká za použití tepla nebo tlaku nerozebíratelný spoj. Spoj může vzniknout za pomoci přídavného materiálu nebo i bez něj. Potřebnou podmínkou pro vznik svaru je, aby byl materiál schopen přejít do tekutého stavu [1]. Postup svařování se může lišit (kontakt materiálů a následné působení tepla, působení tepla a následný kontakt, kontakt a zahřívání naráz). Potřebné teplo je předáváno přímým kontaktem (např. nahřátým povrchem) nebo na určitou vzdálenost (např. horkým plynem). Základní dělení metod svařování termoplastů se dle způsobu předávání tepla dělí do tří skupin [2].

1.1.1 Dělení svařovacích metod

Základní dělení [2]

- Svařování horkým tělesem
- Svařování horkým plynem
- Ostatní metody svařování

Podrobnější dělení metod svařování [2]



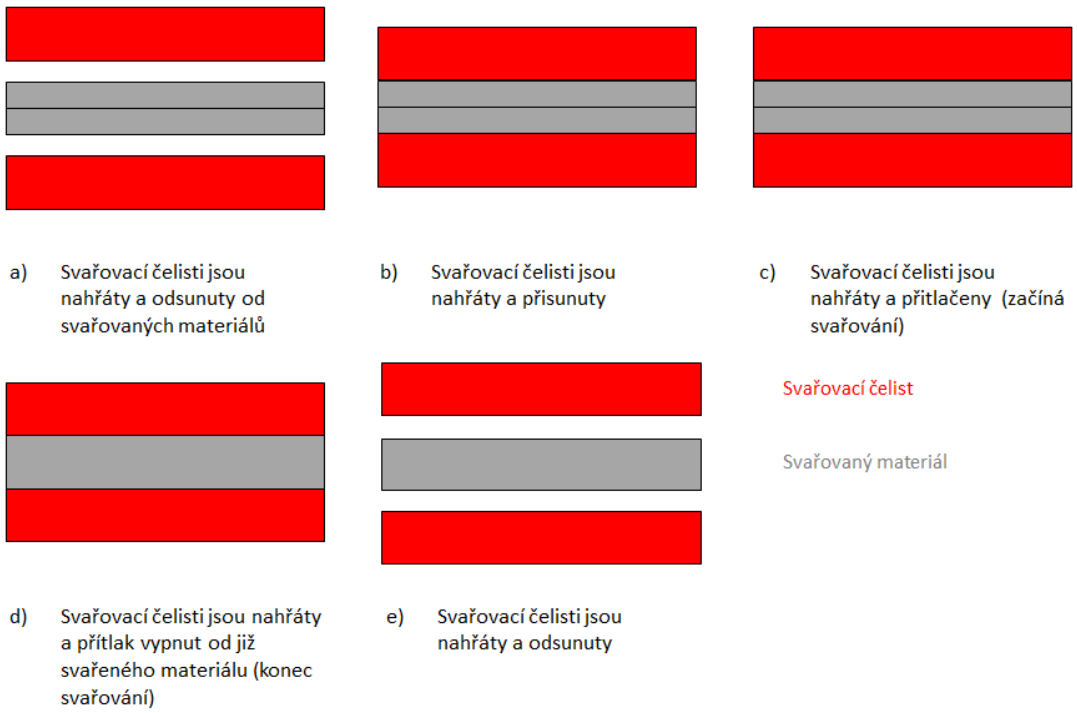
Obr. 1 Metody svařování termoplastů

Pozn.: V této práci jsou níže popsány metody svařování používané ve firmě, pro kterou práce vznikla. Ostatní metody z tabulky výše zde popsány nejsou z důvodu omezení počtu stran práce a kromě toho podrobný výpis dalších metod není stěžejní pro téma mé práce.

1.1.1.1 Nepřímé svařování horkým tělesem

Popis

Horké těleso přenáší svařovací teplo na svařovanou plochu materiálu. Patrona uvnitř čelistí se nahřeje a jeho teplota je udržována. Patrona přímo ovlivňuje teplotu čelistí, které realizují samotné sváření. Z důvodu pomalého chladnutí tělesa je tato teplota regulována i mezi jednotlivými cykly svařování. Z toho plyne omezení doby styku svařovacích čelistí se svařovaným materiálem. Naopak výhodou je skutečnost, že nahřívané těleso může mít veliké rozměry, což umožňuje dosažení vysoké plochy svaru. Nastavitelnými parametry jsou teplota čelistí, doba sváření, svařovací tlak.



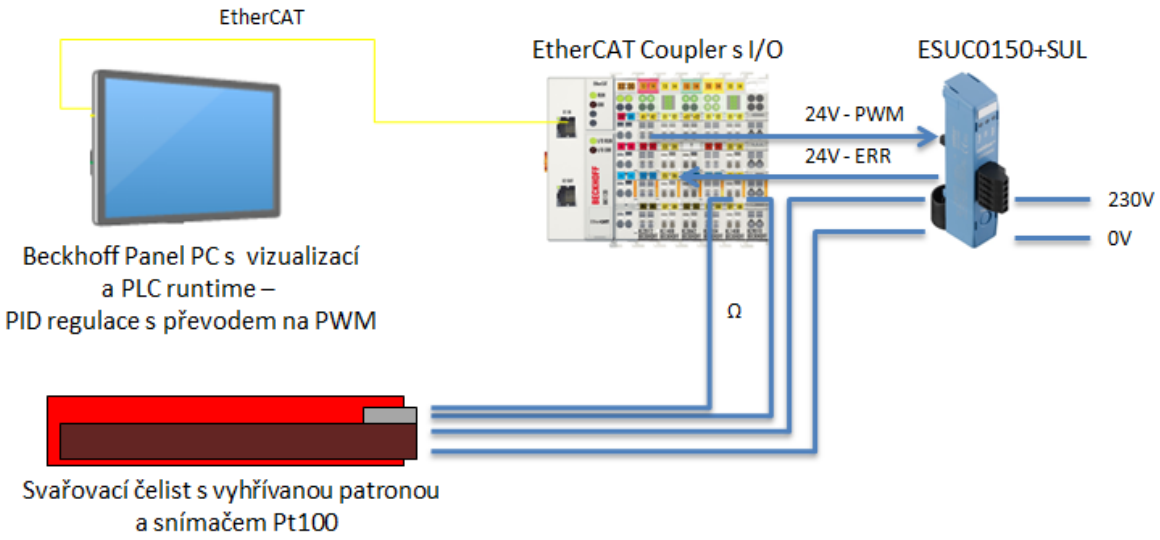
Obr. 2 Svařovací cyklus horkým tělesem

Výhody: možnost sváření větší plochy, nízká cena.

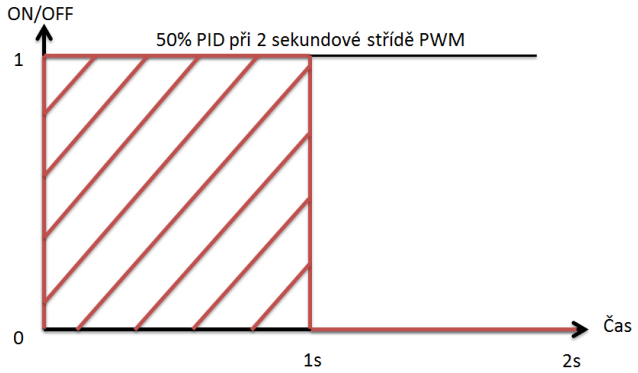
Nevýhody: pomalé chladnutí, dlouhá doba nahřívání tělesa, nelze svářet kontinuálně, potřeba přídavných pohonů (či pneumatiky) pro realizaci cyklu, u podélného svařování je limitující výška čelistí, která ovlivňuje výšku stroje.

Realizace

V PLC probíhá výpočet PID regulace teploty na základě žádané hodnoty z vizualizace a aktuální hodnoty z čidla PT100. Výstup z PID je převeden na PWM (pulzně šířková modulace) s 2 sekundovou střídou, viz obr. 4. Hodnota PWM jde po EtherCATu z PLC runtime na panelovém PC do EK1100 (EtherCAT Coupler) a následně do výstupní karty EL2004. PWM jde binárním signálem z této karty do polovodičového spínače SUL. Ten realizuje nahřívání tepelných patron, uložených ve svařovacích čelistech. Každý polovodičový spínač nahřívá jednu čelist. Na spínači je nasazen diagnostický modul ESUC0150. Z diagnostického modulu jde zpět na vstupní kartu EL1014 binární signál, který reprezentuje vadnou patronu. Vedle patrony je umístěno čidlo PT100, které vede zpět na vstupní kartu EL3204, jehož funkce je popsána výše. Ve vizualizaci lze nastavit teplotu vyhřívání, dobu sváření a tlak (pokud je tlak realizován pomocí servomotorů). V případě tlačení čelistí pomocí stlačeného vzduchu, se tlak nastavuje ručně. Informace z vizualizace jsou pro všechny naše typy sváření stejné, takže je nebudu již dále zmiňovat. Princip je znázorněn na obr. 2.



Obr. 3 Schéma zapojení – horkým tělesem

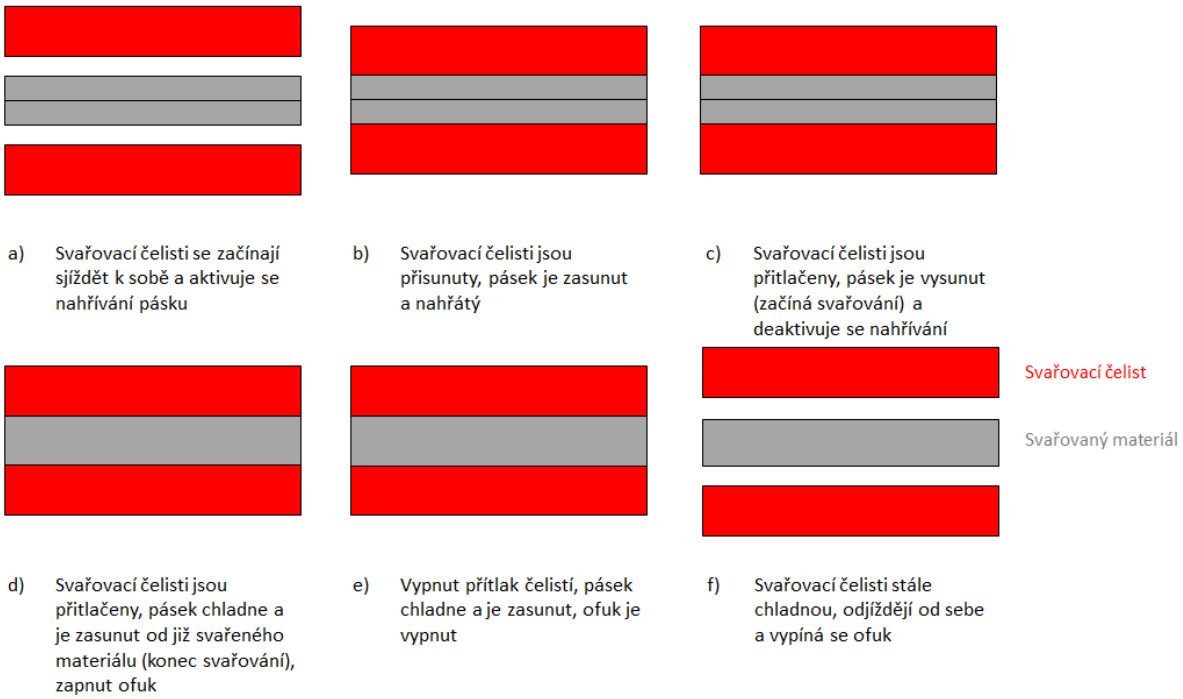


Obr. 4 50% PID při 2 sekundové střídě PWM

1.1.1.2 Impulzní sváření

Popis

Při impulzním sváření se nevyhřívá celá čelist, jako tomu je u předchozího případu, ale jen kantalové pásky, upevněné na přední ploše svařovací čelisti. Díky tomu dochází k mnohem rychlejšímu náběhu teploty a zároveň i rychlejšímu chladnutí. Chladnutí může také dopomáhat ofuk, který může být zapnut již ve fázi, kdy se sice již vypnul přítlak pásku na materiál, ale čelisti stále ještě přidržují sáček.



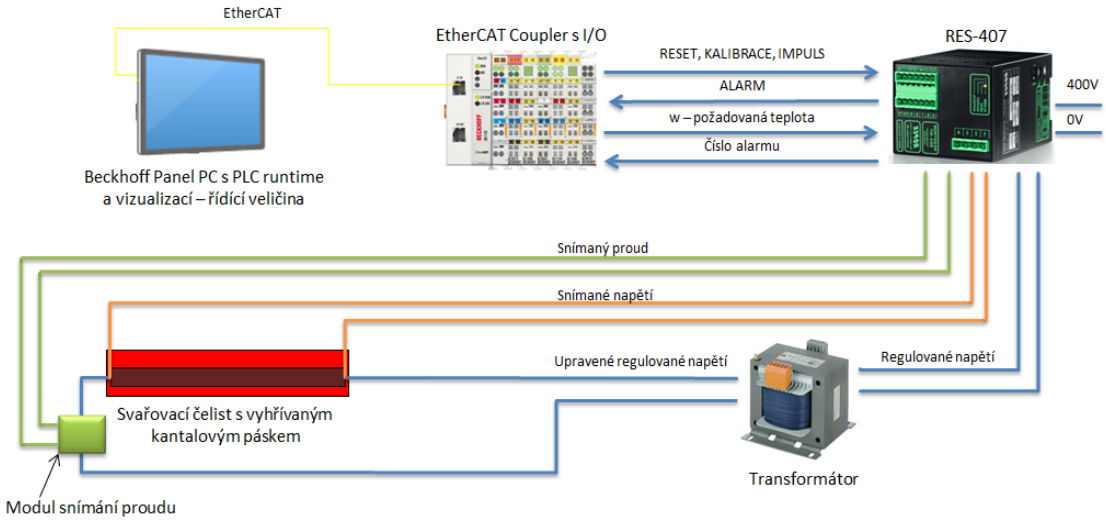
Obr. 5 Svařovací cyklus impulzní metodou

Výhody: rychlejší nahřívání a chladnutí oproti předchozí metodě.

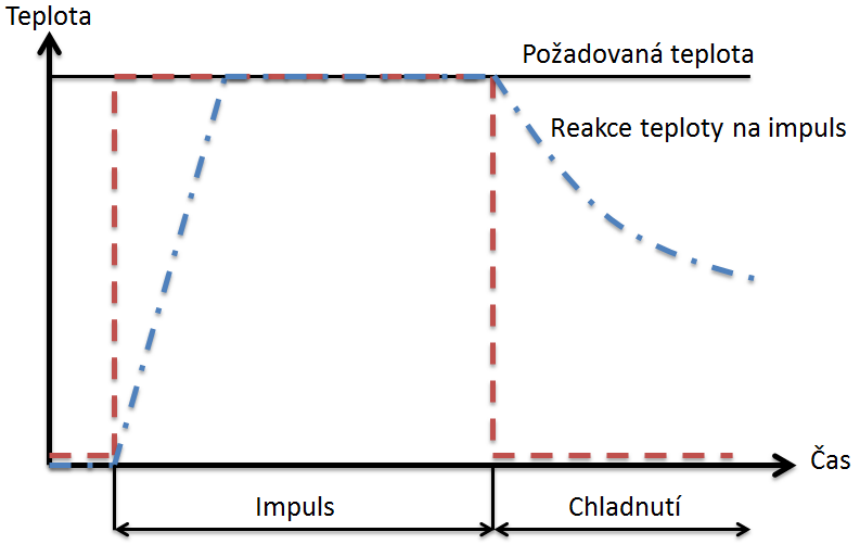
Nevýhody: při vyšším svaru je potřeba více pásků pod sebou, u podélného svařování je limitující výška čelistí, která ovlivňuje výšku stroje, nelze svářet kontinuálně, potřeba přídavných pohonů (či pneumatiky) pro realizaci cyklu.

Realizace

Regulace je realizována v regulátoru RES-407 na základě vstupní hodnoty, která jde z PLC přes EtherCAT Coupler, z výstupní analogové karty a aktuální hodnoty získané z naměřeného proudu a napětí. Z RES-407 jde regulované napětí do transformátoru, kde je následně upraveno pro vyhřívání kantalového pásku na čelistech. Jeden regulátor ovlivňuje vždy protilehlé pásky. To znamená, že například při výšce svaru, u které jsou zapotřebí dva pásky nad sebou, jsou nutné dva RES-407. Jeden z regulátorů ovlivňuje horní pásek na první čelisti a horní pásek na čelisti druhé. Druhý reguluje teplotu na spodních páscích. Mezi transformátorem a páskem je modul, který snímá procházející proud a dává zpětnou vazbu regulátoru. Dále je z pásku snímáno napětí, které je také zavedeno jako zpětná vazba do regulátoru. Pomocí informací o procházejícím proudu a napětí si regulátor spočítá aktuální teplotu. Z PLC do regulátoru jde tedy řídící veličina a binární povely RESET, KALIBRACE, IMPULS. Do PLC je zavedena binární informace ALARM a analogová hodnota vyjadřující číslo chyby. Ohřev probíhá pouze při aktivním vstupu IMPULS. IMPULS je aktivován, když se svařovací čelisti sjíždějí k sobě a deaktivován při dosažení požadovaného tlaku. Závislost teploty na čase při aktivaci impulzu je vidět na obr. 7.



Obr. 6 Schéma zapojení – impulzní metoda

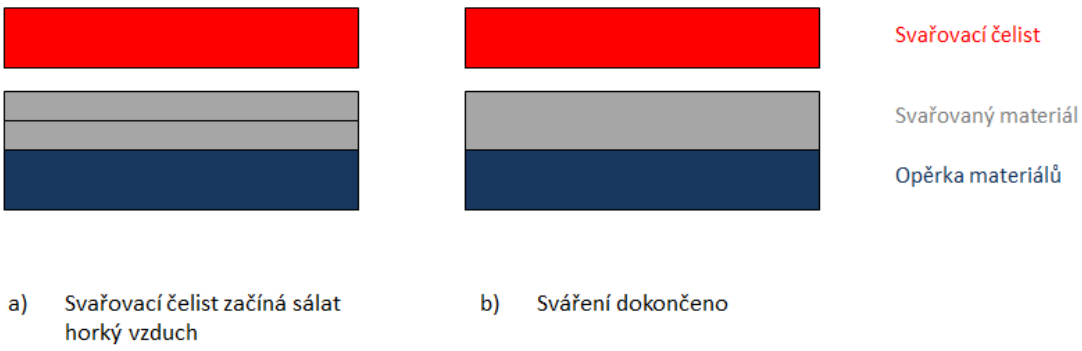


Obr. 7 Průběh teploty v jednom pulzním cyklu

1.1.1.3 Svařování horkovzduchem

Popis

Ke sváření se využívá vzduch, který prochází topným tělesem. Svařování neprobíhá za přímého tlaku, ale v určité vzdálenosti od materiálu. Možností je tedy kontinuální sváření, což může výrazně ovlivnit délku svařovacího cyklu. Dalšími výhodami jsou odstranění potřeby pohybu čelistí (ať už od sebe/k sobě nebo nahoru/dolů) a rychlá tepelná reakce (prakticky ihned při vypnutí/sepnutí elektromagnetického ventilu).



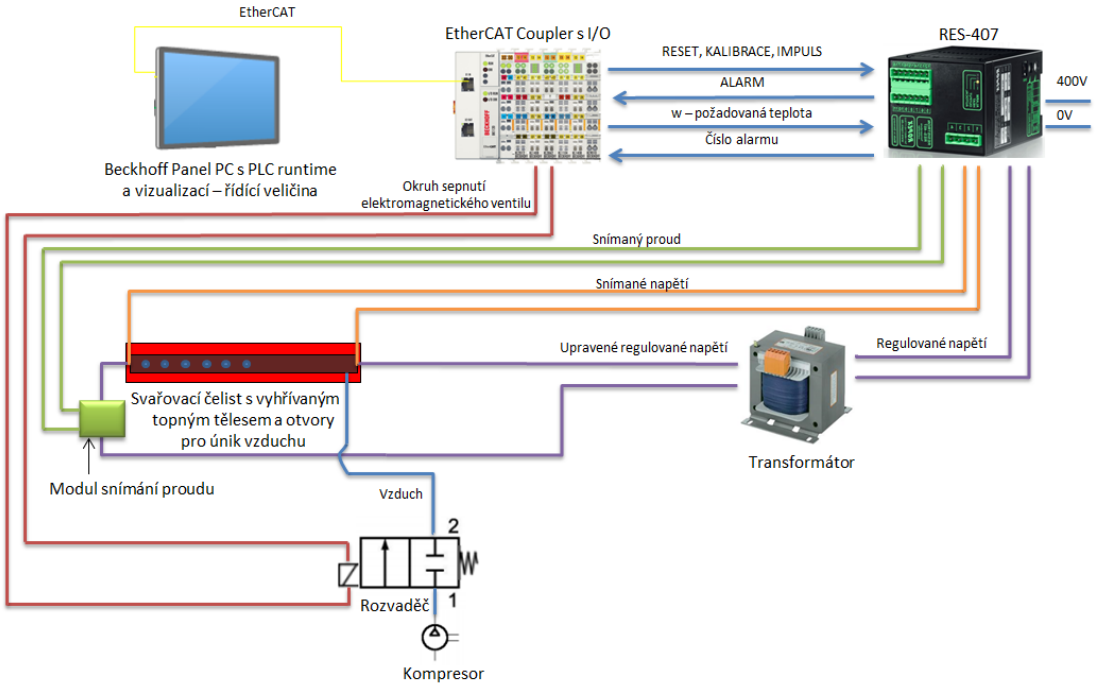
Obr. 8 Svařovací cyklus horkovzduchem

Výhody: možnost kontinuálního svařování, nepotřebnost přidání pohonu čelistí, rychlé tepelné reakce.

Nevýhody: vyšší cena, potřeba instalace dodávky vzduchu procházejícího nahřátým tělesem.

Realizace

Použit je opět regulátor RES-407, mezi nímž a PLC je interface realizován stejně jako u impulzního sváření. Změna je v řízení výstupu z PLC "IMPULS", který nevypínáme mezi jednotlivými svařovacími cykly, ale máme ho trvale sepnutý. Díky tomu je těleso neustále nahříváno na pracovní teplotu. Svařovací cyklus nám v tomto případě určuje sepnutí/vypnutí elektromagnetického ventilu, který nám žene vzduch do nahřátého tělesa.



Obr. 9 Schéma zapojení - horkovzduch

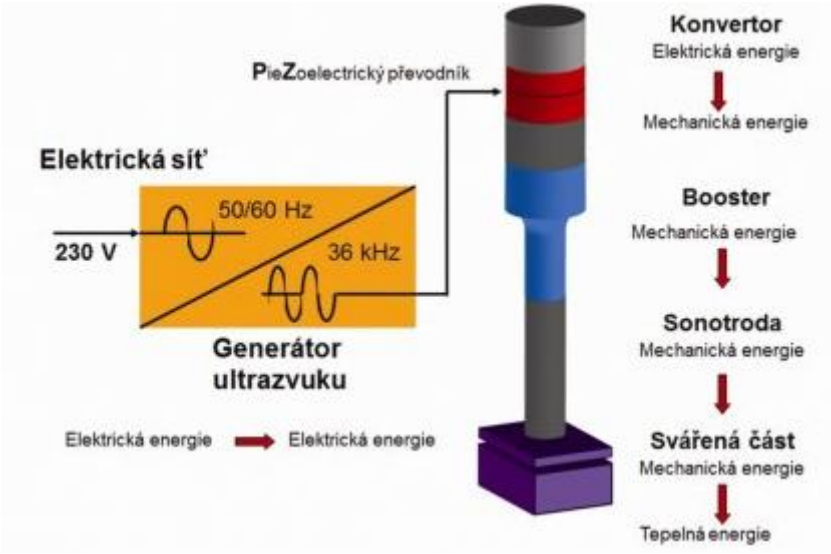
1.1.1.4 Ultrazvukové svařování

Jelikož je ultrazvukové svařování nosným tématem této práce, je realizace přesunuta do praktické části. V ní jsou uvedeny parametry potřebné k nastavení svářecího cyklu a případně jejich výpočet.

Popis

Tento princip se řadí mezi nejrychlejší metody svařování. Metoda je založena na mechanických kmitech o vysoké frekvenci (ultrazvuku). Těchto kmitů je docíleno pomocí generátoru a konvertoru. Generátor přeměňuje střídavý nízkofrekvenční proud na vysokofrekvenční energii. Kmitočet na výstupu generátoru se pohybuje mezi 4 - 100 kHz. Konvertor tuto elektrickou energii přetváří, pomocí piezoelektrického jevu, na vibrace [3]. Vibrace sice mají dostatečný kmitočet, ale malou amplitudu. Proto se využívá zesilovače, který

vysokofrekvenční kmity s již dostatečnou amplitudou předává sonotrodě. Úkolem sonotrody je přenos vibrací na svařovaný úsek. Úsek je ovlivněn teplem a tlakem vibrací. Z obecné teorie popsané výše víme, že teplo a tlak jsou činnými prvky svařování. Části sonotrody jsou vidět na obr. 10.



Obr. 10 Složení sonotrody [16]

Níže je uveden přehled vhodnosti materiálů ke sváření ultrazvukem [3].

	Technologie ultrazvukem			
	dobrá		*omezená*	*žádná*
Druh plastu	PVC – polyvinylchlorid	PMMA – polymethylmethakrylát	PE-polyethylen	měkčený PVC
	PS – polystyren	POM – polyoxymethylen	PP-polypropylen	PI – polymid
	houževnatý PS	PBTP – polybutylenereftalát	PA – polyamid	PTFE – polytetrafluorethylen
	ABS – akrylonitrilbutadienstyren	PPO – polyfenylenoxid	PC – polykarbonát	
	SAN – styrenakrylonitril	PS – polysulfid		

Tab. 1 Vhodnost materiálu pro svařovací metodu ultrazvukem

Výhody: velmi rychlý průběh svařování, možnost sváření ploch, majících mezi sebou materiál (nečistoty, mastnoty, atd.), teplotní vliv na svařované materiály není tak markantní jako u ostatních metod
Nevýhody: vysoká cena, velikost svarového spoje je ovlivněna konstrukcí sonotrody, potřeba přídavných pohonů (či pneumatiky) pro realizaci cyklu.

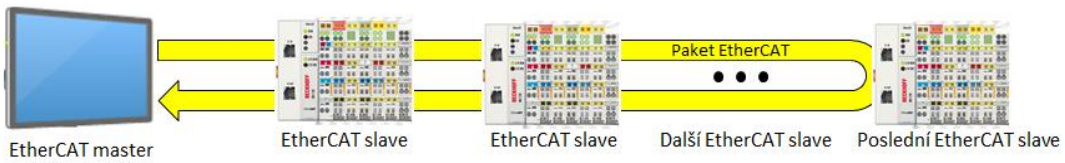
Realizace
Viz praktická část.

1.2 EtherCAT

Sběrnici EtherCAT vyvinula firma Beckhoff. V současné době je EtherCAT standardem a o rozvoj se stará ETG (EtherCAT Technology Group), což je skupina dodavatelů této sběrnice. Je postaven na technologii Ethernet, z něhož plyne i název (Ethernet for Control Automation Technology). V technologii se nahradil přístup pomocí MAC, který pracuje na linkové vrstvě Ethernetu, díky čemuž je možné komunikovat v reálném čase s vysokým výkonem [4].

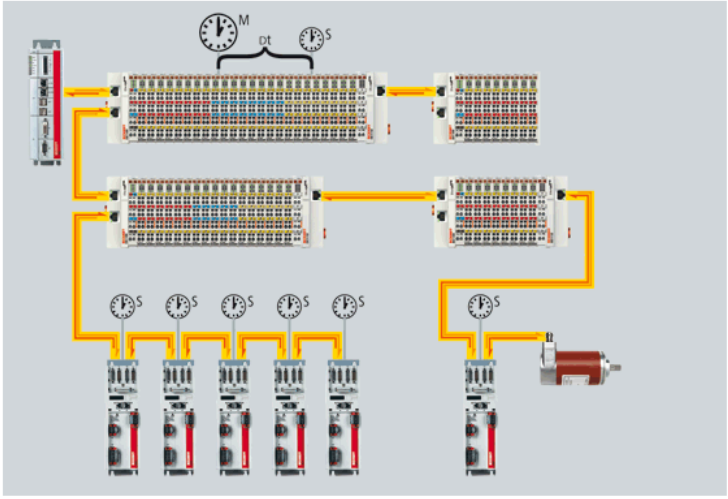
1.2.1 Princip

Základním rozdílem, oproti klasickému Ethernetu, je fakt, že se neodesílá každému komunikovanému prvku rámec zvlášť, ale jeden velký rámec, který prochází všemi zařízeními. Jedná se o komunikaci typu master–slave. Master pošle rámec prvnímu portu v řadě, ten si svá data z rámce vyjme a případně upraví. Výstupním portem rámec pošle dalšímu vstupnímu portu v řadě. Takto komunikace pokračuje až k poslednímu zařízení v řadě (má volný výstupní port). Poslední prvek pošle data zpět. Každé zařízení má tedy minimálně dva porty. Topologie je vždy logický kruh, i když fyzická topologie je nejčastěji neuzavřený kruh (topologie mohou být různé, viz kapitola 1.2.3) [4].



Obr. 11 Logická topologie

Aby byly zaručeny krátké komunikační časy a vysoké výkony, nepoužívají komunikační karty softwarový systém *přijmout, zpracovat a odeslat*, ale jsou tvořeny integrovanými obvody nebo programovatelnými hradlovými poli. Zpoždění mezi přijmutím a odesláním rámce je udáváno v jednotkách nanosekund. Díky technologii postavené na Ethernetu je možné připojit až 65 535 komunikujících prvků. Kabely mohou být mezi dvěma sousedícími zařízeními až 100 metrů dlouhé. Komunikace je deterministická i díky distribuovaným hodinám. Hodiny vlastní každé EtherCAT zařízení a díky tomu je zajištěna synchronizace. Výhodou je možnost měření zpoždění rámce mezi jednotlivými prvky a reagovat na něj [5].



Obr. 12 Synchronizace EtherCAT zařízení [5]

Komunikace je Full-duplexová, takže může být během jednoho komunikačního cyklu aktivních více paketů.

1.2.2 Výkonnost

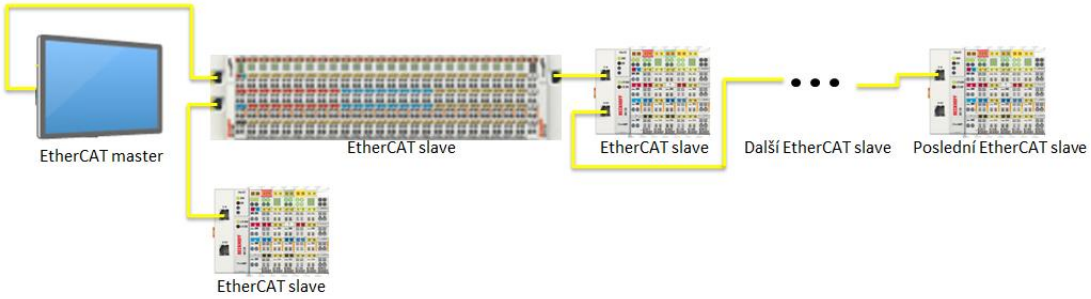
Pro představu o výkonu sítě uvádím pár příkladů o časech zpracování [6]:

- 256 digitálních vstupů/výstupů - 12 mikrosekund
- 1000 digitálních vstupů/výstupů - 30 mikrosekund
- 200 (16-ti bitových) analogových vstupů/výstupů – 50 mikrosekund

1.2.3 Topologie

Realizovat lze díky minimálně dvěma portům téměř libovolnou fyzickou topologii. Nejtypičtějšími jsou linie, strom nebo hvězda. Jak je popsáno výše, logická topologie je vždy logický kruh. Pokud bychom chtěli vytvořit hvězdicovou topologii, je nutno použít rozbočovač, či switch. Pozornému čtenáři jistě neuniklo, že jsem neuvedl topologii kruh. Je to z toho důvodu, že se kruhová topologie používá k redundanci. Komunikace se stále bude chovat jako neuzavřený kruh, jen je schopna posílat data oběma směry (redundance na úrovni kabelů). Výhoda redundance se projeví například při porušení kabelu (data lze oběma směry

vykomunikovat všechna), nebo při poruše zařízení (pokud lze řídit technologii i bez daného zařízení. V případě, že by redundance použita nebyla, pak by všechna ostatní zařízení za porušeným byla také pro komunikaci „mrtvá“) [4].



Obr. 13 Fyzická topologie

1.2.4 Protokoly

Tato kapitola slouží ke krátkému seznámení s protokoly, které jsou dostupné na sběrnici EtherCAT. Podrobněji popíši CoE, jelikož je použit ke komunikaci s naším generátorem.

Jak jsem popisoval výše, EtherCAT je založen na Ethernetu a má oproti němu určité rozdíly. Stejně, jako je rámec EtherCAT zapouzdřen v Ethernetu, jsou další protokoly schovány v EtherCATu. Níže je rychlý přehled protokolů [4].

- EoE (Ethernet over EtherCAT)
- FoE (File Access over EtherCAT)
- SoE (Servo Drive Profile according to IEC 61491-7-204 over EtherCAT)
- CoE (CAN Application Protocol over EtherCAT), nebo také (CANopen over EtherCAT)

1.2.4.1 EoE

V EtherCATu je zabalen klasický Ethernet rámec. Jedná se tedy o komunikační rozhraní Ethernet, přičemž zůstává zachován real-time proces.

1.2.4.2 FoE

Protokol, díky kterému lze přistupovat k souborům. Využívá se například při nahrávání novějšího firmware.

1.2.4.3 SoE

Podpora real-time komunikačního rozhraní SERCOS, využívaného hlavně pro řízení pohonů.

1.2.4.4 CoE

V této kapitole je uveden seznam a popis objektů, které využívá CANopen protokol.

CAN využívá komunikačních objektů. Objekty jsou různého typu a mají svůj přesný význam. Každý objekt má jeden nebo více identifikátorů, které jsou určeny pro definování priority [7].

- PDO (Process Data Objects)
- SDO (Service Data Objects)
- NMO (Network Management Objects)
- SYNC Object
- Emergency Object
- Time Stamp Object

PDO - Procesní data jsou přenášena v jediné CAN zprávě v reálném čase. Mají přesně danou velikost a unikátní identifikátor. Odeslání zprávy může být zapříčiněno vnitřním časovačem, vnitřní událostí, synchronizační zprávou, nebo požadavkem zařízení v síti.

SDO - Servisní data lze upravovat jednotlivě. Komunikace neprobíhá v reálném čase. Každý objekt má unikátní identifikátor a může mít rozdílnou délku. Pokud objekt přesáhne určitou délku, je rozdělen na segmenty se stejným identifikátorem. Komunikace je potvrzovaná typu klient-server.

NMO (Network Management Objects) - Obsahuje nástroje pro správu sítě.

Boot-up Object - Zpráva má stejný identifikátor jako Node/Life nebo Heartbeat. Shodná je i délka 1 bajt, neobsahuje však žádná data. Je poslána jako informace, že došlo k inicializaci (například po zapnutí).

Node/Life-guarding Object - Periodická zpráva, která má za úkol zjistit, zda jsou ostatní účastníci aktivní. Odpovědí je stav zařízení a pro rozpoznání aktuálnosti je přidán bit, který pokaždé změní svou hodnotu.

HeartBeat Object - Zpráva, která sděluje, že zařízení není v chybě a je aktivní. Zpracovává se periodicky.

NMT (Network Management) Object - Zpráva, jejíž identifikátor je nastaven na 0, což znamená nejvyšší prioritu. Má délku 2 bajty. První bajt je naplněn příkazem a druhý adresou cílového zařízení. Pokud je adresa nastavena na 0, pak je příkaz určen všem zařízením. Po zapnutí je každé zařízení ve stavu INITIALIZATION, ze kterého přechází do stavu PRE-OPERATIONAL. Master pomocí NMT nastaví zařízení do stavu OPERATIONAL.

Možné stavy CANopen

- INITIALIZATION – Stav po zapnutí
- STOPPED - Komunikace omezena jen na posílání NMT
- PRE-OPERATIONAL - Automatický přechod ze stavu INITIALIZATION, možno posílat SDO
- OPERATIONAL - Povolena komunikace s PDO

SYNC Object - Zajišťuje pomocí taktování synchronizaci sítě. Periodu odesílání je možno nastavit. Obsahuje identifikátor 128.

Emergency Object - Zpráva signalizující kritickou závadu. Má pevně danou délku 8 bajtů. Je poslána jen jednou za výskyt dané závady a nevyžaduje potvrzení o přijetí. Chyby, které jsou povoleny odesílat pomocí Emergency Object jsou definovány protokolem CANopen.

Time Stamp Object - Zpráva nesoucí časovou značku. Značka obsahuje aktuální datum a čas. Na tuto zprávu se neodpovídá. Má identifikátor 256. Délka zprávy je 6 bajtů.

1.2.5 Bezpečný přenos dat

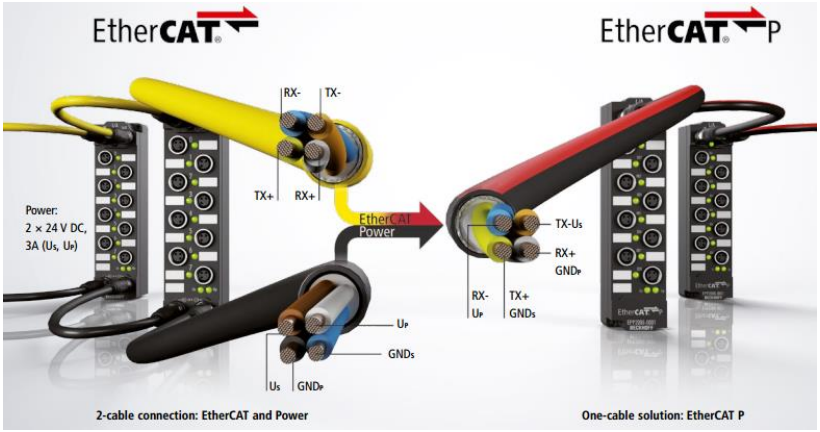
FSoE (Safety over EtherCAT) - Komunikační systém pro bezpečnostní funkce do úrovně SIL 3. Výhodou je, že využívá jednu sběrnici pro bezpečnostní i pro běžnou komunikaci. Je součástí normy IEC 61784-3 [4].

1.2.6 Protokol postavený na EtherCATu

EAP (EtherCAT automation Protocol) - Nejedná se o protokol zabalený v EtherCATu, ale o protokol, který je postaven na jeho základech. Využívá IP nezávisle na fyzické vrstvě (možno i bezdrátově). Vznikl z důvodu urychlení výměny procesních řídících dat mimo reálný čas [4].

1.2.7 EtherCAT P

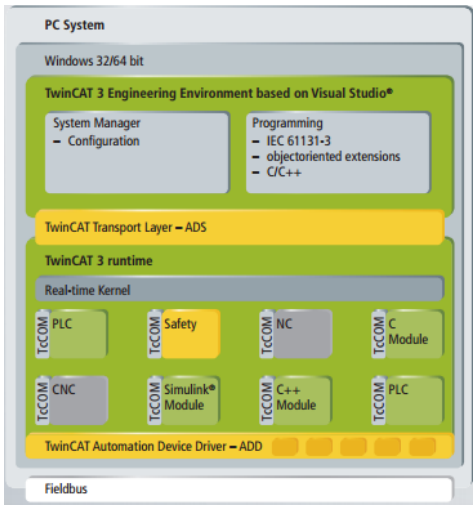
Jeden kabel kombinující komunikační a napájecí kabel. Jde o náhradu stávajícího řešení, kdy se komunikační kabely a napájecí kabely musely natahovat v rozvaděči zvlášť [8].



Obr. 14 Porovnání klasického EtherCAT vs EtherCAT P [8]

1.3 TwinCAT

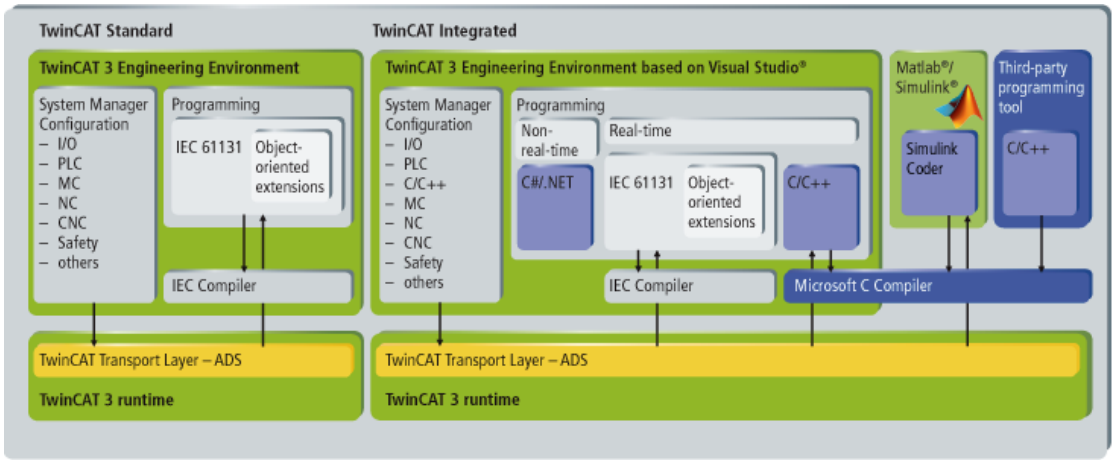
Je zkratkou (The Windows Control Automation Technology) pro sadu nástrojů určených k realizaci automatizačních systémů na platformě PC. Architektura TwinCAT 3 (eXtended Automation Architecture – XAA, na obrázku 15) je složena ze tří sekcí. Vývojové prostředí (eXtended Automation Engineering - XAE), real-time runtime (eXtended Automation Runtime - XAR) a transportní vrstvy (Automation Device Specification - ADS) [9]. Tato celá kapitola, myšleno včetně jejích podkapitol, bude rozepsána více. Je to z toho důvodu, aby čtenář, který se systémy Beckhoff nikdy nepracoval, pochopil na jakém základu jsou postaveny



Obr. 15 Struktura XAA [9]

1.3.1 XAE

XAE je rozdělen na dva základní moduly. Správcovský a programovací. Ve správcovském modulu se nastavují parametry modulů I/O, pohybových úloh, CNC úloh, bezpečnostních úloh, průmyslových sběrnic, vlastnosti C/C++ a PLC úloh. V programovacím modulu je rozšíření normy IEC 61131-3 o objektové programování. Na výběr jsou z normy jak grafické, tak i textové jazyky. V grafických jazycích si můžeme vybrat mezi FBD (Function Block Diagram), LD (Ladder Logic Diagram), SFC (Sequential Function Chart), CFC (Continuous Function Chart) a SC (UML Statechart). Z textových máme na výběr mezi IL (Instruction List) a ST (Structured Text) [10]. Strukturu XAE vidíme na obr. 16.



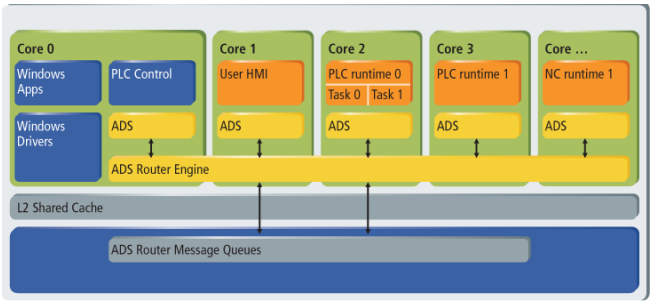
Obr. 16 Struktura XAE [10]

Vlastnosti XAE [9]

- Vývojové prostředí je integrováno do Visual Studia
- V jednom prostředí probíhá konfigurace hw (System Manager), programování (PLC Control), monitorování (Scope View) i vizualizace (visualization)
- Podpora normy IEC 61131-3 a rozšíření o objektové programování PLC
- Rozšíření o programovací jazyk C/C++ pro úlohy reálného času
- Podpora pohybových úloh (Motion Control)
- Lze navázat na vývojové prostředí Matlab/Simulink
- Migrace z prostředí TwinCAT 2
- Programování bezpečnostních funkcí TwinSafe

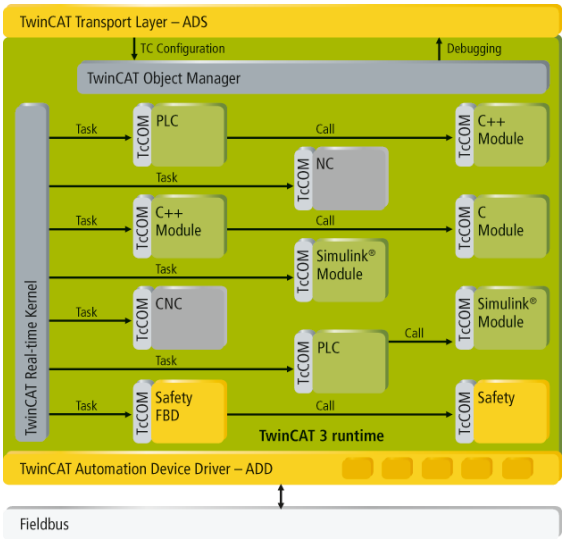
1.3.2 XAR

Výsledný XAE projekt je zkompilován a nahrán do XAR, který běží vedle operačního systému Windows. Podporovány jsou jednojádrové i vícejádrové procesory. U vícejádrových procesorů lze u každého zvlášť nastavit výkon určený pro Windows a pro runtime [10]. Možnosti rozdělení zdrojů jsou uvedeny na obr. 17



Obr. 17 Realizace využití vícejádrových procesorů [10]

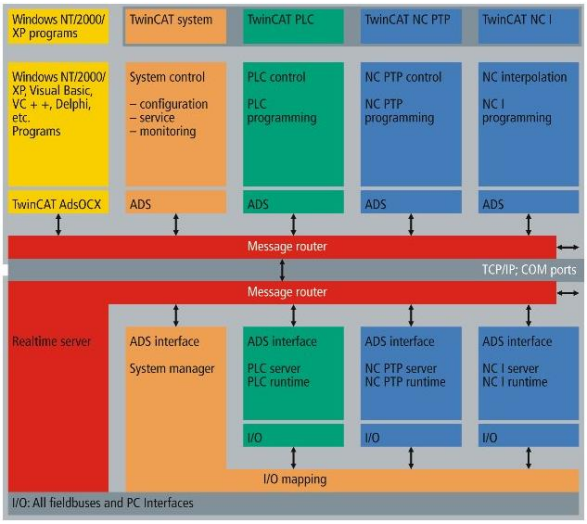
Použít lze 32bitové i 64bitové systémy. Runtime svou činnost může vykonávat na IPC s operačním systémem Windows: XP se Service Pack 3 (x86), Windows 7 Professional (x86, x64), Windows 7 Ultimate (x86, x64) nebo Windows 10 (x86, x64). Na Embedded zařízeních lze využít runtime na Windows: Embedded Standard 2009 nebo Embedded Standard 7 [10].



Obr. 18 Struktura XAR [10]

1.3.3 ADS

Automation Device Specification protokol slouží k výměně dat mezi zařízeními, které v sobě mají implementované ADS rozhraní. Rozhraní posílá zprávy pomocí message routeru. Identifikace zařízení je dána ADS adresou, která je složena z NetID a Portu [11].



Obr. 19 Komunikace pomocí message routeru [11]

1.3.3.1 Port

Dle portu je určena úloha, kterou má daný modul v síti. Například pokud budu nahrávat PLC projekt do dvou různých zařízení, bude NetID jiné, ale port stejný. Tabulka přiřazení úloh k portům je níže [12].

ADS-PortNr	ADS device description
100	Logger
110	Eventlogger
350	IO
351	additional Task 1
352	additional Task 2
500	NC
851	PLC RuntimeSystem 1
852	PLC RuntimeSystem 2
853	PLC RuntimeSystem 3
854	PLC RuntimeSystem 4
...	
900	Camshaft controller
10000	System Service
14000	Scope

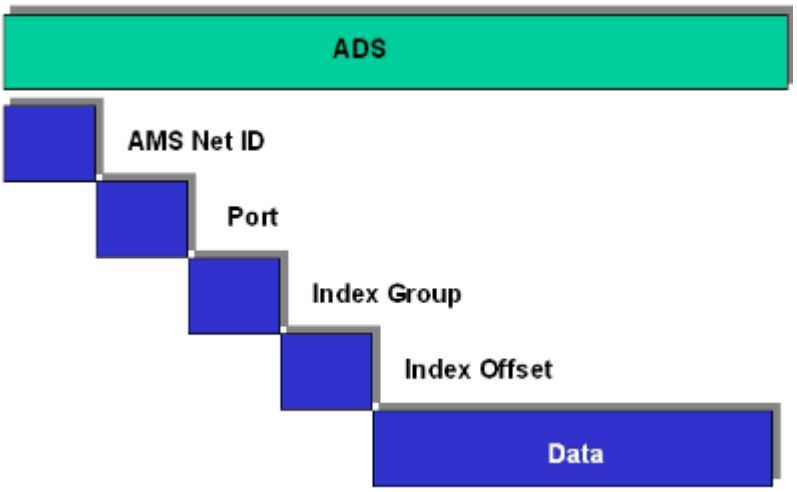
Obr. 20 Přehled úloh pro vyhraněné porty [12]

1.3.3.2 NetID

Je typu AMSNETID, což je pole 6-ti bytů. Nepsaným pravidlem je, že první 4 byty je dobré vyplnit IP adresou zařízení. Příkladem je zařízení s IP adresou "10.10.53.2". Jeho NetID by byla tedy například "10.10.53.2.1.1". ADS komunikaci lze provozovat mezi jednotlivými moduly TwinCAT (například NC a PLC) i mezi TwinCAT moduly s okolím (například PLC a C# aplikací) [12].

1.3.3.3 Protokol

Protokol běží nad TCP/IP nebo UDP/IP protokolem. Díky tomu lze použít libovolnou spojovací cestu (nezávisle na užitých sběrnicích) ke komunikaci s připojenými zařízeními. Jak je poznamenáno výše, identifikace se skládá z NetID a portu. Kombinací *Index Group* a *Index Offset* vzniká adresa proměnné [13].



Obr. 21 Složení ADS zprávy [13]

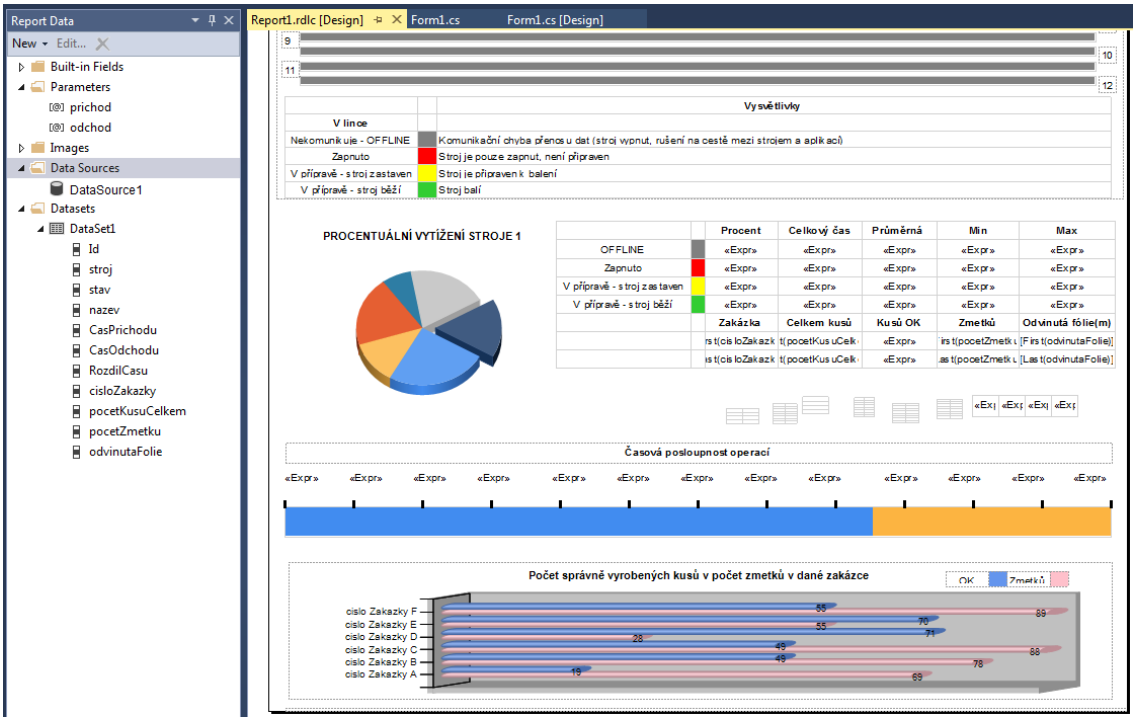
1.3.3.4 Implementace

ADS interface nabízí následující možnosti implementace:

- PLC library (v TwinCAT PLC)
- ADS-DLL (C/C++, Delphi, atd.)
- ADS.NET komponenta(VB.NET, C#, Delphi.NET, atd.)
- ADS-OCX [ActiveX prvek] (Visual Basic, C++, Delphi, atd.)
- ADS-Script-DLL (VBScript, Jscript, atd.)
- ADS-WebService (pomocí http ve Visual C#, Delphi.NET, atd.)
- ADS-Java-DLL
- OPC

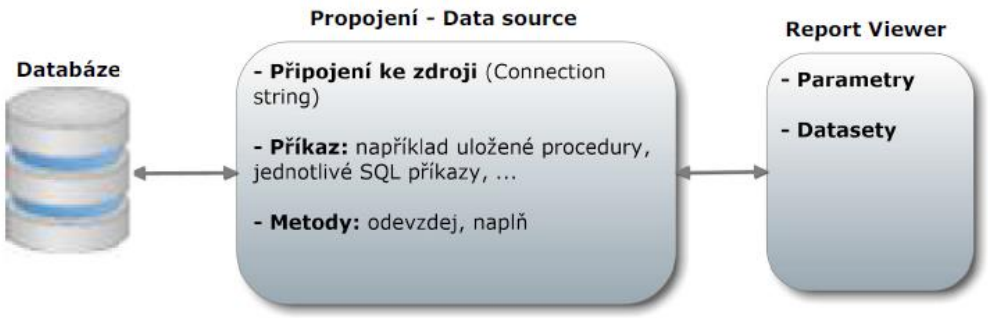
1.4 SSRS

Je zkratka pro Microsoft SQL Server Reporting Services. SSRS v sobě implementuje sadu nástrojů a služeb pro vytváření a správu různých interpretací dat. Nástroje jsou dostupné v prostředí Microsoft Visual Studio a jsou plně kompatibilní s SQL Serverem. Slouží k vývoji reportů z XML dat, či dat z relačních zdrojů. Reporty lze zobrazit jako desktopovou aplikaci, či webovou stránku. SSRS se skládá z grafické interpretace, propojení a zdrojů dat. Grafická úprava dat je realizována v ReportVieweru (tvorba tabulek, grafů, atd.).



Obr. 22 Vývojové prostředí ReportViewer

Zdrojem dat může být například databáze. Propojení obsahuje připojení ke zdroji (Connection string), příkaz (například stored procedures) a metody (Fill a Get). Propojení reportu s daty může vypadat například takto:



Obr. 23 Propojení reportu s databází

2 PRAKTICKÁ ČÁST

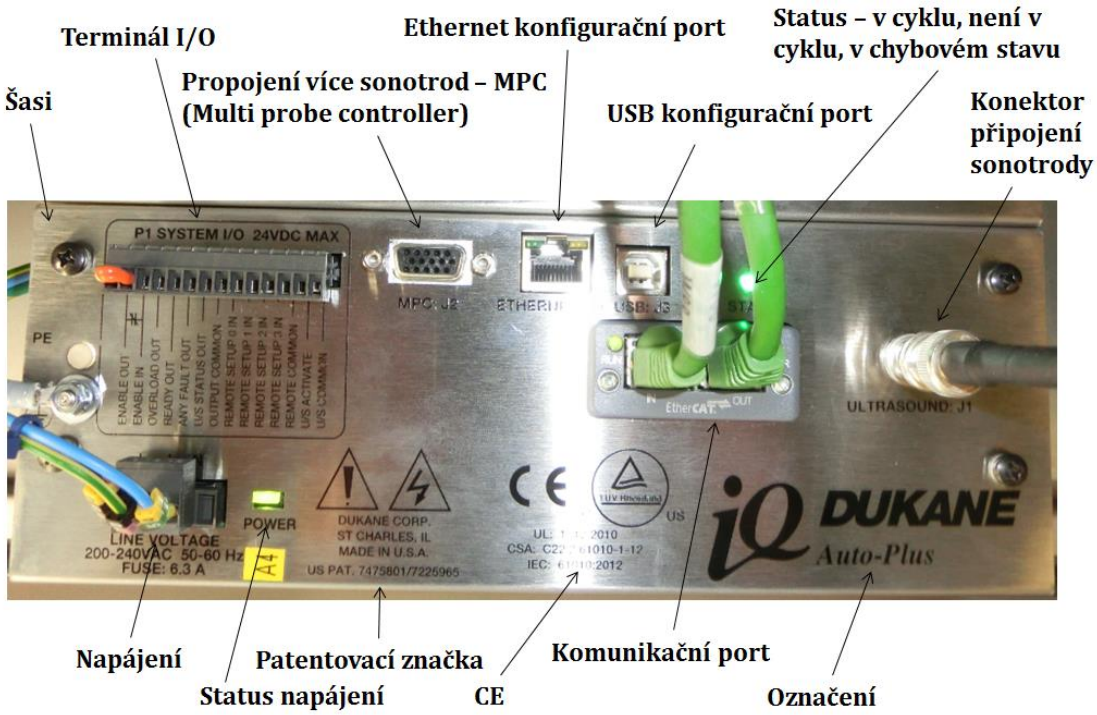
Cílem praktické části je analýza stávajícího stavu strojů a navržení řešení, které by odstranilo současné limity balení problémových druhů zboží. Po návrhu řešení a průzkumu trhu vytypovat dodavatele svařovací komponenty s ohledem na současný stav řídícího systému. V současné době se systém programuje v softwaru TwinCAT 3, ovšem podmínkou je možnost vložení do starších strojů. Je tedy nutno využít knihoven, které jsou podporovány v TwinCat 2. Do systému poté zařadit objekt, naprogramovat potřebnou řídící funkci a komunikační interface pro potřebná data. Jedna z podmínek je, že celek musí být umožněn jednoduše zařadit do ŘS jako nová volba stávajících řešení svařování. Dalším bodem je grafický návrh zobrazení dat ve vizualizaci. Novou část také vytvořit a poté zakomponovat do současné vizualizace. Poslední částí práce je doplnění o tzv. chytrá data z výroby.

2.1 Analýza

Největším problémem při balení zboží je dodržení opakovatelné kvality svaru u problémových druhů zboží. Problémové zboží je takové, které zasahuje nežádoucím způsobem do místa svaru. Například takové bonbony problémovým zbožím nejsou, protože svou vlastní hmotností spadnou na dno sáčku a nezanechávají na fólii stopy materiálu zabraňující vytvoření svaru. Naopak olej, či prášek se zachytává po celé fólii a je větší pravděpodobnost, že vznikne sáček, který neprojde kontrolou. V současném stavu se používají metody: nepřímé horkým tělesem, impulzní a horkovzduchem. V části, kde popisuji druhy sváření, které se používají u termoplastů, zmiňuji i sváření ultrazvukem. Tuto metodu jsem vybral, protože teorie říká, že má značný vliv na působení zboží ve svaru. Tuto skutečnost potvrdily testy při balení. Po průzkumu trhu byl vybrán generátor od firmy Dukane IAS. Firem, které se specializují na sváření ultrazvukem je více. Jednou z nich je například společnost Herrmann Ultrasonic. Ve firmě, pro kterou píši tuto diplomovou práci, je využíván ŘS od firmy Beckhoff. Proto jedním z plusů pro firmu Dukane IAS je ten, že vybraný generátor s označením *iQ Auto-Plus* umožňuje implementaci EtherCAT karty do generátoru. Tím pádem je umožněno kompletní řízení po komunikaci a nejsou potřeba další vstupní / výstupní karty. Protokol, který je využíván generátorem je CoE (CANopen over EtherCAT). Sběrnice EtherCat podporuje mnoho dalších protokolů, viz kapitola EtherCAT.

2.1.1 Popis vybraného generátoru

Jak vybraný generátor vypadá, je zobrazeno na obrázku níže. Obsahuje terminál vstupů/výstupů pro základní ovládání a diagnostiku. My tento terminál nevyužijeme, jelikož máme celé řízení postavené na EtherCATu. Dále je obsažen MPC, díky němuž lze připojit více sonotrod. V generátoru jsou dvě možnosti diagnostiky. Pomocí ethernetového kabelu a portu USB. Dioda STATUS hlásí, zda se generátor nachází ve stavu svařování, či nikoliv. Poslední, pro nás zajímavou součástí je komunikační port.



Obr. 24 Vybraný generátor - iQ Auto Plus (Dukane)

Nyní, když máme vybrán generátor, můžeme přistoupit k realizaci projektu.

2.2 Projekt

2.2.1 Seznámení s TwinCAT 3 a plán práce

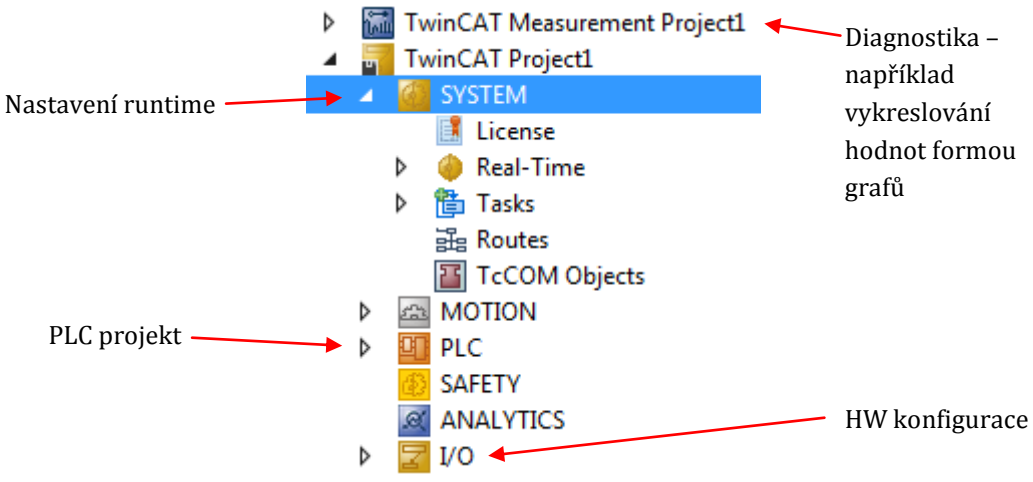
Nejprve dovolte shrnout informace z kapitoly 1.3, která se zabývá filozofií ŘS Beckhoff. S těmito informacemi budeme totiž dále pracovat. Beckhoff nenabízí ve svém produktovém portfoliu klasické PLC jak je známe, tedy fyzický hardware do kterého nahrajeme PLC projekt. Veškeré ŘS jsou postaveny na průmyslových PC, na kterých běží vedle operačního systému tzv. real-time runtime, který po sběrnici EtherCAT komunikuje s ostatním HW (v našem případě I/O a generátor). Do runtime se nahraje projekt, který může kromě HW konfigurace a PLC projektu obsahovat například NC úlohy, safety, atd..

2.2.1.1 Struktura TwinCAT 3

Na obr. 25 je vidět, jak vypadá prostředí TwinCAT 3 a že obsahuje všechny prvky z kapitoly 1.3.1. Abychom mohli propojit HW část s PLC částí, je potřeba, aby proměnné, které chceme propojovat (např. binární vstup ze vstupní karty, nebo proměnné z generátoru) byly označeny pomocí %I*, případně %Q*.

Například:
viditelnáVstupníProměnná AT %I*: INT;
viditelnáVýstupníProměnná AT %Q*: INT;

oproti
neviditelnáProměnná: INT;

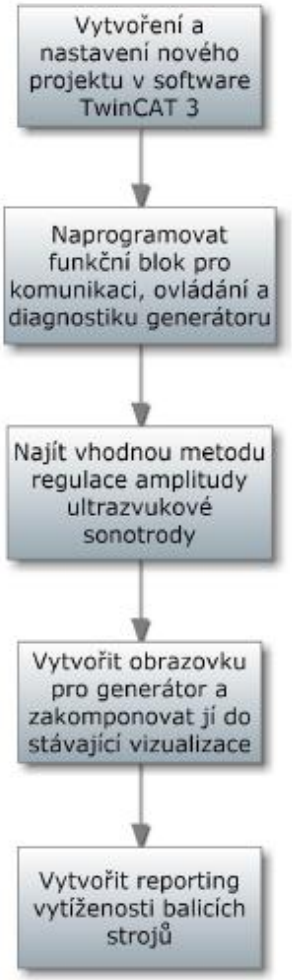


Obr. 25 Jedno prostředí pro celý projekt

2.2.1.2 Plán práce

Níže je uveden chronologicky sestavený plán práce. V kapitole 2.2.2 se budu zabývat krok po kroku tvorbou nového projektu od samotného založení až po nastavení. Kapitola 2.2.3 slouží k popisu naprogramovaného bloku pro komunikaci a řízení generátoru. V 2.2.4 je znázorněna metoda řízení svářecího cyklu pomocí změny amplitudy. Dále je ukázka výpočtu dopravního zpoždění změny řízení amplitudy. Kapitola 2.2.5 slouží k popisu vytvořené vizualizace.

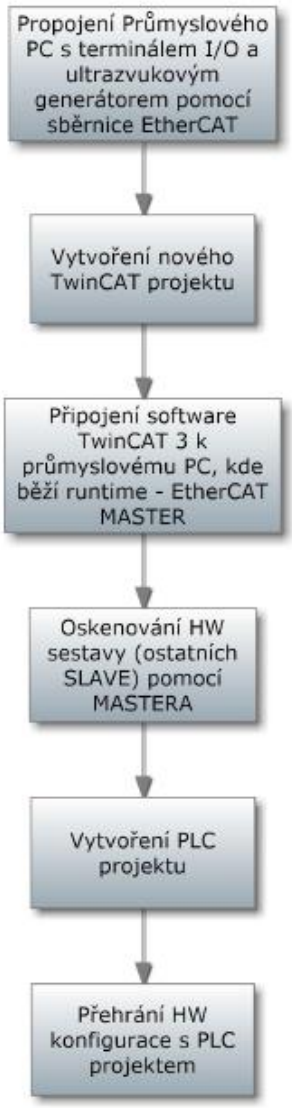
Reporting jsem umístil do samostatné kapitoly 2.3. Důvod je ten, že nesouvisí přímo s ultrazvukovým řízením, ale se stroji jako takovými. V 2.3 bude opět krok po kroku znázorněn vývoj report systému včetně ukázek zdrojových kódů.



Obr. 26 Plán práce

2.2.2 Vytvoření projektu

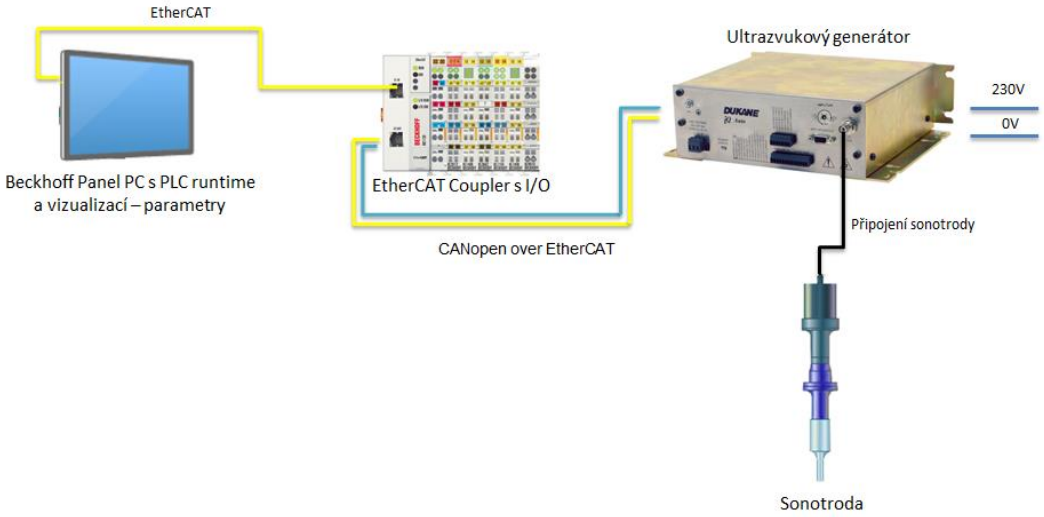
Na obr. 27 je diagram postupu pro vytvoření projektu. Nejprve propojíme účastníky na sběrnici EtherCAT. Dále založíme nový projekt, který budeme po oskenování HW konfigurace a naprogramování PLC nahrávat do průmyslového PC.



Obr. 27 Postup při vytváření projektu

2.2.2.1 Propojení členů sběrnicí EtherCAT

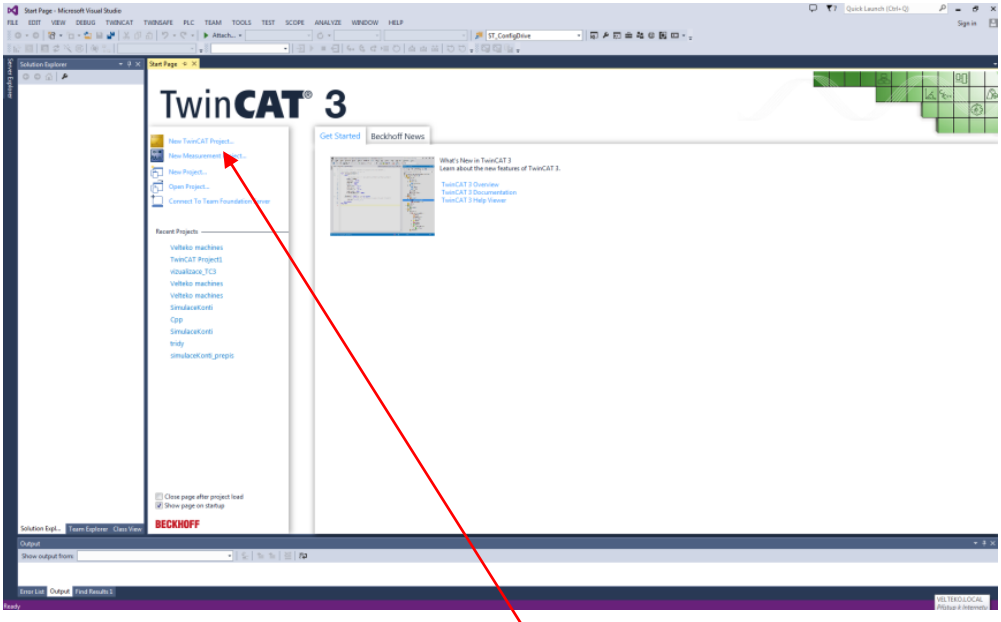
Tato kapitola obsahuje schéma zapojení komponent potřebných pro řízení generátoru. K PC je jako první v řadě připojen terminál se vstupy/výstupy a následuje generátor, k němuž je připojená sonotroda popsaná v 1.1.1.4.



Obr. 28 Schéma zapojení ultrazvuku a sonotrody s ŘS

2.2.2.2 Nový projekt

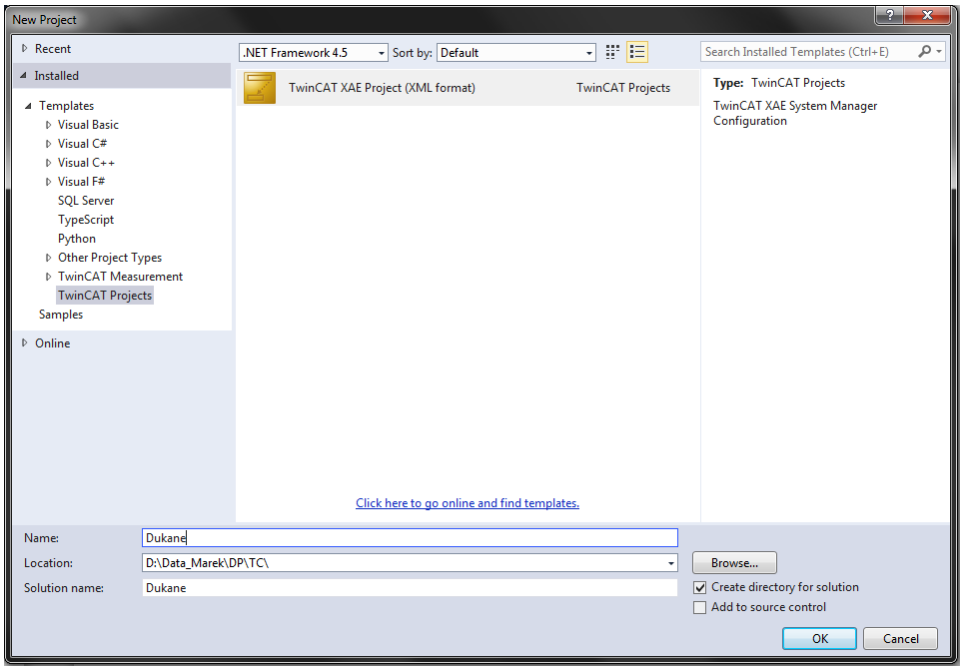
Začneme spuštěním software s označením *TwinCAT XAE*. Po spuštění vidíme prostředí Visual Studia s integrovaným TwinCAT 3. Náhled je na obr. 29.



Obr. 29 Založení nového projektu

Visual Studiu řekneme, že se jedná o *XAE* (New TwinCAT Project).

Po zvolení *New TwinCAT Project* pojmenujeme projekt a zvolíme umístění, viz obr. 30.

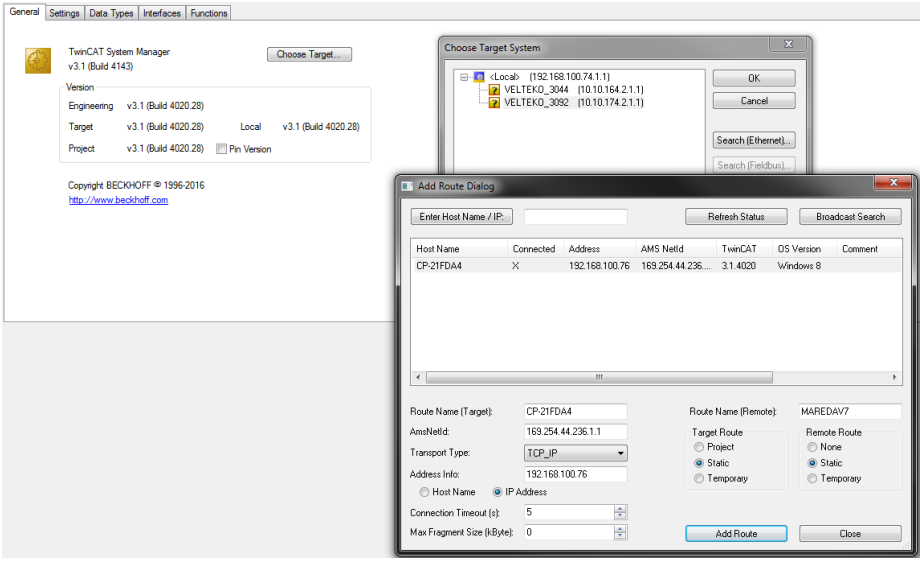


Obr. 30 Výběr XAE projektu

Nyní máme projekt založen, můžeme tedy pokračovat dále.

2.2.2.3 Propojení projektu s cílovým průmyslovým PC

Čistý projekt neobsahuje PLC projekt ani HW konfiguraci. Nejprve je potřeba se spojit s průmyslovým PC, který nám po EtherCATu najde ostatní HW. Cíl (PC) najdeme pomocí *SYSTEM -> Choose Target -> Search Ethernet -> Broadcast Search*. Program sám najde všechny dostupné PC, na kterých běží runtime. Po zvolení *Add Route* se náš projekt propojí se zvoleným runtime. Pokud vše proběhne v pořádku, uvidíme, stejně jako na obr. 31, ve sloupci *Connected* křížek.



Obr. 31 Připojení k cílovému runtime zařízení

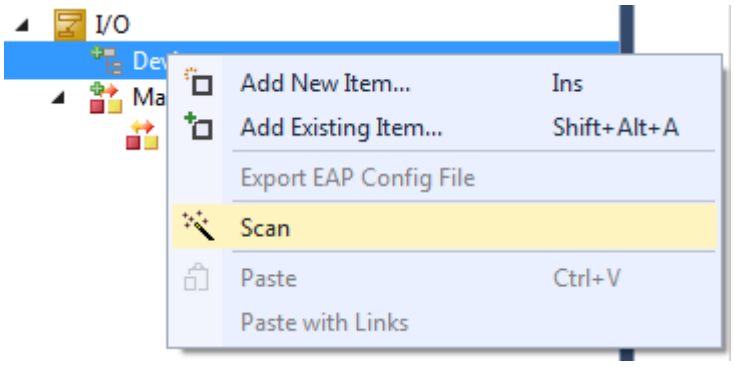
2.2.2.4 Sken HW sestavy

V současné době máme propojenou síť EtherCAT, založen projekt a jsme připojeni k ŘS. Následujícím krokem je nalezení ostatních EtherCAT účastníků.

Pokud chceme oskenovat sestavu, je potřeba být v konfiguračním módu. Konfigurační mód je znázorněn modrou barvou. Run mód (runtime je aktivní) se značí zeleně. Obrázek schodů se šipkou dává softwaru pokyn pro nahrání HW konfigurace do runtime. Všechny režimy jsou uvedeny na obr. 32.

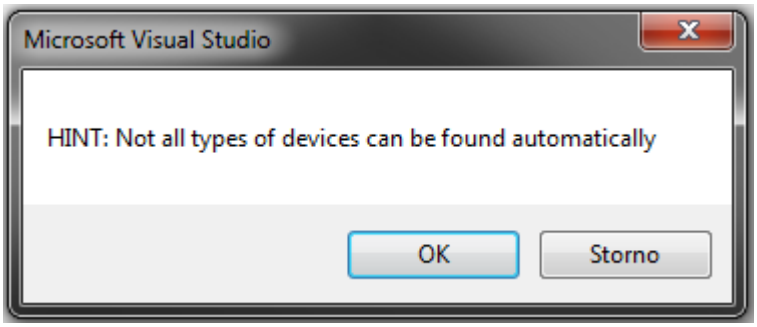
Obr. 32 Nahrání konfigurace a přepínání mezi módy

Po přepnutí do konfiguračního módu můžeme načíst připojenou sestavu. Samotné skenování sestavy je znázorněno na obr. 33. Umístění je v *I/O -> Devices*.



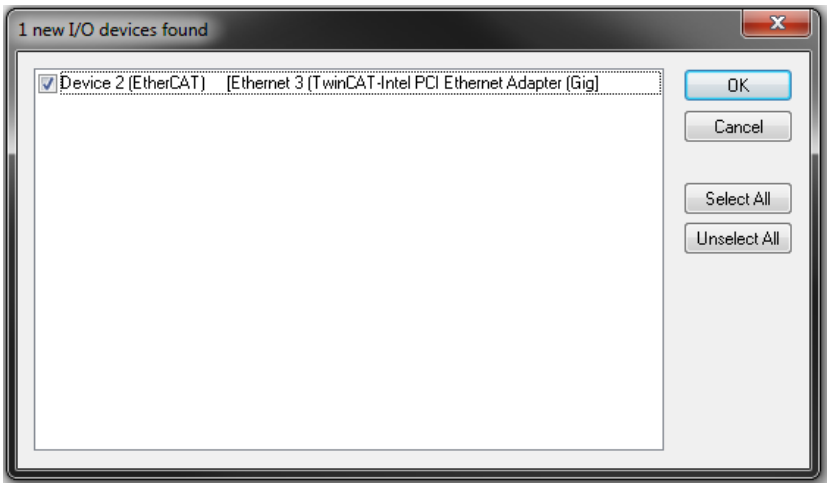
Obr. 33 Vyhledání připojených zařízení na EtherCATu

Po aktivaci skenování na nás vyskočí upozornění, že není zaručeno nalezení veškerých zařízení, viz obrázek níže. Proto je dobré si po oskenování zkontrolovat shodu HW konfigurace v softwaru s realitou.



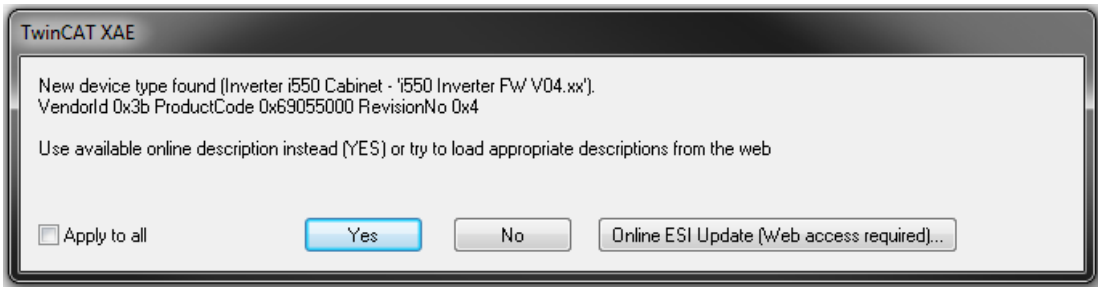
Obr. 34 Hláška pro kontrolu nalezení všech zařízení

Po potvrzení hlášky nám software nalezne naše PC (respektive jeho EtherCAT adaptér). Při připojení více ŘS do sítě se může stát, že dostaneme na výběr mezi více adaptéry. My na výběr nemáme, jak můžeme vidět na obr. 35, odsouhlasíme tedy použití nalezeného adaptéru. Tento adaptér bude na síti jako MASTER a najde nám zbylé SLAVE.



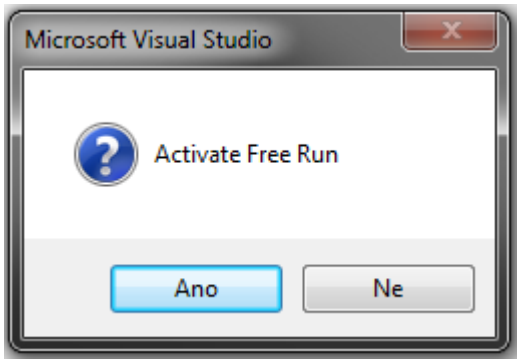
Obr. 35 Volba EtherCAT adaptéru pro vyhledávání

Zde je potřeba podoktknout, že HW konfigurace v projektu je popsána xml soubory. Každé EtherCAT zařízení má svůj xml soubor, který se dá stáhnout na stránkách výrobce daného zařízení. Pokud MASTER zjistí, že popis v xml souboru v projektu je zastaralý, nebo soubor v projektu vůbec není obsažen, máme možnost nechat si stáhnout nový popis automaticky. Vyzvání ke stažení je simulováno na obr. 36.



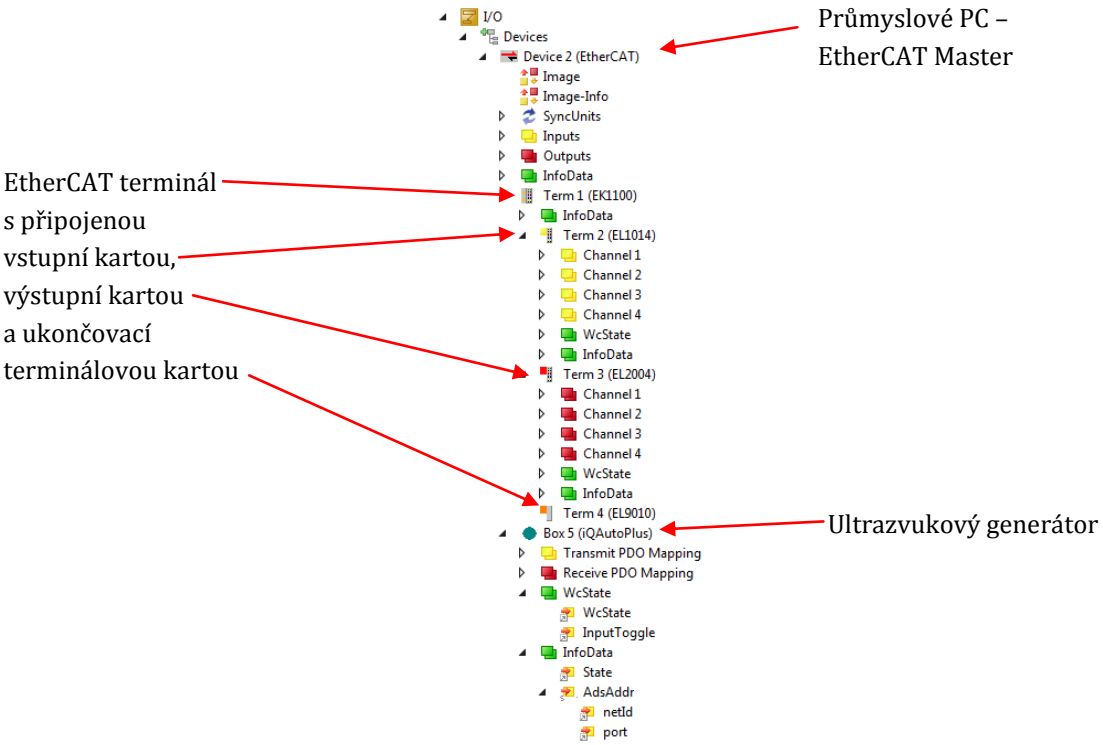
Obr. 36 Stažení nového popisu zařízení

Po nalezení všech účastníků na síti je HW konfigurace dokončena. Nyní můžeme zůstat v konfiguračním módu, nebo ho opustit. My zatím v konfiguračním módu zůstaneme. Klikneme tedy na *Ne*.



Obr. 37 Aktivace Free Run módu

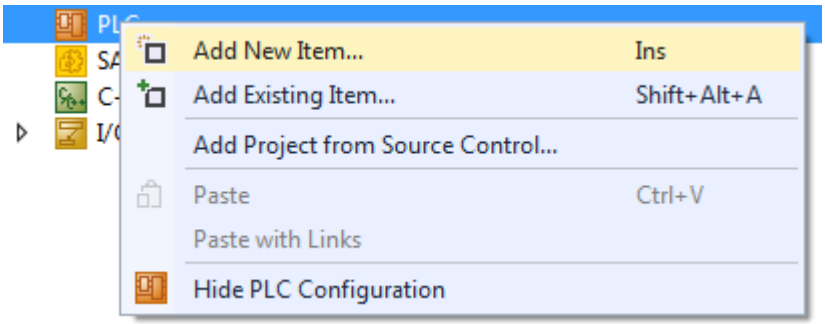
HW konfiguraci se vstupy, výstupy a ultrazvukovým generátorem (iQ Auto-Plus) vidíme na obr. 38. Konfigurace nám sedí s obr. 28, takže máme jistotu, že sken proběhl v pořádku.



Obr. 38 Přehled nalezených zařízení

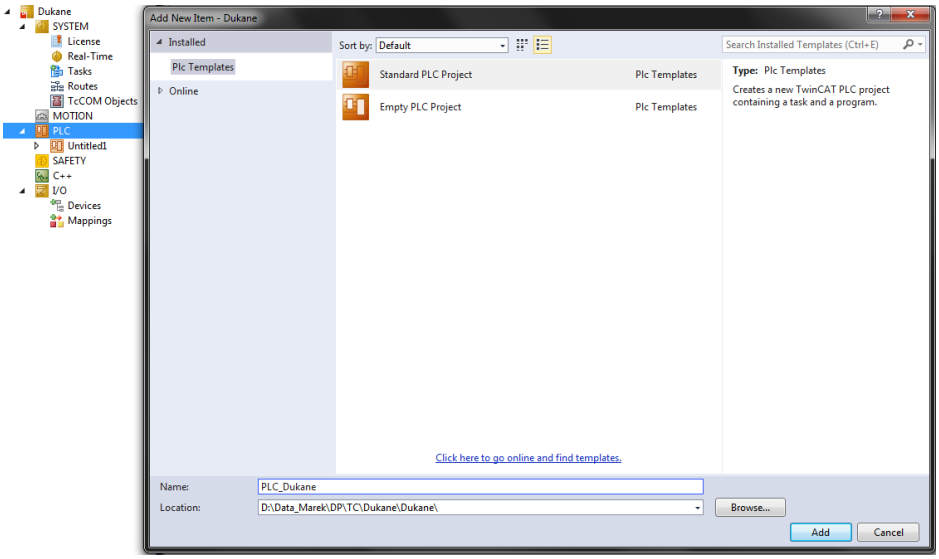
2.2.2.5 PLC projekt

Abychom mohli generátor řídit, musíme si založit PLC projekt, ve kterém později namapujeme (propojíme s PLC algoritmem) proměnné PDO (Process Data Objects), vstupy, výstupy, atd.. Přidání PLC projektu je popsáno a zakresleno níže.



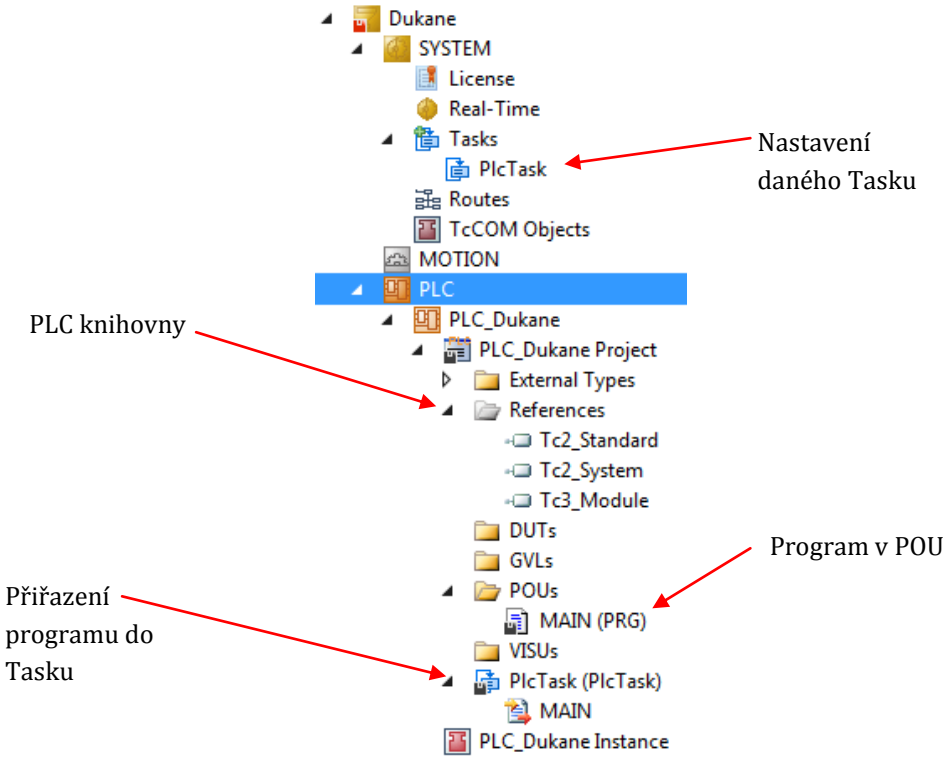
Obr. 39 Vložení PLC do projektu

Můžeme si založit úplně prázdné PLC, nebo se standardní strukturou. Vybereme se standardní strukturou, viz obr. 40, abychom měli alespoň základní kostru.



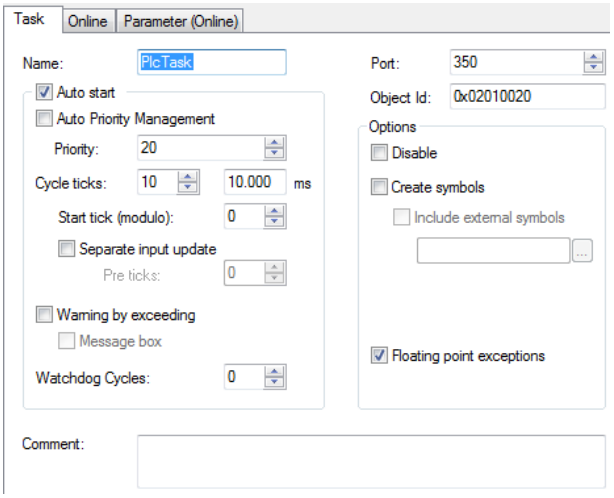
Obr. 40 Pojmenování PLC

Automaticky vygenerovaná struktura obsahuje *References*, což jsou použité knihovny, jejichž funkce a funkční bloky využíváme. Nyní si pro jistotu vysvětlíme termín *POUs*, který je sice definován obecně užívanou normou IEC 61 131-3, ale zde v textu ještě nebyl použit. *POUs* (ENG: Program Organization Units, CZ: programové organizační jednotky) je jednotný název pro programy, funkční bloky a funkce. V *POUs* se zmíněné programy, funkce a funkční bloky tvářejí z důvodu jasné struktury projektu. Základním programem je *MAIN*, který neustále dokola vykonává vnořený kód. Poslední důležitou položkou je *PlcTask (úloha)*, do kterého vkládáme programy a určujeme, jak se program bude vykonávat. Úloh může být více než jedna a do každé z nich lze vložit několik programů. Vkládání programů do úloh provádíme v *PLC*, ovšem vlastnosti dané úlohy nastavujeme v *SYSTEM -> Task* a vybraná úloha. Jasnější pohled na věc dostaneme, pokud se blíže podíváme na obr. 41 a obr. 42. Na obr. 41 je vidět struktura PLC a obr. 42 slouží k nahlédnutí do možností nastavení úloh.



Obr. 41 Struktura PLC

V nastavení úlohy se vybírá, zda se má úloha spustit automaticky po aktivaci runtime, nebo jestli ji budeme volat programově dle libosti. Další vlastností je priorita. Priorita se uplatňuje v případě, pokud dvě úlohy soupeří o zdroje procesoru. Poslední věcí, kterou nastavíme, je *Cycle Ticks*. Čas určuje frekvenci vykonávání dané úlohy. Hodnota na obrázku nám říká, že se úloha bude volat každých 10ms. Vše je přehledně vykresleno na obr. 42.



Obr. 42 Nastavení Task pro PLC

Nejdůležitější body máme nastaveny. Nyní lze aktivovat konfiguraci. Před vlastní aktivací sestavíme projekt pro nahrání do runtime. Na hlavním panelu zvolíme *BUILD -> Build Solution*. Pokud při sestavení nedošlo k žádné chybě, aktivujeme konfiguraci, viz popis obrázku 32. Po úspěšné aktivaci se *XAE* zeptá, zda chceme runtime přepnout do módu run. Po potvrzení přepnutí do run módu se runtime spustí. Je sice aktivní, ale v *PLC* se nevykonává žádný kód.

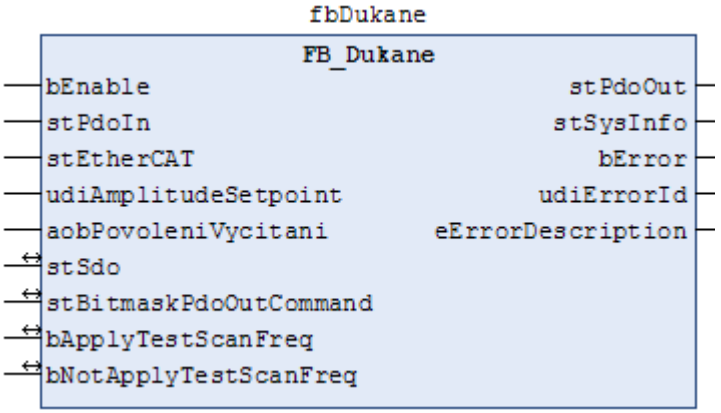
2.2.3 FB řízení generátoru

Jak bylo uvedeno v kapitole Plán práce, pokračujeme vytvořením FB, který bude vykonávat potřebnou funkci pro náš generátor. Níže je uveden seznam možností a vlastností funkčního bloku:

- Cyklická komunikace s SDO
- Testování funkčnosti sonotrody
- Skenování optimální frekvence sonotrody
- Nastavení pracovní, minimální a maximální frekvence
- Nastavení náběhové a doběhové rampy amplitudy
- Diagnostika a resetování chyb
- Řízení svařovacího cyklu

2.2.3.1 Popis FB

Rozhraní funkčního bloku je možno vidět na obr. 43.



Obr. 43 Popis FB

Interface jsem volil tak, aby bylo na první pohled vidět, jakého typu je daná proměnná. Název začíná datovým typem. První velké písmeno odděluje typ od vlastního popisu proměnné. Použité typy: b=bool, st=struktura (datový typ obsahující další datové typy - strukturu je potřeba zavést do projektu), udi=unsigned double integer, aob=array of bool (potřeba zkontrolovat velikost

pole, aby došlo ke správnému přiřazení proměnných), e=enum (proměnná, jejíž číselnou hodnotu vyjadřuje slovní popis). V této kapitole stručně popíši význam jednotlivých vstupních a výstupních proměnných a struktur. Uvedeny budou pouze důležité nebo zajímavé proměnné. Bližší informace jsou v příloze.

bEnable

Slouží k aktivaci bloku. Je to z toho důvodu, aby se nemusel měnit projekt při různých metodách svařování. Blok je v projektu vždy a je aktivován dle potřeby.

stPdoIn

Do stPdoIn se kopírují hodnoty ze vstupních PDO generátoru. Níže je tabulka se seznamem proměnných ve struktuře. S některými proměnnými se dále pracuje, jsou tedy uvedeny v kapitole 2.2.3.2.

ST_Dukane_PdoIn	
Název	Index objekt: podobjekt
System_Error_Status	0x2001:0x01
Initiate_Error_Status	0x2001:0x02
Process_Error_Status	0x2001:0x03
Generator_State	0x2001:0x04
Real_Time_Amplitude	0x2001:0x05
Real_Time_Frequency	0x2001:0x06
Real_Time_Power	0x2001:0x07
System_Status	0x2001:0x08
Temperature	0x2001:0x09
Cycle_Count	0x2001:0x0A
Good_Part_Count	0x2001:0x0B
Bad_Part_Count	0x2001:0x0C
Suspect_Part_Count	0x2001:0x0D
Aborted_Part_Count	0x2001:0x0E
Cycle_Part_Status	0x2001:0x0F
Cycle_Weld_Time	0x2001:0x10
Cycle_Weld_Peak_Power	0x2001:0x11
Cycle_Weld_Energy	0x2001:0x12
Active_Setup	0x2001:0x13
Running_Setup	0x2001:0x14
Running_Probe	0x2001:0x15

Tab. 2 Vstupní procesní data

stEtherCAT

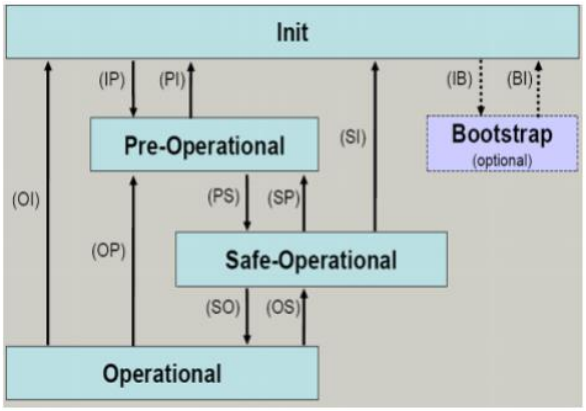
Obsahuje adresu, port a stav komunikace. Datové typy nejsou všechny stejné jako v případě PDO (UDINT), proto jsou obsaženy v tabulce.

ST_Dukane_EtherCAT	
EtherCAT_ID	ARRAY [0..5] OF USINT
EtherCAT_port	WORD
State	UINT
WcState_validData	BOOL
InputToggle	BOOL

Tab. 3 Struktura komunikačních proměnných

Pro zajímavost je uvedena proměnná *State*.

State: Stav komunikace. Základními stavy komunikace jsou Init, Pre-Operational, Safe-Operational, Operational a Boot. Na obr. 44. je znázorněn způsob přepínání mezi stavy [14].



Obr. 44Přepínání mezi komunikačními stavy [14]

K těmto základním stavům může EtherCAT nabývat ještě dodatečných informací. Stav EtherCAT zařízení je složen ze stavu zařízení jako takového a stavu jeho vstupního a výstupního portu. Výsledek může vypadat například následovně: *DeviceStatus: OP=(8), LinkStatus: Port 1_Missing link at port X(defined by EC_LINK_STATE_PORT_A/B/C/D)=(36)*. Číslo 36 je v hexadecimálním vyjádření 24 (20+4 => *Communication port B[port1]+Slave signals missing link*). Je tedy jasné, že *DeviceStatus* je pravý byte a *LinkStatus* levý byte, jak je uvedeno v tabulce 4.

ETHERCAT_STATE	
Stav	Hodnota
INIT	0x___1
PREOP	0x___2
BOOT	0x___3
SAFEOP	0x___4
OP	0x___8
Signal error	0x001_
Invalid system info read	0x002_
Initialization error occurred	0x004_
Slave disabled	0x008_
Slave not present	0x010_
Slave signals link error	0x020_
Slave signals missing link	0x040_
Slave signals unexpected link	0x080_
Communication port A	0x100_
Communication port B	0x200_
Communication port C	0x400_
Communication port D	0x800_

Tab. 4 Stavy komunikace

udiAmplitudeSetpoint

Požadovaná svařovací amplituda. Hodnota je udávána v [%]. Možný rozsah je 20% - 100%. Cokoliv mimo tento rozsah je zahozeno.

aobPovoleniVycitani

Možnost měnit vyčítání proměnných za běhu nám dává aobPovoleniVycitani. V tomto poli si vybereme, které proměnné z SDO chceme zrovna vyčítat a které ne. Díky tomu se nám výrazně urychlí obnova právě vyčítaných dat ve vizualizaci. Proměnné, které jsou určeny k vyčítání, jsou uloženy ve struktuře stSdo.

stSdo

Struktura, do které se ukládají vybrané SDO. Tato data zapisujeme do generátoru pomocí bloku *FB_EcCoESdoWrite* a vyčítáme blokem *FB_EcCoESdoRead*. Jedná se o standardní bloky, proto jejich popis nebudu z důvodu rozsahu práce uvádět. Obsah struktury je zobrazen v tabulce 5. S obsahem stSdo se dále v textu pracuje, proto proměnné popíši.

ST_Dukane_Sdo	
Název	Index objekt: podobjekt
RampUpTime	0x2101:0x00
RampDownTime	0x2102:0x00
FreeRun_Frequency	0x2117:0x00
Minimum_Frequency	0x2118:0x00
Maximum_Frequency	0x2119:0x00
Amplitude	0x2100:0x00
TestScanResult	0x2128:0x00
FreeRun_Previous	
Minimum_Previous	
Maximum_Previous	

Tab. 5 Vybraná servisní data

RampUpTime: Náběhová rampa amplitudy (čas, který má generátor na plynulou změnu žádané amplitudy, bližší info v kapitole 2.2.4.2).

RampDownTime: Doběhová rampa amplitudy (čas, který má generátor na plynulé přechod do nulové hodnoty amplitudy).

FreeRun_Frequency: Požadovaná frekvence. Jak tuto hodnotu určíme je blíže specifikováno v kapitole 2.2.4.1.

Minimum_Frequency: Spodní limita frekvence. Při překročení se vygeneruje chyba. Bližší specifikace v 2.2.4.1.

Maximum_Frequency: Horní limita frekvence. Při překročení se vygeneruje chyba. Bližší specifikace v 2.2.4.1.

Amplitude: Amplituda použitá při svařovacím cyklu v případě neaktivního bitu *Amp_Otf_Enable* z *stBitmaskPdoOutCommand*. Hodnota se nastavuje ve vizualizaci.

TestScanResult: Výsledek testu sonotrody. Výsledkem je optimální frekvence užívání. Požadavek na test se nastavuje se provádí z vizualizace.

Proměnné, u kterých je za podtržítkem *Previous* jsou jen k testování změny hodnoty.

stBitmaskPdoOutCommand

Slouží ke zjednodušení ovládání. Příkazy do generátoru se posílají jako číslo vyjádřené v hexadecimálním tvaru (viz Command v StPdoOut). Číslo je vytvořeno pomocí kombinace bitů s přiřazenými významy. Přiřazení čísla k bitu je znázorněno v tabulce 6. *Front Panel Lockout* se využívá u generátorů, které mají na sobě umístěn panel s nastavovacími tlačítky. My máme verzi bez tohoto panelu. Seznam bitů převáděných na číslo je v tabulce 6,.

ST_Dukane_BitmaskPdoOutCommand			
Name	Hexadecimal	Bit	Decimal
Run_Continuous	0x00000001	0000 0000 0000 0000 0000 0000 0000 0001	1
Run_Weld	0x00000002	0000 0000 0000 0000 0000 0000 0000 0010	2
Run_Test	0x00000004	0000 0000 0000 0000 0000 0000 0000 0100	4
Clear_Error	0x00000008	0000 0000 0000 0000 0000 0000 0000 1000	8
Amp_Otf_Enable	0x00000010	0000 0000 0000 0000 0000 0000 0001 0000	16
Auto_Stop	0x00000020	0000 0000 0000 0000 0000 0000 0010 0000	32
Front_Panel_Lockout		Not applicable	
Run_Scan	0x00000100	0000 0000 0000 0000 0000 0001 0000 0000	256
Clear_Setup_Refreshed_Output	0x80000000	1000 0000 0000 0000 0000 0000 0000 0000	2147483648

Tab. 6 Maskování výstupního PDO příkazu

Run_Weld: Svařovací cyklus je aktivován a vypínán společne s tímto bitem.

Run_Test: Test běží, pokud je bit v log.1. Test slouží k vyzkoušení generátoru a sontrody mimo svařovací cyklus. Testování se spouští z vizualizace. Amplituda je automaticky brána z *stSdo*, ať už je nahozen bit *Amp_Otf_Enable*, či ne.

Clear_Error: Při náběžné hraně dojde k resetování chyb v generátoru. Reset chyb se provádí z vizualizace

Amp_Otf_Enable: je–li tento bit nahozen, svařuje se amplitudou v PDO, pokud ne, bere si generátor hodnotu z SDO. Oba významy jsou popsány v daných sekcích.

Run_Scan: Generátor začně skenovat sonotrodu a vyhledávat optimální parametry algoritmem, který výrobce neudává. Sken je spuštěn náběžnou hranou. Výsledek je v *TestScanResult* v *stSdo*. Požadavek na sken se nastavuje z vizualizace.

Výsledné číslo získáme pomocí operací AND a OR. TwinCAT 2 nepodporuje bitový součin a součet typu UDINT, je potřeba jí převést na typ DWORD. Operaci přiřazení bitu *Amp_Otf_Enable* do výstupního čísla lze vidět na obr. 45.

```
IF stBitmaskPdoOutCommand.AMP_OTF_ENABLE THEN
    stPdoOut.Command := DWORD_TO_UDINT(UDINT_TO_DWORD(stPdoOut.Command) OR 16#00000010);
ELSE
    stPdoOut.Command := DWORD_TO_UDINT(UDINT_TO_DWORD(stPdoOut.Command) AND (NOT 16#00000010));
END_IF
```

Obr. 45 Algoritmus maskování výstupního PDO příkazu

bApplyTestScanFreq
Uložení optimální frekvence sonotrody *TestScanResult* z *StSdo*.

bNotApplyTestScanFreq
Zahození optimální frekvence sonotrody *TestScanResult* z *StSdo*.

stPdoOut
PDO, které se zapisují do generátoru. Z celé tabulky nás nejvíce zajímá *Command* a *Otf_Amplitude_Setpoint*.

ST_Dukane_PdoOut	
Název	Index objekt: podobjekt
Command	0x2002:0x01
Otf_Amplitude_Setpoint	0x2002:0x02
Active_Setup	0x2002:0x03
Running_Setup	0x2002:0x04
Running_Probe	0x2002:0x05

Tab. 7 Výstupní procesní data

Command: Číselná hodnota vyjádřená ovládacími bity z *stBitmaskPdoOutCommand*

Otf_Amplitude_Setpoint: Žádaná hodnota amplitudy - použitá v případě aktivace bitu *Amp_Otf_Enable* v *stBitmaskPdoOutCommand*.

stSysInfo

Výrobní informace generátoru jsou uloženy v stSysInfo. Jsou to ve skutečnosti také SDO jako v stSdo, ovšem zařadil jsem sem data, která se nevyčítají cyklicky, ale jen jednou a to po zapnutí generátoru. Popis proměnných je v tomto případě zbytečný. Názvy mluví za vše.

ST_Dukane_SysInfo	
Název	Index objekt: podobjekt
ModelNumber	0x2080:0x00
SerialNumber	0x2081:0x00
MB_App_Firmware_Version	0x2082:0x00
FP_App_Firmware_Version	0x2083:0x00
MP_Boot_Firmware_Version	0x2084:0x00
FP_Boot_Firmware_Version	0x2085:0x00

Tab. 8 Struktura systémových informací z SDO

bError

Aktivace bError nastane v případě jakékoliv chyby generátoru.

udiErrorId

Aby bylo možné najít zdroj chyby, byla zavedena proměnná udiErrorId, ve které je uloženo číslo chyby.

eErrorDescription

Jelikož je zdrojů chyb více (chyba v generátoru (například překročení teploty), chyba komunikace, atd.) a každý zdroj uvádí číslo chyby, může dojít k tomu, že jedno číslo bude mít více významů. Pro přesnější diagnostiku je zaveden eErrorDescription, kde je uveden zdroj chyby. Na obr. 46 je seznam rozlišených zdrojů chyb.

```
TYPE E_Dukane_ErrorDescription :
(
    NoError:=0,                     (* bez závady *)
    EcCoeSdoRead:=1,                (* chyba při provádění komunikačního funkčního bloku "FB_EcCoESdoRead" - kód závady v parametru "ErrorId" *)
    EcCoESdoWrite:=2,               (* chyba při provádění komunikačního funkčního bloku "FB_EcSoEWrite" - kód závady v parametru "ErrorId" *)
    UltrasonicGenerator:=3,         (* ultrazvukový generátor DUKANE je v závadě *)
    Device_NotOP:=4                 (* obsluhované zařízení na komunikační sběrnici EtherCat není připravené k provozu *)
);
END_TYPE
```

Obr. 46 Rozlišované zdroje chyb

2.2.3.2 Bližší popis bitově významových proměnných a stavů

Tato kapitola se bude stručně zabývat pouze principem vymaskování jednotlivých bitů ze vstupních PDO objektů.

Bitově významové vstupní proměnné jsou *System_Error_Status, Initiate_Error_Status*, *Process_Error_Status* a *Cycle_Part_Status*.

Nyní uvedu seznam bitů, které jsou v daných proměnných skryty a popíši algoritmus pro jejich vymaskování. V tabulkách je kromě názvu i hexadecimální vyjádření polohy daného bitu.

2.2.3.2.1 Seznam proměnných s maskou

stPdoIn -> System_Error_Status

ST_Dukane_BitmaskSystemErrorStatus	
Název	Maska
Internal_Error	0x0002000F
Overload_Freq_Lock_Failed	0x00000010
OverLoad_Freq_Lock_Lost	0x00000020
Peak_Overload_Positive	0x00000040
Peak_Overload_Negative	0x00000080
Average_Overload	0x00000100
Current_Loop	0x00000200
Power_Not_Ok	0x00000400
Over_Temperature	0x00000800
Over_Amplitude	0x00001000
Frequency_Bounds_Exceeded	0x01000000
Under_Voltage	0x10000000

Tab. 9 Maskování systémových chyb

stPdoIn -> Generator_State

Generator_State	
Hodnota	Název
0	Idle
1	Test
2	Auto_Start_Delay
3	Pre-PreTrigger
4	PreTrigger
5	Trigger_Delay
6	Continuous
7	Weld 1
8	Weld 2
9	Scrub
10	Dynamic_Hold
11	Static_Hold
12	AfterBurst_Delay
13	AfterBurst_Duration
14	PreWeld_Scan
15	Done
16	RESERVED
17	Downstroke_Pause

Tab. 10 Stavy generátoru

stPdoIn -> System_Status

ST_Dukane_BitmaskSystemStatus	
Název	Maska
Good_Part	0x00000001
Bad_Part	0x00000002
Suspect_Part	0x00000004
Cycle_Start_Rejct	0x00000008
Amp_Regulation	0x00000010
Power_Regulation	0x00000020
Trigger	0x00000040
US_Status	0x00000080
Power_Ok	0x00000100
MPC_Ready	0x00000200
Ready	0x00000400
Anyfault	0x00000800
OverLoad	0x00001000
ESTOP	0x00002000
Online	0x00004000
In_Hold	0x00008000
In_Cycle	0x00010000
In_Cycle_no_AB	0x00020000
Overtemp	0x00040000
In_After_Burst	0x00080000
In_Test	0x00100000
In_Test_Scan	0x00200000
Setup_Changed	0x80000000

Tab. 11 Maskování systémových statusů

stPdoIn -> Cycle_Part_Status

ST_Dukane_BitmaskCyclePartStatus	
Název	Maska
Good_Part	0x00000001
Bad_Part	0x00000002
Suspect_Part	0x00000004
Lower_Time_Bad_Limit	0x00000100
Lower_Time_Suspect_Limit	0x00000200
Upper_Time_Suspect_Limit	0x00000400
Upper_Time_Bad_Limit	0x00000800
Lower_Peak_Power_Bad_Limit	0x00010000
Lower_Peak_Power_Suspect_Limit	0x00020000
Upper_Peak_Power_Suspect_Limit	0x00040000
Upper_Peak_Power_Bad_Limit	0x00080000
Lower_Energy_Bad_Limit	0x01000000
Lower_Energy_Suspect_Limit	0x02000000
Upper_Energy_Suspect_Limit	0x04000000
Upper_Energy_Bad_Limit	0x08000000

Tab. 12 Maskování výsledku svařovacího cyklu

2.2.3.2.2 Algoritmus vymaskování

V projektu jsou použity dva způsoby vymaskování hodnot. První způsob maskuje jen jeden přesně určený bit, viz obr. 47. Jednoduše řečeno – každý bit ve vstupním čísle má svou vlastní reprezentaci. Toto odmaskování se provede jednoduchou operací AND mezi dvěma WORDY.

```
stBitmaskSysteErrorStatus.Overload_Freq_Lock_Failed := (UDINT_TO_DWORD(stPdoIn.System_Error_Status) AND 16#00000010) = 16#00000010;
```

Obr. 47 Algoritmus odmaskování do jednoho bitu

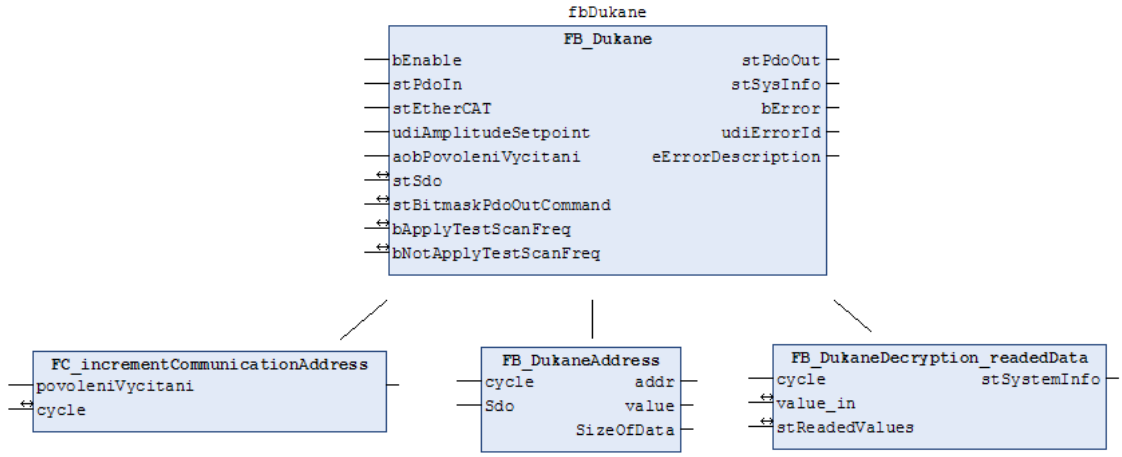
Druhým způsobem vymaskujeme bit na základě množiny vícero aktivních bitů. V tomto případě je výstupní bit aktivní při alespoň jednom aktivním bitu z daného rozsahu. To nám pokryje situaci, kdy například prvních pět bitů má svůj význam a druhých pět bitů neumíme rozlišit, takže jim přiřadíme jeden výstupní bit. Prvních pět bitů bychom vymaskovali první metodou, zbytek metodou č. 2, která je znázorněna na obr. 47.

```
stBitmaskSysteErrorStatus.Internal_Error := (UDINT_TO_DWORD(stPdoIn.System_Error_Status) AND 16#0002000F) <> 16#00000000;
```

Obr. 48 Algoritmus odmaskování z množiny bitů

2.2.3.3 Integrované FB

FB_Dukane má v sobě integrovány další dva bloky (FB) a jednu funkci (FC).



Obr. 49 Integrace bloků

FC_incrementCommunicationAddresss

Je funkce, jejímž výstupem je číslo, představující pozici další vyčítané / zapisované proměnné. Číslo je určeno na základě povolených bitů v poli aobPovoleniVycitani.

FB_DukaneAddress

Vybere adresu, přiřadí k ní proměnnou a velikost proměnné. Tyto informace vygeneruje pomocí výstupu z *FC_incrementCommunicationAddresss*.

FB_DukaneDecryption_readedData

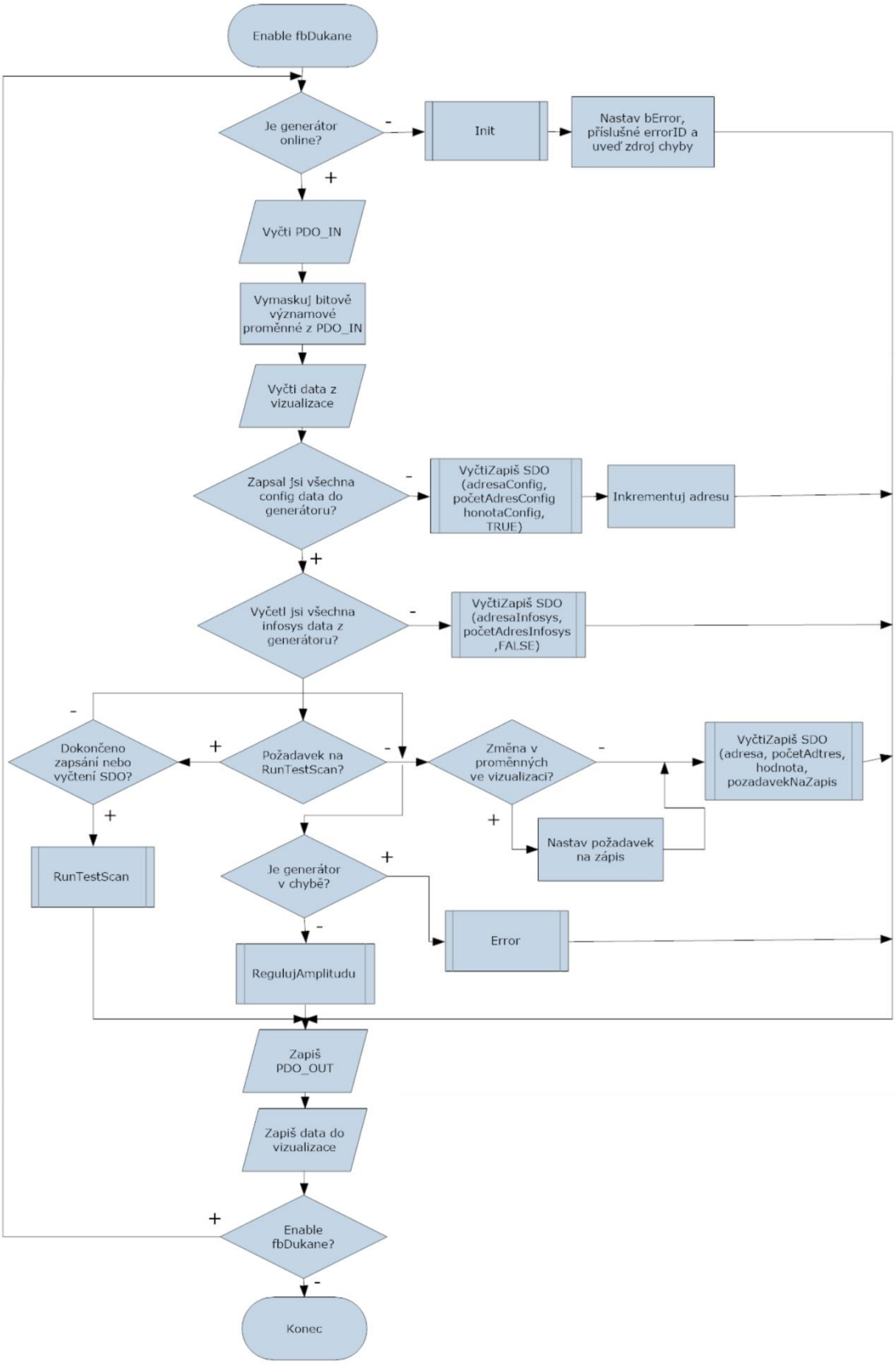
Dešifruje přijaté zprávy. Přijaté zprávy se ukládají do pole složeného z unsigned double integer, jelikož je tento datový typ nejčastěji vyčítaný. Důvod použití pole je ten, že zprávy jsou různě veliké. Po každém přijetí nové zprávy se toto pole přečte, dešifruje, přesune do žádané proměnné a vynuluje. Nulování je z toho důvodu, aby nedošlo k ovlivnění nové (kratší) zprávy. Dešifrování je v tomto případě převedení přijaté zprávy na potřebný význam. Příkladem může být přijatá zpráva vyjádřená polem UDINT na textový řetězec správného významu. V případě použití ASCII kódování je třeba nejprve pole UDINT zkopírovat do pole BYTE. Číslu v BYTE podobě přiřadíme číslo, nebo písmeno, které v ASCII reprezentuje. Jedním z příkladů může být číslo 65, které vyjadřuje písmeno A.

2.2.3.4 Vnitřní algoritmus

Tato kapitola slouží k seznámení s vnitřním algoritmem bloku. Jako první v krátkosti slovně popíši chování a následně budou uvedeny podrobnější vývojové diagramy.

Blok začne pracovat povolením bitu Enable. Po aktivaci blok zkontroluje, zda je generátor na sběrnici EtherCAT dostupný. V případě, že generátor nekomunikuje, zavolá se podprogram *Init*. Pokud generátor odpovídá, tak se vyčtou vstupní PDO. U potřebných proměnných se číselná hodnota převede na bity. Vyčtou se data z vizualizace a případné změny se zanesou do generátoru. Po aktualizaci konfiguračních proměnných se ze servisních důvodů vyčtou systémové informace o generátoru. Tyto úkony se provedou vždy jen jednou a to po spuštění bloku. Následující kroky se provádějí stále dokola, dokud je povolen bit Enable. Program zjišťuje, zda je požadavek na otestování sonotrody. Pokud ne, pak se neustále dokola vyčítají/zapisují data z/do generátoru. Kontroluje se, zda je generátor v chybovém stavu. Pokud ano, ukončí se svářecí cyklus (amplituda je automaticky vynulována) a patřičně se na chybu zareaguje, viz. *Error*. Pokud je generátor v pořádku, probíhají svářecí cykly, viz. *RegulujAmplitudu*. V případě, že je požadavek na sken sonotrody, pak je proveden *RunTestScan*. Hlavní algoritmus, popsaný výše, je zobrazen na obr. 50. Zmíněné podprogramy jsou na obrázcích 51-55.

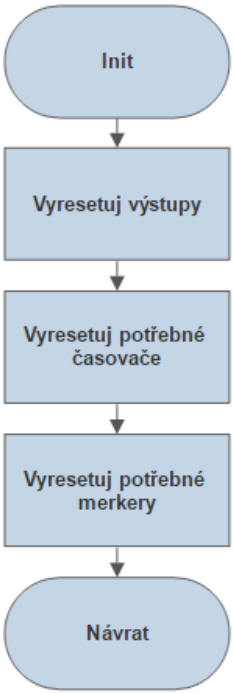
Hlavní algoritmus



Obr. 50 Vnitřní algoritmus

Podprogram Init

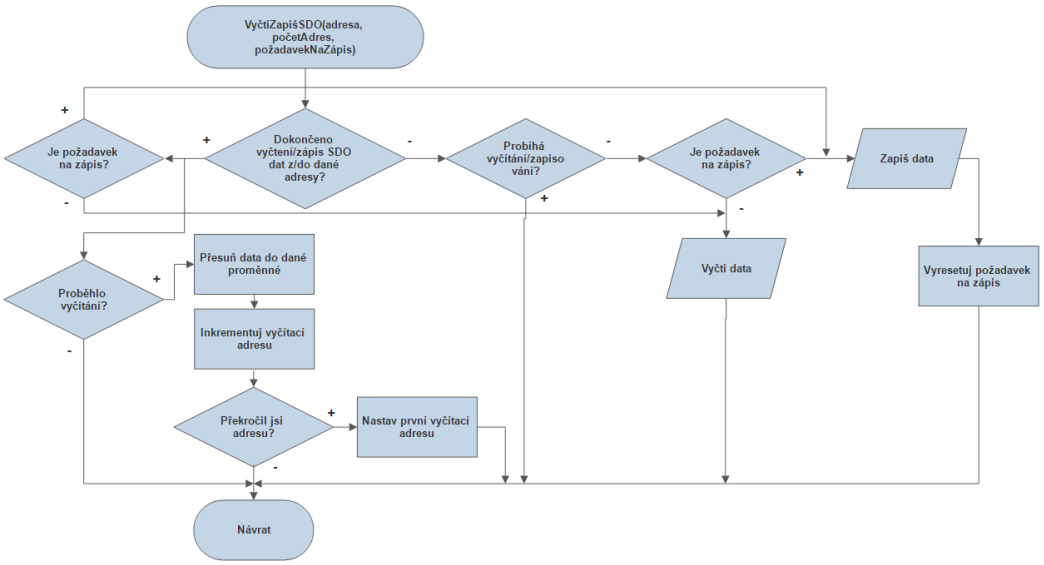
V init módu jsou vyresetovány/vynulovány potřebné PDO, časovače, atd.. Je to z důvodu správného znovuspuštění cyklu. Důvod je jasný. Nemůžeme vědět, kdy se generátor znovu připojí (servisní zásah) a nemůžeme například nechat nastavenou pracovní amplitudu a nahozen bit *SVÁŘEJ*. Po naběhnutí komunikace by nám stroj začal svářet a mohlo by dojít ke zranění člověka, nebo ke zničení sonotrody. Časovače nám kontrolují uplynutí doby sváření (chceme, aby další sáček byl svařen správně).



Obr. 51 Podprogram Init

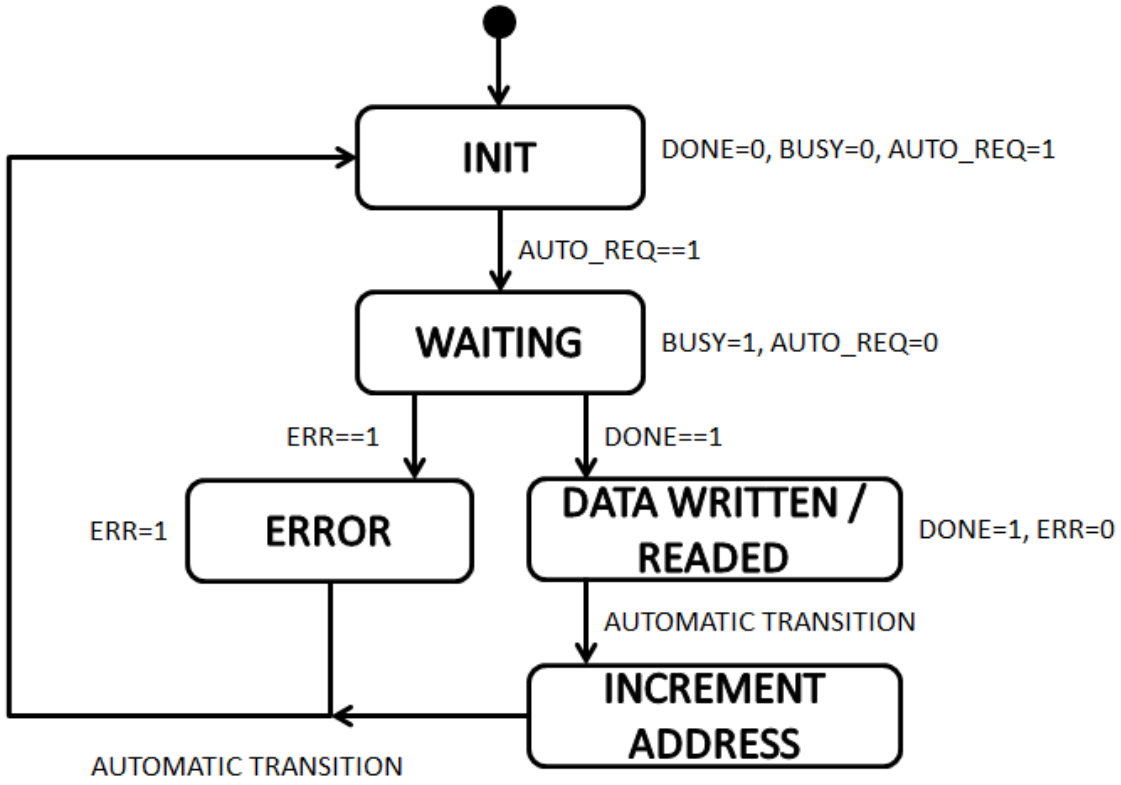
Podprogram zajišťující komunikaci s SDO generátoru

Pro vyčítání/zapisování SDO jsou využity bloky *FB_EcCoeSdoRead /FB_EcCoeSdoWrite*. Zapisování je aktivní pouze v případě, že dojde ke změně hodnoty v proměnné. Jinak se cyklicky provádí vyčítání dat. Kontroluje se překročení chtěné adresy a zareaguje se na ní nastavením adresy první. Uveden je vývojový diagram na obr. 52 i stavový diagram na obr. 53.



Obr. 52 Podprogram VyčtiZapiš

Na stavovém diagramu (obr. 53) můžeme vidět způsob řešení komunikace. Pro vysvětlení: znak „=“ znamená přiřazení hodnoty proměnné v daném stavu a „==“ je podmínka přechodu mezi stavy. Jelikož se vyčítání a zapisování chová autonomně, neboli bez účasti požadavků na čtení/zápis (zapíše se automaticky při změně hodnoty), je požadavek na výměnu dat automaticky nastaven po spuštění bloku, nebo při dokončení operace, ať už skončí vyčtením/zapsáním nebo chybou. Z tohoto důvodu si můžeme všimnout, že se chyba ERR automaticky neresetuje v inicializačním kroku INIT, ale až po následující správné výměně dat. Kdyby tomu tak nebylo, tak by chyba nezůstala v log. 1 a nebylo by možné jí diagnostikovat.



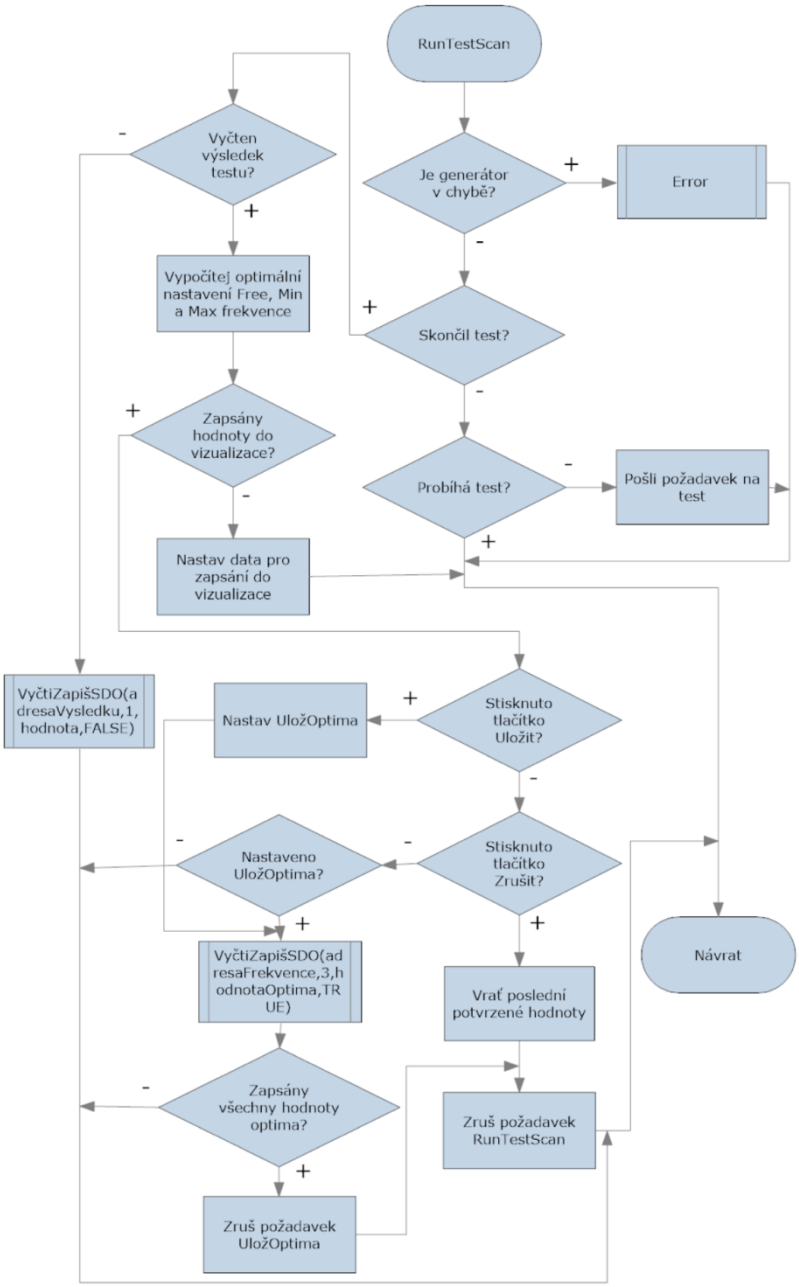
Obr. 53 Stavový diagram výměny SDO

Na obrázku 54 jsou zobrazena vyčítaná a zapisovaná data. Data, která se zapisují se také vyčítají. Je to z důvodu porovnání změny hodnoty. Uvedeny jsou pouze proměnné, které se komunikují pomocí SDO. Data, která se komunikují pomocí PDO je zbytečné uvádět, jelikož se o komunikaci stará ŘS sám.

Obr. 54 Komunikace s generátorem pomocí SDO

Podprogram skenování ideální frekvence dané sonotrody

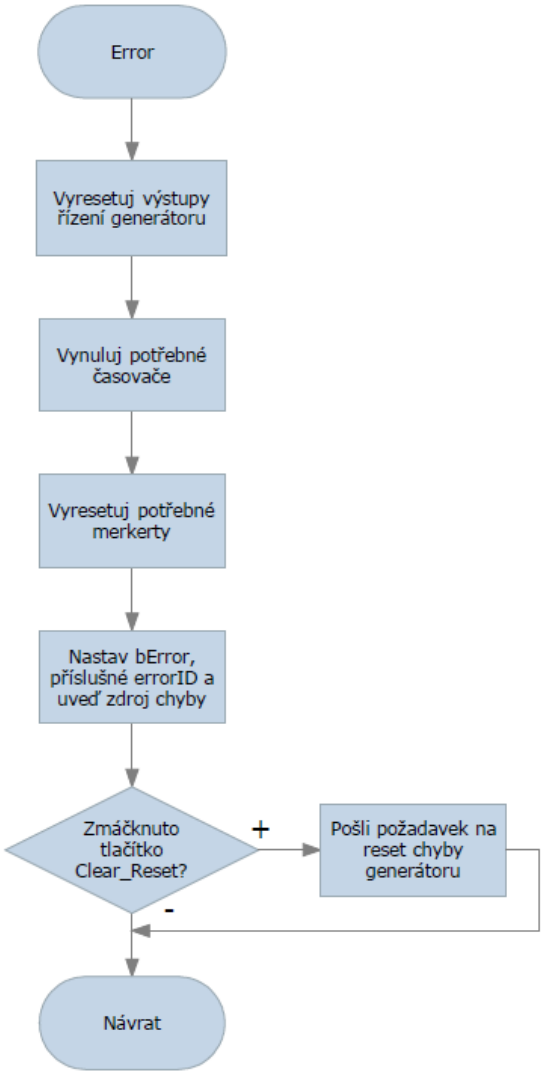
Požadavek na sken se nastavuje z vizualizace. Po odeslání požadavku generátoru se kontroluje ukončení procesu. Po ukončení se z generátoru vyčte výsledek a vypočítají se nové hodnoty frekvencí (*Free run*, *Max* a *Min*). Tyto hodnoty se zapíší do vizualizace a čeká se na jejich potvrzení. Pokud obsluha rozhodne, že se nové hodnoty nepoužijí, pak se vrátí předchozí nastavené. V případě, že ano, uloží se do generátoru a použijí se pro svařovací cyklus. Proč je dobré přenastavit frekvenci sonotrody je vysvětleno v kapitole 2.2.4.1.



Obr. 55 Podprogram RunTestScan

Podprogram Error

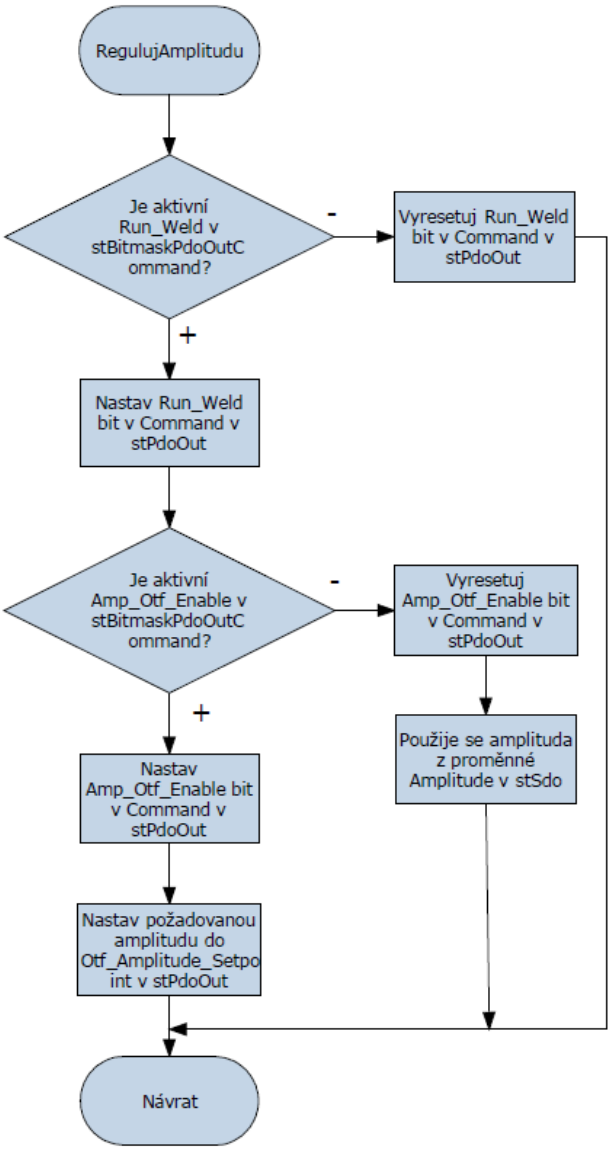
Po zjištění chyby se vyresetují potřebné PDO výstupy, časovače, atd. a nastaví se výstupní bit *bError*. Vyčte se *errorID* a určí zdroj chyby, viz kapitola 2.2.3.1. Čeká se na pokyn pro reset chyby, viz. *Clear_Error* v *stBitmaskPdoOutCommand*.



Obr. 56 Podprogram Error

Podprogram volby regulace amplitudy

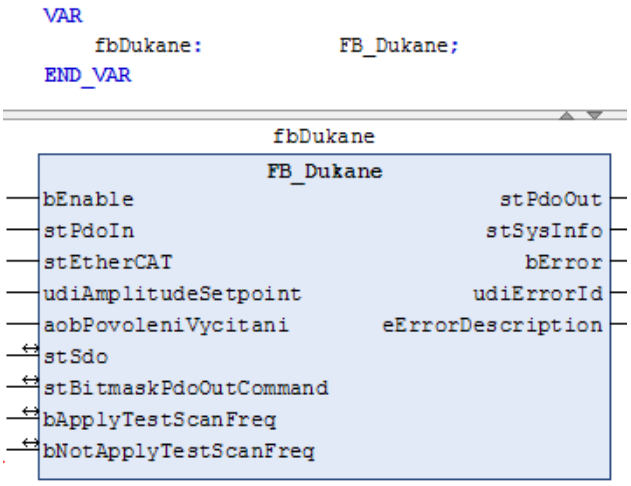
V podprogramu „RegulujAmplitudu“ je možnost dvou regulací. Buď regulujeme pomocí PDO objektu, který můžeme měnit v reálném čase, nebo pomocí SDO objektu, jehož změna se projeví vždy po dokončeném cyklu. Tato možnost je z důvodu testování. Aby nedošlo k přepisování testovací hodnoty během automatického cyklu, jsou zavedeny dvě nezávislé proměnné. Do PDO objektu zapisuje program v PLC, viz obrázek 57. Do SDO objektu se zapíše vždy při změně hodnoty ve vizualizaci. Přepínání mezi použitím PDO a SDO je vysvětleno v kapitole 2.2.3.1.



Obr. 57 Podprogram RegulujAmplitudu

2.2.3.5 Volání bloku v programu

Po vytvoření bloku ho musíme v projektu zavolat a definovat (například v programu *MAIN*), jinak by sice byl v projektu zaveden, ale nevykonával by se. Volání a definice jsou znázorněny na obr. 58. Definice je mezi řádky *VAR* a *END_VAR*. Pro přehlednost jsem volání uskutečnil pomocí jazyka FBD.



Obr. 58 Vložení instance FB do projektu

2.2.4 Regulace svařovacího cyklu

Za generátorem je připojena sonotroda, jejíž chování ovlivňuje generátor na základě příkazů a parametrů z PLC. Svařovací etapa začíná příkazem *Svářej*. Základní parametry potřebné k řízení jsou:

- Čas náběhu (Ramp up time)
- Čas doběhu (Ramp down time)
- Čas svařování (Welding time)
- Amplituda (Amplitude)
- Užitá frekvence (Free run frequency)
- Maximální frekvence (Max frequency)
- Minimální frekvence (Min frequency)

S těmito parametry budeme dále pracovat v kapitole 2.2.4.2.

2.2.4.1 Zkoumané metody řízení generátoru

Během testování svařovacích cyklů bylo zkoumáno několik metod. Jako první jsem zvolil časové řízení. Výhodou byla jednoduchost. V podstatě šlo o svařování konstantní amplitudou a frekvencí s nastavitelným časem. Nevýhodu jsem viděl v tom, že do budoucna by tato metoda mohla být jedním z faktorů, které zdržují rychlost balení.

V druhém případě jsem zvolil konstantní čas a nastavitelnou amplitudu. Tato metoda sice zaručuje, že bude u všech fólií dodržena stejná délka svařovacího cyklu, ovšem byla by potřeba opětovné úpravy amplitudy při změně výkonu stroje.

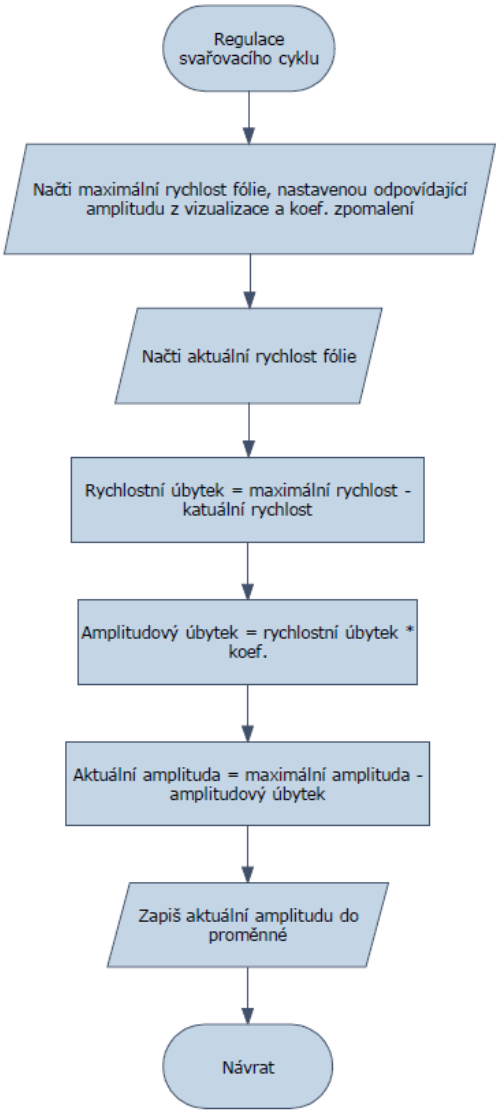
U obou předchozích metod je pro danou fólii potřeba nastavit operátorem buď čas, nebo amplitudu. Hodnoty jsou v cyklu neměnné.

Nejlepších výsledků bylo dosaženo programovou úpravou zvolené amplitudy. Operátor mění amplitudu (maximální) do doby, dokud při maximální rychlosti posuvu fólie nevznikne požadovaná kvalita svaru sáčku. Hodnota amplitudy je poté programově měněna na základě změny rychlosti posuvu fólie. Konstantou je v tomto případě délka odvinuté fólie během svařování. Jinými slovy, na čím větší výkon je stroj nastaven, tím je větší amplituda, ale zároveň kratší čas svařování (daná délka fólie se odvine rychleji). Možností, jak lze ovlivnit vypočítanou amplitudu (například, když při určité rychlosti není svar tak kvalitní), lze upravit koeficient úbytku (dále jen koef), jehož hodnota je defaultně nastavena na hodnotu 1. Níže je uveden vývojový diagram a vzorec pro výpočet požadované amplitudy.

$$v\_ubytek = v\_\max - v\_akt \quad [Hz] \tag{1}$$

$$amp\_ubytek = [v\_ubytek].\{koef\} \quad [Hz] \tag{2}$$

$$amp\_akt = amp\_\max - amp\_ubytek \quad [Hz] \tag{3}$$



Obr. 59 Regulace svařovacího cyklu

Nyní máme vypočtenou amplitudu. Než se ovšem přikročí k samotnému svařování, je dobré upravit svařovací frekvenci vzhledem k použité sonotrodě. Jak se ideální frekvence zjistí, je popsáno v kapitole 2.2.3.4. Žádné dvě sonotrody nejsou úplně totožné, proto se ladí každá jedna sonotroda zvlášť. Dle doporučení výrobce nenecháváme vypočítanou frekvenci jako užitou, ale upravujeme jí, společně s minimální a maximální frekvencí, dle vztahů níže.

$$FreeRunFrequency = OptimalFrequency - 100 \quad [Hz] \quad (4)$$

$$MinimumFrequency = FreeRunFrequency - 500 \quad [Hz] \quad (5)$$

$$MaximumFrequency = FreeRunFrequency + 500 \quad [Hz] \quad (6)$$

2.2.4.2 Dopravní zpoždění amplitudy

Jen pro shrnutí zopakuji, že parametry *Čas náběhu* a *Čas doběhu* se nastaví při spuštění generátoru. *Amplituda* je proměnná v čase na základě rychlosti posuvu fólie a užitého materiálu.

U svařovacích cyklů musíme počítat i s časy potřebnými ke změně amplitudy. Časová změna je znázorněna na výpočtech (7), (8), (9) a (10) [15]. V příkladu je použit parametr *RampUpTime* (náběhová rampa), *StartAmplitude* (startovací amplituda) a *EndAmplitude* (cílová amplituda). Reálná změna (*RampTime*) je ovlivněna nejen parametrem *RampUpTime*, ale ještě startovací amplitudou (od které nárůst, či pokles začíná). Při řízení používám náběhovou rampu 100ms, z důvodu jasných přepočtů (aby nedošlo k výsledku *RampRate=1*) používám ve vzorcích rampu 150ms.

$$RampRate = \frac{StartAmplitude}{RampUpTime} = \frac{100\%}{150ms} = 0.667\,\%/ms \qquad (7)\ [15]$$

$$RampTime = \frac{StartAmplitude - EndAmplitude}{RampRate} = \frac{100\% - 20\%}{0.667\,\%/ms} = 120ms \qquad (8)\ [15]$$

Je vidět, že změna ze 100% amplitudy na 20% amplitudy bude trvat 120ms. Přesvědčíme se o tom, že obráceným postupem se čas bude lišit. Nastavíme tedy startovací amplitudu na 20%. Nejprve ovšem musíme znovu přepočítat *RampRate*.

$$RampRate = \frac{StartAmplitude}{RampUpTime} = \frac{20\%}{150ms} = 0.133\,\%/ms \qquad (9)\ [15]$$

$$RampTime = \frac{StartAmplitude - EndAmplitude}{RampRate} = \frac{20\% - 100\%}{0.133\,\%/ms} = 600ms \qquad (10)\ [15]$$

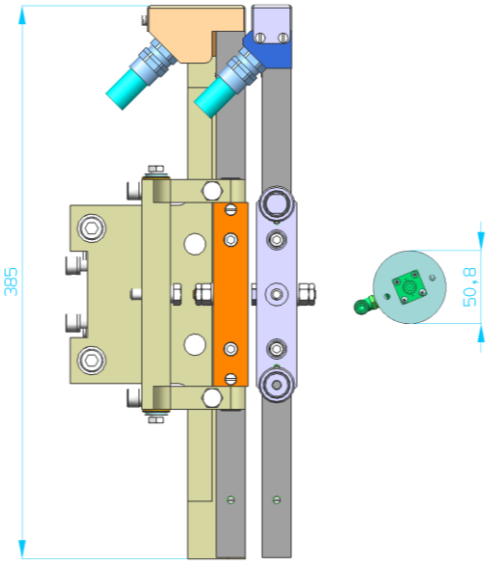
2.2.5 Výsledný sáček

V této kapitole je popsán sáček s výsledným svarem. Zmíněn je i vedlejší přínos použití ultrazvukové metody.

Na obr. 60 je sáček, na jehož realizaci byly použity dvě různé metody svařování. Spodek sáčku je svařen pomocí ultrazvukové metody. Boční svary jsou realizovány tepelnými čelistmi. Svar po tepelných čelistech je na první pohled rozeznatelný svým vzorkem. Snímky sáčku byly vytvořeny na výstavě Interpack 2017, kde byl stroj uveden.

Obr. 60 Výsledný sáček

Použití sonotrody má oproti tepelným (a ostatním, v kapitole 1.1, popsaným) čelistem další výhodu. Tou je potřebná výška k realizaci svaru. Na obr. 61 je na první pohled vidět úspora výšky. Tato výška se může využít pro použití výkonnějšího dávkovače produktu (výkonnější dávkovače bývají prostorově náročnější).



Obr. 61 Úspora výšky stroje

Shrnutí výhod: možnost balení problematického zboží, nižší výška stroje, kontinuální režim svařování.

2.2.6 Vizualizace

Cílem této kapitoly je usnadnit práci programátorovi, který tvoří vizualizační část balicích strojů. Bylo potřeba implementovat a graficky rozvrhnout rozmístění použitých tlačítek, bitů, které reprezentují stav generátoru, informačních hlášek a číselných hodnot proměnných v generátoru.

Z důvodu omezeného prostoru na panelu je vizualizace opatřena posuvníkem. Horní polovina je zobrazena na obr. 62, dolní polovinu můžeme vidět na obr. 63. Níže je popis struktury a nejzajímavějších proměnných. Myslím si, že s ohledem na přílohu není nutno popisovat vše.

Jelikož ultrazvukový generátor není jediné připojené zařízení, je generátor jednou z možností v záložkách. V záložce *Ultrazvukový generátor* si můžeme vybrat požadovaný generátor (pokud existuje více, než jeden). Nejvýše položená část pod vybraným generátorem slouží k rychlé diagnostice. Jelikož zdrojů závad může být větší množství a tím pádem by se do vizualizace všechny nevešly, rozdělil jsem chyby na mazatelné (*Any fault* – chyby, které se dají vyresetovat) a nejpravděpodobnější nemazatelné (např. *Power not ok, Under voltage,* atd.). Kromě závad můžeme vidět například stav *Ready* a ve svařovacím cyklu *In cycle*. Druhá část je určena nejen pro diagnostiku aktuálních hodnot, ale i pro nastavování generátoru. Modře označená proměnná *Regulation amplitude* je nastavitelná a popisoval jsem ji v kapitole 2.2.3.1. Tlačítkem *Run test* aktivujeme sonotrodu s amplitudou ze zmíněné proměnné. Aktuální amplitudu sonotrody vidíme v *Realtime amplitude*. Lze tedy jednoduše vyzkoušet jak protečení dat do generátoru, tak i shodu nastavené a aktuální amplitudy. Význam *Ramp up time* je osvětlen v kapitole 2.2.4.2.

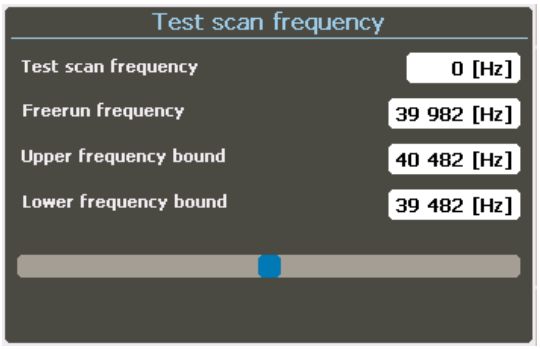
Obr. 62 Vizualizace - část 1

Další tlačítko je *Run test scan*, popsané v kapitole 2.2.3.1 jako *RunScan*. Po stisknutí tohoto tlačítka se vynoří vyskakovací okno (obr. 64). Posledními informacemi jsou sériové číslo, model, aktuálně použitý firmware, atd.. (obr. 63)

Obr. 63 Vizualizace - část 2

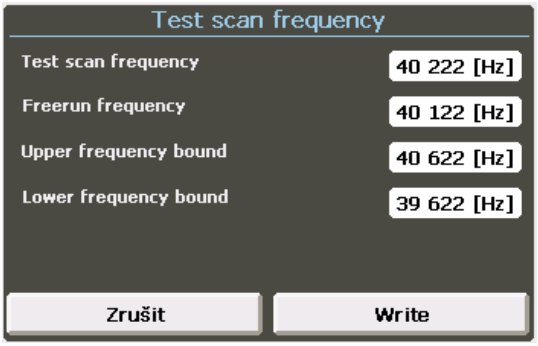
Obr. 64 Vynoření vyskakovacího okna

Vyskakovací okno z obrázku 64 můžeme opustit tlačítkem *Zrušit*. Tlačítkem *Run test* spustíme skenování sonotrody. Na obrázku 65 vidíme, jak se okno změní v případě žádosti o sken.



Obr. 65 Čekání na výsledek skenování sonotrody

Po dokončení operace skenování můžeme výslednou hodnotu frekvence zahodit pomocí tlačítka *Zrušit*, nebo uložit stisknutím na *Write* (obr. 66). Na obrázku 66 můžeme dále vidět, že náš algoritmus správně spočítal *FreeRun*, *Maximum* a *Minimum* frekvencí na základě rovnic (4), (5), a (6).



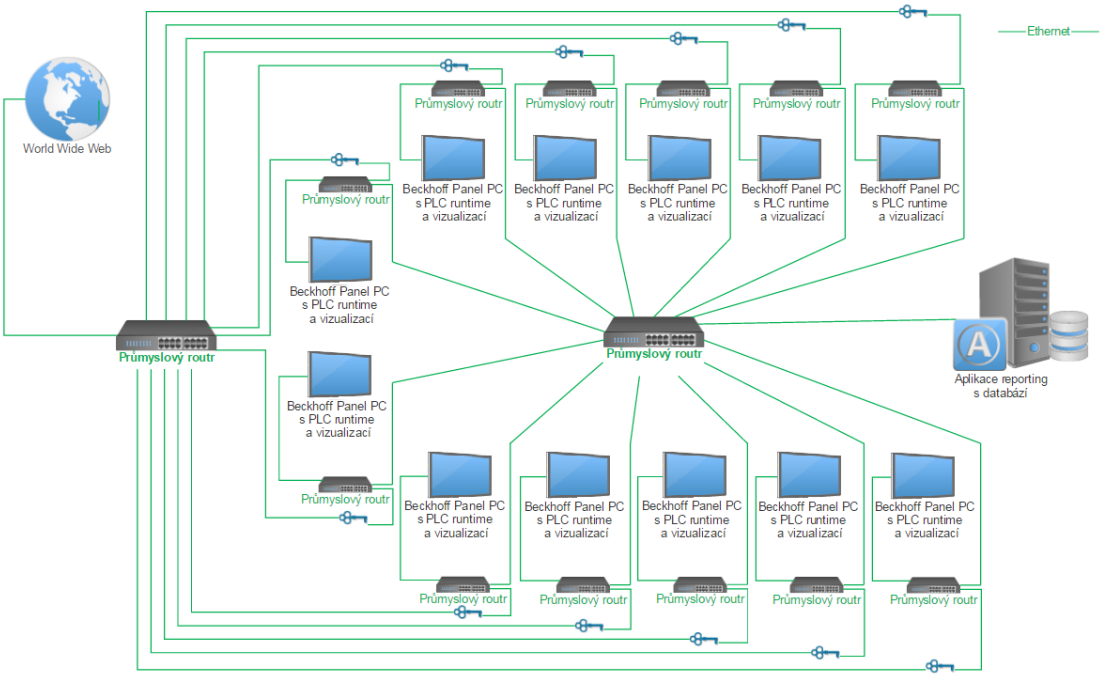
Obr. 66 Uložení / Zahození výsledku skenování sonotrody

2.3 Reporting

Reporting zajišťuje sběr a vyhodnocení dat. Důvod vybrání platformy SSRS byl ten, že není potřeba dalších vývojových prostředků, než se ve firmě v současné době používají. SSRS je součástí Visual Studia, ve kterém se tvoří vizualizace pro balicí stroje. V první podkapitole je znázorněna topologie propojení aplikace se stroji.
V kapitole 2.3.2 je ukázáno vytvoření databáze s procedurami. Kapitola 2.3.3 nás provede vytvořením aplikace včetně propojení s databází, ukázkou tvorby reportu a zdrojových kódů. Poslední část kapitoly 2.3.3 je ukázka výsledné aplikace včetně vygenerovaného reportu.

2.3.1 Sběr dat

Na obrázku 67 je zobrazen princip propojení strojů a reportingu. Komunikace mezi stroji a aplikací je pomocí ADS, které je popsáno v kapitole 1.3.3. Aplikace je spuštěná na stejném serveru, kde je uložená databáze. Aplikace se neustále dotazuje strojů na jejich stavy, a pokud je potřeba, uloží je do databáze. Zákazník požadoval možnost uzamknout stroj před přístupem zvenčí. Proto je topologie zvolena tak, aby mohla aplikace sbírat data nezávisle na vnějším přístupu.



Obr. 67 Schéma propojení aplikace a strojů

Druhým požadavkem byl samozřejmě obsah reportu.

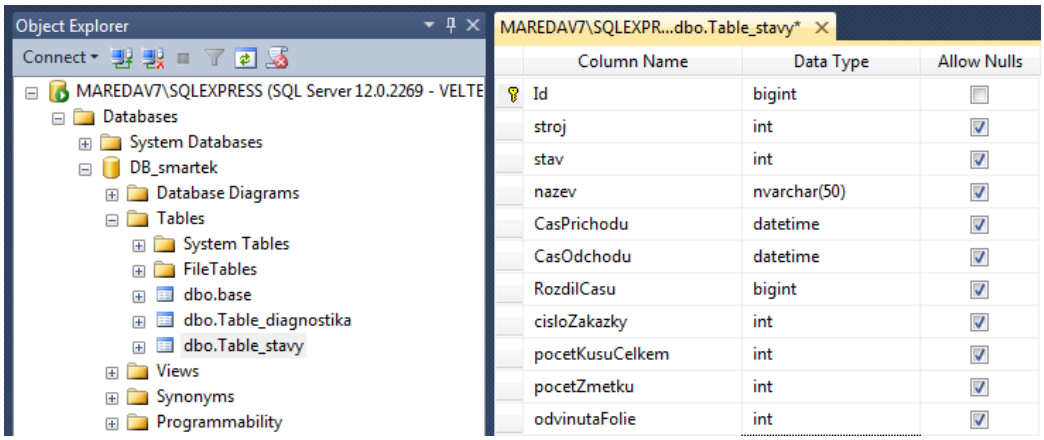
Obsah reportu:

- Vytížení stroje během směny – celkové i časové
- Aktivní zakázka / zakázky na stroji během směny
- Počet vyrobených kusů v zakázce
- Počet zmetkových kusů v zakázce
- Délka odvinuté fólie

Všechna tato data se budou ukládat do databáze. Nyní přistoupíme k jednotlivým krokům tvorby aplikace včetně uložení dat do databáze.

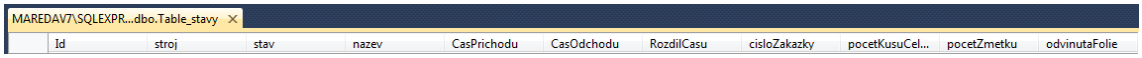
2.3.2 Vytvoření databáze

Jako první si vytvoříme databázi. Použijeme *Microsoft SQL Server Management Studio*. Ve složce *Databases* si vytvoříme novou databázi a v ní tabulky s proměnnými. Seznam proměnných je vidět na obr. 68.



Obr. 68 Obsah tabulky

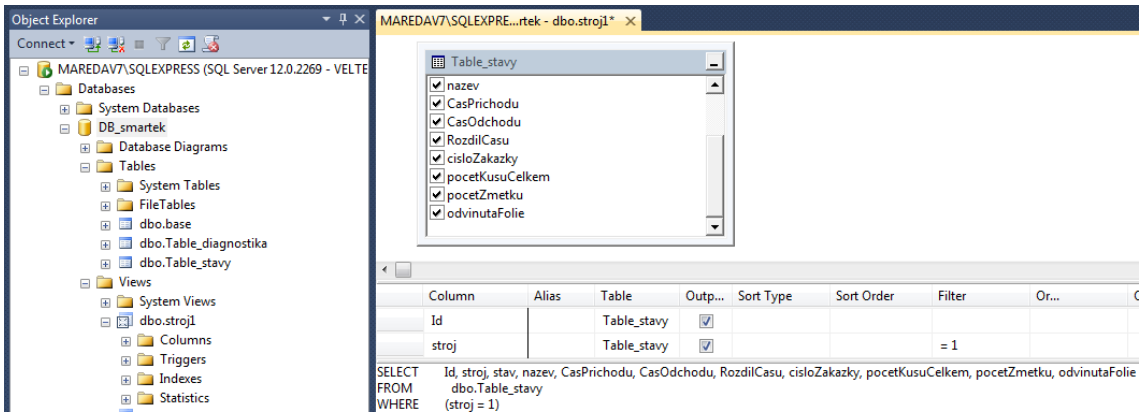
Počet proměnných značí počet sloupců tabulky, viz obr. 69.



Obr. 69 Sloupce tabulky

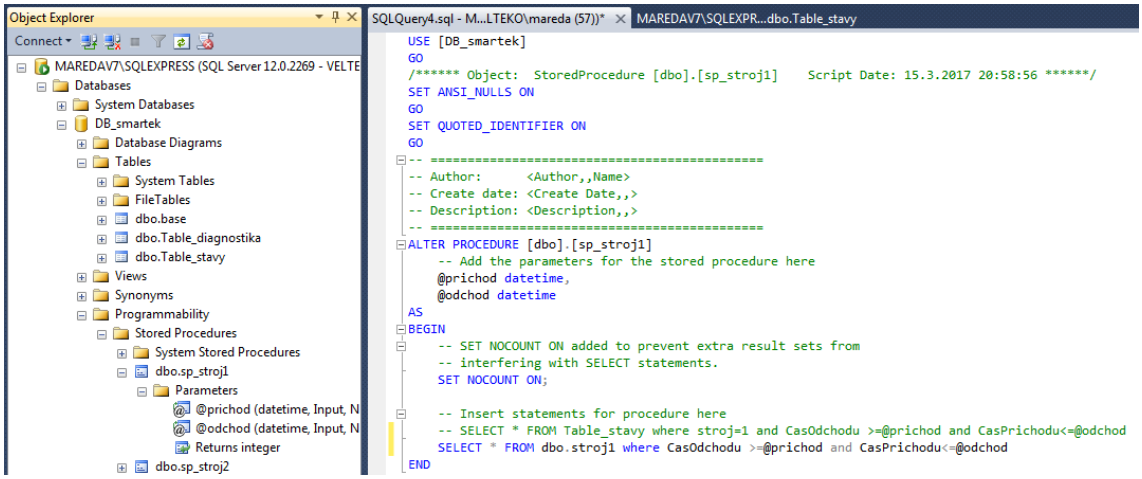
Abychom měli časový přehled o všech strojích a zároveň jsme nemuseli tvořit *n* tabulek, vytvoříme jen jednu a k ní tzv. *Views*. *Views* slouží k náhledu do tabulky s podmínkou. V mé aplikaci se ukládají všechna data do tabulky *Table_stavy* a dělám si z ní „výtahy dat“ na základě čísla stroje. Takže kolik strojů v reportu, tolik

Views. Podmínku lze vidět na Obr. 70 v řádku *stroj* (=1 znamená, že vytáhne pouze řádky, kde je v kolonce *stroj* číslo 1).



Obr. 70 Views tabulky

Výsledný výtažek dat do reportu je závislý na vstupních parametrech *Příchod* a *Odchod* (začátek a konec reportu). *Views* nepodporují parametry, k tomu slouží *Stored Procedures*. V nich si napíšeme podmínky na základě parametrů. *Stored procedures* lze vidět na obr. 71.



Obr. 71 Stored Procedures

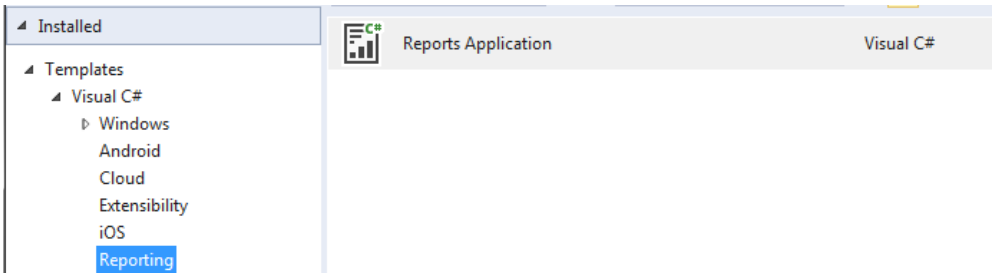
V obrázku 71 je dále vidět, že se odkazujeme na *Views* (dbo.stroj1, viz. obr. 70). *Stored Procedures* lze programovat i přímo na danou tabulku (lze vidět v komentáři nad podmínkou). *Views* jsem si tedy vytvořil jen pro přehlednost o daném stroji.

Databáze je vytvořena. Nyní naprogramujeme aplikaci, která jí bude plnit a vyčítat z ní data.

2.3.3 Aplikace

2.3.3.1 Založení nového projektu

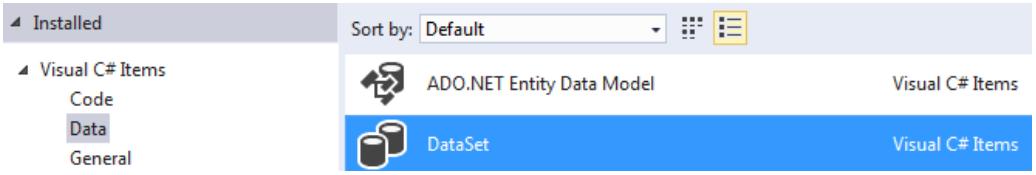
Abychom mohli naprogramovat aplikaci, je potřeba založit nový projekt. Ve Visual Studiu si tedy vybereme *Reports Application*, viz obr. 72. Po založení nového projektu na nás vyskočí tzv. Wizard. Wizard je určen k rychlému vytvoření základní kostry reportu. My průvodce zavřeme tlačítkem Cancel a vytvoříme si celý report ručně.



Obr. 72 Nový report

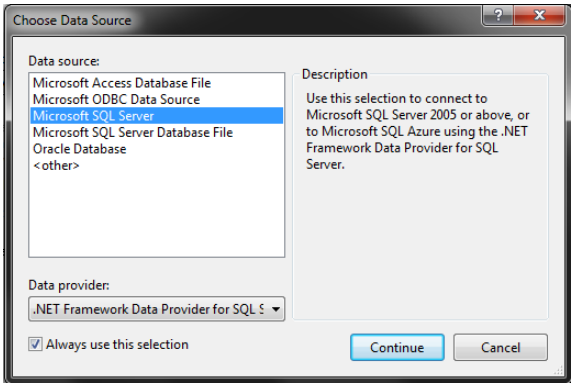
2.3.3.2 Propojení projektu s databází

Abychom měli co zobrazovat, je potřeba definovat zdroj zobrazovaných dat. Pomocí *Add -> New Item*, které nalezneme v hlavním menu projektu, přidáme do projektu DataSet, který se nám bude plnit daty z databáze, viz obr. 73.



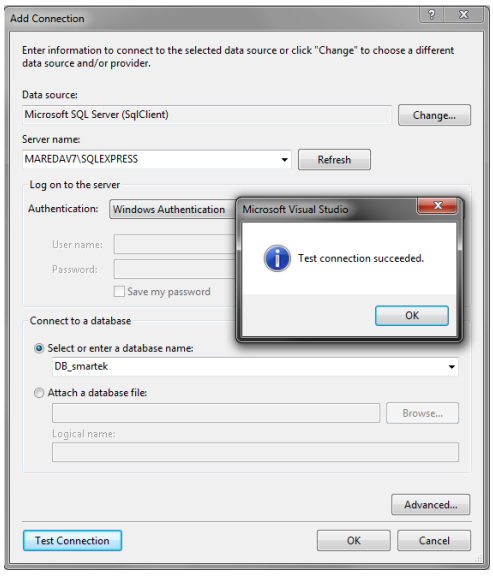
Obr. 73 DataSet

Po vytvoření se nám automaticky otevře okno s obsahem DataSetu. Pokud se nám okno automaticky neotevře, poklikáme na *NášDataSet.xsd* (v hlavním menu). Zatím je DataSet prázdný. Musíme ho propojit s naší databází. Klikneme tedy pravým tlačítkem myši a objeví se volba *Add -> TableAdapter*. Volbu potvrdíme a v dalším okně si vytvoříme cestu k databázi (klikneme na *New Connection*) a zvolíme, stejně jako na obr. 74, *Microsoft SQL Server*.



Obr. 74 Data Source

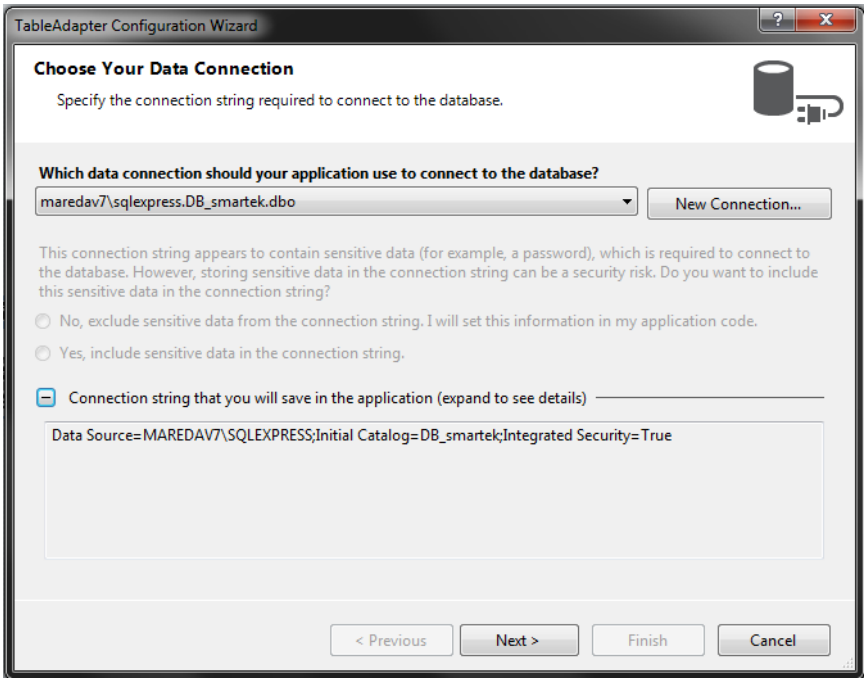
Po kliknutí na tlačítko *Continue* vyskočí okno *Add Connection.* V tomto okně zvolíme jméno serveru a příslušnou databázi. Pro ověření spojení můžeme, stejně jako na obr. 75, zvolit *Test Connection*.



Obr. 75 Připojení k databázi

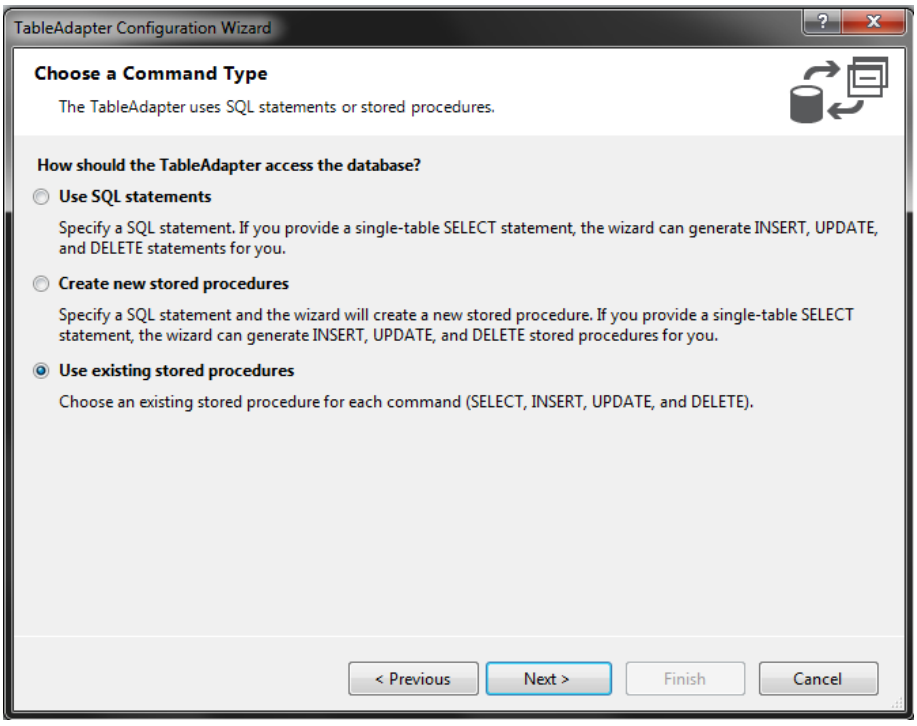
Potvrdíme pomocí *OK* a nalezneme se v okně *TableAdapter Configuration Wizard*.

Toto okno slouží k náhledu a případnému uložení *Connection stringu*. Ten se nám bude hodit v aplikaci pro připojení k databázi.



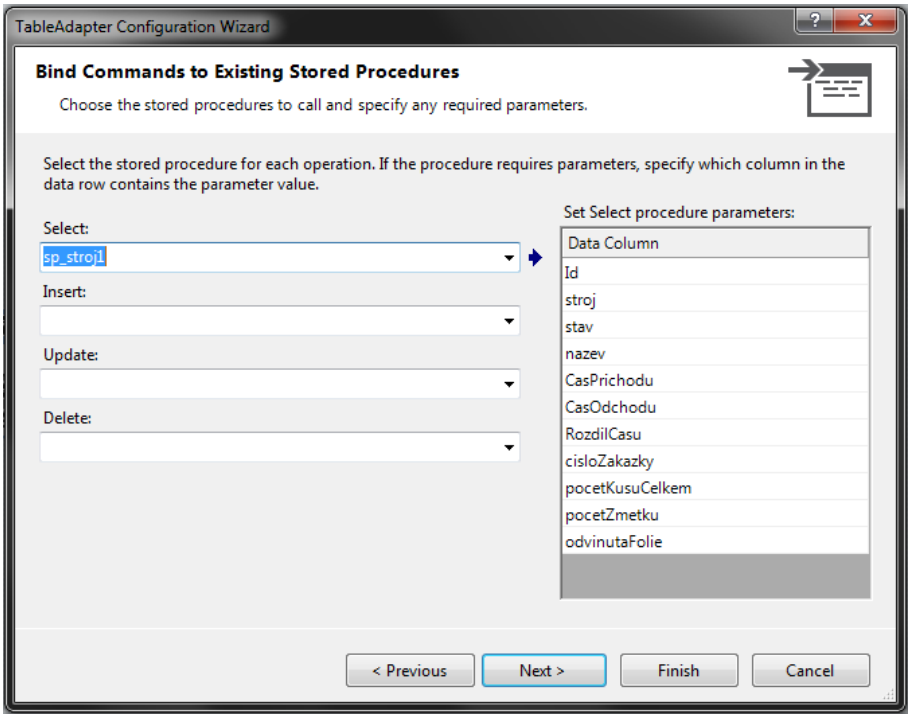
Obr. 76 Connection string

Poté se nás program zeptá, jakým způsobem budeme vyčítat data z databáze. Vybereme *Stored Procedures*, viz obr. 77.



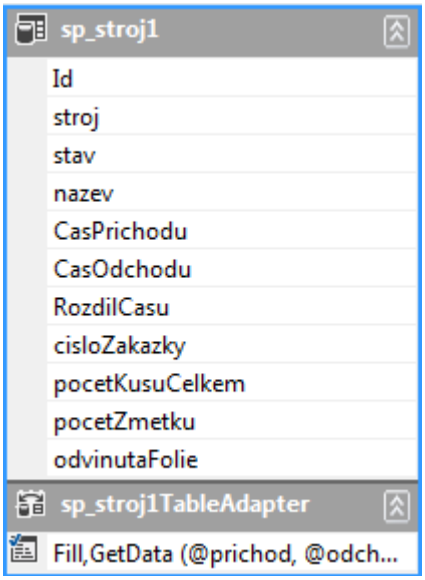
Obr. 77 Stored procedures v Table adapteru

V dalším okně zvolíme naší proceduru (*sp_stroj1* na obr. 78) a již můžeme vidět proměnné z databáze.



Obr. 78 Vybraná procedura v Table adapteru

Následně zvolíme *Finish* a máme vytvořené spojení s databází. V DataSetu na obr. 79 vidíme propojený obsah.

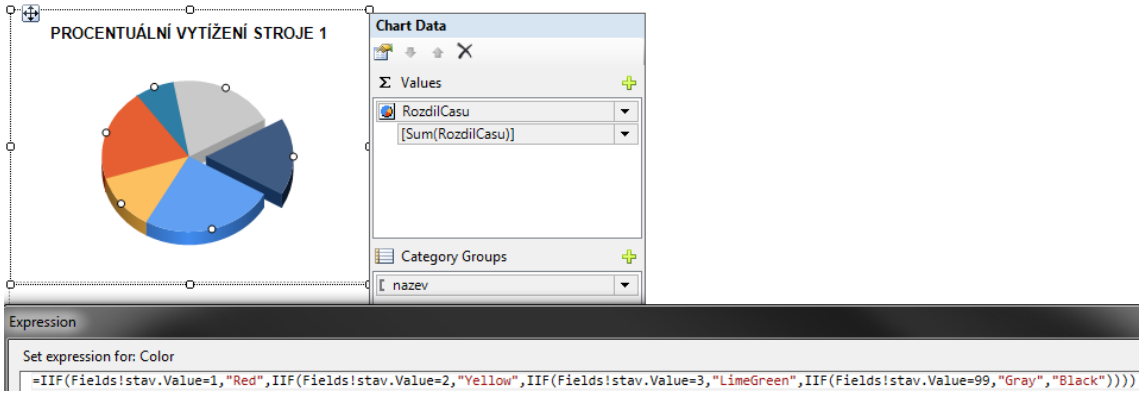


Obr. 79 Propojení reportu s databází

Stejně tak vytvoříme datasety pro ostatní stroje.

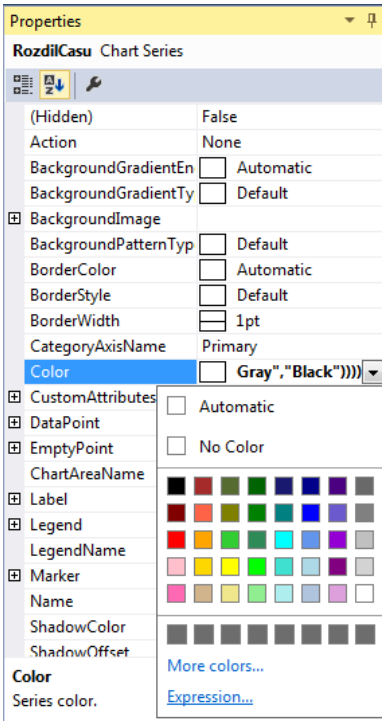
2.3.3.3 ReportViewer – Reprezentace dat

Reprezentace dat se upravuje v souboru *Report.rdl*. Do reportu můžeme vkládat textboxy, tabulky, grafy, obrázky, subreporty, atd.. Objekty lze vkládat pomocí Toolboxu, nebo *pravé tlačítko myši -> Insert*. Na obr. 80 je znázorněno, jak jednoduché je nastavit graf, ve kterém uvidíme, jak dlouho byl stroj v daném stavu. Můžeme vidět součet časů v daných stavech (*Sum(RozdilCasu)* v ∑ *Values*) a rozdělení do kategorií dle názvů stavů (*Category Groups*). Pokud nechceme nechat automatické rozložení barev stavů, použijeme *Expression* ve vlastnostech barev. Na základě daného stavu určíme barvu v grafu (například pomocí několika podmínek IF – v reportu se IF značí jako *IIF*).



Obr. 80 Nastavení grafu

Obrázek 81 značí, jak se do *Expression* barev proklikáme. Vlastností a možností jejich úprav je mnoho. Jejich popis by vystačil na samostatnou práci, takže si ukážeme nejdůležitější z použitých metod.



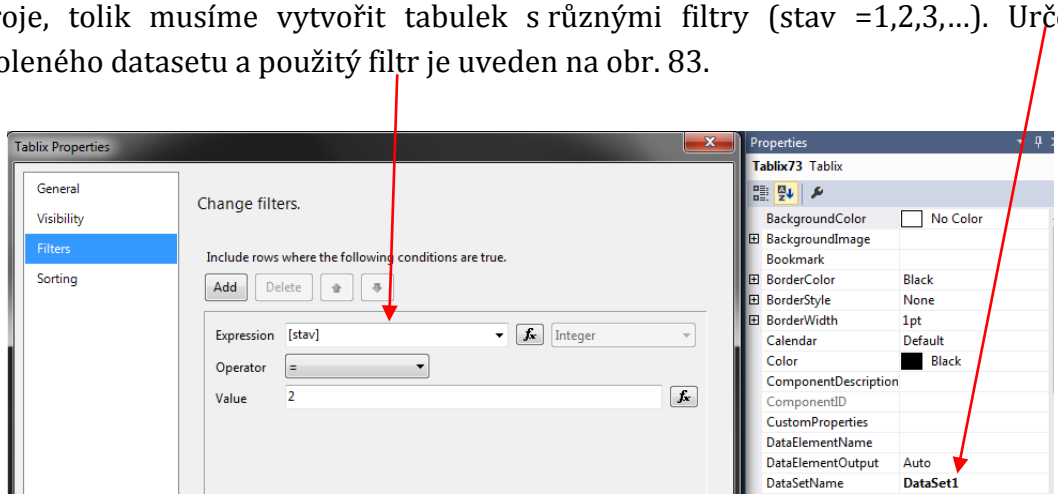
Obr. 81 Programování vlastností

Pokud chceme používat parametry z databáze, musíme si je vytvořit v záložce *Report Data*. V případě, že *Report Data* nevidíme, pomůže kombinace kláves *Ctrl+Alt+D*. Ve složce *Parameters* zvolíme *Add Parameter*, určíme název parametru a datový typ. Pokud se vytváří tabulka jen pro vedlejší výpočty a nechceme ji celou zobrazovat (s každou hodnotou nám narůstá počet řádků), tak si tabulku označíme jako neviditelnou a program ví, že nemá svými rostoucími rozměry ovlivňovat velikost výsledného vzhledu reportu. Na Obr. 82 je vidět pomocná tabulka. Řádek, který je označen rovnítkem, slouží k nárustu hodnot (co hodnota, to nový řádek). Dokážete si představit, jak by vypadal výsledný report, kdyby muselo být v reportu například 100 řádků, které slouží jen pro pomocný výpočet. Na Obr. 82 je tabulka, ze které vypočítáme celkový čas (suma dílčích časů), jeho průměr, procentuální podíl, minimum a maximum. Všechny výpočty se týkají jednoho stavu.

		Rozdil Casu
☰		[RozdilCasu]
	Celkem	«Expr»
	Prumer	«Expr»
	Procent	«Expr»
	Min	«Expr»
	Max	«Expr»

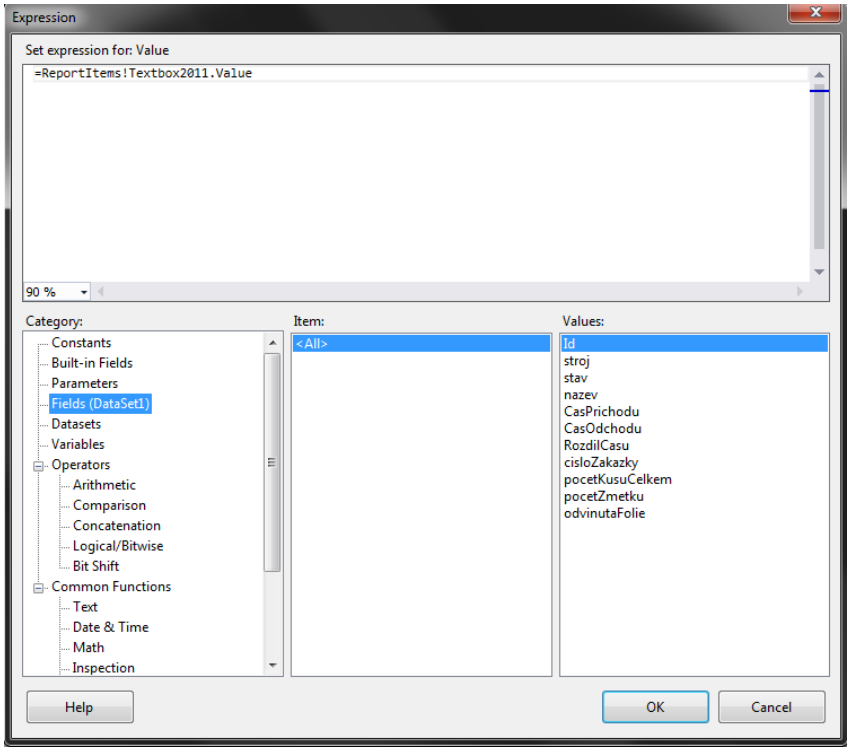
Obr. 82 Pomocná tabulka

Abychom mohli určit jen jeden stav, tak musíme tabulku označit (stejně jako graf) datasetem (dataset prvního stroje = DataSet1) a vytvořit filtr. Kolik stavů daného stroje, tolik musíme vytvořit tabulek s různými filtry (stav =1,2,3,…). Určení zvoleného datasetu a použitý filtr je uveden na obr. 83.



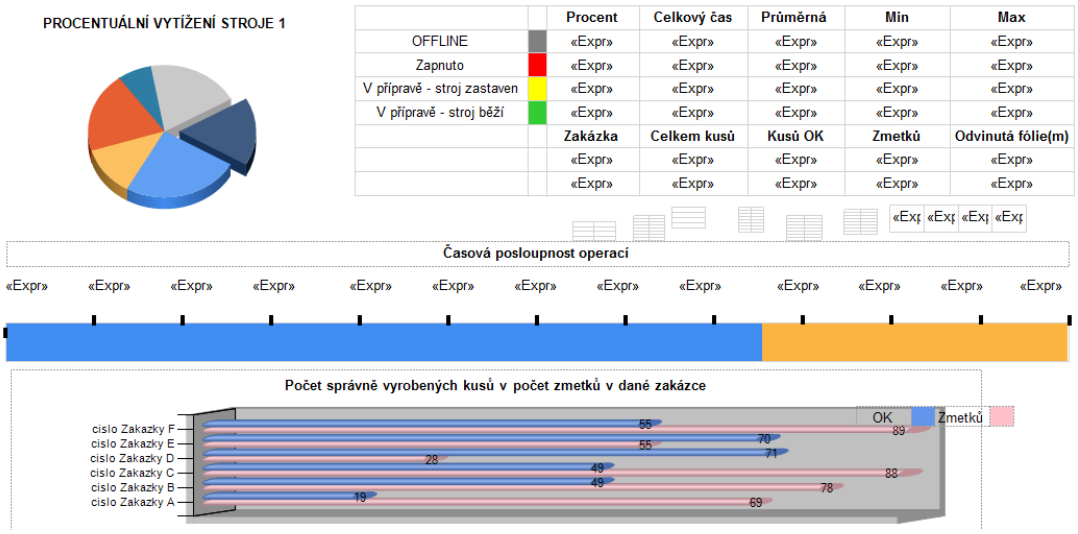
Obr. 83 Filtr tabulky

ReportItems! Používáme k odvolání se na prvek v reportu jako například hodnotu v Textboxu jak je vidět na Obr. 84. Já jsem tuto možnost použil kvůli vedlejším výpočtům v jedné tabulce a její výsledek jsem použil v tabulce jiné pro výpočty další, nebo pro zobrazení.



Obr. 84 Možnosti Expression

Po vytvoření výpočtů, tabulek a grafů může kousek reportu vypadat například jako Obr. 85.



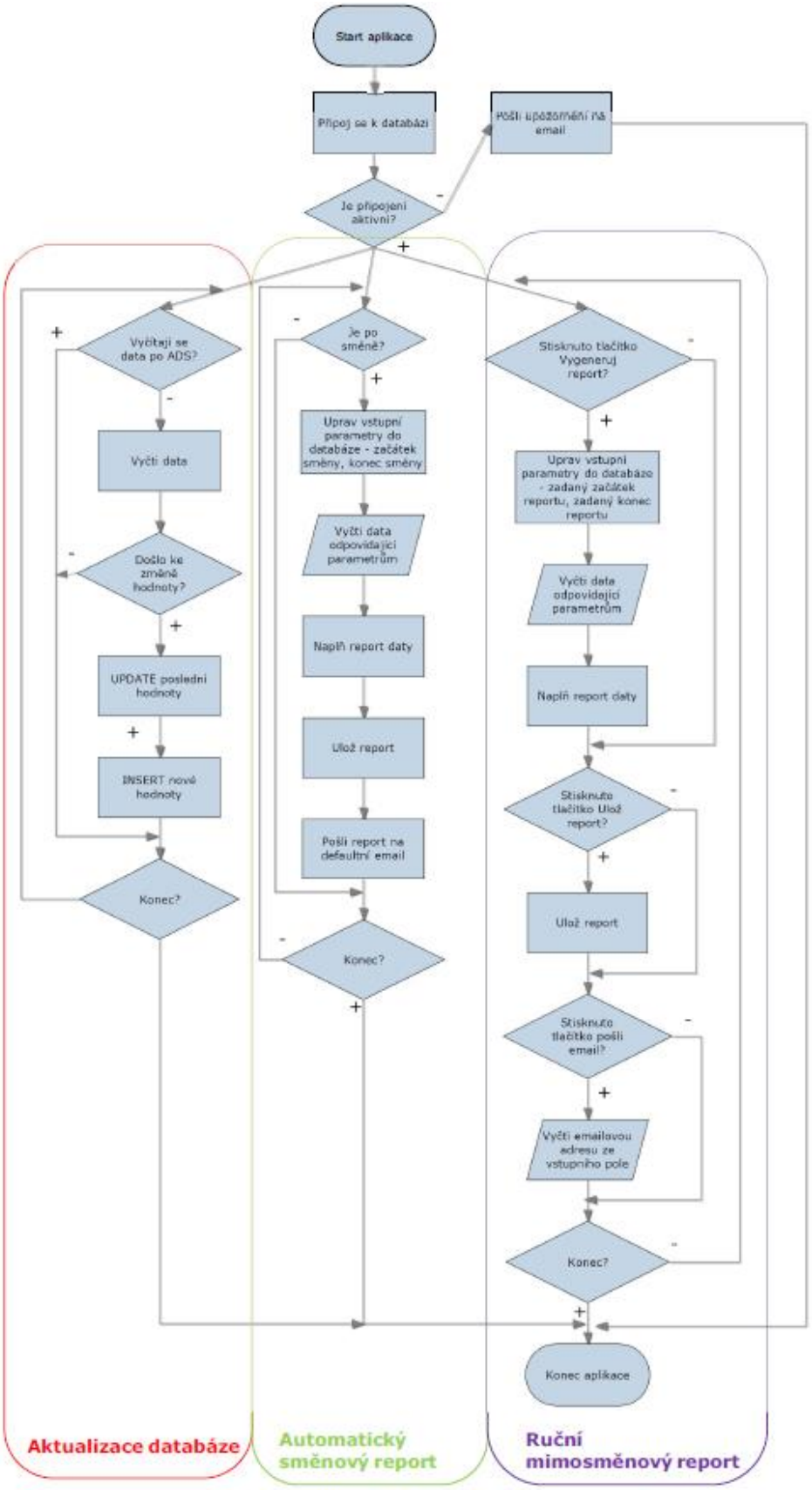
Obr. 85 Část tvorby reportu

Poslední věc, kterou musíme udělat je propojení jednoho souboru *report.rdlc* s *reportviewerem* v aplikaci. Na hlavní obrazovce se zvolí *Choose Report -> report.rdlc.*

2.3.3.4 Vnitřní algoritmus

2.3.3.4.1 Popis funkčnosti aplikace

Po zapnutí aplikace se program pokusí navázat spojení s databází. Pokud připojení nelze navázat, pošle se upozorňující email obsluze a program se ukončí. V případě otevření komunikace k databázi se program cyklicky dotazuje strojů na jejich stav (k němu si přiřadí název stavu) a číslo zakázky. Pokud se jedna z těchto hodnot změní, vyčte se počet vyrobených kusů a počet zmetků do změny (z rozdílu se vypočítá počet správně vyrobených kusů). Dále vznikne pokyn k vynulování těchto hodnot, abychom měli stále aktuální hodnoty pro ručně vytvořený report. Data se zapíší do databáze. Program automaticky generuje report po skončení směny. Po vygenerování reportu se uloží jako *PDF* a odešle na defaultní emailovou adresu. Možností je i ručně generovaný report. Více nám řekne vývojový diagram na obr. 86.



Obr. 86 Vývojový diagram reportingu

2.3.3.4.2 Ukázka zdrojových kódů

Připojení k databázi

Pro připojení k databázi musíme vložit do projektu *„using System.Data.SqlClient;“*. Na obr. 87 je znázorněno otevření komunikace s naší databází a testovací vytažení dat včetně nepodařeného spojení.

```
public void connDB()
{
    try
    {
        con = new SqlConnection(@"Data Source=MAREDAV7\SQLEXPRESS;Initial Catalog=DB_smartek;Integrated Security=True");
        con.Open();
        cmd = new SqlCommand("SELECT * FROM Table_stavy");
        cmd.Connection = con;
        da = new SqlDataAdapter(cmd);
        dt = new DataTable();
        da.Fill(dt);
        cmd = null;
        da = null;
    }
    catch
    {
        System.Windows.Forms.MessageBox.Show("databaze nepripojena"); // pro testovani
        posliEmailNepripojenaDatabazeNenahrano();
        Application.Exit();
    }
}
```

Obr. 87 Připojení k databázi

Aktualizace dat v databázi

Pokud dojde ke změně hodnot, nejprve se aktualizují data předchozí změny. Aktualizuje se čas odchodu události (daný stav, nebo zakázka byl/a ukončen/a), rozdíl času (jak dlouho byl stroj ve stavu a zakázce {pomocí sql příkazu *DateDiff*}), počet vyrobených kusů (od posledního uložení), počet zmetků (od posledního uložení) a délka odvinuté fólie (také od posledního uložení). Nebudu zde uvádět celý kód aktualizace dat. Vždy je použito klíčové slovo *UPDATE*, použitá tabulka, použitá proměnná a její hodnota. Jelikož jsou aktualizovaná data stejná pro všechny stroje, je kód v cyklu *FOR*, jehož hodnotu udává *n*. Aktualizace hodnot pomocí jazyka C# je vidět na obr. 88.

```
cmd = new SqlCommand("UPDATE Table_stavy SET [CasOdchodu] = SYSDATETIME() WHERE [Id]=(SELECT MAX(IIF([stroj]=" + n + ",[Id],0)) FROM Table_stavy)");
cmd.Connection = con;
da = new SqlDataAdapter(cmd);
da.Fill(dt);
```

Obr. 88 Aktualizace dat v databázi

Vložení nových dat do databáze

Po aktualizaci předchozích dat vložíme novou změnu stavu, nebo čísla zakázky, viz. obr. 89.

```
// po kazdem zapisu zmeny stavu (nazvu stavu) nebo cisla zakazky se nuluje pocet kusu, pocet zmetku a odvinuta folie v PLC stroje
//- aby se spravne vygeneroval rucni report
cmd = new SqlCommand("INSERT INTO Table_stavy([nazev],[stav],[CasPrichodu],[stroj],[cisloZakazky]) VALUES ('" + nazev[n] + "','"
    + stav[n] + ", SYSDATETIME()," + n + ","+ cislo_zakazky[n] +")");
cmd.Connection = con;
da = new SqlDataAdapter(cmd);
da.Fill(dt);
```

Obr. 89 Vložení nového řádku do databáze

Naplnění dat v datasetu reportu

Na obrázku níže je nastíněn způsob naplnění dat v reportu parametry a daty z databáze. Z důvodu velikosti je obrázek upraven o odstranění plnění ostatních *TableAdpaptérů*.

```
public void vygenerujReport(object rozsah)
{
    ReportParameter prichod = new ReportParameter("prichod");
    ReportParameter odchod = new ReportParameter("odchod");

    string pr, od;
    pr = ((rozsah as string[])[0]);
    od = ((rozsah as string[])[1]);
    prichod.Values.Add(pr);
    odchod.Values.Add(od);

    reportViewer1.LocalReport.SetParameters(prichod);
    reportViewer1.LocalReport.SetParameters(odchod);

    this.sp_stroj1TableAdapter.Fill(this.DataSet1.sp_stroj1, Convert.ToDateTime(pr), Convert.ToDateTime(od));
    reportViewer1.RefreshReport();
}
```

Obr. 90 Generování reportu

Ukládání reportu do PDF

Součástí SSRS je sice možnost uložení do PDF, ale já jsem ji musel naprogramovat znovu ručně z důvodu automatického ukládání reportu. Vnitřní kód *ulozPDF()*, který je na obr. 91, není má práce, ale jde o oficiální veřejně dostupný algoritmus podpory SSRS ze stránek Microsoftu.

```
public void ulozPDF()
{
    try
    {
        string _path = Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData);
        _path += @"\output.pdf";
        Warning[] warnings;
        string[] streamids;
        string mimeType;
        string encoding;
        string extension;

        byte[] bytes = reportViewer1.LocalReport.Render(
            "PDF", null, out mimeType, out encoding,
             out extension,
            out streamids, out warnings);

        FileStream fs = new FileStream(_path,
           FileMode.Create);
        fs.Write(bytes, 0, bytes.Length);
        fs.Close();
        fs = null;
    }
    catch { }
}
```

Obr. 91 Uložení PDF

Odesílání reportu na email

Odesílání emailů je možné díky *„using System.Net.Mail;“*. Vytvoří se nová zpráva, přidá odesilatel a příjemce. Předmět je vyplněn příkazem *msg.Subject* a tělo zprávy *msg.Body*. Příloha se přidává pomocí *msg.Attachments.Add(data)*. Zbytek kódu se týká nastavení emailu (klient, port a povolení ssl) a pokynu k odeslání.

```
public void posliEmail()
{
    try
    {
        string _path = Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData);
        _path += @"\output.pdf";

        MailMessage msg = new MailMessage();
        msg.From = new MailAddress("report@reportingsystem.cz");
        msg.To.Add("zakaznik@firmaz.cz");
        msg.Subject = "Směnový report";
        msg.Body = "Dobrý den, \n\n\t posíláme report za uplynulou směnu.\n\n\t S pozdravem '
        Attachment data = new Attachment(_path);
        msg.Attachments.Add(data);
        SmtpClient sc = new SmtpClient("posta");
        sc.Port = 25;

        sc.EnableSsl = false;

        sc.Send(msg);
        msg.From = null;
        msg = null;
        data = null;
        sc = null;
    }
    catch (Exception ex)
    {
    }
}
```

Obr. 92 odeslání emailu

Na obrázcích 90, 91 a 92 jsou uvedeny metody, které se volají ručně tlačítky. Metody *vygenerujReport_aut, ulozPDF_aut* a *posliEmail_aut*, které zapříčiňují automaticky generovaný report jsou velmi podobné, tudíž zde uvedeny nejsou.

Automaticky generovaný report

Pomocí časovače se cyklicky volá metoda *Automatika()*. Pokud metoda zjistí, že je konec směny, upraví vstupní parametry (příchod reportu a odchod reportu). Dle směny se časy upraví na 6:00 až 14:00, nebo 14:00 až 22:00, nebo 22:00 až 6:00. Po úpravě časů se vygeneruje report, převede se na formát PDF a odešle na zvolený email. Zdrojový kód je uveden na obr. 93.

```
public void Automatika()
{
    try
    {
        if (DateTime.Now.Hour == 6 || DateTime.Now.Hour == 14 || DateTime.Now.Hour == 22)
        {
            if (DateTime.Now.Hour == 6)
            {
                d1 = new TimeSpan(6, 0, 0);
            }
            else if (DateTime.Now.Hour == 14)
            {
                d1 = new TimeSpan(14, 0, 0);
            }
            else
            {
                d1 = new TimeSpan(22, 0, 0);
            }
            datum1 = DateTime.Now;
            datum1 = datum1.Date + d1;
            po = datum1.ToString();
            datum2 = datum1.AddHours(-8);
            od = datum2.ToString();

            vygenerujReport_aut(new string[] { od, po });
            ulozPDF_aut();
            posliEmail_aut();
        }
    }
    catch { }
}
```

Obr. 93 automaticky generovaný report

2.3.3.5 Výsledná aplikace

Tuto kapitolu tvoří popis výsledné aplikace a z ní vygenerovaný report.

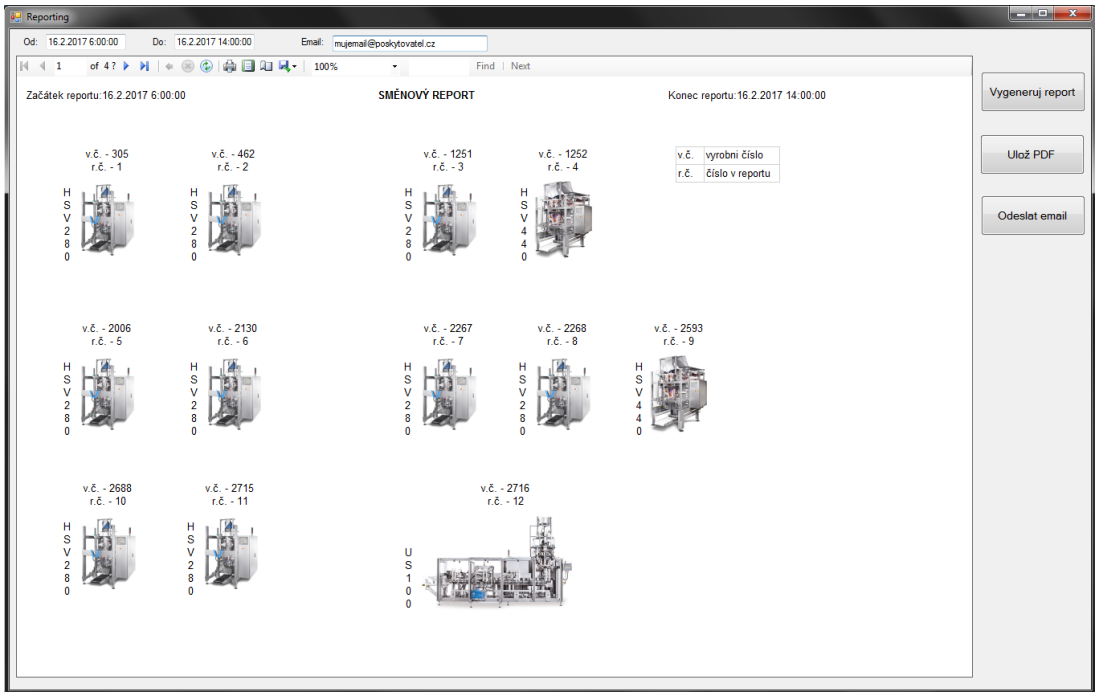
2.3.3.5.1 Výsledná aplikace

Aplikaci je možno vidět na obr. 94. Na tomto obrázku vidíme první, tzv. přehledovou stranu reportu. K čemu tento náhled slouží je uvedeno v následující kapitole. Jak bylo uvedeno v předchozím textu, aplikace umožňuje kromě automatického zasílání směnového reportu také report ruční. Pokud chceme vygenerovat report ručně, postupujeme dle následujícího textu. V kolonkách si zadáme čas začátku a konce reportu. Dále stiskneme tlačítko *Vygeneruj report*. Nyní máme na výběr dvě možnosti.

Možností číslo jedna je prohlížení reportu přímo v aplikaci pomocí šipek doleva a doprava, viz obr. 94.

Druhou a mnohem příjemnější možností je uložení reportu do PDF.

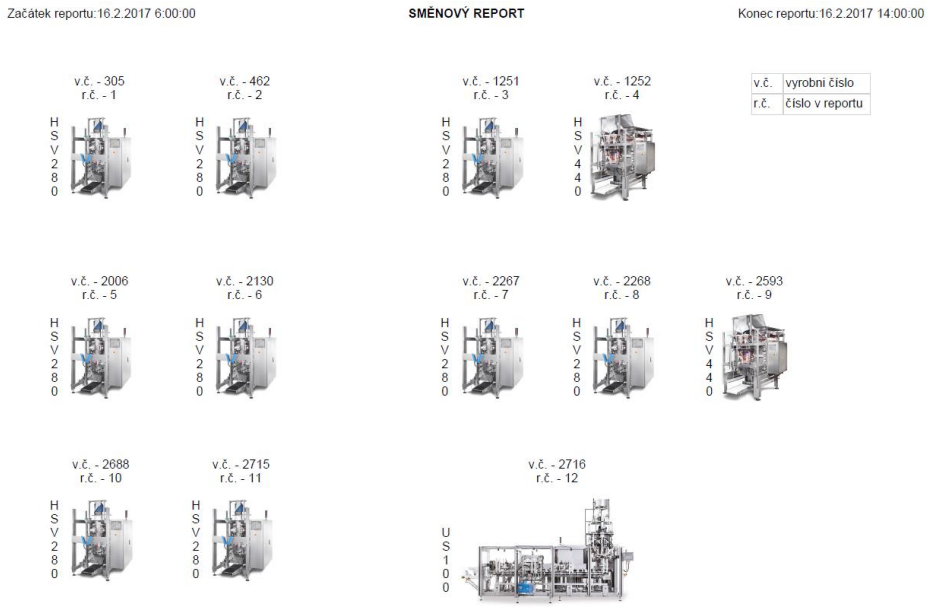
Poslední funkcí aplikace je možnost odeslání reportu na email. K tomu nám stačí zadat emailovou adresu do kolonky *Email* a stisknout tlačítko *Odeslat email*.



Obr. 94 Výsledná aplikace

2.3.3.5.2 Vygenerovaný report

Na první straně (obr. 95) je rozmístění strojů ve výrobě s přiřazenými výrobními a report čísly. Tato strana slouží k rychlejší orientaci v reportu. Kdyby byla data přiřazena pouze výrobnímu číslu stroje, bylo by mnohem náročnější na čas najít stránku, na které se daný stroj nachází. Takto se stačí podívat, jaké report číslo má daný stroj a vím, na které straně se informace o stroji nachází. Například stroj s report číslem 1 je zobrazen v reportu na straně č. 3. Jedná se až o třetí stranu, z důvodu rychlého přehledu, který je popsán v následujícím textu.

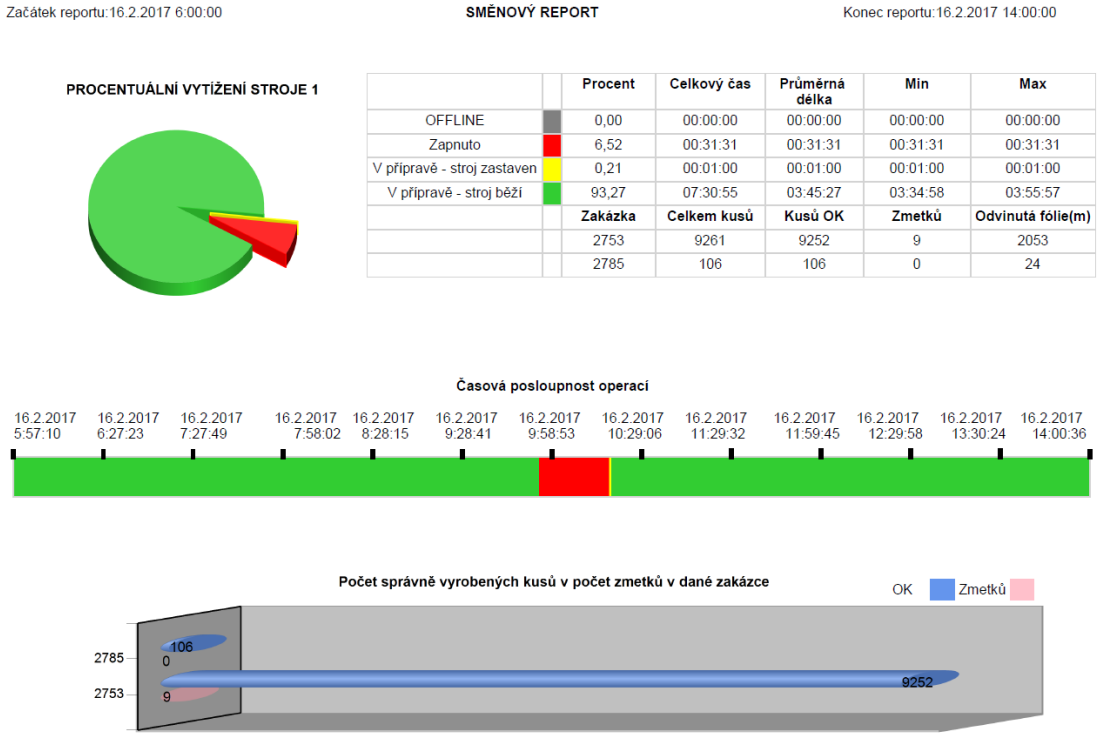


Obr. 95 Report - přehled rozmístění strojů

Druhá strana (obr. 96) slouží k prvnímu nahlédnutí, tzv. rychlému přehledu. Tento přehled jsem zařadil, aby majitel reportu nemusel pokaždé prohlížet všechny strany, ale mohl se zaměřit jen na ty „zajímavé“. Rychlý přehled ukazuje poměr stavů, ve kterých se stroj nacházel a zmenšený Ganttův diagram. Jak je vidět, data jsou posbírána z doby, kdy byl reporting nasazen na prvním stroji.

Obr. 96 Report - rychlý přehled

Následující obrázek ukazuje podrobnější informace o vytížení stroje za danou směnu. Obsahuje koláčový graf vyjadřující procentuální vytížení stroje, které je svázáno s tabulkou. V tabulce je výpis stavů s celkovým poměrem a časem stavů, dále průměrnou, minimální a maximální dobu v určitém stavu. Tabulka také obsahuje celkový počet vyrobených kusů za směnu, zmetkovitost, počet kusů prošlých kontrolou a odvinutou fólii. Tyto informace jsou navíc rozděleny do zakázek. Pod tabulkou je vidět časová posloupnost přicházejících stavů (výše zmíněný Ganttův diagram). Ve spodní části je poměr správných kusů a zmetků v zakázce.



		Procent	Celkový čas	Průměrná délka	Min	Max
OFFLINE		0,00	00:00:00	00:00:00	00:00:00	00:00:00
Zapnuto		6,52	00:31:31	00:31:31	00:31:31	00:31:31
V přípravě - stroj zastaven		0,21	00:01:00	00:01:00	00:01:00	00:01:00
V přípravě - stroj běží		93,27	07:30:55	03:45:27	03:34:58	03:55:57
		Zakázka	**Celkem kusů**	**Kusů OK**	**Zmetků**	**Odvinutá fólie(m)**
		2753	9261	9252	9	2053
		2785	106	106	0	24

Obr. 97 Report - podrobný přehled

3 ZÁVĚR

Práce se zabývá problematikou svařování termoplastů u balicích strojů. Hlavním cílem bylo seznámení s ultrazvukovou metodou a praktické ověření jejích výhod. Navržení funkčního bloku pro řízení svařovacího cyklu je realizováno v ŘS od firmy Beckhoff. Firma Beckhoff je známá mimo jiné díky vlastní sběrnici EtherCAT, přes kterou je zajištěna komunikace s ultrazvukovým generátorem. Použitým protokolem je CoE. Nastavení generátoru bylo realizováno přes SDO, takže bylo potřeba ručně vytvořit výměnu dat mezi generátorem a PLC. Některá data v generátoru byla vícevýznamová, takže bylo zapotřebí informace rozkódovat a navrhnout vhodný interface interpretace dat pro přívětivou implementaci bloku. Kromě možnosti balení problematického produktu došlo také ke snížení výšky balícího stroje. Díky tomu je možné uvažovat o zástavbě další automatizace (například dopravníky zboží do násypky), či o použití výkonnějších a tedy i rozměrnějších dávkovačů zboží.

Dále bylo zapotřebí ověřit možnosti řízení svařovacího cyklu s ohledem na výkon stroje.

Dalším bodem bylo navržení grafického vzhledu a doprogramování do současné vizualizace. K vizualizaci se nepoužívá software třetích stran, jako je např. *WinCC*, či *ZENON*. Proto bylo zapotřebí se více do hloubky seznámit s programovacím jazykem C#, ve kterém je vizualizace tvořena.

Posledním úkolem byla tvorba reportingu, který poskytuje údaje o výrobě. Způsob realizace reportingu byl čistě v mých rukou, proto jsem vybral SSRS, který je programován také v jazyce C# a nebylo tedy potřeba dokupovat další drahý software.

Jako možnost dalšího pokračování v práci vidím ve vynoření a rozšíření komunikačního bloku. Rozšíření by spočívalo v přidání vstupní proměnné, která by rozlišovala druh komunikace. Díky tomu by se pomocí jednoduché podmínky IF nebo CASE snadno realizovala implementace dalších standardních funkčních bloků, které zajišťují komunikaci jiným způsobem, než *FB_EcCoESdoWrite/ FB_EcCoESdoRead*. Dále přidáním vstupní struktury se seznamem komunikovaných adres (v současné době je seznam integrován v mém bloku), které by se svázali s již použitým polem aobPovoleniVycitani. Vznikl by tak univerzální blok zajišťující autonomní komunikaci s různými protistranami.

Pokračovat lze také report systémem. Zde jsou možnosti, řekl bych skoro neomezené. Už jen z pohledu různorodosti dat z výroby a jejich chtěné reprezentace. Další možností je integrace reportu do vizualizace.

Veškeré teoretické poznatky byly aplikovány na vývojové verzi stroje, který byl již předveden na výstavě Interpack 2017 v Düsseldorfu.

Literatura

[1] BACHMANN, Jiří, Jaroslav BERÁNEK, Michal DANĚČEK a kol. *Plasty pro rozvod médií a svařované konstrukce*. Vyd. 1. Praha: GAS spol. s. r. o., 2001. 530 s. ISBN 80-86176- 97-5.

[2] LOYDA, Miloslav, Vlastimil ŠPONER, Ladislav ONDRÁČEK a kol. *Svařování termoplastů*. Vyd. 1. Praha: UNO spol. s. r. o., 2001. 496 s. ISBN 80–238–6603–6.

[3] *Ultrazvukové svařování - 41* [online]. 2009 [cit. 2017-03-24]. Dostupné z: http://homen.vsb.cz/~hla80/2009Svarovani/2-06-41.pdf

[4] *EtherCAT Technology Group* [online]. [cit. 2017-03-24]. Dostupné z: https://www.ethercat.org/en/technology.html

[5] *Beckhoff Information System - English* [online]. [cit. 2017-03-24]. Dostupné z: https://www.beckhoff.com/english.asp?ethercat/aufbau2.htm

[6] *Beckhoff Information System - English: EtherCAT – Ultra high-speed for automation* [online]. [cit. 2017-03-24]. Dostupné z: https://www.beckhoff.com/english.asp?ethercat/highlights.htm?id=20433492043386

[7] *Automa: CANopen – vyšší komunikační protokol pro vestavné sítě* [online]. [cit. 2017-03-24]. Dostupné z: http://automa.cz/cz/casopis-clanky/canopen-vyssi-komunikacni-protokol-pro-vestavne-site-2004_04_32279_854

[8] *BECKHOFF PC Control: EtherCAT P combines ultra-fast communication and power supply in a single cable* [online]. [cit. 2017-03-24]. Dostupné z: https://www.pc-control.net/pdf/012016/technology/pcc_0116_ethercatp_e.pdf

[9] *Automa: TwinCAT 3 – eXtended Automation Technology (XAT)* [online]. [cit. 2017-03-24]. Dostupné z: http://automa.cz/Aton/FileRepository/pdf_articles/53801.pdf

[10] *Beckhoff Information System - English: TwinCAT 3 Philosophy* [online]. [cit. 2017-03-24]. Dostupné z: https://infosys.beckhoff.com/english.php?content=./content/1033/tc3_overview/html/tc3_overview_philosophy.htm&id=

[11] *Beckhoff Information System - English: ADS Introduction* [online]. [cit. 2017-03-24]. Dostupné z: https://infosys.beckhoff.com/english.php?content=./content/1033/tcadscommon/html/tcadscommon_intro.htm&id=

[12] *Beckhoff Information System - English: ADS device identification* [online]. [cit. 2017-03-24]. Dostupné z: https://infosys.beckhoff.com/english.php?content=./content/1033/tcadscommon/html/tcadscommon_intro.htm&id=

[13] *Beckhoff Information System - English: ADS Protocol* [online]. [cit. 2017-03-24]. Dostupné z: https://infosys.beckhoff.com/english.php?content=./content/1033/bc9000/html/bt_bx9000adsprotocol.htm&id=13128393460564065 63

[14] *Beckhoff Information System - English: EtherCAT State Machine* [online]. [cit. 2017-03-24]. Dostupné z: https://infosys.beckhoff.com/english.php?content=./content/1033/bc9000/html/bt_ethernet%20ads%20potocols.htm&id=

[15] *Dukane iQ EtherCAT User Manual*. St. Charles, Illinois 60174, 2900 Dukane Drive.

[16] *MM Průmyslové spektrum: Spojování a řezání ultrazvukem* [online]. 2013 [cit. 2017-04-21]. Dostupné z: http://www.mmspektrum.com/clanek/spojovani-a-rezani-ultrazvukem.html

Seznam symbolů, veličin a zkratek

FEKT - Fakulta elektrotechniky a komunikačních technologií
VUT - Vysoké učení technické v Brně
PLC - Programmable Logic Controller
EtherCAT - Ethernet for Control Automation Technology
UDP – User Datagram Protocol
TCP – Transmission Control Protocol
EoE - Ethernet over EtherCAT
FoE - File Acces over EtherCAT
SoE - Servo Drive Profile according to IEC 61491-7-204 over EtherCAT
CoE - CAN over EtherCAT
PDO - Process Data Objects
SDO - Service Data Objects
NMO - Network Management Objects
TwinCAT - The Windows Control Automation Technology
XAA - eXtended Automation Architecture
XAE - eXtended Automation Engineering
XAR - eXtended Automation Runtime
ADS - Automation Device Specification
FBD - Function Block Diagram
LD - Ladder Diagram
SFC - Sequential Function Chart
CFC - Continuus Function Chart
SC - UML Statechart
IL - Instruction List
ST - Structured Text
FB - Function Block
FC - Function
SSRS - Microsoft SQL Server Reporting Services

Seznam příloh

4 PŘÍLOHY

stPdoIn

System_Error_Status: Je různý od nuly v případě výskytu chyby. Číselná hodnota je vyjádřena kombinací bitů viz tabulka 9.

Initiate_Error_Status: Chyba při inicializaci generátoru. Hodnota vyjadřuje číslo chyby. Pokud u proměnných není uvedena tabulka, nejedná se o bitově významovou proměnnou.

Process_Error_Status: Chyba při svařovacím procesu.

Generator_State: Aktuální stav generátoru. Možné stavy jsou uvedeny v tabulce 10.

Real_Time_Amplitude: Aktuální amplituda. Hodnota je udávána v %.

Real_Time_Frequency: Aktuální frekvence. Fyzikální veličinou je [Hz].

Real_Time_Power: Aktuální výkon. Jednotkou je Watt [W].

System_Status: Podrobnosti o stavu a aktivitě systému. Bližší informace v tabulce 11.

Temperature: Aktuální teplota [°C].

Cycle_Count: Celkový počet cyklů. Celkový počet, *Good*, *Bad*, *Suspect* a *Aborted* jsou počítány od inicializace (Po restartu se nuluje).

Good_Part_Count: Celkový počet dokončených cyklů, které nepřekročilo limity ani nejsou označeny jako podezřelé.

Bad_Part_Count: Celkový počet chybných cyklů – překročení limit.

Suspect_Part_Count: Celkový počet „podezřelých“ cyklů – například neočekávaný průběh.

Aborted_Part_Count: Celkový počet cyklů, které se nepodařilo spustit.

Cycle_Part_Status: Podrobnosti posledního svařovacího cyklu. Informace o překročení limit a výsledku cyklu.

Cycle_Weld_Time: Délka posledního cyklu v [ms].

Cycle_Weld_Peak_Power: Špičkový výkon posledního cyklu [W].

Cycle_Weld_Energy: Celková energie za poslední cyklus.

Active_Setup: Aktuálně nastavovaný řídící proces (na výběr mezi 16 pozicemi -> 16 pozic pro ukládání parametrů).

Running_Setup: Aktuálně vybraný řídící proces (použijí se parametry z *Active_Setup*).

Running_Probe: Aktuálně vybraná sonotroda (16 možností, má význam pouze při použití více sonotrod).

StEtherCAT
EtherCAT_ID: Adresa EtherCAT zařízení. Jde o NetID složku, která je popsána v kapitole 1.3.3.2.

EtherCAT_port: Port zařízení. Bližší informace viz. kapitola 1.3.3.1.

State: Viz kapitola 2.2.3.1.

WcState_validData: Hodnota True/False značí, zda komunikace je s chybou/bez chyby. True = chybná komunikace. False = komunikace bez chyby.

InputToggle: Mění svou hodnotu, aby poznal aktuálnost dat. Při výměně informací se při každém cyklu hodnota změní a zařízení pozná, že jsou data aktuální.

StBitmaskPdoOutCommand

Run_Continuous: Na náběžnou hranu začne svařovací cyklus. Cyklus je aktivní po nastavenou dobu.

Run_Weld: Svařovací cyklus je aktivován a vypínán společne s tímto bitem.

Run_Test: Test běží, pokud je bit v log.1. Test slouží k vyzkoušení generátoru a sontrody mimo svařovací cyklus.

Clear_Error: Při náběžné hraně dojde k resetování chyb v generátoru.

Amp_Otf_Enable: Povolení řízení amplitudy pomocí OTF.

Auto_Stop: Na náběžnou hranu dojde k ukončení svařovacího cyklu.

Run_Scan: Generátor začně skenovat sonotrodu a vyhledávat optimální parametry. Sken je spuštěn náběžnou hranou.

Clear_Setup_Refreshed_Output: Potvrzení změn v *Setup_Changed* ve struktuře v *BitmaskSystemStatus*, aby PLC rozlišilo další informaci o změně.

stPdoOut
Command: viz 1.3.3.1.

Otf_Amplitude_Setpoint: viz 1.3.3.1.

Active_Setup: Výběr nastavovaného řídícího procesu - parametry se nastavují pomocí SDO.

Running_Setup: Výběr aktuálního řídícího procesu.

Running_Probe: Výběr sonotrody, která bude použita v dalším cyklu.

stPdoIn -> System_Error_Status

Internal_Error: Došlo k vnitřní chybě. Jakýkoliv z těchto bitů znamená hlavní alarm. Bity jsou blíže nespecifikované.

Overload_Freq_Lock_Failed: Generátor se nemohl přepnout na požadovanou frekvenci.

Overload_Freq_Lock_Lost: Generátor neudržel požadovanou frekvenci.

Peak_Overload_Positive: Vnitřní proud generátoru překročil horní bezpečnou hranici (špatný převodník nebo nevhodné svařovací podmínky).

Peak_Overload_Negative: Vnitřní proud generátoru překročil spodní bezpečnou hranici (špatný převodník nebo nevhodné svařovací podmínky).

Average_Overload: Výkon překročil jmenovitou hodnotu pro danou amplitudu.

Current_Loop: Karta analogových vstupů je mimo rozsah 4-20mA.

Power_Not_Ok: Vnitřní napětí generátoru je mimo akceptovatelný limit.

Over_Temperature: Vnitřní teplota mimo akceptovatelný limit.

Over_Amplitude: Amplituda překročila hranici 100%.

Frequency_Bounds_Exceeded: Rezonanční frekvence překročila skokovou hranici.

Under_Voltage: Napájecí napětí je menší než požadované.

stPdoIn -> Generator_State

Idle: Není spuštěn cyklus.

Test: Spuštěn test generátoru.

Auto_Start_Delay: V cyklu, čeká na nastavený zpožděný start.
Pre-PreTrigger: Čeká na událost, která aktivuje svařovací akci před spuštěním cyklu.

PreTrigger: V cyklu, čeká na událost, která zahájí akci.

Trigger_Delay: V cyklu, podmínky ke spuštění akce splněny. Čeká na událost, která spustí akci.

Continuous: V cyklu, akce běží a je nastaven Run Continuously v Command_bitmask.

Weld1: V cyklu, akce Run Weld spuštěna, kontroluje ukončovací podmínky.

Weld2: V cyklu, akce Run Weld spuštěna, kontroluje ukončovací podmínky (při užití pneumatiky-lis).

Scrub: V cyklu, po dokončení akce, aktivován proces začišťování.

Dynamic_Hold: V cyklu, akce dokončena, dynamické přidržování svaru.

Static_Hold: V cyklu, akce dokončena, přidržování svaru bez pohybu.

AfterBurst_Delay: Svařovací cyklus dokončen a čeká se na uplynutí nastavené doby k aktivaci AfterBurst.

AfterBurst_Duration: Dokončuje se svařovací akce.

PreWeld_Scan: Spuštěn sken sonotrody.

Done: Operace dokončena.

RESERVED: Rezervováno.

DownStroke_Pause: Přerušení.

stPdoIn -> System_Status

Good_Part: Pokud během předchozího cyklu nedojde k chybě, nebo cyklus není označen jako podezřelý, je tento bit v log.1.

Bad_Part: Během předchozího cyklu došlo k překročení limit.

Suspec_Part: Minulý cyklus je označen jako podezřelý

Cycle_Start_Reject: Generátor nemohl spustit cyklus.

Amp_Regulation: Aktuální amplituda je v rozmezí +/- 1% od požadované hodnoty.

Power_Regulation: Aktuální výkon je v rozmezí +/- 10% od požadované hodnoty.

Trigger: Přijat požadavek na spuštění cyklu.

US_Status: Energie je dodávána připojené zátěži (materiál je svařován).

Power_Ok: Napájecí napětí je v přijatelných mezích.

MPC_Ready: MPC (Multi Probe Controller) přepl na vybranou sonotrodu a ta je připravena na svařovací cyklus (má smysl jen při použití více sonotrod).

Ready: Generátor je připraven.

Anyfault: Během posledního cyklu došlo k chybě.

Overload: Během posledního cyklu došlo k přetížení. Více informací viz *System_Error_Status*.

ESTOP: Aktivní "Central Stop".

Online: Generátor je online.

In_Hold: Generátor je v „Hold“ fázi cyklu.

In_Cycle: Probíhá cyklus, mimo svářecí akci je bit v log.1.

In_Cycle_no_AB: Probíhá cyklus, mimo svářecí akci je bit v log.0.

Overtemp: Překročena vnitřní teplota generátoru.

In_After_Burst: V cyklu, není pod zátěží (mimo svářecí akci).

In_Test: Probíhá testování.

In_Test_Scan: probíhá skenování sonotrody.

Setup_Changed: Změny byly akceptovány.

stPdoIn -> Cycle_Part_Status

Good_Part: Stejný význam jako v ST_Dukane_BitmaskSystemStatus. Na začátku každého cyklu je tento bit vyresetován. Na konci každého cyklu je jeden z *Good*, *Bad* a *Suspect* nastaven do log.1.

Bad_Part: viz *Good_Part*.

Suspect_Part: viz. *Good_Part*.

Lower_Time_Bad_Limit: Nedodržen čas svařování pro *Bad_Time* během posledního cyklu.

Lower_Time_Suspect_Limit: Nedodržen čas svařování pro *Suspect_Time* během posledního cyklu.

Upper_Time_Suspect_Limit: Překročen čas svařování pro *Suspect_Time* během posledního cyklu.

Upper_Time_Bad_Limit: Překročen čas svařování pro *Bad_Time* během posledního cyklu.

Lower_Peak_Power_Bad_Limit: Během posledního cyklu byl překročen spodní limit výkonu pro *Bad_Part*.

Lower_Peak_Power_Suspect_Limit: Během posledního cyklu byl překročen spodní limit výkonu pro *Suspect_Part*.

Upper_Peak_Power_Suspect_Limit: Během posledního cyklu byl překročen horní limit výkonu pro *Suspect_Part*.

Upper_Peak_Power_Bad_Limit: Během posledního cyklu byl překročen horní limit výkonu pro *Bad_Part*.

Lower_Energy_Bad_Limit: Během posledního cyklu byl překročen spodní limit energie pro *Bad_Part*.

Lower_Energy_Suspect_Limit: Během posledního cyklu byl překročen spodní limit energie pro *Suspect_Part*.

Upper_Energy_Suspect_Limit: Během posledního cyklu byl překročen horní limit energie pro *Suspect_Part*.

Upper_Energy_Bad_Limit: Během posledního cyklu byl překročen horní limit energie pro *Bad_Part*.