Vysoké učení technické v Brně

Fakulta elektrotechniky a komunikačních technologií

Ústav automatizace a měřící techniky

Řízení separace chemických vzorků dvojicí vysokonapěťových zdrojů

Název

Obor: Kybernetika, automatizace a měření

Student: Bc. Jiří Zálešák

Vedoucí: Ing.Radek Štohl, Ph.D.

Abstrakt :

První část práce je zaměřena na hardwarové řešení a realizaci desky plošných spojů. Řešeným problémem je galvanické oddělení komunikačních signálů mezi vysokonapěťovým zdrojem CZE2000 od firmy Spellman a systémem PLC AMiRiS99 od firmy AMiT.

Druhá část projektu je zaměřena na komunikaci PLC s aplikací LabVIEW na PC, která je zde realizována pomocí prvků ActiveX..

Třetí část práce je zaměřena na programování PLC a vytvoření řídícího algoritmu pro separaci ve vývojovém prostředí DetStudio, jedná se o volně stažitelný software firmy AMiT s.r.o..

Čtvrtá část je zaměřena na tvorbu aplikace v programu LabVIEW, pomocí které probíhá komunikace a ovládání řídícího systému.

Poslední část práce se zabývá testováním navrženého systému na reálné separaci chemických vzorků.

Klíčová slova: Separace, AMiRiS99, ATOUCHX, LabVIEW, CZE2000

Brno University of Technology

Faculty of Electrical Engineering and Communication

Department of Control, Measurement and Instrumentation

Separation control of chemical solutions by high voltage supply pair

Thesis

Specialisation of study: Cybernetics, Control and Measurement

Student: Bc. Jiří Zálešák

Supervisor: Ing.Radek Štohl, Ph.D.

Abstract :

The first part of thesis is intent on hardware solution and realization PCB. Solve problem is galvanic separation communications signals between high - voltage supply CZE2000 from Spellman and system PLC AMIRIS99 from AMiT.

The second part project is intent on communications PLC with application LabVIEW on PC, there is realized via ActiveX.

Third part is intent on programming PLC and creation control algorithm. There are programmed in DetStudio. This program is freeware software firm AMiT Ltd.

The fourth part is intent on creation application in software LabVIEW. This application communication with control system.

The last part of thesis is intent on testing system on Separation of chemical solutions.

Keywords: Separation, AMiRiS99, ATOUCHX, LabVIEW, CZE2000

Bibliografická citace mé práce:

ZÁLEŠÁK, J. *Řízení separace chemických vzorků dvojicí vysokonapěťových zdrojů.* Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2008. 78 s. Vedoucí diplomové práce Ing. Radek Štohl, Ph.D.

P r o h l á š e n í

„Prohlašuji, že svou diplomovou práci na téma "Řízení separace chemických vzorků dvojicí vysokonapěťových zdrojů" 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 diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové 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 § 152 trestního zákona č. 140/1961 Sb.“

V Brně dne : Podpis:

Poděkování

Tímto děkuji mému vedoucímu semestrálních projektů a následné diplomové práce Ing. Radkovi Štohlovi, Ph.D. za pomoc při realizaci zadání.

Dále děkuji Doc. Ing. Janu Pospíchalovi, CSc. z Mendelovy zemědělské a lesnické univerzity v Brně za pomoc při řešení záležitostí týkajících se chemie.

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

1. ÚVOD

V semestrální projektu číslo jedna jsem se seznámil s několika metodami a principem elektroforézy. Patří sem např. kapilární zónová elektroforéza (CZE), micelární elektrokinetická kapilární chromatografie (MECC), kapilární izoelektrická fokusace (CIEF) a kapilární izotachoforéza (CITP). Tyto metody bývají často v literatuře označovány společným názvem kapilární elektroforéza.

První část této práce je zaměřena na návrh detailní koncepce řízení v reálném čase, návrhu hardwaru a následné realizaci desky plošných spojů. Použitý systém je PLC AMiRiS99 od firmy AMiT, který bude ovládán pomocí počítače a programu LabVIEW. Řízení se týká dvou vysokonapěťových zdrojů CZE2000 od firmy Spellman, kdy první zdroj slouží k samotné elektroseparaci a druhý zdroj ke změně Ph elektrolytu. Návrh a realizace hardwaru se týká galvanického oddělení PLC od vysokonapěťových zdrojů.

Druhá část je zaměřena na popis komunikace systému PLC AMiRiS99 od firmy AMiT s.r.o. Komunikaci PLC s aplikacemi na PC zajišťují knihovny nazvané ATOUCH. V případě komunikace s aplikací LabVIEW se jedná přímo o ATOUCHX využívajících prvků ActiveX.

Třetí část se zabývá programováním řídícího algoritmu v PLC. Zde popsané a použité vývojové prostředí se nazývá DetStudio. Jedná se o software, který firma AMiT nabízí bezplatně ke stažení na svých internetových stránkách.

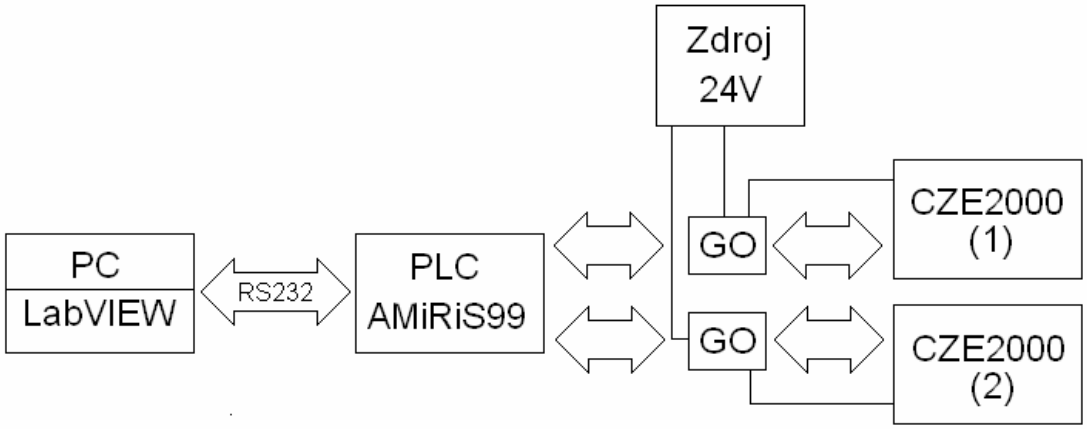
Čtvrtá část je zaměřena na realizaci a popis funkčních bloků v programu LabVIEW sloužících ke komunikaci. Dále je zde realizována konečná aplikace sloužící k ovládání PLC a řízení procesu separace.

Poslední část práce se zabývá testováním celého navrhnutého systému a realizovaného řídícího algoritmu. Testování systému probíhá na reálné separaci chemických vzorků. Zvolenou separační metodou je v tomto případě kapilární izotachoforéza (CITP).

2. KONCEPCE ŘÍZENÍ

2.1 ZAPOJENÍ SYSTÉMU

Obrázek 1 zobrazuje blokové schéma zapojení systému. Systém PLC AMiRiS99 je spojen s PC přes komunikační linku RS232. Oba vysokonapěťové zdroje jsou zde galvanicky odděleny od PLC. Každý blok galvanického oddělení obsahuje 6 analogových a 8 digitálních kanálů (podrobný popis propojení konkrétních signálů PLC s CZE2000 viz kap. 2.4.3). Zdroj č.1 slouží k samotné separaci a zdroj č. 2 ke změně Ph použitého elektrolytu. Vzhledem k tomu, že oba zdroje jsou konstrukčně shodné, ale liší se maximálními hodnotami napětí a proudu (zdroj č.1 – 30 kV, 300 μA, zdroj č.2 – 20 kV, 500 μA), veškerý popis týkající se zdroje je zaměřen na zdroj č.1.



Obrázek 1: Blokové schéma zapojení systému

2.2 VN ZDROJ

Vysokonapěťový zdroj obsahuje pouze dva konektory. Jeden je pro výstup vysokého napětí a druhý je CANON 25, který obsahuje řídicí piny a piny pro napájení zdroje. Druhý vodič vysokého napětí je přiveden na konstrukci zdroje. Zdroj není napájen přímo ze sítě, hodnota napájecího napětí je 24V.

Zdroj umožňuje plynulé nastavení napětí od 0 do 30 kV a proudu od 0 do 300µA, což plně dostačuje pro proces kapilární elektroforézy. Popis všech pinů uvádí tabulka 1.

J2	Signál
1, 2, 3	Kostra & zem 24V DC
4	Vysoké napětí - Povolit/Zakázat
5	Napěťový kontrolní pin
6	Proudový kontrolní pin
7	Kostra
8	Ovládání napětí
9	Ovládání proudu
10	+10,24V DC referenční
11	Návratový vodič k ovládacím a kontrolním pinům
12	Ovládání polarity
13	Indikátor kladné polarity
14, 15	+24V DC
16	Kostra
17	Indikátor záporné polarity
18	I – mód indikátor
19	V – mód indikátor
20	Kontrolní pin ztrácejícího se proudu
21	Opačný vodič zátěže (při použití pinu 20)
22	Indikátor ztrácejícího se proudu
23, 24, 25	Rezervní vývody

Tabulka 1: Konektor CANON VN zdroje Spellman CZE2000 [1]

2.2.1 Popis ovládání VN zdroje

Literatura Spellman [1] uvádí následující popis ovládání zdroje. VN zdroj je napájen 24V ±10%, jištěn 2A pojistku a jeho celkový výkon dosahuje 9W. Vstupní proud je max. 1,25A.

Ovládání napětí a proudu je prováděno přes externí zdroj napětí. Přivedením napětí od 0 do 10V na řídicí pin 8 lze plynule měnit výstupní napětí v celém rozsahu 0 až 30 kV. Stejně tak probíhá i ovládání proudu přivedením napětí na pin 9. Přesnost nastavení je 0,1%. Nastavené napětí a proud je možno snímat z kontrolních pinů 5 a 6 opět v rozsahu 0 – 10V s přesností 1% u napětí a 2% u proudu.

Povolení vysokého napětí lze provádět na pinu 4. Hodnoty napětí odpovídají TTL logice, kdy při logické nule je vysoké napětí zakázáno.

Změna polarity je prováděna přes pin 12, kdy logická 1 nastavuje kladnou a logická nula zápornou polaritu výstupního napětí. Kladná polarita je indikována +24 V na pinu 13 a záporná polarita na pinu 17.

Napěťový režim zdroje je indikován na pinu 19 a to +15 V. Obdobně je proudový režim indikován na pinu 18. Zdroj obsahuje vnitřní automatický regulační obvod mezi napěťovým a proudovým režimem. Pokud v napěťové režimu dojde ke zvětšení zatěžovacího proudu a napětí se sníží pod nastavenou hodnotu, vnitřní regulační obvod automaticky přepíná zdroj do proudového režimu. Stejně tak pokud při proudovém režimu dojde k poklesu zatěžovacího proudu a tím ke zvýšení napětí přes nastavenou hodnotu, zdroj automaticky přepíná do napěťového režimu.

Zdroj umožňuje měření „ztrácejícího se proudu“. Opačný vodič od zatížení přiveden na pin 21, způsobí, že na pinu 20 je napětí odpovídající ztrácejícímu se proudu, který neteče pinem 21, ale vrací se přes zem na konstrukci přístroje. Indikátor ztrácejícího se proudu pin 22 má na výstupu hodnotu +15 V, když rozdílový proud překročí 20 μA.

Dle literatury SPELLMAN [1] zde uvádím **postup pro bezpečné zacházení** se zdrojem:

a) vysokonapěťový kabel musí být zatížen a zabezpečen proti náhodnému dotyku
b) zdroj musí být jištěn 2A pojistkou
c) konstrukce zdroje musí být uzemněna
d) návratový vodič zatížení musí být připojen ke konstrukci
e) napětí a proud po připojení řízení musí být nastaveno na 0 a zakázáno vysoké napětí
f) nastavení požadované polarity
g) může být přivedeno napájecí napětí zdroje
h) následuje povolení vysokého napětí
i) pomalé nastavení výstupního napětí a proudu až k požadované hodnotě
j) při vypnutí zdroje nejdříve zakázat vysoké napětí
k) při delší nečinnosti odpojit zdroj od napájení

2.3 SYSTÉM AMIRIS99

K řízení vysokonapěťových zdrojů je použit PLC AMiRiS99 firmy AMiT. Jedná se o kompaktní řídicí systém v celokovovém provedení určený pro menší aplikace. Svými výpočetními schopnostmi překračuje třídu klasických PLC. Systém s řadou doplňujících funkcí. Technické údaje PLC [6] jsou uvedeny v tabulce 2.

Číslicové vstupy Univerzální střídavý / stejnosměrný	16 × 24 V ss/st Logická 0 min -30 V, max 5 V Logická 1 min 16 V, max 30 V
Ochrana číslicových vstupů	Transil 600 W
Pevnost galvanického oddělení	600 V
Číslicové výstupy tranzistorové	16 × 24 V/0,5 A ss
Ochrana tranzistorových výstupů	Transil 600 W
Číslicové výstupy reléové	9 × na modulu AREL7S2P-X
Analogové vstupy	8 × 0..10 V / 0..5 V / 0..20 mA / Ni1000
Ochrana analogových vstupů	Diody + odpor 10 kΩ
Analogové výstupy s modulem AM-AO2U s modulem AM-AO2I	Max 4 × 0..10 V, zatížení maximálně 10 mA 0..20 mA, maximální impedance 500 Ω
Ochrana analogových výstupů	Transil 600 W
Sériový komunikační kanál	RS232 RS485 galv. odd. 600 V (s modulem AM-RS485) CAN galv. odd. 600 V (s modulem AM-CAN) RS232 bez galv. odd. (s modulem AM-RS232) M-BUS galv. odd. 600 V (s modulem AM-MBUS)
Krytí	IP20, Umístěno v kovovém krytu
Připojení signálů	Pružinové konektory WAGO
Napájení	24 V ss ±20 %
Odběr (bez výstupů)	Max. 250 mA při 24 V
Pracovní teplota	0 ÷ 70 °C
Maximální vlhkost okolí	95 % nekondenzující
Hmotnost	1,9 kg
Rozměry (š × v × h)	316 × 193 × 50 mm
Zálohování RAM	5 let
Programování	PSP3 (NOS), jazyk C (AC166)

Tabulka 2: Technické údaje systému AMiRiS99 [6]

Z tabulky 2 je vidět, že PLC obsahuje 16 číslicových vstupů, 16 číslicových výstupů, 8 analogových vstupů a 4 analogové výstupy, což bohatě postačuje k řízení a snímání všech potřebných funkcí u obou napěťových zdrojů.

K indikaci kladné/záporné polarity, napěťového/proudového režimu a ztrácejícího se proudu bude použito pro oba zdroje 10 číslicových vstupů.

K nastavení polarity, povolení/zakázání vysokého napětí a zapnutí/vypnutí napájecího napětí pro oba zdroje bude použito 6 číslicových výstupů.

K ovládání napětí a proudu pro oba zdroje budou použity 4 analogové výstupy, u kterých se hodnota výstupního napětí mění od 0 do10 V.

K snímání napětí, proudu, referenčního napětí a ztrácejícího se proudu pro oba zdroje bude použito 8 analogových vstupů.

Popis zapojení pinů ze zdrojů ke konkrétním vstupům/výstupům PLC AMiRiS99 je uveden v kapitole 2.4.3.

2.4 HW ŘEŠENÍ GALVANICKÉHO ODDĚLENÍ

2.4.1 Popis zapojení

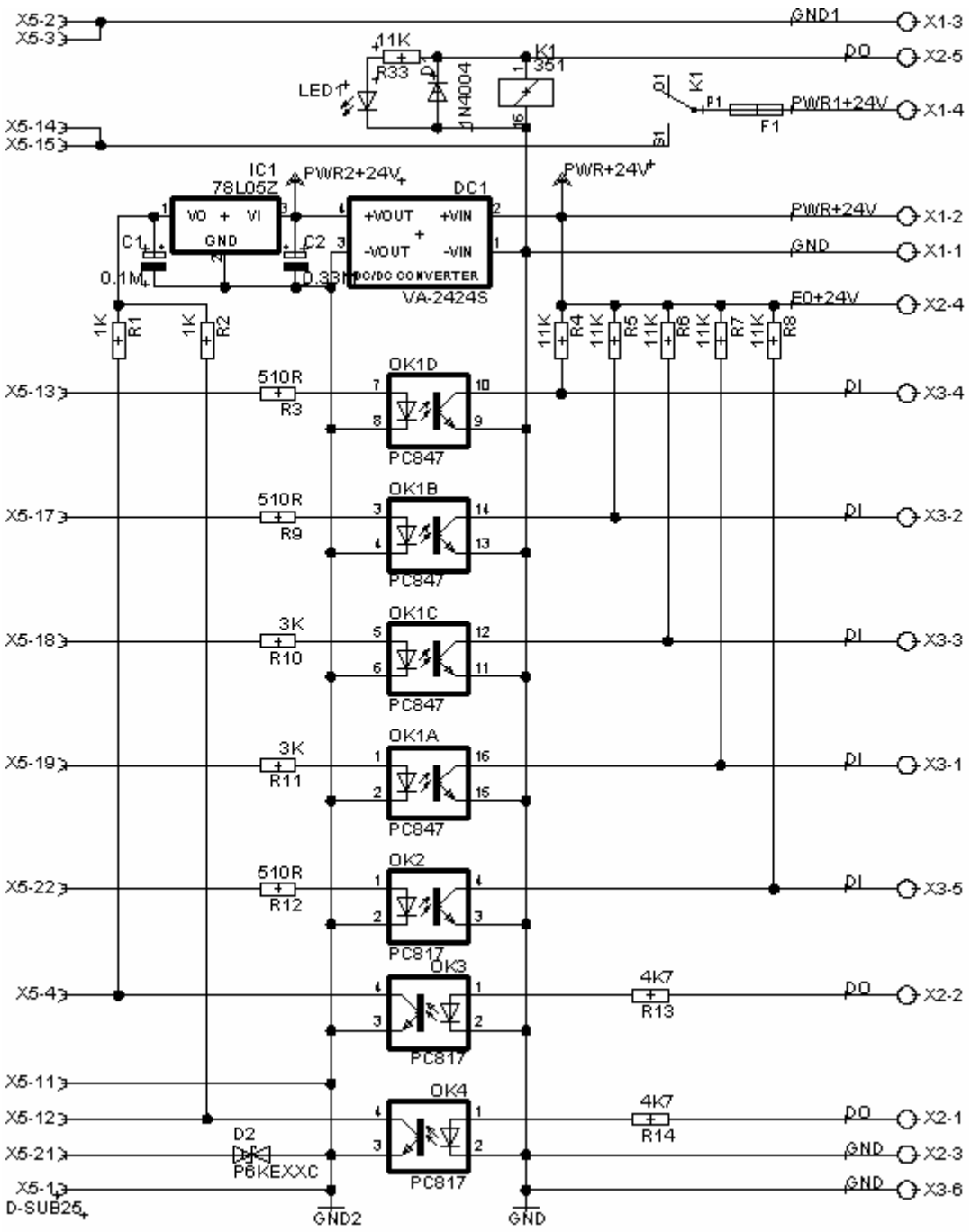
Cílem návrhu bylo galvanické oddělení mezi vysokonapěťovým zdrojem a systémem AMiRiS99. Byl navržen obvod, který obsahuje 13 galvanicky oddělených kanálu, z toho 7 číslicových (obr.2) a 6 analogových (obr.3).

Navržený obvod je s vysokonapěťovým zdrojem propojen přes kabel CANON25V-CANON25Z a spojení s PLC je realizováno pomocí svorkovnic.

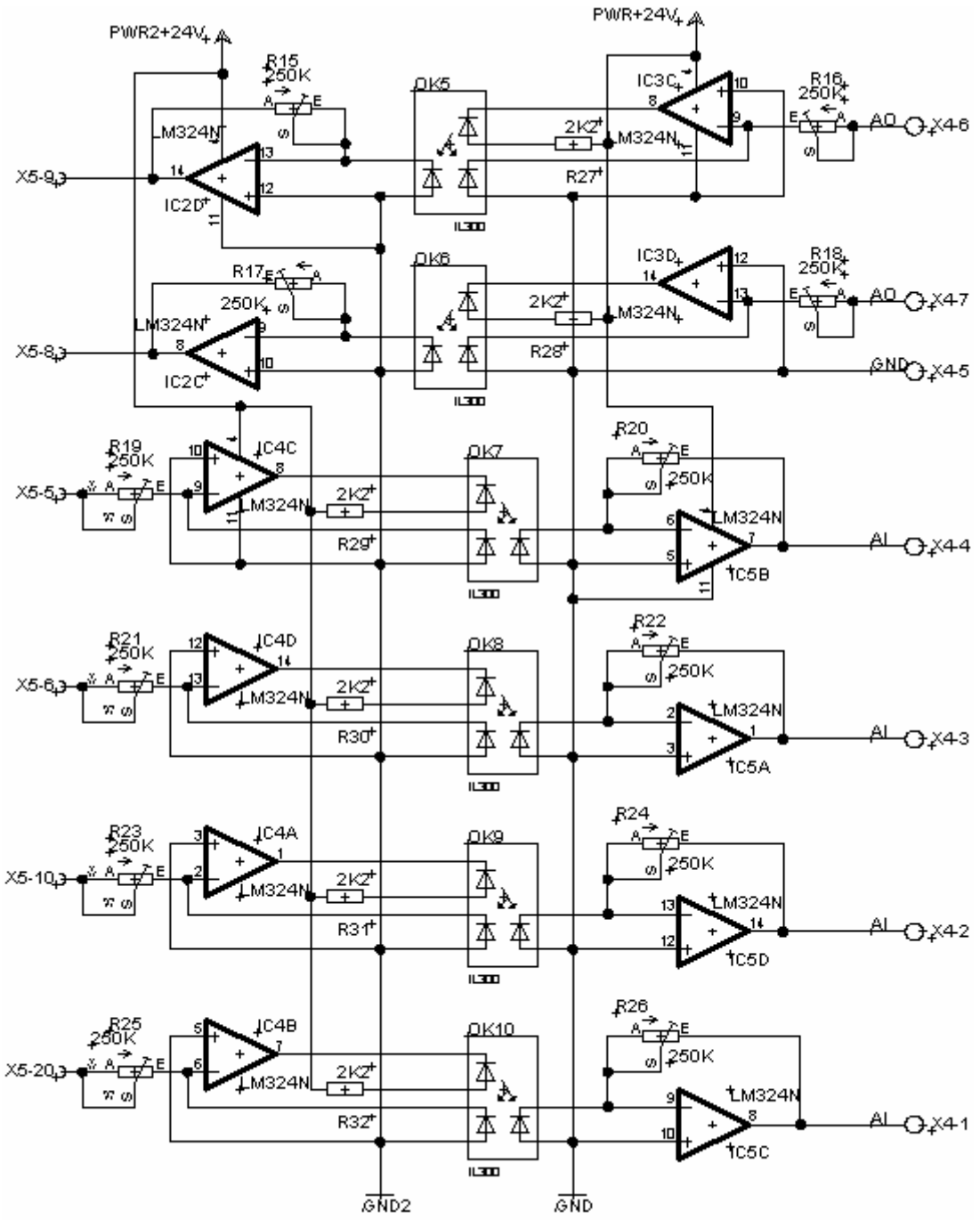
Hodnota napětí pro logické 1 číslicových vstupů PLC zde byla zvolena 24V. K oddělení napájecích napětí obvodu je zde použit DC24V/DC24V měnič. Tento měnič jsem zvolil z důvodu stejných hodnot napájení operačních zesilovačů na obou stranách. Povolení/zakázání vysokého napětí a změna polarity zdroje jsou ovládány napětím 5 V. Z tohoto důvodu je na tyto piny přivedeno napětí z 24 V výstupu DC/DC měniče přes 5V stabilizátor. Pro oddělení číslicových signálů byly zvoleny optočleny PC847 a PC817.

Pro galvanické oddělení analogových signálů jsem se rozhodl pro použití lineárního optočlenu IL300 spolu s operačními zesilovači LM324. Obvod se vyznačuje velmi dobrou linearitou. Použitý obvod vychází z doporučeného zapojení udávaného výrobcem. [2]

2.4.2 Schéma zapojení



Obrázek 2: Schéma zapojení galvanického oddělení (1/2)



Obrázek 3: Schéma zapojení galvanického oddělení (2/2)

2.4.3 Popis propojení vstupů/výstupů PLC s CZE2000

Tabulka 3 obsahuje detailní popis připojení konkrétních vstupů/výstupů systému AMiRiS99 ke konkrétním ovládacím a indikačním pinům obou vysokonapěťových zdrojů CZE2000.

Svorkovnice	Signál PLC - zdroj č.1	Signál PLC - zdroj č.2	Pin zdroje	Popis
X1-1	GND	GND		zem od napájení GO
X1-2	PWR+24V	PWR+24V		napájení GO +24V
X1-3	GND1	GND1	X5-2, X5-3	zem od napájení zdroje
X1-4	PWR1+24V	PWR1+24V	X5-14, X5-15	napájení zdroje +24V
X2-1	DO0.0	DO1.0	X5-12	ovládání polarity
X2-2	DO0.1	DO1.1	X5-4	povolení/zakázání vysokého napětí
X2-3	E0GND	E1GND		zem od digitálních výstupů PLC
X2-4	E0+24V	E1+24V		napájení digitálních výstupů PLC
X2-5	DO0.2	DO1.2		ovládání relé - napájení zdroje
X3-1	DI0.0	DI1.0	X5-19	indikace napěťového režimu
X3-2	DI0.1	DI1.1	X5-17	indikace záporné polarity
X3-3	DI0.2	DI1.2	X5-18	indikace proudového režimu
X3-4	DI0.3	DI1.3	X5-13	indikace kladné polarity
X3-5	DI0.4	DI1.4	X5-22	indikace ztrácejícího se proudu
X3-6	I0GND	I1GND		zem od digitálních vstupů
X4-1	AI0.0	AI0.4	X5-20	ztrácející se proud
X4-2	AI0.1	AI0.5	X5-10	reference +10,24V DC
X4-3	AI0.2	AI0.6	X5-6	kontrola proudu
X4-4	AI0.3	AI0.7	X5-5	kontrola napětí
X4-5	AGND	AGND		zem od analogových vstupů/výstupů
X4-6	AO0.0	AO0.2	X5-9	ovládání proudu
X4-7	AO0.1	AO0.3	X5-8	ovládání napětí

Tabulka 3: Popis propojení vstupů/výstupů PLC s CZE2000

3. KOMUNIKACE

3.1 ATOUCH

Ovladače ATOUCH slouží ke komunikaci systému s aplikacemi na PC. Zajišťují přenos údajů z a do systému a pracují s ActiveX prvky.

ATOUCH jsou součástí i ovladačů do prostředí LabVIEW a mnoha dalších programů. Mimo to jsou tyto knihovny bezplatně nabízeny na internetových stránkách firmy AMiT. [2]

3.2 ATOUCHX

Knihovna AtouchX obsahuje ActiveX objekty, které zprostředkovávají připojení počítače s řídicími systémy firmy AMiT. Dále pak knihovna obsahuje ActiveX objekt pro zpětné čtení archivů.

Všechny objekty knihovny AtouchX mají funkcionální charakter. To znamená, že objekty nemají žádné vlastnosti nebo jen několik málo vlastností a to spíše provozního charakteru a veškerá práce s nimi probíhá pomocí metod.

Všechny metody vracejí kód chyby, je to číslo určující důvod selhání metody. Speciálním případem je hodnota 0, která znamená, že metoda dopadla bez chyby. Pokud metoda vrací nějakou hodnotu, pak obsahuje odpovídající parametr předávaný odkazem. [2]

3.2.1 AtouchDir

Objekt AtouchDir zprostředkovává připojení počítače k řídicím systémům firmy AMiT. Připojení je implementováno jako direktivní. Při direktivním způsobu připojení se vyvolává komunikační metoda. Metoda zabezpečí přenos dat a počká na konec komunikace. Po návratu z metody jsou k dispozici data a údaj, jak komunikace dopadla. [2]

Objekt je třeba použít tak, že po vytvoření se vyvolá některá z inicializačních metod. Po úspěšné inicializaci je možno používat ostatní metody objektu (existují však i metody pracují bez inicializace). Při ukončení je třeba nejprve vyvolat metodu Done a pak teprve objekt zrušit. [2]

Objekt AtouchDir obsahuje následující metody:

Inicializace, deinicializace
InitFromFile - Inicializace objektu ze souboru.
InitFromString - Inicializace objektu z řetězce.
Done - Ukončení činnosti objektu.
Verze a informace
GetKernelVersion - Zjistí verzi komunikačního jádra .
GetObjectVersion - Zjistí verzi objektu.
GetInfo - Zjistí informace o komunikačním jádře.
GetHWInfo - Zjistí informace o HW připojení stanice.
Konverze času
TimeFromDbNet, TimeFromDbNetV - Převede čas z DbNet formátu do VBA formátu.
TimeToDbNet, TimeToDbNetV - Převede čas z VBA formátu do DbNet formátu.
Databáze
DbGetInfo - Zjistí informace o databázové proměnné.
Komunikace
GetData, GetDataV - Přečte celou databázovou proměnnou.
GetDataMtx, GetDataMtxV - Přečte výřez databázové matice.
PutData, PutDataV - Zapíše celou databázovou proměnnou.
PutDataMtx, PutDataMtxV - Zapíše výřez databázové matice.
GetTime, GetTimeV - Přečte čas stanice.
PutTime, PutTimeV - Zapíše čas do stanice.
Identify, IdentifyV - Přečte identifikaci stanice.
Správa připojení
StationStatus - Zjistí stav připojení stanice.
StationConnect - Zahájí proces připojování stanice.
StationDisconnect - Zahájí proces odpojování stanice.
StationReserve - Nastaví nebo zruší rezervaci pro spojení se stanicí.
StationSetPriority, StationSetPriorityV - Nastaví prioritu pro spojení se stanicí.

Tabulka 4: Metody objektu AtouchDir [2]

4. PROGRAMOVÁNÍ

4.1 DETSTUDIO

DetStudio je návrhové prostředí pro řídicí stanice z produkce firmy AMiT. Na internetových stránkách firmy AMiT je možnost získat zdarma nejnovější verzi vývojového prostředí a knihovny funkčních modulů.

DetStudio umožňuje tvorbu aplikací pro řídicí stanice AMiT. Prostředí se skládá z několika nástrojů.

Hlavním nástrojem je editor procesů a podprogramů, ve kterém se do procesů a podprogramů zapisují příkazy programovacího jazyka a vytváří se algoritmická část řídící aplikace.

Dalším nástrojem je WYSIWYG (jedná se o program, kde se vkládají jednotlivé bloky zobrazovací jednotky na obrazovku PC a verze zobrazená na obrazovce PC je vzhledově totožná s výslednou verzí na zobrazovací jednotce) editor obrazovek. Zde se skládají jednotlivé grafické prvky a pomocí proměnných jsou propojeny s algoritmickou částí aplikace.

4.2 PROGRAMOVÁNÍ PLC

Jak již bylo uvedeno v kapitole 4.1, použitým vývojovým prostředím je software DetStudio. Při zápisu programu je možnost vybrat si ze tří editorů, jedná se o dva textové – ST (strukturovaný text), LA (jazyk logických adres) a jeden grafický – RS (jazyk reléových schémat). Zvláštním editorem je zde editor příkazu Let. Jedná se o přiřazovací příkaz typu *proměnná = výraz*. Používá se k veškerým výpočtům, logickým a srovnávacím operacím.

4.3 ŘÍDÍCÍ ALGORITMUS V PLC

Celý řídící algoritmus pro PLC AMiRiS99 firmy AMiT byl vytvořen ve vývojovém prostředí DetStudio. Krátký popis prostředí je uveden v kapitole 4.1.

Pro programování jsem si vybral editor ST - strukturovaný text. Z mého pohledu poskytuje nejsnadnější programování a hlavně nabízí nejširší možnosti.

Jedná se o zápis sekvencí modulů. V každém modulu je proměnný počet parametrů. Dále se jedná o moduly tzv. pseudojazyka. V těchto modulech dochází k větvení programu nebo k vytváření programových smyček. Posledním modulem je modul Let, který má jako parametr výraz.

4.4 POPIS PROGRAMU

DetStudio umožňuje rozdělit celý algoritmus do tzv. procesů. Proces je tedy část programu, která pracuje relativně samostatně a nezávisle na ostatních procesech. Do těchto procesů se pak umístní program vytvořený v jednom z možných editorů. Téměř všechny procesy jsou spouštěny periodicky. Přehled všech procesů udává tabulka 7.

Označení	Počet	Popis
Normal	16	Obvyklé regulační či měřící smyčky.
Idle	1	Nejméně prioritní děje (typicky obsluha vizualizace).
Quick	1	Rychlý děj na úrovni 50 ms.
HiSpeed	2	Velmi rychlé děje až na úrovni 1ms.
Interrupt	16	Rychlá reakce na vybraných digitálních vstupech.
Init	1	Inicializace aplikace.

Tabulka 5: Seznam procesů [7]

Při tvorbě programu jsem z tabulky 5 použil dva typy procesů. Patří mezi ně procesy Normal, kterých jsem z možných 16 použil celkem 6 (popis uveden v kapitole 5.2) a z důvodu možnosti ovládání systému přes přídavný terminál i proces Idle.

Procesy typu Normal umožňují nastavit periodu spouštění v intervalu 100 ms až 1000000 s (přibližně 11dní). Veškeré procesy jsem nastavil na nejrychlejší periodu spouštění – 100 ms. Vzhledem ke koncepci řízení jsou tyto rychlosti plně dostačující.

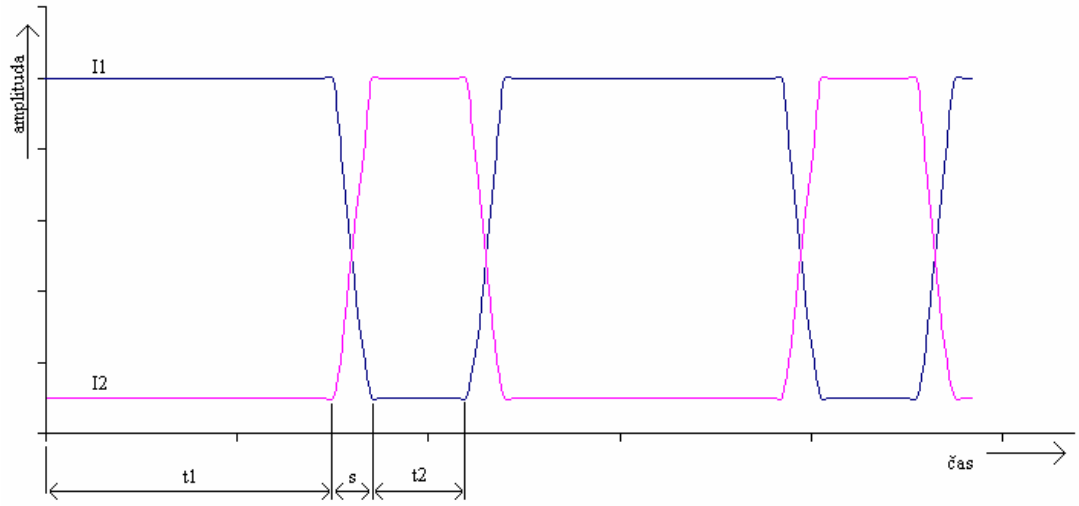
4.5 POPIS PROCESŮ

Proces – Hlavni

Tento proces obsahuje jedno z hlavních větvení celého programu. Umožňuje provádět separaci v tzv. „normálním režimu“ a „pulsním režimu“.

Část programu obsluhující normální režim obsahuje algoritmus, který zajišťuje po zadání hodnot napětí a proudů jejich pomalé a plynulé nastavení na výstupu PLC a tím i na výstupu vysokonapěťového zdroje. Požadavek na tento krok vychází z postupu pro bezpečné zacházení s vysokonapěťovým zdrojem, který je uveden v kapitole 2.2.1. Dalším požadavkem pro tento krok bylo také to, aby došlo k minimalizaci časového zpoždění mezi hodnotou v PLC a hodnotou na výstupu VN zdroje. Dobu, za kterou dojde k nastavení požadované hodnoty, jsem určil experimentálně a stanovil ji na 4 s. Tento čas je s rezervou a plně dostačující k tomu, aby se výstupní napětí zdroje změnilo z 0 na 30 kV. Změnu napětí a proudů jsem zvolil lineární.

Druhá část tohoto procesu obsahu algoritmus zajišťující pulsní režim. V tomto režimu dochází k výměně zadaných proudů dle obrázku 8. Proud I_1 slouží k separaci a proud I_2 ke změně pH. Hodnoty časů t_1 a t_2 jsou uživatelsky nastavitelné. Hodnota s je určena pevně a její hodnota je 4 s. Algoritmus pulsního režimu využívá k plynulé změně hodnot proudů část zdrojového kódu z normálního režimu.



Obrázek 4: Průběh proudů v pulsním režimu

Proces – in_out

Proces in_out obsahuje pouze moduly pro zápis/čtení binárních a analogových signálů. Tyto moduly obsahují také převod fyzikálních veličin.

Jako příklad zde uvedu moduly pro čtení/zápis jednoho binárního a jednoho analogové vstupu/výstupu.

BinOut - zapíše hodnotu zvoleného bitu proměnné typu *I* na výstup logického kanálu DO. Obsahuje tři parametry – jméno proměnné, příznak negace výstupního signálu, číslo zapisovaného signálu DO.

BinIn - načte hodnotu jednoho digitálního vstupního signálu z logického kanálu do zvoleného bitu proměnné typu *I*. Obsahuje tři parametry – signál logického kanálu z něhož se bude číst, příznak negace vstupního signálu, bit proměnné do kterého bude zapsána hodnota načteného signálu.

AnOut - zapisuje hodnotu proměnné jejíž fyzikální rozměr přepočítává na rozsah převodníku do logického kanálu. Obsahuje celkem sedm parametrů – číslo logického vstupního kanálu AO, hodnota ve fyzikálních jednotkách, horní a dolní hranici výstupního signálu v elektrických jednotkách, horní a dolní hranici ve fyzikálních jednotkách.

AnIn – čte analogový signál z příslušného AI kanálu a přepočítává jej na fyzikální rozměr. Obsahuje celkem sedm parametrů, které jsou shodné jako u modulu *AnOut.*

```
BinOut @polarita, 0x0000, #DO00_0          //ovládání polarity
BinIn  #DI00_1, 0x0001, @zap_pol           //záporná pol., neg.
AnOut  #AO00_0,p_i_put[0,0],10.000,0.000,10.000,0.000,500.000
AnIn   #AI00_2,c_proud[0,0],10.000,0.000,10.000,0.000,500.000
```

Proces – citace

V tomto procesu jsou inkrementovány proměnné, které slouží pro plynulé nastavení napětí a proudů, nastavení časů *t1* a *t2* a nastavení mezí.

```
IncDec Hodnota, Běh, Přírůstek
```

Uvedený modul inkrementuje/dekrementuje proměnou *Hodnota* o hodnotu *Přírůstek.* Bit *Běh* modul spouští nebo zastavuje.

Proces – prepinac

Proces umožňuje výběr zda jednotlivé proudy (proud na separaci I_1 a proud na změnu pH I_2) budou v daném poměru, nebo ne. Pokud je požadavek a poměr je zapnut, pak $I_2 = I_1$/pomer, kde pomer je uživatelsky zadaná hodnota. V tomto případě se již hodnota I_2 nemusí zadávat a je automaticky vypočtena z proudu I_1.

Dále proces obsahuje algoritmus rozhodovaní, zda-li mají být použity hodnoty pro zadání mezí pro napětí a proud stejné nebo se zadává každá hodnota zvlášť. Pokud je požadavek na stejné meze, není pak nutné nastavovat jednotlivé meze a pro všechny je nastavená stejná hodnota. Zadávané hodnoty jsou v procentech.

Veškeré rozhodování, nejen v tomto procesu je v programu realizováno modulem *If-EndIf.*

```
If Test.0, :00000
      .
      .
      .
:00000 EndIf
```

Tento modul realizuje podmíněný příkaz. Modul testuje podmínku - bit *Test.0,* je-li podmínka splněna, je vykonána sekvence následujících modulů. Při nesplnění podmínky dochází k vykonávání modulů bezprostředně za *EndIf.*

Proces – kontrola

V procesu kontrola dochází k testování, zda-li je hodnota napětí a proudu v daných mezích. Tato kontrola zajišťuje snížení napětí a proudu na nulové hodnoty a vypnutí vysokého napetí pokud dojde k překročení zadaných mezí.

Pokud např. dojde při separaci ke zplynatění elektrolytu v kapiláře, dochází k rychlé změně vodivosti, a tím i nárůstu/poklesu napětí a proudu. Protože vysokonapěťový zdroj obsahuje vnitřní regulátor, je nutné hlídat současně napětí i proud. Při zvýšení odporu se snaží vnitřní logika zdroje udržet proud na konstantní úrovni tím, že zvýší napětí. Právě rychlé zvýšení napětí je signálem, že nastala v elektrolytu změna, např. již zmíněné zplynatění elektrolytu.

Nastavení mezí pro hlídání je do jisté míry věcí cviku. Pokud uživatel nastaví meze příliš vysoké, může se stát že k ochraně vypnutím nedojde a v kapiláře budou probíhat výboje, které jednak znehodnotí zkoumaný vzorek, ale může se i stát, že při použití plastové kapiláry dojde vlivem vznikajícího tepla k jejímu poškození. Na druhou stranu, při nastavení mezí na příliš nízkou hodnotu může docházet k vypínaní vysokého napětí vlivem vnějšího rušení. Při nastavení malých hodnot proudu, v jednotkách μA, doporučuji nastavit meze na vyšší hodnotu, nejméně 30% a naopak u vyšších hodnot v řádu stovek, nejvýše 10%. U napětí je to obdobné.
Jednotlivé hodnoty mezí nejsou vypočítávány z nastavovaných hodnot, ale z hodnot, které jsou zpětně čtené a jednotlivé přenastavení probíhá jednou za 30 s. K tomuto kroku mě vedla změna vodivosti elektrolytu během celé separace. Při zvyšování odporu elektrolytu, které je zcela běžné, bude docházet vlivem udržení konstantního proudu ke zvyšování napětí. Při konci separace by mohlo dojít k překročení mezí,

které by byly vypočteny na začátku separace ze zadaných hodnot a došlo by k nežádanému vypnutí vysokého napětí.

Ke změnám hodnot mezí kromě automatického přenastavení jednou za 30 s dochází také, pokud uživatel změní některou z hodnot napětí nebo proudu.

Zde popsaná kontrola probíhá ve všech okamžicích separace s výjimkou doby *s* dle obrázku 4.

Pokud dojde k neplánovanému vypnutí vysokého napětí je vygenerováno hlášení obsahující důvod vypnutí

K testování, zda je daná proměnná v mezích jsem použil modul Limits.

```
Limits Vstup,VystupNad,VystupPod,DolniMez,HorniMez,hystereze,negace
```

Modul porovnává hodnotu *Vstup* s hodnotami *DolniMez* /*HorniMez*. Pokud dojde k prekročení jedné z nich, je nastaven bit *VystupNad* / *VystupPod* . Modul také umožňuje zadat pásmo necitlivosti a negaci testu.

Proces – osetreni

Proces obsahuje různá ošetření proti událostem, které by mohli způsobit pád nebo špatný chod systému.

Patří mezi ně např.

- záporné hodnoty jsou pomocí absolutní hodnoty převedeny na kladné
- nezapnutí poměru proudů při nulové hodnotě poměru
- nezapnutí pulsního režimu pokud jsou hodnoty t_1 a t_2 nulové
- nepovolení vysokého napětí pokud jsou hodnoty napětí a proudů nenulové
- a další ošetření které slouží pro vnitřní chod programu

Proces – Idle

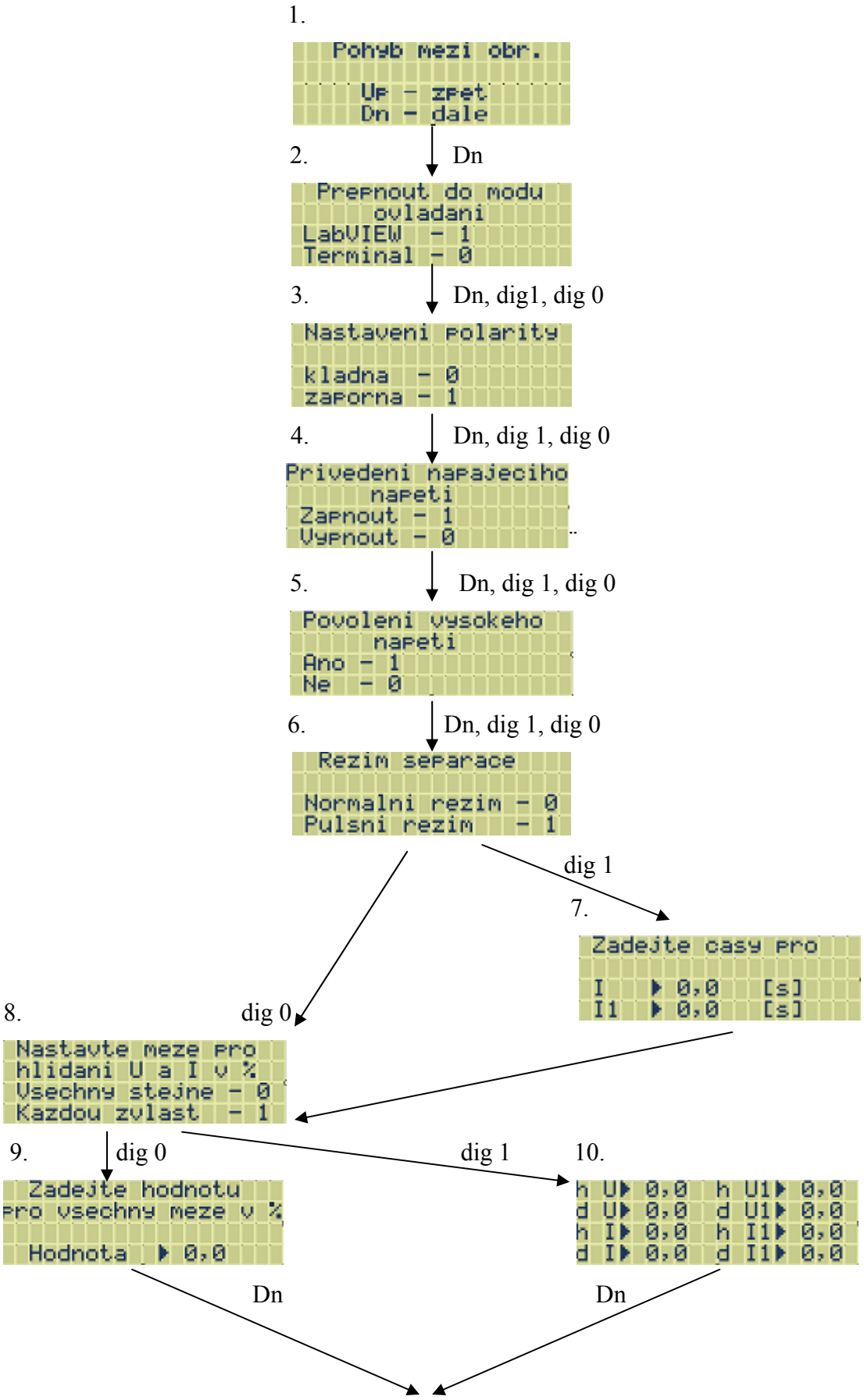
Proces Idle obsahuje pouze jeden modul - *Lcw3Idle*, který zajišťuje obsluhu terminálu.

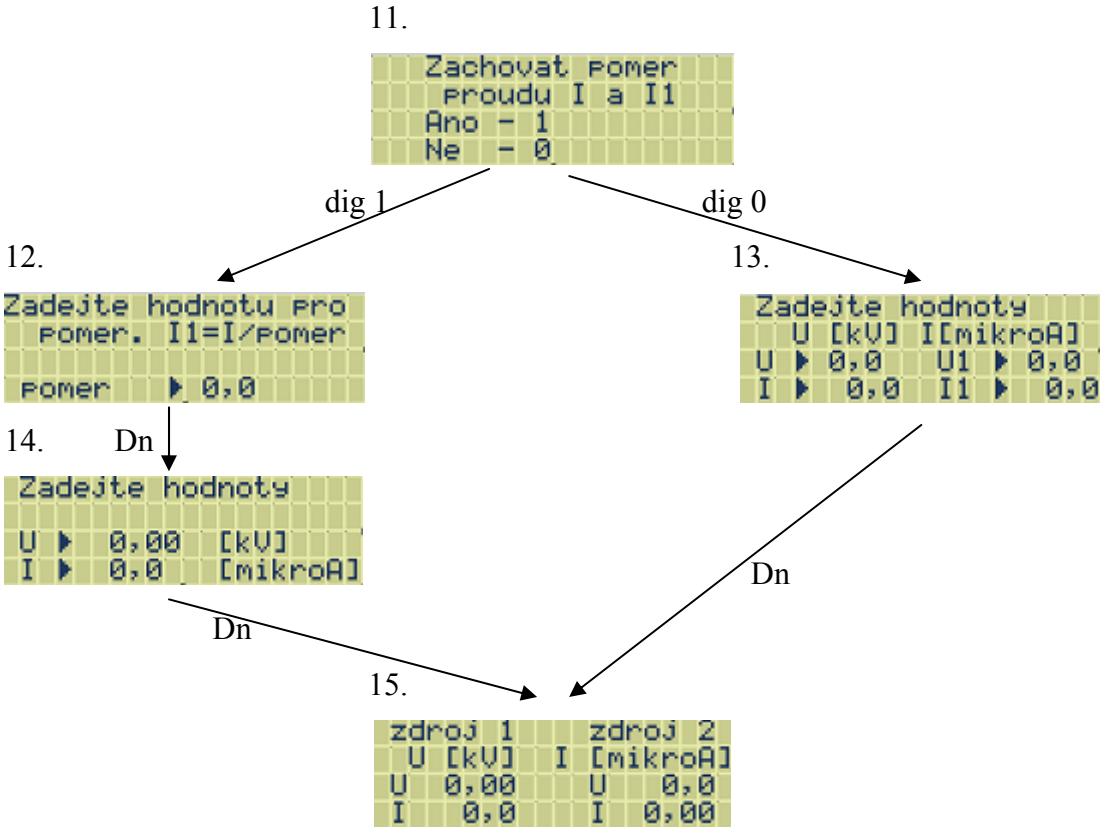
4.6 OBSLUHA POMOCÍ TERMINÁLU

Jak již bylo uvedeno v kapitole 4.1 program DetStudio obsahuje také editor obrazovek. Terminál, který je k dispozici má označení APT130, jedná se o terminál 4x20 se sériovým rozhraním. [2]

Návrh uživatelské aplikace, pomocí které lze ovládat PLC spočívá v umístění jednotlivých prvků na obrazovku, která je pak shodná s obrazovkou na terminálu. Prvky jsou na obrazovku přenášeny uživatelem z toolboxu. Každý prvek má své specifické vlastnosti, které lze měnit v editoru vlastností. Některé prvky umožňují reakci na změny v systému, k těmto prvkům pak může být přiřazen skript, který např. změní hodnotu proměnné, přepne na jinou obrazovku apod.

Pro ovládání pomocí tohoto terminálu byla navržena uživatelská aplikace, pomocí které lze nastavit veškeré hodnoty potřebné pro separaci. Pomocí poslední obrazovky lze také sledovat hodnoty nastavených napětí a proudů. Sekvence obrazovek spolu s ovládáním je na obrázku 5.


1.
Pohyb mezi obr.
Up - zpet
Dn - dale
2.
Dn
Prepnout do modu ovladani
LabVIEW - 1
Terminal - 0
3.
Dn, dig1, dig 0
Nastaveni polarity
kladna - 0
zaporna - 1
4.
Dn, dig 1, dig 0
Privedeni napajeciho napeti
Zapnout - 1
Vypnout - 0
5.
Dn, dig 1, dig 0
Povoleni vysokeho napeti
Ano - 1
Ne - 0
6.
Dn, dig 1, dig 0
Rezim separace
Normalni rezim - 0
Pulsni rezim - 1
dig 1
7.
Zadejte casy pro
I ▶ 0,0 [s]
I1 ▶ 0,0 [s]
8.
dig 0
Nastavte meze pro hlidani U a I v %
Vsechny stejne - 0
Kazdou zvlast - 1
9.
dig 0
dig 1
10.
Zadejte hodnotu pro vsechny meze v %
Hodnota ▶ 0,0
h U▶ 0,0 h U1▶ 0,0
d U▶ 0,0 d U1▶ 0,0
h I▶ 0,0 h I1▶ 0,0
d I▶ 0,0 d I1▶ 0,0
Dn
Dn



Obrázek 5: Sekvence obrazovek

Popis ovládání

Obrazovka č.1 je startovní a zobrazuje, jak se lze mezi jednotlivými obrazovkami pohybovat. Pokud nedochází k větvení obrazovek, dojde při stisku klávesy Down k přepnutí obrazovky na další v pořadí. Pokud obrazovka obsahuje některé z rozhodování, např. obrazovka č. 6, dochází k přepnutí na další obrazovku po stisku klávesy 0 nebo 1. Klávesa Down zde z důvodu rozhodování nefunguje. Obdobně se lze mezi obrazovkami pohybovat směrem zpět, stiskem klávesy Up. Po stisku této klávesy dojde k zobrazení předchozí obrazovky. Pokud dochází ke zpětnému kroku z obrazovky, na kterou je navázáno více obrazovek z předchozího rozhodování, dochází ke zpětnému skoku na nejbližší rozhodovací obrazovku.

Číslo obrazovky	Popis
1	Pohyb mezi obrazovkami: klávesou Down se pohybuje směrem dále a klávesou Up směrem zpět
2	Přepnout do modu ovládání: klávesa 1 zajistí společné ovládání z programu LabVIEW i terminálu. Klávesa 0 zajistí ovládání pouze z terminálu.
3	Nastavení polarity: klávesa 0 nastaví kladnou polaritu, klávesa 1 nastaví zápornou polaritu
4	Přivedení napájecího napětí: klávesa 0 - vypne, klávesa 1 - zapne
5	Povolení vysokého napětí: klávesa 0 – zakáže, klávesa 1 - povolí
6	Režim separace: klávesa 0 – normální režim, klávesa 1 – pulsní režim. Popis režimů viz. kapitola 5.2
7	Zadejte časy pro: obrazovka se zobrazí při pulsním režimu. Zadaná hodnota pro I = t1, I1 = t2 dle obr. 16. I – separace, I1 - pH
8	Nastavte meze pro hlídání U a I v %: klávesa 0 zajistí možnost zadat pro všechny meze stejnou hodnotu. Klávesa 1 zajistí zadání mezí pro každou hodnotu zvlášť.
9	Zadejte hodnotu pro všechny meze v %: nastaví veškeré meze na zadanou hodnotu.
10	h U–horní mez pro napětí U, d U–dolní mez pro napětí U, h I–horní mez pro proud I, d I–dolní mez pro proud I, h U1–horní mez pro napětí U1, d U1–dolní mez pro napětí U1, h I1–horní mez pro proud I1, d I1–dolní mez pro proud I1
11	Zachovat poměr proudů: klávesa 1 zajistí, že I1 = I/pomer, kde pomer bude zadán v následující obrazovce. Klávesa 0 zajistí zadání jednotlivých proudů. I – separace, I1 – pH
12	Zadejte hodnotu pro pomer. I1=I/pomer. Po předchozí žádosti na udržování proudů v daném poměru se proud I1 nastaví dle zadané hodnoty. Proud I1 a napětí U1 se již dále nenastavuje. I – separace, I1,U1 - pH
13	Zadejte hodnoty U[kV] I[mikroA]: nastavení jednotlivých hodnot napětí a proudů. U,I – separace, U1,I1 - pH
14	Zadejte hodnoty: nastavení U a I. U1, I1 se již nenastavuje. U,I – separace, U1,I1 - pH
15	zdroj 1 zdroj 2: zobrazení napětí a proudů pro oba zdroje. Zdroj 1 – separace, Zdroj 2 – pH.

Tabulka 6: Popis obrazovek

5. LABVIEW

Programy v LabVIEW se nazývají virtuální přístroje nebo VI. Svým vzhledem a činností jsou obdobou skutečných přístrojů, jako jsou např.osciloskopy a multimetry. VI je obdobou programování v jiných programovacích jazycích. K vytváření takových programů obsahuje LabVIEW nástroje pro čtení, analýzu, zobrazení a ukládání dat.

V LabVIEW se vytváří uživatelské rozhraní programu pomocí ovládacích prvků a indikátorů. Ovládací prvky jsou zde např. otočné knoflíky, tlačítka, stupnice a další vstupní zařízení. Indikátory jsou např. grafy, LED diody a jiné zobrazovače. Po vytvoření čelního panelu se přidá programový kód, který čelní panel řídí. Programový kód se zapisuje (kreslí) do okna blokového diagramu.

5.1 KOMUNIKACE LABVIEW – PLC

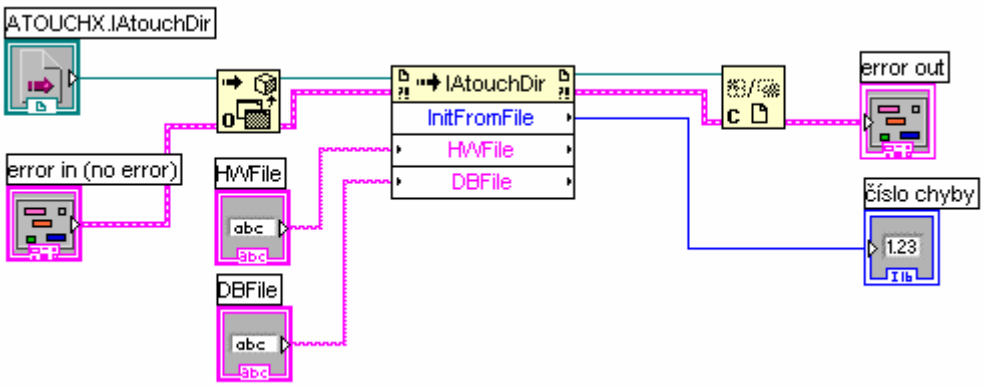
Jak vyplývá z kapitoly 3.2, pro komunikaci programu LabVIEW se systémem AMiRiS99 je využito knihovny ATOUCHX pracující s ActiveX prvky. V programu LabVIEW byly vytvořeny funkční bloky pro použití základních metod, které nám nabízí objekt AtouchDir. Jsou to např. metody inicializace, čtení databázových proměnných, zápis do databázových proměnných apod. Realizované metody jsou uvedeny v tabulce 5, kap. 3.2.1 (nebyly realizovány metody pro *správu připojení*, tyto metody slouží pro správu připojení více stanic v síti).

Dále uvedu popis všech realizovaných metod. K metodám použitým v při tvorbě výsledné aplikace v programu LabVIEW jsou také uvedeny jejich blokové diagramy.

InitFromFile, InitFromString

Obě metody provedou inicializaci objektu. Bez inicializace nelze pracovat s ostatními metodami objektu - skončí s chybou "neinicializováno". Výjimku tvoří metody pro zjištění verzí, pro zjištění informací a metody pro konverze času, které lze používat i bez inicializace. Obě metody vrací kód chyby.

InitFromFile provádí počáteční inicializaci ze souboru. [2]



Obrázek 6: Blokový diagram - metoda InitFromFile

HWFile obsahuje jméno a cestu k souboru, který popisuje HW připojení počítače k PLC.

Příklad obsahu souboru s HW připojením pro komunikaci s LabVIEW

```
[General]
MyStation=31            číslo pod kterým se PC připojuje k síti DbNet

[COM]
Com=1                   číslo COM portu počítače
Speed=19200             rychlost v Bd
Station=0               číslo stanice
```

DBFile obsahuje popis databáze, která obsahuje veškeré databázové proměnné. Popis je členěn do řádků, na každém z nich je popsána právě jedna proměnná ve tvaru:

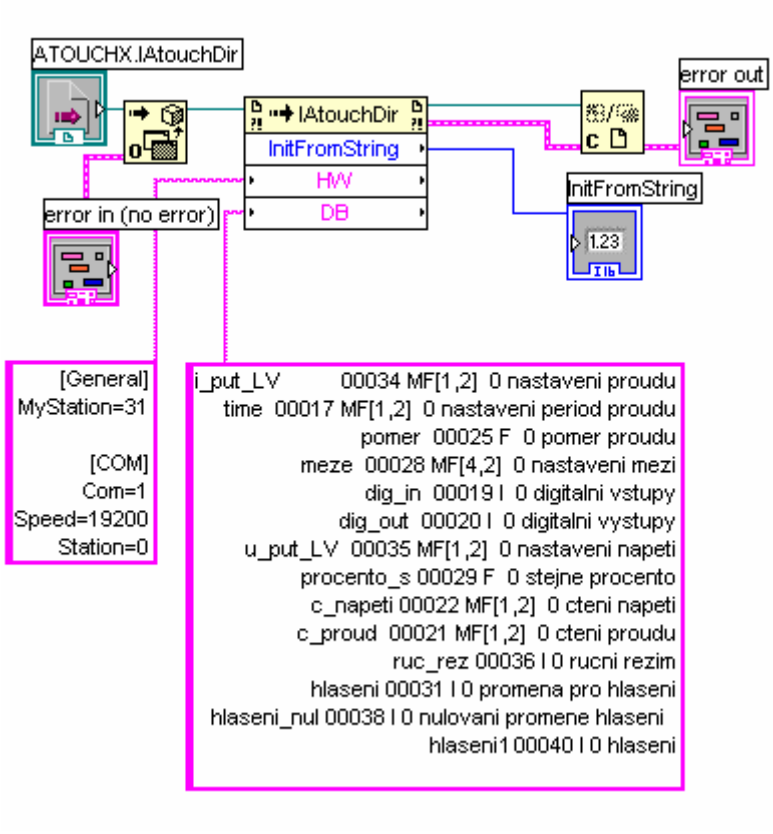
Jméno WID typ [řádky,sloupce] zdrojová_stanice komentář

Jméno – jméno proměnné
WID – WID dané proměnné
Typ – datový typ proměnné. Může být *I, L, F, MI, ML, MF*
[řádky,sloupce] – rozměr databázových matic – *MI, ML, MF*
zdrojová_stanice – číslo stanice obsahující danou proměnnou
komentář – nepovinný údaj, vlastní popis proměnné

Jako příklad zde uvedu obsah DB souboru pro inicializaci navrženého řídícího algoritmu.

i_put_LV	00034	MF[1,2]	0	nastaveni proudu
time	00017	MF[1,2]	0	nastaveni period proudu
pomer	00025	F	0	pomer proudu
meze	00028	MF[4,2]	0	nastaveni mezi
dig_in	00019	I	0	digitalni vstupy
dig_out	00020	I	0	digitalni vystupy
u_put_LV	00035	MF[1,2]	0	nastaveni napeti
procento_s	00029	F	0	stejne procento
c_napeti	00022	MF[1,2]	0	cteni napeti
c_proud	00021	MF[1,2]	0	cteni proudu
ruc_rez	00036	I	0	rucni rezim
hlaseni	00031	I	0	promena pro hlaseni
hlaseni_nul	00038	I	0	nulovani promene hlaseni
hlaseni1	00040	I	0	promena hlaseni1

Jak je vidět z obrázku 7, metoda InitFromString používá k inicializaci stejného zápisu HW a DB jako metoda InitFromFile, s tím rozdílem, že popis HW připojení a Db proměnných lze přímo vložit např. do konstantního textového pole.



Obrázek 7: Blokový diagram - metoda InitFromString

GetKernelVersion

Metoda vrací přímo číslo verze komunikačního jádra nebo objektu AtouchDir v BCD formátu tak, že horní bajt obsahuje číslo verze a spodní bajt obsahuje číslo podverze. Například verzi 1.60 odpovídá hodnota &H160. [2]

GetObjectVersion

Metoda vrací přímo číslo verze komunikačního jádra nebo objektu AtouchDir v BCD formátu tak, že horní bajt obsahuje číslo verze a spodní bajt obsahuje číslo podverze. Například verzi 1.60 odpovídá hodnota &H160. [2]

GetInfo

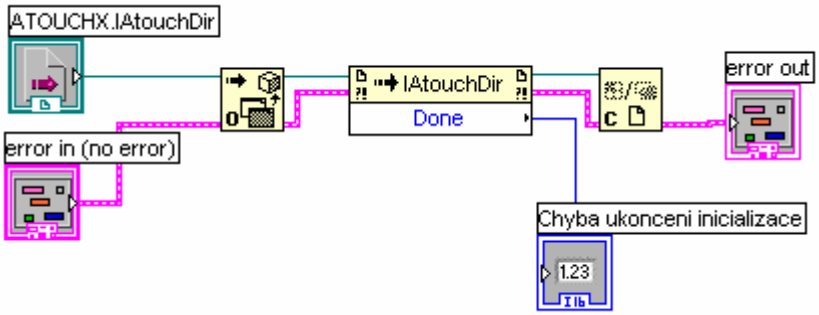
Metoda zjistí dostupné informace o komunikačním jádře a vytvoří proměnnou INFO jako jednorozměrnou matici o velikosti 1 prvek. Matice obsahuje verzi komunikačního jádra v BCD formátu tak, že horní bajt obsahuje číslo verze a spodní bajt obsahuje číslo podverze. Například verzi 1.60 odpovídá hodnota &H160. [2]

GetHWInfo

Metoda zjistí dostupné informace o HW připojení zadané stanice Station a vytvoří proměnnou INFO jako jednorozměrnou matici o velikosti 2 prvky. Matice INFO(0) obsahuje verzi HW ovladače poskytující připojení stanice Station v BCD formátu tak, že horní bajt obsahuje číslo verze a spodní bajt obsahuje číslo podverze. Například verzi 1.60 odpovídá hodnota &H160. Verze je obvykle shodná s verzí celého komunikačního jádra.Matice INFO(1) obsahuje typ HW připojení, např. &H200 odpovídá připojení pomocí modemu, &H400 odpovídá připojení pomocí COM, apod. [2]

Done

Metoda Done ukončuje připojení k síti DbNet. Uvolňuje všechny přidělené prostředky, a tím umožňuje komunikaci ostatním zařízením s požadovanou stanicí. Metoda se musí vyvolat vždy před zrušením objektu a to bez ohledu na úspěšnost inicializace. Metoda vždy vrací kód chyby 0 - vše dopadlo bez chyby. [2]



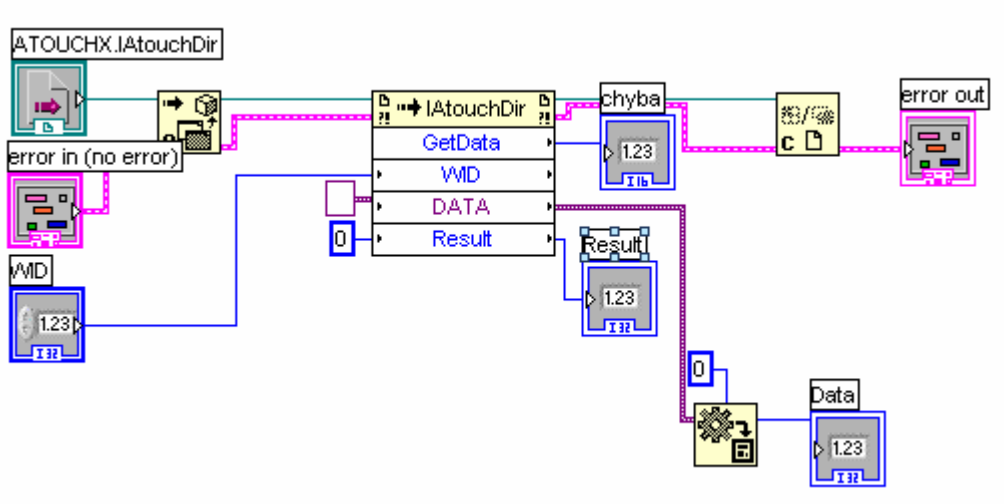
Obrázek 8: Blokový diagram - metoda Done

TimeFromDbNet, TimeToDbNet

Metoda TimeFromDbNet převede čas z DbNet formátu do VBA formátu a metoda TimeToDbNet převede čas z VBA formátu do DbNet formátu. Metody lze používat i bez inicializace připojení k síti DbNet. Informační systém DbNet uchovává datum a čas s přesností na 1s jako počet sekund od 1.1.1980. [2]

GetData

Metoda přečte obsah celé db. proměnné a vrátí její hodnotu v proměnné DATA. Čtení probíhá direktivně. Při direktivním způsobu připojení uživatel vyvolá komunikační metodu. Metoda mu zabezpečí přenos dat a počká na konec komunikace. Po návratu z metody má uživatel k dispozici případně čtená data a údaj, jak komunikace dopadla. Výsledek komunikace se ukládá do proměnné Result. [2]



Obrázek 9: Blokový diagram - metoda GetData

PutTime

Metoda zapíše nový čas do stanice. Čtení probíhá direktivně, výsledek komunikace se ukládá do proměnné Result. [2]

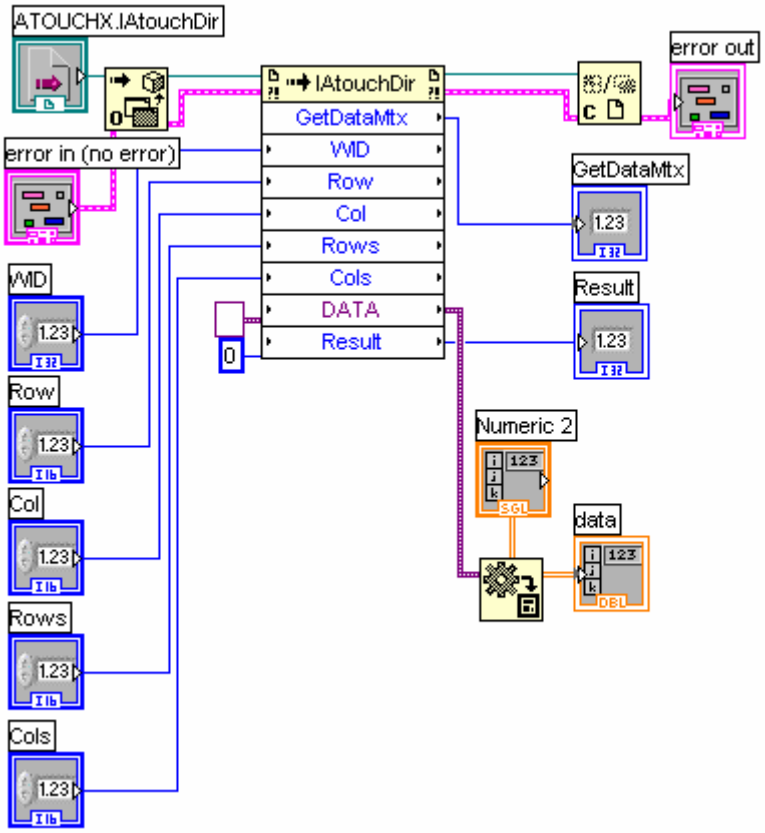
DbGetInfo

Metoda zjistí dostupné informace o db. proměnné zadané WIDem a vrátí je ve vytvořené matici INFO. [2]

GetDataMtx

Metoda přečte celou nebo část db. matice a vrátí její hodnotu v proměnné DATA. Čtení probíhá direktivně, výsledek komunikace se ukládá do proměnné Result.

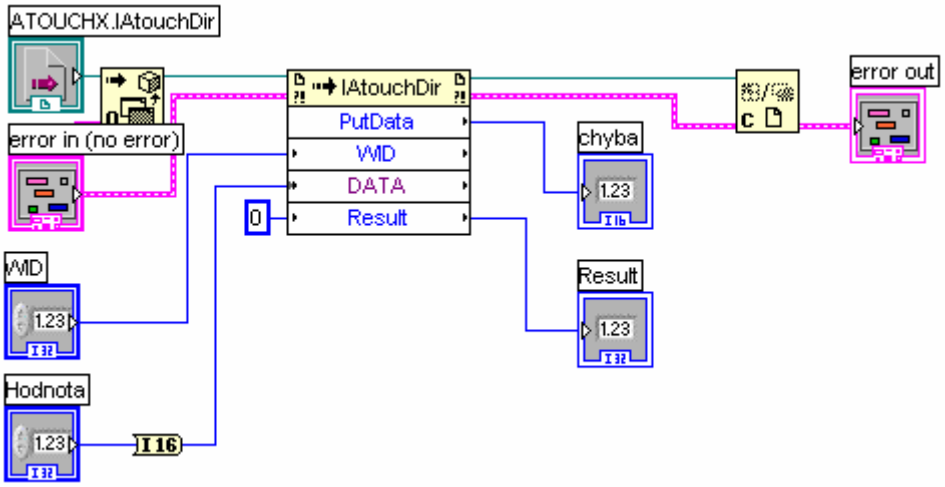
Pokud se čte db. matice (ať už celá nebo jen její část), metoda vytvoří proměnnou DATA jako dvourozměrnou matici. Prvky matice jsou typu odpovídající typu čtené db. proměnné. Pokud se čte db. matice, je proměnná DATA vždy indexována od prvku (0,0) bez ohledu, zda byla čtena celá db. matice nebo její část od počátku (od pozice (0,0)) nebo její část zevnitř. [2]



Obrázek 10: Blokový diagram - metoda GetDataMtx

PutData

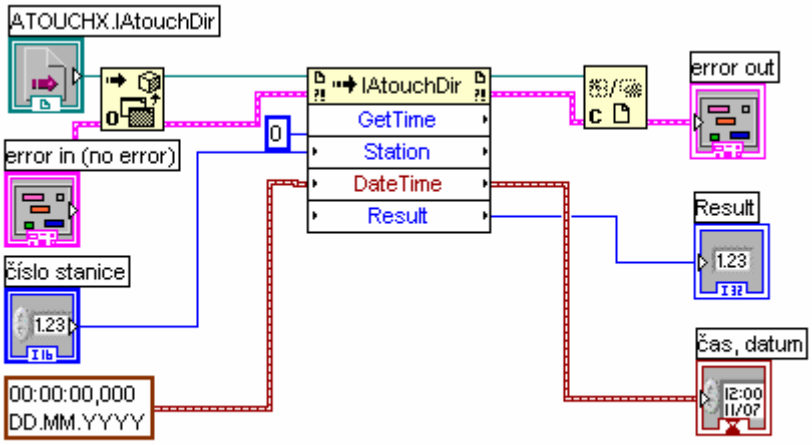
Metoda zapíše obsah celé db. proměnné. Zápis probíhá direktivně, výsledek komunikace se ukládá do proměnné Result. Zapisovaný obsah musí odpovídat zapisované db. proměnné. Pokud se zapisuje do jednoduché db. proměnné, musí být DATA jednoduchá hodnota. DATA musí být takového typu, aby bylo možno jeho hodnoty zapsat bez ztráty informace do db. proměnné. [2]



Obrázek 11: Blokový diagram - metoda PutData

GetTime

Metoda přečte čas stanice a vrátí jej v proměnné DateTime. Čtení probíhá direktivně, výsledek komunikace se ukládá do proměnné Result. [2]



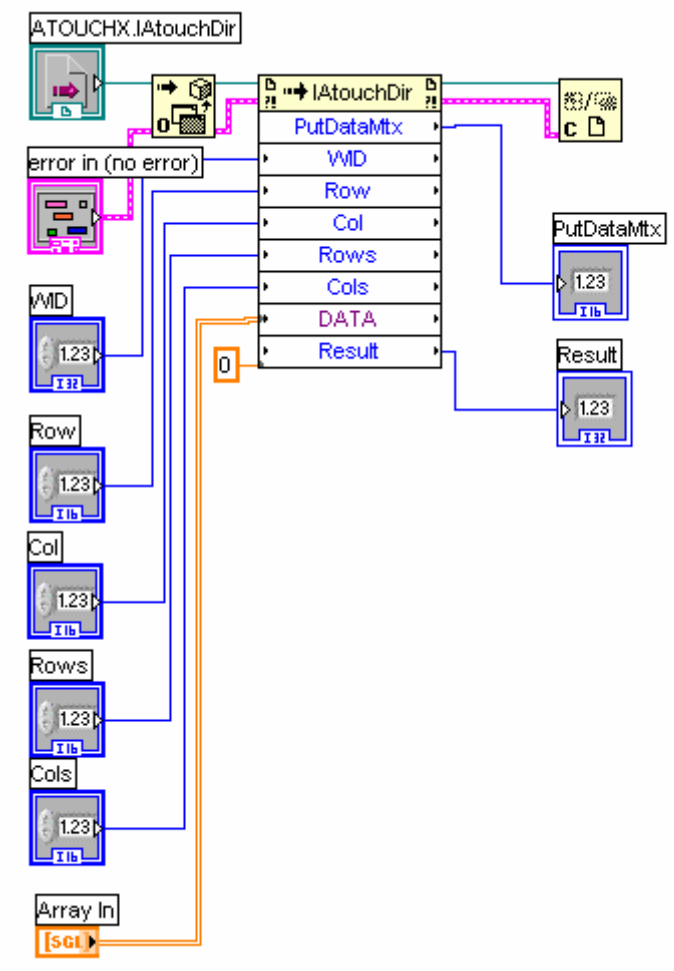
Obrázek 12: Blokový diagram - metoda GetTime

Identify

Metoda přečte aplikační nebo systémové identifikační řetězce stanice a vrátí je v proměnné DATA. Čtení probíhá direktivně, výsledek komunikace se ukládá do proměnné Result. [2]

PutDataMtx

Metoda zapíše obsah celé nebo část db. matice. Zápis probíhá direktivně, výsledek komunikace se ukládá do proměnné Result. Pokud se zapisuje matice (i její část), musí DATA být dvourozměrné pole, jehož počet řádků a sloupců odpovídá počtu zapisovaných řádků a sloupců. V tomto případě nezáleží na počátečních indexech DATA ale jen a pouze na rozměrech. DATA musí být takového typu, aby bylo možno jeho hodnoty zapsat bez ztráty informace do db. proměnné. [2]



Obrázek 13: Blokový diagram - metoda PutDataMtx

5.2 LABVIEW – VI

Jak již bylo naznačeno v úvodu kapitoly 5, software LabVIEW k programování využívá grafický editor. Tento editor se skládá ze dvou částí, je to Front panel, zde se umísťují ovládací a indikační prvky jako jsou např. tlačítka nebo různé stupnice, světelné indikátory apod.. V tomto panelu jsem navrhl vzhled aplikace pomocí které je možno ovládat, nastavovat hodnoty pro PLC a také sledovat nastavené hodnoty pomocí indikačních prvků a grafů.

Druhou částí grafického editoru je okno blokový diagram, kde dochází k propojení jednotlivých ovládacích a indikačních prvků z Front panelu pomocí funkčních bloků. Tyto funkční bloky zajišťují různé výpočty, komunikaci s daným systémem apod..

K zajištění komunikace s PLC AMiRiS99 jsem zde použil již dříve navržené funkční bloky, jejichž popis spolu s blokovými diagramy je uveden v kapitole 5.1. Veškeré navržené funkční bloky byly implementovány do jedné knihovny, kterou dále využívá výsledná aplikace.

V další části této kapitoly uvedu použité struktury a bloky spolu s jejich popisem a blokovými diagramy, které jsem při tvorbě aplikace použil. Jedná se hlavně o popis jednotlivých řídících a událostních struktur. Základní strukturou použitou při tvorbě aplikace byla tzv. Event Structure, která dále obsahuje Case Event - struktury reagující na určitou událost. V aplikaci jsou to události jako je změna hodnoty ovládacího prvku, stisk klávesy nebo tlačítka myši a v poslední řadě událost Timeout, která zajišťuje periodické provádění funkčních bloků umístěných v dané struktuře. Jako příklad je od každé zmíněné události uveden blokový diagram spolu s krátkým popisem.

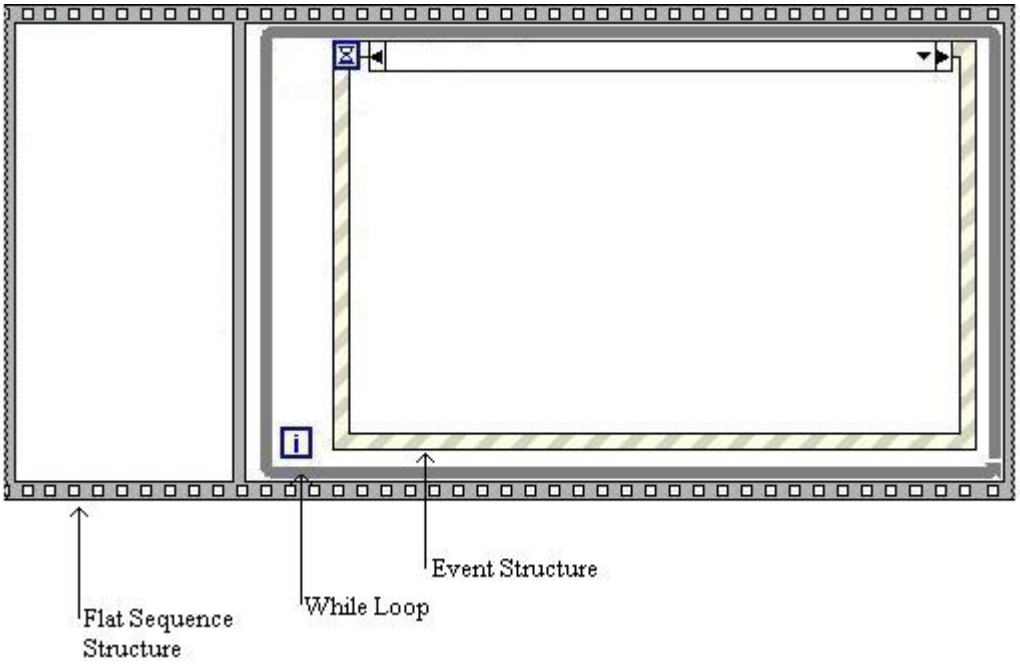
Struktury

Veškeré funkční bloky zajišťující komunikaci s PLC jsou umístěny ve strukturách zobrazených na obrázku 14.

Flat Sequence Structure zajišťuje provedení funkčních bloků v dané sekvenci. První okno obsahuje bloky, které musí být spuštěny jako první a pouze jednou, patří sem např. blok inicializace.

Struktura While Loop zde zajišťuje funkci nekonečné smyčky a jsou zde umístěny veškeré ovládací a indikační bloky.

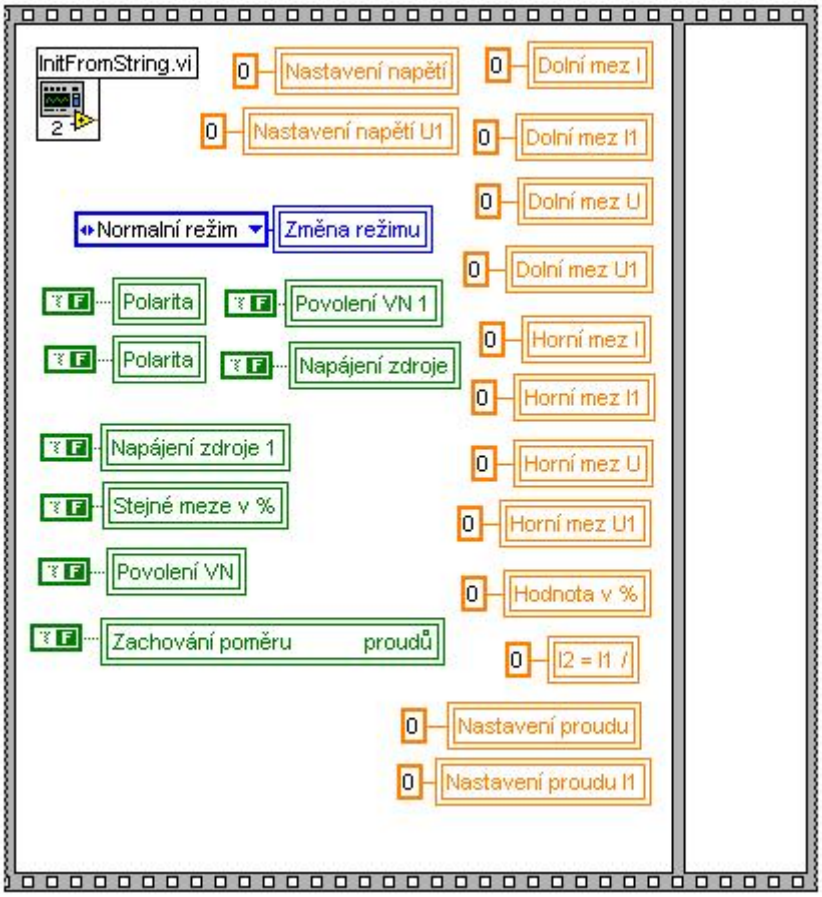
Event Structure zajišťuje spouštění jednotlivých bloků při dané události. Výhodou použití této struktury je, že nedochází k neustálému vysílání a čtení dat mezi PC a PLC a tím dochází k menšímu zatížení PC. Struktura obsahuje celkem 11 událostních struktur, některé z nich budou popsány dále.



Obrázek 14: Blokový diagram – struktury

Inicializace

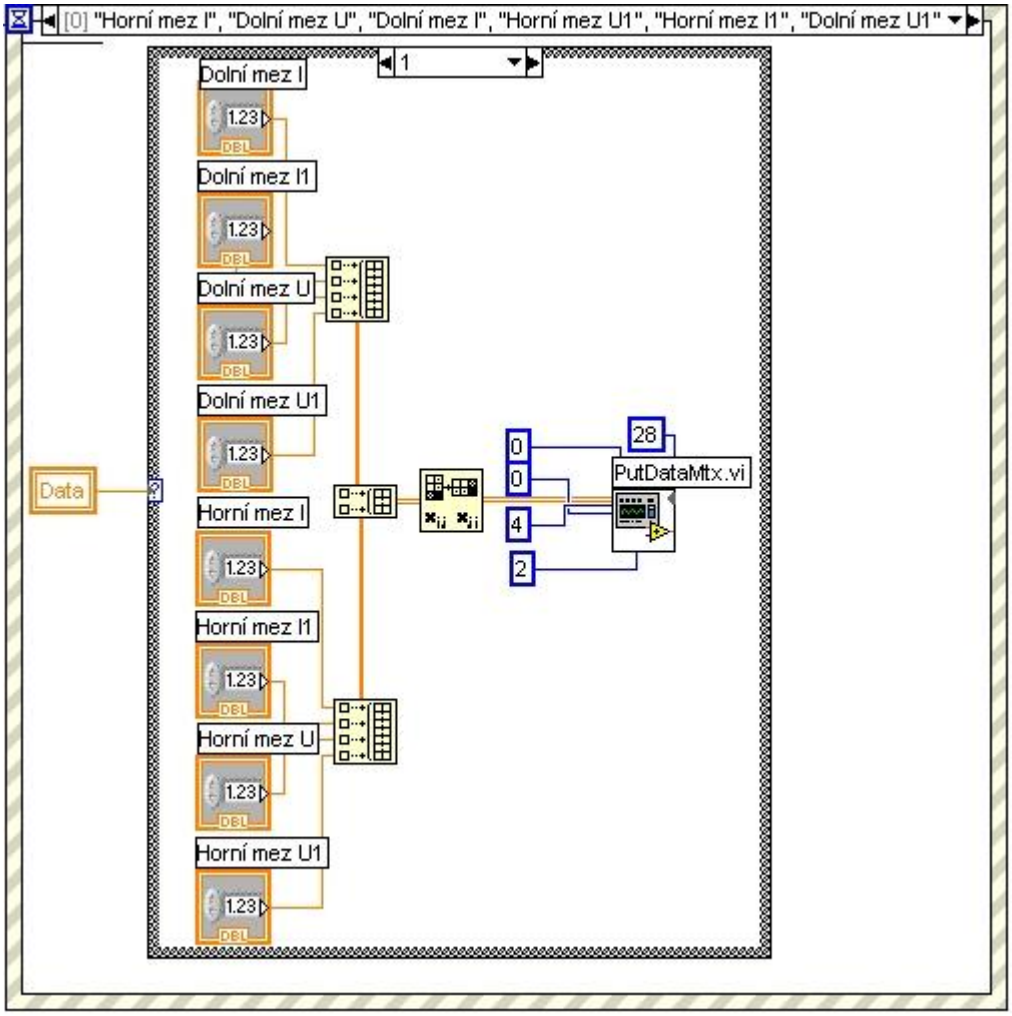
Jak již bylo uvedeno Flat Sequence Structure zajišťuje postupné vykonávání funkčních bloků. První okno této struktury obsahuje SubVI – InitFromString, které zajistí počáteční inicializaci komunikačního jádra, musí být spuštěno pouze jednou, při opakovaném spuštění skončí s chybou – komunikační jádro bylo již inicializováno. Dále jsou v tomto okně umístěny lokální proměnné ke všem ovládacím a indikačním prvkům, které zajišťují nulování těchto prvků při spuštění aplikace.



Obrázek 15: Blokový diagram - Inicializace

Event Structure – změna hodnoty

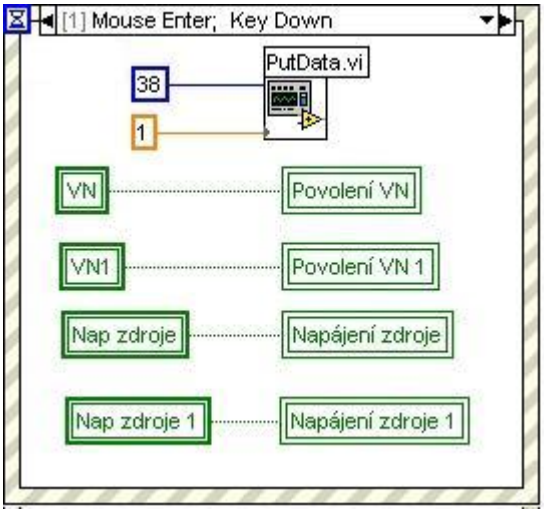
Událostní struktura číslo 0 obsahuje funkční bloky pro nastavení jednotlivých mezí spolu s SubVI - PutDataMtx zajišťující vyslání maticové proměnné. Struktura je nastavena na události změny některé z hodnot mezí. Tyto bloky jsou ještě navíc umístěny v Case struktuře, která zamezuje vysílání hodnot v případě, že PLC je v režimu ručního ovládání.



Obrázek 16: Blokový diagram: Event Structure – změna hodnoty

Event Structure – klávesa, tlačítko myši

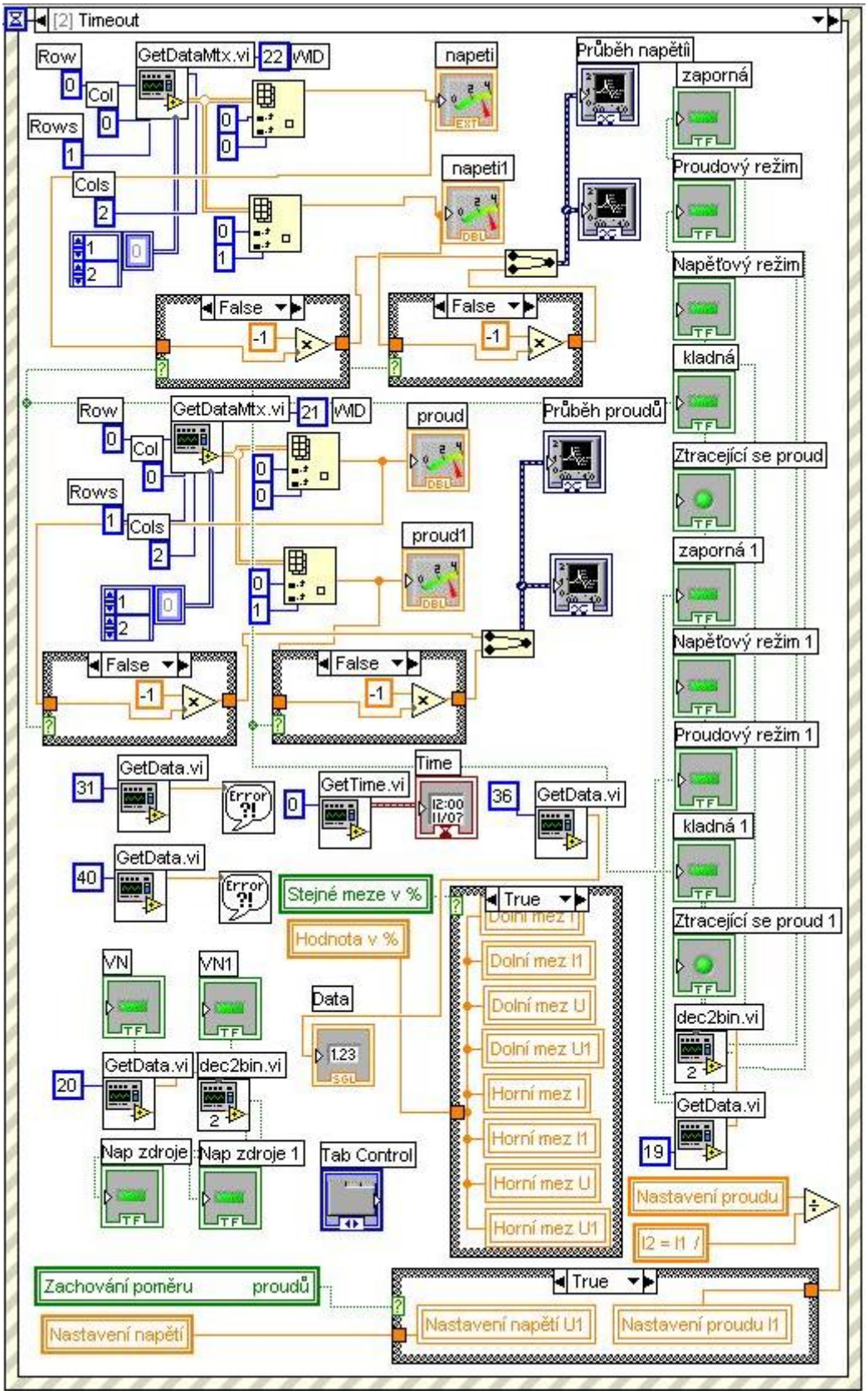
Událostní struktura 1 je nastavena dvě události, kliknutí myší nebo zmáčknutí klávesy. Obsahuje SubVI PutData, které zajišťuje vyslání hodnoty 1 do proměnné nulování hlášení při stisku klávesy nebo tlačítka myši, a tím zamezí opětovné vyslání stejného hlášení. Dále obsahuje lokální proměnné, které zajistí správné nastavení ovládacích prvků dle skutečného nastavení v PLC.



Obrázek 17: Blokový diagram: Event Structure – klávesa, tlačítko myši

Event Structure - Timeout

Událostní struktura 2 je nastavena na reakci Timeout, znamená to, že struktura je prováděna v pravidelné periodě. Struktura obsahuje veškeré indikační prvky které jsou v aplikaci použity. Dochází zde tedy ke čtení databázových proměnných a jejich následné indikaci v 100 ms periodě.

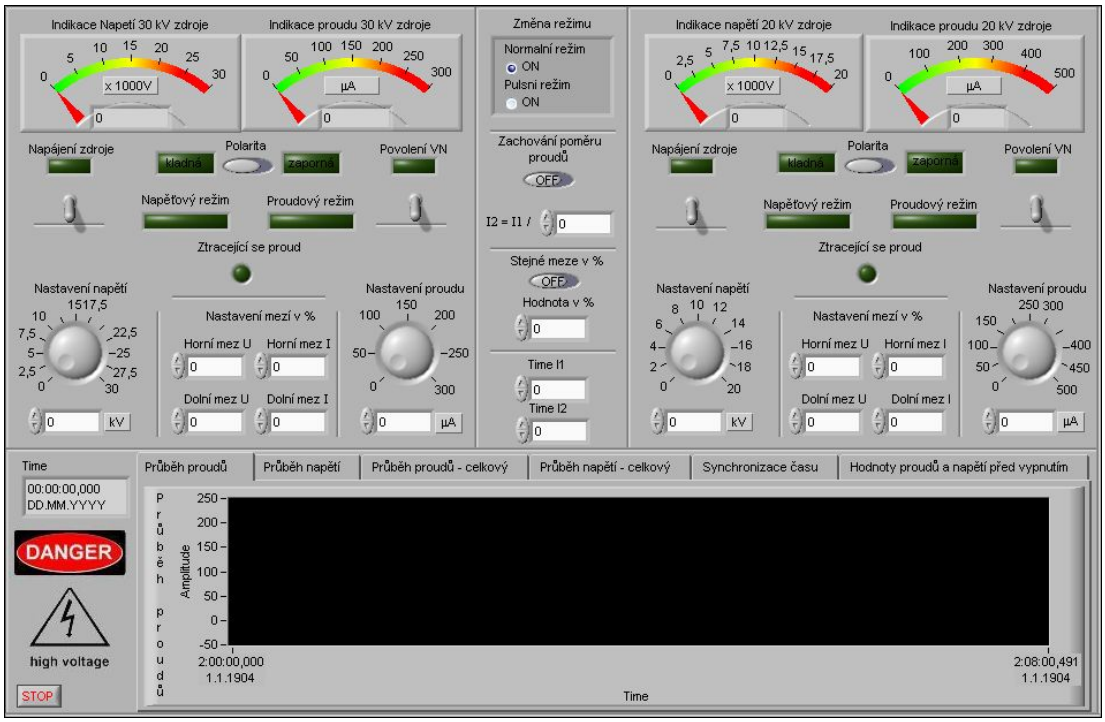


Obrázek 18: Blokový diagram: Event Structure – Timeout

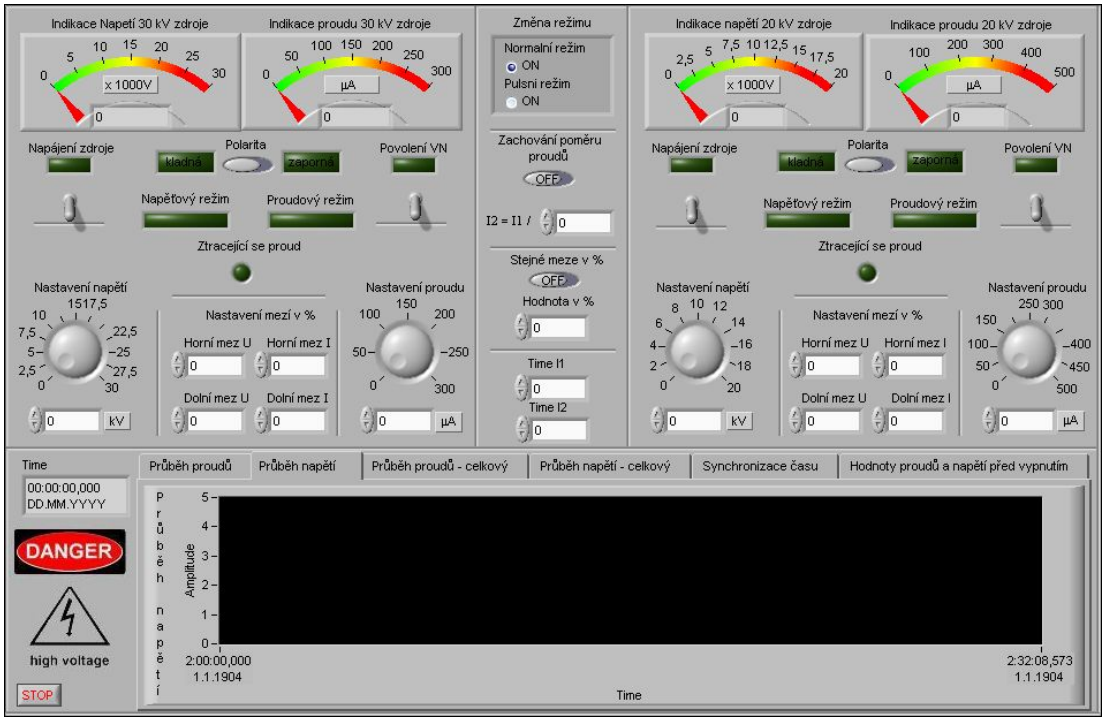
Front panel

Na obrázku 19 je vzhled navrženého čelního panelu. Jak je vidět panel je rozdělen na dvě části. Levá část slouží k ovládání zdroje - 30kV a pravá část k ovládání zdroje – 20kV. Střední část je zaměřena na nastavování hodnot společných pro oba zdroje.

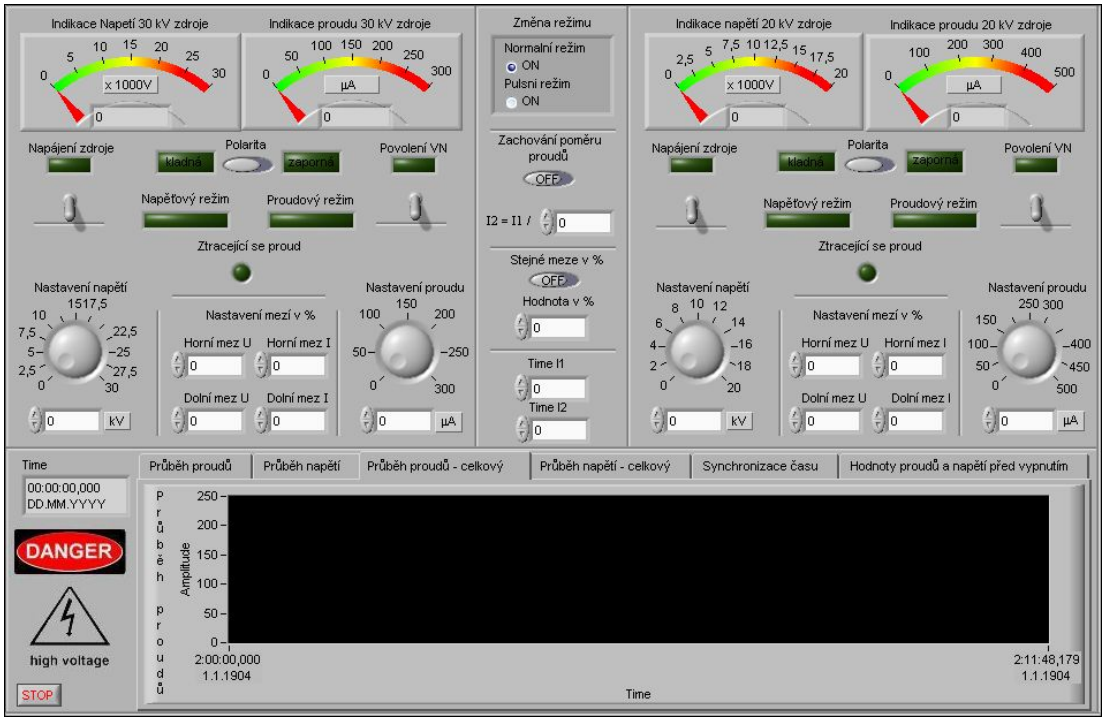
Spodní část panelu obsahuje záložkové menu. Záložky jedna až čtyři obsahují grafické znázornění průběhu proudů a napětí (I1 – I2, U1 – U2). U záložky jedna a dva je to průběh za určitý časový interval a u záložek tři a čtyři za celou dobu separace. Pátá záložka obsahuje přepnutí ovládání pomocí terminálu a synchronizaci času PC – PLC. V poslední položce je obsaženo pole, do kterého jsou ukládány hodnoty napětí a proudů pokud dojde k neplánovanému vypnutí vysokého napětí při překročení nastavených mezí. Záložky jsou postupně ukázány na obrázcích 19 až 25.



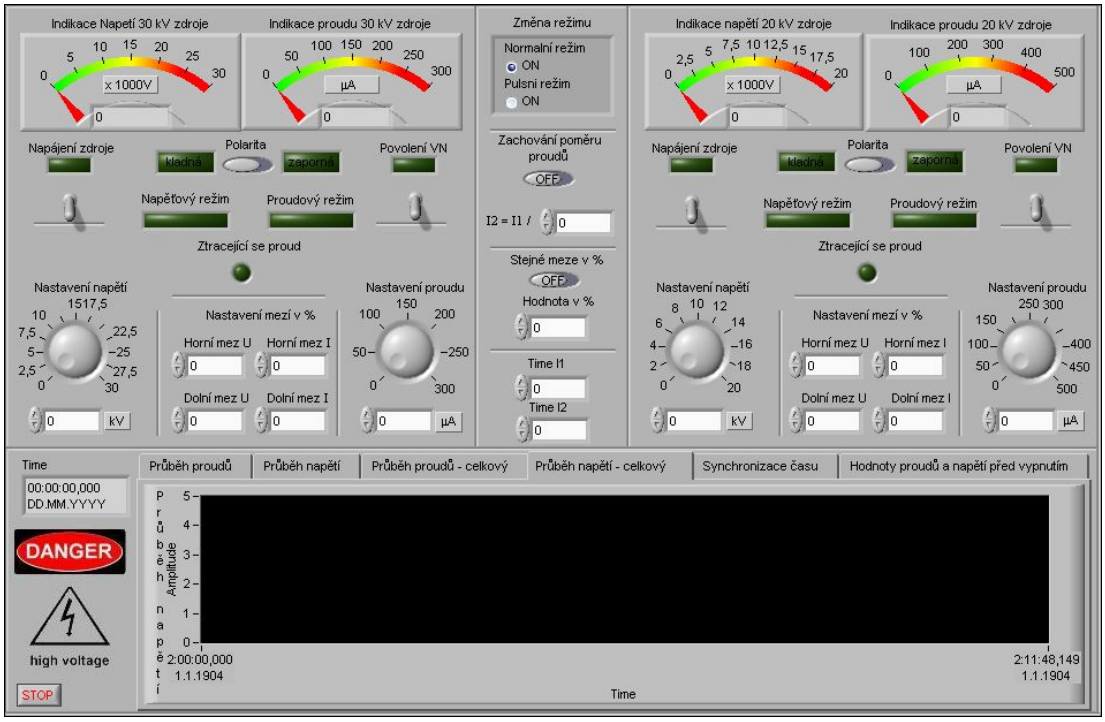
Obrázek 19: Front panel Z1



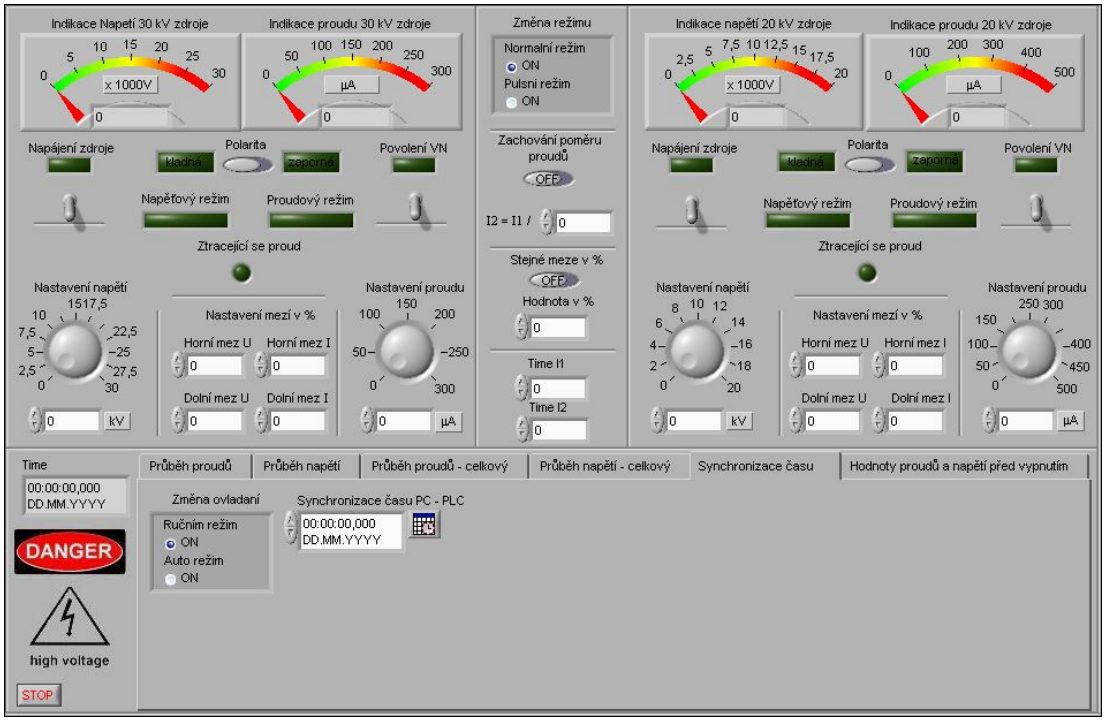
Obrázek 20: Front panel Z2



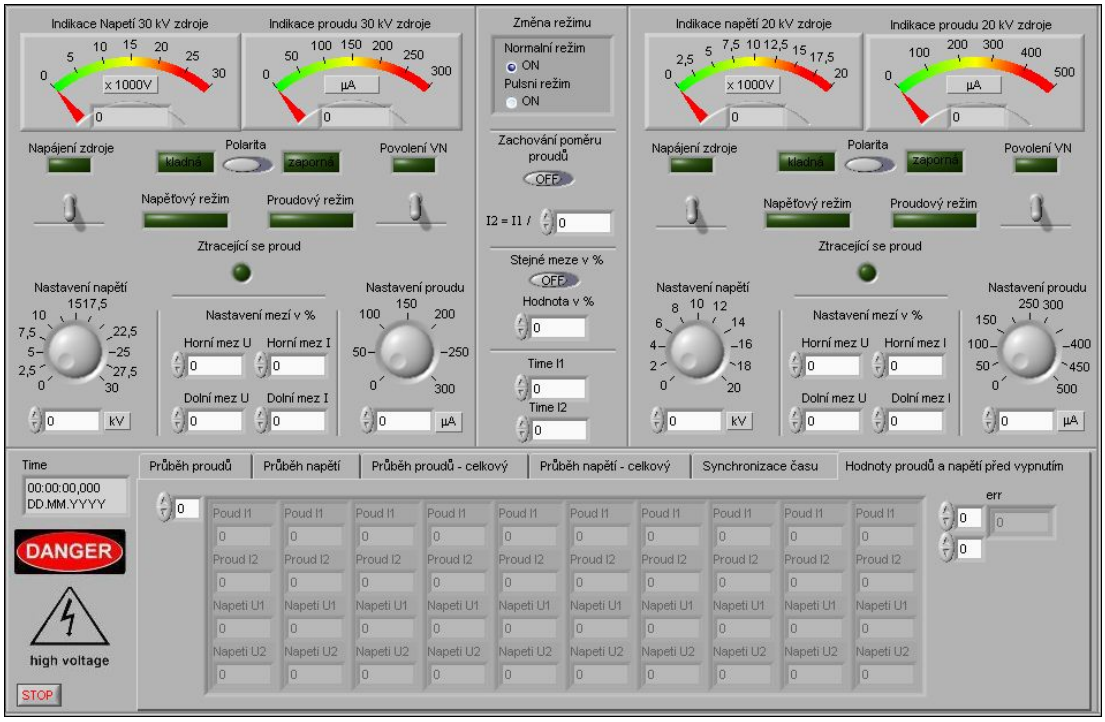
Obrázek 21: Front panel Z3



Obrázek 22 : Front panel Z4



Obrázek 23: Front panel Z5



Obrázek 24: Front panel Z6

6. SEPARACE CHEMICKÝCH VZORKŮ

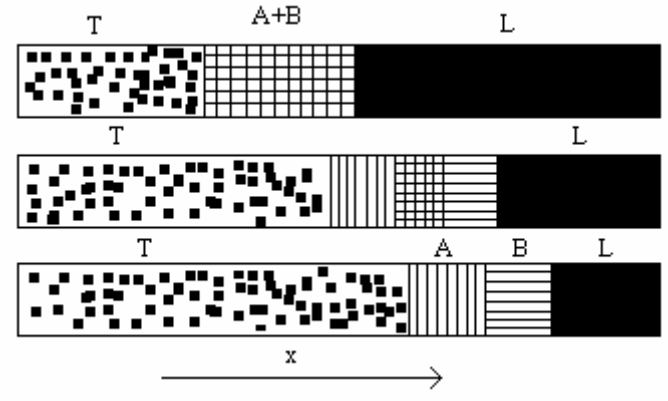
6.1 KAPILÁRNÍ IZOTACHOFORÉZA

Metodou na testování celého systému byla zvolena kapilární izotachoforéza. Izotachoforéza se liší od ostatních elektroforetických metod tím, že vzorek je vnášen mezi dva různé elektrolyty – vedoucí (leading L) a koncový (terminátor T). [5]

V jednom experimentu mohou být separovány pouze ionty jediného znaménka, buď anionty nebo kationty. Po připojení elektrického pole na systém začne probíhat izotachoforetický proces. [5]

Průběh lze rozdělit na dvě části. První část, kde dochází k separaci složek vzorku – migrační rychlosti jednotlivých částic ve směsné zóně jsou různé. V druhé části se již částice pohybují stejnou rychlostí – ustálená stav. [5]

Separace směsi dvou složek A a B je ukázána na obr. 26.



Obrázek 25: Izotachoforéza [6]

V průběhu separace se rychlejší částice oddělují ze směsné zóny a vytvářejí tak čistou zónu složky A mezi směsnou zónou a vedoucím elektrolytem. V zadní části směsné zóny vzorku se oddělují pomalejší částice a vzniká tak čistá zóna složky B mezi zónou směsi a koncovým elektrolytem. V dalším průběhu se čisté zóny prodlužují, směsná zóna se zkracuje a po určité době dojde ke kompletní separaci části A a B, kdy směsná zóna úplně vymizí. Ustáleného stavu je dosaženo, když oddělené zóny s ostrými hranami následují jedna za druhou. Od této doby se všechny zóny pohybují konstantní rychlostí. [5]

6.2 REÁLNÁ SEPARACE CHEMICKÝCH VZORKŮ

Celkem byla provedena separace pěti různých chemických vzorků. Separace probíhala pouze s jedním vysokonapěťovým zdrojem 30kV. Zdroj 20kV, který je určen na změnu pH nebyl do procesu separace zapojen z důvodu nekompletnosti separační aparatury.

Délka kapiláry byla 210 mm, vnější průměr 1 mm a vnitřní průměr 0,8 mm. Průměrná doba všech pěti separací byla 20 minut. V další části této kapitoly uvedu naskenované kapiláry spolu s ukázkou ovládacího panelu v LabVIEW. Tyto obrázky byly postupně skenovány v průběhu separace. První ze čtyř kapilár je vždy počátek separace a poslední kapilára zobrazuje proces u konce. Z čelního panelu jsem vždy vybral jeden nejzajímavější. Zobrazují aktuální nastavení řídicího systému a průběh napětí nebo proudů během separace.

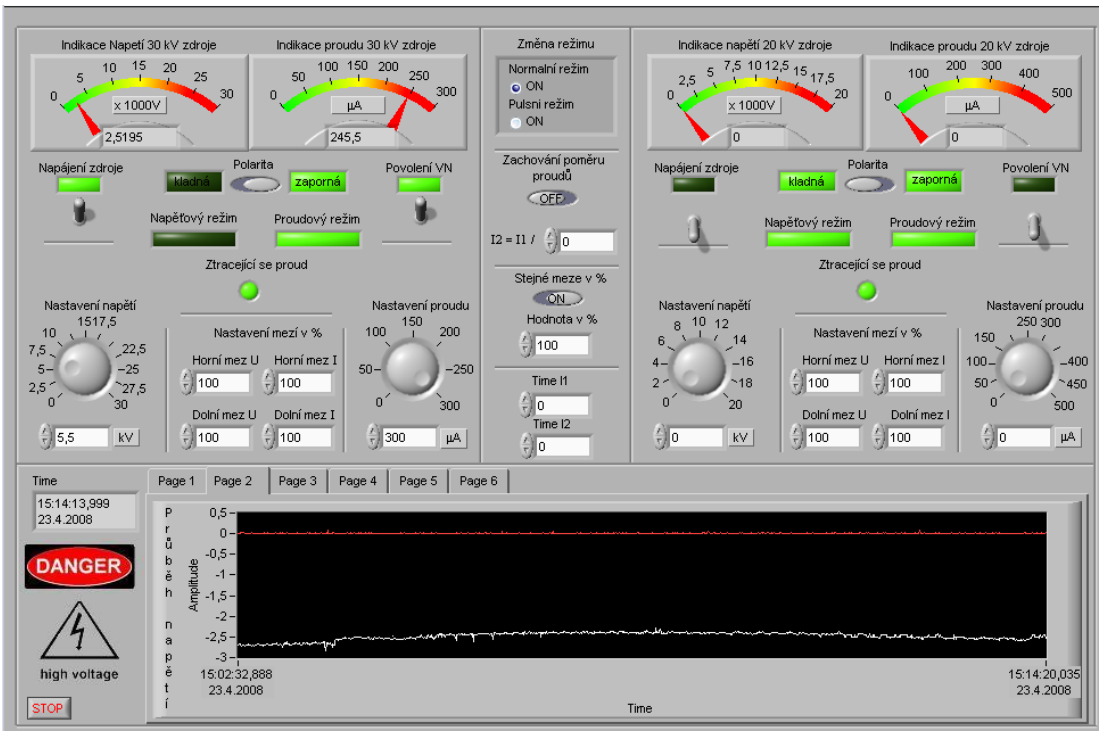
Složení chemických vzorků

č.	Terminátor	Leading
1	kyselina octová(0.02 mol)	červené barvivo SPADNS (0,01 mol)
2	kyselina octová(0.02 mol)	octan amonný(0,01mol) + hydroxid amonný(0.01mol) + bromthymolova modř(0.001mol)
3	kyselina octová(0.02 mol)	octan amonný(0,01mol) + hydroxid amonný(0,01mol) + bromthymolova modř(0.001 mol) + methyl
4	kyselina octová(0.02 mol)	octan amonný(0,01mol) + hydroxid amonný(0,01mol) + bromthymolova modř(0.001mol) + feroin
5	kyselina octová(0.02 mol)	octan amonný(0,01mol) + hydroxid amonný(0,02mol) + kresolftalexon(0,001 mol) + vápník(0.0001mol)

Tabulka 7: Složení použitých chemických vzorků

Separace č.1

Obrázek 26 ukazuje nastavení hodnot, které byly použity při první separaci. Polarita zde byla zvolena záporná a hodnota napětí a proudu nebyla během celé separace měněna.



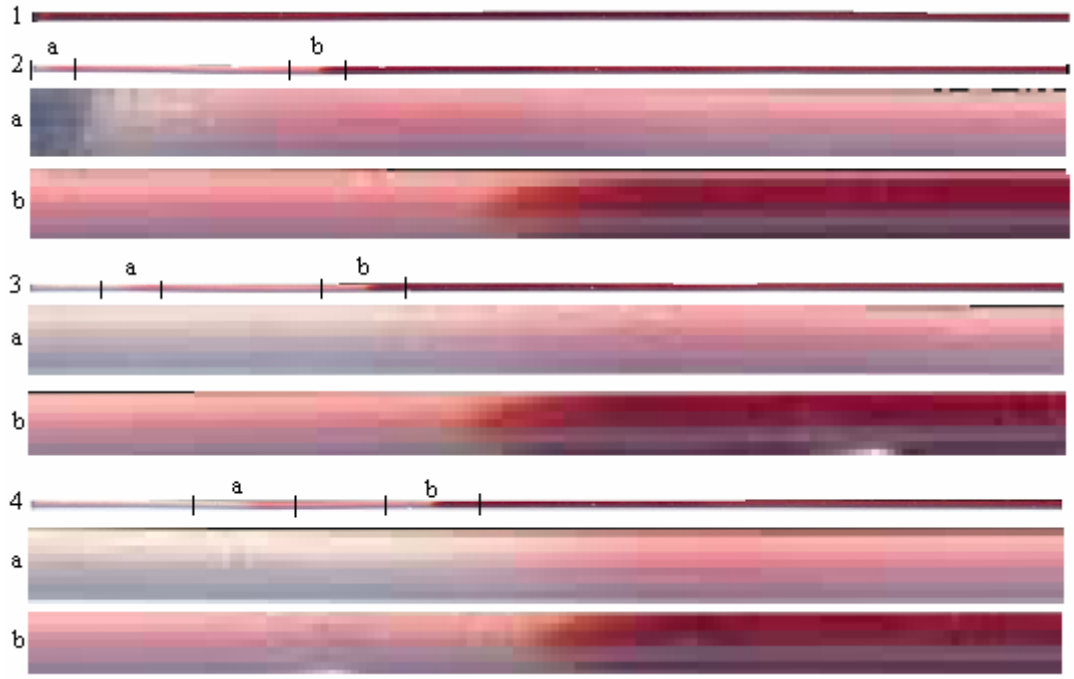
Obrázek 26: Front panel s1

Obrázek 27 ukazuje neskenované kapiláry z první separace. Kapilára číslo 1 zobrazuje začátek separace, obsahuje pouze leading.

Druhá kapilára obsahuje již částečně separované zóny. Pro větší přehlednost jsem přechody zvětšil, jsou to části *a* a *b* . I když leading se skládá pouze z červeného barviva SPADNS jsou zde vidět dvě zóny. Je to způsobeno tím, že použité barvivo obsahuje dvě složky které jsou od sebe separovány.

Ve třetí kapiláře dochází k prodloužení zón. Poslední, čtvrtá kapilára již zobrazuje dokončenou separaci kdy dochází pouze k posunu jednotlivých zón.

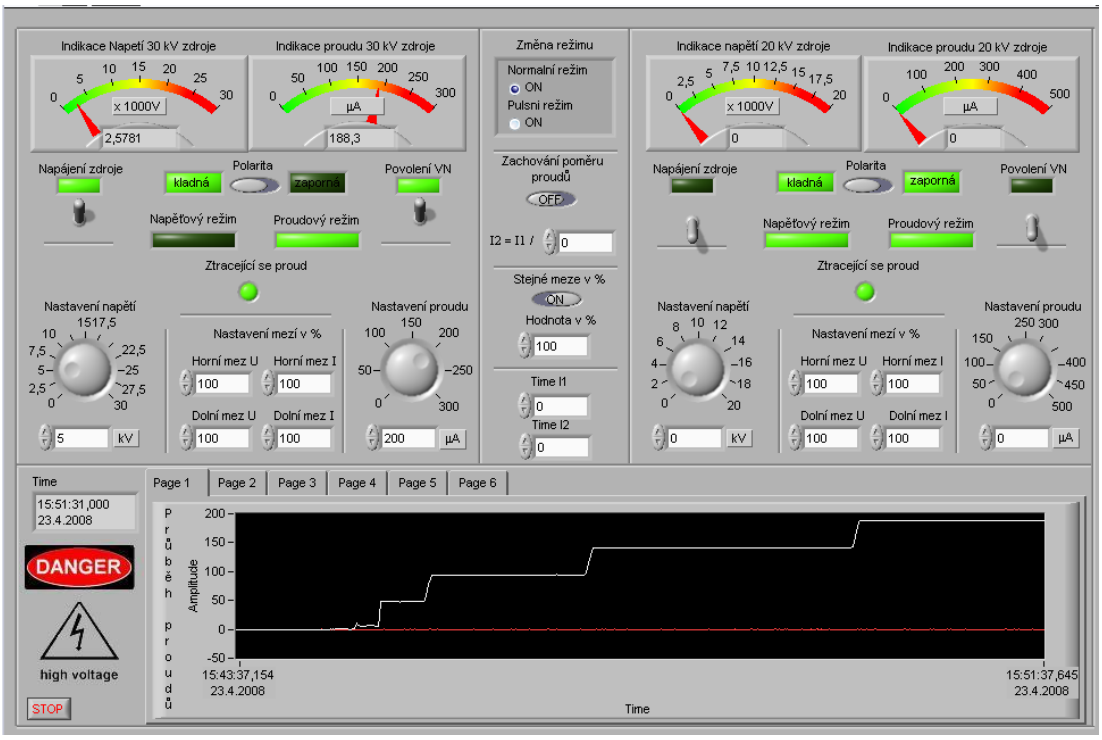
terminátor - kyselina octová(0.02 mol),
leading - červené barvivo SPANDS(0,01 mol)



Obrázek 27: Kapiláry s1

Separace č.2

Obrázek 28 ukazuje nastavení hodnot, které byly použity při druhé separaci. Polarita zde byla zvolena kladná. Hodnota napětí a proudu byla na počátku separace nastavena na 5 kV. Proud byl v průběhu separace několikrát zvýšen, což zajistilo rychlejší separaci.



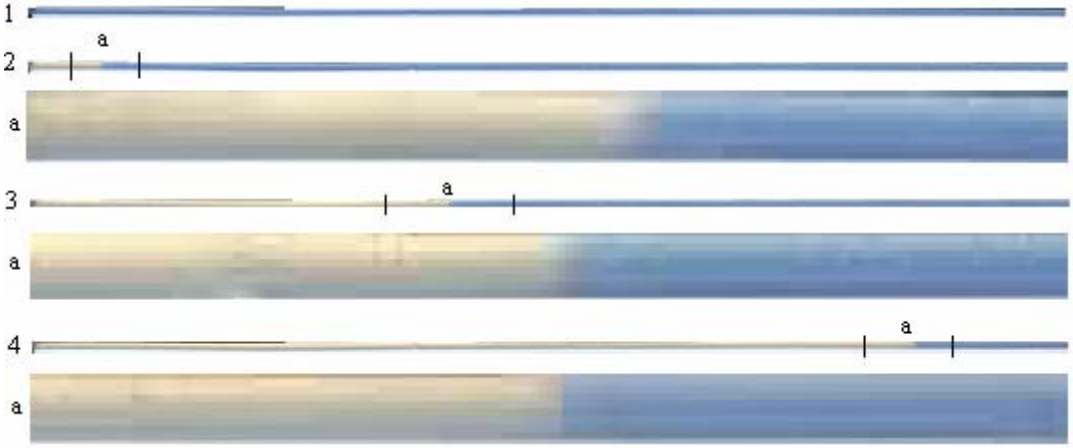
Obrázek 28: Front panel s2

Obrázek 29 ukazuje neskenované kapiláry ze druhé separace. Kapilára číslo 1 opět zobrazuje počátek separace.

V kapiláře 2 jde již vidět že již dochází k separaci složek. Ve třetí kapiláře dochází k dalšímu prodlužování zóny. Je zde vidět, že přechod mezi jednotlivými složkami není ještě zcela čistý.

Ve třetí kapiláře dochází již k téměř úplné separaci. Poslední, čtvrtá kapilára již obsahuje ostrý přechod mezi jednotlivými zónami.

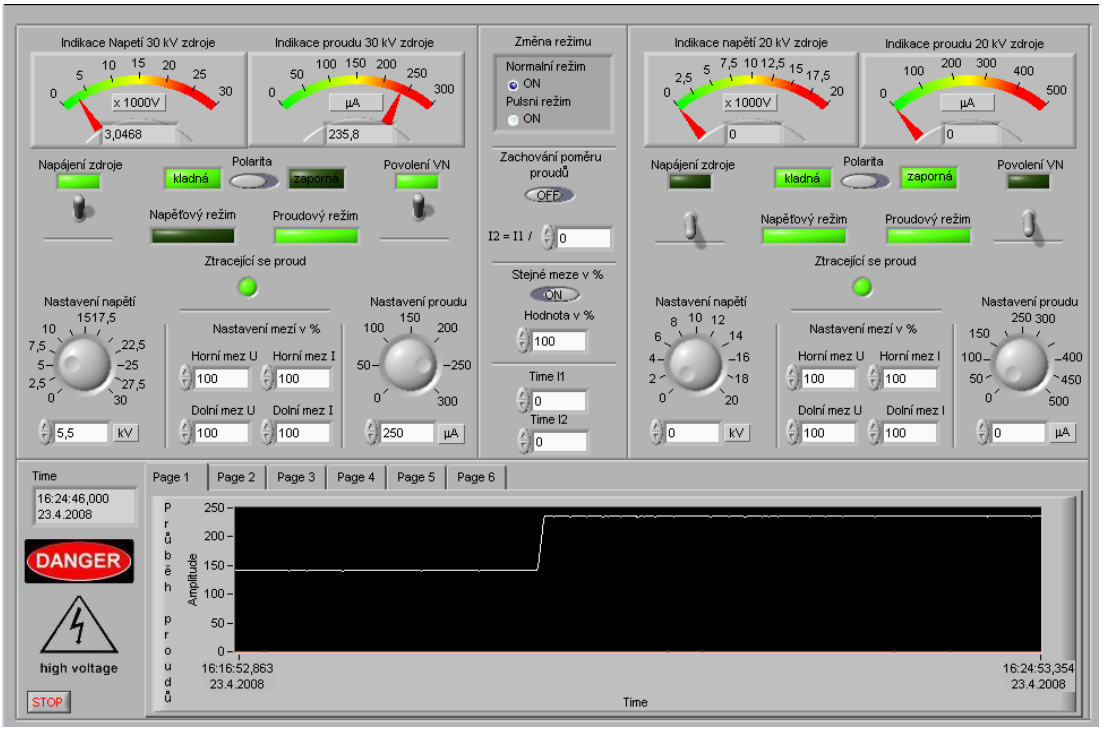
terminátor - kyselina octová(0.02 mol),
leading - octan amonný(0,01mol) + hydroxid amonný(0.01mol) + bromthymolova modř(0.001mol)



Obrázek 29: Kapiláry s2

Separace č.3

Obrázek 30 ukazuje nastavení hodnot, které byly použity při třetí separaci. Polarita zde byla opět zvolena kladná. Hodnota napětí a proudu byla na počátku separace nastavena na 5,5 kV. Proud byl v průběhu separace jedenkrát zvýšen z hodnoty 150 µA na 250 µA, opět pro urychlení děje.



Obrázek 30: Front panel s3

Třetí experiment obsahuje stejné složky jako separace číslo 2, pouze s přidaným methylem. Na první kapiláře je opět vidět počátek procesu.

Ve druhé kapiláře je vidět, že již dochází k částečné separaci složek. Ve třetí kapiláře se tyto složky prodlužují.

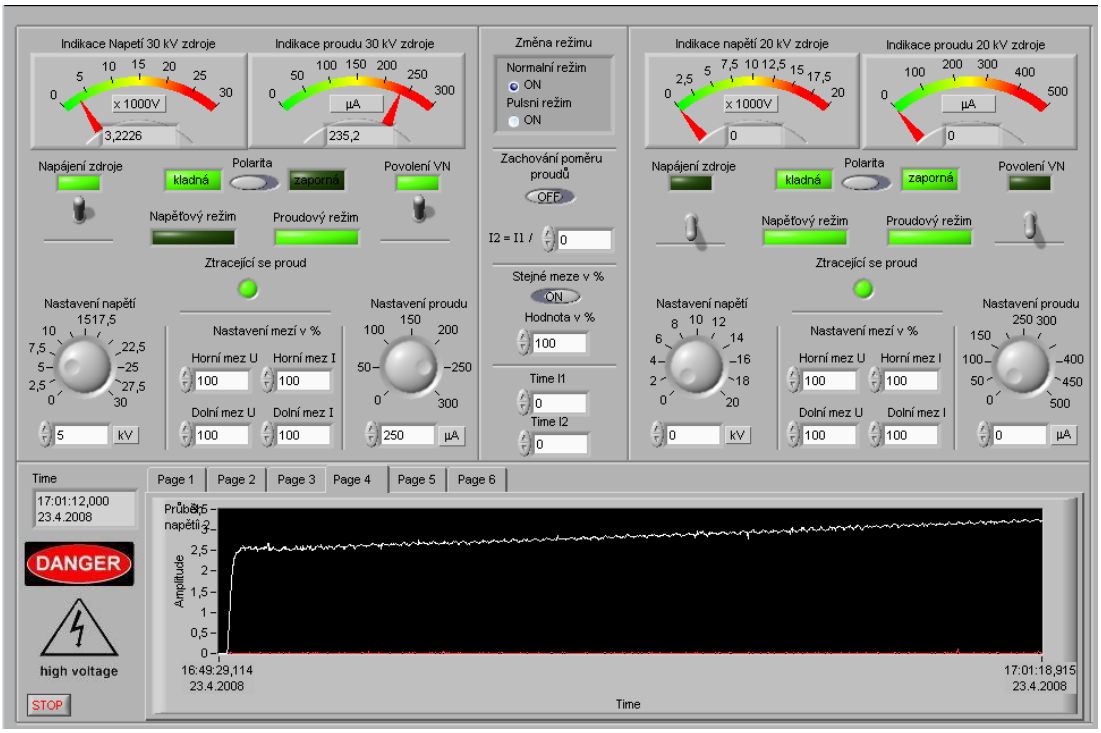
V poslední kapiláře je vidět ustálený stav separace kdy již dochází pouze k posouvání jednotlivých zón.

terminátor - kyselina octová (0.02 mol)
leading - octan amonný(0,01mol) + hydroxid amonný(0,01mol) + bromthymolova modř(0.001 mol) + methyl

Obrázek 31: Kapiláry s3

Separace č.4

Obrázek 32 ukazuje nastavení hodnot, které byly použity při čtvrté separaci. Polarita zde byla opět zvolena kladná. Hodnoty napětí a proudu byly nastaveny na počátku separace a nebyly po dobu separace měněny.

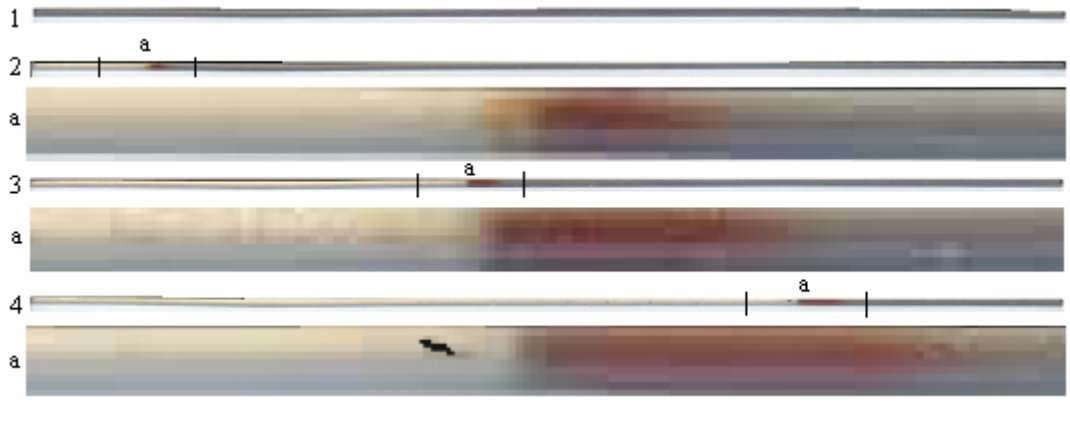


Obrázek 32: Front panel s4

Čtvrtá separace obsahuje stejné složky jako předchozí, s tím rozdílem, že složka methylu byla zaměněna za feroin.

Z obrázku 33 je vidět, že průběh separace je téměř shodný s průběhem předchozí separace. Opět je vidět, že dochází k postupnému prodlužování zón. Poslední kapilára již zobrazuje konec separace, kdy dochází pouze k posouvání zón.

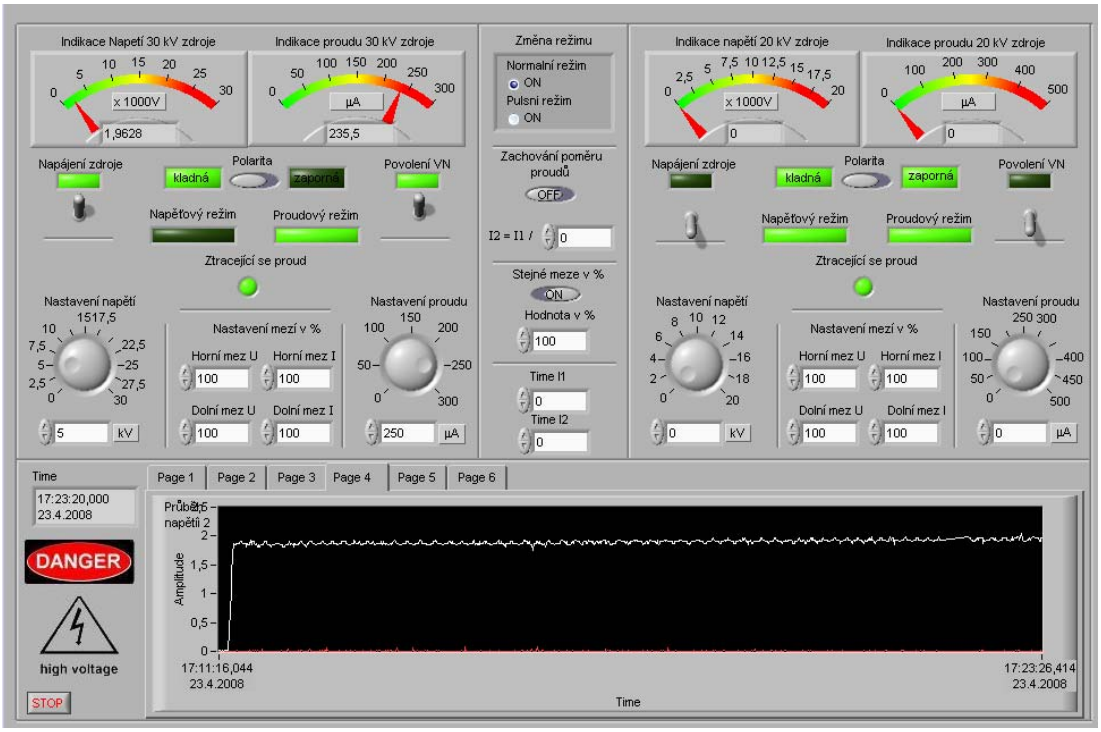
terminátor - kyselina octová(0.02 mol)
leading - octan amonný(0,01mol) + hydroxid amonný(0,01mol) + bromthymolova modř(0.001mol) + feroin



Obrázek 33: Kapiláry s4

Separace č.5

Obrázek 34 ukazuje nastavení hodnot, které byly použity při poslední, páté separaci. Polarita zde byla opět zvolena kladná. Hodnoty napětí a proudu byly nastaveny na počátku separace a nebyly po dobu separace měněny.

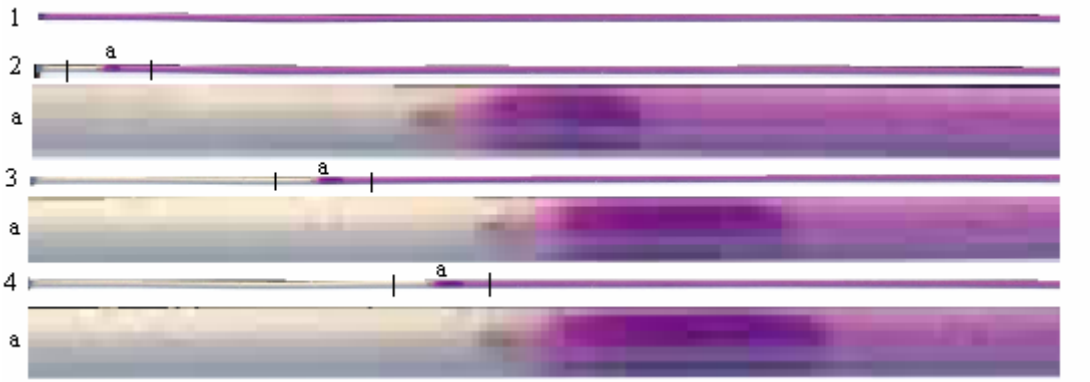


Obrázek 34: Front panel s5

Poslední, pátá separace je ukázána na obrázku 35. Je vidět, že průběh separace je opět velmi podobný předchozím separacím.

První kapilára zobrazuje počátek separace. Dále dochází k postupnému prodlužování zón. Poslední kapilára již zobrazuje ustáleny stav, kde dochází pouze k posouvání jednotlivých zón.

terminátor - kyselina octová(0.02 mol)
leading - octan amonný(0,01mol) + hydroxid amonný(0,02mol) + kresolftalexon(0,001mol) + vápník(0.0001mol)



Obrázek 35: Kapiláry s5

7. ZÁVĚR

Cílem této práce bylo navrhnout systém pro proces kapilární elektroseparace. Základní instrumentací celého systému je řídicí systém PLC AMiRiS99 firmy AMiT s.r.o. a vysokonapěťový zdroj CZE2000 firmy Spellman.

V první části jsem navrhnul detailní koncepci celého systému. Seznámil jsem se z ovládáním vysokonapěťového zdroje a základními vlastnostmi PLC. Na základě těchto informací jsem navrhnul hardwarové řešení galvanického oddělení mezi těmito instrumenty. Cílem bylo galvanicky oddělit 7 digitálních a 6 analogových kanálů. Po návrhu následovala realizace desky plošných spojů a úspěšné oživení.

V další části jsem se seznámil z možnostmi komunikace PLC s PC. Firma AMiT na svých internetových stránkách nabízí bezplatně knihovny sloužící ke komunikaci s různými aplikacemi na PC. Pro mé zadání, ovládání pomocí software LabVIEW jsem z těchto knihoven použil ATOUCHX. Tato knihovna využívá tzv. prvky ActiveX. Pomocí této knihovny jsem realizoval základní funkční bloky v LabVIEW, které jsem dále použil v konečné aplikaci.

Dále jsem se seznámil s programováním PLC. Vývojové prostředí se nazývá DetStudio, jedná se o volně stažitelný software. V tomto programu jsem realizoval řídicí algoritmus pro proces separace.

V neposlední řadě jsem navrhnul vzhled (Front Panel) výsledné VI aplikace v softwaru LabVIEW. Pomocí již dříve navržených funkčních bloků jsem zajistil komunikaci této aplikace s PLC.

Poslední část práce byla zaměřena na testování celého navrhnutého systému. Testování jsem uskutečnil na reálné separaci chemických vzorků. Při tomto testování byl odstraněny menší, většinou estetické nedostatky. Celkem jsem provedl pět úspěšných separací na různých vzorcích.

8. SEZNAM POUŽITÝCH ZDROJŮ

[1] Instruction manual CZE series, SPELLMAN, 2000 [online]. URL:<www.spellmanhv.com/tech/pdf/CZEMAN.pdf>

[2] AMiT s.r.o., Uživatelský manuál ATOUCHX, 2008 [online]. URL: <www.amit.cz>

[3] Linear optocoupler IL300, Siemens, [online]. URL: <http://www.datasheetcatalog.com/datasheets_pdf/I/L/3/0/IL300.shtml >

[4] Quadruple Operational Amplifiers LM324, Texas Instruments, 2002 [online]. URL:<http://www.datasheetcatalog.com/datasheets_pdf/L/M/3/2/LM324.shtml>

[5] Václav Říha, Izatochoforéza, [on-line]. URL:<www.upce.cz/priloha/kalch-anal-itp>

[6] Dokumentace AMiRiS99, AMiT s.r.o., 2008 [on-line] <http://www.amit.cz/docs/cz/kompakty/amiris99_d_cz_100.pdf>

[7] AMiT s.r.o., DetStudio [počítačový program], Ver. Beta 1.0.45, 2008. Dostupné z <http://www.amit.cz/inet_dir/cz/sw/paramsw.htm#detstudio>

9. SEZNAM PŘÍLOH

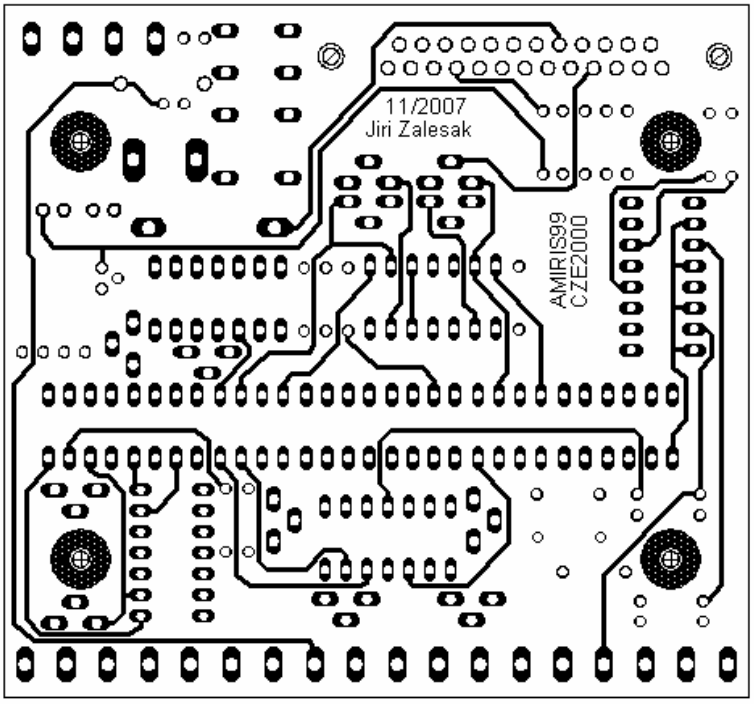
P1 – Desky plošných spojů

P2 – Soupis součástek

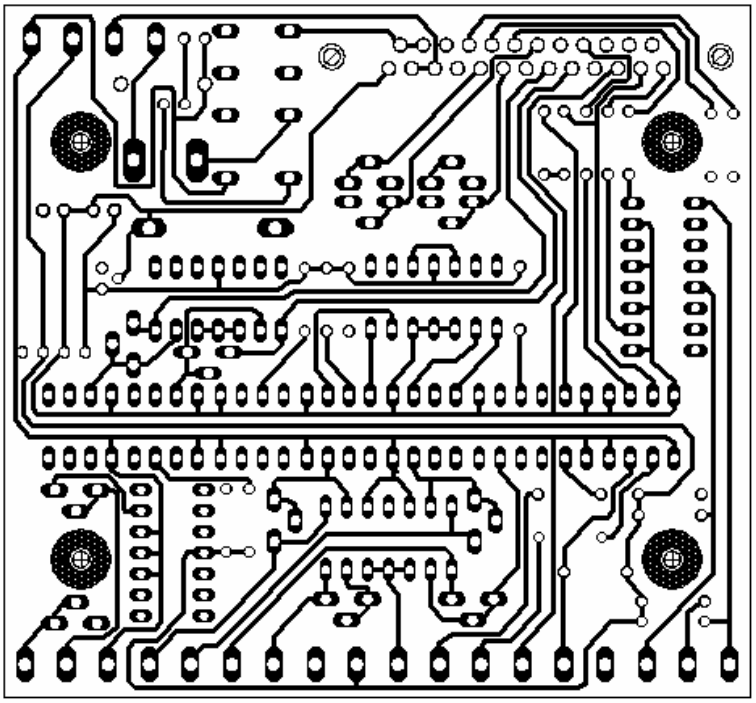
P3 – Výpis zdrojového kódu řídicího algoritmu

P4 – Dokumentace VIs

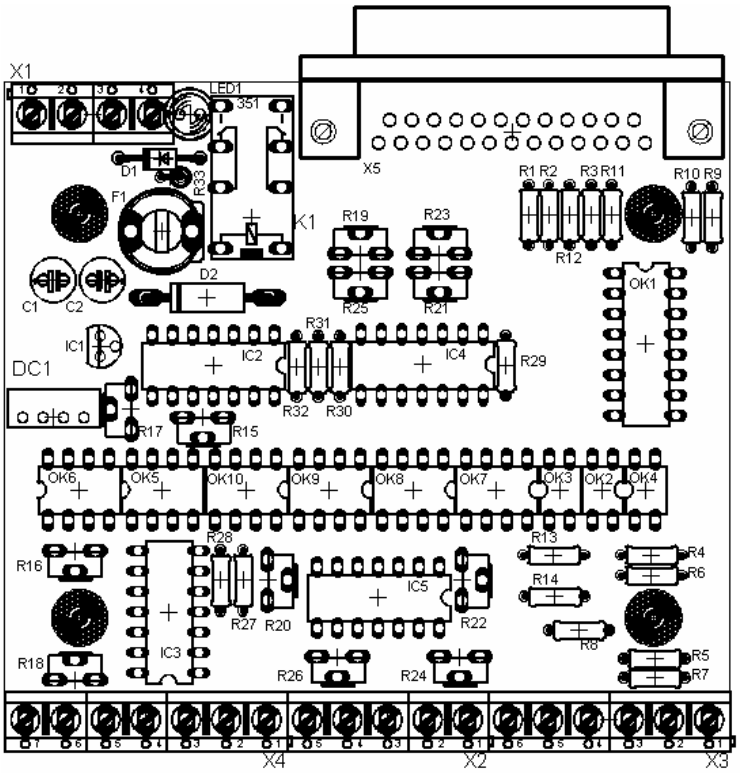
Příloha P1 - Desky plošných spojů



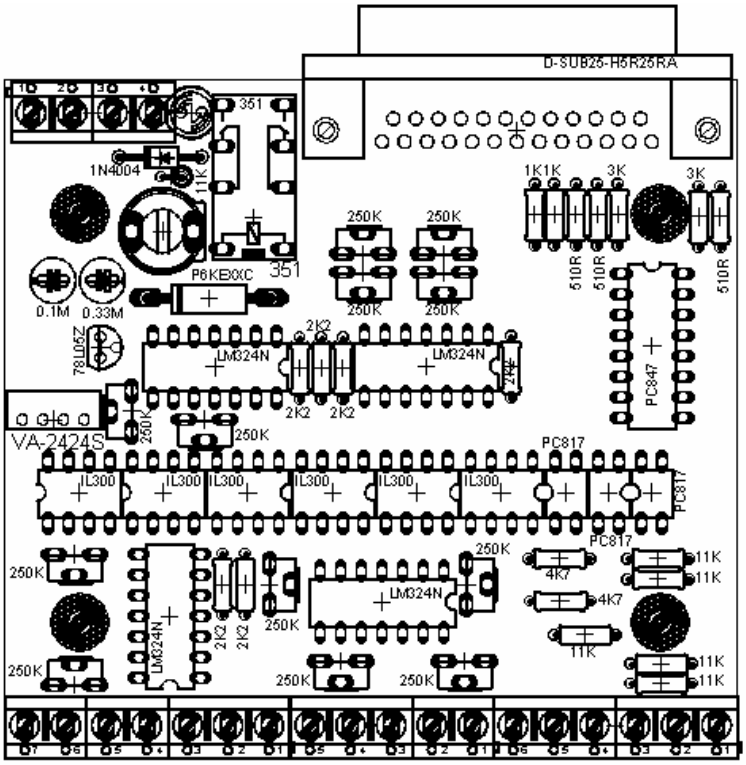
Pohled TOP na DPS



Pohled BOTTOM na DPS



Rozmístění součástek na DPS



Hodnoty součástek na DPS

Příloha P2 - Soupis součástek

Součástka	Hodnota		Součástka	Hodnota
R1	1K		D1	1N4004
R2	1K		D2	P6KE15CA
R3	510R		C1	0,1M/50V
R4	11K		C2	0,33M/50V
R5	11K		K1	G5V-2
R6	11K		F1	5x20 2A
R7	11K		DC1	SIM1-2424
R8	11K		OK1	PC847
R9	510R		OK2	PC817
R10	3K		OK3	PC817
R11	3K		OK4	PC817
R12	510R		OK5	IL300
R13	4K7		OK6	IL300
R14	4K7		OK7	IL300
R15	250K		OK8	IL300
R16	250K		OK9	IL300
R17	250K		OK10	IL300
R18	250K		IC1	78L05
R19	250K		IC2	LM324N
R20	250K		IC3	LM324N
R21	250K		IC4	LM324N
R22	250K		IC5	LM324N
R23	250K		X1	2xARK500/2
R24	250K		X2	ARK500/2, ARK500/3
R25	250K		X3	2xARK500/3
R26	250K		X4	2xARK500/2, ARK500/3
R27	2K2		X5	CAN 25 Z 90
R28	2K2			
R29	2K2			
R30	2K2			
R31	2K2			
R32	2K2			
R33	11K			

Příloha P3 - Výpis zdrojového kódu řídícího algoritmu

```
Proces Hlavni
//zdroj 1
Let @i_pNRp_i_p = i_put[0,0] != p_i_put[0,0]
Let @i_pRp_i_p  = i_put[0,0] == p_i_put[0,0]
Let @u_pNRp_u_p = u_put[0,0] != p_u_put[0,0]
Let @u_pRp_u_p  = u_put[0,0] == p_u_put[0,0]
//zdroj 2
Let @i_pNRp_i_p1 = i_put[0,1] != p_i_put[0,1]
Let @i_pRp_i_p1  = i_put[0,1] == p_i_put[0,1]
Let @u_pNRp_u_p1 = u_put[0,1] != p_u_put[0,1]
Let @u_pRp_u_p1  = u_put[0,1] == p_u_put[0,1]
//nastaveni hodnot matice pro interpolaci proudu a napeti
If @i_pRp_i_p
Let krivka_I[0,1] = i_put[0,0]
Let citac_1 = 0
EndIf
If @i_pRp_i_p1
Let krivka_I1[0,1] = i_put[0,1]
Let citac_3 = 0
EndIf
If @u_pRp_u_p
Let krivka_U[0,1] = u_put[0,0]
Let citac_2 = 0
EndIf
If @u_pRp_u_p1
Let krivka_U1[0,1] = u_put[0,1]
Let citac_4 = 0
EndIf

Let krivka_I[1,0]  = smernice
Let krivka_U[1,0]  = smernice
Let krivka_I1[1,0] = smernice
Let krivka_U1[1,0] = smernice

Let krivka_I[1,1]  = i_put[0,0]
Let krivka_U[1,1]  = u_put[0,0]
Let krivka_I1[1,1] = i_put[0,1]
Let krivka_U1[1,1] = u_put[0,1]
//zajistuje zadany prubeh proudu a napeti
Interpol citac_1, p_i_put[0,0], krivka_I
Interpol citac_2, p_u_put[0,0], krivka_U
Interpol citac_3, p_i_put[0,1], krivka_I1
Interpol citac_4, p_u_put[0,1], krivka_U1

//////////////////////Pulsnirezim//////////////////////////////////////
If @zap_puls, :00001                                //zapnuti pulsniho rezimu
If @kon_perioda                                     //pri konci periody se hodnoty
Let i_put[0,0] = puls_I[0,0]                        //nastavi na puvodni hodnoty
Let i_put[0,1] = puls_I[0,1]
Let citac_5 = 0
EndIf
//doba trvani proudu I1, zapoctena doba narustu proudu
Let @perioda_t1 = citac_5 <= time[0,0] + smernice
//doba trvani proudu I2, zapoctena doba narustu proudu
Let @perioda_t2 = citac_5 > time[0,0] + smernice
//celkova doba I1 a I2
Let @kon_perioda = citac_5 > time[0,0] + time[0,1] + 3*smernice
If @perioda_t1
//doba proudu I1, hodnoty odpovidaji nastavenym hodnotam
Let puls_I[0,0] = i_put[0,0]
Let puls_I[0,1] = i_put[0,1]
EndIf

If @perioda_t2                        //doba proudu I2, hodnoty proudu jsou prohozeny
Let i_put[0,0] = puls_I[0,1]
Let i_put[0,1] = puls_I[0,0]
EndIf
:00001 EndIf
```

Proces Citace

```
//proces norm rez
IncDec citac_1, @i_pNRp_i_p, 0.1            //citac pro interpol proudu I1 - norm_rez
IncDec citac_2, @u_pNRp_u_p, 0.1            //citac pro interpol napeti U1 - norm_rez
IncDec citac_3, @i_pNRp_i_p1, 0.1           //citac pro interpol proudu I2 - norm_rez
IncDec citac_4, @u_pNRp_u_p1, 0.1           //citac pro interpol napeti U2 - norm_rez


//proces puls rez
IncDec citac_5, @citac_5pr, 0.1      //citac pro vymenu proudu, cita pri spusteni
pulsniho rez. a musi byt zadany periody a hodnoty proudu

//proces kontrola
IncDec citac_6, 1, 0.100             //pro nastaveni mezi jednou za zadanou dobu

IncDec citac_7, @i_putRnul, 0.1
IncDec citac_8, @i_put1Rnul, 0.1
```

Proces Osetreni

```
Let @zmena_i_put = citac_5 <= time[0,0] + smernice  //osetreni proti zamene proudu
pri zmenach zadanych hodnot v pulsnim

//nastaveni hodnot probehne pouze v dobe proudu I1, aby nedoslo k zamene proudu pri
//zmene hodnot
If @zmena_i_put
Let i_put[0,0] = i_put_LV[0,0]
Let i_put[0,1] = i_put_LV[0,1]

Let u_put[0,0] = u_put_LV[0,0]
Let u_put[0,1] = u_put_LV[0,1]
EndIf

Let i_put[0,0] = abs(i_put[0,0])                    //proti zadani zaporne hodnoty
Let i_put[0,1] = abs(i_put[0,1])
Let u_put[0,0] = abs(u_put[0,0])
Let u_put[0,1] = abs(u_put[0,1])

Let time[0,0] = abs(time[0,0])                      //proti zadani zaporne hodnoty
Let time[0,1] = abs(time[0,1])

Let pomer = abs(pomer)                              //proti zadani zaporne hodnoty

Let @i_putVnul  = i_put[0,0] > 0
Let @u_putVnul  = u_put[0,0] > 0
Let @i_put1Vnul = i_put[0,1] > 0
Let @u_put1Vnul = u_put[0,1] > 0

Let @i_putRnul  = i_put[0,0] == 0
Let @u_putRnul  = u_put[0,0] == 0
Let @i_put1Rnul = i_put[0,1] == 0
Let @u_put1Rnul = u_put[0,1] == 0



Let @ui_putVnul  = @i_putVnul  and @u_putVnul
Let @ui_put1Vnul = @i_put1Vnul and @u_put1Vnul

Let @timeVnul = time[0,0] * time[0,1] > 0    //pro zapnuti citace pro pulsni rezim

//pro spusteni pulsniho rez. musi byt zadany periody a hodnoty proudu
Let @citac_5pr = @i_putVnul and @i_put1Vnul and @timeVnul and @zap_puls

Let @pomerVnul = pomer > 0                  //zapnuti pomeru pouze pro zadany pomer
Let @zap_puls_ng = not @zap_puls

If @zap_puls_ng
Let citac_5 = 0
EndIf
```

```
Let @zmen_mez_ng = not @zmena_mezi
Let @i_get = c_proud[0,0] > 1
Let @i_get1 = c_proud[0,1] > 1

Let @zmen_mez_ng = not @zmena_mezi
Let @u_get = c_napeti[0,0] > 30
Let @u_get1 = c_napeti[0,1] > 30

//osetreni proti zadani hodnot u a i pred zapnutim VN
Let @vn_vyp = not @vn_zap
Let @u_or_i_pVn = @i_putVnul or @u_putVnul
Let @vn_vANDuiVn = @vn_vyp and @u_or_i_pVn

If @vn_vANDuiVn
Let @vn_zap = bool(0)
Let hlaseni1 = 2010
Let i_put_LV[0,0] = 0
Let u_put_LV[0,0] = 0
EndIf

//osetreni proti zadani hodnot u a i pred zapnutim VN1
Let @vn_vyp_1 = not @vn_zap_1
Let @u_or_i_pVn1 = @i_put1Vnul or @u_put1Vnul
Let @vn1vANDuiVn = @vn_vyp_1 and @u_or_i_pVn1

If @vn1vANDuiVn
Let @vn_zap_1 = bool(0)
Let hlaseni1 = 2011
Let i_put_LV[0,1] = 0
Let u_put_LV[0,1] = 0
EndIf

Let @vse = @rele and @vn_zap and @i_putVnul and @u_putVnul
Let @vse1 = @rele_1 and @vn_zap_1 and @i_put1Vnul and @u_put1Vnul

If hlaseni_nul.0
Let hlaseni = 0
Let hlaseni1 = 0
Let hlaseni_nul = 0
EndIf

Let @timeout_1 = citac_7 > 3600
Let @timeout_2 = citac_8 > 3600

If @vn_vyp
Let c_proud[0,0] = 0
Let c_napeti[0,0] = 0
Let @u_mod = bool(1)
Let @zap_pol = bool(1)
Let @i_mod = bool(1)
Let @klad_pol = bool(1)
Let @z_proud = bool(1)
EndIf

If @vn_vyp_1
Let c_proud[0,1] = 0
Let c_napeti[0,1] = 0
Let @u_mod_1 = bool(1)
Let @zap_pol_1 = bool(1)
Let @i_mod_1 = bool(1)
Let @klad_pol_1 = bool(1)
Let @z_proud_1 = bool(1)
EndIf


Proces Term
Lcw3Idle NONE
```

Proces Prepinac

```
If @pomer                           //zapnuti pomeru proudu
If @pomerVnul, :NONE
Let i_put[0,1] = i_put[0,0] / pomer
Let u_put[0,1] = u_put[0,0]
EndIf
EndIf

If @s_procento                      //pri zapnutem stejnem procentu pro meze
Let procento[0,0] = abs(procento_s) //dochazi k nastaveni vsech mezi na stejnou
Let procento[0,1] = abs(procento_s) //hodnotu
Let procento[1,0] = abs(procento_s)
Let procento[1,1] = abs(procento_s)
Let procento[2,0] = abs(procento_s)
Let procento[2,1] = abs(procento_s)
Let procento[3,0] = abs(procento_s)
Let procento[3,1] = abs(procento_s)
EndIf


If @i_putVnul
Let citac_7 = 0
EndIf

If @i_put1Vnul
Let citac_8 = 0
EndIf


If @timeout_1
Let @rele = bool(0)
Let citac_7 = 0
Let hlaseni = 2012
EndIf

If @timeout_2
Let @rele_1 = bool(0)
Let citac_8 = 0
Let hlaseni = 2013
EndIf
```

Proces In_out

```
BinOut @polarita, 0x0000, #DO00_0     //ovládání polarity
BinOut @vn_zap, 0x0001, #DO00_1       //povolení vysokého napětí, neg.
BinOut @rele, 0x0000, #DO00_2         //ovládání relé

BinOut @polarita_1, 0x0000, #DO11_0   //ovládání polarity 2
BinOut @vn_zap_1, 0x0001, #DO11_1     //povolení vysokého napětí, neg. 2
BinOut @rele_1, 0x0000, #DO11_2       //ovládání relé 2

BinIn #DI00_0, 0x0001, @u_mod         //U - mod, neg.
BinIn #DI00_1, 0x0001, @zap_pol       //záporná polarita, neg.
BinIn #DI00_5, 0x0001, @i_mod         //I - mod, neg.
BinIn #DI00_3, 0x0001, @klad_pol      //kladná polarita, neg.
BinIn #DI00_4, 0x0001, @z_proud       //ztrácející se proud, neg.
BinIn #DI12_0, 0x0001, @u_mod_1       //U - mod, neg. 2
BinIn #DI12_1, 0x0001, @zap_pol_1     //záporná polarita, neg. 2
BinIn #DI12_5, 0x0001, @i_mod_1       //I - mod, neg. 2
BinIn #DI12_3, 0x0001, @klad_pol_1    //kladná polarita, neg. 2
BinIn #DI12_4, 0x0001, @z_proud_1     //ztrácející se proud, neg. 2

AnOut #AO00_0, p_i_put[0,0],10.000,0.000,10.000,0.000,300.000   //nastavení proudu I
AnOut #AO00_1, p_u_put[0,0],10.000,0.000,10.000,0.000,30.000    //nastavení napětí U
AnOut #AO00_2, p_i_put[0,1],10.000,0.000,10.000,0.000,500.000   //nastavení proudu I2
AnOut #AO00_3, p_u_put[0,1],10.000,0.000,10.000,0.000,20.000    //nastavení napětí U2

AnIn #AI00_2, c_proud[0,0], 10.000,0.000,10.000,0.000,300.000   //kontrola proudu I
AnIn #AI00_3, c_napeti[0,0], 10.000,0.000,10.000,0.000,30.000   //kontrola napětí U
AnIn #AI00_7, c_napeti[0,1], 10.000,0.000,10.000,0.000,20.000   //kontrola napětí U2
AnIn #AI00_6, c_proud[0,1], 10.000,0.000,10.000,0.000,500.000   //kontrola proudu I2
```

Proces In_out

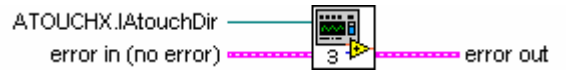
```
Let @zmena_mez_t = citac_6 >= 25     //aktualizace mezi jednou za zadanou hodnotu

//hlidani zmeny nastaveni hodnot
Limits p_u_put[0,0], @z_p_u_put, @z_p_u_put, u_meze[0,0], u_meze[0,1], 0.000, 0x0000
Let u_meze[0,0] = p_u_put[0,0] - 0.01
Let u_meze[0,1] = p_u_put[0,0] + 0.01

Limits p_i_put[0,0], @z_p_i_put, @z_p_i_put, i_meze[0,0], i_meze[0,1], 0.000, 0x0000
Let i_meze[0,0] = p_i_put[0,0] - 0.01
Let i_meze[0,1] = p_i_put[0,0] + 0.01

Limits p_u_put[0,1], @z_p_u_put1,@z_p_u_put1, u_meze[1,0], u_meze[1,1], 0.000, 0x0000
Let u_meze[1,0] = p_u_put[0,1] - 0.01
Let u_meze[1,1] = p_u_put[0,1] + 0.01

Limits p_i_put[0,1], @z_p_i_put1,@z_p_i_put1, i_meze[1,0], i_meze[1,1], 0.000, 0x0000
Let i_meze[1,0] = p_i_put[0,1] - 0.01
Let i_meze[1,1] = p_i_put[0,1] + 0.01

Let @zmena_mezi = @z_p_u_put or @z_p_i_put or @z_p_u_put1 or @z_p_i_put1 or
@zmena_mez_t

//dochazi k prenastaveni hodnot mezi jednou za zadanou dobu nebo pri zmene zadaneho
//napeti nebo proudu
If @zmena_mezi
Let meze[0,0] = (c_proud[0,0]  - (c_proud[0,0]  * procento[0,0] / 100)) - 2
//dolni mez pro I
Let meze[0,1] = (c_proud[0,0]  + (c_proud[0,0]  * procento[0,1] / 100)) + 2
//horni mez pro I
Let meze[1,0] = (c_proud[0,1]  - (c_proud[0,1]  * procento[1,0] / 100)) - 2
//dolni mez pro I2
Let meze[1,1] = (c_proud[0,1]  + (c_proud[0,1]  * procento[1,1] / 100)) + 2
//horni mez pro I2
Let meze[2,0] = (c_napeti[0,0] - (c_napeti[0,0] * procento[2,0] / 100)) - 30
//dolni mez pro U
Let meze[2,1] = (c_napeti[0,0] + (c_napeti[0,0] * procento[2,1] / 100)) + 30
//horni mez pro U
Let meze[3,0] = (c_napeti[0,1] - (c_napeti[0,1] * procento[3,0] / 100)) - 30
//dolni mez pro U2
Let meze[3,1] = (c_napeti[0,1] + (c_napeti[0,1] * procento[3,1] / 100)) + 30
//horni mez pro U2
Let citac_6 = 0
EndIf


//hlidani zdali jsou promene v nastavenych mezich
If @zmen_mez_ng, :00000              //pokud nedochazi ke zmene hodnot
If @vn_zap, :00100                   //pokud je zapnuto VN
If @ui_putVnul                       //poku jsou napeti a proudy vetsi nez nula
Limits c_proud[0,0],  @nad_I,  @pod_I, meze[0,0], meze[0,1], 0.000, 0x0000//hlidani I
Limits c_napeti[0,0], @nad_U,  @pod_U, meze[2,0], meze[2,1], 0.000, 0x0000 /hlidani U
EndIf
:00100 EndIf
If @vn_zap_1, :00101                 //pokud je zapnuto VN
If @ui_put1Vnul                      //poku jsou napeti a proudy vetsi nez nula
Limits c_proud[0,1], @nad_I1,@pod_I1, meze[1,0], meze[1,1], 0.000, 0x0000//hlidani I2
Limits c_napeti[0,1],@nad_U1,@pod_U1, meze[3,0], meze[3,1], 0.000, 0x0000//hlidani U2
EndIf
:00101 EndIf
:00000 EndIf
//Vypnuti VN pri prekroceni mezi
If @vse, :01000              //pokud je zapnuto VN, rele, hodnoty jsou vetsi nez nula
If @i_get, :00011            //hodnota cteneho proudu je vetsi nez 1 mikroA
If @pod_I                    //pokud proud I1 klesne pod zadanou mez
Let prokontrolu[0,0] = c_proud[0,0]
Let prokontrolu[0,1] = c_proud[0,1]
Let prokontrolu[1,0] = c_napeti[0,0]
Let prokontrolu[1,1] = c_napeti[0,1]
Let @vn_zap = bool(0) //dochazi k vypnuti VN
Let hlaseni = 2002    //a je vyslana zprava
EndIf
```

```
If @nad_I
Let prokontrolu[0,0] = c_proud[0,0]
Let prokontrolu[0,1] = c_proud[0,1]
Let prokontrolu[1,0] = c_napeti[0,0]
Let prokontrolu[1,1] = c_napeti[0,1]
Let @vn_zap = bool(0) //dochazi k vypnuti VN
Let hlaseni = 2003            //a je vyslana zprava
EndIf
:00011 EndIf
If @u_get, :00025             //pokud hodnota cteneho napeti je vetsi nez 30V
If @pod_U                     //pokud proud U1 klesne pod zadanou mez
Let prokontrolu[0,0] = c_proud[0,0]
Let prokontrolu[0,1] = c_proud[0,1]
Let prokontrolu[1,0] = c_napeti[0,0]
Let prokontrolu[1,1] = c_napeti[0,1]
Let @vn_zap = bool(0)         //dochazi k vypnuti VN
Let hlaseni = 2006            //a je vyslana zprava
EndIf
If @nad_U                     //pokud proud U1 vzroste nad zadanou mez
Let prokontrolu[0,0] = c_proud[0,0]
Let prokontrolu[0,1] = c_proud[0,1]
Let prokontrolu[1,0] = c_napeti[0,0]
Let prokontrolu[1,1] = c_napeti[0,1]
Let @vn_zap = bool(0) //dochazi k vypnuti VN
Let hlaseni = 2007            //a je vyslana zprava
EndIf
:00025 EndIf
:01000 EndIf

If @vse1, :02000              //pokud je zapnuto VN, rele, hodnoty jsou vetsi nez nula
If @i_get1, 00002
If @pod_I1                    //pokud proud I2 klesne pod zadanou mez
Let prokontrolu[0,0] = c_proud[0,0]
Let prokontrolu[0,1] = c_proud[0,1]
Let prokontrolu[1,0] = c_napeti[0,0]
Let prokontrolu[1,1] = c_napeti[0,1]
Let @vn_zap_1 = bool(0)       //dochazi k vypnuti VN
Let hlaseni = 2004            //a je vyslana zprava
EndIf

If @nad_I1                    //pokud proud I2 vzroste nad zadanou mez
Let prokontrolu[0,0] = c_proud[0,0]
Let prokontrolu[0,1] = c_proud[0,1]
Let prokontrolu[1,0] = c_napeti[0,0]
Let prokontrolu[1,1] = c_napeti[0,1]
Let @vn_zap_1 = bool(0)       //dochazi k vypnuti VN
Let hlaseni = 2005            //a je vyslana zprava
EndIf
:00002 EndIf

If @u_get1, :00026
If @pod_U1                    //pokud proud U2 klesne pod zadanou mez
Let prokontrolu[0,0] = c_proud[0,0]
Let prokontrolu[0,1] = c_proud[0,1]
Let prokontrolu[1,0] = c_napeti[0,0]
Let prokontrolu[1,1] = c_napeti[0,1]
Let @vn_zap_1 = bool(0)       //dochazi k vypnuti VN
Let hlaseni = 2008            //a je vyslana zprava
EndIf
If @nad_U1                    //pokud proud U2 vzroste nad zadanou mez
Let prokontrolu[0,0] = c_proud[0,0]
Let prokontrolu[0,1] = c_proud[0,1]
Let prokontrolu[1,0] = c_napeti[0,0]
Let prokontrolu[1,1] = c_napeti[0,1]
Let @vn_zap_1 = bool(0)       //dochazi k vypnuti VN
Let hlaseni = 2009            //a je vyslana zprava
EndIf
:00026 EndIf
:02000 EndIf
Let pro_meze = 0
```

Příloha P4 – Dokumentace VIs

Done.vi

Connector Pane



Controls and Indicators

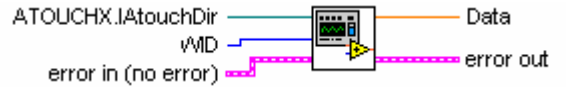
ATOUCHX.IAtouchDir Enabling directive communication via ActiveX

error in (no error) The **error in** cluster can accept error information wired from VIs previously called. Use this information to decide if any functionality should be bypassed in the event of errors from other VIs. The pop-up option **Explain Error** (or Explain Warning) gives more information about the error displayed.

error out The **error in** cluster can accept error information wired from VIs previously called. Use this information to decide if any functionality should be bypassed in the event of errors from other VIs. The pop-up option **Explain Error** (or Explain Warning) gives more information about the error displayed.

GetData.vi

Connector Pane



Controls and Indicators

ATOUCHX.IAtouchDir Enabling directive communication via ActiveX

WID Variable - identification number

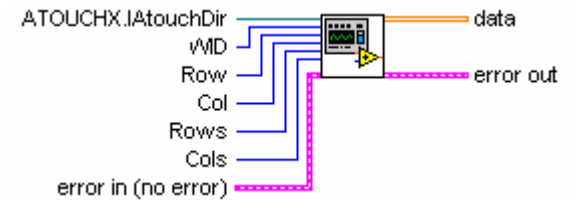
error in (no error) The **error in** cluster can accept error information wired from VIs previously called. Use this information to decide if any functionality should be bypassed in the event of errors from other VIs. The pop-up option **Explain Error** (or Explain Warning) gives more information about the error displayed.

Data Read variable value.

error out The **error in** cluster can accept error information wired from VIs previously called. Use this information to decide if any functionality should be bypassed in the event of errors from other VIs. The pop-up option **Explain Error** (or Explain Warning) gives more information about the error displayed.

GetDataMtx.vi

Connector Pane



Controls and Indicators

ATOUCHX.IAtouchDir Enabling directive communication via ActiveX

error in (no error) The **error in** cluster can accept error information wired from VIs previously called. Use this information to decide if any functionality should be bypassed in the event of errors from other VIs.

The pop-up option **Explain Error** (or Explain Warning) gives more information about the error displayed.

WID Variable - identification number.

Row Number of rows, that are reads.

Col Number of cols, that are reads.

Rows Number first row, from where reads. (indexed from 0)

Cols Number first cols, from where reads. (indexed from 0)

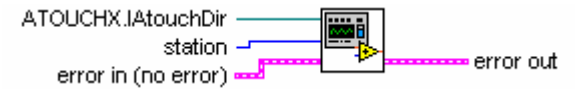
Data Value matrix, that are reads.

error out The **error in** cluster can accept error information wired from VIs previously called. Use this information to decide if any functionality should be bypassed in the event of errors from other VIs.

The pop-up option **Explain Error** (or Explain Warning) gives more information about the error displayed.

GetTime.vi

Connector Pane



Controls and Indicators

ATOUCHX.IAtouchDir Enabling directive communication via ActiveX

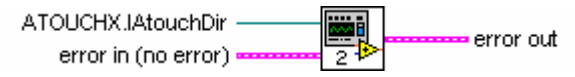
station Number station.

error in (no error) The **error in** cluster can accept error information wired from VIs previously called. Use this information to decide if any functionality should be bypassed in the event of errors from other VIs. The pop-up option **Explain Error** (or Explain Warning) gives more information about the error displayed.

error out The **error in** cluster can accept error information wired from VIs previously called. Use this information to decide if any functionality should be bypassed in the event of errors from other VIs. The pop-up option **Explain Error** (or Explain Warning) gives more information about the error displayed.

InitFromString.vi

Connector Pane



Controls and Indicators

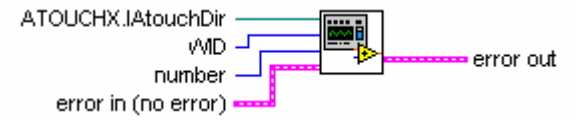
ATOUCHX.IAtouchDir Enabling directive communication via ActiveX

error in (no error) The **error in** cluster can accept error information wired from VIs previously called. Use this information to decide if any functionality should be bypassed in the event of errors from other VIs. The pop-up option **Explain Error** (or Explain Warning) gives more information about the error displayed.

error out The **error in** cluster can accept error information wired from VIs previously called. Use this information to decide if any functionality should be bypassed in the event of errors from other VIs. The pop-up option **Explain Error** (or Explain Warning) gives more information about the error displayed.

PutData.vi

Connector Pane



Controls and Indicators

ATOUCHX.IAtouchDir Enabling directive communication via ActiveX

WID Variable - identification number

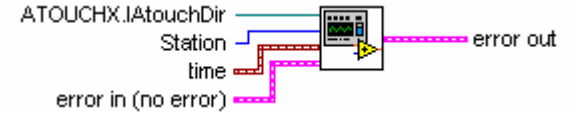
number Write variable value

error in (no error) The **error in** cluster can accept error information wired from VIs previously called. Use this information to decide if any functionality should be bypassed in the event of errors from other VIs. The pop-up option **Explain Error** (or Explain Warning) gives more information about the error displayed.

error out The **error in** cluster can accept error information wired from VIs previously called. Use this information to decide if any functionality should be bypassed in the event of errors from other VIs. The pop-up option **Explain Error** (or Explain Warning) gives more information about the error displayed.

PutTime.vi

Connector Pane



Controls and Indicators

ATOUCHX.IAtouchDir Enabling directive communication via ActiveX

Station Number station.

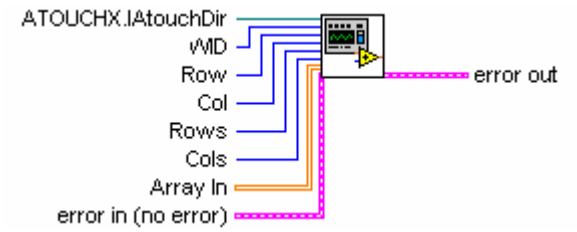
time Value time.

error in (no error) The **error in** cluster can accept error information wired from VIs previously called. Use this information to decide if any functionality should be bypassed in the event of errors from other VIs. The pop-up option **Explain Error** (or Explain Warning) gives more information about the error displayed.

error out The **error in** cluster can accept error information wired from VIs previously called. Use this information to decide if any functionality should be bypassed in the event of errors from other VIs. The pop-up option **Explain Error** (or Explain Warning) gives more information about the error displayed.

PutDataMtx.vi

Connector Pane



Controls and Indicators

ATOUCHX.IAtouchDir Enabling directive communication via ActiveX

WID Variable - identification number.

error in (no error) The **error in** cluster can accept error information wired from VIs previously called. Use this information to decide if any functionality should be bypassed in the event of errors from other VIs.

The pop-up option **Explain Error** (or Explain Warning) gives more information about the error displayed.

Row Number of rows, that are write.

Col Number of cols, that are write.

Rows Number first row, from where write. (indexed from 0)

Cols Number first cols, from where write. (indexed from 0)

Array In Value matrix, that are write.

error out The **error in** cluster can accept error information wired from VIs previously called. Use this information to decide if any functionality should be bypassed in the event of errors from other VIs.

The pop-up option **Explain Error** (or Explain Warning) gives more information about the error displayed.