Univerzita Palackého v Olomouci

Přírodovědecká fakulta

Integrace systému µLab do LabVIEW

The Integration of µLab into LabView

Obor: Aplikovaná fyzika
Rok: 2016
Autor: B Eng Fredericus Marinus Anne Linderhof
Vedoucí: Mgr. Milan Vůjtek, Ph.D.

Bibliografická identifikace

Jméno a příjmení autora	B Eng Fredericus Marinus Anne Linderhof
Název práce	Integrace systému µLab do LabVIEW
Typ práce	Diplomová
Pracoviště	Katedra experimentální fyziky
Vedoucí práce	Mgr. Milan Vůjtek, Ph.D.
Rok obhajoby práce	2016
Počet stran	80
Počet příloh	0
Jazyk	Český
Abstrakt	Práce je zaměřena na ovládání systému µLab, respektive modulu A&DDU z prostředí LabVIEW. Probírá se problém komunikace, rozbor komunikačního protokolu a poté je popisován nově vytvořený program pro ovládání modulu A&DDU.
Klíčová slova	LabVIEW, µLab, rc2000

Bibliographical Identification

Author's first name and surname	B Eng Fredericus Marinus Anne Linderhof
Title	The Integration of μLab into LabView
Type of thesis	Master
Department	Department of Experimental Physics
Supervisor	Mgr. Milan Vůjtek, Ph.D.
Year of presentation	2016
Number of pages	80
Number of appendices	0
Language	Czech
Abstract	This thesis is focused on controlling of the μLab system, the A&DDU module respectively from development environment LabVIEW. It addresses communication problems, the analysis of the communication protocol and it describes newly created software for controlling the A&DDU module.
Keywords	LabVIEW, μLab, rc2000

Prohlašuji, že jsem předloženou diplomovou práci vypracoval samostatně pod vedením mgr. Milana Vůjtka, Ph.D., a že jsem použil zdrojů, které cituji a uvádím v seznamu použitých zdrojů.

V Olomouci dne ....

podpis

Poděkování

Za pomoc s nastavením osciloskopu bych chtěl poděkovat Michalu Dudkovi, Vlastimilu Vrbovi a Leně Stránské bych chtěl poděkovat za jejich pomoc s jazykovou kontrolou této práce. Nejvíce si ale zaslouží poděkování vedoucí Milan Vůjtek za své cenné rady po celou dobu vedení mé diplomové práce.

Obsah

1. Úvod

Ve výuce *Praktik z elektřiny a magnetismu* KEF/FP2 se na Katedře experimentální fyziky používá v dnešní době modulový výukový systém rc2000 – μLAB firmy *RC společnost s. r. o. přístroje pro vědu a vzdělávání* [1]. Tento systém se skládá z různých hardwarových modulů, ze kterých je nejvýznamnější *Analog and digital data unit* (česky: analogová a digitální datová jednotka, zkratka A&DDU), a počítačového programu rc2000, který s A&DDU komunikuje. Přestože je program rc2000 plně funkční a pro řadu úloh dostačuje, pro výuku na VŠ má ovšem některé nevýhody. Není například možné měřit s jedním počítačem pomocí více modulů A&DDU nebo měřit více než jen jeden kanál v měřicím režimu *Oscilloscope + Gen*. Také se v některých úkolech stává, že systém nenaměří veškerá data. Zároveň pro výuku předmětu *Číslicové měřicí systémy* by bylo vhodné mít k dispozici zařízení, které je dostupné na všech pracovištích a ovládatelné z počítače pomocí vývojového prostředí LabVIEW. Proto byla navržena diplomová práce, která by po prostudování a vyzkoušení komunikačního protokolu A&DDU vedla k vytvoření nového programu v prostředí LabVIEW, který by mohl dílčí nevýhody systému rc2000 obejít.

Za tímto účelem byl napsán plán skládající se z 5 bodů:

1. seznámení se s rc2000 a studium ovládání A&DDU,
2. studium LabVIEW, zejména komunikace s RS-232,
3. studium komunikačního protokolu, případně navázání kontaktu s firmou RC společnost,
4. napsání programu v LabVIEW ke komunikaci s A&DDU, rozšiřující stávající možnosti a
5. případné využití Data Dashboard for LabVIEW v systému Android.

Vedle těchto pěti bodů je potřeba napsat studentům krátkou příručku.

2. Teoretická část

2.1. Systém µLab a program rc2000

Výukový systém µLab je navržen pro výuku elektroniky a elektřiny na základních až vysokých školách. Je tvořen počítačem s ovládacím programem rc2000, hardwarovým modulem A&DDU, který je připojen k počítači přes sériové rozhraní RS-232 a v podstatě se jedná o dvoukanálový osciloskop s jedním analogovým výstupem. Dále je součástí systému řada samostatných modulů, které buď jsou aktivní (funkční generátor, zdroj napětí, voltmetr, atd.), nebo pasivní (modul tranzistoru, operačního zesilovače, propojovací modul atd.). Žádný z modulů kromě A&DDU se nedá připojit k počítači, ale napětí z nich se dají měřit pomocí osciloskopických vstupů A&DDU.

A&DDU a rc2000 umožňují pracovat (měřit a generovat napětí) v sedmi různých režimech:

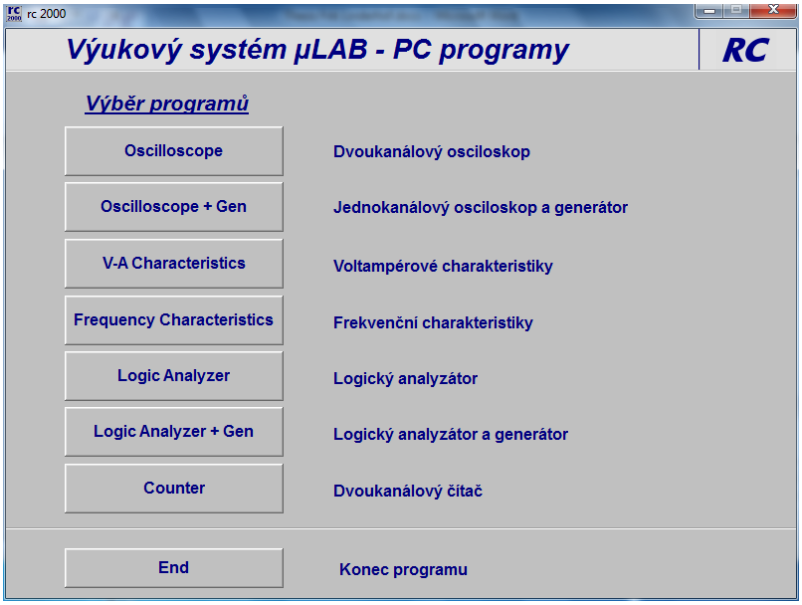
- Oscilloscope (Dvoukanálový osciloskop),
- Oscilloscope + Gen (Jednokanálový osciloskop + generátor),
- V-A Characteristics (Voltampérové charakteristiky),
- Frequency Characteristics (Frekvenční charakteristiky),
- Logic Analyzer (Logický analyzátor),
- Logic Analyzer + Gen (Logický analyzátor + generátor) a
- Counter (Dvoukanálový čítač).

Každý režim tvoří v rc2000 jeden subprogram, dále mají režimy Oscilloscope + Gen a Logic Analyzer + Gen svoje vlastní subprogramy Analog Waveform Editor, respektive Digital Waveform Editor, ve kterých lze vytvářet výstupní signály pro generátory. Většina režimů je analogová (Oscilloscope, Oscilloscope + Gen, V-A Characteristics, Frequency Characteristics and Counter) a ostatní jsou digitální.

Dále se využívá terminologie **skupina** a **podskupina**. V hlavním programu a subprogramech jsou některé funkce, přepínatelné tlačítky, seskupeny pod jednotný název. Takový blok budeme označovat jako *skupinu* (například *Measurement* na obr. 2). Tyto skupiny jsou často ještě rozděleny do menších *podskupin* (například *Sequence* na obr. 2).

2.1.1. Hlavní program

Hlavní program (obr. 1) je v porovnání s ostatními subprogramy relativně malý. Originální hlavní program ukazuje jednoduché menu, odkud je pouze možné spustit další subprogram nebo program zavřít. Hlavní program má jednu nevýhodu, a to, že není možné spustit více subprogramů zároveň, resp. měřit s více moduly A&DDU.



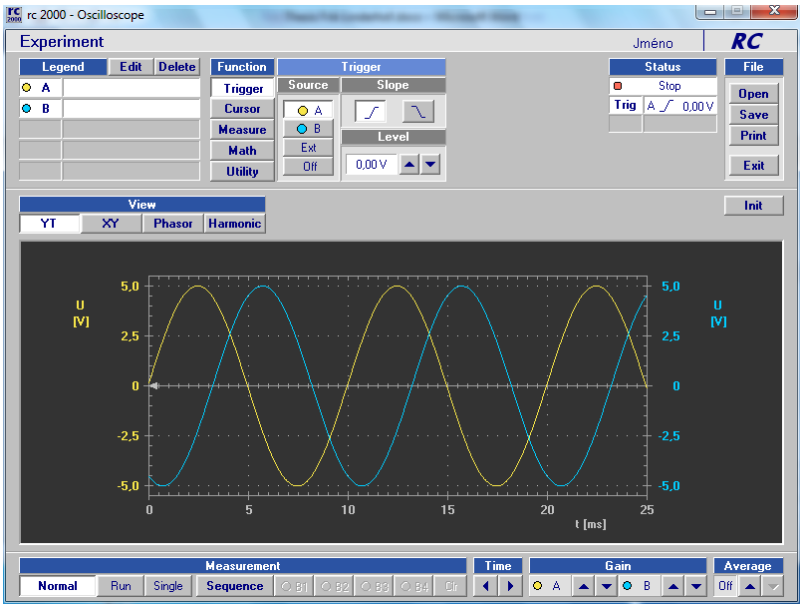
Obrázek 1: Snímek obrazovky originálního hlavního programu rc2000.

2.1.2. Režim *Oscilloscope*

Subprogram *Oscilloscope* je režim virtuálního dvoukanálového osciloskopu (obr. 2). Obsahuje několik skupin s možnostmi přizpůsobit nastavení měření. První skupina je *Measurement*, ve které se nalézají možnosti vlastního měření. Dělí se do dvou podskupin, *Normal* a *Sequence*. V podskupině *Normal* jsou dva způsoby měření, *Run* a *Single*. *Single* je jednorázové měření a *Run* je kontinuální měření. Podskupina *Sequence* nabízí možnost měřit jednorázově a uchovávat a zobrazovat až čtyři různá měření pomocí tlačítek *B1*, *B2*, *B3* a *B4*. Dále nabízí *Clr*, kterým lze uchovávaná měření smazat a inicializovat měření *Sequence*.

Druhá skupina je *Time* se dvěma tlačítky, < a >. Těmito tlačítky lze měřicí dobu přizpůsobit od standardní hodnoty 25 ms do minimální 0,5 ms a maximální 500 s. Celkový počet jednotlivých hodnot nastavení *Time* je 19.

Další skupina *Gain* je opět rozdělena na dvě podskupiny *A* a *B*, odpovídající dvěma vstupním kanálům osciloskopu. V obou podskupinách lze *Gain* přizpůsobit nahoru a dolů a tím měnit vstupní napěťový rozsah. Nastavit lze 7 hodnot, a to 100 mV, 200 mV, 500 mV, 1 V, 2 V, 5 V a 10 V.



Obrázek 2: Snímek obrazovky originálního subprogramu Oscilloscope.

Skupina *Average* nabízí uživateli možnost průměrovat naměřené hodnoty jednoho, dvou, čtyř, osmi nebo šestnácti po sobě jdoucích měření.

View je další skupina a má čtyři nastavení *XT*, *XY*, *Phasor* a *Harmonic*. Skupina určuje způsob zobrazení naměřených dat:

- *XT* je časový rozvoj signálu; v dalších režimech se vyskytuje také, ale jen zkrácený,
- *XY* přidá nový graf se zobrazením kanálu *A* jako osy X a kanálu *B* jako osy Y,
- *Phasor* přidá nový graf se zobrazením fázového posuvu harmonických signálů a
- *Harmonic* přidá nový graf se zobrazením frekvenčního spektra signálu.

Poslední skupina je *Function*, která je rozdělena na 5 podskupin. Pouze první podskupina *Trigger* je relevantní. Trigger (spouštěč) představuje podmínku okamžiku spuštění měření a podskupina *Trigger* je dále rozdělena na tři funkce, *Source*, *Slope* a *Level*. *Source* nabízí uživateli možnost vybrat zdroj triggeru, a to *A* (pomocí *Channel A*), *B* (pomocí *Channel B*), *Ext* (pomocí externího zdroje) a *Off* (bez triggeru). *Slope* nabízí dvě nastavení triggeru zdola nahoru (_/‾) a shora dolů (‾_), či náběžnou nebo sestupnou

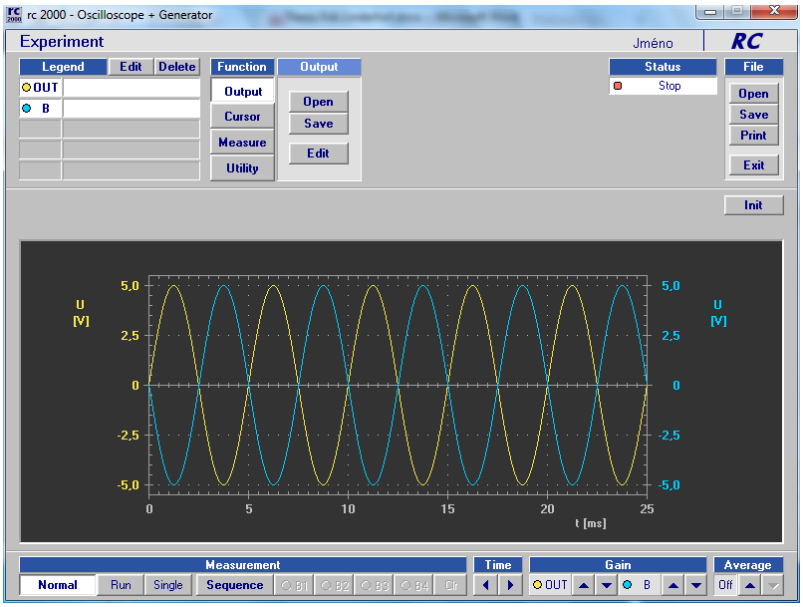
hranu. Poslední nastavení je *Level*, který umožňuje nastavit prahovou hodnotu triggeru. Celkový počet jednotlivých hodnot nastavení *Level* je 221. Krok mezi jednotlivými hodnotami nastavení *Level* závisí na nastavení *Gain*, krok *Level* je jedna setina nastavení *Gain* (když například *Gain* má nastavený rozsah od -5 V do $+5$ V, tak je krok Level 0,05 V, viz tab. 1).

Tabulka 1: Typické hodnoty nastavení Level a související napětí.

		Nastavení *Gain*						
		100 mV	200 mV	500 mV	1 V	2 V	5 V	10 V
Typické hodnoty nastavení *Level*	−110	−110 mV	−220 mV	−550 mV	−1,1 V	−2,2 V	−5,5 V	−11 V
	−100	−100 mV	−200 mV	−500 mV	−1 V	−2 V	−5 V	−5 V
	−1	−1 mV	−2 mV	−5 mV	−0,01 V	−0,02 V	−0,05 V	−0,1 V
	0	0 mV	0 mV	0 mV	0 V	0 V	0 V	0 V
	1	1 mV	2 mV	5 mV	0,01 V	0,02 V	0,05 V	0,1 V
	100	100 mV	200 mV	500 mV	1 V	2 V	5 V	10 V
	110	110 mV	220 mV	550 mV	1,1 V	2,2 V	5,5 V	11 V

2.1.3. Režim *Oscilloscope + Gen*

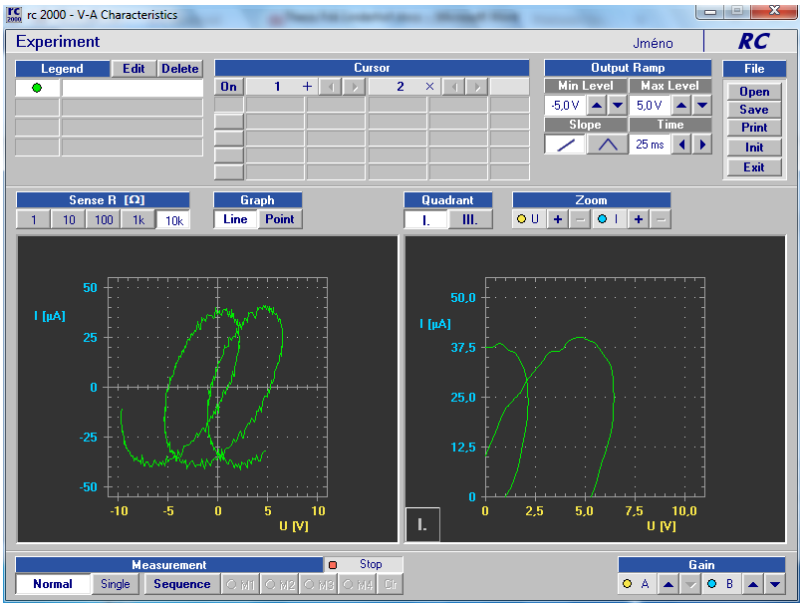
Originální subprogram *Oscilloscope + Gen* je jednokanálový osciloskop a generátor (obr. 3), na rozdíl od subprogramu *Oscilloscope*, což je dvoukanálový osciloskop. Některé skupiny možnosti přizpůsobení nastavení měření jsou jiné, například zde není přítomná podskupina *Trigger* (spouštění je dáno generovaným signálem) a v skupině *Gain* je podskupina *A* nahrazena podskupinou *Out*, která nastavuje rozsah výstupního napětí. Další skupiny jako například *Measurement*, *Time* a *Average* jsou totožné. V porovnání se subprogramem *Oscilloscope* je dále přidána podskupina *Output* s funkcemi *Open*, *Save* a *Edit*. Pomocí funkce *Open* lze vybrat předem vytvořený analogový výstupní signál pro generátor. Funkce *Save* nabízí uživateli možnost uložit změny přímo v subprogramu, jako například nastavení *Time* a *Gain*. Pomocí funkce *Edit* lze otevírat subprogram *Analog Waveform Editor*, v kterém lze další nastavení a průběh výstupního signálu změnit.



Obrázek 3: Snímek obrazovky originálního subprogramu Oscilloscope + Gen.

2.1.4. Režim *V-A Characteristics*

Subprogram *V-A Characteristics* (obr. 4) slouží k měření voltampérových charakteristik součástek a používá některé skupiny stejné jako subprogramy *Oscilloscope* a *Oscilloscope + Gen*. Podskupina *Normal* v skupině *Measurement* má jen jedno tlačítko: *Single*. Skupina *Gain* je stejná jako v subprogramu *Oscilloscope*, neobsahuje podskupinu *Out*. Skupina *Time* je v tom případě podskupina v skupině *Output Ramp*. Další podskupiny této skupiny jsou *Min Level*, *Max Level* a *Slope*. Skupinou *Output Ramp* lze řídit výstupní signál. Další příklady nové skupiny jsou *Quadrant*, s níž je možné vybrat první nebo třetí kvadrant grafu jako sekundární graf, a *Sense R*, s jejíž pomocí lze vybrat velikost rezistoru, sloužícího k převodu proudu na napětí, v hodnotách 1 Ω, 10 Ω, 100 Ω, 1 kΩ nebo 10 kΩ.



Obrázek 4: Snímek obrazovky originálního subprogramu V-A Characteristics.

2.1.5. Režim *Frequency Characteristics*

Subprogram *Frequency Characteristics* (obr. 5) je čtvrtý subprogram, který slouží k měření frekvenčních charakteristik obvodů. Skupina *Measurement* je stejná jako u *V-A Characteristics*. Skupina *Decades* nabízí uživatelům 3 možnosti: 1, 2 a 3, a to možnosti měření jednoho, dvou nebo tří dekád ve frekvenční doméně. Skupina *Begin* také nabízí tři možnosti: 10 Hz, 100 Hz a 1 kHz. Tyto hodnoty jsou počáteční hodnoty frekvence. Nastavení skupin *Decades* a *Begin* jsou vzájemně provázaná (tab. 2). V případě, že je vybráno měření se třemi dekádami, může být počáteční hodnota jenom 10 Hz (celý systém je totiž navržen pro práci se signály do 10 kHz).

Tabulka 2: Možné nastavení skupin Begin a Decades

		Decades		
		1	2	3
Begin	10 Hz	Ano	Ano	Ano
	100 Hz	Ano	Ano	–
	1 kHz	Ano	–	–

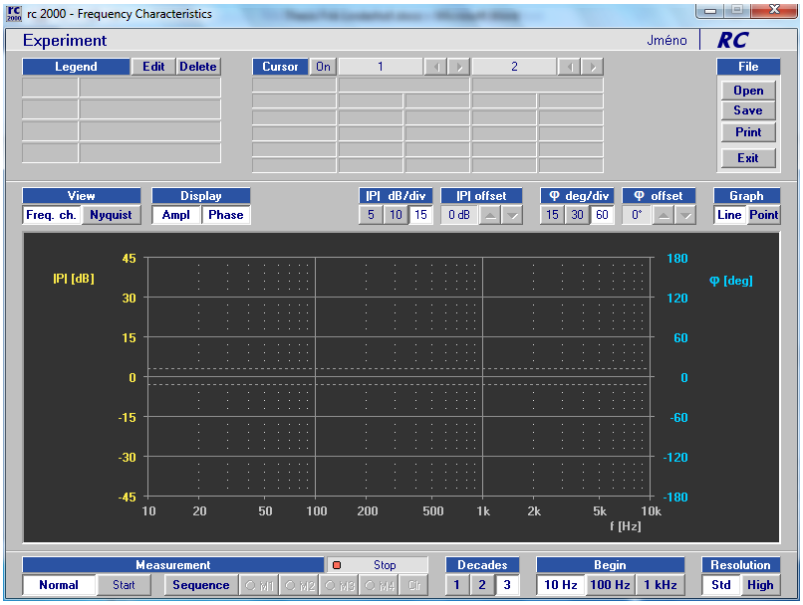
Skupina *Resolution* má dvě nastavení: *Standard* a *High*. Pomocí této skupiny lze změnit celkový počet bodů měření charakteristiky. Počet měření nezávisí jenom na *Resolution*, ale také na skupinách *Begin* a *Decades* (tab. 3).

Tabulka 3: Počet měření na základě nastavení Resolution, Begin a Decades.

***Resolution*: Std**		***Decades***		
		1	2	3
Begin	10 Hz	28	34	27
	100 Hz	27	36	–
	1 kHz	27	–	–

***Resolution*: High**		***Decades***		
		1	2	3
Begin	10 Hz	109	102	73
	100 Hz	104	100	–
	1 kHz	73	–	–

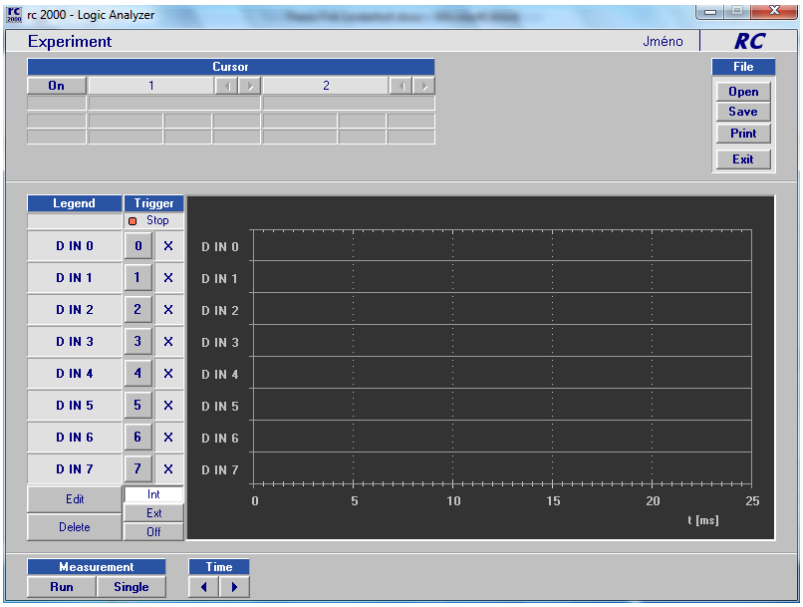
Skupina *View* má, namísto čtyř možností u subprogramu *Oscilloscope*, pouze dvě nastavení – *Freq. Ch.* a *Nyquist*. Skupina *Display* umožňuje vypnout/zapnout zobrazení amplitudy/fáze, ale nejméně jedna musí být zobrazena. Další skupiny $|P|$ dB/div, $|P|$ *offset*, φ deg/div, φ *offset* a *Graph* nejsou důležité.



Obrázek 5: Snímek obrazovky originálního subprogramu Frequency Characteristics.

2.1.6. Režim *Logic Analyzer*

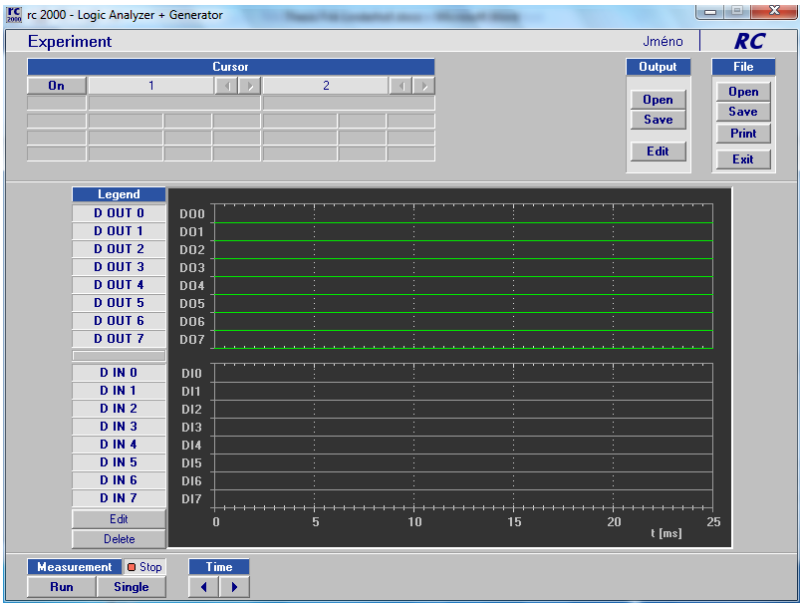
Subprogram *Logic Analyzer* (obr. 6) je první digitální subprogram, zachycují časový průběh osmi logických kanálů. *Logic Analyzer* má jen dvě skupiny *Measurement* a *Time*. Skupina *Measurement* je stejná jako podskupina *Normal* u subprogramů *Oscilloscope* a nabízí uživatelům možnost měření kontinuálně, *Run*, a jednorázově, *Single*. Skupina *Time* je stejná jako u subprogramů *Oscilloscope* a *Oscilloscope + Gen*.



Obrázek 6: Snímek obrazovky originálního subprogramu Logic Analyzer.

2.1.7. Režim *Logic Analyzer + Gen*

Subprogram *Logic Analyzer + Gen* (obr. 7) je stejný jako subprogram *Logic Analyzer* s tím rozdílem, že *Logic Analyzer + Gen* má jednu skupinu navíce: *Output*. Skupina *Output* nabízí tři možnosti: *Open*, *Save* a *Edit* a je stejná jako podskupina *Output* u subprogramu *Oscilloscope + Gen*. Pomocí funkce *Open* lze vybrat předem vytvořený digitální výstupní signál pro generátor. Funkce *Save* nabízí uživateli možnost uložit změny přímo v subprogramu, jako nastavení *Time*. Pomocí funkce *Edit* lze otevírat subprogram *Digital Waveform Editor*, ve kterém lze průběh výstupního signálu změnit.



Obrázek 7: Snímek obrazovky originálního subprogramu Logic Analyzer + Gen.

2.1.8. Režim *Counter*

Poslední subprogram je *Counter* (obr. 8). Skupina *Measurement* je stejná jako skupina *Measurement* subprogramu *Logic Analyzer*. Skupina *Time* má jen 7 kroků od 5 s do 500 *s*. Další dvě skupiny *Range* a *Offset* mají obě dvě podskupiny *T1* a *T2*. Pomocí skupiny *Range* lze přizpůsobit měřicí rozsah v 8 krocích od 300 do 60 000. Pomocí *Offset* lze měřící rozsah posunout s kroky jedné šestiny měřící rozsah, to je 50 při 300 a 10 000 při 60 000. Počet kroků *offset* záleží na nastavení *Range* (tab. 4).

Tabulka 4: Počet kroků a maximální rozsah Offset

Range	**Maximum rozsahu**	**Počet kroků**
300	65 550	1305
600	65 600	650
1200	65 600	322
3000	66 000	126
6000	66 000	60
12 000	66 000	27
30 000	70 000	8
60 000	70 000	1

Skupina *Trigger Source* je podobná podskupině *Source* u subprogramu *Oscilloscope*. Na rozdíl od té podskupiny jsou *A* a *B* nahrazeny *T1* a *T2*.

Obrázek 8: Snímek obrazovky originálního subprogramu Counter.

2.2. Rozhraní RS-232

Komunikace mezi A&DDU a rc2000 je prováděna pomocí rozhraní RS-232. RS-232 je jeden ze standardů sériových portů a používá se jako komunikační rozhraní osobních počítačů a další elektroniky [2–4]. Písmena RS znamenají *radio standard* nebo *recommended standard* (radiový standard nebo doporučený standard). Úrovně signálu ve standardu RS-232 jsou definovány takto: logická jednička je úroveň mezi -3 a -15 V, logická nula je úroveň mezi $+3$ a $+15$ V.

Komunikace pomocí RS-232 může využívat model podobný *master/master* nebo *master/slave*. V případě RS-232 se zařízení typu master nazývají DTE (*Data Terminal Equipme*nt) a zařízení typu slave DCE (*Data Circuit-Terminal Equipment* nebo *Data Communication(s) Equipment*). První možnost, master/master, je komunikace mezi dvěma počítači, kde oba jsou DTE. V tom případě je potřeba křížit některé dráty mezi některými piny pomocí Crossover Cable (křižovací kabel). U druhé možnosti, master/slave je počítač většinou DTE a přístroj, s kterým se pokusí komunikovat, DCE. V případě komunikace mezi DTE a DCE není křížení potřeba a používá se normální kabel. Komunikace mezi A&DDU a počítačem je podobné modelu master/slave.

Pro RS-232 jsou definovány také standardní konektory. Typické zakončení pro DTE je samec a pro DCE samice. Používají se konektory s různým počtem pinů. Existují

například konektory s 25 piny (DB-25), 9 piny (DB-9, obr. 9) a 8 piny (RJ-45). Jednotka A&DDU je vybavena konektorem DB-9. V tab. 5 je přehled pinů konektoru DB-9 a jejich směr u DTE a DCE.

Tabulka 5: Piny RS-232 DB-9.

Pin	Zkratka	Jméno	Směr u DTE	Směr u DCE
1	DCD	Data Carrier Detect	Vstup	Výstup
2	RD	Recieve Data	Vstup	Výstup
3	TD	Transmit Data	Výstup	Vstup
4	DTR	Data Terminal Ready	Výstup	Vstup
5	GND	Ground	–	–
6	DSR	Data Set Ready	Vstup	Výstup
7	RTS	Request to Send	Výstup	Vstup
8	CTS	Clear to Send	Vstup	Výstup
9	RI	Ring Indicator	Vstup	Výstup

Obrázek 9:Umístění pinů u konektoru typu samec (vlevo) a samice (vpravo).

Komunikace pomocí RS-232 může probíhat velkým rozsahem rychlostí. Maximální rychlost (nebo *Baud Rate*), kterou lze dosáhnout pomocí RS-232 připojenému k běžnému sériovému portu počítače, je 115 200 b/s. Rychlost nemusí být v obou směrech stejná, ale když není, je potřeba použít *Flow Control* (řízení toku). Komunikace s A&DDU je limitována na rychlosti 9600, 19 200, 38 400, 57 600 a 115 200 b/s.

Komunikační protokol RS-232 vyžaduje, aby první bit, který se posílá, byl takzvaný *Start Bit*. *Least Significant Bit* (nejméně významný bit) je odesílán jako druhý bit a postupně za ním se posílají všechy *Data Bits* (datové bity), kde *Most Significant Bit* (nejvýznamnější bit) je poslední z nich. Počet datových bitů není striktně definován, ale se může lišit mezi různými komunikačními protokoly. V závislosti na používaném komunikačním protokolu se (ne)posílá takzvaný *Parity bit* (paritní bit). Paritní bit je

jednoduchá kontrola, se kterou lze zkontrolovat správnost datových bitů (při nízké intenzitě poruch). Jako poslední se posílají *Stop Bits* (závěrné bity). Může to být jeden, jeden a půl nebo dva bity.

Dále je možné používat *Flow Control*. Nejznámější formy jsou RTS/CTS a DTR/DSR. RTS/CTS a DTR/DSR jsou porovnatelné v tom, že oba používají dva kanály na nastavení řízení toku, jeden kanál ovládá DTE, druhý DCE. Inicializace začíná na straně DTE, který nastaví jeho DTR, respektive RTS do logické jedné. DTE se pak ptá, jestli DCE může přijmout data. DCE odpovídá, v případě, že může přijmout data, nastavením DSR, respektive CTS do logické jedné. Rozdíl mezi RTS/CTS a DTR/DSR je, že DTR/DSR se nastaví pro celou dobu komunikace, a RTS/CTS se nastaví do logické jedné pro každý datový blok zvlášť. Ovšem tyto formy jsou pouze konvencí, nikoliv standardem a celá řada zařízení je využívá jinak.

2.3. Vývojové prostředí LabVIEW

LabVIEW znamená *Laboratory Virtual Instrumentation Engineering Workbench*. LabVIEW je vývojové prostředí softwaru firmy National Instruments pro její programovací jazyk *G* [5, 6]. G je grafický programovací jazyk na rozdíl od většiny programovacích jazyků, které jsou textové. Jazyk G je vhodný programovací jazyk pro řízení měřicí techniky, získávání dat a komunikaci s měřicími přístroji. Tato práce je prováděna pomocí verze LabVIEW 2013.

Programy, které jsou programovány v LabVIEW, se nazývají *Virtual Instruments* (virtuální přístroje). Každý *Virtual Instrument* se skládá z tří částí: *Front Panel* (přední panel), *Block Diagram* (blokový diagram) a *Connector Panel* (konektorový panel). *Front Panel* je ta část, kterou uživatel vidí. Ukazují se tam data v grafech a v indikátorech a uživatel může ovlivnit chování programu pomocí tlačítek a dalších ovládacích prvků. *Block Diagram* je vlastní programový kód, který uživatel nevidí a programátor pomocí něho mimo jiné spojuje tlačítka s grafy a indikátory a řeší komunikaci s externími přístroji.

Connector Panel je ikona, která ukazuje vstupní a výstupní kanály. Ikony virtuálního přístroje lze používat v *Block Diagramu* dalšího virtuálního přístroje a to dvěma postupy. První postup je jako normální funkce. *Connector Panel* proto umožňuje objektově orientované programování v LabVIEW. Druhý postup je jako subprogram, kde se otevírá *Front Panel* virtuálního přístroje jako nové okno. V prvním postupu nemůže

uživatel přímo ovlivnit fungování virtuálního přístroje, ale fungování je ovlivněno hodnotami na vstupních kanálech. Druhým postupem může uživatel ovlivnit fungování nastavením tlačítek.

V porovnání s dalšími vývojovými prostředími má LabVIEW výbornou pomůcku pro debuggování *Highlight Execution*. Toto nastavení nechá celý program pomalu provádět a v *Block Diagramu* ukazuje výstupní hodnoty funkcí a zvýrazní se pozice, kde se v programu něco děje.

Kompilátor v LabVIEW kompiluje kód automaticky a v případě, že chybí připojení nelze kód provádět. Místo tlačítka *Run* (provádět) nabízí LabVIEW možnost ukázat takzvaný *Error List* (seznam chyb).

Nevýhoda LabVIEW je, že chybí možnost zvětšení a zmenšení. Taková možnost by vylepšila přehlednost velkých programů a zjednodušila dělání snímků obrazovek.

3. Postup a výsledky

3.1. Způsob analýzy komunikace

Analýzu a sledování komunikace po sériové lince lze provádět dvěma metodami:

1. Softwarovou analýzou pomocí programů v počítači, které zachycují a zobrazují komunikaci na sériové lince. Tímto způsobem lze odchytit použité kódy, způsob prezentace dat atd., ale je obtížné zjistit případné časování
2. Hardwarovou analýzou lze pomocí digitálního osciloskopu zachytávat napěťové průběhy přímo na pinech použitých konektorů. K rozumnému použití je ale třeba mít k dispozici osciloskop, který umožní vhodnou prezentaci dat (například převod sledů bitů na hexadecimální hodnotu bytu).

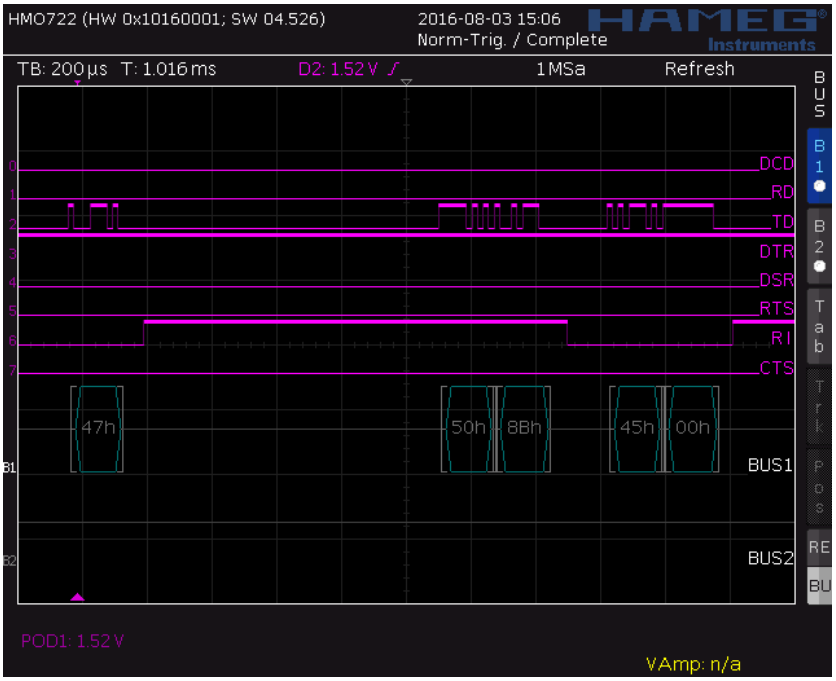
Softwarová analýza byla prováděna pomocí programu *Realterm* (emulátor terminálu), kterým lze posílat data do zařízení a zachycovat odpovědi z něj, a dále programy *Device Monitoring Studio* a *Serial Port Monitor*, které pouze zachycují komunikaci a také nastavení jiných než datových linek.

Pro hardwarovou analýzu bylo třeba zajistit uchycení sond osciloskopu, k čemuž byla použita dvojice spájených konektorů (obr. 10 vlevo). Při prvních pokusech o komunikaci mezi LabVIEW a A&DDU bylo zjištěno, že komunikace nefunguje tak, jako při použití emulátoru terminálu. Analýzou pomocí osciloskopu se ukázalo, že LabVIEW před komunikací krátce nastaví piny 4 a 7 do nevhodné logické úrovně, čímž zamezí další komunikaci s modulem. Podle literatury se zřejmě jedná o *purge puls* (čisticí pulz) na všech výstupech RS-232, které LabVIEW samo vkládá. Jako prozatímní řešení byla vyrobena propojka (obr. 10 vpravo), kde dráty spojující piny 4 a 7 jsou na straně DCE spojeny s pinem 1, který je v naší komunikaci trvale ve vhodné logické úrovni. Pomocí těchto dvou pinů se neodesílá žádná důležitá informace, a proto neovlivní komunikaci špatným způsobem.

Obrázek 10: Propojky konektorů umožňující odposlech (vlevo) a s propojenými piny(vpravo).

3.2. Studium komunikačního protokolu

Ke studiu komunikačního protokolu byly používány hardware (osciloskop HAMEG Rohde & Schwarz HM0722, snímek obrazovky osciloskopu na obr. 11) a software k odposlechnutí signálu.



Obrázek 11: Snímek obrazovky prvních tří parametrů inicializačního kódu subprogramu Oscilloscope pomocí HAMEG Rohde & Schwarz HM0772.

Ukazuje se, že řízení toku je zde nestandardní – kontrolování je jen jednosměrné. Místo toho, že se používá RTS/CTS nebo DTR/DSR, změní se stav CTS od logické jedné do logické nuly a naopak po každém bloku dat.

Protože není k dispozici podrobná dokumentace, je nutné zjistit význam kódů. To lze provádět tak, že se v originálním programu postupně využívají funkce s různými

parametry/nastavením a pomocí odposlechu se zjišťuje, kde se kódy mění a jaké se používají. Dobrým příkladem, jak postupně je určován význam různých kódů, je dekódování triggerů. V tab. 6 jsou uvedeny čtyři sekvence kódů, kterými lze měnit nastavení zdroje triggeru. V případě, že se porovnají první dva kódy, které nastaví jako zdroj triggeru kanál *A*, respektive kanál *B*, je vidět, že jen u jednoho parametru, označeného červeně, se kód mění. Všechna důležitá informace proto musí být určena tím parametrem. Parametr 45[1] určuje zdroj triggeru, kde kód 45 00 nastaví kanál *A* jako zdroj triggeru a kód 45 01 nastaví kanál *B* jako zdroj triggeru.

Když srovnáme předchozí sekvence se sekvencí pro vypnutí triggeru (Off), vidíme, že kód za Off je kratší. Dále vidíme, že se mění jen jeden parametr, označený zeleně, což znamená, že všechna důležitá informace musí být nastavena tímto parametrem. Kód 43 určuje zřejmě stav triggeru, kde kód 43 00 zapne trigger a kód 43 01 vypne trigger. Poslední sekvence kódu pro externí trigger (Ext) je stejně jako sekvence kódů pro Off kratší a dále se liší dvěma parametry označenými žlutě a modře: 4B a 50. Poslední parametr z těch dvou, 50, se ale také vyskytuje v kódech pro startování subprogramů. S největší pravděpodobností lze ovládat nastavení externí/interní triggerování parametrem 4B, kde kód 4B 00 nastaví interní trigger a kód 4B 01 nastaví externí trigger.

Tabulka 6: Nastavení triggeru a odpovídající kód

Nastavení	**Odpovídající kód**
A	53 | 4A 80 | 45 00 | 4B 00 | 43 00 | 59 | 50 8B
B	53 | 4A 80 | 45 01 | 4B 00 | 43 00 | 59 | 50 8B
Ext	53 | 4B 01 | 43 00 | 59 | 50 89
Off	53 | 4B 00 | 43 01 | 59 | 50 8B

V tab. 6 zbývají ještě tři kódy, které nebyly v předchozím odstavci identifikovány, označené černě, a to 53, 4A 80 a 59. Podíváme se proto na další nastavení, které tyto kódy může ovlivnit. První je nastavení hrany triggeru, která může být náběžná nebo sestupná. Tab. 7 ale ukazuje čtyři nastavení. Ukazuje se, že sekvence kódu odpovídající hraně triggeru je také závislá na zdroji triggeru. Mezi těmito čtyřmi kódy se mění dva parametry, a to 45 a 59/5A. O parametru 45 je známo, že se týká zdroje triggeru, proto musí parametr 59/5A, označený fialově, odpovídat hraně triggeru. To je zajímavé, protože se ukazuje, že

[1] Všechny kódy, parametry a sekvence kódů v této práci jsou dané v hexadecimální soustavě.

existují dva způsoby, jak mohou parametry nést informaci. První způsob je sufixem a používá se například v parametrech 43, 45, 4B a 50. Druhý způsob je změna vlastního parametru a dobrý příklad je 59/5A.

Tabulka 7: Nastavení hrany triggeru a odpovídající kód

Nastavení	**Odpovídající kód**
A a _/‾	53 | 4A 80 | 45 00 | 4B 00 | 43 00 | 59
A a ‾_	53 | 4A 80 | 45 00 | 4B 00 | 43 00 | 5A
B a _/‾	53 | 4A 80 | 45 01 | 4B 00 | 43 00 | 59
B a ‾_	53 | 4A 80 | 45 01 | 4B 00 | 43 00 | 5A

Pomocí nové informace získané v předchozím odstavci se znovu podíváme na kódy nastavení zdroje triggeru v tab. 6. Ukazuje se, že kódy jsou variabilní a závisí na ostatních nastaveních; parametr 59/5A nezůstává stejný, ale je taky závislý na dřívějších nastaveních (tab. 8). Dále je vidět, že stále je význam parametrů 53 a 4A neznámý.

Tabulka 8: Nastavení zdroje triggeru a odpovídající kód

Nastavení	**Odpovídající kód**
A a _/‾	53 | 4A 80 | 45 00 | 4B 00 | 43 00 | 59 | 50 8B
A a ‾_	53 | 4A 80 | 45 00 | 4B 00 | 43 00 | 5A | 50 8B
B a _/‾	53 | 4A 80 | 45 01 | 4B 00 | 43 00 | 59 | 50 8B
B a ‾_	53 | 4A 80 | 45 01 | 4B 00 | 43 00 | 5A | 50 8B

Pro dešifrování parametru 4A se podíváme na sekvence kódů pro nastavení úrovně triggeru. Tab. 9 ukazuje příklad tří nastavení úrovně triggeru. Stejně jako u předchozích příkladů se mění kódy 45 a 59/5A v případě, že jako zdroj triggeru je místo kanálu *A* nastaven kanál *B* a v závislosti na nastavení hrany. Nové v kódu ale je, že se dvakrát opakují parametry: 4A a 43. Důvod pro tento fakt není jasný. Parametr 43 má dvakrát přesně stejný kód, 43 00 a význam není zřejmý. Parametr 4A má dvakrát různý kód, kde první je vždycky starý kód starého nastavení a druhý nový kód nového nastavení.

Tabulka 9: Nastavení úrovně triggeru a odpovídající kód (nahoru).

Staré nastavení	**Nové nastavení**	**Odpovídající kód**
0,00 V	0,05 V	4A 80 | 43 00 | 4A 81 | 45 00 | 4B 00 | 43 00 | 59
0,05 V	0,10 V	4A 81 | 43 00 | 4A 82 | 45 00 | 4B 00 | 43 00 | 59
0,15 V	0,15 V	4A 82 | 43 00 | 4A 83 | 45 00 | 4B 00 | 43 00 | 59

Tabulka 10: Nastavení úrovně triggeru a odpovídající kód (dolů).

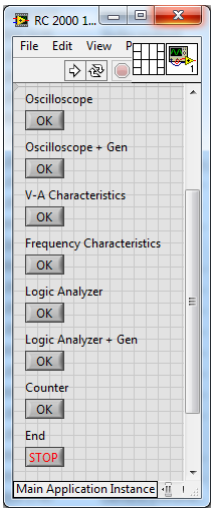
Staré nastavení	Nové nastavení	Odpovídající kód
0,15 V	0,10 V	4A 83 | 43 00 | 4A 82 | 45 00 | 4B 00 | 43 00 | 59
0,10 V	0,05 V	4A 82 | 43 00 | 4A 81 | 45 00 | 4B 00 | 43 00 | 59
0,05 V	0,00 V	4A 81 | 43 00 | 4A 80 | 45 00 | 4B 00 | 43 00 | 59

3.3. Nový program v LabVIEW

Stejně jako originální program rc2000 je nový program v LabVIEW rozdělen do subprogramů odpovídajících výše uvedeným režimům. Celý program je napsán tak, aby se daly používat soubory **.aio* a **.dio*, které jsou používané pro generování výstupních signálů, vytvářené původním programem rc2000.

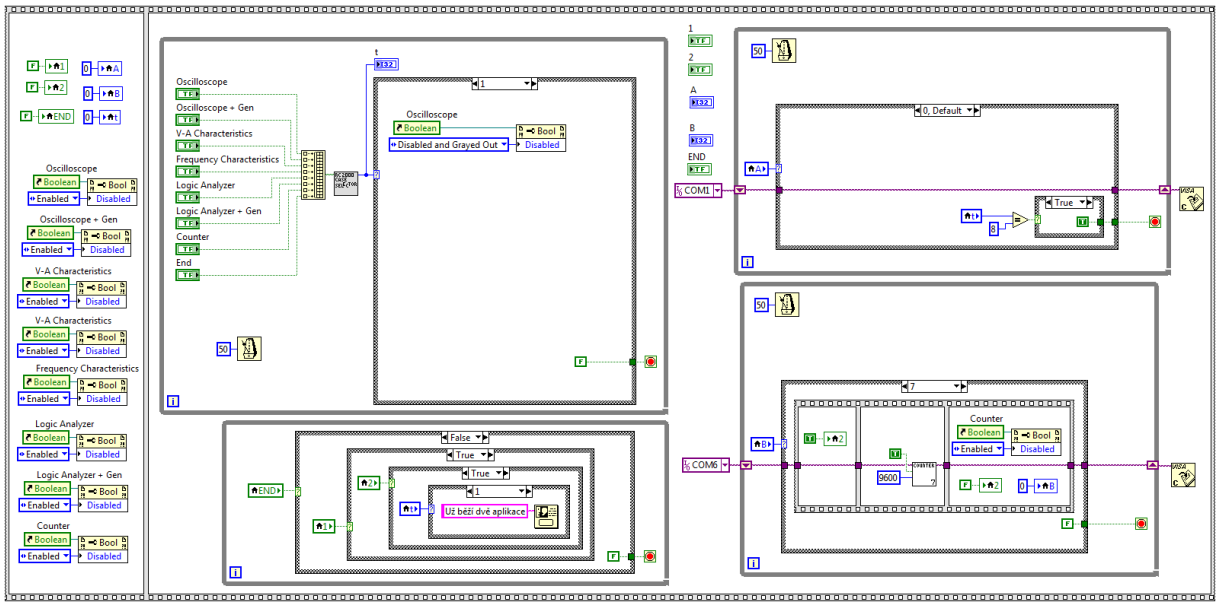
3.3.1. Hlavní program

V novém programu (obr. 12) je možné spustit dva subprogramy zároveň. Není ale možné spustit stejný subprogram dvakrát. Dále funguje stejně jako originální hlavní program.



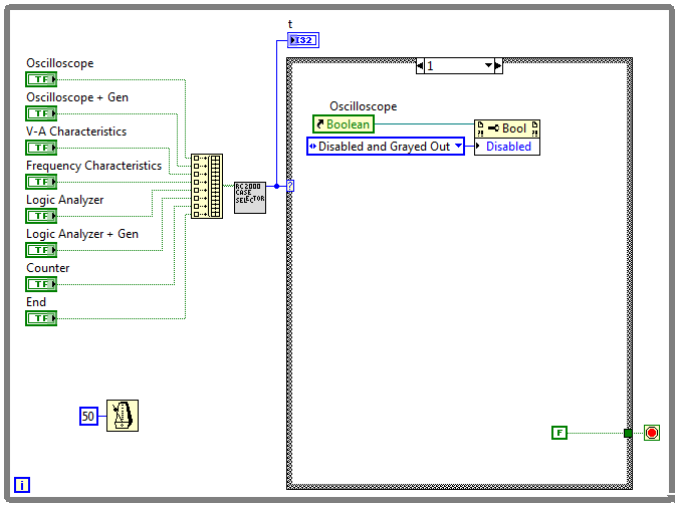
Obrázek 12: Snímek obrazovky Front Panelu nového hlavního programu.

Kód hlavního programu (obr. 13) se skládá ze dvou částí: inicializace (vlevo) a vlastní hlavní program (vpravo). V části inicializace se vynulují všechny proměnné. Vlastní hlavní program je komplexnější a se skládá ze čtyř procesů, které probíhají zároveň.



Obrázek 13: Blokový diagram nového hlavního programu.

První proces (obr. 14) kontroluje každých 50 ms, jestli bylo nějaké z tlačítek stlačeno. V případě, že bylo, bude dané tlačítko deaktivováno, zašednuto a umožní spuštění příslušného subprogramu. První proces potom pokračuje kontrolováním, jestli bylo stlačeno nějaké z dalších tlačítek.



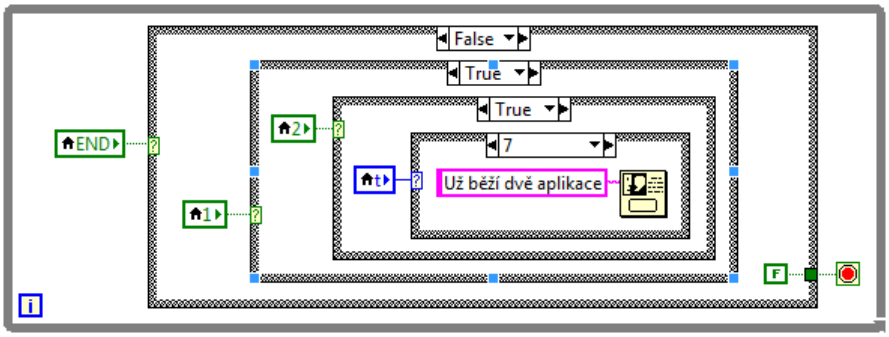
Obrázek 14: Blokový diagram prvního procesu nového hlavního programu.

Druhý proces (obr. 15) běží zároveň vedle prvního procesu a skládá se z různých podmínek (if-loop). Od vnější podmínky dovnitř se testuje: jestli je stlačeno tlačítko *End*, jestli *boolean 1* je pravda, jestli *boolean 2* je pravda a jestli je znovu stlačeno jedno ze zbývajících tlačítek. V případě, že je tlačítko *End* stlačeno, tento proces se zavře a ukončí

celý program, v případě, že tlačítko není stlačeno, kontroluje se další podmínka. V té se zkontroluje, jestli *boolean 1* je pravda (to znamená, že ve *slotu 1* už „běží“ subprogram).

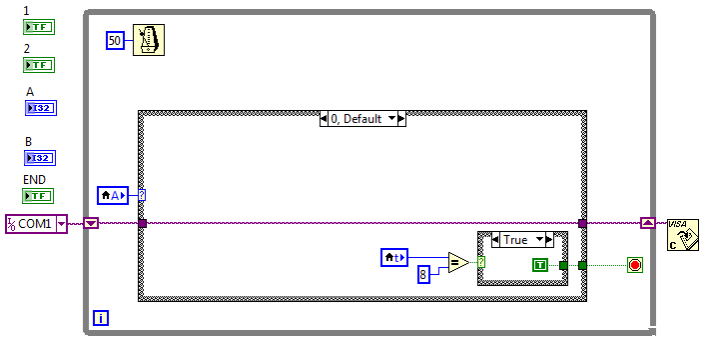
- V případě, že *boolean 1* není pravda, bude se *integer A* rovnat *integer t*, kde *integer t* je celé číslo, které reprezentuje vybraný subprogram a *integer A* reprezentuje první *slot* pro nastartování subprogramu.
- V případě, že *boolean 1* je pravda, otestuje se další podmínka, jestli *boolean 2* je pravda. V případě, že *boolean 2* není pravda, bude se *integer B* rovnat *integer t*, v případě, že *boolean 2* je pravda, druhý slot už je používán a ukáže se hlášení chyby, že už dvě aplikace běží.

Na druhý proces se lze dívat jako na telefonního operátora ze starých časů: první proces ho volá a on rozhodne, se kterou telefonní linkou zavolaný subprogram spojí.



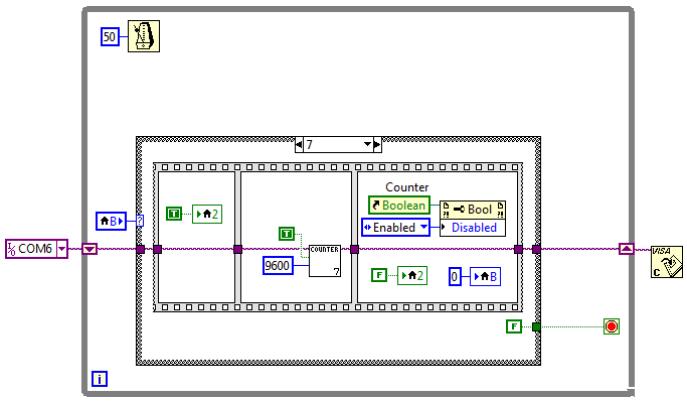
Obrázek 15: Blokový diagram druhého procesu nového hlavního programu.

Třetí proces (obr. 16) je zodpovědný za vlastní spuštění subprogramu na prvním COM portu. Proces začíná před smyčkou *While* inicializací COM portu. Potom vstoupí do smyčky a testuje se *integer A*. V případě, že je *integer A* nulový, zkontroluje se, jestli se hodnota *integer t* rovná 8. V případě, že ano, ukončí se proces, v případě, že ne, proces se zopakuje a znovu se testuje *integer A*. V případě, že *integer A* je teď v intervalu 1 až 7, tak se nastartuje specifický subprogram. Děje se to ve třech krocích. V prvním kroku se nastaví *boolean 1* na pravdu, v druhém kroku se nastartuje subprogram, a to *Oscilloscope*, když *integer A* je 1, *Oscilloscope + Gen*, když *integer A* je 2, atd. Tento proces pokračuje, dokud není subprogram uzavřen. V posledním kroku se znovu aktivuje tlačítko specifického subprogramu, vynuluje se *integer A* a nastaví se *boolean 1* do nepravdy. Když ale *integer A* byl 8, tak se proces ukončí.



Obrázek 16: Blokový diagram třetího procesu nového hlavního programu.

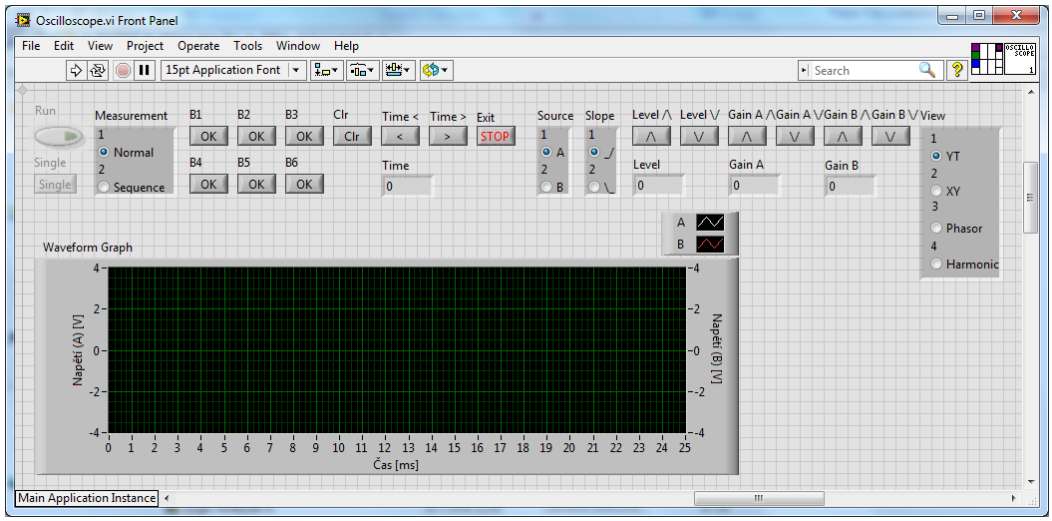
Čtvrtý proces (obr. 17) je zodpovědný za startování subprogramu na druhém COM portu. Čtvrtý proces funguje úplně stejně jako třetí proces, jen jsou některé parametry jiné. *Integer A* je nahrazen *integrem B* a *Boolean 1* je nahrazen *Booleanem 2*.



Obrázek 17: Blokový diagram čtvrtého procesu nového hlavního programu.

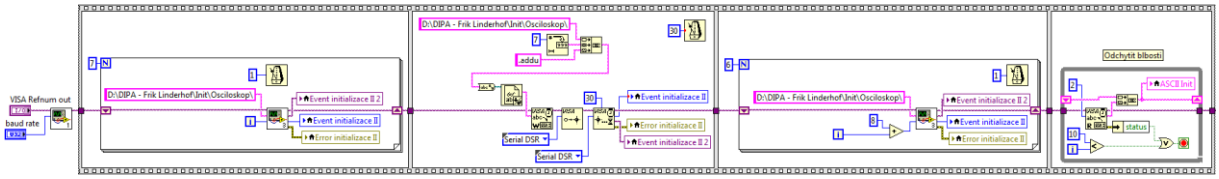
3.3.2. Režim *Oscilloscope*

Nový subprogram *Oscilloscope* (obr. 18) kopíruje největší část možností přizpůsobení měření podle originálního programu. Nejdůležitějším rozdílem je rozšíření podskupiny *Sequence* dvěma tlačítky *B5* a *B6*, čímž je umožněno ukládání 6 měření, které zjednoduší měření třífázových soustav. Další rozšíření je možnost nastartovat subprogram zvlášť, na rozdíl od všech originálních subprogramů, které jen mohou být nastartovány pomocí hlavního programu.



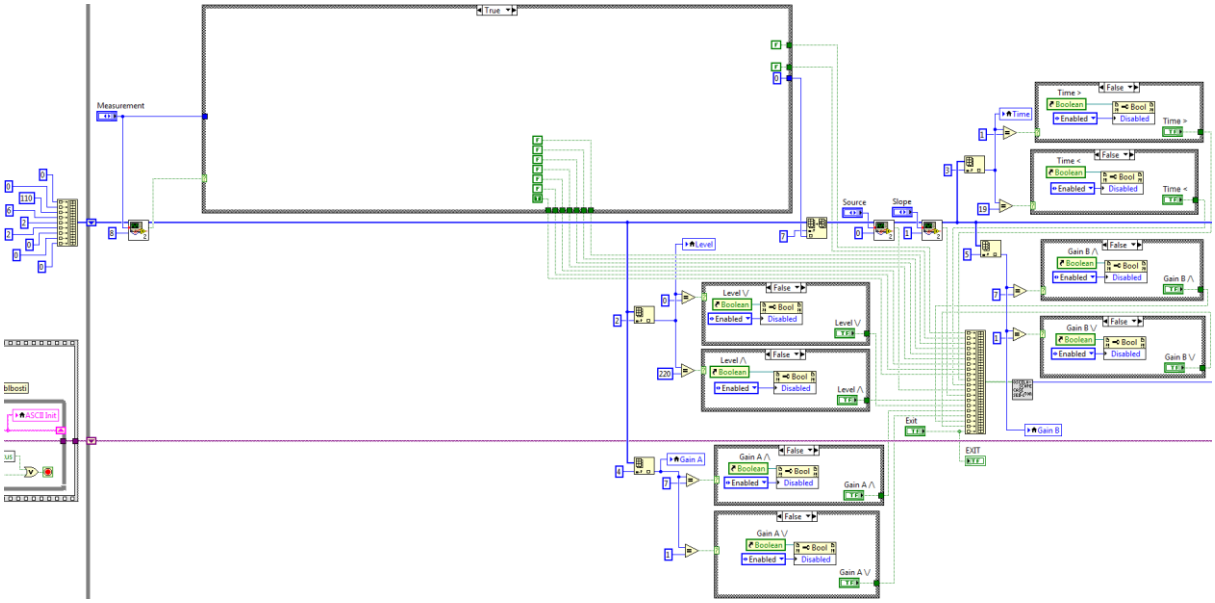
Obrázek 18: Snímek obrazovky Front Panelu nového subprogramu Oscilloscope.

Vlastní program nového subprogramu *Oscilloscope* je rozdělen do dvou procesů, vlastní program a zobrazování. Vlastní program lze rozdělit na tři části, první část je inicializace, druhá vlastní program a třetí část je ukončení subprogramu. Inicializace (obr. 19) se skládá ze čtyř kroků. První krok je inicializace COM portu a je prováděn pomocí SubVI *Initializace Serialoveho portu (+error handling).vi*. Další tři kroky se týkají časování a posílání inicializačních kódů do A&DDU. Poslední krok je určen k čtení odpovědi z A&DDU.



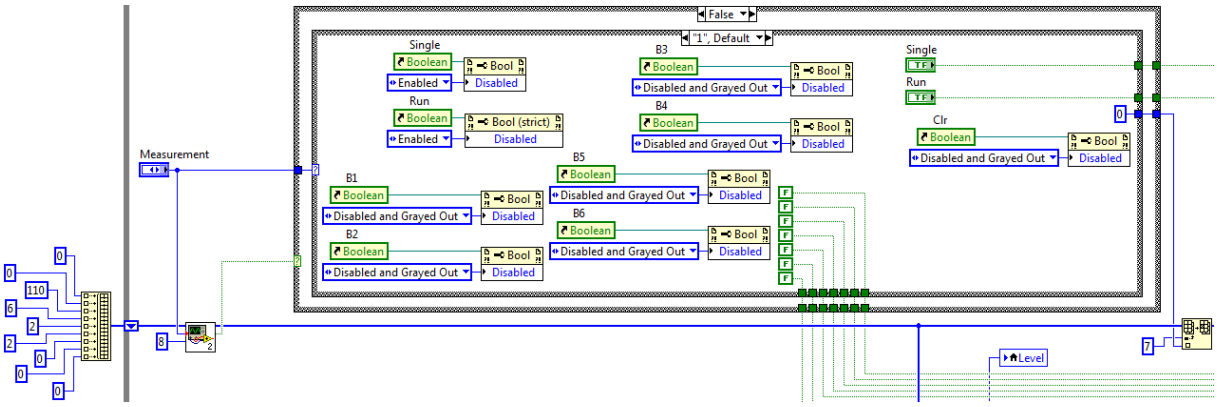
Obrázek 19: Blokový diagram první části prvního procesu nového subprogramu Oscilloscope.

Druhá část (obr. 20) je rozdělena do dvou kroků. První krok je určen k stanovení, který proces musí být prováděn v druhém kroku. Všechna tlačítka (mimo *View*, které řídí druhý proces) odtud ovládají průběh subprogramu.



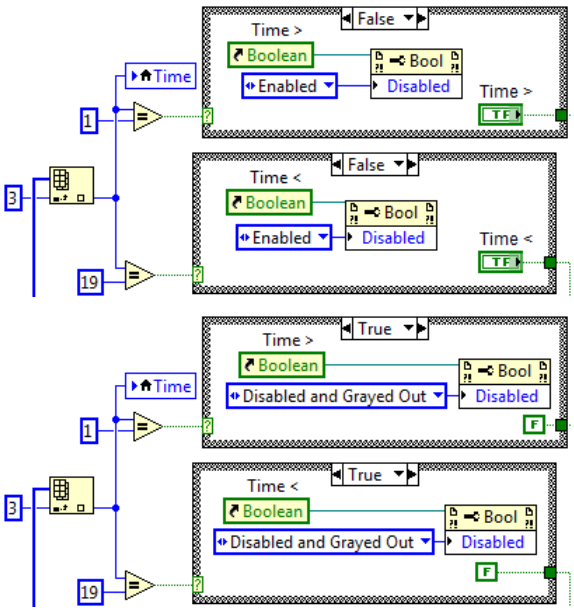
Obrázek 20: Blokový diagram prvního kroku druhé části prvního procesu nového subprogramu Oscilloscope.

První skupina tlačítek je skupina *Measurement* (obr. 21). Tato skupina je rozdělena do dvou podskupin *Normal* a *Sequence*. Přepínání mezi podskupinami se děje na *Front Panelu* pomocí *Radio Buttons*. Po každé se pomocí SubVI *Array Comparator.vi* kontroluje, jestli se stav *Radio Buttons* změnil. V případě, že ano, je prováděna část *pravda* podmínky a je v druhém kroku vyvolán druhý *Case* (2: *Measurement / Clear*). V případě, že se stav *Radio Buttons* nezměnil, je prováděna část *nepravda* podmínky. V této části se nachází další podmínka, která souvisí s podskupinami *Normal* a *Sequence*. V případě, že tlačítko *Measurement* má hodnotu 1, je prováděna podskupina *Normal*. Tlačítka *B1*, *B2*, *B3*, *B4*, *B5*, *B6* a *Clr* jsou deaktivována a zašednuta a tlačítka *Run* a *Single* jsou aktivována. V případě, že tlačítko *Measurement* má hodnotu 2, je prováděna podskupina *Sequence*. V tom případě je stav tlačítek opačný.



Obrázek 21: Blokový diagram skupiny Measurement prvního kroku druhé části prvního procesu nového subprogramu Oscilloscope.

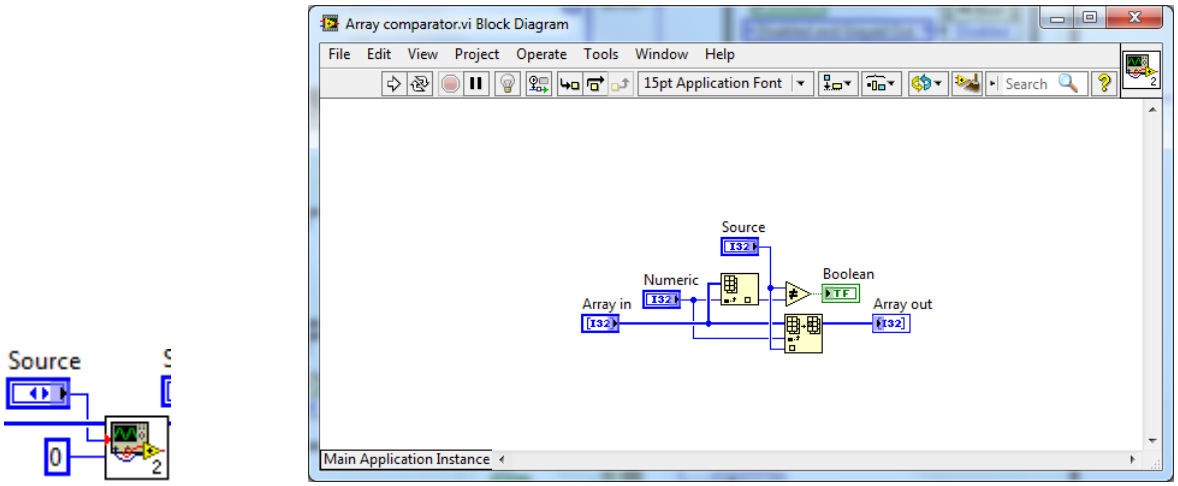
Další skupina tlačítek je skupina *Time* (obr. 22). V skupině se nachází jen dvě tlačítka, < a >. Cíl této části subprogramu je deaktivování a zašednutí tlačítek v případě, že je dosažena mezní hodnota. To je 1 (25 ms) pro < a 19 (500 s) pro >. Stejným způsobem fungují také podskupiny *Trigger - Level*, *Gain - A* a *Gain - B*.



Obrázek 22: Blokový diagram skupiny Time prvního kroku druhé části prvního procesu nového subprogramu Oscilloscope.

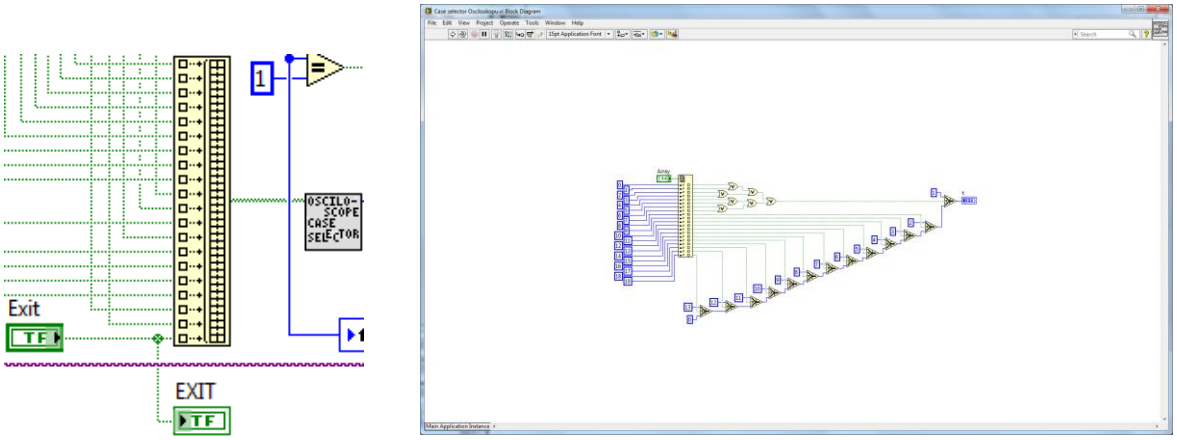
Jako poslední skupina tlačítek je podskupina *Trigger – Source*. V této části se jen vyvolá SubVi *Array comparator.vi* (obr. 23), který porovnává již uloženou hodnotu *Source* se současnou hodnotou. V případě, že tyto dvě hodnoty jsou stejné, nic se neděje,

v případě, že tyto dvě hodnoty jsou odlišné, je posílána *pravda* do SubVI *Case selector Osciloskopu.vi*. Podskupina *Trigger – Slope* funguje stejným způsobem.



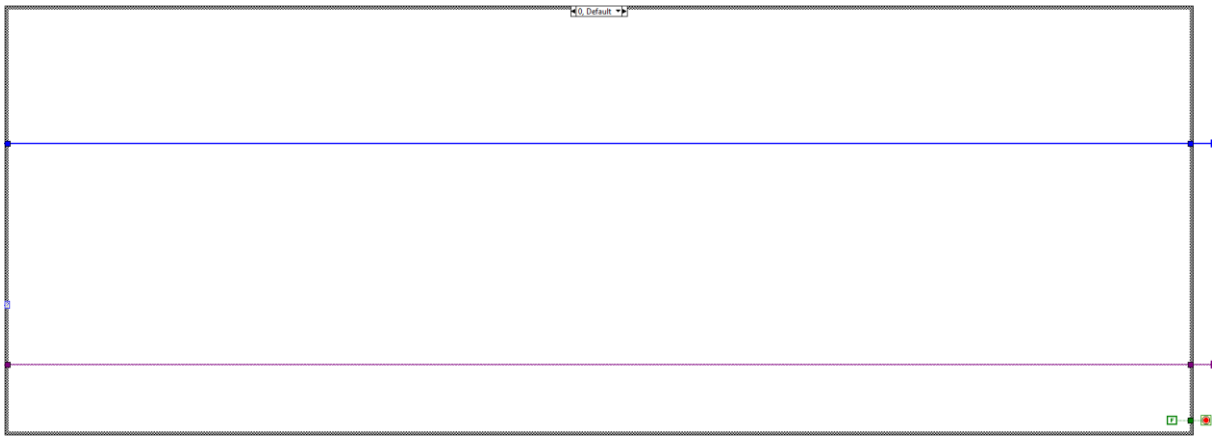
Obrázek 23: Vlevo) Blokový diagram skupiny Source prvního kroku druhé části prvního procesu nového subprogramu Oscilloscope, vpravo) Blokový diagram SubVI Array comparator.vi.

Všechna předchozí tlačítka a tlačítko *Exit* po úpravě nabízí *Case Selectoru* (obr. 24) logické hodnoty *pravda* / *nepravda*. Všechny tyto hodnoty jsou uloženy do pole, pomocí kterého *Case Selector* vybere *Case* pro druhý krok druhé části prvního procesu.



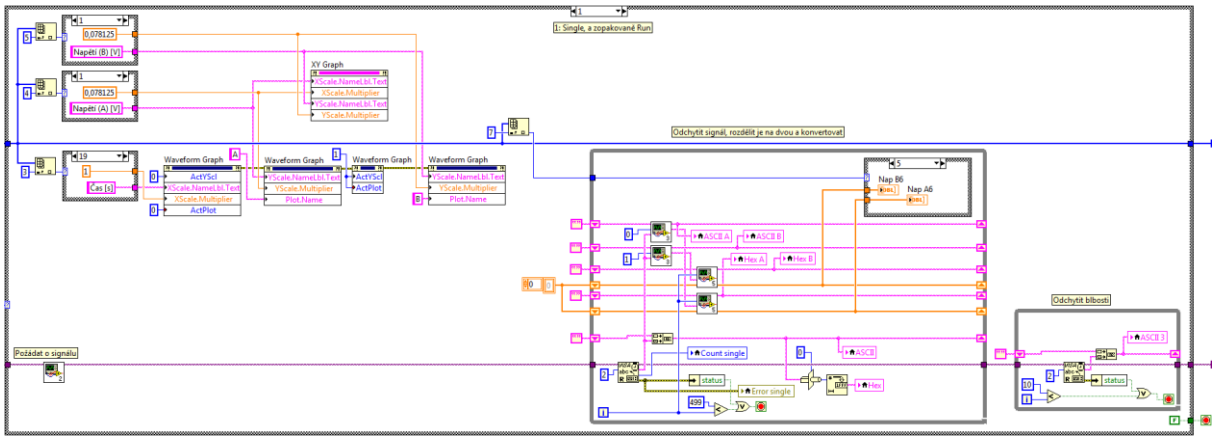
Obrázek 24: Vlevo) Blokový diagram Case Selector prvního kroku druhé části prvního procesu nového subprogramu Oscilloscope, Vpravo) Blokový diagram Case SubVI Selector Osciloskopu.vi.

Druhý krok druhé části je takzvaný *Case structure*. Pomocí *Case selector* v prvním kroku ho řídí. *Case structure* se skládá z 14 různých případů. První *case* je *case 0* (obr. 25) a je *default*, provádí se v případě, že žádný další *case* není vybrán.



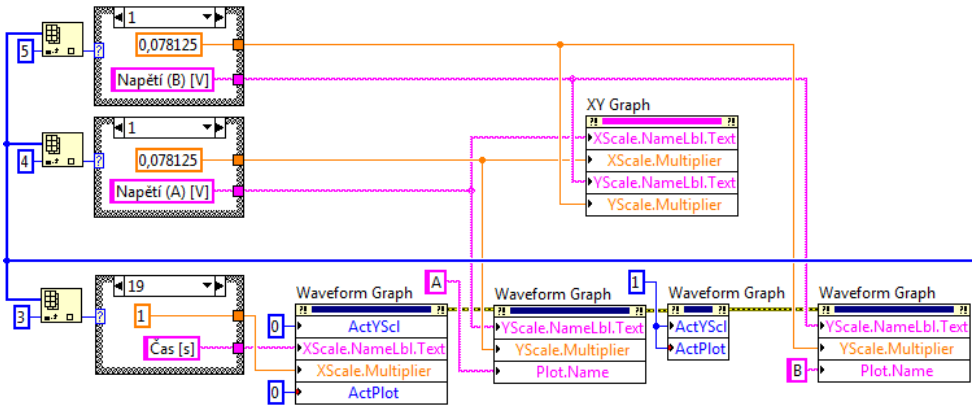
Obrázek 25: Blokový diagram Case 0 druhého kroku druhé části prvního procesu nového subprogramu Oscilloscope.

Case 1 (obr. 26) je vlastní měření osciloskopu a lze ho rozdělit do čtyř bloků, a to *přizpůsobení škál*, *žádost o měření*, *vlastní měření* a *ukončení*.



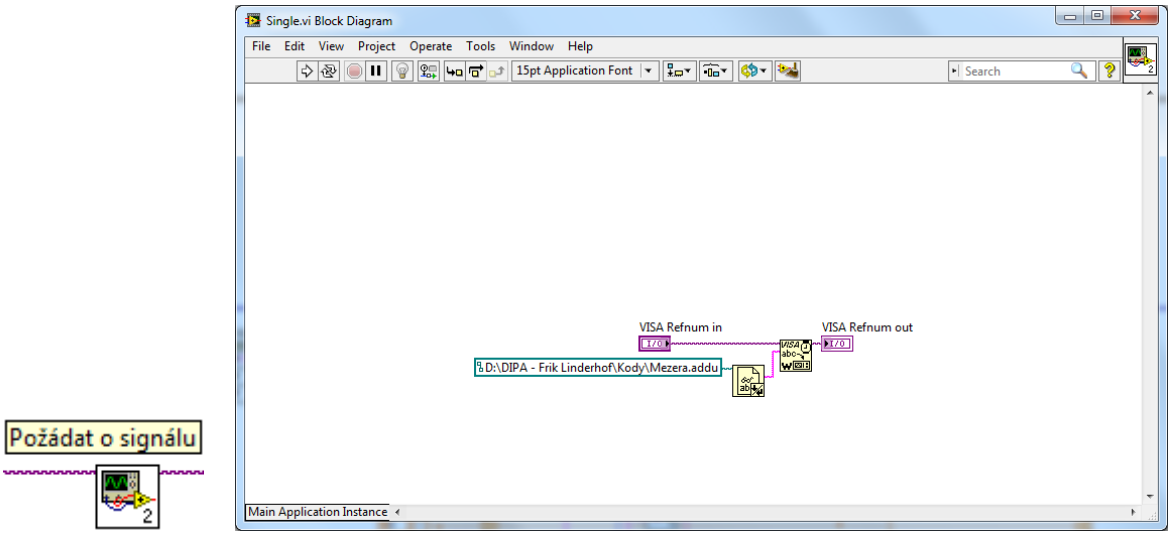
Obrázek 26: Blokový diagram Case 1 druhého kroku druhé části prvního procesu nového subprogramu Oscilloscope.

V bloku přizpůsobení škál je pomocí tří *Case structure* a *Index Array Function* zjištěno současné nastavení *Gain A*, *Gain B* a *Time*, a související hodnoty v (mili-)voltech a (mili-)sekundách. Tyto hodnoty jsou potom pomocí *Property Node* spojeny s grafy (obr. 27).



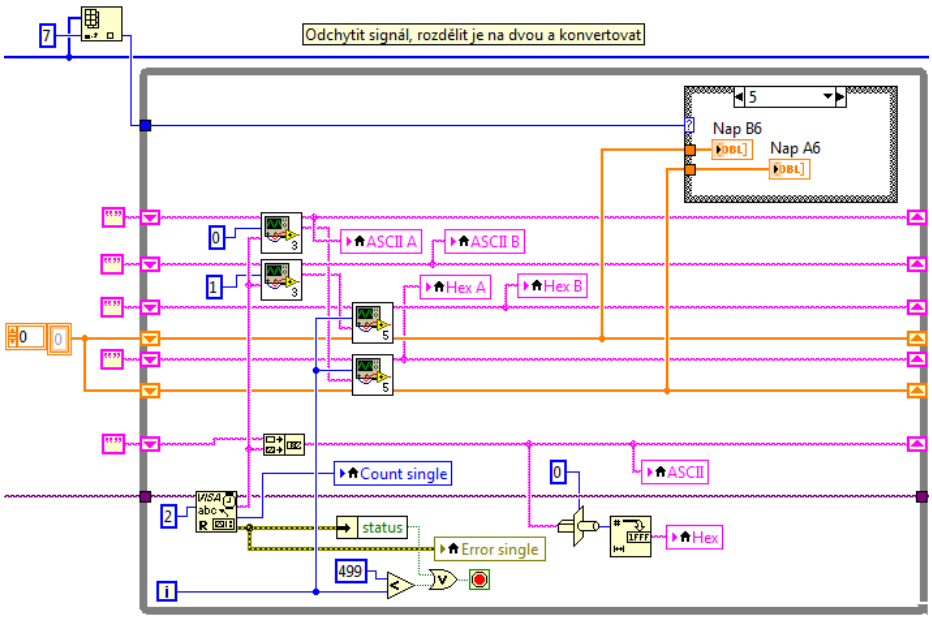
Obrázek 27: Blokový diagram prvního bloku Case 1 druhého kroku druhé části prvního procesu nového subprogramu Oscilloscope.

Další blok je žádost o měření a je posílána z SubVI *Single.vi* (obr. 28), které pošle do A&DDU hexadecimální kód 20 (kód žádosti o měření).



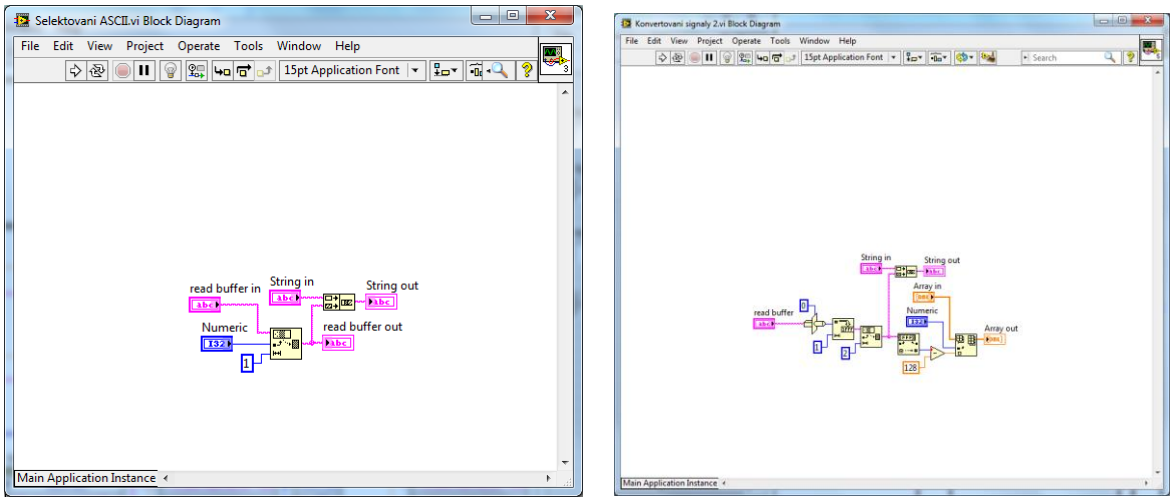
Obrázek 28: Vlevo) Blokový diagram druhého bloku Case 1 druhého kroku druhé části prvního procesu nového subprogramu Oscilloscope, vpravo) Blokový diagram SubVI Single.vi.

Vlastní měření je prováděno ve čtvrtém bloku (obr. 29). Základ je *VISA read function*, která postupně čte data posílaná z A&DDU. Čtená data jsou uložena jako ASCII znaky.



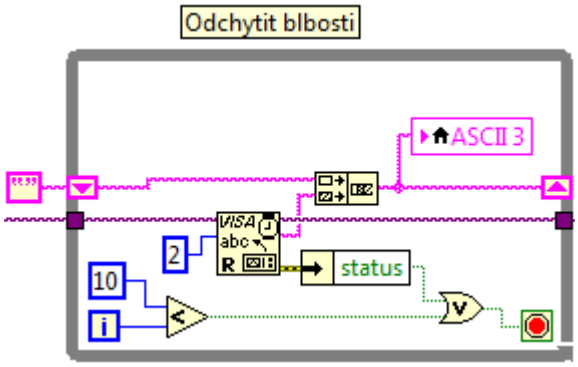
Obrázek 29: Blokový diagram třetího bloku Case 1 druhého kroku druhé části prvního procesu nového subprogramu Oscilloscope.

Tyto ASCII znaky jsou rozděleny do dvou skupin, A a B, pomocí SubVI *Selektovani ASCII.vi* (obr. 30 vlevo), protože modul A&DDU posílá data z obou kanálů prokládaně, vždy jeden byte jako jedna naměřená hodnota. Následně jsou data konvertována na hexadecimální a decimální hodnoty pomocí SubVI *Konvertovani signaly 2.vi* (obr. 30 vpravo). Všechny hodnoty jsou uloženy do různých polí. Pomocí *Case Structure*, ve čtvrtém bloku, se nakonec uloží decimální hodnoty do polí *Nap A1* a *Nap B1*, v případě, že měření bylo inicióváno pomocí tlačítka *Single, Run* nebo *B1*. V ostatních případech, kdy měření bylo inicióváno pomocí tlačítka *B2, B3, B4, B5* nebo *B6*, se mění jména polí do *Nap A2* / *Nap B2*, *Nap A3* / *Nap B3* atd.



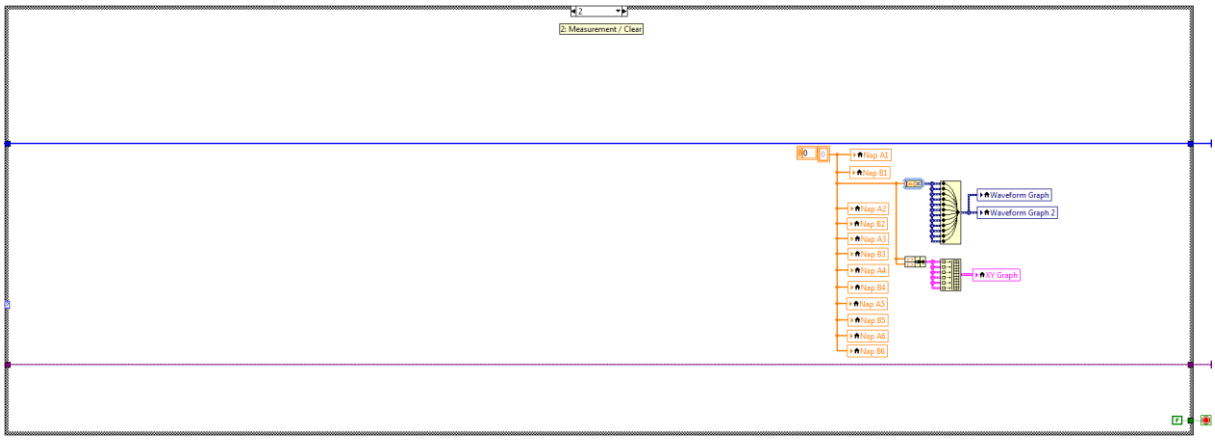
Obrázek 30: Vlevo) Blokový diagram SubVI Selektovani ASCII.vi, vpravo) Blokový diagram SubVI Konvertovani signaly 2.vi.

Poslední blok, ukončení (obr. 31), zpracuje po vlastním měření ve třetím bloku zbylá data poslaná z A&DDU.



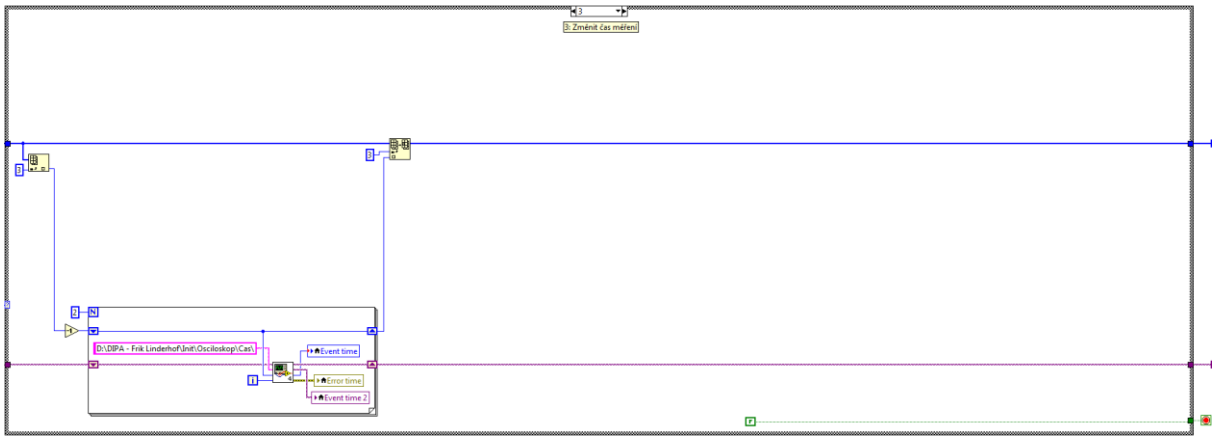
Obrázek 31: Blokový diagram čtvrtého bloku Case 1 druhého kroku druhé části prvního procesu nového subprogramu Oscilloscope.

Case 2 (obr. 32) provádí jak vymazání dat uložených v *double Nap A1* / *Nap B1* až *Nap A6* / *Nap B6*, tak vymazání grafů. Tento *case* je prováděn po stlačení tlačítka *Clr* nebo po změně z režimu *Sequence* na *Normal* nebo naopak.



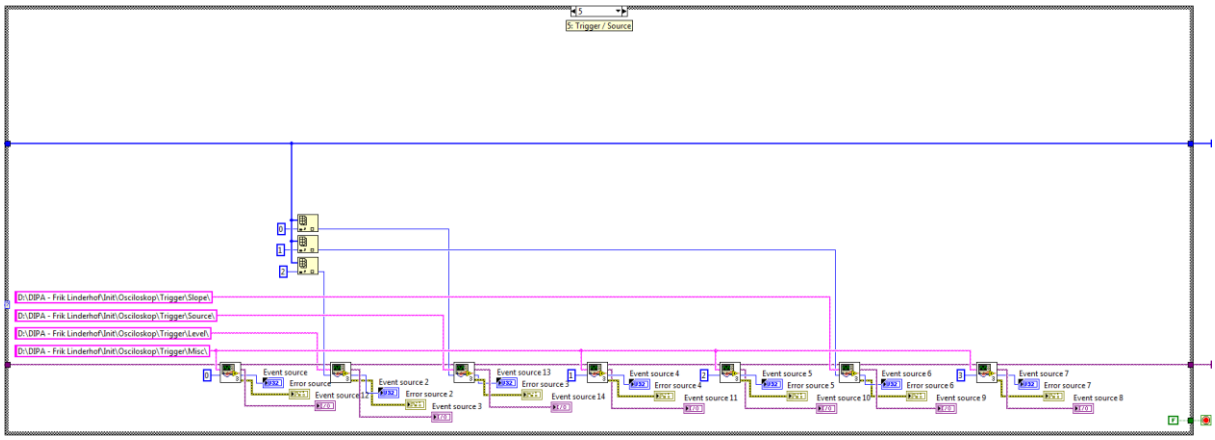
Obrázek 32: Blokový diagram Case 2 druhého kroku druhé části prvního procesu nového subprogramu Oscilloscope.

Case 3 (obr. 33) je spojen s tlačítkem < a posílá A&DDU hexadecimální kód k přizpůsobení pro kratší měřicí dobu. Stejným způsobem prodlouží *Case 4* měřicí dobu.



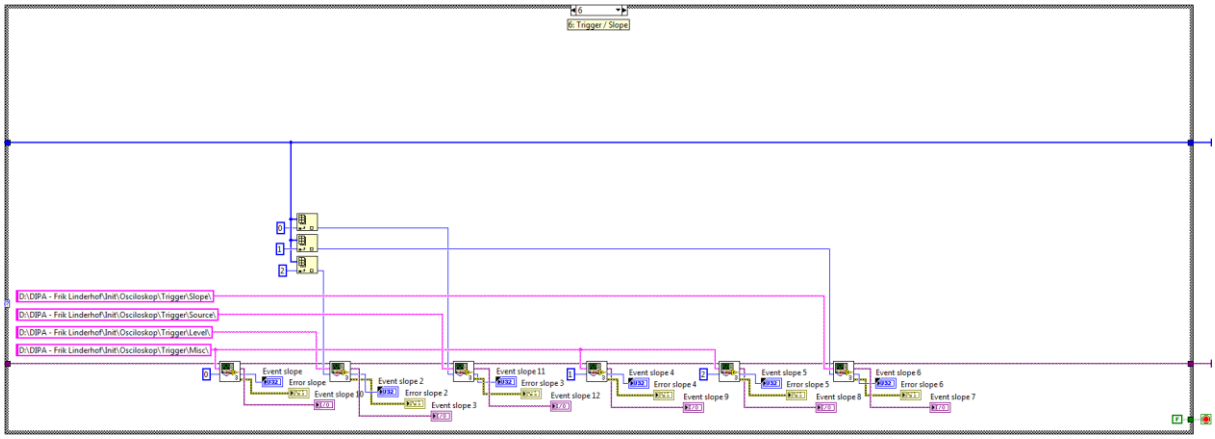
Obrázek 33: Blokový diagram Case 3 druhého kroku druhé části prvního procesu nového subprogramu Oscilloscope.

Přizpůsobení *source triggeru* je prováděno v *Case 5* (obr. 34). Na stránce 23 už bylo zmíněno, že sekvence kódu posílaná pro přizpůsobení triggeru je složitá. Data ze čtyř složek (tj. konkrétní sekvence kódu) jsou vybrána pomocí tří parametrů uložených v polích *integerů: trigger – sloup*, *trigger – source* a *trigger – level* a posílána v sedmi krocích.



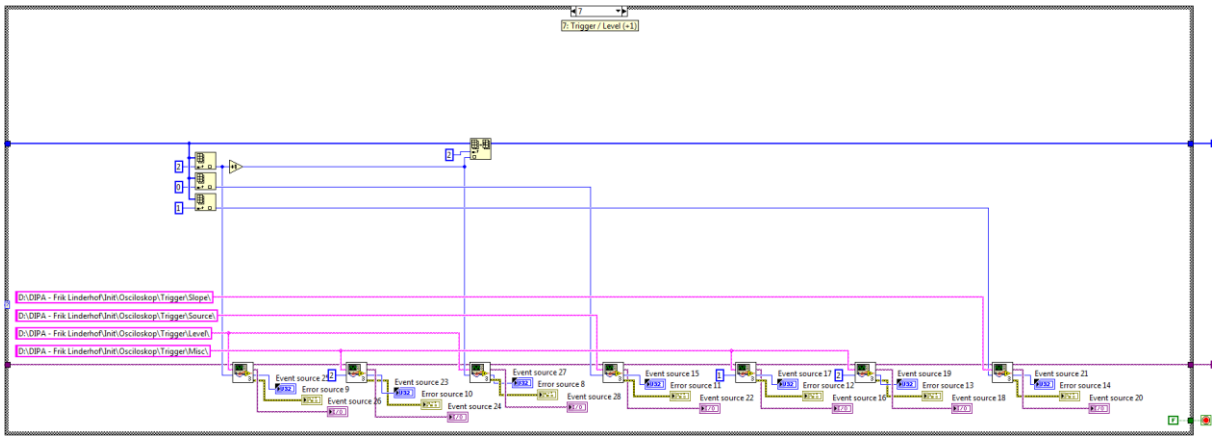
Obrázek 34: Blokový diagram Case 5 druhého kroku druhé části prvního procesu nového subprogramu Oscilloscope.

Case 6 (obr. 35) je hodně podobný *Case 5* v tom, že používá stejné parametry. Posílání dat je nakonec o trochu snadnější, protože místo sedmi kroků stačí jen šest kroků, poslední krok je vynechán. Pomocí tohoto *case* se sekvence přizpůsobí zvolené hraně *triggeru*.



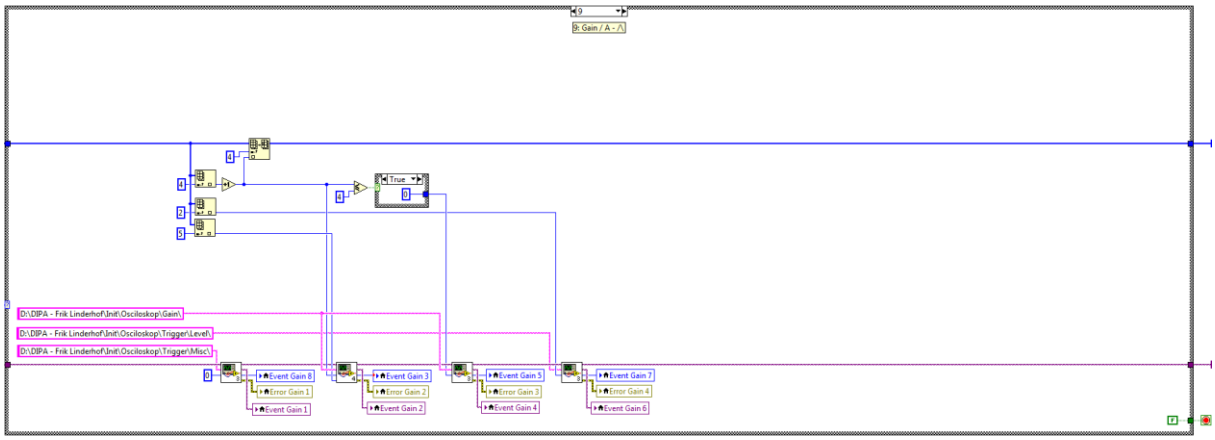
Obrázek 35: Blokový diagram Case 6 druhého kroku druhé části prvního procesu nového subprogramu Oscilloscope

Case 7 (obr. 36) znovu závisí na stejných parametrech a znovu se skládá ze sedmi kroků. Tímto se nastaví úroveň triggeru o jeden stupeň vyšší. Pomocí *Increment Function* je zvýšen parametr 4A. *Case 8* je úplně stejný jako *Case 7*, s výjimkou, že *Increment Function* je nahrazena *Decrement Function* a že úroveň *triggeru* se nastaví o jeden stupeň nižší.



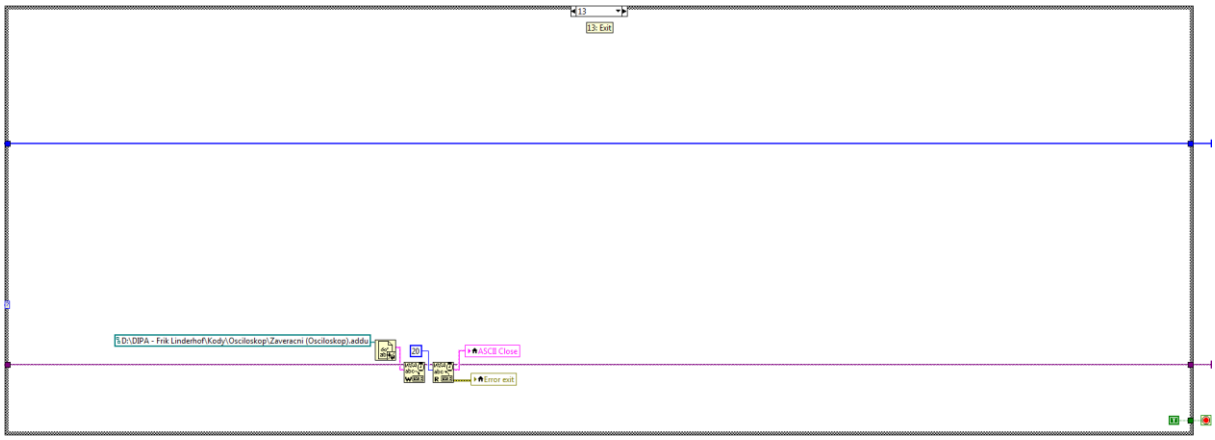
Obrázek 36: Blokový diagram Case 7 druhého kroku druhé části prvního procesu nového subprogramu Oscilloscope.

Zvýšení měřicího rozsahu *Gain – A* je prováděno pomocí *Case 9* (obr. 37). Data ze tří složek jsou vybrána pomocí tří parametrů uložených v polích typu *integer: Gain – A*, *Gain – B* a *trigger – level* a posílána ve čtyřech krocích. *Case 10* je stejný jako *Case 9*, s výjimkou, že *Increment Function* je nahrazena *Decrement Function* a že *Gain – A* se nastaví o jeden stupeň nižší. *Case 11* je stejný jako *Case 9*, s výjimkou, že *Increment Function* je aplikován na páté místo pole typu *integer* místo na čtvrté místo a tím se mění *Gain – B* místo *Gain – A*. *Case 12* je stejný jako *Case 11*, s výjimkou, že *Increment Function* je nahrazena *Decrement Function* a že *Gain – B* se nastaví o jeden stupeň nižší.



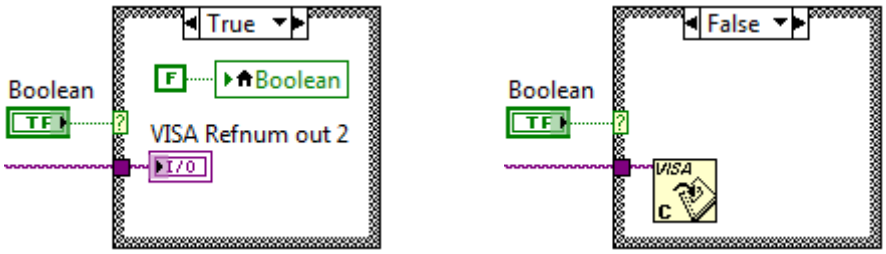
Obrázek 37: Blokový diagram Case 9 druhého kroku druhé části prvního procesu nového subprogramu Oscilloscope.

Poslední *case* je *Case 13* (obr. 38), který je aktivován v případě, že je stlačeno tlačítko *Exit*. Posílá se v něm zavírací kód a ukončí se druhá část prvního procesu.



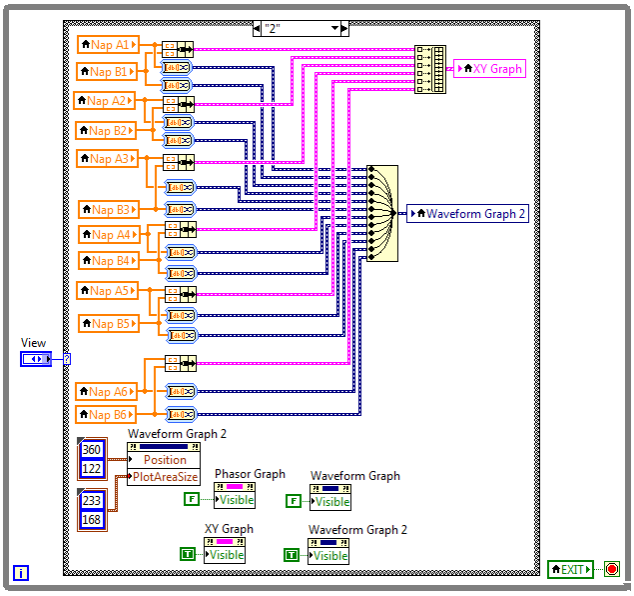
Obrázek 38: Blokový diagram Case 13 druhého kroku druhé části prvního procesu nového subprogramu Oscilloscope.

Třetí část prvního procesu (obr. 39) je přidána proto, aby bylo možné spustit bez problémů subprogram *Oscilloscope* jak samostatně, tak jako subprogram *RC 2000 1.3.vi*. V prvním případě je potřeba ukončit spojení přes COM port už v subprogramu, v druhém případě bude spojení přes COM port uzavřeno hlavním programem.



Obrázek 39: Blok diagram třetí části prvního procesu nového subprogramu Oscilloscope.

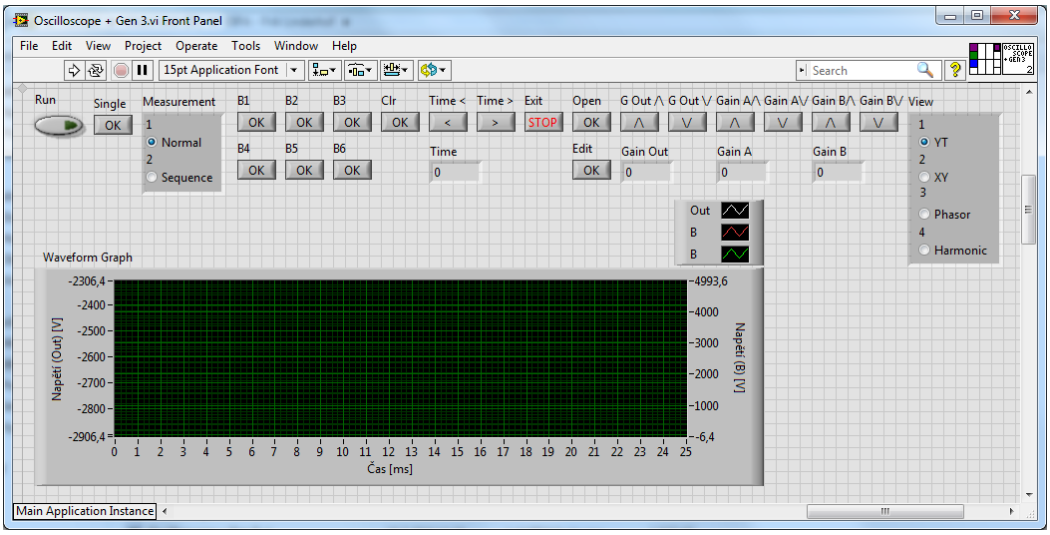
Druhý proces je zobrazování naměřených dat (obr. 40). Vyřeší se zde dva různé problémy: který graf se obrazí na *Front Panelu* a jak se v grafu zobrazí naměřené hodnoty. Zobrazení grafů je vyřešeno pomocí *Property Node*, kde zobrazení naměřených hodnot je vyřešeno pomocí *Merge Signals Function* a *Bundle Function*.



Obrázek 40: Blokový diagram druhého procesu nového subprogramu Oscilloscope.

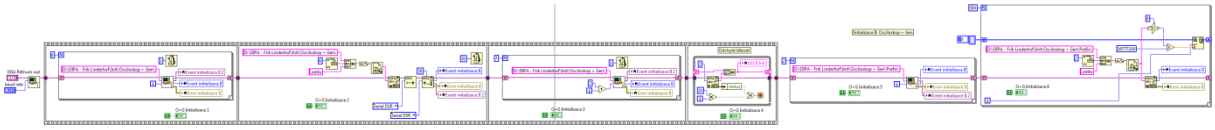
3.3.3. Režim *Oscilloscope + Gen*

Nový subprogram *Oscilloscope + Gen* (obr. 41) se v některých detailech od originálního subprogramu liší. Stejně jako nový subprogram *Oscilloscope* jsou do podskupiny *Sequence* přidána dvě tlačítka, *B5* a *B6*. Dále je v porovnání s originálním subprogramem přidána skupina *View*. Poslední důležitá změna je rozšíření skupiny *Gain* podskupinou *A*, která změní originální jednokanálový osciloskop a generátor na dvoukanálový osciloskop a generátor.



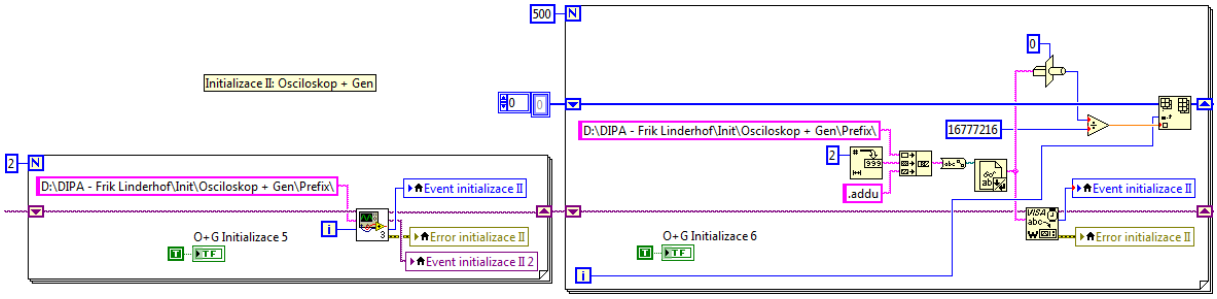
Obrázek 41: Snímek obrazovky Front Panelu nového subprogramu Oscilloscope + Gen.

Stejně jako u subprogramu *Oscilloscope* je kód subprogramu *Oscilloscope + Gen* rozdělen do dvou procesů *vlastní program* a *zobrazování*. Vlastní program se rozdělí do tří částí *inicializace*, *vlastní program* a *ukončení*. První část, *inicializace* (obr. 42), lze rozdělit do dvou kroků, *vlastní inicializace* a *inicializace výstupního signálu*. První krok, *vlastní inicializace*, je funkčně stejný jako první části prvního procesu, *inicializace*, v subprogramu *Oscilloscope* a rozdíly jsou jen v odlišném časování. Tento krok v kódu proto nebude vysvětlován zvlášť.



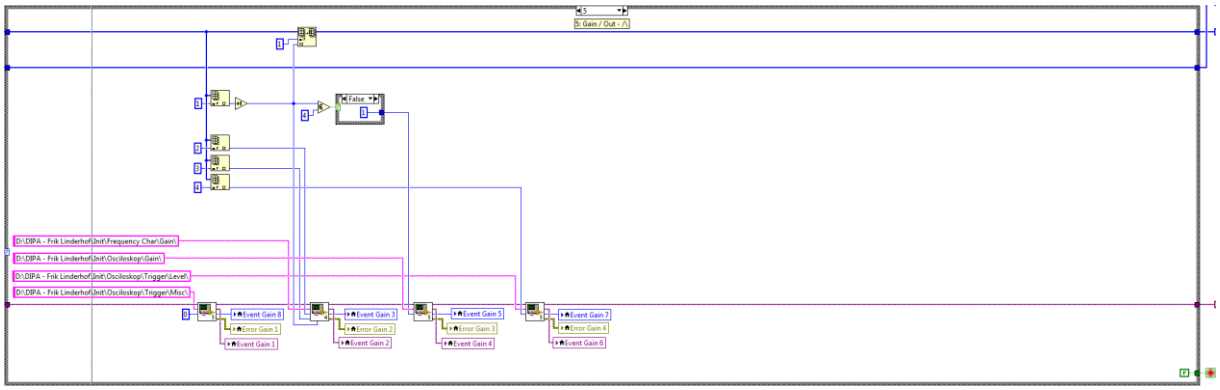
Obrázek 42: Blokový diagram první části prvního procesu nového subprogramu Oscilloscope + Gen.

Druhý krok, *inicializace výstupního signálu* (obr. 43), se rozdělí do dvou dílů *posílání prefixu* a *posílání hexadecimálních hodnot*. Posílání prefixu je potřeba, ať A&DDU ví, že další posílané informace nejsou parametry, ale hodnoty reprezentující nějaké napětí konkrétního průběhu (waveform). Dále se ve druhém dílu konvertují tyto hexadecimální hodnoty do decimálních hodnot a uloží se do pole typu *integer*.



Obrázek 43: Blokový diagram druhého kroku první části prvního procesu nového subprogramu Oscilloscope + Gen.

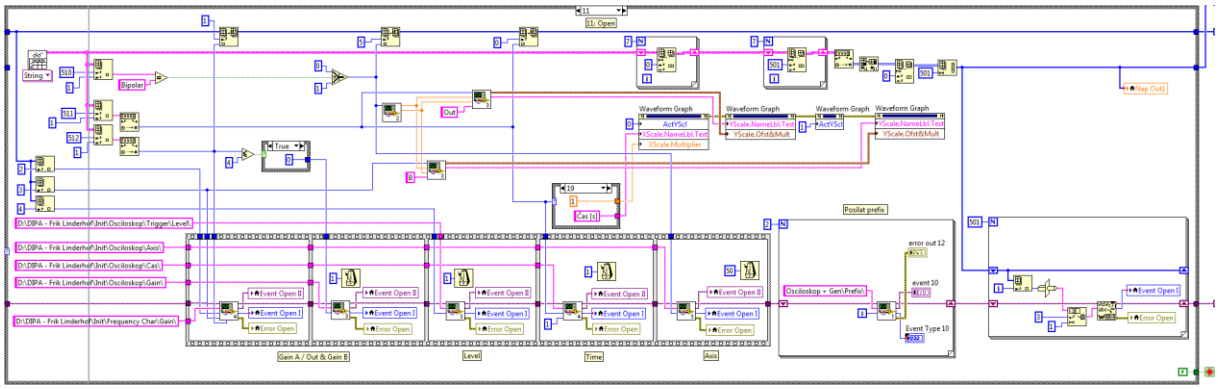
První krok druhé části prvního procesu nového subprogramu *Oscilloscope + Gen* je funkčně hodně podobný stejnému kroku subprogramu *Oscilloscope* (obr. 20). Totéž platí pro *Case 0: Default*, *Case 1: Single*, *Case 2; Measurement – clear*, *Case 3: Time – <*, *Case 4: Time – >* v druhém kroku subprogramu *Oscilloscope + Gen*. *Case 5: Gain Out – /* (obr. 44) je v porovnání se subprogramem *Oscilloscope* přidán. Tento *case* vybere komunikační kódy ze čtyř složek pomocí čtyř parametrů a posílá kódy ve čtyřech krocích. *Case 6*: *Gain Out – \/* je stejný jako *Case 5*, s tím rozdílem, že *Increment Function* je nahrazena *Decrement Function*. *Case 7: Gain A – /* a *Case 9: Gain B – /* jsou stejné jako *Case 5*, ale v obou případech je *Increment Function* aplikována na jiném *integeru*. *Case 8: Gain A – \/* a *Case 10: Gain B – \/ se* liší tím, že *Increment Function* je nahrazena *Decrement Function* a že tato funkce je aplikována na jiném *integeru*.



Obrázek 44: Blokový diagram Case 5 druhého kroku druhé části prvního procesu nového subprogramu Oscilloscope + Gen.

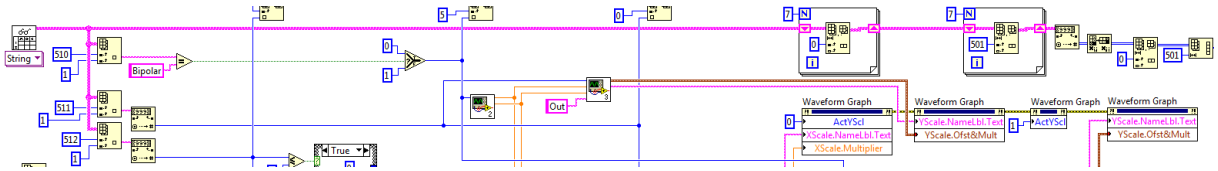
Case 11: Open (obr. 45) je úplně nový *case*. Tento *case* je hodně komplexní. Data použitá v něm pochází z tří zdrojů: ze složky se soubory **.aio* (editorem vytvořené

průběhy výstupního signálu), ze složek se soubory *.*addu* (částí sekvencí kódů) a z pole *integerů*.



Obrázek 45: Blokový diagram Case 11 druhého kroku druhé části prvního procesu nového subprogramu Oscilloscope + Gen.

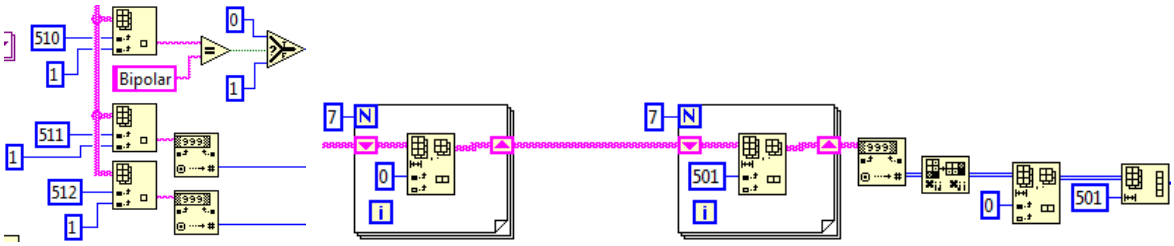
Nejprve je potřeba se podívat na první zdroj dat ze souboru *.*aio* (obr. 46). V něm lze najít decimální hodnoty, které budou posílány do A&DDU a tři parametry: *Axis*, *Time* a *Gain Out*. Pomocí funkce *Read From Spreadsheet File (string).vi* je celý soubor *.*aio* naskenován do pole typu *string*.



Obrázek 46: Blokový diagram prvního dílu Case 11 druhého kroku druhé části prvního procesu nového subprogramu Oscilloscope + Gen.

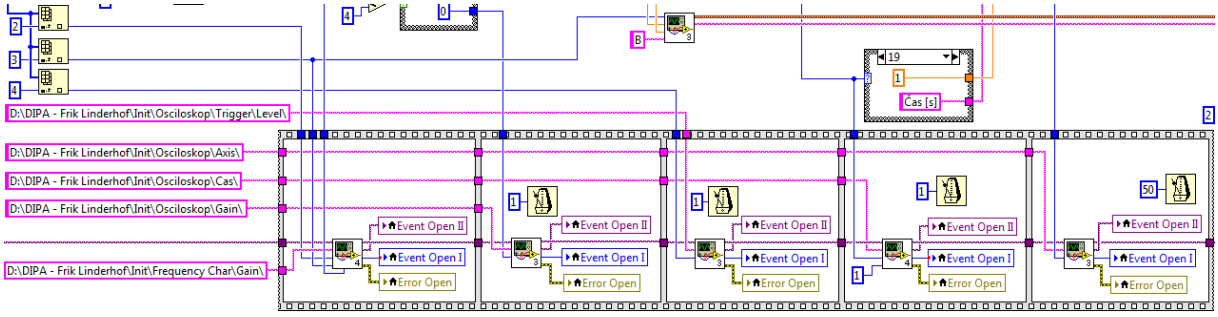
Z pole typu *string* lze jednoduše vybrat žádané hodnoty parametrů (obr. 47 vlevo) pomocí funkce *Index Array Function* a pomocí funkce *Decimal String To Number Function* je konvertovat na typ *integer*.

Decimální hodnoty jsou prosévány pomocí dvou smyček, které odstraní všechna další data mimo hodnot průběhu signálu (obr. 47 vpravo). Potom se konvertuje *string* do pole typu *integer* a to je transponováno pomocí funkce *Transpose 2D Array Function*. Pomocí funkce *Delete From Array Function* jsou odstraněny indexy a nakonec je pole přetvořeno pomocí funkce *Reshape Array Function*.



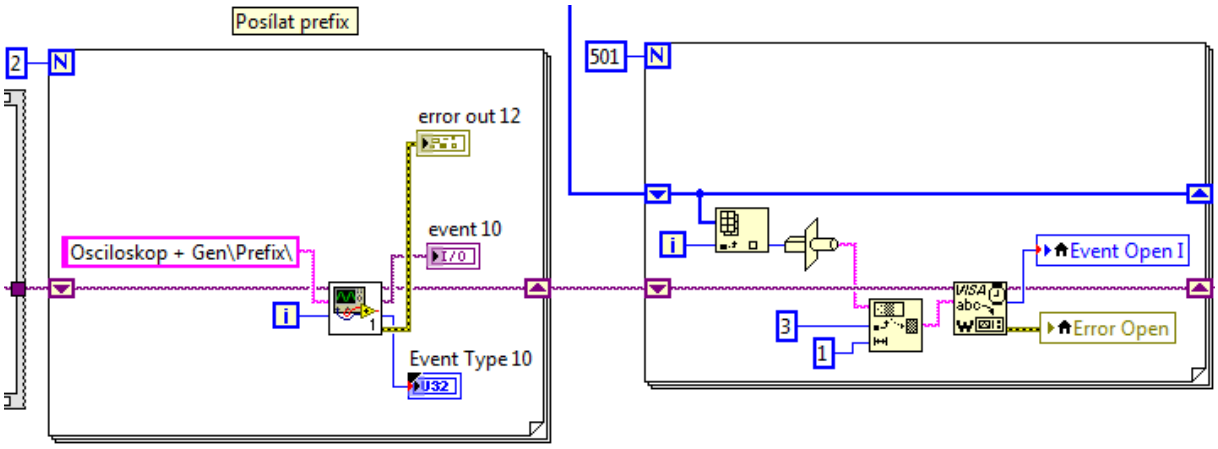
Obrázek 47: Vlevo) Blokový diagram prvního pustupu prvního dílu Case 11 druhého kroku druhé části prvního procesu nového subprogramu Oscilloscope + Gen, vpravo) Blokový diagram druhého postupu prvního dílu Case 11 druhého kroku druhé části prvního procesu nového subprogramu Oscilloscope + Gen.

Pomocí tří parametrů *Axis*, *Time* a *Gain Out*, získaných ze souboru **.aio*, spolu s parametry *Gain A*, *Gain B* a *Trigger Level*, získaných z pole typu *integer*, jsou vybrány z pěti složek hexadecimální kódy pro změnu nastavení (obr. 48).



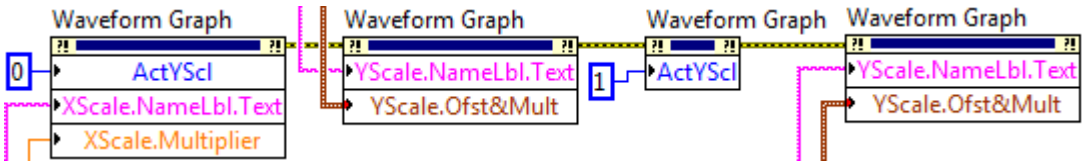
Obrázek 48: Blokový diagram druhého dílu Case 11 druhého kroku druhé části prvního procesu nového subprogramu Oscilloscope + Gen.

Poslední díl posílá nejprve *prefix* do A&DDU ve smyčce (obr. 49). Potom jsou data průběhu signálu postupně konvertována z decimální hodnoty na hexadecimální hodnotu a posílána do A&DDU.



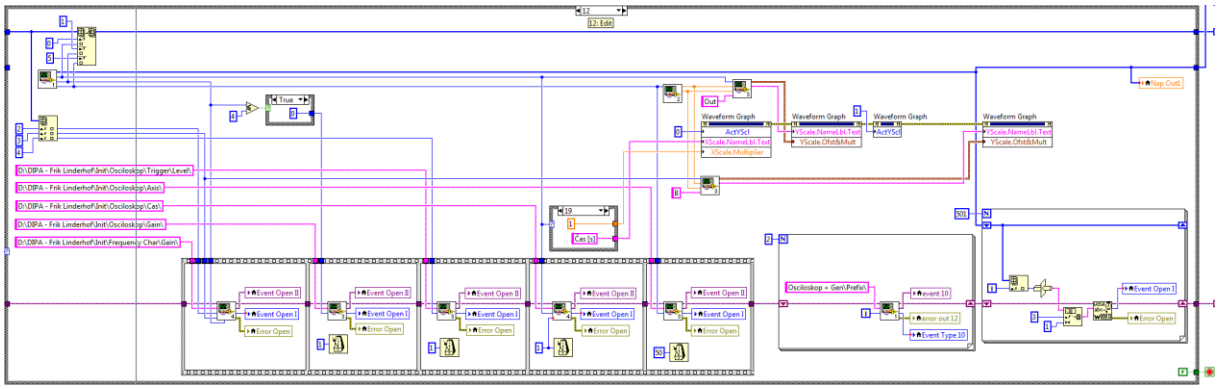
Obrázek 49: Blokový diagram třetího dílu Case 11 druhého kroku druhé části prvního procesu nového subprogramu Oscilloscope + Gen.

Pomocí čtyři funkcí *Property Node* jsou současně přizpůsobeny osy grafu (obr. 50).



Obrázek 50: Blokový diagram čtvrtého dílu Case 11 druhého kroku druhé části prvního procesu nového subprogramu Oscilloscope + Gen.

Case 12: Edit (obr. 51) je velmi podobný k *Case 11: Open*. Všechna data a parametry, které jsou pomocí funkce *Read From Spreadsheet File (string).vi* nalezeny v *Case 11: Open*, jsou v *Case 12: Edit* poskytnuty subprogramu *Analog Waveform Editor.vi*. *Case 13: Exit* v subprogramu *Oscilloscope + Gen* je stejný jako v subprogramu *Oscilloscope* (obr. 38).

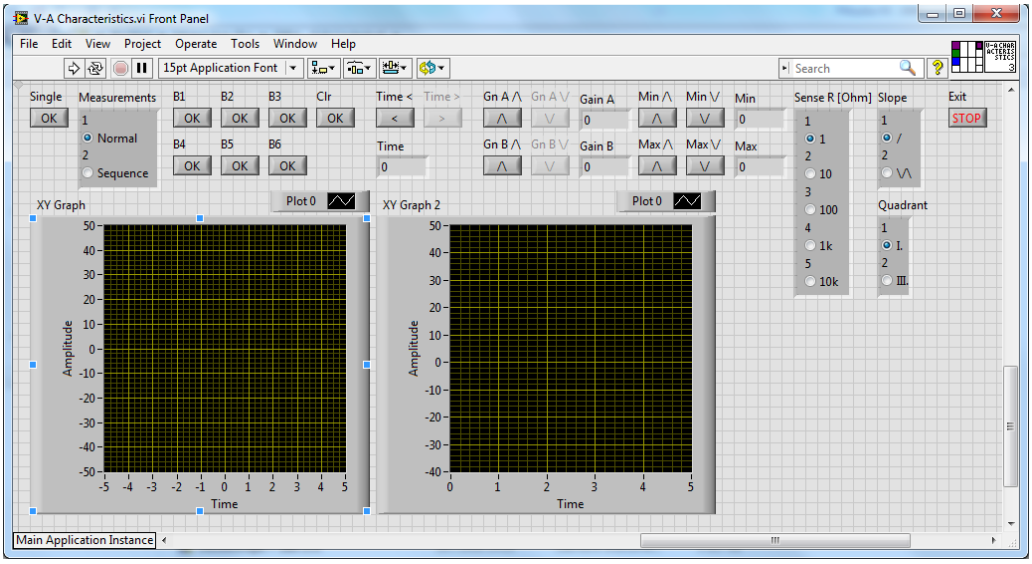


Obrázek 51: Blokový diagram Case 12 druhého kroku druhé části prvního procesu nového subprogramu Oscilloscope + Gen.

Třetí krok druhé části prvního procesu subprogramu *Oscilloscope + Gen* je stejný jako tentýž krok v subprogramu *Oscilloscope* (obr. 39). Druhý proces, *zobrazování*, je stejný jako druhý proces subprogramu *Oscilloscope* (obr. 40).

3.3.4. Režim *V-A Characteristics*

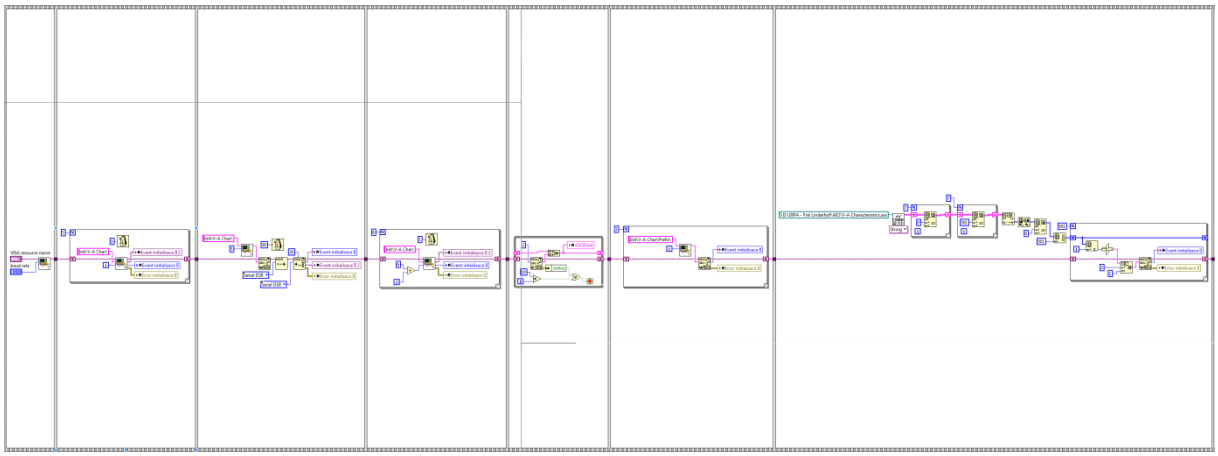
Stejně jako nové subprogramy *Oscilloscope* a *Oscilloscope + Gen* je nový subprogram rozšířen v podskupině *Sequence* tlačítky *B5* a *B6* (obr. 52).



Obrázek 52: Snímek obrazovky Front Panelu nového subprogramu V-A Characteristics.

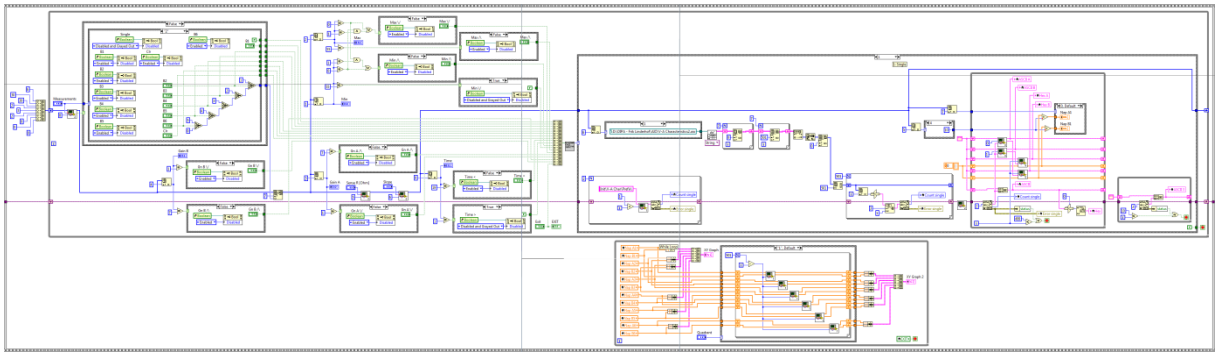
Blokový diagram subprogramu je rozdělen do *Sequence Structure* s devíti částmi, ze kterých prvních sedm provádí inicializaci. Další dvě části jsou *vlastní program* a *ukončení*. Inicializace (obr. 53) se skládá z dvou kroků: *vlastní inicializace* a *inicializace*

výstupního signálu. Vlastní inicializace je prvních pět částí a inicializace výstupního signálu jsou poslední dvě části.



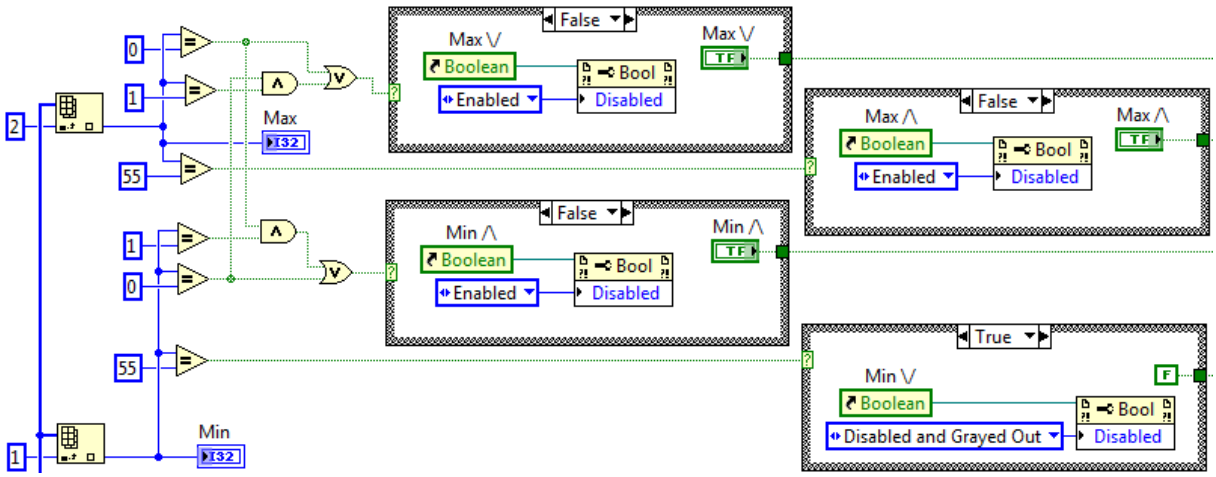
Obrázek 53: Blokový diagram první sedmi částí nového subprogramu V-A Characteristics.

Osmá část, *vlastní program* (obr. 54), je rozdělena do dvou procesů: *vlastní program* a z*obrazení*. První proces, *vlastní program*, se skládá ze dvou kroků: *stanovení* a *provádění*. První krok, stanovení, funguje stejně jako u subprogramů *Oscilloscope* a *Oscilloscope + Gen*, kde je jen potřeba upozornit na podskupiny *Min Level* a *Max Level* u skupiny *Output Ramp*.



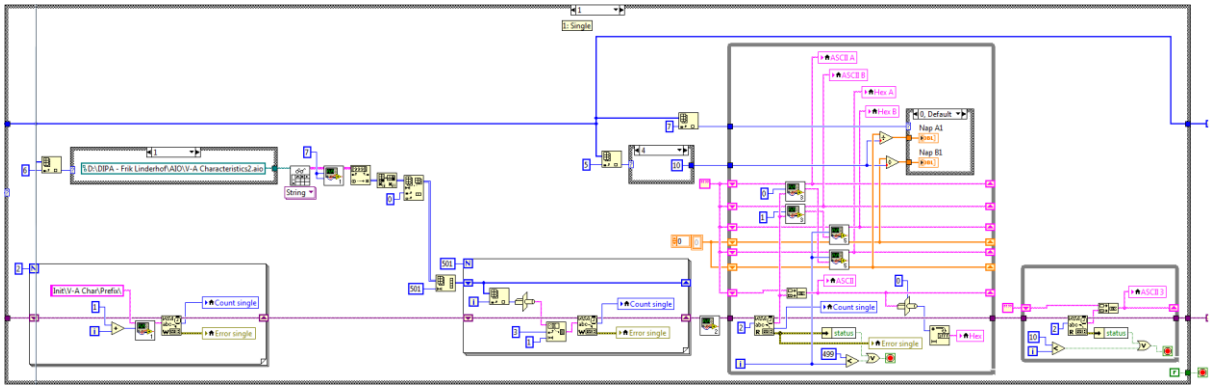
Obrázek 54: Blokový diagram osmého části nového subprogramu V-A Characteristics.

Podskupiny *Min Level* a *Max Level* jsou vzájemné spojené (obr. 55). Minimální hodnota obou podskupin je 0, ale není možné, aby obě podskupiny současně měly tuto hodnotu. Proto je vytvořena konstrukce před podmínkami, která toto ošetří.



Obrázek 55: Blokový diagram podskupin Min Level a Max Leven skupiny Output Ramp prvního kroku prvního procesu osmé části nového subprogramu V-A Characteristics.

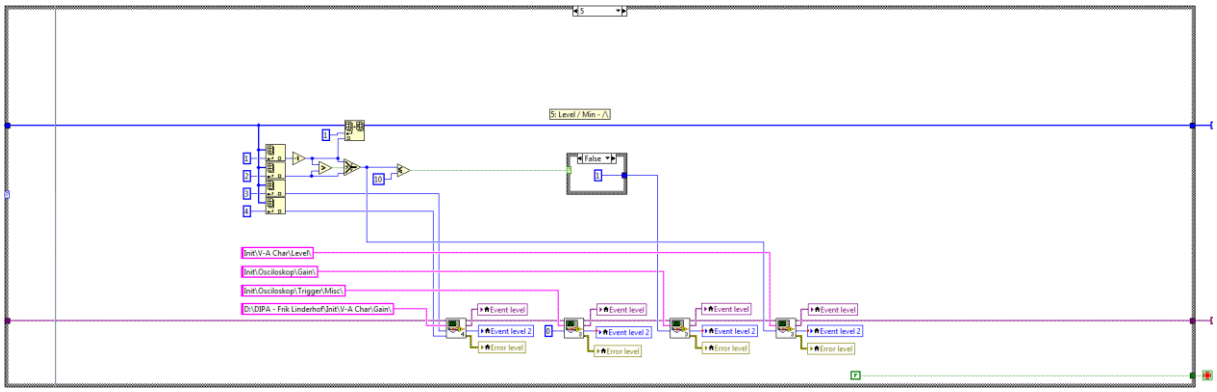
Druhý krok se znovu skládá z *Case Structure*. *Case 0: Default* je stejný jako u subprogramu *Oscilloscope*. *Case 1: Single* (obr. 56), ale liší se od předchozích subprogramů. Před posláním příkazu pro měření se posílá sekvence složená ze dvou dílů. První díl je poslání prefixu, druhý díl je poslání analogového výstupního signálu. Další díly, *posílat příkaz měření*, *vlastní měření* a *ukončení* jsou stejné jako u subprogramů *Oscilloscope* a *Oscilloscope + Gen*.



Obrázek 56: Blokový diagram Case 1 druhého kroku prvního procesu osmé části nového subprogramu V-A Characteristics.

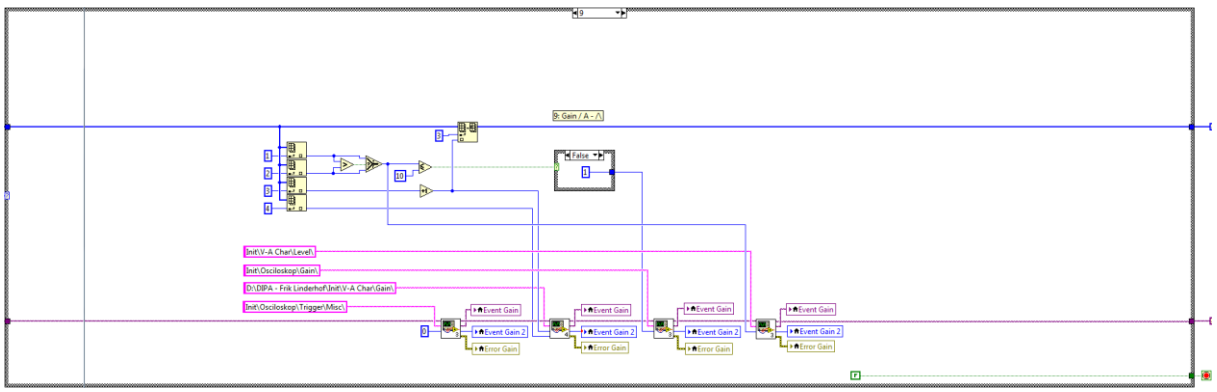
Case 2: Measurement / Clear, *Case 3: Time – <* a *Case 4: Time – >* jsou stejné jako u subprogramů *Oscilloscope* a *Oscilloscope + Gen*. *Case: 5 Level / Min – /* (obr. 57) je v subprogramu *V-A Characteristics* nový *case*. Ze čtyř složek se posílají po sobě čtyři komunikační kódy pomocí čtyř parametrů uložených v polích typu *integer*: *Level / Min*, *Level / Max*, *Gain / A* a *Gain / B*. *Case 6*: *Level / Min – \/* je stejný jako *Case 5*, s rozdílem,

že *Decrement Function* je nahrazena *Increment Function*. *Case 7: Level / Max – /* je stejné jako *Case 5*, ale v obou případech je *Increment Function* aplikován na jiném *integeru*. *Case 8: Level / Max – \/ se* liší s tím, že *Increment Function* je nahrazena *Decrement Function* a že tato funkce je aplikována na jiném *integeru*.



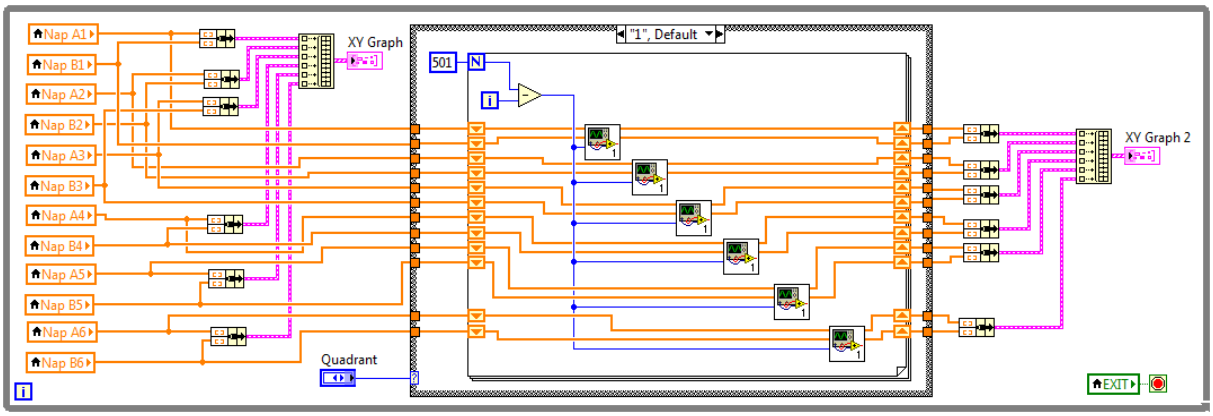
Obrázek 57: Blokový diagram Case 5 druhého kroku prvního procesu osmé části nového subprogramu V-A Characteristics.

Case 9: Gain A – / (obr. 58) se liší od stejného *case* předchozích subprogramů v tom, že data přichází ze čtyř složek místo tří a pomocí čtyř parametrů uložených v polích integerů místo tří. V porovnání s *Case 5* jsou složky a parametry stejné, ale pořadí, ve kterém se posílají komunikační kódy a pozice *Increment Function*, jsou jiné. *Case 10: Gain A – \/* je stejný jako *Case 9* s rozdílem, že *Increment Function* je nahrazena *Decrement Function*. U *Case 11: Gain B – /* je posunuta Increment Function a u *Case 12: Gain B – \/* je tato *Increment Function* nahrazena *Decrement Function*. *Case 13: Exit* je stejný jako u subprogramů *Oscilloscope* a *Oscilloscope + Gen*.



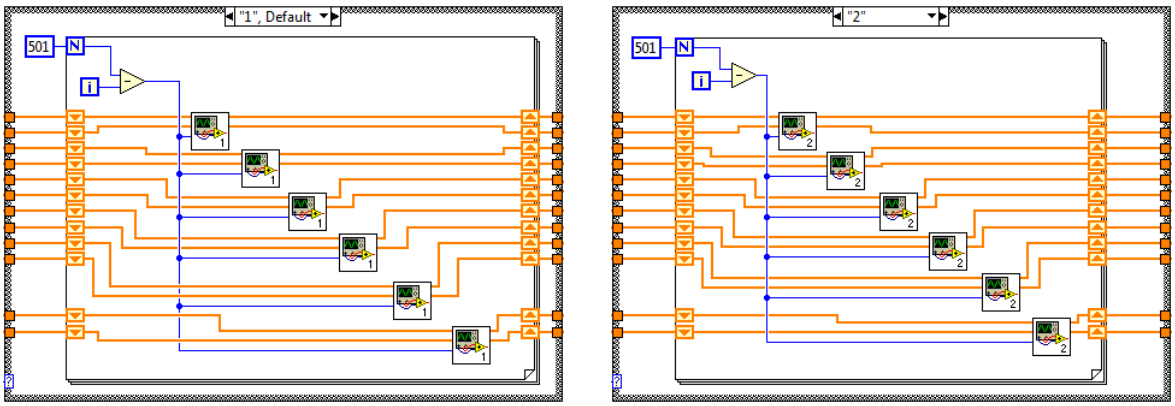
Obrázek 58: Blokový diagram Case 9 druhého kroku prvního procesu osmé části nového subprogramu V-A Characteristics.

Druhý proces osmé části je zodpovědný za zobrazování (obr. 59). Zobrazí se dva grafy *XY Graph* a *XY Graph 2*. V *XY Graph* se kreslí celý graf. Všechna naměřená data se spojují do šesti polí. Těchto šest polí se spojuje do jednoho, které lze vykreslit v grafu. Vykreslení grafu v *XY Graph 2* je komplikovanější, tady je potřeba nejdříve všechna data zpracovat.



Obrázek 59: Blokový diagram druhého procesu osmé části nového subprogramu V-A Characteristics.

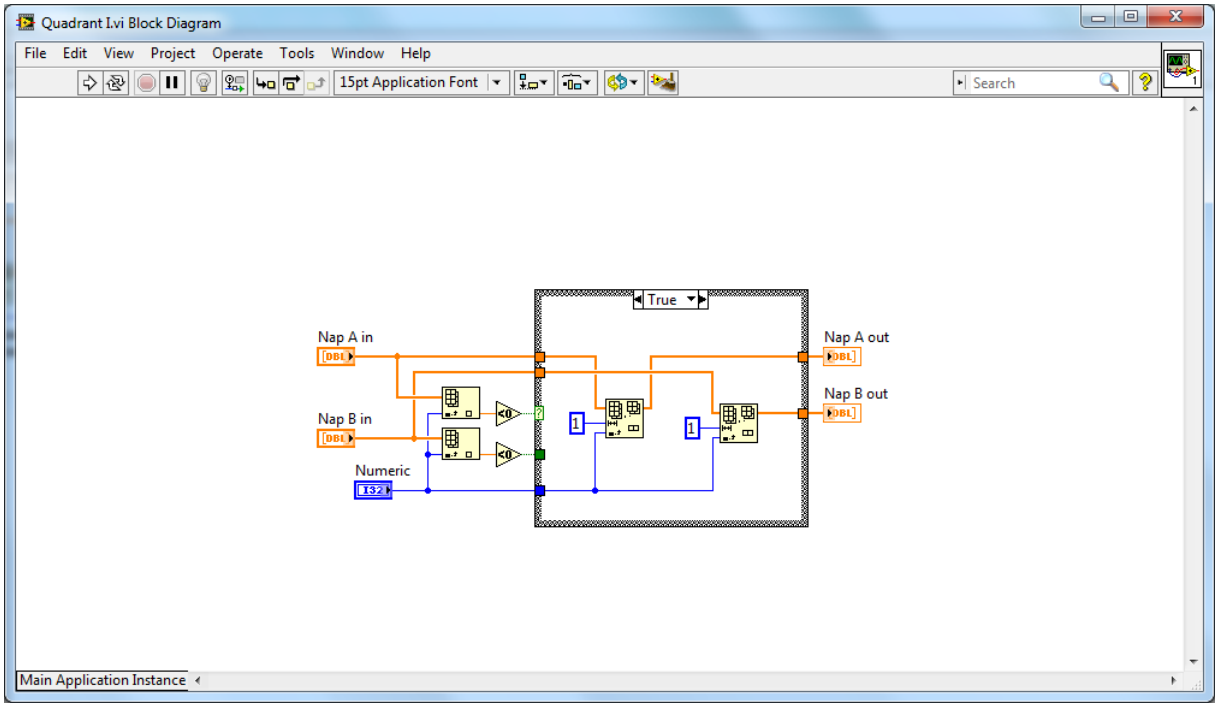
Zpracování dat se provádí na základě nastavení *Radio Buttons Quandrant*. Nastavení tlačítka ovládá podmínku (obr. 60). První *case* se provádí u prvního kvadrantu a druhý *case* u třetího kvadrantu. Signály z kanálů *A* a *B* jsou párovány a krok za krokem jsou ve smyčce kontrolovány pomocí subVI *Quadrant I.vi* a *Quandrant III.vi*.



Obrázek 60: Blokový diagram prvního a druhého case druhého procesu osmé části nového subprogramu V-A Characteristics.

V těchto subVI (obr. 61) jsou data vyřazena v případě, že hodnoty z kanálu *A* jsou menší (*Quadrant I*) nebo větší (*Quadrant III*) než nula. V případě, že hodnoty jsou vyřazeny, se kontrolují další data. V případě, že hodnoty nejsou vyřazeny, zkontroluje se

hodnota z kanálu *B*. V případě, že hodnota z kanálu *B* je menší, respektive větší než nula, obě hodnoty se vyřadí. V případě, že hodnota z kanálu *B* je větší, respektive menší než nula, hodnoty se ponechají.



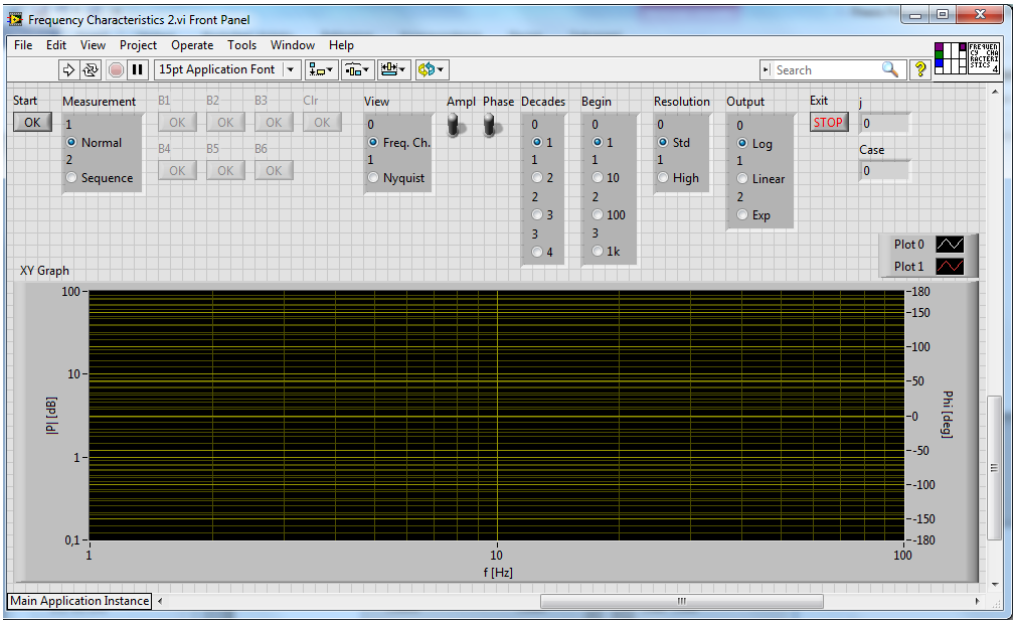
Obrázek 61: Blokový diagram subprogramu Quadrant I.vi.

Všechny hodnoty jsou potom párovány pomocí funkce *Bundle function*, pomocí *Build Array Function* jsou seskupeny a zobrazeny. Devátá část subprogramu *V-A Characteristics* funguje stejně jako třetí část subprogramu *Oscilloscope*.

3.3.5. Režim *Frequency Characteristics*

Nový subprogram *Frequency Characteristics* (obr. 62) je v porovnání se starým subprogramem rozšířen. Obě skupiny *Decades* a *Begin* jsou rozšířeny o jednu další možnost: měřit čtyři dekády a měřit od 1 Hz. Dále je přidána nová skupina *Output*, s kterou lze nastavit rozložením frekvencí, při kterých se provádí měření: *logaritmické*, *lineární* a *exponenciální*. Nastavení ovládané pomocí skupiny *Resolution* se zjednodušilo. Místo variabilního počtu měření na základě *Resolution*, *Begin* a *Decades*, závisí nové nastavení jenom na *Resolution* a *Decades*, viz rovnice (1), kde *Resolution* je 1 v případě, že nastavení je *Standard* a kde *Resolution* se rovná 3 v případě, že nastavení je *High*

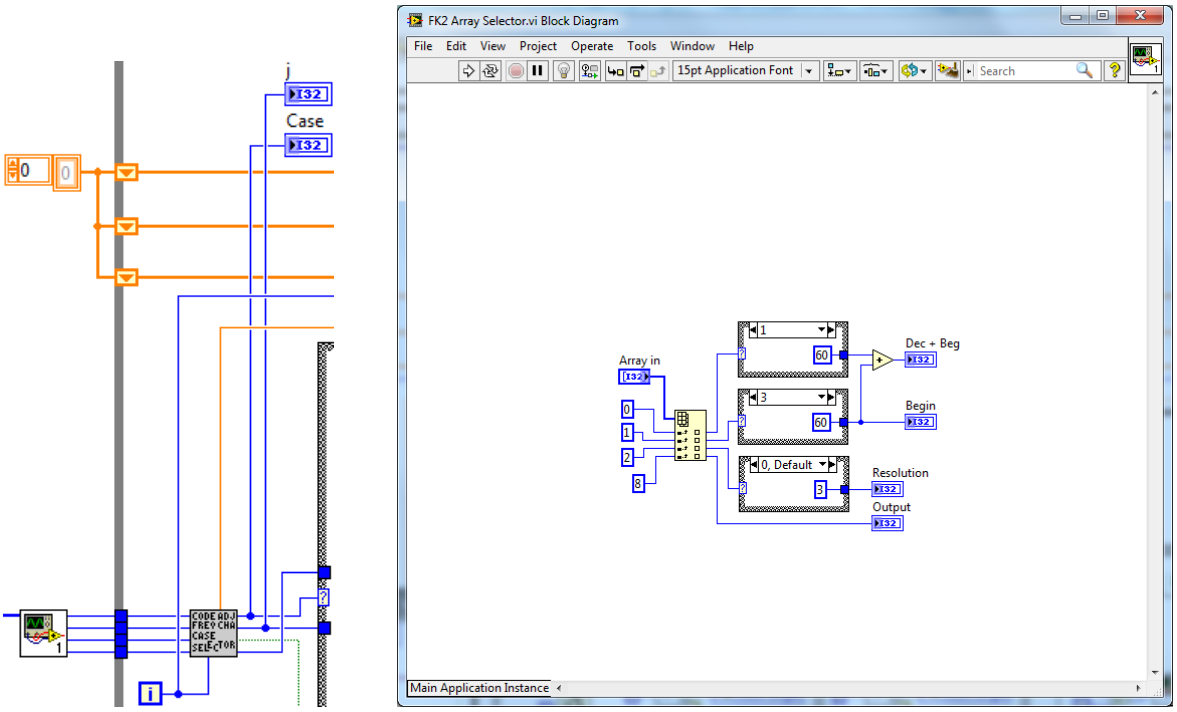
$$1 + 10 \cdot Decades \cdot Resolution. \quad (1)$$



Obrázek 62: Snímek obrazovky Front Panelu nového subprogramu Frequency Characteristics.

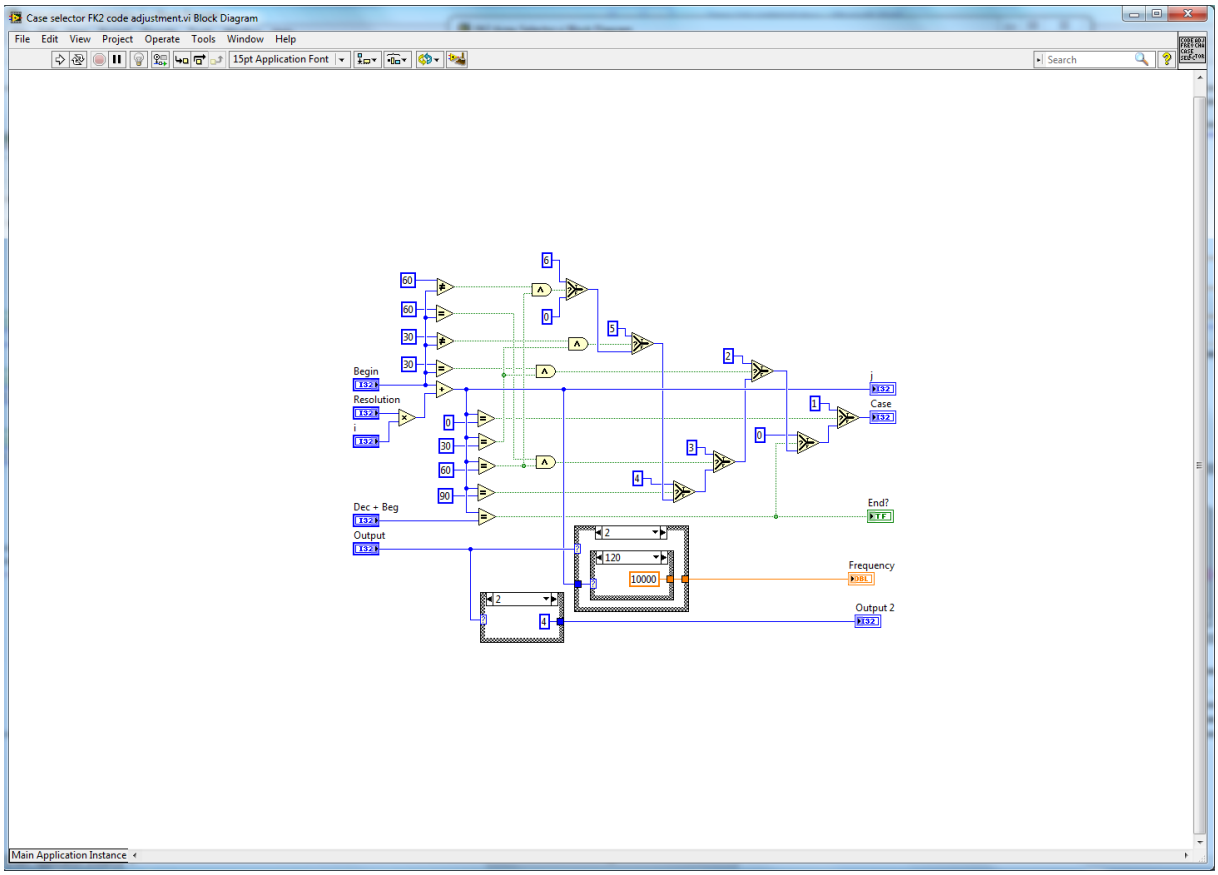
Subprogram *Frequency Characteristics* se skládá ze sedmi částí, prvních pět jsou *inicializace*, další je *vlastní program* a poslední je *ukončení*. Inicializace funguje stejně jako u předchozích subprogramů. Šestá část *vlastní program* je rozdělena na dva procesy: *vlastní program* a *zobrazení*. Proces *vlastní program* se skládá ze dvou kroků: *stanovení* a *provádění*. První krok *stanovení* funguje podobně jako v předchozích subprogramech. Druhý krok provádění se skládá z *Case structure*. *Case 0: Default* je stejný jako u předchozích subprogramů. *Case 1: Start* je rozdělen na dva díly. První díl je *vlastní měření* a druhý díl je *příprava na zobrazení*. Vlastní měření se štěpí do tří součástí: *stanovení*, *inicializace* a *provádění*.

Stanovení se skládá ze dvou SubVI *FK2 Array Selector.vi* (obr. 63) a *Case selector FK2 code adjustment.vi*. První SubVI se nachází před smyčkou *While*, která tvoří hlavní část kroku *provádění*. Cílem tohoto subVI je připravit nastavení celého měření.



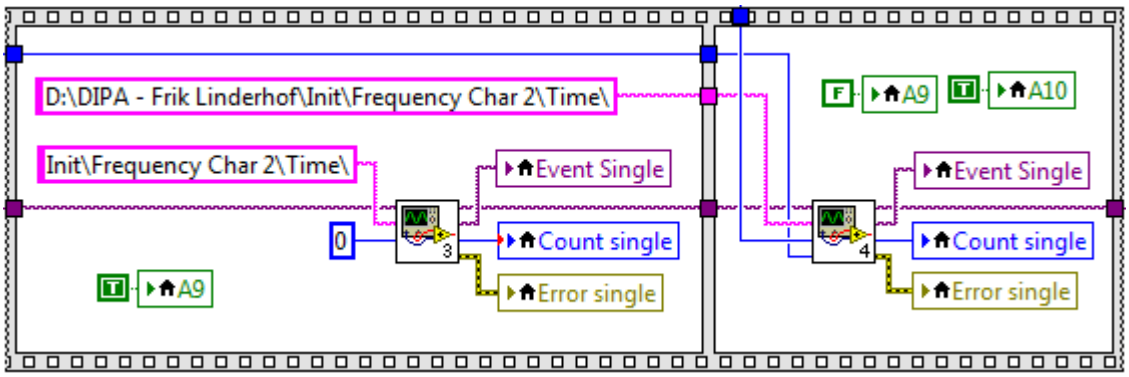
Obrázek 63: Vlevo) Blokový diagram první součásti prvního dílu Case 1 druhého kroku prvního procesu šesté části nového subprogramu Frequency Characteristics, vpravo) Blokový diagram FK2 Array Selector.vi.

Druhý subVI *Case selector FK2 code adjustment.vi* (obr. 64) má některé povinnosti. První povinnost je stanovení, kdy má smyčka *While*, ve které se nachází, skončit. Druhá povinnost je určení *case* další součásti *inicializace*. Další důležitá povinnost tohoto subVI je stanovení frekvence měření.



Obrázek 64: Blokový diagram Case selector FK2 code adjustment.vi.

Další součást *inicializace* subprogramu se skládá z *Case Structure*. Tento *Case Structure* obsahuje 7 *casů*. *Case 0: Default* (obr. 65) je nejjednodušší a nejčastěji používaný *case*. V *case* se nachází *Sequence Structure* s dvěma částmi. Pomocí parametrů, určených v SubVI *Case selector FK2 code adjustment.vi*, se v tomto *case* posílá časové nastavení.

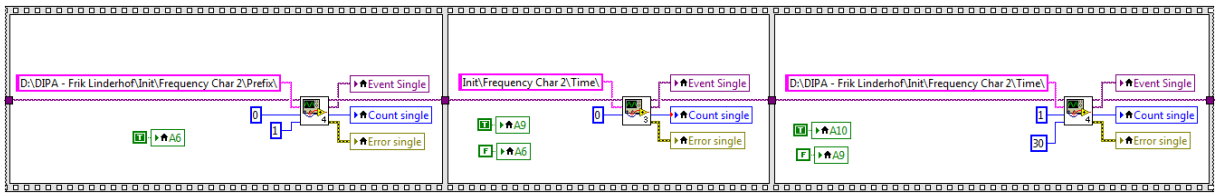


Obrázek 65: Blokový diagram Case 0 druhé součásti prvního dílu Case 1 druhého kroku prvního procesu šesté části nového subprogramu Frequency Characteristics.

Case 1: Begin = 1 Hz (obr. 66) obsahuje *Case Structure* s deseti částmi. Prvních pět částí lze považovat za *inicializaci* a další dvě fungují jako *prefix*, pomocí kterých se posílá nastavení. V sedmé části se posílá výstupní signál a v posledních dvou částech se posílá časové nastavení. *Case 2: Begin = 10 Hz*, *Case 3: Begin = 100 Hz* a *Case 4: Begin = 1 kHz* fungují v podstatě stejně.

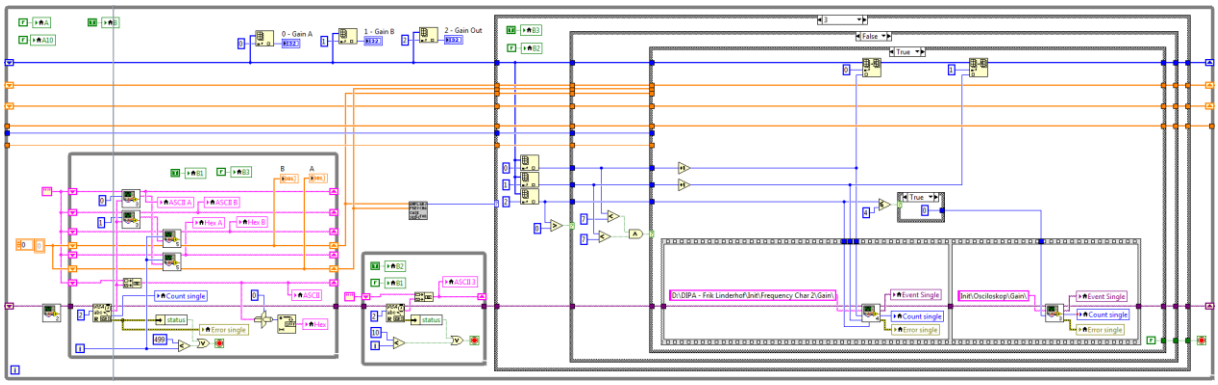
Obrázek 66: Blokový diagram Case 1 druhé součásti prvního dílu Case 1 druhého kroku prvního procesu šesté části nového subprogramu Frequency Characteristics.

V *Case 5: j = 10 Hz* (obr. 67) nachází *Case Structure* se třemi částmi. První část posílá prefix a další dvě posílají časové nastavení. *Case 6: j = 100 Hz* funguje stejně jako *Case 5*.



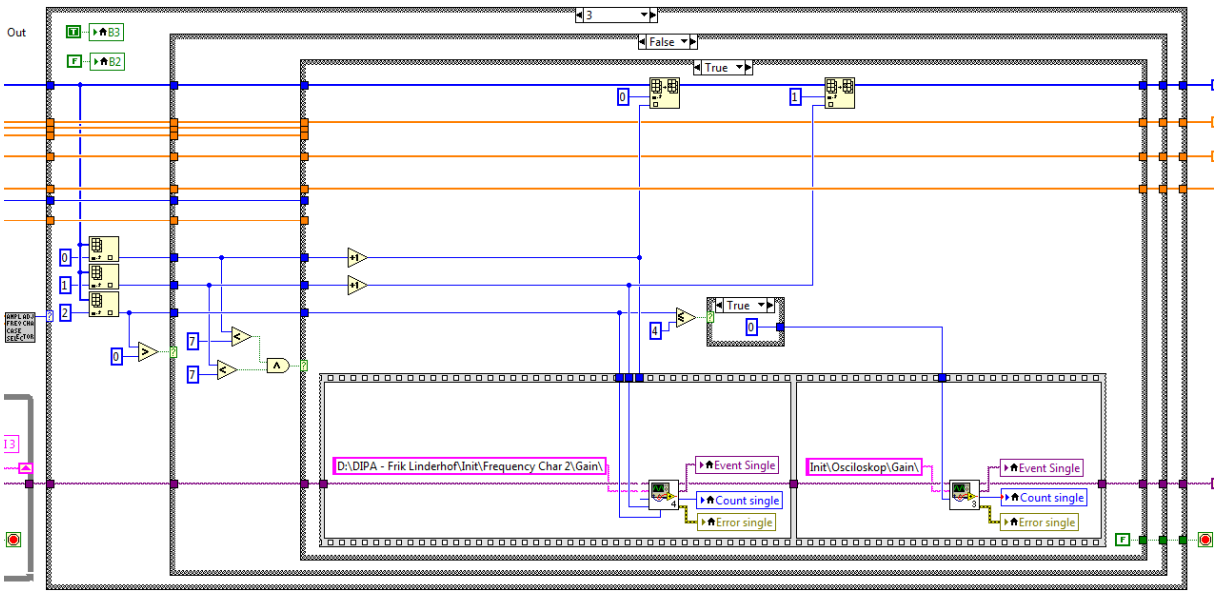
Obrázek 67: Blokový diagram Case 5 druhé součásti prvního dílu Case 1 druhého kroku prvního procesu šesté části nového subprogramu Frequency Characteristics.

Třetí součást *provádění* (obr. 68) lze rozdělit na čtyři bloky. První tři jsou už známé, a to *žádost o měření*, *vlastní měření* a *ukončení*. Poslední je *přizpůsobení měřicí škály*.



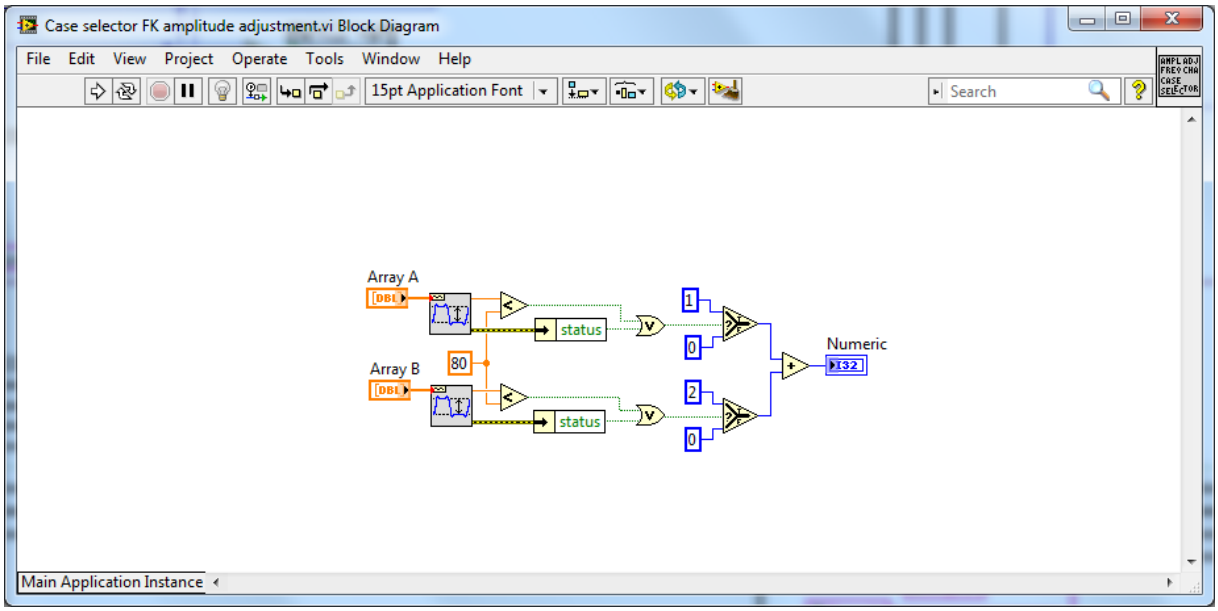
Obrázek 68: Blokový diagram třetí součásti prvního dílu Case 1 druhého kroku prvního procesu šesté části nového subprogramu Frequency Characteristics.

Hlavní část tohoto bloku (obr. 69) tváří sekvence podmínek a *Case structure*, které se řídí pomocí subVI *Case selector FK amplitude adjustment.vi*.



Obrázek 69: Blokový diagram čtvrtého bloku třetí součásti prvního dílu Case 1 druhého kroku prvního procesu šesté části nového subprogramu Frequency Characteristics.

SubVI *Case selector FK amplitude adjustment.vi* (obr. 70) kontroluje amplitudy obou kanálů. V případě, že na obou kanálech je amplituda menší než 80 (celý rozsah je 0 až 255), nebo žádná, vybere se *Case 3*. V případě, že na obou kanálech je amplituda větší než 80, vybere se *Case 0: Default*. *Case 1*, respektive *Case 2* se vyberou v případě, že amplituda je menší než 80 na kanálu *A*, respektive *B*.

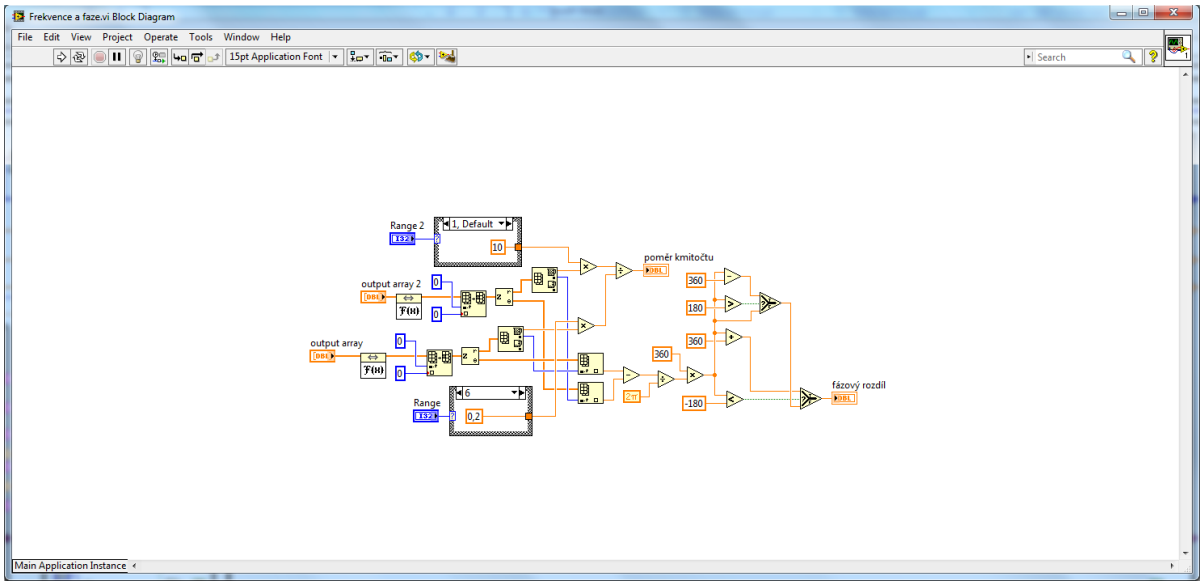


Obrázek 70: Blokový diagram Case selector FK amplitude adjustment.vi.

Case 0: Default (obr. 71) se provádí v případě, že na obou kanálech je amplituda větší než 80. V tom případě je měření u tohoto nastavení času ukončeno a pomocí nastavení *Gain A* a *Gain B* v subVI *Frekvence a fáze.vi* jsou vypočítány *poměr amplitud* a *fázový rozdíl*.

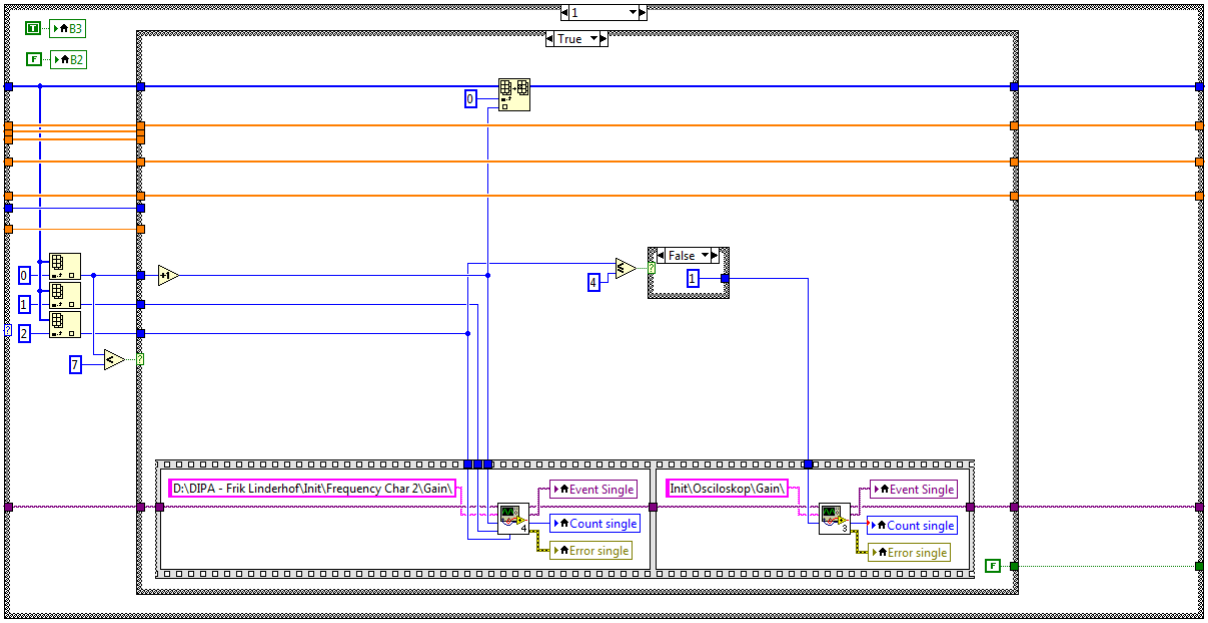
Obrázek 71: : Blokový diagram Case 0 čtvrtého bloku třetí součásti prvního dílu Case 1 druhého kroku prvního procesu šesté části nového subprogramu Frequency Characteristics.

Nastavení *Gain A* a *Gain B* jsou ve subVI *Frekvence a fáze.vi* (obr. 72) přepočítány na maximální napěťový rozsah. Ta jsou vynásobena maximálními amplitudami na obou kanálech. Tyto dvě hodnoty jsou vzájemně vyděleny a udávají *poměr amplitud*, tj, přenos. *Fázový rozdíl* je vypočítán pomocí fází maximální amplitudy. V *Case 0* jsou poměr amplitud, fázový rozdíl a dříve určená frekvence přidány do tří polí.



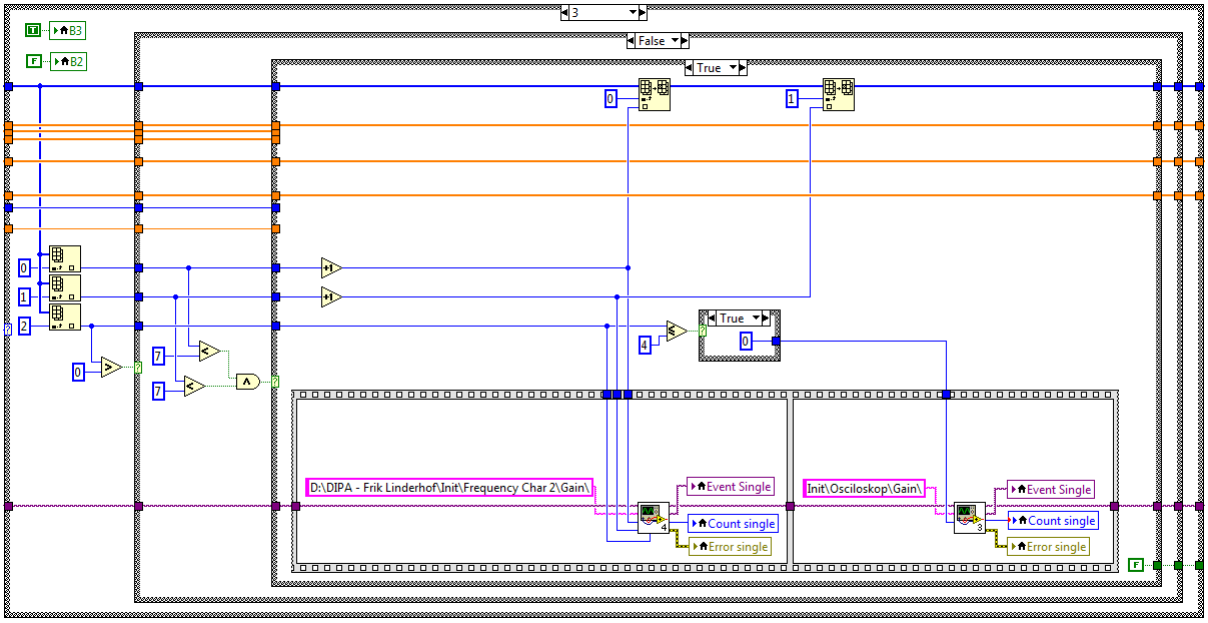
Obrázek 72: Blokový diagram Frekvence a fáze.vi.

Case 1 (obr. 73) se provádí v případě, že na kanálu *A* je signál s amplitudou menší než 80, ale na kanálu *B* je signál s amplitudou větší než 80. Nejprve se kontroluje, jestli nastavení *Gain A* už dosáhlo jeho nejmenší krajní meze – 100 mV. V případě, že ano, jsou pomocí subVI *Frekvence a fáze.vi* vypočítány *poměr amplitud* a *fázový rozdíl*. V případě, že ne, je posílán kód k přizpůsobení nastavení *Gain A*. *Case 2* funguje stejně jako *Case 1* s rozdílem, že kanál *B* má amplitudu menší než 80 a *Gain B* je přizpůsoben.



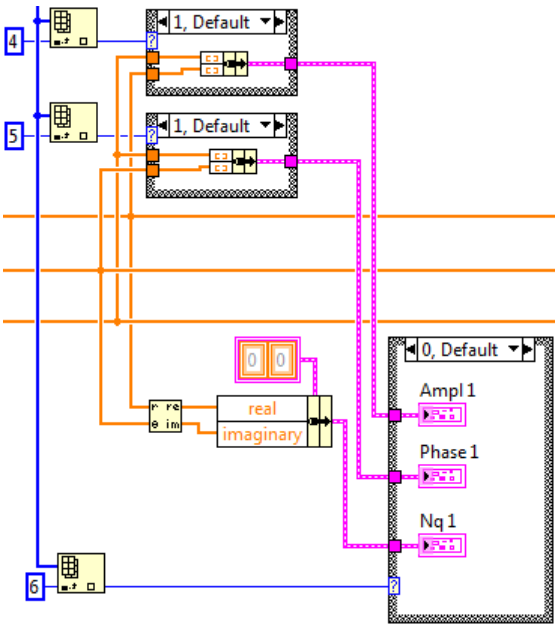
Obrázek 73: Blokový diagram Case 1 čtvrtého bloku třetí součásti prvního dílu Case 1 druhého kroku prvního procesu šesté části nového subprogramu Frequency Characteristics.

Case 3 (obr. 74) se provádí v případě, že na obou kanálech jsou signály s amplitudou menší než 80. Nejprve se kontroluje, jestli nastavení *Gain Out* už dosáhlo jeho největší krajní meze – 10 V. V případě, že ne, je posílán kód k přizpůsobení nastavení *Gain Out*. V případě, že ano, jsou kontrolovány, jestli nastavení *Gain A* i *Gain B* už dosáhla jejich nejmenší krajní meze – 100 mV. V případě, že ano, jsou pomocí subVI *Frekvence a fáze.vi* vypočítány *poměr amplitud* a *fázový rozdíl*. V případě, že ne, jsou posílány kódy k přizpůsobení nastavení *Gain A* a *Gain B*.



Obrázek 74: Blokový diagram Case 3 čtvrtého bloku třetí součásti prvního dílu Case 1 druhého kroku prvního procesu šesté části nového subprogramu Frequency Characteristics.

Poslední díl tohoto *case* (obr. 75) je příprava na zobrazování. Hodnoty v polích *poměr amplitud* a *fázový rozdíl* jsou spárovány a uloženy do polí *Ampl n* a *Phase n* na základě nastavení skupiny *Measurement*. Dále jsou hodnoty pomocí funkce *Polar To Re/Im Function* konvertovány a uloženy do polí *Nq n*.

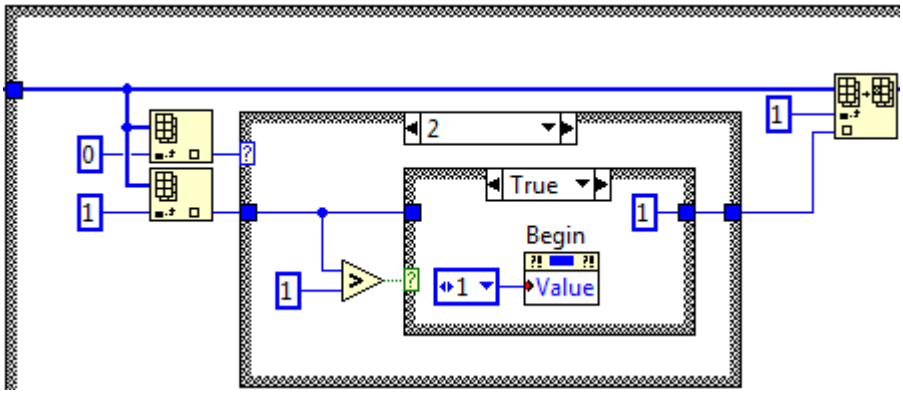


Obrázek 75: Blokový diagram druhého dílu Case 1 druhého kroku prvního procesu šesté části nového subprogramu Frequency Characteristics.

Case 2: Measurement / Clear je stejný jako v předchozích subprogramech. *Case 3: Decades* (obr. 76) je malý *case*, ve kterém se přizpůsobí hodnoty skupiny *Begin* tak, jak jsou označené v tab. 11. *Case 4: Begin* funguje stejným způsobem, ale opačně. *Case 5: View* funguje podobný jako v předchozích subprogramech.

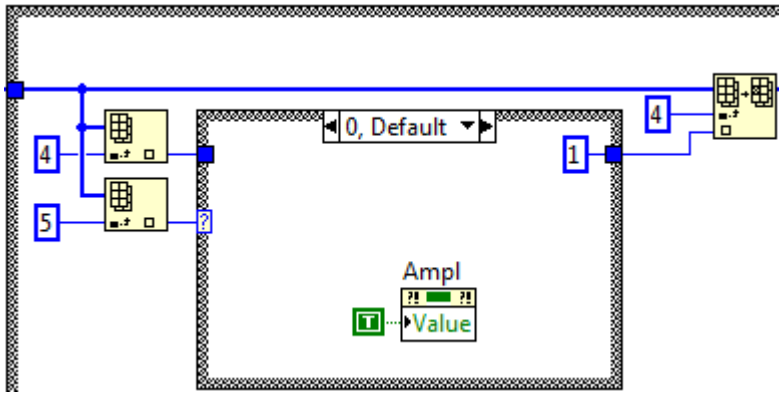
Tabulka 11: Možné nastavení skupin Begin a Decades.

		Decades			
		1	2	3	4
Begin	1 Hz	Ano	Ano	Ano	Ano
	10 Hz	Ano	Ano	Ano	–
	100 Hz	Ano	Ano	–	–
	1 kHz	Ano	–	–	–



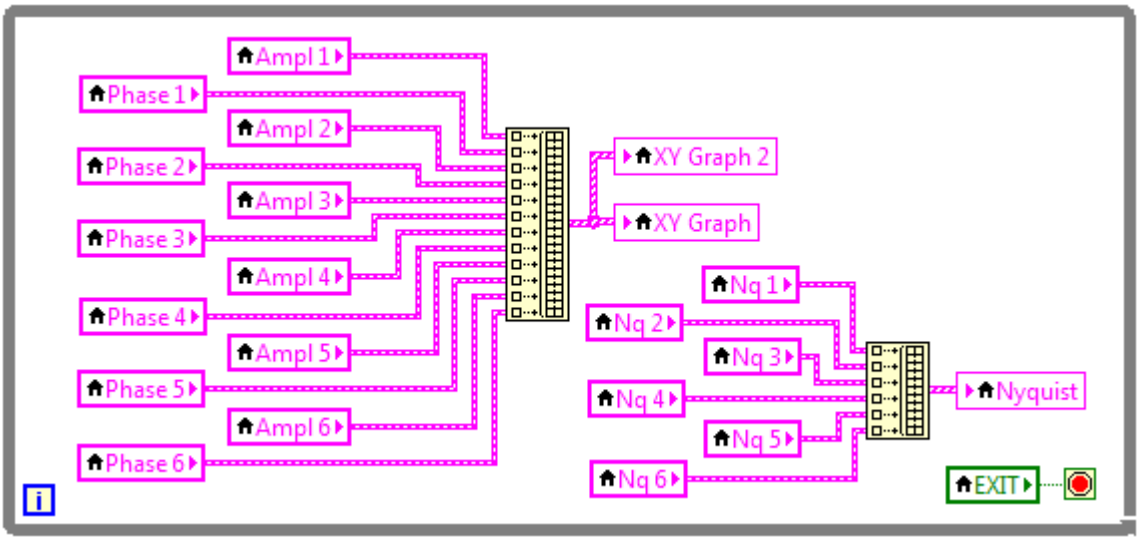
Obrázek 76: Blokový diagram Case 3 druhého kroku prvního procesu šesté části nového subprogramu Frequency Characteristics.

Case 6: Display – Ampl (obr. 77) kontroluje nové nastavení tlačítka *Ampl*. V případě, že nové nastavení je vypnuté, zapíná *Phase*. *Case 7: Display – Phase* funguje stejně, kde zapíná *Ampl*, v případě, že nové nastavení tlačítka *Phase* je vypnuté. Poslední *Case 8: Exit*, funguje stejně jako v předchozích subprogramech.



Obrázek 77: Blokový diagram Case 6 druhého kroku prvního procesu šesté části nového subprogramu Frequency Characteristics.

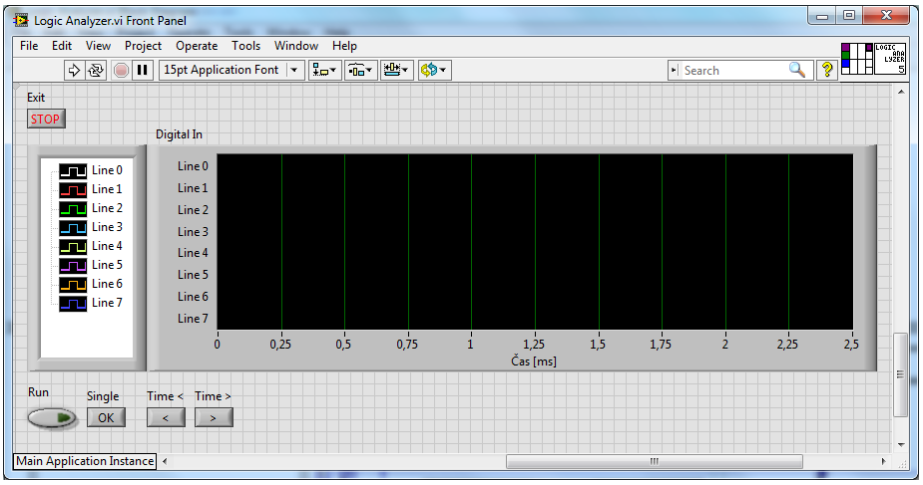
Druhý proces šesté části, *zobrazování* (obr. 78), seskupí pomocí funkce *Build Array Function* všechna pole hodnot *Ampl n* a *Phase n* do grafů *XY Graph* a *XY Graph 2* a všechna pole hodnot *Nq n* do grafu *Nyquist*. Sedmá část, *ukončení*, funguje stejně jako v předchozích subprogramech.



Obrázek 78: Blokový diagram druhého procesu šesté části nového subprogramu Frequency Characteristics.

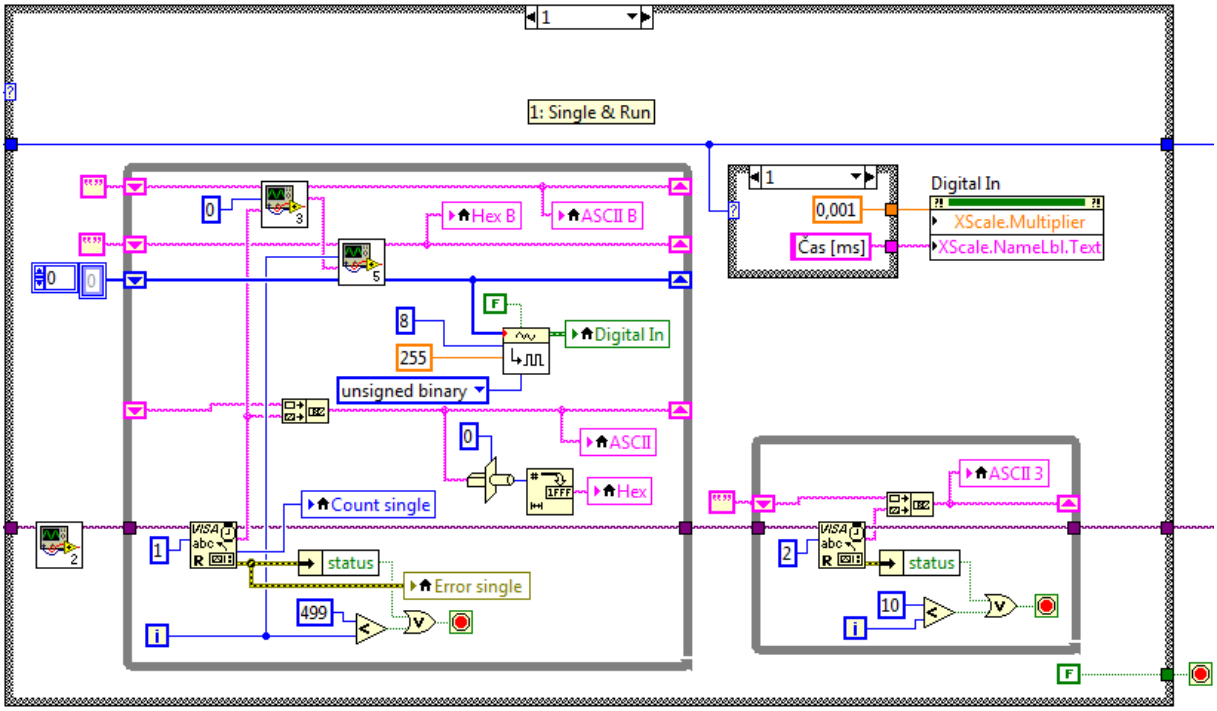
3.3.6. Režim *Logic Analyzer*

Nový subprogram *Logic Analyzer* (obr. 79) není rozšířen v porovnání s originálním subprogramem. Celý program se skládá ze sedmi částí, kde prvních pět se používá pro inicializaci. Další část je *vlastní měření*, která je rozdělena do dvou kroků: *stanovení* a *provádění*. První krok *stanovení* funguje stejně jako v předchozích subprogramech.



Obrázek 79: Snímek obrazovky Front Panelu nového subprogramu Logic Analyzer.

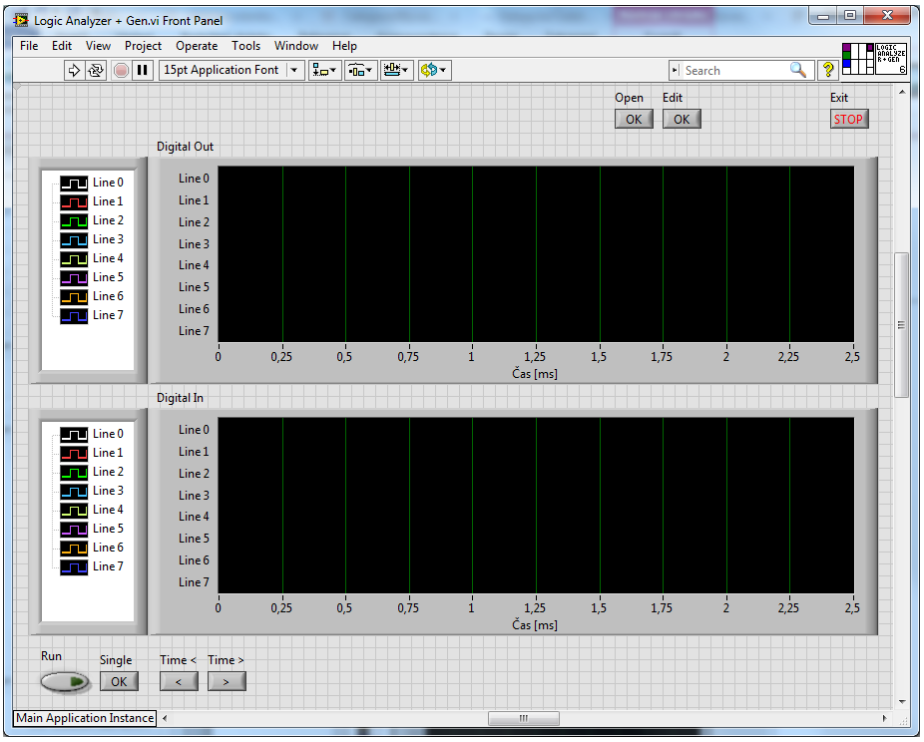
Krok provádění se skládá z pěti *casů*: *Case 0: Default*, *Case 1: Single & Run*, *Case 2: Změnit čas (-1)*, *Case 3: Změnit čas (+1)* a *Case 5: Exit*. Všechny *case* jsou podobné jako v předchozích subprogramech. V *Case 1: Single* (obr. 80) se pomocí funkce *DWDT Analog to Digital.vi* konvertuje hexadecimální naměřený signál do 8bitového signálu. Poslední část *ukončení* funguje stejně jako u předchozích subprogramů.



Obrázek 80: Blokový diagram Case 1 druhého kroku šesté části nového subprogramu Logic Analyzer.

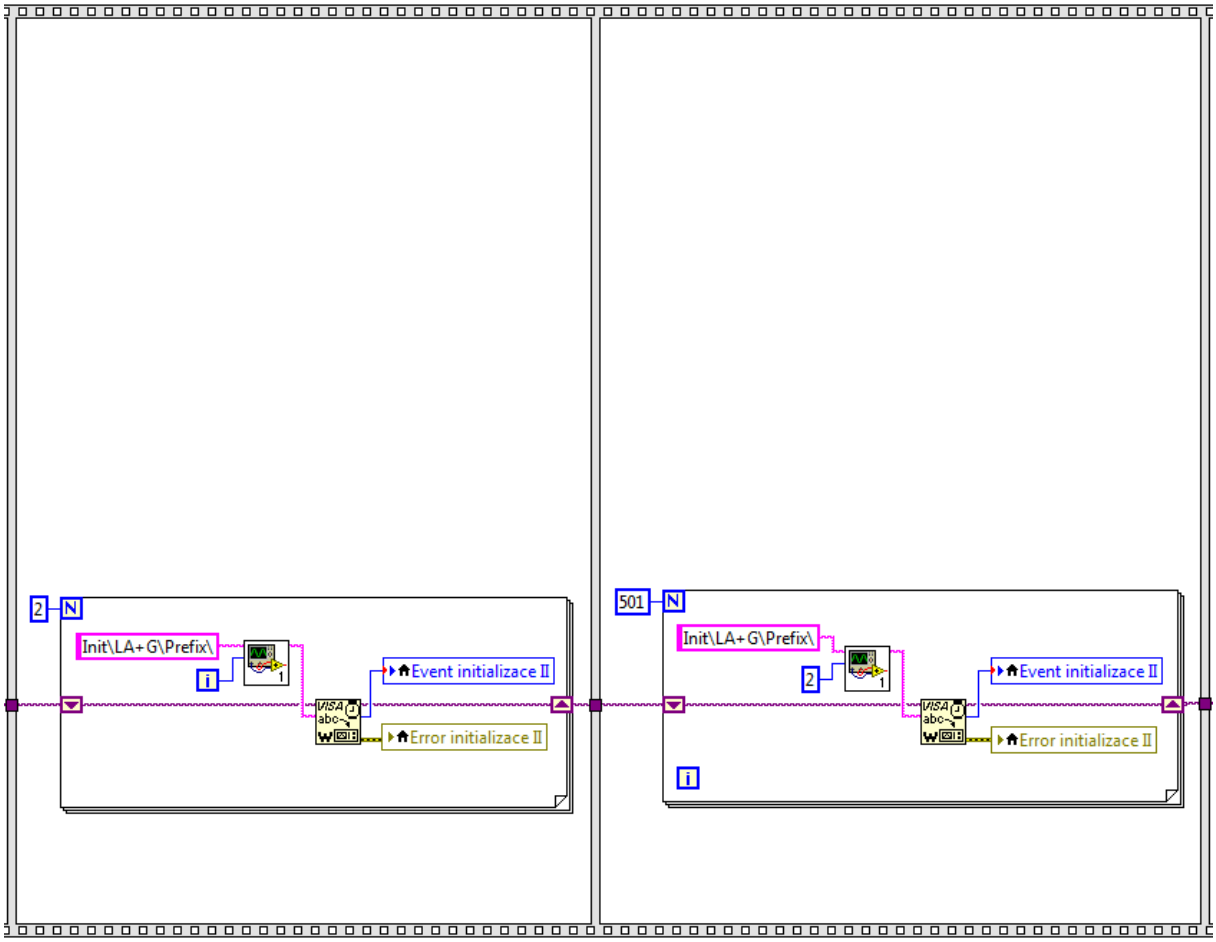
3.3.7. Režim *Logic Analyzer + Gen*

Stejně jako subprogram *Logic Analyzer* není nový subprogram *Logic Analyzer + Gen* (obr. 81) rozšířen novými funkcemi. V porovnání se subprogramem *Logic Analyzer* jsou přidány další graf *Digital Out* a skupina *Output* s tlačítky *Open* a *Edit*. Celý subprogram se skládá z devíti částí, ze kterých je pro inicializaci použito prvních sedm. *Inicializaci* lze rozdělit do dvou kroků: *vlastní inicializace* a *inicializace výstupního signálu*. Vlastní inicializace je rozdělena do pěti částí.



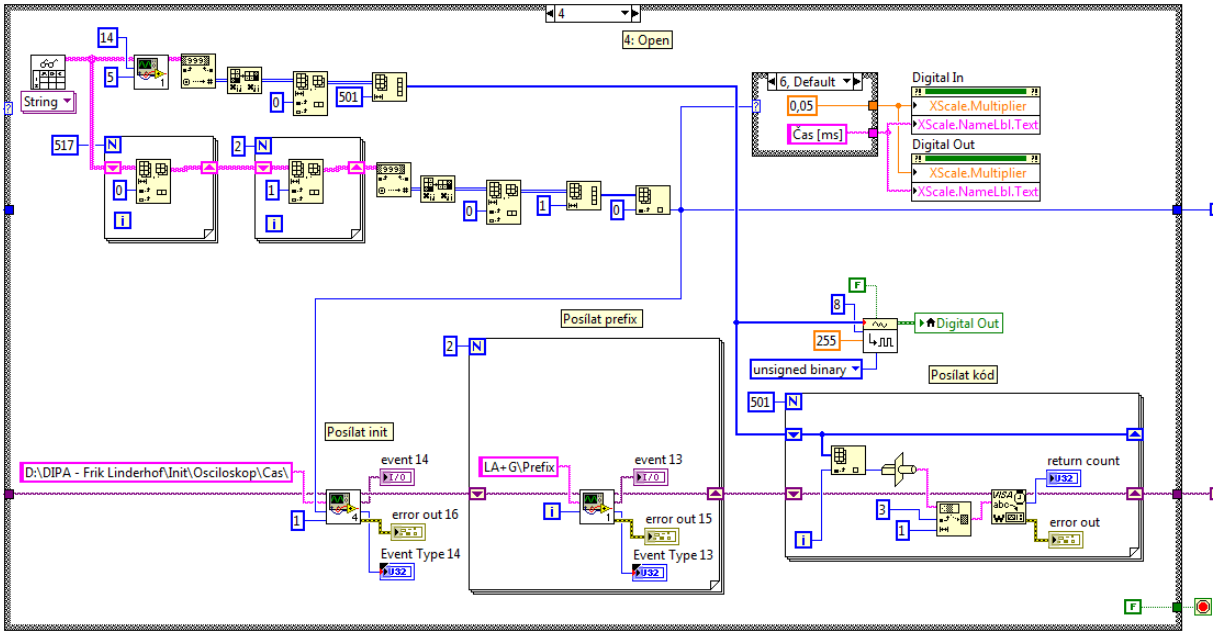
Obrázek 81: Snímek obrazovky Front Panelu nového subprogramu Logic Analyzer + Gen.

Inicializace výstupního signálu se skládá z dvou částí (obr. 82). První (a celkově šestá) část posílá do A&DDU kód prefixu a v druhé (celkově sedmé) části je posílán hexadecimální výstupní dummy signál.



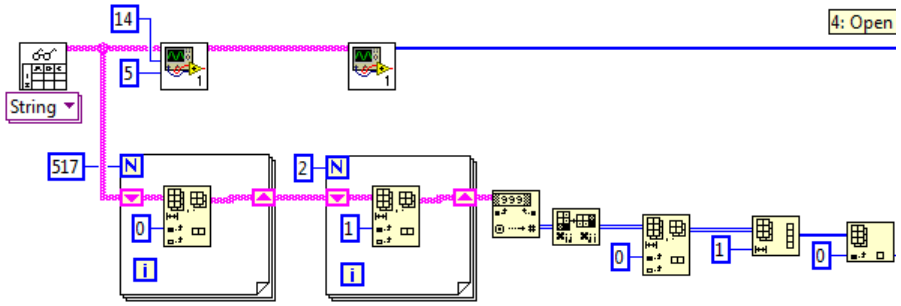
Obrázek 82: Blokový diagram šesté a sedmé části nového subprogramu Logic Analyzer + Gen.

V osmé části se nachází *vlastní měření* s kroky *stanovení* a *provádění*. Krok *provádění* se skládá ze sedmi *casů*: *Case 0: Default*, *Case 1: Single & Run*, *Case 2: Změnit čas (-1)*, *Case 3: Změnit čas (+1)*, *Case 4: Open*, *Case 5: Edit* a *Case 6: Exit*. *Case 0* až *3* a *Case 6* jsou stejné jako v subprogramu *Logic Analyzer*. *Case 4: Open* (obr. 83) lze rozdělit do šesti bloků; *čtení*, *posílání inicializace*, *posílání prefixu*, *posílání kódu*, *přizpůsobení škály* a *zobrazování*.



Obrázek 83: Blokový diagram Case 4 druhého kroku osmé části nového subprogramu Logic Analyzer + Gen.

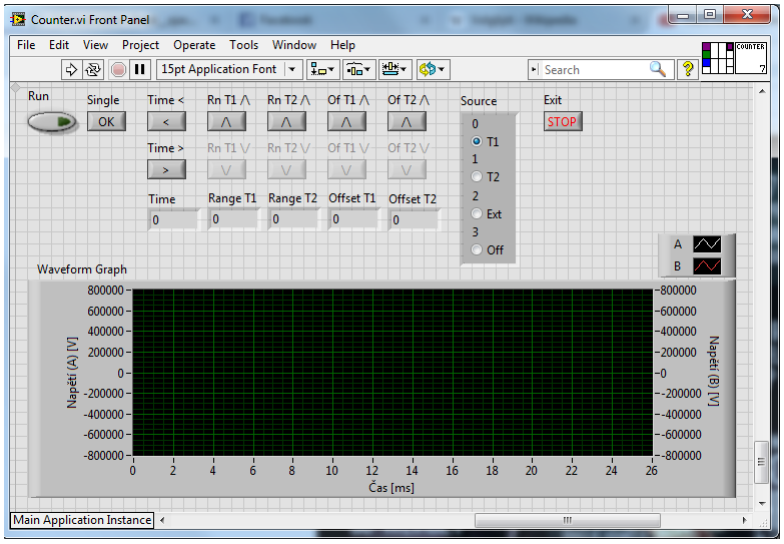
První krok *čtení* (obr. 84) konvertuje data ze souboru **.dio* do pole typu *integer* a nastaví parametr *Time* pomocí subVI *StringArrayTrimmer.vi* a *String2Array.vi*. V dalším kroku *posílání inicializace* se posílá parametr *Time* a zároveň se přizpůsobí škála krokem *přizpůsobení škály*. Ve třetím kroku posílání prefixu se posílá do A&DDU kód prefixu. Krok *posílání kódu* posílá do A&DDU kód, který je konvertován v prvním kroku a v posledním kroku *zobrazování* se stejný kód zobrazí v grafu *Digital Out*. *Case 5: Edit* funguje stejně jako *Case 4*, jen je první krok *čtení* nahrazen subprogramem *Digital Waveform Editor*. Poslední část je *ukončení* a to funguje stejně jako v předchozích subprogramech.



Obrázek 84: Blokový diagram prvního bloku Case 4 druhého kroku osmé části nového subprogramu Logic Analyzer + Gen.

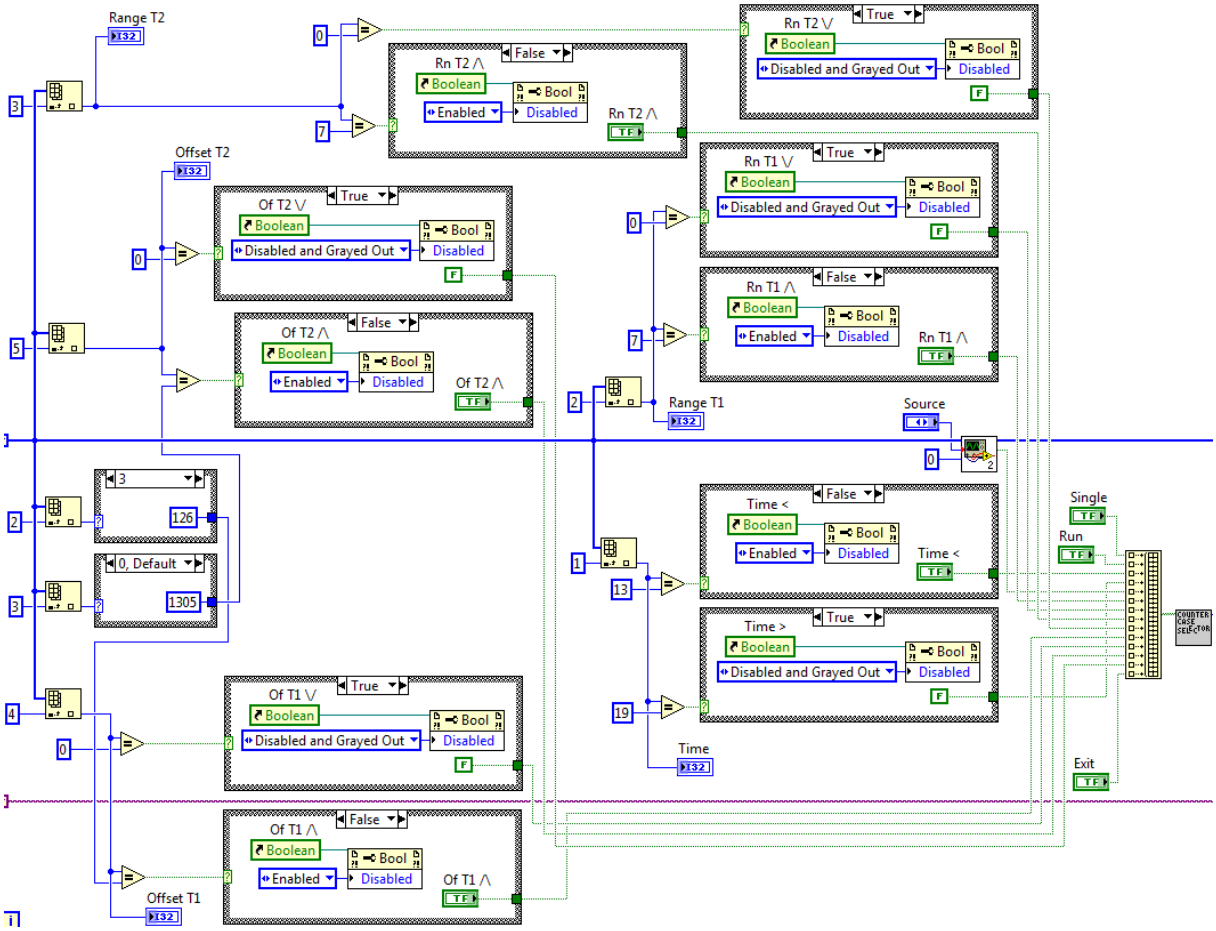
3.3.8. Režim *Counter*

Poslední subprogram *Counter* (obr. 85) nemá rozšířené možnosti. Subprogram se skládá ze sedmi částí. První tři části jsou používané pro *inicializaci*, další je *vlastní program* a poslední je *ukončení*. Čtvrtá část *vlastní program* je rozdělena do dvou kroků: *stanovení* a *provádění*.



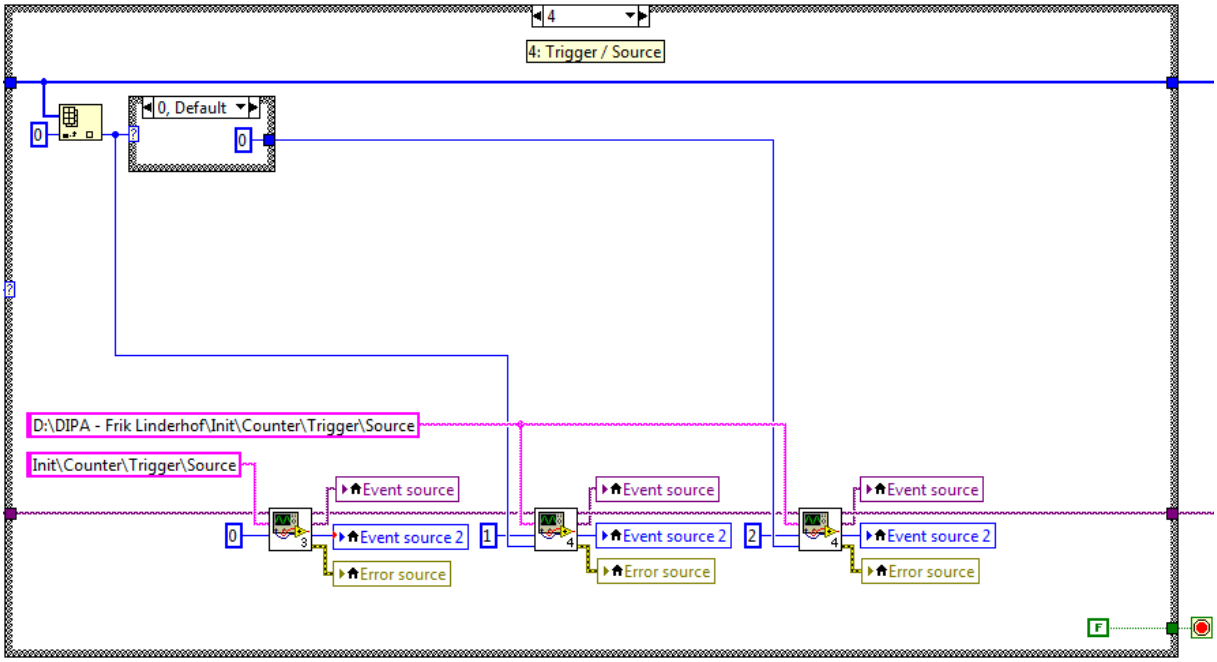
Obrázek 85: Snímek obrazovky Front Panelu nového subprogramu Counter.

Krok *stanovení* (obr. 86) vnáší jednu zajímavou věc, a to nastavení maximální krajní meze *Offsetu*. Nastavení maximální krajní meze je variabilní a závisí na nastavení *Range*. Protože existují dva *Triggery*, má každý *Trigger* variabilní krajní mez. Variabilita nastavení je prováděna pomocí *Case Structure*.



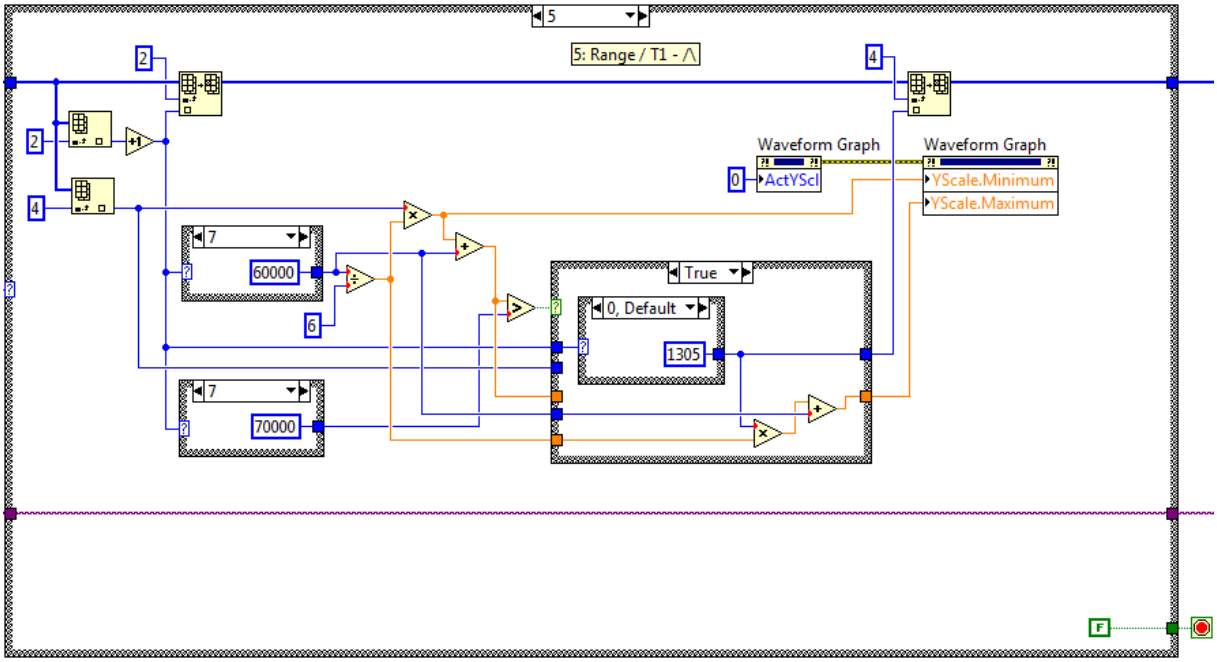
Obrázek 86: Blokový diagram prvního kroku čtvrté části nového subprogramu Counter.

Druhý krok *provádění* se skládá z *Case Structure*. *Case 0: Default*, *Case 1: Single*, *Case 2: Time – <* a *Case 3: Time – >* jsou podobné jako v ostatních subprogramech. *Case 4: Trigger / Source* (obr. 87) se ale liší. Data jsou posílána do ADD&U ve třech krocích.



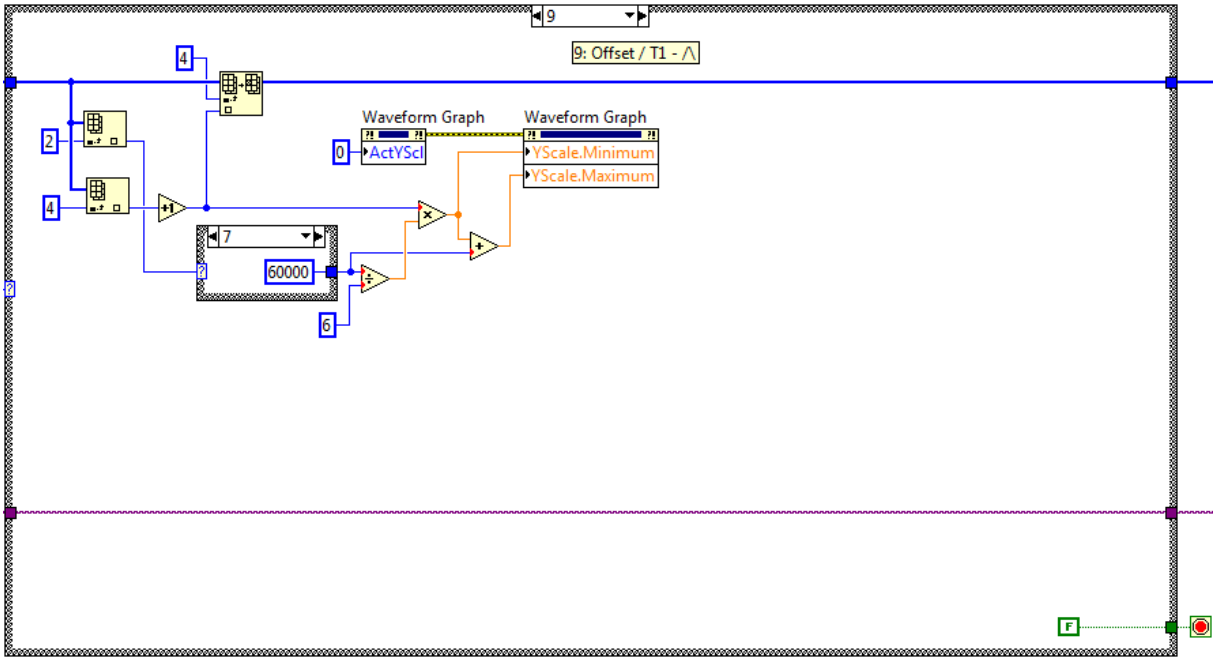
Obrázek 87: Blokový diagram Case 4 druhého kroku čtvrtého části nového subprogramu Counter.

Další case, *Case 5: Range / T1 – /* (obr. 88), zvětšuje *Range* prvního *Triggeru*. Komplexita tohoto *case* je způsobena nutností přizpůsobení maximální krajní meze *Offsetu* prvního *Triggeru*. Dále se v tomto *case* ještě přizpůsobí škála grafu. *Case 6: Range / T1 – \/* je stejná jako *Case 5* s rozdílem, že se nepřizpůsobí maximální krajní mez. *Case 7: Range / T2 – /* funguje stejně jako *Case 5*, kde *Range T1* je nahrazen *Range T2* a *Offset T1* je nahrazen *Offset T2*. *Case 8*: *Range / T2 – \/* je kombinace těchto úprav. *Range T1*, respektive *Offset T1* jsou nahrazeny *Range T2*, respektive *Offset T2* a maximální krajní mez se nepřizpůsobí.



Obrázek 88: Blokový diagram Case 5 druhého kroku čtvrtého části nového subprogramu Counter.

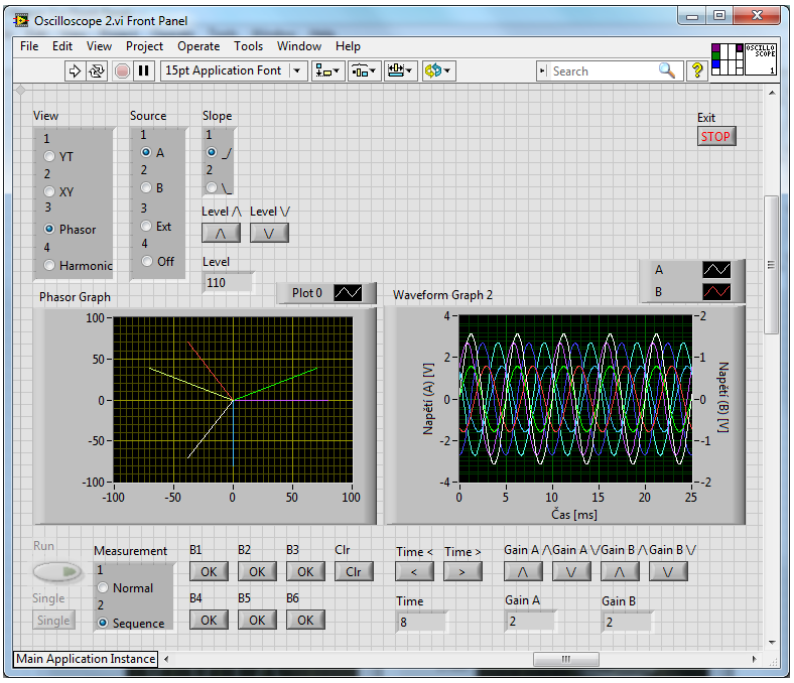
Case 9: Offset / T1 – / (obr. 89) přizpůsobí nastavení *Offset* prvního *Triggeru* pomocí funkce *Increment Function* a přizpůsobí škálu grafu. *Case 10: Offset / T1 – \/* je podobný, ale přizpůsobí se pomocí funkce *Decrement Function*. *Case 11: Offset / T2 – /*, respektive *Case 12: Offset / T2 – \/* jsou identické ke *Case 9: Offset / T1 – /*, respektive *Case 10: Offset / T1 – \/* s rozdílem, že je přizpůsoben nastavení *Offset* druhého *Triggeru*, místo prvního *Triggeru*. Poslední case, *Case 13: Exit* je podobný jako v předchozích subprogramech. Pátá část *ukončení* funguje stejně jako u ostatních subprogramů.



Obrázek 89: Blokový diagram Case 9 druhého kroku čtvrtého části nového subprogramu Counter.

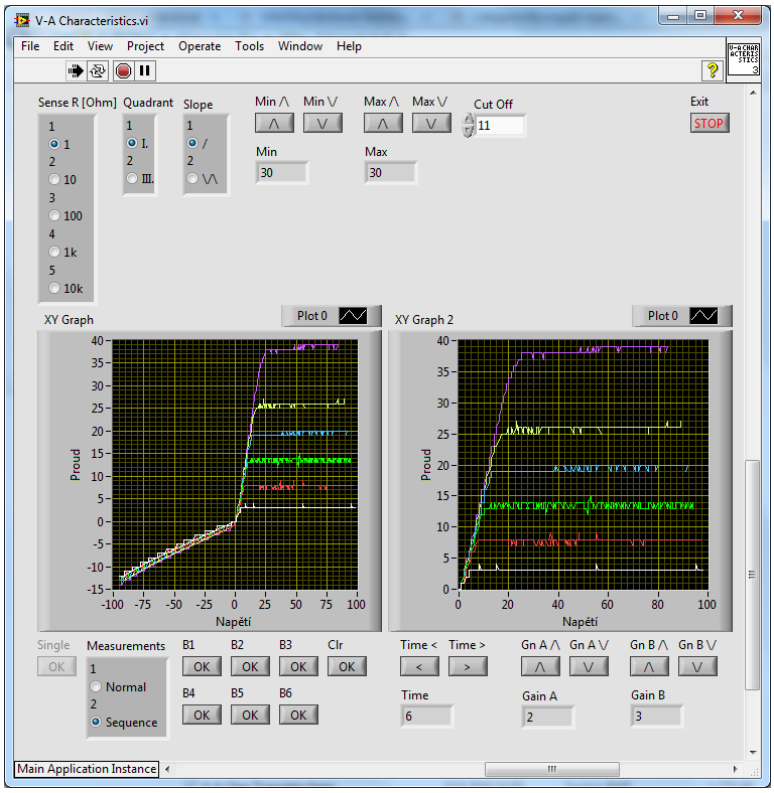
3.4. Diskuze výsledků

Základní funkčnost programu byla otestována pomocí sady základních měření. Jako ukázku lze uvést například měření trojfázové soustavy (obr. 90). Měření bylo prováděno pomocí modulu *3 Phase System* v subprogramu *Oscilloscope*. Toto měření ukazuje výhodu možnosti ukládat šest měření v novém subprogramu v porovnání s originálním programem. Skupina *View* je nastavena na *Phasor*. V tomto měření fungovalo šesté měření jako referenční (fialové), vzhledem ke kterému je počítán fázový rozdíl ostatních. První (bílé) a druhé (červené) měření (fázová napětí) jsou vůči referenčnímu měření posunutá o $\pm 120°$. Další tři měření (sdružená napětí) jsou zase vzájemně posunutá o $\pm 120°$ a vůči předchozím posunuté o $30°$.



Obrázek 90: Snímek obrazovky Front Panelu nového subprogramu Oscilloscope s měřením trojfázové soustavy.

Další ukázka (obr. 91) je měření výstupních charakteristik bipolárního tranzistoru. Měření bylo prováděno pomocí modulů *Transistor Bipolar* a *Programmable DC* Supply, a digitálního multimetru v subprogramu *V-A Characteristics*. Stejně jako v předchozím případě ukazuje tato ukázka výhodu ukládání šesti, což umožní podrobnější parametrizaci výstupních charakteristik. Pomocí kanálu *A* je měřeno napětí U_{CE} a kanálem *B* je měřeno napětí na rezistoru 100 Ω, které odpovídá kolektorovému proudu I_C.



Obrázek 91: Snímek obrazovky Front Panelu nového subprogramu s měřením bipolárního transistoru.

Je však třeba zmínit také některé problémy. Nejvýznamnější z nich je potřeba upravené propojky konektorů. Zatím se mi nepodařilo zjistit, jak nakonfigurovat LabVIEW funkce pro práci s RS-232 tak, aby na sběrnici neposílaly nežádoucí puls.

Hlavní program momentálně nabízí možnost měřit na dvou A&DDU zároveň. Teoreticky bylo by možné tento počet rozšířit podle počtu portů (USB a RS-232) na počítači. V tom případě by ale bylo potřeba vymyslet nový způsob synchronizace, protože současný způsob vyžaduje ještě důkladné debuggování.

Závěr

Hlavním cílem této práce bylo integrovat systém µLab do LabVIEW a rozšířit stávající možnosti, které nabízí program rc2000. Plán práce se skládal z pěti bodů, z nichž první čtyři byly splněny. Pátý bod, který byl jen volitelný a zahrnoval využití *Data Dashboard for LabVIEW* v systému *Android,* nebyl udělán, a to hlavně z důvodu časového omezení a komplikací během programování.

V teoretické části byly uvedeny popisy tří základních kamenů této práce: originální program rc2000, rozhraní RS-232 a vývojové prostředí softwaru LabVIEW. V *systému µLab a programu rc2000* jsou popsány všechny subprogramy a funkce. Rozhraní RS-232 je popsáno hlavně ze strany hardwaru.

V postupu a výsledcích jsou popsány poznatky získané studiem komunikačního protokolu, problémy s komunikací a jejich řešení a popsán nový program v LabVIEW. Studium komunikačního protokolu ukazuje příklad zjištění významu parametrů protokolu pro skupinu *Trigger*. U nového programu jsou popsány blokové diagramy, jejich funkce a případná rozšíření oproti původnímu programu rc2000. K rozšířením patří například možnost spustit dva subprogramy zároveň (se dvěma jednotkami A&DDU, připojenými k jednomu počítači), v podskupině *Sequence* tlačítka *B5* a *B6* ve všech analogových subprogramech, nezávislá možnost startovat subprogram, přidání skupiny *View* a podskupiny *Gain – B* do subprogramu *Oscilloscope + Gen*, a přidání dalších možností do skupin *Decades* a *Begin* a nové skupiny *Output* do subprogramu *Frequency Characteristics*.

Dále je vhodné zmínit možnost dalších případných rozšíření, které nebyly z časových důvodů realizovány. Je možné například nabídnout v subprogramech *Oscillscope* nebo *Oscilloscope + Gen* možnost přepínat mezi unipolárním a bipolárním měřením, testovat, na jaké rychlosti pracují připojené A&DDU, rozšířit možnosti měření V-A charakteristik na měření parametrických charakteristik (například pro tranzistor) a měření aktivních součástek.

Summary

The main goal of this thesis was to integrate the μLab system into LabVIEW and to expand the possibilities, which are present in the program rc2000. The plan of the thesis consisted of five points, from which the first four were executed. The fifth point however was only optional; it comprised the use of *Data Dashboard for LabVIEW* in *Android*, and was due to limited time and complications during programming never implemented.

In the second chapter *teoretická část* are the three corner stones of this thesis, the original program rc2000, the serial port RS-232 and the development environment LabVIEW, introduced. In the first section *systém μLab a program rc2000* all the subprograms and functions are described. The serial port RS-232 is described mainly from a hardware point of view.

In the third chapter *postup a výsledky* are the findings described of the study of the communication protocol, problems with the communication and solution to these problems and the new program in LabVIEW is described. The study of the communication protocol showcases an example of the meaning of parameters of the protocol for the function group *Trigger*. For the new program the block diagrams are described, how they work and, if added, what new functionality has been added, in comparison with the original software rc2000. Examples of additional functionality are the possibility to run two subprograms at the same time (with two A&DDU, connected to one computer), in the function group Sequence the button *B5* and *B6* in all analogue subprograms, a possibility to start a subprogram independently from the main program, the addition of the function group *View* and subgroup *Gain – B* to the subprogram *Oscilloscope + Gen*, and the expansion of the function groups *Decades* and *Begin* and addition of a new function group *Output* to the subprogram *Frequency Characteristics*.

It stands to reason to mention some of the other possibilities of additional expansions, which were not implemented due to time restrains. It is for example possible to add the possibility to switch between the regimes for unipolar and bipolar measurements, to test at which speed a connected A&DDU works, add the possibility for measuring parametric characteristics (for example of a transistor) or for measuring active components in V-A Characteristics regime.

Seznam použitých zdrojů

[1] http://www.rcdidactic.cz/cz/

[2] https://cs.wikipedia.org/wiki/RS-232

[3] https://nl.wikipedia.org/wiki/RS-232

[4] https://en.wikipedia.org/wiki/RS-232

[5] https://nl.wikipedia.org/wiki/LabVIEW

[6] https://en.wikipedia.org/wiki/LabVIEW

Seznam použitých přístrojů a softwarů

Seznam použitých přístrojů

- Analog & Digital Data Unit, výrobce RC společnost s r. o. přístroje pro vědu a vzdělání, inv. číslo 1109915
- Analog & Digital Data Unit, výrobce RC společnost s r. o. přístroje pro vědu a vzdělání, inv. číslo 1109908
- 3 Phase System, výrobce RC společnost s r. o. přístroje pro vědu a vzdělání, inv. číslo 3184014
- Moduly systému µLab (například panel, Transistor Bipolar, Programmable DC Supply)
- Zdroj Z 5, výrobce ZPA Brno, ser. číslo 8703049 (Napěťový zdroj)
- Zdroj Z 5, výrobce ZPA Brno, ser. číslo 0209086 (Napěťový zdroj)
- HM0722, výrobce HAMEG Rohde & Schwarz, inv. číslo 3207605 (Osciloskop)
- HO3508, výrobce HAMEG Rohde & Schwarz, inv. číslo 3207606 (Modul přídavný)
- Digitus USB/RS-232 převodník
- AXIO MET AX-585B Digital Multimeter

Seznam použitých softwarů

- rc2000 – ovládací program systému µLab
- National Instruments LabVIEW 2013 – vývojové prostředí
- Device Monitoring Studio – program pro sledování sériové komunikace
- Hex Editor Neo
- Serial Port Monitor – program pro sledování sériové komunikace
- Realterm – emulátor terminálu