JIHOČESKÁ UNIVERZITA V ČESKÝCH BUDĚJOVICÍCH

PEDAGOGICKÁ FAKULTA

KATEDRA FYZIKY

Získání a zpracování digitálních dat pro určení přesné efektivní hodnoty napětí

Bakalářská práce

Autor: Pavel Čurda

Vedoucí práce: doc. PaedDr. Petr Adámek, Ph.D.

Konzultant: Ing. Václav Král

2006

Anotace

Tato bakalářská práce vznikla jako externí projekt při realizaci energetického zařízení: Terminál Automatizovaných Funkcí Transformátoru 112 kV (TAFT 112®). Toto zařízení je určeno pro regulaci a řízení zhášecí tlumivky elektrorozvodného transformátoru. Vlastní bakalářská práce se zabývá především problematikou analogově-digitálního (AD) převodu a jeho řízení. Dále se pak zabývá analogovými i digitálními metodami měření přesné efektivní hodnoty napětí (true RMS). Součástí bakalářské práce je i praktická aplikace uvedené problematiky na terminálu TAFT 112®.

Prohlášení

Prohlašuji, že jsem tuto práci vypracoval samostatně a že jsem všechny použité zdroje uvedl v seznamu použité literatury na konci této práce.

Pavel Čurda

Poděkování

Na tomto místě bych rád poděkoval doc. PaedDr. Petru Adámkovi, Ph.D., vedoucímu bakalářské práce, za odborné připomínky. Dále bych rád poděkoval Ing. Václavu Královi za možnost pracovat na externím projektu, cenné rady a trpělivost. Na závěr bych rád ještě zmínil Ing. Ondřeje Staška, se kterým jsem spolupracoval při vývoji softwaru.

Obsah

Úvod

Tato bakalářská práce vznikla jako externí projekt na základě požadavku energetické společnosti EGC-EnerGoConsult ČB s.r.o. Náplní tohoto projektu byl vývoj digitálního energetického regulačního zařízení: Terminál Automatizovaných Funkcí Transformátoru 112 kV (TAFT 112®).

Účelem tohoto zařízení je automatická obsluha elektrorozvodného transformátoru. Zejména se jedná o regulaci zhášecí tlumivky a případném připínání odporníku k uzlu transformátoru. Cílem bakalářské práce na tomto zařízení bylo zajištění automatické synchronizace vzorkování, tak aby každá perioda měřeného síťového napětí byla zachycena přesně v požadovaném počtu vzorků. Dalším úkolem pak byla realizace rychlého výpočtu přesné efektivní hodnoty napětí (true RMS) na základě naměřených dat.

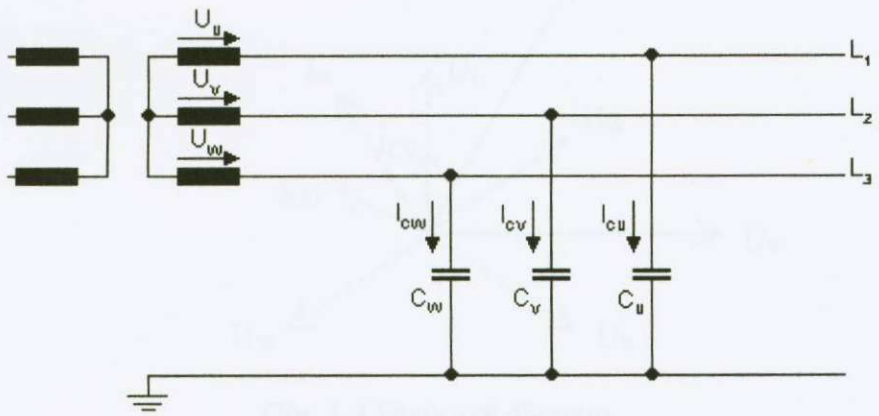
Snahou vlastní bakalářské práce je vždy shrnout teoretické údaje a názorně vyložit jejich praktickou aplikaci při realizaci terminálu TAFT 112®. Úvodní dvě kapitoly jsou proto věnovány spíše teoretickým základům a obecným principům. Dále následuje bližší se seznámení s konkrétní realizací hardwaru terminálu a řídícím mikrokontrolérem ATMEL®. Závěrečné kapitoly pak plně využívají předem uvedené teoretické poznatky při konkrétních realizacích daných problematik.

1. Způsoby uzemnění uzlu VN transformátoru

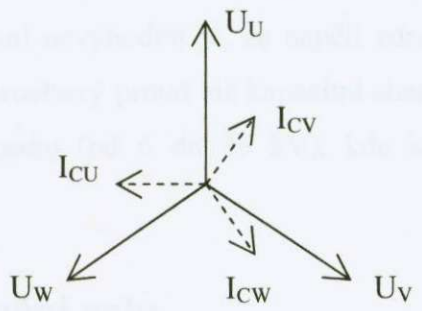
Tato kapitola si v žádném případě neklade za cíl podrobně rozvádět danou problematiku. Pouze se snaží jednoduše nastínit principy uzemnění uzlu VN transformátoru [1][2]. Dále jsou zde naznačeny principy ovládání zhášecí tlumivky terminálem.

1.1 Izolovaný uzel

Takto zapojený transformátor (obr. 1.1) nemá mezi uzlem a zemí připojeny žádné impedance [1]. Síť se pak v řádném provozu projevuje symetricky rozloženými kapacitami fází vůči zemi ($C_W = C_V = C_U$). Jednotlivá fázová napětí i parazitní kapacitní proudy mají v tomto případě prakticky stejnou velikost. Z jejich znázornění ve fázorovém diagramu (obr. 1.2) plyne, že se vzájemně kompenzují a napětí uzlu proti zemi má pak v ideálním případě nulovou hodnotu.



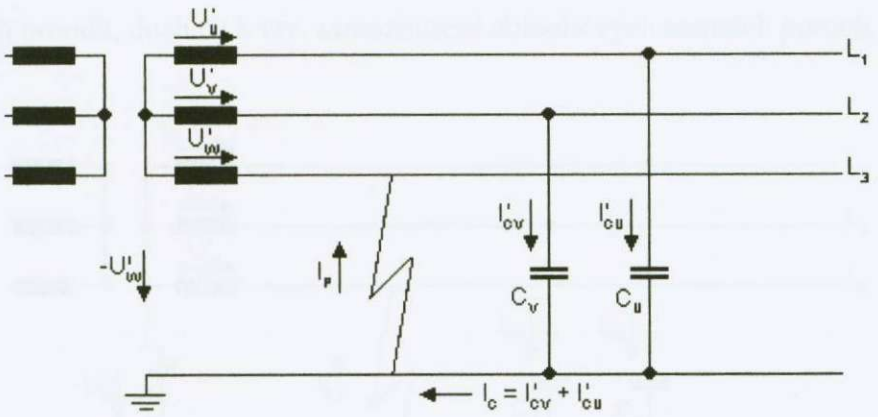
Obr. 1.1 Izolovaný uzel.



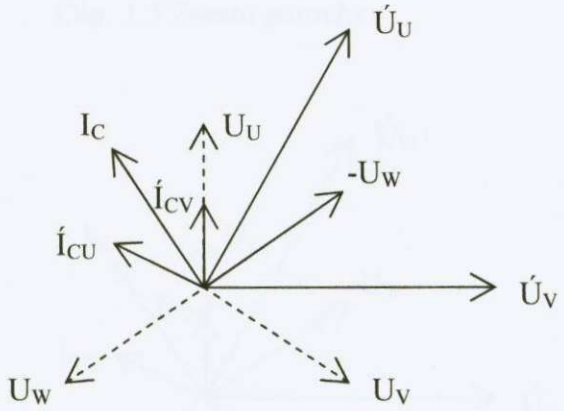
Obr. 1.2 Fázorový diagram.

V případě spojení jedné fáze (např. L_3) se zemí (obr. 1.3), vzroste napětí uzlu transformátoru na zápornou hodnotu fázového napětí ($-U_W$). Na zbylých dvou fázích (L_1

a L_2) vzroste napětí vůči zemi na sdruženou hodnotu a dokonce se změní i jejich vzájemný fázový posuv [1]. Situace je zachycena ve fázorovém diagramu na (obr. 1.4).



Obr. 1.3 Zemní porucha.



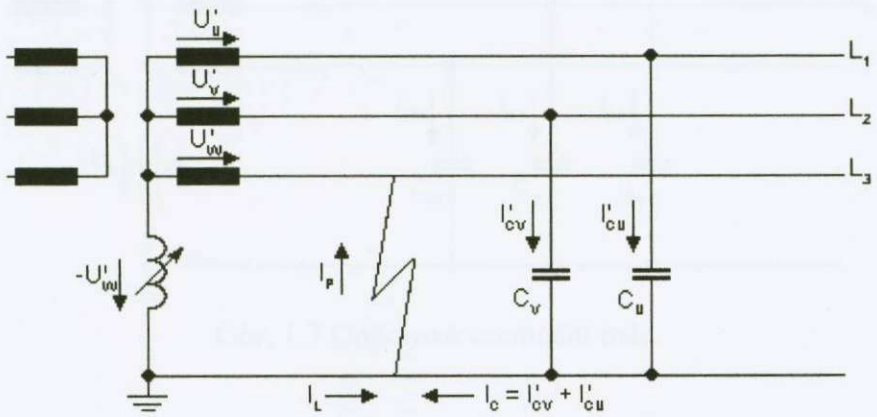
Obr. 1.4 Fázorový diagram.

Rozvodnou síť s izolovaným uzlem lze provozovat i v případě zemního spojení jedné fáze [1][2]. Její hlavní nevýhodou je, že napětí zdravých fází stoupne na sdruženou hodnotu a dále že poruchový proud má kapacitní charakter. Jako izolované sítě se realizují sítě menšího rozsahu (od 6 do 35 kV), kde kapacitní proud nepřesahuje hodnotu 20A.

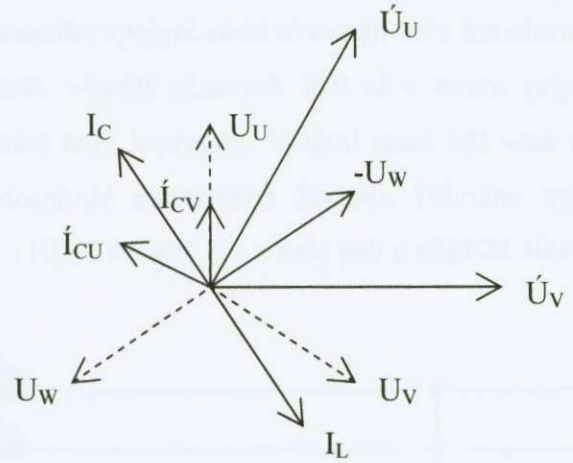
1.2 Indukční uzemnění uzlu

Indukční spojení uzlu transformátoru je provedeno pomocí laditelné zhášecí tlumivky (Petersenovy cívky) [1][2]. V případě jednofázové zemní poruchy (obr. 1.5), protéká zhášecí tlumivkou proud indukčního charakteru I_L. Vhodně nastavenou (vyladěnou) indukčností tlumivky lze docílit kompenzace kapacitních proudů I_C proudem I_L

(obr. 1.6). Výsledný poruchový proud má pak pouze činnou složku (nezakreslena ve fázorovém diagramu), která je daná především ztrátami v tlumivce a svodovými odpory jednotlivých fází. Protože velikost tohoto proudu je mnohonásobně menší, než velikost kapacitních proudů, dochází k tzv. samozhášení obloukových zemních poruch.



Obr. 1.5 Zemní porucha.



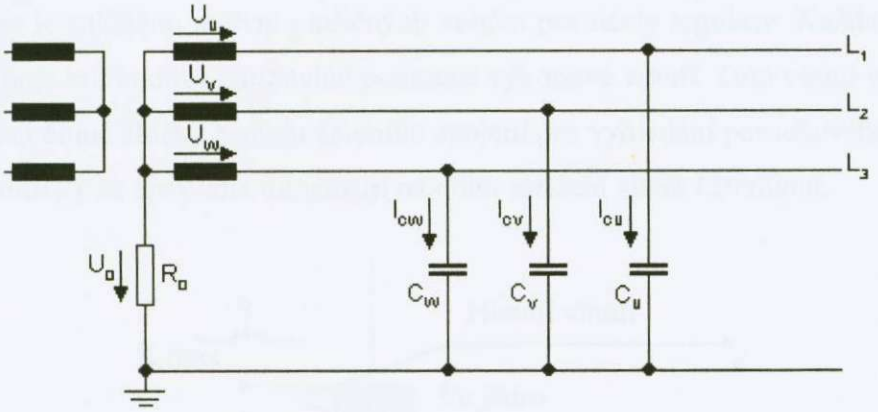
Obr. 1.6 Fázorový diagram.

Kompenzace kapacitního proudu [1][2] se využívá u sítí venkovního vedení, kde kapacitní proud nepřesahuje hodnotu 100A. Dále se využívá u kabelových sítí menšího rozsahu s kapacitním proudem do 450A.

1.3 Odporové uzemnění uzlu

Odporové uzemnění uzlu (obr. 1.7) se používá především u kabelových sítí většího rozsahu [1]. Charakter poruch je zde obvykle trvalého charakteru a nelze zde využít principu samozhášení. Rezistor zastává funkci omezení velikosti poruchových

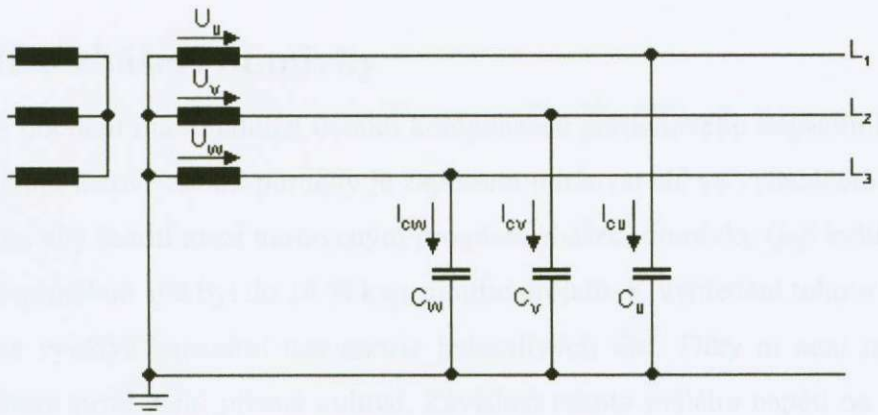
proudů při případném zemním spojením. Hlavní nevýhodou tohoto způsobu je nutnost odstavení sítě v případě poruchy.



Obr. 1.7 Odporové uzemnění uzlu.

1.4 Přímé uzemnění uzlu

Takto zapojený transformátor [1] má uzel přímo spojený se zemí (obr 1.8). V případě jednofázového zemního spojení zůstává napětí uzlu transformátoru proti zemi na prakticky nulové hodnotě. Napětí zdravých fází si v tomto případě udržuje své jmenovité hodnoty, nedochází tedy k přepětí. Vedení musí být však rychle odpojeno, protože zkratový proud dosahuje extrémních hodnot. Přímého uzemnění uzlu se zpravidla užívá u sítí VVN 110, 220 a 400 kV a dále pak u sítí NN 400V.

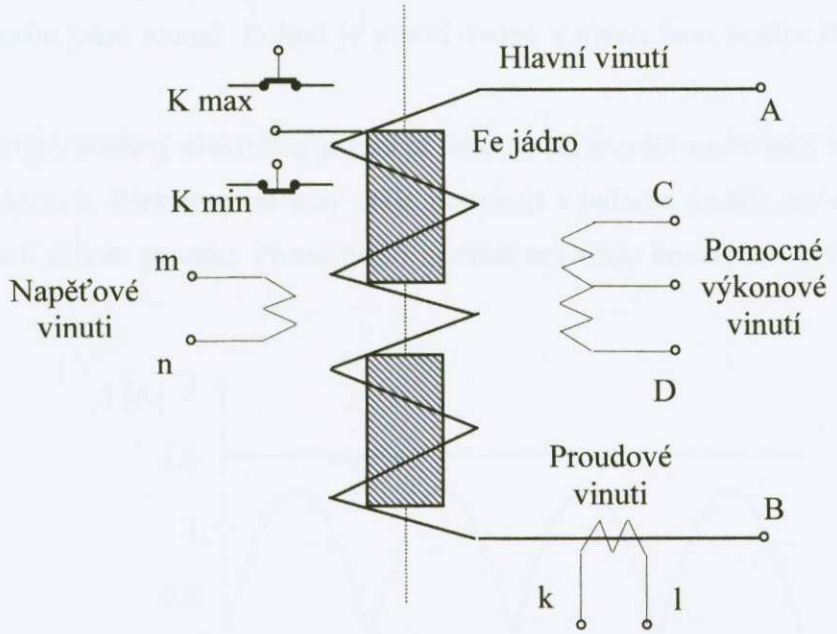


Obr. 1.8 Přímé uzemnění uzlu.

1.5 Provedení zhášecí tlumivky

Konstrukce zhášecí tlumivky [2] je patrná z (obr. 1.9). Tlumivka se zapojuje mezi uzel transformátoru a zem pomocí vývodů A a B. Požadovaná indukčnost je

nastavována pohyblivým železným jádrem. Pohyb jádra je zajištěn elektrickými motory, jejichž rozsah působnosti je omezen koncovými spínači K_{max} a K_{min}. Tlumivka je dále vybavena pomocným měřícím transformátorem proudu (vývody k a l) a napětí (vývody m a n). Tím je zajištěno měření patřičných veličin pro účely regulace. Každá tlumivka navíc obsahuje krátkodobě zatížitelné pomocné výkonové vinutí. Toto vinutí se využívá ke zvyšování činné složky proudu zemního spojení pro vyhledání poruchového vývodu. Zhášecí tlumivky se zpravidla dimenzují na dobu zatížení 30 až 120 minut.



Obr. 1.9 Zhášecí tlumivka.

1.6 Ladění zhášecí tlumivky

Pro docílení maximálních účinků kompenzace poruchového kapacitního proudu v případě jednofázové zemní poruchy je zapotřebí udržovat síť ve vyladěném stavu [2]. Tj. takovém, aby rozdíl mezi nastaveným proudem zhášecí tlumivky (její indukčností) a kapacitním proudem sítě byl do 10 % kapacitního proudu. K vyhledání tohoto vyladěného stavu se využívá kapacitní nesymetrie jednotlivých fází. Díky ní není napětí uzlu transformátoru proti zemi přesně nulové. Závislost tohoto malého napětí na nastavené indukčnosti je charakterizována rezonanční křivkou, jejíž vrchol odpovídá právě vyladěnému stavu. Ladění je možno provádět ručně, případně speciální automatikou např. terminálem TAFT 112®.

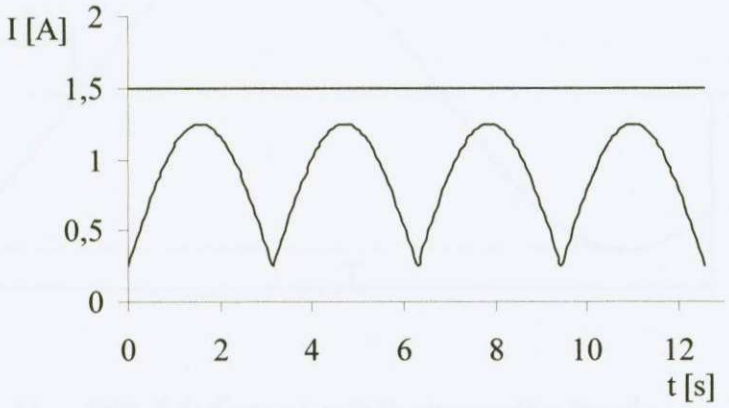
2. Veličiny střídavého proudu

Cílem této kapitole je definovat základní pojmy a velečiny popisující střídavé proudy. Tyto pojmy budou využívány v dalších kapitolách, zejména pak v kapitole 6.

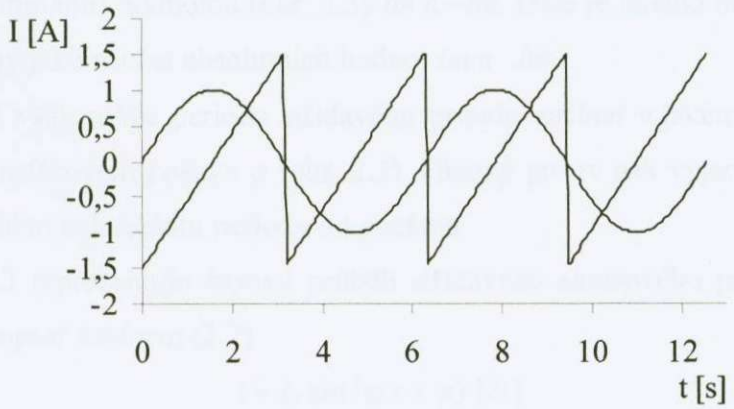
2.1 Stejnosměrný a střídavý proud

Elektrickým proudem rozumíme usměrněný pohyb elektricky nabitých částic (elektronů a iontů) [3]. V případě vodičů jsou těmito částicemi elektrony obsažené ve vodivostním pásu atomů. Pokud je proud veden v plynu jsou těmito částicemi kladné ionty.

Stejnosměrný elektrický proud je takový, při kterém nedochází ke změně směru toku elektronů. Elektrony se tedy stále pohybují v jednom směru, netvrdí se však nic o velikosti tohoto proudu. Proud proto nemusí mít vždy konstantní velikost (obr. 2.1).



Obr. 2.1: Příklad dvou průběhů stejnosměrného proudu.



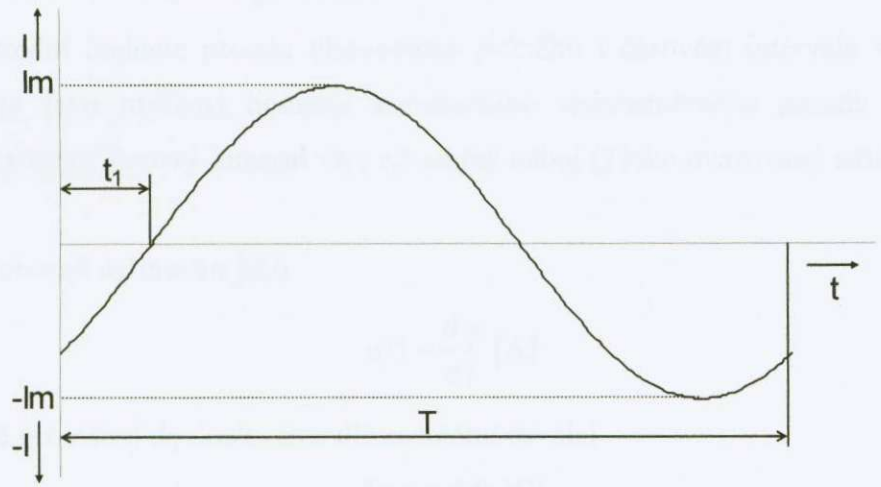
Obr. 1.2: Příklad dvou průběhů střídavého proudu.

Naopak střídavý elektrický proud je takový, při kterém dochází ke změně směru toku elektronů. Elektrony tedy mění (střídají) směr svého toku (obr. 2.2).

2.2 Časový průběh střídavého proudu

Základní charakteristikou střídavého proudu je perioda T [3][4]. Perioda je čas, který uběhne během jednoho kmitu střídavé veličiny (obr. 2.3.). S periodou úzce souvisí frekvence f střídavého proudu. Frekvence určuje počet period za jednu sekundu. Jedná se tedy o převrácenou hodnotu periody.

$$f = \frac{1}{T} \text{ [Hz]} \tag{2.1}$$



Obr. 2.3: Časový průběh sinusového proudu.

Střídavý proud lze dále charakterizovat jeho maximální, respektive zápornou maximální (minimální) hodnotou (obr. 2.3) *Im* a *–Im*. Dále je možno definovat rozkmit střídavé veličiny jako součet absolutních hodnot *Im* a *–Im*.

Obecně však může perioda střídavého proudu začínat v jakémkoli čase $t \neq 0$, pak hovoříme o fázovém posuvu φ (obr. 2.3). Fázový posuv pak vyjadřuje předbíhání, popřípadě zpožďování začátku periody od počátku.

Obr. 2.3 reprezentuje časový průběh střídavého sinusového proudu, který lze matematicky popsat vztahem (2.2)

$$i = I_m \sin(\varpi t + \varphi) \text{ [A]} \tag{2.2}$$

kde i je okamžitá hodnota proudu v čase, Im je maximální hodnota proudu, ω je úhlová frekvence, t je čas a φ je počáteční fázový posun. Úhlovou frekvenci ω lze dále vyjádřit známým vztahem (2.3) [3]

$$\omega = 2\pi f = \frac{2\pi}{T} \text{ [rad s}^{-1}\text{]} \tag{2.3}$$

což vyjadřuje úhlovou rychlost oběhu rotoru v alternátoru. Pro fázový posun φ z (2.2) bude platit [3]

$$\varphi = 2\pi \frac{t_1}{T} \text{ [rad]} \tag{2.4}$$

kde t_1 vyjadřuje čas o který se perioda předbíhá, popřípadě zpožďuje.

2.3 Střední hodnota proudu

Střední hodnota proudu libovolného průběhu v časovém intervalu $<t_1 ; t_2>$ je definována jako myšlená hodnota konstantního stejnosměrného proudu I_{AV}, který přenese za stejný časový interval $<t_1 ; t_2>$ stejný náboj Q jako uvažovaný střídavý proud [3][4].

Proud je obecně definován jako

$$i(t) = \frac{\mathrm{d}q}{\mathrm{d}t} \text{ [A]} \tag{2.5}$$

Po úpravě pro náboj dq dostáváme diferenciální rovnici

$$\mathrm{d}q = i(t)\mathrm{d}t \text{ [C]} \tag{2.6}$$

a integrací v mezích daných časovým intervalem $<t_1 ; t_2>$ platí pro přenesený náboj q

$$Q = \int_{t_1}^{t_2} \mathrm{d}q = \int_{t_1}^{t_2} i(t)\mathrm{d}t \text{ [C]} \tag{2.7}$$

Pro stejnosměrný proud platí $i(t) = I_{AV}$, má tedy stále konstantní hodnotu. Pro náboj přenesený stejnosměrným proudem dostáváme

$$Q_{DC} = I_{AV}(t_2 - t_1) \text{ [C]} \tag{2.8}$$

Při stanovení náboje přeneseného střídavým proudem Q_{AC} vyjdeme ze vztahu (2.7), do kterého dosadíme za $i(t)$ námi vyšetřovaný střídavý proud v časovém intervalu $<t_1 ; t_2>$

$$Q_{AC} = \int_{t_1}^{t_2} i(t)\mathrm{d}t \text{ [C]} \tag{2.9}$$

Z definice víme, že náboj přenesený stejnosměrným proudem má být stejně velký jako náboj přenesený střídavým proudem

$$Q_{DC} = Q_{AC} \tag{2.10}$$

Po dosazení a drobné úpravě dostáváme pro střední hodnotu proudu

$$I_{AV} = \frac{1}{t_2 - t_1} \int_{t_1}^{t_2} i(t)\mathrm{d}t \ [\mathrm{A}] \tag{2.11}$$

Vzhledem k tomu, že střídavé proudy jsou většinou symetrické podle osy x (obr. 2.3) je střední hodnota takovýchto proudů rovna nule. Proto má smysl tuto hodnotu počítat za kratší časový interval, a to za půlperiodu *<0 ; T/2>*. V tomto případě přejde vztah (2.11) do tvaru

$$I_{AV} = \frac{2}{T} \int_{0}^{\frac{T}{2}} i(i)\mathrm{d}t \ [\mathrm{A}] \tag{2.12}$$

Jako příklad lze uvést výpočet střední hodnoty proudu sinusového průběhu. Tento proud je popsán vztahem (2.2). Dosazením (2.2) do (2.12) postupně dostáváme

$$I_{AV} = \frac{2}{T} \int_{0}^{\frac{T}{2}} I_m \sin(\omega t)\mathrm{d}t = \frac{2}{T}\frac{I_m}{\omega}\left[-\cos(\omega t)\right]_0^{\frac{T}{2}} = \frac{2}{T}\frac{I_m T}{2\pi}\left[-\cos\frac{2\pi}{T}t\right]_0^{\frac{T}{2}} = \frac{I_m}{\pi}(1+1) \ [\mathrm{A}]$$

Střední hodnota střídavého sinusového (harmonického) proudu je tedy

$$I_{AV} = \frac{2}{\pi} I_m \cong 0{,}637\, I_m \ [\mathrm{A}] \tag{2.13}$$

2.4 Efektivní hodnota střídavého proudu

Efektivní hodnota střídavého proudu *i(t)* je rovna velikosti stejnosměrného proudu *I*, jenž na činné zátěži o odporu *R* vyvine za dobu jedné periody *T* stejnou tepelnou energii jako střídavý proud *i(t)* [3][4].

Pro výkon platí zcela obecně

$$P = \frac{\mathrm{d}E}{\mathrm{d}t} \ [\mathrm{W}] \tag{2.14}$$

Dále pro elektrický výkon platí

$$P = R I^2 \ [\mathrm{W}] \tag{2.15}$$

Rovnice (2.14) a (2.15) se musí nutně rovnat. Za proud *I* dosadíme náš střídavý elektrický proud *i(t)* a vyjádříme diferenciál elektrické práce d*E*

$$dE = R\, i^2(t)dt \text{ [J]} \tag{2.16}$$

Integrací (2.16) v mezích daných časem jedné periody, konečně dostaneme vztah pro vyjádření elektrické práce, tj. tepelné energie vzniklé průchodem proudu *i(t)* na činné zátěži *R*

$$E = \int_0^T R\, i^2 dt \text{ [J]} \tag{2.17}$$

V případě stejnosměrného proudu *I* bude pro tepelnou energii E_{DC} na činné zátěži *R* platit

$$E_{DC} = R\, I^2 T \text{ [J]} \tag{2.18}$$

Pro tepelnou energii E_{AC} způsobenou průchodem střídavého proudu *i(t)* za dobu jedné periody *T*

$$E_{AC} = R \int_0^T i^2(t)dt \text{ [J]} \tag{2.19}$$

Podle definice víme, že obě tepelné energie se mají rovnat

$$E_{DC} = E_{AC} \tag{2.20}$$

Po dosazení a drobné úpravě dostáváme pro efektivní hodnotu proudu

$$I = \sqrt{\frac{1}{T}\int_0^T i^2(t)dt} \text{ [A]} \tag{2.21}$$

Jako příklad lze uvést výpočet efektivní hodnoty proudu sinusového průběhu. Tento proud lze popsat vztahem (2.2). Dosazením (2.2) do (2.21) postupně dostáváme

$$I = \sqrt{\frac{1}{T}\int_0^T I_m^2 \sin^2(\varpi t)dt} = I_m \sqrt{\frac{1}{T}\left(\frac{1}{2}[t]_0^T - \frac{1}{2\varpi}[\sin(\varpi t)\cos(\varpi t)]_0^T\right)} =$$

$$= I_m \sqrt{\frac{1}{T}\left(\frac{T}{2} - 0\right)} = I_m \sqrt{\frac{1}{2}} \text{ [A]}$$

Efektivní hodnota střídavého sinusového (harmonického) proudu je tedy

$$I = I_m \frac{\sqrt{2}}{2} \cong 0{,}707\, I_m \text{ [A]} \tag{2.22}$$

2.5 Střední a efektivní hodnota střídavého napětí

Analogicky jako u časově proměnného proudu *i(t)* lze definovat i střední a efektivní hodnoty napětí [3]. Můžeme užít i následující úvahu: Jaký úbytek napětí

vyvolá střídavý proud *i(t)* na jednotkovém rezistoru *R*. Z ohmova zákona víme, že napětí *U* je přímo úměrné elektrickému odporu *R* a procházejícímu proudu *I*

$$U = R\,I \text{ [V]} \tag{2.23}$$

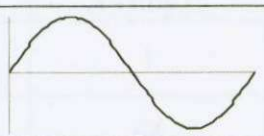
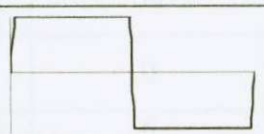
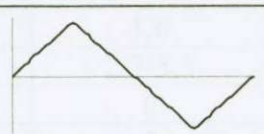
Pak vztah (2.11) pro výpočet střední hodnoty proudu přejde na tvar pro výpočet střední hodnoty napětí

$$U_{AV} = \frac{1}{t_2 - t_1}\int_{t_1}^{t_2} u(t)\mathrm{d}t \text{ [V]} \tag{2.24}$$

A vztah (2.21) pro výpočet efektivní hodnoty proudu přeje na tvar pro výpočet efektivní hodnoty napětí

$$U = \sqrt{\frac{1}{T}\int_0^T u^2(t)\mathrm{d}t} \text{ [V]} \tag{2.25}$$

Na závěr jsou v (tab. 2.1) uvedeny vztahy pro výpočet střední a efektivní hodnoty napětí tří základních průběhů napětí.

Průběh napětí	Střední hodnota napětí U_{AV}	Efektivní hodnota napětí U
	$\frac{2}{\pi}U_M$	$\frac{\sqrt{2}}{2}U_M$
	U_M	U_M
	$\frac{1}{2}U_M$	$\frac{\sqrt{3}}{3}U_M$

Tab. 2.1: Porovnání středních a efektivních hodnot napětí několika signálů.

3. Mikrokontroléry ATMEL® AVR®

Tato kapitola se zabývá především AVR® jádrem mikrokontrolérů ATMEL®, které je takřka ve všech typech mikrokontrolérů této rodiny totožné. Dále jsou zde nastíněny možnosti mikrokontroléru ATmega 128 a základní informace o jeho programování.

3.1 Obecné vlastnosti

Korporace ATMEL® založená v roce 1984 patří k celosvětové špičce v polovodičové technice [5]. U nás je známa především díky svým polovodičový pamětem a jednočipovým mikrokontrolérům.

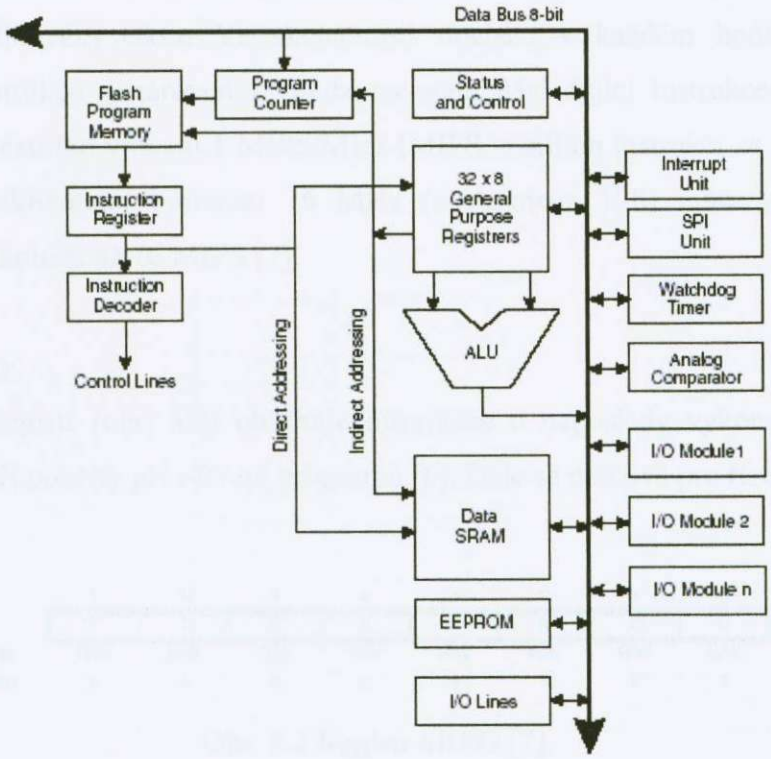
Následující tabulka (tab. 3.1) ukazuje vlastnosti vybraných čtyř zástupců těchto mikrokontrolérů včetně ceny k listopadu 2005. Jednotlivými vlastnostmi se budeme dále zabývat v následujících kapitolách. Z uvedených parametrů jasně vyplývá, že tyto mikrokontroléry jsou značně univerzální, výkonné a cenově dostupné, což je činí velice oblíbenými.

	ATtiny15	ATtiny2313	ATmega8535	ATmega128
Kapacita FLASH [kB]	1	2	8	128
Kapacita EEPROM [B]	64	128	512	4096
Kapacita SRAM [B]	0	128	512	4096
Max. I/O pins	6	18	32	53
F_{max} [MHz]	1,6	20	16	16
Vcc [V]	2,7-5,5	1,8-5,5	2,7-5,5	2,7-5,5
16-bit. č/č	0	1	1	2
8-bit. č/č	2	1	2	2
SPI	0	0	1	1
UART	0	1	1	2
TWI	0	1	1	1
ISP	1	1	1	1
A/D převodník	4	0	0	8
Analogový komparátor	1	1	0	1
Watchdog	1	1	1	1
Integrovaný RC oscilátor	1	1	1	1
Pouzdro	PDIP8	PDIP20	PDIP40	TQFP 64
Cena	45 Kč	50 Kč	86 Kč	280 Kč

Tab. 3.1 Srovnání parametrů několika vybraných mikrokontrolérů [5].

3.2 Jádro AVR

Toto jádro (obr. 3.1) bylo navrženo pro vysoký výpočetní výkon s ohledem na programování vyššími programovacími jazyky jako je C, Pascal nebo Basic [5]. Tohoto úkolu bylo dosaženo především díky použití harvardské architektury, redukované instrukční sady (RISC) a rychle přístupného 32 bytového registrového pole [6].



Obr. 3.1 Obecné AVR® jádro [7].

Harvardská architektura vychází z myšlenky oddělené programové a datové paměti. Nevýhodou tohoto oddělení je větší technologická složitost vlastního čipu, jelikož jsou potřeba dvě sběrnice: programová a datová. Naopak výhodou dvou oddělených sběrnic je rychlejší zpracování programu, neboť se nemusí data i program dělit o jedinou sběrnici. Další výhodou je, že sběrnice mohou mít různou délku. Osmibitová šířka programové sběrnice (osmibitová instrukce) by byla nedostatečná pro zakódování celého instrukčního souboru, proto se využívá šestnáctibitová programová sběrnice (šestnáctibitové instrukce). Z toho také vyplývá jednotná délka všech instrukcí a to 16 bitů.

Redukovaný instrukční soubor (RISC) znamená, že instrukční soubor obsahuje pouze jednodušší instrukce, které lze vykonat rychleji. Ukazuje se, že je rychlejší

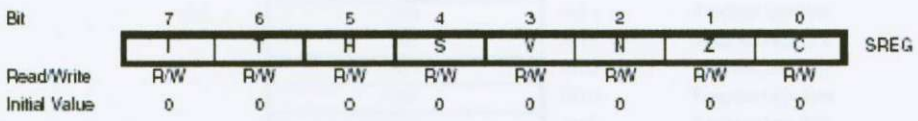
vykonat více jednoduchých instrukcí než jednu komplikovanou. Použitím redukovaného instrukčního souboru se velmi zjednoduší dekodér instrukce, což samozřejmě přinese zrychlení vlastního dekódování.

Registrové pole obsahuje 32 všeobecně použitelných registrů. V postatě se jedná o 32 bytový akumulátor. Přístup do tohoto registrového pole je možno provést v jediném hodinovém taktu.

Díky těmto třem hlavním výhodám bylo dosaženo načtení a vykonání instrukce v jediném hodinovém taktu. Ve skutečnosti dochází v každém hodinovém taktu k vykonání instrukce a zároveň k předzpracování následující instrukce. Tímto bylo dosaženo výpočetního výkonu 1 MIPS/MHz (MIPS = milion instrukcí za sekundu). Při maximálním taktovacím kmitočtu 16 MHz (u ATmega 128) můžeme disponovat výpočetním výkonem až 16 MIPS [7].

Stavový registr

Tento registr (obr. 3.2) obsahuje informace o naposledy vykonané instrukci, které mohou být použity při větvení programu [6]. Dále se používá pro řízení přerušení.



Obr. 3.2 Registr SREG [7].

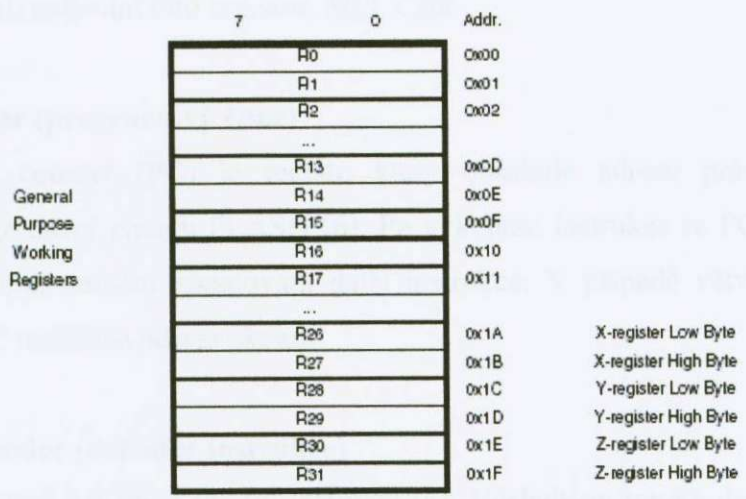
Význam jednotlivých bitů:

- C - příznak přetečení. Indikuje přetečení po aritmetické nebo logické operaci.
- Z - příznak nulového výsledku. Indikuje nulový výsledek po aritmetické nebo logické operaci.
- N - příznak negativního výsledku. Indikuje záporný výsledek po aritmetické nebo logické operaci.
- V - příznak přeplnění čísla v druhém doplňku. Podporuje aritmetiku druhého doplňku.
- S - znaménkový bit. Indikuje znaménko čísla v druhém doplňku.
- H - pomocný příznak přetečení. Indikuje přenos mezi dolní a horní polovinou bytu (BCD aritmetika).

- T - kopírovací bit. Instrukce *BLD* a *BST* používají bit T jako zdrojový nebo cílový bit.
- I - globální povolení přerušení. Pro záchyt přerušení musí být tento bit nastaven (I = 1). Po vstupu do obslužné rutiny přerušení je tento bit automaticky nulován a tím je znemožněn záchyt dalšího přerušení.

Pole registrů

Pole registrů (obr. 3.3) je složeno ze 32 všeobecně použitelných osmibitových registrů [6]. Nad registrovým polem jsou definovány všechny dostupné aritmetické a logické operace, které probíhají v jediném hodinovém cyklu.



Obr. 3.3 Pole registrů [7].

Výjimku tvoří pouze instrukce pracující s konstantami (*SBCI, SUBI, CPI, ANDI, ORI, LDI, CPI*). Tyto instrukce pracují pouze s horní polovinou registrového pole (R16 – R31). Posledních 6 bytů registrového pole může být použito jako ukazatele (pointer) X, Y a Z. Těchto ukazatelů se využívá pro nepřímou adresaci.

Aritmetickologická jednotka (ALU)

Tato jednotka umožňuje vykonávání následujících operací mezi dvěma registry nebo mezi registrem a konstantou [6]:

- Aritmetické operace
  - Sčítání: *ADD, ADC*
  - Odčítání: *SUB, SUBI, SBC, SBCI*

- Inkrementace a dekrementace: *INC, DEC*
- Druhý doplněk: *NEG*

- Logické operace
  - Logický součet: *AND, ANDI*
  - Logický součet: *OR, ORI*
  - Výlučný logický součet: *EOR*
  - Negace: *COM*
  - Nastavení a nulování: *SER, CLR*
- Bitové operace
  - Logický posuv a rotace: *LSL, LSR, ROL, ROR*
  - Přehození horního a dolního půlbytu: *SWAP*
  - Nastavení, nulování bitů registru: *SBR, CBR*

Program counter (programový čítač)

Program counter (PC) je registr, který obsahuje adresu právě prováděné instrukce v programové paměti FLASH [6]. Po vykonání instrukce se PC automaticky inkrementuje, čímž umožní zpracování další instrukce. V případě větvení programu (skoků) je do PC umístěna adresa skoku.

Instruction Decoder (dekodér instrukce)

Tento obvod dekóduje vlastní instrukci [6]. Výsledkem tohoto dekódování jsou elektrické impulsy na řídících linkách, které zabezpečí požadovanou funkci. Například instrukce *ADD R0, R1* má za úkol sečíst registry *R0* a *R1*, přičemž součet uloží do registru *R1*. Dekodér instrukce vygeneruje patřičné signály, které připojí na vstupy ALU registry *R0* a *R1* a zároveň nasměruje výstup ALU přes datovou sběrnici do registru *R0*.

Jednotka přerušení

Přerušení je reakcí na nějakou událost, která může být vyvolána např. čítačem/časovačem, příjmem bytu sériovým kanálem, změnou stavu log. vstupů atd. Přerušení je tedy pevně spjato s periferiemi mikrokontroléru [6]. V případě záchytu požadavku pře-rušení dojde ke skoku v programu na vektor přerušení (adresu) odpovídající dané události. Současně se uloží návratová adresa (obsah PC) do zásobníku. Následně je vykonána obslužná rutinka přerušení. Na závěr se program vrátí

na původní pozici obnovením obsahu PC ze zásobníku. Počet vektorů přerušení je závislý na konkrétně použitém typu, např. ATmega 128 disponuje 35 vektory [7].

Na datovou sběrnici je dále připojeno množství periferních obvodů, jejichž počet je závislý na konkrétním typu mikrokontroléru. Mezi základní periférie, které jsou obsaženy na každém typu mikrokontroléru patří [5]:

- 8 bitový čítač/časovač
- Watchdog
- EEPROM
- Vstupně-výstupní brána

Většina mikrokontrolérů však nezůstává u základní výbavy a jsou doplněny mnoha dalšími periferiemi jako jsou:

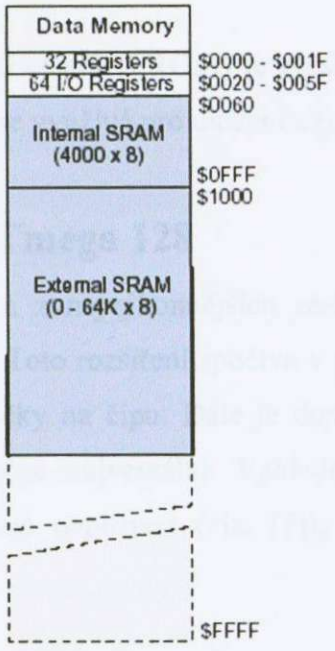
- Datová SRAM
- SPI jednotka
- UART
- 16 bitový čítač/časovač
- Analogový komparátor
- A/D převodník

3.3 Paměťový prostor

Jak již bylo řečeno mikrokontroléry ATMEL AVR se vyznačují odděleným datovým a programovým prostorem. Navíc obsahují paměť typu EEPROM.

Datová paměť

Datová paměť je tvořena statickými buňkami SRAM [6]. Struktura paměti je zřejmá z obrázku (obr. 3.4). Prvních 32 bytů zaujímá pole registrů. Dalších 64 bytů je využito vstupně/výstupní registry. Tyto registry slouží ke konfiguraci vlastního mikrokontroléru a jeho periférií. Na konkrétním mikrokontroléru nemusejí být všechny implementovány, nicméně tento prostor zaujímá vždy 64 bytů. Následující část tvoří libovolně použitelná paměť RAM. Velikost této paměti se liší podle použitého typu mikrokontroléru, dokonce může být i nulová u ATtiny 15 [5]. Poslední část paměti je vyhrazena externímu datovému prostoru. Ten může dosahovat velikosti až 64 kB [5].



Obr. 3.4 Struktura datové paměti [7].

Zásobník

Zásobník je část paměti, která slouží k ukládání a pozdějšímu obnovování obsahů registrů [6]. Dále se využívá při volání podprogramů a obslužných rutin přerušení (ukládá se do něj návratová adresa). Chová se jako paměť typu LIFO (poslední dovnitř, první ven).

Ukazatel zásobníku SP (Stack Pointer)

Je registr, jehož obsahem je adresa vrcholu zásobníku v RAM [6]. Při vložení bytu do zásobníku se SP snižuje, při vyjmutí se SP zvyšuje. Zásobník tedy roste směrem dolů. Obvykle se proto vrchol zásobníku umisťuje na konec datové paměti RAM.

Programová paměť

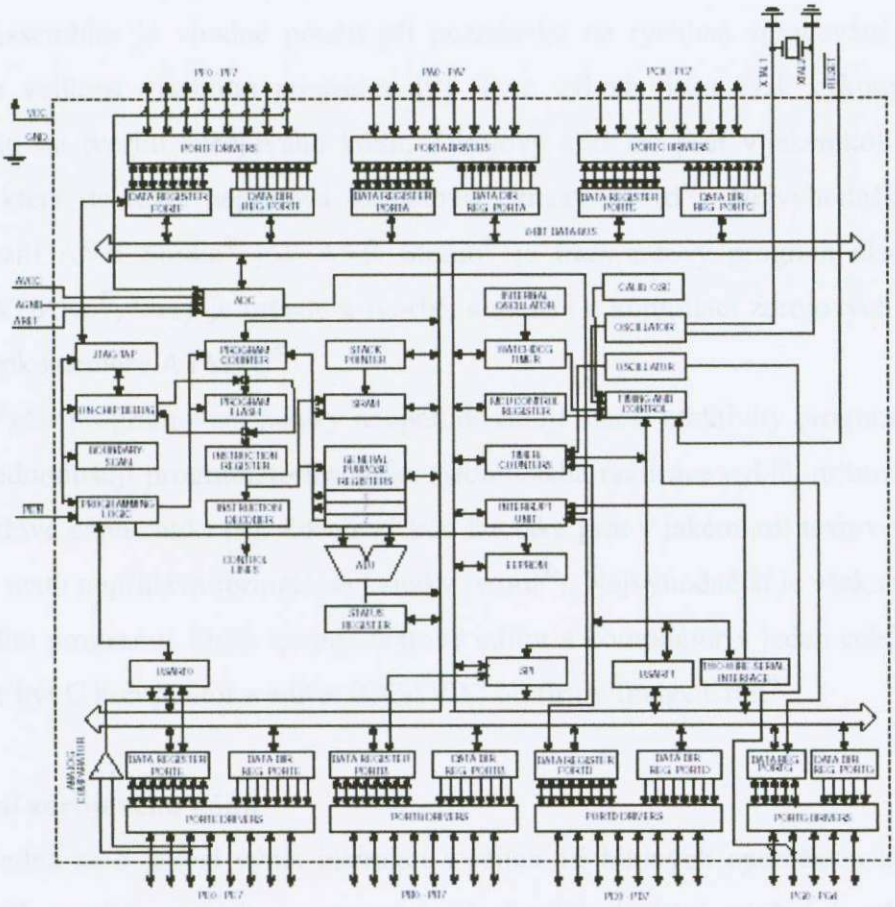
Jedná se o paměť typu FLASH adresovanou po dvojbytech [6]. Kapacita této paměti je samozřejmě závislá na použitém mikrokontroléru (tab 3.1). Její nesmírnou výhodou je možnost ji programovat přímo v aplikaci pomocí sériového rozhraní SPI [6] nebo pomocí JTAGu [5].

EEPROM

EEPROM (Electrically Erasable PROM) je elektricky programovatelná nedestruktivní paměť [6]. Nejčastěji se využívá pro uložení uživatelských dat.

3.4 Mikrokontrolér ATmega 128

ATmega 128 je jedním z nejvýkonnějších zástupců mikrokntolérů ATMEL s rozšířeným AVR jádrem [7]. Toto rozšíření spočívá v pestřejší instrukční sadě a dále v umístění jednobytové násobičky na čipu. Dále je doplněn množstvím periférií, což činí tento mikrokontrolér značně univerzální. Vzhledem k rozsahu této práce není možné mikrokontrolér podrobně popisovat (viz. [7]), proto je zde uvedeno pouze blokové schéma (obr. 3.5).



Obr. 3.5 Blokové schéma ATmega 128 [7].

3.5 Programování mikrokontrolérů

Vlastní programování by se dalo rozdělit na několik částí [6][9]:

- Vytvoření zdrojového kódu
- Přeložení zdrojového kódu (kompilace)
- Odladění programového kódu
- Nahrání zkompilovaného kódu do mikrokontroléru

Vytvoření zdrojového kódu

Pro psaní zdrojového kódu je důležitá volba programovacího jazyka [9]. Můžeme použít základní jazyk procesoru, kterým je assembler (jazyk symbolických adres), nebo se přikloníme k vyššímu programovacímu jazyku jako je C, Pascal nebo Basic. Každá z těchto variant má své výhody i nevýhody.

Assembler je vhodné použít při požadavku na rychlost zpracování programu a nízkou velikost vlastního programu [9]. Tyto výhody jsou však vykoupeny větší náročností na tvorbu zdrojového kódu. Zdrojový kód lze psát v jakémkoli textovém editoru, který do textu nepřidává formátovací znaky (word®). Nejvýhodnější se však jeví použití AVR Studia® [5]. AVR Studio® je freewareový program distribuovaný firmou ATMEL®, který je určený k tvorbě, simulaci a kompilaci zdrojových programů pro mikrokontroléry ATMEL®.

Vyšší programovací jazyky naopak dosahují menší efektivity programu, ale výrazně zjednodušují programátorům práci (jednoduchá realizace cyklů, aritmetika v plovoucí řádové čárce, atd.) [9]. Zdrojový kód lze také psát v jakémkoli textovém editoru, který do textu nepřidává formátovací znaky (word®). Nejvýhodnější je však také použití speciálního programu, který spojuje textový editor a kompilátor v jeden celek. Příkladem může být C kompilátor a editor ICCAVR® od firmy Image Craft®.

Přeložení zdrojového kódu

Jedná se o jakýsi výpis instrukcí v jejich 16 bitových operačních kódech [6]. V případě použití vyšších programovacích jazyků dochází napřed k překladu do assembleru a následně k „překódování". Produktem tohoto překladu je soubor typu *.hex, který je již připraven k nahrání do paměti mikrokontroléru.

Odladění zdrojového kódu

Odladěním zdrojového kódu ověříme jeho správnost a otestujeme jeho chování v nestandardních situacích [6]. Provádí se buď čistě softwarově v AVR® Simulátoru, který je součástí AVR Studia®, nebo pomocí speciálního emulátoru či vývojového kitu. Zajímavé je také použití JTAGu [5], což umožňuje snadné odladění programu přímo v aplikaci. Navíc ho lze použít i pro nahrání zkompilovaného kódu do mikroprocesoru.

Nahrání zkompilovaného kódu do mikrokontroléru

Nahrání zkompilovaného kódu do mikrokontroléru je posledním krokem, který je třeba provést [6]. Všechny mikrokontroléry disponují několika možnostmi nahrání programu do FLASH paměti. První možností je paralelní programátor. Tato varianta vyžaduje speciální typ programátoru, do kterého se vkládá vlastní mikrokontrolér. Znamená to tedy vždy mikrokontrolér vyjmout z aplikace, naprogramovat ho a vrátit zpět do aplikace. Tento postup není příliš vhodný pro vývoj aplikací, ale na druhou stranu umožňuje naprogramovat i paměťové zámky [6]. Další možností je sériové programování, které se provádí přes SPI jednotku (sériová kanál) [5]. S určitými kompromisy lze sériového programování využívat i přímo v aplikaci. Poslední a nejlepší možností je programování pomocí JTAGu [5], který je primárně určen pro činnost přímo v aplikaci a navíc je plně podporován AVR Studiem®.

4. Hardwarová koncepce terminálu

Cílem této kapitoly není vyčerpávajícím způsobem popsat zapojení jednotlivých částí terminálu, nýbrž získání základní představy o funkci zařízení jako celku [2]. Konkrétní zapojení zde uvedeno není a ani by nebylo správné ho zde uvést, protože je výsledkem několikaměsíčního vývoje firmy EGC. Proto se v této kapitole setkáme nanejvýš se zjednodušenými blokovými schématy znázorňujícími danou funkci.

4.1 Základní požadavky

Terminál Automatizovaných Funkcí Transformátoru TAFT 112 byl vyvinut firmou EGC (EnerGoConsult ČB, s.r.o.), především Ing. Václavem Králem za účelem co největší univerzality a rychlosti [2]. Díky této snaze se stal TAFT univerzálním hardwarem pro takřka jakékoli řídící (regulační) zařízení v elektrotechnice. Po nahrání konkrétního softwaru se hardware promění v dané zařízení, ať již automatizovaný regulátor zhášecí tlumivky transformátoru nebo cokoliv jiného.

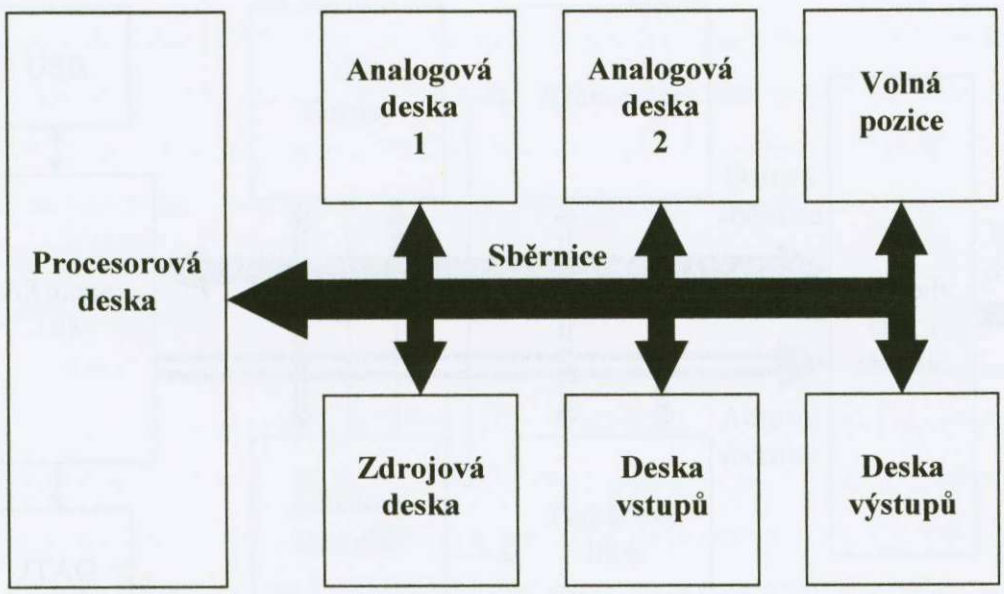
Díky použití moderního mikrokontroléru ATMEL® ATmega 128 (viz. kapitola 3), který dosahuje výpočetního výkonu až 16 MIPS (MIPS = milion instrukcí za sekundu) a správné hardwarové stavby, jež umožňuje přístup do externího datového prostoru (obsahuje i AD převodníky) v jediné instrukci, bylo docíleno rychlého zpracování měřených dat. Díky této rychlosti bylo možno implementovat do obslužného softwaru i poměrně výpočtově náročné operace jako je Fourierova harmonická analýza, aplikovatelná na každou periodu měřeného signálu.

4.2 Principiální řešení terminálu

Vlastní terminál je řešen sběrnicovou strukturou, kde veškeré periferie jsou umístěny v externím datovém prostoru procesoru na samostatných deskách (obr. 4.1) [2]. Tato architektura velice připomíná sestavy osobních počítačů (PC). Kde procesorová deska představuje základní desku PC a ostatní periferní desky představují zásuvné karty do sběrnic PC.

Funkce terminálu by se dala rozdělit do několika dílčích částí. Předně je to vlastní práce procesoru. Sběrnice je v této fázi nečinná, periférie pracují v zadaném režimu a procesor provádí výpočty. Další činností je zápis nebo čtení periferních desek. Procesor dospěl v této části výpočtu do stavu, kdy potřebuje např. změnit stav relátek na

výstupní desce. Všechna periferní zařízení jsou umístěna v jeho externím datovém prostoru, musí tedy vykonat instrukci pro zápis do tohoto prostoru. V průběhu vykonávání tohoto příkazu se automaticky vygeneruje daná adresa, vybaví se příslušný signál povolující přístup k desce a zároveň se „otevře" datová část sběrnice pro zápis na danou desku. Posledním druhem činnosti je generování přerušení externí událostí. Tato situace nastává např. při stisku tlačítka klávesnice. V tom případě se aktivuje signál přerušení, který přeruší činnost procesoru. Procesor pak již výše popsaným způsobem tj. zápisem nebo čtením zareaguje na danou událost a pokračuje v normální činnosti.



Obr. 4.1 Sběrnicová struktura terminálu [2].

4.3 Sběrnice

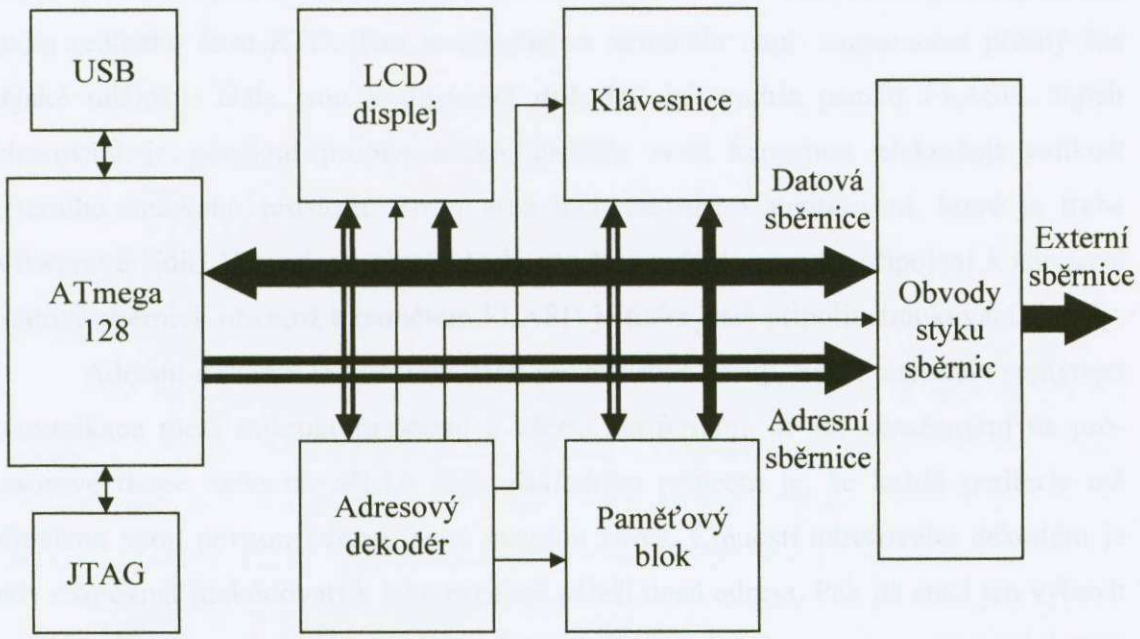
Sběrnice terminálu tvoří jakousi komunikační dálnici mezi procesorem a jednotlivými periferiemi [2]. Pro přesnost je třeba ještě říci, že v terminálu existují celkem dvě sběrnice vnitřní a externí. Vnitřní sběrnice je obsažena na procesorové desce a budeme se jí ještě zabývat v kapitole 5.4. Naopak externí sběrnice je právě ta, která propojuje procesorovou desku (vnitřní sběrnici) se všemi periferiemi (obr. 4.1).

Mezi hlavní části této sběrnice patří datová část obsahující osm bitů, tedy jeden byte. Její funkce je zřejmá, zápis a čtení příslušných periferií. Z principu funkce musí zajišťovat obousměrnou komunikaci. Další částí externí sběrnice je adresní dekodér, jehož úkolem je dekódovat z adresní části vnitřní sběrnice signál pro povolení přístupu

k dané periferní desce. Dále následuje skupina řídících a pomocných signálů jako jsou reset, sériová komunikace, přerušení a pomocné signály vzorkování viz. kapitola 5.5. Poslední částí této sběrnice je napájení +12V a +5V.

4.4 Procesorová deska

Procesorová deska zastupuje nejdůležitější část terminálu [2]. Zajišťuje správnou funkci jednotlivých částí zařízení a zároveň koordinuje jejich vzájemnou komunikaci.



Obr. 4.2 Blokové schéma procesorové desky [2].

Hlavní součásti této desky (obr. 4.2) je výkonný řídící mikrokontrolér ATMEL® ATmega 128 (viz. kapitola 3). Dalo by se říci, že zastupuje funkci mozku, který celý terminál oživuje. Procesor pracuje v režimu s externí datovým prostorem, ve kterém je umístěna většina externích periferií. Toto uspořádání předpokládá existenci dvou oddělených vnitřních sběrnic procesorové desky, a to sice datové a adresové. Obousměrná datová sběrnice je tvořena 8 bity (1 byte) a propojuje patřičné části této desky. Druhá adresní sběrnice je tvořena již 2 byty, což umožňuje adresovat až 64 kB. Vygenerovaná adresa určuje s jakým zařízením se bude komunikovat.

Procesor je dále vybaven vývojový rozhraním JTAG (viz. kapitola 3), díky němuž je možno provádět pohodlné ladění a nahrávání programu přímo v aplikaci. Dále

se na procesorové desce vyskytuje galvanicky oddělený převodník sériového kanálu na USB. Sériový kanál je integrován již v použitém řídícím mikrokontroléru jako jedna z jeho periferií. Tento převodník je tvořen specializovaným integrovaným obvodem, díky kterému je obvodové řešení velice jednoduché. Terminál dále obsahuje na zdrojové desce (viz. kapitola 5.8) galvanicky oddělený převodník napěťových úrovní 0V a +5V sériového kanálu (tyto úrovně jsou dány mikrokontrolérem) na úrovně odpovídající standartu RS232. Tím vzniká dvojí možnost připojení terminálu ke konfiguračnímu počítači, buď pomocí portu USB nebo pomocí sériové linky COM.

Paměťový blok obsahuje externí datovou paměť SRAM a dále dvě paměti typu FLASH. Datová paměť SRAM o velikosti 32 kB obsahuje navíc zaintegrovaný obvod hodin reálného času RTC. Tím je umožněno terminálu např. zaznamenat přesný čas nějaké události. Dále jsou k dispozici dvě 512 kB rychlé paměti FLASH. Jejich adresování je poněkud problematické, protože svou kapacitou překračují velikost externího datového prostoru. Proto je u nich zavedeno stránkování, které je třeba softwarově řídit. Vlastní zapojení všech pamětí spočívá pouze v připojení k adresové i datové sběrnici, přičemž k pamětem FLASH je třeba ještě připojit stránkovací signály.

Adresní dekodér je klíčovou částí procesorové desky, neboť zajišťuje správnost komunikace mezi mikrokontrolérem a všemi periferiemi, ať již obsaženými na procesorové desce nebo na nějaké jiné. Základním princem je, že každá periferie má přidělenu svou pevnou adresu, resp. skupinu adres. Činností adresového dekodéru je tedy rozpoznat (dekódovat) k jaké periferii náleží daná adresa. Pak již stačí jen vybavit patřičný signál povolující komunikaci dané periferie a lze danou adresu pomocí datové sběrnice číst či zapisovat. Příkladem může být zápis na LCD displej. Procesor v tomto případě vygeneruje danou adresu na adresové sběrnici, adresní dekodér ji dekóduje. Rozpozná, že daná adresa odpovídá LCD displeji, vyšle patřičný signál povolující komunikaci s ním. Na závěr proběhne patřičná datová výměna prostřednictvím datové sběrnice. Vzhledem k tomu, že dekodér plní funkci kombinační logiky, tak je k jeho realizaci použito programovatelné hradlové pole. Diky tomu se poměrně složitý dekodér vměstná do jediného integrovaného obvodu, který pak již stačí jenom připojit k adresové sběrnici a patřičným řídícím signálům.

Rozhraní mezi procesorovou deskou a dalšími deskami tvoří obvody styku sběrnic. Úkolem těchto obvodů je oddělit vnitřní a vnější sběrnice, tak aby externí sběrnice byla v činnosti pouze pokut je to nutné. Datová sběrnice je propojena s externí sběrnicí v plném rozsahu, nýbrž adresní sběrnice pouze v polovičním rozsahu tj. pouze

8 bitů. Prakticky se jedná pouze budiče sběrnic a propojení patřičných řídících signálů. Mezi tyto signály patří reset, sériová komunikace, přerušení, signály vzorkování a řízení sběrnice.

Vizuální rozhraní terminálu je vytvořeno pomocí inteligentního grafického LCD displeje o rozlišení 128*128 bodů. Tento displej je využíván výhradně v textovém režimu, umožňující zobrazení 22*16 znaků. LCD displej obsahuje integrovaný mikrořadič, díky němuž se obsluha značně zjednoduší (obsahuje např. definice znakových sad). Obvodově je zapojen velice jednoduše. Pouze je připojen k datové sběrnici a patřičným řídícím signálům, které zabezpečují komunikaci.

Styk obsluhy s terminálem je umožněn pomocí 8 tlačítkové membránové klávesnice. Tlačítka jsou připojena proti zemi na vstup střadače, který je dále napojen na datovou sběrnici. Dále jsou připojena na osmi vstupové hradlo NAND, které v případě stisku kteréhokoliv z nich generuje signál přerušení. V nastalém stavu přerušení si pak mikrokontrolér přečte stav střadače a pokračuje v předešlé činnosti.

4.5 Analogová deska

Obě analogové desky jsou klíčovou částí terminálu, neboť zprostředkovávají analogově digitální převod měřených napětí [2]. První deska obsahuje 3 střídavé napěťové vstupy a jeden proudový, druhá deska obsahuje také 3 střídavé napěťové vstupy a navíc jeden stejnosměrný. Samozřejmostí musí být galvanické oddělení všech vstupů.

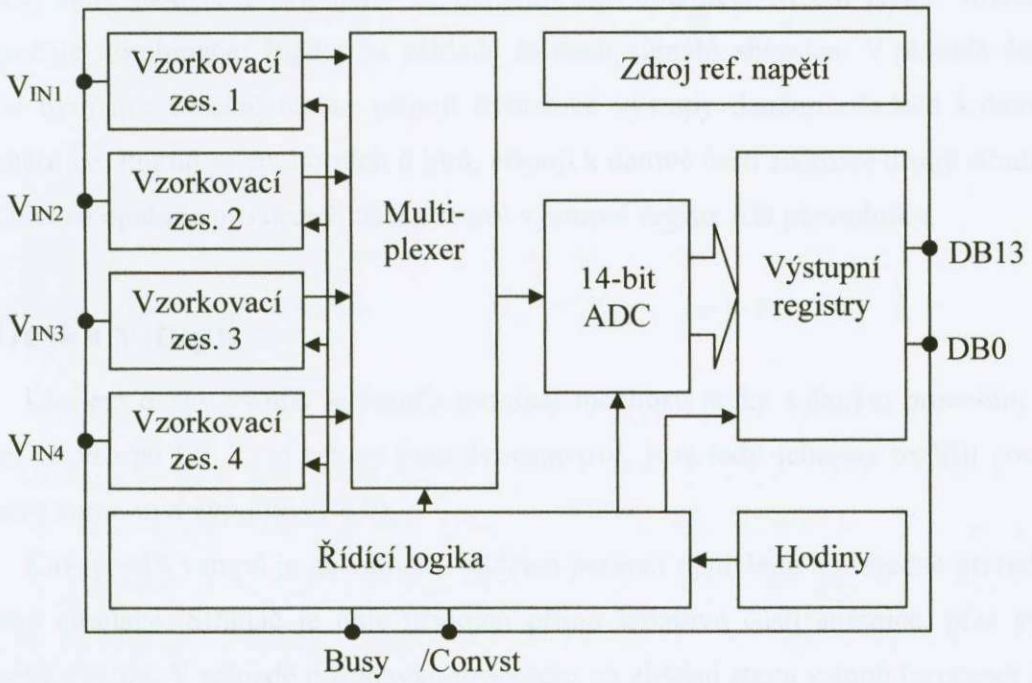
Vstupní obvody předcházející AD převodníku jsou identické pro všechny kanály, tedy s výjimkou proudového a stejnosměrného vstupu. Tvoří je vstupní napěťový transformátor, který tak plní funkci galvanického oddělení a zároveň snižuje velikost vstupního napětí. Následuje napěťový zesilovač s digitálně nastavitelným zesílením, což umožňuje volbu několika měřících rozsahů. Dáje je patřičně upravený analogový signál přiveden na vstup AD převodníku. V případě proudového vstupu je kanál zapojen identicky, pouze je místo napěťového transformátoru použit transformátor proudový. I pro stejnosměrný vstup je zapojení kanálu stejné, místo vstupního napěťového transformátoru se zde však nachází integrovaný izolační zesilovač.

Jako AD převodník je zde použit obvod firmy Analog Devices® AD7865 (obr. 4.3) [11]. Jedná se o rychlý 4 kanálový 14 bitový AD převodník. Navíc je schopen odebrat všechny vzorky naráz v jediném okamžiku. Díky této vlastnosti je možno ze

získaných dat určit např. fázový posuv mezi měřenými signály. Základní parametry tohoto obvodu jsou patrné z tab. 4.1.

Parametr	Hodnota	Jednotka
Rozlišení	14	Bit
Vzorkovací rychlost	100	kSPs
Doba převodu	4 * 2,4	ms
INL	±2	LSB
DNL	±1	LSB

Tab. 4.1 Základní parametry obvodu AD7865 [11].



Obr. 4.3 Zjednodušené blokové schéma obvodu AD7865 [11].

Z blokového schématu na (obr. 4.3) je patrné, že pro řízení celého AD převodníku jsou použity pouze dva piny. Ve skutečnosti je jich více, ale tyto jsou klíčové v daném módu. Jedná se o výstupní signál Busy indikující průběh AD převodu a o vstupní signál /Convst. Pokut AD převodník zaznamená vzestupnou hranu na pinu /Convst, dojde k zahájení AD převodu. Následně dostanou vzorkovací zesilovače od řídící logiky impuls pro odebrání vzorků. Toto vzorkování se provede naráz v jediném okamžiku. Dále řídící logika nasměruje výstup prvního vzorkovacího zesilovače přes analogový multiplexer na vstup rychlého 14 bitového aproximačního AD převodníku. Z něj se po převodu 14 bitové slovo uloží do prvního ze čtyř výstupních registrů. Dále

se cyklus opakuje pro druhý, třetí a čtvrtý kanál. Probíhající převod je po celou dobu indikován vysokou logickou úrovní na pinu Busy.

Po dokončení AD převodu je možné vyčíst data z výstupu AD převodníku. Datová sběrnice je však 8 bitová a výstup AD převodníku je 14 bitový, proto se čtení provádí dvakrát. Napřed se čte dolní byte a následně horních 6 bitů. Výstupní registry v AD převodníku jsou čtyři a každý se musí číst na dvakrát, celkem se tedy musí pro přečtení celého AD převodu číst osmkrát, přičemž po přečtení prvního 14 bitového slova se čte druhé atd.

Hardwarově je toto vyřešeno tak, že výstup AD převodníku je přiveden na dva střadače, které jsou dále připojeny na datovou část sběrnice. Řízení těchto střadačů zabezpečuje kombinační logika na základě řídících signálů sběrnice. V případě čtení dolního bytu kombinační logika připojí třístavové výstupy daného střadače k datové části sběrnice. Pokud se čte horních 6 bitů, připojí k datové části sběrnice druhý střadač. Toto čtení se opakuje pro druhý, třetí a čtvrtý výstupní registr AD převodníku.

4.6 Deska vstupů

Úkolem desky vstupů je opatřit terminál možností styku s daným procesem za pomoci 16 vstupů [2]. Tyto vstupy jsou dvoustavové, jsou tedy schopny rozlišit pouze dva stavy zapnuto a vypnuto (1 a 0).

Každý z 16 vstupů je galvanicky oddělen pomocí optočlenu a následně přiveden na vstup střadače. Střadač je dále připojen přímo k datové části sběrnice, přes svůj třístavový výstup. V případě požadavku procesoru na zjištění stavu vstupů (procesor čte z externího datového prostoru) dojde k uvolnění výstupů střadače a tím k vybavení požadovaných dat na sběrnici.

Jistou nevýhodou by se mohlo zdát, že deska nedisponuje možností vyvolat přerušení procesoru. Aby se procesor dozvěděl o případné změně stavu na vstupu musí cyklicky v pravidelných časových intervalech vyčítat vstupní stav a následně ho vyhodnocovat. Možností vyvolat přerušení by se činnost procesoru přerušila pouze při dané změně na vstupu.

4.7 Deska výstupů

Stejně jako deska vstupů i deska výstupů zajišťuje styk terminálu s daným procesem [2]. Výstup je realizovaný za pomoci přepínacích kontaktů 8 relátek. Terminál tak dostává možnost provádění regulačních zásahů na základě svých výpočtů.

Struktura desky výstupů je obdobná jako u desky vstupů. Datová část sběrnice je přivedena na vstupy střadače. Jeho výstupy jsou připojeny přes patřičné obvody na 8 relátek s přepínatelnými kontakty. Pokut procesor dospěje při zpracovávání programového kódu do bodu, kdy potřebuje změnit stav výstupů, vykoná tak zápisem patřičných dat do svého externího datového prostoru na příslušnou adresu. Při tomto zápisu se automaticky uvolní vstup střadače a po skončení se uzavře. Střadač si tímto způsobem zapamatuje vstupní stav až do dalšího zápisu.

4.8 Zdrojová deska

Jak již název napovídá zdrojová deska obsahuje napájecí zdroj, který napájí celý terminál [2]. Navíc ještě obsahuje galvanicky oddělený převodník napěťových úrovní sériové komunikace.

Napájecí zdroj je řešen klasickým způsobem za pomocí transformátoru, usměrňovače a dvojce stabilizátorů +5V a +12V. Vlastní sériové rozhraní je implementováno jako integrovaná periferie přímo v řídícím mikrokontroléru. Jeho patřičné piny jsou prostřednictvím sběrnice připojeny přes optočleny, které zajišťují galvanické oddělení, na známý převodník napěťových úrovní MAX 232. Tím je zajištěn převod úrovní z napětí 0V a +5V na napětí +12V a -12V, které odpovídají standartu RS232.

5. AD převod a jeho řízení

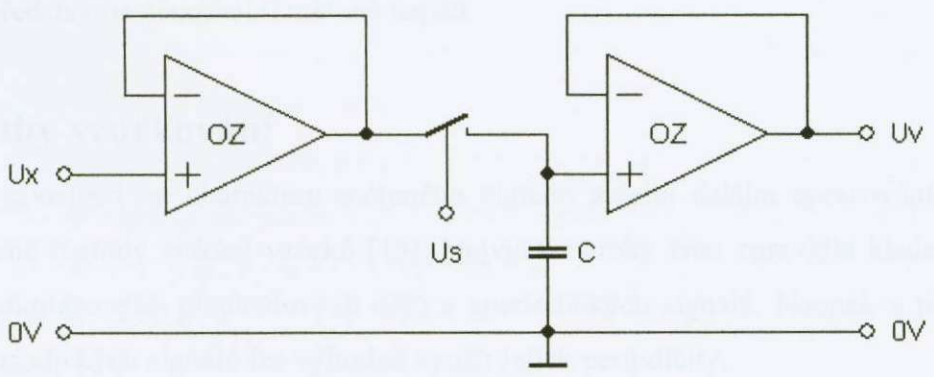
V úvodu této kapitoly jsou vysvětleny základní pojmy týkající se problematiky analogově digitálního (AD) převodu a jeho řízení. Dále jsou zde uvedeny vybrané metody AD převodníků. Na závěr je pak uvedeno řízení AD převodu v terminálu včetně ovládacího algoritmu.

5.1 Vzorkování

Analogově digitální (AD) převod spočívá v číslicové reprezentaci spojitého analogového signálu [12]. Tato reprezentace je tvořena diskrétní časovou posloupností okamžitých číslicových hodnot měřeného signálu, které jsou od sebe vzdáleny o jistý časový interval. Celý proces AD převodu by se pak dal rozdělit na tyto části

- Vzorkování - spočívá v odběru vzorku vstupního signálu v definovaných časových okamžicích. V podstatě se jedná o převod signálu v čase spojitém na nespojitý [12].
- Kvantování - jednotlivých vzorků zabezpečuje přiřazení jejich velikosti jistou nejbližší číslicovou hodnotu. Převádí tedy signál v úrovních spojitý na nespojitý [12].
- Kódování - spočívá ve vyjádření kvantovacích hodnot v jistém číslicovém kódu [12]. Zpravidla se využívá přímého a doplňkového kódu.

Funkce vzorkovače tedy spočívá jednak v odběru vzorku vstupního signálu v předem definovaném časovém okamžiku, ale také v udržení jeho hodnoty po celou tuto dobu na konstantní úrovni. Obvod, který danou funkci realizuje se nazývá vzorkovač s analogovou pamětí S/H (sample and hold), případně vzorkovací zesilovač (obr. 5.1).



Obr. 5.1 Ideální vzorkovací zesilovač.

Oba operační zesilovače zde plní funkci sledovače, ve kterém se vyznačují velkým vstupním a malým výstupním odporem. První operační zesilovač na svém výstupu přesně kopíruje vstupní napětí *Ux*. V případě požadavku k odběru vzorku (signál *Us*), dojde k otevření spínače a takřka okamžitém nabití paměťového kondenzátoru *C* na požadované napětí *Ux*. Druhý operační zesilovač již jen přenáší napětí z paměťového kondenzátoru na výstup. Pro optimální funkci vzorkovacího zesilovače je nutné, aby se spínač vyznačoval co možná nejmenším odporem v sepnutém stavu. V případě nesplnění této podmínky je třeba prodlužit dobu jeho sepnutí. Prakticky je spínač obvykle realizován pomocí diod nebo rychlého unipolárního tranzistoru.

Základní podmínkou pro správné vzorkování je, aby vzorkovaný signál mohl být zpětně rekonstruován bez ztráty informace. Tuto podmínku vyjadřuje tzv. Nyquistův, Shannon-Kotělnikovův vzorkovací teorém [13], který stanovuje minimální vzorkovací frekvenci f_V na základě maximálního kmitočtu f_{MAX} spektra měřeného signálu.

$$f_V \geq 2 f_{MAX} \text{ [Hz]} \tag{5.1}$$

Vlivem konečné periody vzorkování $T_V = 1 / f_V$ se v měřeném signálu vytváří periodické spektrum s periodou f_V. Při nesplnění vzorkovacího teorému, dojde k překrývání těchto sousedních spekter (aliasing) a tím k znehodnocení měřeného signálu. Tento signál pak již nelze přesně rekonstruovat. Někdy se z těchto důvodu začleňuje před vzorkovací zesilovač filtr typu dolní propust (antialiasingový filtr).

Jak již bylo řečeno kvantování spočívá v lineárním přiřazení velikosti jednotlivých vzorků jisté nejbližší číslicové hodnotě. Počet těchto číslicových hodnot *M* (kvantovacích hladin) je dán bitovým rozsahem *N* použitého převodníku a to sice $M = 2^N$. Velikost kvantovacího kroku *q* je pak možno vyjádřit jako

$$q = \frac{U_M}{2^N - 1} \text{ [V/bit]} \tag{5.2}$$

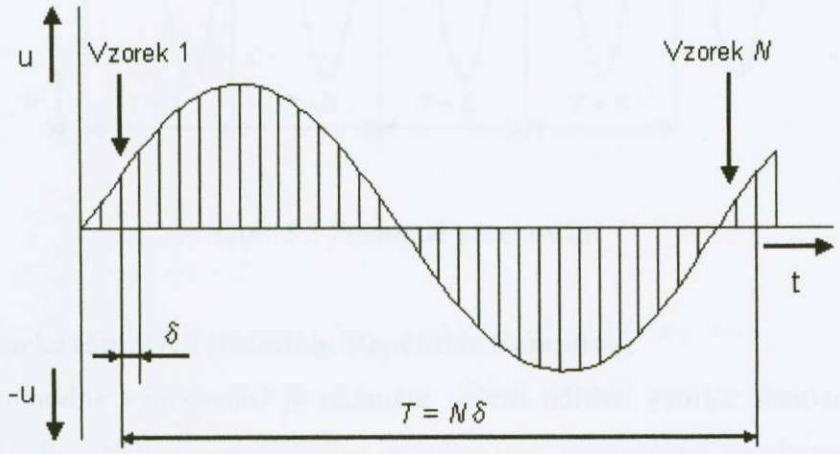
Kde U_M představuje maximální měřené napětí.

5.2 Druhy vzorkování

V závislosti na charakteru měřeného signálu a jeho dalším zpracováním lze použít různé metody získání vzorků [13]. Nejvyšší nároky jsou zpravidla kladeny na měření jednorázových přechodových dějů a aperiodických signálů. Naopak v případě měření periodických signálů lze výhodně využít jejich periodicity.

Vzorkování v reálném čase RTS (Real Time Sampling)

Vzorkování v reálném čase (obr. 5.2) je charakteristické konstantní vzorkovací periodou δ [13]. V případě měření periodických signálů jsou všechny vzorky odebrány během jediné periody a je tedy nutno splnit podmínku danou vzorkovacím teorémem.



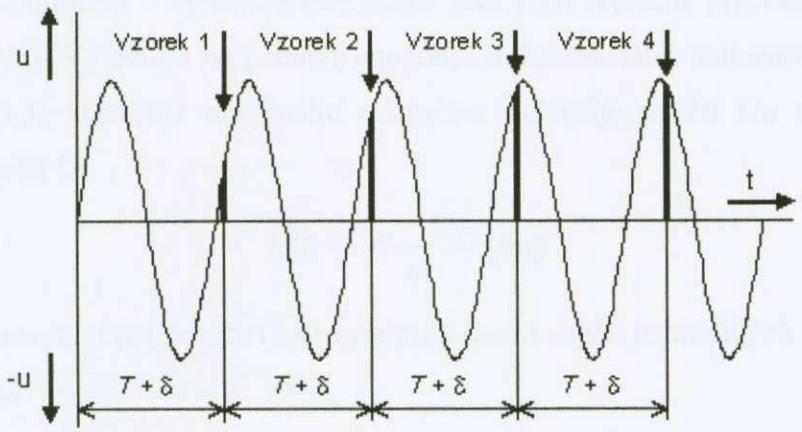
Obr. 5.2 Vzorkování v reálném čase.

Vzorkování v reálném čase je dále vhodné pro měření přechodových jevů a neperiodických signálů. Vyznačuje se snadnou technickou realizací a jednoduchým následným zpracováním měřeného signálu. Nevýhodou je však vysoký požadavek na rychlost použitého AD převodníku.

Postupné vzorkování RS (Repetite Sampling)

Tohoto principu vzorkování lze využít výhradně pouze u stabilních periodických signálů [13]. V každé periodě je vždy odebrán pouze jediný vzorek (obr. 5.3). K odběru dalšího vzorku dochází vždy až v následující periodě a to sice v čase $T+\delta$. Kde δ reprezentuje tzv. efektivní vzorkovací periodu. K získání požadovaného počtu N vzorků na periodu je tedy potřebný také počet N period.

Hlavním nedostatkem postupného vzorkování je omezení metody na periodické signály. Naopak nespornou výhodou je možnost vzorkování signálů o vysokých frekvencích s použitím ne příliš rychlého AD převodníku.



Obr. 5.3 Postupné vzorkování.

Náhodné vzorkování RRS (Random Repetitive Sampling)

Pro náhodné vzorkování je okamžik určení odběru vzorku stanoven pseudonáhodně [13]. Po příchodu spouštěcího impulsu jsou generovány vzorkovací impulsy s náhodným, ale přesně známým časovým rozdělením. Z takto získaného velkého souboru dat je pak následně rekonstruován původní signál. Hlavní nevýhodou náhodného vzorkování jsou vysoké nároky na výpočetní výkon procesoru.

5.3 Chyby AD převodníků

Každý AD převodník se v podstatě skládá ze dvou částí. První je analogová, ve které dochází k zesílení či zeslabení měřeného signálu, případně k jeho impedančnímu oddělení. Dále by se do této části dal také zařadit vzorkovací zesilovač. Následující digitální část má za úkol provést kvantování a kódování. Průchodem signálu těmito obvody dochází k následujícím chybám [12].

- Chyba zesílení (gain error) - je dána rozdílem sklonu skutečné převodní charakteristiky a ideální převodní charakteristiky. Zpravidla je způsobena špatně nastaveným ziskem předcházejících zesilovačů a dá se tedy korigovat. Problémem však může být její teplotní stabilita.
- Chyba nuly (zero error) - způsobuje posunutí převodní charakteristiky o konstantní hodnotu. Může být způsobena špatně vykompenzovanou napěťovou nebo proudovou nesymetrií operačních zesilovačů. Případně ji může způsobit nestabilita zdroje referenčního napětí.

- Chyba linearity - vyjadřuje maximální odchylku skutečné převodní charakteristiky od ideální. Vyjadřuje se pomocí integrální a diferenciální nelinearity. Integrální nelinearita (5.3) vyjadřuje maximální odchylku měřeného napětí U_M od teoreticky měřeného napětí U_T.

$$INL = \frac{U_M - U_T}{q} \text{ [bit]} \tag{5.3}$$

Naopak diferenciální nelinearita (5.4) vyjadřuje rozdíl napětí jednotlivých kvantovacích kroků U_i a U_{i-1}.

$$DNL = \frac{U_i - U_{i-1}}{q} - 1 \text{ [bit]} \tag{5.4}$$

Kde q reprezentuje kvantovací krok.

- Chyba kvantování - je způsobena podstatou kvantování, která spočívá v zaokrouhlení měřeného napětí na nejbližší zobrazitelnou diskrétní hodnotu. Maximální takto způsobená chyba je dána polovinou kvantovacího kroku $\pm q/2$.

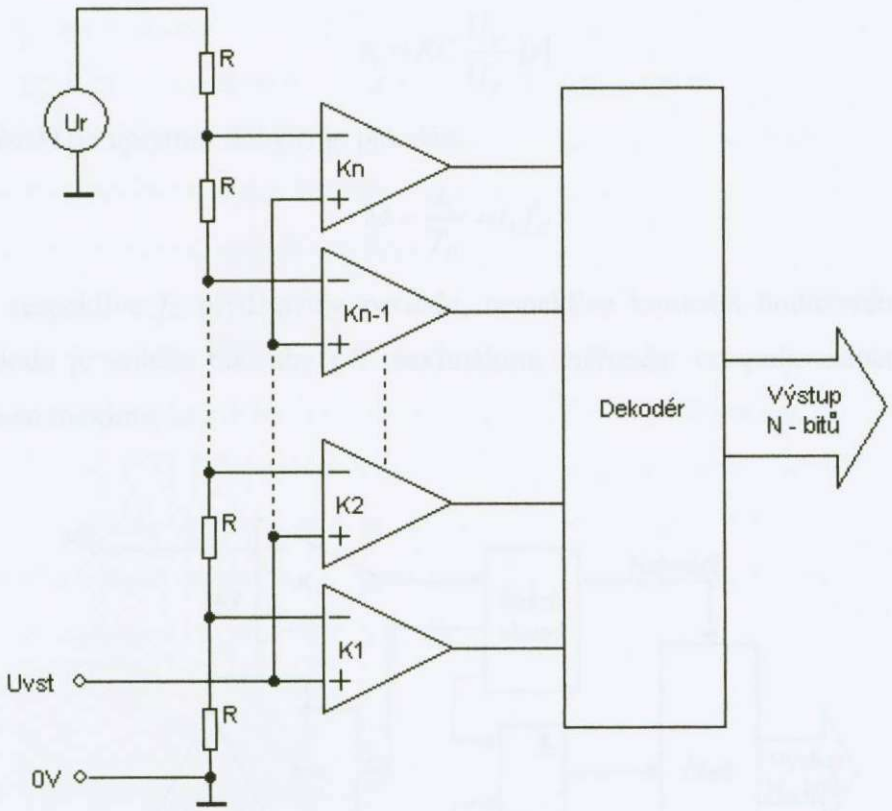
5.4 AD převodníky

Nyní následuje seznámení se s vybranými principy AD převodníků. Jednotlivé principy AD převodu sebou nesou své výhody i nevýhody a podle nich je lze aplikovat v různých měřících systémech.

Paralelní komparační převodníky

Paralelní komparační převodník (obr. 5.4) neustále porovnává vstupní napětí U_{VST} s kvantovacími úrovněmi [12]. Jednotlivé kvantovací úrovně jsou získány z přesného referenčního zdroje napětí U_R pomocí přesné odporové sítě. Počet vstupních komparátorů musí odpovídat počtu kavntovacích hladin. Dvouhodnotový výstup komparátorů je následně převeden dekodérem do některého z výhodných dvojkových kódů.

Celý AD převod probíhá prakticky v jediném okamžiku, je omezen pouze zpožděním komparátorů, dekodéru a ustálením přechodových jevů. Tyto převodníky běžně dosahují 10^7 až 10^9 převodů za sekundu. To je přímo předurčuje pro digitalizaci rychlých signálů. Prakticky se vyrábějí v 6, 8 a 10 bitovém rozlišení. Jejich nevýhodou je, že obsahují velký počet komparátorů.



Obr. 5.4 Paralelní komparační převodník.

Integrační převodníky s jednotaktní integrací

Integrační převodníky (obr. 5.5) převádějí napětí na časový interval (obr. 5.6) pomocí integrátoru [12]. Vstupní napětí U_x je neustále porovnáváno komparátorem K s rostoucím integračním napětím U_{INT}, které je odvozeno od přesného referenčního zdroje napětí $-U_r$. Pro integrační napětí U_{INT} platí

$$U_{INT} = -\frac{1}{RC}\int(-U_r)\mathrm{d}t = \frac{U_r}{RC}t_1 \text{ [V]} \tag{5.5}$$

Na začátku převodu řídící obvod vygeneruje signály pro vynulování čítače, vybití integračního kondenzátoru C a otevření hradla. Tím se zabezpečí lineární růst integračního napětí U_{INT} a načítání hodinových impulsů čítačem. V okamžiku shody napětí U_x a U_{INT} je řídícím obvodem zablokováno hradlo.

$$U_x = U_{INT} \tag{5.6}$$

$$U_x = \frac{U_r}{RC}t_1 \tag{5.7}$$

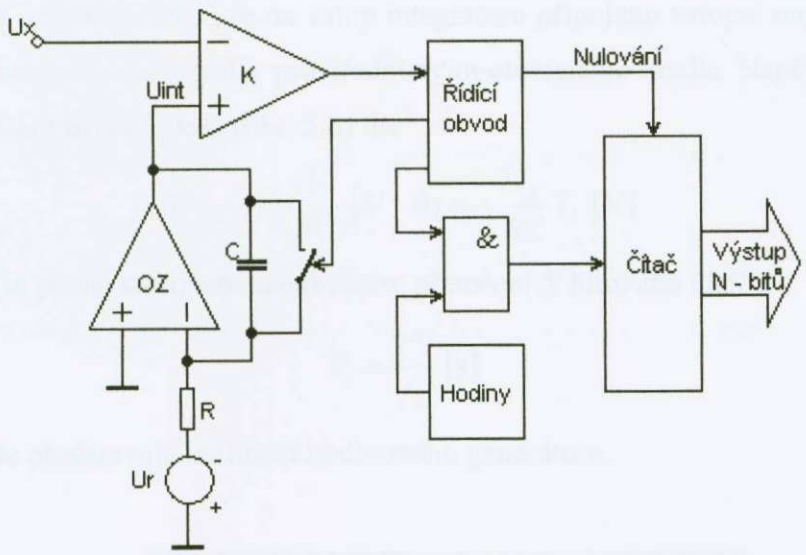
Pro dubu otevření hradla t_1 musí platí

$$t_1 = RC\frac{U_x}{U_r}\ [\mathrm{s}] \tag{5.8}$$

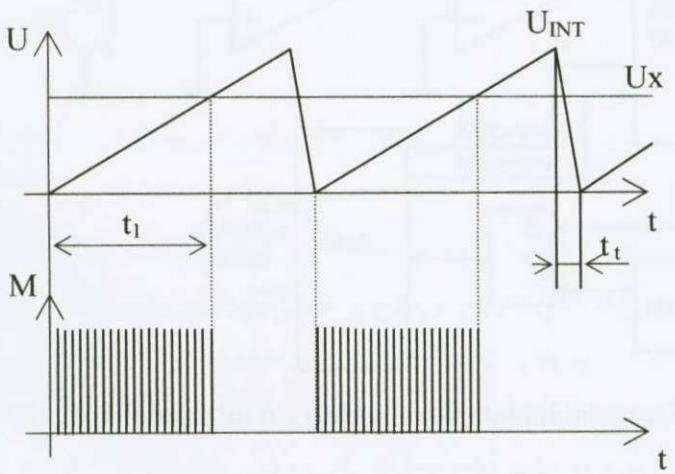
Stav čítače M po uplynutí doby t_1 je pak dán

$$M = \frac{t_1}{T_G} = t_1 f_G \tag{5.9}$$

Kde T_G, respektive f_G představuje periodu, respektive kmitočet hodinového signálu. Tato perioda je volena tak, aby při maximálním měřeném vstupním napětí byl stav čítače roven maximu.



Obr. 5.5 Integrační převodník.



Obr. 5.6 Časové průběhy integračního převodníku.

Hlavní nevýhodou tohoto typu převodníku je obtížnost přesného vynulování integračního kondenzátoru C, na kterém závisí doba otevření čítače. Další problém nastává v případě „zašumělého" vstupního napětí, pak totiž zpravidla dochází k předčasnému ukončení převodu. Dále jsou zde kladeny vysoké požadavky na časovou a teplotní stabilitu RC článku. Z těchto důvodů se tento typ převodníku prakticky nevyrábí a je nahrazen dokonalejším kolegou s dvoutaktní integrací.

Integrační převodníky s dvoutaktní integrací

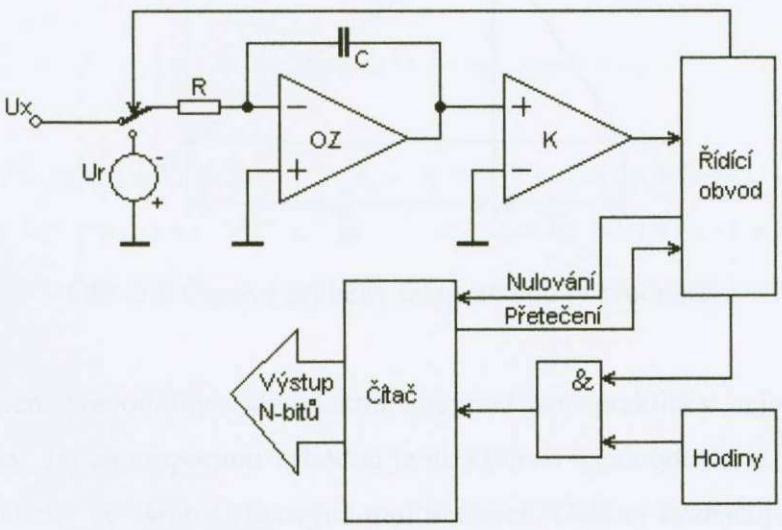
Tento typ integračního převodníku (obr. 5.7) již podle názvu pracuje ve dvou taktech (cyklech) [12]. Před začátkem převodu je integrační kondenzátor C vybit a čítač vynulován. V prvním taktu T_1 je na vstup integrátoru připojeno vstupní napětí $U_x > 0$ a čítač se plní hodinovými impulsy prostřednictvím otevřeného hradla. Napětí na výstupu integrátoru U_{INT} lineárně klesá (obr. 5.8) dle

$$U_{INT} = -\frac{1}{RC}\int U_X \, \mathrm{d}t = -\frac{U_X}{RC} T_1 \ [\mathrm{V}] \tag{5.10}$$

Kde doba T_1 je pevně stanovena okamžikem přetečení N bitového čítače.

$$T_1 = \frac{2^N}{f_h} \ [s] \tag{5.11}$$

Přičemž f_h zde představuje kmitočet hodinového generátoru.



Obr. 5.7 Integrační převodník s dvoutaktní integrací.

Přetečením čítače zároveň začíná druhý takt převodu. Při něm se stav čítače automaticky vynuluje, k integrátoru je připojeno přesné referenční napětí $U_x < 0$ a hodinový

kmitočet je stále připojen k čítači pomocí hradla. Velikost výstupního napětí U_{INT} se dále zvětšuje až do nulové hodnoty po dobu T_2.

$$U_{INT} = -\frac{1}{RC}\int(-U_R)\,dt + U_0 \text{ [V]} \tag{5.12}$$

U_0 zde představuje počáteční stav integrátoru tedy hodnotu ze vztahu (5.10).

$$U_{INT} = -\frac{1}{RC}\int(-U_R)\,dt - \frac{U_X}{RC}T_1 = 0 \tag{5.13}$$

Z čehož po integraci získáváme

$$\frac{U_R}{RC}T_2 - \frac{U_X}{RC}T_1 = 0 \tag{5.14}$$

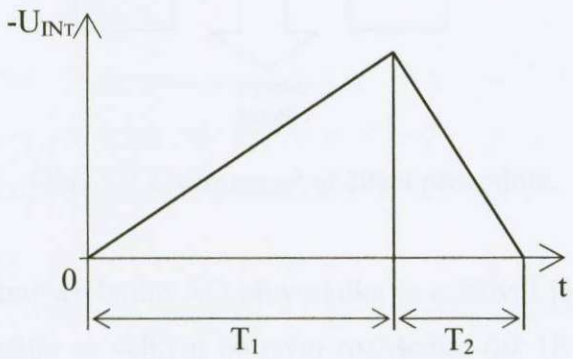
A dále úpravou pro T_2 musí platit

$$T_2 = \frac{U_X}{U_R}T_1 \text{ [s]} \tag{5.15}$$

V čase T_2 končí druhá fáze převodu a tím i celý AD převod. Stav čítače M pak bude

$$M = \frac{T_2}{T_h} = f_h\frac{U_X}{U_R}T_1 = f_h\frac{U_X}{U_R}\frac{2^N}{f_h} = \frac{U_X}{U_R}2^N \tag{5.16}$$

Kde T_h, respektive f_h je perioda hodinového signálu, respektive jeho kmitočet.

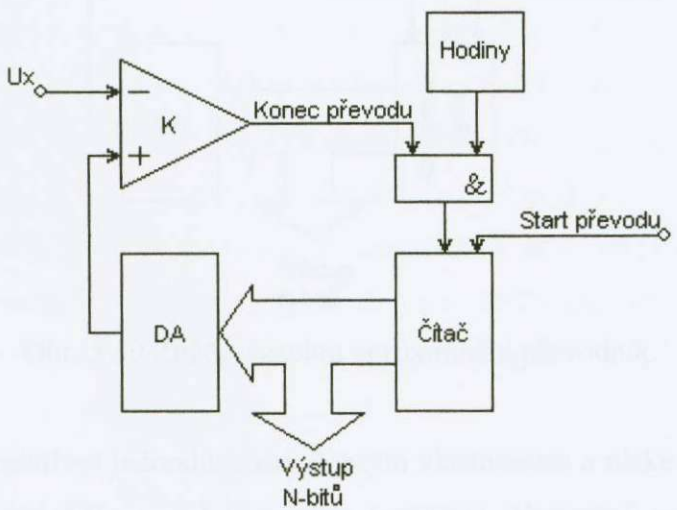


Obr. 5.8 Časové průběhy integračního převodníku.

Integrační převodníky s dvoutaktní integrací jsou prakticky nejpoužívanějšími AD převodníky. Jejich nespornou výhodou je nízká cena a jednoduchost, díky čemuž je lze nalézt prakticky ve všech číslicových multimetrech. Dále se vyznačují velmi dobrou linearitou, vysokým bitovým rozlišením (až 22 bitů) a odolností vůči vstupnímu rušení. Při pohledu na vztah (5.16) je dále patrné, že jejich přesnost se odvíjí pouze od stability referenčního zdroje napětí. Nevýhodou je však rychlost převodu, která činí maximálně stovky převodů za sekundu.

Zpětnovazební (kompenzační) čítací převodníky

Principem zpětnovazebního čítacího převodníku je neustálé porovnávání měřeného napětí U_x a zpětnovazebního napětí pomocí komparátoru (obr. 5.9) [12]. Převod je spuštěn pomocí externího signálu start převodu. Tím se vynuluje čítač a zároveň se otevře hradlo, přes které se čítač plní hodinovými impulsy. Výstupní číslicové slovo čítače je okamžitě převedeno do analogové podoby pomocí DA převodníku na zpětnovazební porovnávací napětí. S postupným zvyšováním obsahu čítače se tedy zároveň zvyšuje i ono zpětnovazební porovnávací napětí. V případě, že velikost tohoto napětí převýší velikost měřeného napětí, uzavře komparátor hradlo. Tím se ukončí AD převod, přičemž stav čítače je úměrný vstupnímu napětí U_x.



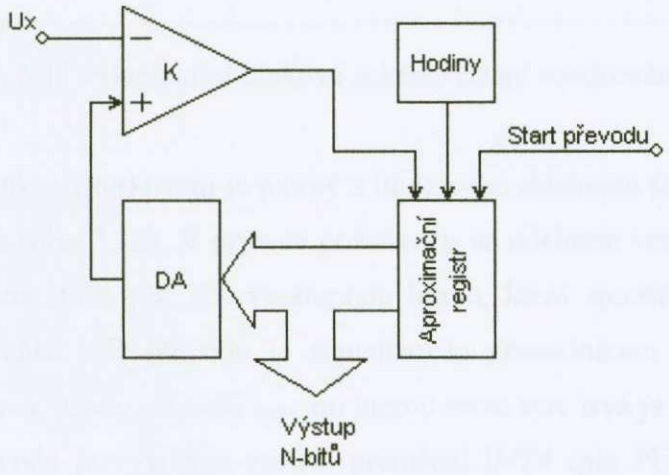
Obr. 5.9 Zpětnovazební čítací převodník.

Výhodou zpětnovazebního AD převodníku je relativní jednoduchost a tím také příznivá cena. Vyznačuje se velkým bitovým rozlišením (až 18 bitů) a vcelku dobrou linearitou. Nevýhodné je však, že doba převodu není z principu konstantní.

Zpětnovazební (kompenzační) aproximační převodníky

Použitím zpětnovazebního aproximačního převodníku lze dosáhnout značného zrychlení AD převodu oproti např. čítacímu převodníku [12]. Rozdíl vůči předchozímu čítacímu převodníku spočívá především v řízení změny číslicového slova, tedy i zpětnovazebního porovnávacího napětí, pomocí aproximačního registru (obr. 5.10). Funkce převodníku pak spočívá v postupném nastavování jednotlivých váhových bitů N bitového aproximačního registru. Na začátku převodu se nastaví nejvyšší MSB váhový

bit, tomu zároveň odpovídá zpětnovazební porovnávací napětí $U_{MAX}/2$. Toto napětí je dále porovnáno se vstupním napětím U_x a pokud komparátor určí jako větší napětí U_x, ponechá MSB bit aproximačního registru nastaven. V opačném případě ho vynuluje. V dalším kroku se nastaví MSB-1 bit a opět dojde k výše popsanému porovnání a případnému vynulování tohoto bitu. Posloupnost těchto kroků se opakuje až k nejnižšímu váhovému bitu LSB. Počet těchto kroků je tedy stejný jako šířka použitého aproximačního registru.

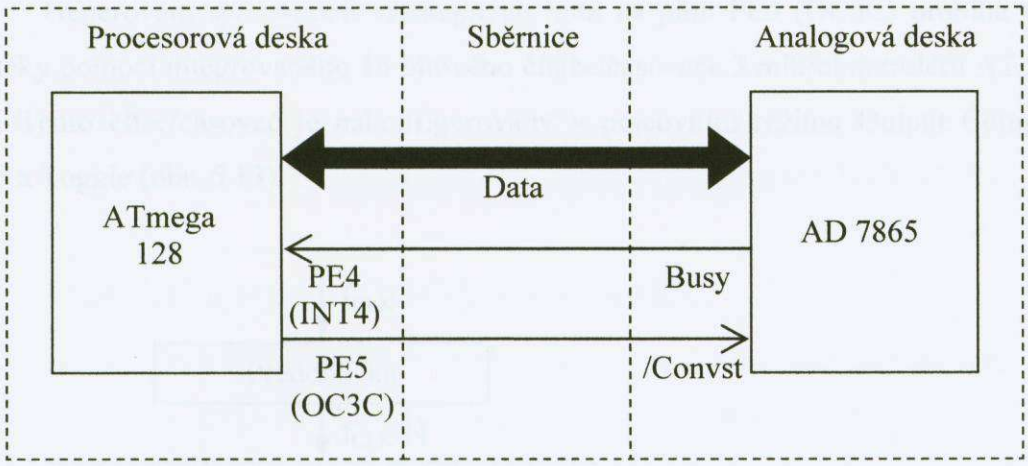


Obr. 5.10 Zpětnovazební aproximační převodník.

Díky své relativní jednoduchosti, dobrým vlastnostem a nízké ceně jsou zpětnovazební aproximační AD převodníky velice rozšířeny. Vyznačují se velkým bitovým rozlišením (až 18 bitů), velmi dobrou linearitou a rychlostí převodu až 10^7 za sekundu.

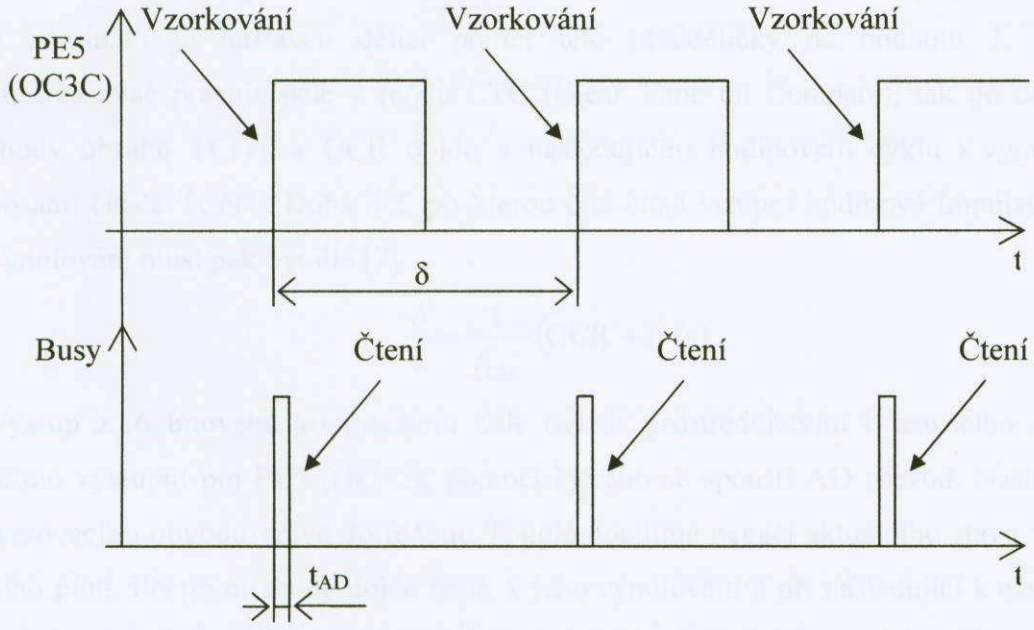
5.5 Hardwarové řízení vzorkování v terminálu

Vzhledem k zaměření funkce terminálu a požadavku na rychlé zpracování naměřených dat bylo zvoleno vzorkování v reálném čase RTS (kapitola 5.2). Úkolem řídícího mikrokontroléru ATmega 128 je tedy zabezpečení správného vzorkování, tak aby byla celá perioda měřeného signálu zachycena v požadovaném počtu vzorků N. Kde pro N byla stanovena hodnota 128 vzorků s ohledem na splnění Nyquistova, Shannon Kotělnikovova vzorkovacího teorému (kapitola 5.2) a konečné výpočetní rychlosti mikrokontroléru ATmega 128 [2].



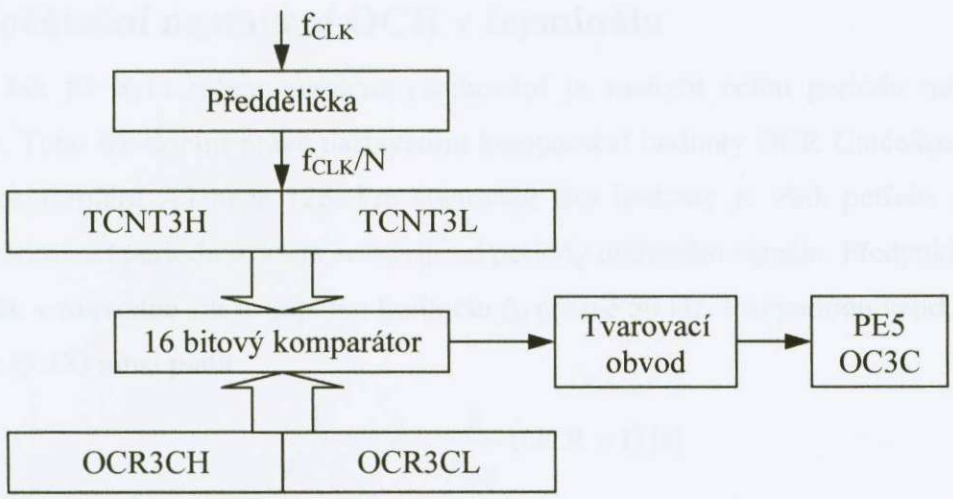
Obr. 5.11 Principiální blokové schéma řízení vzorkování [2].

Princip funkce vzorkování je patrný z blokového schématu (obr. 5.11) a z časového diagramu na (obr. 5.12). V případě požadavku na odebrání vzorku, generuje procesor na svém pinu PE5 (OC3C) vzestupnou hranu, která spouští AD převod (viz. kapitola 4.5). Průběh AD převodu je signalizován převodníkem vysokou logickou úrovní na pinu Busy. Doba převodu t_{AD}, po kterou tento stav trvá je zhruba 10 μs [11]. Po ukončení převodu je vyvoláno externí přerušení INT4 (pin PE4) činnosti mikrokontroléru od sestupné hrany signálu Busy. V tomto přerušení dochází ke čtení obsahu AD převodníků a uložení těchto dat do paměti.



Obr. 5.12 Časový průběh řídících signálů [2].

Generování spouštěcích vzestupných hran na pinu PE5 (OC3C) probíhá automaticky pomocí integrovaného 16 bitového čítače/časovače 3 mikrokontroléru ATmega 128. Tento čítač/časovač je nakonfigurovaný v pracovním režimu Output Compare, CTC a Toggle (obr. 5.13).



Obr. 5.13 Zjednodušené blokové schéma čítače/časovače v režimu Output Compare [7].

Základním prvkem čítače/časovače v tomto režimu je 16 bitový komparátor, který porovnává nastavenou (komparační) hodnotu v registrech OCR3CL a OCR3CH s hodnotou vzestupně čítajícího čítače TCNT3L a TCNT3H. Vstupní impulsy čítače jsou odvozeny pomocí předděličky přímo z krystalového oscilátoru. V případě aplikace v terminálu je nastaven dělící poměr této předděličky na hodnotu 1. Jelikož čítač/časovač pracuje dále v módu CTC (Clear Time on Compare), tak po dosažení shody obsahů TCNT a OCR dojde v následujícím hodinovém cyklu k vynulování obsahu čítače TCNT. Doba $\delta/2$, po kterou čítá čítač vstupní hodinové impulsy až do vynulování musí pak být dle [7]

$$\frac{\delta}{2} = \frac{1}{f_{CLK}}(\mathrm{OCR}+1)\,[\mathrm{s}] \tag{5.17}$$

Výstup z 16 bitového komparátoru dále ovládá prostřednictvím tvarovacího obvodu přímo výstupní pin PE5 (OC3C), pomocí kterého se spouští AD převod. Nastavením tvarovacího obvodu právě do režimu Toggle docílíme negaci aktuálního stavu výstupního pinu. Při první shodě dojde např. k jeho vynulování a při následující k nastavení. Doba mezi dvěmi vzestupnými hranami je pak dvojnásobkem doby $\delta/2$ a určuje vzorkovací periodu.

$$\delta = \frac{2}{f_{CLK}}(\mathrm{OCR}+1)\,[\mathrm{s}] \tag{5.18}$$

Tímto jednoduchým způsobem je zajištěno automatické spouštění AD převodu v daných časových okamžicích v závislosti na nastavení komparační hodnoty OCR.

5.6 Počáteční nastavení OCR v terminálu

Jak již bylo řečeno úkolem vzorkování je zachytit celou periodu měřeného signálu. Toho lze docílit právě nastavením komparační hodnoty OCR čítače/časovače3 v mikrokontroléru ATmega 128. Pro stanovení této hodnoty je však potřeba předem znát vzorkovací periodu δ, která se odvíjí od periody měřeného signálu. Předpokládejme zatím, že v rozvodné síti je napětí o kmitočtu f_M přesně 50 Hz. Pro periodu vzorkování δ pak dle (5.18) musí platit

$$\frac{1}{f_M N} = \delta = \frac{2}{f_{CLK}}(\mathrm{OCR}+1)\,[\mathrm{s}] \tag{5.19}$$

Kde N je stanovený počet vzorků (viz. kapitola 5.5).

Vyjádříme-li nyní OCR z rovnice (5.19)

$$\mathrm{OCR} = \frac{f_{CLK}}{2Nf_M} - 1 \tag{5.20}$$

Po dosazení konkrétních hodnot do (5.20) získáme hodnotu 1249. Právě touto hodnotou je třeba inicializovat komparační registr OCR3C na počátku měření. Od této hodnoty se bude dále odvíjet další nastavení OCR v případě měření napětí o odlišném kmitočtu.

Dále je potřeba stanovit přesnost zachycení periody $\Delta f/f_{OCR}$ v případě jednotkové změny OCR. Přičemž výrazem $\Delta f = f_{OCR} - f_{OCR-1}$ se rozumí změna kmitočtu, které by odpovídala jednotková změna OCR. Pro frekvenci f_{OCR}, při které by byla perioda přesně zachycena platí podle (5.19)

$$f_{OCR} = \frac{f_{CLK}}{2N(\mathrm{OCR}+1)}\,[\mathrm{Hz}] \tag{5.21}$$

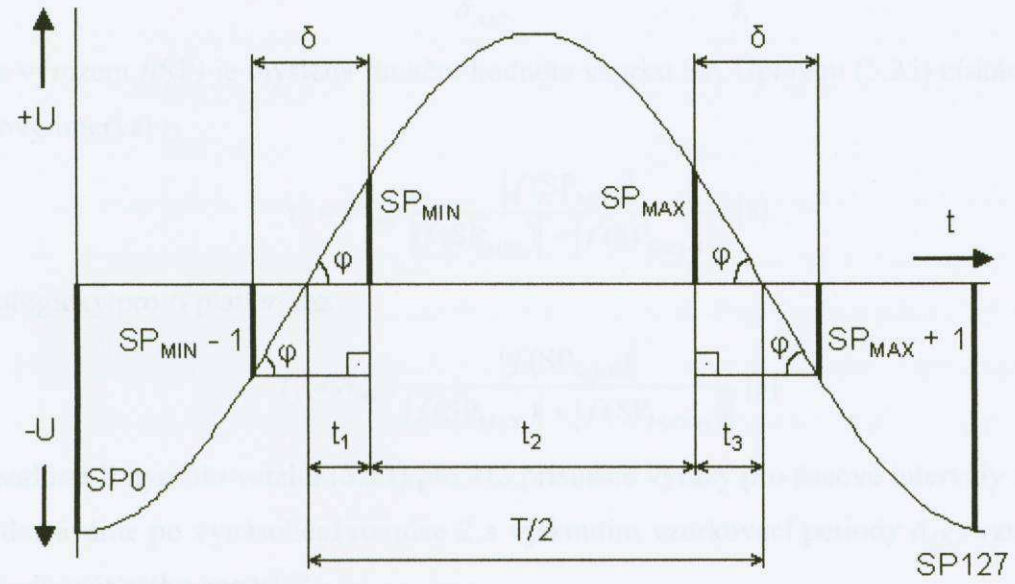
Výraz pro přesnost zachycení pak po značných úpravách nabude tvaru

$$\frac{f_{OCR} - f_{OCR-1}}{f_{OCR}} = \frac{-1}{\mathrm{OCR}} \tag{5.22}$$

V případě přesného zachycení měřeného napětí s kmitočtem 50 Hz (OCR = 1249) dostáváme přesnost zachycení zhruba 0,08 %. Tato hodnota značí, že zachycení periody tímto způsobem je velice přesné.

5.7 Výpočet nové hodnoty OCR v terminálu

Bohužel v rozvodné síti není zaručen kmitočet napětí přesně 50 Hz [2]. Příslušná norma povoluje hodnotu síťového kmitočtu v rozmezí 50 Hz ± 5 % (50 Hz ± 2,5 Hz). Vzhledem k tomu, že vzorkovací perioda δ je na tomto kmitočtu závislá, musí ho terminál neustále vyhodnocovat. Ve skutečnosti není potřeba měřit přímo kmitočet, ale pouze přesný počet vzorků na polovinu periody.



Obr. 5.14 Stanovení přesného počtu vzorků.

Uvažujme situaci, kdy nedochází k zachycení přesně jedné periody. V tom případě není splněna podmínka N = 128 vzorků na celou periodu. Těchto vzorků bude třeba jen 127 a nebo 128,4. Tento přesný počet vzorků lze stanovit na základě navzorkované posloupnosti dat uložené v patřičném bufferu. Prakticky v každém takto získaném bufferu je možno vyhledat dva vzorky SP_{MIN} a SP_{MAX} těsně u průchodů nulou, které ohraničují půlperiodu měřeného napětí zevnitř (obr. 5.14). Algoritmem vyhledání těchto vzorků se podrobně zabývá kapitola 5.9. Pro půlperiodu $T/2$ měřeného napětí musí platit

$$\frac{T}{2} = t_1 + t_2 + t_3 \text{ [s]} \tag{5.23}$$

Časový interval t_2 určíme jednoduše na základě vzdálenosti jednotlivých vzorků SP_{MIN} a SP_{MAX}. Přičemž vzdálenost dvou sousedních vzorků je aktuální vzorkovací perioda δ_{AKT}, které odpovídá hodnota OCR_{AKT} komparačního registru OCR3C.

$$t_2 = \delta_{AKT}(SP_{MAX} - SP_{MIN}) \text{ [s]} \tag{5.24}$$

Pro výpočet časových intervalů t_1 a t_3 využijeme skutečnosti, že směrnice funkce sinus má hodnotu prakticky konstantní v okolí průchodů nulou. Pak na základě podobnosti trojúhelníku platí

$$tg\,\varphi = \frac{|f(SP_{MIN})| + |f(SP_{MIN-1})|}{\delta_{AKT}} = \frac{|f(SP_{MIN})|}{t_1} \tag{5.25}$$

Kde výrazem $f(SP)$ je myšlena funkční hodnota vzorku SP. Úpravou (5.25) získáme pro časový interval t_1

$$t_1 = \delta_{AKT}\frac{|f(SP_{MIN})|}{|f(SP_{MIN})| + |f(SP_{MIN-1})|} \text{ [s]} \tag{5.26}$$

Analogicky pro t_3 platí výraz

$$t_3 = \delta_{AKT}\frac{|f(SP_{MAX})|}{|f(SP_{MAX})| + |f(SP_{MAX+1})|} \text{ [s]} \tag{5.27}$$

Dosadíme-li nyní do vztahu (5.23) pro $T/2$ příslušné výrazy pro časové intervaly t_1, t_2 a t_3, dostáváme po vynásobení rovnice 2 a vytknutím vzorkovací periody δ_{AKT} vztah pro periodu měřeného napětí T

$$T = 2\delta_{AKT}\left(\frac{|f(SP_{MIN})|}{|f(SP_{MIN})| + |f(SP_{MIN-1})|} + f(SP_{MAX}) - f(SP_{MIN}) + \right.$$
$$\left. + \frac{|f(SP_{MAX})|}{|f(SP_{MAX})| + |f(SP_{MAX+1})|}\right) \text{ [s]} \tag{5.28}$$

Přičemž výraz v závorce, pojmenujme ho α, má význam právě skutečného počtu vzorků na půlperiodu. S využitím této substituce platí pro (5.28)

$$T = 2\delta_{AKT}\alpha \text{ [s]} \tag{5.29}$$

Se znalostí skutečného počtů vzorků α na půlperiodu můžeme nyní vypočíst hodnotu nové komparační hodnoty OCR_{NEW} registru OCR3C. Přičemž tuto hodnotu stanovíme právě tak, aby byla splněna podmínka $N = 128$ vzorků přesně na periodu. Pro komparační hodnotu OCR_{NEW} platí dle (5.20)

$$OCR_{NEW} = \frac{f_{CLK}}{2Nf_M} - 1 \tag{5.30}$$

Kde N je námi požadovaný počet vzorků na periodu, tedy hodnota 128. Za frekvenci měřeného napětí $f_M = 1/T_M$ můžeme na základě vztahu (5.29) dosadit do (5.30)

$$\mathrm{OCR}_{\mathrm{NEW}} = \frac{\delta_{AKT}\,\alpha\, f_{CLK}}{N} - 1 \tag{5.31}$$

Dosazením aktuální vzorkovací periody δ_{AKT} na základě nastavené aktuální hodnoty $\mathrm{OCR}_{\mathrm{AKT}}$ komparačního registru OCR3C (5.19), obdržíme po úpravě

$$\mathrm{OCR}_{\mathrm{NEW}} = \frac{2}{N}\alpha\,(\mathrm{OCR}_{\mathrm{AKT}} + 1) - 1 \tag{5.32}$$

Tímto způsobem lze stanovit nové nastavení komparačního registru OCR3C, tak aby byla splněna podmínka N vzorků na celou periodu. Nevýhodou uvedeného principu je nutnost vyhledání průchodů nulou v naměřených datech, které trvá jistou dobu (viz. kapitola 5.9). Odstranění této nevýhody by mohlo být začlenění napěťového komparátoru do analogové desky. Tento komparátor by neustále porovnával měřené napětí s nulou. V případě jeho překlopení, detekovaný průchod nulou, by vyvolal např. přerušení mikrokontroléru. Měřením časového intervalu mezi těmito přerušeními by se dala pomocí dalšího integrovaného čítače/časovače (režim Input Compare) mikrokontroléru mnohem jednodušeji a hlavně rychleji stanovit perioda měřeného signálu. Tato úprava by však znamenala značný zásah do hardwaru terminálu.

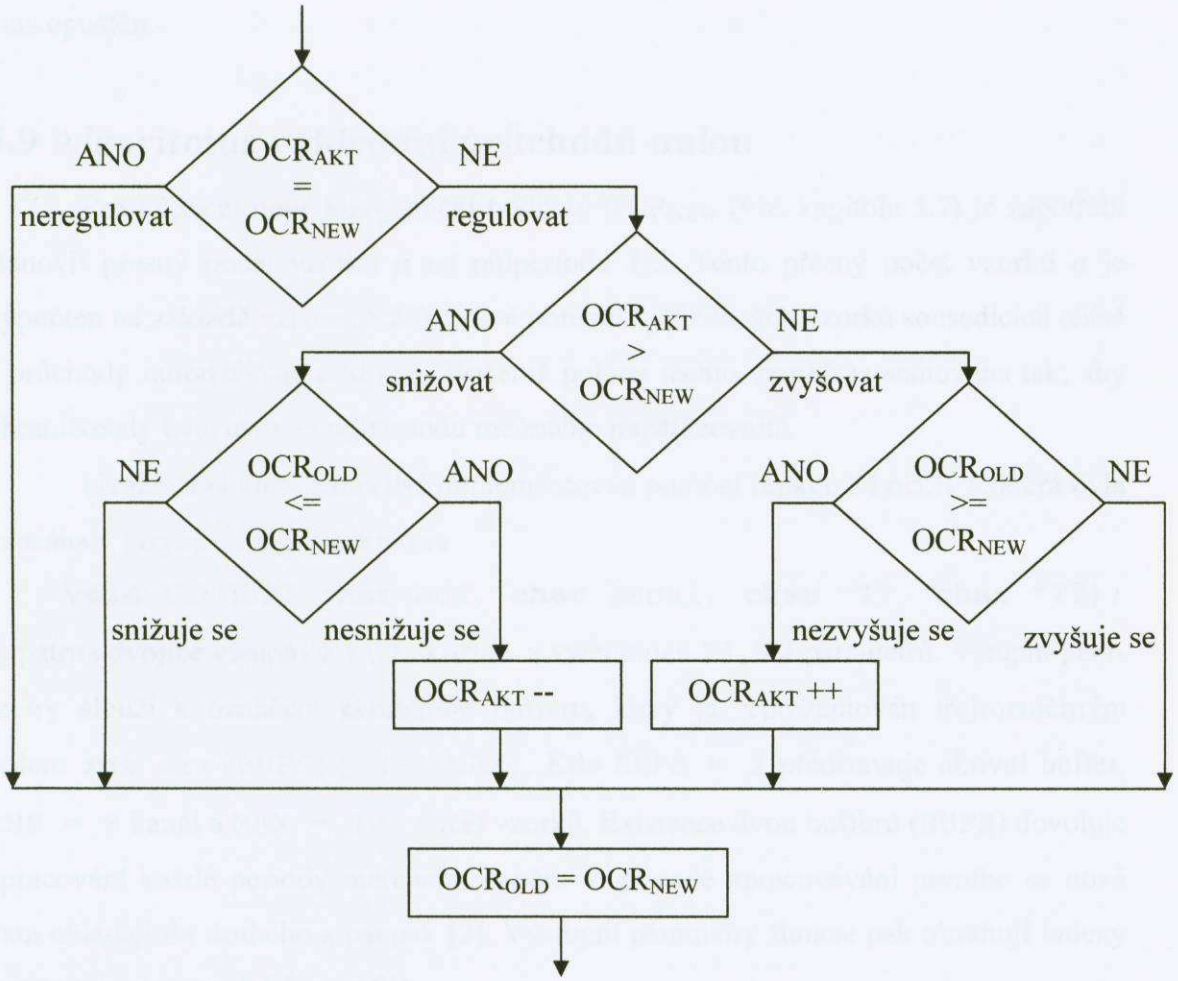
5.8 Regulace OCR v terminálu

Pro účely zachycení periody v terminálu byl stanoven jistý porovnávací algoritmus, který zajišťuje přesné zachycení právě jedné periody v požadovaném počtu vzorků N = 128. Vzhledem k tomu, že v rozvodné síti je hodnota kmitočtu dlouhodobě konstantní (za jistých okolností však může změnit svou hodnotu), byla stanovena velikost regulačního kroku OCR na jedničku. Takto malý regulační krok umožňuje velice přesné zachycení periody a navíc v případě přechodových jevů nezpůsobí výrazné chyby. Jistá nevýhoda je však při změně sledovaného kmitočtu, která je velice pomalá. To však v dané aplikaci nečiní žádné problémy z výše uvedeného důvodu.

Regulační algoritmus (obr. 5.15) pracuje celkem se třemi komparačními hodnotami OCR

- $\mathrm{OCR}_{\mathrm{AKT}}$ – je komparační hodnota, při které byla získána aktuální naměřená data.
- $\mathrm{OCR}_{\mathrm{NEW}}$ – je komparační hodnota nově vypočtená na základě aktuálních naměřených dat (viz. kapitola 5.7).

- OCR_{OLD} – je hodnota předcházející nově vypočtené komparační hodnoty OCR_{NEW}.



Obr. 5.15 Vývojový diagram regulace OCR.

Na počátku algoritmu dochází k porovnání aktuální komparační hodnoty OCR_{AKT} a nově vypočtené komparační hodnoty OCR_{NEW}. Pokud mají stejnou hodnotu je vše v pořádku a není potřeba vykonávat regulační zásah. Po přiřazení komparační hodnoty OCR_{NEW} do komparační hodnoty OCR_{OLD} je algoritmus opuštěn. V případě, že si však hodnoty OCR_{AKT} a OCR_{NEW} nejsou rovny, pokračuje algoritmus dále k určení směru regulace. Pokud je OCR_{AKT} větší než OCR_{NEW} je celkem jasné, že se hodnota OCR_{AKT} musí snižovat. V opačném případě pak zvyšovat. Dále před snížením (zvýšením) hodnoty OCR_{AKT} je třeba se ujistit, zda již ke snižování (zvyšování) nedochází. Proto je zde zařazena další podmínka, která zjišťuje zda předcházející hodnota nové OCR_{OLD} je menší nebo rovna (větší nebo rovna) nové hodnotě OCR_{NEW}. V případě nesplnění této podmínky dochází k snižování (zvyšování) hodnoty OCR_{AKT} a není tedy třeba provádět

regulační zásah. Pokud však podmínka je splněna, pak nedochází ke snižování (zvyšování) a je potřeba provést regulační zásah v podobě dekrementace (inkrementace) hodnoty OCR_{AKT}. Po přiřazení hodnoty OCR_{NEW} do hodnoty OCR_{OLD} je tento algoritmus opuštěn.

5.9 Algoritmus vyhledání průchodů nulou

Pro výpočet nové komparační hodnoty OCR_{NEW} (viz. kapitola 5.7) je zapotřebí stanovit přesný počet vzorků α na půlperiodu $T/2$. Tento přesný počet vzorků α je vypočten na základě vztahu (5.28), právě pomocí vyhledaných vzorků sousedících těsně s průchody nulou SP_{MIN} a SP_{MAX}. Přičemž poloha těchto vzorků je stanovena tak, aby ohraničovaly uvažovanou půlperiodu měřeného napětí zevnitř.

Uvažovaný algoritmus byl implementován pomocí funkce `FindSPT`, která byla napsána v jazyce C. Z její deklarace

```
void FindSPT(char buf, char kanal, char *T1, char *T2);
```

je patrná dvojice vstupních `buf`, `kanal` a výstupních `T1`, `T2` parametrů. Vstupní parametry slouží k označení aktuálního bufferu, který je reprezentován trojrozměrným polem `int arr[BUFS][CHS][MAX]`. Kde `BUFS = 2` představuje aktivní buffer, `CHS = 8` kanál a `MAX = 128` počet vzorků. Existence dvou bufferů (`BUFS`) dovoluje zpracování každé periody měřeného napětí. V případě zpracovávání prvního se nová data ukládají do druhého a naopak [2]. Výstupní parametry funkce pak obsahují indexy uvažovaných vzorků SP_{MIN} a SP_{MAX}.

Princip funkce `FindSPT` je vcelku prostý. Spočívá pouze v neustálém prohledávání zvoleného bufferu navzorkovaných dat, při kterém je nutno respektovat všechny možnosti výskytu hledané půlperiody. Výpis této funkce je možno nalézt v Příloze I. Funkce musí z principu používat pomocné počítadlo `i` (ř.3), jehož datovým typem je char.

První možností výskytu průchodu nulou je, že bude první vzorek nulový (podmínka na ř.5). Tato varianta sice není příliš pravděpodobná, nicméně je potřeba s ní počítat. V tomto případě je potřeba se dále rozhodnout, zda napětí bude klesat nebo stoupat. To je zajištěno další podmínkou (ř.6) podle znaménka 26 vzorku. 26 vzorek odpovídá právě jedné čtvrtině periody při nastaveném OCR pro přesných 50 Hz, ale skutečnému kmitočtu měřeného napětí 55 Hz. Pokud je tedy znaménko kladné, je možno uložit polohu prvního průchodu nulou (ř.7). Dále je patrné, že druhý hledaný

průchod nulou se bude nacházet někde za 26 vzorkem (ř.8). Tento druhý průchod je nalezen cyklem while (ř.9). V případě výskytu prvního záporného vzorku je cykl ukončen, přičemž poloha vzorku SP_{MAX} bude o krok zpět. Do podmínky cyklu je nutno zařadit i možnost, že daný průchod se v získaných datech nevyskytuje např. vlivem poruchy. Proto po dosažení 128 vzorku je cykl také opuštěn, jinak by vznikla nekonečná smyčka a činnost terminálu by zkolabovala. V případě záporného 26 vzorku je situace velice podobná (ř.13 až ř.19), pouze podmínka v cyklu while nyní testuje záporné hodnoty.

Pravděpodobnější možností je však výskyt prvního průchodu nulou mezi dalšími vzorky, tedy ne na první pozici. Tento stav nastává při nesplnění podmínky na (ř.5). I v tomto případě se odvíjí další postup vyhledávání dle polarity prvního vzorku (podmínka na ř.22). Pokud je kladný, je vyhledáván první záporný vzorek cyklem while (ř.23). Tento vzorek bude zároveň polohou vzorku SP_{MIN} (ř.25). Dále je zřejmé , že poloha dalšího průchodu nulou je dosti vzdálená od současné polohy, proto je možno ušetřit čas pomocí skoku o 26 vzorků (viz. výše). Poté je vyhledán první kladný vzorek dalším cyklem while (ř.27), přičemž poloha vzorku SP_{MAX} bude o krok zpět. V případě záporného prvního vzorku (ř.31) je situace analogická jako v případě kladného prvního vzorku, pouze jsou otočené podmínky v cyklech while.

Dále je třeba zajistit chybový výstup z funkce v případě, že nebyly nalezeny očekávané průchody nulou. Tato situace může nastat např. při poruše nebo při připojení analogového vstupu na stejnosměrné napětí. Dále může tento stav nastat v případech, kdy se vzorkuje přesně nebo více (velká vzorkovací perioda δ) a první průchod nulou leží těsně před prvním vzorkem. Pak se zákonitě může v navzorkovaných datech nalézt pouze jeden průchod. Tato situace je spíše výjimečná, ale je nutno na ni pamatovat. V těchto případech bude hodnota pomocného počítadla `i` větší nebo rovna hodnotě 128 a do výstupního parametru `T1` se uloží chybová hodnota 200.

Hlavní nevýhodou uvedené funkce FindSPT je, že neustále musí přistupovat a hodnotit jednotlivé vzorky. To samozřejmě neblaze přispívá k časovému vytížení řídícího mikrokontroléru. Délka zpracování této funkce je různá v závislosti na konkrétních naměřených vzorcích. V případě, že se nepodaří vyhledat průchody nulou je potřebný čas k jejímu vykonání maximální a činí 850 μs. V normálním režimu je pak hodnota tohoto časového intervalu typicky 450 μs. Další nevýhodou této funkce je, že nezohledňuje případné zákmity v okolí průchodů nulou. Pokud by se tyto případné zákmity objevily, došlo by k vyhledání falešného průchodu nulou. Vzhledem k tomu, že

v rozvodné síti se tyto zákmity neobjevují, nečiní tato nevýhoda žádné potíže v dané aplikaci. Odstranění těchto nedostatků by se dalo celkem jednoduše řešit např. způsobem uvedeným v závěru kapitoly 5.7.

5.10 Algoritmus určení přesného počtu vzorků

Tento algoritmus spočívá ve vyčíslení přesného počtu vzorků α na půlperiodu. Výpočet probíhá přesně dle výrazu v závorce vztahu (5.28). Jeho realizace je provedena pomocí funkce `GetSamplu`, která byla vytvořena v jazyce C. Deklarace této funkce má tvar

```
float GetSamplu(char buf, char kanal, char T1, char T2);
```

Návratová hodnota funkce má význam právě přesného počtu vzorků α na půlperiodu, proto je zde použit datový typ float. Význam dalších dvou parametrů `buf` a `kanal` je stejný jako u funkce `FindSPT` v předcházející kapitole. Jedná se tedy o určení aktivního befferu a měřeného kanálu. Pomocí poslední dvojice parametrů `T1` a `T2` se funkci předávají předem nalezené vzorky, které těsně ohraničují danou půlperiodu zevnitř.

```
1  float GetSamplu(char buf, char kanal, char T1, char T2)
2  {
3    float t1, t3;
4
5    t1 = (float)abs(arr[buf][kanal][T1])/
6         (abs(arr[buf][kanal][T1-1])+
7          abs(arr[buf][kanal][T1]));
8
9    t3 = (float)abs(arr[buf][kanal][T2])/
10        (abs(arr[buf][kanal][T2+1])+
11         abs(arr[buf][kanal][T2]));
12
13   return (t1 + (T2 - T1) + t3);
14 }
```

Z jejího výpisu je patrné rozdělení výpočtu do tří částí. V první z nich (ř.5 až ř.7) dochází k vyčíslení počtu vzorků na časový interval `t1`. V druhé části (ř.9 až ř.11) je pak toto vyčíslení pro časový interval `t3`. Na závěr jsou pak jednotlivé počty vzorků na dané časové intervaly t1, t2 a t3 sečteny (ř. 13) a výsledek je následně předán jako návratová hodnota funkce. Přičemž přesný počet vzorků na časový interval `t2` je vypočten přímo v příkazu return.

Časová náročnost uvedené funkce GetSamplu činí 185 μs. Tato hodnota je pro daný účel přijatelná a nečiní tedy žádné potíže. Je jí zdánlivě veliká hodnota je způsobena výpočtem v plovoucí řádové čárce (datový typ float).

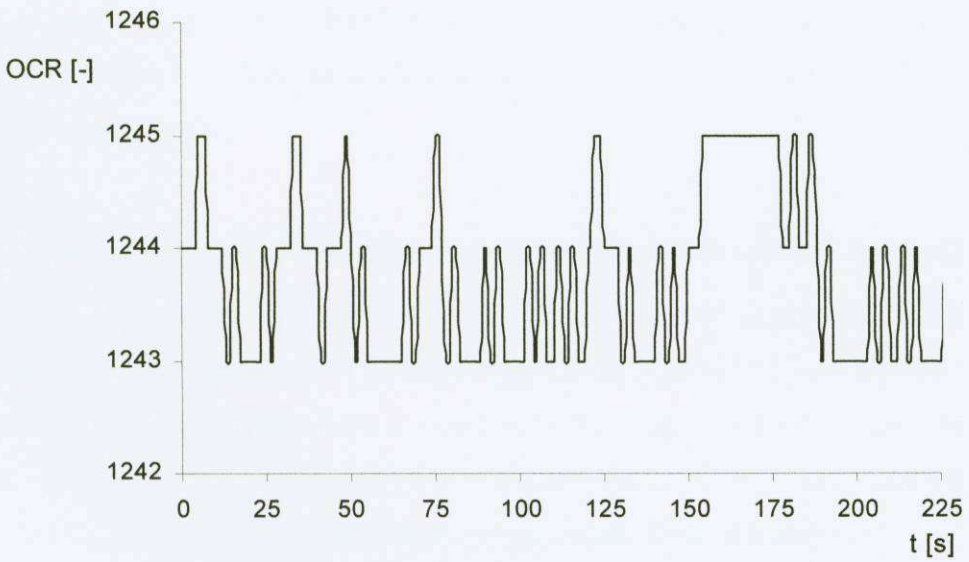
5.11 Algoritmus řízení vzorkování

Řízení vzorkování terminálu spočívá v softwareové implementaci postupu naznačeného v kapitole 5.8. Tento algoritmus je sestaven pomocí jazyka C, jehož výpis je možné nalézt v Příloze II.

Na začátku uvedeného výpisu (ř.2 a ř.3) jsou definovány dvě symbolické konstanty `CYKL` a `KANAL`. První z nich určuje počet měřících cyklů pro účely průměrování. Druhá pak určuje referenční kanál, podle kterého se bude synchronizovat činnost vzorkování. Dále následuje seznam definic použitých proměnných (ř.5 až ř.8). Jejich význam se ozřejmí v dalším popisu.

Po patřičných inicializacích (vyznačeno . . .) se běh programu musí zacyklit do nekonečné smyčky (ř.10 až ř.42), ve které se budou neustále zpracovávat naměřená data a synchronizovat vzorkování. Na počátku této smyčky se vynuluje pomocné počítadlo `Samplu` (ř.11). Poté následuje 5 měřících cyklů (ř.12 až ř.21). V těchto měřících cyklech je nutné nejdříve vyčkat na navzorkovaný buffer pomocí funkce `Wait4Buf` (ř.13) [2], se kterým je pak možno dále pracovat. Poté je možné vyhledat průchody nulou (ř.14) pomocí funkce `FindSPT` (viz. kapitola 5.9). Dále je otestován případný chybový stav (ř.15) této funkce, ke kterému může dojít např. při odpojeném vstupu atd. Jeli hodnota parametru `SPT1` různá od 200 dojde k výpočtu skutečného počtu vzorků na půlperiodu (ř.16) pomocí funkce `GetSamplu` (viz. kapitola 5.10) a přičtení této hodnoty do průměrovací sumy `Samplu`. V opačném případě (chybový stav) se předpokládá, že se vzorkuje přesně a přičítá se tedy hodnota 64 (ř.18). Dále následuje prostor (ř.19) pro vlastní zpracování naměřených dat terminálem (výpočet efektivních

hodnot, stanovení akčních zásahů, atd.). Na závěr každého měřícího cyklu musí být použitý buffer uvolněn pomocí funkce `FreeBuf` (ř.20) pro další vzorkování [2]. Po ukončení měřících cyklů je vypočtena (ř.23 a ř.24) nová komparační hodnota OCR_{NEW}, dle vztahu (5.32). Přičemž proměnná `Samplu` obsahuje součet jednotlivých skutečných počtů vzorků na půlperiodu za 5 měřících cyklů, proto je třeba tuto proměnnou podělit 5 (konstanta CYKL). Tím se stanoví průměrná komparační hodnota OCR_{NEW} za 5 měřících cyklů. Při předpokladu měření napětí o kmitočtu přesně 50 Hz (perioda 20 ms), by 5 měřících cyklů trvalo přesně 100 ms. Tento časový interval je pro regulaci prakticky neměnící se periody měřeného kmitočtu příliš krátký, proto se vypočtené nové komparační hodnoty OCR_{NEW} dále průměrují. Proto je nově spočtená hodnota OCR_{NEW} z (ř.23 a ř.24) dále přičtena do pomocné proměnné OCR_{SUM} (ř.25). Vlastní regulační zásah se provádí až ve smyčce na (ř.26 až ř.41), která je zpracována po každých 10 průchodech nekonečným cyklem. Celkem tedy po 50 měřících cyklech, což by odpovídalo při měření napětí o kmitočtu 50 Hz časovému intervalu 1 s. Vlastní regulační algoritmus je sestaven pomocí podmínek přesně dle vývojového diagramu na (obr. 5.15) z kapitoly 5.8. Přičemž proměnná AKT_{INT} v uvedeném výpisu odpovídá proměnné OCR_{AKT} ve vývojovém diagramu. Na závěr vykonání tohoto regulačního algoritmu je nutné vynulování proměnných OCR_{SUM} (ř.39) a počítadla `poc` (ř.40).



Obr. 5.16 Průběh OCR.

Ačkoliv se časový interval mezi dvěmi regulačními zásahy (1 s) zdá značný, je doba potřebná pro synchronizaci vzorkování při změně měřeného kmitočtu z 50 Hz na 51 Hz rovna 25 s. Tento časový interval je pro daný účel, kde se změny kmitočtu objevují spíše výjimečně, zcela postačující.

Uvedený algoritmu byl experimentálně ověřen na prototypu terminálu s vynikajícími výsledky zachyceni periody. Na obrázku (obr. 5.16) je znázorněn průběh komparační hodnoty OCR v průběhu činnosti terminálu. Z grafu je patrné, že maximální odchylka OCR činila ± 1, což odpovídá velice přesnému zachycení.

6. Měření efektivní hodnoty napětí

Tato kapitola by se dala rozdělit na dvě dílčí části. V první jsou uvedeny obecné metody měření efektivní hodnoty [12][14], ať již analogové či digitální. Druhá část je pak věnována implementaci algoritmu výpočtu přesné efektivní hodnoty napětí (true RMS) v terminálu TAFT 112®.

6.1 Problematika měření efektivní hodnoty napětí

Efektivní hodnota měřeného napětí vyjadřuje výkonové poměry měřeného napětí. S její definicí (6.1) jsme se již podrobně seznámili v kapitole 2.5.

$$U = \sqrt{\frac{1}{T}\int_0^T u^2(t)\mathrm{d}t} \ [\mathrm{V}] \tag{6.1}$$

Uvážíme-li, že obecný periodický signál *u(t)* je na základě Fourierovy harmonické analýzy součtem harmonických kmitů [12], lze vztah pro výpočet efektivní hodnoty napětí převést na tvar (6.2).

$$U = \sqrt{U_0^2 + U_1^2 + U_2^2 + \ldots + U_n^2} \tag{6.2}$$

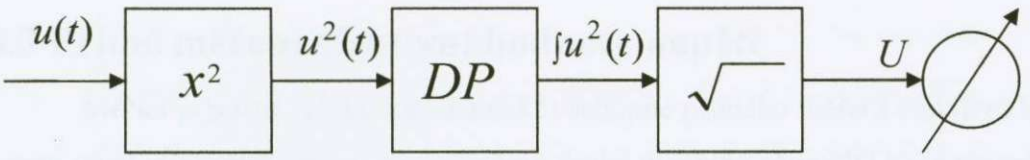
Přičemž U_0 odpovídá stejnosměrné složce napětí, U_1 pak efektivní hodnotě základní harmonické a U_2 až U_n jsou efektivní hodnoty vyšších harmonických. Dále víme, že kmitočet *n*-té harmonické odpovídá *n* násobku základní harmonické. Z toho plyne, že měřící přístroje určené pro měření efektivní hodnoty napětí neharmonických napětí musí měřit efektivní hodnotu jednotlivých harmonických složek, proto musí mít dostatečný kmitočtový rozsah.

6.2 Převodníky efektivní hodnoty na stejnosměrná napětí

Tyto převodníky zajišťují měření efektivní hodnoty napětí přístrojům, které neumožňují její přímé měření [12]. Různé typy těchto převodníků jsou schopny přesně převést neharmonické napětí pouze v jistém rozsahu. Toto omezení je dané především četností výskytu harmonických složek v měřeném napětí. Čím se tedy měřené napětí více liší od harmonického, tím větší chyby se převodník zpravidla dopouští.

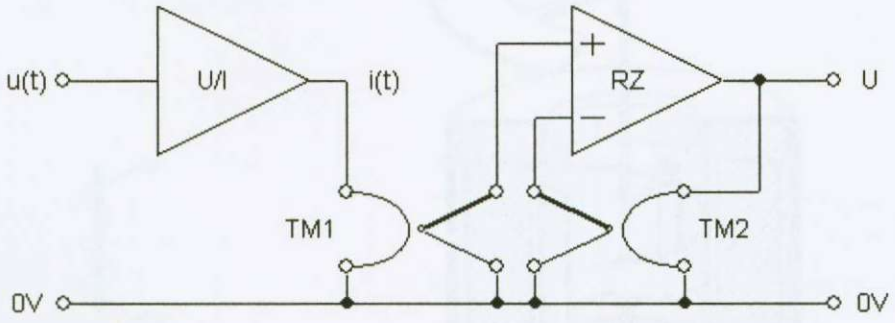
Na (obr. 6.1) je znázorněno blokové schéma explicitního převodníku. Tento převodník pracuje přesně podle definice (6.1). Druhou mocninu i odmocninu můžeme realizujeme pomocí vhodně zapojených exponenciálních a logaritmických zesilovačů. Funkci dolní propusti zastoupí integrační zesilovač. Vlastnosti celého převodníků jsou

především závislé na kvalitě exponenciálního a logaritmického zesilovače, které musí být vždy teplotně stabilizovány. Obvykle se explicitní převodník vyznačuje malým dynamickým rozsahem [12].



Obr. 6.1 Blokové schéma explicitního převodníku.

Dále se v měřících přístrojích často využívá zpětnovazební převodník s teplotně závislými prvky (obr. 6.2). V převodníku je použita dvojice shodných tepelně izolovaných termoelementů TM1 a TM2. TM1 je ohříván proudem *i(t)* úměrným vstupnímu napětí *u(t)* prostřednictvím převodníku napětí/proud. Oproti tomu TM2 je ohříván proudem *I*, který je úměrný výstupnímu napětí *U*. Výstupy termoelementů jsou připojeny na vstupy rozdílového zesilovače RZ, který zajišťuje aby ohřev obou prvků byl stejný. V tom případě bude dle definice efektivní hodnoty napětí odpovídat velikost stejnosměrného napětí *U* efektivní hodnotě vstupního střídavého napětí *u(t)*.



Obr. 6.2 Převodník s teplotně závislými prvky.

Jako termoelementů lze využít nepřímo vyhřívané termočlánky, popřípadě polovodičových prvků obsažených zpravidla ve specializovaných integrovaných obvodech. S převodníky tohoto typu lze dosáhnou přesnosti až 0,01% na síťových kmitočtech, 1% asi do 500Hz a 5% do kmitočtu 100MHz [12].

Pro praktické použití vyrábí např. firma Analog Devices® integrované převodníky na efektivní hodnotu AD636, AD736 atd. Jejich výhodou je dobrá přesnost a jednoduchost začlenění do měřícího obvodu. Nevýhodou je jejich poměrně vysoká cena (AD736 za 325Kč k listopadu 2005).

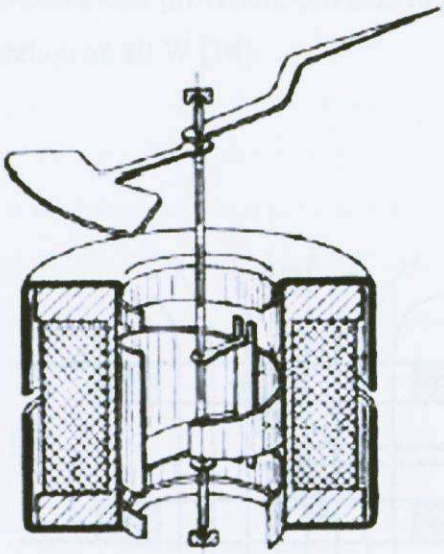
6.3 Přímé měření efektivní hodnoty napětí

Měřící přístroje (elektromechanické) schopné přímého měření efektivní hodnoty napětí musí mít pohybový moment úměrný druhé mocnině okamžité hodnoty měřeného napětí a současně jejich dynamické vlastnosti musí vytvořit střední hodnotu (6.1) [12][14].

Feromagnetická soustava

Princip feromagnetického měřícího ústrojí spočívá v působení sil magnetického pole cívky na dvojici feromagnetických plíšků (obr. 6.4) [12]. Jeden plíšek je připevněn k cívce, druhý otočný je spojen s ukazatelem.

Obr. 6.3 Značka soustavy.



Obr. 6.4 Feromagnetická soustava [14].

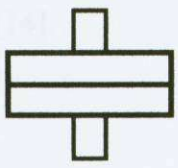
Po připojení proudu I se plíšky souhlasně zmagnetují a otočný plíšek se začne vzdalovat od pevného. Pohybový moment M_P takto vyvolaný je úměrný změně energie magnetického pole W_M cívky

$$M_P = \frac{\mathrm{d}W_M}{\mathrm{d}\varphi} = \frac{1}{2}\frac{\mathrm{d}L}{\mathrm{d}\varphi} I^2 \ [\mathrm{Nm}] \qquad (6.3)$$

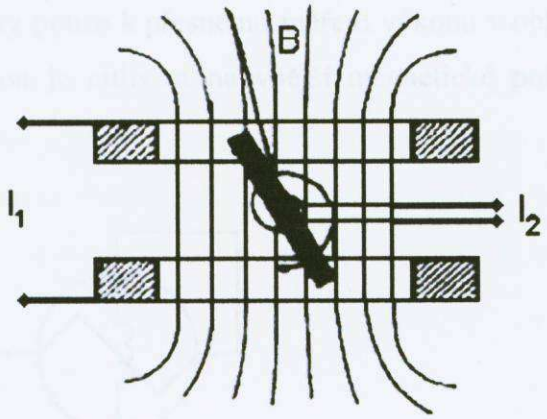
Kde φ představuje úhel natočení. Pokut by výraz $\mathrm{d}L/\mathrm{d}\varphi$ byl roven konstantě měla by stupnice kvadratický průběh. Jistým způsobem ji lze linearizovat vhodným tvarem plíšků, počátek stupnice však bude vždy zhuštěný. I v případě buzení cívky střídavým proudem libovolného průběhu *i(t)* bude pohybový moment M_P dán vztahem (6.3), kde proud I je nyní vyjádřen svou efektivní hodnotou. Ideální feromagnetické ústrojí by měřilo efektivní hodnotu měřené veličiny bez ohledu na její kmitočet a tvar. Vlivem ztrát způsobených vířivými proudy však měří tato soustava efektivní hodnotu jen v jistých mezích s danou přesností (max. stovky Hz) [14]. Uložení otočné části může být hrotové nebo na napjatých vláknech. V případě hrotového uložení je direktivní moment vyvoláván spirálovitou pružinou. Napjatých vláken se užívá pouze pro účely přesných laboratorních přístrojů. Tlumení je vzduchové pomocí křidélka. Tato soustava se dále vyznačuje velkou vlastní spotřebou a používá se hlavně pro měření na síťových kmitočtech [14].

Voltmetr se s feromagnetickým měřícím ústrojím realizuje za pomoci předřadného rezistoru. Převážně se vyrábějí v rozváděčovém provedení pro rozsah napětí 15 až 600 V, s třídou přesnosti 2,5 a vlastní spotřebou až 10 W [14].

Elektrodynamická soustava



Obr. 6.5 Značka soustavy.



Obr. 6.6 Princip elektrodynamické soustavy [12].

Elektrodynamická soustava (obr. 6.6) využívá vzájemné silové působení dvou cívek, kterými protéká el. proud [12][14]. Soustava je tvořena pevnou cívkou (proudovou) rozdělenou z důvodu homogenity na dvě části a z pohyblivé cívky (napěťové), která je spojená s ukazatelem. Direktivní moment pohyblivé cívky je dán spirálovitými pružinami, přes které je zároveň zajištěno její napájení.

Pohybový moment M_P napěťové cívky vyvolaný silovým působením magnetického pole, vyvolaného průchodem proudu I_1 pevnou cívkou a I_2 pohyblivou cívkou, je dán změnou magnetické energie W_M při změně výchylky φ (6.5). Přičemž pro celkovou energii magnetického pole těchto dvou cívek platí.

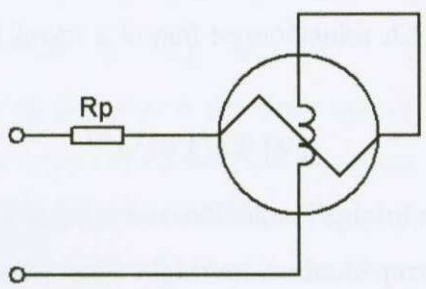
$$W_M = \frac{1}{2} L_1 I_1^2 + \frac{1}{2} L_2 I_2^2 + M I_1 I_2 \text{ [J]} \tag{6.4}$$

Kde L_1 představuje indukčnost pevné cívky, L_2 indukčnost pohyblivé cívky a M pak vzájemnou indukčnost obou cívek. Pohybový moment M_P pak bude.

$$M_P = \frac{\mathrm{d}W_M}{\mathrm{d}\varphi} = I_1 I_2 \frac{\mathrm{d}M}{\mathrm{d}\varphi} \text{ [Nm]} \tag{6.5}$$

Ze vztahu (6.5) je patrné, že s výchylkou se mění pouze vzájemná indukčnost M. Při vhodném geometrickém uspořádání, tak aby platilo $\mathrm{d}M/\mathrm{d}\varphi$ = konst., lze dosáhnout lineárního uspořádání stupnice. Pokut budou proudy I_1 a I_2 střídavé neharmonické, bude výsledný pohybový moment úměrný součinu jejich středních hodnot. V případě, že budou mít stejnou velikost bude tento moment úměrný efektivní hodnotě tohoto proudu.

Této soustavy se využívá prakticky pouze k přesnému měření výkonu v oblasti síťového kmitočtu. Její hlavní nevýhodou je citlivost na vnější magnetické pole a vysoká cena [14].

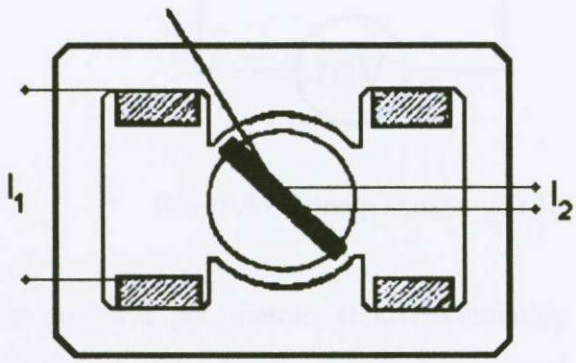


Obr. 6.7 Zapojení elektrodynamického voltmetru.

Elektrodynamické voltmetry lze zapojit sériovým spojením pevné a pohyblivé cívky za pomoci předřadného odporu (obr. 6.7). Cívkami pak teče stejný proud a soustava udává efektivní hodnotu měřeného napětí.

Ferodynamická soustava

Její princip je totožný jako u soustavy elektrodynamické [12][14]. Rozdíl spočívá pouze v přítomnosti magnetického obvodu (obr. 6.9).



Obr. 6.8 Značka soustavy. Obr. 6.9 Princip elektrodynamické soustavy [12].

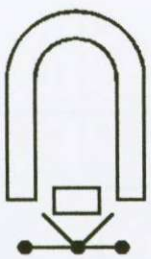
Magnetický tok je buzen pevnou cívkou a prakticky nevybočuje svou dráhou z magnetického obvodu. Tím získává soustava větší pohybový moment M_P a vyznačuje se větší odolností vůdčí vnějšímu magnetickému poli.

Magnetoelektrická soustava s termočlánkem

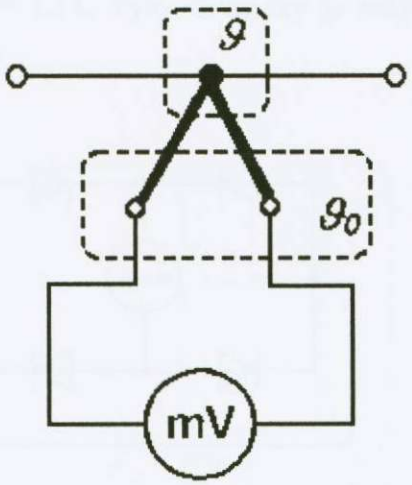
Termočlánek je využíván k přeměně střídavého elektrického proudu na stejnosměrné napětí [12][14], které se pak snadno měří magnetoelektrickým voltmetrem (obr. 6.11). Termočlánek je přímo ohříván topným vodičem, kterým protéká měřený proud. V případě rozdílnosti teplot hrotu a konců termočlánku $\Delta\vartheta$, se objeví na jeho koncích termoelektrické napětí U_t.

$$U_t = k_t \Delta\vartheta \text{ [V]} \tag{6.6}$$

Kde k_t představuje konstantu daného termočlánku. Teplotní rozdíl $\Delta\vartheta$ bude dále úměrný teplu, které se na něm vyzáří, tedy efektivní hodnotě proudu procházejícím topným vodičem.



Obr. 6.10 Značka soustavy.



Obr. 6.11 Princip soustavy [12].

Voltmetry této konstrukce lze používat pro měření efektivní hodnoty napětí kmitočtu až řádově desítek kHz (vliv povrchového jevu) s nevalnou přesností. Tento způsob měření efektivní hodnoty napět se dnes již prakticky nepoužívá [12].

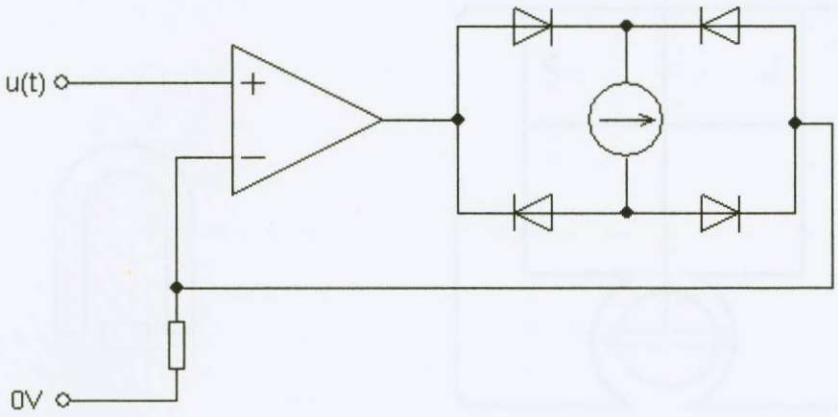
6.4 Nepřímé měření efektivní hodnoty napětí

Nepřímé měření efektivní hodnoty napětí se dá realizovat pomocí stejnosměrného voltmetru ať již analogového nebo digitálního a převodníku efektivní hodnoty napětí na stejnosměrné napětí (viz. kapitola 6.2). Tyto převodníky mohou být realizovány buď výpočtovým způsobem pomocí operačních zesilovačů nebo mohou pracovat na tepelném principu. Bohužel mají jistá omezení týkající se maximálního kmitočtu měřeného napětí. Dále špatně registrují krátké napěťové špičky. Přístroje vybavené těmito převodníky nesou označení true RMS (True Root Mean Square) což je zařazuje do vyšší cenové i kvalitnější kategorie.

Další možností nepřímého měření efektivní hodnoty napětí je doplnění stejnosměrného voltmetru měřícím usměrňovačem [12]. Voltmetr v tomto případě měří střední hodnotu napětí, přičemž jeho stupnice je cejchována v efektivních hodnotách. Souvislost efektivní U a střední U_{AV} hodnoty napětí udává činitel tvaru K_T

$$K_T = \frac{U}{U_{AV}} [-] \tag{6.7}$$

Hodnota tohoto činitele je závislá na konkrétním průběhu měřeného napětí a pro harmonický sinusový průběh má hodnotu K_T = 1,11. Tyto voltmetry je tedy možné používat pouze pro měření harmonických napětí.



Obr. 6.12 Zapojení aktivního usměrňovače.

Obrázek (obr. 6.12) znázorňuje zapojení aktivního dvoucestného usměrňovače vhodného např. pro magnetoelektrický systém (viz. níže). Zapojením usměrňovacího můstku do zpětnovazební větve operačního zesilovače docílíme lineární převodní charakteristiku usměrňovače.

Magnetoelektrická soustava

Funkce této soustavy spočívá v silovém působení magnetického pole na vodič, kterým protéká elektrický proud I [12][14]. Základ soustavy tvoří permanentní magnet s pólovými nástavci. V jejich dutině je umístěna měřící cívka navinutá na hliníkovém rámečku (obr. 6.14). Napájení cívky je zajištěno pomocí dvojice spirálovitých pružin, které zároveň vytvářejí direktivní moment M_D. Síla působící na jednu stranu cívky o N závitech bude

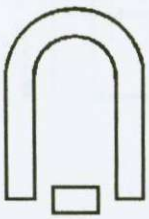
$$F = N\,I\,l\,B \text{ [N]} \tag{6.7}$$

kde l je délkou rámečku kolmá na směr působení síly F. Pohybový moment způsobený působením síly F o rameni r (poloměr cívky) bude roven

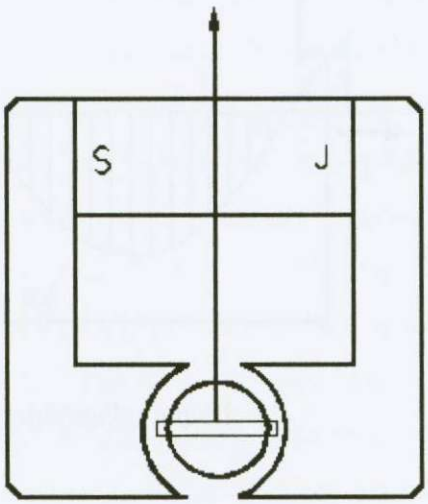
$$M_P = 2\,F\,r = 2\,N\,I\,l\,B\,r \text{ [Nm]} \tag{6.8}$$

Z tohoto vztahu vyplývá, že pohybový moment M_P a tedy i výchylka přístroje je přímo úměrná protékajícímu proudu I včetně jeho polarity. Jestliže je proud procházející cív-

kou obecného periodického průběhu *i(t)*, pak otočná část měřícího ústrojí vlivem setrvačnosti a tlumení vytvoří jeho střední hodnotu.



Obr. 6.13 Značka soustavy.



Obr. 6.14 Princip soustavy [12].

Magnetoelektrické měřící přístroje patří k nejpoužívanějším typům analogových přístrojů. Mezi jejich hlavní přednosti patří malá vlastní spotřeba, vysoká přesnost a jednoduchost.

6.5 Číslicové vyhodnocení efektivní hodnoty napětí

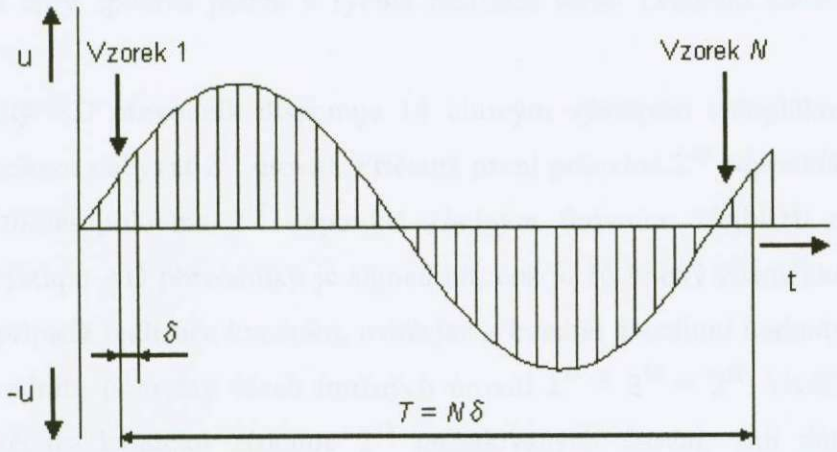
Číslicovým zpracováním efektivní hodnoty napětí rozumíme její výpočet na základě získané posloupnosti dat $X[N]$ z AD převodu [12]. Vzorkování je prováděno v reálném čase RTS (viz. kapitola 5.2), tak aby celá perioda měřeného napětí T byla přesně zachycena (obr. 6.15). Délka posloupnosti N, tedy počet vzorků, se volí s ohledem na Nyquistův, Shannon-Kotělnikovův vzorkovací teorém (5.1) a tak aby platilo

$$N = 2^n \tag{6.9}$$

Kde n je kladné přirozené číslo. Touto volbou si zajistíme rychlou realizaci dělení a násobení pomocí logických posuvů. Souvislost mezi periodou vzorkování δ a periodou měřeného signálu T je pak dána vztahem

$$T = N\,\delta \text{ [s]} \tag{6.10}$$

Jako vhodným kompromisem mezi výpočetní rychlostí procesoru a splněním vzorkovacího teorému bylo zvoleno na terminálu $N = 128$.



Obr. 6.15 Vzorkování měřeného napětí.

Pro určení efektivní hodnoty napětí z diskrétních hodnot posloupnosti *X[N]* vyjdeme z definičního vztahu (6.11)

$$U = \sqrt{\frac{1}{T}\int_0^T u^2(t)\mathrm{d}t} \text{ [V]} \tag{6.11}$$

V tomto vztahu nahradíme integrál sumou, diferenciál d*t* pak přejde v periodu vzorkování δ a za periodu měřeného napětí T dosadíme ze vztahu (6.10).

$$U = \sqrt{\frac{1}{N\delta}\sum_{i=1}^{N} u_i^2\,\delta} \tag{6.12}$$

Kde napětí u_i představuje *i*-tý vzorek z datové posloupnosti *X[N]*. Po vykrácení periody vzorkování δ z (6.12) dostáváme výsledný vztah pro výpočet efektivní hodnoty napětí z diskrétní datové posloupnosti

$$U = \sqrt{\frac{1}{N}\sum_{i=1}^{N} u_i^2} \tag{6.13}$$

Při odvození tohoto vztahu byla předpokládána schodovitá aproximace signálu, tedy konstantní úroveň u_i po dobu vzorkovací periody δ [12].

6.6 Algoritmus výpočtu ef. hodnoty napětí terminálem

Za účelem dosažení co největší efektivity a rychlosti výpočtu efektivní hodnoty napětí terminálem byl tento algoritmus realizován pomocí funkce `SumKv` (`SumKvM`) vytvořené v assembleru. Výpis funkce `SumKvM` je možné nalézt v Příloze III. Výpočet probíhá přesně dle (6.13), přičemž odmocnina a dělení *N* není realizována touto funkcí.

Její podstata tedy spočívá pouze v rychlé realizaci sumy kvadrátů navzorkovaných napětí.

Použitý AD převodník disponuje 14 bitovým výstupem v doplňkovém kódu, může tedy celkem nabývat 2^{14} úrovní. Přičemž první polovina 2^{13} odpovídá záporným úrovním a druhá polovina 2^{13} odpovídá kladným úrovním. Nejbližší datový typ vyhovující výstupu AD převodníku je signed int, což je 16 bitový znaménkový datový typ [10]. V případě realizace kvadrátu, uvažujeme kvadrát absolutní hodnoty napětí, to odpovídá kvadrátu poloviny všech možných úrovní $2^{13} * 2^{13} = 2^{26}$. Uvážíme-li dále sumu 128 těchto kvadrátů získáme 2^{33} požadovaných úrovní. Ani datový typ s největším rozsahem unsigned long (2^{32} úrovní) však není schopen tento požadavek uspokojit. Tato situace by nastala např. v případě měření obdélníkového napětí s amplitudou odpovídající maximální možné měřené hodnoty. Pokud bychom měřili sinusové napětí s amplitudou odpovídající maximu, bude počet požadovaných úrovní roven téměř 100 % maximálního rozsahu datového typu unsigned long. Z tohoto důvodu provádí funkce realizující sumu kvadrátů celočíselné dělení dvěma, čímž se zabezpečí možnost použití standardního datového typu unsigned long se zanedbatelnou ztrátou přesnosti. Díky tomu je dále zabezpečeno, že suma nepřeteče ani při měření obdélníkového napětí s maximální úrovní.

V průběhu tvorby obslužného softwaru terminálu dále vyvstal požadavek na vyhledání maximálního vzorku měřeného napětí pro účely automatické volby měřících rozsahů. Vzhledem k tomu, že není nutné toto vyhledávání provádět vždy, byly napsány dvě funkce. Funkce `SumKv`, která realizuje pouze sumu kvadrátů a funkce `SumKvM`, která realizuje sumu kvadrátů a navíc vyhledává maximální vzorek. Z jejich deklarací

```
unsigned long SumKv(int buf[]);
unsigned long SumKvM(int buf[], int *max);
```

jsou patrné jak návratové typy unsigned long, tak předání polí navzorkovaných dat pomocí ukazatelů int *. Funkce `SumKvM` obsahuje navíc ukazatel pro předání maximálního vzorku.

Výpočet efektivní hodnoty napětí pak spočívá v podělení sumy kvadrátů 64 (`SumKv` dělí dvěma, celkem tedy 128) a provedením odmocniny. Odmocnina i dělení jsou realizovány přesně v plovoucí řádové čárce pomocí datového typu float. V následující ukázce je naznačen způsob výpočtu efektivní hodnoty napětí prvního kanálu v nultém bufferu.

```
    ...
extern int arr[BUFS][CHS][MAX];
unsigned int Uef, MaxSamp;
    ...
Uef = (unsigned int)sqrt(
      (float)SumKvM(&arr[0][1][0], &MaxSamp)/64));
    ...
```

Pole arr představuje buffer navzorkovaných dat, kde `BUFS = 2` představuje buffer, `CHS = 8` kanál a `MAX = 128` počet vzorků. Existence dvou bufferů dovoluje zpracování každé periody měřeného napětí [2]. V případě zpracovávání prvního se nová data ukládají do druhého a naopak.

Funkce `SumKv` a `SumKvM` jsou umístěny v samostatné assemblerovské jednotce, přičemž v Příloze III je uvedena pouze funkce `SumKvM`. Začátek jednotky obsahuje seznam definic alternativních jmen registrů (ř.1 až ř.17). Tyto definice výrazně zlepšují čitelnost a orientaci v programu. Dále následuje inicializační část (ř.19 až ř.30). V ní se předají adresy parametrů (buffer a maximum) příslušným ukazatelům (X a Z). Následuje vynulování proměnných sumy, maxima a pomocného bitu T, který zde slouží jako pomocný 33. bit sumy. Na závěr inicializační části je aktualizováno počítadlo hodnotou 128. Od návěští `SmyckaM` se provádí vlastní výpočet sumy kvadrátů. Nejdřív dochází k nahání zpracovávaného vzorku z pamětí (ř.32 a ř.33). Pokut je vzorek záporný (podmínka na ř.34), je vytvořena jeho absolutní hodnota. Poté následuje porovnání vzorku s dočasným maximem (ř.40 až ř.46). Dále dochází k vyčíslení vlastního kvadrátu. Jeho realizace je provedena pomocí tří součinů *MUL* a vhodně volených součtů. Pak již stačí pouze přičíst k sumě vypočtený kvadrát (ř.61 až ř.66). Při tomto přičtení nesmíme zapomenout na případné přetečení a nastavení 33. bitu T (ř. 65 a ř.66). Podmínkou na (ř.68 a ř.69) je smyčka zacyklena pro uvažovaných 128 průchodů, tedy zpracování přesně jedné navzorkované periody. Po opuštění smyčky dojde k uložení maxima pomocí ukazatele (r.71 a ř.72). Na závěr je součin z výše popsaných důvodů celočíselně podělen dvěma (ř.74 až ř.80) s přihlédnutím k případnému nastavení 33. bitu T. Funkce je na závěr ukončena instrukcí *RET*.

Funkce `SumKv` je prakticky identická s funkcí `SumKvM`. Jediný rozdíl spočívá v nepřítomnosti detekce maxima. Z tohoto důvodu není uveden její výpis v Příloze III.

Díky použití assembleru při tvorbě funkcí `SumKv` a `SumKvM` bylo docíleno jejich velice rychlé zpracování (tab. 6.1).

	Čas [μs]
SumKv	276
SumKvM	327

Tab. 6.1 Časy výpočtu funkcí.

V (tab. 6.2) jsou dále uvedeny časy výpočtu efektivní hodnoty napětí s využitím těchto funkcí.

	Čas [μs]
U_{EF} (SumKv)	711
U_{EF} (SumKvM)	762

Tab. 6.2 Časy výpočtu napětí.

Z uvedených údajů jasně vyplývá, že čas potřebný k výpočtu efektivní hodnoty napětí je zhruba o 435 μs delší než čas potřebný k vykonání funkcí `SumKv` (`SumKvM`). V tomto časovém úseku dochází k realizaci odmocniny a dělení v plovoucí řádové čárce. Dalšího zrychlení výpočtu by se dalo docílit napsáním rychlé assemblerské funkce pro realizaci celočíselné odmocniny. Tím by se onen časový úsek 435 μs podstatně zkrátil, ale na druhou stranu by poklesla přesnost výpočtu a to především při měření malých napětí.

Závěr

Projekt vývoje terminálu TAFT 112® je dlouhodobá a problematická záležitost. Vlastní vývoj by se dal rozčlenit na tři samostatné části. V první řadě je potřeba učinit důkladnou analýzu dané problematiky. Jedná se především o kvantitativní vyjádření požadavků na danou funkci. Na základě této analýzy je nutno dále vyvinout konkrétní hardwarové řešení terminálu. To spočívá ve výběru vhodných elektronických prvků a jejich správné začlenění do elektroniky terminálu. Posledním úkolem při realizaci takovéhoto typu zařízení je vytvoření řídícího softwaru, který pak „oživí" terminál. Hlavní problém při tomto vývoji zpravidla bývá v dodatečných požadavcích na danou funkci nebo v problémech, které se začnou projevovat až v závěrečné fázi. Řešení těchto nedostatků pak zpravidla spočívá v zásahu do již vyrobeného hardwaru a tím i značné zpoždění celého projektu.

Prvním úkolem této bakalářské práce bylo zajištění automatické synchronizace vzorkování (viz. kapitola 5), tak aby každá perioda měřeného napětí byla zachycena v požadovaném počtu vzorků. Tento počet vzorků byl stanoven až v průběhu tvorby softwaru jako kompromis mezi splněním Nyquistova, Shannon-Kotělnikovova vzorkovacího teorému a konečné výpočetní rychlosti použitého řídícího mikrokontroléru ATMEL® ATmega 128 na hodnotu 128 vzorků. Tohoto úkolu bylo vcelku elegantně docíleno díky značné univerzalitě řídícího mikrokontroléru s přesnosti zachycení periody ± 0,04 Hz na kmitočtu 50 Hz. Ona elegance spočívala v hardwarovém využití automatického spouštění vzorkování a čtení navzorkovaných dat. Jistým nedostatkem by se mohlo jevit čistě softwarové detekování průchodů nulou za účelem výpočtu pro řízení hardwarového spouštění vzorkování. To samozřejmě neblaze přispívá k časovému vytížení řídícího mikrokontroléru. Řešením by bylo např. přidání napěťového komparátoru, který by zastával funkci detektoru průchodů nulou. To by však znamenalo značný zásah do hardwaru terminálu a výrazné zpoždění projektu.

Druhým úkolem pak byla realizace rychlého výpočtu přesné efektivní hodnoty napětí na základě navzorkovaných dat. Pro daný výpočet byla sestavena dvojice rychlých assemblerovských funkcí SumKv a SumKvM (viz. kapitola 6). Obě funkce realizují rychlou sumu kvadrátů z naměřených dat, přičemž funkce SumKvM navíc vyhledá maximální vzorek. Výsledná efektivní hodnota je následně stanovena jako odmocnina ze sumy kvadrátů přesně v plovoucí řádové čárce. Tímto způsobem se podařilo docílit maximální rychlosti výpočtu 762 μs.

Seznam použité literatury

[1] ORSÁGOVÁ J.: Rozvodná zařízení. Brno, Elektronická skripta VUT, 2003.

[2] Firemní materiály EGC-EnerGoConsult ČB s.r.o.

[3] ZAJÍC J.: Fyzika II (Elektřina a magnetismus). Pardubice, EMADO, 2004.

[4] LÁNÍČEK R.: Elektronika, obvody, součástky, děje. Praha, BEN, 1998.

[5] http://www.atmel.com

[6] MATOUŠEK D.: Práce s mikrokontroléry ATMEL AVR. Praha, BEN, 2003.

[7] Data sheet: ATmega 128

[8] Konstrukční Elektronika A Radio: 1/2003. Praha, AMARO s.r.o., 2003.

[9] BURKHARD M.: C pro mikrokontroléry. Praha, BEN, 2003.

[10] HEROUT P.: Učebnice jazyka C. České Budějovice, Kopp, 2001.

[11] Data sheet: AD7865

[12] GESCHEIDTOVÁ E., REZ J., STEINBAUER M.: Měření v elektrotechnice. Brno, Nakladatelství VUTIUM, 2002.

[13] ĎAĎO S., VEDRAL J.: Číslicové měření, přístroje a metody. Praha, Vydavatelství ČVUT, 2002.

[14] Elektrotechnická měření. Praha, BEN, 2002.

Seznam příloh

Příloha I: Výpis funkce FindSPT

```
1  void FindSPT(char buf, char kanal, char *T1, char *T2)
2  {
3    char i = 1;
4
5    if (arr[buf][kanal][0] == 0) {                    //první je 0
6      if (arr[buf][kanal][26] > 0)  {                 //+
7        *T1 = 0;                                      //SP_MIN
8        i = 26;
9        while ((arr[buf][kanal][i] >= 0) && (i < 128))
10         i++;
11       *T2 = i - 1;                                  //SP_MAX
12     }
13     else {                                          //-
14       *T1 = 0;                                      //SP_MIN
15       i = 26;
16       while ((arr[buf][kanal][i] <= 0) && (i < 128))
17         i++;
18       *T2 = i - 1;                                  //SP_MAX
19     }
20   }
21   else {                                            //první není 0
22     if (arr[buf][kanal][0] > 0) {                   //+
23       while ((arr[buf][kanal][i] > 0) && (i < 128))
24         i++;
25       *T1 = i;                                      //SP_MIN
26       i += 26;
27       while ((arr[buf][kanal][i] <= 0) && (i < 128))
28         i++;
29       *T2 = i - 1;                                  //SP_MAX
30     }
31     else {                                          //-
32       while ((arr[buf][kanal][i] < 0) && (i < 128))
```

```
33          i++;
34        *T1 = i;                                    //SPMIN
35        i += 26;
36        while ((arr[buf][kanal][i] >= 0) && (i < 128))
37          i++;
38        *T2 = i - 1;                                //SPMAX
39      }
40    }
41
42    if (i >= 128)
43      *T1 = 200;
44  }
```

Příloha II: Výpis algoritmu řízení vzorkování

```
1       ...
2   #define CYKL   5                                  //měřící cykly
3   #define Kanal  0                                  //ref. kanál
4       ...
5   char b, i, poc = 0, SPT1 = 0, SPT2 = 0;
6   unsigned int OCRsum, OCRnew, OCRold;
7   extern unsigned int OCRakt;
8   float Samplu;
9       ...
10    for (;;){                                       //nek. cykl
11      Samplu = 0;
12      for (i = 0; i < CYKL; i++){                   //5 mer. cyklu
13        b = Wait4Buf();                             //ceka na buf.
14        FindSPT(b, Kanal, &SPT1, &SPT2);            //průchody 0
15        if (SPT1 != 200)                            //α
16          Samplu += GetSamplu(b, Kanal, SPT1, SPT2);
17        else
18          Samplu += 64;
19            ...
20        FreeBuf(b);
21      }
22
23      OCRnew = (unsigned int)fround((AktInt+1)*
24                Samplu/(CYKL*64))-1;
25      OCRsum += OCRnew;
26      if (++poc == 10){
27        OCRnew = OCRsum / 10;
28
29        if (AktInt != OCRnew)
30          if (AktInt > OCRnew){
31            if (OCRold <= OCRnew)
32              AktInt--;
33          }
```

```
34          else{
35            if (OCRold >= OCRnew)
36              AktInt++;
37            }
38          OCRold = OCRnew;
39          OCRsum = 0;
40          poc = 0;
41        }
```

Příloha III: Výpis funkce SumKvM

```
1             .define KvLL   R2
2             .define KvL    R3
3             .define KvH    R4
4             .define KvHH   R5        ;kvadrát
5             .define MaxL   R6
6             .define MaxH   R7        ;maximum
7             .define SumLL  R16
8             .define SumL   R17
9             .define SumH   R18
10            .define SumHH  R19       ;mezisoučty
11            .define ValueL R8
12            .define ValueH R24       ;vzorek
13            .define Poc    R25       ;počítadlo průchodů
14            .define pMaxL  R26
15            .define pMaxH  R27       ;ukazatel maxima
16            .define BufL   R30
17            .define BufH   R31       ;ukazatel bufferu
18
19  _SumKvM:: MOVW BufL, R16           ;adresa bufferu
20            MOVW pMaxL, R18          ;adresa maxima
21
22            CLR SumLL
23            CLR SumL
24            CLR SumH
25            CLR SumHH                ;suma = 0
26            CLR MaxL
27            CLR MaxH                 ;maximum = 0
28
29            CLT                      ;T = 0 (33. bit sumy)
30            LDI Poc, 128             ;počítadlo průchodů
31
32  SmyckaM:  LD ValueL, Z+
33            LD ValueH, Z+            ;Value = buffer[?][?][i]
```

```
34              SBRS ValueH, 7          ;je záporné
35              RJMP MaximumM
36              COM ValueH
37              NEG ValueL
38              SBCI ValueH, 0xFF       ;absolutní hodnota
39
40  MaximumM:   MOV R0, ValueL
41              MOV R1, ValueH
42              SUB R0, MaxL
43              SBC R1, MaxH
44              BRCS KvadratM           ;je větší?
45              MOV MaxL, ValueL
46              MOV MaxH, ValueH
47
48  KvadratM:   CLR KvH             ;KvLL a KvL není nutné mazat
49              CLR KvHH                ;Kv = 0
50              MUL ValueL, ValueL      ;1. mezisoučin
51              MOVW KvLL, R0
52              MUL ValueL, ValueH
53              ADD KvL, R0
54              ADC KvH, R1             ;2. mezisoučin
55              ADD KvL, R0
56              ADC KvH, R1             ;2. a 3. mezisoučin
57              MUL ValueH, ValueH
58              ADD KvH, R0
59              ADC KvHH, R1            ;4. mezisoučin
60
61              ADD SumLL, KvLL
62              ADC SumL, KvL
63              ADC SumH, KvH           ;suma += kvadrát
64              ADC SumHH, KvHH
65              BRCC DalM               ;(C = 0) -> DalM
66              SET                     ;T = 1, (33. bit)
67
```

```
68  DalM:     DEC Poc
69            BRNE SmyckaM              ;zacyklení
70
71            ST X+, MaxL
72            ST X+, MaxH               ;uložení maxima
73
74            CLC                       ;C = 0
75            BRTC NepreM               ;(T = 0) -> NepreM
76            SEC                       ;C = 1
77  NepreM:   ROR SumHH
78            ROR SumH
79            ROR SumL
80            ROR SumLL                 ;Sum /= 2
81
82            RET
```