PRAXE

č. 2	STŘEDNÍ PRŮMYSLOVÁ ŠKOLA CHOMUTOV	David Herko
17. 12. 2024	GENERÁTOR PRŮBĚHŮ	V4

Zadání

S využitím vývojového kitu AVR a dvou paralelních DAC převodníků sestavte v jazyku C program fungující jako jednoduchý dvoukanálový funkční generátor ovládaný ze sériové linky. Pro její obsluhu použijte knihovnu, kterou máte k dispozici v rámci ukázkových kódů. Ověření parametrů generovaného průběhu provádějte osciloskopem. Výsledné řešení musí obsahovat následující funkcionalitu:

- Nezávislé nastavení parametrů pro každý kanál v reálných jednotkách (Hz, mV)

- Podpora tří různých průběhů na kanál. Vyžadován sinus a dva další dle vlastního výběru

- Generování průběhu v reálném čase pro oba kanály současně

- Nastavení parametrů průběhu pomocí textových příkazů z terminálu připojeného sériovou linkou (např. CH1:TYPE:SIN, CH2:FREQ:12, CH1:AMP:1500)

Teorie

Pomocí kontroleru ATMega128 se posílají generované signály do osciloskopu na dva různé kanály tak, že se na kontroler posílá hodnota napětí. Programuje se v jazyce C - pomocí externích knihoven `avr/io.h` pro komunikaci s kontrolerem a `util/delay.h` pro funkci zpoždění kódu. Nesmíme zapomenout nastavit registry směru DDRx a datové registry PORTx.

Popis řešení

Pro vytváření nastavení kanálu jsem se rozhodl vytvořit struct `ChannelStruct`. V `main` voidu se vytvoří dva tyto structy, každý pro jeden kanál. Nastaví se do něj amplituda (v milivoltech), perioda (v milisekundách), typ a port (to jest kanál, na kterém se má signál generovat). Typy jsou následující: -1 pro nulový signál, 0 pro obdélníkový průběh. 1 měla být pro pilu a 2 měla být sinusovka, ty jsem ale neimplementoval (viz závěr). Hlavní void `main` následně přejde do nekonečné smyčky, ve které se podle typu průběhu v `ChannelStructech` volají metody pro generování průběhu. Počká se jednu milisekundu a smyčka jede od znovu.

Rozbor proměnných a metod

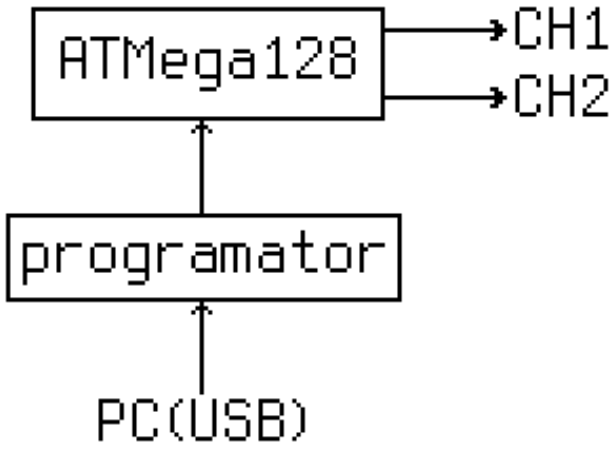
main.c

`void setup(void)`	Výchozí nastavení portů
`struct ChannelStruct`	Struct obsahující nastavení pro kanály generátoru
`void nullWave(Channel ch)`	Nulový signál (žádný průběh)
`void squareWave(Channel ch)`	Průběh typu obdélník
`int main(void)`	Hlavní metoda programu

struct ChannelStruct

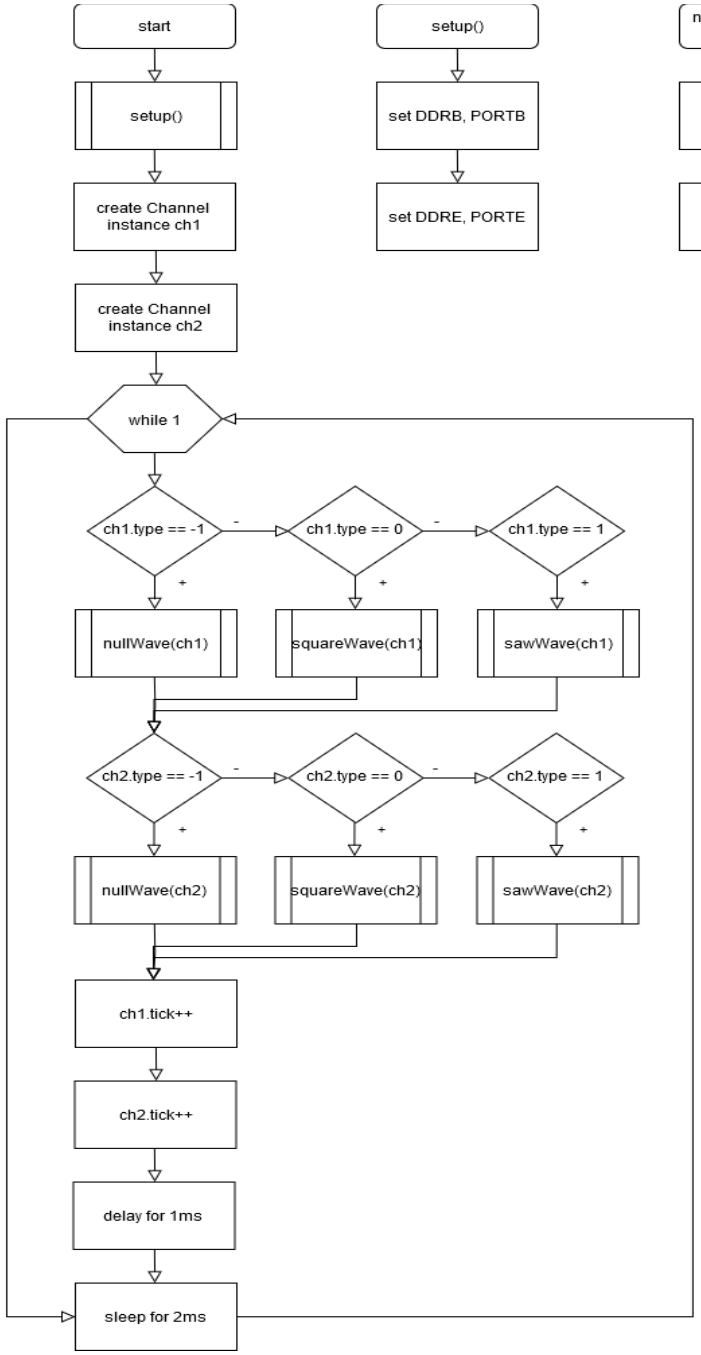
`int amplitude`	Amplituda [mV]
`int period`	Perioda [ms]
`int type`	Typ průběhu
`uint8_t port`	Port, na kterém se má průběh generovat
`int tick`	Proměnná pro tikání hodin kanálu

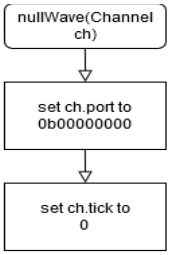
Schéma zapojení

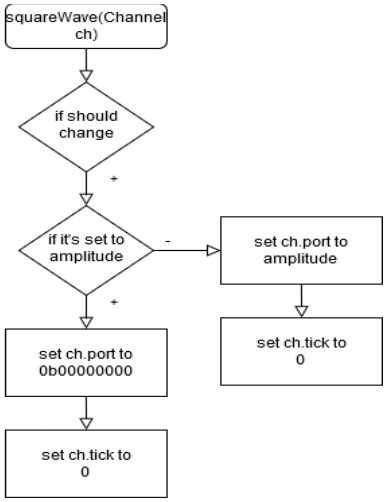


Vývojový diagram

main.c







Komentovaný výpis kódu

```
#include <avr/io.h>
#include <util/delay.h>
#include <stdio.h>
#include <math.h>
#include <stdint.h>

//configure CPU and devices
void setup(void) {
	DDRB = 0b11111111;
	PORTB = 0b00000000;

	DDRE = 0b11111111;
	PORTE = 0b00000000;
}

//channel info object
typedef struct ChannelStruct {
	int amplitude;
	int period;
	int type;
		//TYPE -1 = disabled
		//TYPE 0 = square
	uint8_t* port;
	int tick;
} Channel;

//resets wave to nothing
void nullWave(Channel ch) {
	//set to nothing
	*ch.port = 0b00000000;

	//reset tick
	ch.tick = 0;
}

//create a square wave
void squareWave(Channel ch) {
	//if should change wave
	if(ch.tick % ch.period == 0) {
		//either set it to the amplitude or nothing
		if(*ch.port == (ch.amplitude / 8)) {
			*ch.port = 0b00000000;
		} else if(*ch.port == 0b00000000) {
			*ch.port = (ch.amplitude / 8);
		}

		//reset tick
		ch.tick = 0;
	}
}

int main(void) {
  setup();

  //create ch1 with variables
  Channel ch1;
```

```
  ch1.amplitude = 2000; //in milivolts
  ch1.period = 1000; //in miliseconds
  ch1.type = 0;
  ch1.port = &PORTB;

  //create ch2 with variables
  Channel ch2;
  ch2.amplitude = 500; //in milivolts
  ch2.period = 320; //in miliseconds
  ch2.type = 0;
  ch2.port = &PORTE;

  //main loop for doing absolutely everything
  while(1) {
      //set function according to channel1 type
      if(ch1.type == -1) {
            nullWave(ch1);
      } else if(ch1.type == 0) {
            squareWave(ch1);
      }

      //set function according to channel2 type
      if(ch2.type == -1) {
            nullWave(ch2);
      } else if(ch2.type == 0) {
            squareWave(ch2);
      }

      //tick both channels
      ch1.tick++;
      ch2.tick++;
      _delay_ms(1);
      }

  return 0;
}
```

Odpovědi na otázky

1) Pro úsporu ovládacích IO linek se dnes běžně používají sériově řízené převodníky (např. MCP4921). Zkuste se zamyslet a popsat změny, které byste ve svém programu museli udělat při použití podobného typu převodníku.

Řízení pomocí sériové linky v mém programu není, tudíž nemám kód jak měnit.

2) Rozhodněte jaké rozlišení převodníku budete potřebovat v případě kdy budete chtít generovat signál v rozsahu ±1.5V s nejmenším rozlišením (velikostí kroku) alespoň 0.5mV. Uveďte postup jakým jste k potřebnému rozlišení dospěli.

Matematika, jenž nerozumím.

3) Na výstupu DA převodníků se často používá operační zesilovač. Pokuste se vysvětlit proč.

Je potřeba převést mezi analogovým na digitální signál, nebo něco podobného.[1]

1 http://352lab.vsb.cz/Podklady/03_PAR/Prevodniky.html

Závěr

S generátorem jsem se moc daleko nedostal. Tenhle typ úlohy mi nesedl, protože je to kombinace matematiky a elektroniky, což jsou dva obory kterým nerozumím a jdou úplně mimo mě. Také to je moje první úloha v jazyce C, se kterým nemám předchozí zkušenosti a tak mi programování trvá ještě déle než obvykle. Povedlo se mi implementovat jenom jeden průběh (obdélník) a oba kanály jdou nastavovat nezávisle na sobě.