Instytut Informatyki Politechniki Śląskiej

Zespół Mikroinformatyki i Teorii Automatów Cyfrowych

Rok akademicki:	Rodzaj studiów*: SSI/NSI/NSM	Przedmiot (Języki Asemblerowe/SMiW):	Grupa	Sekcja
2018/2019	**SSI**	**SMiW**	**3**	**6**

Imię:	**Mirosław**	**Prowadzący**: OA/JP/KT/GD/BSz/GB	**BZ**
Nazwisko:	**Ściebura**		

Raport końcowy

Temat projektu:

Stoper

Data oddania: dd/mm/rrrr	**06/02/2019**

1. Temat projektu

Tematem projektu było zaprojektowanie stopera wyświetlającego wszelkie informacje na wyświetlaczu LCD.

Główną funkcją było odmierzanie czasu do naciśnięcia odpowiedniego przycisku z możliwością zapisu międzyczasu przy naciśnięciu innego przycisku, kontynuacja odmierzania lub reset w zależności od naciśniętego przycisku gdy czas został zatrzymany oraz wyświetlenie międzyczasów po resecie.

Głównym założeniem było wykorzystanie mikrokontrolera (w tym wypadku była to Atmega8A) oraz wyświetlacza LCD 2x16. W planach było wykorzystanie też modułu RTC, jednak jego wykorzystanie było niepotrzebne (wewnętrzny timer procesora) a nawet bezużyteczne (czas podawany jedynie z dokładnością do sekund). Układ był realizowany przy pomocy płytki stykowej.

2. Analiza zadania

Do wykonania projektu użyłem następujących elementów:

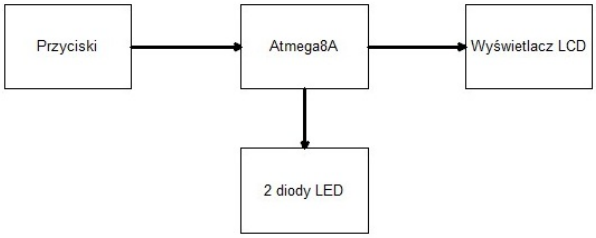
- mikrokontroler Atmega8A-PU – mikrokontroler w obudowie przewlekanej (PDIP-28) w celu użycia na płytce stykowej, wybrany ze względu na prostotę budowy i wykorzystania oraz niską cenę i dużą liczbę materiałów dydaktycznych, układ zapewnia dużą funkcjonalność która pozwoliła na wykonanie projektu
- wyświetlacz LCD 2x16 znaków oparty o sterownik Hitachi HD44780 – wybrany ze względu na możliwość pokazywania wielu informacji oraz oparty o dość popularny sterownik, do jego obsługi wykorzystane zostały biblioteki autorstwa Radosława Kwietnia (http://radzio.dxp.pl/hd44780/)
- potencjometr obrotowy 10kΩ liniowy 1/8W – potrzebny do odpowiedniej regulacji kontrastu na wyświetlaczu
- Tact Switch 12x12mm / 7mm THT 4pin – przyciski monostabilne – w celu komunikacji użytkownika z układem
- 2 żółte diody LED – w celu zobaczenia czy nie następują drgania zestyków (przełączenie przy naciśnięciu przycisku)
- rezystory i kondensatory – w celu zapewnienia odpowiednich parametrów elektrycznych

Ze względu na brak większego doświadczenia w projektowaniu i uruchamianiu takich układów wybrałem elementy najbardziej popularne (duża liczba materiałów dydaktycznych i pomocniczych). Oprócz tych elementów użyłem także programatora USBASP w celu zaprogramowania układu oraz dostarczenie zasilania. Jest to również jeden z podstawowych programatorów. Do wgrywaia wsadu .hex użyłem MkAvrCalculator który bardzo dobrze współpracował z programatorem (jest to nakładka graficzna na program avrdude). Jako środowisko programistyczne wybrałem Atmel Studio 7.0, dostarczane przez producenta mikrokontrolera – firmę Atmel (firma wykupiona przez Microchip – można spotkać nazwę tego producenta).

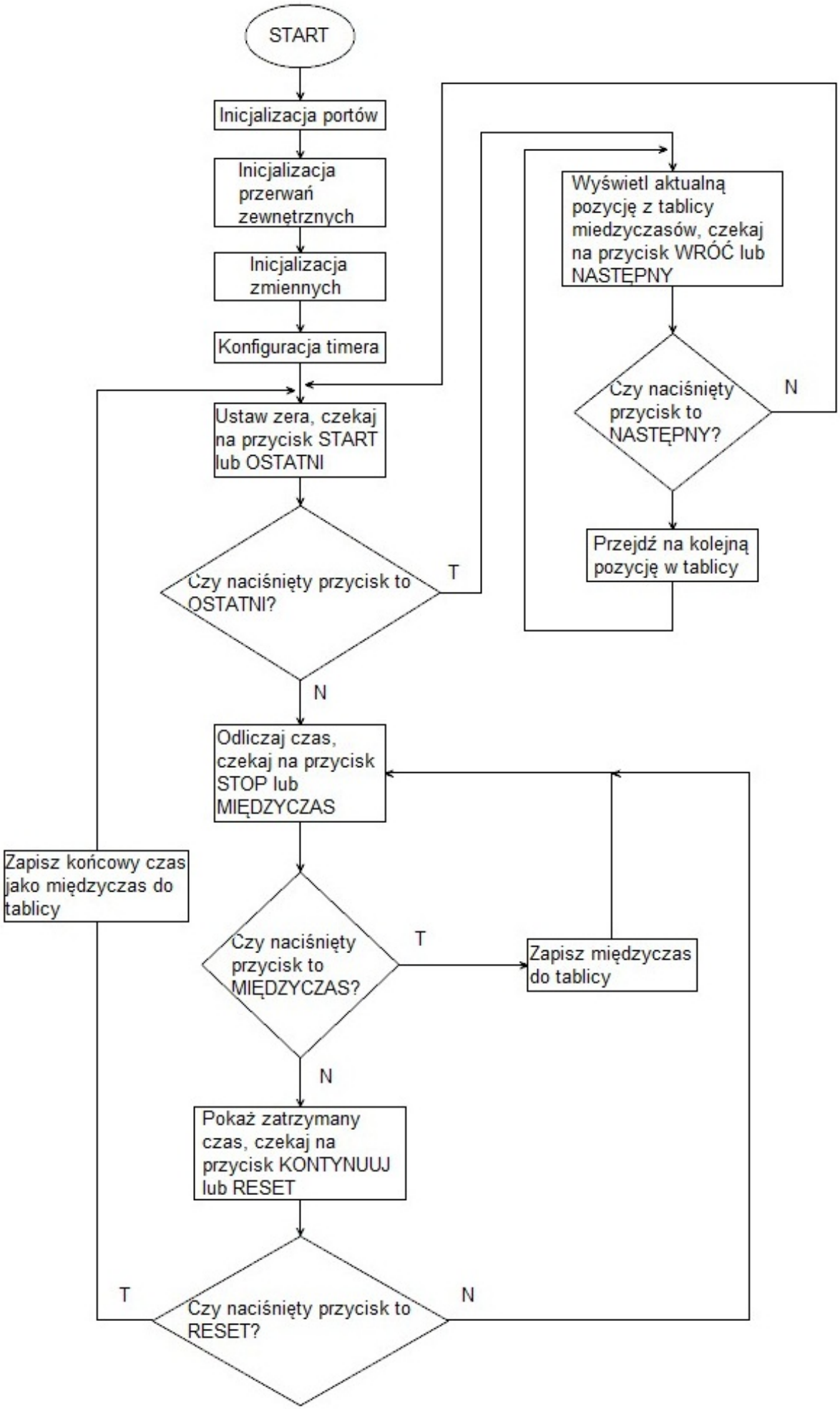
Zadanie polegało na podłączeniu wszystkich elementów w celu dobrej komunikacji, a następnie ustaleniu roli konkretnych portów i napisaniu odpowiedniego algorytmu w języku C obejmującego konkretne działanie, obsługę przerwań zewnętrznych i licznikowych oraz wyświetlanie odpowiednich informacji na wyświetlaczu.

3. Specyfikacja wewnętrzna

a) Schemat ideowy



b) Schemat blokowy



c) Funkcje poszczególnych bloków układu

Atmega8A – Mikrokontroler, główna część sterująca programem i realizująca określony algorytm. Zawiera też kondensatory ceramiczne 100nF i elektrolityczne 1uF oraz rezystor 10k przy /RESET w celu zapewnienia odpowiedniego zasilania.

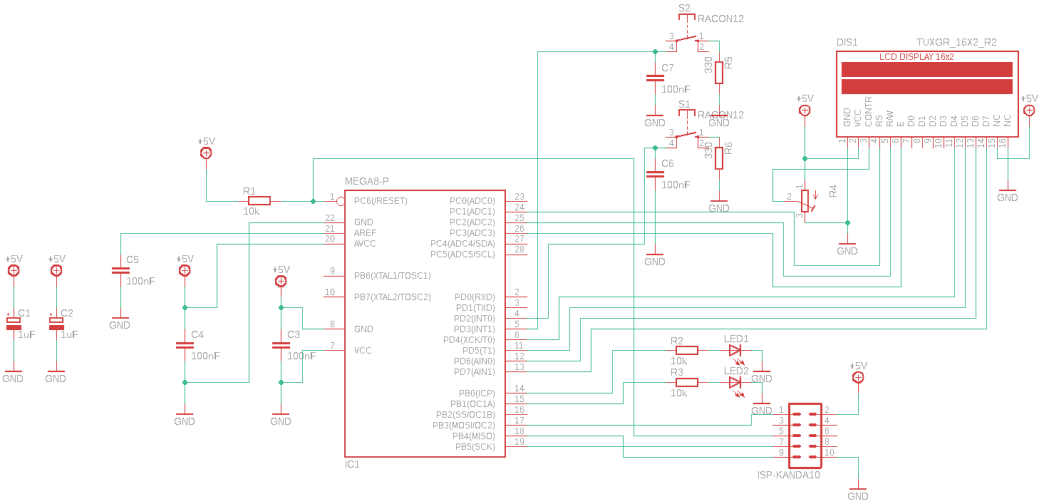
Wyświetlacz LCD – Odpowiada za prezentowanie informacji o zmierzonym i zapisanym czasie oraz opis aktualnej funkcji klawiszy. Składa się z wyświetlacza opartego o sterownik Hitachi HD44780 oraz potencjometru obrotowego 10kΩ do regulacji kontrastu.

Przyciski – Zestaw 2 przycisków do interakcji z użytkownikiem wywołujących przerwanie (INT0 lub INT1). Odpowiadają za START/OSTATNI, STOP/MIĘDZYCZAS, KONTYNUUJ/RESET. Przyciski mają podłączone kondensatory ceramiczne 100nF i rezystory 330Ω w celu ograniczenia drgań zestyków. Linie sterujące podpięte są do linii PC1 – PC3 mikrokontrolera, natomiast linie danych do PD4 – PD7.

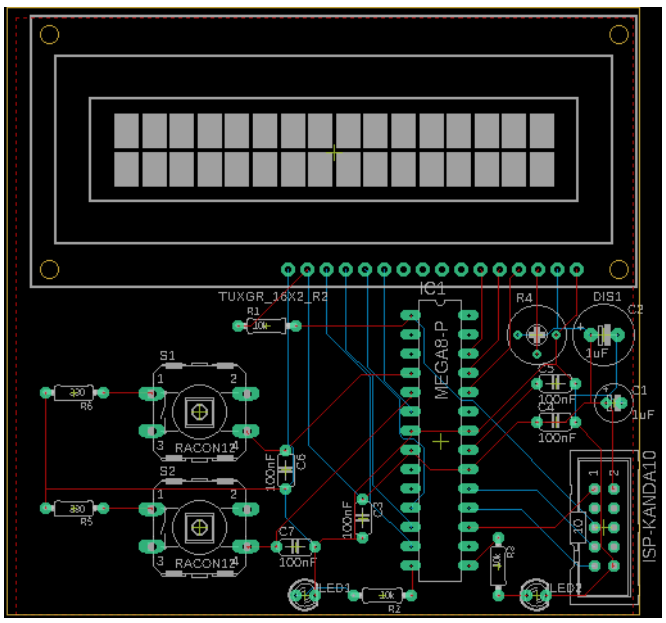
Diody LED – Zestaw 2 diod LED. Zmieniają stan na przeciwny (świeci się jedna albo druga) przy naciśnięciu któregoś z przycisków (sprawdzanie braku drgań zestyków). Podłączone przez rezystory 10k w celu ograniczenia prądu.

d) Schematy

Schemat logiczny układu

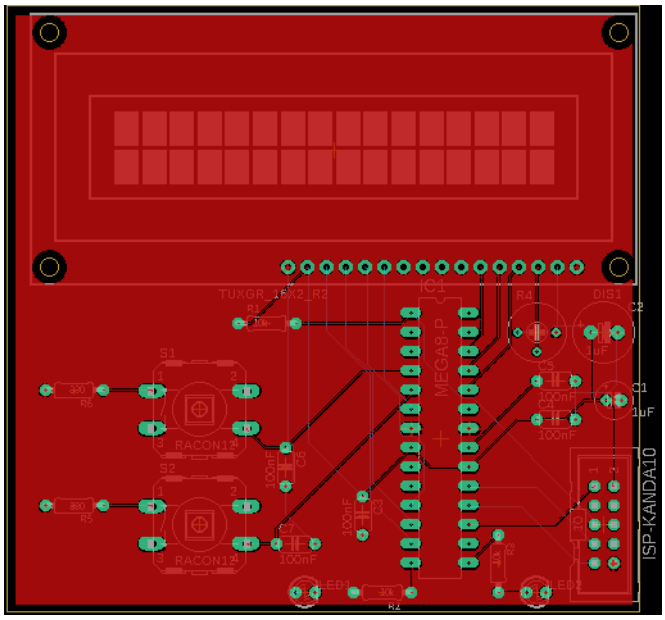


Schemat płytki PCB wraz z rozmieszczeniem elementów na płytce i obudowie



Obudowa mogłaby obejmować całą płytkę i posiadać jedynie otwór na wyświetlacz i przyciski oraz ew. złącze programowania (jednak zakładamy jednokrotne programowanie układu przez co mógłby być ukryty). Piny 15 i 16 na schemacie logicznym są podłączone do zasilania i masy ze względu na użycie podświetlenia (piny te to kolejno anoda i katoda). Wszystkie wejścia mają wewnętrzny rezystor podciągający więc ustawiona programowo jedynka przy braku podłączenia portu będzie poprawna.

Płytka z naniesionym obszarem masy (polygon GND)



e) Algorytm oprogramowania urządzenia

Działanie algorytmu został wyjaśnione w dużej części na schemacie blokowym. W celu początkowej konfiguracji wyłączamy przerwania. Wstępnie inicjalizowane są odpowiednie linie portów(PB0, PB1 wyjściowe, PD2, PD3 – wejściowe) i przerwania INT0 i INT1. Zerowany jest też licznik 0. Następnie mamy 50 ms opóźnienie w celu stabilizacji stanów oraz uzupełnienie znaków tablicy znaki (string z czasem). Następnie zerujemy zmienne programu i inicjalizujemy wyświetlacz funkcją LCD_Initalize() oraz konfigurujemy timer2 (zerujemy, ustawiamy prescaler 128 i włączamy przerwanie przy przepełnieniu). Po włączeniu przerwań następuje pętla nieskończona w której sprawdza się potrzebę odświeżenia, a następnie w zależności od trybu i potrzeby wywołuje funkcję do wyświetlania startu (tryb 0), funkcję wyświetlania międzyczasu (tryb 1), funkcje utworzenia czasu pisemnego i wypisania na ekran w odpowiednim miejscu (tryb 2) lub wyświetlenie stopu (tryb 3). po tych sprawdzeniach zeruje się też znacznik odświeżania. W trakcie naciśnięcia przycisku wywoływana jest procedura obsługi przerwania INT0 lub INT1 (obsługa działań wyświetlanych na ekranie) a w trakcie przepełnienia liczników 0 lub 2 ich procedury przerwań.

f) opis zmiennych

Zmienne są globalne ze względu na potrzebę obsługi większości z poziomu przerwań (brak przekazywania parametrów).

volatile unsigned char tryb – zmienna określająca aktualny tryb w którym znajduje się stoper (0 – wyświetlanie wartości startowej, 1 – przeglądanie zapisanych międzyczasów, 2 – odliczanie czasu, 3 – zatrzymanie), volatile w celu uniknięcia agresywnej optymalizacji (przyjęcie z góry wartości przez kompilator)
volatile unsigned char godziny – bufor przechowujący ilość godzin zliczonego czasu, volatile w celu uniknięcia agresywnej optymalizacji (przyjęcie z góry wartości przez kompilator)
volatile unsigned char minuty – bufor przechowujący ilość minut zliczonego czasu, volatile w celu uniknięcia agresywnej optymalizacji (przyjęcie z góry wartości przez kompilator)

volatile unsigned char sekundy – bufor przechowujący ilość sekund zliczonego czasu, volatile w celu uniknięcia agresywnej optymalizacji (przyjęcie z góry wartości przez kompilator)
volatile unsigned int milisekundy – bufor przechowujący ilość milisekund zliczonego czasu, volatile w celu uniknięcia agresywnej optymalizacji (przyjęcie z góry wartości przez kompilator)
volatile unsigned int mikrosekundy – bufor przechowujący ilość mikrosekund zliczonego czasu, volatile w celu uniknięcia agresywnej optymalizacji (przyjęcie z góry wartości przez kompilator)
char znaki[13] – string z pisemną reprezentacją czasu (hh:mm:ss:mmm), jako tablica ze względu na stałą długość
char historiaznaki[15] – string z pisemną reprezentacją czasu (hh:mm:ss:mmm) dla kolejnych międzyczasów – indeks na początku, jako tablica ze względu na stałą długość
unsigned char historiagodziny[MIEDZYCZASY] – tablica o z góry określonej długości, przechowująca do określonej liczby czasów (część godzinowa)
unsigned char historiaminuty[MIEDZYCZASY] – tablica o z góry określonej długości, przechowująca do określonej liczby czasów (część minutowa)
unsigned char historiasekundy[MIEDZYCZASY] – tablica o z góry określonej długości, przechowująca do określonej liczby czasów (część sekundowa)
unsigned int historiamilisekundy[MIEDZYCZASY] – tablica o z góry określonej długości, przechowująca do określonej liczby czasów (część milisekundowa)
unsigned int historiamikrosekundy[MIEDZYCZASY] – tablica o z góry określonej długości, przechowująca do określonej liczby czasów (część mikrosekundowa)
volatile unsigned char czywyswietlonostart – blokada dla wyświetlania startu, mówi czy start był wyświetlony (potrzeba odświeżenia), volatile w celu uniknięcia agresywnej optymalizacji (przyjęcie z góry wartości przez kompilator)
volatile unsigned char czywyswietlonostop – blokada dla wyświetlania stopu, mówi czy stop był wyświetlony (potrzeba odświeżenia), volatile w celu uniknięcia agresywnej optymalizacji (przyjęcie z góry wartości przez kompilator)
volatile unsigned char pozycja – indeks pozycji w tablicy na którą będziemy wpisywać między czas (maksymalnie MIEDZYCZASY – 1 w celu zapisania na koniec czasu ostatecznego), volatile w celu uniknięcia agresywnej optymalizacji (przyjęcie z góry wartości przez kompilator)
volatile unsigned char wyswietlanapozycja – indeks pozycji wyświetlanej w trybie 1, jeżeli przekroczy ostanią pozycję zapisania w tablicy to zostanie wyzerowany, volatile w celu uniknięcia agresywnej optymalizacji (przyjęcie z góry wartości przez kompilator)
volatile unsigned char wyswietlaniemiedzyczasu – blokada dla wyświetlania międzyczasu, mówi czy trzeba wyświetlić nowy międzyczas (potrzeba odświeżenia), volatile w celu uniknięcia agresywnej optymalizacji (przyjęcie z góry wartości przez kompilator)
volatile unsigned char odswiez – blokada odświeżania ekranu, odświeżenie tylko przy zmianie informacji, volatile w celu uniknięcia agresywnej optymalizacji (przyjęcie z góry wartości przez kompilator)
volatile unsigned char bufor – bufor w celu przepisania timera, volatile w celu uniknięcia agresywnej optymalizacji (przyjęcie z góry wartości przez kompilator)

Domyślnie stała MIEDZYCZASY ustawiona jest na 8.

g) Opis funkcji

void uaktualnienie() – funkcja uaktualniająca stan bufora czasu (inkrementacja liczby mikrosekund, milisekund)
void czaspisemny() – wykonuje pisemną reprezentację czasu w zmiennej globalnej znaki
void wlacztimer() – włącza timer0 - ustawienie odpowiedniego prescalera wewnątrz
void wylacztimer() – włącza timer0 - ustawia 000 w miejscu prescalera i zeruje bit overflow
void ekran(char tablicaznakow[], unsigned char poczatek, char* napis) – wypisuje na ekran LCD tablicę znaków (c-string) w pierwszej linii od podanego znaku w niej i drugi napis w dolnej
void przepiszbufor() – przepisuje do bufora czasu informację z TCNT0 i zeruje ten rejestr
void wyswietlstart() – zeruje bufor i wyświetla informację startową
void wyswietlstop() – oczyt czasu i stopuje
void wyswietlmiedzyczas() – wpisuje miedzyczas na ekran LCD

void wpiszmiedzyczas() – ładuje do bufora odpowiedni czas z tablicy miedzyczasów
void wpiszdohistorii() – wpisuje do tablicy z historią na aktualną pozycję
void odnotujmiedzyczas() – wpisuje miedzyczas do tablicy (jak jest miejsce)
ISR(INT0_vect) – procedura obsługi przerwania INT0, przepisuje bufor (przepisanie i zerowanie zawartości TCNT0), potem w zależności od trybu w jakim była wykonywana (0 – ustawienie 0 na TCNT0 i pozycji oraz włączenie timera, 1 – wpisanie międzyczasu oraz wyzerowanie wyswietlaniemiedzyczasu, 2 – wyłączenie timera, przepisanie bufora, wyzerowanie czywyswietlonostop, 3 – włączenie timera), następnie zmienia tryb na odpowiedni (z 0 na 2, z 1 na 1, z2 na 3, z 3 na 2) i zmiana stanu LED
ISR(INT1_vect) – procedura obsługi przerwania INT1, przepisuje bufor (przepisanie i zerowanie zawartości TCNT0), potem w zależności od trybu w jakim była wykonywana (0 – wpisanie miedzyczasu i wyzerowanie wyswietlaniemiedzyczasu, 1 – wyzerowanie wyswietlona pozycja i czywyswietlonostart, 2 – odnotowanie miedzyczasu, 3 – wpis do historii, wyzerowanie wyswietlona pozycja i czywyswietlonostart), następnie zmienia tryb na odpowiedni (z 0 na 1, z 1 na 0, z2 na 2, z 3 na 0) i zmiana stanu LED
ISR(TIMER0_OVF_vect) – procedura obsługi przerwania przepełnienia timera 0, zwiększa bufor milisekund o 65 i mikrosekund o 536 oraz uaktualnia czas bufora
ISR(TIMER2_OVF_vect) – procedura obsługi przerwania przepełnienia timera 2, zmienia na 1 wartość zmiennej odswiez (potrzeba odświeżenia informacji na ekranie)

h) opis interakcji oprogramowania z układem elektronicznym

Oprogramowanie odpowiada m.in. za poprawne wyświetlanie informacji na wyświetlaczu LCD. Szczególnie pomocne są tu funkcje z biblioteki HD44780.h (autorstwa Radosława Kwietnia – radzio.dxp.pl). Te funkcje to LCD_Clear, LCD_GoTo i LCD_WriteText. Mamy też wykorzystanie wbudowanych timerów 0 i 2 jako odpowiednio liczników czasu i odświeżania treści na ekranie (zmienna odswiez). Mamy też zmianę stanu na przeciwny 2 diód LED, co pozwala nam sprawdzić brak drgań zestyków przy naciśnięciu któregoś z przycisków. Jako interakcje możemy również uznać narzędzie do programowania jakim jest programator USBASP, który komunikuje się (programuje) z układem przez łącze KANDA.

i) Szczegółowy opis działania ważniejszych procedura

Za najważniejsze funkcję uznałbym procedury obsługi przerwań INT0 i INT1.

ISR(INT0_vect) – procedura obsługi przerwania INT0, przepisuje bufor (przepisanie i zerowanie zawartości TCNT0), potem w zależności od trybu w jakim była wykonywana wewnątrz instrukcji switch:
0 – ustawienie 0 na TCNT0 i pozycji oraz włączenie timera, zmiana na tryb 2;
1 – wpisanie międzyczasu oraz wyzerowanie wyswietlaniemiedzyczasu, zmiana na tryb 1;
2 – wyłączenie timera, przepisanie bufora, wyzerowanie czywyswietlonostop, zmiana na tryb 3;
3 – włączenie timera, zmiana na tryb 2.
Pod koniec następuje zmiana stanu diód LED.

ISR(INT1_vect) – procedura obsługi przerwania INT1, przepisuje bufor (przepisanie i zerowanie zawartości TCNT0), potem w zależności od trybu w jakim była wykonywana wewnątrz instrukcji switch:
0 – wpisanie miedzyczasu i wyzerowanie wyswietlaniemiedzyczasu, zmiana na tryb 1;
1 – wyzerowanie wyswietlona pozycja i czywyswietlonostart, zmiana na tryb 0;
2 – odnotowanie miedzyczasu, zmiana na tryb 2;
3 – wpis do historii, wyzerowanie wyswietlona pozycja i czywyswietlonostart), zmiana na tryb 0.
Pod koniec następuje zmiana stanu diód LED.

4. Specyfikacja zewnętrzna

a) Opis funkcji elementów sterujących urządzeniem

Główną częścią układu odpowiedzialną za sterowanie jest mikrokontroler Atmega8 korzystający z wewnętrznego oscylatora RC o częstotliwości 1MHz. Układ ten jest w tej konfiguracji w stanie realizować wszelkie potrzebne działania. Sterowanie urządzeniem odbywa się przez 2 przyciski, które w zależności od aktualnego trybu w jakim znajduje się urządzenie odpowiadają za konkretną akcję po wciśnięciu.

b) Opis funkcji elementów wykonawczych

Głównym elementem wykonawczym jest wyświetlacz LCD 2x16 znaków. Odpowiada on za przekazywanie użytkownikowi informacji nt. mierzonego czasu, zapisanych międzyczasów, możliwości kolejnych działań układu.

Jak już wcześniej wspomniano, sterowanie urządzeniem odbywa się przez 2 przyciski, które w zależności od aktualnego trybu w jakim znajduje się urządzenie odpowiadają za konkretną akcję po wciśnięciu.

Oprócz tego mamy 2 diody LED w celu sprawdzania czy nie nastąpiły drgania zestyków (czysto kontrolnie – w układzie jest rezystor 330Ω i kondensator 100 nF równoległy dla każdego przycisku, co powinno eliminować drgania). Zmieniają one stan na przeciwny po każdorazowym naciśnięciu dowolnego przycisku

c) Opis reakcji na zdarzenia zewnętrzne

Zdarzeniami zewnętrznymi które sterują układem są naciśnięcia przycisków (zwarcia do masy), które powodują przerwania INT0 i INT1. W programie zawarte mamy procedury obsługi tych przerwań, które odpowiadają odpowiednim działaniom (opisanym na ekranie) i zmianom trybów pracy urządzenia. Reakcje te są dokładniej opisane w opisie funkcji ISR(INT0_vect) i ISR(INT1_vect).

d) Skrócona instrukcja obsługi urządzenia

1. Podłącz urządzenie do zasilania.
2. Wyświetlony zostaje ekran startowy z wyzerowanym czasem i napisami START oraz OSTATNI.
   a) Po wciśnięciu przycisku odpowiadającemu START rozpocznie się odliczanie czasu.
   b) Po wciśnięciu przycisku odpowiadającemu OSTATNI nastąpi przejście do historii ostatniego pomiaru (która będzie pusta ze względu na pierwsze użycie). Wyświetlony zostanie wyzerowany czas z indeksem 1 i napisy NASTĘPNY oraz WRÓĆ. Po wciśnięciu przycisku odpowiadającemu NASTĘPNY nic się nie stanie (próba przejścia na następną pozycję w tablicy zapisanych międzyczasów, która jest pusta). Po wciśnięciu przycisku odpowiadającemu WRÓĆ nastąpi powrót do ekranu startowego.
3. Po wciśnięciu START następuje odliczanie czasu i wyświetlenie napisów STOP i MIĘDZYCZAS.
   a) Po wciśnięciu przycisku odpowiadającemu STOP czas zostanie zatrzymany.
   b) Po wciśnięciu przycisku odpowiadającemu MIĘDZYCZAS aktualny stan zegara zostanie wpisany do tablicy międzyczasów (na kolejną pozycję). Odliczanie czasu nie będzie wstrzymane. Po 7 zapisach międzyczasu)wartość domyślna) naciśnięcie zostanie zignorowane (brak pozycji w tablicy).
4. Po wciśnięciu przycisku STOP czas zostaje zatrzymany i wyświetlony na wyświetlaczu z napisami KONTYNUUJ i RESET.
   a) Po wciśnięciu przycisku odpowiadającemu RESET czas zostanie wyzerowany i nastąpi powrót do ekranu głównego. Czas ostateczny zostanie zapisany na ostatniej pozycji w tablicy międzyczasów (domyślnie na 8 pozycji, ostatnia pozycja zarezerwowana dla czasu końcowego). Po naciśnięciu przycisku OSTATNI będziemy mogli przeglądać te międzyczasy.
   b) Po wciśnięciu przycisku odpowiadającemu KONTYNUUJ pomiar będzie kontynuowany (bez utraty odnotowanych międzyczasów ale też bez zapisu zatrzymanej wartości).
5. Po wciśnięciu przycisku RESET wracamy do ekranu głównego. Po wciśnięciu START rozpoczyna się nowy pomiar (usunięcie wszystkich zapisanych międzyczasów i mierzenie od nowa). Po wciśnięciu

OSTATNI możemy przeglądać zapisane międzyczasy i czas końcowy (wraz z kolejnymi indeksami – domyślnie od 1 do 8). Przycisk NASTĘPNY umożliwia przejście na następną pozycję w tablicy i wyświetlenie wartości międzyczasu z niej. Przycisk WRÓĆ przechodzi do ekranu startowego (jak po naciśnięciu RESET możemy po raz kolejny nacisnąć OSTATNI i jeszcze raz przejrzeć międzyczasy – dopiero naciśnięcie START usuwa historię).
6. Aby zakończyć odłącz urządzenie od zasilania.

5. Opis montażu i uruchamiania

a) Jakie problemy wystąpiły podczas montażu i uruchamiania i jak zostały rozwiązane?

Podczas montażu nie wystąpiły większe problemy. Po każdej modyfikacji i podłączeniu układ był kilkukrotnie sprawdzany, co pozwoliło uniknąć usterek. Ważnym było zapewnienie odpowiedniego zasilania i jego filtracji zgodnie z zaleceniami producenta. Ważnym też było pobranie odpowiednich sterowników do programatora i konfiguracja programu MkAvrCalculator, pozwalającego na zaprogramowanie układu. Istotną kwestią była eliminacja drgań zestyków w trakcie używania przycisku rozwiązana przy pomocy dołączenia równolegle kondensatora ceramicznego 100 nF i szeregowego rezystora 330Ω Podczas uruchamiania na bieżąco były wychwytywane drobne błędy w kodzie (np. wyświetlanie znaków zamiast cyfr na wyświetlaczu, nie odpowiedni zapis międzyczasów do tablicy) i poprawiane wraz z kolejnymi programowaniami układu.

b) Jak przeprowadzano testy poprawności działania urządzenia?

Testy poprawności polegały głównie na kilkukrotnym użyciu wszystkich możliwych trybów działania (ścieżek wykonania), w różnych konfiguracjach. Czasy odnotowywane były porównywane ze stoperem wzorcowym. Wszystkie te pomiary były wykonywane kilkukrotnie co pozwoliło na uznanie działania układu za poprawne.

6. Wnioski z uruchamiania i testowania

Wbrew wcześniejszym przypuszczeniom, realizacja projektu nie wiązała się z większymi problemami. Przy dość ostrożnym podłączaniu i modyfikacjach układu udało się uniknąć większych strat. Mimo, iż na początku wydawało się koniecznym zastosowanie układu zegara czasu rzeczywistego DS1307 to do odliczania czasu wystarczyło taktowanie mikrokontrolera i wykorzystanie timerów, przez co można myło pozwolić na zrezygnowanie z tego układu (zmniejszenie skomplikowania i kosztów). Sama realizacja projektu była bardzo potrzebna i kształcąca, gdyż dzięki w niej można było w praktyce zobaczyć mechanizmy układu mikroprocesorowego, jego wejść i wyjść oraz peryferiów. Samo testowanie było także bardzo potrzebne aby wychwycić drobne błędy w oprogramowaniu i działaniu systemu. Realizacja tego projektu bardzo mnie zaciekawiła co może skutkować kolejnymi projektami.