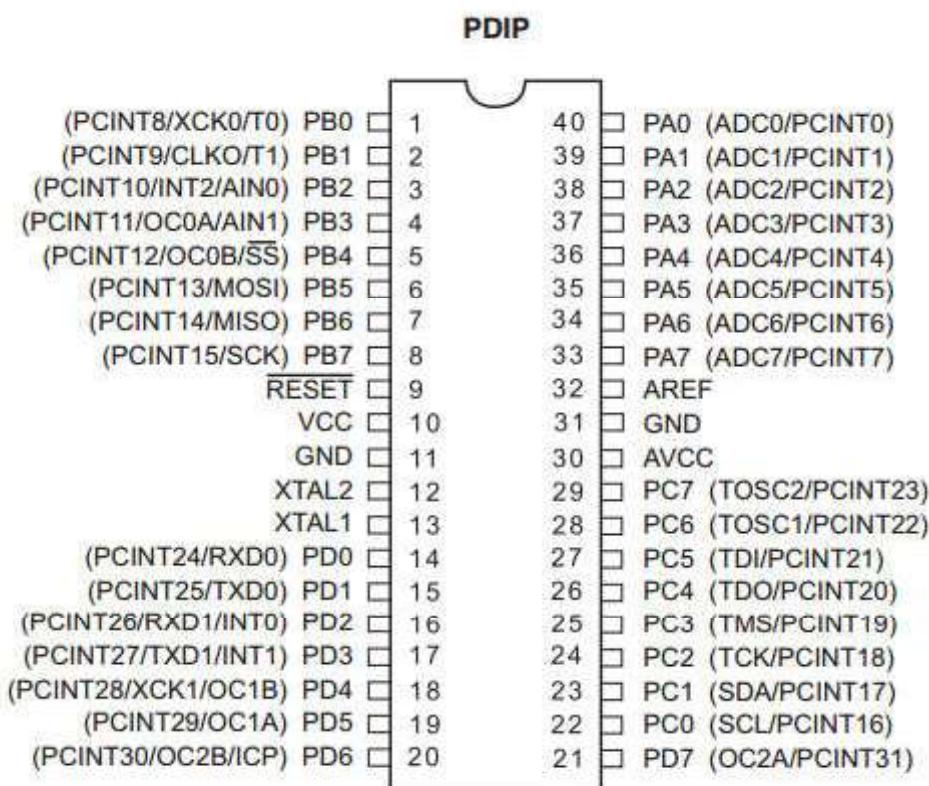


## 4. Projekt części elektronicznej urządzenia

### 4.1 Dobór układu sterującego

Do sterowania układem został wybrany mikrokontroler ATmega 644PA z serii AVR produkowany przez firmę Atmel. Schemat wyprowadzeń został przedstawiony na rysunku 4.1. Układ ten posiada cztery trójstanowe ośmioróżkowe porty wejścia/wyjścia, które mogą pełnić inne role takie, jak wejścia sygnałów przerwań bądź wewnętrznych interfejsów. W swojej strukturze posiada on 64kB pamięci flash, 2kB EEPROM oraz 4kB SRAM, które są dostępne w oddzielnych przestrzeniach adresowych [12].

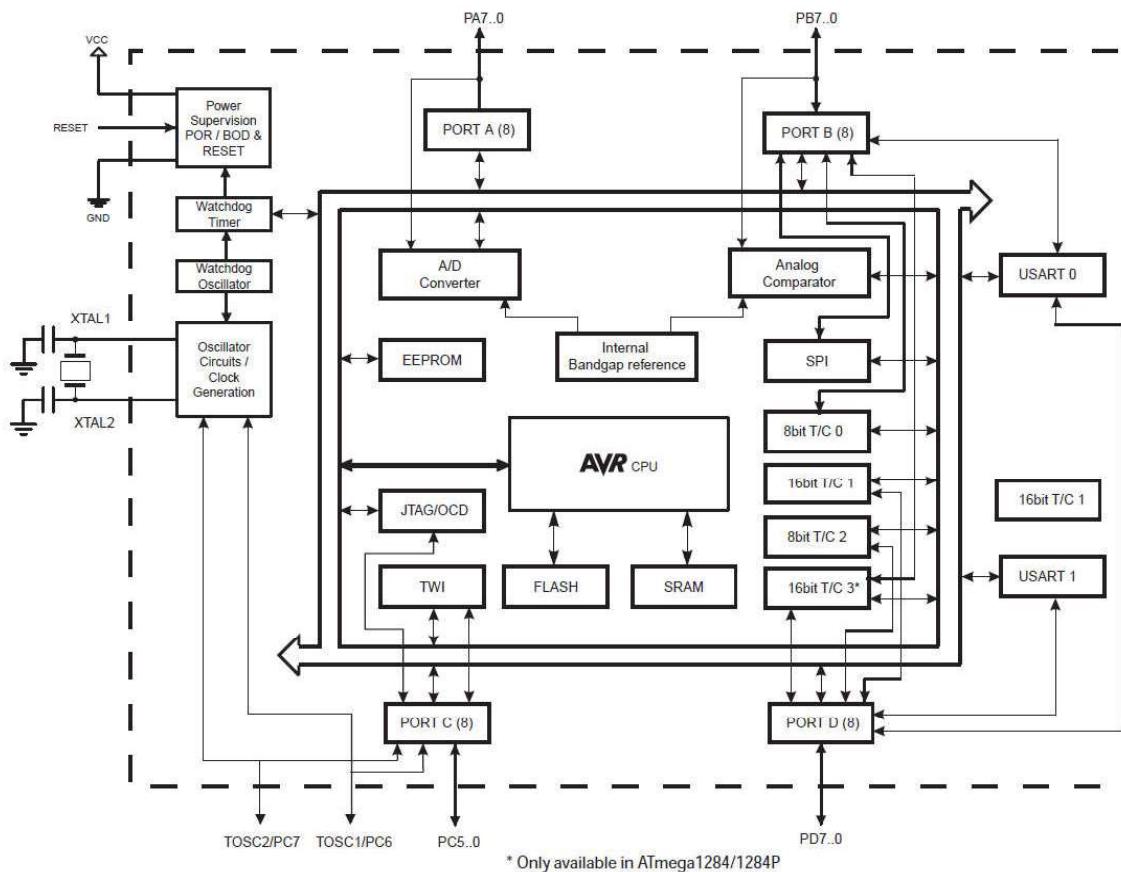


Rys. 4.1. Pinout mikrokontrolera ATmega 644PA w obudowie DIP [12].

Dostępne są trzy opcje źródeł sygnału zegarowego takie jak wewnętrzny oscylator RC, zewnętrzny generator kwarcowy oraz zewnętrzny rezonator kwarcowy podłączany do wejść XTAL1 oraz XTAL2. Maksymalna częstotliwość taktowania według dokumentacji wynosi 20MHz dla zewnętrznego generatora bądź oscylatora kwarcowego. W projekcie użyty został rezonator kwarcowy o częstotliwości drgań równej 24MHz. Stabilne przetaktowanie mikrokontrolera możliwe jest jedynie w przypadku, gdy nie jest

wykorzystywany zapis komórek do pamięci EEPROM. Zapis przy zegarze większym niż podawany przez producenta powoduje dużo szybsze uszkodzenie tejże pamięci.

ATmega posiada w swojej strukturze wiele układów peryferyjnych. Blokowy schemat wewnętrzny mikrokontrolera przedstawia rysunek 4.2.



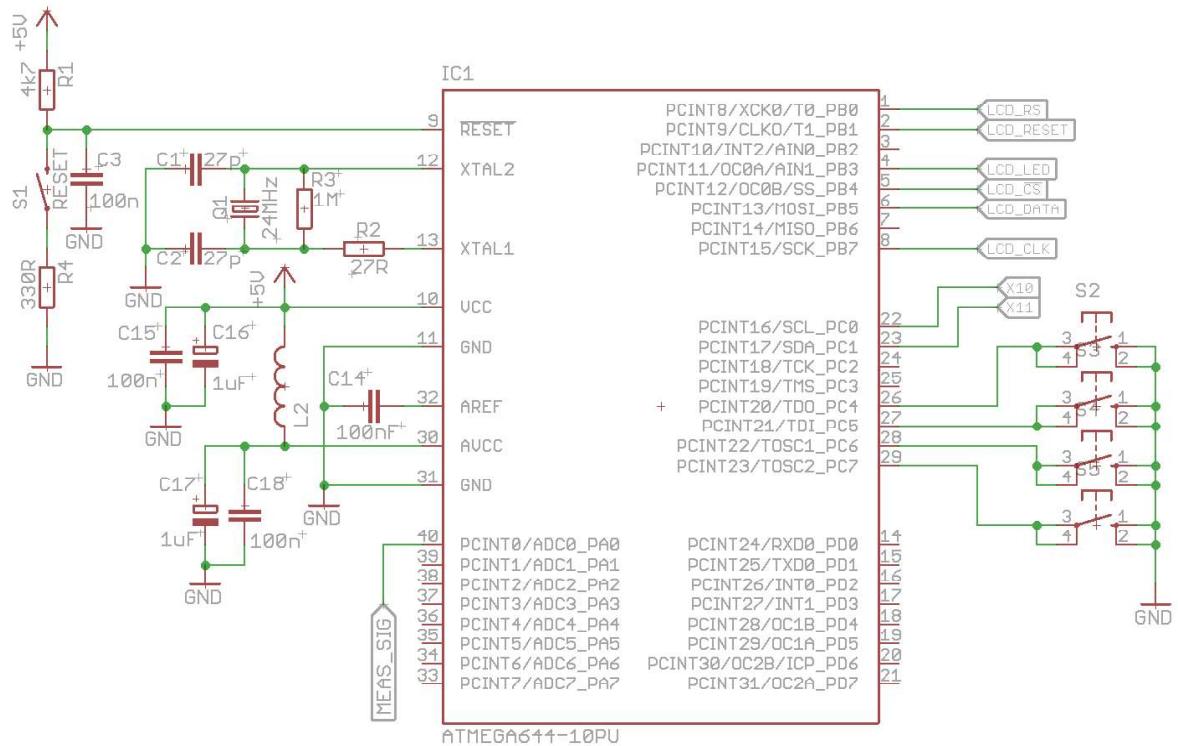
Rys. 4.2. Schemat blokowy mikrokontrolera ATmega 644PA [12].

Dostępne są sprzętowe interfejsy SPI, TWI, JTAG oraz USART. Interfejs SPI został użyty do komunikacji mikrokontrolera z wyświetlaczem. Ponadto dostępne są dwa ośmioróżkowe oraz jeden szesnastoróżkowy timer/licznik, który użyty został jako generator podstawy czasu.

Wewnętrzny przetwornik A/C posiada 10-bitową rozdzielcość i charakteryzuje się dużą szybkością próbkowania jak na przetwornik wbudowany w mikrokontroler. Według dokumentacji maksymalna częstotliwość taktowania przetwornika wynosi 200kHz, jednakże przy wykorzystaniu ośmioróżkowej rozdzielcości pomiaru, przetwornik pracuje poprawnie taktowany nawet zegarem 1MHz.

Przetwornik wbudowany w mikrokontroler działa na zasadzie sukcesywnej aproksymacji (kolejnego przybliżenia), czyli doprowadzony do niego sygnał jest porównywany z napięciem przetwornika C/A. Napięcie przetwornika C/A zawiera się w przedziałach od 0 V do wartości napięcia referencyjnego VREF, które doprowadzone jest do przeznaczonego na nie pinu mikrokontrolera, bądź do wartości jednego z wybranych wewnętrznych źródeł tego napięcia. Z każdym kolejnym krokiem iteracyjnym jest ono porównywane przez komparator z napięciem mierzonym. Jeżeli napięcie mierzone jest większe lub równe napięciu dostarczonemu przez przetwornik C/A, na odpowiednim biecie w rejestrze ADCW (16-bitowy rejestr składający się z dwóch 8-bitowych) ustawiana jest jedynka. W przypadku, gdy mierzone napięcie jest mniejsze, ustawiane jest zero. Przechodząc tak przez wszystkie bity dostępne w przetworniku w rejestrze ADCW ostatecznie otrzymywana jest wartość, która odpowiada mierzonemu napięciu wejściowemu [12].

Schemat połączeń mikrokontrolera do wyświetlacza, toru wejściowego, klawiatury oraz poprawnego jego zasilania został narysowany w ewaluacyjnej wersji programu Cadsoft Eagle w komplikacji 7.5.0 i przedstawiony na rysunku 4.3.



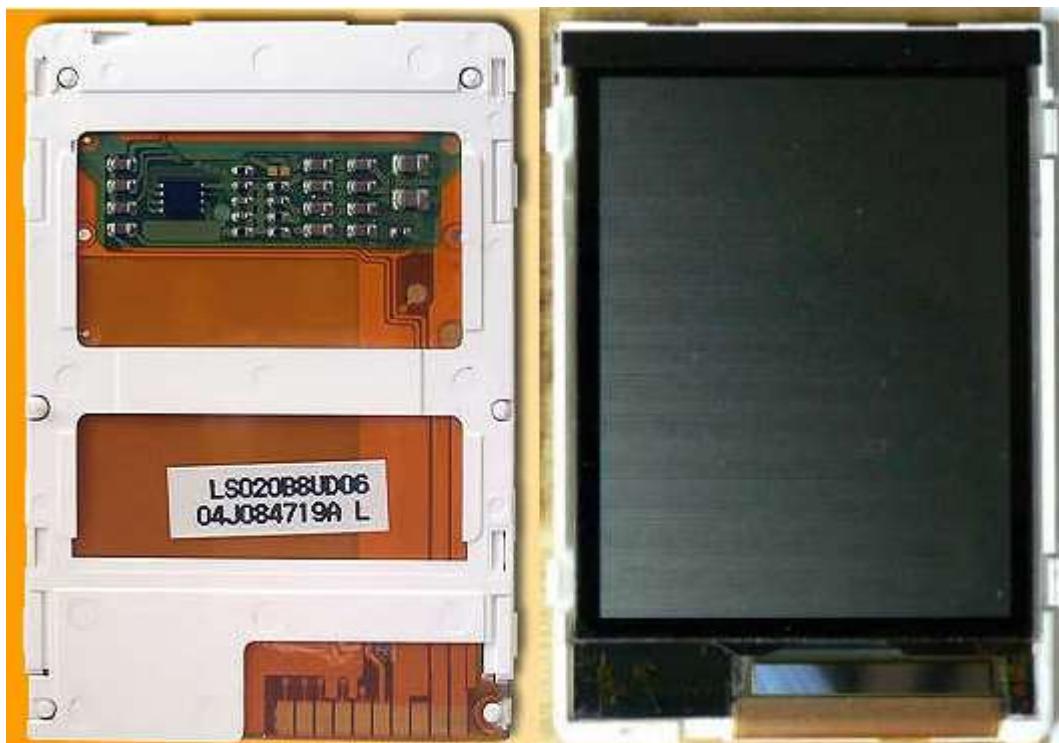
Rys. 4.3. Schemat połączeń peryferii do mikrokontrolera.

Linie odpowiadające za komunikację z wyświetlaczem zostały podłączone do pinów, które wewnętrznie połączone są ze sprzętowym interfejsem SPI, co pozwala na

szybszą transmisję danych, niż w przypadku transmisji programowej z użyciem zwykłych pinów wejścia/wyjścia. Komunikacja z użytkownikiem odbywa się poprzez klawiaturę zbudowaną z czterech przycisków połączonych z liniami PC3 -PC7 mikrokontrolera. Nie są do nich dołączone dodatkowe elementy w postaci pojemności i rezystancji dla wyeliminowania zjawiska drgania styków, gdyż przewidziana została programowa obsługa debouncingu. Do etykiety MEAS\_SIG doprowadzony zostaje mierzony sygnał z toru wejściowego. Dodatkowe rezystancje R3 oraz R2 połączone z rezonatorem kwarcowym eliminują częściowo zjawisko dryfu częstotliwościowego i gwarantują zwiększenie stabilności sygnału zegarowego [10]. Sposób połączenia zasilania i jego filtracji został zaczerpnięty z noty katalogowej mikrokontrolera[12].

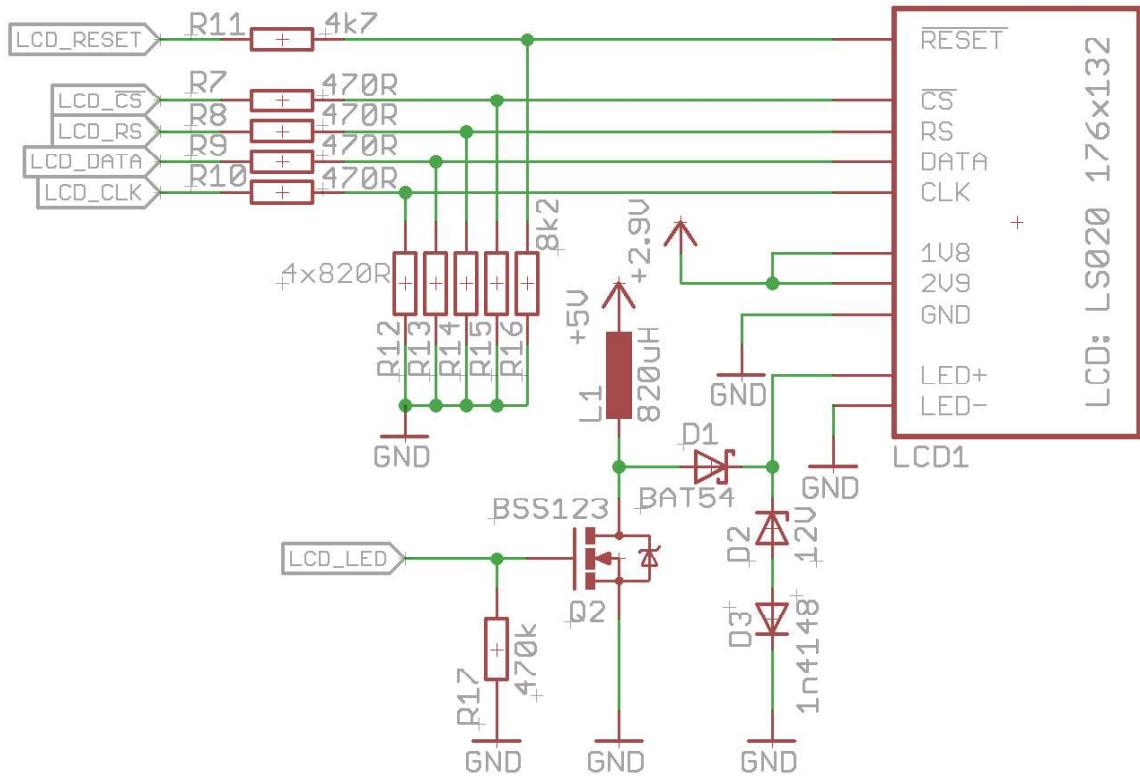
## 4.2 Użyty wyświetlacz LCD

Jako ekran wykorzystany został wyświetlacz o wymiarach 38,2mm x 55,8mm i oznaczeniu LS020B8UD05 montowany w telefonach CX65 firmy Siemens. Posiada on 16 bitową paletę barw, czyli możliwe jest wyświetlenie 65536 odcienni oraz rozdzielcość wynoszącą 132x176 pikseli. Na rysunku 4.3 został przedstawiony jego wygląd.



Rys. 4.4. Wygląd wyświetlacza [16].

Na rysunku 4.6 ukazany został sposób połączenia wyświetlacza z mikrokontrolerem, oraz jego zasilania.



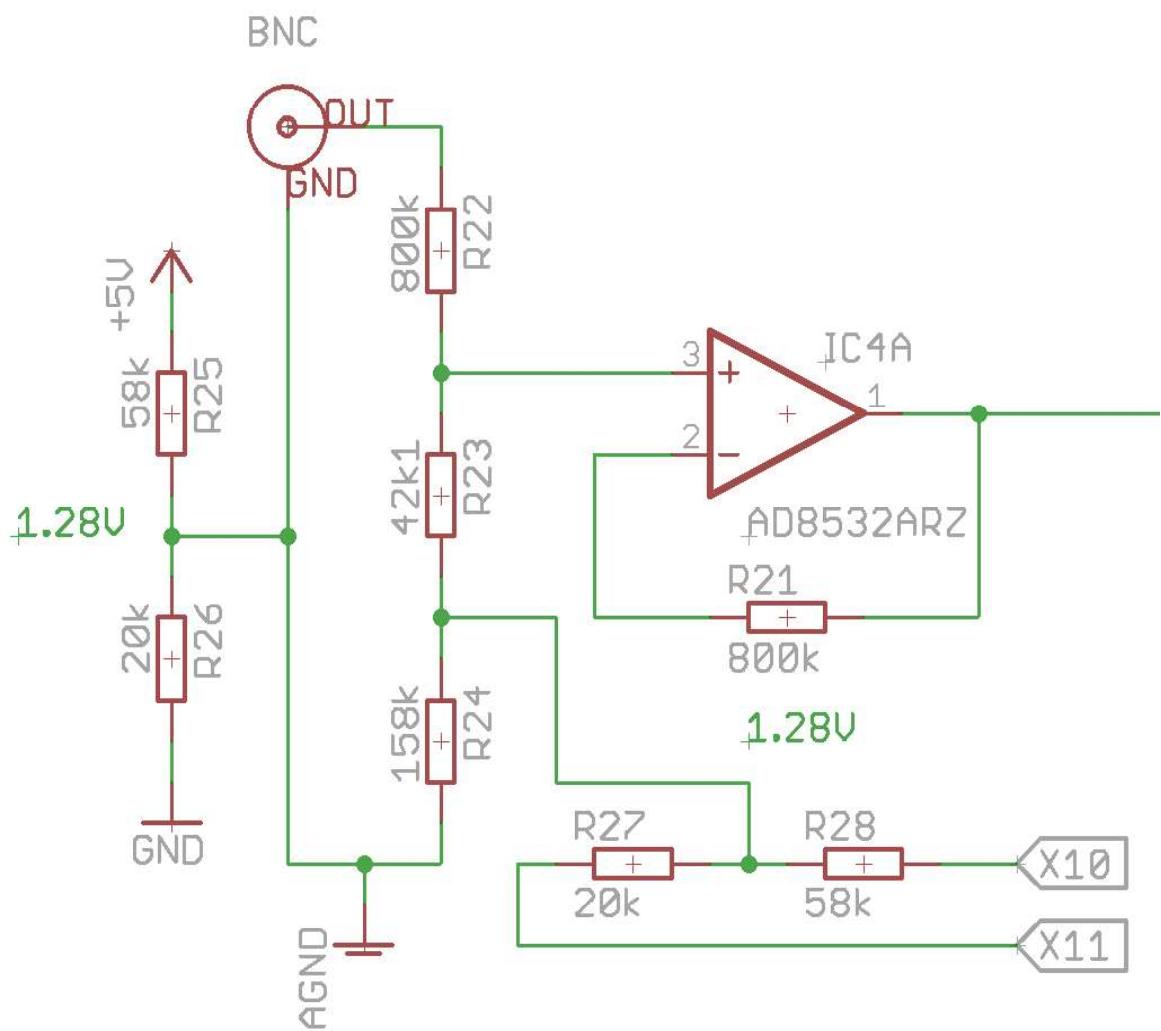
Rys. 4.6. Schemat połączeń wyświetlacza.

Logika wyświetlacza pracująca na napięciach od 0 do 3,3 V, co wymusza dopasowanie poziomów napięć mikrokontrolera pracującego w zakresie 0 - 5 V. Funkcję tę realizują dzielniki napięciowe zbudowane na rezystorach R7 - R16. Zasilanie podświetlenia wymaga dodatkowego zasilania 12V, które zostało wytworzzone przy użyciu prostej przetwornicy podwyższającej napięcie sterowanej z portu mikrokontrolera. Zbudowana została ona z elementów R17, Q2, L1 oraz D1. Diody D2 oraz D3 pełnią rolę zabezpieczenia przed przepięciami. Doprowadzenie do bramki tranzystora sygnału prostokątnego o częstotliwości około 62kHz i wypełnieniu 50% powoduje podbicie napięcia do granicy 12V i pozwala na poprawną pracę podświetlenia wyświetlacza [11], [16].

### 4.3 Tor pomiarowy

Projektując tor należało uwzględnić, aby impedancja wejściowa była typowa dla urządzeń dostępnych na rynku. Dodatkowo niezbędne było zaprojektowanie toru, tak aby

pozwalał on na zmianę tłumienia sygnału wejściowego. Co prawda użyty mikrokontroler posiada w strukturze przetwornika A/C wbudowany wzmacniacz sygnału w sygnale wejściowym dla sygnału różnicowego, jednakże jego budowa nie pozwala na pomiar w ten sposób napięć ujemnych.**[ZRÓDŁO]** Sam przetwornik potrafi zmierzyć napięcia jedynie napięcia w zakresie od dolnej wartości napięcia zasilania, do wartości przyłączonego napięcia referencyjnego. Jako, że w projekcie zostało użyte wewnętrzne napięcie referencyjne 2,56 V, pozwala to na pomiar sygnałów w zakresie 0 - 2,56 V. Ograniczenia te powodują potrzebę zastosowanie sztucznej masy, na poziomie 1,28 V tak, aby móc poprawnie mierzyć dodatnią oraz ujemną połowę doprowadzonego sygnału. Schemat uwzględniający wszystkie powyższe aspekty został przedstawiony na rys. 4.7.



Rys. 4.7. Schemat pierwszego stopnia toru pomiarowego.

Sygnał podawany jest na dwustopniowy dzielik rezystorowy składający się z rezystorów R22, R23 oraz R24. Wartości rezystancji zostały wyliczone z układu równań powstałego z przyjęcia następujących warunków:

- impedancja wejściowa toru ma mieć wartość  $1 \text{ M}\Omega$ ,
- w przypadku pozostawienia linii X10 oraz X11 w stanie wysokiej impedancji, dzielik powstaje z rezystorów R22, R23, R24 ma tłumić sygnał pięciokrotnie,
- w przypadku, gdy linia X10 zostaje podłączona do VCC, a X11 do gnd, dzielik powstaje z rezystorów R22 oraz R23 ma tłumić sygnał dwudziestokrotnie.

Po analizie powyższych warunków powstał następujący układ równań:

$$\left\{ \begin{array}{l} R_{22} + R_{23} + R_{24} = 1000 \\ \frac{R_{23}}{R_{22}+R_{23}} = \frac{1}{20} \\ \frac{R_{23}+R_{24}}{R_{22}+R_{23}+R_{24}} = \frac{1}{5} \end{array} \right. \quad (4.1)$$

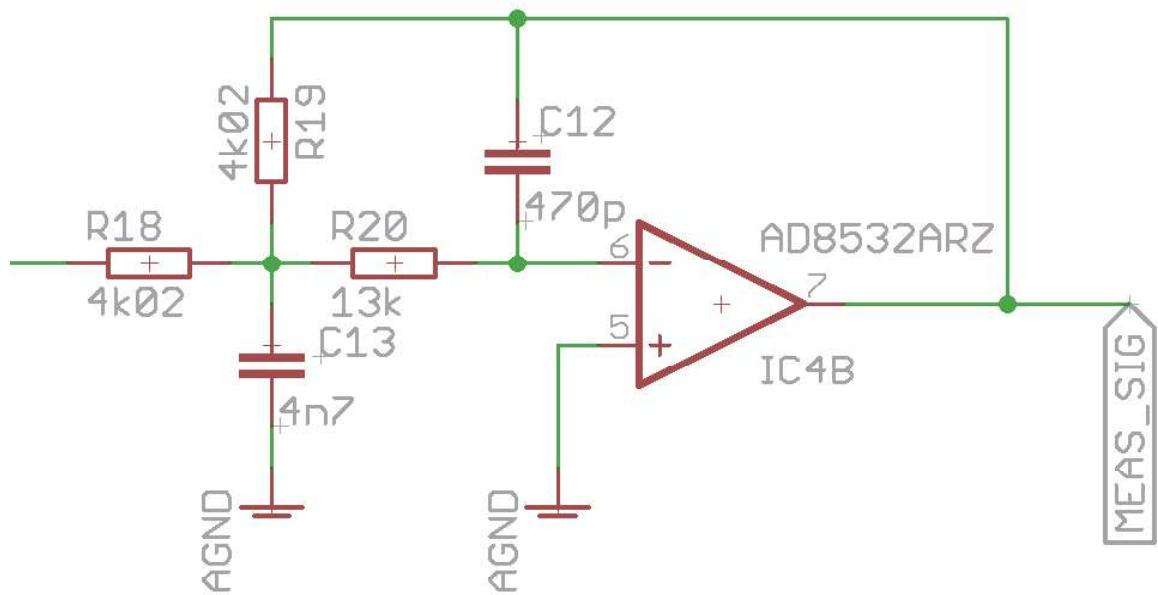
Rozwiążanie tego układu jednoznacznie określa wartości rezystancji, które po skorygowaniu do typoszeregu wynoszą  $800 \text{ k}\Omega$  dla R22,  $42,1 \text{ k}\Omega$  dla R23 oraz  $158 \text{ k}\Omega$  dla R24.

Dzielik zbudowany z rezystorów R25 oraz R26 służy dotworzenia sztucznej masy na poziomie  $1,28\text{V}$  dla sygnału wejściowego. Dodatkowy dzielik utworzony z pary rezystorów R27 i R28 pozwala na przełączanie tłumienia sygnału wejściowego doprowadzonego na wzmacniacz operacyjny. Linie X10 oraz X11 zostały podłączone do linii sterujących mikrokontrolera. Ustawienie X10 w stan wysoki, a X11 niski powoduje odcięcie rezystora R24, i uzyskanie tłumienia rzędu  $26 \text{ dB}$ , co przekłada się na dwudziestokrotne zmniejszenie sygnału wejściowego. Dzięki użyciu takiego rozwiązania maksymalna amplituda mierzonego sygnału wejściowego ma wartość w granicach  $51,2\text{V}$ . W momencie, gdy mikrokontroler wystawi na linię X10 oraz X11 stan wysokiej impedancji, gdzie jest ona rzędu kilku gigaomów, można przyjąć, że dodatkowy dzielik zostaje odłączony i nie ma wpływu na tor pomiarowy. Tłumienie sygnału dla tego przypadku wynosi  $14 \text{ dB}$  i pozwala na mierzenie maksymalnej amplitudy sygnału wejściowego wynoszącej  $12,8\text{V}$ .

Kolejnym stopniem toru pomiarowego jest wtórnik napięciowy zbudowany na wzmacniaczu operacyjnym AD8532ARZ[8]. Jest to wzmacniacz pracujący przy pojedynczym zasilaniu od  $2,7$  do  $6 \text{ V}$ , typu rail to rail, czyli jego napięcie wyjściowe może przyjmować wartości bliskie wartościom zasilania. Charakteryzuje się on niskim poborem

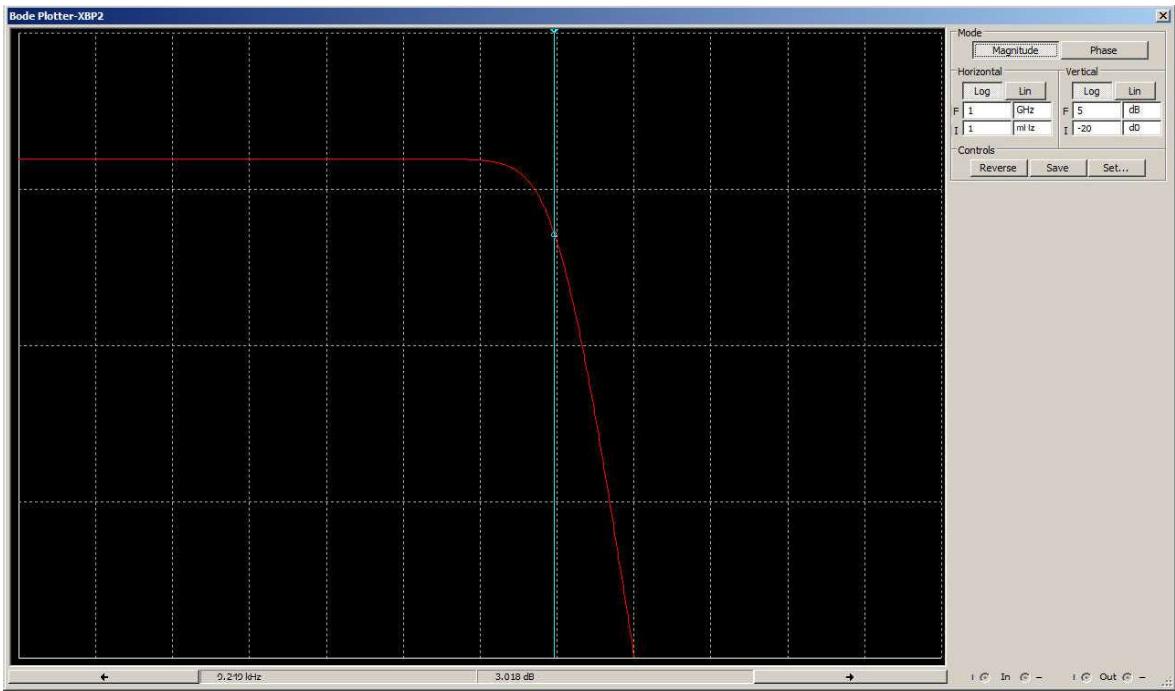
prądu i pasmem przenoszenia równym 3 MHz, oraz bardzo dużym czasem narastania rzędu 5 V/us.

Ostatnim stopniem toru wejściowego jest dolnoprzepustowy filtr antialiasingowy o trzydecybelowym paśmie przenoszenia równym 9,25 kHz oraz tłumieniu 18 dB na oktawę, przedstawiony na rys. 4.8.



Rys. 4.8. Schemat drugiego stopnia toru pomiarowego.

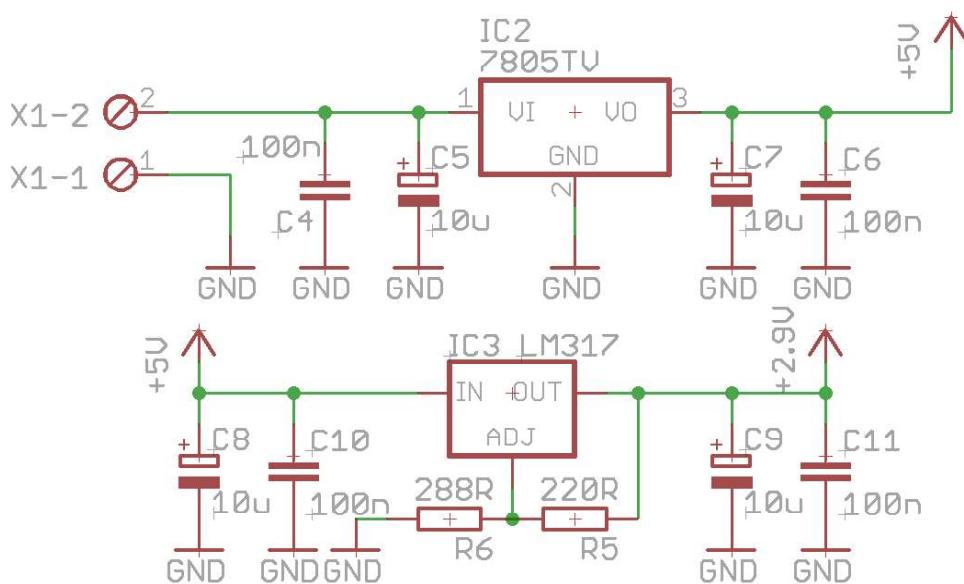
Jest to dolnoprzepustowy filtr aktywny LP-MFP z wielokrotnym sprzężeniem zwrotnym o charakterystyce Czebyszewa i dobroci wynoszącej 0,707. Wartości poszczególnych elementów zostały obliczone w uproszczony sposób [2]. Filtr został zasymulowany w edukacyjnej wersji programu NI Multisim 13.0, gdzie zostało wykreślone jego pasmo przenoszenia przedstawione na rysunku 4.9.



Rys. 4.9. Pasmo przenoszenia filtru drugiego stopnia toru pomiarowego.

#### 4.4 Układ zasilania

Zasilanie zostało zaprojektowane przy użyciu stabilizatorów liniowych. Co prawda powoduje to o wiele większe straty energii niż w przypadku zasilania z przetwornicy, jednakże wprowadza do obwodu o wiele mniejsze zakłócenia. Przyjęte zostało, że urządzenie będzie zasilane z baterii o napięciu 9 V i symbolu 6F22. Schemat układu zasilania został przedstawiony na rys 4.10.



Rys. 4.10. Schemat zasilania urządzenia.

Do wytwarzania napięcia 5 V potrzebnego do zasilania mikrokontrolera i toru pomiarowego użyty został stabilizator napięcia o oznaczeniu 7805 pracujący w standardowej dla niego aplikacji. Dodatkowe napięcie zasilania 2,9 V do zasilania wyświetlacza uzyskane zostało przy użyciu układu LM317 [13]. Napięcie wyjściowe wyznaczone jest według wzoru:

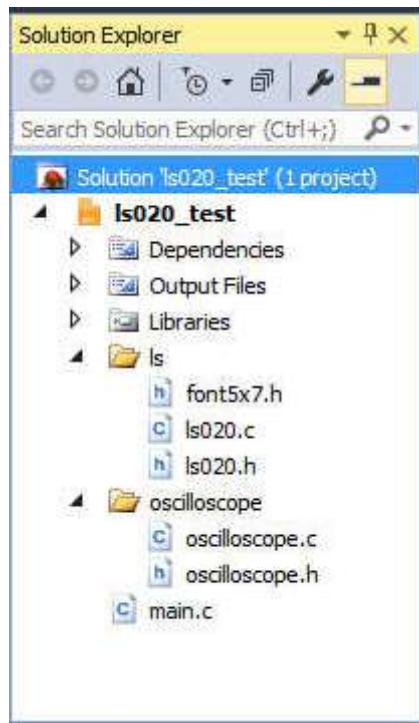
$$U_{wy} = 1.25 \left(1 + \frac{R_6}{R_5}\right) = 1.25 \left(1 + \frac{288}{220}\right) = 2,89 \text{ V} \quad (4.2)$$

Całkowity schemat urządzenia został przedstawiony w załączniku pierwszym.

## 5. Oprogramowanie

### 5.1 Wiadomości ogólne na temat oprogramowania

Program dla mikrokontrolera został napisany w środowisku Atmel Studio w wersji 7.0 w języku C. Pliki programu zostały podzielone na oddzielne pliki odpowiadające za realizację poszczególnych funkcji. Drzewo projektu przedstawia rys. 5.1.



Rys. 5.1. Struktura programu.

Podzielenie programu na oddzielne pliki ma za zadanie zwiększenie czytelności kodu programu. W folderze *ls* znajdują się pliki odpowiedzialne za transmisję danych i obsługę wyświetlacza LCD. Dodatkowo została zaimplementowana obsługa wyświetlania napisów. Folder *oscilloscope* zawiera funkcje programu odpowiedzialne za kwantowanie sygnału wejściowego, generowanie podstawy czasu, obsługę klawiszy oraz wyświetlanie danych. Plik *main.c* jest głównym plikiem programu, gdzie znajduje się początkowa konfiguracja i inicjalizacja peryferii.

Kompilacja została przeprowadzona w kompilatorze Atmel Toolchain w wersji 3.4.2. Kod wynikowy zajmuje 4084 bajtów pamięci flash oraz 1541 bajtów pamięci ram.

## 5.2 Obsługa klawiszy

Klawiatura podłączona do portów PC4 - PC7 mikrokontrolera służy do konfiguracji parametrów urządzenia takich jak podstawa czasu, wzmacnienie, sposób wyzwalania oraz stopień wyzwalania. Obsługa klawiszy została zrealizowana podczas przerwania przy przepełnieniu ośmiobitowego licznika oznaczonego jako timer0. W głównym pliku programu przed startem funkcji odpowiadającej za działanie oscyloskopu, zostają wywołane dwie funkcje inicjujące, które przygotowują procesor do obsługi klawiszy. Zostały one przedstawione na listingu 5.1.

Listing 5.1. Inicjalizacja i konfiguracja klawiszy.

```
1. void timer0_init(void){
2.     TCCR0B = (1<<CS02) | (1<<CS00);
3.     TIMSK0 =(1<<TOIE0);
4. }
5. void keys_init(void){
6.     DDR(KEY_PORT) &= ~(KEY_1|KEY_2|KEY_3|KEY4); // as outputs
7.     PORT(KEY_PORT)|=(KEY_1|KEY_2|KEY_3|KEY_4); // pull ups
8. }
```

Funkcja *keys\_init()* konfiguruje piny procesora do których zostały podłączone klawisze jako wejścia podciągnięte poprzez wewnętrzne wbudowane w mikrokontroler rezistory do VCC. Pozwala to na sterowanie rejestrem PINC przez zadanie stanu niskiego na dany pin. Funkcja *timer0\_init()* ustawia timer w taki sposób, aby był taktowany zegarem mikrokontrolera podzielonym przez 1024 i zgłaszał przerwanie w momencie przepełnienia licznika. Oznacza to, że przerwanie będzie wykonywane z częstotliwością około 92 razy na sekundę [1], [4].

Stan klawiszy sprawdzany jest w przerwaniu od przepełnienia licznika w sposób przedstawiony na listingu 5.2.

Listing 5.2. Obsługa klawiszy w przerwaniu

```
1. ISR (TIMER0_OVF_vect){
2.     if((~PIN(KEY_PORT))&(KEY_1|KEY_2|KEY_3)){
3.         timer1_disable();
4.         draw_flag =1;
5.         key_flag=1;
6.         _delay_ms(25);
7.         if((~PIN(KEY_PORT))&(KEY_1|KEY_2|KEY_3)){
8.             key_status=(~PINC);
```

```

9.           while((~PIN(KEY_PORT))&(KEY_1|KEY_2|KEY_3));
10.      }
11.  }
12. }
```

Podczas każdego wykonywania przerwania sprawdzane jest, czy którykolwiek z klawiszy został naciśnięty. Jeżeli jest to prawdą zatrzymywane jest zbieranie bufora z próbkami i wywoływanie natychmiastowe wyrysowanie przebiegu na wyświetlaczu LCD. Ustawiana jest także zmienna *key\_flag*, zgłaszająca do programu, że któryś z klawiszy został wciśnięty. W dalszej kolejności oczekiwany jest czas 25 ms w celu programowej eliminacji wpływu drgań styków na zachowanie klawiszy i po raz drugi sprawdzane jest, czy którykolwiek z klawiszy został wciśnięty. Jeżeli tak, wartość rejestru PINC przepisywana jest do zmiennej *key\_status* i uruchomiona zostaje pętla oczekująca do momentu zwolnienia wszystkich klawiszy. Ostatnim etapem jest analiza zamiennej *key\_status* w głównej funkcji oscyloskopu i odpowiednie zmodyfikowanie tablicy przechowującej ustawienia urządzenia.

### 5.3 Generator podstawy czasu

Generator podstawy czasu został zrealizowany przy użyciu szesnastobitowego licznika oznaczonego jako timer1. Inicjalizacja timera została przedstawiona na listingu 5.3.

Listing 5.3. Inicjalizacja timera 1.

```

1. void timer1_init(void){
2.   TIMSK1 = (1<<OCIE1A); // interrupt output compare A
3.   TCCR1B = (1<<WGM12); // CTC mode
4. }
```

Licznik zostaje skonfigurowany do pracy w trybie CTC co oznacza, że z każdym jego uruchomieniem zaczyna liczyć od 0 w górę i z każdą kolejną iteracją porównuje wartość z wartością zapisaną w rejestrze OCR1A. W momencie stwierdzenia zgodności tych wartości wywoływanie zostaje przerwanie, a licznik zostaje zerowany. Konfiguracja taka pozwala na precyzyjne odmierzanie czasu pomiędzy kolejnymi wywoływaniami procedury obsługi przerwania [1], [4], [12].

Odpowiednie wywoływanie kolejnych konwersji przetwornika poprzez przetwornik A\C wymaga przeliczenia wartości potrzebnych do wpisania w rejestrze OCR1A. Dla

lepszego zobrazowania obliczeń przyjmijmy, że na ekranie zostanie wyświetlony przebieg o częstotliwości 31.25 Hz. Czas trwania okresu takiego przebiegu wynosi 32 ms. Ekran oscyloskopu posiada 8 działek, gdzie w każdej zostaje wyrysowane 120 próbek. Oznacza to, że aby poprawnie wyświetlić taki przebieg należy dokonywać kolejne konwersje co 200 us. Sposób obliczenia wartości rejestru OCR1A, w sposób spełniający powyższe warunki przedstawia równanie (RÓWNAINIAE)

$$OCR1A = \left( \frac{F_{CPU}}{N * FREQ} \right) - 1 = \left( \frac{24 * 10^6}{2 * 5000} \right) = 2399 \quad (5.1)$$

Zmienna F\_CPU określa częstotliwość taktowania mikrokontrolera, N jest wartością prescalera ustawionego w liczniku, a FREQ pożądaną częstotliwością wywoływania przerwania. Podstawiając wartości liczbowe do równania otrzymana została wartość 2399, która wpisana do rejestru OCR1A i uruchomienia licznika, będzie wywoływać przerwanie z częstotliwością 5 kHz. W taki sposób zostały przeliczone wszystkie wartości dla różnych podstaw czasu potrzebnych w urządzeniu i przedstawione w tabeli 5.1.

Tabela 5.1. Wartości rejestru OCR1A.

Podstawa czasu	Okres przerwań	Częstotliwość przerwań	Prescaler	Wartość OCR1A
50 ms	2,5 ms	400 Hz	2	29999
20 ms	1 ms	1 kHz	2	11999
10 ms	500 us	2 kHz	2	5999
5 ms	250 us	4 kHz	2	2999
2 ms	100 us	10 kHz	2	1199
1 ms	50 us	20 kHz	2	599
500 us	25 us	40 kHz	2	299
200 us	10 us	100 kHz	2	119
100 us	5 us	200 kHz	2	59

Obliczone wartości umieszczone zostały w tablicy `ocr_values[]` przedstawionej na listingu 5.4.

Listing 5.4. Tablica wartości rejestru OCR oraz funkcja kwantująca sygnał.

```
1.                                     // 50ms/div 20ms 10ms 5ms 2ms 1ms ...
2. const uint16_t ocr_values[] = {59999, 23999, 11999, 5999, 2399, 1199, 599
   , 239, 119, 59 };
3.
4. void fill_probe_buffer(uint16_t timebase){
5.     timer1_disable();
6.     OCR1A = ocr_values[timebase];
7.     probe_counter = 0;
8.     timer1_enable();
9. }
```

Listing 5.4 dodatkowo przedstawia funkcję *fill\_probe\_buffer()*, która korzystając z tablicy *ocr\_values[]* ustawia wartość rejestru OCR1A oraz uruchamia timer. Wywoływana jest ona przed rozpoczęciem każdej sekwencji zebrania tablicy próbek. Początkowe wyłączenie timera przed dokonaniem wpisania wartości do rejestru wymuszone jest atomową obsługą klawiszy. Oznacza to, iż sprawdzenie warunków wcisnięcia klawiszy i zmiana parametrów urządzenia może wystąpić w dowolnym momencie wykonywania programu. Wyłączenie licznika na czas wpisywania wartości do rejestrów oraz wyzerowanie licznika zebranych próbek, powoduje uniemożliwienie sytuacji, gdzie podczas trwania sekwencji zbierania próbek nagle zostanie zmieniona wartość rejestru OCR1A, a co za tym idzie zmiana podstawy czasu. W przypadku zaistnienia takiej kolizji licznik i tak w dalszej kolejności jest zerowany, a cała sekwencja kwantowania zostaje powtórzona od zerowej próbki.

## 5.4 Kwantyzacja sygnału

Badany sygnał doprowadzony jest do wejścia PA0 mikrokontrolera, które wewnętrznie połączone jest z przetwornikiem A/C. Parametry pracy przetwornika konfigurowane są poprzez odpowiednie ustawienie rejestrów konfiguracyjnych. Za inicjalizację przetwornika i konfigurację jego parametrów odpowiedzialna jest funkcja *adc\_init()* przedstawiona na listingu 5.5.

Listing 5.5. Inicjalizacja przetwornika ADC.

```
1. void adc_init(void){
2.     DDRA &=~ (1<<PA0);
3.     ADMUX |= (1<<REFS1)|(1<<REFS0)|(1<<ADLAR);
4.     ADCSRA |= (1<<ADEN)|(1<<ADPS2)|(1<<ADPS0)|(1<<ADATE)|(1<<ADSC);
5. }
```

Pin PA0 zostaje ustawiony jako wejście bez rezystora podciągającego co oznacza, że trakcie dokonywania konwersji, na tym pinie panuje stan wysokiej impedancji. W rejestrze ADMUX skonfigurowane zostało wewnętrzne źródło napięcia referencyjnego 2,56 V oraz przesunięcie wyniku konwersji do lewej strony w rejestrach ADCL oraz ADCH. Pozwala to na szybszy odczyt bardziej znaczącego bajtu wyniku konwersji niż w normalnym przypadku. Idea tego rozwiązania została przedstawiona na rysunku 5.1.

a)									
<b>ADLA</b>									
<b>R</b>	Bit (0x79)	15	14	13	12	11	10	9      8	
=	(0x78)	-	-	-	-	-	ADC9	ADC8	
<b>0</b>	Read/Write	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADC1      ADC0	
		7	6	5	4	3	2	1      0	
		R	R	R	R	R	R	R      R	
		R	R	R	R	R	R	R      R	
	Initial Value	0	0	0	0	0	0	0      0	
		0	0	0	0	0	0	0      0	
b)									
<b>ADLA</b>									
<b>R</b>	Bit (0x79)	15	14	13	12	11	10	9      8	
=	(0x78)	ADC9	ADC8	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2
<b>1</b>	Read/Write	ADC1	ADC0	-	-	-	-	-	-
		7	6	5	4	3	2	1      0	
		R	R	R	R	R	R	R      R	
		R	R	R	R	R	R	R      R	
	Initial Value	0	0	0	0	0	0	0      0	
		0	0	0	0	0	0	0      0	

Rys. 5.1. Ilustracja prezentująca sposób umieszczania danych w rejestrze ADCX [12].

Jak widać ustawienie bitu ADLAR w rejestrze ADMUX skutkuje umieszczeniem starszego bajtu wyniku konwersji bezpośrednio w rejestrze ADCH. Obsługa odczytu wartości w takim przypadku polega jedynie na bezpośrednim przepisaniu wartości z rejestru ADCH do pamięci ram, gdzie znajduje się tablica z kolejnymi próbками. Gdy bit ADLAR nie jest ustawiony, aby odczytać wartość starszego bajtu konwersji należałyby dokonywać dodatkowe operacje logiczne w postaci przesunięcia oraz sumy bitowej. Użycie tych operacji skutkowałoby zwiększeniem zajętości pamięci ram oraz dłuższym wykonywaniem kodu programu.

Kolejnym krokiem konfiguracji jest ustawienie zegara taktującego przetwornik. Według dokumentacji maksymalna częstotliwość taktująca wynosi 200 kHz. Wartość ta

odnosi się do prawidłowego przeprowadzania konwersji na pełnych dziesięciu bitach rozdzielczości. W przypadku gdy wykorzystywana jest mniejsza rozdzielczość przetwornik pracuje poprawnie nawet dla większych częstotliwości taktowania. W urządzeniu prescaler doprowadzający sygnał zegarowy do przetwornika został ustawiony na wartość 32. Częstotliwość taktowania przetwornika oblicza się według wzoru 5.2 [1], [4], [12].

$$F_{ADC} = \frac{F_{CPU}}{ADC_{PRESCALER}} = \frac{24*10^6}{32} = 750 \text{ kHz} \quad (5.2)$$

Z obliczeń wynika, że sygnał taktujący przetwornika ma częstotliwość 75 kHz. W rejestrze ADCSRA dodatkowo konfiguracja wymusza na przetworniku aby działał w trybie "free run", gdzie kolejne konwersje następują same po sobie, a wartości poprzednich są nadpisywane. Nadpis jest blokowany jedynie na moment odczytu wartości z rejestru ADCH. W trybie free run czas konwersji trwa 13.5 taktu zegarowego. Podstawiając więc do wzoru 5.3 tę wartość otrzymamy maksymalną ilość konwersji zdolną do wykonania przez przetwornik w czasie jednej sekundy.

$$SPS = \frac{F_{ADC}}{CONV\_CYCLES} = \frac{75*10^4}{13,5} = 55555 \quad (5.3)$$

Ostatecznie zgodnie z twierdzeniem Kotielnikowa-Shannona można obliczyć, że maksymalna częstotliwość sygnału, który można odtworzyć z próbek zebranych przez przetwornik jest równa około 27,5 kHz. W praktyce ze względu na sposób wyświetlania przebiegu na zaprojektowanym urządzeniu, częstotliwość ta wynosi około 5 kHz.

Sposób sprzągnięcia ze sobą algorytmu realizującego generator podstawy czasu oraz kwantowanie został przedstawiony na listingu 5.6.

Listing 5.6. Obsługa przerwania od timera 1 oraz zbieranie próbek.

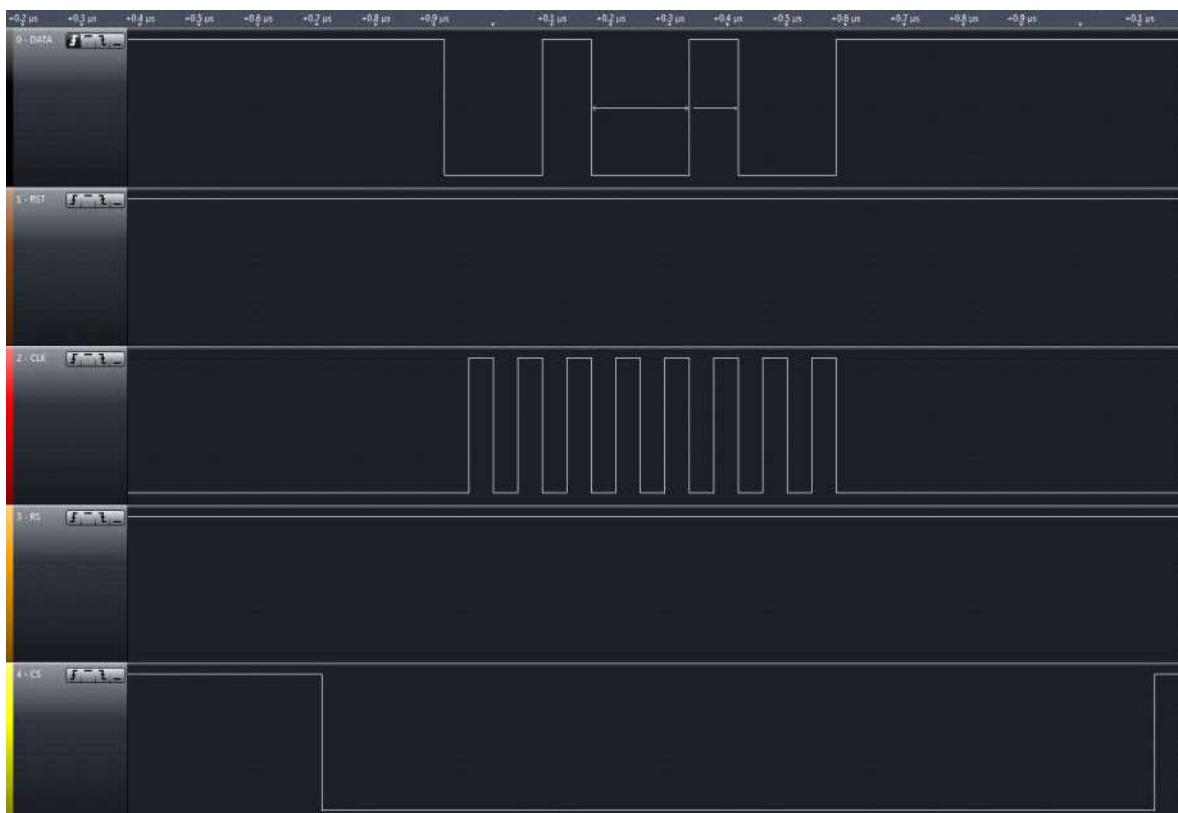
```

1. ISR (TIMER1_COMPA_vect)
2. {
3.     if(probe_counter==1024){
4.         timer1_disable();
5.         draw_flag=1;
6.     }
7.     adc_buf[probe_counter] = ADCH;
8.     probe_counter++;
9.
10. }
```

Całym zadaniem zajmuje się przerwanie od timera1. Na początku sprawdzany jest warunek, czy licznik próbek nie wskazuje poza tablice z próbkami. Jeżeli warunek jest prawdziwy sekwencja zbierania próbek zostaje przerwana i ustawiona zostaje flaga *draw\_flag*, która przekazuje informacje do głównego kodu programu, że może zająć się obróbką i wyświetlaniem danych na ekranie. W momencie, gdy warunek nie jest spełniony wartość rejestru ADCH, w którym znajduje się wynik konwersji, przepisywana jest do tablicy z próbkami *adc\_bufor*, a licznik wskazujący na bieżący element w tablicy zostaje inkrementowany.

## 5.5 Obsługa wyświetlacza

Użyty w projekcie wyświetlacz nie posiada dostępnej dokumentacji, gdyż zastrzeżona jest u producenta. Aby napisać procedury jego obsługi posłużono się analizatorem stanów logicznych firmy Saleae, którym przechwycona została ramka komunikacji telefonu Siemens CX65 z wyświetlaczem. Przechwycona ramka przedstawiona została na rys 5.2.



Rys. 5.2. Przechwycona ramka komunikacji między wyświetlaczem a telefonem.

Na pierwszy rzut oka widoczne jest, że przebieg przedstawiony w pierwszym rzędzie przedstawia linię danych, trzeci linię sygnału zegarowego, a ostatni sygnału enable. Transmisja przeprowadzana jest w szeregowo, gdzie dane wysyłane są wraz ze zboczem

narastającym sygnału zegarowego. W ciągu jednej ramki przesyłany jest zawsze jeden bajt, a aktywowanie transmisji odbywa się poprzez ustawieniu na linii enable stanu niskiego. Dalsza analiza przechwyconego sygnału pozwoliła określić, że linia druga odpowiada sygnałowi resetu wyświetlacza, a czwarta określa czy wysłany bajt jest danymi, czy też komendą. Tak przeprowadzona analiza pozwoliła napisać procedurę wysłania bajtu do wyświetlacza przedstawioną na listingu 5.7, wraz z funkcją inicjalizacyjną.

Listing 5.7. Funkcje obsługujące podstawowe procedury wyświetlacza.

```

1. void spi_init(void){
2.     //ports configuration
3.     LCD_RS_DDR |= (1<<LCD_RS);           // RS pin as output
4.     LCD_RESET_DDR |= (1 << LCD_RESET);    // RESET pin as output
5.     LCD_CS_DDR |= (1 << LCD_CS);         // CS pin as output
6.     LCD_DATA_DDR |= (1 << LCD_DATA);      // DATA pin as output
7.     LCD_CLK_DDR |= (1 << LCD_CLK);        // CLK pin as output
8.
9.     // SPI interface configuration
10.    SPCR = (1<<MSTR) | (1<<SPE);       // master mode, SPI enable
11.    SPSR = (1<<SPI2X);                  // double SPI frequency
12. }
13.
14. void spi_send_byte(uint8_t byte){
15.     SPDR = byte;                         // send data
16.     while(!(SPSR & (1 << SPIF)));       // wait for end transfer
17. }
```

Funkcja *spi\_init()* konfiguruje wewnętrzny interfejs SPI dostępny w mikrokontrolerze do współpracy z wyświetlaczem. Wszystkie linie danych zostają ustawione jako wyjścia po czym aktywowana zostaje transmisja wraz z podwojeniem maksymalnego sygnału zegarowego który ją taktuje. Funkcja *spi\_send\_byte()* jako parametr przyjmuje wartość która ma zostać wysłana przez interfejs SPI do wyświetlacza i umieszcza ją w rejestrze SPDR, co powoduje automatyczne wystartowanie transmisji. Następnie oczekuje na zakończenie transmisji i opuszcza funkcję. Przy użyciu tak przygotowanej procedury napisane zostały kolejne funkcje wysyłające dane oraz komendy przedstawione na listingu 5.8.

Listing 5.8. Funkcje komunikacyjne z wyświetlaczem.

```

1. void lcd_command_byte(uint8_t command_byte){
2.     LCD_RS_PORT |= (1<<LCD_RS);          // send as command
```

```

3.     LCD_CS_PORT &= ~(1<<LCD_CS);           // transfer enable
4.     spi_send_byte(command_byte);             // send byte
5.     LCD_CS_PORT |= (1<<LCD_CS);            // transfer disable
6. }
7.
8. void lcd_data_byte(uint8_t data_byte){
9.     LCD_RS_PORT &= ~(1<<LCD_RS);           // send as data
10.    LCD_CS_PORT &= ~(1<<LCD_CS);          // transfer enable
11.    spi_send_byte(data_byte);               // send data byte
12.    LCD_CS_PORT |= (1<<LCD_CS);            // transfer disable
13. }

```

Funkcje te są bardzo do siebie podobne. Obie na czas transmisji ustawiają linię enable w stan niski i wywołują procedurę wysłania bajtu. Różnica pomiędzy nimi polega, na zmianie linii RS w stan wysoki dla wysyłania komendy i niski dla wysyłania danych. Tak przygotowane funkcje wystarczą do poprawnej implementacji inicjalizacji oraz funkcji graficznych obsługujących wyświetlaczy.

## 5.6 Wyzwalanie, obróbka danych i ich wyświetlanie

Po zebraniu bufora przechowującego próbki w tablicy *adc\_buf[]* o rozmiarze 1024 bajtów niezbędne jest jej przeanalizowanie pod względem wystąpienia warunków wyzwalania i poprawne wyświetlenie przebiegu na ekranie. Jako, że wyzwalanie może nastąpić w dowolnym indeksie próbki począwszy od 0 do 863, należy wyznaczyć offset, który określi od jakiego momentu ma być kreślony przebieg. Ostatnim indeksem musi być liczba nie większa niż 863, gdyż na wyświetlaczu w jednym przebiegu rysowane jest 160 próbek. Zwiększenie tej wartości powodowałoby przekroczenie odczytu bufora z próbками i rysowanie losowych wartości z pamięci ram. Fragment funkcji ilustrującej ideę wyzwalania przedstawiony został na listingu 5.9.

Listing 5.9. Obsługa wyzwalania oscyloskopu.

```

1. void trigger(void){
2.     max_adc=adc_buf[0];
3.     min_adc=adc_buf[0];
4.     uint8_t trigger=0;
5.
6.     for(int cnt=0;cnt<1024;cnt++){
7.         if(adc_buf[cnt]>max_adc) max_adc=adc_buf[cnt];

```

```

8.         if(adc_bufor[cnt]<min_adc) min_adc=adc_bufor[cnt];
9.     }
10.    if(max_adc != min_adc)
11.        trigger = (((max_adc - min_adc)/2)+ min_adc);
12.    else
13.        trigger = max_adc;
14.
15.    for(int cnt=1;cnt<863;cnt++){
16.        if((adc_bufor[cnt]>trigger+settings[2])&&(adc_bufor[cnt-
1]<trigger-settings[2])){
17.            probe_offset=cnt;
18.            break;
19.        }
20.    }
21. }
```

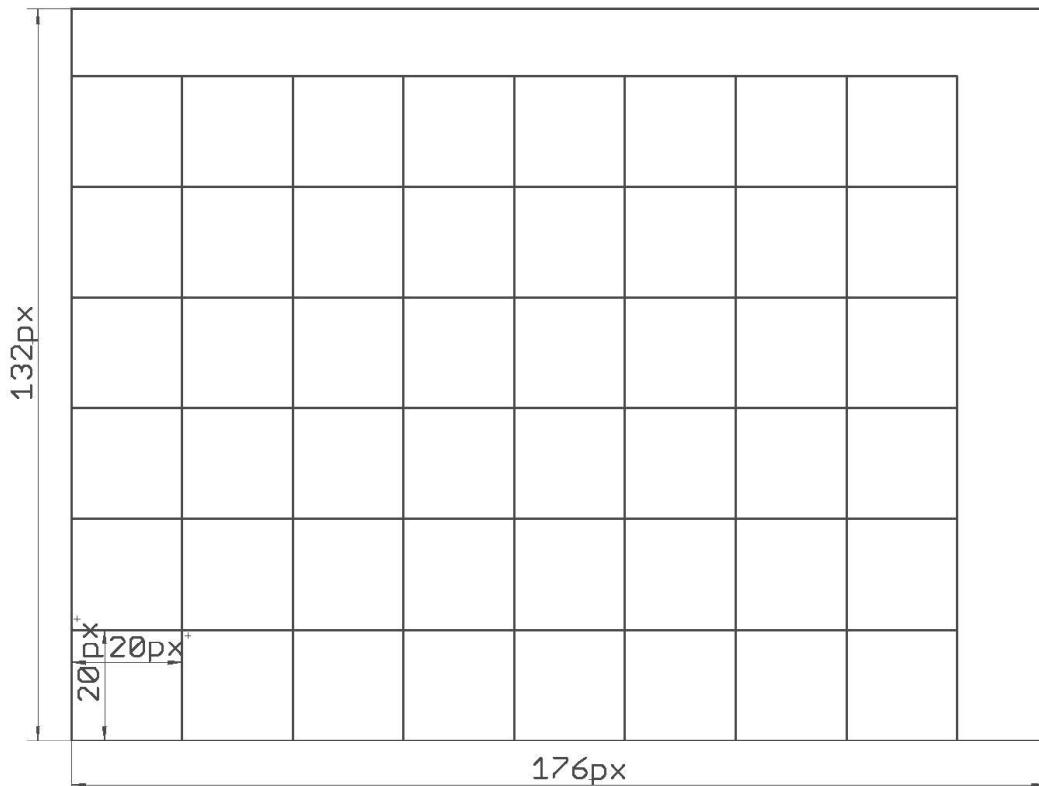
Początkowo tablica próbek jest przeszukiwana w celu odnalezienia największej oraz najmniejszej zmierzonej wartości, które umieszczane są w zmiennych *adc\_max* oraz *adc\_min*. Następnie z tych wartości wyznaczana jest wartość pośrednia między nimi i zapisywana w zmiennej *trigger*, która w rzeczywistości jest poziomem wyzwalania. Urządzenie nie daje możliwości zmiany poziomu wyzwalania wyświetlanego sygnału i jest to zawsze jego wartość średnia. W kolejnej pętli operującej na tablicy próbek, sprawdzany jest warunek wyzwalania. Sprawdza on, czy dwie następujące po sobie próbki są kolejno większe i mniejsze od poziomu wyzwalania o wartość głębokości wyzwalania zapisanego w tablicy *settings[]*. Im mniejsza jest ta wartość, tym wyzwalanie cechuje się większą czułością. Jeżeli warunek zostanie spełniony offset zostaje ustawiony na bieżącą próbkę i pętla jest przerwana. Przedstawiony algorytm odpowiada wyzwalaniu zboczemu narastającemu sygnału. Dla zbocza opadającego funkcja będzie się różnić jedynie kolejnością sprawdzania próbek.

Kolejnym etapem jest przygotowanie wyświetlacza do wyrysowania przebiegu. Ekran zostaje podzielony na 8 działek w poziomie oraz 6 w pionie. Podział taki spowodowany jest jego rozdzielcością wynoszącą 132x176 pikseli. Na każdą działkę przypada 20 pikseli w pionie oraz poziomie. Funkcja *lcd\_draw\_grid()* przedstawiona na listingu 5.10 przygotowuje wyświetlacz poprzez narysowanie na nim siatki składającej się z 16 linii.

Listing 5.10. Funkcja rysująca podziałki na ekranie oscyloskopu.

```
1. void lcd_draw_grid(void){  
2.     for(uint8_t a=0;a<160;a=a+20){  
3.         lcd_line(a,0,a,119); // vertical  
4.     }  
5.     for(uint8_t a=0;a<120;a=a+20){  
6.         lcd_line(0,a,159,a); // horizontal  
7.     }  
8. }
```

Sposób podziału dodatkowo został przedstawiony na rys 5.3.



Rys. 5.3. Sposób podziału siatki na oscyloskopie.

Na tak przygotowanym wyświetlaczu w dalszej kolejności następuje rysowanie przebiegu poprzez funkcję `lcd_draw_scope()`, która została ukazana na listingu 5.11.

Listing 5.11. Funkcja rysująca przebieg na ekranie oscyloskopu.

```
1. void lcd_draw_scope(uint8_t *data, uint8_t colour, uint8_t background){  
2.     int8_t a=0;  
3.     for (uint8_t index=0; index<160; index++){
```

```

4.         lcd_pixel(index,lcd_bufor[index],BLACK_COLOUR);
5.     }
6.
7.     for ( uint8_t index=0; index<160;index++){
8.         a=adc_bufor[probe_offset+index]/2;
9.         lcd_pixel(index,a,WHITE_COLOUR);
10.        lcd_bufor[index]=a;
11.    }
12.

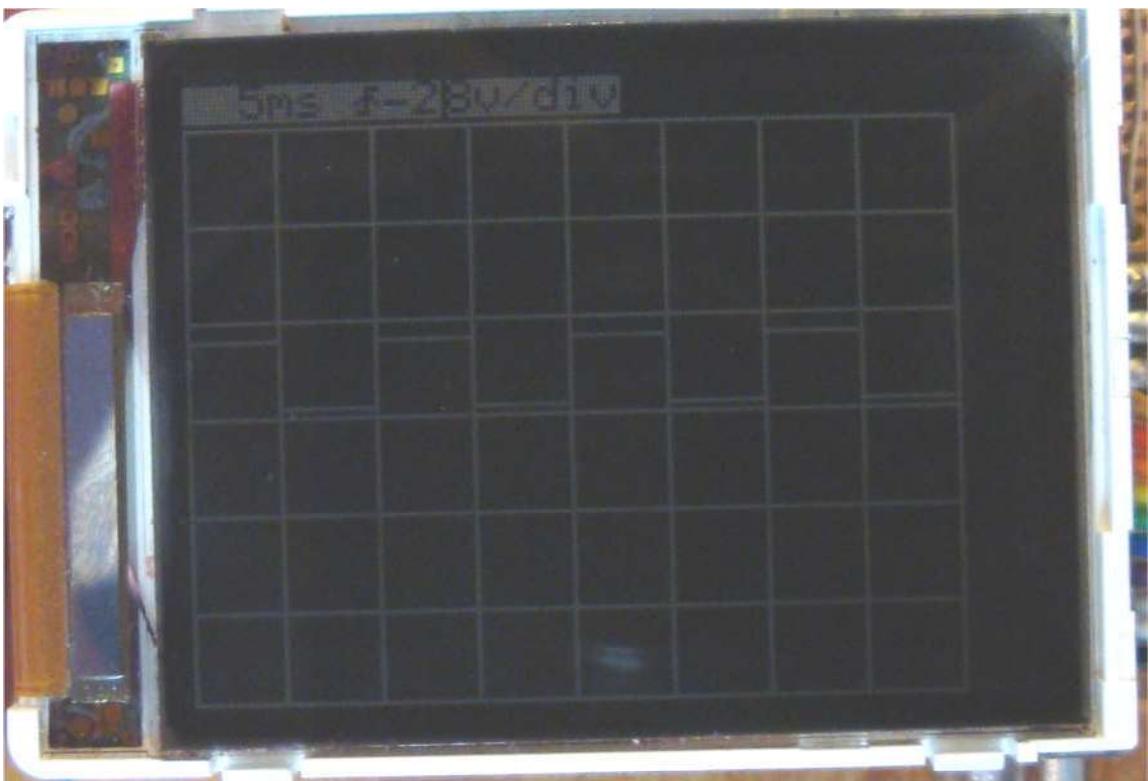
```

W pierwszej kolejności pętla czyści wyświetlacz z przebiegu narysowanego w poprzednim cyklu wyświetlania. Wartości poprzedniego cyklu przechowywane są w tablicy *lcd\_bufor[]*, która jest uzupełniana w momencie rysowania przebiegu na wyświetlaczu. Kolejna pętla odczytuje następujące po sobie wartości z bufora próbek uwzględniają przy tym wyznaczony przez funkcję *trigger()* offset. Następnie wartości są formatowane i kreślone na wyświetlaczu przy jednoczesnym zapisie ich w tablicy *lcd\_bufor[]*. Użycie takiego rozwiązania znacznie przyspiesza odświeżanie ekranu. W przypadku czyszczenia całego wyświetlacza niezbędne byłoby wysłanie ponad 23 kB danych do wyświetlacza, a tak wysyłane jest jedynie około 800 bajtów.

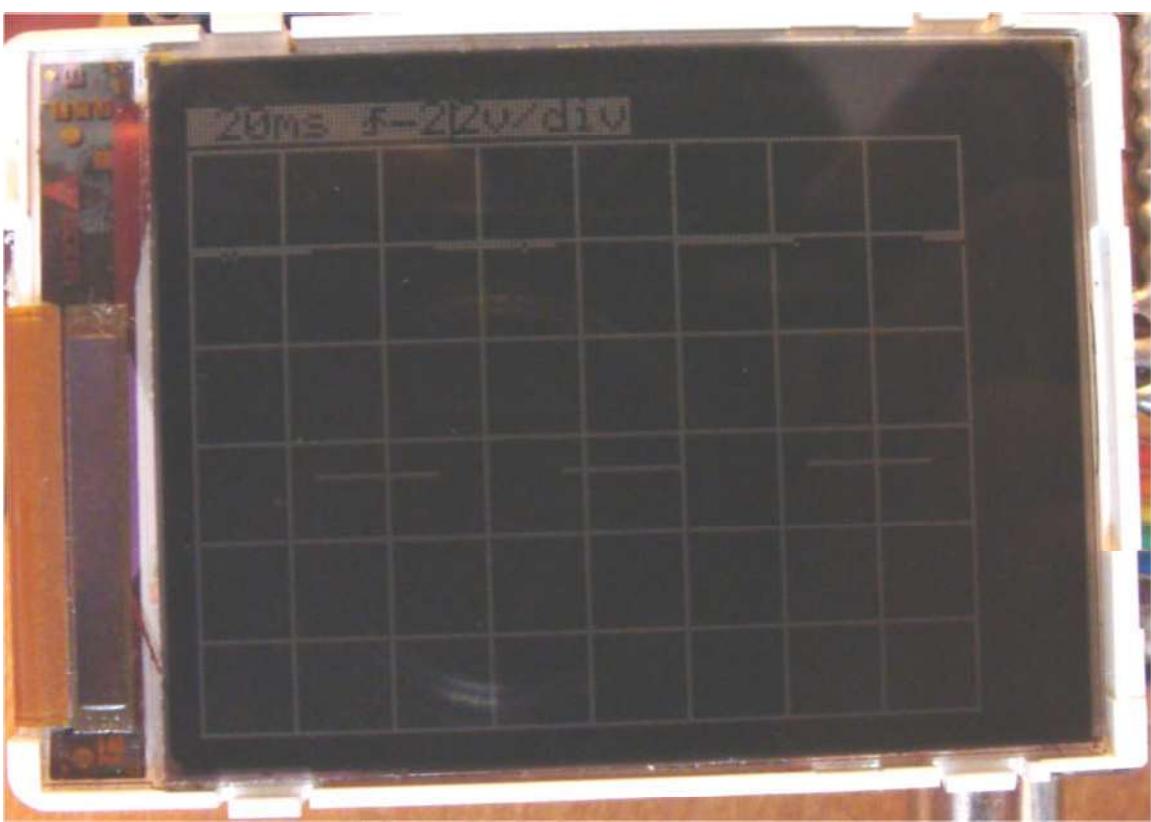
Ostatecznie wywoływana zostaje funkcja *lcd\_draw\_settings()* wypisująca bieżące parametry konfiguracyjne oscyloskopu w wolnych miejscach na wyświetlaczu. Aby było to możliwe konieczna była implementacja obsługi czcionek w bibliotece wyświetlacza. Funkcja ta oraz reszta kodu programu została umieszczona w załączniku drugim.



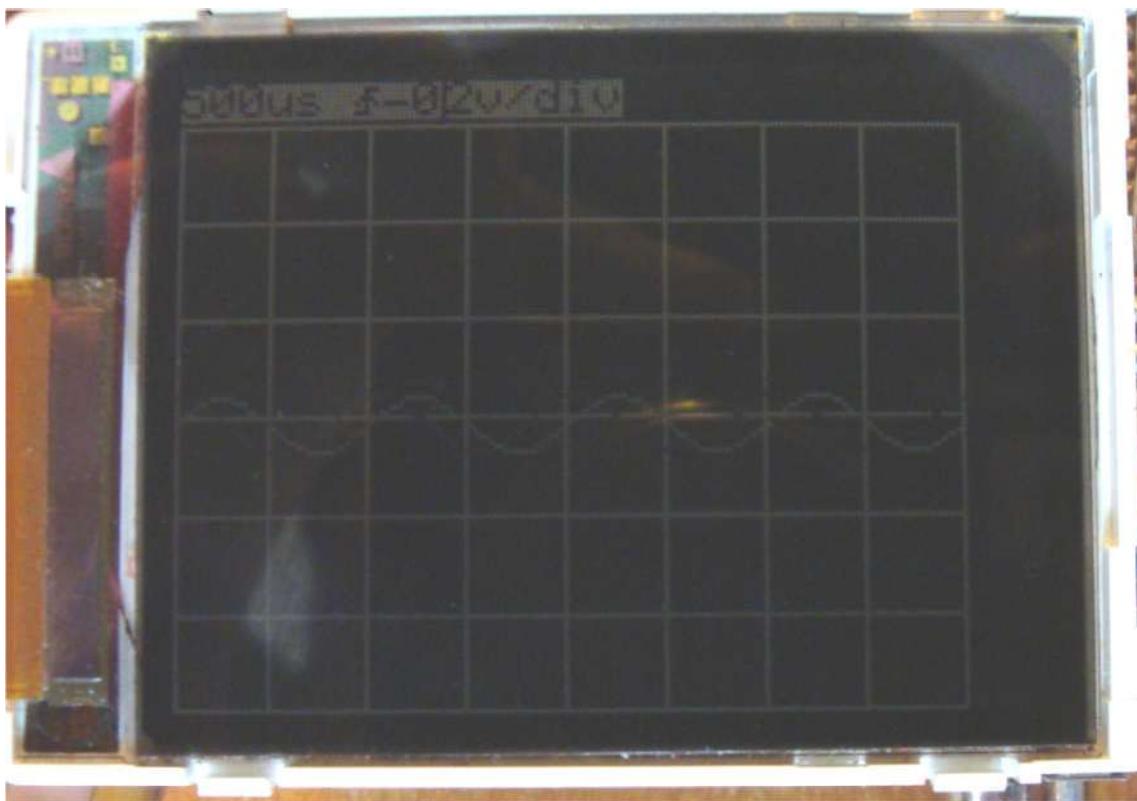
Rys. 6.5. Przebieg prostokątny: amplituda 5 V, częstotliwość 2 kHz.



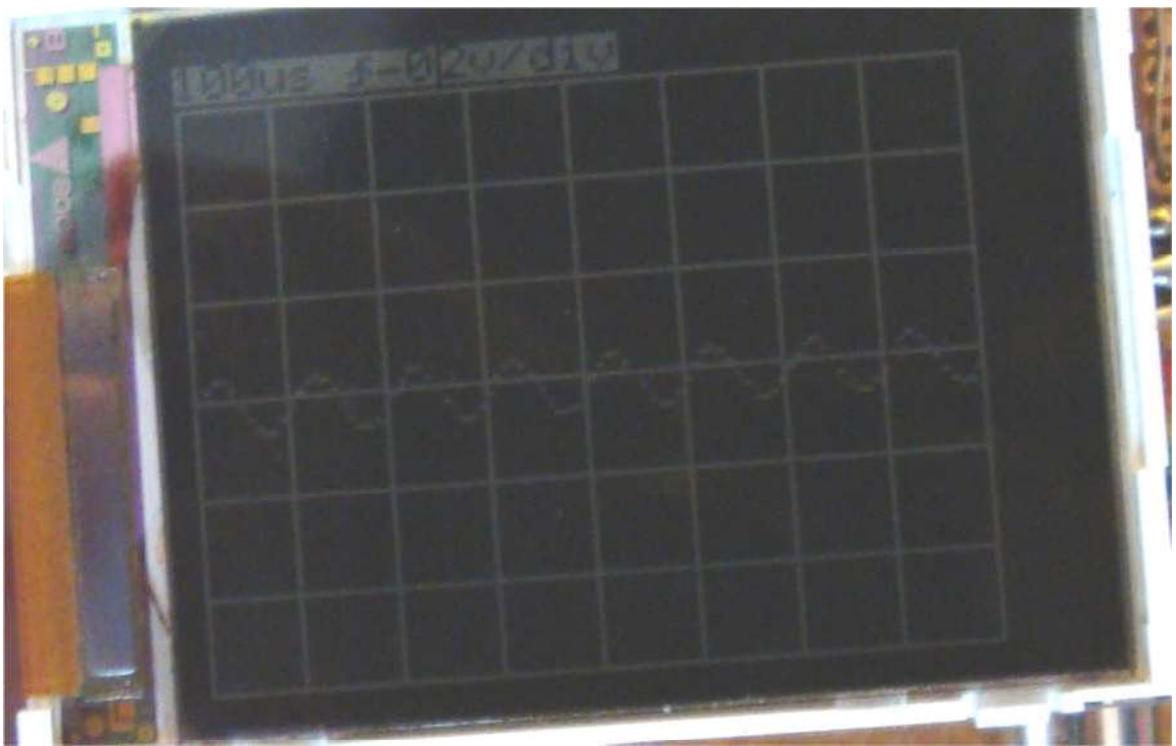
Rys. 6.6. Przebieg prostokątny: amplituda 5 V, częstotliwość 100 Hz.



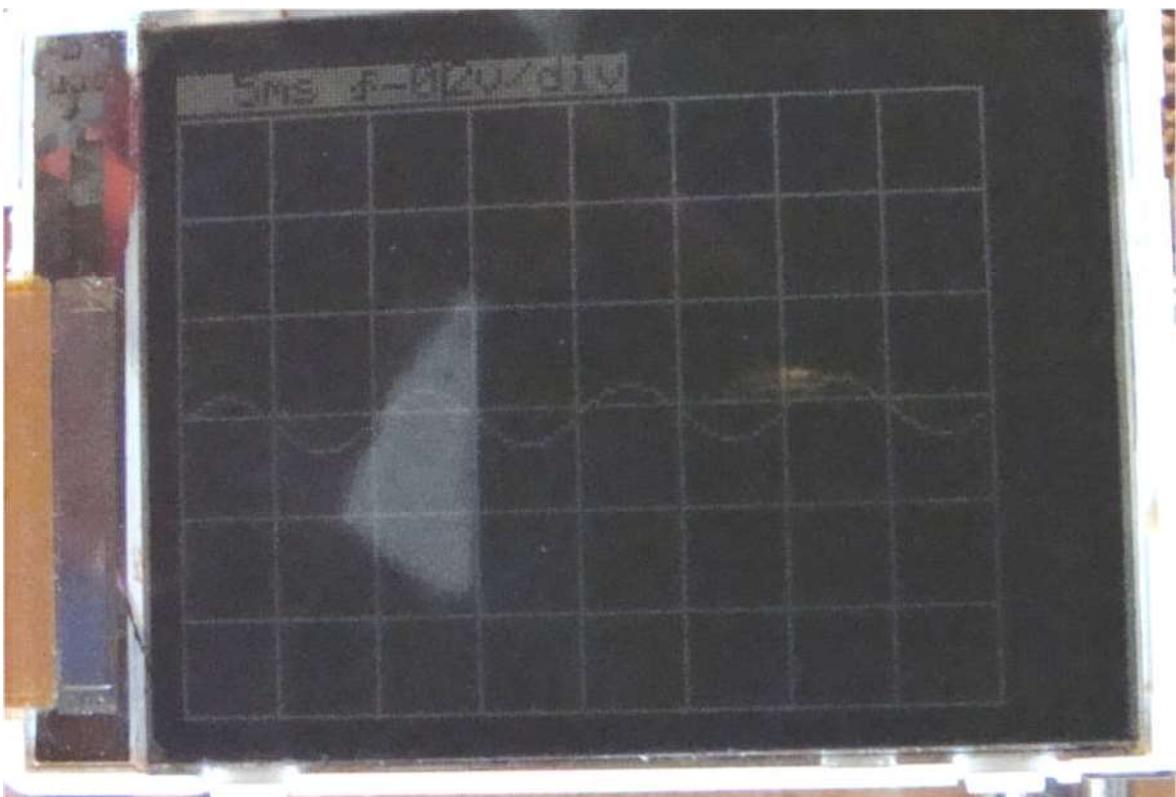
Rys. 6.7. Przebieg prostokątny: amplituda 5 V, częstotliwość 20 Hz.



Rys. 6.8. Przebieg sinusoidalny: amplituda 2 V, częstotliwość 1 kHz.



Rys. 6.9. Przebieg sinusoidalny: amplituda 2 V, częstotliwość 10 kHz.



Rys. 6.10. Przebieg sinusoidalny: amplituda 2 V, częstotliwość 100 Hz.