Temat: Obsługa płytki STRC51 i kompilatora SDCC. **Ćwiczenie 1. (sd)**

1. Konfiguracja systemu operacyjnego.

Przed przystąpieniem do wszelkich prac podczas ćwiczenia należy przygotować stanowisko komputerowe. Głównym elementem jest płytka STRC51 dla której przygotowywane będą programy (w C). Narzędziem niezbędnym do tworzenia kodu wynikowego jest pakiet SDCC (oficjalna strona z której można pobrać pakiet to: http://sdcc.sourceforge.net, **Uwaga!** Pakiet ten jest zainstalowany w lokacji opisanej poniżej). Przed przystąpieniem do edycji i kompilacji programów należy odpowiednio skonfigurować swoje konta komputerowe.
Do poprawnej pracy kompilatora potrzebne są ustawienia zmiennych środowiskowych (ścieżka dostępu). Należy sprawdzić czy w ścieżce dostępu jest program SDCC. Najprostszym sposobem jest uruchomienie programu CMD (linia poleceń) i wydanie polecenia PATH. W otrzymanym raporcie należy odszukać wpis: SDCC/BIN lub podobny. W przypadku braku takiego wpisu należy dla komputerów laboratoryjnych dodać wpisy:

```
PATH=n:\sdcc\bin
```

W przypadku komputerów działających pod kontrolą Windows XP, należy ewentualne zmiany wykonywać tylko dla profilu użytkownika lub z uprawnieniami administratora dla profili wszystkich użytkowników.
Następną czynnością jest ustalenie ścieżki (linku) do w właściwego miejsca składowania plików pakietu SDCC. Czynności te wykonujemy logując się do komputera udostępniającego pliki, z poziomu katalogu domowego.

```
ln -s /opt/windows/staff/sdcc  sdcc
```

Następnie należy sprawdzić w jakiej wersji jest program wydając polecenia:

```
n:
sdcc –v
```

w rezultacie czego pojawią się informacje o wersji oprogramowania (i potwierdzą że ścieżki są poprawnie ustawione). Jednym z możliwych komunikatów może być:

```
SDCC : mcs51/gbz80/z80/avr/ds390/pic14/TININative/xa51 2.3.3
```

Na końcu wpisu ujawniony jest numer wersji pakietu (tu: 2.3.3).

2. Konfiguracja komunikacji z płytką STRC51.

2.1. Konfiguracja programu komunikacyjnego.

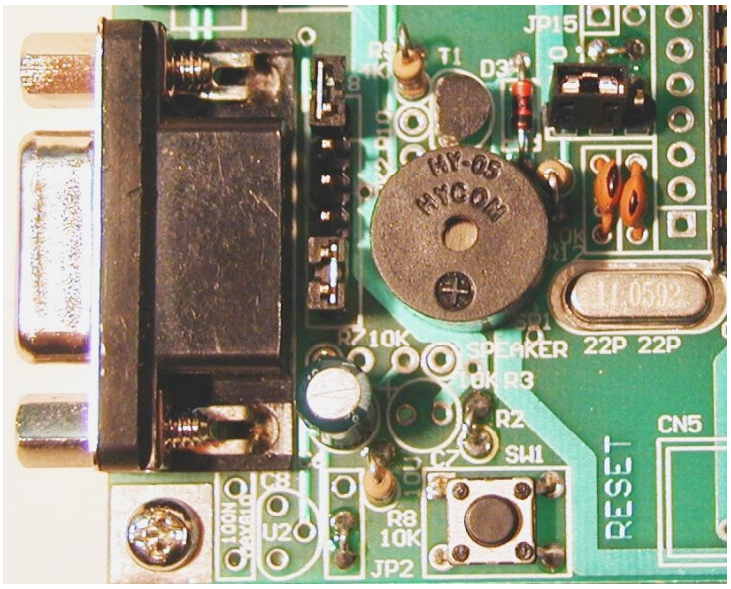
Używając pakietu TERA TERM PRO (dostępny na komputerach w laboratorium) należy skonfigurować następujące parametry:
-Rodzaj połączenia: COM1 (zestawiając nowe połączenie: FILE->NEW CONNECTION...),
-Parametry transmisji:

-prędkość transmisji 19200,
-bez badania parzystości,
-8 bitów danych,
-1 bit sotpu,
-bez kontroli przepływu informacji,
-opóźnienie wysyłania znaków 0ms, opóźnienie wysyłania linii 20ms.

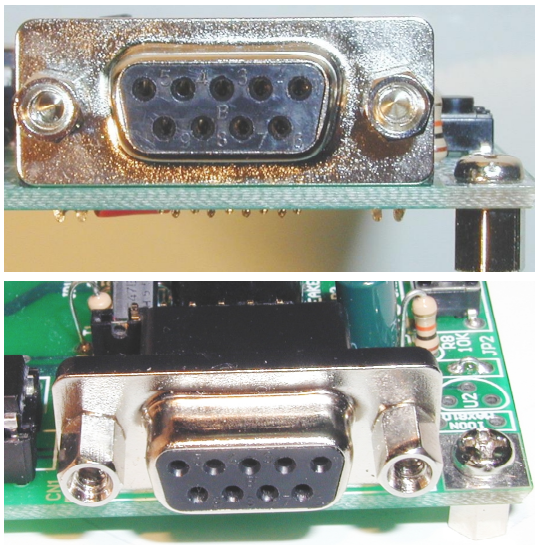
Dla zmian tych opcji należy wejść w: SETUP -> SERIAL PORT.
Po dokonaniu wszelkich prac konfiguracyjnych tego pakietu można zapisać konfigurację:
SETUP -> SAVE SETUP z własną nazwą pliku konfiguracyjnego (tak aby nie zniszczyć utworzonego przez administratora pliku konfiguracyjnego). Można eksperymentalnie dobrać inne parametry opóźnień.

2.2. Konfiguracja połączenia z płytką STRC51.

Aby możliwa stała się komunikacja z płytka należy ją połączyć z komputerem kablem (**przedłużaczem** – stosowanie innego kabla może zniszczyć komputery lub płytkę STRC51) RS232. Łącząc komputer przez złącze COM1 (zgodnie z ustawieniami w punkcie 2.1) z płytką STRC51 – gniazdo CN1 (pokazane na rysunkach 1. i 2.).

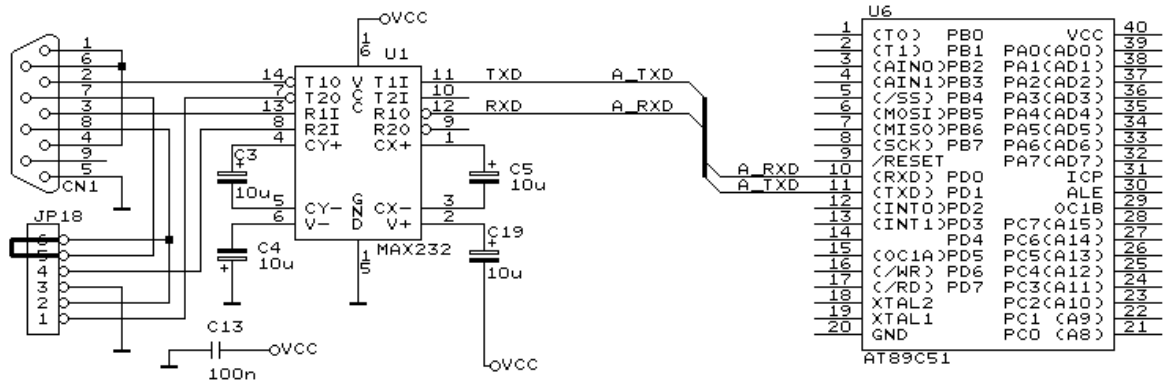


Rys.1. Widok z góry.



Rys. 2. Widok z boku.

Schemat na rysunku 3. pokazuje elektryczne połączenie gniazda CN1 w STRC51, dla głównej ścieżki komunikacyjnej.



Rys.3. Schemat elektryczny otoczenia CN1

3. Konfiguracja – wywoływanie kompilatora.

Używanym w ramach ćwiczenia, kompilatorem jest program SDCC (wchodzącym w skład pakietu o takiej samej nazwie). Aby skompilować plik zapisany pod nazwą main.c (znajdujący się w katalogu sdcc/source jako przykład do łagodnego wejścia z zagadnienie), należy wywołać polecenie (plik main.c należy skopiować do swojego katalogu na dysku n:, i z tego samego miejsca wywołać polecenie):

```
sdcc -c main.c
```

Wynikiem działania kompilatora są między innymi pliki:

main.asm	-	plik będący skompilowaną do asemblera wersją pliku źródłowego,
main.lst	-	plik będący skompilowaną do asemblera wersją pliku źródłowego z dodanymi kodami binarnymi poleceń asemblerowych,
main.sym	-	plik z symbolami użytymi w pliku: main.asm,

`main.rel` - plik z wynikiem kompilacji (znany z platformy `X86`, jako plik `OBJ` – o innym formacie).

Tak skompilowany program należy poddać linkowaniu z bibliotekami:

```
sdcc --model-small --code-loc 0x4000 --xram-loc 0 main.rel
```

Znak '\' na końcu pierwszej linii wskazuje na dalszą część polecenia w nastepnej linii.
Po poprawnie wykonanym polecenia, powinny pojawić się między innymi pliki:

`main.rst` - plik asemblerowego listingu z danymi dodanymi przy linkowaniu,
`main.map` - plik utworzony prze linker, z informacjami o użyciu pamięci,
`main.lnk` - plik pomocniczy dla linkera,
`main.ihx` - wynik pracy linkera – w formacie `HEX`.

Warto zwrócić uwagę na parametry wywołania:

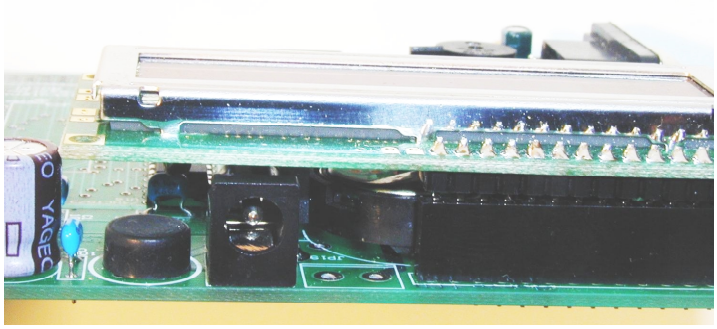
`--model-small` - tryb pracy procesora `80C51`, możliwy jest też tryb `LARGE`,
`--code-loc 0x4000` - lokalizacja pamięci `CODE` (zawierającej skompilowany kod programu do wykonania), **UWAGA!** W płytkach używanych w ramach ćwiczeń ten parametr nie może być zmieniany – tam procesor skacze po wykonaniu ładowania programu z PC,
`--xram-loc 0` - umiejscowienie danych w zewnętrznej pamięci danych `XDATA`.

Celem dokładniejszej analizy parametrów wywołania należy udać się do podkatalogu `DOC` w katalogu `SDCC`, tam przejrzeć dokumenty `HTML` przeglądarką - będące anglojęzyczną dokumentacją kompilatora.
Końcowym etapem jest uporządkowanie wynikowego pliku `main.ihx`. Czynność ta nie jest obowiązkowa ale przyspiesza w przypadku niektórych programów, ładowanie do `STRC51`.

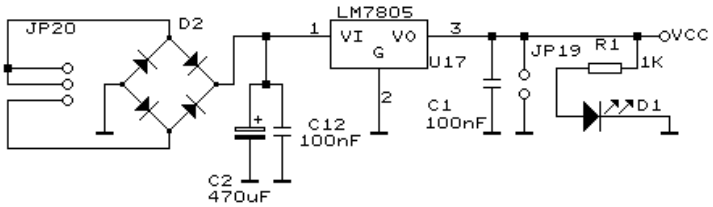
```
packihx main.ihx > main.hex
```

4. Ładowanie programu do płytki z procesorem 80C51.

Po podłączeniu zasilania i połączeniu z komputerem, płytka testowa gotowa jest do pracy. Do zasilania urządzenia należy używać zasilaczy o napięciu wyjściowym między: 8..12V przy wydolności prądowej minimum 300mA. Rysunek 4. przedstawia złącze zasilania JP20 do którego podłączamy zasilacz (UWAGA! Przed włączeniem zasilania podłączyć płytkę do PC przez złącze RS232). Schemat z rysunku 5. przedstawia blok zasilania.

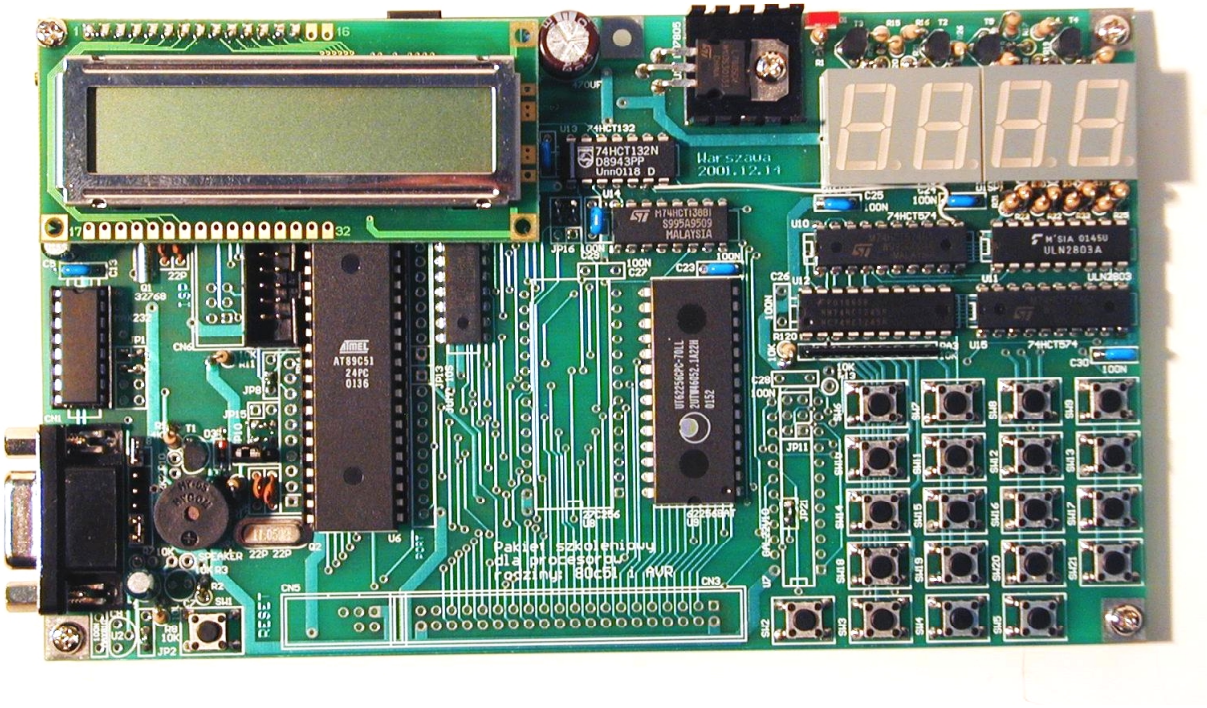


Rys. 4. Widok gniazda zasilania JP20



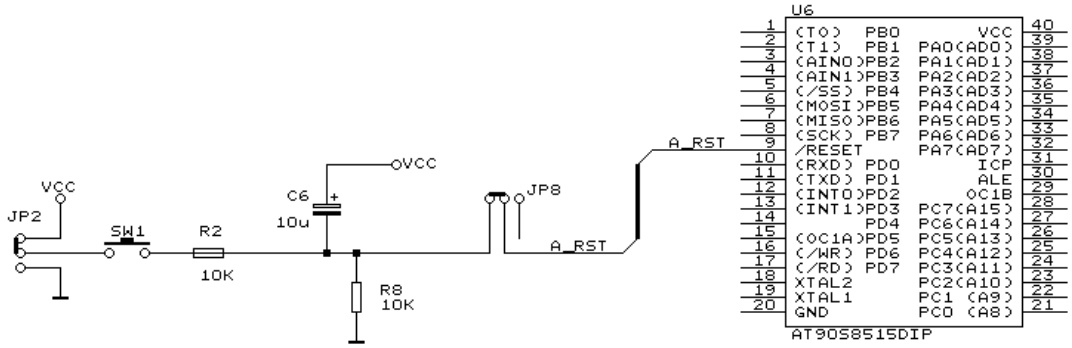
Rys. 5. Schemat bloku zasilania

O poprawnej pracy (w tym poprawnym zasilaniu) układu świadczy świecenie diody D1 jej umiejscowienie pokazuje rysunek 6. (czerwony obiekt w okolicach górnego prawego rogu).



Rys. 6. Widok płytki STRC51

Aby inicjować działanie STRC51 (między innymi przed ładowaniem programów) używamy klawisza RESET – SW1. Jego lokalizacja jest widoczna na płytce – opis na warstwie miedzi. Schemat na rysunku 7. pokazuje obwód związany z generowaniem sygnału RESET.



Rys. 7. Schemat obwodu generowania sygnału RESET

Po naciśnięciu SW1 na ekranie terminala pojawią się komunikaty generowany przez STRC51. Od tego momentu możliwe jest:
-odczyt pamięci XRAM (D),
-odczyt pamięci IRAM (R),
-skok pod określony adres w pamięci kodu (J),
-załadowanie pamięci RAM zbiorem w formacie HEX-INTEL z różnymi opcjami (P, Q, N),
-odczyt i zapis z pamięci „bezpośredniej" (I, O),
-raport zawartości banków rejestrów i pamięci wewnętrznej (X),
-zapełnienie pamięci zewnętrznej XRAM (F).

Aby poprawnie załadować program (w formacie HEX) należy kolejno:
-nacisnąć klawisz RESET – SW1,
-wybrać Q – celem załadowania pamięci CODE programem do uruchomienia,
-system po komendzie Q wypisze komunikat:

```
Poczatek ladowania programu
```

-skopiować do schowka zawartość pliku, np.:
-wydając polecenie (kopiujące automatycznie plik do schowka):

```
copyclip main.hex,
```

-wkleić zawartość schowka do "połączenia" w programie TERA TERM PRO (EDIT->PASTE),
-poczekać na pojawienie się komunikatu (podczas kopiowania widać – mało czytelnie zawartość pliku: main.hex kopiowanego do STRC51):

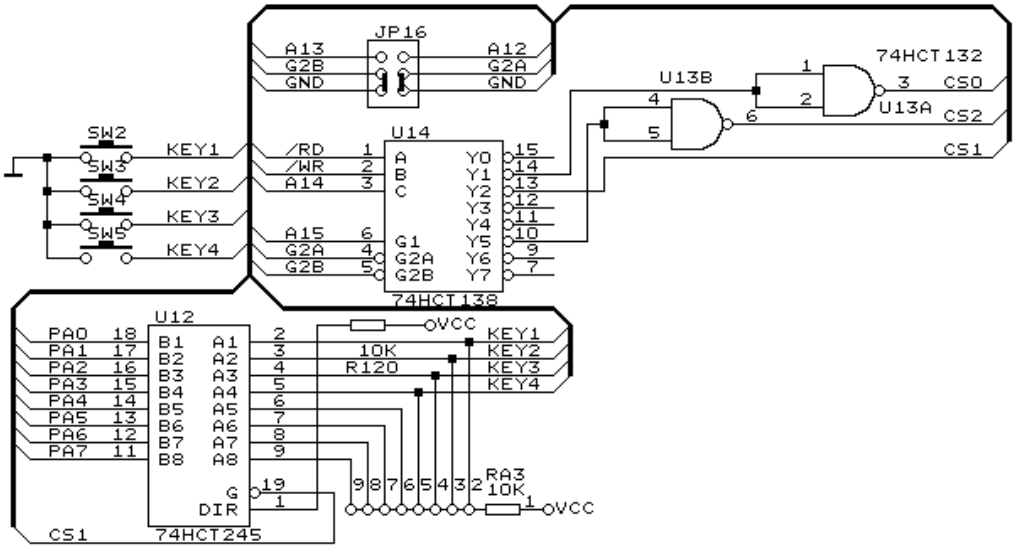
```
Koniec ladowania programu
```

Ponowne załadowanie – jeżeli potrzebne – może nastąpić na dwa sposoby: używając przycisku RESET, bądź umieszczenie w programie wynikowym skoku pod adres 0x0000.
Zamiast polecenia P, można używać poleceń Q lub N – które automatyzują powyższe polecenia – łącząc ładowanie z automatycznym uruchomieniem programu.

5.Uwagi dotyczące programowania w języku C.

Używaną odmianą języka jest ANSI-C z rozszerzeniami i pewnymi ograniczeniami (patrz: SDCC\DOC\index.html). Z ważniejszych różnic jest możliwość alokacji zmiennych (nie mówimy tu o alokacjach dynamicznych: malloc-free) w określonych obszarach pamięci. Dla przykładu klawiatura przez programistę jest widziana jako zmienna (co wynika z konstrukcji systemu – patrz rysunek 8. przedstawiający schemat obwodu związanego z U12):

```
xdata at 0x8000 unsigned char U12;
```



Rys. 8. Schemat obwodów związanych z `U12`

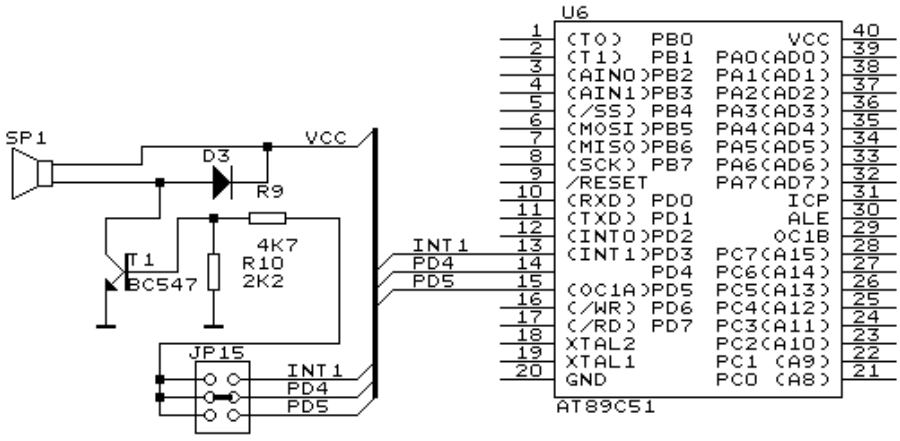
Dla kompilatora oznacza, że jest umiejscowiona ta zmienna będzie w przestrzeni `XDATA` procesora (są jeszcze przestrzenie: `CODE`, `DATA` i inne). Dyrektywa `at` wskazuje, że zmienna ta znajdzie się pod pewnym wyspecyfikowanym adresem (tutaj: `0x8000`).

Po dokładnej analizie schematu (rys. 8.) dostrzeżemy, że układ scalony `U12` (`74HCT245`) jest podłączony do klawiszy: `SW2, SW3, SW4, SW5`. Czytając ze zmiennej `U12` wartość można sprawdzić stan tych klawiszy (naciśnięcie klawisza powoduje wyzerowanie odpowiedniego mu bitu w `U12`).

Innym rozszerzeniem jest możliwość (wspomagana sprzętowego) używania zmiennych bitowych. Dla przykładu:

```
sbit at 0xB5 T1;
```

definiuje jednobitowa zmienną `T1` pod adresem `0xB5` w przestrzeni pamięci wewnętrznej. Ten obszar jest podłączony do pinu nr. 14 (`PD4`).



Rys. 9. Obwody generowania dźwięków (buzzer)

Na schemacie rys. 9. można dostrzec iż ten pin steruje głośnikiem `SP1`. Zmieniając stan tego pinu można generować efekty dźwiękowe. Należy zwrócić uwagę, że sterowanie głośnikiem odbywa się binarnie – wiec program sterujący musi sam zadbać o zmianę położenia membrany głośnika. Drgania membrany słyszalne dla ucha ludzkiego są w zakresie od 300Hz do 14000Hz. Ze względu na uwarunkowania konstrukcyjne najlepiej dobrać częstotliwość w okolicach 2000..4000Hz.

Poniższy fragment ilustruje sposób generowania drgań:

```
for(;;){
   T1=0;
   Czekaj();
   T1=1;
   Czekaj();
}
```

Gdzie funkcja `Czekaj();` jest "nic nie robiącą" funkcją opuźniającą, której czas wykonania jest w ścisłym związku z częstotliwością generowanego przebiegu akustycznego. Należy też zwrócić uwagę, że rozwiązanie przytoczone tutaj można nieco zmodyfikować od postaci następującej:

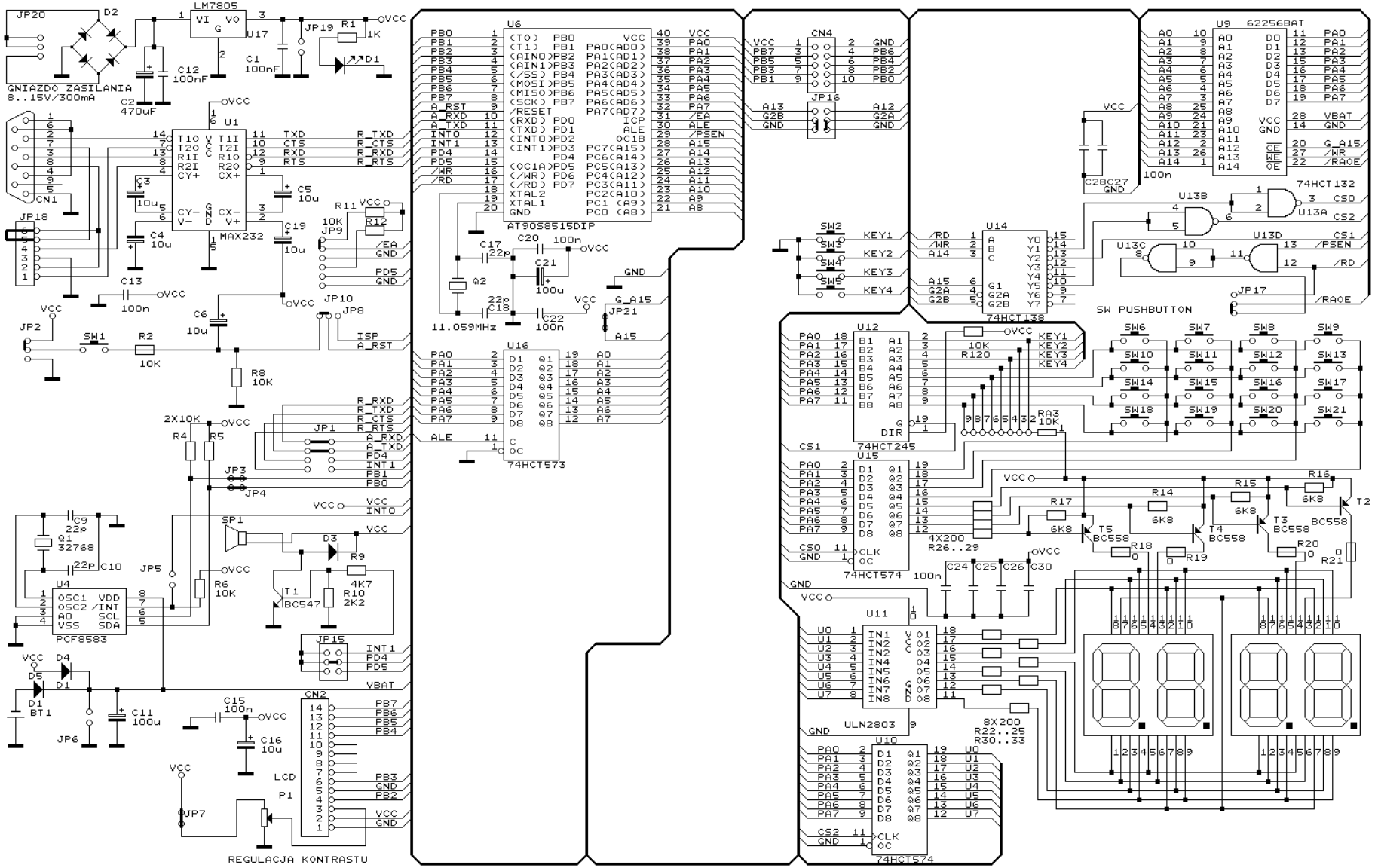
```
char r=0;
...
for(;;){
  if((r & 0x01)==0)
    T1=0;
  else
    T1=1;
  r++;
  Czekaj();
}
```

Rozwiązanie takie wymaga zastosowania dodatkowej zmiennej ale jest łatwiejsze do przekształcenia w rozwiązanie z zastosowaniem przerwań, lub do zastosowania w systemach czasu rzeczywistego.
Warto zwrócić uwagę na dość enigmatyczny zapis:

```
if((r & 0x01)==0)
```

Taka instrukcja warunkowa w ogólności sprawdza najmłodszy bit zmiennej `r`. Wykrywanie czy mamy wartość w zmiennej parzystą czy nie parzystą, warunkuje czy membrana głośnika ma być wciągnięta czy wypchnięta.

6. Zadania do wykonania:
Zapoznać się z działaniem systemu, napisać prosty program (działanie określa prowadzący) na bazie przytoczonych przykładów.


GNIAZDO ZASILANIA
8..15V/300mA
LM7805
AT90S8515DIP
11.059MHz
MAX232
PCF8583
74HCT573
74HCT245
74HCT574
74HCT138
74HCT132
62256BAT
ULN2803
SW PUSHBUTTON
REGULACJA KONTRASTU
LCD