KOLEGIUM KARKONOSKIE
w Jeleniej Górze

Państwowa Wyższa Szkoła Zawodowa

LABORATORIUM SYSTEMÓW MIKROPROCESOROWYCH

Aleksander DZIUDA

Waldemar KRUPA

JELENIA GÓRA 2007

RECENZENT

Jan Jagielski

Niniejsze wydawnictwo można nabyć w Bibliotece Uczelnianej
Kolegium Karkonoskiego PWSZ
w Jeleniej Górze
ul. Lwówecka 18 tel. 0 75 645 33 52

ISBN 978-83-924736-3-3

SPIS TREŚCI

OD AUTORÓW

Niniejsze opracowanie jest przeznaczone dla studentów przygotowujących się i wykonujących ćwiczenia laboratoryjne w ramach kursów *"Systemy mikroprocesorowe II"* na kierunku ELEKTRONIKA I TELEKOMUNIKACJA oraz *"Technika cyfrowa"* z zakresu systemów mikroprocesorowych na kierunku EDUKACJA TECHNICZNO-INFORMATYCZNA, prowadzonych w Instytucie Techniki Kolegium Karkonoskiego.

Przyjmuje się, że studenci znają architekturę mikrokontrolera 80C51 oraz podstawowe zasady programowania mikrokontrolerów w języku asemblera dla mikrokontrolera 80C51 z informacji przekazywanych na wykładach i zajęciach projektowych poprzedzających zajęcia laboratoryjne.

Pierwszy rozdział opracowania zapoznaje z Regulaminem Laboratorium Systemów Mikroprocesorowych oraz przepisami BHP.

Rozdział drugi przedstawia opis modułowego systemu µM-DYD wraz z opisem modułu sterownika mikrokontrolera rodziny ’51.

Rozdział trzeci omawia zintegrowane środowisko programowe IDE-51 łączące w jedną całość edytor tekstowy, asembler, linker oraz debugger płytkowy.

Rozdziały 2 i 3 opracowany został na podstawie dokumentacji technicznej firmy MicroMax *Moduł dydaktyczny µM-DYD. Wersja 4.0*. MicroMax 1995.

Rozdział czwarty zapoznaje studentów z tematyką poszczególnych ćwiczeń laboratoryjnych.

Każde z praktycznie realizowanych ćwiczeń zawiera podany cel dydaktyczny realizowanego ćwiczenia, przedmiot ćwiczenia, program ćwiczenia oraz zakres materiału obowiązujący studenta.

Dodatkowo opracowanie zawiera następujące dodatki:

Dodatek A - predefiniowane rejestry specjalne (SFR) mikrokontrolerów rodziny ’51.

Dodatek B - listę rozkazów mikrokontrolera 80C51.

Dodatek C - komunikaty o błędach kompilatora oraz programu MONITOR - 51.

Informacje zawarte w pracy ułatwiają wykonanie ćwiczeń przewidzianych programem kształcenia. Załączony spis literatury powinien zachęcić studentów do samodzielnego pogłębiania i rozszerzania swojej wiedzy z zakresu Systemów Mikroprocesorowych.

1. REGULAMIN LABORATORIUM SYSTEMÓW MIKROPROCESOROWYCH

Niniejszy regulamin obowiązuje studentów odbywających zajęcia laboratoryjne w sali 31 z przedmiotu *"Systemy mikroprocesorowe"* na kierunku ELEKTRONIKA I TELEKOMUNIKACJA oraz *"Technika cyfrowa"* z zakresu systemów mikroprocesorowych na kierunku EDUKACJA TECHNICZNO-INFORMATYCZNA Instytutu Techniki Kolegium Karkonoskiego.

Laboratorium z Systemów Mikroprocesorowych jest ilustracją wykładów i zajęć projektowych, prowadzonych w Instytucie Techniki Kolegium Karkonoskiego.

W laboratorium obowiązuje następujący regulamin:

1. Obecność na zajęciach jest obowiązkowa.
2. Zajęcia laboratoryjne odbywają się zgodnie z harmonogramem wywieszonym na tablicy ogłoszeń.
3. Ćwiczenia laboratoryjne odbywają się w podgrupach laboratoryjnych, których skład ustalony jest na pierwszych zajęciach.
4. Płaszcze i inne okrycia wierzchnie należy pozostawić w szatni.
5. W laboratorium nie wolno spożywać żadnych posiłków (dotyczy to również napojów).
6. W laboratorium należy zachować ciszę. Konwersację z innymi studentami należy ograniczyć do niezbędnego minimum i prowadzić

głosem ściszonym, tak, aby nie zakłócić ciszy i nie rozpraszać pozostałych ćwiczących.

7. Nie wolno samowolnie zamieniać ani przenosić przyrządów oraz korzystać ze sprzętu laboratoryjnego nie wchodzącego w zestaw danego ćwiczenia.
8. Łączenie badanych układów studenci dokonują używając wyłącznie elementów znajdujących się na danym stanowisku pomiarowym.
9. Studenci mogą włączać napięcia zasilające oraz włączać i wyłączać komputery wyłącznie za zgodą prowadzącego.
10. Studenci są zobowiązani do zachowania szczególnej ostrożności w użytkowaniu przyrządów i komputerów. O wszelkich uszkodzeniach należy natychmiast powiadomić prowadzącego zajęcia. W przypadku uszkodzenia przyrządu lub komputera z winy studenta, ponosi on odpowiedzialność materialną zgodnie z obowiązującymi przepisami.
11. Winni zniszczenia lub uszkodzenia elementów układu wskutek samowolnego włączenia napięcia lub niestosowania się do wskazówek prowadzącego ćwiczenia - będą odpowiadali materialnie.
12. Zabrania się dokonywania jakichkolwiek zmian w konfiguracji systemu Windows 98SE (dotyczy to również pulpitu!) i w zainstalowanych programach.
13. Rozłączenie badanego układu jest dozwolone po zatwierdzeniu wyników przez prowadzącego ćwiczenie.
14. Zakazuje się wgrywania, pobierania z Internetu oraz instalowania oprogramowania na dyskach komputerów znajdujących się w laboratorium.
15. Zabrania się tworzenia nowych katalogów oraz wgrywania plików do komputerów laboratorium.

16. W trakcie zajęć zabrania się z korzystania z Internetu (przeglądarek WWW, poczty elektronicznej, a szczególnie z usług typu: Gadu-Gadu, IRC, Chat, itp.). Można korzystać z przeglądarek WWW w przypadku, gdy jest to wymagane przez ćwiczenie.
17. Wykonując dane ćwiczenie należy pracować wyłącznie w katalogu roboczym wskazanym w instrukcji tego ćwiczenia i zgodnie z tą instrukcją.

Celem uniknięcia nieszczęśliwym wypadkom należy przestrzegać następujących zasad (wyciąg z instrukcji BHP):

1. Pracę na stanowiskach laboratoryjnych powinna cechować szczególna ostrożność i rozwaga.
2. Przed przystąpieniem do modelowania badanego układu, należy sprawdzić czy źródła zasilania są odłączone.
3. Zabrania się używania przewodów łączących z uszkodzoną izolacją lub oberwanymi końcówkami.
4. Zabrania się włączenia napięcia bez sprawdzenia układu przez prowadzącego ćwiczenia i uzyskania jego zgody.
5. Wszystkie zmiany w układzie, jak również jego rozłączenie jest dozwolone jedynie przy odłączeniu napięcia.
6. Włączenie napięcia po dokonanej zmianie w układzie może nastąpić dopiero po sprawdzeniu układu przez prowadzącego ćwiczenia.
7. Zabrania się dotykać części elementów układu znajdującego się pod napięciem, a w szczególności zacisków przyrządów i końcówek przewodów łączących.
8. W razie odłączenia źródeł zasilania przez bezpiecznik automatyczny, znajdujący się na listwie zasilającej należy niezwłocznie powiadomić o tym fakcie prowadzącego ćwiczenia.
9. W przypadku porażenia prądem elektrycznym należy:

- natychmiast wyłączyć napięcie zasilające za pomocą wyłącznika głównego lub wyłączników znajdujących się na listwach zasilających;
- uwolnić porażonego spod działania prądu elektrycznego;
- powiadomić prowadzącego ćwiczenia oraz lekarza telefon 999;
- przystąpić do udzielania pierwszej pomocy zgodnie z zasadami udzielania pomocy w przypadku porażenia prądem;
- poddać poszkodowanego opiece lekarskiej, niezależnie od tego czy stracił przytomność czy nie.

10. W razie pożaru należy:
    - wyłączyć napięcie zasilające;
    - nieść pomoc osobom zagrożonym, zachować spokój, nie wywoływać paniki;
    - przystąpić do gaszenia pożaru przy pomocy dostępnych środków gaśniczych, np.: kocy gaśniczych, gaśnic.

      Uwaga! do gaszenia urządzeń elektrycznych nie wolno używać gaśnic pianowych.
    - powiadomić straż pożarną – telefon 998;
    - studenci, którzy nie biorą udziału w akcji ratowniczej muszą bezwzględnie opuścić laboratorium.
11. W razie powstania innego wypadku (urazy mechaniczne, chemiczne, itp.) należy:
    - usunąć czynniki szkodliwe;
    - zawiadomić o wypadku pracownika prowadzącego ćwiczenia;
    - udzielić pierwszej pomocy poszkodowanym;
    - w razie potrzeby wezwać lekarza, ewentualnie przewieźć poszkodowanego do szpitala.

Organizacja zajęć

W ciągu semestru odbywa się 15 ćwiczeń (15 spotkań). Każde ćwiczenie trwa 2 godziny lekcyjne. Pierwsze spotkanie organizacyjne stanowi wprowadzenie do ćwiczeń, podczas którego następuje omówienie organizacji i przebiegu ćwiczeń, sposobu oceniania oraz dokonywany jest podział na podgrupy laboratoryjne. Dwa ostatnie spotkania przeznaczone są na ewentualne odrobienie zaległych ćwiczeń.

Studenci wykonują ćwiczenia w Laboratorium Systemów Mikroprocesorowych w podgrupach składających się z zespołów dwuosobowych lub w szczególnych przypadkach trzyosobowych pod kierunkiem pracownika Instytutu Techniki.

Podgrupy laboratoryjne wykonują ćwiczenie zgodnie z harmonogramem zajęć określającym termin, miejsce, temat ćwiczenia i osobę prowadzącą zajęcia. Harmonogram wywieszony jest w gablocie przedmiotu.

Ćwiczenia wykonywane są w systemie całościowym, tzn. ustawiony jest w laboratorium jeden temat ćwiczeń, realizowany przez wszystkie podgrupy.

Każde ćwiczenie polega na napisaniu i uruchamianiu programów w języku asemblera realizujących otrzymane zadania. Każda grupa otrzymuje indywidualne zadania projektowe. Treść i formę prezentacji zadań określa prowadzący zajęcia.

W przypadkach usprawiedliwionej nieobecności na ćwiczeniu istnieje możliwość odrobienia ćwiczenia z inną podgrupą laboratoryjną, jeżeli będą wolne miejsca. Przypadek taki należy omówić z prowadzącym ćwiczenia w danej grupie.

Odrobienie dwóch nieobecności nieusprawiedliwionych następuje podczas ostatnich zajęć w tzw. terminach obróbczych.

Każdy student przed rozpoczęciem ćwiczeń laboratoryjnych obowiązany jest zapoznać się z *Regulaminem Laboratorium Systemów Mikroprocesoro-*

wych oraz *Przepisami BHP obowiązującymi w laboratorium*, co winno być udokumentowane podpisem.

Osoby, które nie przestrzegają regulaminu mogą być pozbawione przywileju korzystania z laboratorium (konsekwencją tego faktu może być brak zaliczenia ćwiczenia i całego laboratorium).

Przebieg ćwiczenia

Realizacja ćwiczeń laboratoryjnych, to samodzielna działalność i podejmowanie decyzji przez studentów w trakcie wykonywania ćwiczeń.

Ćwiczenie rozpoczyna się od sprawdzenia obecności i przygotowania do zajęć (obowiązuje znajomość teorii zawartej w niniejszej instrukcji i materiału wykładu związanego z danym ćwiczeniem).

Przygotowanie się studentów do ćwiczeń laboratoryjnych polega na:

- ✓ zapoznaniu się z tematem i celem ćwiczenia;
- ✓ przyswojeniu teoretycznych podstaw budowy, sterowania i działania badanych elementów składowych systemów mikroprocesorowych;
- ✓ przygotowanie protokołu ćwiczenia;
- ✓ określeniu przewidywanych efektów działania systemu mikroprocesorowego;
- ✓ przyswojeniu sobie wiadomości teoretycznych dotyczących wykorzystania środowiska programowego i sprzętowego wykorzystywanego podczas zajęć laboratoryjnych;
- ✓ przygotowaniu protokołu pomiarów z opracowanymi algorytmami oraz przygotowanymi programami w języku asemblera.

W przypadku niedostatecznego przygotowania teoretycznego prowadzący ćwiczenia może nie dopuścić studenta do wykonywania ćwiczenia. Przypadek taki jest równoznaczny z nieobecnością nieusprawiedliwioną. Student, który nie został dopuszczony do wykonywania ćwiczenia ma prawo uczestniczyć w wykonywaniu ćwiczenia, jednak bez możliwości oddania protokołu i spra-

wozdania. Takie ćwiczenie musi być powtórzone w czasie wyznaczonym przez prowadzącego w ramach limitu nieobecności nieusprawiedliwionych.

Podstawowym dokumentem wykonania ćwiczenia jest protokół ćwiczenia. Każda podgrupa prowadzi jeden protokół ćwiczenia, w którym posiada opracowane algorytmy i napisane programy w języku asemblera zgodnie z otrzymanymi zadaniami dotyczącymi danego ćwiczenia, protokół służy dodatkowo w czasie zajęć do notowania wyników działania programów, uwag dotyczących przebiegu ćwiczenia, poleceń prowadzącego zajęcia oraz prowadzenia obliczeń. Protokół podlega zatwierdzeniu przez prowadzącego ćwiczenie. Zatwierdzenie protokołu oznacza zaliczenie części praktycznej ćwiczenia.

Na podstawie protokołu każda podgrupa wykonuje sprawozdanie. Wystarczy jedno sprawozdanie na podgrupę, jakkolwiek każdy student może przygotować własne sprawozdanie. W sprawozdaniu zawarte winny być tylko wyłącznie wyniki z zatwierdzonego protokołu. Sprawozdanie (z dołączonym protokołem) w formie pisemnej należy złożyć prowadzącemu do oceny na następnych zajęciach. W przypadku niespełnienia tego warunku studenci nie są dopuszczeni do wykonywania kolejnego ćwiczenia. Sprawozdania są oceniane i przechowywane przez prowadzącego ćwiczenia do końca semestru, w którym odbywają się zajęcia laboratoryjne.

Szczegółowe zasady wykonywania sprawozdania określa prowadzący ćwiczenia.

Zalecany układ sprawozdania:

- ✓ tabela nagłówkowa sporządzona według wzoru:

KOLEGIUM KARKONOSKIE INSTYTUT TECHNIKI			
LABORATORIUM SYSTEMÓW MIKROPROCESOROWYCH			
Numer grupy:	*Numer ćwiczenia:*	*Prowadzący:*	
Skład podgrupy: *1.* *2.*	*Temat ćwiczenia:*		
	Data wykonania:	*Ocena:*	*Podpis:*

- ✓ cel ćwiczenia;
- ✓ algorytmy otrzymanych do rozwiązania zadań;
- ✓ rozwiązania zadań w języku asemblera otrzymanych w ramach przygotowania się do ćwiczeń laboratoryjnych;
- ✓ schematy połączeń układów faktycznie stosowanych w ćwiczeniu;
- ✓ niezbędne obliczenia;
- ✓ wnioski i uwagi z przebiegu ćwiczenia.

Oceniając sprawozdanie bierze się pod uwagę:

- poprawność i sposób dojścia do wyników;
- stopień wyczerpania tematu we wnioskach;
- staranność wykonania.

W przypadku rażąco nieudolnego wykonywania ćwiczenia, nieumiejętnego posługiwania się sprzętem laboratoryjnym, niszczenia sprzętu pomiarowego, wyposażenia laboratorium lub innego postępowania szkodliwego z punktu widzenia pozostałych osób korzystających z laboratorium studenci mogą zostać usunięci z laboratorium.

Warunki zaliczenia laboratorium

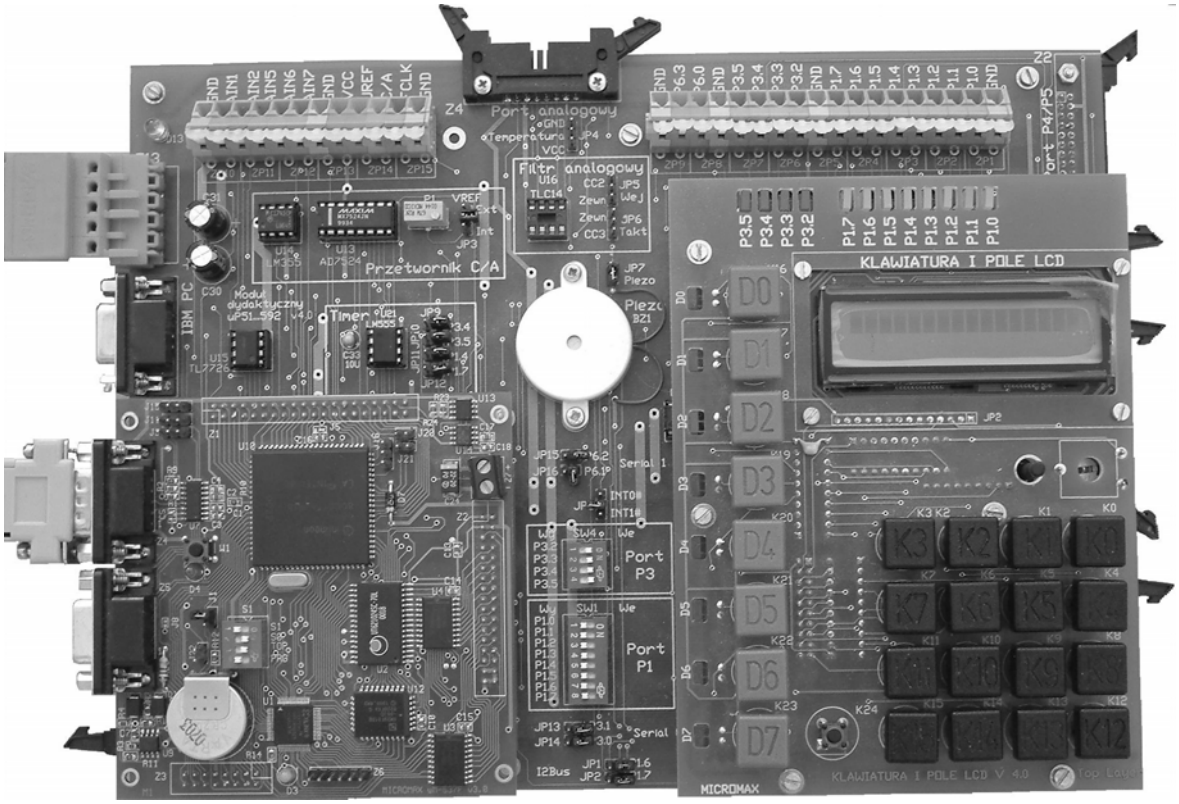
Do zaliczenia laboratorium wymagane jest wykonanie wszystkich ćwiczeń i uzyskanie z nich pozytywnych ocen. Na ocenę ćwiczenia składa się: ocena z przygotowania się do zajęć, ocena ze sprawozdania, jak również ocena aktywności studenta na zajęciach, o ile prowadzący wystawi taką ocenę. Stopień końcowy z laboratorium jest średnią ocen uzyskanych z poszczególnych ćwiczeń.

2. OPIS SPRZĘTU
- MODUŁOWY SYSTEM μM-DYD

Modułowy system μM-DYD składa się z modułu dydaktycznego oraz z modułu sterownika mikrokontrolera rodziny ’51.

Przeznaczony jest do nauki i wstępnego testowania oprogramowania dla systemów mikroprocesorowych z mikrokontrolerami rodziny ‘51.

Wygląd zewnętrzny modułowego systemu μM-DYD przedstawia rysunek 2.1.



Rys. 2.1. Modułowy system μM-DYD.

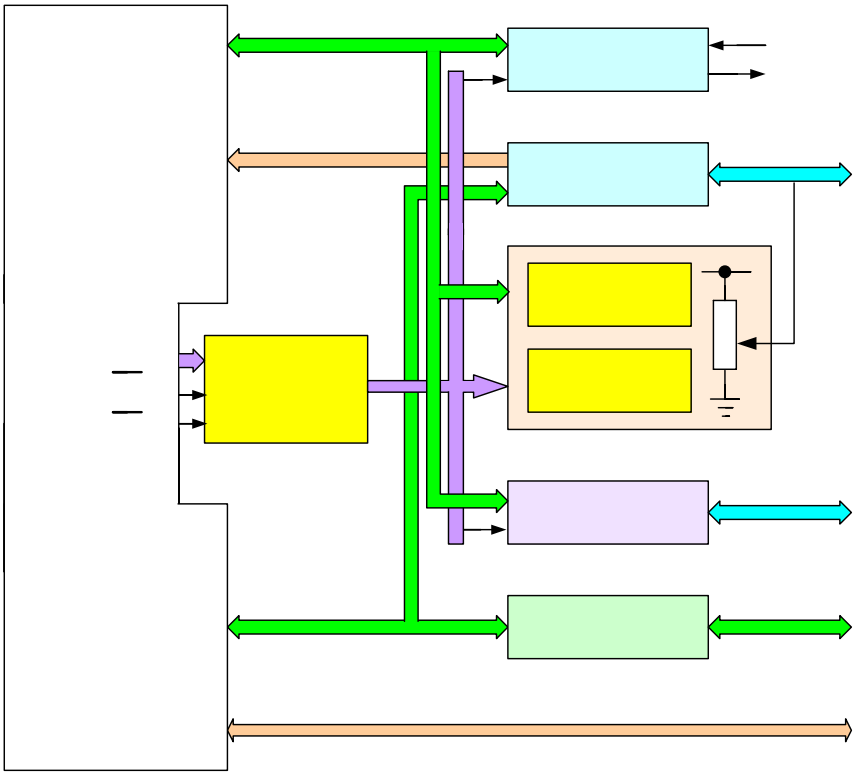
Jako moduł sterownika wykorzystany jest moduł **µM – 537F** z mikrokontrolerem Infineon C537 (Siemens SAB 80C537).

System µM-DYD może dodatkowo współpracować z dołączanymi do niego następującymi makietami dydaktycznymi:

- makietą dydaktyczną „Wyświetlacz LED”;
- makietą dydaktyczną „Skrzyżowanie”;
- makietą dydaktyczną „Tęcza”.

2.1. Struktura wewnętrzna modułu dydaktycznego

Schemat blokowy systemu µM-DYD przedstawia rysunek 2.2.



Rys. 2.2. Schemat blokowy systemu µM-DYD.

Płyta modułu zawiera typowe, najczęściej wykorzystywane układy towarzyszące mikrokontrolerom w systemach sterowania i kontrolno-pomiarowych:

- pole odczytu do wyświetlania komunikatów i informacji o stanie systemu lub nadzorowanych obiektów;

- wieloelementową klawiaturę, która służy użytkownikowi do komunikacji z systemem mikroprocesorowym, wprowadzania lub zmiany parametrów nadzorowanych procesów;
- wyświetlacze dwustanowe informujące o poziomach logicznych wybranych linii;
- przetwornik cyfrowo-analogowy generujący sygnały o zadanych kształtach lub realizujący operacje mnożenia analogowego sygnałów;
- łącze szeregowe w standardzie RS232 będące elementem wewnętrznym mikrokontrolera do indywidualnego wykorzystania przez użytkownika, w wielu systemach uruchomieniowych łącze to wykorzystywane jest do komunikacji mikrokontrolera z programem nadzorującym działanie mikrokontrolera umieszczonym w komputerze PC;
- piezoelement (buzzer) emitujący dźwięki o częstotliwości programowanej przez użytkownika, sygnalizujący wystąpienie stanów alarmowych, np. przekroczenie ustalonych zakresów itp.

Dla systemów pomiarowo-sterujących wykorzystujących mikrokontrolery z przetwornikami analogowo-cyfrowymi moduł dydaktyczny μM-DYD wyposażono w:

- regulowane źródło napięcia stałego;
- układ programowalnego, dolnoprzepustowego filtru analogowego;
- rezystory korekcyjne do połączenia modułu z termoelementami, układami, których rezystancja lub prąd zależne są od temperatury.

Szczegółową charakterystykę modułu dydaktycznego zawiera [1]. Poniżej przedstawiono jedynie podstawowe informacje o tych układach, które są wykorzystywane w ramach ćwiczeń i znajomość, których jest niezbędna do poprawnego wykonania ćwiczenia.

2.1.1. Porty cyfrowych Wejść - Wyjść równoległych

Dostępne są wszystkie linie P1.0 - P1.7 portu P1 oraz linie P3.2 - P3.5 portu P3. Linie te posiadają układy zabezpieczające (bufory) przed przepięciami i przypadkowymi zwarciami. Do każdej z tych linii dołączona jest dioda LED sygnalizująca stan linii. Aktywnym poziomem (powodującym świecenie diody) jest stan niski L (logiczne zero). Sposób buforowania linii i dołączenie diody LED przedstawiono na rysunku 2.1 i rysunku 2.2 w [1].

Konfigurowanie pojedynczej linii jako wejściowej albo wyjściowej dokonuje się alternatywnie:

- dla portu P1 - za pomocą mikroprzełącznika SW1 (znajduje się na płycie) albo za pomocą sygnału P1Sj (na złączu Z1), j = 0,1,...,7.
- dla portu P3 - za pomocą mikroprzełącznika SW4 (na płycie) albo za pomocą sygnału P3Sk (na złączu Z3), k = 2,3,4,5.

Buforowane linie portu P1 i linie P3.2 ÷ P3.5 są wyprowadzone na złącze samozaciskowe, a także - łącznie z sygnałami sterującymi - odpowiednio na złącza Z1 i Z3. Topografię wyprowadzeń na złącza Z1 i Z3 przedstawiono odpowiednio na rysunku 2.3 i rysunku 2.4.

Rys. 2.3.Wyprowadzenie buforowanych linii portu P1 i sygnałów sterujących (widok złącza Z1 od przodu).

Rys. 2.4. Wyprowadzenie buforowanych linii portu P3 i sygnałów sterujących (widok złącza Z3 od przodu).

UWAGA: Linie P3.0 i P3.1 nie są buforowane. Mogą być wykorzystane jedynie w ćwiczeniu „Szeregowa transmisja informacji".

Porty P4, P5 i P6 są również portami buforowanymi. Kierunek przesyłania danych ustalany jest na wielostykowym złączu Z2 i Z3, brak jest natomiast na płycie modułu (dla tych portów) dodatkowych przełączników.

2.1.2. Klawiatura

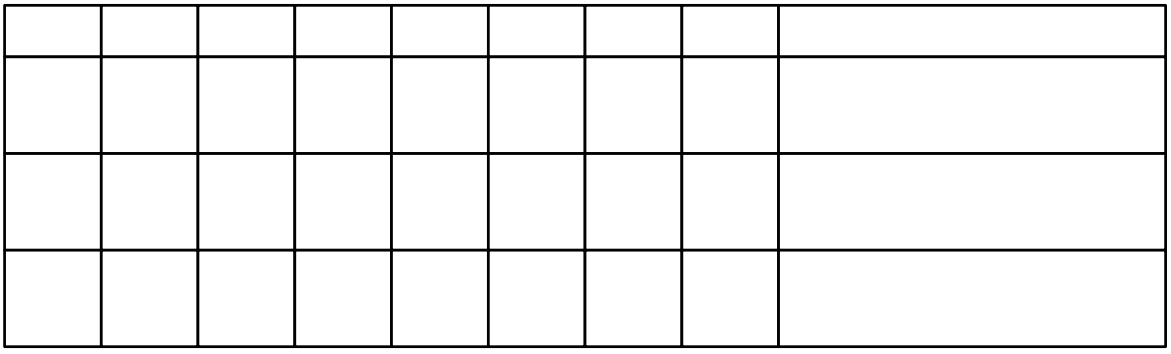
Klawiatura zawiera 25 klawiszy i składa się z:

- klawiatury chwilowej - klawisze K0 ÷ K15;
- klawiatury statycznej - klawisze K16 ÷ K23;
- klawisza zerowania mikrokontrolera - RESET.

Klawisze K0 ÷ K15 zgrupowano w 2 kolumny po 8 klawiszy w każdej:

- w kolumnie 1-szej - klawisze K0 ÷ K7;
- w kolumnie 2-giej - klawisze K8 ÷ K15.

Programowa obsługa klawiatury chwilowej wymaga okresowego - co 8ms albo 10ms - przeglądania stanu klawiszy w obu kolumnach. Zaadresowanie odpowiedniej kolumny i odczyt stanu klawiszy przypisanych do tej kolumny (adresu) umożliwia identyfikację wciśniętego klawisza. Wyróżnikiem wciśniętego klawisza jest poziom L na określonej pozycji odczytanego 8-bitowego słowa. Zasadę tę objaśniono na rysunku 2.5.



Rys. 2.5. Zasada identyfikacji wciśniętego klawisza.
Wciśnięty klawisz: K3 albo K11.

Klawisze K16 ÷ K23 klawiatury statycznej są dołączone do wejść taktujących przerzutników JK. Natomiast do wyjść przerzutników są dołączone diody LED. Każde naciśnięcie klawisza powoduje zmianę stanu przerzutnika i zaświecenie albo wygaszenie dołączonej do jego wyjścia diody LED. Stan klawiszy można odczytać programowo adresując klawiaturę statyczną i dynamiczną zgodnie z przypisanym jej adresem. Adresy przypisane klawiaturze zestawiono w tabeli 2.1.

Tabela 2.1. Adresy przypisane klawiaturze chwilowej i statycznej.

Klawiatura	Klawisze	Adres szesnastkowy
Chwilowa	K0 - K7 K8 – K15	0FFE1H 0FFE2H
Statyczna	K16 - K23	0FFB0H

2.1.3. Wyświetlacz alfanumeryczny LCD

Moduł wyświetlacza alfanumerycznego stanowi ciekłokrystaliczny, 32 znakowy ekran (pole odczytu) LCD o organizacji 2 wiersze po 16 znaków oraz scalony programowalny sterownik standardu HD44780. Układ programowalnego sterownika w wewnętrznej strukturze zawiera między innymi:

- pamięć ekranu DD RAM;
- generator znaków ROM;
- generator programowanych znaków CG RAM.

Pamięć DD RAM ma pojemność 80×8 bitów i służy jako pamięć chwilowa, do której zapisywane są kody - pobierane z generatora znaków ROM - wyprowadzane na ekran znaków. Adresy pamięci DD RAM przypisane poszczególnym znakom ekranu zestawiono w tabeli 2.2.

Generator znaków ROM zawiera 160 wzorców znaków ASCII o organizacji matrycy znakowej 5×7 i 32 wzorce znaków o organizacji matrycy znakowej 5×10.

Tabela 2.2. Adresy pamięci DD RAM przypisane poszczególnym pozycjom znaku na ekranie.

Pozycja znaku w wierszu		1	2	3	4	…	13	14	15	16
Górnym	Adresy	00	01	02	03	…	0C	0D	0E	0F
Dolnym		40	41	42	43	…	4C	4D	4E	4F

Generator znaków CG RAM umożliwia użytkownikowi definiowanie własnych znaków. Możliwe jest zaprogramowanie 8 znaków (np. polskie znaki diakrytyczne: **ą**, **ę**, **ć**, **ń**, itp.) o organizacji matrycy znakowej 5×7 albo 4 znaków o organizacji matrycy znakowej 5×10. Pamięć CG RAM, do której można zapisać wzorce własnych znaków, ma pojemność 8×8 bitów i umieszczona jest podwójnie w przestrzeni adresowej: 00H ÷ 07H i 08H ÷ 0FH. Wybór jednej z opcji przestrzeni adresowej należy do użytkownika.

Programowalny sterownik zawiera dwa rejestry służące do programowania wyświetlacza alfanumerycznego LCD: rejestr instrukcji (sterujący) i rejestr danych. Oba te rejestry są dostępne dla zapisu i dla odczytu informacji. Adresy przypisane tym rejestrom zamieszczono w tabeli 2.3.

Tabela 2.3. Adresy rejestrów służące do programowania wyświetlacza alfanumerycznego LCD.

Rejestr	Adres szesnastkowy	Funkcja
Instrukcji	0FFF0H 0FFF1H	Zapis Odczyt
Danych	0FFF2H 0FFF3H	Zapis Odczyt

Zasady programowania wyświetlacza alfanumerycznego LCD i charakterystykę instrukcji (rozkazów) przedstawiono w [1].

2.1.4. Porty szeregowe

Do komunikacji systemu μM-DYD z komputerem PC służy łącze szeregowe o standardzie RS232, dołączane do portu szeregowego mikrokontrolera.

Standardowo portem mikrokontrolera zarządza program MONITOR-51. Użytkownik może jednakże napisać i uruchomić własny program sterujący portem szeregowym. Możliwa jest przy tym realizacja transmisji szeregowej jednoczesnej (*duplex*) - między modułem μM-DYD a komputerem PC - albo transmisji jednokierunkowej (*simplex*), tylko w ramach modułu μM-DYD.

Transmisję jednokierunkową umożliwiają wyprowadzone na złącze Z3 linie P3.0 i P3.1 portu szeregowego (muszą być zwarte zworki JP13-P3.1 i JP14-P3.0). Możliwe jest również połączenie bezpośrednio na płycie modułu linii P3.0 z P3.1.

2.1.5. Generator sygnału i brzęczyk

Na płycie modułu μM-DYD umieszczono układ wolnozmiennego generatora (*timera*), rys.2.10 [1]. Umożliwia on dołączenie przebiegu prostokątnego o częstotliwości około 1.5Hz do:

- wejścia licznika T0 - JP9 zwarte z linią P3.4;
- wejścia licznika T1 - JP10 zwarte z linią P3.5.

Brzęczyk (piezoelement) emitujący dźwięki o częstotliwości programowanej przez użytkownika można za pomocą zworki JP7 dołączyć (na płycie μM-DYD) do linii P1.1 portu P1.

2.1.6. Tor analogowy przetwornika A/C mikrokontrolera

Moduł dydaktyczny umożliwia dołączenie do mikrokontrolera 6 sygnałów analogowych, oznaczonych symbolami AN0 ÷ AN2, AN5 ÷ AN7.

Układ U15 wraz z rezystorami R6..R11 zabezpiecza wejścia analogowe mikrokontrolera przed przepięciami. Sygnały analogowe mogą być dołączane do wielostykowego złącza Z4 lub do złącz samozaciskowych ZP10 ÷ ZP15.

Wszystkie wejścia analogowe modułu dydaktycznego są wewnętrznie połączone z innymi elementami modułu:

- wejście AN0 połączone jest z wyjściem przetwornika cyfrowo-analogowego;
- wejście AN1 umożliwia pomiar zewnętrznego sygnału doprowadzonego do wejścia programowalnego filtru dolnoprzepustowego;
- wejście AN2 umożliwia pomiar sygnału na wyjściu programowalnego filtru dolnoprzepustowego;
- wejście AN5 nie ma połączeń wewnętrznych, przeznaczone jest tylko dla użytkownika;
- wejście AN6 połączone jest z potencjometrem znajdującym się na płytce klawiatury; w ten sposób można zadawać dowolne napięcia w całym zakresie przetwarzania przetwornika;
- wejście AN7 wraz z rezystorem korekcyjnym służy do pomiarów temperatury elementami wrażliwymi na zmianę temperatury, np. termorezystory, źródła prądowe itp.

Taki sposób wewnętrznych połączeń zwiększa zakres zastosowań przetwornika analogowo-cyfrowego mikrokontrolera bez konieczności użycia dodatkowych przewodów i elementów.

Jednym z istotniejszych elementów toru analogowego jest programowalny filtr dolnoprzepustowy TLC 14 firmy Texas Instruments (układ U16). Filtr ten stosowany jest przede wszystkim do kształtowania sygnału analogowego uzyskiwanego metodą modulacji szerokości impulsów PWM (*Pulse Width Modulation*). Typowym przykładem jest generowanie sygnału sinusoidalnego metodą modulacji szerokości impulsów na wyjściu P1.2/CC2 z równoczesnym generowaniem sygnału taktującego na wyjściu P1.3/CC3.

Zastosowana w module dydaktycznym konfiguracja pozwala na filtrowanie sygnałów zewnętrznych lub taktowanie filtru dolnoprzepustowego ze-

wnętrznym sygnałem prostokątnym w zależności od położenia zworek: JP5 i JP6.

2.1.7. Przetwornik cyfrowo-analogowy

Moduł dydaktyczny µM-DYD zawiera przetwornik cyfrowo-analogowy AD7524.

Jest to ośmiobitowy mnożący przetwornik z równoległym buforem, przeznaczony do bezpośredniej współpracy z układami mikroprocesorowymi.

Przetwornik A/C jest umieszczony w przestrzeni adresowej pamięci zewnętrznej RAM mikrokontrolera pod adresem 0FFD0H. Sygnały sterujące wejścia $\overline{CS}$ i $\overline{WR}$, są wytwarzane w układzie GAL20V8 (U19).

Dokładny opis i schematy połączeń przetwornika A/C znajdują się w [1].

2.2. Moduł sterownika µM – 537F

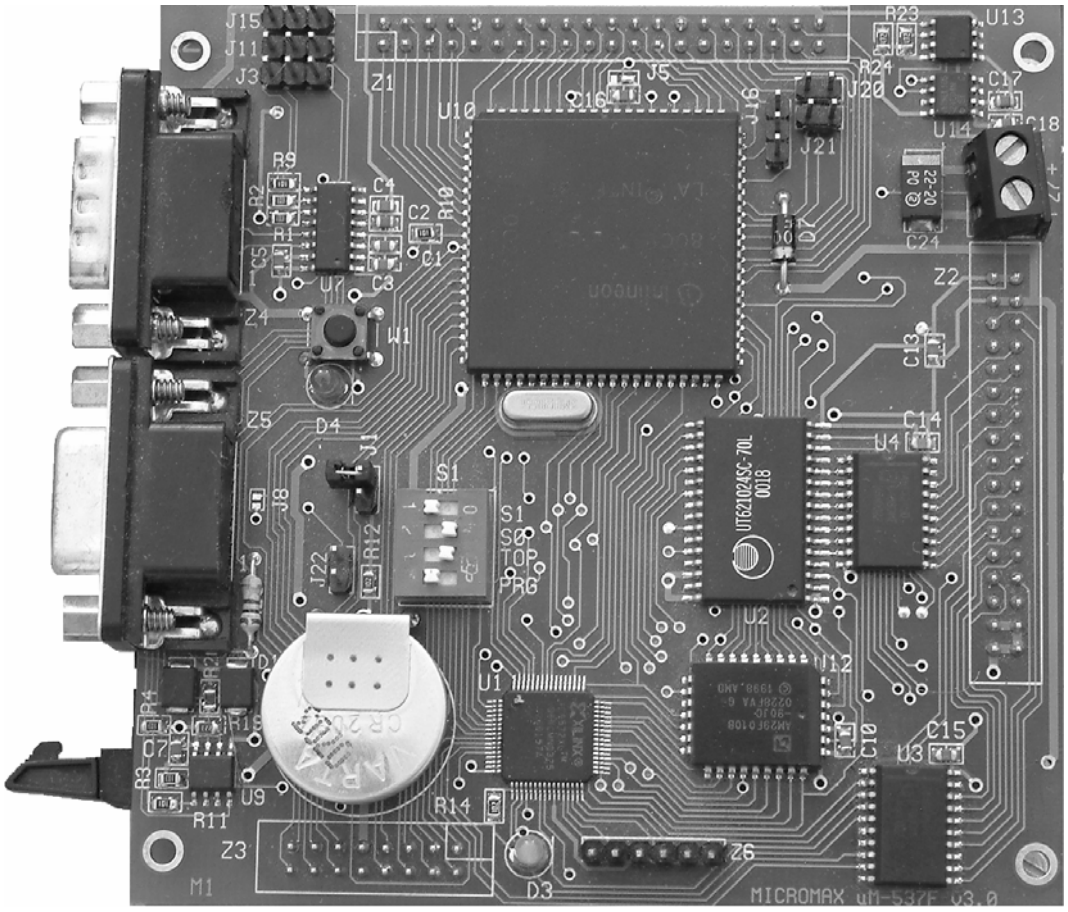
2.2.1. Wstęp

Moduł µM-537F firmy MICROMAX jest uniwersalnym sterownikiem mikroprocesorowym przeznaczonym do wielorakich zastosowań w dziedzinie pomiarów i sterowania. Moduł µM-537F jest rozwinięciem produkowanego przez firmę MICROMAX modułu µM-537. Oba moduły są kompatybilne pod względem mechanicznym i elektrycznym. Nowy produkt został rozszerzony tak, aby wcześniejsze aplikacje przeznaczone dla µM-537 działały bez modyfikacji kodu. Zachowano w nim istniejące modele pamięci. Zaletą nowego modułu jest zastąpienie kłopotliwej w użyciu pamięci EPROM pamięcią typu FLASH, której programowanie odbywa się przez złącze RS232. Nowy moduł posiada powiększoną do 128kB pamięć programu i danych (bankowane po 32kB). Wygląd zewnętrzny modułu sterownika µM-537F przedstawia rysunku 2.6.

Moduł µM-537F może pracować jako niezależny, samodzielny sterownik lub współpracować z innymi modułami obiektowymi. Do programowania i uruchamiania oprogramowania użytkownika niezbędny jest komputer PC

wyposażony w port szeregowy RS232. Z modułem dostarczane jest oprogramowanie do programowania pamięci FLASH. Dodatkowo dostępne jest oprogramowanie IDE51, które umożliwia edycję, kompilację, ładowanie i krokowe uruchamianie własnych programów.

Moduł może również współpracować z innymi narzędziami programistycznymi np. firmy KEIL.



Rys. 2.6. Wygląd zewnętrzny modułu sterownika µM-537F.

2.2.2. Warunki pracy

- **Zasilanie**

Moduł µM-537F wymaga zasilania napięciem stabilizowanym +5V. Maksymalny pobór prądu wynosi około 100mA. Sposób podłączenia napięcie zasilającego przedstawia tabela 2.4.

Tabela 2.4. Zasilanie modułu sterownika µM-537F

	Podłączenie zasilania do µM-537F	
	Złącze Z2	Złącze Z7
+ 5V	3, 4	+ 5V
GND	1, 2	GND

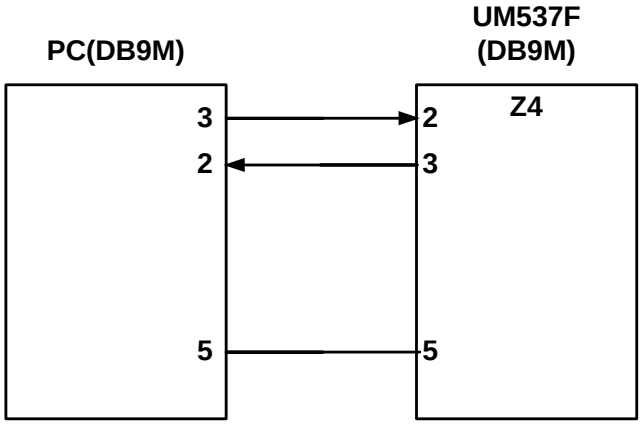
- **Połączenie z komputerem PC**

Aby połączyć moduł µM-537F z komputerem PC należy wykorzystać przewód trójżyłowy lub dwużyłowy w ekranie.

Wykorzystywane są tylko linie TX, RX i GND.

Moduł posiada dwa złącza szeregowe Z4 i Z5. Do programowania pamięci FLASH oraz do komunikacji ze środowiskiem IDE51 wykorzystywane jest złącze Z4.

Połączenie modułu sterownika z komputerem PC przedstawia rysunek 2.7.



Rys. 2.7. Połączenie modułu sterownika z komputerem PC poprzez złącze Z4.

- **Uruchamianie oprogramowania użytkowego**

Programy użytkownika mogą być uruchamiane w trybie testowym lub docelowym.

Tryb testowy umożliwia współpracę ze środowiskiem programowym IDE51 firmy MICROMAX lub innym np. firmy KEIL. W obu przypadkach do modułu musi zostać zaprogramowany odpowiedni program MONITOR.

Za pomocą przełącznika konfiguracyjnego Sl należy wybrać odpowiedni model pamięci. Z poziomu IDE51 lub KEIL możliwe jest krokowe uruchamianie aplikacji, ustawianie pułapek, etc.

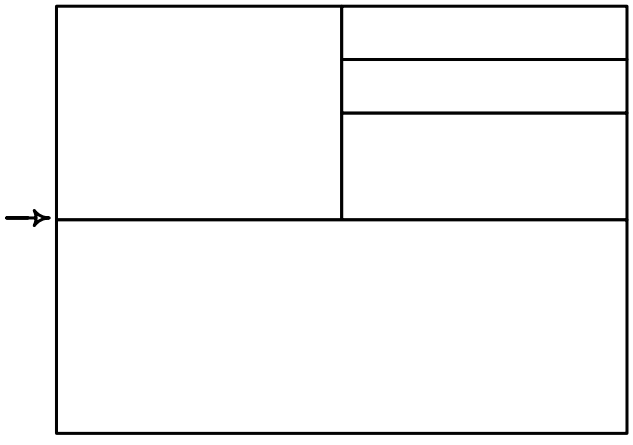
Tryb docelowy pozwala na samodzielną pracę modułu w układzie docelowym.

- **Modele pamięci**

Standardowo dla modułu µM-537F przewidziane są dwa modele pamięci M0 i M8. Dodatkowo dostępne są jeszcze dwa modele R0 i R1, które umożliwiają programowanie modułu µM-537F bez konieczności użycia pamięci FLASH.

Pozycje S1, S0, TOP decydują o modelu pamięci. Pozycja PRG jest wykorzystywana do programowania pamięci. W przypadku wyboru modelu pamięci PRG należy ustawić w pozycji wyłączony - logiczne 0.

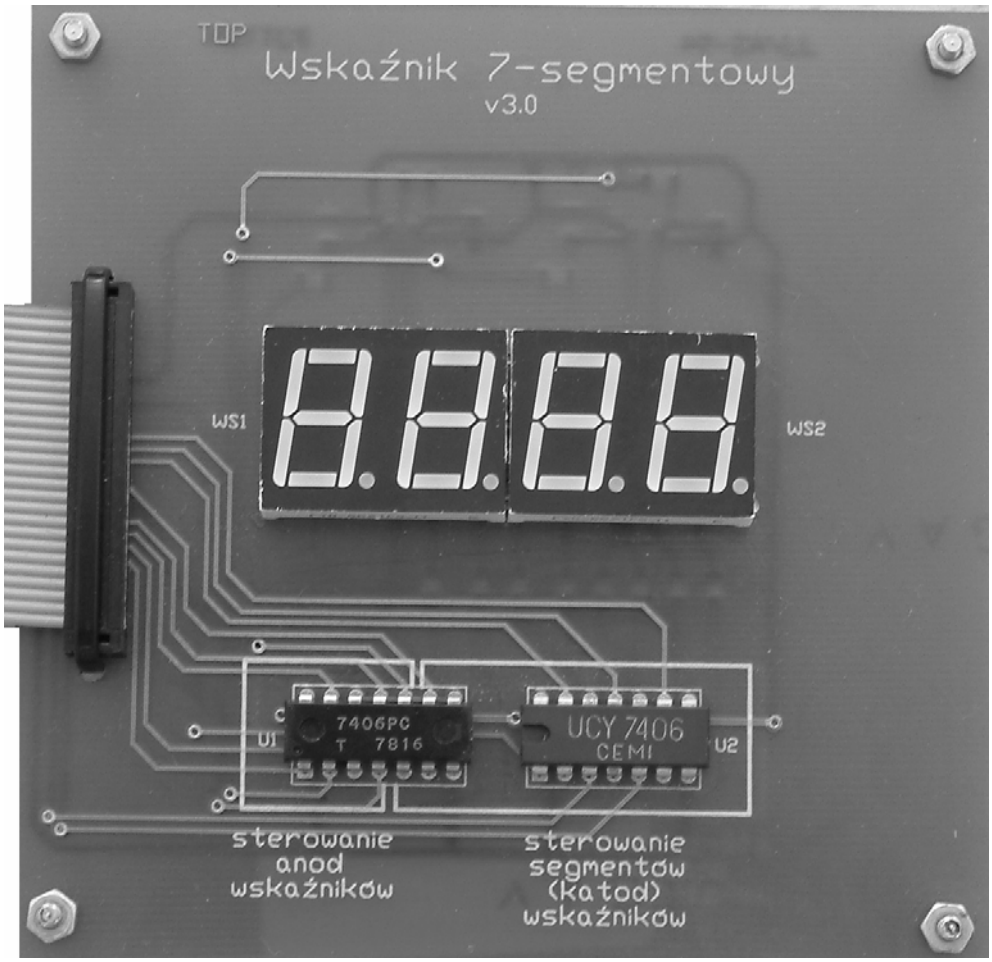
Podczas zajęć laboratoryjnych wykorzystywany jest model M8 w trybie testowym przedstawiony na rysunku 2.8.



Rys. 2.8. Organizacja modelu pamięci M8 w trybie testowym.

2.3. Makieta dydaktyczna „Wyświetlacz LED”

Makieta przeznaczona jest do testowania programów sterujących statycznie i dynamicznie wskaźniki 7-segmantowe. Wygląd zewnętrzny makiety przedstawia rysunek 2.9.



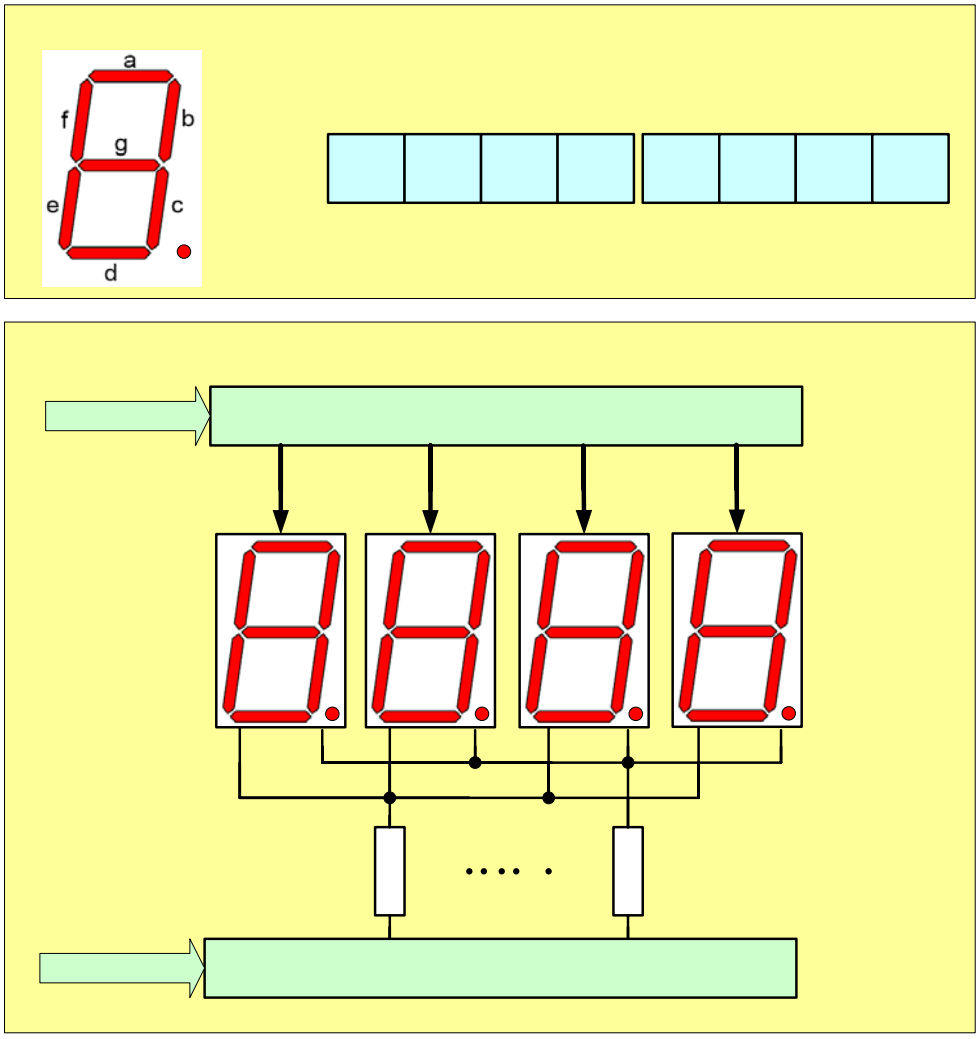
Rys. 2.9. Makieta dydaktyczna „Wyświetlacz LED”.

Makieta zawiera 4 wskaźniki 7-segmentowe. Schemat układu sterowania wyświetlaczami przedstawia rysunek 2.10.

Sterowanie wskaźników i segmentów jest następujące:

- cztery wskaźniki W1 ÷ W4 sterowane są niezależnymi liniami portu P5.0 ÷ P5.3, wskaźnik lewy skrajny W1 linią P5.0, wskaźnik prawy skrajny W4 linią P5.3;

- stan jedynki logicznej na liniach P5.0 ÷ P5.3 uaktywnia skojarzony z linią wskaźnik, a stan zera logicznego wygasza wskaźnik;
- segmenty wskaźników sterowane są liniami portu P4.0..P4.7, linia P4.0 steruje segmentem a, linia P4.1 - segmentem b, ... , a linia P4.7 steruje segmentem h;
- stan jedynki logicznej na linii P4.0..P4.7 wywołuje przepływ prądu przez wybrany segment, a tym samym zaświecenie segmentu, jeśli wskaźnik jest uaktywniony; stan zera logicznego wygasza segment;

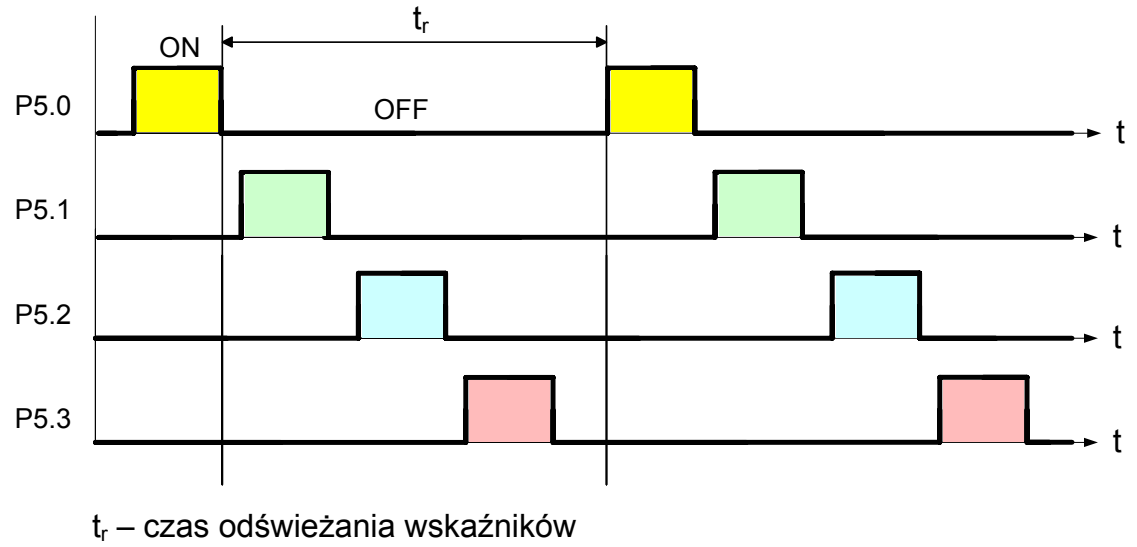


Rys. 2.10. Schemat blokowy układu sterowania wskaźników 7 - segmentowych.

Jak już zostało podane na wstępie makieta przeznaczona jest do testowania programów sterujących statycznie i dynamicznie wskaźniki 7 - segmentowe. Wyświetlanie statyczne znaków zakłada, że informacja wyświetlana

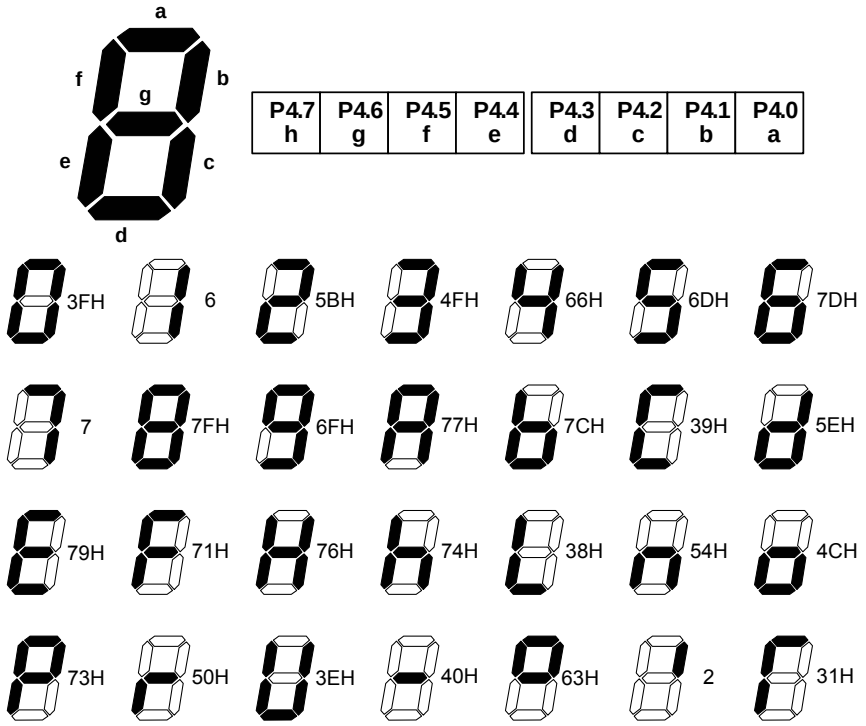
jest na wszystkich wskaźnikach równocześnie. W wyświetlaniu dynamicznym w jednej chwili aktywny jest tylko jeden wskaźnik, pozostałe są wygaszone. W następnej chwili wygasza się bieżący, aktywny wskaźnik i uaktywnia następny. Wygaszanie i uaktywnianie powtarza się dla pozostałych wskaźników. Pamiętając o bezwładności oka, odpowiednio szybkie zmiany wyświetlanej treści powodują widzenie całej treści, bez zjawiska migotania pola odczytowego. Jako dolną granicę, minimalną szybkość zmian należy przyjmować częstotliwość 20 Hz.

Na rysunku 2.11 przedstawiono sygnały sterujące wyświetleniem informacji na 4 wskaźnikach.



Rys. 2.11. Zasada wyświetlania dynamicznego.

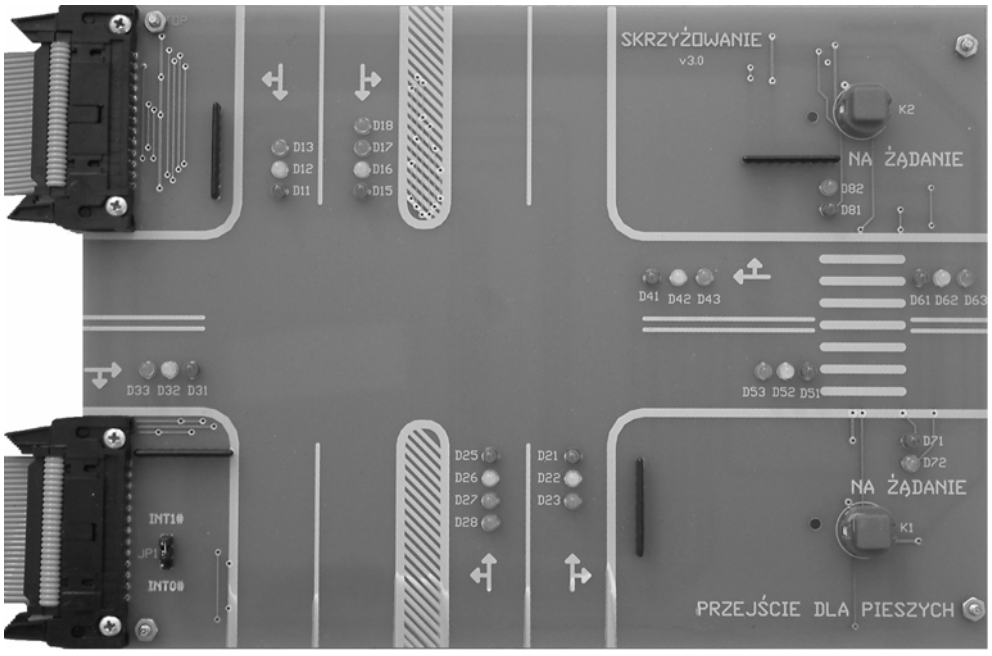
Wskaźniki siedmiosegmentowe służące do wyświetlania informacji w postaci znaków świetlnych nadają się szczególnie do prezentacji cyfr i niestety bardzo niewielu innych znaków, niektórych małych i dużych liter oraz symboli. Na rysunku 2.12 przedstawiono kształty wyświetlanych znaków oraz ich kody sterujące.



Rys. 2.12. Kształty oraz kody sterujące znaków wyświetlanych na wskaźniku 7-segmentowym.

2.4. Makieta dydaktyczna „Skrzyżowanie”

Makieta umożliwia poznanie zasad sterowania czasowego obiektem, jakim jest skrzyżowanie ulic. Wygląd zewnętrzny makiety przedstawia rysunek 2.13.



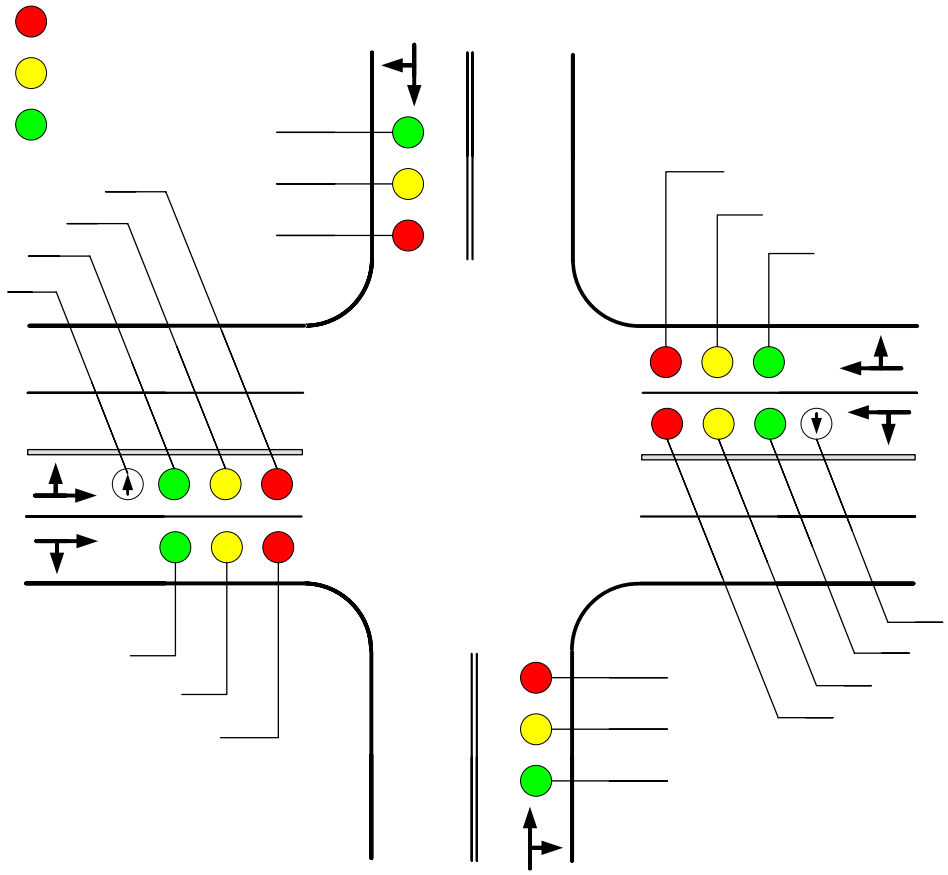
Rys. 2.13. Makieta dydaktyczna „Skrzyżowanie”.

Wykorzystując makietę możemy symulować następujące skrzyżowania:

- tylko dla ruchu kołowego z wyłączeniem pieszych;
- przejście dla pieszych;
- połączenie przejścia dla pieszych (na żądanie) ze sterowanym ruchem kołowym.

Potrzebne do przygotowania i testowania oprogramowania wzorcowe odcinki czasu generowane są sprzętowo lub programowo. W zależności od stopnia przygotowania możliwe jest również wykorzystanie przerwań lub testowanie programowej sytuacji na skrzyżowaniu.

Rysunek 2.14 przedstawia skrzyżowanie dla ruchu kołowego. Skrzyżowanie nie należy do całkiem typowych, ponieważ ma rozdzielone pasy ruchu i oddzielne sygnalizatory do skrętu w lewo.



Rys. 2.14. Pojedyncze skrzyżowanie ulic.

Poszczególne diody elektroluminescencyjne (LED) imitują odpowiednie światła i sterowane są przez mikrokontroler liniami portu P4 oraz P5. Przyporządkowanie portom linii przedstawia tabela 2.4.

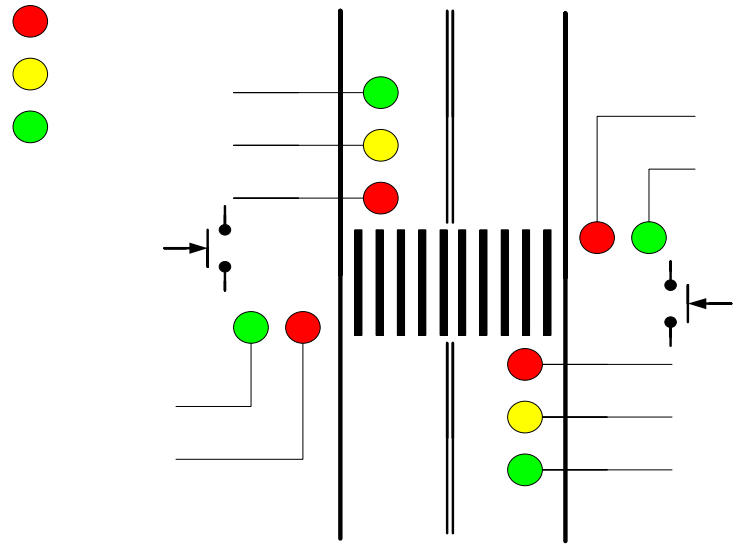
Tabela 2.4. Sterowanie diod LED na skrzyżowaniu dla ruchu kołowego.

Kolor diody LED	Kierunek pionowy	Kierunek poziomy	
		prawoskręt	lewoskręt
czerwony	P5.0	P4.0	P4.4
żółty	P5.1	P4.1	P4.5
zielony	P5.2	P4.2	P4.6
zielona strzałka	P5.3	P4.3	P4.7

Zaświecenie i zgaszenie wybranej diody wynika ze stanu linii:

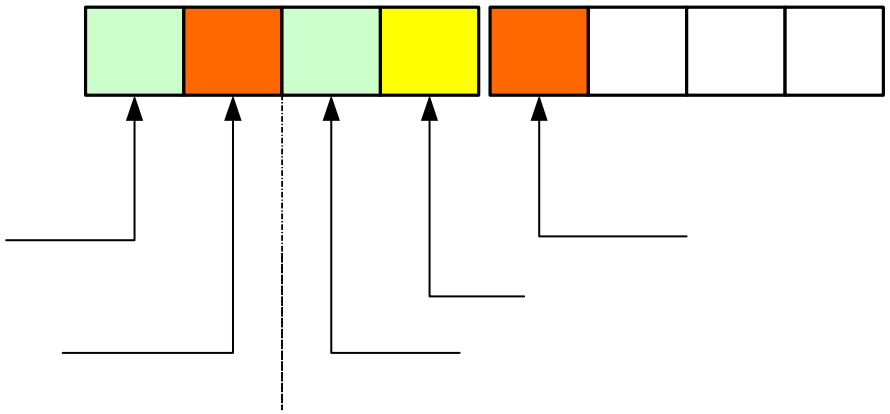
- zera logicznego, zaświecenie diody LED;
- jedynki logicznej, zgaszenie diody LED.

Kolejny rysunek 2.15 przedstawia przejście dla pieszych.



Rys. 2.15. Przejście dla pieszych.

Poszczególne diody elektroluminescencyjne (LED) imitują odpowiednie światła i sterowane są przez mikrokontroler liniami portu P5. Przyporządkowanie linii portu P5 przedstawia rysunek 2.16.



Rys. 2.16. Przyporządkowanie linii portu P5 na przejściu dla pieszych.

Zapalanie zielonego światła na przejściu dla pieszych może być sterowane przez samych pieszych. Przeważnie na sygnalizatorze dla pieszych znajduje się przycisk STOP umożliwiający włączenie czerwonego światła dla pojazdów i zielonego dla pieszych. Obsługa przycisku STOP odbywa się poprzez programowy odczyt linii portu P3.2 lub P3.3 w zależności od stanu zworki JP1 umieszczonej na płytce makiety „Skrzyżowanie". Obie linie mogą być również wykorzystane jako wejścia zgłaszające przerwania:

- P3.2 jako wejście $\overline{INT0}$
- P3.3 jako wejście $\overline{INT1}$

Czas świecenia się światła zielonego dla pieszych jest ograniczony. Po ponownym zapaleniu się światła zielonego dla pojazdów musi upłynąć pewien odcinek czasu, po którym pieszy może ponownie zatrzymać pojazdy. Pominięcie takiej blokady przycisku pieszego może spowodować całkowite zatrzymanie ruchu pojazdów.

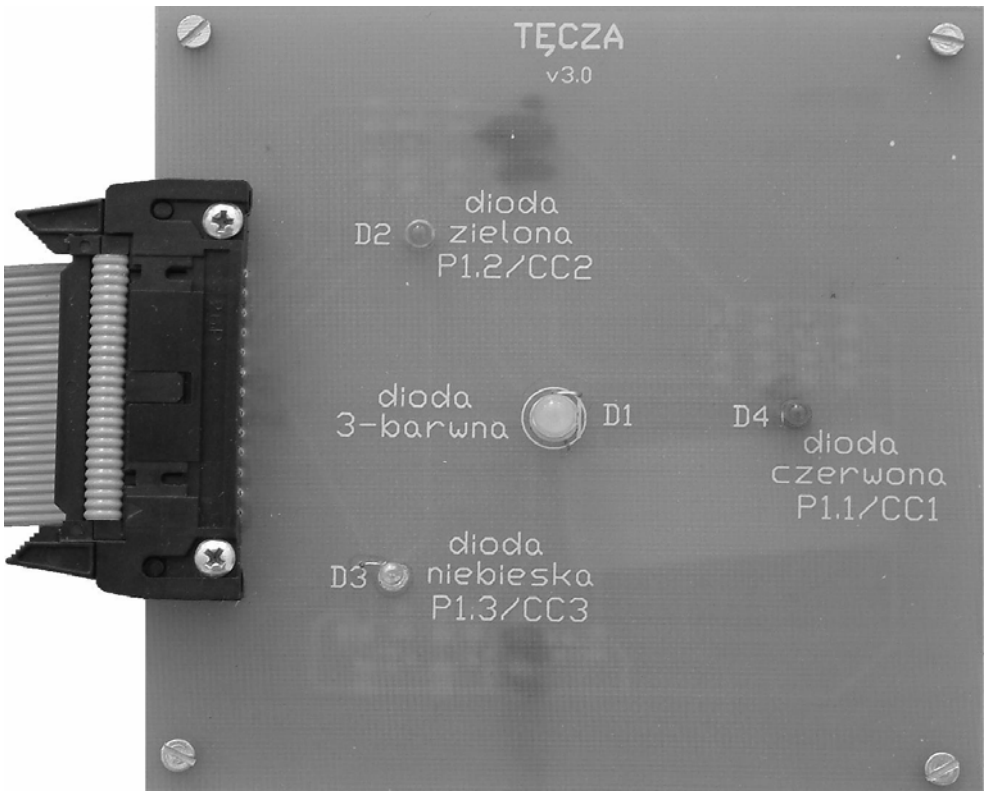
Port P5

P5.7

zielony

2.5. Makieta dydaktyczna „Tęcza”

Makieta „Tęcza” przeznaczona jest do wizualnej prezentacji efektów modulacji szerokości impulsów PWM (*Pulse Width Modulation*) przez zmianę jaskrawości świecenia i zmianę barwy diod elektroluminescencyjnych (LED). Wygląd zewnętrzny makiety przedstawia rysunek 2.17.



Rys. 2.17. Makieta dydaktyczna „Tęcza”.

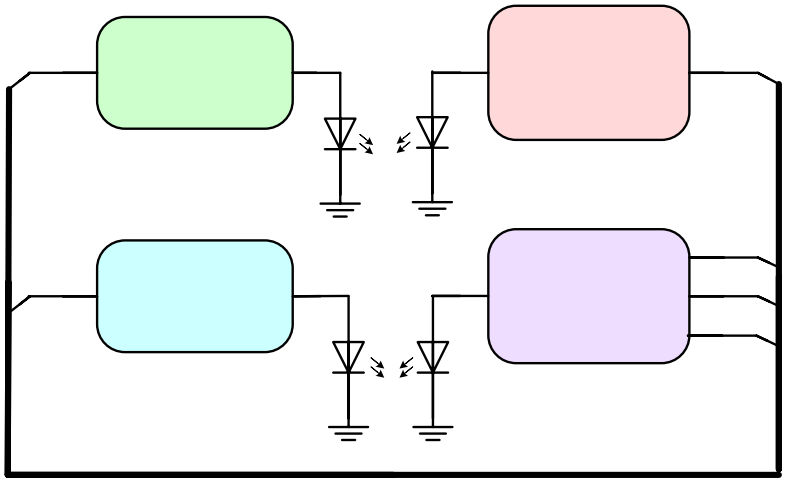
Makieta zawiera jedną diodę trójbarwną D1 złożoną z czterech wewnętrznych diod: zielonej, czerwonej i dwóch diod niebieskich oraz trzy diody monochromatyczne:

- D2 - zieloną;
- D3 - niebieską;
- D4 - czerwona.

Diody elektroluminescencyjne D1 ÷ D4 sterowane są liniami portu P1, przedstawione zostało to na rysunku 2.18:

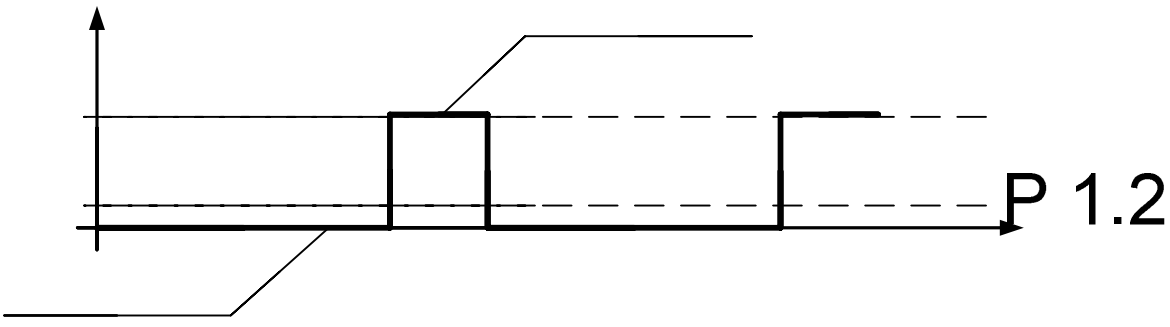
- P1.1 - dioda czerwona D4 i wewnętrzna dioda czerwona diody trójbarwnej D1;

- P1.2 - dioda zielona D2 i wewnętrzna dioda zielona diody trójbarwnej D1;
- P1.3 - dioda niebieska D3 i obie wewnętrzne diody niebieskie diody trójbarwnej D1.



Rys. 2.18. Schemat blokowy makiety „Tęcza".

Do sterowania wszystkich diod wykorzystano klucze tranzystorowe (tranzystory p-n-p) sterowane bezpośrednio z linii portu P1. Zerowej wartości sygnału na liniach portu P1 odpowiada świecenie diody, a stanowi wysokiemu - zgaszenie diody rysunek 2.19.



Rys. 2.19. Wartość średnia prądu płynącego przez diodę w zależności od współczynnika wypełnienia impulsów.

Zmieniając współczynnik wypełnienia impulsów ε zmianie ulega również wartość średnia prądu płynącego przez diodę $I_{śr}$. Rolę układu całkującego pełni oko, które nie zauważa zmian częstszych niż 16 ÷ 20 zmian na sekundę.

2.6. Zasilacz

Dla modułowego systemu µM-DYD standardowym napięciem zasilającym jest +5V oraz +12V. Maksymalna odchyłka napięcia zasilania wynosi ±0,25V. Napięcia te pobierane są z modułu zasilacza µM-UNI wyposażonego w stabilizator napięcia +5V oraz +12V.

3. SYSTEM IDE51
- ŚRODOWISKO PROGRAMOWE

3.1. Wstęp

System **IDE51** jest zintegrowanym środowiskiem programowym, łączącym w jedną całość edytor tekstowy, asembler, linker oraz debugger płytkowy. Dzięki takiemu połączeniu jest możliwe szybkie i efektywne przygotowywanie oprogramowania użytkowego dla systemów opartych o mikrokontrolery 8 bitowe z rodziny ‘51. System umożliwia pisanie wielomodułowych programów w języku asemblera, kompilację, wykrywanie błędów syntaktycznych na poziomie źródłowym oraz błędów logicznych na poziomie programu wynikowego, przesyłanie i uruchamianie programów w systemie docelowym z mikrokontrolerem rodziny ‘51.

3.2. Zasady posługiwania się systemem IDE51

Wywołanie systemu IDE51 odbywa się po wykonaniu dyrektywy IDE51.EXE. Bezpośrednio po tej czynności pojawia się okno z nazwą i numerem wersji systemu oraz z nazwą właściciela - rysunek 3.1.

Rys. 3.1. Okno powitalne systemu IDE51.

Naciśnięcie dowolnego klawisza powoduje wywołanie okna głównego systemu - rysunek 3.2.

Rys. 3.2. Okno główne systemu IDE51.

Okno główne systemu **IDE51** jest podzielone na trzy następujące części:

- wiersz menu (u góry ekranu), składający się z pól **F**ile, **E**dit, **R**un, **C**ompile, **P**roject, **O**ptions, **D**ebug, **B**reak/**W**atch;

- obszar środkowy przeznaczony na rozwijanie okien;
- wiersz opisów kluczy (u dołu ekranu), zawierający informacje na temat funkcji przypisanych wybranym klawiszom Fn.

Przejście do poszczególnych opcji wiersza menu następuje na kilka sposobów. Najprostszym jest wprowadzenie z klawiatury wyróżnionej litery w nazwie opcji, np.: **F**ile. Innym sposobem jest wykorzystanie klawiszy kierunkowych poziomych, a po przemieszczeniu się na wybrane pole menu, naciśnięcie klawisza Enter. Wybranie funkcji określonej przez opcję podmenu odbywa się w sposób analogiczny do wyboru funkcji z menu głównego (tym razem używa się klawiszy kierunkowych pionowych) i może powodować wywołanie kolejnego podmenu. Należy dodać, że jeśli po wybraniu pierwszego podmenu zostaną użyte klawisze kierunkowe poziome, to na ekran będą przywołane sąsiednie podmenu głównego menu.

W celu opuszczenia podmenu albo w przypadku omyłkowego wyboru podmenu, można cofnąć się na wyższy poziom menu naciskając klawisz Esc.

Wybieranie funkcji głównego menu może być znacznie przyspieszone przez użycie klawisza Alt. Naciśnięcie go w dowolnym momencie wraz z pierwszą litera pola głównego menu powoduje natychmiastowe wykonanie funkcji związanej z tym polem np.: Alt-F, Alt-O, Alt-P.

Poza wymienionymi klawiszami wiele użytecznych funkcji spełniają klawisze Fn, Alt-Fn oraz Ctrl-Fn, których lista pojawia się w wierszu opisu kluczy. Po naciśnięciu klawisza Alt w wierszu opisu kluczy pojawia się lista klawiszy dostępnych w połączeniu z klawiszem Alt, natomiast po naciśnięciu klawisza Ctrl lista klawiszy dostępnych z klawiszem Ctrl.

Standardowe znaczenie klawiszy funkcyjnych:

F2 - zapamiętanie na dysku tekstu znajdującego się w edytorze (w oknie aktywnym),

F3 - otwarcie nowego okna edycyjnego i ewentualne wczytanie zbioru tekstowego (z domyślnym rozszerzeniem ***.asm),**

F4 - uruchomienie programu z punktem zatrzymania w miejscu położeniu kursora,

F5 - powiększenie/zmniejszenie okna aktywnego,

F6 - praca z ograniczoną prędkością,

F7 - praca krokowa,

F8 - praca krokowa z omijaniem podprogramów (podprogramy są wykonywane z szybkością maszynową),

F9 - kompilacja programu w tekstowym oknie aktywnym,

F10 - przełączanie **menu główne/okno edycyjne**,

Ctrl-F2 - transmisja kodu wynikowego do mikrokontrolera,

Ctrl-F5 - zmiana położenia i rozmiaru okna aktywnego,

Ctrl-F6 - praca z ograniczoną prędkością (podprogramy są wykonywane w jednym kroku),

Ctrl-F7 - śledzenie obszarów pamięci,

Ctrl-F8 - wstawianie/usuwanie „pułapki" programowej,

Ctrl-F9 - uruchamianie programu z szybkością maszynową,

Alt-0..9 - wykaz otwartych okien wraz z możliwością ich uaktywnienia lub usunięcia,

Alt-F7 - wyświetlenie poprzedniego komunikatu o błędzie (po procesie kompilacji),

Alt-F8 - wyświetlenie następnego komunikatu o błędzie,

Alt-X - opuszczenie systemu IDE51.

Zarządzanie systemem okien:

Okno jest ograniczoną powierzchnią ekranu, którą można otworzyć, uaktywnić, zamknąć, skasować, przesunąć, zmienić rozmiar. Otwarcie okna wiąże się z utworzeniem nowego zbioru przeznaczonego do edycji. W systemie

IDE51 może być otwartych wiele okien jednocześnie, ale tylko jedno w danej chwili może być aktywne. Okno aktywne posiada podwójną ramkę. Zamknięcie okna polega na jego "zamrożeniu" tzn. okno jest pamiętane, ale nie jest widoczne. Okno zamknięte może być uaktywnione.

Skasowanie okna wiąże się z usunięciem go z pamięci programu. Okno skasowane nie może być uaktywnione. Aby ponownie pracować na oknie skasowanym należy go najpierw otworzyć.

W systemie IDE51 wyróżnia się dwa rodzaje okien:

- okna edycyjne oznaczane kolejno liczbami począwszy od l;
- okna specjalne do których zalicza się okno: MESSAGES, REGS, TERMINAL i WATCH.

Okna specjalne są zawsze otwarte i nie mogą być kasowane. Możliwe jest ich uaktywnianie lub zamykanie.

Dostępne są następujące operacje związane z manipulacją oknami:

Otwarcie nowego okna do edycji - wybór File/Load lub naciśnięcie F3.

Uaktywnienie okna - naciśnięcie Alt-0 i wybór żądanego okna poprzez użycie klawiszy kierunkowych oraz naciśnięcie Enter lub naciśnięcie Alt oraz numeru okna 1 ÷ 9 - dla okien edycyjnych lub wybrane litery - dla okien specjalnych.

Zamknięcie okna - naciśnięcie Alt-0 i wybór żądanego okna poprzez użycie klawiszy kierunkowych oraz wybranie opcji Closed (klawisz C).

Kasowanie okna - naciśnięcie Alt-0 i wybór żądanego okna poprzez użycie klawiszy kierunkowych oraz wybranie opcji Delete (klawisz D).

Przesunięcie aktywnego okna - naciśnięcie Ctrl-F5 i użycie klawiszy kierunkowych w celu pożądanego umieszczenia okna, zakończenie operacji przez naciśnięcie Enter.

Zmiana rozm. aktywnego okna - naciśnięcie Ctrl-F5 i użycie klawiszy kierunkowych z klawiszem Shift w celu pożądanego umieszczenia okna, zakończenie operacji przez naciśnięcie Enter.

Zoom aktywnego okna - naciśnięcie F5 powoduje rozwinięcie aktywnego okna na cały ekran. Ponowne naciśnięcie F5 powoduje powrót do poprzedniego rozmiaru okna.

W praktyce, najczęściej są używane następujące okna: **okno edycyjne** (pisanie, kompilacja, poprawianie programu, praca krokowa), **okno watch** (śledzenie i modyfikacja zawartości obszarów pamięci), **okno rejestrów** (odczyt zawartości podstawowych rejestrów mikroprocesora - uaktualniane tylko przy pracy krokowej).

3.3. Znaczenie menu i opcji

3.3.1. Menu File (Alt-F)

Menu **File** pozwala na utworzenie nowego i otwarcie już istniejącego zbioru w oknie edycyjnym - rysunek 3.3.

Dodatkowo możliwe jest zapisanie zbioru oraz operacja wyjścia z programu **IDE51**.

Rys. 3.3. Okno menu File.

Load (F3)

Opcja **File/Load** wyświetla okno dialogowe umożliwiające wybór zbioru w celu otwarcia w oknie edycyjnym.

Save (F2)

Opcja **File/Save** umożliwia zapamiętanie zbioru na dysku z aktualnie aktywnego okna edycyjnego.

Quit (Alt-X)

Opcja **File/Quit** pozwala opuścić program IDE51 i powrócić do systemu operacyjnego. Jeżeli w zbiorach dokonano zmian bez zapamiętania to program zapyta czy wyjść bez zapamiętania.

3.3.2. Menu Edit (Alt-E)

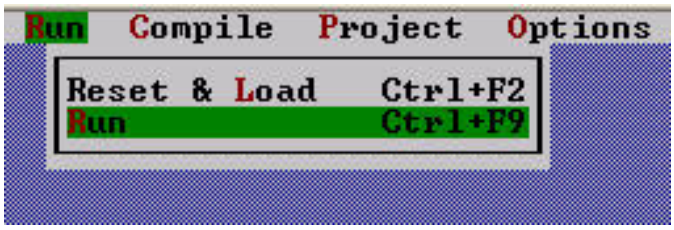
Rys. 3.4. Okno menu Edit.

Menu **Edit** pozwala wykonywać operacje na tekście w aktywnym oknie edycyjnym rysunek 3.4. Aby operować tekstem należy najpierw go zaznaczyć. Początek tekstu zaznacza się przy użyciu klawiszy Ctrl-K B, przy czym kursor powinien znajdować się na początku tekstu. Następnie przesuwa się kursor na koniec zaznaczanego tekstu i naciska klawisze Ctrl-K K. W celu

zaznaczenia pojedynczego słowa przesuwa się kursor na koniec słowa i naciska klawisze Ctrl-K T. Całą linię zaznacza się naciskając Ctrl-K L.

3.3.3. Menu Run (Alt-R)

Menu **Run** pozwala na zerowanie programowe mikrokontrolera w systemie docelowym oraz ładowanie i uruchamianie programu rysunek 3.5.



Rys. 3.5. Okno menu Run.

Reset&Load (Ctrl-F2)

Opcja **Reset & Load** pozwala wyzerować mikrokontroler w systemie docelowym (zerowanie programowe). Opcja jest aktywna o ile system znajduje się pod kontrolą programu MONITOR. Wtedy system docelowy zostanie doprowadzony do stanu początkowego, a następnie przeniesiony będzie program z komputera do systemu docelowego. Jeżeli program MONITOR nie kontroluje systemu docelowego to należy zastosować tzw. „zimny start" tj. zerowanie sprzętowe.

Run (Ctrl-F9)

Opcja **Run** uruchamia program znajdujący się w systemie docelowym. Program zostaje uruchomiony praktycznie bez kontroli programu IDE51.

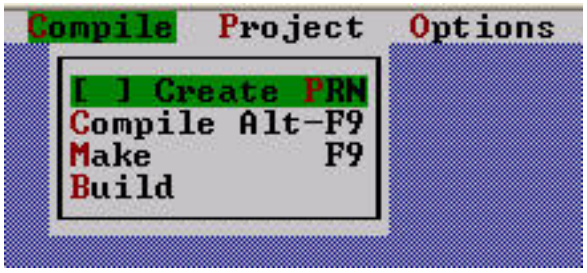
Sterowanie programu IDE51 zostaje przekazane do okna TERMINAL komunikacji bezpośredniej z procesorem peryferyjnym. Praca programu zostaje zatrzymana, gdy:

1. Napotkana zostanie pułapka.
2. Zostanie wyzerowany mikrokontroler.

Dopóki program IDE51 pracuje w oknie TERMINAL, to nie są odświeżane okna: WATCH oraz REGS, ani „ślady" realizacji programu w zbiorze źródłowym. Jest to spowodowane uprzywilejowanym trybem komunikacji okna TERMINAL z procesorem peryferyjnym.

3.3.4. Menu Compile (Alt-C)

Menu **Compile** pozwala kompilować program znajdujący się w aktywnym oknie edycyjnym - rysunek 3.6.



Rys.3.6. Okno menu Compile.

[ ] Create PRN

Opcja **[ ] Create PRN** jest przełącznikiem włączającym lub wyłączającym możliwość tworzenia zbioru typu *.PRN będącego wynikiem kompilacji.

Włączanie/wyłączanie odbywa się przy użyciu klawisza Enter.

Compile (Alt-F9)

Opcja **Compile** kompiluje program zawarty w oknie aktywnym do postaci pośredniej (*.OBJ). Podczas kompilacji pojawia się okno informujące o jej przebiegu i rezultatach. W przypadku wykrycia błędów zostanie otwarte i uaktywnione okno MESSAGES, które wyświetla listę błędów.

Make (F9)

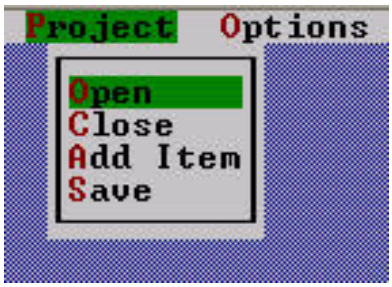
Opcja **Make** kompiluje do postaci wynikowej (tj. *.HEX) program zawarty w aktywnym oknie lub zdefiniowany w postaci projektu. Opcja **Make** tłumaczy tylko te moduły, które zostały zaktualizowane.

Build

Opcja **Build** jest podobna do **Make** oprócz tego, że jest bezwarunkowa. Realizuje kompilację wszystkich modułów niezależnie od tego, czy były już kompilowane. Opcję **Build** należy używać szczególnie wtedy, gdy w sposób istotny zostały zmienione opcje kompilatora.

3.3.5. Menu Project (Alt-P)

Menu **Project** służy do definicji struktury modułów składających się na program docelowy, rysunek 3.7. Struktura modułów jest przechowywana w zbiorach typu *.PRJ. Programy jednomodułowe mogą być uruchamiane bez definiowania projektu. Postać menu:



Rys. 3.7. Okno menu Project.

Open

Opcja **Open** powoduje otwarcie okna PROJECT i wczytanie do niego danych ze zbioru typu *.PRJ. Do podania nazwy projektu służy okno analogiczne do okna zadawania nazwy zbioru dla edytora (File/Load). Po wejściu w okno PROJECT można przeglądać strukturę zbiorów tworzących projekt, dodać następny zbiór przez wciśnięcie klawisza Insert lub usunąć zbiór z listy za pomocą klawisza Delete. Położenie zbioru dodawanego jest determinowane aktualnym położeniem linii kursorowej w oknie PROJECT. Wciśnięcie klawisza Enter powoduje rozpoczęcie edycji wskazanego zbioru.

Close

Opcja **Close** zamyka projekt otwarty opcją **Open**.

Add

Opcja **Add Item** powoduje dodanie nazwy zbioru do listy zbiorów tworzących projekt, analogicznie jak klawisz Insert w oknie PROJECT.

Save

Opcja **Save** powoduje zapamiętanie w zbiorze aktualnej postaci okna PROJECT.

3.3.6. Menu Options (Alt-O)

Menu **Options** zawiera szereg opcji, które umożliwiają podgląd i zmianę ustawień dla asemblera i debuggera - rysunek 3.8.

Rys. 3.8. Okno menu Options.

Compiler

Opcja **Compiler** umożliwia wybór typu mikrokontrolera z rodziny ‘51, którego predefiniowane symbole będą rozpoznawane przy kompilacji, rysunek 3.9. Wybór typu mikrokontrolera następuje przy użyciu klawiszy kierunkowych lub wyróżnionych pełnych nazw. Naciśnięcie klawisza Enter zmienia stan podopcji na przeciwny.

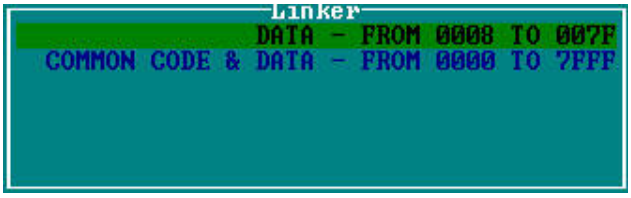
Wyjście z opcji następuje poprzez wybór pola Ok i potwierdzenie przez naciśnięcie Enter lub przez naciśnięcie klawisza Esc i zaniechanie zmian.

Rys. 3.9. Okno wyboru typu mikrokontrolera.

Wybranie typu mikrokontrolera powoduje automatyczne rozpoznawanie predefiniowanych nazw rejestrów SFR charakterystycznych dla tego mikrokontrolera.

Linker

Po wywołaniu opcji **Linker** zostaje wyświetlona ostatnio skonfigurowana mapa obszarów pamięci, w której zezwala się linkerowi umieszczanie segmentów relokowalnych programu - rysunek 3.10.



Rys. 3.10. Okno konfiguracji obszarów pamięci.

Segmenty absolutne są umieszczane zawsze w miejscu wskazanym przez treść programu. Mapa pamięci zapisana w postaci rekordów opisuje typ segmentu pamięci, adres początku obszaru oraz końca obszaru dostępnego dla linkera. Adresy są podawane w kodzie szesnastkowym.

Po naciśnięciu klawisza Enter lub Insert pojawi się okno edycji opisu obszaru pamięci - rysunek 3.11.

Rys. 3.11. Okno edycji obszaru pamięci.

Wyróżniono następujące typy segmentów pamięci:

BIT - pamięć obszaru danych adresowalna bitowo. O ile brakuje w trakcie łączenie programu pamięci w segmentach typu BIT, linker próbuje umieścić segmenty bitowe w obszarze pamięci DATA;

DATA - wewnętrzna pamięć danych mikrokontrolera;

CODE - pamięć programu mikrokontrolera. W tych obszarach linker umieści tylko segmenty CODE zakładając, że fizyczne pamięci programu i danych w tym zakresie są rozróżniane strobami dostępu (sygnał $\overline{PSEN}$ - dla pamięci CODE, sygnał $\overline{RD}$ - dla pamięci XDATA), a więc mogą się nakładać adresami bez utraty jednoznaczności.

XDATA - zewnętrzna pamięć danych mikrokontrolera. W tych obszarach linker umieści tylko segmenty XDATA, zakładając, że fizyczne pamięci programu i danych w tym zakresie są rozróżniane strobami dostępu (sygnał $\overline{PSEN}$ - dla pamięci CODE, sygnał $\overline{RD}$ - dla pamięci XDATA), a więc mogą się nakładać adresami bez utraty jednoznaczności.

COMMON CODE & DATA - obszar pamięci zewnętrznej, w którym mogą rezydować zarówno dane z segmentu XDATA jak i program z segmentu CODE. Linker zakłada, że w tym obszarze stroby

dostępu do pamięci XDATA oraz CODE są zsumowane, a więc nie można umieszczać danych i programu pod tym samym adresem bez utraty jednoznaczności.

Wyboru rekordu opisującego dany obszar pamięci dokonuje się za pomocą strzałek pionowych. Następnie klawiszem Delete można usunąć zaznaczony rekord, klawiszem Enter poprawić, klawiszem Insert wstawić nowy rekord opisujący obszar pamięci. Edycję w opcji **Linker** kończy się klawiszem Esc.

Należy pamiętać, że program zaraz po wykryciu sklejających lub nakładających się obszarów tego samego typu połączy je w jeden obszar.

Opcja **Memory From** służy do ustalenia adresu początku obszaru, natomiast opcja **To** do ustalenia adresu końca obszaru. Wybór opcji odbywa się za pomocą klawiszy kierunkowych (góra, dół) i naciśnięciu klawisza Enter lub przez użycie klawisza wyszczególnionej litery. Po wybraniu opcji ustawiania adresu można wpisać wartość adresu w postaci liczby hexadecymalnej.

Typ obszaru pamięci określa się jedną z opcji:

Data Segment - segment pamięci DATA;

Code - segment pamięci CODE (niedostępny dla danych);

External Data - segment pamięci XDATA (niedostępny dla programu);

CoMmon Code & External Data - obszar pamięci RAM wspólny dla programu i danych zewnętrznych;

Bit Data - segment pamięci BIT.

Pracę w oknie edycji opisu obszaru pamięci kończy wybór pola OK - zakończenie pozytywne lub klawiszem Esc - zaniechanie operacji.

Target

Pojawia się okno konfiguracji sprzętowej, w którym należy ustawić numer łącza szeregowego służącego do połączenie z systemem docelowym - rysunek 3.12.

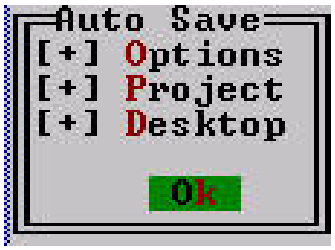
Rys. 3.12. Okno wyboru numeru łącza szeregowego.

Read

Wczytywanie zbioru konfiguracyjnego.

AutoSave

Opcja **AutoSave** określa, jakie elementy systemu będą zapamiętane automatycznie przy opuszczaniu IDE51, rysunek 3.13. Klawiszami kierunkowymi pionowymi wybiera się podopcje. Użycie klawisza Enter zmienia stan na przeciwny. Znak [+] oznacza zapamiętanie. Wyjście z opcji następuje poprzez wybór pola Ok i potwierdzenie przez Enter lub przez Esc i zaniechanie zmian.



Rys. 3.13. Okno wyboru opcji zapisu podczas zamykania systemu IDE51.

Save

Składowanie zbioru konfiguracyjnego.

3.3.7. Menu Debug (Alt-D)

Menu **Debug** służy do uruchamiania programu zawartego w aktywnym oknie edycyjnym pod kontrolą debugger'a - rysunek 3.14.

Oprócz typowych procedur pracy krokowej umożliwia bezpośrednie połączenie się z systemem docelowym w trybie emulacji terminala.

Rys. 3.14. Okno menu Debug.

Terminal (Alt-N)

W oknie TERMINAL emulowany jest terminal TTY komunikujący się z systemem docelowym. O ile aktualnie wykonywanym programem w systemie jest program MONITOR, to można wydawać wszystkie jego komendy. Można również komunikować się z innymi programami pracującymi w systemie docelowym. O ile zostanie w tym oknie wciśnięty klawisz F2, to zawartość zbioru zamiast być wczytana do okna edycji, zostanie przeniesiona do systemu docelowego, jako sekwencja rozkazów. Można w ten sposób załadować wprost program w postaci INTEL-HEX.

Chwilowo, w okno TERMINAL, jest wprowadzany także program IDE51 w trakcie wykonywania komend uruchamiających pisany program. Ma to na celu umożliwienie własnym programom komunikowanie się ze środowiskiem. W tym przypadku przesyłane są do systemu docelowego symbole wciśniętych klawiszy bez żadnej interpretacji. Praca w tym trybie kończy się z chwilą przejęcia sterowania przez program MONITOR.

Przy pracy krokowej może nastąpić zamazywanie wiersza ekranu po ostatnio otrzymanym znaku nowej linii. Jest to związane z działaniem procedur przechwytujących sygnały z systemu docelowego.

Step (F7)

Każde wywołanie rozkazu pracy krokowej powoduje wykonanie jednej instrukcji uruchamianego w systemie docelowym programu. Okna REGS,

WATCH oraz położenie kursora w programie źródłowym są na bieżąco uaktualniane.

Step Over Procedure (F8)

Każde wywołanie rozkazu pracy krokowej z pominięciem podprogramów powoduje wykonanie jednej procedury lub instrukcji programu uruchamianego, nie będącej procedurą. Okna REGS, WATCH oraz wskaźnik położenia w programie źródłowym są uaktualniane po wykonaniu instrukcji. W trakcie pracy debugger ustawia za wywołaniem procedury pułapkę programową. Z tego względu, odwołanie się procedury do 3 bajtu bezpośrednio za jej wywołaniem może spowodować całkowicie błędne działanie systemu. Zaleca się zatem, w przypadku wystąpienia takiej możliwości, na czas uruchamiania programu dodać 3 instrukcje puste, aby odsunąć ewentualny adres skoku.

Auto-Step (F6)

Wywołanie rozkazu pracy krokowej powoduje sekwencyjne wykonanie kolejnych instrukcji programu uruchamianego, jedna po drugiej. Okna REGS, WATCH oraz wskaźnik położenia w programie źródłowym są na bieżąco uaktualniane po wykonaniu każdej instrukcji.

Auto-StepOver (Ctrl-F6)

Program wykonuje kolejne procedury i uaktualnia wszystkie okna śledzenia wraz z położeniem wskaźnika rozkazów na tle zbioru źródłowego.

Wykonywanie opcji przerywa się przez wciśnięcie dowolnego klawisza.

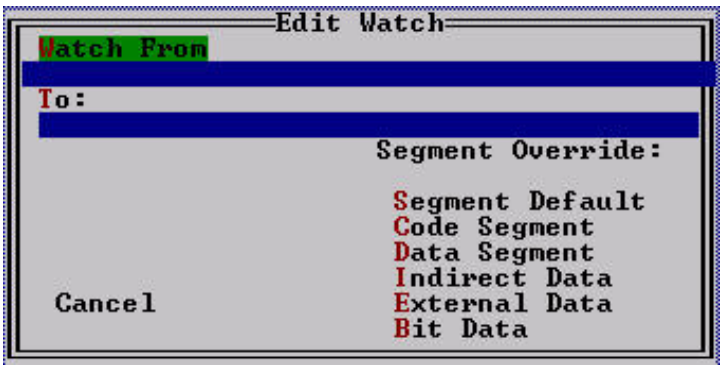
3.3.8. Menu Break/watch (Alt-B)

Menu **Break/watch** zawiera opcje umożliwiające przerywanie programu oraz śledzenie i modyfikację zawartości rejestrów i pamięci - rysunek 3.15.

Rys. 3.15. Okno menu Break/watch.

Set Watch (Ctrt-F7)

Po wywołaniu opcji **Set Watch** pojawia się okno edycji obszaru śledzenia pamięci, rysunek 3.16. Program automatycznie proponuje ustawienie początku obszaru śledzenia, na zmienną określoną słowem znajdującym się pod kursorem w oknie edycji. Z tego względu niedopuszczalne jest wywołanie opcji bez wcześniejszego otwarcia jakiegokolwiek okna edycji.



Rys. 3.16. Okno edycji obszaru śledzenia pamięci.

W oknie edycji obszaru śledzenia obowiązują następujące opcje:

Watch From - ustalenie początku obszaru śledzonego;

To - adresu końca obszaru śledzonego;

Segment Default - śledzenie w obszarze pamięci odbywa się zgodnie z typem wyrażenia w **Watch From** - jeżeli wyrażenie nie jest związane z określonym typem segmentu, to sygnalizowany jest błąd;

Code Segment - śledzenie w pamięci typu **CODE**;

Data Segment - śledzenie w pamięci typu **DATA**;

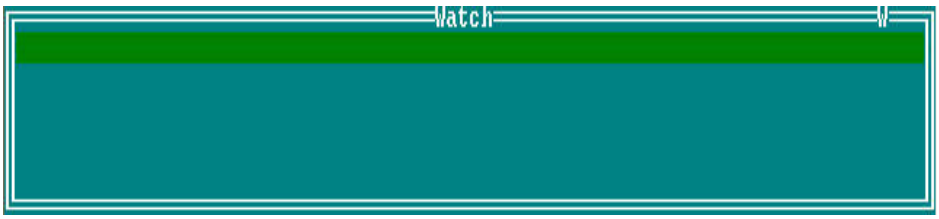
Indirect Data - śledzenie pamięci typu **IDATA**;

External Data - śledzenie pamięci typu **XDATA**;

Bit Data - śledzenie pamięci typu **BIT**.

Edit Watches

W oknie śledzenia pamięci wyświetlane są podglądane obszary pamięci - rysunek 3.17.



Rys. 3.17. Okno śledzenia pamięci.

Wewnątrz okna śledzenia pamięci można przesuwać linię kursorową, kasować zadane obszary śledzenia - klawisz Delete, definiować nowe obszary śledzenia - klawisz Insert oraz poprawiać parametry już zdefiniowanych obszarów śledzenia - klawisz Enter. Klawisze Insert i Enter działają analogicznie do funkcji Ctr-F7 z wyjątkiem tego, że w okienku edycji obszaru w przypadku klawisza Enter pojawiają się poprzednie parametry obszaru bezpośrednio po wywołaniu funkcji.

Toggle Break-Point (Ctr-F8)

Opcja pozwala zaznaczyć, a jeżeli już była zaznaczona, to anulować, daną linię w oknie edycji jako linię pułapki programowej. Program uruchamiany w trakcie realizacji przerwie pracę po osiągnięciu instrukcji zaznaczanej jako pułapka programowa. W trakcie zakładania pułapki debugger zamienia 3 bajty pamięci programu. Z tego powodu skok do dwóch bajtów, znajdujących się bezpośrednio za adresem pułapki, spowoduje nieobliczalne działanie systemu. Z tego względu pomiędzy przewidywanym adresem pułapki a najbliższym

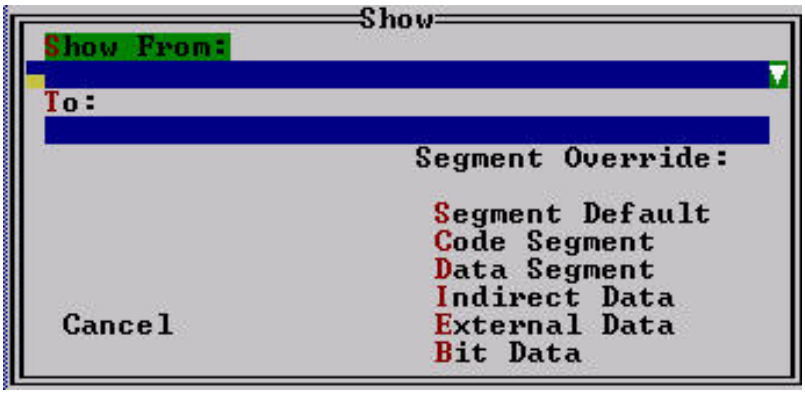
wejściem dla skoku należy, na czas uruchamiania, umieścić dwie instrukcje puste.

Clear Break- Points

Opcja powoduje usunięcie wszystkich znaczników pułapek programowych.

Modify Memory (Alt-M)

Po wybraniu opcji modyfikowania pamięci pojawia się okno z następującymi opcjami - rysunek 3.18.



Rys. 3.18. Okno modyfikacji obszaru pamięci.

Show From - adres początku obszaru modyfikacji;
To - adres końca obszaru modyfikacji;
Segment Default - modyfikacja odbywa się według typu adresu początku;
Code Segment - modyfikacja w pamięci typu **CODE**;
Data Segment - modyfikacja w pamięci typu **DATA**;
Indirect Data - modyfikacja w pamięci typu **IDATA**;
External Data - modyfikacja w pamięci typu **XDATA**;
Bit Data - modyfikacja w pamięci typu **BIT**;
Cancel - zaniechanie modyfikacji pamięci.

Po zakończeniu modyfikacji pamięci korygowana jest zawartość okna rejestrów oraz położenie wskaźnika w treści programu.

Po ustaleniu typu segmentu pojawia się okno modyfikacji pamięci, zawierające aktualne wartości pamięci.

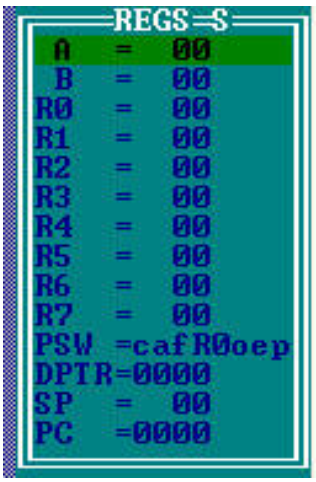
W oknie modyfikacji pamięci wyświetlany jest z lewej strony kod segmentu oraz adres pierwszej wyświetlanej komórki w tej linii. Za adresem wyświetlane są kolejne bajty pamięci (lub bity w przypadku segmentu bitowego) oddzielone spacjami. Wartości są wyświetlane w postaci hexadecymalnej. W obrębie zakresu modyfikowanej pamięci można poruszać się kursorami pionowymi i poziomymi. W miarę konieczności przeprowadzany jest automatycznie „scroll". Bajty zawierające zmodyfikowane wartości są podświetlane. Nowe wartości wprowadza się poprzez nadpisywanie nowej wartości w miejscu poprzedniej (tylko liczby hexadecymalne lub 0/1 w przypadku segmentu danych bitowych).

Wciśnięcie klawisza Enter powoduje przeniesienie zmodyfikowanych wartości do systemu docelowego oraz powrót do okna wyboru zakresu modyfikacji.

Wciśnięcie klawisza zaniechania Esc powoduje powrót do okna wyboru zakresu modyfikacji bez przesiania zmodyfikowanych wartości do systemu docelowego.

Registers (Alt-S)

Po otwarciu okna REGS można zmieniać zawartość wszystkich rejestrów mikrokontrolera oraz flag rejestru PSW - rysunek 3.19.



Rys. 3.19. Okno modyfikacji rejestrów mikrokontrolera.

Zawartość rejestru wyświetlana jest w postaci liczby hexadecymalnej. Rejestr PSW jest wyświetlany w postaci zakodowanej. Litery symbolizują poszczególne flagi w rejestrze, o ile flaga jest zapalona to wyświetlana jest duża litera, jeżeli natomiast jest zgaszona to wyświetlana jest mała litera. Na pozycji odpowiadającej bitom wyboru banku rejestrów wyświetlany jest numer wybranego banku poprzedzony literą R.

Wybór rejestru do modyfikacji dokonuje się poprzez użycie klawiszy kierunkowych pionowych. Wciśnięcie klawisza Enter powoduje wejście w tryb edycji zawartości rejestru. Po wyborze rejestru zmienia się kolor cyfr określających aktualną zawartość rejestru. Możemy zmienić zawartość rejestru poprzez nadpisywanie nowej wartości na starej (stosuje się zapis w kodzie hexadecymalnym). Zmianę cyfry poprawianej dokonujemy klawiszami kierunkowymi poziomymi. Zmiany rejestru można zaniechać przyciskiem Esc. Zmiana wartości rejestru w systemie docelowym jest dokonywana po wciśnięciu klawisza Enter lub klawisza kierunkowego pionowego.

W przypadku zmiany zawartości rejestru PC, korekcja wskaźnika położenia w treści programu jest nanoszona dopiero po wyjściu z okna REGS.

Rejestr PSW jest modyfikowany bezpośrednio przez ustawianie flag za pomocą odpowiednich liter:

C - Zmiana flagi przeniesienia CY;

A - Zmiana pomocniczej flagi AC;

O - Zmiana flagi nadmiaru OV;

F - Zmiana flagi ogólnego przeznaczenia FO;

E - Zmiana flagi ogólnego przeznaczenia Fl (PSW. l). Dostępna jest tylko w niektórych mikrokontrolerach;

P - Flaga parzystości.

Wyboru banku rejestrów dokonujemy poprzez wciśnięcie klawisza od 0 do 3 oznaczającego numer wybranego banku.

3.4 Edytor tekstowy

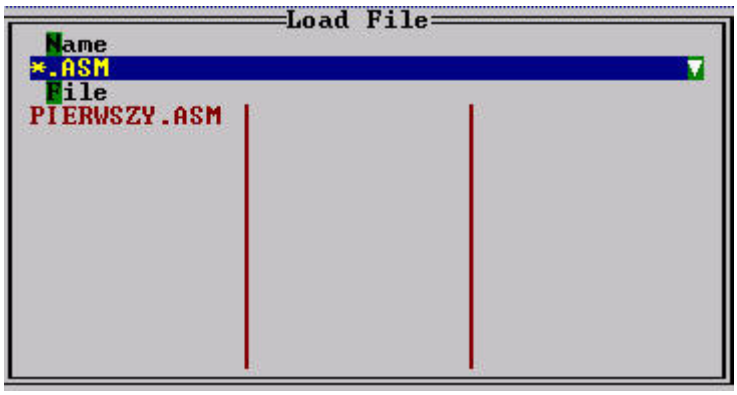
Integralną częścią systemu IDE51 jest edytor ekranowy, który pozwala na przygotowanie programu źródłowego przez użytkownika. W systemie IDE51 zaimplementowano edytor typu WORDSTAR. Oprócz standardowych funkcji edycyjnych edytor umożliwia wykonywanie operacji blokowych na tekście, wyszukiwanie i zamianę wybranej frazy itp. Wejście w tryb edycji może nastąpić albo przez otwarcie nowego okna edycyjnego lub poprzez uaktywnienie okna zamkniętego. Jednocześnie może być otwartych kilka okien edycyjnych, ale aktywnym jest tylko jedno. Zmiana aktywnego okna edycyjnego następuje poprzez użycie klawiszy Alt- numer okna.

Otwieranie nowego okna edycyjnego

Otwarcia nowego okna edycyjnego można dokonać na dwa sposoby:

- przez naciśnięcie klawisza F3 lub;
- przez wybranie opcji File/Load - rysunek 3.20.

W oknie znajdują się dwa pola: Name oraz File.



Rys. 3.20. Plansza otwierania nowego okna edycyjnego.

W polu Name pojawia się wzorzec wg którego wyświetlane są zbiory w polu File (standardowo *.ASM). Naciśnięcie klawisza Enter powoduje przejście do pola File. Użycie klawiszy kierunkowych pionowych rozwija

pole Name i pozwala wybrać inne wzorce (o ile istnieją). Użycie klawiszy znakowych kasuje bieżący wzorzec i pozwala wpisać nową nazwę. Jeśli dana nazwa jest nazwą zbioru (np. MOJ.ASM) to zostanie otwarte okno edycyjne. W przeciwnym przypadku nazwa będzie potraktowana jako „filtr" i w polu File zostaną wyświetlone odpowiadające zbiory. Za pomocą klawiszy kierunkowych pionowych można wybrać żądany zbiór i (naciskając klawisz Enter) otworzyć okno edycyjne. Poruszanie się między polami odbywa się za pomocą kombinacji klawiszy: Alt-N - pole Name, Alt-F - pole File.

Wprowadzanie, zmiana i usuwanie tekstu

Poniżej przedstawiono zestaw dyrektyw umożliwiających wykonanie podstawowych operacji na zbiorze tekstowym.

Operacje przesuwania:

Ctrl-E lub ↑ - przesunięcie kursora o jeden wiersz do góry;

Ctrl-X lub ↓ - przesuniecie kursora o jeden wiersz do dołu;

Ctrl-S lub ← - przesunięcie kursora o jedną kolumnę w lewo;

Ctrl-D lub → - przesunięcie kursora o jedną kolumnę w prawo;

Ctrl - → - przesunięcie kursora o słowo w prawo;

Ctrl - ← - przesunięcie kursora o słowo w lewo;

Ctrl-A - przesunięcie kursora do początku słowa w lewo;

Ctrl-F - przesunięcie kursora do początku słowa w prawo;

Home - przesunięcie kursora do początku linii;

End - przesunięcie kursora na koniec linii;

Ctrl-Home - przesunięcie kursora do początku tekstu;

Ctrl-End - przesunięcie kursora do końca tekstu;

Ctrl-R lub PgUp - przesunięcie tekstu o jedną stronę do góry;

Ctrl-C lub PgDn - przesunięcie tekstu o jedną stronę w dół;

Ctrl-W - przesunięcie tekstu o jeden wiersz w dół;

Ctrl-Z - przesunięcie tekstu o jeden wiersz do góry;

Ctrl-Q B - przesunięcie kursora do początku bloku;

Ctrl-Q K - przesunięcie kursora do końca bloku.

Operacje kasowania:

Backspace - kasowanie znaku przed kursorem;

Del - kasowanie znaku pod kursorem;

Ctrl-T - usunięcie znaku i słowa znajdującego się pod kursorem;

Ctrl-Y - usunięcie wiersza, w którym znajduje się kursor;

Ctrl-Q Y - kasowanie tekstu do końca wiersza;

Ctrl-K Y - kasowanie bloku tekstu.

Operacje wstawiania:

Enter - wstawienie nowej linii

Złożone operacje na tekście

Poniżej opisane są operacje dotyczące większych fragmentów tekstu, takie jak np.: przestawianie lub kopiowanie grup sąsiadujących wierszy, wczytywanie zbioru lub zapis do zbioru zaznaczonego bloku albo wyszukiwanie lub zmiana określonych słów i fraz:

Ctrl-Q F - poszukiwanie zadanej frazy w tekście;

Ctrl-Q A - poszukiwanie zadanej frazy i zamiana na zadaną frazę;

Ctrl-L - powtórzenie ostatniej operacji szukaj/zamień;

Ctrl-Q B - przesuń kursor do znacznika początku bloku;

Ctrl-Q K - przesuń kursor do znacznika końca bloku;

Ctrl-K B - oznaczenie początku bloku;

Ctrl-K K - oznaczenie końca bloku;

Ctrl-K T - zaznacz słowo jako blok;

Ctrl-K H - usunięcie wyróżnienia słowa;

Ctrl-K Y - usunięcie wyróżnionego bloku;

Ctrl-K C - wykonanie dyrektywy powoduje skopiowanie bloku wyróżnionego przez dyrektywy Ctrl-K B i Ctrl-K K na pozycję przed znak wyróżniony przez kursor;

Ctrl-K R - wykonanie dyrektywy powoduje otwarcie okna dialogowego (podobnie jak w opcji Load/File), które umożliwia wybór zbioru wczytywanego w miejsce kursora;

Ctrl-K W - działanie dyrektywy powoduje otwarcie okna dialogowego (podobnie jak w opcji Load/File), które umożliwia podanie nazwy zbioru, gdzie będzie przechowywany zaznaczony w oknie edycyjnym blok;

Ctrl-K V - wykonanie dyrektywy powoduje przeniesienie bloku wyróżnionego przez dyrektywy Ctrl-K B i Ctrl-K K na pozycję przed znak wyróżniony przez kursor;

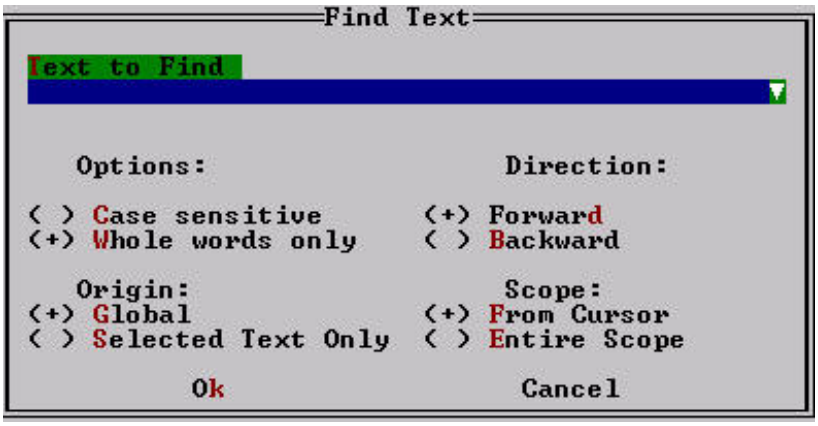
Alt-K C - wykonanie dyrektywy powoduje skopiowanie bloku wyróżnionego przez dyrektywy Ctrl-K B i Ctrl-K K do bufora;

Alt-K V - wykonanie dyrektywy powoduje przeniesienie bloku wyróżnionego przez dyrektywy Ctrl-K B i Ctrl-K K do bufora;

Alt-K P - wykonanie dyrektywy powoduje, wstawienie zawartości bufora na pozycję przed znak wyróżniony przez kursor.

Operacje wyszukiwania/zamiany

W przypadku wywołania funkcji szukania lub zamiany łańcucha znaków pojawia się okno dialogowe służące do ustawienia opcji przeszukiwania - rysunek 3.21.



Rys. 3.21. Okno ustawiania opcji wyszukiwania.

Znaczenie opcji:

Text to Find - tekst do znalezienia - pojawia się okno edycji łańcucha znaków, w którym definiuje się tekst do znalezienia;

Options:

Case sensitive - jeśli opcja jest włączona to rozróżniane będą duże i małe litery w trakcie szukania;

Whole words only - jeśli opcja została wybrana, to przy szukaniu porównywane są tylko całe słowa;

Direction:

Forward - poszukiwanie prowadzone będzie w kierunku do końca zbioru;

Backward - poszukiwanie prowadzone będzie w kierunku do początku zbioru;

Origin:

Global - poszukiwanie prowadzone będzie w całym zbiorze;

Selected Text Only - poszukiwanie prowadzone będzie tylko w zaznaczonym bloku;

Scope:

From Cusor - poszukiwanie prowadzone będzie od położenia kursora;

Entire Scope - poszukiwanie prowadzone będzie od początku zbioru;

Ok - wykonaj pojedynczą operację szukania;

Przycisk Cancel powoduje zaniechanie operacji.

3.5. Asembler

3.5.1. Wprowadzenie

Jedną z podstawowych funkcji systemu IDE51 jest przetwarzanie zbioru źródłowego na zbiór wynikowy. Pierwszy etap przetwarzania realizuje tzw. asembler, którego zadaniem jest analiza składni zbioru tekstowego i wyprodukowanie tzw. zbioru pośredniego (*object*), który następnie przekształcany jest w zbiór wynikowy za pomocą programu łączącego (*linkera*).

Efektem kompilacji są trzy zbiory o rozszerzeniach: ***.obj**, ***.hex** oraz ***.map.** W przypadku uaktywnienia funkcji **[+]Create PRN** (menu **Compile**) - rysunek 3.6, dodatkowo jest tworzony zbiór z rozszerzeniem ***.prn.**

W systemie IDE51 występuje asembler, który posiada następujące właściwości:

- procesor makrorozkazów;
- kompilacja warunkowa;
- predefiniowane symbole dla pochodnych z rodziny ’51;
- kompilacja programów wielomodułowych;
- sygnalizacja błędów kompilacji w zbiorze źródłowym.

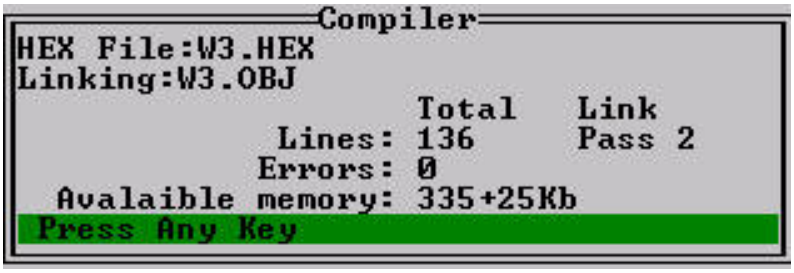
Z asemblerem związana jest opcja Compile w menu głównym. Zawiera ona trzy podopcje: Compile/Compile, Compile/Make i Compile/Build.

Podopcja Create PRN dotyczy tworzenia zbioru typu *.PRN. W systemie IDE51 istnieje możliwość kompilacji bez linkowania lub z linkowaniem.

Kompilację bez linkowania wybiera się podopcją Compile/Compile (Alt-F9). Kompilacja z linkowaniem dotyczy podopcji Compile/Make (F9) i Compile/Build.

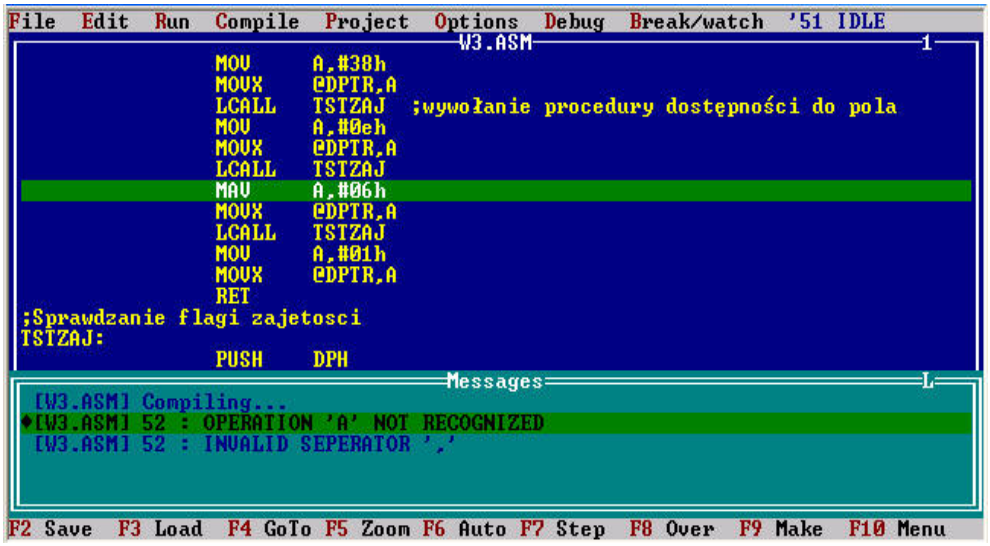
Podczas kompilacji na środku ekranu wyświetla się okno rysunek 3.22, które informuje o efektach kompilacji i linkowania, ilości linii programu źródłowego, które już zostały skompilowane, braku błędów, dostępnej pamięci operacyjnej komputera. W przypadku uzyskania bezbłędnej kompilacji nastę-

puje proces linkowania, co w końcu doprowadza do wygenerowania zbioru wynikowego.



Rys. 3.22. Okno informujące o efektach kompilacji i linkowania.

Gdy pojawią się jakiekolwiek błędy kompilacji nie dochodzi do wygenerowania zbioru wynikowego. W tej sytuacji otwiera się okno MESSAGES, a w nim numery linii w programie źródłowym gdzie wystąpił błąd oraz krótki opis błędu. Sytuację taką ilustruje - rysunek 3.23.



Rys. 3.23. Okno edycyjne i MESSAGES z komunikatami o błędach.

Za pomocą klawiszy kierunkowych (góra, dół) można przejrzeć listę błędów. Jednocześnie z przeglądaniem listy błędów podświetla się odpowiednia linia programu źródłowego, gdzie wystąpił błąd. Z okna MESSAGES przechodzi się bezpośrednio do okna edycji poprzez naciśnięcie klawisza Enter, dzięki czemu możliwe jest natychmiastowe usunięcie błędu. W oknie edycyjnym istnieje możliwość przeglądania listy błędów wraz z podświetlaniem od-

powiedniego wiersza przy użyciu klawiszy Alt-F8 (przeglądanie do przodu) oraz Alt-F7 (przeglądanie do tyłu).

Po usunięciu błędów należy przeprowadzić kompilację od nowa. Poszczególne błędy są wyszczególniane również w zbiorze ***.prn,** który zawiera kolejne adresy poszczególnych rozkazów programu wraz z kodami tych rozkazów.

Asembler akceptuje predefiniowane symbole rejestrów dla następujących mikrokontrolerów z rodziny ‘51:

- INTEL 8031;
- INTEL 8032;
- SIEMENS 80C535;
- SIEMENS 80C537;
- PHILIPS 83C552.

Wybór typu mikrokontrolera następuje w opcji Option/Compiler - rysunek 3.9. Zaznaczenie opcji [+] None umożliwia kompilację bez rozpoznawania jakichkolwiek symboli rejestrów. Ponadto istnieje możliwość rozpoznawania symboli rejestrów innych mikrokontrolera z rodziny ‘51 poprzez użycie dyrektywy **INCLUDE Nazwa_zbioru**. Zbiór ze zdefiniowanymi symbolami będzie wczytywany w czasie kompilacji.

3.5.2. Argumenty, adresy, wyrażenia

Uwaga: *Asembler nie rozróżnia pomiędzy literami małymi i dużymi.*

I. Liczby

Przyjmowane są tylko liczby całkowite:

xxxxd lub xxxx gdzie x = 0 ÷ 9 dziesiętne, np. 1234 lub 1234d

xxxxh gdzie x = 0 ÷ 9,a,b,c,d,e,f szesnastkowe, np. 25h, 0a1h

xxxxb gdzie x = 0 lub 1 binarne, np. 01110110b

Pierwsza cyfra musi być zawsze z zakresu 0 ÷ 9. Jeżeli w linii instrukcji liczba oznacza wartość, to musi ona być poprzedzona znakiem „#”, w prze-

ciwnym razie zostanie przyjęta przez asembler jako bezpośredni adres rejestru.

Przykłady:

MOV A,0fh wprowadzenie do akumulatora zawartości rejestru o adresie 0fh;

MOV A,#0fh wprowadzenie do akumulatora liczby hexadecymalnej 0fh;

MOVA,#10101010b wprowadzenie do akumulatora liczby binarnej 10101010.

II. Łańcuchy znakowe

Łańcuchy znaków są to ciągi znaków umieszczone pomiędzy znakami apostrofu np. ‘x’, ‘xx’, ‘xxx’. Do programu asembler wstawia odpowiednie kody ASCII ciągu znaków, zawartych pomiędzy znakami apostrof. Łańcuchy mogą też być używane w dyrektywach **DB i DW**.

Przykłady:

```
tekst: DB   'To jest przykład'
tekst1:DW   'itd',12
```

III. Etykiety

Poszczególnym punktom programu, podprogramom, predefiniowanym stałym, rejestrom można przypisać **etykietę.** Etykietą jest unikalną, nie będącą słowem kluczowym, nazwą, przy czym pierwszym znakiem musi być **litera**. Etykietę należy umieszczać w pierwszej kolumnie linii tekstu programu (bez spacji). Wskazane jest (choć nie jest konieczne) umieszczanie na końcu etykiety znaku dwukropka („**:**”) - poprawia to czytelność programu.

Przykłady:

```
tekst: DB 10,10101100B,03DH,250
dane: DS 4
skok: LCALL skok1
```

Dana etykieta może być zdefiniowana w programie tylko jeden raz, natomiast ilość odwołań do niej nie jest ograniczona. Gdy użytkownik pragnie skierować wykonanie programu w miejsce oznaczone etykietą, posługuje się jej nazwą a nie adresem fizycznym.

IV. Symbole specjalne

Niektóre predefiniowane symbole (w tym nazwy rejestrów specjalnych) są słowami kluczowymi i nie mogą być używane jako etykiety, np.: **A, ACC, PC, DPTR, C, CY, AB, R0..R7, TCON, TMOD.**

Specjalne znaczenie ma np. znak: **$**, który oznacza skok do adresu aktualnie wykonywanej instrukcji. Np. instrukcja: **DJNZ R0,$** będzie powtarzana tak długo, dopóki zawartość rejestru R0 nie osiągnie wartości równej zero.

Znak średnika „;” oznacza początek komentarza, tzn. dalszy ciąg linii nie jest przez asembler analizowany. Znak ten może także być używany np. do czasowego wyłączania poszczególnych linii programu - jest to często niezbędne na etapie wstępnego uruchamiana programu.

3.5.3. Operatory

W języku asemblera do zapisywania wyrażeń stosuje się dwa rodzaje operatorów:

Arytmetyczne: + (dodawania), - (odejmowania), * (mnożenia), / (dzielenia), **MOD** (dzielenia modulo).

Binarne:

high, low	(wydzielenie starszego i młodszego bajtu adresu lub liczby;
and	logiczny **AND**;
or	logiczny **OR**;
xor	logiczny **XOR**;
not	logiczny **NOT.**

Przykład:

MOV TL0,#low(4095) do TL0 zostanie wpisany młodszy bajt liczby 4095 (czyli 0ffh);

MOV TH0,#high(4095) do TH0 zostanie wpisany starszy bajt liczby 4095 (czyli 0fh);

MOV TL1,#low(not(4095) do TL1 zostanie wpisany zanegowany młodszy bajt liczby 4095 (czyli 00h);

MOV TH1,#high(not(4095)) do TH1 zostanie wpisany zanegowany starszy bajt liczby 4095 (czyli 0f0h).

3.5.4. Dyrektywy asemblera

Dyrektywy języka asemblera umożliwiają:

Przypisanie nazwie wartości:

- **EQU**

Przykład:

stała EQU 31h przypisanie liczby 31h nazwie *stała*;

- **SET**

Przykład:

```
dana SET 10h
```

(linie programu)

dana SET 20h przypisanie nazwie *dana* kolejno dwóch, różnych wartości;

- **DATA**

Przykład:

ilość DATA 21h przypisanie nazwie *ilość* adresu komórki pamięci z **wewnętrznego** obszaru pamięci danych adresowanej bezpośrednio 00-0ffh;

- **IDATA**

Przykład:

`dane IDATA 81h` przypisanie nazwie *dane* adresu komórki pamięci z **wewnętrznego** obszaru pamięci danych adresowanej pośrednio 00-0ffh;

- **XDATA**

Przykład:

`lcd XDATA 1fffh` przypisanie nazwie *lcd* adresu komórki pamięci z **zewnętrznego** obszaru pamięci danych 0000-0ffffh;

- **CODE**

Przykład:

`start CODE 0100h` przypisanie nazwie *start* adresu komórki pamięci z obszaru **pamięci programu** 0000-0fffh;

- **BIT**

Przykład:

`stop BIT 00` przypisanie nazwie *stop* adresu bitu z obszaru **wewnętrznej** pamięci, adresowanej bitowo: bity 00-7fh w obszarze pamięci 20h-30h oraz bity rejestrów specjalnych w obszarze 80h-0ffh.

Zamiast dyrektyw **DATA, IDATA, XDATA, CODE, BIT** można oczywiście użyć dyrektywy **EQU.** Jednak używanie wcześniej podanych dyrektyw poprawia czytelność programu (podobnie jak stosowanie komentarzy w programie).

Inne

- **DB**

Przykład:

`tab: DB 10,1fh,10101010b,'ABC'` umieszcza kolejno w pamięci programu bajty wyrażeń;

- **DW**

Przykład:

`tab1: DW 1234,1f1fh,'abcd'` umieszcza w pamięci programu kolejno podane słowa (po 2 bajty) wyrażenia;

- **ORG**

Przykład:

`ORG 1000h` instrukcje są umieszczane w pamięci programu kolejno począwszy od adresu podanego dyrektywą **ORG.**

- **END** linie programu po tej dyrektywie nie są przez kompilator analizowane.

3.6. Linker

System IDE51 zawiera linker tj. program, który wykonuje następujące funkcje:

- łączenie kilku modułów w jeden;
- przydzielanie obszarów pamięci dla poszczególnych segmentów;
- definiowanie adresów absolutnych dla segmentów relokowalnych;
- przypisywanie symboli zewnętrznych do symboli typu PUBLIC;
- tworzenie zbioru typu *.HEX.

W trakcie łączenia modułów linker najpierw określa dostępną pamięć dla modułów relokowalnych na podstawie mapki pamięci deklarowanej w oknie Options/Linker - rysunek 3.8. Adresy obszarów relokowalnych w tym oknie zostają przydzielone tylko wewnątrz tych obszarów, natomiast segmenty absolutne zostaną umieszczone według wskazań określonych w treści programu źródłowego.

Przy przydziale pamięci stosowana jest następująca kolejność:

1. Segmenty absolutne - rezerwowana jest pamięć od adresu początku segmentu (opcja AT deklaracji segmentu absolutnego w programie źródłowym) do końca zajętego obszaru.
2. Zadeklarowane banki rejestrów danych (USING).
3. Segmenty DATA zadeklarowane jako BIT-ADDRESSABLE.
4. Segmenty BIT.
5. Segmenty DATA.
6. Segmenty IDATA.
7. Segmenty CODE.
8. Segmenty XDATA.

W obrębie jednego segmentu kolejność umieszczania jest zgodna z kolejnością występowania modułów w oknie PROJECT.

W przypadku braku wolnego miejsca w obszarze pamięci relokowalnej sygnalizowany jest błąd. Linker tworzy zbiór typu *.HEX zawierający program wynikowy w standardzie INTEL-HEX oraz zbiór typu *.MAP, zawierający adresy segmentów w kolejnych modułach oraz wartości nazw globalnych. Zbiór *.MAP jest w postaci znakowej. Jest on wykorzystywany przez debugger do tworzenia łączników pomiędzy realizowanym zadaniem a postacią źródłową programu.

Dodatkowo linker określa adres instrukcji, od której zostanie rozpoczęte wykonywanie programu w trakcie jego realizacji. Adres ten jest określony jako najmniejszy adres rezerwowany przez segmenty typu CODE.

Praca z projektem

Złożone programy wygodnie jest podzielić na kilka modułów zawartych w osobnych zbiorach (komunikację między modułami zapewniają dyrektywy PUBLIC / EXTERN). W takim przypadku należy utworzyć projekt. Zaleca się utworzenie oddzielnego katalogu dla każdego projektu np.: ...\PROJEKT, gdzie będą przechowywane wszystkie zbiory związane z bieżącym projektem.

Otwieranie projektu

Informacje dotyczące nazw modułów wchodzących w skład projektu zawarte są w zbiorze z rozszerzeniem *.PRJ. Wczytanie zbioru typu *.PRJ następuje po wybraniu opcji Project/Open – rysunek 3.7. Wybór zbioru jest analogiczny do sposobu wykorzystywanego w opcji File/Load. W przypadku otwierania nowego zbioru należy podać jego nazwę z rozszerzeniem *.PRJ. Otwarcie okna PROJECT umożliwia wybranie zbiorów tworzących projekt - klawisz Insert.

Zamykanie projektu

Zakończenie pracy z projektem następuje po wybraniu opcji Project/Close - rysunek 3.7. Zamknięcie projektu konieczne jest w przypadku pracy z programem jednomodułowym. W przeciwnym razie nie będzie możliwa kompilacja programu jednomodułowego.

3.7. Program MONITOR - 51

3.7.1. Wprowadzenie

MONITOR - 51 umożliwia uruchamianie skompilowanego programu źródłowego, napisanego w języka asemblera dla mikrokontrolerów z rodziny ’51 wybranych w opcji Options/Compiler – rysunek 3.9. Wykorzystanie tej opcji jest możliwe jedynie w przypadku podłączenia do komputera za pomocą złącza RS-232 płytki z systemem zbudowanym w oparciu o mikrokontroler z rodziny ‘51. Ponadto system uruchamiany musi być wyposażony w program MONITOR - 51.

Program MONITOR-51 umożliwia:

- przyjęcie przesyłanego z komputera PC (łączem RS-232) za pomocą programu IDE51 (polecenie Ctrl-F2) kodu wykonywalnego i umieszczenie go w pamięci RAM pod wskazanym adresem;
- uruchomienie z komputera (polecenie Ctrl-F9) programu umieszczonego w pamięci RAM;
- pracę krokową (polecenia F6, F7, F8) programu;

- start i zatrzymanie programu pod dowolnym adresem (adresy poszczególnych rozkazów są dostępne w pliku ***.prn**);
- odczyt oraz zmianę zawartości wszystkich komórek pamięci - z wyjątkiem komórek umieszczonych w obszarze pamięci programu monitora (jest możliwy tylko odczyt zawartości komórek pamięci programu MONITOR-51).

3.7.2. Ładowanie programu do systemu docelowego

Przed przystąpieniem do wykorzystania MONITORA-51 należy upewnić się czy komputer został połączony z systemem docelowym poprzez łącze RS-232. Następnie należy załadować uprzednio skompilowany program w oknie edycyjnym do systemu docelowego. W tym celu można użyć kombinacji klawiszy Ctrl-F2 lub z menu głównego uaktywnić opcję Run/Reset & Load - rysunek 3.5. O ile program nie został jeszcze skompilowany lub zlinkowany to zostanie skompilowany i zlinkowany. Następnie zostanie otwarte (o ile czynność jest wykonywana po raz pierwszy) lub uaktywnione okno TERMINAL oraz wykonane programowe zerowanie procesora i ładowanie programu do systemu docelowego. Rejestr PC będzie ustawiony na najniższy adres segmentu CODE załadowanego programu.

3.7.3. Uruchamianie programu w systemie docelowym

Po załadowaniu programu do systemu docelowego kursor liniowy ustawia się na pierwszej wykonywanej linii programu źródłowego. W tym momencie można uruchomić program z pełną szybkością poprzez naciśnięcie kombinacji klawiszy Ctrl-F9 lub w jednym z trybów pracy krokowej. W przypadku uruchomienia z pełną szybkością program MONITOR-51 przekazuje sterowanie do programu użytkownika i traci kontrolę nad wykonywanym programem.

W tym trybie nie można obserwować zmiennych w oknie WATCH. Wykonywanie programu użytkownika pod kontrolą IDE51 dokonuje się poprzez wybór trybu pracy krokowej:

- F7 - zatrzymanie po każdej rozkazie;
- F8 - zatrzymanie po każdej procedurze.

Po naciśnięciu klawisza F7 (F8) następuje wykonanie rozkazu (procedury) i przesunięcie kursora liniowego do następnego rozkazu.

Ponadto możliwe jest uruchamianie programu w trybie z ograniczoną prędkością tzn. cykliczne wykonywanie pracy krokowej:

- F6 - praca z ograniczoną prędkością;
- Ctrl-F6 - praca z ograniczoną prędkością z omijaniem procedur.

Zatrzymanie trybu pracy z ograniczoną prędkością następuje poprzez naciśnięcie dowolnego klawisza. Kursor liniowy zatrzymuje się po ostatnio wykonywanym rozkazie (procedurze). Wszystkie wymienione tryby pracy (F7, F8, F6, Ctrl-F6) odświeżają okno REGS oraz WATCH, co umożliwia śledzenie zawartości rejestrów procesora oraz zmiennych zdefiniowanych w oknie WATCH.

W oknie TERMINAL pojawiają się komunikaty wysyłane przez program uruchamiany oraz w przypadku, gdy program uruchamiany jest zamrożony. Komunikaty debuggera umieszczonego w pamięci FLASH. W przypadku, gdy okno TERMINAL zostało zainicjowane przez uruchomienie programu (Ctrl-F9), to dalsza praca z debuggerem jest możliwa tylko po zawieszeniu pracy programu uruchamianego (na pułapce lub po zerowaniu mikroprocesora). W tym czasie wszelkie znaki z klawiatury są przesyłane bezpośrednio do systemu docelowego bez analizy.

Wydanie komendy zmiany okna powoduje odświeżenie okna REGS, okna WATCH oraz wskaźnika instrukcji wykonywanej w programie źródłowym.

UWAGA: program IDE51 nie interpretuje komunikatów w oknie TERMINAL i zakłada, że informacje zawarte w zbiorach *.OBJ i *.MAP są ciągle aktualne.

3.7.4. Zakładanie pułapek programowych

W przypadku uruchamiania większych programów bardzo pomocne mogą okazać się tzw. pułapki programowe. Umożliwiają one zatrzymanie (uruchomionego poprzez opcję Ctrl-F9) programu na wybranym rozkazie.

Ustawianie pułapek programowych dokonuje się w aktualnie aktywnym oknie edycyjnym. Kursor należy umieścić w odpowiedniej linii programu źródłowego - punkcie zatrzymania i nacisnąć Ctrl-F6. Ustawienie pułapki programowej sygnalizowane jest podświetleniem wybranej linii programu. Kasowanie pułapki wykonuje się przez ustawienie w linii rozkazu kursora znakowego i naciśnięcie klawiszy Ctrl-F8. Ze względu na realizację pułapki programowej poprzez wstawienie rozkazu LCALL istnieje możliwość zniekształcenia programu użytkownika, co w konsekwencji może doprowadzić do błędnej pracy programu. Rozkaz LCALL zajmuje trzy bajty i dlatego nie można użyć dwóch pułapek w sekwencji jedna za drugą. Drugi przykład ograniczenia przedstawiono poniżej:

```
BEG:    ACALL SER          fragment programu PROGRAM.ASM
        ACALL DELAY
        SJMP  BEG
        N O P              rozkaz pusty
DELAY:  MOV   R1,#12
DEL1:   MOV   R2,#0FFH
```

Wstawienie pułapki programowej na pozycji rozkazu SJMP BEG w przypadku braku rozkazu NOP spowoduje, że program zatrzyma się na pułapce, ale dalsze działanie będzie błędne. Wstawienie rozkazu NOP umożliwi dalsze wykonywanie programu.

3.7.5. Obserwacja zmiennych programowych i obszarów pamięci

System IDE51 posiada możliwość deklaracji zmiennych programowych, obszarów pamięci lub rejestrów, których zawartość będzie obserwowana w specjalnie do tego celu przeznaczonym oknie WATCH. Wprowadzenie symbolu lub wprost wartości liczbowej do okna WATCH dokonuje się poprzez użycie kombinacji klawiszy Ctrl-F7 (z poziomu okna edycyjnego) lub z menu głównego poprzez wybór opcji Break/Watch/Set Watch - rysunek 3.15. Zmienne, rejestry, obszary pamięci mogą być przedstawiane w postaci bajtowej lub bitowej o ile zawarte są w obszarze danych bitowych.

Przykład:

P3 - obserwacja bajtu rejestru;

P3.1 - obserwacja bitu rejestru;

P3.0.. P3.7 - obserwacja bajtu w postaci bitowej.

Po zadeklarowaniu symbolu lub wartości liczbowej należy przypisać segment pamięci, w którym następuje obserwacja. Jeśli zmienną podano w sposób symboliczny to program sam rozpoznaje typ segmentu o ile zmienna jest zdefiniowana.

3.7.6. Modyfikacja zawartości rejestrów i pamięci

Oprócz możliwości obserwacji MONITOR - 51 posiada również możliwość zmiany zawartości zmiennych, rejestrów lub pamięci. Modyfikacja wybranych, najważniejszych rejestrów procesora, odbywa się w oknie REGS, do którego przechodzi się używając klawiszy Alt-S lub z menu głównego poprzez wybór opcji Break/Watch/Registers - rysunek 3.15.

Modyfikacja pozostałych rejestrów, zmiennych lub obszarów pamięci dokonuje się poprzez użycie klawiszy Alt-M lub z menu głównego poprzez wybór opcji Break/Watch/Modify Memory - rysunek 3.15.

4. Ćwiczenia laboratoryjne

4.1. Cele kształcenia

Zajęcia laboratoryjne mają na celu zdobycie praktycznych umiejętności programowania mikrokontrolera z rodziny ’51 (SIEMENS SAB 80C517/80C537) i testowanie oprogramowania, w nakreślonym w kursie, zakresie oraz praktyczne wykorzystanie mikrokontrolera (sprzęt i oprogramowanie) do sterowania obiektami, z którymi współpracują mikrokontrolery rodziny ’51.

Ilustracja praktycznych aspektów projektowania oprogramowania i programowania mikrokontrolerów w typowych, reprezentatywnych zastosowaniach związanych z aparaturą kontrolno-pomiarową i sterownikami przemysłowymi.

Oczekuje się, że uczestnictwo w zajęciach umożliwi studentowi uzyskanie praktycznych umiejętności obsługi podstawowych narzędzi wspomagających projektowanie i uruchamianie oprogramowania dla systemów mikroprocesorowych.

4.2. Charakterystyka ćwiczeń

Treści programowe

W pierwszej fazie zajęcia mają na celu rozwijanie umiejętności projektowania programów pisanych w języku asemblera. Obejmują one konstrukcje typowych procedur arytmetycznych i sterujących: dodawanie, mnożenie i dzielenie jednobajtowych i wielobajtowych argumentów, kopiowanie da-

nych pomiędzy różnymi obszarami i typami pamięci, odmierzanie zadanych interwałów czasowych, konwersję kodów.

W drugiej fazie zajęć przewidziano projekty programów wykorzystujących wszystkie elementy zawarte w strukturze wewnętrznej mikrokontrolera oraz sterowanie elementami zewnętrznymi: klawiaturą (statyczną i dynamiczną), wyświetlaczami LED i alfanumerycznymi LCD, sygnalizatorami, oraz układem transmisji szeregowej.

4.2.1. Wprowadzenie do ćwiczeń laboratoryjnych, wymagania, literatura, zasady BHP

Omówienie spraw organizacyjnych związanych z wykonaniem i zaliczeniem ćwiczeń laboratoryjnych: podział na grupy i podgrupy, zapoznanie z *Regulaminem Laboratorium Systemów Mikroprocesorowych oraz Przepisami BHP obowiązującymi w laboratorium*, przedstawienie tematyki i zasad zaliczenia ćwiczeń laboratoryjnych, zapoznanie z literaturą podstawową i uzupełniającą.

4.2.2. Moduł dydaktyczny μM-DYD

Cel dydaktyczny ćwiczenia:

Celem ćwiczenia jest zapoznanie się z strukturą sprzętową modułu dydaktycznego μM-DYD oraz makiet z nim współpracujących.

Przedmiot ćwiczenia :

- moduł sterownika μM-537F;
- pole odczytowe do wyświetlania komunikatów i informacji o stanie systemu;
- wieloelementowa klawiatura;
- wyświetlacze dwustanowe;
- łącze szeregowe w standardzie RS-232;

- piezoelement (buzzer) emitujący dźwięki o programowanej częstotliwości;
- makieta dydaktyczna „Wyświetlacz LED”;
- makieta dydaktyczna „Skrzyżowanie”;
- makieta dydaktyczna „Tęcza”.

Program ćwiczenia obejmuje:

- ♦ dokładne rozpoznanie umiejscowienia poszczególnych zespołów funkcjonalnych na płycie modułu dydaktycznego i płytce sterownika;
- ♦ zapoznanie się z strukturą i wyprowadzeniami we-wy wybranych zespołów funkcjonalnych, wskazanych przez prowadzącego zajęcia;
- ♦ uruchamianie przykładowych programów, wskazanych przez prowadzącego zajęcia, ilustrujących działanie wybranych podzespołów funkcjonalnych modułu dydaktycznego (programy znajdują się na dysku C komputera w katalogu C:/IDE51/DEMO).

Zakres materiału obowiązujący studenta:

Studenci winni zapoznać się z opracowaniem „Laboratorium Systemów Mikroprocesorowych” ze szczególnym zwróceniem uwagi na rozdział 2 - Opis sprzętu - Modułowy system µM-DYD.

Wymagana jest znajomość sposobu zasilania i połączenia modułu z komputerem PC oraz umiejętność scharakteryzowania podstawowych elementów struktury wewnętrznej modułu dydaktycznego.

Ponadto studenci powinni znać przeznaczenie oraz sposób wykorzystania makiet dydaktycznych podłączanych do modułu dydaktycznego.

4.2.3. System programów IDE51

Cel dydaktyczny ćwiczenia:

Celem ćwiczenia jest zapoznanie się z programami narzędziowymi systemu IDE51 wspomagającymi projektowanie i uruchamianie oprogramowania dla systemów mikroprocesorowych.

Przedmiot ćwiczenia :

- edytor tekstowy;
- asembler (program tłumaczący);
- program MONITOR -51.

Program ćwiczenia obejmuje:

♦ szczegółowe zapoznanie się z strukturą i możliwościami funkcjonalnymi systemu IDE51;

♦ edycję i kompilację przykładowych programów wskazanych przez prowadzącego ćwiczenia;

♦ szczegółowe zapoznanie się z zleceniami programu MONITOR-51;

♦ uruchamianie i wykonywanie skompilowanych przykładowych programów (programy znajdują się na dysku C komputera w katalogu C:/IDE51/DEMO).

Zakres materiału obowiązujący studenta:

Studenci winni zapoznać się z opracowaniem „Laboratorium Systemów Mikroprocesorowych" ze szczególnym zwróceniem uwagi na rozdział 3 - System IDE51 - Środowisko programowe.

Wymagana jest znajomość:

- uruchamiania systemu IDE51;
- przeznaczenia standardowych klawiszy funkcyjnych, menu i opcji systemu IDE51;

- umiejętność zarządzania oknami systemu IDE51;
- otwierania nowego okna edycyjnego;
- wykonywania prostych i złożonych operacji na tekście znajdującym się w otwartym oknie edycyjnym;
- kompilacji programu;
- przesyłania programu do systemu mikroprocesorowego (µM-DYD);
- uruchamiania programu w systemie mikroprocesorowym;
- śledzenia i modyfikacji zawartości rejestrów i komórek pamięci.

4.2.4. Elementy programowania w języku asemblera

Cel dydaktyczny ćwiczenia:

Celem ćwiczenia jest zapoznanie się z podstawami programowania w języku asemblera, na przykładzie języka asemblera 8051 i rozwijanie umiejętności uruchamiania programów.

Przedmiot ćwiczenia :

- rejestry robocze i rejestry ogólnego przeznaczenia SFR mikrokontrolera 8051;
- język asemblera:
  - argumenty,
  - adresy,
  - dyrektywy przypisania nazwie wartości,
  - operatory logiczne,
  - etykiety,
- opcje programu MONITOR-51.

Program ćwiczenia obejmuje:

♦ konstrukcję i uruchamianie programów liniowych;

- konstrukcję i uruchamianie prostych programów zawierających moduły warunkowe i pętle.

Zakres materiału obowiązujący studenta:

Znajomość cyklu tworzenia oprogramowania w języku asemblera.

Wymagana jest znajomość rejestrów specjalnych SFR i rejestrów roboczych ich organizacji w banki rejestrów oraz sposób zmiany aktualnego banku mikrokontrolera 8051.

Umiejętność uruchamiania programu w trybie pracy krokowej, podglądu oraz modyfikacji wybranych rejestrów specjalnych SFR i komórek wewnętrznej pamięci mikrokontrolera.

4.2.5. Wymiana danych w systemie mikroprocesorowym

Cel dydaktyczny ćwiczenia:

Celem ćwiczenia jest zapoznanie się z organizacją i trybami adresowania wewnętrznej pamięci danych i programu mikrokontrolera oraz z wymianą danych pomiędzy mikrokontrolerem a pamięciami zewnętrznymi.

Przedmiot ćwiczenia :

- wewnętrzna pamięć danych DATA;
- wewnętrzna pamięć danych IDATA;
- zewnętrzna pamięć danych XDATA;
- pamięć kodu programu CODE;
- organizacja stosu;
- tryby adresowania;
- dyrektywy inicjacji i rezerwacji obszarów pamięci.

Program ćwiczenia obejmuje:

- konstrukcję i uruchamianie programów kopiowania danych w ramach wewnętrznej pamięci danych;

- ♦ konstrukcję i uruchamianie programów kopiowania danych z pamięci programu do wewnętrznej pamięci danych;
- ♦ konstrukcję i uruchamianie programów kopiowania danych z zewnętrznej do wewnętrznej pamięci danych.

Zakres materiału obowiązujący studenta:

Wymagana jest znajomość organizacji wewnętrznej i zewnętrznej pamięci danych i programu mikrokontrolera 8051 oraz podstawowe tryby ich adresowania. Sposób dołączania zewnętrznych pamięci do mikrokontrolera oraz przeznaczenie sygnałów generowanych przez mikrokontroler służących do obsługi tych pamięci. Znajomość dyrektyw języka asemblera.

4.2.6. Operacje arytmetyczne i logiczne

Cel dydaktyczny ćwiczenia:

Celem ćwiczenia jest zapoznanie się z podstawowymi operacjami arytmetycznymi i logicznymi wykonywanymi na argumentach jednobajtowych oraz z zamianą kodu znaków.

Przedmiot ćwiczenia :

- wewnętrzna pamięć danych DATA;
- flagi rejestru stanu PSW;
- język asemblera:
  - instrukcje arytmetyczne i logiczne,
  - operatory arytmetyczne,
  - operatory logiczne,
  - wyrażenia,
- okno Register programu IDE51.

Program ćwiczenia obejmuje:

- konstrukcję i uruchamianie programów realizujących dodawanie argumentów jednobajtowych umieszczonych w różnych obszarach pamięci mikrokontrolera;
- konstrukcję i uruchamianie programów realizujących operacje mnożenia i dzielenia argumentów jednobajtowych umieszczonych w różnych obszarach pamięci mikrokontrolera;
- konstrukcję i uruchamianie programów realizujących konwersję kodów znaków.

Zakres materiału obowiązujący studenta:

Wymagana jest znajomość rozkazów arytmetycznych i logiczny. Znajomość i przeznaczenie poszczególnych flag rejestru stanu PSW podczas wykonywania operacji arytmetycznych jak również umiejętność odczytania i dokonania interpretacji wspomnianych flag w systemie IDE51.

4.2.7. Dołączanie do mikrokontrolera układów prostych we-wy

Cel dydaktyczny ćwiczenia:

Celem ćwiczenia jest zapoznanie z strukturą portów mikrokontrolera 8051 i zasadami dołączania do mikrokontrolera układów prostych we-wy.

Przedmiot ćwiczenia :

- porty mikrokontrolera;
- wyświetlacze dwustanowe modułu dydaktycznego μM-DYD;
- piezoelement (buzzer) emitujący dźwięki o programowanej częstotliwości;
- makieta dydaktyczna „Wyświetlacz LED”;
- makieta dydaktyczna „Skrzyżowanie”;
- składnia języka asemblera;

- program obsługujący sygnalizację świetlną na makiecie „Skrzyżowanie".

Program ćwiczenia obejmuje:

- ♦ szczegółowe zapoznanie się z strukturą portów P1 oraz P3, technikami buforowania ich linii we-wy i dołączania do nich układów typu wskaźniki LED, klawiatura statyczna i brzęczyk;
- ♦ konstrukcję i uruchamianie programów obsługujących wyświetlacze 7-segmentowe;
- ♦ konstrukcję i uruchamianie programów sterujących sygnalizacją świetlną na skrzyżowaniu;
- ♦ konstrukcję i uruchamianie programów generujących akustyczne sygnały alarmowe oraz proste melodie.

Zakres materiału obowiązujący studenta:

Wymagana jest znajomość budowy portów mikrokontrolera 8051, zasady ich obsługi oraz przeznaczenia. Zasady wykorzystania portów P1 i P3 (sposób buforowania i zobrazowywania informacji wyjściowej) w module dydaktycznym µM-DYD.

Studenci winni zapoznać się z opracowaniem „Laboratorium Systemów Mikroprocesorowych" ze szczególnym zwróceniem uwagi na opis budowy, sposób sterowania i wykorzystania makiet dydaktycznych „Wyświetlacz LED" oraz „Skrzyżowanie".

4.2.8. Liczniki T0 i T1

Cel dydaktyczny ćwiczenia:

Celem ćwiczenia jest zapoznanie się z budową, zasadą działania i możliwościami funkcjonalnymi zegarów-liczników T0 i T1.

Przedmiot ćwiczenia :

- licznik T0;
- licznik T1;
- wyświetlacze dwustanowe modułu dydaktycznego µM-DYD;
- makieta dydaktyczna „Skrzyżowanie”;
- makieta dydaktyczna „Tęcza”;
- składnia języka asemblera.

Program ćwiczenia obejmuje:

- ♦ szczegółowe zapoznanie się z strukturą zegarów-liczników oraz z zasadami ich programowania;
- ♦ konstrukcję i uruchamianie programów odmierzania czasu;
- ♦ konstrukcję i uruchamianie programów zliczania zdarzeń zewnętrznych.

Zakres materiału obowiązujący studenta:

Wymagana jest znajomość budowy liczników T0 i T1, sposób sterowania licznikami oraz rejestrami SFR wykorzystywanymi do obsługi wymienionych liczników. Studenci winni zapoznać się z instrukcją laboratoryjną „Laboratorium Systemów Mikroprocesorowych” ze szczególnym zwróceniem uwagi na opis budowy, sposób sterowania i wykorzystania makiet dydaktycznych „Tęcza” oraz „Skrzyżowanie”.

4.2.9. Licznik T2

Cel dydaktyczny ćwiczenia:

Celem ćwiczenia jest zapoznanie się z budową, zasadą działania i możliwościami funkcjonalnymi licznika T2.

Przedmiot ćwiczenia :

- licznik T2;
- wyświetlacze dwustanowe modułu dydaktycznego µM-DYD;
- składnia języka asemblera.

Program ćwiczenia obejmuje:

Program ćwiczenia obejmuje:

- ♦ szczegółowe zapoznanie się ze strukturą licznika T2 oraz z zasadą jego programowania;
- ♦ konstrukcję i uruchamianie programów odmierzania czasu;
- ♦ konstrukcje i uruchamianie programów generujących sygnały PWM.

Zakres materiału obowiązujący studenta:

Wymagana jest znajomość budowy licznika T2 oraz sposób sterowania licznikiem. Znajomość rejestrów wykorzystywanych przez układ licznika T2. Sposób generacji sygnałów PWM przy pomocy układu licznika T2 pracującego w trybie porównania. Studenci winni zapoznać się z instrukcją laboratoryjną „Laboratorium Systemów Mikroprocesorowych" ze szczególnym zwróceniem uwagi na opis budowy, sposób sterowania i wykorzystania makiet dydaktycznych „Tęcza" oraz „Skrzyżowanie".

4.2.10. System przerwań

Cel dydaktyczny ćwiczenia:

Celem ćwiczenia jest zapoznanie się z budową, zasadą działania i możliwościami funkcjonalnymi oraz logiką przerwań mikrokontrolera 8051.

Przedmiot ćwiczenia :

- system przerwań;
- wyświetlacze dwustanowe modułu dydaktycznego uM-DYD;
- makieta dydaktyczna „Skrzyżowanie";
- makieta dydaktyczna „Tęcza";

- składnia języka asemblera.

Program ćwiczenia obejmuje:

Program ćwiczenia obejmuje:

- ♦ szczegółowe zapoznanie się z logiką przerwań oraz z zasadami ich programowania i wykorzystania;
- ♦ konstrukcję i uruchamianie programów wykorzystujących system przerwań.

Zakres materiału obowiązujący studenta:

Budowa systemu przerwań mikrokontrolera 8051, źródła przerwań, sposób odblokowywania systemu przerwań jak również umiejętność zmiany priorytetu poszczególnych przerwań.

4.2.11. Klawiatura

Cel dydaktyczny ćwiczenia:

Celem ćwiczenia jest kształtowanie umiejętności dołączania i obsługi układów we-wy, takich jak klawiatura chwilowa i statyczna.

Przedmiot ćwiczenia :

- klawiatura chwilowa;
- klawiatura statyczna;
- uruchamiane programy obsługowe klawiatury;
- liczniki T0 i T1;
- wyświetlacze dwustanowe modułu dydaktycznego μM-DYD;
- składnia języka asemblera.

Program ćwiczenia obejmuje:

Program ćwiczenia obejmuje:

♦ szczegółowe zapoznanie się z klawiaturą chwilową i statyczną modułu dydaktycznego;

♦ konstrukcję i uruchamianie programów obsługujących klawiaturę chwilową;

♦ konstrukcję i uruchamianie programów obsługujących klawiaturę statyczną.

Zakres materiału obowiązujący studenta:

Znajomość budowy oraz zasad odczytu informacji z klawiatury chwilowej (przyciski K0 – K15) oraz statycznej (przyciski K16 – K23) modułu dydaktycznego μM-DYD.

4.2.12. Wyświetlacz alfanumeryczny LCD

Cel dydaktyczny ćwiczenia:

Celem ćwiczenia jest rozwijanie umiejętności konstruowania elementarnych programów zarządzających i obsługujących pracę wyświetlacza LCD. Ćwiczenie zapoznaje również szczegółowo z architekturą wyświetlacza ciekłokrystalicznego LCD.

Przedmiot ćwiczenia :

- wyświetlacz alfanumeryczny LCD;
- liczniki T0 i T1;
- wyświetlacze dwustanowe modułu dydaktycznego μM-DYD;
- składnia języka asemblera.

Program ćwiczenia obejmuje:

Opracowanie i uruchomienie elementarnego programu obsługującego konsolę (klawiatura i wyświetlacz LCD) i umożliwiającego:

♦ identyfikację naciśniętego klawisza klawiatury statycznej i realizację przypisanej mu funkcji:

- przesuwanie kursora wyświetlacza o jedną pozycję w prawo,
- przesuwanie kursora wyświetlacza o jedną pozycję w lewo,
- przesuwanie kursora z dolnego wiersza do górnego i odwrotnie,
- kasowanie wskazanego przez kursor znaku,
- wstawianie znaku spacji w miejsce wyróżnione kursorem,
- przenoszenie tekstu z dolnego wiersza do górnego,

♦ identyfikację naciśniętego klawisza klawiatury chwilowej i wyprowadzenie na wyświetlacz w miejscu wskazanym przez kursor numeru klawisza;

♦ konstrukcję i uruchamianie programu programowania i wyświetlenia na wyświetlaczu polskich znaków np. ą, ć, ż, ź.

Zakres materiału obowiązujący studenta:

Budowa wyświetlacza alfanumerycznego LCD modułu dydaktycznego µM-DYD.

Umiejętność podania przeznaczenia poszczególnych elementów składowych programowalnego sterownika wyświetlacza.

Zasada sterowania pracą wyświetlacza:

- ustalenie szerokości słowa sterującego;
- włączenie i wyłączenie oraz wybór rodzaju kursora przedstawiającego aktualną pozycję wpisywanego znaku;
- określenie, czy podczas wpisywania znaków ma się przesuwać kursor i w którym kierunku, a wyświetlane znaki (ekran) mają nie zmieniać pozycji, czy kursor jest na stałej pozycji a wyświetlone już znaki (ekran) przesuwają się w lewo lub w prawo;
- ustawianie kursora na wybranej pozycji.

Umiejętność wyświetlenia znaku pod określonym adresem z pamięci generatora znaków. Znajomość sposobu sprawdzania gotowości wyświetlacza do dalszej pracy po wysłaniu danej lub rozkazu do wyświetlacza.

Wymagana jest znajomość zasad wyświetlania znaków umieszczonych w pamięci XDATA oraz CODE. Ponadto studenci powinni umieć zaprogramować polskie litery w matrycy 5×7 i wyświetlić na wyświetlaczu zaprogramowany znak.

4.2.13. Szeregowa transmisja informacji

Cel dydaktyczny ćwiczenia:

Celem ćwiczenia jest zapoznanie się z zasadami organizacji szeregowej transmisji informacji oraz ze strukturą i sposobami programowania układu transmisji szeregowej mikrokontrolera 8051.

Przedmiot ćwiczenia :

- układ transmisji szeregowej mikrokontrolera 8051;
- klawiatura chwilowa;
- klawiatura statyczna;
- wyświetlacz LCD;
- terminal TTY;
- składnia języka asemblera.

Program ćwiczenia obejmuje:

- ♦ szczegółowe zapoznanie się z strukturą i zasadami programowania układu transmisji szeregowej mikrokontrolera 8051;
- ♦ konstrukcję i uruchamianie programów umożliwiających realizację szeregowej transmisji asynchronicznej jednokierunkowej (simplex);
- ♦ konstrukcję i uruchamianie programów umożliwiających realizację szeregowej transmisji asynchronicznej jednoczesnej (duplex).

Zakres materiału obowiązujący studenta:

Wymagana jest znajomość:

- budowy oraz sposóbu konfiguracji układu transmisji szeregowej mikrokontrolera 8051;
- tryby pracy układu transmisji szeregowej;
- rodzaje transmisji szeregowej (8 - bitowa i 9 - bitowa);
- umiejętność ustawienia określonej częstotliwości taktującej układ transmisji szeregowej mikrokontrolera 8051 i 80C537;
- przesyłania przez układ transmisji szeregowej informacji umieszczonej w pamięci CODE z jednoczesnym wyświetleniem jej w oknie terminala;
- odbioru informacji przez port szeregowy np. z klawiatury komputera PC.

4.2.14. Termin dodatkowy obróbczy i poprawkowy

Terminy dodatkowe umożliwiają poprawienie i odrobienie zaległości. W terminach obróbczych istnieje możliwość poprawienia dwóch ćwiczeń laboratoryjnego - poprawa jednego tematu w czasie jednego spotkania.

LITERATURA

1. *Moduł dydaktyczny μM-DYD. Wersja 4.0*. MicroMax 1995.
2. Rydzewski A.: *Mikrokomputery jednuukładowe rodziny MCS51*. Seria: Podręczny Katalog Elektronika. WNT. Warszawa 1992.
3. Janiczek J., Stępień A.: *Mikrokontroler SAB 80(C)515/535*. Wydawnictwo WZN. Wrocław 1995.
4. Janiczek J., Stępień A.: *Laboratorium systemów mikroprocesorowych. Cz. I.* Wydawnictwo EZN. Wrocław 1995.
5. Janiczek J., Stępień A.: *Laboratorium systemów mikroprocesorowych. Cz. II.* Wydawnictwo EZN. Wrocław 1997.
6. Gałka P., Gałka P.: *Podstawy programowania mikrokontrolera 8051.* ZNI „Mikon", Warszawa 1995.
7. Doliński J.: *Mikrokomputer jednoukładowy 8051.* Wydawnictwo PLJ. Warszawa 1993.
8. Starecki T.: *Mikrokomputery jednoukładowe INTEL 8051.* „NAZOMI", Warszawa 1996.
9. *IDE-51 Zintegrowane środowisko programowe dla procesorów rodziny 8051. Wersja 2.1*. MicroMax 1995.
10. *Moduł sterownika uM-537F. Instrukcja użytkowania. Wersja 3.0*. Micro-Max 2003.
11. *Makieta dydaktyczna 'SKRZYŻOWANIE'. Wersja 3.0*. MicroMax 1994.
12. *Makieta dydaktyczna 'WYŚWIETLACZ LED'. Wersja 3.0*. MicroMax 1994.
13. *Makieta dydaktyczna 'TĘCZA'. Wersja 3.0*. MicroMax 1994.

DODATEK A

Predefiniowane rejestry specjalne SFR mikrokontrolerów rodziny '51

A1. Rejestry bajtowe 80(C)51

Nazwa SFR	Adres
ACC	**0E0h**
RB	**0F0h**
DPH	**083h**
DPL	**082h**
IE	**0A8h**
IP	**0B8h**
P0	**080h**
P1	**080H**
P2	**0A0H**
P3	**0B0H**
PCON	087H
PSW	**0D0H**
SBUF	099H
SCON	**098H**
SP	081H
TCON	**088H**
TH0	08CH
TH1	08DH
TL0	08AH
TL1	08BH
TMOD	089H

A2. Rejestry bitowe 80(C)51

Nazwa SFR	Adres
ACY	0D0H.6
CY	0D0H.7
EAL	0A8H.7
ES	0A8H.4
ET0	0A8H.1
ET1	0A8H.3
EX0	0A8H.0
EX1	0A8H.2
FL0	0D0H.3
IE0	088H.1
IE1	088H.3
INT0	0B0H.2
INT1	0B0H.3
IT0	088H.0
IT1	088H.2
OV	0D0H.2
P	0D0H.0
PS	0B8H.4
PT0	0B8H.1
PT1	0B8H.3
PX0	0B8H.0
PX1	0B8H.2
RB8	098H.2
RD	0B0H.7
REN	098H.4
RI	098H.0
RS0	0D0H.3
RS1	0D0H.4
RXD	0B0H.0
SM0	098H.7
SM1	098H.6
SM2	098H.5
T0	0B0H.4
T1	0B0H.5
TB8	098H.3
TF0	088H.5
TF1	088H.7
TI	098H.1
TR0	088H.4
TR1	088H.6
TXD	0B0H.1
WR	0E0H.6

A3. Dodatkowe rejestry bajtowe 80(C)515/80(C)535

Nazwa SFR	Adres
ADCON	**0D8H**
ADDAT	0D9H
CCEN	0C1H
CCH1	0C3H
CCH2	0C5H
CCH3	0C7H
CCL1	0C2H
CCL2	0C4H
CCL3	0C6H
CRCH	0CBH
CRCL	0CAH
DAPR	0CDH
IEN0	**0A8H**
IEN1	**0B8H**
IP0	0A9H
IP1	0B9H
IRCON	**0C0H**
P4	**0E8H**
P5	**0F8H**
T2CON	**0C8H**
TH2	0CDH
TL2	0CCH

A4. Dodatkowe rejestry bitowe 80(C)515/80(C)535

Nazwa SFR	Adres
ADM	0D8H.4
BRE	0D8H.7
BSY	0D8H.5
CLK	0D8H.6
CLKOUT	090H.6
EADC	0B8H.0
EAL	0A8H.7
ET2	0A8H.5
EX2	0B8H.1
EX3	0B8H.2
EX4	0B8H.3
EX5	0B8H.4
EX6	0B8H.5
EXEN2	0B8H.7
EXF2	0C0H.7
FL1	0D0H.1
I2FR	0C8H.5
I3FR	0C8H.6
IADC	0C0H.0
IEX2	0C0H.1
IEX3	0C0H.2
IEX4	0C0H.3
IEX5	0C0H.4
IEX6	0C0H.5
INT2	090H.4
INT3	090H.0
INT4	090H.1
INT5	090H.2
INT6	090H.3
MX0	0D8H.1
MX1	0D8H.2
M X1	0D8H.3
SWDT	0B8H.6
T2	090H.7
T2CM	0C8H.2
T2EX	090H.5
T2I0	0C8H.0
T2I1	0C8H.1
T2PS	0C8H.7
T2R0	0C8H.3
T2R1	0C8H.4
T2F2	0C0H.6
WDT	0A8H.6

A5.Rejestry 80(C)517/80(C)537

Nazwa SFR	Adres
ACC	**0E0H**
B	**0F0H**
DPH	83H
DPL	82H
DPSEL	92H
PSW	**0D0H**
SP	81H
ADCON0	**0D8H**
ADCON1	0DCH
ADDAT	0D9H
DAPR	0DAH
IEN0	**0A8H**
CTCON	0E1H
IEN1	**0B8H**
IEN2	9AH
IP0	0A9H
IP1	0B9H
IRCON	**0C0H**
TCON	**88H**
T2CON	**0C8H**
ARCON	0EFH
MD0	0E9H
MD1	0EAH
MD2	0EBH
MD3	0ECH
MD4	0EDH
MD5	0EEH
CCEN	0C1H
CC4EN	0C9H
CCH1	0C3H
CCH2	0C5H
CCH3	0C7H
CCH4	0CFH
CCL1	0C2H
CCL2	0C4H
CCL3	0C6H
CCL4	0CEH
CMEN	0F6H
CMH0	0D3H
CMH1	0D5H
CMH2	0D7H
CMH3	0E3H
CMH4	0E5H
CMH4	0E7H
CMH5	0F3H
CMH6	0F5H
CMH7	0D2H

Nazwa SFR	Adres
CML0	0D2H
CML1	0D4H
CML2	0D6H
CML3	0E2H
CML4	0E4H
CML5	0E6H
CML6	0F2H
CML7	0F4H
CMSEL	0F7H
CRCH	0CBH
CRCL	0CAH
CTCON	0E1H
CTRELH	0DFH
CTRELL	0DEH
TH2	0CDH
TL2	0CCH
P0	**80H**
P1	**90H**
P2	**0A0H**
P3	**0B0H**
P4	**0E8H**
P5	**0F8H**
P6	0FAH
P7	0DBH
P8	0DDH
ADCON0	**0D8H**
PCON	87H
S0BUF	99H
S0CON	**98H**
S0RELL	0AAH
S0RELH	0BAH
S1BUF	9CH
S1CON	9BH
S1REL	9DH
S1RELH	0BBH
TCON	**88H**
TH0	8CH
TH1	8DH
TL0	8AH
TL1	8BH
TMOD	89H
IEN0	**0A8H**
IEN1	**0B8H**
IP0	0A9H
IP1	0B9H
WDTREL	86H

DODATEK B

Lista instrukcji mikrokontrolera Intel 80(C)51

Oznaczenia stosowane na listach rozkazów:

Wszystkie oznaczenia rejestrów, wskaźników, portów i układów funkcjonalnych są identyczne z oznaczeniami stosowanymi w opisie architektury mikrokomputera [2]. Pozostałe oznaczenia mają następujący sens:

a - adres 8 - bitowy; służy do wyznaczenia efektywnego adresu pamięci programu,

a8 - 8 - bitowy bezpośredni adres danej. Wyróżnia komórkę wewnętrznej pamięci danych (lokacje 0 - 127) lub rejestr specjalny,

a11 - adres 11 - bitowy,

aaa - bity a_{10}, a_9 i a_8 adresu 11 - bitowego,

a16 - adres 16 - bitowy,

Bb - bit akumulatora o numerze b; b = 0 - 7,

bit - adres bezpośredni bitu wewnętrznej pamięci danych - obszar adresowania bitów - lub adres rejestru specjalnego,

d8 - dana 8 - bitowa,

d16 - dana 16 - bitowa,

rel - 8 - bitowy adres względny (liczba z przedziału -128.....+127),

Ri - rejestr adresowy aktualnie wybranego zestawu rejestrów; i = 0 - 1,

Rr - rejestr roboczy zpośród aktualnie wybranego zestawu rejestrów; r = 0 - 7,

(N) - zawartość komórki pamięci adresowanej zawartością rejestru o nazwie N lub adresem N,

- prefiks argumentu bezpośredniego,

@ - prefiks adresowania rejestrowego bezpośredniego,

x - iloczyn arytmetyczny,

: - iloraz,

/ - negacja logiczna; NOT,

$\vee$ - suma logiczna; OR,

$\wedge$ - iloczyn logiczny; AND,

$\oplus$ - wyłączne lub; XOR,

$\leftrightarrow$ - wymiana zawartości (bajtami lub tetradami),

XSEG - zewnętrzna pamięć danych,

CSEG - pamięć programu,

BSEG -obszar adresowania bitów (wewnętrznej pamięci danych lub rejestrów specjalnych).

Ustawianie wskaźników:

- - zawartość wskaźnika nie ulega zmianie,
- + wskaźnik jest ustawiany stosownie do wyniku operacji.

W kolumnie „**operacja**" - w tabelach instrukcji - zapis po znaku „**;**" jest komentarzem do opisywanej operacji. Wskaźniki **RS** i **FO** słowa stanu nie zmieniają się z wyjątkiem rozkazów określających operacje bezpośrednio dotyczące słowa stanu lub poszczególnych jego bitów.

INSTRUKCJE PRZESŁAŃ

Tabela B.1

Mnemonika	Liczba bajtów	Liczba cykli	Kod szesnast-kowy	CY	AC	OV	P	Operacja
MOV A,Rr	1	1	E8-EF	-	-	-	+	A := Rr; r = 0 -7
MOV A,a8	2	1	E5	-	-	-	+	A := (a8)
MOV A,@Ri	1	1	E6-E7	-	-	-	+	A := (Ri); i = 0 - 1
MOV A,#d8	2	1	74	-	-	-	+	A := d8
MOV Rr,A	1	1	F8-FF	-	-	-	-	Rr := A; r = 0 – 7
MOV Rr,a8	2	2	A8-AF	-	-	-	-	Rr := (a8); r = 0 -7
MOV Rr,#d8	2	1	78-7F	-	-	-	-	Rr := d8; r = 0-7
MOV a8,A	2	1	F5	-	-	-	-	(a8) := A
MOV a8,Rr	2	2	86-8F	-	-	-	-	(a8) := Rr; r = 0-7
MOV a8,a8	3	2	85	-	-	-	-	(a8) := (a8); różne
MOV a8,@Ri	2	2	86-87	-	-	-	-	(a8) := (Ri); i = 0 - 1
MOV a8,#d8	3	2	75	-	-	-	-	(a8) := d8
MOV @Ri,A	1	1	F6-F7	-	-	-	-	(Ri) := A; i = 0 - 1
MOV @Ri,a8	2	2	A6-A7	-	-	-	-	(Ri) := (a8); i = 0 – 7
MOV @Ri,#d8	2	1	76-77	-	-	-	-	(Ri) := d8; i = 0 – 1
MOV DPTR,#d16	3	2	90	-	-	-	-	DPTR := d16
MOVC A,@A+DPTR	1	2	93	-	-	-	+	A := (A+DPTR); CSEG
MOVC A,@A+PC	1	2	83	-	-	-	+	A := (A+PC); CSEG
MOVX A,@Ri	1	2	E2-E3	-	-	-	+	A := (P2,Ri); XSEG
MOVX A,@DPTR	1	2	E0	-	-	-	+	A := (DPTR); XSEG
MOVX @Ri,A	1	2	F2-F3	-	-	-	-	(P2,Ri) := A; XSEG
MOVX @DPTR,A	1	2	F0	-	-	-	-	(DPTR) := A; XSEG
PUSH a8	2	2	C0	-	-	-	-	SP := SP+1; (SP):= (a8)
POP a8	2	2	D0	+	+	+	+	(a8) := (SP); SP := SP-1
XCH A,Rr	1	1	C8-CF	-	-	-	+	A ↔ Rr; r = 0 – 7
XCH A,a8	2	1	C5	-	-	-	+	A ↔ (a8)
XCH A,@Ri	1	1	C6-C7	-	-	-	+	A ↔ (Ri); i = 0 – 1
XCHD A,@Ri	1	1	D6-D7	-	-	-	+	$A_{0-3} \leftrightarrow (Ri)_{0-3}$; i = 0-1

INSTRUKCJE ARYTMETYCZNE

Tabela B.2

Mnemonika	Liczba bajtów	Liczba cykli	Kod szesnastko-wy	CY	AC	OV	P	Operacja
ADD A,Rr	1	1	28-2F	+	+	+	+	A := A+Rr; r = 0-7
ADD A,a8	2	1	25	+	+	+	+	A := A + (a8)
ADD A,@Ri	1	1	26-27	+	+	+	+	A := A + (Ri); i = 0-1
ADD A,#d8	2	1	24	+	+	+	+	A := A + d8
ADDC A,Rr	1	1	38-3F	+	+	+	+	A := A + Rr + CY; r = 0-7
ADDC A,a8	2	1	35	+	+	+	+	A := A + (a8) + CY
ADDC A,@Ri	1	1	36-37	+	+	+	+	A := A + (Ri) + CY; i = 0-1
ADDC A,#d8	2	1	34	+	+	+	+	A := A + d8 + CY
SUBB A,Rr	1	1	98-9F	+	+	+	+	A := A – Rr - CY; r = 0-7
SUBB A,a8	2	1	95	+	+	+	+	A := A –d8 – CY
SUBB A,@Ri	1	1	96-97	+	+	+	+	A := A - (Ri) - CY; i = 0-1
SUBB A,#d8	2	1	94	+	+	+	+	A := A - d8 – CY
INC A	1	1	04	+	+	+	+	A := A + 1
INC Rr	1	1	08-0F	-	-	-	-	Rr := Rr + 1; r = 0 – 7
INC a8	2	1	05	-	-	-	-	(a8) := (a8) + 1
INC @Ri	1	1	06-07	-	-	-	-	(Ri) := (Ri) + 1
INC DPTR	1	2	A3	-	-	-	-	DPTR := DPTR + 1
DEC A	1	1	14	-	-	-	+	A := A – 1
DEC Rr	1	1	18-1F	-	-	-	-	Rr := Rr – 1; r = 0 – 7
DEC a8	2	1	15	-	-	-	-	(a8) := (a8) – 1
DEC @Ri	1	1	16-17	-	-	-	-	(Ri) := (Ri) - 1; i = 0 -1
MUL AB	1	4	A4	0	-	+	+	BA := AxB
DIB AB	1	4	84	0	-	+	+	AB := A/B
DA	1	1	4	+	-	-	+	Korekcja dziesiętna A

INSTRUKCJE LOGICZNE Tabela B.3

Mnemonika	Liczba bajtów	Liczba cykli	Kod szesnastkowy	CY	AC	OV	P	Operacja
ANL A,Rr	1	1	58-5F	-	-	-	+	A := A ∧ Rr; r = 0 – 7
ANL A,a8	2	1	55	-	-	-	+	A : = A ∧ (a8)
ANL A,@Ri	1	1	56-57	-	-	-	+	A := A ∧ (Ri); i = 0 – 1
ANL A,#d8	2	1	54	-	-	-	+	A := A ∧ d8
ANL a8,A	2	1	52	-	-	-	-	(a8) := (a8) ∧ A
ANL a8,#d8	3	2	53	-	-	-	-	(a8) ;= (a8) ∧ d8
ORL A,Rr	1	1	48-4F	-	-	-	+	A := A ∨ Rr; r = 0 – 7
ORL A,a8	2	1	45	-	-	-	+	A := A ∨ (a8)
ORL A,@Ri	1	1	46-47	-	-	-	+	A := A ∨ (Ri); i = 0 – 1
ORL A,#d8	2	1	44	-	-	-	+	A := A ∨ d8
ORL a8,A	2	1	42	-	-	-	-	(a8) := (a8) ∨ A
ORL a8,#d8	3	2	43	-	-	-	-	(a8) := (a8) ∨ d8
XRL A,Rr	1	1	68-6F	-	-	-	+	A := A ⊕ Rr; r = 0 – 7
XRL A,a8	2	1	65	-	-	-	+	A := A ⊕ (a8)
XRL A,@Ri	1	1	66-67	-	-	-	+	A := A ⊕ (Ri); i = 0 - 1
XRL A,#d8	2	1	64	-	-	-	+	A := A ⊕ d8
XRL a8,A	2	1	62	-	-	-	-	(a8) := (a8) ⊕ A
XRL a8,#d8	3	2	63	-	-	-	-	(a8) := (a8) ⊕ d8
CLR A	1	1	E4	-	-	-	+	A := 0
CPL A	1	1	F4	-	-	-	-	A := /A
RL A	1	1	23	-	-	-	-	$A_0 := A_7$, $A_n := A_{n-1}$; n = 1 – 7
RLC A	1	1	33	+	-	-	+	$A_0 := CY$, $A_n := A_{n-1}$; $CY := A_7$, n = 1 – 7
RRA	1	1	03	-	-	-	-	$A_7 := A_0$, $A_n := A_{n+1}$, n = 0 – 6
RRC A	1	1	13	+	-	-	+	$A_7 := CY$, $A_n := A_{n+1}$, $CY := A_0$, n = 0 – 6
SWAP A	1	1	C4	-	-	-	-	$A_{4-7} \leftrightarrow A_{0-3}$

INSTRUKCJE USTAWIAJĄCE WSKAŹNIKI SŁOWA STANU Tabela B.4

Mnemonika	CY	AC	OV	Mnemonika	CY	AC	OV
ADD	+	+	+	**CLR C**	0	-	-
ADDC	+	+	+	**CPL C**	+	-	-
SUBB	+	+	+	**ANL C,bit**	+	-	-
MUL	0	-	+	**ANL C,/bit**	+	-	-
DIV	0	-	+	**ORL C,bit**	+	-	-
DA	+	-	-	**ORL C,/bit**	+	-	-
RRC	+	-	-	**MOV C,bit**	+	-	-
RLC	+	-	-	**INC A**	+	+	+
SETB C	1	-	-	**POP a8**	+	+	+

INSTRUKCJE ROZGAŁĘZIENIA LUB ZATRZYMANIA PROGRAMU Tabela B.5

Mnemonika	Liczba bajtów	Liczba cykli	Kod szesnast-kowy	CY	AC	OV	P	Operacja
ACALL a11	2	2	aaa10001	-	-	-	-	SP:= SP + 1, (SP) := PC_{0-7}, SP:= SP + 1, (SP) := PC_{8-15} PC_{0-10} := a11
LCALL a16	3	2	12	-	-	-	-	SP := SP+1, (SP) := PC_{0-7} SP := SP+1, (SP) := PC_{8-15} PC := a16
RET	1	2	22	-	-	-	-	PC_{8-15} := (SP), SP := SP-1 PC_{0-7} := (SP), SP := SP-1
RETI	1	2	32	-	-	-	-	PC_{8-15} := (SP), SP := SP-1 PC_{0-7} := (SP), SP := SP-1
AJMP a11	2	2	aaa00001	-	-	-	-	PC_{0-10} := a11
LJMP a16	3	2	02	-	-	-	-	PC := a16
SJMP rel	1	2	80	-	-	-	-	PC := PC + rel
JMP @A+DPTR	1	2	73	-	-	-	-	PC := DPTR + A
JZ rel	2	2	60	-	-	-	-	Jeśli A=0, to PC := PC+rel
JNZ rel	2	2	70	-	-	-	-	Jeśli A#0, to PC := PC+rel
CJNE A,a8,rel	3	2	B5	+	-	-	-	Jeśli A#(a8), to PC:=PC+rel
CJNE A,#d8,rel	3	2	B4	+	-	-	-	Jeśli A#d8, to PC:=PC+rel
CJNE Rr,#d8,rel	3	2	B8-BF	+	-	-	-	Jeśli Rr#d8, to PC:=PC+rel r = 0 -7
CJNE @Ri,#d8,rel	3	2	B6-B7	+	-	-	-	Jeśli (Ri)#d8, to PC:=PC+rel, i = 0 – 1
DJNZ Rr,rel	2	2	8-DF	-	-	-	-	Rr := Rr-1. Jeśli Rr # 0 to PC := PC + rel, r = 0 -7
DJNZ a8,rel	3	2	D5	-	-	-	-	(a8) := (a8) - 1. Jeśli (a8)#0 to PC := PC + rel
NOP	1	1	00	-	-	-	-	Nic nie rób

INSTRUKCJE DZIAŁAŃ NA POJEDYNCZYCH BITACH Tabela B.6

Mnemonika	Liczba bajtów	Liczba cykli	Kod szesnast-kowy	CY	AC	OV	P	Operacja
MOV C,bit	2	1	A2	+	-	-	-	CY := (bit); BSEG
MOV bit,C	2	2	92	-	-	-	-	(bit) := CY; BSEG
CLR C	1	1	C3	0	-	-	-	CY := 0
CLR bit	2	1	C2	-	-	-	-	(bit) := 0; BSEG
SETB C	1	1	D3	1	-	-	-	CY := 1
SETB bit	2	1	D2	-	-	-	-	(bit) := 1; BSEG
CPL C	1	1	B3	+	-	-	-	CY := /CY
CPL bit	2	1	B2	-	-	-	-	(bit) := /(bit); BSEG
ANL C,bit	2	2	82	+	-	-	-	CY := CY $\wedge$ (bit);BSEG
ANL C,/bit	2	2	B0	+	-	-	-	CY:=CY $\wedge$ /(bit); BSEG
ORL C,bit	2	2	72	+	-	-	-	CY:=CY $\vee$ (bit); BSEG
ORL C,/bit	2	2	A0	+	-	-	-	CY:=CY $\vee$ /(bit);BSEG
JC rel	2	2	40	-	-	-	-	Jeśli CY = 1, to PC := PC + rel
JNC rel	2	2	50	-	-	-	-	Jeśli CY = 0, to PC := PC + rel
JB bit,rel	3	2	20	-	-	-	-	Jeśli (bit) = 1, to PC := PC + rel; BSEG
JNB bit,rel	3	2	30	-	-	-	-	Jeśli (bit) = 0, to PC := PC + rel; BSEG
JBC bit,rel	3	2	10	-	-	-	-	Jeśli (bit) = 1 to PC := PC + rel oraz (bit) := 0; BSEG

DODATEK C
Komunikaty o błędach Asemblera i Monitora:

C1. Komunikaty Asemblera:

1. VALUE HAS BEEN TRUNCATED TO 8 BITS

Przekroczono zakres zmiennej bajtowej. Nadmiarowe bity zostały odrzucone.

2. INVALID CHARACTER IN NUMERIC CONSTANT

Niedopuszczalny znak w stałej liczbowej.

3. OPERATION 'kod' NOT RECOGNIZED

Nie rozpoznano operacji 'kod'.

4. SYNTAX ERROR

Ogólny błąd składni.

5. DEFINITION STATEMENT - NAME EXPECTED

Rozkaz definiujący - brak nazwy w polu etykiety.

6. WRONG TYPE/NUMBER OF OPERANDS FOR THIS INSTRUCTION

Zły typ lub liczba operandów dla tego rozkazu.

7. TOO MANY OPERANDS

Zbyt wiele operandów dla tej dyrektywy.

8. TARGET OUT OF 2-KBYTE RANGE

Argument 10 bitowy i przekroczony zakres bloku 2 kB (np. przy acall, ajmp).

9. TARGET OUT OF +127/-127 RANGE

Przekroczony zakres argumentu krótkiego relatywnego (np w sjmp).

10. NAME/LABEL SYNTAX ERROR

Błąd w budowie nazwy lub etykiety (np. niedopuszczalny znak w obszarze etykiety).

11. ILLEGAL USE OF A RESERVED SYMBOL

Niedopuszczalne użycie symbolu zastrzeżonego (np. użycie nazwy rejestru jako etykiety).

12. 'symbol' - ALREADY DEFINED SYMBOL

Próba zdefiniowania już wcześniej zdefiniowanego symbolu.

13. INVALID SEPARATOR 'znak'

Użyto niewłaściwego znaku 'znak' do rozdzielenia pól lub argumentów.

14. INVALID BYTE BASE IN BIT ADDRESS EXPRESSION

W wyrażeniu bitowym jako adres bajtu wskazano wartość spoza obszaru adresowalnego bitowo.

15. OUT OF RANGE OR NON-TYPELESS BIT-OFFSET

W wyrażeniu bitowym jako numer bitu użyto wartości spoza dopuszczalnego zakresu lub posiadająca typ (np. adres w segmencie).

16. RESTRICTED SYMBOL IN EXPRESSION

Symbol zastrzeżony użyty w wyrażeniu.

17. DEFINITION STATEMENT EXPECTED

Oczekiwane wyrażenie definiujące (brak argumentu w dyrektywie definiującej).

18. PHASE ERROR (PASS-2)

Wartość zmieniła się pomiędzy przebiegami.

19. ')' EXPECTED

Brak zamykającego nawiasu.

20. MISSING FACTOR

Brak drugiego argumentu w wyrażeniu z operandem dwuargumentowym.

21. DIVIDE BY ZERO ERROR

Błąd dzielenia przez zero.

22. UNEXPECTED ‘)’

Nieoczekiwany nawias zamykający - brak nawiasu otwierającego.

23. UNRESOLVED EXPRESSION

Wartości wyrażenia nie udało się obliczyć - zwykle przyczyną jest brak definicji jakiegoś użytego symbolu.

24. STRING CONTAINSMORE THAN 2 CHARACTERS

W wyrażeniu (a nie w instrukcji inicjującej pamięć) użyto łańcucha dłuższego niż 2 bajty.

25. BAD INDIRECT REGISTER

Użyto niewłaściwego rejestru do adresacji pośredniej.

27. ILLEGAL FACTOR

Nielegalny składnik wyrażenia - w wyrażeniu arytmetycznym oczekuje się składnika (liczby lub operanda symbolicznego).

28. SYMBOL SHOULD BE DEFINED DURING PASS-1

Symbol powinien być zdefiniowany już w trakcie analizy składni.

29. VALUE HAS BEEN TRUNCATED 16 BITS

Wartość przekroczyła zakres liczby 16 bitowej.

30. ILLEGAL BACKWARD DEFINITION

Nielegalna definicja wstecz już użytego symbolu - wcześniej użyta „domyślna” wartość atrybutów symbolu koliduje z obecnie nadaną.

31. SEGMENT TYPE EXPECTED

Oczekiwano nazwy typu segmentu (CODE, DATA,.....).

32. BAD RELOCATION -TYPE

Podano niedopuszczalny typ relokacji segmentu (UNIT, PAGE, INPAGE, INBLOCK, BITADDRESSABLE) dla danego segmentu.

33. COMPILER TABLES OVERFLOW

Przekroczona pojemność tablic kompilatora (pamięci operacyjnej). Jest zalecana segmentacja programu na mniejsze moduły.

34. SEGMENT - SYMBOL EXPECTED

Operandem dyrektywy RSEG musi być nazwa segmentu.

35. EXPRESSION EXPECTED

Oczekiwano wyrażenia.

36. INSTRUCTION UNSUITABLE FOR THIS SEGMENT

Instrukcja/dyrektywa niedopuszczalna dla tego segmentu (np. dyrektywy inicjujące pamięć w obszarze pamięci DATA, XDATA).

37. BAD RELOCATABLE EXPRESSION

Nieprawidłowe wyrażenie relokowalne. Wyrażenie relokowalne (którego wartość jest obliczana przez linker) może zawierać tylko jeden symbol relokowalny (nazwy segmentów, ‘externale’, etykiety wewnątrz segmentu relokowalnego). Symboli tych nie można mnożyć ani dodawać. Dopuszczalne jest tylko dodawanie/odejmowanie wartości absolutnych oraz użycie operatorów LOW i HIGH.

38. CONFLICTING ADDRESS TYPE/SEGMENT IN EXPRESSION

Wykluczające się typy lub przynależności do segmentów argumentów wyrażenia (np. próba dodania adresu z DATA do adresu z CODE).

39. ILLEGAL NUMBER OF A REGISTER BANK

Numer banku rejestrów musi być z zakresu 0 -3.

40. NOR LABEL NOR NAME ARE PERMITTED

Dyrektywa wyklucza równoczesne użycie nazwy lub etykiety.

41. CONFLICTING ATTRIBUTES

Ustawiono sprzeczne atrybuty.

42. POSITIVE ABSOLUTE EXPRESSION EXPECTED

Oczekiwano nieujemnego wyrażenia arytmetycznego.

43. ‘>‘ EXPECTED

Brak makro-nawiasu zamykającego ciąg specjalny wywołania makrorozkazu.

44. 'ELSE', 'ELSEIF', OR 'ENDIF' WITHOUT MATCHING 'IF'

Użyto komendy ELSE, ELSEIF, lub ENDIF bez otwarcia kompilacji warunkowej instrukcją IF.

45. 'ELSE' OR 'ELSEIF' AFTER 'ELSE'

Po ELSE nie można precyzować dalszych rozgałęzień programu na tym poziomie kompilacji warunkowej.

46. 'LOCAL' OUTSIDE A MACRO

Dyrektywę LOCAL wolno używać tylko wewnątrz makrodefinicji.

47. ADDRESS BELOW LOWER LIMIT

Dyrektywa ORG nie może ustawiać adresu inicjacji pamięci poniżej wartości zadeklarowanej za dyrektywy AT.

48. CAN'T CREATE OBJECT FILE nazwa zbioru

Kompilator nie potrafi utworzyć zbioru kompilacji pośredniej.

C2. Komunikaty MONITORA-51

1. ERROR * BAD COMMAND**

Podano nieznaną komendę.

2. ERROR * BAD DIGIT**

Podana liczba zawiera nieważną cyfrę.

3. ERROR * MISSING PARAMETER**

Komenda oczekuje na parametr, który nie został podany.

4. ERROR * LIMIT EXPECTED**

Próba zdefiniowania więcej niż 10 punktów wstrzymań.

5. ERROR * BAD HEX RECORD**

Podany rekord w formacie INTEL - HEX zawiera błędny typ rekordu lub niewłaściwą sumę kontrolną.

6. ERROR * NO CODE MEMORY AT ADDRESS: XXXX**

Nie można zapisać bajtu pod wskazanym adresem w zakresie pamięci programu.

7. ERROR * CONFLICT BREAKEADDRESS**

Podany adres punktu wstrzymań koliduje z wcześniej zdefiniowanym punktem wstrzymań.

8. ERROR * NOT EXIST**

Podany punkt wstrzymań nie występuje.

9. ERROR * OUT OF RANGE**

Podany numer punktu wstrzymań leży w zakresie 0..9; lub adres docelowy rozkazu skoku znajduje się poza możliwym zakresem pamięci.

10. ERROR * BAD ARGUMENT**

Podany argument na danej pozycji jest niedopuszczalny lub nieznany.

11. ERROR * WRONG REGISTER**

Podany rejestr na danej pozycji jest niedopuszczalny lub nieznany.

12. ERROR * NUMBER OF PARAETERS**

Podany rozkaz jest niepełny lub podano zbyt wiele parametrów.

13. ERROR * REGISTER EXPECTED**

Podany rozkaz oczekuje nazwy rejestru.

14. ERROR * DATA - ADDRESS EXPECTED**

Podany rozkaz oczekuje adresu danych.

15. ERROR * BIT - ADDRESS EXPECTED**

Podany rozkaz oczekuje adresu bitowego

16. ERROR * DELIMITER EXPECTED**

Nie podano ograniczenia

17. ERROR * # - VALUE EXPECTED**

Nie podano stałej.