Mikrokontroler Motorola M68HC05

Technika mikroprocesorowa

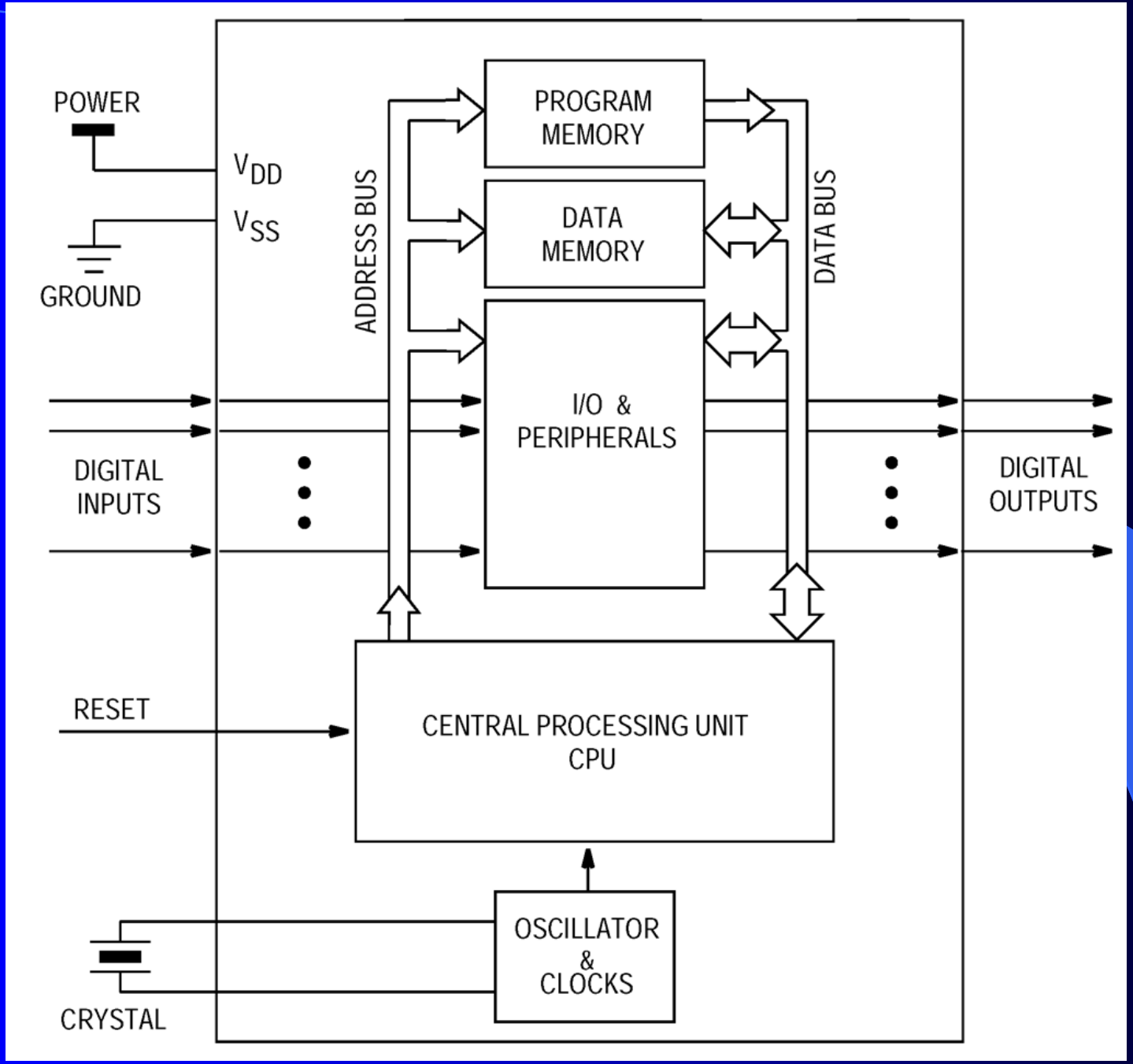
dr inż. Wiesław Madej

Mikrokontroler MC68HC05 jest pełnym systemem komputerowym mieszczącym się w pojedynczym układzie scalonym. W większości zastosowań aplikacji sterujących może funkcjonować bez dodatkowych układów zewnętrznych. W związku z tym zmniejszają się koszty produkcji, co przyczynia się do wysokiego stopnia elastyczności, umożliwia to wyprodukowanie układów z przeznaczeniem do prostych zastosowań inżynierskich. Nowe właściwości mikrokontrolerów MC68HC05 znajdują zastosowanie u wielu doświadczonych inżynierów, którzy potrzebują rodziny układów z funkcjami i aplikacjami zgodnymi ze starszymi modelami mikrokontrolerów firmy Motorola. Technologia HCMOS wykorzystana w tych mikrokontrolerach charakteryzuje się małymi wymiarami, dużą szybkością działania i małym poborem mocy oraz wysoką odpornością układów CMOS. Mikrokontrolery MC68HC05 mogą być sterowane przez różne częstotliwości zegara od jak najmniejszych wartości po wartości maksymalne. Ta właściwość może być wykorzystana przy zmniejszeniu poboru mocy w układach pracujących na wysokich częstotliwościach zegara. Układ posiada dwa kontrolery programowe zarządzające zmniejszeniem poboru mocy WAIT i STOP.

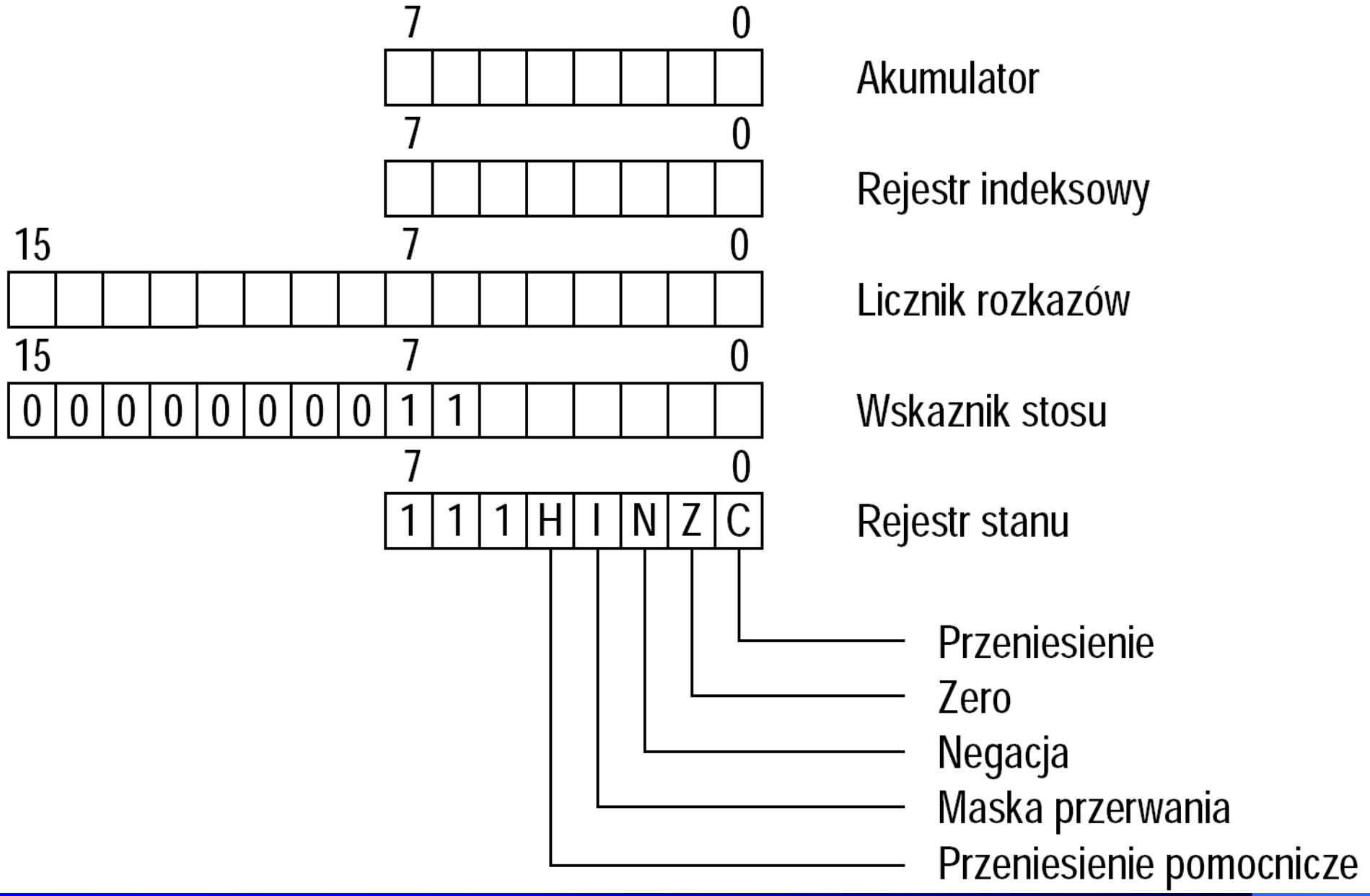
Moduły te mogą być atrakcyjne dla aplikacji zarządzających energią.

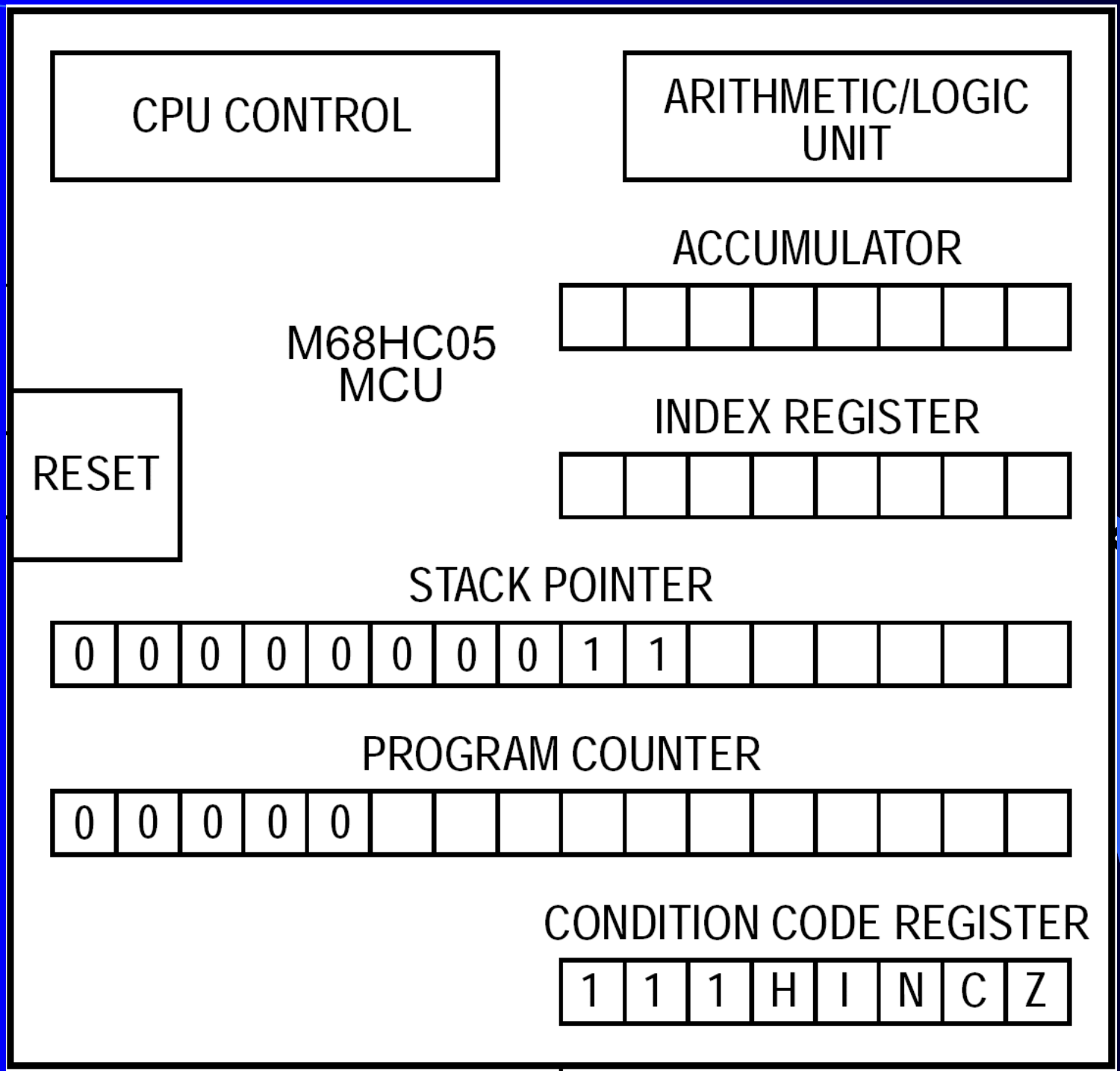
Wybrane cechy mikrokontrolera MC68HC05:

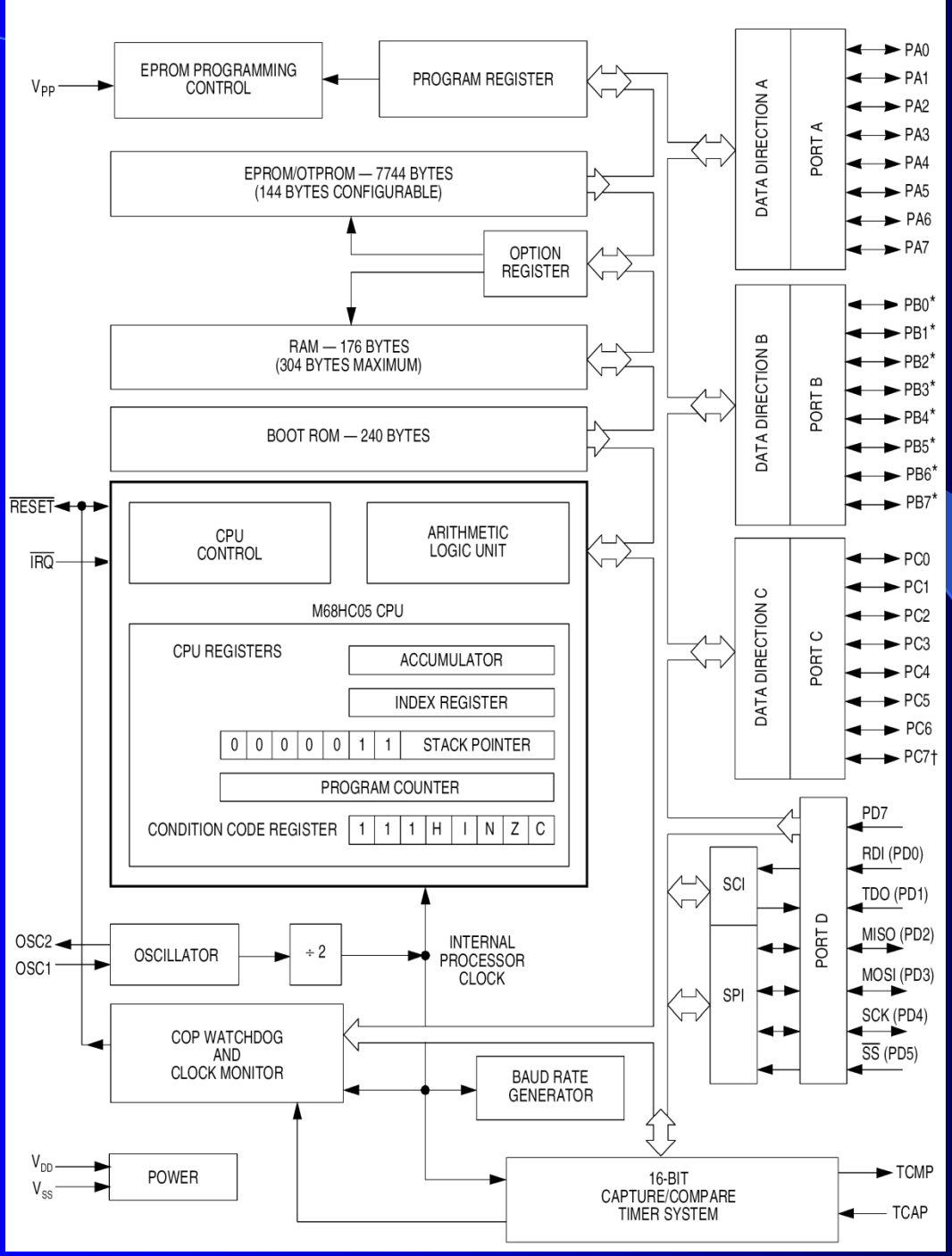
- mały koszt produkcji, wykonanie w technologii HCMOS,
- mikrokontroler oparty o architekturę 8-bitową,
- dwa tryby zarządzanie energią,
- 24 linie dwukierunkowe I/O,
- 7 linii pracujących w trybie odczytu,
- dwa wyprowadzenia czasowe,
- częstotliwość operacyjna 2 MHz przy 5V; 1 MHz przy 3V,
- 16 bitowy licznik,
- szeregowy, asynchroniczny i synchroniczny interfejs komunikacyjny,
- jednorazowa pamięć ROM lub kasowalna pamięć promieniami UV,
- wybieralna konfiguracja pamięci,
- układ kontroli WATCHDOG,
- oscylator wewnątrz układu,
- 1240 bajtów wewnętrznej pamięci programu,
- 64 bajtowy obszar pamięci RAM,
- programy kompatybilne z procesorami rodziny CMOS M146805,
- możliwość zmiany poszczególnych bitów,
- mapowanie portów I/O jako pamięć,


POWER
V_{DD}
V_{SS}
GROUND
ADDRESS BUS
PROGRAM MEMORY
DATA MEMORY
I/O & PERIPHERALS
DATA BUS
DIGITAL INPUTS
DIGITAL OUTPUTS
RESET
CENTRAL PROCESSING UNIT CPU
OSCILLATOR & CLOCKS
CRYSTAL

Jednostka centralna mikrokontrolera, składa się z 8-bitowego bloku arytmetyczno-logicznego: akumulatora, rejestru indeksowego, licznika rozkazów, wskaźnika stosu, układu sterowania CPU. W mikrokontrolerze MC68HC05 znajdują się pamięci: stała – ROM (EPROM lub OTPROM) i pamięć operacyjna – RAM. Do wyprowadzeń układu dołączone są porty wejścia – wyjścia, asynchroniczny interfejs komunikacyjny SCI, porty separowane SPI, 16–bitowy wewnętrzny programowalny licznik czasowy. Układ ma możliwość samokontroli przeciwko błędom systemowym, dlatego posiada układ WATCHDOG zabezpieczający przed błędami programowymi. Monitor zegara generuje sygnał do systemu o wznowienie pracy, jeśli system się zagubił lub chodzi za wolno. Przy wykrytym niewłaściwy kodzie wysyła nie maskowane przerwanie.


7
0
Akumulator
7
0
Rejestr indeksowy
15
7
0
Licznik rozkazów
15
7
0
0 0 0 0 0 0 0 0 1 1
Wskaznik stosu
7
0
1 1 1 H I N Z C
Rejestr stanu
Przeniesienie
Zero
Negacja
Maska przerwania
Przeniesienie pomocnicze


CPU CONTROL
ARITHMETIC/LOGIC UNIT
ACCUMULATOR
M68HC05 MCU
RESET
INDEX REGISTER
STACK POINTER
0 0 0 0 0 0 0 0 1 1
PROGRAM COUNTER
0 0 0 0 0
CONDITION CODE REGISTER
1 1 1 H I N C Z


EPROM PROGRAMMING CONTROL
PROGRAM REGISTER
EPROM/OTPROM — 7744 BYTES (144 BYTES CONFIGURABLE)
OPTION REGISTER
RAM — 176 BYTES (304 BYTES MAXIMUM)
BOOT ROM — 240 BYTES
CPU CONTROL
ARITHMETIC LOGIC UNIT
M68HC05 CPU
CPU REGISTERS
ACCUMULATOR
INDEX REGISTER
STACK POINTER
PROGRAM COUNTER
CONDITION CODE REGISTER
OSCILLATOR
÷ 2
INTERNAL PROCESSOR CLOCK
COP WATCHDOG AND CLOCK MONITOR
BAUD RATE GENERATOR
POWER
16-BIT CAPTURE/COMPARE TIMER SYSTEM
DATA DIRECTION A
PORT A
DATA DIRECTION B
PORT B
DATA DIRECTION C
PORT C
PORT D
SCI
SPI
RESET
IRQ
OSC2
OSC1
VPP
VDD
VSS
PA0
PA1
PA2
PA3
PA4
PA5
PA6
PA7
PB0*
PB1*
PB2*
PB3*
PB4*
PB5*
PB6*
PB7*
PC0
PC1
PC2
PC3
PC4
PC5
PC6
PC7†
PD7
RDI (PD0)
TDO (PD1)
MISO (PD2)
MOSI (PD3)
SCK (PD4)
SS (PD5)
TCMP
TCAP

Piny i wyprowadzenia

Ponieważ mikrokontroler MC68HC0705 jest wykonany w technologii CMOS, nieużywane końcówki mogą być źródłem przypadkowych oscylacji na tych wyprowadzeniach. Preferowaną metodą eliminacji zakłóceń jest podłączenie do tych końcówek stanów wysokich lub niskich.

V_{DD} i V_{SS}

Zasilanie procesora jest podawane przez te dwa wyprowadzenia, z czego na wyprowadzenie V_{DD} jest podawane napięcie dodatnie, a wyprowadzenie V_{SS} jest dołączone do masy zasilania. Mikrokontroler może być zasilany pojedynczym napięciem +5V.

V_{PP}

Wyprowadzenie V_{PP} jest używane do jednorazowego programowania pamięci ROM, OTPROM, EPROM. Napięcie programowania wynosi 14,75 V_{DC} i jest akceptowane przy programowaniu pamięci. Normalnie pin (końcówka) ten jest podłączony do V_{DD}. Zabrania się podłączania tego pinu do zasilania V_{SS}, gdyż grozi to uszkodzeniem układu.

IRQ

Jest to wyprowadzenie przyjmujące zewnętrzny sygnał przerwania zgłoszony na to wejście. Zewnętrzne przerwanie jest nieobsługiwane, gdy w rejestrze CCR bit I jest ustawiony na „1”, ale sygnał zgłoszenia przerwania jest zapamiętany w zatrzasku wejściowym tego wyprowadzenia.

RESET

Wejście to jest aktywowane stanem logicznym "0", pozwala ręcznie wyzerować cały układ. Zewnętrzny obwód RC może być podłączony do końcówki generującej sygnał „reset”. W chwili włączenia zasilania (ang. power- on- reset), system opóźnia start o 4064 cykle, aby obwód oscylatora mógł się ustabilizować.

OSC1, OSC2

Mikrokontroler MC68HC05 ma możliwość dołączenia do wyprowadzeń układu kilku rodzajów źródeł częstotliwości. Istnieje możliwość korzystania z zewnętrznego źródła częstotliwości. Częstotliwość zegara procesora jest otrzymywana przez podział częstotliwości oscylatora na dwa. Wewnętrzny oscylator jest zaprojektowany jako wejście do podłączenia równoległego rezonatora kwarcowego lub ceramicznego o maksymalnej częstotliwości 4MHz. Kryształ i jego komponenty powinny znajdować się jak najbliżej układu w celu eliminacji pojemności pasożytniczych. W przypadku dołączenia zewnętrznego zegara należy podłączyć go do wyjścia OSC1, a wyprowadzenie OSC2 pozostawić nie podłączone.

PA7-PA0, PB7-PB0, PC7-PC0

Wyprowadzenia PA7-PA0, PB7-PB0, PC7-PC0 tworzą porty A, B i C. Są to ośmioliniowe wejścia- wyjścia ogólnego przeznaczenia. Każda linia portu może być ustawiona programowo jako wejście lub wyjście, w zależności od wpisanych wartości w rejestrach DDRA, DDRB, DDRC.

PD5-PD0, PD7

Wyprowadzenia PD7, PD5-PD0 tworzą port D. Port ten jest mapowany w mapie pamięci i jest on ustawiony jako wejście tylko do odczytu, może też być wykorzystany jako szeregowy port komunikacyjny synchroniczny i asynchroniczny.

TCAP, TCMP

Wyprowadzenie oznaczone jako TCAP jest wejściem sygnałowym zgłaszającym żądanie obsługi licznika wejściowego. Wyjście układu oznaczone jako TCMP jest wyprowadzeniem sygnałowym komparatora porównującego zawartość rejestrów OCR z aktualną wartością licznika. W chwili wystąpienia zgodności zawartości wyżej wymienionych rejestrów może zostać wystawiony stan logiczny na końcówce TCMP.

Moc obliczeniowa komputerów jest zależna do dostępu pamięci. Tryby adresacji procesora definiują, w jaki sposób dane zostaną dostarczone do procesora. Z powodu różnych trybów adresacji instrukcje mogą mieć różny dostęp do operandów. Mikrokontrolery MC68HC05 posiadają 62 instrukcje podstawowe; z czego każda instrukcja wymaga kodu operacji. W zależności od trybu adresacji mamy 210 odrębnych kodów operacji. Mikrokontroler MC68HC05 używa 6 trybów adresacji przy odwoływaniu się do pamięci.

Tryby adresowania:

- rejestrowe,
- natychmiastowe,
- bezpośrednie strony 0,
- rozszerzone,
- indeksowe (bez offsetu),
- indeksowe (z 8-bitowym offsetem),
- indeksowe (z 16-bitowym offsetem),
- względne.

W podstawowej wersji tego mikrokontrolera wszystkie zmienne programu i rejestry wejścia-wyjścia mieszczą się w pamięci pod adresem $0000 do $00FF, także używa się bezpośredniego trybu adresacji.

Tryb Adresowania	**Skrót**	**Argument**
Rejestrowe	**INH**	*brak*
Natychmiastowe	**IMM**	ii
Bezpośrednie strony 0	**DIR**	dd
(Z testem bitu)		dd rr
Rozszerzone	**EXT**	hh ll
Indeksowe (bez offsetu)	**IX**	*brak*
Indeksowe (z 8-bitowym offsetem)	**IX1**	ff
Indeksowe (z 16-bitowym offsetem)	**IX2**	ee ff
Względne	**REL**	rr

dd — wartość 8-bitowa dla pierwszych 256 komórek z przedziału od $0000 do $00FF.

ee — starsza 8-bitowa wartość adresu 16-bitowego.

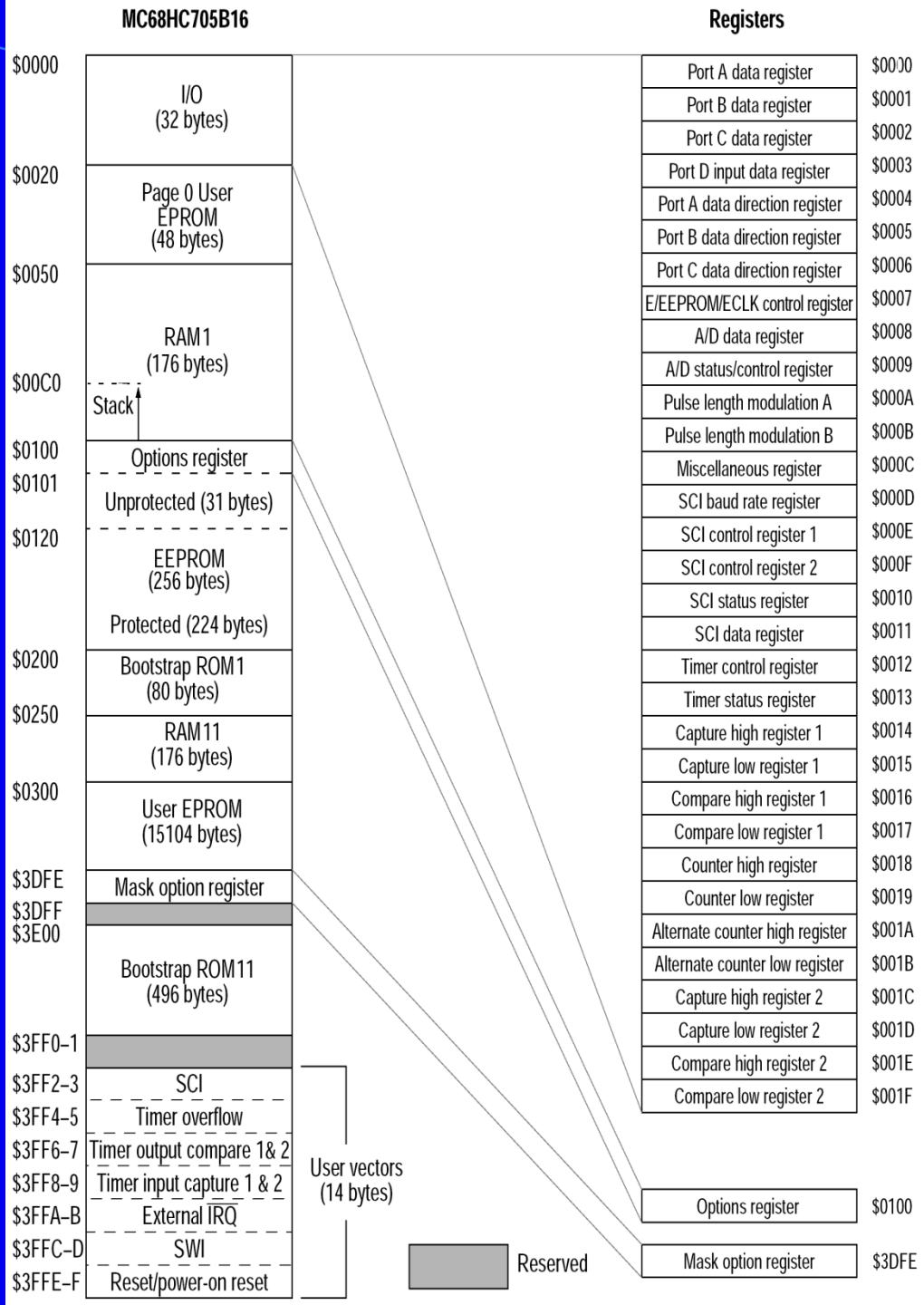
ff — młodsza 8-bitowa wartość adresu 16-bitowego lub adres 8-bitowy.

ii — wartość 8-bitowa przy adresowaniu bezpośrednim.

hh — starsza 8-bitowa wartość adresu 16-bitowego adresowania rozszerzonego.

ll — młodsza 8-bitowa wartość adresu 16-bitowego adresowania rozszerzonego.

rr — wartość 8-bitowa adresu względnego.


MC68HC705B16
Registers
$0000
I/O
(32 bytes)
$0020
Page 0 User
EPROM
(48 bytes)
$0050
RAM1
(176 bytes)
$00C0
Stack
$0100
Options register
$0101
Unprotected (31 bytes)
$0120
EEPROM
(256 bytes)
Protected (224 bytes)
$0200
Bootstrap ROM1
(80 bytes)
$0250
RAM11
(176 bytes)
$0300
User EPROM
(15104 bytes)
$3DFE
Mask option register
$3DFF
$3E00
Bootstrap ROM11
(496 bytes)
$3FF0–1
$3FF2–3
SCI
$3FF4–5
Timer overflow
$3FF6–7
Timer output compare 1& 2
$3FF8–9
Timer input capture 1 & 2
$3FFA–B
External IRQ
$3FFC–D
SWI
$3FFE–F
Reset/power-on reset
User vectors
(14 bytes)
Reserved
Port A data register $0000
Port B data register $0001
Port C data register $0002
Port D input data register $0003
Port A data direction register $0004
Port B data direction register $0005
Port C data direction register $0006
E/EEPROM/ECLK control register $0007
A/D data register $0008
A/D status/control register $0009
Pulse length modulation A $000A
Pulse length modulation B $000B
Miscellaneous register $000C
SCI baud rate register $000D
SCI control register 1 $000E
SCI control register 2 $000F
SCI status register $0010
SCI data register $0011
Timer control register $0012
Timer status register $0013
Capture high register 1 $0014
Capture low register 1 $0015
Compare high register 1 $0016
Compare low register 1 $0017
Counter high register $0018
Counter low register $0019
Alternate counter high register $001A
Alternate counter low register $001B
Capture high register 2 $001C
Capture low register 2 $001D
Compare high register 2 $001E
Compare low register 2 $001F
Options register $0100
Mask option register $3DFE

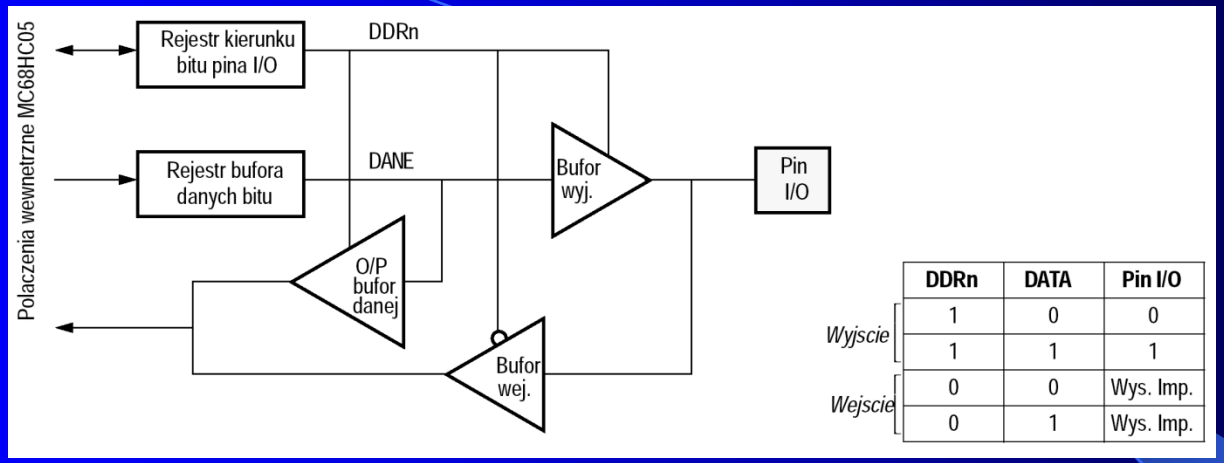
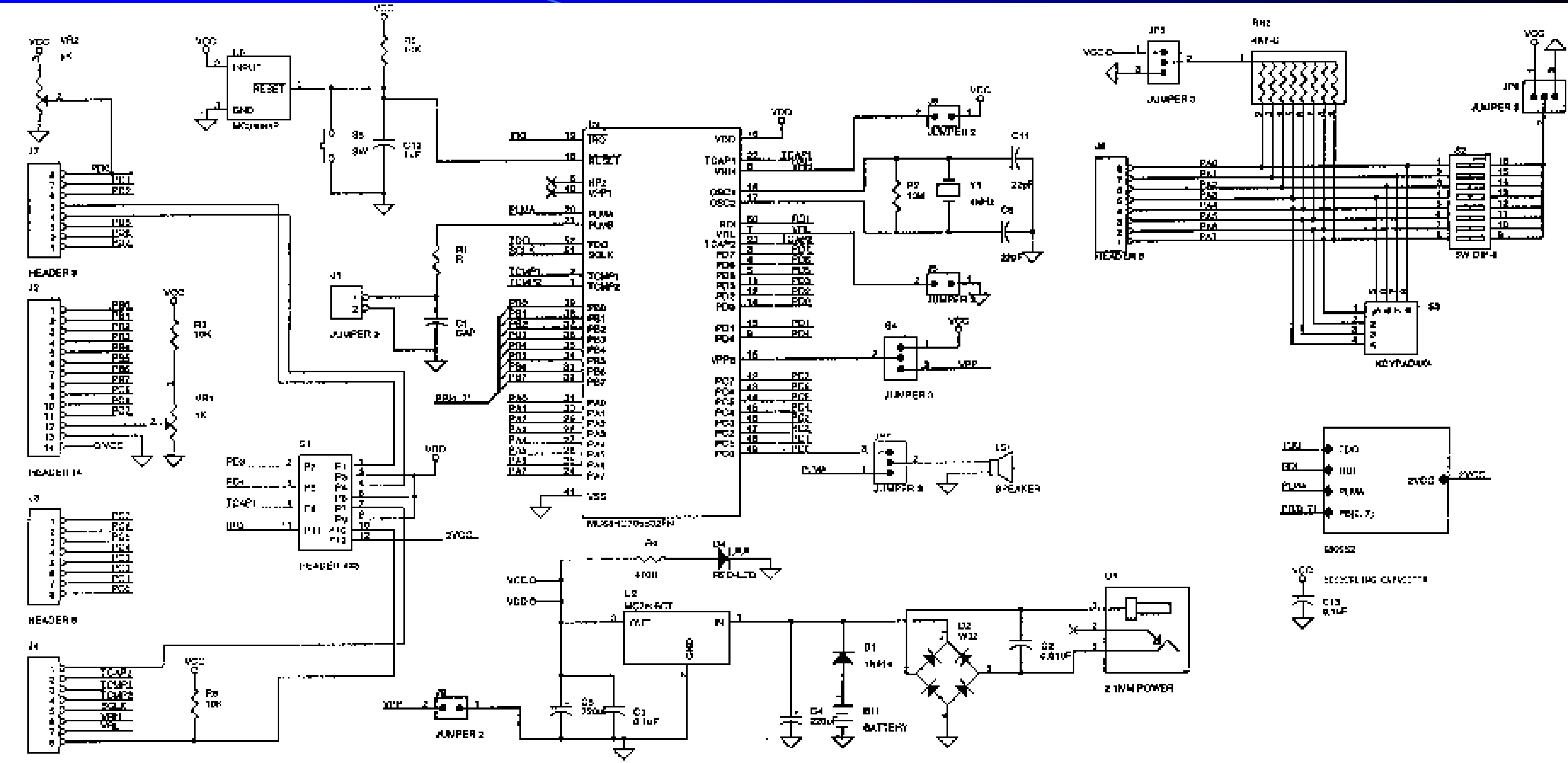
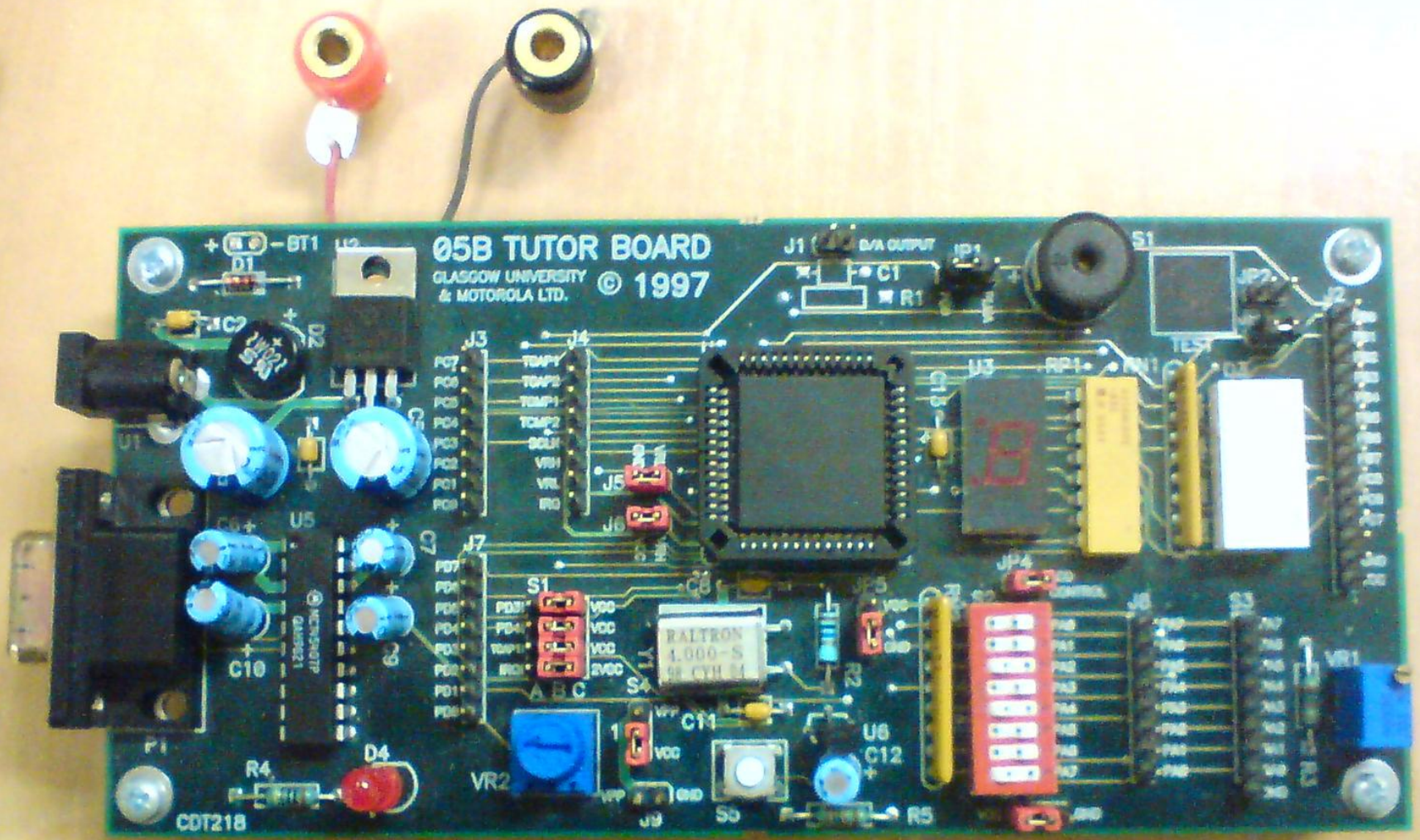


Tabela 2.5.2 Określenie kierunku przesyłania danych przez Port A.

R/W	DDRn	Działanie MCU zapis do/odczyt z bitu danych
0	0	Pin I/O jest w trybie wejścia. Dane są zapisane do bufora wyjściowego.
0	1	Dana jest zapisana do bufora wyjściowego i jest wystawiana na pin I/O.
1	0	Zawartość pinu I/O jest odczytywana.
1	1	Pin I/O jest w trybie wyjścia. Dane z bufora wyjściowego są odczytywane



RESET
HEADER 8
HEADER 14
JUMPER 2
SPEAKER
KEYPAD4X4
SW DIP-8
BATTERY
RED LED
2.1MM POWER


05B TUTOR BOARD
GLASGOW UNIVERSITY & MOTOROLA LTD.
© 1997
CDT218

Do napisania programu możemy użyć dowolnego edytora tekstu np. posłużyć się notatnikiem. Pisząc program należy pamiętać jak jest zbudowana linia asemblera. Pola linii są oddzielone znakami białymi (spacja, tabulacja). Wewnątrz pola znaki te nie występują. Linia programu zawiera cztery pola, kolumny, z których niektóre z nich mogą zostać puste: etykieta, mnemonik, operand i komentarz. Wartość może występować jako symbol, stała, lub znak '*'.

Operatory.

Oznaczenia stosowane przy pisaniu programu:

#	wartość (adresowanie natychmiastowe)
+	dodawanie
-	odejmowanie
*	mnożenie
⊕	funkcja EX-OR

Stała może być zapisana w kodzie asemblera na pięć różnych formatów: decymalnie, binarnie, heksadecymalnie, oktalnie, lub jako znak ASCII. Programista określa wartość w jakim formacie jest podana za pomocą następujących przedrostków:

!	DZIESIĘTNIE
$	HEKSADECYMALNIE
%	BINARNIE
@	OKTALNIE
'	ASCII

!27 lub 27T = zapis liczby 27 dziesiętnie
$27 lub 27H = zapis liczby 39 heksadecymalnie
%00001010 = zapis liczby 10 binarnie

Asembler konwertuje wszystkie stałe na format binarny kodu maszynowego i wyświetla stałe w listingu asemblera w postaci heksadecymalnej. Bez przedrostka są zapisywane wartości dziesiętnie.

Oznaczenia dyrektyw stosowane przy pisaniu programu:

*	- linia komentarza ignorowana przez kompilator
;	- znaki za znakiem są traktowane jako komentarz

CSEG AT - określa adres początku segmentu rozkazów w pamięci programu.
DB - umieszcza w kodzie programu jedną lub ciąg 8-bitowych danych. Wymienione w ciągu dane zostaną umieszczone w tej samej kolejności w pamięci programu bajt po bajcie.

DS - rezerwuje wyspecyfikowaną ilość bajtów w pamięci programu. Rozmiar segmentu danych jest wyrażeniem, na które mogą składać się *liczby*, wcześniej deklarowane *stałe* lub kody znaków ujętych w cudzysłów. Nazwie stałej zostaje przypisany adres początku segmentu danych. Wystąpienie etykiety nie jest wymagane.

DSEG AT - określa adres początku segmentu danych w pamięci programu.

DW - umieszcza w kodzie programu jedną lub ciąg 16-bitowych danych. Wymienione w ciągu dane zostaną umieszczone w tej samej kolejności w pamięci programu słowo po słowie. Elementami ciągu mogą być *liczby*, *stałe* lub kody znaków ujętych w cudzysłów. Po ostatnim elemencie ciągu nie powinno być przecinka. Wystąpienie etykiety nie jest wymagane.

END - kończy zapis tekstu programu, wszystkie występujące po niej znaki i rozkazy są ignorowane.

ENDM - kończy zapis makroinstrukcji. Każde wywołanie nazwy makroinstrukcji powoduje wstawienie kodu rozkazów w niej umieszczonych. Wywołanie makroinstrukcji może wystąpić dopiero po jej deklaracji.

EQU - służy do deklarowania nazw stałych.

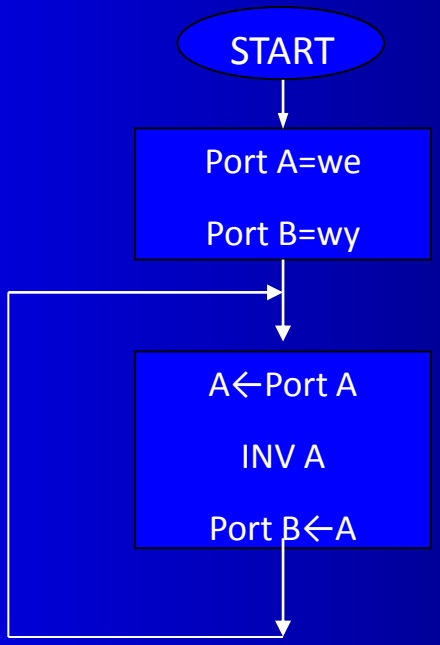
$INCLUDE - włącza do tekstu programu wyspecyfikowany plik.

MACRO - rozpoczyna zapis *makroinstrukcji*.

ORG - określa adres początku segmentu rozkazów lub danych w pamięci programu.

Przykładowe programy:

Pierwszy z przykładowych programów wykorzystuje przełączniki SW i wyświetlacz siedmiosegmentowy. Program odczytuje stan przełączników i wyświetla jakie pozycje zostały włączone. Nie jest to zbytnio czytelne, ale w trakcie pracy zmieniając ustawienia przełączników zmieniają nam się zapalane segmenty na wyświetlaczu. Programem realizującym tą funkcję jest program ‘inout.asm’. Wykorzystując mikrokontroler MC68HC05 należy określić kierunek pracy portów A i B, oraz określić, w którym miejscu pamięci ma się znajdować kod programu.

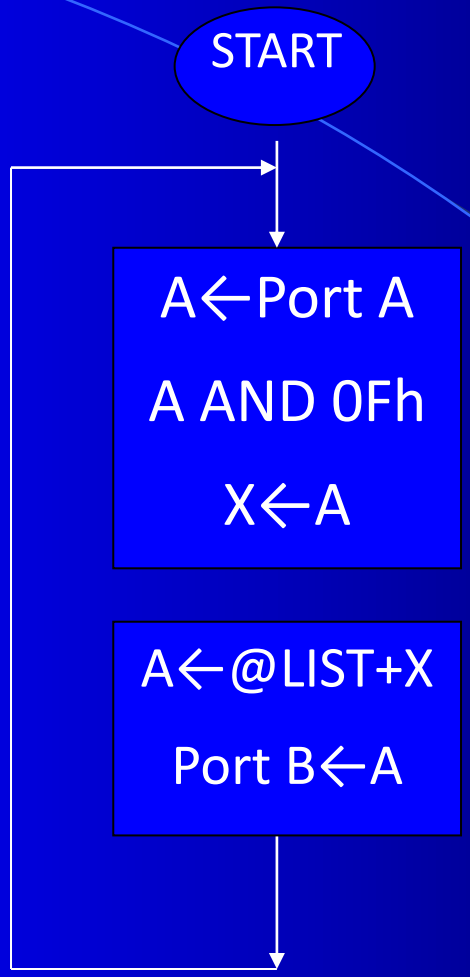


```
*************************************************************************
*Program 1 INOUT.asm - odczytuje wartość z przełączników (Port A)       *
*i wyświetla na wyświetlaczu LED (Port B)                               *
*Na module JP2 zwarty, JP4 opcja, JP5 masa, JP6 Zasianie                *
*************************************************************************
*Definicja Adresów Portów*
PORTA    EQU     0                ;Port A (dołączone są przełączniki)
PORTB    EQU     1                ;Port B (dołączone wyświetlacze Led)
DDRA     EQU     4                ;Rejestr konfigurujący rodzaju pracy portu A
DDRB     EQU     5                ;Rejestr konfigurujący rodzaju pracy portu B
         ORG     $250             ;Wskazanie miejsca początku programu w pamięci
START:   LDA     #0               ;Zapisz wartość 0 do akumulatora A
         STA     DDRA             ;Zapisz wartość z Akumulatora do DDRA
                                  ;Zostaje skonfigurowany Port A  do odczytu
         LDA     #$FF             ;Zapisz wartość 0 do akumulatora A
         STA     DDRB             ;Zapisz wartość z Akumulatora do DDRB
                                  ;Zostaje skonfigurowany Port B jako tylko do
                                          zapisu
LOOP:    LDA     PORTA            ;Odczytaj wartość z Portu A i zapisz do Ak. A
         COMA                     ;Dokonuje inwersji wszystkich bitów w Ak. A
         STA     PORTB            ;Zapisz wartość z Akumulatora A do portu B
         BRA     LOOP             ;Skocz do etykiety LOOP
         END
```

Program - adresowanie indeksowe

Program odczytuje binarną wartość na przełączniku i zamienia ją na wartość wyświetlacza siedmiosegmentowego. Jest to wartość o wiele bardziej czytelna niż wyniki prezentowane w poprzednim programie. W układzie jest jeden wyświetlacz siedmiosegmentowy i może wyświetlić wartości heksadecymalnie od 0 do F. Z przełącznika będziemy uwzględniać wartość tylko z czterech linii.

Program możemy podzielić na dwie części: pierwsza odczytująca wartość z przełącznika, a druga dekodująca z kodu binarnego na siedmiosegmentowy z wykorzystaniem adresowania indeksowego. W celu odrzucenia starszej części portu A użyjemy iloczynu logicznego AND z maską bitów $0F. Chcąc dokonać dekodowania kodu binarnego na siedmiosegmentowy najlepiej użyć 16 bajtów pamięci podając wartości odpowiadające danym wartościom w kolejności od 0 do F. W momencie pobrania wartości należy jedynie dodać wartość początku tablicy z kodami i pobrać ją, następnie wystawić na port B, gdzie jest wyświetlacz. Do tego celu znakomicie można wykorzystać adresowanie indeksowe.



Algorytm programu SEVENSEG.ASM

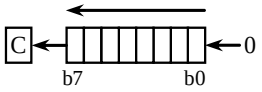
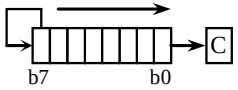
```
*Program  SEVENSEG.ASM  odczytuje młodsze 4 linie przełącznika A       *
* i zapala zdekodowaną informację na wyświetlaczu LED na porcie B  *
* Uses indexed addressing mode to access segment codes                 *
* Hardware JP2 Optional, JP4 In, JP5 ground, JP6 Vcc                      *
*************************************************************************
PORTA     EQU       0          ;Definiuje adresy dla portu A i B
PORTB     EQU       1
DDRA      EQU       4           ;Określenie kierunków pracy portów
DDRB      EQU       5
*
*
          org       $250       ;początek programu w pamięci
START:
          lda       #$0        ; ustaw Port A
          sta       DDRA       ; jako wejście
          lda       #$FF       ; ustaw port B
          sta       DDRB       ; jako wyjście
LOOP:
          lda       PORTA      ; odczyt klawiszy
          and       #$0F       ; wyzeruj 4 starsze bity
          tax                  ; zapisz do X 4 młodsze bity portu A
          lda       LIST,X     ; odczytaj kod znaku LED z LIST
          sta       PORTB      ; wystaw wartość zdekodowaną na port B
          bra       LOOP       ; Powrót do LOOP
*
* Kody dla wyświetlacza 7 segmentowego, 0 zapala segment
*
LIST:     fcb       %11000000,%11111001,%10100100,%10110000; '0', '1', '2', '3'
          fcb       %10011001,%10010010,%10000010,%11111000;'4', '5', '6', '7
          fcb       %10000000,%10010000,%10001000,%10000011; '8', '9', 'A', 'b'
          fcb       %11000110,%10100001,%10000110,%10001110;'C' , 'd' , 'E' , 'F‘
```

MC 68HC05

OBSŁUGA GŁOŚNICZKA

```
*Definicja Adresów Portów*
PORTC    EQU      2          ;Port C (dołączony głośniczek)
DDRC     EQU      6          ;Rejestr konfigurujący rodzaju pracy portu C
OPOZN    EQU      $50
   ORG   $250                ;Wskazanie miejsca początku programu w pamięci
START:   LDA      #$FF
   STA   DDRC                ;Zapisz wartość z Akumulatora do DDRC
LOOP:
   LDA   #1                  ;wpisz do Ak. A wartość 1
   sta PORTC
   LDA OPOZN                 ;wpisz opóźnienie do A
P1:DECA
   BNE P1
   LDA   #0                  ;wpisz do Ak. A wartość 1
   sta PORTC
   LDA OPOZN                 ;wpisz opóźnienie do A
P2:DECA
   BNE P2
   BRA    LOOP               ;Skocz do etykiety LOOP
   END
```

Skrócona lista rozkazów mikrokontrolera MC68HC05

Mnemonik	Operacja	Deskrypcja	Typ. Adres.	Kod Maszynowy		Cyk.	Znaczniki				
				Kod	Arg.		H	I	N	Z	C
ADC opr.	Dodawanie, uwzględnia znak przeniesienia (dodaje), dodawanie liczb > niż 8 bit.	A ← (A) + (M) + C	IMM DIR EXT IX2 IX1 IX	A9 B9 C9 D9 E9 F9	ii dd hh ll ee ff ff	2 3 4 5 4 3	Δ	-	Δ	Δ	Δ
ADD opr.	Dodawanie bez przeniesienia do akumulatora	A ← (A) + (M)	IMM DIR EXT IX2 IX1 IX	AB BB CB DB EB FB	ii dd hh ll ee ff ff	2 3 4 5 4 3	Δ	-	Δ	Δ	Δ
AND opr.	Iloczyn logiczny (akum. *komórka pamięci)	A ← (A) • (M)	IMM DIR EXT IX2 IX1 IX	A4 B4 C4 D4 E4 F4	ii dd hh ll ee ff ff	2 3 4 5 4 3	-	-	Δ	Δ	-
ASL opr. ASLA ASLX ASL opr. ASL opr.	Arytmetyczne przesunięcie w lewo		DIR INH INK IXl IX	38 48 58 68 78	dd ff	5 3 3 6 5	-	-	Δ	Δ	Δ
ASR opr. ASRA ASRX ASR opr. ASR opr.	Arytmetyczne przesunięcie w prawo		DIR INH INH IX1 IX	37 47 57 67 77	dd ff	5 3 3 6 5	-	-	Δ	Δ	Δ

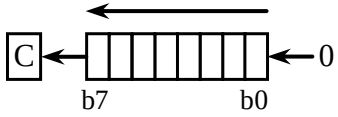
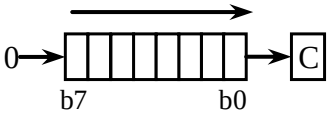
BCC rel.	Skocz, gdy C = 0	? C = 0	REL	24	rr	3	-	-	-	-	-
BCLR n,opr.	Zeruj bit w pamięci.	Mn ← 0	DIR b0 DIR bl DIR b2 DIR b3 DIR b4 DIR b5 DIR b6 DIR b7	11 13 15 17 19 1B 1D 1F	dd dd dd dd dd dd dd dd	5 5 5 5 5 5 5 5	-	-	-	-	-
BCS rel.	Skocz, gdy C =1	? C = 1	REL	25	rr	3	-	-	-	-	-
BEQ rel.	Skocz, gdy Z =1	? Z = 1	REL	27	rr	3	-	-	-	-	-
BHCC rel.	Skocz, gdy H = 0	? H = 0	REL	28	rr	3	-	-	-	-	-
BHCS rel.	Skocz, gdy H = 1	? H = 1	REL	29	rr	3	-	-	-	-	-
BHI rel.	Skocz, gdy C=0 i Z=0	? C+Z = 0	REL	22	rr	3	-	-	-	-	-
BHS rel.	Skocz, gdy większy, równy	? C = 0	REL	24	rr	3	-	-	-	-	-
BIH rel.	Skocz, gdy $\overline{\text{IRQ}}$ Pin =1	? $\overline{\text{IRQ}}$ Stan wysoki	REL	2F	rr	3	-	-	-	-	-
BIL rel.	Skocz, gdy $\overline{\text{IRQ}}$ Pin = 0	? $\overline{\text{IRQ}}$ Stan niski	REL	2E	rr	3	-	-	-	-	-

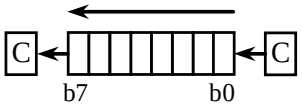
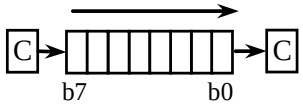
BIT opr.	Testuj A z pamięcią	(A) • (M)	IMM DIR EXT IX2 IX1 IX	A5 B5 C5 D5 E5 F5	ii dd hh ll ee ff ff	2 3 4 5 4 3	-	-	Δ	Δ	-
BLO rel.	Skocz, gdy mniejszy	? C = 1	REL	25	rr	3	-	-	-	-	-
BLS rel.	Skocz, gdy mniejszy, równy	? C+Z= 1	REL	23	rr	3	-	-	-	-	-
BMC rel.	Skocz, gdy I = 0	? I = 0	REL	2C	rr	3	-	-	-	-	-
BMI rel.	Skocz, gdy ujemne	? N= 1	REL	2B	rr	3	-	-	-	-	-
BMS rel.	Skocz, gdy I = 1	? I = 1	REL	2D	rr	3	-	-	-	-	-
BNE rel.	Skocz, gdy różny	? Z = 0	REL	26	rr	3	-	-	-	-	-
BPL rel.	Skocz, gdy dodatnie	? N = 0	REL	2A	rr	3	-	-	-	-	-
BRA rel.	Skocz zawsze	? 1 = 1 (zawsze prawda)	REL	20	rr	3	-	-	-	-	-
BRCLR n,opr,rel.	Skocz, gdy bit n w M=0	? Bit n of M = 0	DIR b0 DIR bl DIR b2 DIR b3 DIR b4 DIR b5 DIR b6 DIR b7	01 03 05 07 09 0B 0D 0F	dd rr dd rr dd rr dd rr dd rr dd rr dd rr dd rr	5 5 5 5 5 5 5 5	-	-	-	-	Δ

BRN rel.	Nie skacz	? 1 = 0 (never true)	REL	21	rr	3	-	-	-	-	-
BRSET n,opr,rel.	Skocz, gdy bit n w M=l	?Bit n of M = l	DIR b0 DIR bl DIR b2 DIR b3 DIR b4 DIR b5 DIR b6 DIR b7	00 02 04 06 08 0A 0C 0E	dd rr dd rr dd rr dd rr dd rr dd rr dd rr dd rr	5 5 5 5 5 5 5 5	-	-	-	-	Δ
BSET n,opr.	Ustaw bit n w pamięci M	Mn ← l	DIR b0 DIR bl DIR b2 DIR b3 DIR b4 DIR b5 DIR b6 DIR b7	10 12 14 16 18 1A 1C 1E	dd dd dd dd dd dd dd dd	5 5 5 5 5 5 5 5	-	-	-	-	-
BSR rel.	Skocz do podprogramu	PC ←(PC)+2 push (PCL); SP=SP-1 push (PCH); SP=SP-1 PC ← (PC)+REL	REL	AD	rr	6	-	-	-	-	-
CLC	Zeruj bit przeniesienia C	C ← 0	INH	98		2	-	-	-	-	0
CLI	Zeruj bit maski przerwania	I ← 0	INH	9A		2	-	0	-	-	-

CLR opr. CLRA CLRX CLR opr. CLR opr.	Zeruj	M ← 00 A ← 00 X ← 00 M ← 00 M ← 00	DIR INH INH IX1 IX	3F 4F 5F 6F 7F	dd ff 	5 3 3 6 5	-	-	0	1	
CMP opr.	Porównaj A z daną M	(A) - (M)	IMM DIR EXT IX2 IX1 IX	A1 B1 C1 D1 E1 F1	ii dd hh ll ee ff ff 	2 3 4 5 4 3	-	-	Δ	Δ	Δ
COM opr. COMA COMX COM opr. COM opr.	Neguj poszczególne bity	M ← $\overline{M}$ = $FF - (M) A ← $\overline{A}$ X ← $\overline{X}$ M ← $\overline{M}$ M ← $\overline{M}$	DIR INH INH IX1 IX	33 43 53 63 73	dd ff 	5 3 3 6 5	-	-	Δ	Δ	1
CPX opr.	Porównaj rejestr X z daną M	(X) - (M)	IMM DIR EXT IX2 IX1 IX	A3 B3 C3 D3 E3 F3	ii dd hh ll ee ff ff 	2 3 4 5 4 3	-	-	Δ	Δ	Δ

DEC opr. DECA DECX DEC opr. DEC opr.	Zmniejsz o 1	M ← (M) - 1 A ← (A) - 1 X ← (X) - 1 M ← (M) - 1 M ← (M) - 1	DIR INH INH IX1 IX	3A 4A 5A 6A 7A	dd ff 	5 3 3 6 5	-	-	Δ	Δ	-
EOR opr.	Suma modulo 2 A z pamięcią	A ← (A) ⊕ (M)	IMM DIR EXT IX2 IX1 IX	A8 B8 C8 D8 E8 F8	ii dd hh ll ee ff ff 	2 3 4 5 4 3	-	-	Δ	Δ	-
INC opr. INCA INCX INC opr. INC opr.	Zwiększ o 1	M ← (M) + 1 A ← (A) + 1 X ← (X) + 1 M ← (M) + 1 M ← (M) + 1	DIR INH INH IX1 IX	3C 4C 5C 6C 7C	dd ff 	5 3 3 6 5	-	-	Δ	Δ	-
JMP opr.	Skok w dowolne miejsce programu	PC ← Adres Efektywny	DIR EXT IX2 IX1 IX	BC CC DC EC FC	dd hh ll ee ff ff 	2 3 4 3 2	-	-	-	-	-
JSR opr.	Skocz do podprogramu	PC ← PC+n (n = l, 2, or 3) push (PCL); SP←SP-1 push (PCH); SP←SP-1 PC← Adres Efektywny	DIR EXT IX2 IX1 IX	BD CD DD ED FD	dd hh ll ee ff ff 	5 6 7 6 5	-	-	-	-	-

LDA opr.	Zapisz do akumulatora	A ← (M)	IMM DIR EXT IX2 IX1 IX	A6 B6 C6 D6 E6 F6	ii dd hh ll ee ff ff	2 3 4 5 4 3	-	-	Δ	Δ	-
LDX opr.	Zapisz do rejestru indeksowego	X←(M)	IMM DIR EXT IX2 IX1 IX	AE BE CE DE EE FE	ii dd hh ll ee ff ff	2 3 4 5 4 3	-	-	Δ	Δ	-
LSL opr. LSLA LSLX LSL opr. LSL opr.	Logiczne przesunięcie w lewo		DIR INH INH X1 IX	38 48 58 68 78	dd ff	5 3 3 6 5	-	-	Δ	Δ	Δ
LSR opr. LSRA LSRX LSR opr. LSR opr.	Logiczne przesunięcie w prawo		DIR INH INH IX1 IX	34 44 54 64 74	dd ff	5 3 3 6 5			0	A	A

MUL	Pomnóż	X:A ← (X) x (A)	INH	42		11	0	-	-	-	0
NEC opr. NEGA NEGX NEG opr. NEG opr.	Zanegowanie (zmiana znaku + na – i odwrotnie)	M ← -(M) = $00 - (M) A ← -(A) X ← -(X) M ← -(M) M ← -(M)	DIR INH INH IX1 IX	30 40 50 60 70	dd ff 	5 3 3 6 5	-	-	Δ	Δ	Δ
NOP	Nic nie rób		INH	9D		2	-	-	-	-	-
ORA opr.	Suma logiczna OR z daną	A ← (A) + (M)	IMM DIR EXT IX2 IX1 IX	AA BA CA DA EA FA	ii dd hh ll ee ff ff 	2 3 4 5 4 3	-	-	Δ	Δ	-
ROL opr. ROLA ROLX ROL opr. ROL opr.	Obrót w lewo przez C w kierunku starszych bitów		DIR INH INH IX1 IX	39 49 59 69 79	dd ff 	5 3 3 6 5	-	-	Δ	Δ	Δ
ROR opr. RORA RORX ROR opr. ROR opr.	Obrót w prawo przez C w kierunku młodszych bitów		DIR INH INH IX1 IX	36 46 56 66 76	dd ff 	5 3 3 6 5	-	-	Δ	Δ	Δ
RSP	Resetuj wskaźnik stosu	SP ← $00FF	INH	9C		2	-	-	-	-	-

RTI	Powrót z obsługi przerwania zarówno z programowego i sprzętowego	SP = SP+l; pull(CCR) SP = SP+l; pull(A) SP = SP+l; pull(X) SP = SP+l; pull(PCH) SP = SP+l; pull(PCL)	INH	80		9	(ze stosu) Δ	Δ	Δ	Δ	Δ
RTS	Powrót z podprogramu	SP = SP+l; pull(PCH) SP = SP+1; pull(PCL)	INH	81		6	-	-	-	-	-
SBC opr.	Odejmowanie z uwzględnieniem przeniesienia	A ← (A) - (M) – C	IMM DIR XT IX2 IX1 IX	A2 B2 C2 D2 E2 F2	ii dd hh ll ee ff ff	2 3 4 5 4 3	-	-	Δ	Δ	Δ
SEC	Ustaw bit przeniesienia C	C ← 1	INH	99		2	-	-	-	-	1
SEI	Ustaw bit maski przerwania	I ← 1	INH	9B		2	-	1	-	-	-
STA opr.	Zapisz akumulator A do komórki pamięci	M ← (A)	DIR EXT IX2 IX1 IX	B7 C7 D7 E7 F7	dd hh ll ee ff ff	4 5 6 5 4	-	-	Δ	Δ	-
STOP	Aktywne $\overline{\text{IRQ}}$;Stop Oscyl.		INH	8E		2	-	0	-	-	-

STX opr.	Zapisz rejestr X do pamięci	M ← (X)	DIR EXT IX2 IX1 IX	BF CF DF EF FF	dd hh ll ee ff ff	4 5 6 5 4	-	-	Δ	Δ	-
SUB opr.	Odejmowanie bez uwzględnienia przeniesienia	A ← (A) - (M)	IMM DIR EXT IX2 IX1 IX	A0 B0 C0 D0 E0 F0	ii dd hh ll ee ff ff	2 3 4 5 4 3	-	-	Δ	Δ	Δ
SWI	Wywołanie przerwania programowego z poziomu programu	PC ← PC+1 Push PCL; SP=SP-1 push PCH; SP=SP-1 push X; SP=SP-1 push A; SP=SP-1 push CCR; SP=SP-1 I Bit ← 1 PCH ← ($xxFC)(vector PCL ← ($xxFD) fetch)	INH	83		10	-	1	-	-	-
TAX	Przeniesienie A do X	X ← (A)	INH	97		2	-	-	-	-	-
TST opr. TSTA TSTX TST opr. TST opr.	Testowanie wartości akumulatora lub wartości pamięci dla negacji lub zera, wpływa tylko na znaczniki	(M) – 0	DIR INH INH IX1 IX	3D 4D 5D 6D 7D	dd ff 	4 3 3 5 4	-	-	Δ	Δ	-
TXA	Przeniesienie X do A	A ← (X)	INH	9F		2	-	-	-	-	-
WAIT	Oczekiwanie na przerwanie		INH	8F		2	-	0	-	-	-

Mikrokontroler Motorola MC68HC05

Dziękuję za uwagę!