инж. Александър Стефанов Керезов

Ръководство за лабораторни упражнения

по

МИКРОПРОЦЕСОРНА СХЕМОТЕХНИКА

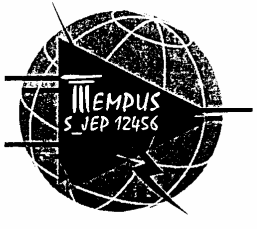
за бакалавър-инженер по Електроника

Технически Университет – София, 2000

В настоящето ръководство са изложени упражненията от курса по Микропроцесорна Схемотехника за специалността бакалавър-инженер по електроника в Технически Университет – София. Курсът е базиран основно на едночиповия микроконтролер MC68HC11 на Motorola и стандартната развойна система M68HC11EVB, произвеждана от същата фирма. Към всяка тема е представена кратка теоретична част, даваща основните необходими познания за провеждане на упражнението.

Ръководството е предназначено за студентите от специалност "Електроника" във висшите учебни заведения. То може да бъде използвано и от студенти от други специалности, както и от специалисти, работещи в тази област.

Издаването на настоящото ръководство стана възможно благодарение на подкрепата, оказана на автора по линията на проекта TEMPUS S_JEP 12456 "RESTRUCTURING DEGREE COURSES IN ELECTRONICS AND COMMUNICATIONS"



МИКРОПРОЦЕСОРНА СХЕМОТЕХНИКА

Лабораторно упражнение No.1

Микропроцесорна система за развитие M68HC11EVB – I част

I. Цел на упражнението

Студентите се запознават с развойната система M68HC11EVB, ориентирана за настройване на апаратна част и програмно осигуряване на електронни устройства, изградени на базата на едночиповите микроконтролери от фамилията MC68HC11. Разучава се организацията на системата, начините за комуникация с нея и част от мониторните команди.

II. Опитна постановка

Лабораторната постановка се състои от:

- Персонален микрокомпютър (ПК) съвместим с IBM-PC със самозареждащ се от хард- или флопи-диск MS-DOS 3.0 или по-нов.
- Лабораторен макет с вграден M68HC11EVB (апаратен симулатор, вътрешносхемен входно/изходен емулатор, комуникатор).
- Програма CONECT.EXE, представляваща комуникационна програма за осъществяване на двупосочен сериен обмен с външни компютърни системи.

Програмата след стартирането си превръща ПК в устройство тип "TERMINAL".

Необходими външни връзки:

- Подаване на 5V, +12V, -12V към макета.
- Връзка чрез специален кабел на COM1 или COM2 на персоналния микрокомпютър към M68HC11EVB (куплунг "TERMINAL").
- Вътрешно е осъществена връзка между изхода TXD и входа RXD на вградения интерфейс за сериен обмен на информация SCI.

Упражнението се провежда със системата за развитие M68HC11EVB на MOTORLA и персонален компютър IBM PC. Връзката между тях се осъществява по сериен интерфейс RS-232. За целта се стартира комуникационната програма CONECT от персоналния компютър:

C:/ connect

Избира се режим 1 (*В линия*) от менюто, при което след натискане на бутон ENTER на дисплея се изобразява знака на монитора > . С това системата е

готова за работа. Командите, които ще се изпълняват се въвеждат от клавиатурата на компютъра.

III. Задачи за изпълнение

1. Да се разучи структурата на развойната система и да се начертае блоковата и схема.
2. Да се стартира комуникационната програма CONECT в персоналния компютър и да се установи връзка с развойната система.
3. Да се разучат и експериментират командите на мониторната програма за работа с клетки от паметта, поставяне/премахване на точки на прекъсване, стартиране на програма, трасиране на програма.
4. Да се зареди в оперативната памет следната програма за сумиране на три числа. Да се изпълни програмата.

```
1  0000                NAME ADD
2  0000                P68H11
3  0000          BEG       EQU  $0000
4  0003          END       EQU  $0003
5  0010          M         EQU  $0010
6                *СЪБИРАНЕ НА 3 ЧИСЛА ОТКЛЕТКИ BEG ДО END
7                *РЕЗУЛТАТА СЕ НАМИРА В КЛЕТКИ M И M+1
8  C000             ORG  $C000

9  C000  CC0000      LDD  #0000  ~ НУЛИРАНЕ НА АССА И
                                                    ACCB
10 C003  CE0000      LDX  #BEG     НАЧАЛЕН АДРЕС
11 C006  EB00    L1  ADDB 0,X      СЪБИРАНЕ
12 C008  8900        ADCA #0
13 C00A  08          INX           IX:=IX+1
14 C00B  8C0003      CPX  #END     КРАЙ?
15 C00E  26F6        BNE  L1       ПРЕХОД АКО НЕ
16 C010  DD10        STD  M        РЕЗУЛТАТ-->в M и M+1
17 C012  3F          SWI           КРАЙ

18 C000          END  ADD
```

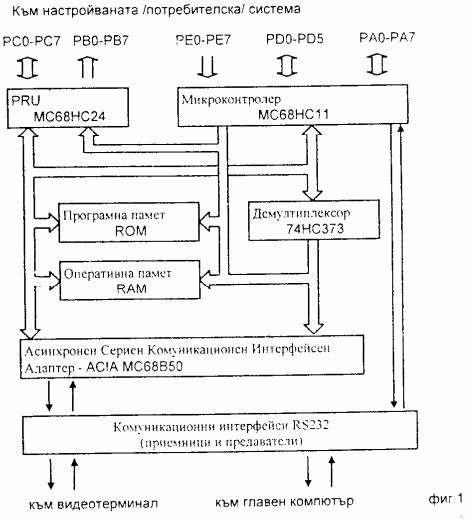
5. Да се постави точка на прекъсване на втората инструкция и да се стартира програмата.
6. Да се изпълни програмата стъпка по стъпка, като се запише състоянието на регистрите на всяка стъпка. Началният адрес на трасиране ($C000) се задава в PC с помощта на командата RM. Резултатите да се представят в табличен вид:

Код на операцията	PC	IY	IX	Акумулатор A	Акумулатор B	CCR S|X|H|I|N|Z|V|C

7. Да се състави блоков алгоритъм на програмата.

IV. Информационна част

1. Описание на развойната система



фиг.1

Развойната система M68HC11EVB е изградена на базата на едночиповия микроконтролер MC68HC11. Структурната и схема е показана на фиг.1. Развойната система може да емулира работата на едночипов микроконтролер MC68HC11 в потребителска система, давайки възможност по този начин да бъде тествана и настроена както апаратната и част, така и програмното осигуряване.

В развойната система едночиповият микроконтролер работи в разширен режим, при което PB0-PB7 и PC0-PC7 изпълняват ролята на адресна и

мултиплексирана /адресна и даннова/ магистрали. За да могат да бъдат емулирани и тези портове в потребителската система е въведена периферна схема, имитираща тяхното поведение /MC68HC24/.

Демултиплексирането на адресната и данновата магистрали се извършва с помощта на регистъра 74HC373. В него се запомня младшата част от адреса и той остава валиден и по времето, когато по същите линии се извършва обмен на данни.

В системата се съдържат по 8 Kbytes постоянна и оперативна памети. В първата е записана мониторната програма, чрез която се извършва диалога с оператора, а втората е свободна за потребителски цели.

Предвидени са два серийни комуникационни канала, удовлетворяващи изискванията на стандарта RS232C. Първият от тях се осъществява чрез асинхронен комуникационен интерфейсен адаптер MC68B50 и е предназначен за връзка с видеотерминал. За изграждане на втория канал се използва вградения в едночиповия микроконтролер интерфейс SCI и е предназначен за връзка с работна станция /централен компютър/.

2. Комуникационна програма CONECT

За да може да се осъществи връзка между развойната система и персоналния компютър, в последния е необходимо да се стартира комуникационната програма CONECT. След стартирането и в диалогов режим се указва номера на комуникационния порт, скоростта и формата на предаваните данни. Програмата пренасочва въвежданите от клавиатурата символи към серийния порт, от където те постъпват в развойната система.

Получаваните от развойната система символи на свой ред се изобразяват на екрана на монитора. Това превръща персоналния компютър в своеобразен видеотерминал, чрез който оператора общува с развойната система.

V. Контролни въпроси

1. За какво служи микропроцесорната система за развитие M68HC11EVB?
2. Как се осъществява връзката между микропроцесорната система за развитие M68HC11EVB и персоналният компютър?
3. Начертайте общата блокова схема на микропроцесорната система за развитие M68HC11EVB.
4. Как се въвежда програма на Асемблер в микропроцесорната система за развитие M68HC11EVB?

МИКРОПРОЦЕСОРНА СХЕМОТЕХНИКА

Лабораторно упражнение No. 2

Микропроцесорна система за развитие M68HC11EVB
II част

I. Цел на упражнението:

Студентите да се запознаят с вътрешната структура на развойната система 68HC11EVB и командите на мониторната програма.

II. Опитна постановка

Лабораторната постановка се състои от:

- Персонален микрокомпютър (ПК) съвместим с IBM-PC със самозареждащ се от хард- или флопи-диск MS-DOS 3.0 или по-нов.
- Лабораторен макет с вграден M68HC11EVB (апаратен симулатор, вътрешносхемен входно/изходен емулатор, комуникатор).
- Програма CONECT.EXE, представляваща комуникационна програма за осъществяване на двупосочен сериен обмен с външни компютърни системи.

Програмата след стартирането си превръща ПК в устройство тип "TERMINAL".

Необходими външни връзки:

- Подаване на 5V, +12V, -12V към макета.
- Връзка чрез специален кабел на COM1 или COM2 на персоналния микрокомпютър към M68HC11EVB (куплунг "TERMINAL").
- Вътрешно е осъществена връзка между изхода TXD и входа RXD на вградения интерфейс за сериен обмен на информация SCI.

Упражнението се провежда със системата за развитие M68HC11EVB на MOTORLA и персонален компютър IBM PC. Връзката между тях се осъществява по сериен интерфейс RS-232. За целта се стартира комуникационната програма CONECT от персоналния компютър:

C:/ connect

Избира се режим 1 (*В линия*) от менюто, при което след натискане на бутон ENTER на дисплея се изобразява знака на монитора > . С това системата е

готова за работа. Командите, които ще се изпълняват се въвеждат от клавиатурата на компютъра.

III. Задачи за изпълнение:

1. Да се разучат и изпълнят дадените в Приложение 1 команди на мониторната програма на развойната система M68HC11EVB. Да се състави кратка таблица с командите по възходящ поименен ред със синтаксис на командата и кратко описание на функцията и.
2. С помощта на командата ASM от адрес $C000 да се въведе програмата дадена по долу. Програмата инициализира SCI-подсистемата за работа и прехвърля данни от областта $C400 в $C200 чрез серийния канал на микроконтролера.

   С подходяща команда (BF) адресната област $C400-$C41F да се запълни с $55, а в областта $C200-$C21F с $00. Да се поставят точки на прекъсване на адреси $C003, $C00A и $C02E (команда BR). Да се стартира програмата (команда G) и се проследи резултата от изпълнението ѝ, като при всяко спиране се визуализира областта от паметта $C200-$C21F (команда MD).

   За коректното изпълнение на програмата е необходимо изводи 20 /RxD/ и 21 /TxD/ на куплунга P1 да се свържат накъсо.

```
1                       .list on
2   C000                org $C000
3
4   C000  CE C4 00          LDX  #$C400    начален адрес на буфера за предаване
5   C003  18 CE C2 00       LDY  #$C200    начален адрес на буфера за приемане
6   C007  B6 10 2E          LDAA $102E     нулиране на RDRF бита
7   C00A  A6 00      Label0 LDAA 0,X       зареждане на байт данни
8   C00C  B7 10 2F          STA  $102F     четене на регистъра за данни на SCI
9   C00F  B6 10 2E   Label1 LDAA $102E     четене на регистъра на състоянието на SCI
10  C012  84 80             ANDA #$80      проверка на състоянието на бит TDRE
11  C014  27 F9             BEQ  Label1    преход към ред 9 ако бит TDRE =0
12  C016  B6 10 2E   Label2 LDAA $102E     четене на регистъра на състоянието на SCI
13  C019  84 20             ANDA #$20      проверка за състоянието на бит RDRF
14  C01B  27 F9             BEQ  Label2    преход към 12 ако бит RDRF =0
15  C01D  B6 10 2F          LDAA $102F     четене на регистъра за данни на SCI
16  C020  18 A7 00          STAA 0,Y       запис в буфера за приемане
17  C023  08                INX
18  C024  18 08             INY
19  C026  8C C4 20          CPX  #$C420    проверка за край на данните
20  C029  27 03             BEQ  Label3    преход към ред 22 ако е край
21  C02B  7E C0 0A          JMP  Label0    преход към ред 7 ако не е край
22  C02E  3F         Label3 SWI            край
```

3. Да се начертае блоковия алгоритъм на управляващата програма от точка 2.

IV. Информационна част

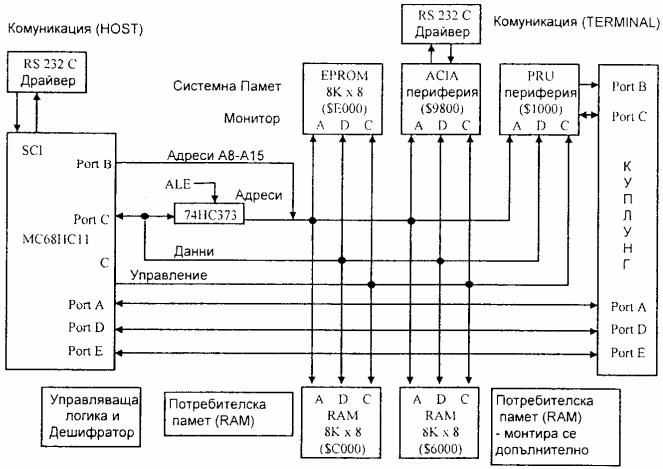
Структурата на системата за развитие M68HC11EVB е показана на фиг.2. В нея са включени:

- 8-битов микроконтролер MC68HC11A8 ;
- 8 Кбайта постоянна памет EPROM;
- 8 Кбайта оперативна памет SRAM за потребителски нужди;
- два серийни интерфейса за връзка с терминал (TERMINAL) и централен компютър (HOST) по стандарт RS232C;

PRU MC68HC24 за емулация на Port B и Port C;

На външен куплунг са изведени всички сигнали на микроконтролера.

Развойната система е проектирана за работа в два режима. В първия режим посредством командите на мониторната програма се дава възможност на потребителя да създава и настройва потребителски програми, да зарежда програми от външен източник, да променя съдържанието на вътрешните регистри на микроконтролера и клетки от оперативната памет. Вторият режим позволява на MC68HC11EVB да се използува като емулатор в системи изградени на базата на MC68HC11. В този случай цялата развойна система може да се разглежда като един микроконтролер 68HC11 работещ в едночипов вариант.



фиг.2. ФУНКЦИОНАЛНА (БЛОКОВА) СХЕМА НА СИСТЕМАТА M68HC11EVB.

Разпределението на адресното пространство на системата е показано на фиг.3.

Начало	Описание	Край
$E000	ROM Монитор - BUFFALO 2.5	$FFFF
$C000	Потребителски RAM 8K	$DFFF
$B800	Не се използва	$BFFF
$B600	EEPROM 512b - вграден	$B7FF
$A000	Не се използва	$B5FF
$9800	Външно ACIA за терминал	$9FFF
$8000	Не се използва	$97FF
$6000	Допълнителен RAM 8K	$7FFF
$4000	Управление на MUX за PD0	$5FFF
$1800	Не се използва	$3FFF
$1000	Регистри и PRU MC68HC24)	$17FF
$0100	Не се използва	$0FFF
$0000	Вграден RAM 256b	$00FF

фиг.3 Адресна карта на паметта

Единия от серийните канали на системата е изграден на базата на вградения сериен комуникационен интерфейс в 68HC11 /SCI/.

SCI-подсистемата е една от двете независими входно-изходни подсистеми за сериен обмен на данни в 68HC11. Предавателната и приемната част работят независимо една от друга, но използуват един и същ формат на данните и една скорост на обмен. SCI има някой особености, като например:

- връзката се осъществява по две линии RxD и TxD без допълнителни управляващи сигнали;
- програмно задаване на формата на данните и една от 32 възможни скорости на обмен;
- възможност за откриване на грешки и шумове по линията за приемане;
- възможност за детектиране на край на съобщение;

Управлението на интерфейса се осъществява с помощта на няколко регистъра, предназначението на които е дадено в специализираната литература [л.1], [л.2].

В упражнението са използувани следните похвати и съкращения при управлението на асинхронния сериен канал:

- ред 6 - нулира RDRF (Receive Data Register Full) бита в регистъра SCSR (SCI Status Register). RDRF=0 означава че регистъра за получаване на данни от HOST комуникатора е празен.
- ред 8 - зарежда в SCDR (SCI Data Register) данни за изпращане.
- ред 9,10,11 - чете SCSR, избира бит TDRE (Transmit Data Register Empty), ако TDRE=0 това означава, че SCDR е още пълен и трябва да се изчака изпразването му от данните.
- ред 12,13,14 - чете SCSR, избира бит RDRF, ако RDRF=0 това означава, че SCDR е още празен и трябва да се изчака изпълването му с данни.
- ред 15 - чете постъпилите вече в SCDR данни.

Най-общо погледнато работата на SCI се извършва чрез контролирането на пет вградени регистъра:

- BAUD- управление на скоростта на обмен
- SCCR1- определя 8-9 битова дължина на данните и др.
- SCCR2- определя формата на предаване-приемане на данните
- SCSR- следи за правилното предаване- приемане
- SCDR- изпраща- приема данни

V. Контролни въпроси

1. Какво е предназначението на интегралната схема MC68HC24 в микропроцесорната система за развитие M68HC11EVB?
2. Какво е разпределението на адресното пространство в микропроцесорната система за развитие M68HC11EVB?
3. Кои са основните команди на Монитора на микропроцесорната система за развитие M68HC11EVB?
4. Как се поставят точки на прекъсване и как се продължава изпълнението на програмата след точка на прекъсване?
5. Каква информация извежда Монитора след приключване изпълнението на потребителската програма?

МИКРОПРОЦЕСОРНА СХЕМОТЕХНИКА

Лабораторно упражнение No. 3

Видове адресации и инструкции на едночиповия микроконтролер MC68HC11

I. Цел на упражнението

Студентите се запознават с видовете адресации и инструкции на микроконтролера MC68HC11.

II. Опитна постановка

Лабораторната постановка се състои от:

- Персонален микрокомпютър (ПК) съвместим с IBM-PC със самозареждащ се от хард- или флопи-диск MS-DOS 3.0 или по-нов.
- Лабораторен макет с вграден M68HC11EVB (апаратен симулатор, вътрешносхемен входно/изходен емулатор, комуникатор).
- Програма CONECT.EXE, представляваща комуникационна програма за осъществяване на двупосочен сериен обмен с външни компютърни системи.

Програмата след стартирането си превръща ПК в устройство тип "TERMINAL".

Необходими външни връзки:

- Подаване на 5V, +12V, -12V към макета.
- Връзка чрез специален кабел на COM1 или COM2 на персоналния микрокомпютър към M68HC11EVB (куплунг "TERMINAL").
- Вътрешно е осъществена връзка между изхода TXD и входа RXD на вградения интерфейс за сериен обмен на информация SCI.

Упражнението се провежда със системата за развитие M68HC11EVB на MOTORLA и персонален компютър IBM PC. Връзката между тях се осъществява по сериен интерфейс RS-232. За целта се стартира комуникационната програма CONECT от персоналния компютър:

C:/ connect

Избира се режим 1 (*В линия*) от менюто, при което след натискане на бутон ENTER на дисплея се изобразява знака на монитора > . С това системата е

готова за работа. Командите, които ще се изпълняват се въвеждат от клавиатурата на компютъра.

III. Задачи за изпълнение

1. Да се разучи програмния модел ма микроконтролера MC68HC11.
2. Да се разучат видовете инструкции и адресации на микроконтролера.
3. Да се стартира програмата CONECT от PC и да се установи връзка с развойната система.
4. Да се въведе от адрес $C000 (ASM C000) и изпълни стъпка по стъпка (RM, T) програмата, илюстрираща работата на различните инструкции и да се попълни таблицата,като:

• * Вписва се адреса и съдържанието на клетката от RAM, която е обект на действие на инструкцията (ако има такава).

• Броя на процсорните цикли за изпълнение на инструкциите се взима от описанието на инструкциите (от приложението)

Инструкция	Код на операцията	Съдържание на регистрите												Действие	Вид адресация	Брой цикли	Брой байтове	Съдържание На RAM*
		PC	IY	IX	A	B	CCR							SP				
							S	X	H	I	N	Z	V	C					

Програма, илюстрираща работата на различните инструкции:

```
1.    LDAA   #$55
2.    STAA   $00      ; Виж съдържанието на клетка $0000 в RAM
3.    STAA   $C400    ; Виж съдържанието на клетка $C400 в RAM

4.    LDX    #$0000
5.    LDY    #$0010
6.    LDAB   0,X      ; Виж съдържанието на клетка $0000 в RAM
7.    STAB   0,Y      ; Виж съдържанието на клетка $0010 в RAM
8.    STAB   $10,Y    ; Виж съдържанието на клетка $0020 в RAM

9.    LDAA   #$AA
10.   TAB
11.   INX
```



```
12. L1 DEX
13.    BEQ    L1
14.    JMP    L2
15.    LDX    #$E000

16. L2 INX
17.    CPX    #$0002
18.    BLO    L2

19.    LDAA   #$55
20.    ANDA   #$A5
21.    ADDA   #05

22.    BSET   $00,01   ; Виж съдържанието на клетка $0000 в RAM
23. L3 DEC    $00      ; Виж съдържанието на клетка $0000 в RAM
24.    BRCLR  $00,01,L3

25.    LDS    #$C800
26.    PSHA            ; Виж съдържанието на клетки $C7F0 - $C800
27.    PULB
28.    PSHX            ; Виж съдържанието на клетки $C7F0 - $C800
29.    PULY

30.    NOP
31.    SWI
```

IV. Информационна част

1. Регистри на процесора

MC68HC11 има 7 програмно достъпни регистъра, а именно:

1.1. Акумулатори А и В

Акумулатор А и Акумулатор В са 8-битови регистри с общо предназначение използвани да съхраняват операндите и резултата от аритметичните операции и обмена на данни. Тези два акумулатора могат също така да бъдат разглеждани и като 16-битов двоен акумулатор D.

1.2. Индексен регистър Х (IX)

16-битовия индексен регистър Х се използва за индексна адресация. Той подава 16-битовата индексна стойност която се сумира с 8-битовото отместване в инструкцията за да се изчисли действителния адрес. Регистъра IX може да бъде използван и като брояч или регистър за временно съхранение.

1.3. Индексен регистър Y (IY)

16-битовия индексен регистър Y се използва за индексна адресация подобно на регистъра IX. Използването му обаче изисква един допълнителен байт в машинния код на инструкцията тъй като всички инструкции с IY са с двубайтови кодове.

1.4. Указател на стека (SP)

Указателя на стека (SP) е 16 битов регистър, съдържащ адреса на следващата свободна клетка от стека. Стека има организация " Първи влязъл - първи излязъл" с достъп четене/запис който позволява съхранение на данните по време прекъсвания и изпълнение на подпрограми. При запис на нов байт в стека указателя на стека автоматично се намалява, докато при четене SP се увеличава.

1.5. Програмен брояч (PC)

Програмния брояч е 16-битов регистър който съдържа адреса на следващата инструкция, чието изпълнение предстои.

1.6. Регистър за кода на условията (CCR)

Регистър за кода на условията (CCR) е 8-битов регистър в който всеки бит указва определено условие от резултата на току що изпълнената инструкция. Тези битове могат да бъдат тествани индивидуално от програмата и в резултат от тези тестове могат да бъдат предприети различни действия.

1.6.1. Бит за пренос/заем (C)

Бит C се установява в логическа 1 ако съществува пренос или заем от аритметично-логическото устройство по време на последната аритметична операция. Бит C може също така да бъде променян и при инструкции за преместване (shift, rotate).

1.6.2. Бит за препълване (V)

Бит V се установява в логическа 1 ако съществува препълване по време на последната аритметична операция; в противен случаи бит V е в логическа 0.

1.6.3. Бит за нулев резултат (Z)

Бит Z се установява в логическа 1 ако резултата от изпълнението на последната аритметична операция или манипулация на данни е нула; в противен случаи бит Z е в логическа 0.

1.6.4. Бит за отрицателен резултат (N)

Бит N се установява в логическа 1 ако резултата от изпълнението на последната аритметична операция или манипулация на данни е отрицателен; в противен

случаи бит N е в логическа 0. Резултата се счита за отрицателен ако най-старшия бит е в логическа 1.

1.6.5. Маска на прекъсването (I)

Бита за маска на прекъсването I се установява в логическа 1 по апаратен или програмен път с цел забрана на всички маскируеми прекъсвания (вътрешни и външни).

1.6.6. Бит за полупренос (H)

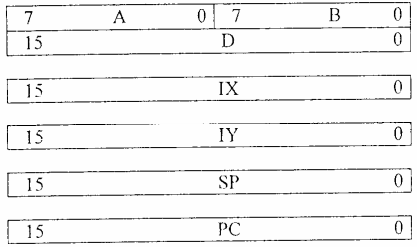
Бит H се установява в логическа 1 ако в резултата от изпълнението на последната аритметична операция е възникнал пренос между бит 3 и бит 4; в противен случаи бит H е в логическа 0.

1.6.7. Маска на прекъсването (X)

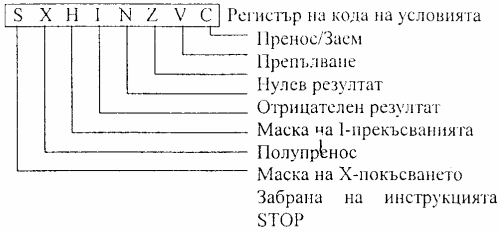
Бита за маска на прекъсването X се установява в логическа 1 по апаратен път по време на начално установяване RESET, изчистването става по програмен път.

1.6.8. Бит за забрана на инструкцията STOP (S)

Битът S се установява с цел забрана на инструкцията STOP и се изчиства за разрешаването и. Бит S е изцяло под програмен контрол. При изпълнение на инструкция STOP при установен S бит резултата от изпълнението е NOP инструкция.



8-битови акумулатори A и B
или 16-битов двоен акум. D
Индексен регистър X
Индексен регистър Y
Стеков Указател
Програмен Брояч



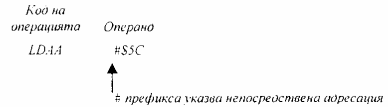
Фиг. 4 Програмен Модел

2. Начини на адресация

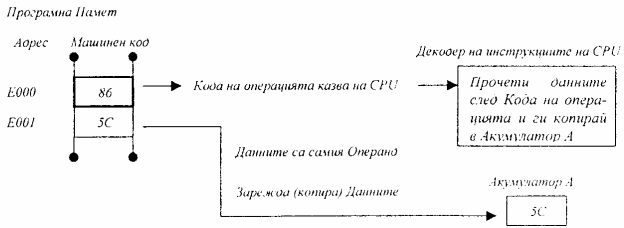
Микроконтролера MC68HC11 има шест възможни начина на адресация, а именно: непосредствена, директна, разширена, индексна, по подразбиране и относителна.

2.1. Непосредствена адресация

Инструкция:



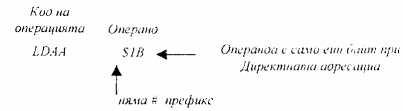
Операция



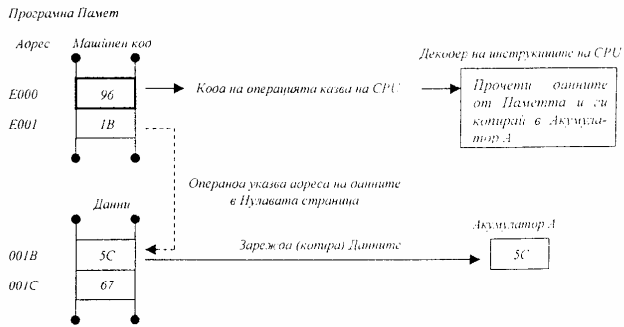
При непосредствената адресация действителния аргумент се съдържа в байта (байтовете) непосредствено след оп-кода на инструкцията, като размера на операнда съответства на размера на регистъра (8 или 16 бита).

2.2. Директна адресация

Инструкция:



Операция

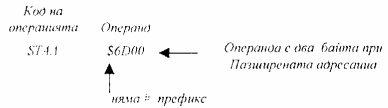


При директната адресация (наричана също така адресация в нулевата страница) действителния аргумент се съдържа в паметта на адреса, чийто младши байт е записан непосредствено след оп-кода на инструкцията, като за старши байт на адреса винаги се възприема $00. Директната адресация дава на потребителя възможност за достъп до областта $0000 - $00FF чрез използване на инструкции, състоящи се само от 2 байта (код на операцията и операнд), спестявайки по този начин програмна памет и време за изпълнение на инструкцията. В повечето приложения в тази област от 256 байта се намира вътрешната оперативна памет на микроконтролера, но тя може да бъде премествана по програмен път и така областта може да бъде освобождавана за външен ресурс.

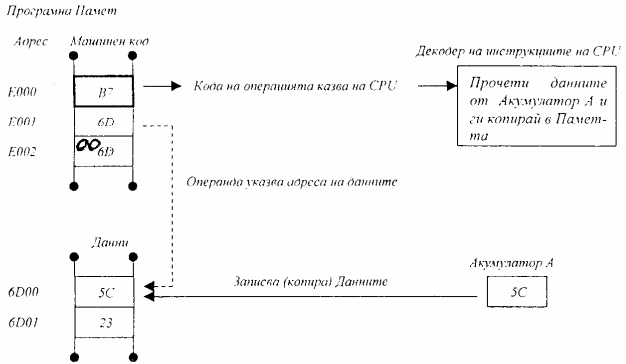
2.3. Разширена адресация

При разширената адресация действителния аргумент се съдържа в паметта на адреса, които се намира записан в двата байта непосредствено след оп-кода на инструкцията. Разширената адресация дава на потребителя възможност за достъп до цялото адресно поле на системата, но е по-дълга и по-бавна от директната адресация.

Инструкция:



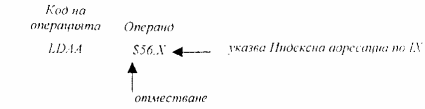
Операция



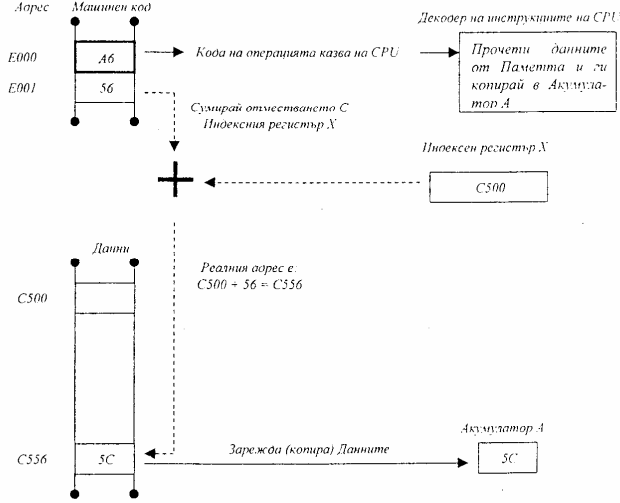
2.4. Индексна адресация

При индексната адресация действителния аргумент се съдържа в паметта на адреса, който се изчислява по време на изпълнение на самата инструкция като сума от стойността на указания 16-битов индексен регистър (IX или IY) и 8-битово отместване, намиращо се записан0 в байта непосредствено след оп-кода на инструкцията. Разширената адресация дава на потребителя възможност за достъп до цялото адресно поле на системата.

Инструкция:



Операция



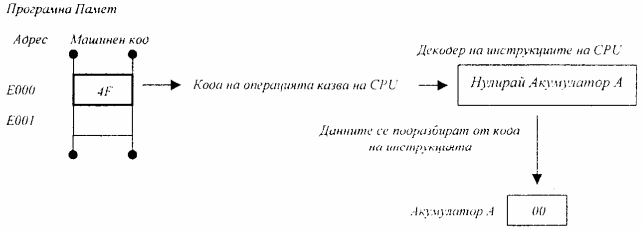
2.5. Адресация по подразбиране

При адресацията по подразбиране операнда се подразбира от самата същност на инструкцията и не е необходимо допълнителното му оказване. Обикновено това са инструкции за работа със самите регистри.

Инструкция:

Код на операцията	*Операнд*
CLRA	*НЯМА*

Операция:



2.6. Относително адресиране

Относителното адресиране се използва при инструкциите за преход. Ако условието за прехода е вярно съдържанието на програмния брояч се сумира с 8-битовото отместване със знак, записано непосредствено след кода на операцията, формирайки по този начин действителния адрес на прехода. Ако условието за преход не е вярно изпълнението продължава със следващата по ред инструкция.

V. Контролни въпроси

1. Какъв е програмния модел на едночиповия микроконтролер MC68HC11?
2. Колко начина за адресиране притежава едночиповия микроконтролер MC68HC11?
3. Кои са програмно достъпните регистри на едночиповия микроконтролер MC68HC11?
4. Каква е разликата между директната и разширената адресации?

5. Какви са основните типове инструкции при едночиповия микроконтролер MC68HC11?
6. За какво се използват флаговете в регистъра за кода на условията (CCR)?
7. Каква е разрядността на акумулаторите на едночиповия микроконтролер MC68HC11?

МИКРОПРОЦЕСОРНА СХЕМОТЕХНИКА

Лабораторно упражнение No.4

Обмен на информация между микропроцесор и оперативна памет

I. Цел на упражнението

Студентите се запознават със синхронизиращите сигнали на микропроцесора MC68HC11. Разучават се апаратните и програмните особености на връзката между микропроцесора и оперативната памет. Изследват се и се снемат времедиаграмите на циклите за четене и запис. Разглежда се предназначението на управляващите сигнали. Запознават се с начина на адресната дешифрация.

II. Опитна постановка

Лабораторната постановка се състои от:

- Персонален микрокомпютър (ПК) съвместим с IBM-PC със самозареждащ се от хард- или флопи-диск MS-DOS 3.0 или по-нов.
- Лабораторен макет с вграден M68HC11EVB (апаратен симулатор, вътрешносхемен входно/изходен емулатор, комуникатор).
- Програма CONECT.EXE, представляваща комуникационна програма за осъществяване на двупосочен сериен обмен с външни компютърни системи.

Програмата след стартирането си превръща ПК в устройство тип "TERMINAL".

Необходими външни връзки:

- Подаване на 5V, +12V, -12V към макета.
- Връзка чрез специален кабел на COM1 или COM2 на персоналния микрокомпютър към M68HC11EVB (куплунг "TERMINAL").
- Вътрешно е осъществена връзка между изхода TXD и входа RXD на вградения интерфейс за сериен обмен на информация SCI.

Упражнението се провежда със системата за развитие M68HC11EVB на MOTORLA и персонален компютър IBM PC. Връзката между тях се осъществява по сериен интерфейс RS-232. За целта се стартира комуникационната програма CONECT от персоналния компютър:

C:/ connect

Избира се режим 1 (*В линия*) от менюто, при което след натискане на бутон ENTER на дисплея се изобразява знака на монитора > . С това системата е готова за работа. Командите, които ще се изпълняват се въвеждат от клавиатурата на компютъра.

Изследваните сигнали са изведени в контролни точки на лицевия панел. Техните времедиаграми се снемат чрез двулъчев осцилоскоп. Изследваната схема е дадена в приложението. Оперативната памет е изградена с интегралната схема 6264 с организация 8к x 8 bit. Демултиплексирането на шината адреси/данни се извършва с помощта на регистъра 74HC373, в който се запомня младша част адресите по време на активното ниво на сигнала AS (адресен строб). Предавателите на буферите за данни се разрешават по време на сигнала E само когато паметта е избрана и операцията е четене. Приемниците са постоянно разрешени. Сигналът за избор на паметта (CSRAM) се изработва от първичния декодер на адресното поле. Записът в паметта е разрешен само по време на сигнала E.

III. Задачи за изпълнение:

1. Да се разучи схемата на свързване на микропроцесора с оперативната памет.
2. Да се наблюдават едновременно сигналите E и AS. Да се снемат осцилограмите и да се определят времевите и амплитудните параметри на двата сигнала.
3. Да се разучи и да се зареди в паметта от адрес $C000 (ASM C000) на микропроцесорната система програма за цикличен запис в клетка. Сигналът CSRAM (избор на ИС) да се включи към единия канал на осцилоскопа, синхронизиран по него. След стартиране на програмата (G C000) на другия лъч на екрана да се наблюдават последователно сигналите R/W, AS, WE, E, D0, A0 и A8. Да се снемат времедиаграмите, като се начертаят една под друга. С тяхна помощ да се определи времето на закъснение на формираните от микропроцесора адреси и данни.

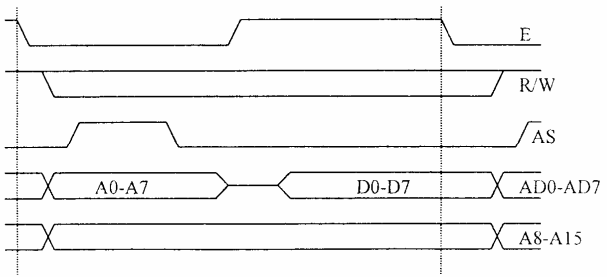
```
00001                NAM   WRITE
00002          *
00003          * ЦИКЛИЧЕН ЗАПИС В КЛЕТКА $6000
00004          *
00005 A   C000                      ORG   $C000

00006 A   C000 86 55                LDAA  #$55
00007 A   C002 97 00          L1    STAA  $6000
00008 A   C004 20 FC                BRA   L1
```

4. Да се разучи и да се зареди в паметта от адрес $C000 (ASM C000) на микропроцесорната система програма за циклично четене на клетка. Сигналът CSRAM (избор на ИС) да се включи към единия канал на осцилоскопа, синхронизиран по него. След стартиране на програмата (G C000) на другия лъч на екрана да се наблюдават последователно сигналите R/W, AS, OE, E, D0, A0 и A8. Да се определи времето на закъснение на формираните от микропроцесора адреси, както и времето на закъснение на получаваните от него данни.

```
00001                NAM   READ
00002          *
00003          * ЦИКЛИЧНО ЧЕТЕНЕ ОТ КЛЕТКА $6000
00004          *
00005 A  C000                    ORG   $C000

00006 A  C000 96 00   A   L1     LDAA  $6000
00007 A  C002 20 FC              BRA   L1
```

IV. Информационна част

Осембитовият микроконтролер MC68HC11 е монолитна интегрална схема от фамилията MC68HCxx на фирмата Motorola. Той е съвместим с TTL/CMOS ИС и се нуждае само от едно захранване +5 V. MC68HC11 притежава 16- битова адресна магистрала за пряко адресиране на 64K адреса, 8- битова двупосочна магистрала за данни и управляваща магистрала. Младшата част на адресите (A0-A7) е мултиплексирана във времето с данновата магистрала (D0-D7). Демултиплексирането им се извършва с помощта на сигнала AS (адресен



Фиг. 5

строб), който указва наличието на валиден адрес на шината.

Аритметичните и логическите операции се изпълняват от 8-битово аритметично логическо устройство (ALU). Режимът на обмена на микропроцесора с паметта се определя от изходящия сигнал R/W (четене/запис). При R/W = 1 микропроцесорът чете, а при R/W = 0 - записва. Съответно микропроцесорните буфери на магистралата за данни се ориентират като входове или като изходи.
Времедиаграмите на сигналите при обмен на микропроцесора с паметта са показани на фиг. 5. За реализиране на операцията четене е необходим един цикъл на синхронизиращият импулс E. При отрицателния фронт на E микропроцесорът формира сигналите R/W и адреса. След известно закъснение сигнала R/W се установява в логическа 1, а на адресната магистрала се появява адресът, от който се чете. По време на AS младшата част на адреса се запомня в регистъра за демултиплексиране 74HC373. След това при положителния фронт на сигнала E на адресираната памет се разрешава да изпрати към микропроцесора данните, които трябва да се установят на магистралата за данни преди следващия отрицателен фронт на E, под чието управление те се зареждат във вътрешен регистър на микропроцесора. Състоянието на магистралата за данни трябва да остане непроменено известна време след отрицателния фронт на E.

Операцията запис се изпълнява по аналогичен начин. При отрицателния фронт на E микропроцесорът формира сигналите R/W и адреса. След известно закъснение сигнала R/W се установява в логическа 0, а на адресната магистрала се появява адресът, в който се записва. По време на AS младшата част на адреса се запомня в регистъра за демултиплексиране 74HC373. След това при положителния фронт на сигнала E микропроцесорът предава данните по магистралата за данни, които се установяват на магистралата от този момент и се задържат поне известно време след отрицателния фронт на E. Адресираната памет трябва да приеме данните под управление на отрицателния фронт на E.

Адресният дешифратор представлява логическа схема, която избира устройствата, свързани в микропроцесорната система. Входовете на това устройство са свързани с адресната шина на микропроцесора, а изходите му се свързват към входовете за избор на всяка периферна или системна схема от микропроцесорната конфигурация.

V. Контролни въпроси

1. Какви синхронизиращи сигнали са необходими за работата на микропроцесора MC68HC11 и какви са изискванията към тях?

2. Какви сигнали се предават по управляващата магистрала на MC68HC11 и какви функции изпълняват?
3. Какво е минималното време за достъп, което трябва да притежава една оперативна памет, за да може да бъде свързана с микропроцесора MC68HC11 и от какво се определя то?
4. Как се определят закъсненията на формираните адреси и данни и от какво зависят те?
5. По кой такт на микропроцесора MC68HC11 се извършва информационен обмен?

МИКРОПРОЦЕСОРНА СХЕМОТЕХНИКА

Лабораторно упражнение No.5

Таймерна подсистема на едночиповия микроконтролер MC68HC11

I. Цел на упражнението:

Студентите да се запознаят с особеностите и функционалните възможности на таймерната подсистема на едночиповия микроконтролер MC68HC11A8.

II. Опитна постановка

Лабораторната постановка се състои от:

- Персонален микрокомпютър (ПК) съвместим с IBM-PC със самозареждаш се от хард- или флопи-диск MS-DOS 3.0 или по-нов.
- Лабораторен макет с вграден M68HC11EVB (апаратен симулатор, вътрешносхемен входно/изходен емулатор, комуникатор).
- Програма CONECT.EXE, представляваща комуникационна програма за осъществяване на двупосочен сериен обмен с външни компютърни системи.

Програмата след стартирането си превръща ПК в устройство тип "TERMINAL".

Необходими външни връзки:

- Подаване на 5V, +12V, -12V към макета.
- Връзка чрез специален кабел на COM1 или COM2 на персоналния микрокомпютър към M68HC11EVB (куплунг "TERMINAL").
- Вътрешно е осъществена връзка между изхода TXD и входа RXD на вградения интерфейс за сериен обмен на информация SCI.

Упражнението се провежда със системата за развитие M68HC11EVB на MOTORLA и персонален компютър IBM PC. Връзката между тях се осъществява по сериен интерфейс RS-232. За целта се стартира комуникационната програма CONECT от персоналния компютър:

C:/ connect

Избира се режим 1 (*В линия*) от менюто, при което след натискане на бутон ENTER на дисплея се изобразява знака на монитора > . С това системата е готова за работа. Командите, които ще се изпълняват се въвеждат от клавиатурата на компютъра.

III. Задачи за изпълнение

1. Да се разучи структурата на таймерната подсистема на MC68HC11A8.
2. Да се генерира периодичен сигнал с честота f = (2 MHz / 65536), като се има предвид, че предварителния делител на таймера работи с коефициент на делене 1:1 и честотата на вътрешния синхронизиращ сигнал е E = 2 MHz. За целта от адрес $C000 да се зареди програмата

```
LDX     #$1000      ;Prepare the register base for index addressing
BSET    $0C,X,$08   ;Set OC1M3 bit in OC1 Action Mask Register-
                    ;enable OC1 action over PA3 output pin
BSET    $0D,X,$08   ;Set OC1D3 bit in OC1 Action Data Register-
                    ;set PA3 output pin in logical 1 if a match with
                    ;TOC1 is found
LDAA    #$02
STAA    $1020       ;Load combination 1:0 in OM5:OL5 bits in
                    ; Timer Control Register 1 - put PA3 output pin
                    ; in logical 0 if a match with TOC5 is found
LDD     $00
STD     $1016       ;Load TOC1 register
LDD     $02
STD     $101E       ;Load TOC5 register

SWI                 ;Return to the monitor program
```

На адрес $00 и $02 да се заредят подходящи 16-битови числа, така че да се реализира коефициент на запълване 50%. Да се стартира програмата от адрес $C000 и да се наблюдава формата на изходния сигнал с осцилоскоп. Да се изчислят стойностите и наблюдават сигнали с продължителност на лог.1 съответно 10 ms и 20 ms.

3. Да се изследва функцията IC на таймерната система. За целта изведените на лицевия панел IC1 и OC2 да се свържат на късо.
   3.1. Да се зареди $01 в регистъра TCTL2 (адрес $1021) - IC1 се конфигурира за работа по преден фронт. Да се наблюдават и запишат стойностите на регистрите TOC1 ($1016), TOC2 ($101E) и TIC1 ($1014).
   3.2. Да се зареди $02 в регистъра TCTL2 (адрес $1021) - IC1 се конфигурира за работа по заден фронт. Да се наблюдават и запишат стойностите на регистрите TOC1 ($1016), TOC2 ($101E) и TIC1 ($1014).
4. Да се генерира периодичен сигнал предварително зададена произволна честота и коефициент на запълване. За целта да се зареди програмата:


```
        LDX     #$1000      ;Prepare the register base for index addressing
        BSET    $0C,X,$08   ;Set OC1M3 bit in OC1 Action Mask Register-
                            ;enable OC1 action over PA3 output pin
        BSET    $0D,X,$08   ;Set OC1D3 bit in OC1 Action Data Register-
                            ;set PA3 output pin in logical 1 if a match with
                            ;TOC1 is found
        LDAA    #$02
        STAA    $1020       ;Load combination 1:0 in OM5:OL5 bits in
                            ; Timer Control Register 1 - put PA3 output pin
                            ; in logical 0 if a match with TOC5 is found

L1      BRCLR   $23,X,$80,L1 ;Wait to set the line in log. 1
        LDAA    #$80
        STAA    $1023       ;Clear OC1F flag in TFLG1 register
        LDD     $1016       ;Load TOC1 register
        ADDD    $00
        STD     $101E       ;Store in TOC5 register

L2      BRCLR   $23,X,$08,L2 ;Wait to put the line in log. 0
        LDAA    #$08
        STAA    $1023       ;Clear OC5F flag in TFLG1 register
        LDD     $101E       ;Load TOC5 register
        ADDD    $02
        STD     $1016       ;Store in TOC1 register

        BRA     L1
```

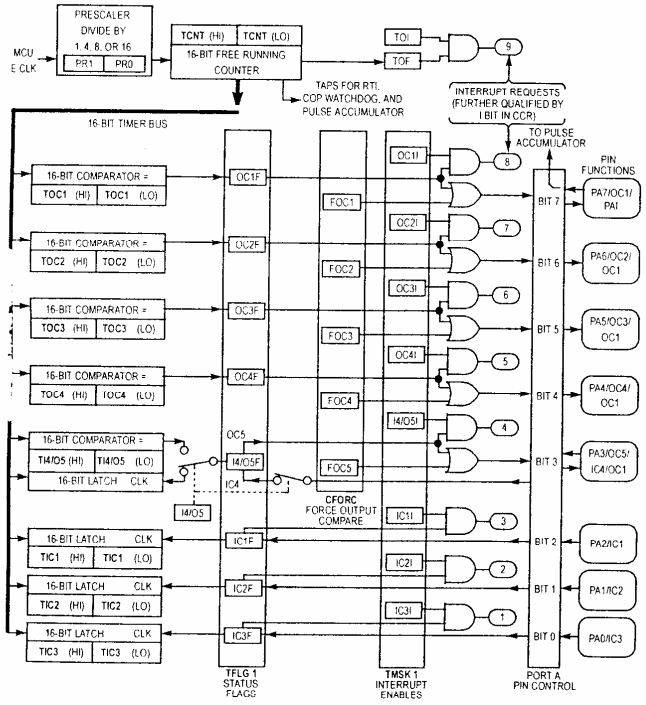
На адрес $00 и $02 да се заредят подходящи 16-битови числа, така че да се реализира периодичен сигнал с честота 1 kHz и коефициент на запълване 50%. Да се стартира програмата и да се наблюдава формата на изходния сигнал с осцилоскоп. Да се изчислят стойностите и наблюдават сигнали с коефициент на запълване 30% и 60% при същата честота.

IV. Информационна част.

Таймерната подсистема на MC68HC11 е базирана на един 16-битов брояч /работещ в непрекъснат режим/. Входа на брояча се тактува от изхода на предварителен делител, чийто коефициент на делене може да се програмира 1:1, 4:1, 8:1, 16:1. Основният таймер работи в режим на сумиране като при препълване може да се генерира прекъсване. Тази особеност позволява програмно да се преодолеят ограниченията наложени от 16-битовия размер на брояча. С таймерната под-система могат да се реализират два основни типа функции. Към първия тип спадат три независими функции INPUT CAPTURE

/IC0 - IC2/. Те се използуват за автоматично записване на текущата стойност на основния брояч, когато на съответната входна линия е детектирано желаното събитие. Към втория тип спадат петте функции OUTPUT COMPARE /OC1 - OC5/. С тяхна помощ могат да се генерират изходни импулсни сигнали с определени от потребителя времеви параметри или пък да се използуват за синтезиране на програмни време закъснения.
Към таймерната подсистема спадат още:

- схема за генериране на периодични прекъсвания /REAL TIME INTERRUPT - RTI/;
- COP - система.



Фиг. 6 Таймерна подсистема на MC68HC11

Таймерната подсистема е най-сложната в структурата на микроконтролера и за нейното управление се използуват най-много регистри. Въпреки че реализацията на голяма част от таймерните функции става с вградена логика, по същество таймерната архитектура на 68HC11 е програмно ориентирана. Това позволява лесното й адаптиране в много широка област на приложение. На фиг.6 е показана блоковата схема на основната част от таймерната подсистема.

Входна функция /INPUT CAPTURE/

Тази функция се използува за регистрация на момента, в който е настъпило някакво външно събитие. Входна детекторна логика следи фронта на сигналите свързани към всяка една от входните линии PA0-PA2. В момента, в който се открие активен фронт логиката изработва стробиращ импулс, по който се извършва запис на текущото състояние на основния таймер в 16 битовия регистър. Входната логика включва управляващи битове, с които потребителя може да програмира желания фронт на входния сигнал, който ще се следи. Избора на режима за всяка входна схема става чрез битове EDGxB и EDGxA от регистъра TCTL2 съгласно табл.1.

EDGxB	EDGxA	Конфигурация
0	0	неактивна функция
0	1	сробиране по преден фронт
1	0	сробиране по заден фронт
1	1	сробиране по всеки фронт

Табл.1

При подходящ избор последователност за следене на входния сигнал могат да се измерват времеви параметри на входни импулсни сигнали /период, коефициент на запалване, ширина на импулси/.

Функция OUTPUT COMPARE

Всяка една от 5-те изходни функции (OCx1- OCx5) се реализира чрез един 16-битов регистър за сравнение, един 16-битов компаратор и изходна програмируема схема. Компаратора сравнява числото записано в регистъра с текущата стойност на основния брояч. Когато те станат равни статус флага на съответната функция (OCxF) се установява в лог. "1", опционно може да се генерира прекъсване, а изхода на таймера се установява в състояние, програмирано предварително. Типа на реакцията на изходната схема се задава чрез два бита (OMx и OLx) от регистъра TCTL1 /табл.2/.

OM2	OL2	OM3	OL3	OM4	OL4	OM5	OL5	TCTL1	$1020

OMx	OLx	Тип на реакцията
0	0	OCx не се променя
0	1	алтернативна промяна на OCx при всяко успешно сравнение
1	0	OCx се форсира в лог. 0
1	1	OCx се форсира в лог. 1

Табл.2

Чрез програмно задаване на типа на реакцията на изходната схема и момента на появата й чрез всяка една от петте изходни функции биха могли да се генерират единични импулси с определена продължителност и ниво, непрекъснати импулсни сигнали с определена честота. Тази функция е подходяща при реализация на широчинно-импулсна модулация -ШИМ.

V. Контролни въпроси

1. Каква е разрядността на основния брояч в таймерната система на едночиповия микроконтролер MC68HC11?
2. За какво служат Входните функции ICx на едночиповия микроконтролер MC68HC11?
3. Какво представляват Изходните функции OCx на едночиповия микроконтролер MC68HC11?
4. Какви са ограниченията на таймерната система на едночиповия микроконтролер MC68HC11 при измерване на времевите параметри на периодичен сигнал?
5. Как се извършва генерирането на сигнал с произволна честота?
6. От какво се определя максималната честота, която може да бъде генерина от таймерната система на едночиповия микроконтролер MC68HC11?

МИКРОПРОЦЕСОРНА СХЕМОТЕХНИКА

Лабораторно упражнение No.6

Асинхронен последователен обмен на информация

I. Цел на упражнението

Студентите се запознават с основните принципи на асинхронния обмен на информация, схемите и форматите за предаване и приемане на информация.

II. Опитна постановка

Лабораторната постановка се състои от:

- Персонален микрокомпютър (ПК) съвместим с IBM-PC със самозареждащ се от хард- или флопи-диск MS-DOS 3.0 или по-нов.
- Лабораторен макет с вграден M68HC11EVB (апаратен симулатор, вътрешносхемен входно/изходен емулатор, комуникатор).
- Програма CONECT.EXE, представляваща комуникационна програма за осъществяване на двупосочен сериен обмен с външни компютърни системи.

Програмата след стартирането си превръща ПК в устройство тип "TERMINAL".

Необходими външни връзки:

- Подаване на 5V, +12V, -12V към макета.
- Връзка чрез специален кабел на COM1 или COM2 на персоналния микрокомпютър към M68HC11EVB (куплунг "TERMINAL").
- Вътрешно е осъществена връзка между изхода TXD и входа RXD на вградения интерфейс за сериен обмен на информация SCI.

Упражнението се провежда със системата за развитие M68HC11EVB на MOTORLA и персонален компютър IBM PC. Връзката между тях се осъществява по сериен интерфейс RS-232. За целта се стартира комуникационната програма CONECT от персоналния компютър:

C:/ connect

Избира се режим 1 (*В линия*) от менюто, при което след натискане на бутон ENTER на дисплея се изобразява знака на монитора > . С това системата е готова за работа. Командите, които ще се изпълняват се въвеждат от клавиатурата на компютъра.

Към магистралите на микропроцесора е свързан асинхронен сериен интерфейсен адаптер (ACIA) MC6850 (CM603). Адресите,на които се избират регистрите на ACIA са:

- $ 9800 - управляващ регистър CR и регистър на състоянието SR;
- $ 9801 - регистър за предаване на данни TDR и регистър за приемане на данни RDR.

III. Задачи за изпълнение

1. Да се разучи принципа на работа на асинхронния последователен интерфейс.
2. Да се наблюдават и снемат осцилограмите на предаване от PC към микропроцесорната система на символите T, U, V. За целта от клавиатурата се подава непрекъснато желания символ (задържа се натиснат съответния бутон) и се наблюдава с осцилоскоп контролната точка RxD (RS232). Да се обърне внимание на нивата на логическата 0 и 1.
3. Да се наблюдава асинхронният последователен формат на извеждана от микропроцесорната система през ACIA информация. За целта да се инициализира ACIA чрез запис на регистъра CR на адрес $ 9800 и да се снемат осцилограмите в точките TxD(CMOS), TxD(RS232), CLK при непрекъснато предаване на числото $ 0F за следните конфигурации:
   3.1. броячно отношение 1:16, 8 даннови бита, 1 стопов бит, контрол по четност;
   3.2. броячно отношение 1:16, 8 даннови бита, 1 стопови бита, контрол по нечетност;
   3.3. броячно отношение 1:16, 8 даннови бита, 1 стопов бит, без контрол;
   3.4. броячно отношение 1:64, 8 даннови бита, 2 стопови бита, без контрол.

Непрекъснатото предаваме на числото $ 0F се осъществява с помощта на програмата:

```
        LDAA    #$0F        ;Load the data to be transmitted
        LDX     #$9800      ;Load ACIA address
        LDAB    $00
        STAB    0, X        ;Write Configuration Register

L1      STAA    1, X        ;Start the transmission
L2      BRCLR   0, X, $02, L2   ;Wait TxDRE
        BRA     L1
```


Преди стартиране на програмата числото, което трябва да се запише в регистъра на конфигурацията да се зареди с помощта на командата MM на адрес $ 00 !

IV. Информационна част

Асинхронният обмен на информация за микропроцесорната фамилия MC6800 (CM600) се осъществява с интерфейсната интегрална схема ACIA MC6850 (CM603). ACIA добавя към информационните битове и следните служебни битове: 1 стартов бит, представян чрез логическото ниво 0; 1 или 2 стопови бита, представяни чрез логическо ново 1; 1 бит за проверка по четност или нечетност (може и да липсва). Всички те образуват информационната дума. Битовете се предават в следната последователност: стартов бит, 7 или 8 информационни бита (от най-младшия към най-старшия), бит за четност или нечетност (ако избраният формат включва такъв) и стопови битове. Интервалите между информационните думи се запълват с логическо ниво 1.

Обменът на данни между ACIA - MC6850 (CM603) и микропроцесора се осъществява чрез 8 двупосочни линии за данни. Посоката на пренос на информацията се управлява от микропроцесора чрез входа R/W на ACIA. Останалите линии за връзка между микропроцесора и ACIA са: CS0, CS1, CS2 - за избор на интегралната схема; RS - за избор на регистрите на ACIA; E - за разрешаване на входно-изходните буфери за данни. Връзката на ACIA с периферията се осъществява чрез две последователни линии за данни и три управляващи линии. Информацията се предава към периферията чрез изхода за предаване на данни TxD и се приема от периферията чрез входа за приемане на данни RxD. Управляващите сигнали CTS - готово за изпращане, DCD - открит носител на информация, RTS - заявка за изпращане осигуряват връзката на микропроцесора с модем. Наличието на два синхронизиращи входа TxC и RxC позволява индивидуални тактови честоти в предавателната и приемната част на ACIA.

Независимо че з микропроцесора ACIA представлява две адресируеми клетки от паметта, вътрешно той има 4 програмно достъпни 8-битови регистъра, два от които са само за запис и два за четене. Регистрите само за четене са: регистър на състоянието SR и регистър за приемане на данни TDR. Регистрите само за запис са: управляващ регистър CR и регистър за предаване на данни TDR. SR отразява състоянието на RDR и TDR, наличието на логическа грешка в обмена на информация и състоянието на линиите за връзка с други периферни устройства и модема. CR управлява операциите за контрол на приемната и предавателна част, разрешение за прекъсване и заявка за изпращане към модем.

По своето предназначение USART 8251 е подобен на СМ603. Основните различия между двете схеми са следните:

- може да работи както в асинхронен, така и в синхронен режим;
- 8251 допуска 5, 6, 7 и 8 бита дължина на думата;
- броят на стоповите битове може да бъде 1, 1.5 и 2;
- добавен е интерфейсния сигнал DTR (запитване за готовност на приемника на модема);

Битове за управление на предавателната част: управляват изхода за прекъсване, изхода RTS и осигуряват предаване на Break.

Бит 6	Бит 5	Функция
0	0	Установява RTS=0 и забранява TIE (разрешение за прекъсване при предаване на данни)
0	1	RTS=0 и разрешава TIE
1	0	RTS=1 и забранява TIE
1	1	RTS=0, забранява TIE и предава Break

Броячно отношение и начално установяване на приемника и предавателя (Rx и Tx)

Бит1	Бит 0	Функция
0	0	/ 1
0	1	/ 16
1	0	/64
1	1	Начално установяване

бит 7	бит 6	бит 5	бит 4	бит 3	бит 2	бит 1	бит 0
RIE	TC2	TC1	WS3	WS2	WS1	CDS2	CDS1

Разрешение за прекъсване на приемника
Бит 7 = 1 : разрешава изхода за прекъсване IRQ в режим на приемане
Бит 7 = 0 : забранява изхода за прекъсване IRQ в режим на приемане

Избор на дължина на думата, контрол по четност/нечетност и брой на стоповите битове:

бит 4	бит 3	бит 2	Дължина на думата	Контрол по четност/нечетност	Стопови битове
0	0	0	7	четност	2
0	0	1	7	нечетност	2
0	1	0	7	четност	1
0	1	1	7	нечетност	1
1	0	0	8	без	2
1	0	1	8	без	1
1	1	0	8	четност	1
1	1	1	8	нечетност	1

Обменът на данни между USART 8251 и микропроцесора се осъществява чрез 8 двупосочни линии за данни. Посоката на пренос на информацията се управлява

от микропроцесора чрез входовете RD и WR. Останалите линии за връзка между микропроцесора и USART CS - за избор на интегралната схема; C/D - за избор на регистри-те на USART; CLK - тактов сигнал (асинхронен); RESET - вход за апаратно начално установяване. Изходите RxRDY и TxRDY служат за организиране на обмен по прекъсване. Двупосочната линия SYNDET се използва в синхронен режим.

За микропроцесора USART заема два адреса. При C/D=1 се адресират регистрите за режима и управление (при запис) и думата на състоянието (при четене). При C/D=0 се адресират регистъра за предавани данни (запис) и приети данни (четене). След апаратен или програмен RESET първият запис е в регистъра за режима, а следващите - в регистъра за управление.

V. Контролни въпроси

1. Как се дефинират понятията формат и коефициент на на кратност при серийните интерфейсни адаптери?
2. За какво служат битовете за контрол по четност или нечетност при последователен обмен на информация в микропроцесорните системи?
3. Каква е максималната скорост за предаване на информация от ACIA и от какво се определя?
4. Как се осъществява основното нулиране на ACIA?
5. В каква последователност се извеждат битовете на предаваната информационна дума от ACIA?

МИКРОПРОЦЕСОРНА СХЕМОТЕХНИКА

Лабораторно упражнение No.7

Синхронен сериен интерфейс (SPI) на едночиповия микроконтролер MC68HC11.

I. Цел на упражнението:

Студентите да се запознаят с особеностите и функционалните възможности на едночиповия микроконтролер MC68HC11. Да се изследва начина на обмен на информация по SPI - интерфейса

II. Опитна постановка

Лабораторната постановка се състои от:

- Персонален микрокомпютър (ПК) съвместим с IBM-PC със самозареждащ се от хард- или флопи-диск MS-DOS 3.0 или по-нов.
- Лабораторен макет с вграден M68HC11EVB (апаратен симулатор, вътрешносхемен входно/изходен емулатор, комуникатор).
- Програма CONECT.EXE, представляваща комуникационна програма за осъществяване на двупосочен сериен обмен с външни компютърни системи.

Програмата след стартирането си превръща ПК в устройство тип "TERMINAL".

Необходими външни връзки:

- Подаване на 5V, +12V, -12V към макета.
- Връзка чрез специален кабел на COM1 или COM2 на персоналния микрокомпютър към M68HC11EVB (куплунг "TERMINAL").
- Вътрешно с осъществена връзка между изхода TXD и входа RXD на вградения интерфейс за сериен обмен на информация SCI.

Упражнението се провежда със системата за развитие M68HC11EVB на MOTORLA и персонален компютър IBM PC. Връзката между тях се осъществява по сериен интерфейс RS-232. За целта се стартира комуникационната програма CONECT от персоналния компютър:

C:/ connect

Избира се режим 1 (*В линия*) от менюто, при което след натискане на бутон ENTER на дисплея се изобразява знака на монитора > . С това системата е

готова за работа. Командите, които ще се изпълняват се въвеждат от клавиатурата на компютъра.

III. Задачи за изпълнение

1. Да се разучат режимите на работа на Синхронния сериен интерфейс (SPI) на микроконтролера.
2. Да се изчислят четирите възможни скорости на обмен на SPI - интерфейса за работа в режим MASTER (системен такт 2 MHz).
3. Да се инициализира SPI - интерфейса за работа в режим MASTER, CPOL = 0 и CPHA = 0. Да се стартира циклично предаване на числото $ 5C с помощта на програмата:

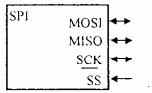
```
    LDAA    #$5C            ;Load the data to be transmitted
    LDX     #$1000          ;Load HC11 register base

L1  BCLR    $00, X, $08     ;Clear Sync pulse
    STAA    $102A           ;Start the transmission
L2  BRCLR   $29, X , $80, L2 ;Wait SPI transfer complete flag
    BSET    $00, X, $08     ;Set Sync pulse
    BRA     L1              ;Send new character
```

   Да се наблюдават и начертаят времедиаграмите на изходите MOSI, SCK и OC, като осцилоскопа се синхронизира по сигнала OC (PA3 се използва като извод с общо предназначение, който се установява програмно с пел синхронизиране).
4. Да се инициализира SPI - интерфейса за работа в режим MASTER, CPOL = 0 и CPHA = 1. Да се стартира циклично предаване на числото $ 5C с помощта на програмата от точка 3. Да се наблюдават и начертаят времедиаграмите на изходите MOSI, SCK и OC, като осцилоскопа се синхронизира по сигнала OC.
5. Да се инициализира SPI - интерфейса за работа в режим MASTER, CPOL = 1 и CPHA = 0. Да се стартира циклично предаване на числото $ 5C с помощта на програмата от точка 3. Да се наблюдават и начертаят времедиаграмите на изходите MOSI, SCK и OC, като осцилоскопа се синхронизира по сигнала OC.
6. Да се инициализира SPI - интерфейса за работа в режим MASTER, CPOL = 1 и CPHA = 1. Да се стартира циклично предаване на числото $ 5C с помощта на програмата от точка 3. Да се наблюдават и начертаят времедиаграмите на изходите MOSI, SCK и OC, като осцилоскопа се синхронизира по сигнала OC.

IV. Информационна част

Интерфейс SPI (Serial Peripheral Interface)

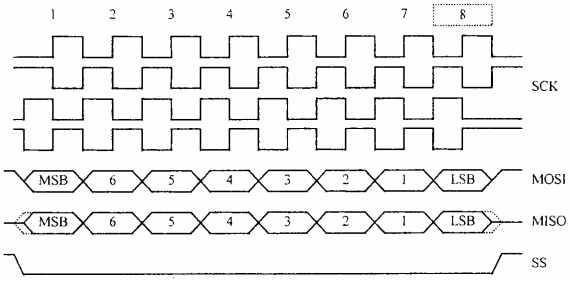
SPI (*Serial Peripheral Interface*) - стандартен синхронен сериен интерфейс на



Motorola, съвместим с интерфейса *Microwire* на National Semiconductor. Състои се от четири сигнала: MOSI (*Master Output Slave Input*), MISO *(Master Input Slave Output*), SCK (*System Clock*) и SS (*Slave Select).*

Сигналите на SPI имат следното предназначение:

- MOSI - изход за данни на водещия / вход за данни на подчинения
- MISO - вход за данни на водещия / изход за данни на подчинения
- SCK - системен такт, генериран от водещия
- SS - сигнал за избор на подчинено устройство или сигнализиране на грешка при водещото устройство



Системата може да работи с всяка една от четирите възможни комбинации на активно ниво и фаза, като изборът се извършва чрез конфигуриращите регистри. Работата на входния и изходния премествши регистри е автономна и след даване на началото на обмена (запис в регистъра за данни на SPI) генерирането на осемте такта на SCK и самия обмен се извършват без намесата на процесора. Сигналът SS служи за разрешение на съответното подчинено устройство, с което искаме да се свържем. Активирането на SS на конфигурирано като водещо устройство е сигнал за наличие на конфликт на

шината SPI (може да се получи само ако имаме повече от едно водещо устройство).

Интерфейса SPI е апаратно реализиран в микроконтролера MC68HC11. Конфигуриращите му регистри са следните:

7	6	5	4	3	2	1	0	
SPIE	SPE	DWOM	MSTR	CPOL	CPHA	SPR1	SPR0	SPCR
								$ 1028

- SPIE - локална маска на прекъсването от SPI
- SPE - включва/изключва системата за сериен синхронен обмен
  0 – системата е изключена.
  1 – системата е включена.
- DWOM - управлява състоянието на отворен дрейн за изходите на порт D (сигналите на SPI са разположени на порт D)
  0 – нормални CMOS изходни стъпала.
  1 – изходи "отворен дрейн".
- MSTR - избор на режим водещ/подчинен за работа на системата
  0 – режим MASTER (водещ).
  1 – режим SLAVE (подчинен).
- CPOL,CPHA - избор на поляритет и фаза за тактовия сигнал SCK
- SPR1,SPR0 - определя тактовата честота SCK - дели системния Е-такт на 2, 4, 16. или 32. Няма значение ако устройството е конфигурирано като подчинено.

Регистъра за състоянието на системата включва:

7	6	5	4	3	2	1	0	
SPIF	WCOL	-	MODF	-	-	-	-	SPSR
								$ 1029

- SPIF - флаг за край на предаването.
- WCOL - грешка при записа - установява се при опит за запис по време на извършване на обмен.
- MODF - грешка в режима на работа - установява се в случай, че конфигурирано като водещо устройство получи логическа 0 на входа си SS.

7	6	5	4	3	2	1	0	
D7	D6	D5	D4	D3	D2	D1	D0	SPDR
								$ 102A

SPDR ($ 102A)- регистър за данни. Записа в този регистър автоматично стартира обмена на един байт информация в режим "MASTER".

7	6	5	4	3	2	1	0	
-	-	D_SS	D_SCK	D_MOSI	D_MISO	D_TxD	D_RxD	DDDR
								$ 1009

DDRD ($ 1009)- регистър за посока (0 - вход, 1 - изход) на порт D. За правилното функциониране на SPI съответните му изводи трябва да бъдат конфигурирани в упражнението както следва:

- SS, SCK, MOSI - изходи
- MISO – вход

V. Контролни въпроси

1. Каква е разликата между синхронното и асинхронното серийно продаване на данни?
2. Колко линии са необходими за еднопосочно предаване на данни по SPI?
3. Какво е необходимо да извърши Водещото устройство, за да прочете данните от Подчиненото такова?
4. Каква е максималната скорост на предаване по SPI в режим Водещ на едночиповия микроконтролер MC68HC11?
5. Колко са възможните комбинации на поляритет и фаза на тактовия сигнал при SPI?

МИКРОПРОЦЕСОРНА СХЕМОТЕХНИКА

Лабораторно упражнение No.8

Паралелен интерфейс на едночиповия микроконтролер MC68HC11

I. Цел на упражнението

Студентите се запознават с основните принципи на паралелния обмен на информация, схемите и форматите за предаване и приемане на информация.

II. Опитна постановка

Лабораторната постановка се състои от:

- Персонален микрокомпютър (ПК) съвместим с IBM-PC със самозареждаш се от хард- или флопи-диск MS-DOS 3.0 или по-нов.
- Лабораторен макет с вграден M68HC11EVB (апаратен симулатор, вътрешносхемен входно/изходен емулатор, комуникатор).
- Програма CONECT.EXE, представляваща комуникационна програма за осъществяване на двупосочен сериен обмен с външни компютърни системи.

Програмата след стартирането си превръща ПК в устройство тип "TERMINAL".

Необходими външни връзки:

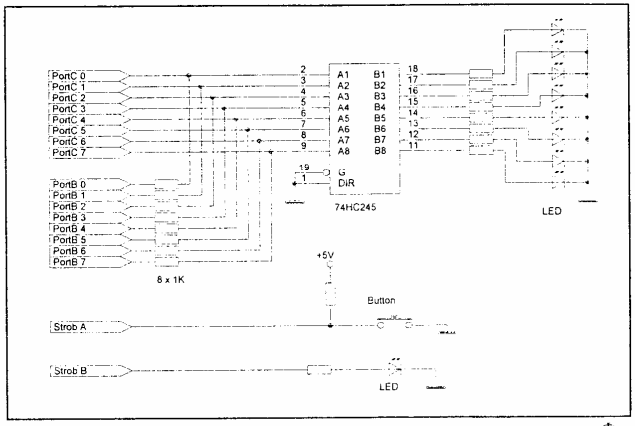
- Подаване на 5V, +12V, -12V към макета.
- Връзка чрез специален кабел на COM1 или COM2 на персоналния микрокомпютър към M68HC11EVB (куплунг "TERMINAL").
- Вътрешно е осъществена връзка между изхода TXD и входа RXD на вградения интерфейс за сериен обмен на информация SCI.

Упражнението се провежда със системата за развитие M68HC11EVB на MOTORLA и персонален компютър IBM PC. Връзката между тях се осъществява по сериен интерфейс RS-232. За целта се стартира комуникационната програма CONECT от персоналния компютър:

C:/ connect

Избира се режим 1 (*В линия*) от менюто, при което след натискане на бутон ENTER на дисплея се изобразява знака на монитора > . С това системата е готова за работа. Командите, които ще се изпълняват се въвеждат от клавиатурата на компютъра.

Към паралелните портове на мокроконтролера е свързана схема за индикация състоянието на изходите посредством светодиоди и бутон за подаване на синхронизиращ импулс STRA – фиг. 7. Когато пиновете на порт С на микроконтролера са ориентирани като входове през буферната схема на светодиодната индикация се подават нивата на изходите на порт В. Когато ориентираме пиновете на порт С на микроконтролера като изходи през буферната схема на светодиодната индикация се подават нивата на изходите на порт С, като евентуалната потенциална разлика в нивата на съответстващите изводи на портовете В и С се поема от токоограничаващите резистори. Контролният сигнал STRB се изобразява на самостоятелен светодиод, разположен на лицевия панел непосредствено до бутона за подаване на импулси на STRA.



Фиг. 7

Вътрешните регистри на микроконтролера, използвани в упражнението имат следните адреси:

$ 1002 - управляващ регистър на паралелния вход/изход (PIOC);
$ 1003 - регистър данни порт С (PORTC);
$ 1004 - регистър данни порт В (PORTB);
$ 1005 - алтернативен регистър данни порт С (PORTCL);
$ 1007 - регистър за посока на изводите на порт С (DDRC);

III. Задачи за изпълнение

1. Да се разучи работата на модула за паралелен обмен на MC68HC11
2. Да се запишат последователно в порт В числата $01, $02, $04, $08, $10, $20, $40, $80 в режим "*Simple strobe*". Да се наблюдават изводите на системата.
3. Да се реализира двоичен брояч на порт В с помощта на програмата:

```
L1    INC   1004
      BSR   DELAY
      BRA   L1

DELAY
      DEY
      BNE   DELAY
      RTS
```

4. Да се реализира бягаща светлина на порт В с помощта на програмата:

```
      LDAA #$ 01
      STAA 1004

L1    ROR   1004
      BSR   DELAY
      BRA   L1

DELAY
      DEY
      BNE   DELAY
      RTS
```

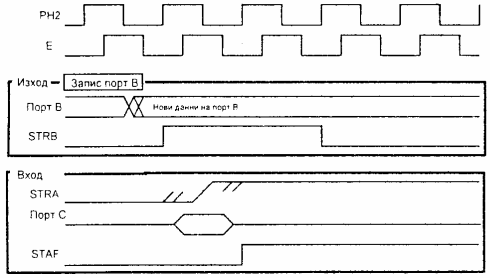
5. Да се изследва паралелния интерфейс в режим "*Full-input handshake*". Да се наблюдават изводите на системата.
   За целта системата да се инициализира за работа в режим "*Full-input handshake*", *SRTB* - по ниво. Да се подават импулси на входа STRA с помощта на бутона. Изчистването на флага за приети данни става чрез последователно четене на регистър PIOC ($ 1002) и PORTCL ($ 1005).

6. Да се изследва паралелния интерфейс в режим "*Full-output handshake*". Да се наблюдават изводите на системата.
   За целта системата да се инициализира за работа в режим "*Full-output handshake*", *SRTB* - по ниво. В PORTB ($1004) да се запише числото $00, а в PORTC - $FF. Да се подават импулси на входа STRA с помощта на бутона.

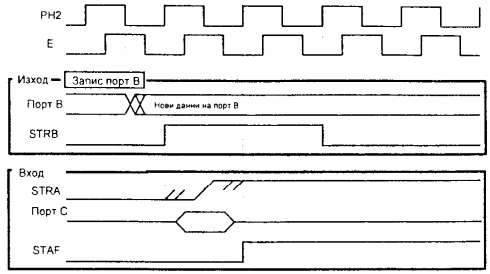
IV. Информационна част

Подсистемата за паралелен обмен на микроконтролера MC68HC11 обхваща *порт B*, *порт C*, *строб A* (*STRA*), *строб B* (*STRB*) и съответните и управляващи регистри. Тъй като в разширен режим на работа на процесора тези портове изцяло се заемат от системната магистрала е разработена специализирана интегрална схема - MC68HC24, която замества изцяло заетите портове и дава възможност на потребителя да използва пълния капацитет на системата и в разширен режим. Работата на подсистемата за паралелен вход/изход се контролира от съдържанието на *PIOC* регистъра. Имаме три основни режима на работа: "*Simple strobe*", "*Full-input handshake*" и "*Full-output handshake*".

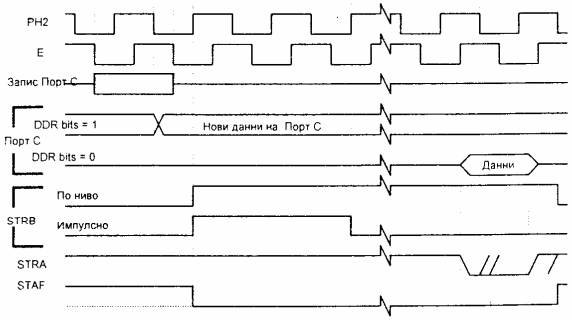
"*Simple strobe*" - при този режим след установяване на съответните битове в



конфигуриращия регистър (*PIOC*) порт B се конфигурира като 8-битов изход с асоциран му контролен сигнал *STRB*, а порт C - 8 битов вход със контрол по *STRA*. При запис в регистъра на порт B данните се установяват на *PB0-PB7* като същевременно се изработва и контролен импулс на *STRB* с продължителност 2 E-такта. Активното ниво на STRB може да бъде логическа 0 или 1 в зависимост от зададената му полярност в конфигуриращия регистър (*PIOC*). При активния фронт на входа *STRA* (отново имаме възможност да избираме поляритета на фронта в конфигуриращия регистър) информацията присъстваща в момента на входовете *PC0-PC7* се запомня във вътрешния регистър на порт C като едновременно с това се установява флага за пристигнали нови данни (*STAF*) в конфигуриращия регистър (*PIOC*) и се издава заявка за прекъсване, която може да бъде локално маскирана чрез бит *STAI*.



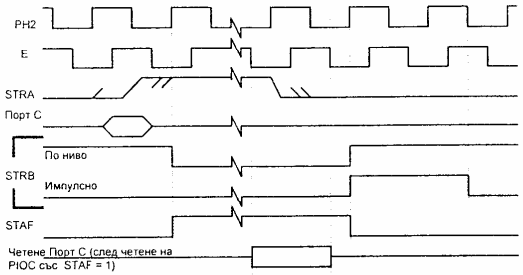
"*Full-input handshake*" - в този режим сигнала *STRB* играе ролята на указание на готовност за приемане на данни от външното устройство. В случая външното устройство не трябва да прави опит да запише нова информация преди да



получи разрешение от *STRB*, т. е. преди данните да са прочетени. Отново имаме пълна адаптируемост към различни видове външни устройства като програмно можем да конфигурираме системата за работа по преден или заден фронт на *STRA*, да подадем разрешени *STRB* по ниво или чрез импулс от 2 E-такта, да променяме активното ниво на сигнала.

Подържа се също така и локално маскируема чрез *STAI* заявка за прекъсване при пристигане на нови данни (установяване на *STAF* в логическа 1).

"*Full-output handshake*" - в този случай порт С се използва като 8 - битов паралелен изход, *STRB* е указание за готовност на данните от страна на процесора, а *STRA* е потвърждение от външното устройство, че е приело данните. Отново можем програмно да избираме полярността на активните нива



на *STRA* и *STRB*, да издаваме разрешение по ниво или импулсно разрешение на STRB, да подаваме локално маскируема заявка за прекъсване. В този случай обаче имаме възможност и да конфигурираме порт С като изходи с 3 изходни състояния, които да се намират във висок импеданс през цялото време с изключение на активния период на заявката за четене от външното устройство (активиране на *STRA*).

Както вече беше многократно споменато цялостното конфигуриране на системата се извършва от регистър *PIOC* (*Parallel Input/Output Control Register*).

7	6	5	4	3	2	1	0	
STAF	STAI	CWOM	HNDS	OIN	PLS	EGA	INVB	PIOC

- STAF - флаг на STRA. Независимо от режима на работа винаги се установява в логическа 1 при активния в момента фронт на STRA. Изчистването на флага става автоматично при прочитане на PIOC със STAF = 1, последвано от четене или запис на алтернативния регистър на порт С (в зависимост от режима - Вход или Изход).
- STAI - разрешение за прекъсване при STAF = 1
- CWOM - конфигурира порт С като нормални изходи (логическа 0) или като отворен дрейн (логическа 1)
- HNDS - Handshake (логическа 1) / Simple strobe (логическа 0) - избор на режим

- OIN - Output(логическа 1)/Input (логическа 0) Handshake - избор на режим
- PLS - избор на начин за активизация на STRB - по ниво (логическа 0) или импулсно (логическа 1)
- EGA - Избор на активен фронт на STRA – заден фронт (логическа 0) или преден (логическа 1)
- INVB - Избор на активно ниво (0/1) на STRB

V. Контролни въпроси

1. Какво е предимството на паралелното пред серийното предаване на данни?
2. В колко режима може да работи системата за паралелен обмен на информация на едночиповия микроконтролер MC68HC11?
3. Начертайте времедиаграмата на обмена в режим "Full Input Handshake".
4. Каква е разликата между активирането на STRB по ниво и импулсното активиране?
5. Какво е предназначението на сигнала STRA?

МИКРОПРОЦЕСОРНА СХЕМОТЕХНИКА

Лабораторно упражнение No.9

Аналогова подсистема на едночиповия микроконтролер MC68HC11

I. Цел на упражнението:

Студентите да се запознаят с особеностите и функционалните възможности на едночиповия микроконтролер MC68HC11A8. Да се изследват особеностите на аналого-цифровия преобразувател. Да се демонстрира принципа на емулация с M68HC11EVB.

II. Опитна постановка

Лабораторната постановка се състои от:

- Персонален микрокомпютър (ПК) съвместим с IBM-PC със самозареждаш се от хард- или флопи-диск MS-DOS 3.0 или по-нов.
- Лабораторен макет с вграден M68HC11EVB (апаратен симулатор, вътрешносхемен входно/изходен емулатор, комуникатор).
- Програма CONECT.EXE, представляваща комуникационна програма за осъществяване на двупосочен сериен обмен с външни компютърни системи.

Програмата след стартирането си превръща ПК в устройство тип "TERMINAL".

Необходими външни връзки:

- Подаване на 5V, +12V, -12V към макета.
- Връзка чрез специален кабел на COM1 или COM2 на персоналния микрокомпютър към M68HC11EVB (куплунг "TERMINAL").
- Вътрешно е осъществена връзка между изхода TXD и входа RXD на вградения интерфейс за сериен обмен на информация SCI.

Упражнението се провежда със системата за развитие M68HC11EVB на MOTORLA и персонален компютър IBM PC. Връзката между тях се осъществява по сериен интерфейс RS-232. За целта се стартира комуникационната програма CONECT от персоналния компютър:

C:/ connect

Избира се режим 1 (*В линия*) от менюто, при което след натискане на бутон ENTER на дисплея се изобразява знака на монитора > . С това системата е

готова за работа. Командите, които ще се изпълняват се въвеждат от клавиатурата на компютъра.

III. Задачи за изпълнение

1. Да се разучат режимите на работа на АЦП на микроконтролера.
2. Стойностите на опорните напрежения са: VRH=5V, VRL=0V. Да се инициализира режим на еднократно преобразуване по втори канал /CH2/ на АЦП. Да се стартира преобразуването и се запишат получените 4-ри стойности от него. Да се изчисли стойността на входното напрежение.
3. Стойностите на опорните напрежения са: VRH=5V, VRL=0V. Да се инициализира в АЦП режим на еднократно многоканално преобразуване на група 0 (CH1 - CH4). Да се стартира преобразуването и се запишат резултатите от него.
4. Стойностите на опорните напрежения са: VRH=5V, VRL=0V. Да се инициализира в АЦП режим на еднократно многоканално преобразуване група 3 (VH**, VL**, 1/2VH**). Да се стартира преобразуването и се запишат резултатите от него. Да се обяснят резултатите и смисъла от измерването на посочените напрежения.

Да се асемблира и зареди от твърдия диск кода на програмата VOLTMETER /VOLT_M.ASM/ - извеждаща напрежението на входа на втори канал (CH 2) на монитора на PC и да се стартира програмата.

Заб. Последователност на зареждането:

а) В DOS се извършва асемблиране на програмата:

```
c:/hc11/as11 volt_m.asm
```

при което се получава преносимия файл VOLT_M.S19 (във формат S19).

б) Стартира се програмата CONECT и се избира режим 1 / ON LINE / от менюто.

в) от командния режим на MC68HC11 се набира командата

```
> LOAD T.
```

с) преход към командния режим на CONECT с <Esc>.

d) избира се режим за прехвърляне на файл от диска <2>.

е) VOLT_M.S19 <enter>

f) преход към управляващ режим на MC68HC11 с <1>, ако не се осъществи връзка с MC68HC11 се налага натискане на бутона "Reset" на макета.

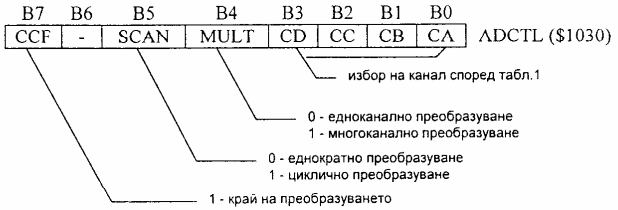
Програмата се стартира от адрес $C000 с помощта на командата

```
>G  C000
```

Резултатите от измерването се наблюдават на монитора на PC.

IV. Информационна част

Вградения АЦП в 68HC11 е 8-битов 8-канален. Общата грешка при преобразуването е +/- 1LSB. Обхвата на измерваните напрежения може да се променя чрез две външни опорни напрежения VRH - горна граница и VRL - долна граница. При промяна на опорните напрежения в граници 2,5V<(VRH-VRL)<5V общата грешка на преобразуването на АЦП се запазва същата Управлението на анологово-цифровата част става с регистъра ADCTL, значението на всеки бит, от който е показано на фиг. 9.



Стартиране на преобразуването става при запис в ADCTL- регистъра (mm 1030 <enter>). При едно канален режим се извършват 4-ри последователни преобразувания на избрания канал. Резултатите от преобразуването се записват в регистри ADR1-ADR4.

CD	CC	CB	CA	избран канал при MULT = 0	резултат в ADRx при MULT = 1
0	0	0	0	PE0	ADR1 - $1031 CH1
0	0	0	1	PE1	ADR2 - $1032 CH2
0	0	1	0	PE2	ADR3 - $1033 CH3
0	0	1	1	PE3	ADR4 - $1034 CH4
0	1	0	0	PE4	ADR1 - $1031 CH5
0	1	0	1	PE5	ADR2 - $1032 CH6
0	1	1	0	PE6	ADR3 - $1033 CH7
0	1	1	1	PE7	ADR4 - $1034 CH8
1	0	0	0	резервирано	ADR1 - $1031
1	0	0	1	резервирано	ADR2 - $1032
1	0	1	0	резервирано	ADR3 - $1033
1	0	1	1	резервирано	ADR4 - $1034

1	1	0	0	VH**	ADR1 - $1031 VH
1	1	0	1	VL**	ADR2 - $1031 VL
1	1	1	0	1/2 VH**	ADR3 - $10331/2 VH
1	0	1	1	резервирано	ADR4 - $1034

Програмата осреднява резултатите от 65536 последователни преобразувания и изчислява стойността на напрежението по линейното уравнение

$$Y = a*X + b$$

където a = (1/256) * 5 V и b = 0. Крайният резултат се преобразува от двоичен в десетичен и в последствие в ASCII код, след което се изпраща по АСИА и съответно се изобразява на монитора.

Листинг на програмния файл VOLT_M.ASM

```
Thu Nov 27 1997 16:59
"MC68HC11 Voltmeter Application"


          2500 A.D. 68c11 Macro Assembler  -  Version 5.00b
          ------------------------------------------------


                    Input  Filename : volt_m.asm
                    Output Filename : volt_m.obj
                    Listing Has Been Relocated


426                                 .LIST ON
427
428
429                                 .LINKLIST
430          1000   REG         EQU     $1000
431          9800   ASIA_ADDR   EQU     $9800
432
433          139B   ALPHA       EQU     $139B ;LINE EQUATION COEF.
434
435                                 .PAGE0
436
437    0000          DISPLAY RMB  4          ;DISPLAY SELLS IN BCD
438    0004          INTEGR  RMB  3  ;INTEGRATION CYCLE TEMPORARY
                                                          RESULT
439
440    0007          MUL1    RMB  2  ;MYLTIPLY 2x2 ROUTINE FIRST
                                                          OPERAND
```

```
0009          MUL2    RMB  2  ;MYLTIPLY 2x2 ROUTINE FIRST
                                                  OPERAND
000B          RES2X2  RMB  4  ;MYLTIPLY 2x2 ROUTINE RESULT

000F          CR      RMB  1  ;SEND CARRAGE RETURN FLAG

                              .ENDS

                              .CODE

        C000  RESET   EQU    *

C000  8E00C0         LDS    #$00C0

C003  CE1000         LDX    #REG

                             ;ENABLE ADC

C006  8680           LDAA   #$80
C008  A739           STAA   OPTION,X

                    ;Configure interrupts of MC68HC11

C00A  867E           LDAA   #$7E
C00C  97F7           STAA   $00F7       ;Illegal opcode.
C00E  CCC000         LDD    #RESET
C011  DDF8           STD    $00F8

 ;*************************************************************

                              ;INTEGRATION CYCLE

        C013         INT_CYC EQU    *
C013  18CE0000           LDY    #$0000
C017  CE1000             LDX    #REG
C01A  7F0004             CLR    INTEGR
C01D  7F0005             CLR    INTEGR+1
C020  7F0006             CLR    INTEGR+2

C023  8684               LDAA   #$81       ;pre-start for
                                                time saving
C025  A730               STAA   ADCTL,X
```



```
                         ;START ADC CONVERSION CYCLE ON PE2
        C027         MEASURE EQU    *
C027  1F3080FC           BRCLR  ADCTL,X,CCF,*   ;WAIT
                                        CONVERTION COMPLEATE FLAF
C02B  8684               LDAA   #$81
C02D  A730               STAA   ADCTL,X

                               ;CALCULATE THE RESULT
C02F  4F                 CLRA
C030  E631               LDAB   ADR1,X
C032  EB32               ADDB   ADR2,X
C034  8900               ADCA   #$00
C036  EB33               ADDB   ADR3,X
C038  8900               ADCA   #$00
C03A  EB34               ADDB   ADR4,X
C03C  8900               ADCA   #$00

                       ;INTEGRATION CYCLE - 65536 MESUREMENTS

C03E  D305               ADDD   INTEGR+1 ;******************
C040  DD05               STD    INTEGR+1 ;* ACUMULATE THE  *
C042  9604               LDAA   INTEGR   ;* RESULT IN INTEG*
C044  8900               ADCA   #$00     ;* RATION SELL    *
C046  9704               STAA   INTEGR   ;******************

C048  1809               DEY
C04A  1809               DEY
C04C  1809               DEY
C04E  1809               DEY
C050  26D5               BNE    MEASURE

                    ;65536 MEASUREMENTS WERE DONE - CALCULATE &
                          WRITE THE RESULT

C052  DC04               LDD    INTEGR
C054  DD07               STD    MUL1
C056  CC139B             LDD    #ALPHA
C059  DD09               STD    MUL2
C05B  BDC0DA             JSR    MUL2X2   ;The result is in
                                                  RES2X2

                               ;HEX TO BCD CONVERTION

C05E  CE03E8             LDX    #1000
```

```
523  C061  02          IDIV
524  C062  3C          PSHX
525  C063  CE0064      LDX     #100
526  C066  02          IDIV
527  C067  3C          PSHX
528  C068  CE000A      LDX     #10
529  C06B  02          IDIV
530  C06C  D703        STAB    DISPLAY+3
531  C06E  8F          XGDX
532  C06F  D702        STAB    DISPLAY+2
533  C071  38          PULX
534  C072  8F          XGDX
535  C073  D701        STAB    DISPLAY+1
536  C075  38          PULX
537  C076  8F          XGDX
538  C077  D700        STAB    DISPLAY+0
539
540                ;BCD TO ASCII CODE CONVERSION - THE RESULT
                     IS SEND TO THE PC MONITOR!
541
542  C079  CE9800      LDX     #ASIA_ADDR ;LOAD ASIA ADDRESS
543  C07C  1F0002FC    BRCLR   0,X,$02,*  ;WAIT TxDRE
544
545  C080  D600        LDAB    DISPLAY+0  ;LOAD THE VALUE TO
                                           CONVERT
546  C082  CB30        ADDB    #$30       ;CONVERT IN ASCII
547  C084  E701        STAB    1,X   ;START THE TRANSMISSION
548  C086  1F0002FC    BRCLR   0,X,$02,*       ;WAIT TxDRE
549
550  C08A  C62C        LDAB    #','            ;LOAD ,
551  C08C  E701        STAB    1,X   ;START THE TRANSMISSION
552  C08E  1F0002FC    BRCLR   0,X,$02,*       ;WAIT TxDRE
553
554  C092  D601        LDAB    DISPLAY+1 ;LOAD THE VALUE TO
                                           CONVERT
555  C094  CB30        ADDB    #$30       ;CONVERT IN ASCII
556  C096  E701        STAB    1,X        ;START THE
                                           TRANSMISSION
557  C098  1F0002FC    BRCLR   0,X,$02,* ;WAIT TxDRE
558
559  C09C  D602        LDAB    DISPLAY+2 ;LOAD THE VALUE TO
                                           CONVERT
560  C09E  CB30        ADDB    #$30       ;CONVERT IN ASCII
561  C0A0  E701        STAB    1,X   ;START THE TRANSMISSION
```



```
562  C0A2  1F0002FC    BRCLR   0,X,$02,* ;WAIT TxDRE
563
564  C0A6  D603        LDAB    DISPLAY+3  ;LOAD THE VALUE TO
                                           CONVERT
565  C0A8  CB30        ADDB    #$30       ;CONVERT IN ASCII
566  C0AA  E701        STAB    1,X   ;START THE TRANSMISSION
567  C0AC  1F0002FC    BRCLR   0,X,$02,*  ;WAIT TxDRE
568
569  C0B0  C620        LDAB    #' '       ;LOAD SPACE
570  C0B2  E701        STAB    1,X   ;START THE TRANSMISSION
571  C0B4  1F0002FC    BRCLR   0,X,$02,*  ;WAIT TxDRE
572
573  C0B8  C656        LDAB    #'V'  ;LOAD CARRAGE RETURN
574  C0BA  E701        STAB    1,X   ;START THE TRANSMISSION
575  C0BC  1F0002FC    BRCLR   0,X,$02,*  ;WAIT TxDRE
576
577  C0C0  C60A        LDAB    #$0A       ;LOAD LINE FEED
578  C0C2  E701        STAB    1,X   ;START THE TRANSMISSION
579  C0C4  1F0002FC    BRCLR   0,X,$02,*  ;WAIT TxDRE
580
581  C0C8  7C000F      INC     CR         ;CHANGE CARRAGE
                                           RETURN FLAG
582  C0CB  130F0108    BRCLR   CR,01,NO_CR ;TEST CARRAGE
                                           RETURN FLAG
583
584  C0CF  C60D        LDAB    #$0D  ;LOAD CARRAGE RETURN
585  C0D1  E701        STAB    1,X   ;START THE TRANSMISSION
586  C0D3  1F0002FC    BRCLR   0,X,$02,*  ;WAIT TxDRE
587
588       C0D7       NO_CR   EQU     *
589  C0D7  7EC013       JMP     INT_CYC
590
591            ;***********************************************
592            ;*  MULTIPLY 2BYTES x 2BYTES ROUTINE STARTS HERE
593            ;***********************************************
594
595       C0DA       MUL2X2  EQU     *
596  C0DA  7F000B       CLR     RES2X2
597  C0DD  7F000C       CLR     RES2X2+1
598  C0E0  9608         LDAA    MUL1+1
599  C0E2  D60A         LDAB    MUL2+1
600  C0E4  3D           MUL
601  C0E5  DD0D         STD     RES2X2+2
602  .
```

```
603  C0E7  9608          LDAA   MUL1+1
604  C0E9  D609          LDAB   MUL2
605  C0EB  3D            MUL
606  C0EC  D30C          ADDD   RES2X2+1
607  C0EE  DD0C          STD    RES2X2+1
608  C0F0  8600          LDAA   #00
609  C0F2  990B          ADCA   RES2X2
610  C0F4  970B          STAA   RES2X2
611
612  C0F6  9607          LDAA   MUL1
613  C0F8  D60A          LDAB   MUL2+1
614  C0FA  3D            MUL
615  C0FB  D30C          ADDD   RES2X2+1
616  C0FD  DD0C          STD    RES2X2+1
617  C0FF  8600          LDAA   #00
618  C101  990B          ADCA   RES2X2
619  C103  970B          STAA   RES2X2
620
621  C105  9607          LDAA   MUL1
622  C107  D609          LDAB   MUL2
623  C109  3D            MUL
624  C10A  D30B          ADDD   RES2X2
625  C10C  DD0B          STD    RES2X2
626
627  C10E  39            RTS
628
629                            .ENDS
630                            .END


        Lines Assembled :  630          Assembly Errors :  0
```

Листинг на преносимия файл VOLT_M.S19

```
S123C0008E00C0CE10008680A739867E97F7CCC000DDF818CE0000CE1
        0007F00047F00054C
S123C0207F00068681A7301F3080FC8681A7304FE631EB328900EB338
        900EB348900D305B7
S123C040DD059604890097041809180918091809260926D5DC04DD07CC139
        BDD09BDC0DACE0371
S123C060E8023CCE0064023CCE000A02D7038FD702388FD701388FD70
        0CE98001F0002FC4A
```



```
S123C080D600CB30E7011F0002FCC62CE7011F0002FCD601CB30E7011
        F0002FCD602CB3025
S123C0A0E7011F0002FCD603CB30E7011F0002FCC620E7011F0002FCC
        656E7011F0002FC92
S123C0C0C60AE7011F0002FC7C000F130F0108C60DE7011F0002FC7EC
        0137F000B7F000C93
S123C0E09608D60A3DDD0D9608D6093DD30CDD0C8600990B970B9607D
        60A3DD30CDD0C8641
S112C10000990B970B9607D6093DD30BDD0B392E
S9030000FC
```

V. Контролни въпроси

1. Колко канала има АЦП на едночиповия микроконтролер MC68HC11?
2. Каква е максималната скорост на преобразуване на един канал на АЦП на едночиповия микроконтролер MC68HC11?
3. Какъв е смисъла на измерване на Vrh, Vrl и ½ Vrh?
4. Как се зарежда външна програма в системата за развитие M68HC11EVB?

ПРИЛОЖЕНИЕ 1

КОМАНДИ НА МОНИТОРА НА M68H11EVB

Всеки команден ред включва:

> <command> [<parameters>] (RETURN)

Където :

>	знак на EVB монитора
<command>	мнемоника на командата
<parameters>	параметър на командата (адрес)
(ENTER)	за въвеждане на командния ред

Всички полета в командния ред се разделят с SPACE.

Асемблер/дизасемблер

Асемблер/дизасемблера е един интерактивен режим на въвеждане на команди. Всяка въведена инструкция се преобразува в машинен код и се записва в оперативната памет на съответния адрес. И всеки валиден код на операция /КОП/ се преобразува в мнемоника на инструкцията и операнд. Всички невалидни КОП се изобразяват на терминала като "ILLOP".

*** При въвеждането на информация трябва да се имат в предвид следните синтактични правила:

- въведените цифрови стойности да са в шестнадесетичен вид;
- отделните операнди трябва да се разделят от един или повече празни символи;
- всеки символ след валидната мнемоника на инструкцията се възприема като операнд, а въведените символи след операнда се възприемат като коментар и се игнорират.

Формат:

ASM [<address>]

Където:

<address> е стартовия адрес, който задължително попада в областта на оперативната памет.

Подкомандите в този режим са следните:

- / - Асемблиране на текущата линия и дизасемблиране на следващата команда

- ^ - Асемблиране на текущата линия и дизасемблиране на предишната команда
- (RETURN) - Асемблиране на текущата линия и дизасемблиране на следващата команда
- (CNTRL)J - Асемблиране на текущата линия. Ако няма нова линия за асемблиране тогава се дизасемблира командата на следващия адрес.
- (CNTRL)A - изход от режима на асемблиране

Пример:

```
>ASM C000
C000 STOP $FFFF
  >LDAA #55          непосредствената адресация изисква
   86 55                    # преди операнда
C002 STOP $FFFF
  >STAA C0           директна адресация
   97 C0
C004 STOP $FFFF
  >LDS  0,X          индексна адресация с отместване 0
   AE 00
C006 STOP $FFFF
  >BRA C500          преход извън допустимия обхват

Branch out of range
C006 STOP $FFFF
  >BRA C030          изчислението на адреса на всеки преход
   20 28                    се прави автоматично
C008 STOP $FFFF
  >(CTRL)A           изход от командата ASM
>
```

Команда за запълване на област от паметта с определена константа.

Формат:

BF <address1> <address2> <data>

Където:

<address1> - долната граница на областта на запълване;
<address2> - горната граница на областта на запълване;
<data> - константа на запълване.

Пример:

```
>BF C000 C0FF 30    запълва всички битове от $C000 до
```

$C0FF с $30
>BF C000 C000 0 установява $C000 в 0

Поставяне точки на прекъсване.
Формат:
BR [-][<address>]...
Където:
[-] премахва всички точки на прекъсване;
[-][<address>] премахва точка на прекъсване на посочения адрес;
По време на изпълнение на програмата когато се достигне някой от адресите, в който е поставена точка на прекъсване, изпълнението текущата програма се спира и на екрана на монитора се изобразява текущото съдържание на регистрите на процесора. Максималния брой точки на прекъсване, които могат да се поставят е 4.
** Точки на прекъсване могат да се поставят само в началото на инструкциите.

Команда за изтриване на EEPROM-а в адресната област $B600-$B7FF.
Формат:
BULK
Тази команда дава възможност на потребителя да изтрие EEPROM-а в адресната област $B600-$B7FF. След изпълнение на командата BF всички клетки от посочената област се установяват в състояние $FF.

Пример:

>BULK
>

Команда за изтриване на EEPROM-а в адресната област $B600-$B7FF и конфигурационния регистър на адрес $103F.
Формат:
BULKALL
Командата е подобна на предишната с тази разлика, че се изтрива и конфигурационния регистър на адрес $103F.

Команда за изпълнение на подпрограма:
Формат:
CALL [<address>]
Където <address> е стартовия адрес на потребителската подпрограма.

Ако не е указан адрес изпълнението на подпрограмата започва от текущото съдържание на програмния брояч (PC). В този режим е възможно поставяне на точки на прекъсване, а самото изпълнение става под управлението на

мониторната програма. Изпълнението на командата завършва след изчерпване на всички точки на прекъсване или при натискане на бутона RESET на EVB.

Примерна програма за изпълнение:
>ASM C000
C000 STX $FFFF
>LDAA #44
86 44
C002 STX $FFFF
>STAA C7FC
B7 C7FC
C005 STX $FFFF
>NOP
01
C006 STX $FFFF
>NOP
01
C007 STX $FFFF
>NOP
01
C008 STX $FFFF
>RTS
39
C008 STX $FFFF
>(CTRL)A
Пример:
>CALL C000 стартиране на подпрограмата
P-C000 Y-DEFE X-F4FF A-AA B-FE C-D0 S-004A изобразяване на вътрешното състояние на регистрите на MCU след RTS
>

Команда за стартиране на програма:
Формат:
G [<address>]
Където <address> е стартовия адрес,от който започва програмата.

G-командата дава възможност на потребителя да изпълнява програми в реално време. Началния адрес се задава в програмния брояч или непосредствено в командния ред. Изпълнението на програмата завършва след изчерпване на всички точки на прекъсване или при натискане на бутона RESET на EVB.

Пример:

```
>G C000                 стартиране на
                        програма от адрес
                        $C000

P-C005 Y-0000 X-00CD A-44 B-FB C-D0 S-004A
>
```

***Заб. Програмата за изпълнение е същата както в предишната. Необходимо е допълнително да се поставят точки на прекъсване на адреси $C005 и $C007 и след това да се изпълни командата G.

Команда за помощ на потребителя:

Формат:

HELP

Командата HELP дава информация на потребителя за всички достъпни команди на мониторната програма.

Пример:

```
>HELP
ASM [<addr>]  Line assembler/disassembler.
  /      Do same address    ^      Do previous address.
  CTRL-J   Do next address    RETURN  Do next opcode.
  CTRL-A   Quit.
BF <addr1> <addr2> [<data>]  Block fill.
BR [-][<addr>]   Set up breakpoint table.
BULK  Erase EEPROM          BULKALL  Erase EEPROM and CONFIG
CALL [<addr>] Call user subroutine. G [<addr>] Execute user code
LOAD, VERIFI <T> or <host download command> Load or Verify S-records
MD [<addr1> [<addr2>]]  Memory dump.
MM [<addr1>]  Memory modify
/      Open same address   CTRL-H or ^ Open previous address.
  CTRL-J   Open next address   SPACE      Open next address
  RETURN   Quit.              <addr>0  Compute Offset to <addr>.
MOVE <s1> <s2> [<d>]   Block move.
P  Proceed/continue executions.
RM [P,Y,X,A,B,C, or S]   Register modify.
T <n>  Trace n instruction.
TM Transparent mode (CTRL-A = exit, CTRL-B = send break).
CTRL-H  Backspace.
CTRL-X or DELETE Abort/cancel command.
```



```
RETURN Repeat last command.
>
```

Команда за зареждане на програми от главен компютър или терминал:

Формат:

LOAD < host download command >

LOAD <T>

Където с < host download command > става зареждане на информация в S-формат през серийния канал за връзка с главен компютър, а с (T) - през канала за връзка с терминал.

Стартовия адрес на зареждане трябва да се намира в оперативната памет (RAM), в противен случай се появява съобщение за грешка.

Пример:

```
>LOAD cat trial.out
cat trial.out
done
>
>LOAD cat trial.out
cat trial.out
error addr E000          невалиден адрес извън областта
>                        на оперативната памет
```

Команда за изобразяване съдържанието на област от паметта:

Формат:

MD [<address1> [<address2>]]

Където <address1> и <address2> са съответно началния и крайния адрес на областта.

Ако не се укаже крайната граница се изобразяват 9 линии с по 16 байта всяка. Ако <address1> е по-голям от <address2>, то по подразбиране се приема за начален адрес първият посочен.

Пример:

```
>MD
E7D0 AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA .....
E7E0 AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA .....
E7F0 AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA .....
E800 AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA .....
E810 AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA .....
E820 AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA .....
```

```
E830 AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA .....
E840 AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA .....
E850 AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA .....
>

>MD C030 C020
C030 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF .....
>

>MD C000 C020
C000 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF .....
C010 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF .....
C020 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF .....
>
```

Команда за смяна съдържанието на паметта:
Формат:

MM [<address>]

Тази команда позволява на потребителя да проверява паметта и да сменя съдържанието й. Тя е достъпна в областите, които са заети с RAM и EEPROM. За улесняване работата с нея могат да се използуват следните подкоманди:

- (CTRL)J or (SPACE BAR) - изобразяване/модифициране съдържанието на следващия адрес
- (CTRL)H or ^ - изобразяване/модифициране съдържанието на предишен адрес
- / - изобразяване/модифициране съдържанието на текущия адрес
- (RETURN) - прекъсване на командата MM
- O - изчисляване на относително изместване при инструкции за преход

При несъвпадение на записаната с контролно прочетената стойност след записа на екрана на терминала се появява съобщение за грешка "ROM".

Пример:

```
>MM C700

C700 44 66/
C700 66 55^
C6FF FF AA(RETURN)      смяна на съдържанието на адрес
                        C6FF и прекъсване на командата MM
>MM C13C

C13C F7 C18EO 51              изчисляване на отместване,
```



```
C13C F7                 резултат = $51

>MM B600                смяна на съдържанието на
                        EEPROM
B600 73 52
>MM B600

B600 52
>
```

Команда за преместване на блок от данни от адрес на адрес:
Формат:

MOVE <address1> <address2> [<destination>]

Където : <address1> начален адрес от паметта
<address2> краен адрес от паметта
[<destination>] начален адрес на новата област

Тази команда позволява на потребителя да копира или премества данни от една област в друга. Ако не се специфицира нов адрес, т.е. <destination> липсва, то новия начален адрес се получава като към <address1> се прибави 1. Тази команда е достъпна и за EEPROM при положение, че бит ROMON в CONFIG регистъра е нулиран.

Пример:

```
>MOVE E000 E7FF C000    преместване на блок от данни
                        от адресна област $E000-$E7FF
                        в област с адреси $C000-$C7FF
>
```

Продължение изпълнението на програмата
Формат:

P

Командата се използува за продължение изпълнението на програми, в които са поставени точки на прекъсване. Тя се използува съвместно с командата G.

Пример:

```
>G C000                 стартиране на
                        програма от
                        адрес $C000
P-C005 Y-7982 X-FF00 A-44 B-70 C-D0 S-004A
>P                      продължение на
                        изпълнението
```

P-C007 Y-7982 X-FF00 A-44 B-70 C-D0 S-004A *точка на*
> *прекъсване на*
адрес $C007

Команда за модификация на регистрите.
Формат:
RM
RM [p,y,x,a,b,c,s]

Командата се използува за промяна на съдържанието на вътрешните регистри на микроконтролера: P-програмен брояч, Y-индексен регистър, X-индексен регистър, A-акумулатор A, B- акумулатор B, C-регистър на кода на условията, S-указател на стека.
Пример:
>RM
P-C007 Y-7982 X-FF00 A-44 B-70 C-C0 S-0054
P-C007 C020

>

>RM X
P-C007 Y-7982 X-FF00 A-44 B-70 C-C0 S-0054
X-FF00 C020

>

>RM
P-C020 Y-DEFE X-C020 A-DF B-DE C-D0 S-0054
P-C020 (SPACE BAR)
Y-DFFE (SPACE BAR)
X-C020 (SPACE BAR)
A-DF (SPACE BAR)
B-DE (SPACE BAR)
C-D0 (SPACE BAR)
S-0054 (SPACE BAR)
>

Команда за стъпково изпълнение на програма.
Формат:
T[<n>]
където <n> - показва броя на командите,които ще се изпълнят ($1-$FF).

Изпълнението започва от адрес еднакъв с текущата стойност на програмния брояч. След изпълнението на всяка команда се извежда кода на изпълнената операция и вътрешното състояние на регистрите на микроконтролера.

Пример:
>T *изпълнение на единична инструкция*
Op-86
P-C002 Y-DEFE X-FFFF A-44 B-00 C-00 S-004B
>

>T 2 *изпълнение на две последователни инструкции*

Op-B7
P-C005 Y-DEFE X-FFFF A-44 B-00 C-00 S-004B
>

Op-01
P-C006 Y-DEFE X-FFFF A-44 B-00 C-00 S-004B

>

Команда за преминаване в "прозрачен режим".
Формат:
TM
Командата прави програмна връзка между устройствата свързани на двата серийни порта. Това дава възможност да се осъществи обмен на информация между главния компютър и терминала. Могат да се използуват следните подкоманди:

- (CTRL)B - изпращане на прекъсване към главния компютър.
- (CTRL)A - изход от прозрачния режим.

Пример:
>TM
appslab login: ED
*Password:*_____

"System message"

_ *$*
_
_
_

_ *$(CTRL)A* *изход от командата*
_ >

Команда за сравнение на информацията в паметта с постъпваща от вън информация

Формат:

VERIFY < host download command >
VERIFY <T>

Където:

< host download command > - сравнява паметта с информацията постъпваща от главния компютър.

<T> - сравнява паметта с информацията постъпваща от терминала.

Пример:

>VERIFY cat trial.out
cat trial.out
done *проверката е завършила без*
> *грешка*

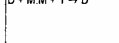
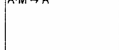
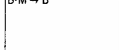
>VERIFY cat trial.out
cat trial.out

error addr E000 *при открита грешка се*
> *изобразява адреса на който е локализирана*

***Заб. Командите - LOAD, VERIFY и TM изискващи работа с външни файлове и връзка с главен компютър се изпълняват само в случаите, когато е налице интерфейсна връзка с главен компютър и предварително са създадени нужните файлове за работа.

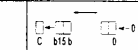
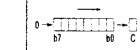
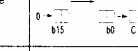
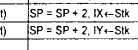
ПРИЛОЖЕНИЕ 2

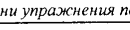
ИНСТРУКЦИИ НА АСЕМБЛЕРА НА M68H11

Source Form(s)	Operation	Boolean Expression	Addressing Mode for Operand	Machine Coding (Hexadecimal) Opcode	Operand(s)	Bytes	Cycle	Cycle by Cycle*	Condition Codes S X H I N Z V C
ABA	Add Accumulators	A + B → A	INH	1B		1	2	2-1	- - ! - ! ! ! !
ABX	Add B to X	IX + 00:B → IX	INH	3A		1	3	2-2	- - - - - - - -
ABY	Add B to Y	IY + 00:B → IY	INH	18 3A		2	4	2-4	- - - - - - - -
ADCA (opr)	Add with Carry to A	A + M + C → A	A IMM	89	ii	2	2	3-1	- - ! - ! ! ! !
			A DIR	99	dd	2	3	4-1	
			A EXT	B9	hh ll	3	4	5-2	
			A IND,X	A9	ff	2	4	6-2	
			A IND,Y	18 A9	ff	3	5	7-2	
ADCB (opr)	Add with Carry to B	B + M + C → B	B IMM	C9	ii	2	2	3-1	- - ! - ! ! ! !
			B DIR	D9	dd	2	3	4-1	
			B EXT	F9	hh ll	3	4	5-2	
			B IND,X	E9	ff	2	4	6-2	
			B IND,Y	18 E9	ff	3	5	7-2	
ADDA (opr)	Add Memory to A	A + M → A	A IMM	8B	ii	2	2	3-1	- - ! - ! ! ! !
			A DIR	9B	dd	2	3	4-1	
			A EXT	BB	hh ll	3	4	5-2	
			A IND,X	AB	ff	2	4	6-2	
			A IND,Y	18 AB	ff	3	5	7-2	
ADDB (opr)	Add Memory to B	B + M → B	B IMM	CB	ii	2	2	3-1	- - ! - ! ! ! !
			B DIR	DB	dd	2	3	4-1	
			B EXT	FB	hh ll	3	4	5-2	
			B IND,X	EB	ff	2	4	6-2	
			B IND,Y	18 EB	ff	3	5	7-2	
ADDD (opr)	Add 16-Bit to D	D + M:M + 1 → D	IMM	C3	jj kk	3	4	3-3	- - - - ! ! ! !
			DIR	D3	dd	2	5	4-7	
			EXT	F3	hh ll	3	6	5-10	
			IND,X	E3	ff	2	6	6-10	
			IND,Y	18 E3	ff	3	7	7-8	
ANDA (opr)	AND A with Memory	A•M → A	A IMM	84	ii	2	2	3-1	- - - - ! ! 0 -
			A DIR	94	dd	2	3	4-1	
			A EXT	B4	hh ll	3	4	5-2	
			A IND,X	A4	ff	2	4	6-2	
			A IND,Y	18 A4	ff	3	5	7-2	
ANDB (opr)	AND B with Memory	B•M → B	B IMM	C4	ii	2	2	3-1	- - - - ! ! 0 -
			B DIR	D4	dd	2	3	4-1	
			B EXT	F4	hh ll	3	4	5-2	
			B IND,X	E4	ff	2	4	6-2	
			B IND,Y	18 E4	ff	3	5	7-2	
ASL (opr)	Arithmetic Shift Left		EXT	78	hh ll	3	6	5-8	- - - - ! ! ! !
			IND,X	68	ff	2	6	6-3	
ASLA			IND,Y	18 68	ff	3	7	7-3	
			A INH	48		1	2	2-1	
ASLB			B INH	58		1	2	2-1	
ASLD	Arithmetic Shift Left Double		INH	05		1	3	2-2	- - - - ! ! ! !
ASR (opr)	Arithmetic Shift Right		EXT	77	hh ll	3	6	5-8	- - - - ! ! ! !
			IND,X	67	ff	2	6	6-3	
ASRA			IND,Y	18 67	ff	3	7	7-3	
			A INH	47		1	2	2-1	
ASRB			B INH	57		1	2	2-1	
BCC (rel)	Branch if Carry Clear	? C = 0	REL	24	rr	2	3	8-1	- - - - - - - -
BCLR (opr) (msk)	Clear Bit(s)	M•(mm) → M	DIR	15	dd mm	3	6	4-10	- - - - ! ! 0 -
			IND,X	1D	ff mm	3	7	6-13	
			IND,Y	18 1D	ff mm	4	8	7-10	
BCS (rel)	Branch if Carry Set	? C = 1	REL	25	rr	2	3	8-1	- - - - - - - -
BEQ (rel)	Branch if = Zero	? Z = 1	REL	27	rr	2	3	8-1	- - - - - - - -
BGE (rel)	Branch if ≥ Zero	? N ⊕ V = 0	REL	2C	rr	2	3	8-1	- - - - - - - -
BGT (rel)	Branch if > Zero	? Z + (N ⊕ V) = 0	REL	2E	rr	2	3	8-1	- - - - - - - -
BHI (rel)	Branch if Higher	? C + Z = 0	REL	22	rr	2	3	8-1	- - - - - - - -
BHS (rel)	Branch if Higher or Same	? C = 0	REL	24	rr	2	3	8-1	- - - - - - - -

Source Form(s)	Operation	Boolean Expression	Addressing Mode for Operand	Machine Coding (Hexadecimal) Opcode	Operand(s)	Bytes	Cycle	Cycle by Cycle*	Condition Codes S X H I N Z V C
BITA (opr)	Bit(s) Test A with Memory	A•M	A IMM	85	ii	2	2	3-1	- - - - ! ! 0 -
			A DIR	95	dd	2	3	4-1	
			A EXT	B5	hh ll	3	4	5-2	
			A IND,X	A5	ff	2	4	6-2	
			A IND,Y	18 A5	ff	3	5	7-2	
BITB (opr)	Bit(s) Test B with Memory	B•M	B IMM	C5	ii	2	2	3-1	- - - - ! ! 0 -
			B DIR	D5	dd	2	3	4-1	
			B EXT	F5	hh ll	3	4	5-2	
			B IND,X	E5	ff	2	4	6-2	
			B IND,Y	18 E5	ff	3	5	7-2	
BLE (rel)	Branch if ≤ Zero	? Z + (N ⊕ V) = 1	REL	2F	rr	2	3	8-1	- - - - - - - -
BLO (rel)	Branch if Lower	? C = 1	REL	25	rr	2	3	8-1	- - - - - - - -
BLS (rel)	Branch if Lower or Same	? C + Z = 1	REL	23	rr	2	3	8-1	- - - - - - - -
BLT (rel)	Branch if < Zero	? N ⊕ V = 1	REL	2D	rr	2	3	8-1	- - - - - - - -
BMI (rel)	Branch if Minus	? N = 1	REL	2B	rr	2	3	8-1	- - - - - - - -
BNE (rel)	Branch if Not = Zero	? Z = 0	REL	26	rr	2	3	8-1	- - - - - - - -
BPL (rel)	Branch if Plus	? N = 0	REL	2A	rr	2	3	8-1	- - - - - - - -
BRA (rel)	Branch Always	? 1 = 1	REL	20	rr	2	3	8-1	- - - - - - - -
BRCLR (opr) (msk) (rel)	Branch if Bit(s) Clear	? M • mm = 0	DIR	13	dd mm rr	4	6	4-11	- - - - - - - -
			IND,X	1F	ff mm rr	4	7	6-14	
			IND,Y	18 1F	ff mm rr	5	8	7-11	
BRN (rel)	Branch Never	? 1 = 0	REL	21	rr	2	3	8-1	- - - - - - - -
BRSET (opr) (msk) (rel)	Branch if Bit(s) Set	? (M) • mm = 0	DIR	12	dd mm rr	4	6	4-11	- - - - - - - -
			IND,X	1E	ff mm rr	4	7	6-14 7-	
			IND,Y	18 1E	ff mm rr	5	8	11	
BSET (opr) (msk)	Set Bit(s)	M + mm → M	DIR	14	dd mm	3	6	4-10	- - - - ! ! 0 -
			IND,X	1C	ff mm	3	7	6-13	
			IND,Y	18 1C	ff mm	4	8	7-10	
BSR (rel)	Branch to Subroutine	See Special Ops	REL	8D	rr	2	6	8-2	- - - - - - - -
BVC (rel)	Branch if Overflow Clear	? V = 0	REL	28	rr	2	3	8-1	- - - - - - - -
BVS (rel)	Branch if Overflow Set	? V = 1	REL	29	rr	2	3	8-1	- - - - - - - -
CBA	Compare A to B	A – B	INH	11		1	2	2-1	- - - - ! ! ! !
CLC	Clear Carry Bit	0 → C	INH	0C		1	2	2-1	- - - - - - - 0
CLI	Clear Interrupt Mask	0 → I	INH	0E		1	2	2-1	- - - 0 - - - -
CLR (opr)	Clear Memory Byte	0 → M	EXT	7F	hh ll	3	6	5-8	- - - - 0 1 0 0
			IND,X	6F	ff	2	6	6-3	
			IND,Y	18 6F	ff	3	7	7-3	
CLRA	Clear Accumulator A	0 → A	A INH	4F		1	2	2-1	- - - - 0 1 0 0
CLRB	Clear Accumulator B	0 → B	B INH	5F		1	2	2-1	- - - - 0 1 0 0
CLV	Clear Overflow Flag	0 → V	INH	0A		1	2	2-1	- - - - - - 0 -
CMPA (opr)	Compare A to Memory	A – M	A IMM	81	ii	2	2	3-1	- - - - ! ! ! !
			A DIR	91	dd	2	3	4-1	
			A EXT	B1	hh ll	3	4	5-2	
			A IND,X	A1	ff	2	4	6-2	
			A IND,Y	18 A1	ff	3	5	7-2	
CMPB (opr)	Compare B to Memory	B – M	B IMM	C1	ii	2	2	3-1	- - - - ! ! ! !
			B DIR	D1	dd	2	3	4-1	
			B EXT	F1	hh ll	3	4	5-2	
			B IND,X	E1	ff	2	4	6-2	
			B IND,Y	18 E1	ff	3	5	7-2	
COM (opr)	1's Complement Memory Byte	$FF – M → M	EXT	73	hh ll	3	6	5-8	- - - - ! ! 0 1
			IND,X	63	ff	2	6	6-3	
			IND,Y	18 63	ff	3	7	7-3	
COMA	1's Complement A	$FF – A → A	A INH	43		1	2	2-1	- - - - ! ! 0 1
COMB	1's Complement B	$FF – B → B	B INH	53		1	2	2-1	- - - - ! ! 0 1
CPD (opr)	Compare D to Memory 16-Bit	D – M:M + 1	IMM	1A 83	jj kk	4	5	3-5	- - - - ! ! ! !
			DIR	1A 93	dd	3	6	4-9	
			EXT	1A B3	hh ll	4	7	5-11	
			IND,X	1A A3	ff	3	7	5-11	
			IND,Y	CD A3	ff	3	7	7-8	

Source Form(s)	Operation	Boolean Expression	Addressing Mode for Operand	Machine Coding (Hexadecimal) Opcode	Operand(s)	Bytes	Cycle	Cycle by Cycle*	Condition Codes S X H I N Z V C
CPX (opr)	Compare X to Memory 16-Bit	IX – M:M + 1	IMM	8C	jj kk	3	4	3-3	----↕↕↕↕
			DIR	9C	dd	2	5	4-7	
			EXT	BC	hh ll	3	6	5-10	
			IND,X	AC	ff	2	6	6-10	
			IND,Y	CD AC	ff	3	7	7-8	
CPY (opr)	Compare Y to Memory 16-Bit	IY – M:M + 1	IMM	18 8C	jj kk	4	5	3-5	----↕↕↕↕
			DIR	18 9C	dd	3	6	4-9	
			EXT	18 BC	hh ll	4	7	5-11	
			IND,X	1A AC	ff	3	7	6-11	
			IND,Y	18 AC	ff	3	7	7-8	
DAA	Decimal Adjust A	Adjust Sum to BCD	INH	19		1	2	2-1	----↕↕↕↕
DEC (opr)	Decrement Memory Byte	M – 1 ® M	EXT	7A	hh ll	3	6	5-8	----↕↕↕-
			IND,X	6A	ff	2	6	6-3	
			IND,Y	18 6A	ff	3	7	7-3	
DECA	Decrement Accumulator A	A – 1 ® A	A INH	4A		1	2	2-1	----↕↕↕-
DECB	Decrement Accumulator B	B – 1 ® B	B INH	5A		1	2	2-1	----↕↕↕-
DES	Decrement Stack Pointer	SP – 1 ® SP	INH	34		1	3	2-3	--------
DEX	Decrement Index Register X	IX – 1 ® IX	INH	09		1	3	2-2	-----↕--
DEY	Decrement Index Register Y	IY – 1 ® IY	INH	18 09		2	4	2-4	-----↕--
EORA (opr)	Exclusive OR A with Memory	A Å M ® A	A IMM	88	ii	2	2	3-1	----↕↕0-
			A DIR	98	dd	2	3	4-1	
			A EXT	B8	hh ll	3	4	5-2	
			A IND,X	A8	ff	2	4	6-2	
			A IND,Y	18 A8	ff	3	5	7-2	
EORB (opr)	Exclusive OR B with Memory	B Å M ® B	B IMM	C8	ii	2	2	3-1	----↕↕0-
			B DIR	D8	dd	2	3	4-1	
			B EXT	F8	hh ll	3	4	5-2	
			B IND,X	E8	ff	2	4	6-2	
			B IND,Y	18 E8	ff	3	5	7-2	
FDIV	Fractional Divide 16 by 16	D/IX ® IX; r ® D	INH	03		1	41	2-17	-----↕↕↕
IDIV	Integer Divide 16 by 16	D/IX ® IX; r ® D	INH	02		1	41	2-17	-----↕0↕
INC (opr)	Increment Memory Byte	M + 1 ® M	EXT	7C	hh ll	3	6	5-8	----↕↕↕-
			IND,X	6C	ff	2	6	6-3	
			IND,Y	18 6C	ff	3	7	7-3	
INCA	Increment Accumulator A	A + 1 ® A	A INH	4C		1	2	2-1	- ↕↕↕-
INCB	Increment Accumulator B	B + 1 ® B	B INH	5C		1	2	2-1	----↕↕↕-
INS	Increment Stack Pointer	SP + 1 ® SP	INH	31		1	3	2-3	--------
INX	Increment Index Register X	IX + 1 ® IX	INH	08		1	3	2-2	-----↕--
INY	Increment Index Register Y	IY + 1 ® IY	INH	18 08		2	4	2-4	-----↕--
JMP (opr)	Jump	See Special Ops	EXT	7E	hh ll	3	3	5-1	--------
			IND,X	6E	ff	2	3	6-1	
			IND,Y	18 6E	ff	3	4	7-1	
JSR (opr)	Jump to Subroutine	See Special Ops	DIR	9D	dd	2	5	4-8	--------
			EXT	BD	hh ll	3	6	5-12	
			IND,X	AD	ff	2	6	6-12	
			IND,Y	18 AD	ff	3	7	7-9	
LDAA (opr)	Load Accumulator A	M ® A	A IMM	86	ii	2	2	3-1	----↕↕0-
			A DIR	96	dd	2	3	4-1	
			A EXT	B6	hh ll	3	4	5-2	
			A IND,X	A6	ff	2	4	6-2	
			A IND,Y	18 A6	ff	3	5	7-2	
LDAB (opr)	Load Accumulator B	M ® B	B IMM	C6	ii	2	2	3-1	----↕↕0-
			B DIR	D6	dd	2	3	4-1	
			B EXT	F6	hh ll	3	4	5-2	
			B IND,X	E6	ff	2	4	6-2	
			B IND,Y	18 E6	ff	3	5	7-2	
LDD (opr)	Load Double Accumulator D	M ® A,M + 1 ® B	IMM	CC	jj kk	3	3	3-2	----↕↕0-
			DIR	DC	dd	2	4	4-3	
			EXT	FC	hh ll	3	5	5-4	
			IND,X	EC	ff	2	5	6-6	
			IND,Y	18 EC	ff	3	6	7-6	

Source Form(s)	Operation	Boolean Expression	Addressing Mode for Operand	Machine Coding (Hexadecimal) Opcode	Operand(s)	Bytes	Cycle	Cycle by Cycle*	Condition Codes S X H I N Z V C
LDS (opr)	Load Stack Pointer	M:M + 1 → SP	IMM	8E	jj kk	3	3	3-2	----↕↕0-
			DIR	9E	dd	2	4	4-3	
			EXT	BE	hh ll	3	5	5-4	
			IND,X	AE	ff	2	5	6-6	
			IND,Y	18 AE	ff	3	6	7-6	
LDX (opr)	Load Index Register X	M:M + 1 → IX	IMM	CE	jj kk	3	3	3-2	----↕↕0-
			DIR	DE	dd	2	4	4-3	
			EXT	FE	hh ll	3	5	5-4	
			IND,X	EE	ff	2	5	6-6	
			IND,Y	CD EE	ff	3	6	7-6	
LDY (opr)	Load Index Register Y	M:M + 1 → IY	IMM	18 CE	jj kk	4	4	3-4	----↕↕0-
			DIR	18 DE	dd	3	5	4-5	
			EXT	18 FE	hh ll	4	6	5-6	
			IND,X	1A EE	ff	3	6	6-7	
			IND,Y	18 EE	ff	3	6	7-6	
LSL (opr)	Logical Shift Left		EXT	78	hh ll	3	6	5-8	----↕↕↕↕
LSLA			IND,X	68	ff	2	6	6-3	
LSLB			IND,Y	18 68	ff	3	7	3-7	
			A INH	48		1	2	2-1	
			B INH	58		1	2	2-1	
LSLD	Logical Shift Left Double		INH	05		1	3	2-2	----↕↕↕↕
LSR (opr)	Logical Shift Right		EXT	74	hh ll	3	6	5-8	----↕↕↕↕
LSRA			IND,X	64	ff	2	6	6-3	
LSRB			IND,Y	18 64	ff	3	7	7-3	
			A INH	44		1	2	2-1	
			B INH	54		1	2	2-1	
LSRD	Logical Shift Right Double		INH	04		1	3	2-2	----0↕↕↕
MUL	Multiply 8 by 8	A:B → D	INH	3D		1	10	2-13	--------↕
NEG (opr)	2's Complement Memory Byte	0 – M → M	EXT	70	hh ll	3	6	5-8	----↕↕↕↕
			IND,X	60	ff	2	6	6-3	
			IND,Y	18 60	ff	3	7	7-3	
NEGA	2's Complement A	0 – A → A	A INH	40		1	2	2-1	----↕↕↕↕
NEGB	2's Complement B	0 – B → B	B INH	50		1	2	2-1	----↕↕↕↕
NOP	No Operation	No Operation	INH	01		1	2	2-1	--------
ORAA (opr)	OR Accumulator A (Inclusive)	A + M → A	A IMM	8A	ii	2	2	3-1	----↕↕0-
			A DIR	9A	dd	2	3	4-1	
			A EXT	BA	hh ll	3	4	5-2	
			A IND,X	AA	ff	2	4	6-2	
			A IND,Y	18 AA	ff	3	5	7-2	
ORAB (opr)	OR Accumulator B (Inclusive)	B + M → B	B IMM	CA	ii	2	2	3-1	----↕↕0-
			B DIR	DA	dd	2	3	4-1	
			B EXT	FA	hh ll	3	4	5-2	
			B IND,X	EA	ff	2	4	6-2	
			B IND,Y	18 EA	ff	3	5	7-2	
PSHA	Push A onto Stack	A → Stk, SP = SP–1	A INH	36		1	3	2-6	--------
PSHB	Push B onto Stack	B → Stk, SP = SP–1	B INH	37		1	3	2-6	--------
PSHX	Push X onto Stack (Lo First)	IX → Stk, SP = SP–2	INH	3C		1	4	2-7	--------
PSHY	Push Y onto Stack (Lo First)	IY → Stk, SP = SP–2	INH	18 3C		2	5	2-8	--------
PULA	Pull A from Stack	SP = SP + 1, A ← Stk	A INH	32		1	4	2-9	--------
PULB	Pull B from Stack	SP = SP + 1, B ← Stk	B INH	33		1	4	2-9	--------
PULX	Pull X from Stack (Hi First)	SP = SP + 2, IX ← Stk	INH	38		1	5	2-10	--------
PULY	Pull Y from Stack (Hi First)	SP = SP + 2, IY ← Stk	INH	18 38		2	6	2-11	--------
ROL (opr)	Rotate Left		EXT	79	hh ll	3	6	5-8	----↕↕↕↕
ROLA			IND,X	69	ff	2	6	6-3	
ROLB			IND,Y	18 69	ff	3	7	7-3	
			A INH	49		1	2	2-1	
			B INH	59		1	2	2-1	

Source Form(s)	Operation	Boolean Expression	Addressing Mode for Operand	Machine Coding (Hexadecimal) Opcode	Operand(s)	Bytes	Cycle	Cycle by Cycle*	Condition Codes S X H I N Z V C
ROR (opr)	Rotate Right		EXT	76	hh ll	3	6	5-8	----↕↕↕↕
			IND,X	66	ff	2	6	6-3	
RORA			IND,Y	18 66	ff	3	7	7-3	
RORB		C b7 b0 C	A INH	46		1	2	2-1	
			B INH	56		1	2	2-1	
RTI	Return from Interrupt	See Special Ops	INH	3B		1	12	2-14	↕ ↕↕↕↕↕↕
RTS	Return from Subroutine	See Special Ops	INH	39		1	5	2-12	--------
SBA	Subtract B from A	A – B ® A	INH	10		1	2	2-1	----↕↕↕↕
SBCA (opr)	Subtract with Carry from A	A – M – C ® A	A IMM	82	ii	2	2	3-1	----↕↕↕↕
			A DIR	92	dd	2	3	4-1	
			A EXT	B2	hh ll	3	4	5-2	
			A IND,X	A2	ff	2	4	6-2	
			A IND,Y	18 A2	ff	3	5	7-2	
SBCB (opr)	Subtract with Carry from B	B – M – C ® B	B IMM	C2	ii	2	2	3-1	----↕↕↕↕
			B DIR	D2	dd	2	3	4-1	
			B EXT	F2	hh ll	3	4	5-2	
			B IND,X	E2	ff	2	4	6-2	
			B IND,Y	18 E2	ff	3	5	7-2	
SEC	Set Carry	1 ® C	INH	0D		1	2	2-1	-------- 1
SEI	Set Interrupt Mask	1 ® I	INH	0F		1	2	2-1	---1----
SEV	Set Overflow Flag	1 ® V	INH	0B		1	2	2-1	------1-
STAA (opr)	Store Accumulator A	A ® M	A DIR	97	dd	2	3	4-2	----↕↕0-
			A EXT	B7	hh ll	3	4	5-3	
			A IND,X	A7	ff	2	4	6-5	
			A IND,Y	18 A7	ff	3	5	7-5	
STAB (opr)	Store Accumulator B	B ® M	B DIR	D7	dd	2	3	4-2	----↕↕0-
			B EXT	F7	hh ll	3	4	5-3	
			B IND,X	E7	ff	2	4	6-5	
			B IND,Y	18 E7	ff	3	5	7-5	
STD (opr)	Store Accumulator D	A ® M, B ® M + 1	DIR	DD	dd	2	4	4-4	----↕↕0-
			EXT	FD	hh ll	3	5	5-5	
			IND,X	ED	ff	2	5	6-8	
			IND,Y	18 ED	ff	3	6	7-7	
STOP	Stop Internal Clocks		INH	CF		1	2	2-1	--------
STS (opr)	Store Stack Pointer	SP ® M:M + 1	DIR	9F	dd	2	4	4-4	----↕↕0-
			EXT	BF	hh ll	3	5	5-5	
			IND,X	AF	ff	2	5	6-8	
			IND,Y	18 AF	ff	3	6	7-7	
STX (opr)	Store Index Register X	IX ® M:M + 1	DIR	DF	dd	2	4	4-4	----↕↕0-
			EXT	FF	hh ll	3	5	5-5	
			IND,X	EF	ff	2	5	6-8	
			IND,Y	CD EF	ff	3	6	7-7	
STY (opr)	Store Index Register Y	IY ® M:M + 1	DIR	18 DF	dd	3	5	4-6	----↕↕0-
			EXT	18 FF	hh ll	4	6	5-7	
			IND,X	1A EF	ff	3	6	6-9	
			IND,Y	18 EF	ff	3	6	7-7	
SUBA (opr)	Subtract Memory from A	A – M ® A	A IMM	80	ii	2	2	3-1	----↕↕↕↕
			A DIR	90	dd	2	3	4-1	
			A EXT	B0	hh ll	3	4	5-2	
			A IND,X	A0	ff	2	4	6-2	
			A IND,Y	18 A0	ff	3	5	7-2	
SUBB (opr)	Subtract Memory from B	B – M ® B	B IMM	C0	ii	2	2	3-1	----↕↕↕↕
			B DIR	D0	dd	2	3	4-1	
			B EXT	F0	hh ll	3	4	5-2	
			B IND,X	E0	ff	2	4	6-2	
			B IND,Y	18 E0	ff	3	5	7-2	
SUBD (opr)	Subtract Memory from D	D – M:M + 1 ® D	IMM	83	jj kk	3	4	3-3	----↕↕↕↕
			DIR	93	dd	2	5	4-7	
			EXT	B3	hh ll	3	6	5-10	
			IND,X	A3	ff	2	6	6-10	
			IND,Y	18 A3	ff	3	7	7-8	
SWI	Software Interrupt	See Special Ops	INH	3F		1	14	2-15	---1----
TAB	Transfer A to B	A ® B	INH	16		1	2	2-1	----↕↕0-
TAP	Transfer A to CC Register	A ® CCR	INH	06		1	2	2-1	↕ ↕↕↕↕↕↕
TBA	Transfer B to A	B ® A	INH	17		1	2	2-1	----↕↕0-

Source Form(s)	Operation	Boolean Expression	Addressing Mode for Operand	Machine Coding (Hexadecimal) Opcode	Operand(s)	Bytes	Cycle	Cycle by Cycle*	Condition Codes S X H I N Z V C
TEST	TEST (Only in Test Modes)	Address Bus Counts	INH	00		1	**	2-20	--------
TPA	Transfer CC Register to A	CCR → A	INH	07		1	2	2-1	--------
TST (opr)	Test for Zero or Minus	M – 0	EXT	7D	hh ll	3	6	5-9	----↕↕00
			IND,X	6D	ff	2	6	6-4	
			IND,Y	18 6D	ff	3	7	7-4	
TSTA		A – 0	A INH	4D		1	2	2-1	----↕↕00
TSTB		B – 0	B INH	5D		1	2	2-1	----↕↕00
TSX	Transfer Stack Pointer to X	SP + 1 → IX	INH	30		1	3	2-3	--------
TSY	Transfer Stack Pointer to Y	SP + 1 → IY	INH	18 30		2	4	2-5	--------
TXS	Transfer X to Stack Pointer	IX – 1 → SP	INH	35		1	3	2-2	--------
TYS	Transfer Y to Stack Pointer	IY – 1 → SP	INH	18 35		2	4	2-4	--------
WAI	Wait for Interrupt	Stack Regs & WAIT	INH	3E		1	***	2-16	--------
XGDX	Exchange D with X	IX → D, D → IX	INH	8F		1	3	2-2	--------
XGDY	Exchange D with Y	IY → D, D → IY	INH	18 8F		2	4	2-4	--------

**Infinity or Until Reset Occurs

***12 Cycles are used beginning with the opcode fetch. A wait state is entered which remains in effect for an integer number of MPU E-clock cycles (n) until an interrupt is recognized. Finally, two additional cycles are used to fetch the appropriate interrupt vector (14 + n total).

dd = 8-Bit Direct Address ($0000 –$00FF) (High Byte Assumed to be $00)

ff = 8-Bit Positive Offset $00 (0) to $FF (255) (Is Added to Index)

hh = High Order Byte of 16-Bit Extended Address

ii = One Byte of Immediate Data

jj = High Order Byte of 16-Bit Immediate Data

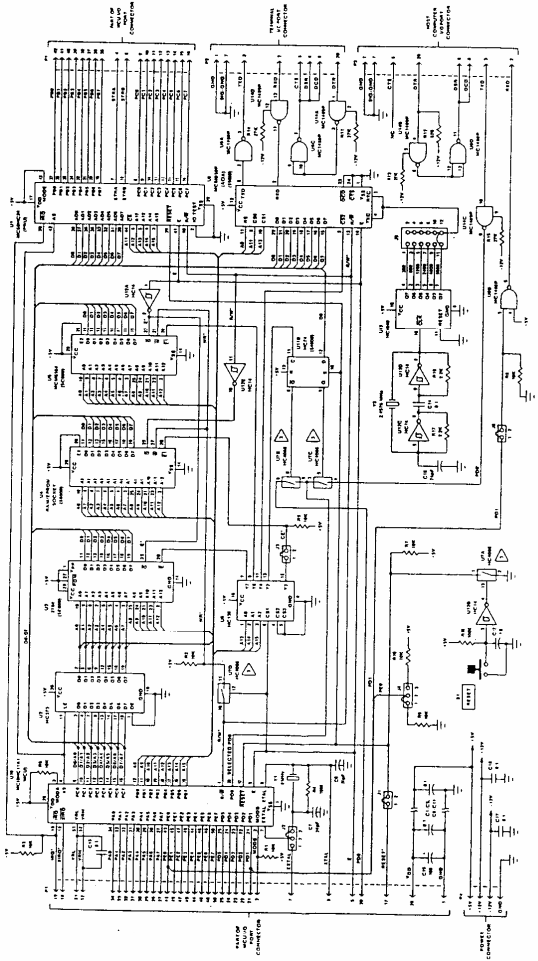
kk = Low Order Byte of 16-Bit Immediate Data

ll = Low Order Byte of 16-Bit Extended Address

mm = 8-Bit Bit Mask (Set Bits to be Affected)

rr = Signed Relative Offset $80 (– 128) to $7F (+ 127)
(Offset Relative to the Address Following the Machine Code Offset Byte)

Схема на развойната система M68HC11EVB



Времеви параметри на едночиповия микроконтролер MC68HC11

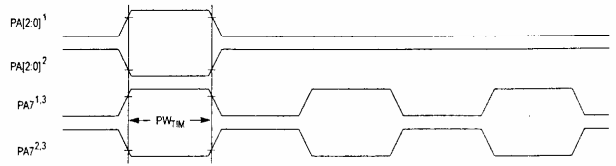
Control Timing

V_{DD} = 5.0 Vdc ± 10%, V_{SS} = 0 Vdc, $T_A = T_L$ to T_H

Characteristic	Symbol	1.0 MHz		2.0 MHz		3.0 MHz		Unit
		Min	Max	Min	Max	Min	Max	
Frequency of Operation	f_o	dc	1.0	dc	2.0	dc	3.0	MHz
E-Clock Period	t_{cyc}	1000	—	500	—	333	—	ns
Crystal Frequency	f_{XTAL}	—	4.0	—	8.0	—	12.0	MHz
External Oscillator Frequency	4 f_o	dc	4.0	dc	8.0	dc	12.0	MHz
Processor Control Setup Time t_{PCSU} = 1/4 t_{cyc} + 50 ns	t_{PCSU}	300	—	175	—	133	—	ns
Reset Input Pulse Width (To Guarantee External Reset Vector) (Minimum Input Time; Can Be Preempted by Internal Reset) (Note 1)	PW_{RSTL}	8 1	— —	8 1	— —	8 1	— —	t_{cyc}
Mode Programming Setup Time	t_{MPS}	2	— 2	— 2	—			t_{cyc}
Mode Programming Hold Time	t_{MPH}	10	— 1	0	— 1	0	—	ns
Interrupt Pulse Width, IRQ Edge-Sensitive Mode $PW_{IRQ} = t_{cyc} + 20$ ns	PW_{IRQ}	1020	—	520	—	353	—	ns
Wait Recovery Start-up Time	t_{WRS}	—	4 —	4	— 4			t_{cyc}
Timer Pulse Width Input Capture, Pulse Accumulator Input $PW_{TIM} = t_{cyc} + 20$ ns	PW_{TIM}	1020	—	520	—	353	—	ns

NOTES:

1. RESET is recognized during the first clock cycle it is held low. Internal circuitry then drives the pin low for four clock cycles, releases the pin, and samples the pin level two cycles later to determine the source of the interrupt. Refer to **9 RESETS, INTERRUPTS, AND LOW POWER MODES** for further detail.
2. All timing is shown with respect to 20% V_{DD} and 70% V_{DD}, unless otherwise noted.

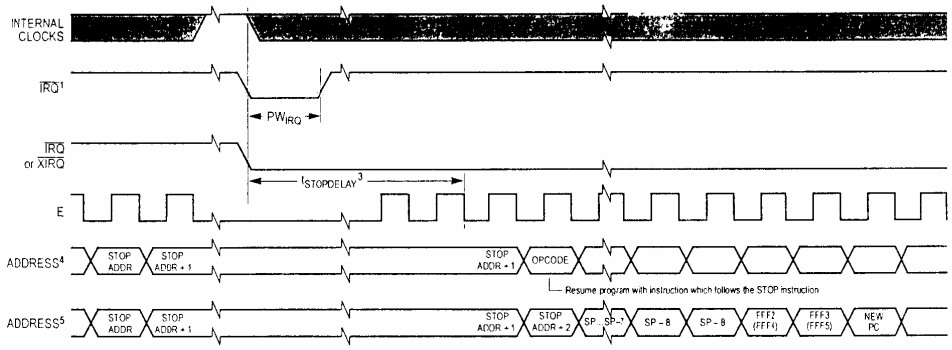


NOTES:

1. Rising edge sensitive input
2. Falling edge sensitive input
3. Maximum pulse accumulator clocking rate is E-clock frequency divided by 2.

TIMER INPUTS TIM

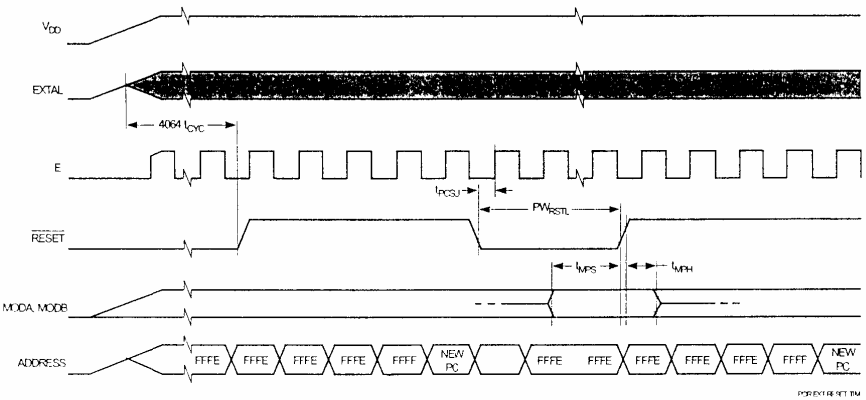
Timer Inputs



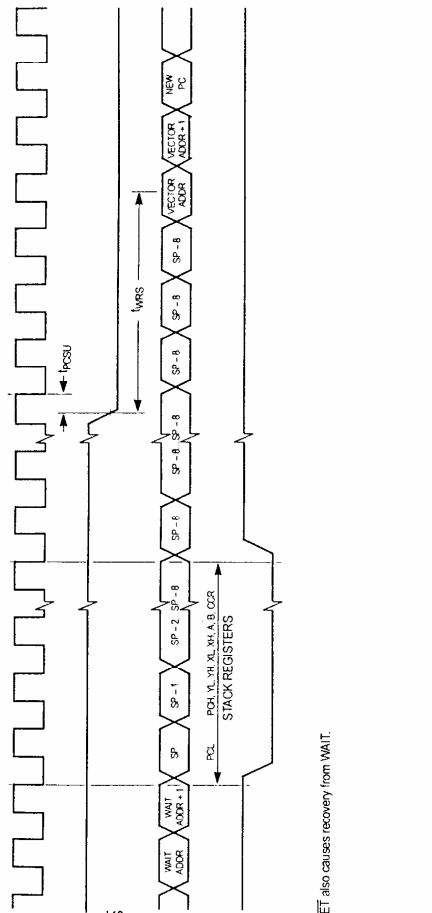
NOTES:
1. Edge Sensitive $\overline{IRQ}$ pin (IRQE bit = 1)
2. Level sensitive $\overline{IRQ}$ pin (IRQE bit = 0)
3. $t_{STOPDELAY}$ = 4064 t_{CYC} if DLY bit = 1 or 4 t_{CYC} if DLY = 0.
4. $\overline{XIRQ}$ with X bit in CCR = 1.
5. $\overline{IRQ}$ or ($\overline{XIRQ}$ with X bit in CCR = 0)

STOP

STOP Recovery Timing Diagram



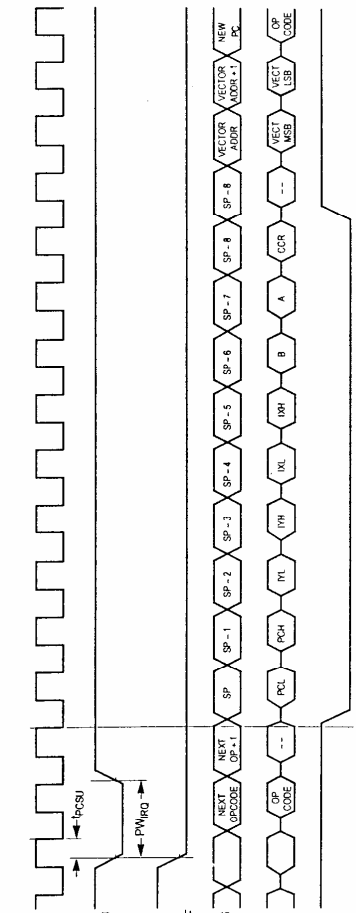
POR and External Reset Timing Diagram



NOTE: RESET also causes recovery from WAIT.

WAIT RECOVERY TIM

WAIT Recovery Timing Diagram



NOTES:
1. Edge sensitive IRQ pin (IRQE bit = 1)
2. Level sensitive IRQ pin (IRQE bit = 0)

INTERRUPT TIM

Interrupt Timing Diagram

Peripheral Port Timing

V_{DD} = 5.0 Vdc ± 10%, V_{SS} = 0 Vdc, $T_A = T_L$ to T_H

Characteristic	Symbol	1.0 MHz		2.0 MHz		3.0 MHz		Unit
		Min	Max	Min	Max	Min	Max	
Frequency of Operation (E-Clock Frequency)	f_o	dc	1.0	dc	2.0	dc	3.0	MHz
E-Clock Period	t_{cyc}	1000	—	500	—	333	—	ns
Peripheral Data Setup Time (MCU Read of Ports A, C, D, and E)	t_{PDSU}	100	—	100	—	100	—	ns
Peripheral Data Hold Time (MCU Read of Ports A, C, D, and E)	t_{PDH}	50	5	0	5	0	—	ns
Delay Time, Peripheral Data Write MCU Write to Port A	t_{PWD}	—	200	—	200	—	200	ns
MCU Writes to Ports B, C, and D t_{PWD} = 1/4 t_{cyc} + 100 ns		—	350	—	225	—	183	
Input Data Setup Time (Port C)	t_{IS}	60	6	0	6	0	—	ns
Input Data Hold Time (Port C)	t_{IH}	100	—	100	—	100	—	ns
Delay Time, E Fall to STRB t_{DEB} = 1/4 t_{cyc} + 100 ns	t_{DEB}	—	350	—	225	—	183	ns
Setup Time, STRA Asserted to E Fall (Note 1)	t_{AES}	0	0	0	—	n		s
Delay Time, STRA Asserted to Port C Data Output Valid	t_{PCD}	—	100	—	100	—	100	ns
Hold Time, STRA Negated to Port C Data	t_{PCH}	10	1	0	1	0	—	ns
Three-State Hold Time	t_{PCZ}	—	150	—	150	—	150	ns

NOTES:

1. If this setup time is met, STRB acknowledges in the next cycle. If it is not met, the response may be delayed one more cycle.
2. Port C and D timing is valid for active drive (CWOM and DWOM bits not set in PIOC and SPCR registers respectively).
3. All timing is shown with respect to 20% V_{DD} and 70% V_{DD}, unless otherwise noted.

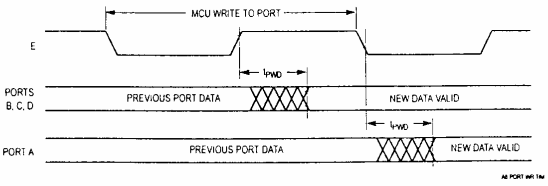


Figure A-7 Port Write Timing Diagram

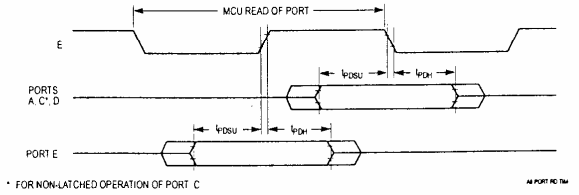
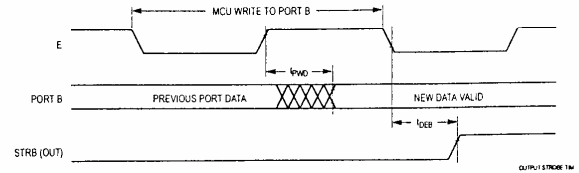


Figure A-8 Port Read Timing Diagram



Simple Output Strobe Timing Diagram

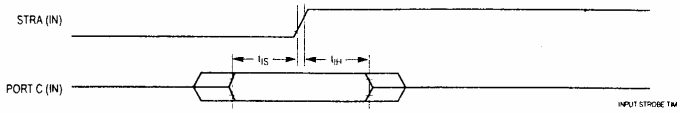
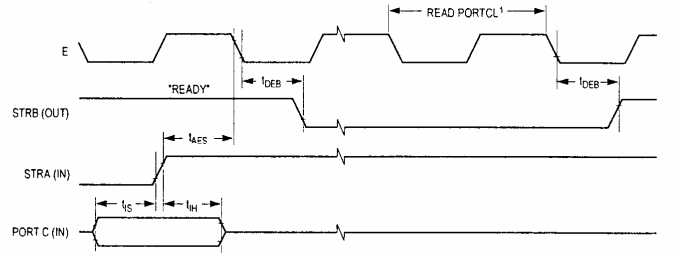
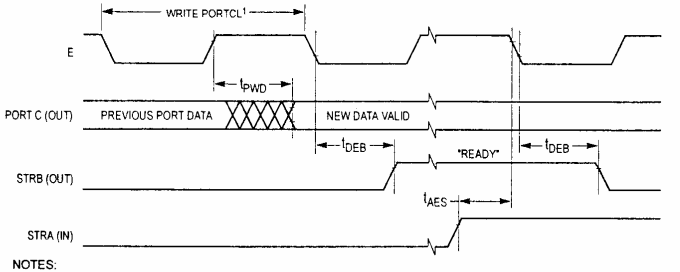


Figure A-10 Simple Input Strobe Timing Diagram



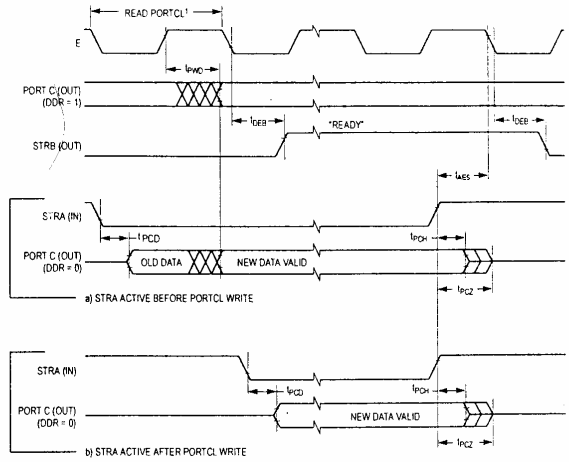
NOTES:
1. After reading PIOC with STAF set
2. Figure shows rising edge STRA (EGA = 1) and high true STRB (INVB = 1).

Figure A-11 Port C Input Handshake Timing Diagram



NOTES:
1. After reading PIOC with STAF set
2. Figure shows rising edge STRA (EGA = 1) and high true STRB (INVB = 1).

Port C Output Handshake Timing Diagram



NOTES:
1. After reading PIOC with STAF set
2. Figure shows rising edge STRA (EGA = 1) and high true STRB (INVB = 1).

Three-State Variation of Output Handshake Timing Diagram (STRA Enables Output Buffer)

Expansion Bus Timing

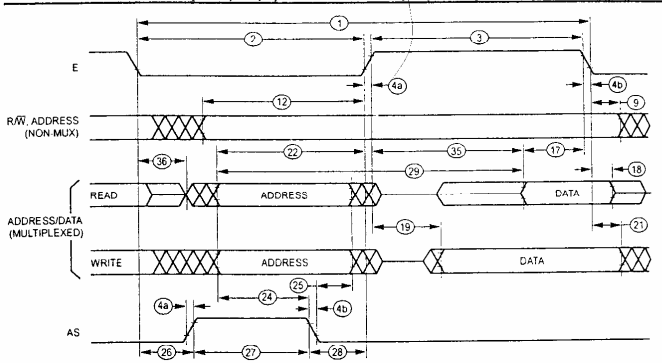
V_{DD} = 5.0 Vdc ± 10%, V_{SS} = 0 Vdc, $T_A = T_L$ to T_H

Num	Characteristic	Symbol	1.0 MHz		2.0 MHz		3.0 MHz		Unit
			Min	Max	Min	Max	Min	Max	
	Frequency of Operation (E-Clock Frequency)	f_o	dc	1.0	dc	2.0	dc	3.0	MHz
1	Cycle Time	t_{cyc}	1000	—	500	—	333	—	ns
2	Pulse Width, E Low PW_{EL} = 1/2 t_{cyc} – 23 ns(Note 1)	PW_{EL}	477	—	227	—	146	—	ns
3	Pulse Width, E High PW_{EH} = 1/2 t_{cyc} – 28 ns(Note 1)	PW_{EH}	472	—	222	—	141	—	ns
4a, b	E and AS Rise and Fall Time	t_r t_f	—	20	—	20	—	15	ns
9	Address Hold Time t_{AH} = 1/8 t_{cyc} – 29.5 ns (Note 1, 2a)	t_{AH}	95.5	—	33	—	26	—	ns
12	Non-Muxed Address Valid Time to E Rise t_{AV} = PW_{EL} – (t_{ASD} + 80 ns)(Note 1, 2a)	t_{AV}	281.5	—	94	—	54	—	ns
17	Read Data Setup Time	t_{DSR}	30	— 3	0	— 3	0	—	ns
18	Read Data Hold Time (Max = t_{MAD}) t	$_{DHR}$	0	145.5	0	83	0	51	ns
19	Write Data Delay Time t_{DDW} = 1/8 t_{cyc} + 65.5 ns (Note 1, 2a)	t_{DDW}	—	190.5	—	128	—	71	ns
21	Write Data Hold Time t_{DHW} = 1/8 t_{cyc} – 29.5 ns(Note 1, 2a)	t_{DHW}	95.5	—	33	—	26	—	ns
22	Muxed Address Valid Time to E Rise t_{AVM} = PW_{EL} – (t_{ASD} + 90 ns)(Note 1, 2a)	t_{AVM}	271.5	—	84	—	54	—	ns
24	Muxed Address Valid Time to AS Fall t_{ASL} = PW_{ASH} – 70 ns(Note 1)	t_{ASL}	151	— 2 6	— 1	3	—		ns
25	Muxed Address Hold Time t_{AHL} = 1/8 t_{cyc} – 29.5 ns(Note 1, 2b)	t_{AHL}	95.5	—	33	—	31		ns
26	Delay Time, E to AS Rise t_{ASD} = 1/8 t_{cyc} – 9.5 ns (Note 1, 2a)	t_{ASD}	115.5	—	53	—	31	—	ns
27	Pulse Width, AS High PW_{ASH} = 1/4 t_{cyc} – 29 ns(Note 1)	PW_{ASH}	221	— 9 6	— 6	3	—		ns
28	Delay Time, AS to E Rise t_{ASED} = 1/8 t_{cyc} – 9.5 ns(Note 1, 2b)	t_{ASED}	115.5	—	53	—	31	—	ns
29	MPU Address Access Time(Note 2a) t_{ACCA} = t_{cyc} – (PW_{EL} – t_{AVM}) – t_{DSR} – t_f	t_{ACCA}	744.5	—	307	—	196	—	ns
35	MPU Access Time t_{ACCE} = PW_{EH} – t_{DSR}	t_{ACCE}	—	442	—	192	—	111	ns
36	Muxed Address Delay (Previous Cycle MPU Read) t_{MAD} = t_{ASD} + 30 ns(Note 1, 2a)	t_{MAD}	145.5	—	83	—	51	—	ns

NOTES:

1. Formula only for dc to 2 MHz.
2. Input clocks with duty cycles other than 50% affect bus performance. Timing parameters affected by input clock duty cycle are identified by (a) and (b). To recalculate the approximate bus timing values, substitute the following expressions in place of 1/8 t_{cyc} in the above formulas, where applicable:
   (a) (1-DC) x 1/4 t_{cyc}
   (b) DC x 1/4 t_{cyc}

   Where:
   DC is the decimal value of duty cycle percentage (high time).
3. All timing is shown with respect to 20% V_{DD} and 70% V_{DD}, unless otherwise noted.



NOTE: Measurement points shown are 20% and 70% of V_{DD}.

MUX BUS TIM

Multiplexed Expansion Bus Timing Diagram

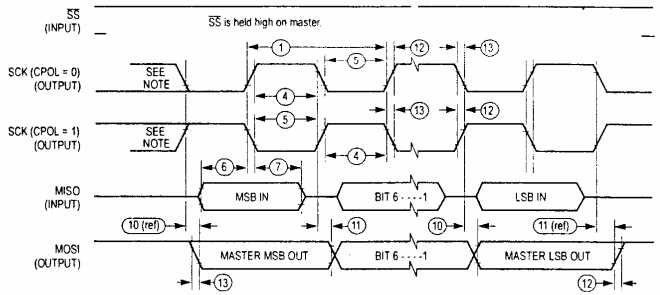
Serial Peripheral Interface (SPI) Timing

V_{DD} = 5.0 Vdc ± 10%, V_{SS} = 0 Vdc, $T_A = T_L$ to T_H

Num	Characteristic	Symbol	2.0 MHz		3.0 MHz		Unit
			Min	Max	Min	Max	
	Operating Frequency Master Slave	$f_{op(m)}$ $f_{op(s)}$	dc dc	0.5 2.0	dc dc	0.5 3.0	f_{op} MHz
1	Cycle Time Master Slave	$t_{cyc(m)}$ $t_{cyc(s)}$	2.0 500	—	2.0 333	—	t_{cyc} ns
2	Enable Lead Time Master (Note 2) Slave	$t_{lead(m)}$ $t_{lead(s)}$	— 250	—	— 240	—	ns ns
3	Enable Lag Time Master (Note 2) Slave	$t_{lag(m)}$ $t_{lag(s)}$	— 250	—	— 240	—	ns ns
4	Clock (SCK) High Time Master Slave	$t_{w(SCKH)m}$ $t_{w(SCKH)s}$	340 190	— —	227 127	— —	ns ns
5	Clock (SCK) Low Time Master Slave	$t_{w(SCKL)m}$ $t_{w(SCKL)s}$	340 190	— —	227 127	— —	ns ns
6	Data Setup Time (Inputs) Master Slave	$t_{su(m)}$ $t_{su(s)}$	100 100	— —	100 100	— —	ns ns
7	Data Hold Time (Inputs) Master Slave	$t_{h(m)}$ $t_{h(s)}$	100 100	— —	100 100	— —	ns ns
8	Access Time (Time to Data Active from High-Impedance State) Slave	t_a	0	120	0	120	ns
9	Disable Time (Hold Time to High-Impedance State) Slave	t_{dis}	—	240	—	167	ns
10	Data Valid (After Enable Edge)(Note 3)	$t_{v(s)}$	—	240	—	167	ns
11	Data Hold Time (Outputs) (After Enable Edge)	t_{ho}	0	— 0	—		ns
12	Rise Time (20% V_{DD} to 70% V_{DD}, CL = 200 pF) SPI Outputs (SCK, MOSI, and MISO) SPI Inputs (SCK, MOSI, MISO, and SS)	t_{rm} t_{rs}	— —	100 2.0	— —	100 2.0	ns µs
13	Fall Time (70% V_{DD} to 20% V_{DD}, C_L = 200 pF) SPI Outputs (SCK, MOSI, and MISO) SPI Inputs (SCK, MOSI, MISO, and SS)	t_{fm} t_{fs}	— —	100 2.0	— —	100 2.0	ns µs

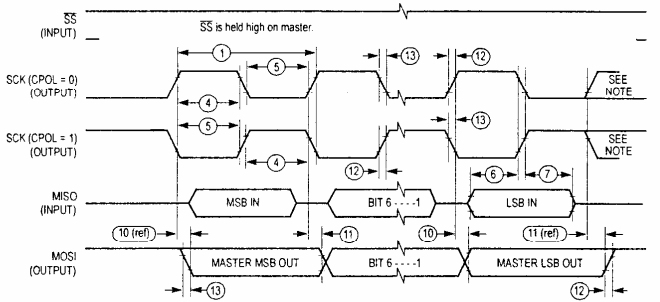
NOTES:

1. All timing is shown with respect to 20% V_{DD} and 70% V_{DD}, unless otherwise noted.
2. Signal production depends on software.
3. Assumes 200 pF load on all SPI pins.



NOTE: This first clock edge is generated internally but is not seen at the SCK pin.

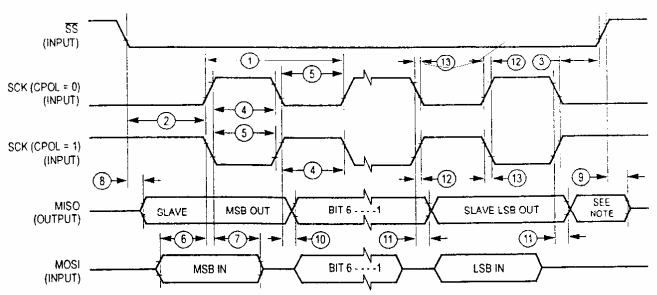
a) SPI Master Timing (CPHA = 0)



NOTE: This last clock edge is generated internally but is not seen at the SCK pin.

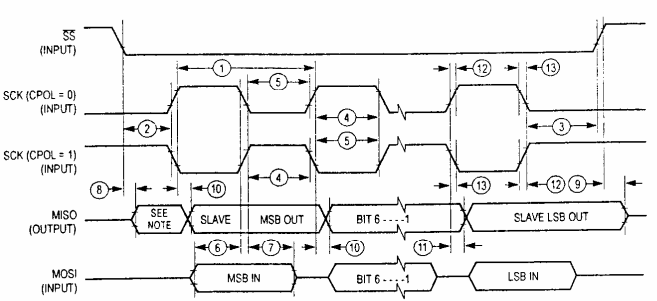
b) SPI Master Timing (CPHA = 1)

SPI Timing Diagram (1 of 2)



NOTE: Not defined but normally MSB of character just received.

c) SPI Slave Timing (CPHA = 0)



NOTE: Not defined but normally LSB of character previously transmitted.

d) SPI Slave Timing (CPHA = 1)

SPI Timing Diagrams (2 of 2)

Литература:

1. M68HC11 Reference Manual, Motorola Inc. 1990
2. MC68HC11A8 HCMOS single-chip Microcontroller, Motorola Inc. 1990
3. M68HC11EVB User's Manual, Motorola Inc.1989
4. Иванов Р., Г. Михов. Електронни цифрови устройства и системи. София, Техника, 1988.
5. Михов Г. Цифрова схемотехника. Технически Университет – София, 1997
6. Driscoll F., R. Coughlin, R. Vilanucci. Data Acquisition and Process Control with the M68HC11 microcontroller, Maxwell Macmillan International, 1994, ISBN 0-02-330555-X
7. P. Spasov. Microcontroller Technology the 68HC11. Prentice Hall, 1996. ISBN 0-13-362724-1
8. Hintz K., D. Tabak. Microcontrolers Architecture, Implementation and Programming. McGraw-Hill, 1992, ISBN 0-07-028977-8

Съдържание