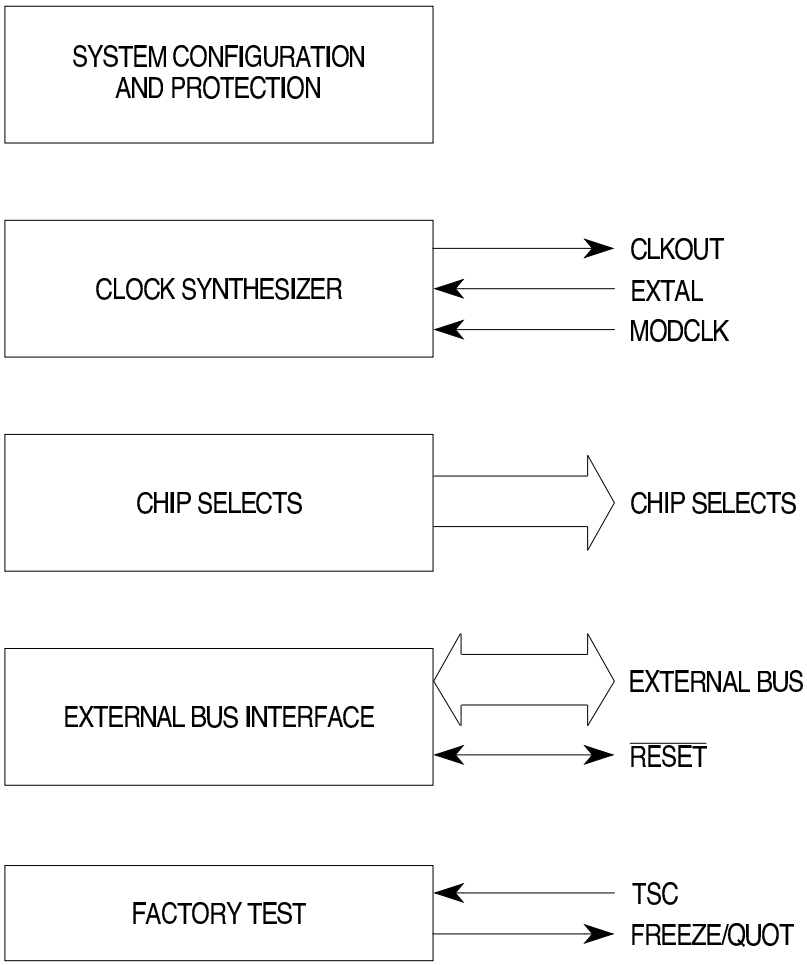
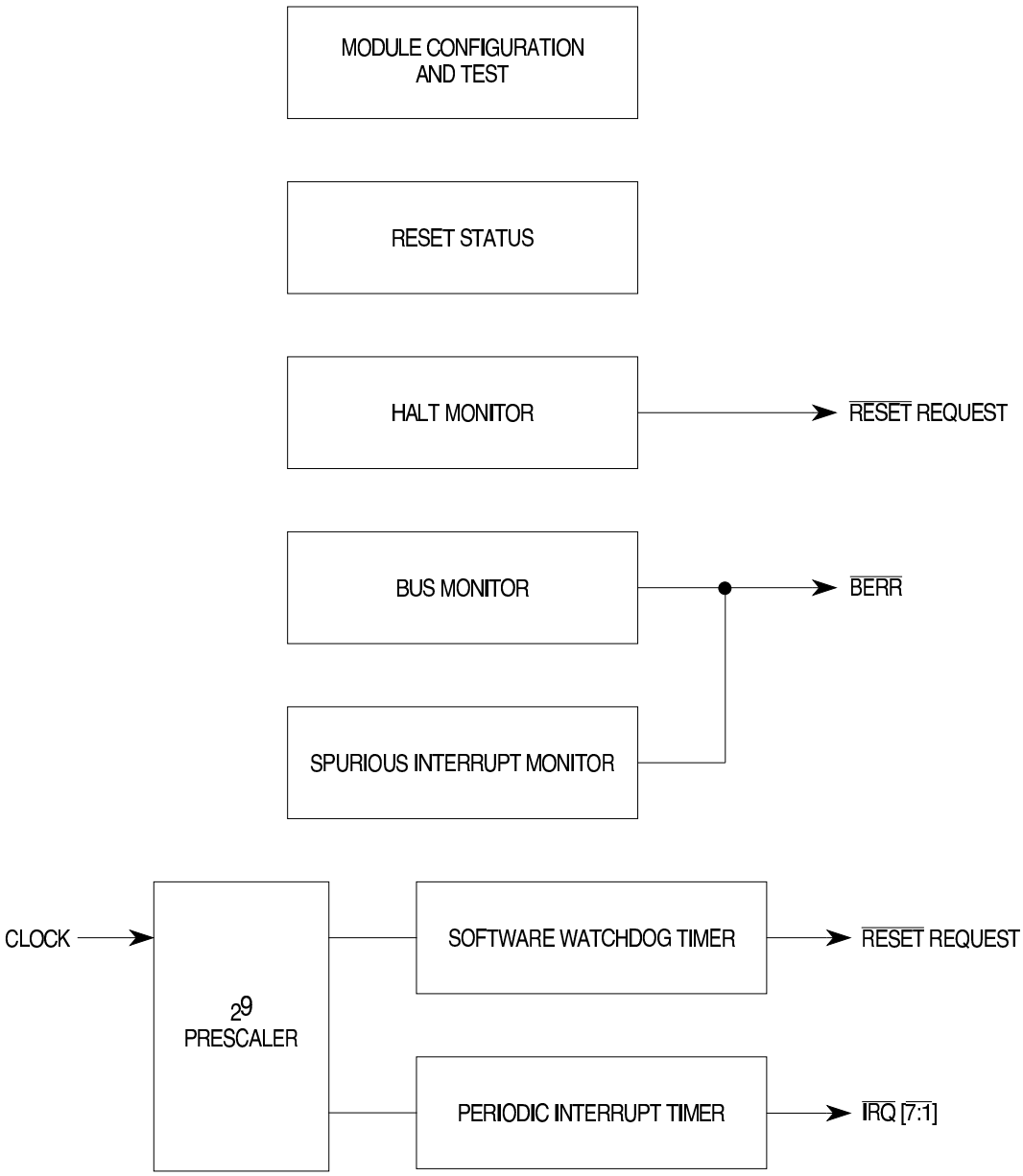
Struktura SIM (*System Integration Module*)



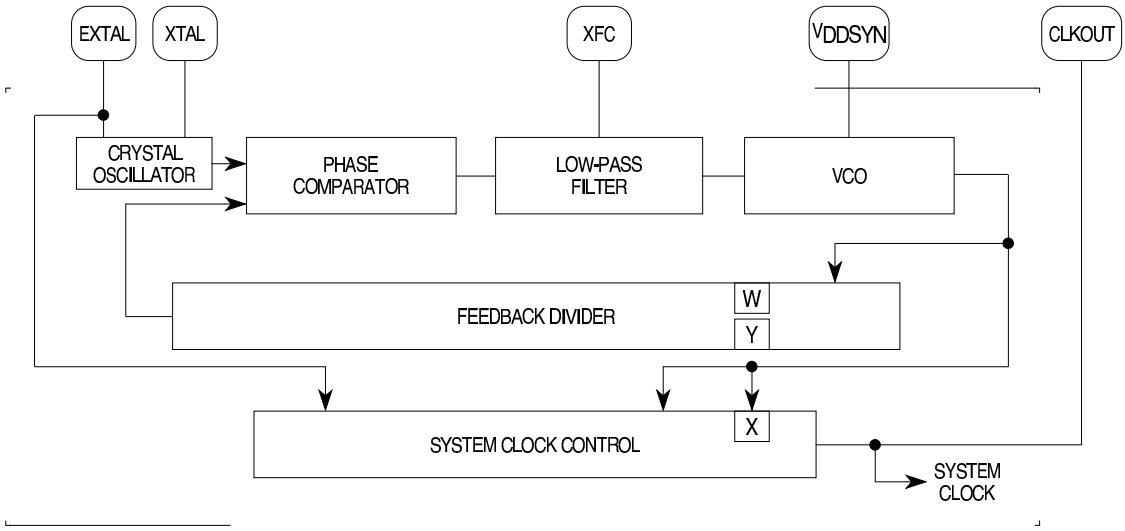
SIMCR — Module Configuration Register **$####00**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXOFF	FRZSW	FRZBM	0	SLVEN	0	SHEN		SUPV	MM	0	0	IARB			
RESET:															
0	0	0	0	DATA 11	0	0	0	1	1	0	0	1	1	1	1

Moduł konfiguracji i zabezpieczeń



Pętla fazowa syntezatora zegara



SYNCR — Clock Synthesizer Control Register **$####04**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
W	X	Y						EDIV	0	0	SLIMP	SLOCK	RSTEN	STSIM	STEXT
RESET:															
0	0	1	1	1	1	1	1	0	0	0	U	U	0	0	0

$$f_{SYS} = \begin{cases} f_{EXT} \cdot (Y+1) \cdot 2^{2W+X+2} & dla \quad 25 \le f_{EXT} \le 50\ [kHz] \\ \frac{f_{EXT}}{128} \cdot (Y+1) \cdot 2^{2W+X+2} & dla \quad 3.2 \le f_{EXT} \le 6.4\ [MHz] \end{cases}$$

Układ *Watchdog* i monitor magistrali

SWSR — Software Service Register **$####26**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NOT USED								0	0	0	0	0	0	0	0
RESET								0	0	0	0	0	0	0	0

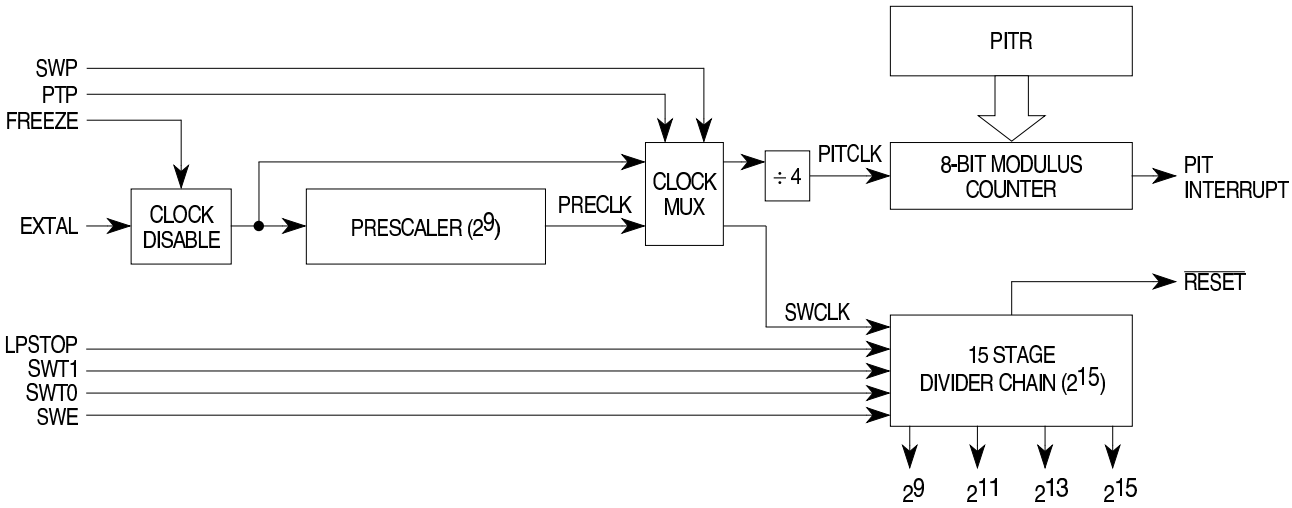
SYPCR — System Protection Control Register **$####20**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NOT USED								SWE	SWP	SWT		HME	BME	BMT	
RESET:								1		0	0	0	0	0	0

$$T_{WDOG} = 512 \cdot \frac{2^{2 \cdot SWT + 9 \cdot SWP}}{f_{EXTAL}}$$

$$T_{BUSMON} = \frac{2^{6 - BMT}}{f_{SYS}}$$

Układ generacji przerwań cyklicznych (PIT)



D.2.13 PICR — Periodic Interrupt Control Register **$YFFA22**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	PIRQL			PIV							
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1

D.2.14 PITR — Periodic Interrupt Timer Register **$YFFA24**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	PTP	PITM							
RESET:															
0	0	0	0	0	0	0	$\overline{\text{MODCLK}}$	0	0	0	0	0	0	0	0

$$T_{PIT} = \frac{4 \cdot 2^{9 \cdot (PTP)} \cdot (PITM)}{f_{EXTAL}}$$

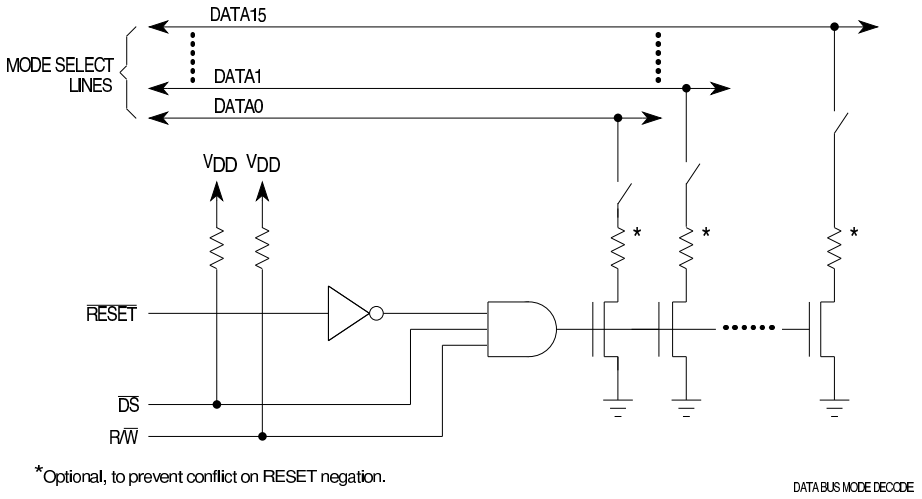
Obsługa programowa PIT

```
#define SIMBASE 0xfffffa00
#define PICR  (*((WORD *) (SIMBASE+0x22)))              /* PIT Control */
#define PITR  (*((volatile WORD *) (SIMBASE+0x24)))     /* PIT Modulus */

#define XTAL        (32768.L)                  /* czestotliwosc kwarcu */
#define PIT_TB      (XTAL/4)                 /* podstawa czasu dla PIT */
#define TPSEC       64                     /* ilosc przerwan na sekunde */
/* wyliczenie stalej dla dzielnika PITR */
#define PI_CONST    ((PIT_TB+(TPSEC>>1))/TPSEC)

#define PI_LEVEL    6                        /* poziom przerwania PIT */
#define PI_VECT     111                      /* wektor przerwania PIT */

int tick;                                          /* licznik taktow */
long secs;                                         /* licznik sekund */

interrupt void myclock()              /* procedura obslugi przerwania */
{
    if(++tick>=TPSEC){ tick = 0; secs += 1; }
}

main()                                      /* glowna funkcja programu */
{
   tick = secs = 0;               /* inicjalizacja zmiennych globalnych */
   *((void (**) ())(4*PI_VECT)) = myclock;        /* ustawienie wektora */
   PICR = (PI_LEVEL<<8)+PI_VECT;      /* poziom i wektor przerwania PIT */
   PITR = PI_CONST;                            /* sta/la dzielnika PIT */
   asm { ANDI #0xf8ff,SR
   }                                /* ustawienie poziomu przerwan na 0 */

   while(1) {}                                      /* petla glowna */
}
```

Reset i konfigurowanie trybu pracy mikrokontrolera

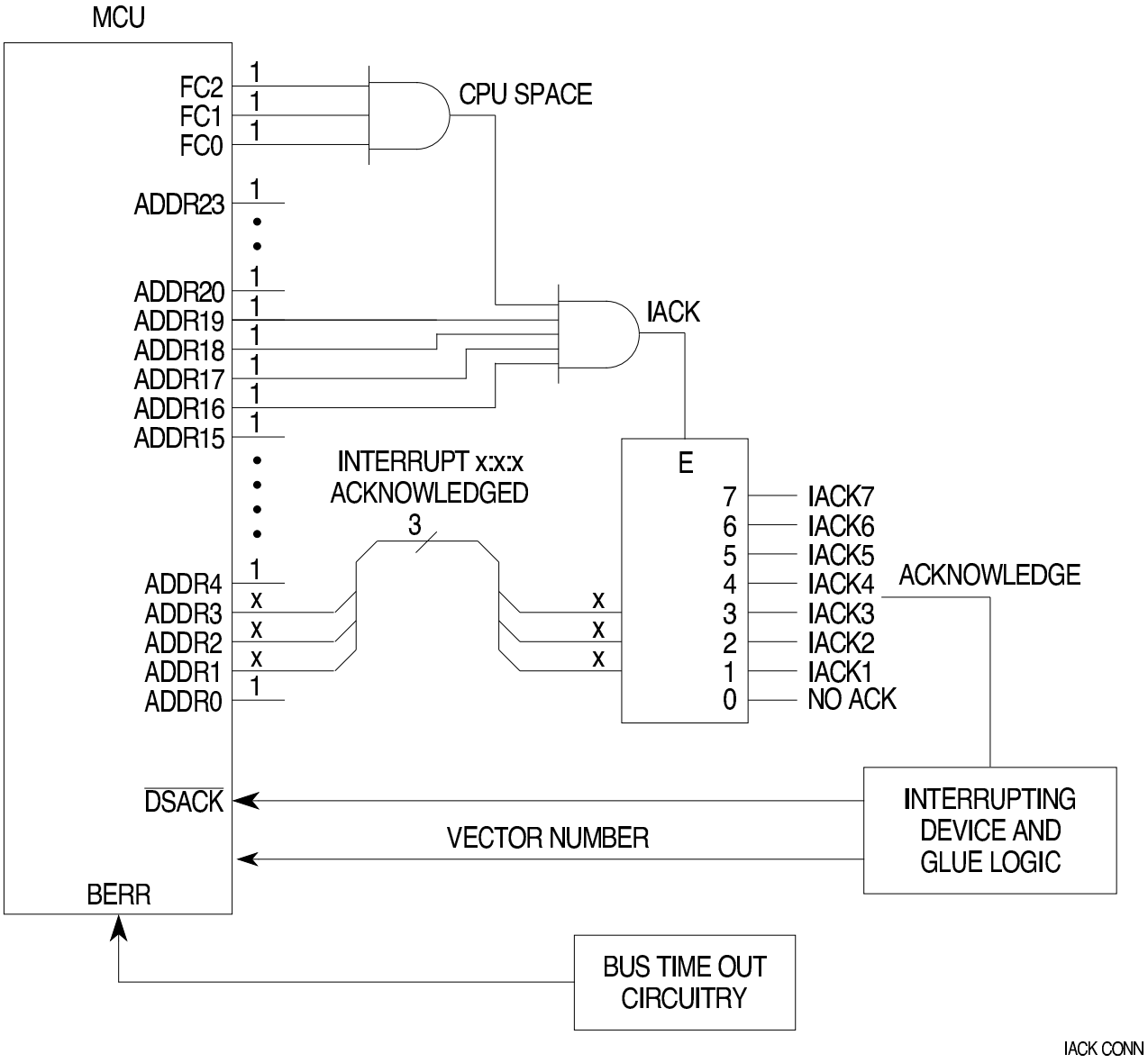
RSR — Reset Status Register **$####06**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NOT USED								EXT	POW	SW	DBF	0	LOC	SYS	TST

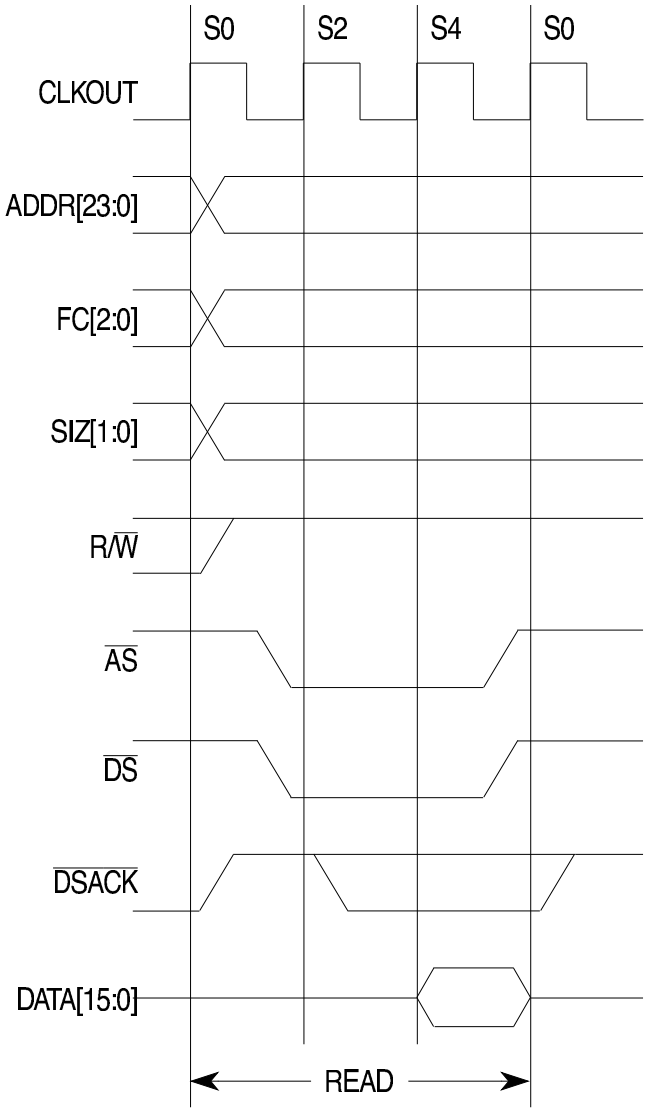


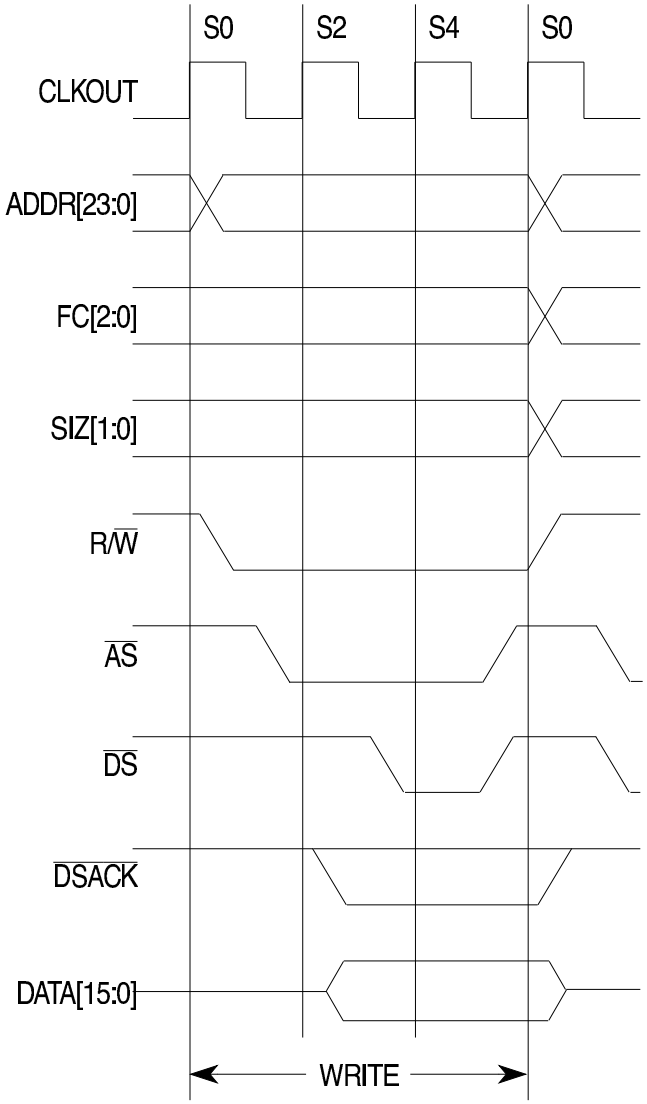
pin	stan domyślny (wysoki)	stan wymuszony (niski)
$DATA0$	$\overline{CSBOOT}$ 16-bitowy	$\overline{CSBOOT}$ 8-bitowy
$DATA1..7$	$\overline{CS}0..10$	sygnały magistrali
$DATA8$	sygnały magistrali	$PORTE$
$DATA9$	$\overline{IRQ}[7:1], MODCLK$	$PORTF$
$DATA11$	tryb testowy wyłączony	tryb testowy włączony
$MODCLK$	SystemClock = VCO	SystemClock = EXTAL
$\overline{BKPT}$	BDM wyłączony	BDM włączony

Układ obsługi przerwań

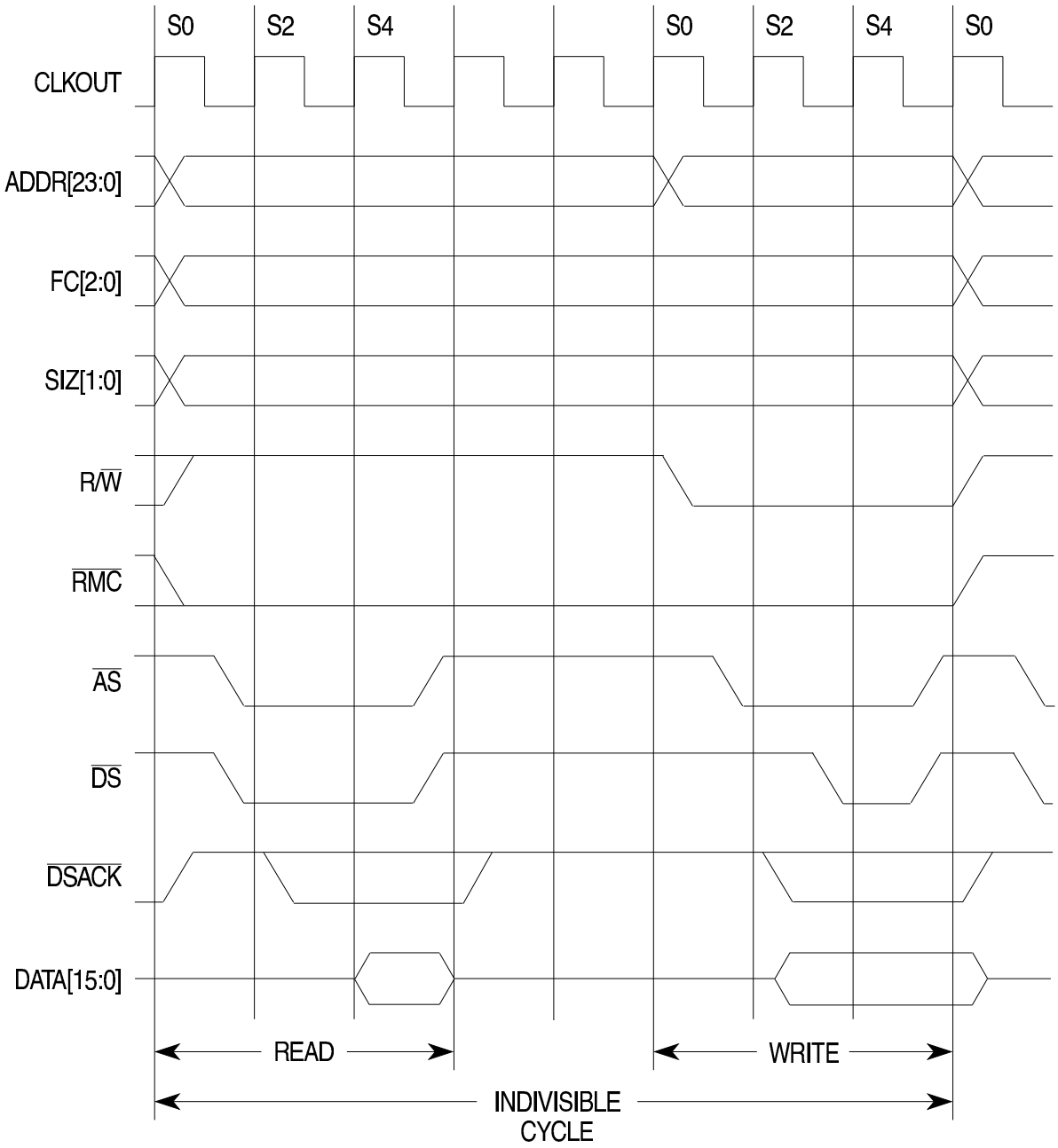


Cykle odczytu i zapisu na magistrali zewnętrznej

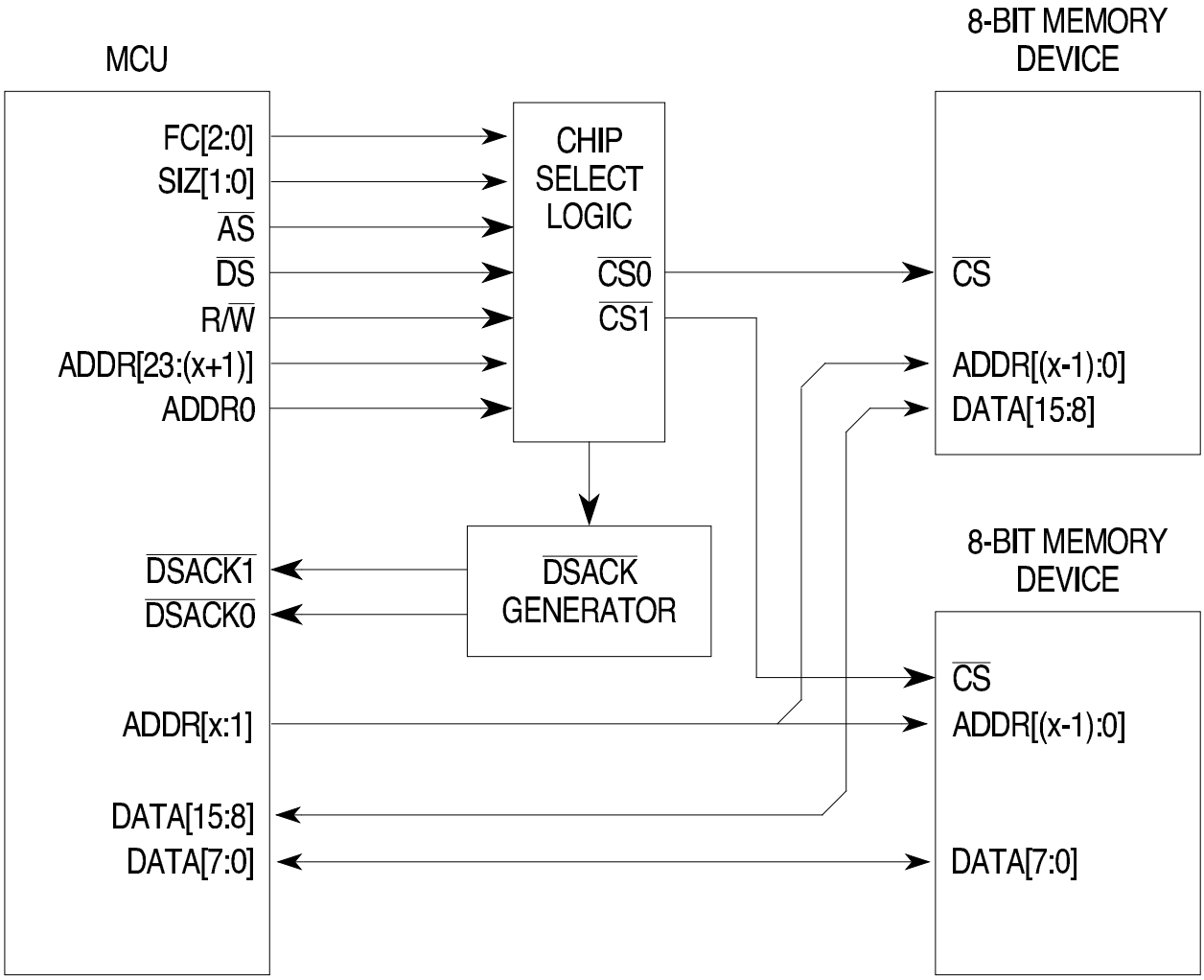




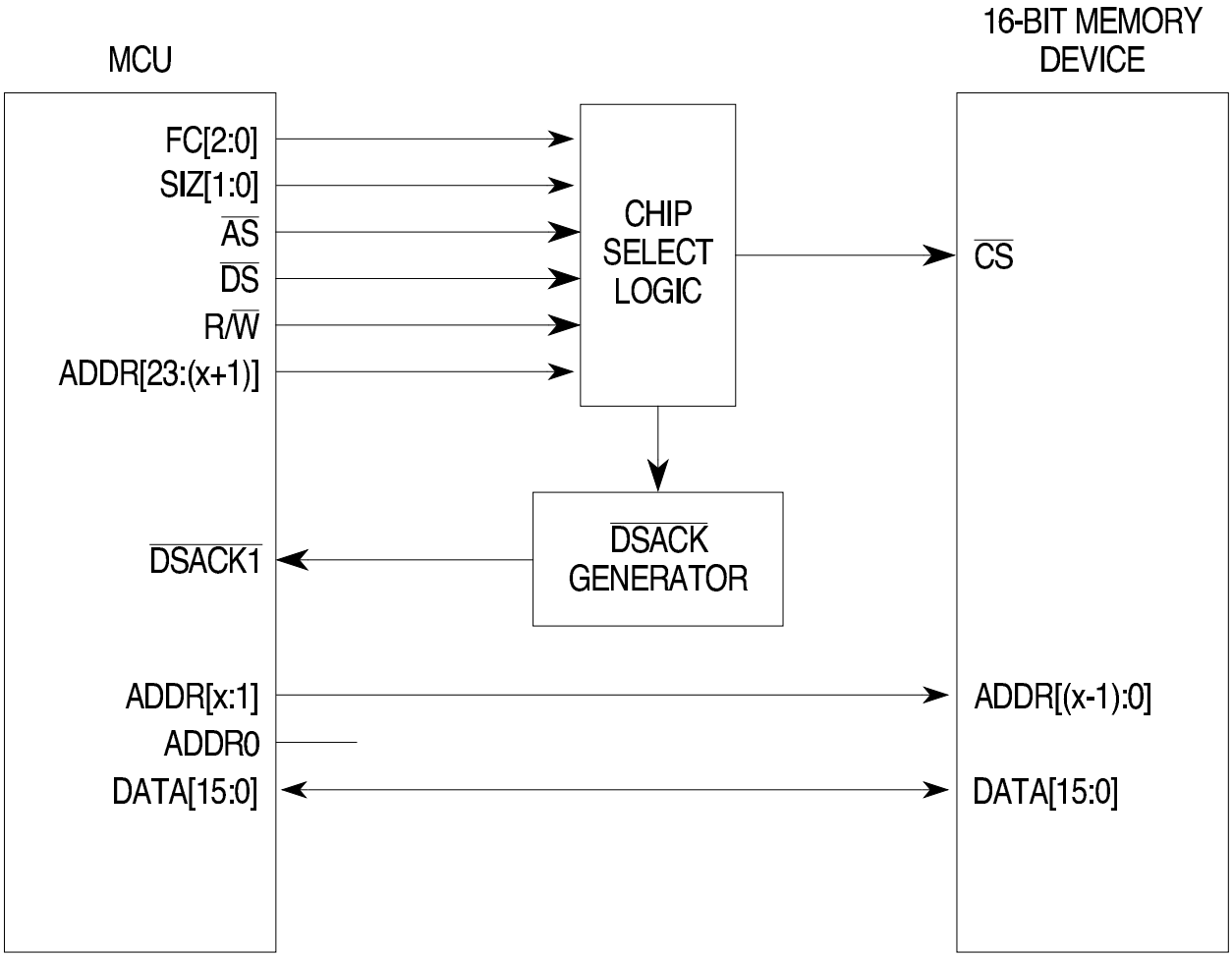
Cykl RMW (*Read–Modify–Write*)



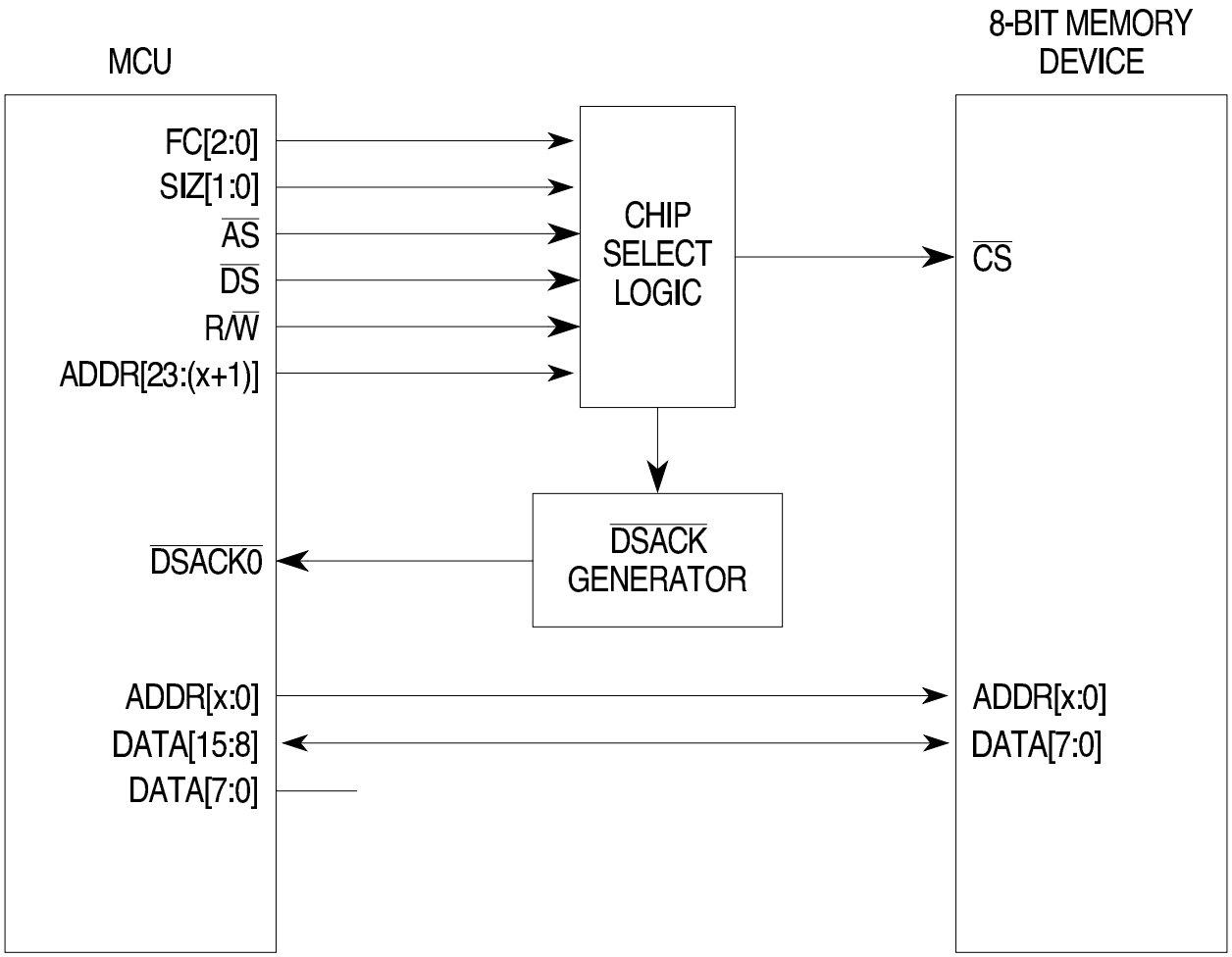
Przyłączenie dwóch pamięci ośmiobitowych



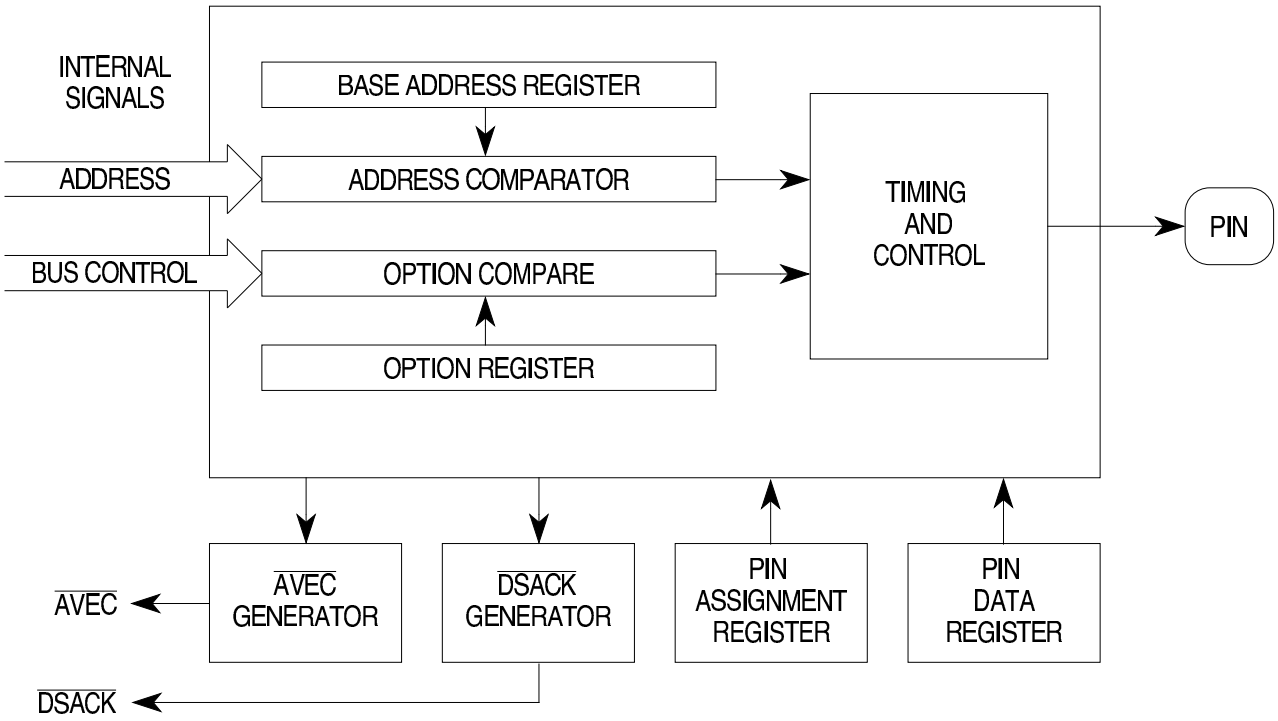
Przyłączenie pamięci szesnastobitowej



Przyłączenie pamięci ośmiobitowej



Układ generowania sygnałów wyboru urządzeń



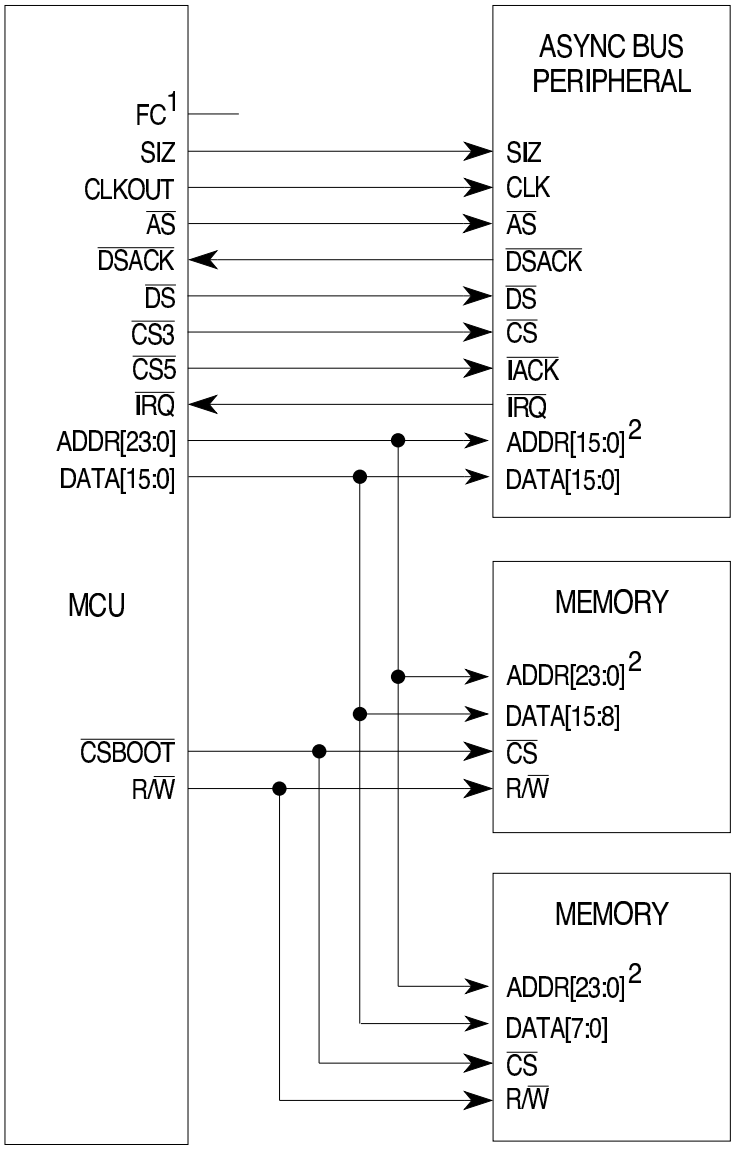
CSBARBT — Chip-Select Base Address Register Boot ROM **$YFFA48**

15	14	13	12	11	10	9	8	7	6	5	4	3	2		0
ADDR 23	ADDR 22	ADDR 21	ADDR 20	ADDR 19	ADDR 18	ADDR 17	ADDR 16	ADDR 15	ADDR 14	ADDR 13	ADDR 12	ADDR 11	BLKSZ		
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1

CSORBT —Chip-Select Option Register Boot ROM **$YFFA4A**

15	14	13	12	11	10	9		6	5	4	3		1	0
MODE	BYTE		R/W		STRB	DSACK			SPACE		IPL			AVEC
RESET:														
0	1	1	1	1	0	1	1	0	1	1	1	0	0	0	0

Przyklad konfiguracji pamięci



1. Can be decoded to provide additional address space.
2. Varies depending upon peripheral memory size.

Programowanie sygnałów CS

```
#define SIMBASE 0xfffffa00
#define CSPAR1  (*((WORD *) (SIMBASE+0x46)))     /* CS Pin Assignment 1 */
#define CSBAR9  (*((WORD *) (SIMBASE+0x70)))              /* CS9 Base */
#define CSOR9   (*((WORD *) (SIMBASE+0x72)))           /* CS9 Option */

#define LCD_IR 0xeff800                  /* adres rejestru danych LCD */
#define LCD_DR 0xeff801              /* adres rejestru sterujacego LCD */

volatile BYTE LcdIr @LCD_IR;                 /* rejestr sterujacy LCD */
volatile BYTE LcdDr @LCD_DR;                    /* rejestr danych LCD */

void PutData(BYTE data)                  /* funkcja wpisu danych do LCD */
{
    while((LcdIr & 0x80) == 0x80);           /* oczekiwanie na gotowosc */
    LcdDr = data;
}

main()                                           /* glowna funkcja programu */
{
    char *ptr = "Hello!";

    /* uruchomienie linii CS9 - wybor LCD */
    CSBAR9 = (LCD_IR & 0xfff800) >> 8;         /* adres bazowy, blok 2k */
    CSOR9  = 0xdbf0;                          /* 1 10 11 0 1111 11 000 0 */
                                              /* s ub rw d dack su  0  n */
    CSPAR1 = (CSPAR1 & 0xff3f) | 0x0080;              /* port 8-bitowy */

    while(*ptr) PutData(*ptr++);
}
```