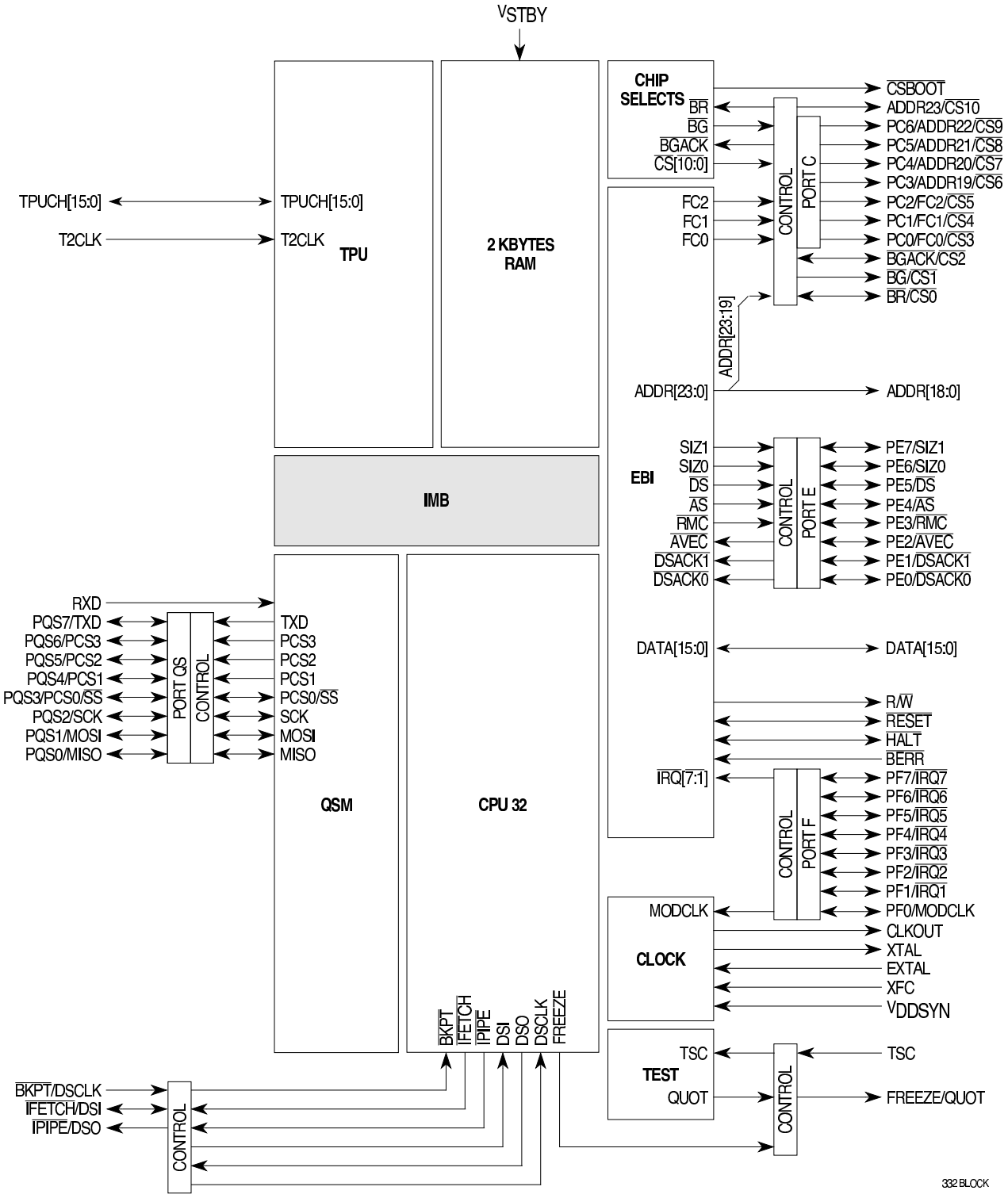
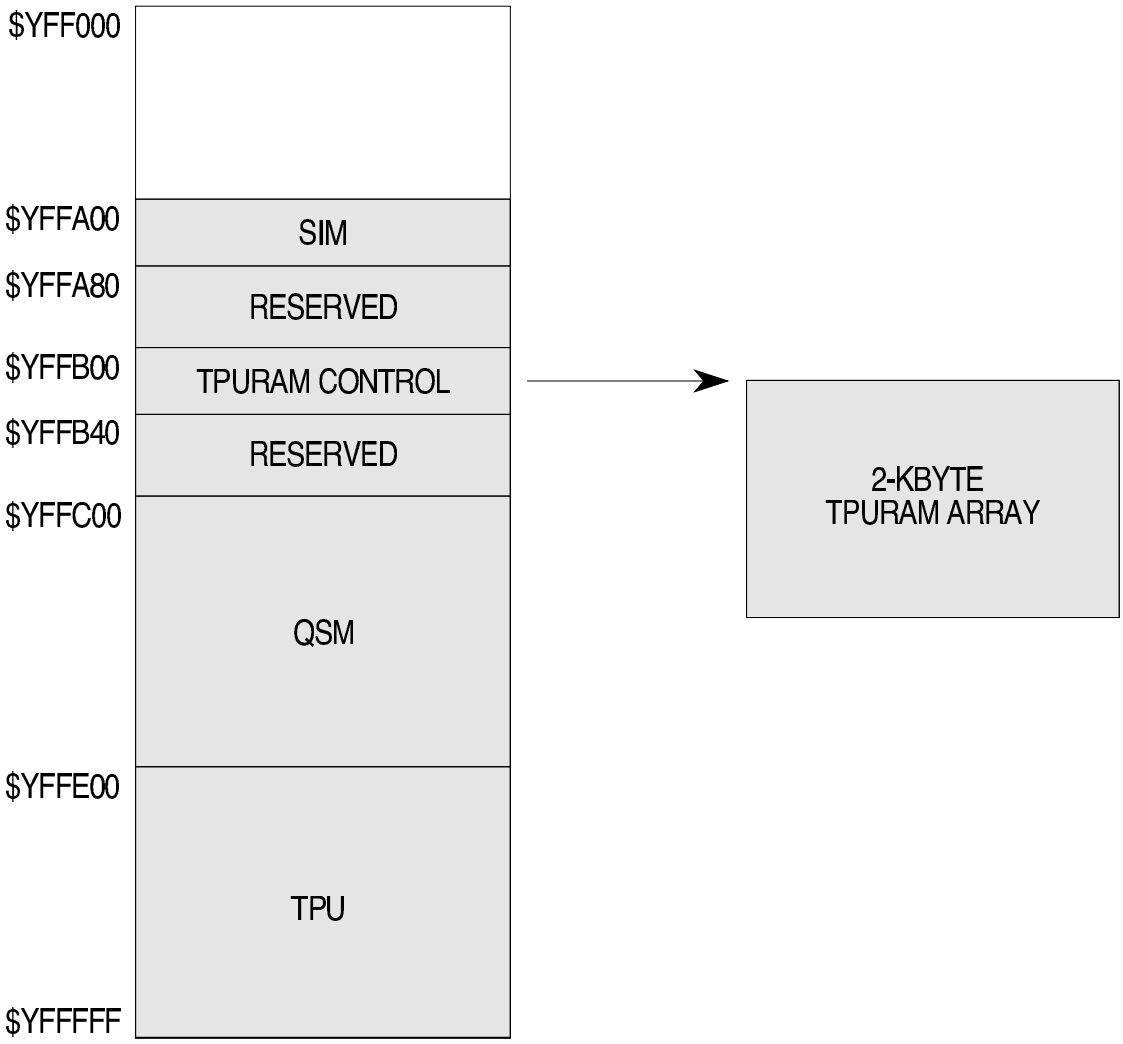
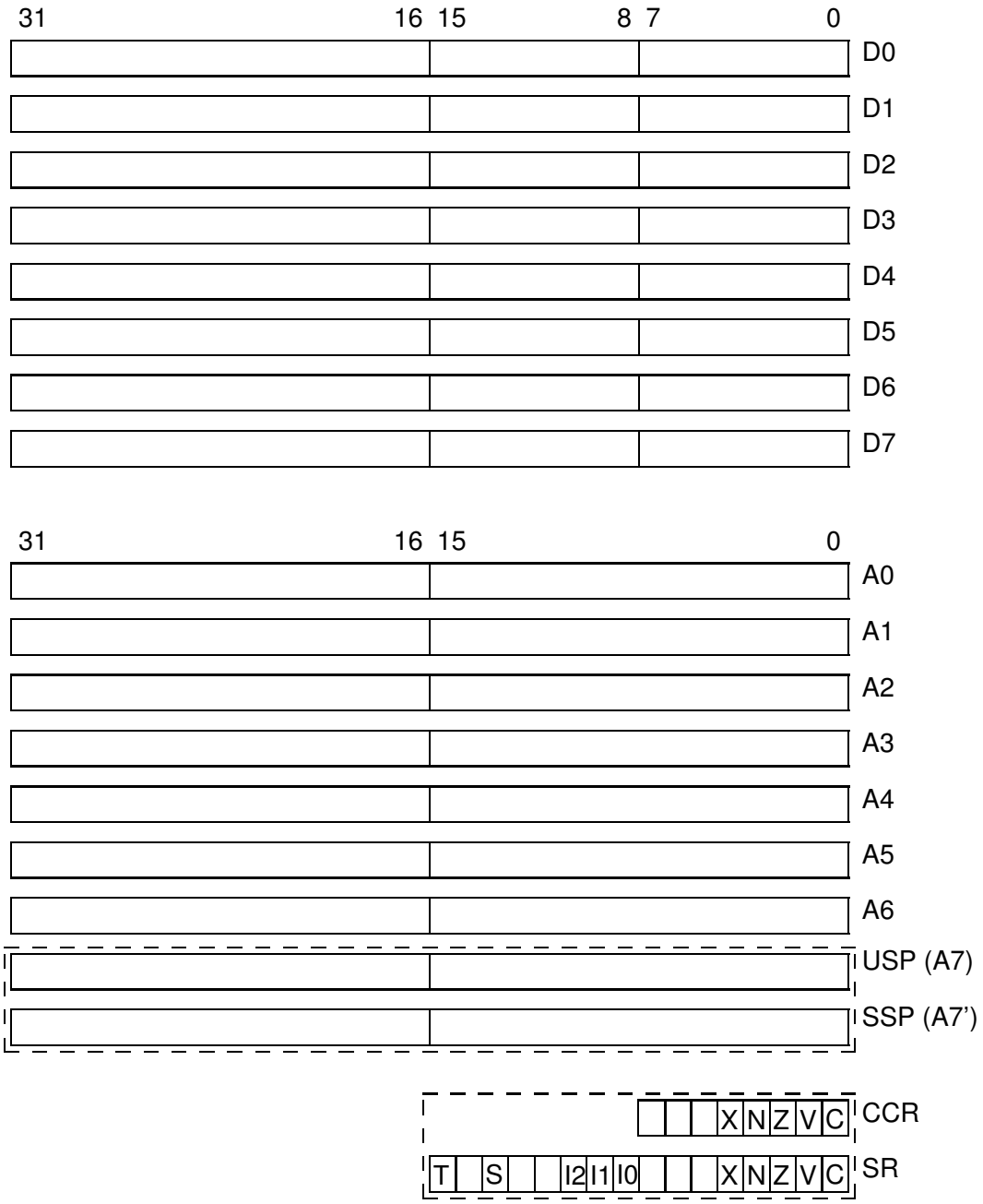
Struktura mikrokontrolera MC68332



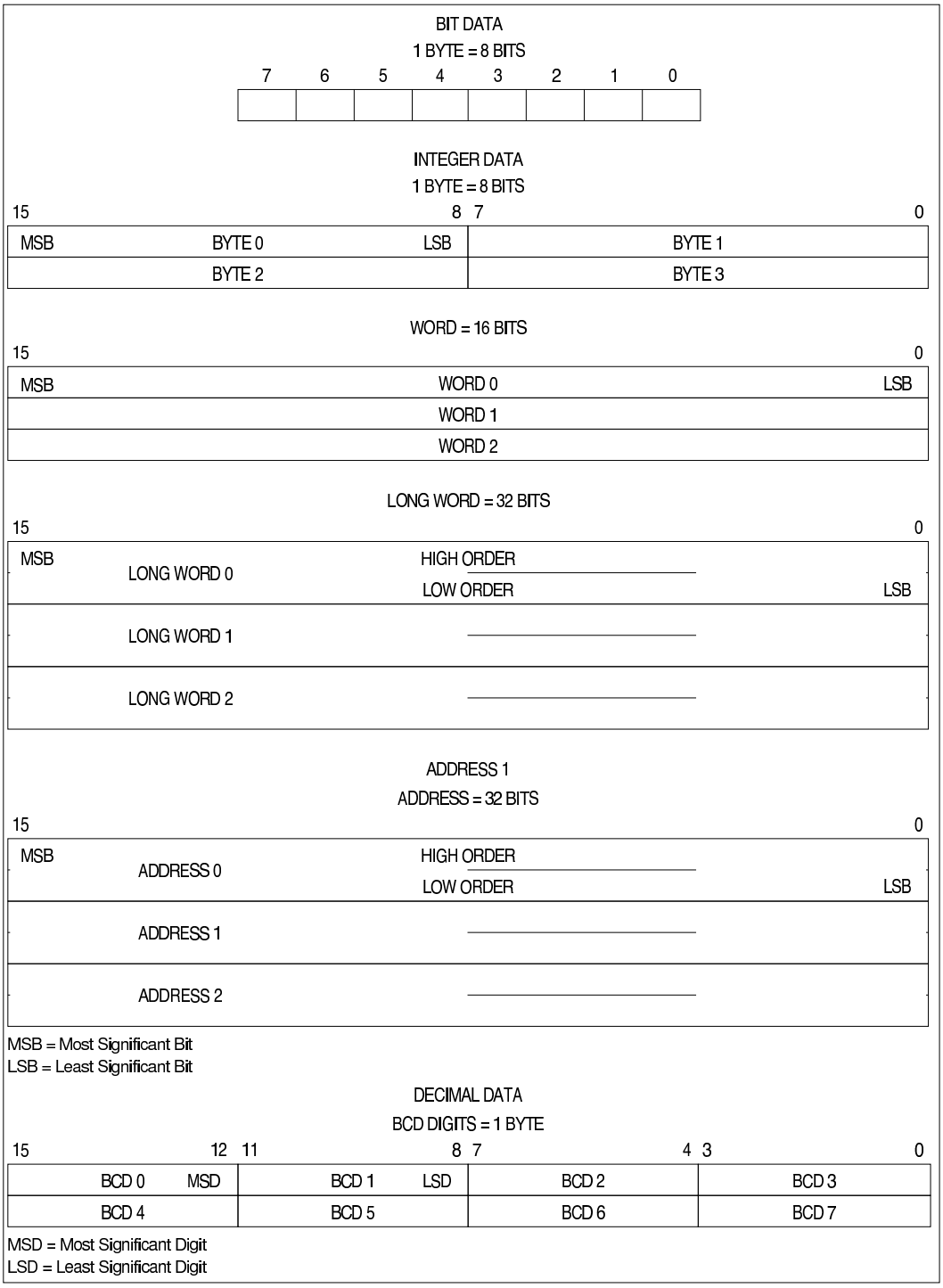
Rozmieszczenie bloków w przestrzeni aderesowej



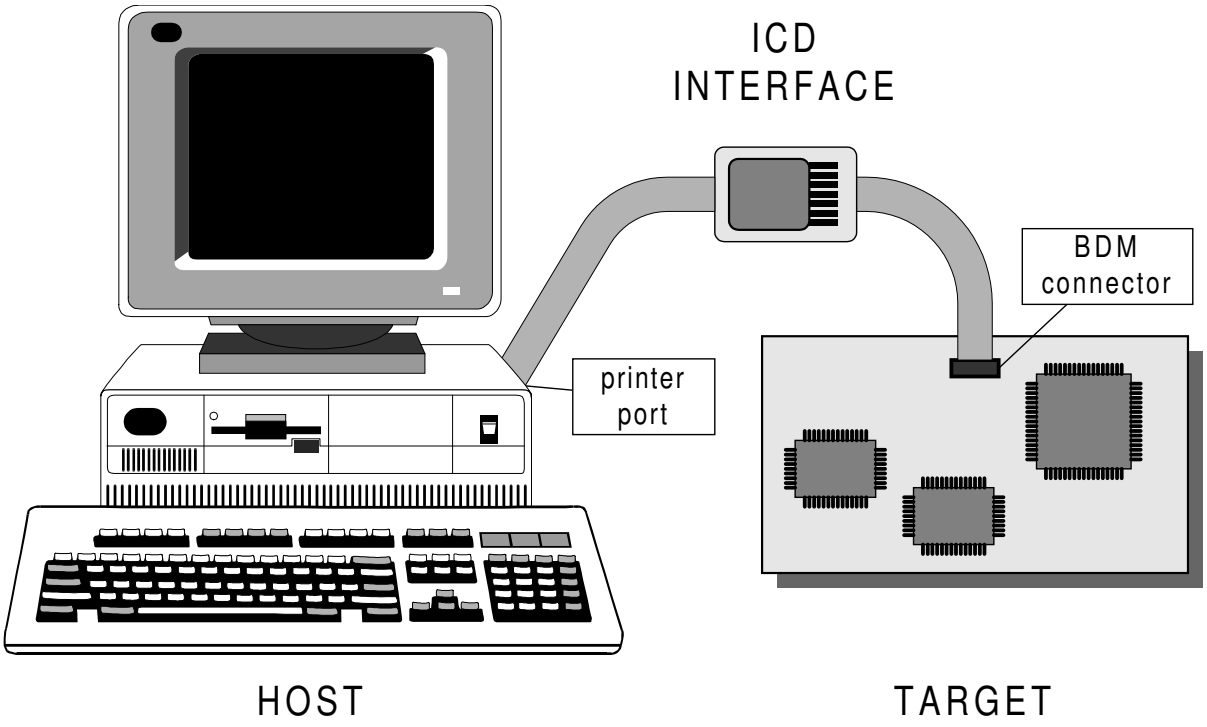
Rejestry procesora CPU32



Organizacja danych w pamięci



Interfejs ICD (*In-Circuit Debugger*) do wbudowanego układu uruchomieniowego BDM (*Background Debug Mode*)



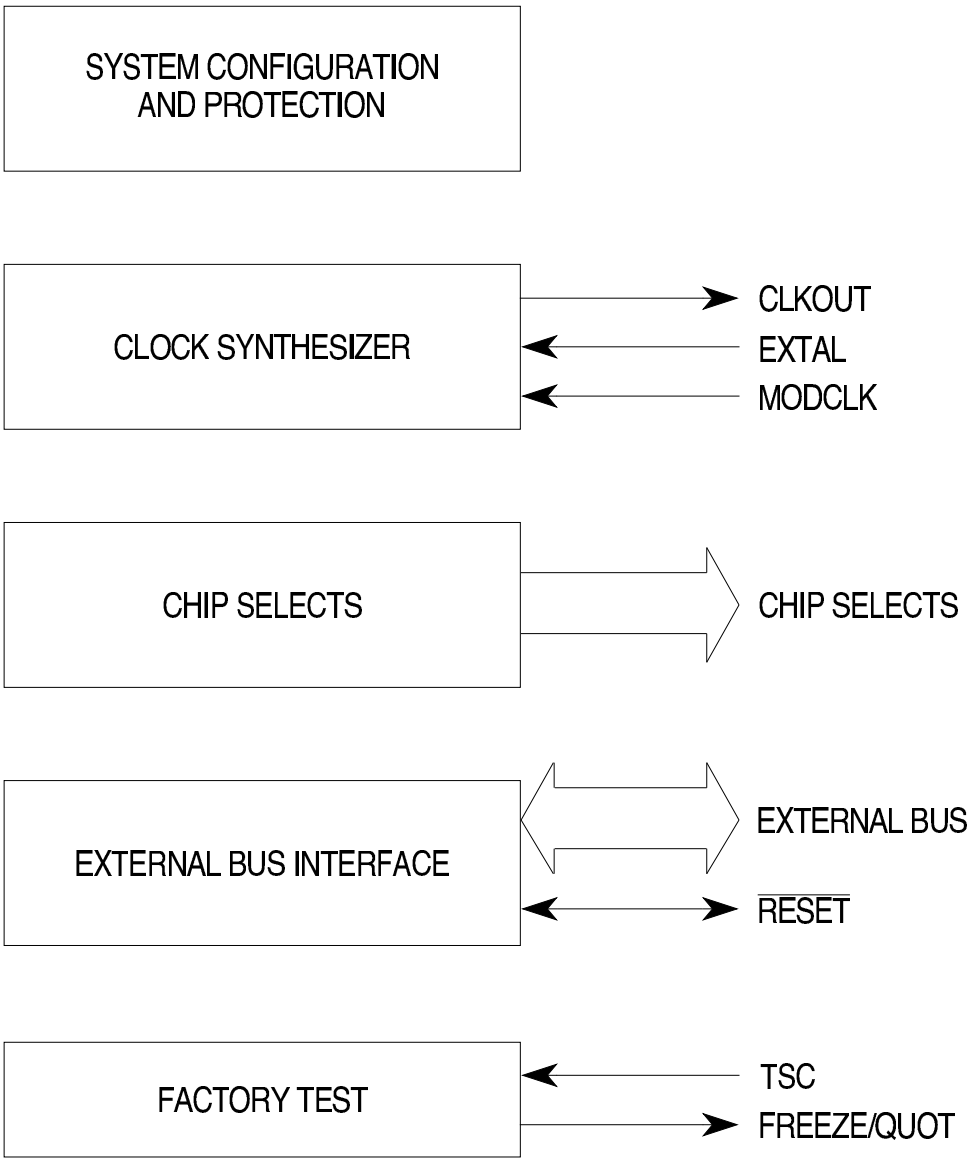
Komendy interfejsu BDM

Command	Mnemonic	Description
Read D/A Register	RDREG/RAREG	Read the selected address or data register and return the results through the serial interface.
Write D/A Register	WDREG/WAREG	The data operand is written to the specified address or data register.
Read System Register	RSREG	The specified system control register is read. All registers that can be read in supervisor mode can be read in background mode.
Write System Register	WSREG	The operand data is written into the specified system control register.
Read Memory Location	READ	Read the sized data at the memory location specified by the long-word address. The source function code register (SFC) determines the address space accessed.
Write Memory Location	WRITE	Write the operand data to the memory location specified by the long-word address. The destination function code (DFC) register determines the address space accessed.
Dump Memory Block	DUMP	Used in conjunction with the READ command to dump large blocks of memory. An initial READ is executed to set up the starting address of the block and retrieve the first result. Subsequent operands are retrieved with the DUMP command.
Fill Memory Block	FILL	Used in conjunction with the WRITE command to fill large blocks of memory. Initially, a WRITE is executed to set up the starting address of the block and supply the first operand. The FILL command writes subsequent operands.
Resume Execution	GO	The pipe is flushed and refilled before resuming instruction execution at the current PC.
Patch User Code	CALL	Current program counter is stacked at the location of the current stack pointer. Instruction execution begins at user patch code.
Reset Peripherals	RST	Asserts RESET for 512 clock cycles. The CPU is not reset by this command. Synonymous with the CPU RESET instruction.
No Operation	NOP	NOP performs no operation and can be used as a null command.

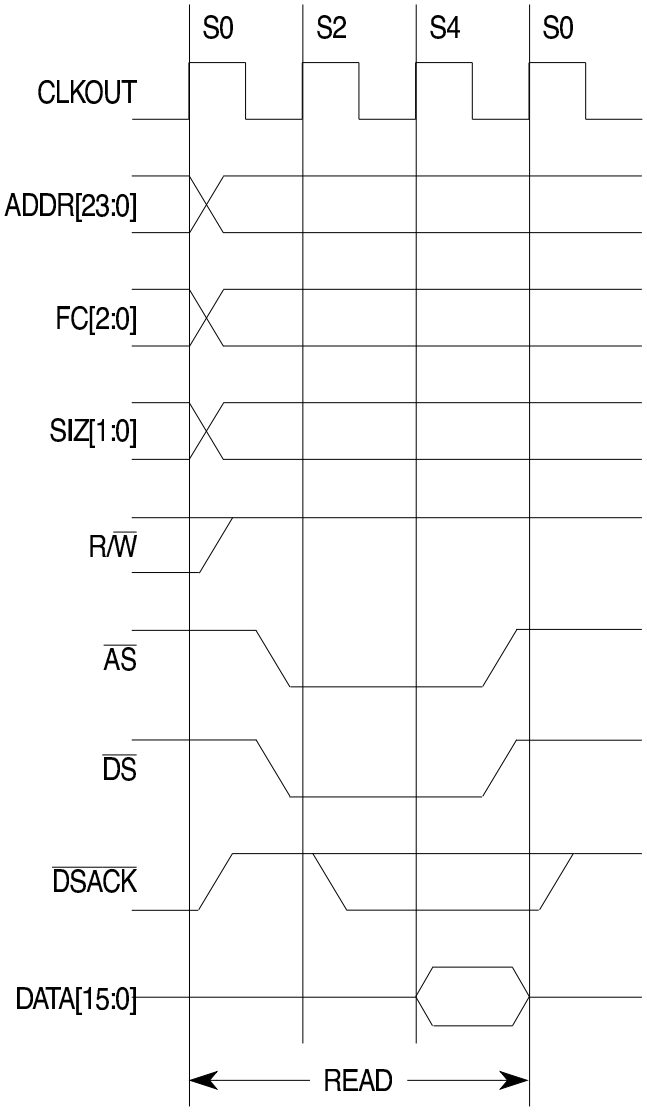
Wektory obsługi zdarzeń specjalnych (*Exception Vectors*)

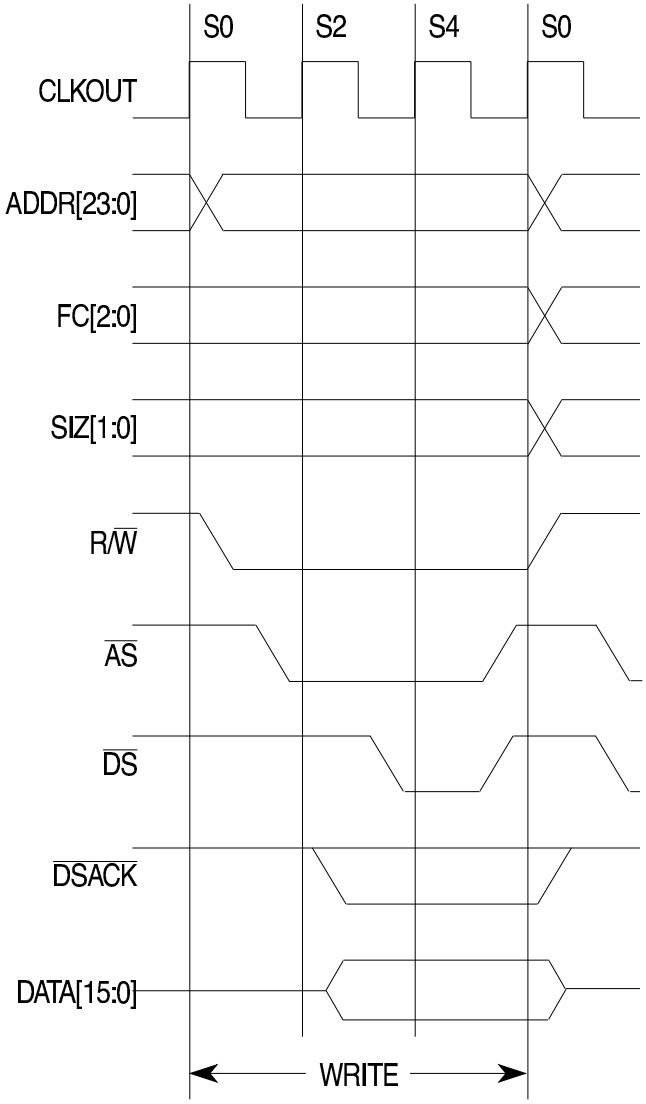
Vector Number	Vector Offset			Assignment
	Dec	Hex	Space	
0	0	000	SP	Reset: Initial Stack Pointer
1	4	004	SP	Reset: Initial Program Counter
2	8	008	SD	Bus Error
3	12	00C	SD	Address Error
4	16	010	SD	Illegal Instruction
5	20	014	SD	Zero Division
6	24	018	SD	CHK, CHK2 Instructions
7	28	01C	SD	TRAPcc, TRAPV Instructions
8	32	020	SD	Privilege Violation
9	36	024	SD	Trace
10	40	028	SD	Line 1010 Emulator
11	44	02C	SD	Line 1111 Emulator
12	48	030	SD	Hardware Breakpoint
13	52	034	SD	(Reserved, Coprocessor Protocol Violation)
14	56	038	SD	Format Error and Uninitialized Interrupt
15	60	03C	SD	Format Error and Uninitialized Interrupt
16–23	64 92	040 05C	SD	(Unassigned, Reserved)
24	96	060	SD	Spurious Interrupt
25	100	064	SD	Level 1 Interrupt Autovector
26	104	068	SD	Level 2 Interrupt Autovector
27	108	06C	SD	Level 3 Interrupt Autovector
28	112	070	SD	Level 4 Interrupt Autovector
29	116	074	SD	Level 5 Interrupt Autovector
30	120	078	SD	Level 6 Interrupt Autovector
31	124	07C	SD	Level 7 Interrupt Autovector
32–47	128 188	080 0BC	SD	Trap Instruction Vectors (0–15)
48–58	192 232	0C0 0E8	SD	(Reserved, Coprocessor)
59–63	236 252	0EC 0FC	SD	(Unassigned, Reserved)
64–255	256 1020	100 3FC	SD	User Defined Vectors (192)

Struktura SIM (*System Integration Module*)

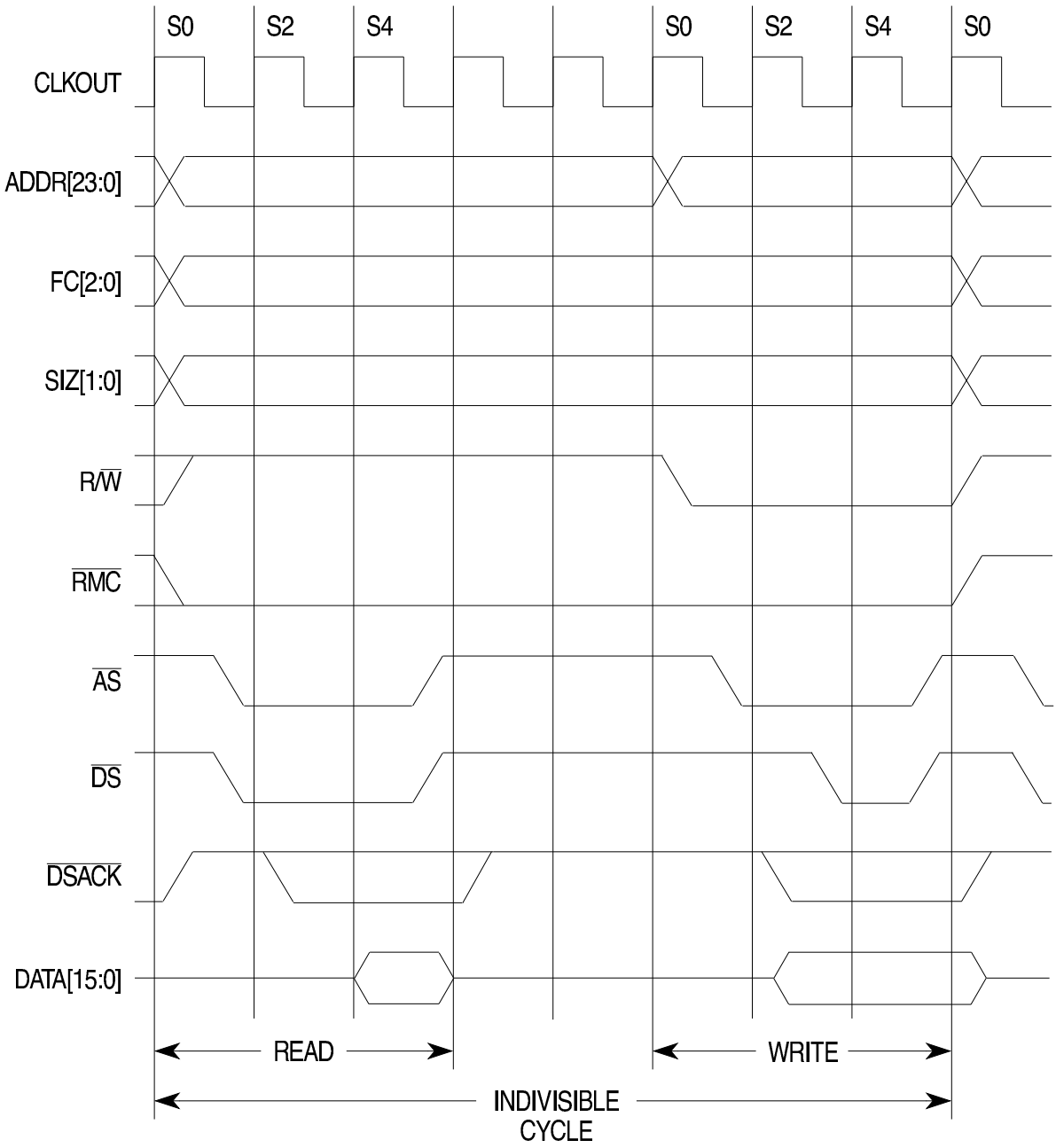


Cykle odczytu i zapisu na magistrali zewnętrznej

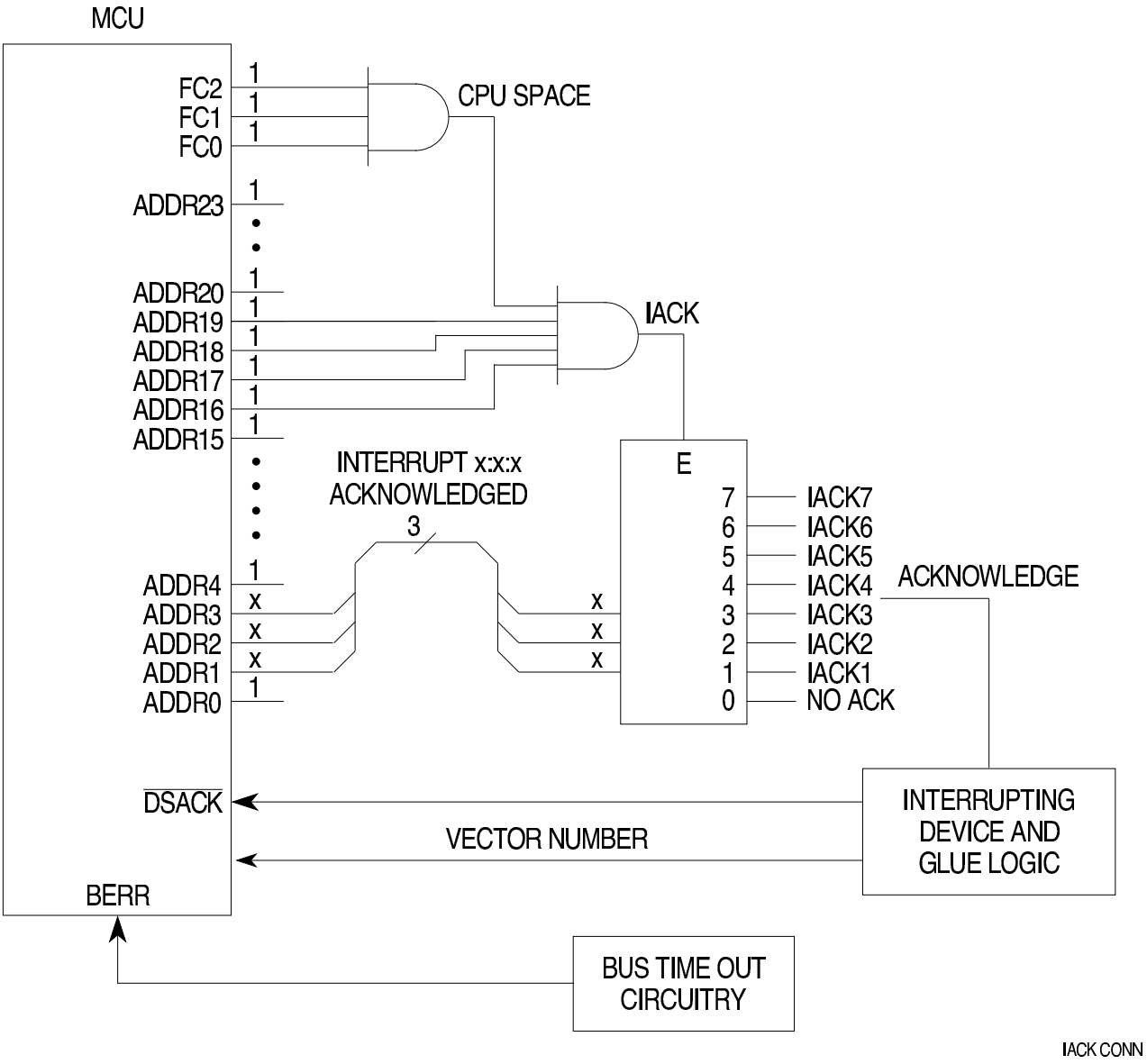




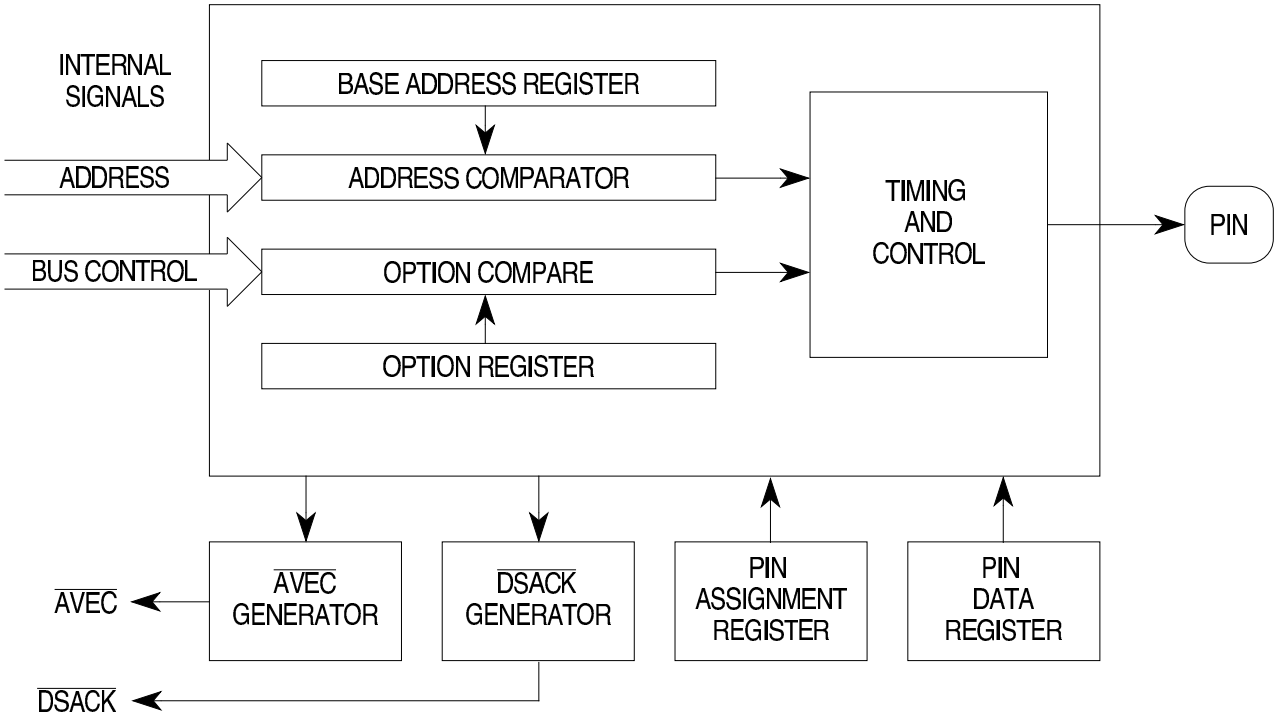
Cykl RMW (*Read–Modify–Write*)



Układ obsługi przerwań



Układ generowania sygnałów wyboru urządzeń



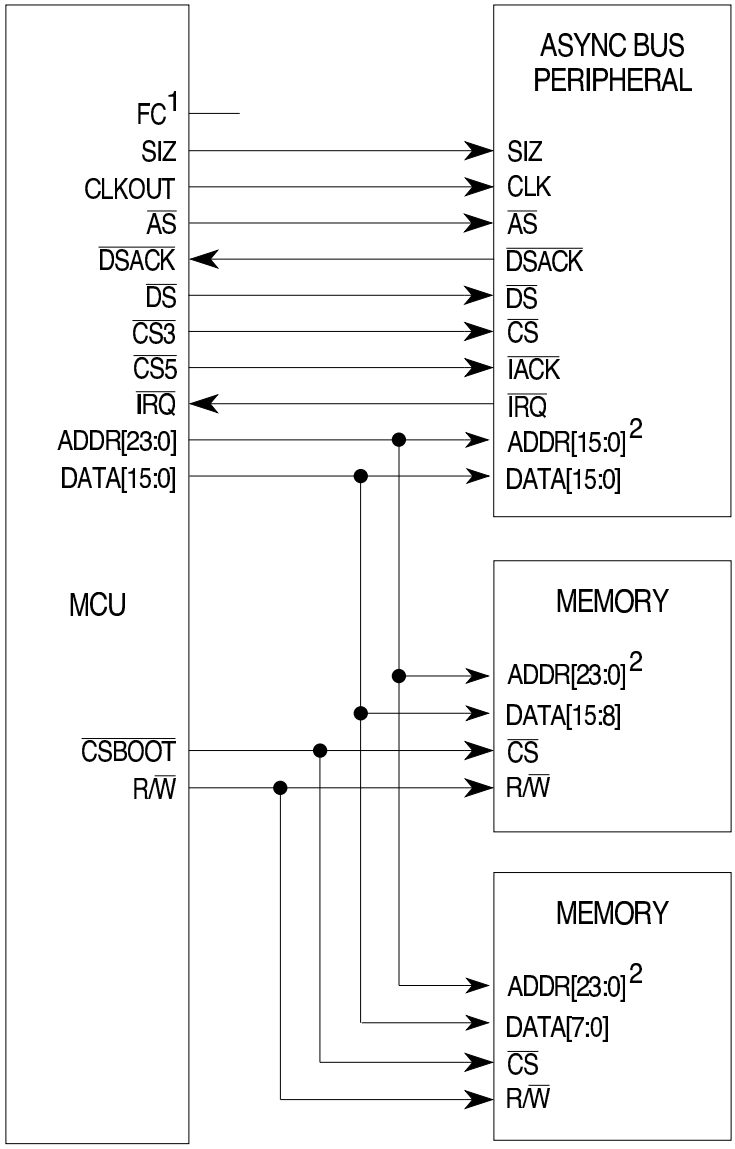
CSBARBT — Chip-Select Base Address Register Boot ROM **$YFFA48**

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

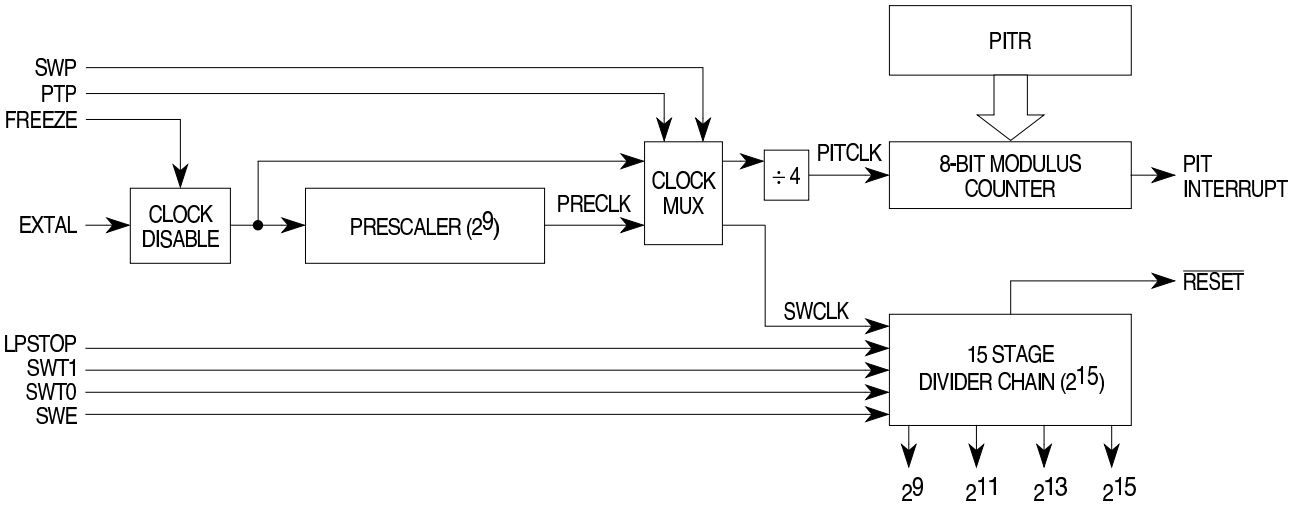


1. Can be decoded to provide additional address space.
2. Varies depending upon peripheral memory size.

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++);
}
```

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**

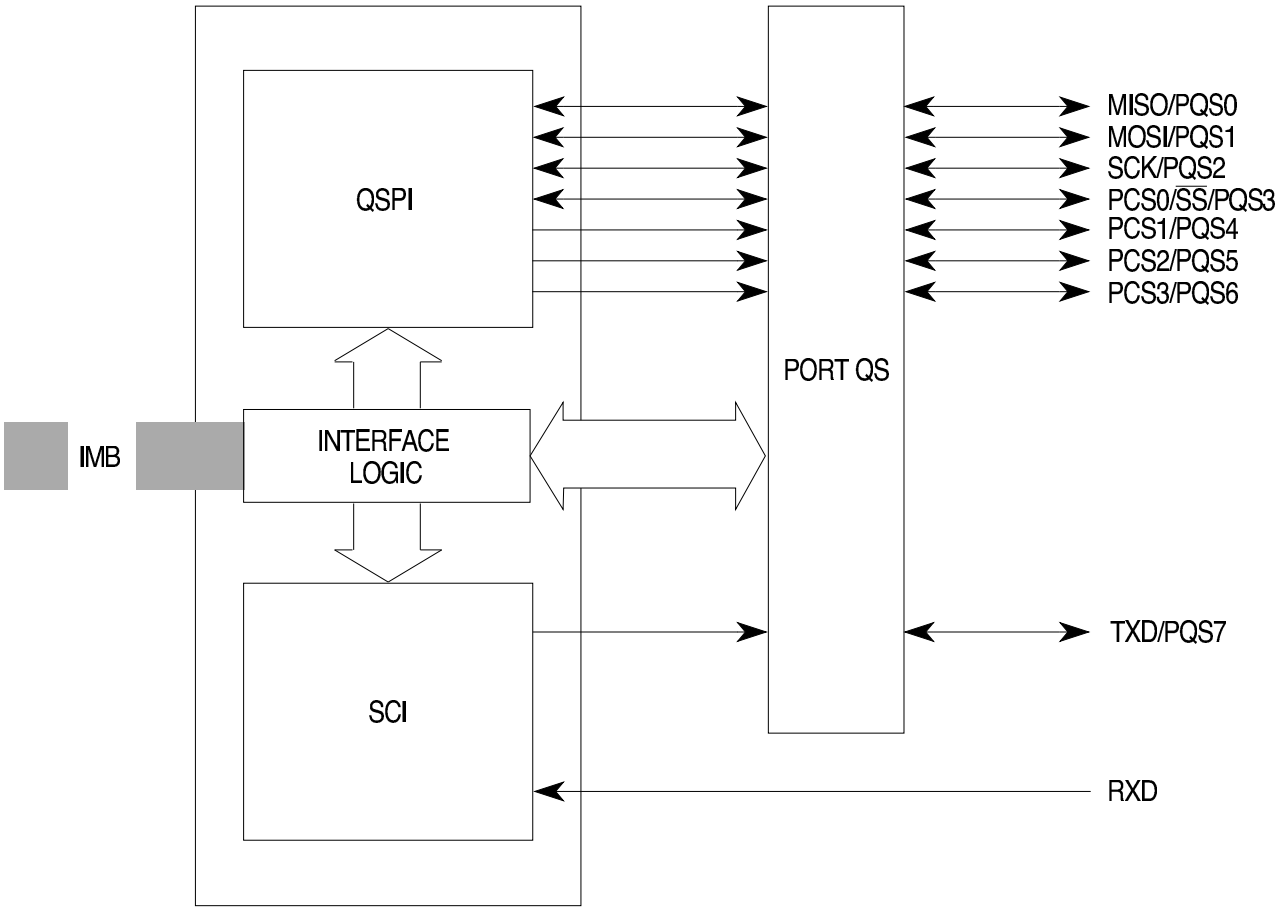
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	PTP	PITM							
RESET:															
0	0	0	0	0	0	0	$\overline{\text{MODCLK}}$	0	0	0	0	0	0	0	0

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

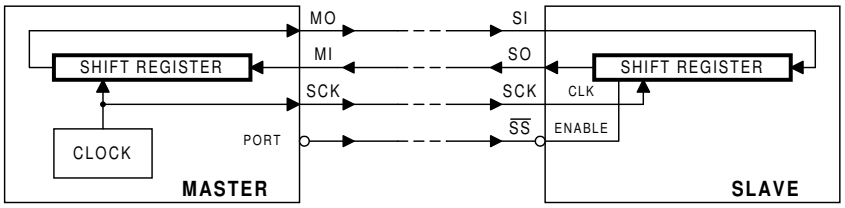
Obsługa programowa PIT

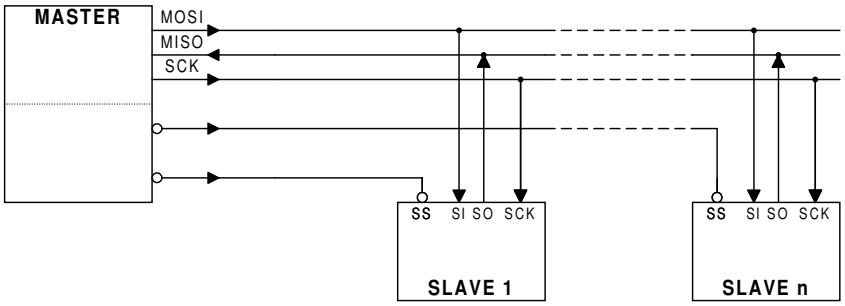
```
#define SIMBASE 0xfffffa00
#define PICR  (*((WORD *) (SIMBASE+0x22)))              /* PIT Control */
#define PITR  (*((volatile WORD *) (SIMBASE+0x24)))     /* PIT Modulus */

#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 */
}
```

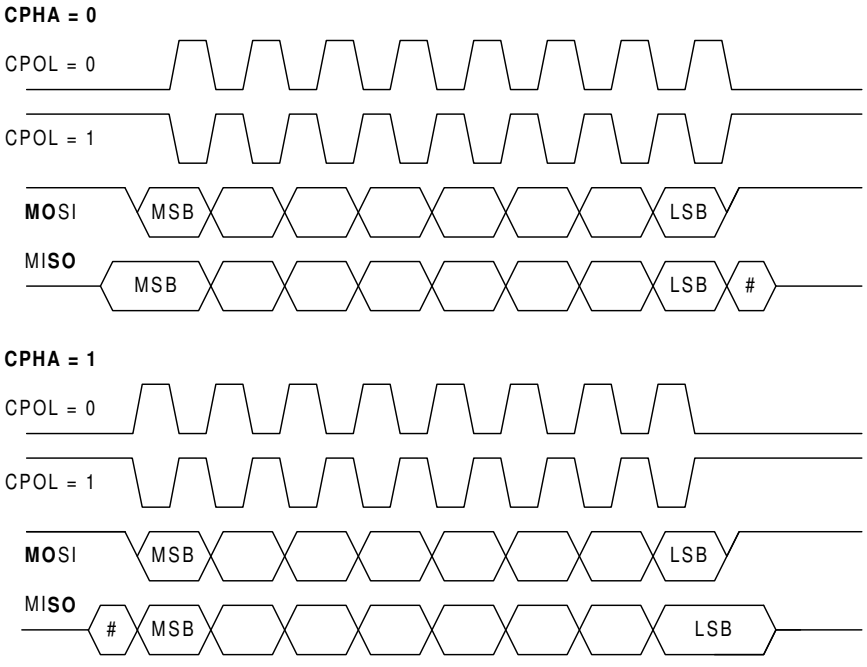
Struktura QSM (*Queued Serial Module*)



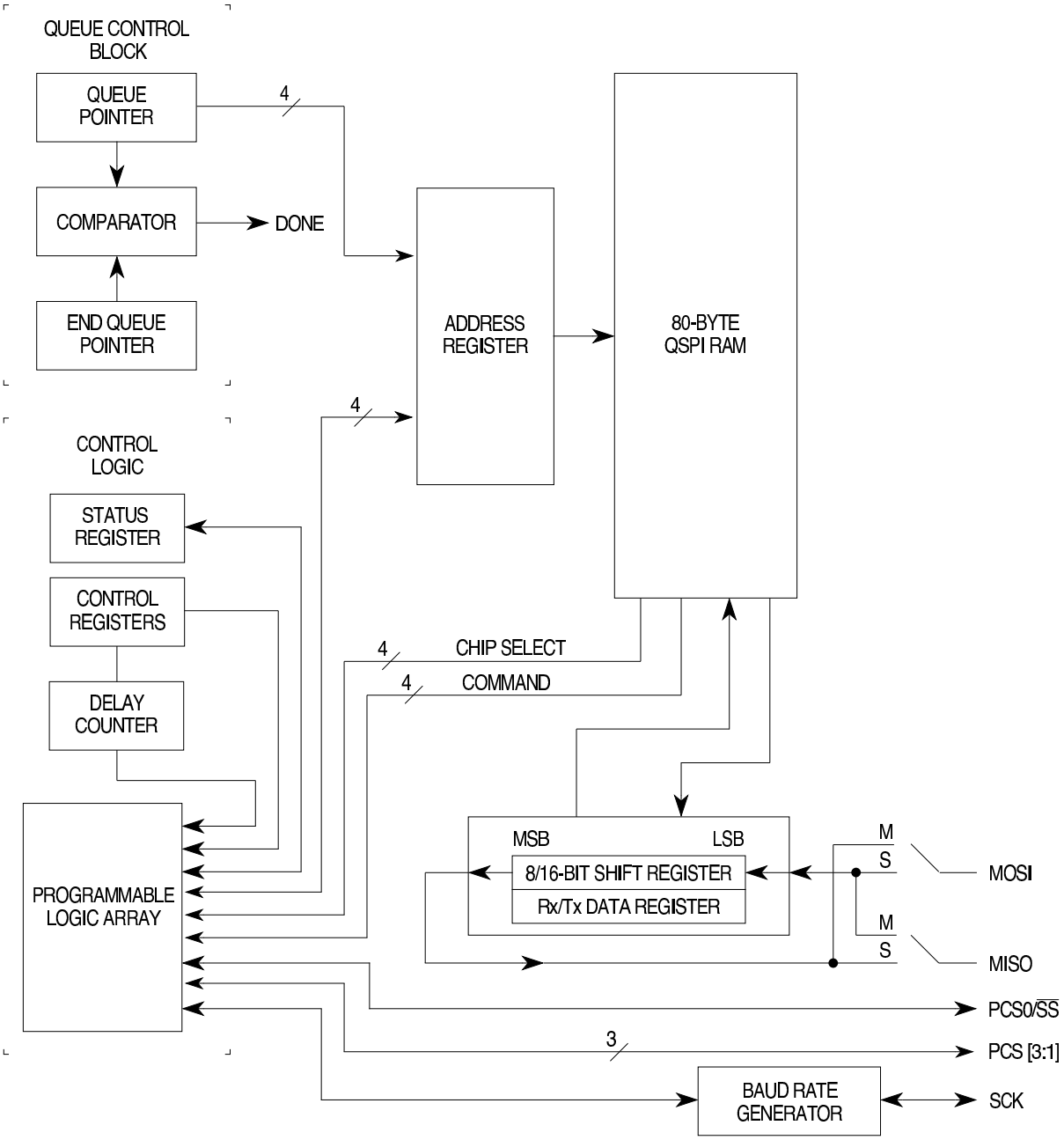
Synchroniczny interfejs urządzeń (SPI)







Blok QSPI (*Queued Serial Peripheral Interface*)



D.4.9 PQSPAR — PORT QS Pin Assignment Register $YFFC16
DDRQS — PORT QS Data Direction Register $YFFC17

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	PQSPA6	PQSPA5	PQSPA4	PQSPA3	0	PQSPA1	PQSPA0	DDQS7	DDQS6	DDQS5	DDQS4	DDQS3	DDQS2	DDQS1	DDQS0
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

D.4.10 SPCR0 — QSPI Control Register 0 $YFFC18

15	14	13	10	9	8	7	0
MSTR	WOMQ	BITS		CPOL	CPHA	SP	
RESET:							
0	0	0	0	0	1	0	0

D.4.11 SPCR1 — QSPI Control Register 1 $YFFC1A

15	14	8	7	0
SPE	DSCKL		DTL	
RESET:				
0	0	0	0	0

D.4.12 SPCR2 — QSPI Control Register 2 $YFFC1C

15	14	13	12	11	8	7	6	5	4	3	0
SPIFIE	WREN	WRTO	0	ENDQP		0	0	0	0	NEWQP	
RESET:											
0	0	0	0	0	0	0	0	0	0	0	0

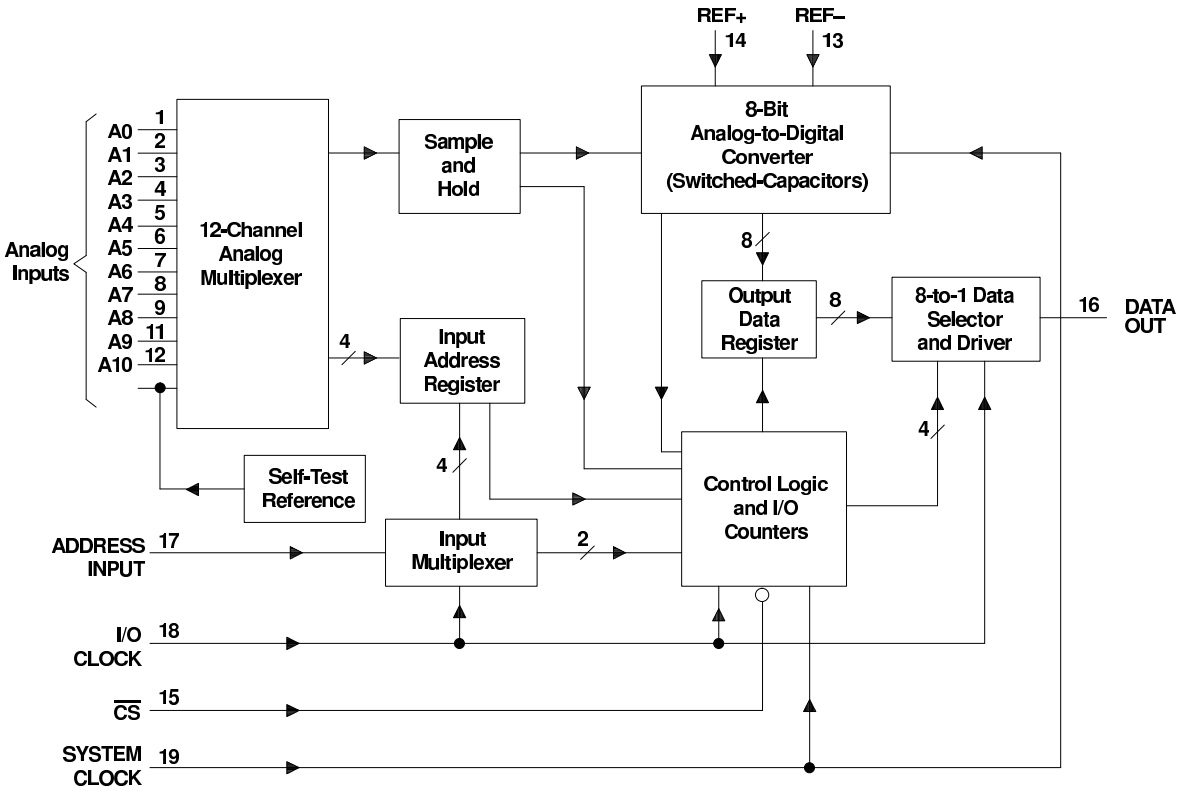
D.4.13 SPCR3 — QSPI Control Register 3 $YFFC1E
SPSR — QSPI Status Register $YFFC1F

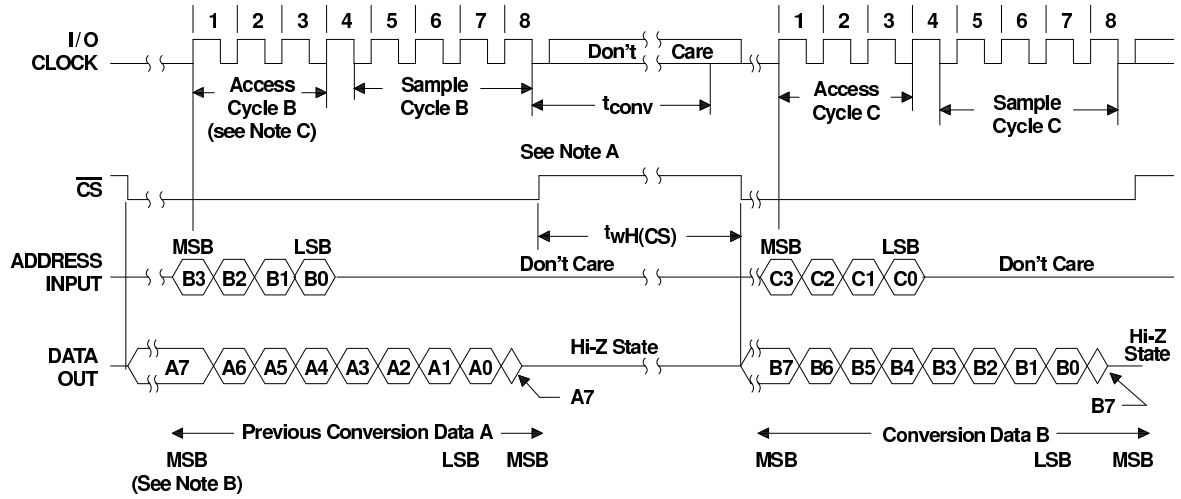
15	14	13	12	11	10	9	8	7	6	5	4	3	0
0	0	0	0	0	LOOPQ	HMIE	HALT	SPIF	MODF	HALTA	0	CPTQP	
RESET:													
0	0	0	0	0	0	0	0	0	0	0	0	0	0

D.4.16 CR[0:F] — Command RAM $YFFD40–$YFFD4F

7	6	5	4	3	2	1	0
CONT	BITSE	DT	DSCK	PCS3	PCS2	PCS1	PCS0*
—	—	—	—	—	—	—	—
CONT	BITSE	DT	DSCK	PCS3	PCS2	PCS1	PCS0*
COMMAND CONTROL				PERIPHERAL CHIP SELECT			

Przetwornik A/C TLC541





NOTES: A. The conversion cycle, which requires 36 system clock periods, is initiated on the 8th falling edge of I/O CLOCK after $\overline{CS}$ goes low for the channel whose address exists in memory at that time. If $\overline{CS}$ is kept low during conversion, I/O CLOCK must remain low for at least 36 system clock cycles to allow conversion to be completed.

B. The most significant bit (MSB) will automatically be placed on the DATA OUT bus after $\overline{CS}$ is brought low. The remaining seven bits (A6–A0) will be clocked out on the first seven I/O CLOCK falling edges.

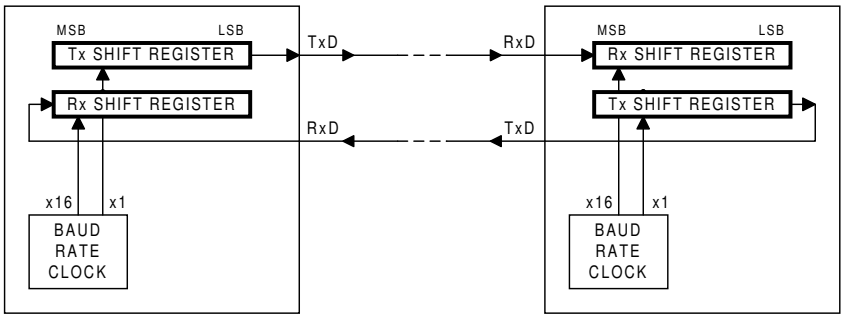
C. To minimize errors caused by noise at $\overline{CS}$, the internal circuitry waits for three system clock cycles (or less) after a chip select falling edge is detected before responding to control input signals. Therefore, no attempt should be made to clock-in address data until the minimum chip-select setup time has elapsed.

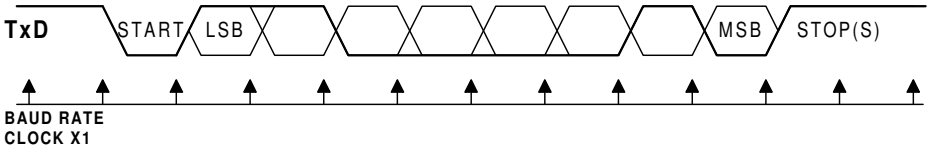
Obsługa programowa QSPI

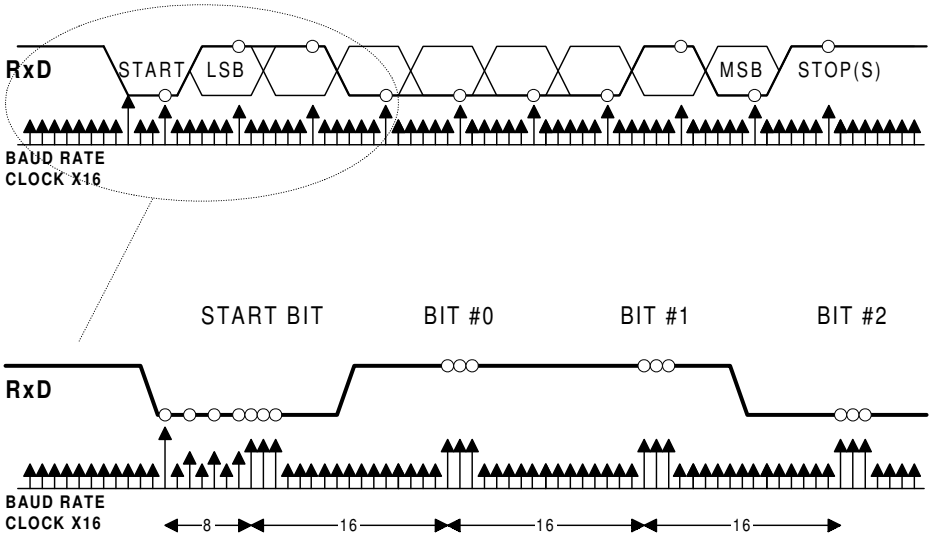
```
#define QSMBASE 0xfffffc00

#define QPDR    (*((volatile BYTE *) (QSMBASE+0x15)))     /* QSM Port Data */
#define QPAR    (*((BYTE *) (QSMBASE+0x16)))         /* QSM Pin Assignment */
#define QDDR    (*((BYTE *) (QSMBASE+0x17)))         /* QSM Data Direction */

#define SPCR0   (*((WORD *) (QSMBASE+0x18)))            /* QSPI Control 0  */

#define SPCR1   (*((WORD *) (QSMBASE+0x1a)))             /* QSPI Control 1 */
#define QsmSPE     0x8000                         /* flaga uruchomienia QSPI */

#define SPCR2   (*((WORD *) (QSMBASE+0x1c)))             /* QSPI Control 2 */

#define SPCR3   (*((BYTE *) (QSMBASE+0x1e)))             /* QSPI Control 3 */

#define SPSR    (*((volatile BYTE *) (QSMBASE+0x1f)))       /* QSPI Status */
#define QsmSPIF    0x80                    /* flaga zakonczenia transmisji */

#define QREC    ( ((volatile WORD *) (QSMBASE+0x100)))  /* QSPI Rx RAM ptr */

#define QTRAN   ( ((WORD *) (QSMBASE+0x120)))           /* QSPI Tx RAM ptr */

#define QCOMD   ( ((BYTE *) (QSMBASE+0x140)))          /* QSPI Cmd RAM ptr */
#define QsmCONT    0x80                              /* nastapi kontynuacja */
#define QsmBITSE   0x40                /* wybor dlugosci slowa wedlug BITS */
#define QsmDT      0x20                  /* wlaczenie opoznienia koncowego */
#define QsmDSCK    0x10               /* wlaczenie opoznienia poczatkowego */
#define QsmSELMASK 0x0f                                 /* maska adresu SLAVE */
```

```
#define ADCSEL      0xc                 /* wybor ADC stanem niskim na PCS1 */
#define DESELECT    0xe                   /* zaden SLAVE nie jest wybrany */

int ReadAdc(WORD kanal)
{
    if(SPCR1 & QsmSPE)                   /* jesli QSPI jest wlaczone, to: */
    {
        while(!(SPSR & QsmSPIF));            /* czekaj na koniec operacji */
        SPSR &= (~QsmSPIF);                     /* zgas flage zakonczenia */
    }

    QDDR = 0xfe;                           /* PCSx, SCK, MOSI - wyjscia */
    QPDR = DESELECT<<3;                       /* wylaczenie wszystkiego */
    QPAR = 0x7b;

    SPCR0 = 0xa817;                     /* master, cpol 0, cpha 0, 493kHz */
    SPCR1 = 0x7fc0;                              /* DSCKL=2us, DTL=22us */
    SPCR2 = 0x0100;                       /* kolejka od 0 do 1 (2 slowa) */

    QTRAN[0] = kanal<<4;                                /* kanal mierzony */
    QTRAN[1] = 11<<4;                                    /* kanal testowy */

    QCOMD[0] = QsmDT | QsmDSCK | ADCSEL;                        /* 8 bitow */
    QCOMD[1] = QsmDT | QsmDSCK | ADCSEL;                        /* 8 bitow */

    SPCR1 |= QsmSPE;                                        /* wlacz QSPI */
    while(!(SPSR & QsmSPIF));                /* czekaj na koniec operacji */
    SPSR &= (~QsmSPIF);                         /* zgas flage zakonczenia */

    return (QREC[1]);                                  /* wynik w QREC[1] */
}
```

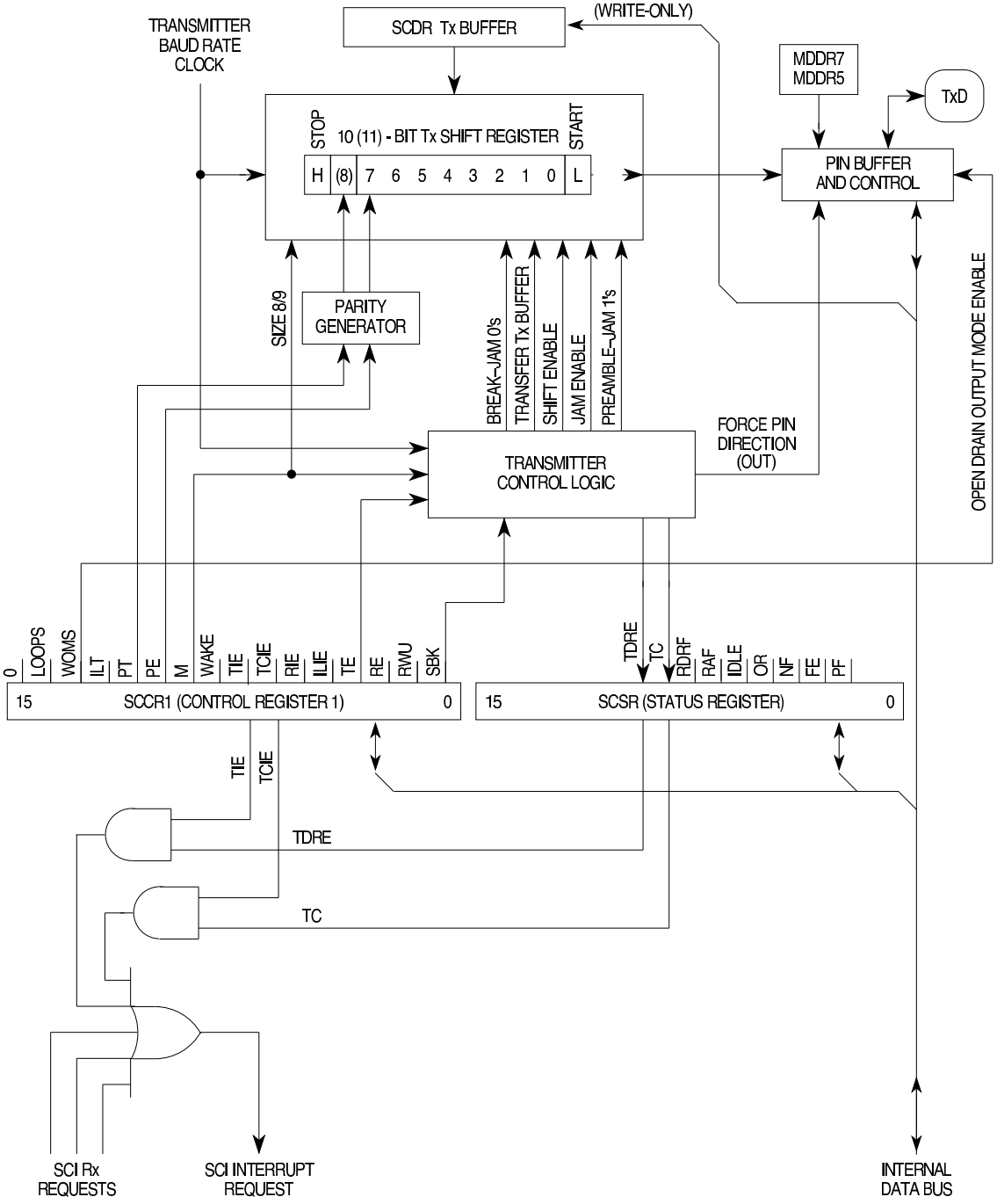
Asynchroniczna transmisja szeregowa







Nadajnik transmisji asynchronicznej (SCI)



Odbiornik transmisji asynchronicznej (SCI)