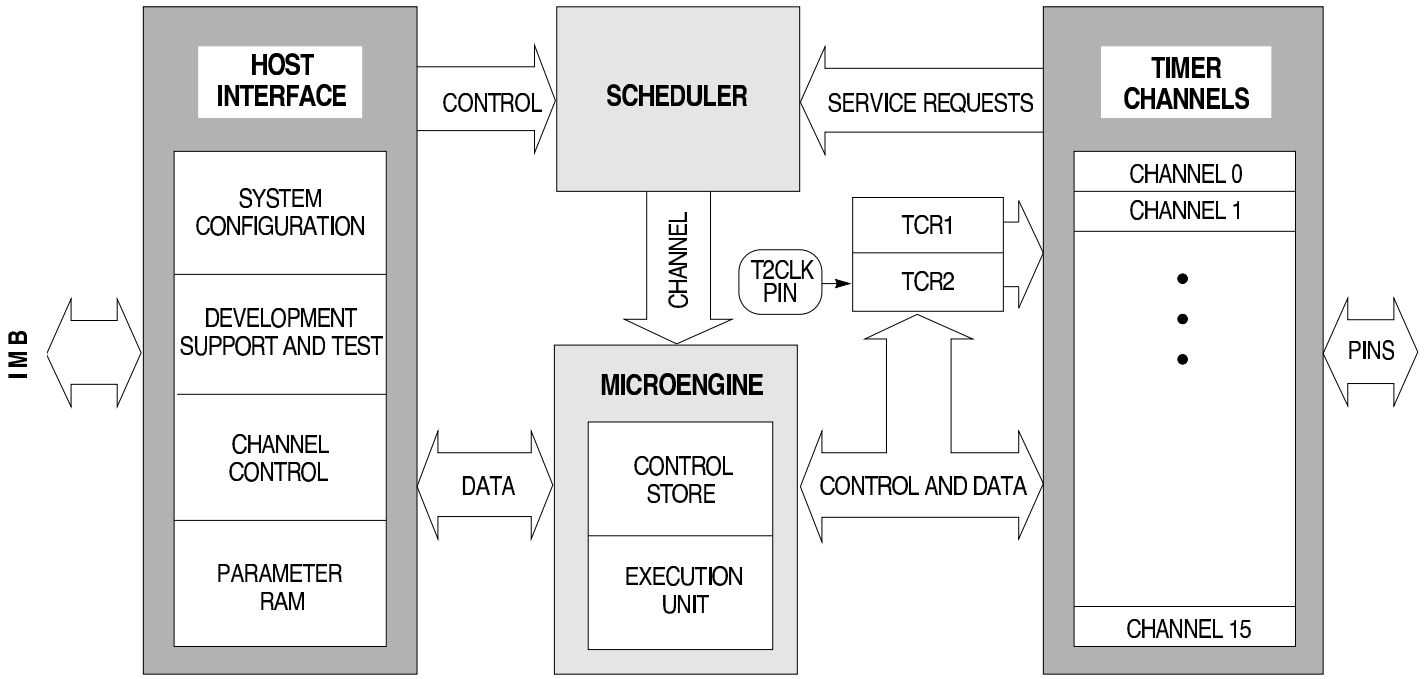
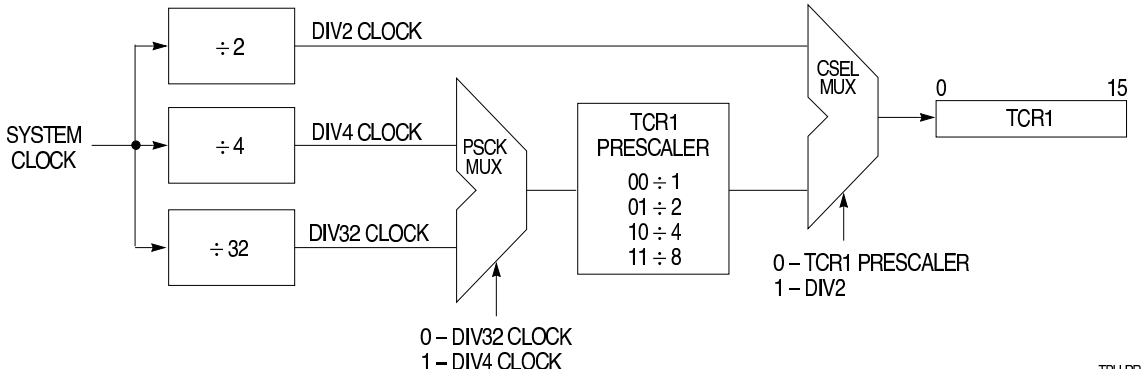
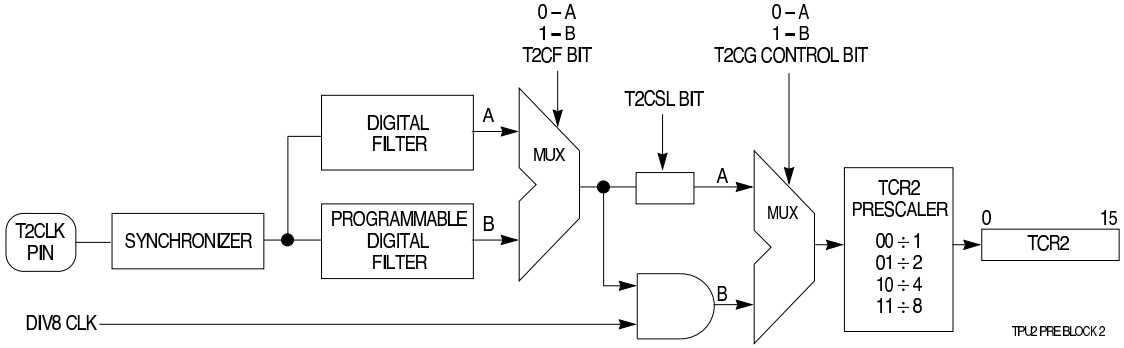
Struktura TPU (*Time Processor Unit*)



TPU BLOCK

Układy generacji podstaw czasu dla TPU





TPUMCR — TPU Module Configuration Register **$###E00**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
STOP	TCR1P[1:0]		TCR2P[1:0]		EMU[1]	T2CG	STF	SUPV	PSCK	TPU2[2]	T2CSL[3]	IARB[3:0]			
RESET:															
0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0

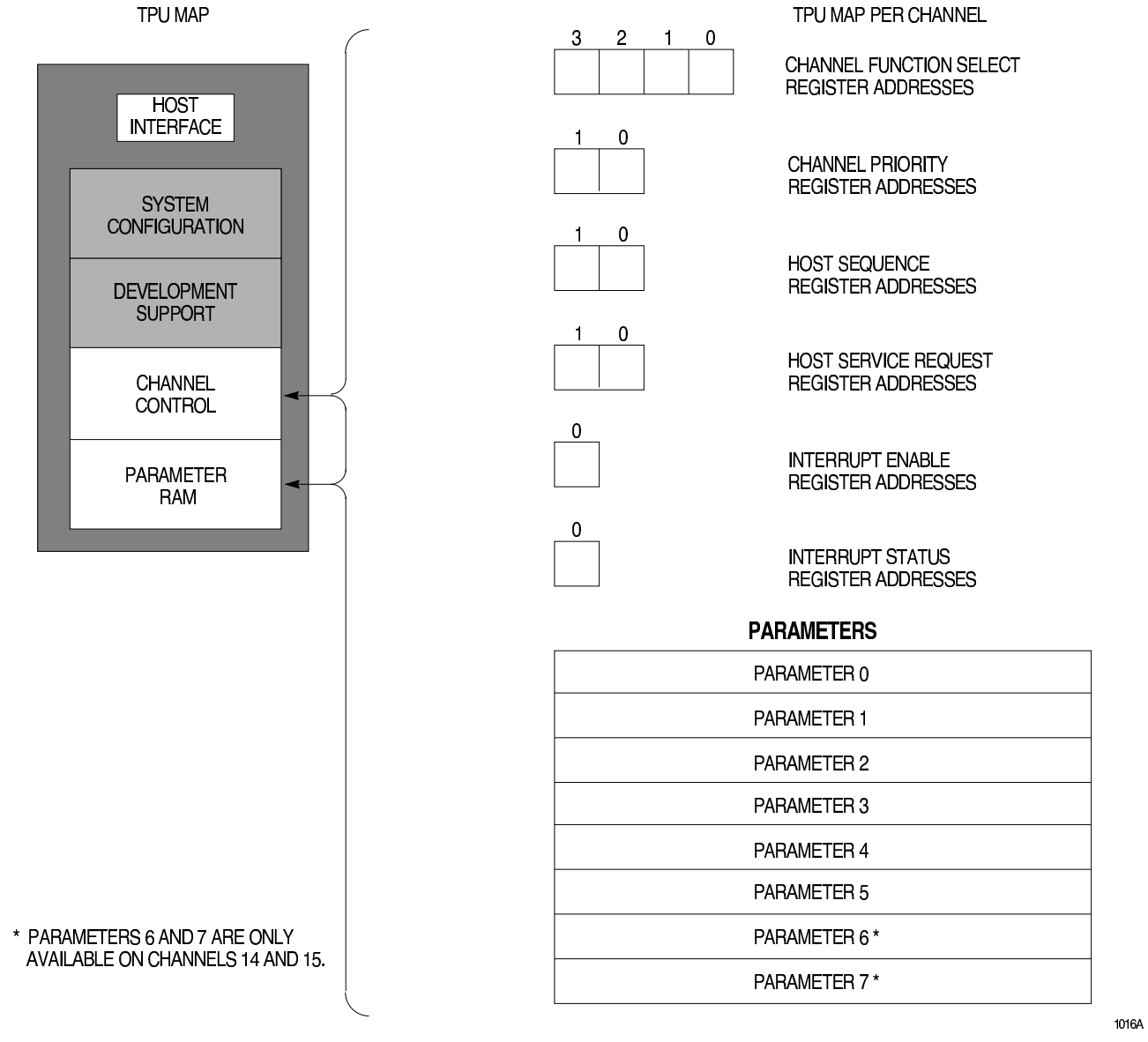
NOTES:

1. On TPU2, this bit is set or cleared according to the shadow bit for bit four of the flash EEPROM module configuration (FEEMCR) register.
2. After reset, the TPU2 enable (TPU2) bit is zero if TPU module is present. In this case, the bit cannot be modified. If the TPU2 module is present, the TPU2 enable bit is one after reset.
3. After reset, TCR2 counter clock edge (T2CSL) bit is zero if the TPU module is present. In this case, the bit cannot be modified. If the TPU2 module is present, this bit is zero after reset and can be modified.

Rejestry bloku TPU (*Time Processor Unit*)

Access	Address	15 8	7 0
S[1]	$###E00	TPU MODULE CONFIGURATION REGISTER (TPUMCR)	
S	$###E02	TPU TEST CONFIGURATION REGISTER (TCR)	
S	$###E04	DEVELOPMENT SUPPORT CONTROL REGISTER (DSCR)	
S	$###E06	DEVELOPMENT SUPPORT STATUS REGISTER (DSSR)	
S	$###E08	TPU INTERRUPT CONFIGURATION REGISTER (TICR)	
S	$###E0A	CHANNEL INTERRUPT ENABLE REGISTER (CIER)	
S	$###E0C	CHANNEL FUNCTION SELECT REGISTER 0 (CFSR0)	
S	$###E0E	CHANNEL FUNCTION SELECT REGISTER 1 (CFSR1)	
S	$###E10	CHANNEL FUNCTION SELECT REGISTER 2 (CFSR2)	
S	$###E12	CHANNEL FUNCTION SELECT REGISTER 3 (CFSR3)	
S/U[2]	$###E14	HOST SEQUENCE REGISTER 0 (HSQR0)	
S/U	$###E16	HOST SEQUENCE REGISTER 1 (HSQR1)	
S/U	$###E18	HOST SERVICE REQUEST REGISTER 0 (HSRR0)	
S/U	$###E1A	HOST SERVICE REQUEST REGISTER 1 (HSRR1)	
S	$###E1C	CHANNEL PRIORITY REGISTER 0 (CPR0)	
S	$###E1E	CHANNEL PRIORITY REGISTER 1 (CPR1)	
S	$###E20	CHANNEL INTERRUPT STATUS REGISTER (CISR)	
S	$###E22	LINK REGISTER (LR)	
S	$###E24	SERVICE GRANT LATCH REGISTER (SGLR)	
S	$###E26	DECODED CHANNEL NUMBER REGISTER (DCNR)	
S	$###E28	TPU2 MODULE CONFIGURATION REGISTER 2 (TPUMCR2) — TPU2 ONLY	
S/U	$###F00 – $###F0E	CHANNEL 0 PARAMETER REGISTERS	
S/U	$###F10 – $###F1E	CHANNEL 1 PARAMETER REGISTERS	
S/U	$###F20 – $###F2E	CHANNEL 2 PARAMETER REGISTERS	
S/U	$###F30 – $###F3E	CHANNEL 3 PARAMETER REGISTERS	
S/U	$###F40 – $###F4E	CHANNEL 4 PARAMETER REGISTERS	
S/U	$###F50 – $###F5E	CHANNEL 5 PARAMETER REGISTERS	
S/U	$###F60 – $###F6E	CHANNEL 6 PARAMETER REGISTERS	
S/U	$###F70 – $###F7E	CHANNEL 7 PARAMETER REGISTERS	
S/U	$###F80 – $###F8E	CHANNEL 8 PARAMETER REGISTERS	
S/U	$###F90 – $###F9E	CHANNEL 9 PARAMETER REGISTERS	
S/U	$###FA0 – $###FAE	CHANNEL 10 PARAMETER REGISTERS	
S/U	$###FB0 – $###FBE	CHANNEL 11 PARAMETER REGISTERS	
S/U	$###FC0 – $###FCE	CHANNEL 12 PARAMETER REGISTERS	
S/U	$###FD0 – $###FDE	CHANNEL 13 PARAMETER REGISTERS	
S/U	$###FE0 – $###FEE	CHANNEL 14 PARAMETER REGISTERS	
S/U	$###FF0 – $###FFE	CHANNEL 15 PARAMETER REGISTERS	

Sterowanie kanałem TPU

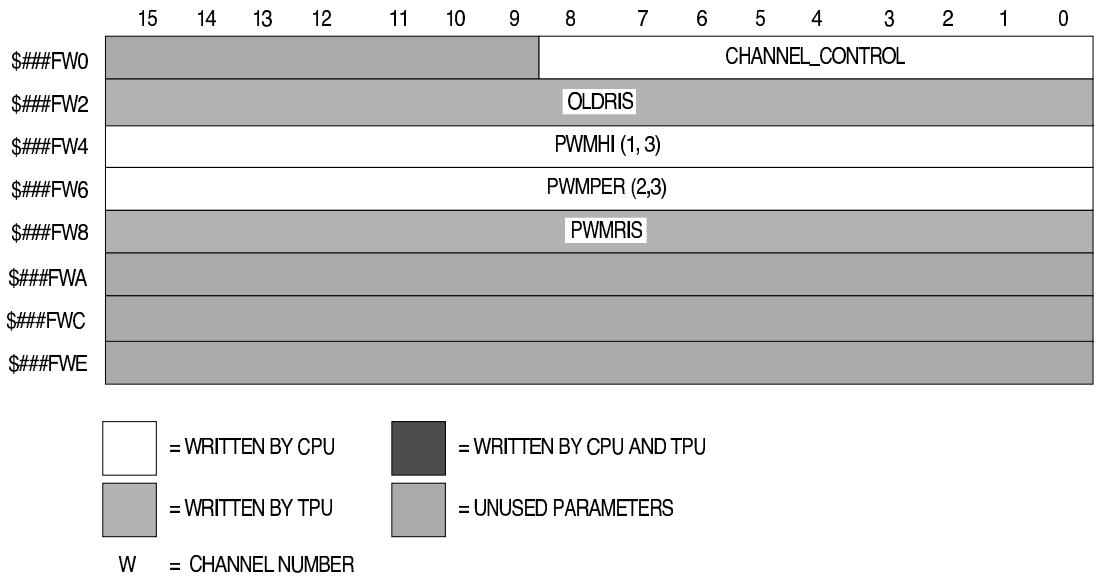


Funkcja PWM (*Pulse Width Modulation*)

CONTROL BITS

	NAME	OPTIONS	ADDRESSES
3 2 1 0	CHANNEL FUNCTION SELECT	PWM FUNCTION NUMBER (ASSIGNED DURING MICRO-CODE ASSEMBLY)	$###E0C–$###E12
1 0	CHANNEL PRIORITY	00 — DISABLE 01 — LOW PRIORITY 10 — MEDIUM PRIORITY 11 — HIGH PRIORITY	$###E1C–$###E1E
1 0	HOST SEQUENCE BITS	xx — NOT USED	$###E14–$###E16
1 0	HOST SERVICE BITS	00 — NOT USED 01 — IMMEDIATE UPDATE OF PWM 10 — INITIALIZE 11 — NOT USED	$###E18–$###E1A
0	INTERRUPT ENABLE	0 — INTERRUPT NOT ASSERTED 1 — INTERRUPT ASSERTED	$###E0A
0	INTERRUPT STATUS		$###E20

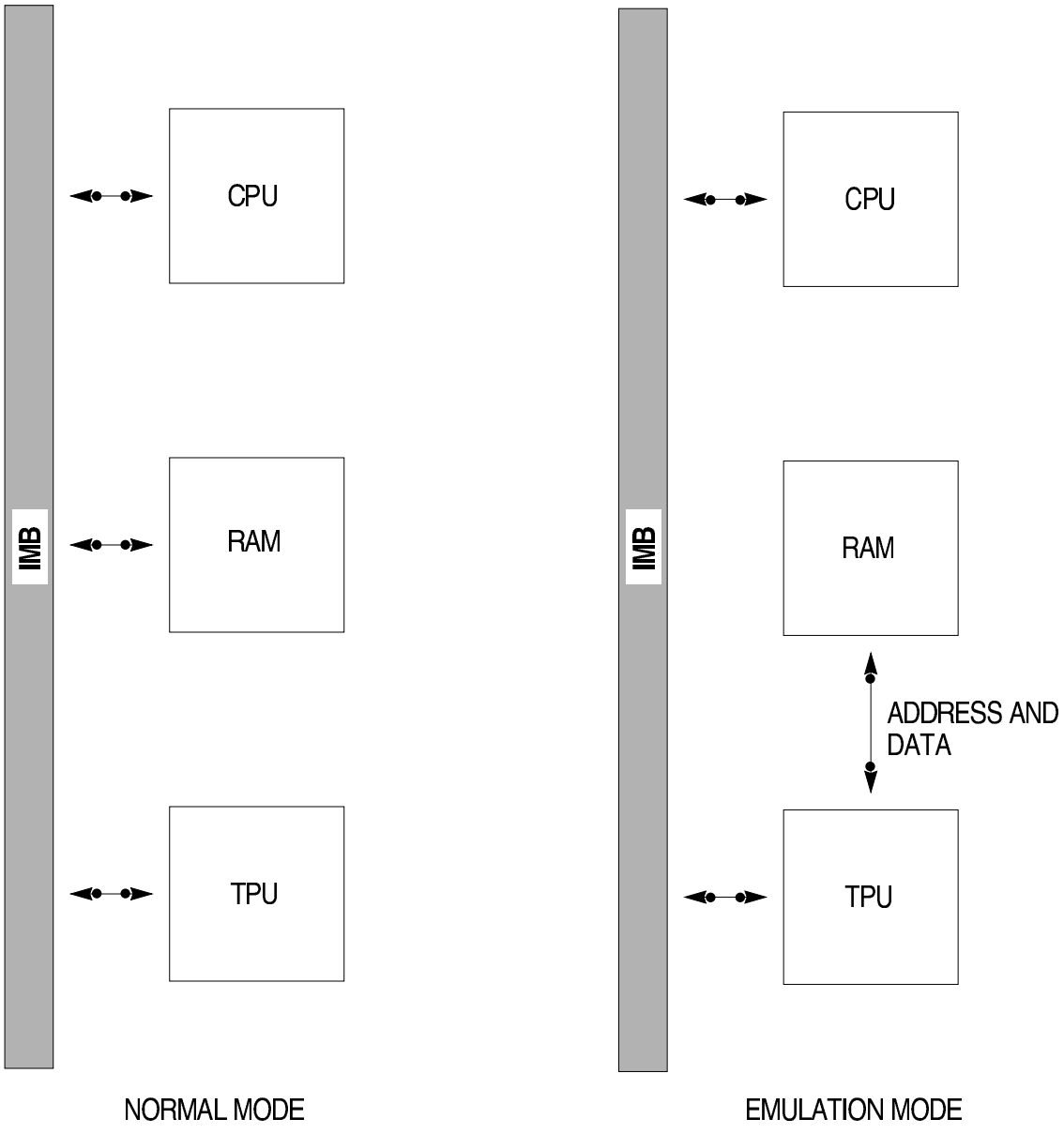
PARAMETER RAM



NOTES:
1. BEST-CASE MINIMUM FOR PWMHI IS 32 SYSTEM CLOCK CYCLES.
2. BEST-CASE MINIMUM FOR PWMPER IS 48 SYSTEM CLOCK CYCLES.
3. PWMHI AND PWMPER MUST BE ACCESSED COHERENTLY.

1027A

Tryb pracy TPU z emulacją mikrokodu w TPURAM



Programowanie TPU (funkcja PWM)

```
/* Definicje rejestrow TPU dla 68332 */
#define TPUBASE 0xfffffe00

#define TPUMCR  (*((WORD *) (TPUBASE+0x00)))      /* TPU Configuration */
#define TpuTCR1P   0x6000                      /* TCR1 prescaler field */
#define TpuEMU     0x0400                         /* emulation flag */
#define TpuIARB    0x000f       /* internal arbitration priority field */

#define CFSR (((WORD *) (TPUBASE+0x0c)))    /* Channel Function Select */

#define HSRR (((volatile WORD *) (TPUBASE+0x18)))  /* Host Svc Request */

#define CPR  (((WORD *) (TPUBASE+0x1c)))           /* Channel Priority */

#define SetFun(chan,code) (CFSR[3-((chan)/4)] = (CFSR[3-((chan)/4)] & \
                      ~(0xf<<(4*((chan)%4)))) | ((code)<<(4*((chan)%4))))
#define SetSer(chan,code) (HSRR[1-((chan)/8)] = ((code)<<(2*((chan)%8))))
#define SetCpr(chan,code) ( CPR[1-((chan)/8)] = ( CPR[1-((chan)/8)] & \
                    ~(0x3<<(2*((chan)%8)))) | ((code)<<(2*((chan)%8))))

#define PARBASE 0xffffff00
#define PARRAM(chan,par) (*(((volatile WORD *)PARBASE)+8*(chan)+(par)))

#define TRAMBAR (*((WORD *)0xfffb04))   /* TPU Emulation Ram Base Reg. */

#define RBASE   0x800000         /* wybrany adres bazowy Emulation RAM */

#define PWMCHAN 0                          /* wybrany kanal TPU dla PWM */
#define PWMFUN  9                          /* kod funkcji PWM w masce A */

extern BYTE tpumska;                    /* obraz TPURAM dla maski A TPU */
```

```
main()
{
    int i;
    BYTE * tmaptr, * ramptr;

        /* Emulacja maski A TPU w RAM */

        TRAMBAR = RBASE>>8;
    tmaptr = (BYTE *)(& tpumska);
        ramptr = (BYTE *)RBASE;
        for(i=0;i<2048;i++) ramptr[i] = *tmaptr++;
    ramptr = tmaptr;

    /* Inicjalizacja i uruchomienie PWM w kanale 0 TPU */

    TPUMCR = 0x200e | TpuEMU;      /* emulacja TPU (mikrokod z TPURAM) */
    CPR[0] = 0;
    CPR[1] = 0;                      /* zatrzymanie wszystkich kanalow */
                                                    /*  TBS  PAC  PSC */
    PARRAM(PWMCHAN,0) = 0x92;     /* sterowanie kanalu: 0100  100  10  */
                                                    /* tcr1 don't low */
    PARRAM(PWMCHAN,2) = 127;                          /* czas impulsu */
    PARRAM(PWMCHAN,3) = 256;                         /* okres impulsu */

    SetFun(PWMCHAN,PWMFUN);                          /* kod funkcji: PWM */
    SetSer(PWMCHAN,2);                          /* zadanie: inicjalizacja */
    SetCpr(PWMCHAN,3);                               /* priorytet: wysoki */

    while(1);
}
```