Dołączanie urządzeń do komputera - karta interfejsowa PCM-3718

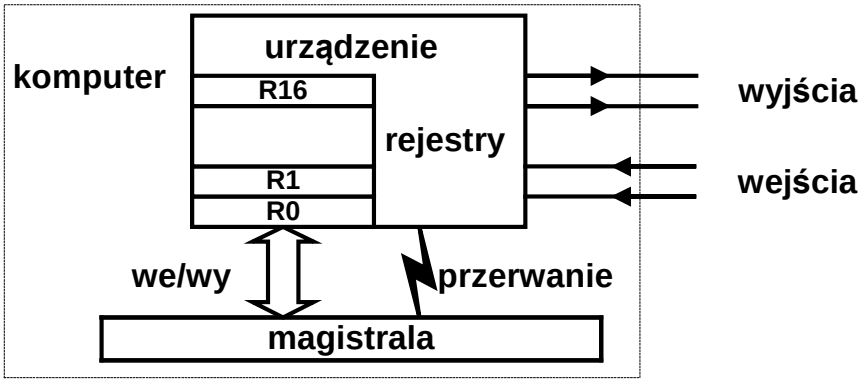
1. Dołączanie urządzeń do komputera

Istnieją dwa podstawowe sposoby podłączenia urządzeń zewnętrznych do komputera.

- Bezpośrednio do magistrali komputera
- Poprzez interfejsy komunikacyjne (RS232, RS485, I2C, SPI, USB, Ethernet).

Gdy urządzenie dołączone jest bezpośrednio do magistrali, widziane jest w przestrzeni adresowej komputera jako:

- Porty wejścia wyjścia
- Obszar pamięci.
- Urządzenie może generować przerwania.



Rysunek 1 Komputer komunikuje się z otoczeniem za pośrednictwem urządzenia podłączonego do magistrali

2. Dostęp do portów we/wy:

Urządzenia wejścia wyjścia posiadają zestawy rejestrów które program może odczytywać i zapisywać. W rejestrach umieszczane są dane konfigurujące urządzenie, statusy, dane przeznaczone do wysłania i odbioru itd.

☞ Próba wykonania w trybie użytkownika operacji wejścia wyjścia lub innej niebezpiecznej operacji spowoduje wygenerowanie wyjątku, wywołanie systemu operacyjnego i w konsekwencji zakończenie procesu.

Procesy które wykonują operacje wejścia wyjścia, muszą należeć do użytkownika **root** (UID=0). Dodatkowo proces powinien wykonać wywołanie funkcji **ThreadCtl(_NTO_TCTL_IO,0).**

W pewnych architekturach urządzenia wejścia wyjścia mogą znajdować się w przestrzeni pamięci wprowadza się ich odwzorowanie na przestrzeń wejścia wyjścia poprzez wykonanie funkcji

Uzyskanie dostępu do rejestrów urządzenia	
int mmap_device_io(int len, uint64_t io)	
len	Liczba bajtów urządzenia która ma być udostępniona
io	Adres początkowy udostępnianego obszaru

Funkcja zwraca wartość będącą argumentem dla funkcji **in*()** i **out*()** czytających i piszących dane do rejestrów lub stałą **MAP_DEVICE_FAILED** gdy nie można uzyskać dostępu do urządzenia.

Odczyt bajtu z portu zewnętrznego	
unsigned char in8(uintptr_t port)	
port	Adres portu w przestrzeni wejścia/wyjścia

Wysłanie bajtu na port zewnętrzny	
void out8(int port, unsigned char val)	
port	Adres portu w przestrzeni wejścia wyjścia.
val	Bajt wyprowadzany na port.

```
#include <hw.inout.h>
...
uintptr_t port;
unsigned char x,y;
ThreadCtl( _NTO_TCTL_IO, 0 );
port = mmap_device_io(1, 0x300);
x = in8(port);
out8(port,y);
```

Przykład 2-1 Odczyt i zapis bajtu z portu urządzenia

3. Opis karty interfejsowej PCM 3718

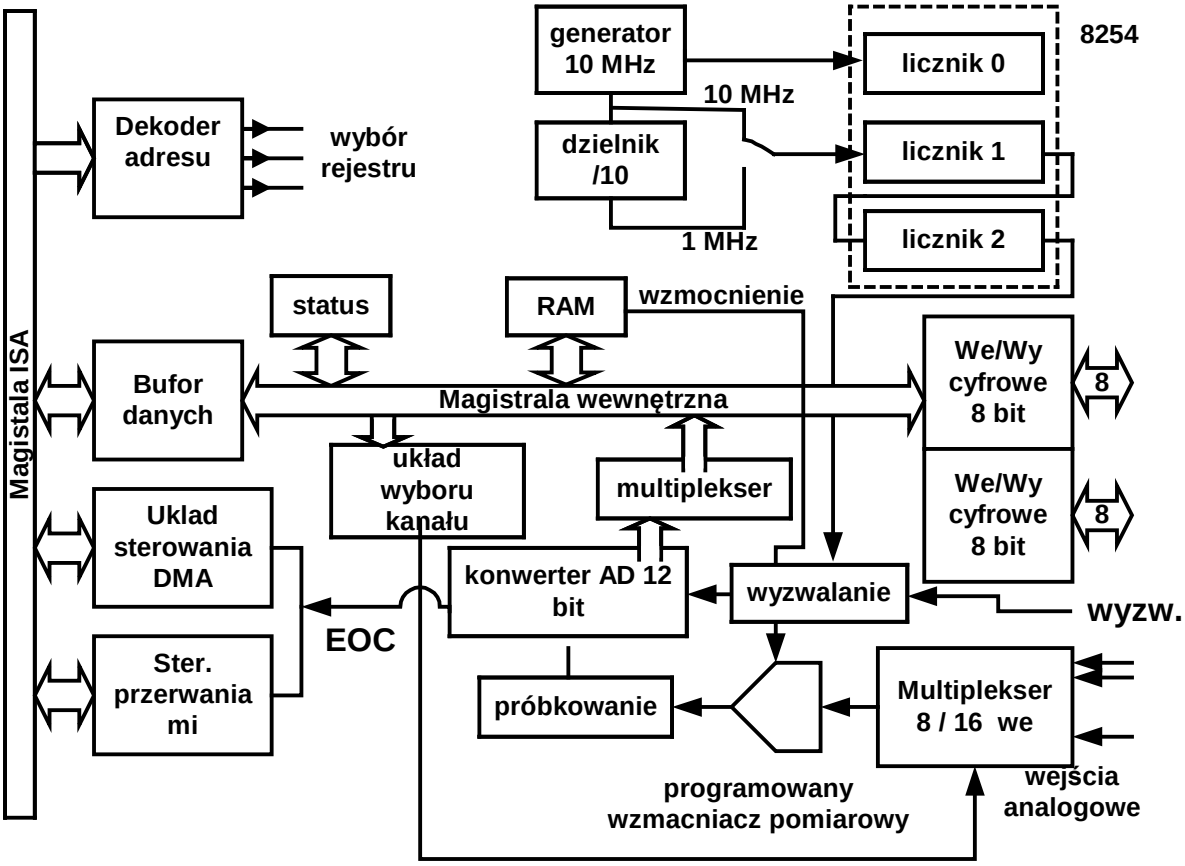
Karta PCM-3718 firmy Advantech Co. Ltd jest to typową kartą przetwornikową stosowana w celach akwizycji danych.

- W systemach przemysłowych,
- Laboratoriach
- Automatyzacji stanowisk badawczo-pomiarowych.

Karta zawiera:

- 16 pojedynczych lub 8 różnicowych wejść analogowych AD dołączonych do multipleksera i dalej poprzez wzmacniacz pomiarowy do 12 bitowego przetwornika analogowo cyfrowego. Maksymalna częstotliwość próbkowania wynosi 60 KHz. Zakres mierzonych napięć to +/- 0.005V, +/- 0.01 V, +/-0.5V, +/-1V, +/-5V, +/-10V dla wejść różnicowych i 0.01V, 0.1V, 1V, 10V dla wejść unipolarnych.
- 16 wejść/wyjść cyfrowych DI poziomu TTL (0V, 5V).
- Układ licznika timera typu 8254 zawierający trzy 16 bitowe liczniki dołączone do generatora 1MHz / 10 MHz. Jeden licznik może być podłączony do źródła zewnętrznego a dwa wykorzystywane są do wyzwalania przetwornika AD.

Z punktu widzenia programisty karta widziana jest jako zestaw 16 rejestrów 8 bitowych poczynając od adresu bazowego **BASE.**



Rysunek 2 Schemat blokowy karty interfejsowej PCM-3718

Adres portu	Odczyt	Zapis
BASE+0	A/D bajt młodszy & kanał	Wyzwalanie programowe A/D
BASE+1	A/D bajt starszy	N/A
BASE+2	Zakres kanałów multipleksera	Zakres kanałów multipleksera
BASE+3	D/I bajt młodszy (DI0-7)	D/O bajt młodszy (DO0-7)
BASE+4	N/A	N/A
BASE+5	N/A	N/A
BASE+6	N/A	N/A
BASE+7	N/A	N/A
BASE+8	Status	Kasowanie źródła przerwania
BASE+9	Rejestr sterujący - zapis	Rejestr sterujący - odczyt
BASE+10	N/A	Konfiguracja liczników
BASE+11	D/I bajt starszy (DI8-15)	D/O bajt starszy (DO8-15)
BASE+12	Licznik 0	Licznik 0
BASE+13	Licznik 1	Licznik 1
BASE+14	Licznik 2	Licznik 2
BASE+15	N/A	Sterowanie licznikami

Tabela 1 Rejestry karty PCM-3718

```
#include <sys/neutrino.h>
#include <hw/inout.h>
#define A0       0  // Przetw. AD bajt lo & kanał
#define A1       1  // Przetw. AD bajt starszy
#define MUXR     2  // Rejestrt multipleksera
#define DOUT1    3  // Wyjścia cyfrowe DO 1
#define DINP1    3  // Wejścia cyfrowe DI 1
#define CONTR    9  // Rejestr ster. przetwornika AD
#define STATR    8  // Rejestr stat. przetwornika AD
#define TIMR    10  // Konfiguracja liczników
#define DINP2   11  // Wejścia cyfrowe DO 2
#define DOUT2   11  // Wyjścia cyfrowe DO 2
#define COUNT0  12  // Sterowanie licznikiem 0
#define COUNT1  13  // Sterowanie licznikiem 1
#define COUNT2  14  // Sterowanie licznikiem 2
#define COUNTC  15  // Sterowanie liczn. układu 8254
```

Przykład 3-1 Plik nagłówkowy pcm3718.h karty PCM-3718

4. Obsługa w trybie odpytywania

4.1 Wejścia i wyjścia cyfrowe

Karta PCM-3718 oferuje:

- Dwa 8 bitowe cyfrowe kanały wejściowe lub wyjściowe

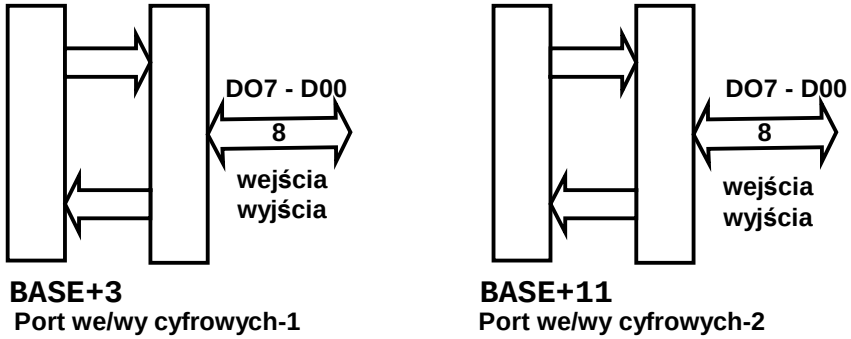
Kanały te używają portów o adresach **BASE+3** i **BASE+11**, które służą zarówno do czytania jak i zapisywania.

Zapis na port **BASE+3** lub **BASE+11** bajtu **DO** powoduje ustawienie linii wyjściowych portu zgodnie z zawartością bitową bajtu **DO**.

Odczyt z portu **BASE+3** lub **BASE+11** powoduje zwrócenie wartości **DI** odpowiadających poziomom logicznym na przyłączeniach portu.

	B7	**B6**	**B5**	**B4**	**B3**	**B2**	**B1**	**B0**
Odczyt	**DI7**	**DI6**	**DI5**	**DI4**	**DI3**	**DI2**	**DI1**	**DI0**
Zapis	**DO7**	**DO6**	**DO5**	**DO4**	**DO3**	**DO2**	**DO1**	**DO0**

Tabela 2 Działanie portów wejść i wyjść cyfrowych DI i DO o adresach BASE+3 i BASE+11




```
 unsigned char dinp(int num) {
// Odczyt portu wejsc cyfrowych, num - port (1,2)
  if(num <= 1)
    return( in8(base + DINP1));
   else
    return( in8(base + DINP2));
}
```

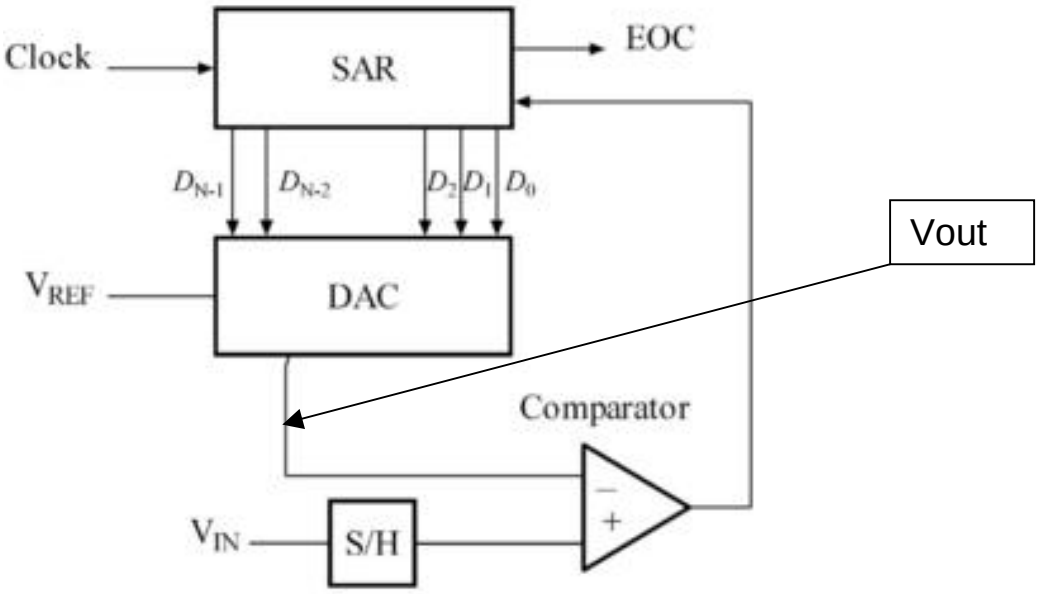
Przykład 1 Odczyt portu wejść cyfrowych num - port (1,2)

```
void dout(int num , unsigned char val) {
// Sterowanie portem wyjsc cyfrowych
// num (1,2) , val (0 - 255)
  if(num <= 1)
    out8(base + DOUT1, val);
  else
    out8(base + DOUT2, val);
}
```

Przykład 2 Zapis do portu wyjść cyfrowych num - port (1,2) wartości val

4.2 Przetwornik A/D – metoda sukcesywnej aproksymacji

SAR – Sucessive approximation register
DAC – Digital Analog Converter
EOC – End of conversion - sygnał końca konwersji
Vin – napięcie wejściowe
SH – sample and hold - układ próbkujący
Vref - napięcie odniesienia
Clock- impulsy zegarowe

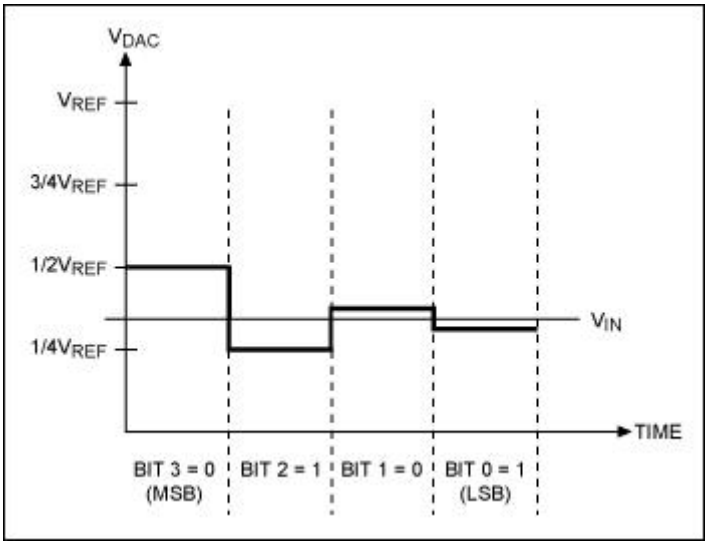


Rys. 4-1 Schemat blokowy przetwornika AD z sukcesywną aproksymacją (z Wikipedia)

Zasada pracy:

Najpierw rejestr SAR jest ustawiany tak że na najstarszej pozycji jest 1 D_{N-1}=1. Przetwornik DA wystawia sygnał Vout = V_{REF} / 2 który jest porównywany z V_{IN} Gdy komparator wskazuje że sygnał wejściowy jest większy od Vout bit ten pozostaje, gdy nie jest zerowany. Następnie ustawiana jest na 1 kolejny bit D_{N-2}=1 i znowu test się powtarza. Po

zakończeniu konwersji rejestr SAR zawiera wynik pomiaru. Generowany jest wtedy sygnał EOC.



Rys. 4-2 Przykład działania przetwornika AD z sukcesywną aproksymacją (z materiałów firmy Maxim)

4.3 Obsługa przetwornika AD– tryb odpytywania

Aby dokonać pomiaru wielkości analogowej za pomoca prztwornika AD należy określić:

- Zakres pomiarowy każdego z wejść (zakres napięć).
- Zakres pracy multipleksera przełączającego wejścia.
- Sposób wyzwalania przetwornika (programowy, układ licznika, zewnętrzny).
- Sposób rozpoznawania końca pomiaru (odpytywanie, przerwanie).
- Sposób przesyłania wyniku pomiaru.

4.3.1 Ustalanie zakresu pomiarowego przetwornika

Każdy z kanałów przetwornika posiada indywidualnie ustawiany zakres pomiarowy. Aby ustalić zakresu pomiarowy przetwornika należy:

1. Wpisać do rejestru BASE+2 numer ustawianego kanału (bity 0-3)
2. Wpisać do rejestru BASE+1 zakres pomiarowy (bity 0-3) zgodnie z poniższa tabelą

Zakres	Unipolar/ bipolar	G3	G2	G1	G0
+/-5	**B**	0	0	0	0
+/-2.5	**B**	0	0	0	1
+/-1.25	**B**	0	0	0	1
+/-0.625	**B**	0	0	1	1
0-10	**U**	0	1	0	0
0-5	**U**	0	1	0	1
0-2.5	**U**	0	1	1	0
0-1.25	**U**	0	1	1	1

Tab. 4-1 Zakresy pomiarowe przetwornika AD

4.3.2 Ustalanie zakresu pracy multipleksera

Karta posiada 8/16 wejść analogowych przełączanych multiplekserem. Aby przygotować układ do pracy należy zaprogramować numer najniższego CL i numer najwyższego mierzonego kanału CH.
Układ zaczyna pomiar od kanału CL. Po dokonaniu pomiaru przechodzi do kolejnego kanału aż do CH po czym powraca do CL. Programowanie zakresu kanałów odbywa się poprzez wpis do rejestru BASE+2 numerów CL (bity 0-3) i CH (bity 4-7).

	B7	**B6**	**B5**	**B4**	**B3**	**B2**	**B1**	**B0**
BASE+2	**CH3**	**CH2**	**CH1**	**CH0**	**CL3**	**CL2**	**CL1**	**CL0**

Tabela 3 Rejestr sterowania multiplekserem karty PCM-3718

4.3.3 Ustalanie źródła wyzwalania, sygnalizacji zakończenia pomiaru i sposobu przesyłania wyniku

Przetwornik AD pracować może w wielu trybach. Tryby te dotyczą:

- Wyzwalania przetwornika,
- Rozpoznawania końca pomiaru
- Przesyłania wyniku pomiaru.

Przetwornik może być wyzwalany:

- programowo
- przez impulsy z umieszczonych na karcie układów

Koniec pomiaru może być:

- Odczytany w rejestrze statusowym
- Sygnalizowany przerwaniem.

Wyniki konwersji mogą być:

- Odczytywane z portów układu
- Zapisywane do pamięci operacyjnej poprzez układ DMA.

O trybie pracy przetwornika decydują wpisy dokonane do rejestru sterującego (adres **BASE+9**).

	B7	B6	B5	B4	B3	B2	B1	B0
BASE+9	INTE	I2	I1	I0	-	DMAE	ST1	ST0

Tabela 4 Rejestr sterujący karty PCM-3718

Bit **INTE** steruje generowaniem przerwań przez kartę.

- Gdy **INTE = 0** generowanie przerwań jest zablokowane.
- Gdy **INTE = 1** oraz **DMAE = 0** oznacza to , że przerwanie jest generowane gdy konwersja AD zostanie zakończona.
- Gdy **INTE = 1** oraz **DMAE = 1** oznacza to, że przerwanie jest generowane gdy z kontrolera DMA przyjdzie impuls T/C wskazujący zakończenie transferu DMA.

Bity **I2, I1, I0** służą do wyboru poziomu przerwania zgodnie z Tabela 5.

I2	I1	I0	Poziom przerwania
0	0	0	-
0	0	1	-
0	1	0	IRQ2
0	1	1	IRQ3
1	0	0	IRQ4
1	0	1	IRQ5
1	1	0	IRQ6
1	1	1	IRQ7

Tabela 5 Poziomy przerwania karty PCL-718

Bity **ST0, ST1** - określenie źródła wyzwalania konwersji przetwornika

ST1	ST0	Źródło wyzwalania
0	X	Programowe
1	0	Zewnętrzne
1	1	Z licznika układu 8254

Tabela 6 Specyfikacja źródeł wyzwalanie karty PCL-718

- Wyzwalanie programowe - zapis dowolnej wartości pod adres **BASE+0**. Wyzwalanie zewnętrzne - pobudzenie linii sterującej **TRIG0** umieszczonej na łączówce karty.
- Wyzwalana wewnętrzne - przez liczniki układu 8254 (licznik 1 i licznik 2).
- Liczniki te dołączone są do generatora kwarcowego o częstotliwości *Fzeg* 10MHz lub 1MHz.
- Wyjście licznika 2 może powodować wyzwolenie konwersji.
- Stopień podziału licznika L_1 i L_2 można zaprogramować i otrzymać żądaną częstotliwość dokonywania konwersji.

$$f = \frac{Fzeg}{L_1 * L_2}$$

Konwersja AD odbywa się metodą sukcesywnej aproksymacji i trwa około 15 μs. Zakończenie konwersji może być wykryte poprzez odczyt rejestru statusowego przetwornika AD lub przez przerwanie.

	B7	B6	B5	B4	B3	B2	B1	B0
BASE+8	**EOC**	**UNI**	**MUX**	**INT**	**CN3**	**CN2**	**CN1**	**CN0**

Tabela 7 Rejestr statusowy przetwornika AD

Operacja zapisu do tego rejestru powoduje wyzerowanie bitu **INT** nie zmieniając pozostałych bitów czyli skasowanie przerwania.

EOC	Wskaźnik zakończenia konwersji. 0 gdy przetwornik jest gotowy a 1 gdy konwersja jeszcze się nie zakończyła .
UNI	Wskaźnik trybu: 0 – tryb bipolarny, 1 – tryb unipolarny.
MUX	Wskaźnik trybu: 0 – 8 kanałów różnicowych, 1 - 16 kanałów pojedynczych z wspólną masą.
INT	Wskaźnik przerwania: 0 – konwersja AD nie jest zakończona od ostatniego wyzerowania tego bitu, 1 – konwersja AD została zakończona i przetwornik jest gotowy do następnego przetwarzania. Jeżeli bit **INTE** rejestru kontrolnego (**BASE+9)** jest ustawiony, wówczas, wraz z ustawieniem bitu **INT** pojawi się przerwanie **IRQ**.
CN0-CH3	Numer kanału, który jest przeznaczony do następnego przetworzenia w przetworniku AD.

Tabela 8 Znaczenie bitów rejestru statusowego przetwornika AD

- Gdy konwersja się zakończy jej wynik może być odczytany z rejestrów danych przetwornika AD.
- Rejestry danych AD służą tylko do czytania i używają adresów **BASE+0** i **BASE+1.**
- Zapis do rejestru spod adresu **BASE+0** powoduje wyzwolenie programowe przetwornika AD (start przetwarzania).

	B7	B6	B5	B4	B3	B2	B1	B0
BASE+0	AD3	AD2	AD1	AD0	C3	C2	C1	C0
BASE+1	AD11	AD10	AD9	AD8	AD7	AD6	AD5	AD4

Tabela 9 Rejestry przetwornika AD karty PCL-718

- Bity **AD11-AD0**-12 bitowa wartość wynikową podawaną przez przetwornik
- Bity **C3-C0** numer kanału AD z którego pochodzi dana wartość.

```
#include "pcl718.h"
#define ADRB 0x300
static int base = ADRB;

card_init(int from, int to, unsigned char zakres) {
// Inicjalizacja karty
// from - kanal poczatkowy, to kanal koncowy
// zakresy pomiarowe:  0-10 V -> 4, 0-5V -> 5,
// 0-2.5 -> 6, 0-1.25 -> 7
   unsigned char val,i;
   printf("inicjacja kanaly od %d  do %d\n",from,to);
   out8(base + CONTR,  0x00);
   val =  in8(base + CONTR);
   if(val != 0x00) {
     printf("Blad inicjalizacji\n");
     exit(0);
   }
   // Ustawienie kan. pocz i konc
   out8(base + MUXR,  (to << 4) | from);
   out8(base + TIMR,  0x00);
   // Odczyt rejestru MUX ----
   val =  in8(base + MUXR);
   // Ustawienie zakresu pomiarowego kanalow
   for(i = from; i<= to; i++) {
      out8(base + MUXR,  i);
      out8(base + RANGE,zakres);
   }
}
```

```
int aread(unsigned int *chan) {
  unsigned int stat,al,ah;
  unsigned int x,xh,xl;
  int i = 0;
  // Start konwersji
   out8(base  ,  0x0);
   do {
       // Odczyt statusu EOC
       stat =  in8(base + STATR);
       i++;
       if(i >= 0xFFFF) return(-1);
   } while((stat & 0x80) != 0);
   al  =  in8(base + ADL);
   ah =  in8(base + ADH);
   xh = ah << 4  ;
   xl = al >> 4;
   // printf("A1: %04X  A0: %4X i= %d \n",ah,al,i);
   *chan = al & 0x0F;
    x = xh + xl;
    return(x);
}
```

```
main() {
   int val,val2, chn,j ;
   unsigned char d1,d2, i = 0;
   printf("Program startuje \n");
   ThreadCtl( _NTO_TCTL_IO, 0 );
   base = mmap_device_io(16,ADRB);
   card_init(0,3,5);
   do {
      for(j=0; j<4; j++) {
           val = aread(&chn);
           printf(" %d - %d ",chn,val);
      }
      d1 = dinp(1);
      printf("we1 %2X  \n",d1);
      usleep(500000);
      // sleep(1);
      dout(2,i);
      i++;
    } while(1);
}
```

Przykład 4-1 Obsługa przetwornika AD karty PCM-3718 w trybie odpytywania

5. Obsługa w trybie przerwań

Obsługa przetwornika AD w trybie odpytywania posiada wady:

- odpytywanie statusu przetwornika powoduje utratę czasu procesora.
- trudno jest uzyskać precyzyjnie określony moment wyzwolenia przetwornika.

Jeżeli chcemy odczytywać wartości z przetwornika AD w ściśle określonych momentach czasu do wyzwalania przetwornika należy użyć liczników układu 8254 które generują impulsy wyzwalające konwersję. Zakończenie konwersji sygnalizowane jest przerwaniem.

Programowanie karty:

- Ustalenie trybu sygnalizowania końca konwersji i wyzwalania
- Ustalenie współczynnika podziału liczników

Ustawienie rejestrów sterujących:

	B7	B6	B5	B4	B3	B2	B1	B0
BASE+9	INTE	I2	I1	I0	-	DMAE	ST1	ST0
	1	1	0	1	-	0	1	1

Tabela 5-1 Zawartość rejestru sterującego karty PCM-3718 w trybie przerwań

INTE = 1 zakończenie konwersji sygnalizowane przerwaniem
Bity B6-B4 numer przerwania
Bity B1 i B0 wyzwalanie konwersji z liczników układu 8254

	B7	B6	B5	B4	B3	B2	B1	B0
BASE+10	-	-	-	-	-	-	TC1	TC0

Tabela 5-2 Rejestr TIMR konfiguracji liczników

TC0=0 układ wyzwalania jest stale włączony
TC0=1 włączony jest wtedy, gdy wejście **TRIG0** ma poziom wysoki.

TC1=0 licznik 0 zlicza impulsy podawane z zewnętrznego źródła
TC1=1 to podłączony jest do wewnętrznego źródła 100KHz

Programowanie liczników układu 8254:

BASE+12	Licznik 0 (odczyt/zapis)
BASE+13	Licznik 1 (odczyt/zapis)
BASE+14	Licznik 2 (odczyt/zapis)
BASE+15	Słowo sterujące

Tabela 5-3 Rejestry układu licznikowego 8254

Proces główny:

- Inicjuje kartę poprzez wykonanie funkcji **`card_init(),`**
- Ustawia stopień podziału liczników funkcją **`pcl_counter(20,10),`**
- Ustawia zakres przemiatanych kanałów - funkcja **`pcl_mux(0,0`**). Inicjuje zdarzenie **`event`**
- Wykonuje funkcję **`InterruptAttach(ADC_INT,handler,NULL,0,0`**). Funkcja ta instaluje handler przerwania **`ADC_INT`**
- Odmaskowuje przerwania **`InterruptUnmask(ADC_INT,id);`**

Obsługa przerwania – funkcja **`handler`**:

1. Odczyt młodszego i starszego bajtu wyniku z rejestrów **BASE** i **BASE+1**. Złożenie wartości razem.
2. Wpis uzyskane z przetwornika AD wartości do bufora cyklicznego.
3. Przekazanie odczytanej wartości na przetwornik DA **`pisz_da(val)`**
4. Wyświetlenie wartości na linijce diodowej **`led_disp(val);`**
5. Skasowanie przerwania przez zapis do rejestru **BASE+8.**

Program odczytuje wartość analogową z przetwornika AD i przekazuje ją na przetwornik DA. Jest to podstawa do realizacji algorytmów DSP.

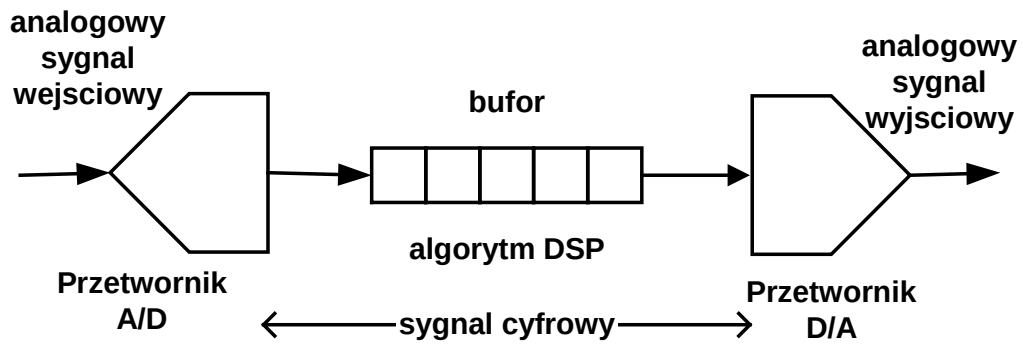
```
// System QNX Neutrino -------------------------------------
// Karta PCM 3718  -  przetwornik  AD - tryb przerwan
// (C) Jedrzej Ulasiewicz 2010
// Przerwanie AD - 5 Odblokowac w BIOS plyty gdy zablokowane
// Przetwornik DA na plycie PCM3718HO, pin 19 złącza P1 analog
#include <sys/neutrino.h>
#include <hw/inout.h>
#include <sys/mman.h>
#include <stdint.h>
#include <stdio.h>
#include <stdlib.h>
#define ADRB 0x300 // Adres bazowy karty
#define ADL    0     // Mlodszy bajt AD + kanal
#define ADH    1     // Starszy
#define RANGE  1     // Wzmocnienie kanalu
#define MUXR   2     // Kanal konc i pocz
#define DALO   4     // Mlodszy bajt DA
#define DAHI   5     // Starszy  bajt DA
#define STATR  8     // Rejestr statusu
#define CONTR  9     // Rejestr sterujacy
#define TIMR  10     // Start timera
#define DIOH  11     // Wyjscia cyfrowe
#define COUNT0 12    // Licznik 0
#define COUNT1 13    // Licznik 1
#define COUNT2 14    // Licznik 3
#define COUNTC 15    // Rej. ster. licznikow

// Adres bazowy karty
static int base = ADRB;

#define BSIZE 1000
#define ADC_INT 5  // Przerwanie karty

static short int buf[BSIZE];
volatile int head,tail,count, id, cnt = 0;
// struct sigevent event;
uintptr_t port;

void card_init(void ) {
// Inicjalizacja karty
// from - kanal poczatkowy, to kanal koncowy
   unsigned char val1,val2 ;
   // INT, IRQ5, DMAE = 0, wyzw z 8 MUXR     54
   val1 = 0xD3;
   out8(base +  CONTR    ,  val1);
   val2  =  in8(base +  CONTR    );
   if(val2 != val1) {
     printf("Blad inicjalizacji karty\n");
     exit(0);
   }
   printf("Status:%x  Control: %x \n",val1,val2);
```

```
   // Pacer enable
   out8(base +  TIMR      , 0x01);
   // INTE I2 I1 I0 X DMA ST1 ST0
   //   1    1   0 1 0  0      1     1
   out8(base +  CONTR,  0xD3);
 }


 void set_range(int from, int to, unsigned char zakres)
 // Ustawienie wzmocnienia kanalow karty
 // from - kanal poczatkowy, to kanal koncowy
 // zakr pom  0-10 V -> 4, 0-5V -> 5, 0-2.5 -> 6, 0-1.25 -> 7
 {
    int i;
    for(i = from; i<= to; i++) {
      out8(base + MUXR,  i);
      out8(base + RANGE,zakres);
   }
 }

 void pcl_counter(int l1, int l2) {
    // Programowanie  licznikow
    //  licznik 1
    out8(base +  COUNTC , 0x74) ;
    out8(base + COUNT1  , l1 & 0xFF);
    out8(base + COUNT1  , l1 >> 8);
     //  licznik 2
    out8(base +  COUNTC , 0xB4) ;
    out8(base + COUNT2  , l2 & 0xFF);
    out8(base + COUNT2  , l2 >> 8);
}

void pcl_mux(int first, int last) {
 out8(base + MUXR,  (last << 4) | (first & 0x0F));
}


void pisz_da(unsigned short int x)
// Zapis kanalu DA
{ unsigned short int yh,yl;
   unsigned short int y;
   y = x;
   y = y & 0x0FFF;
   yl  = (y & 0x0F);
   yl = yl << 4;
   yh = y >> 4;
   out8(base + DALO,yl);
   out8(base + DAHI, yh);
}

void led_disp(unsigned short int val)
```

```
{ unsigned short int y;
  y = val >> 9;
  out8(base + DIOH,0xFF>>(7-y));
}

// struct sigevent event;
volatile int icnt = 0;
int intid = 0;
int sec = 0;

const struct sigevent *handler(void *arg, int id)
// Handler obslugi przerwania
{   unsigned short int chn, val,ah,al;
    unsigned int xh,xl;
    // Odczyt wart. pomiarowych
    al =  in8(base + ADL);
    ah =  in8(base + ADH);
    chn = al & 0x0F;
    xh  = ah << 4;
    xl  = al  >> 4;
    val = xh + xl;
    icnt++;
   // Zapis do bufora cykliczneho
    buf[head] = val;
    head = (head+1) %BSIZE;
    cnt++;
    // Wyswietlenie na linijce diodowej
    led_disp(val);
    pisz_da(val);
    // Skasowanie przerwania
    out8(base  +  STATR      ,  0x00);
    return(NULL);
}

int main() {
   int id;
   time_t t1,t2;
   printf("Start watku odczyt\n");
   ThreadCtl( _NTO_TCTL_IO, 0 );
   port = mmap_device_io(16,base);
   printf("port %x\n",port);
   // Inicjacja trybu pracy karty ------------
   card_init();
   // Ustawienie zakresu kanalow --------
   pcl_mux(0,0);
   // Ustawienie zakresu pomiarowego --
   set_range(0,0,5);
   // Ustawienie czestotliwosci licznikow-
   pcl_counter(10,100);
   id = InterruptAttach(ADC_INT,handler,NULL,0,0);
   if(id < 0) {
```

```
        perror("install"); exit(0);
    }
    printf("Handler zaininstalowany: %d \n",id);
    t1 = time(NULL);
    // Skasowanie przerwania
    out8(base + STATR,0x00);
    // Odmaskowanie przerwania
    InterruptUnmask(ADC_INT,id);
    do {
        printf("icnt: %d cnt: %d\n",icnt,cnt);
       sleep(1);
    } while(icnt < 10000);
    InterruptMask(ADC_INT,id);
    InterruptDetach(id);
    t2 = time(NULL); t2 = t2 -t1;
    printf("pomiarow %d czas %d sek \n",icnt,t2);
    printf("Koniec\n");
    return 0;
}
```

Przykład 5-1 Obsługa przetwornika AD w trybie przerwań



Rys. 5-1 Zasada cyfrowego przetwarzania sygnałów DSP