LABORATORIUM W PC-CIE

Miernik częstotliwości do komputera PC, część 2

kit-AVT269

Kończymy opis karty do pomiaru częstotliwości prezentacją zasad i algorytmu obsługi programowej rejestrów sterujących i danych. Informacje zawarte w artykule są niezwykle pomocne dla konstruktorów zamierzających napisać własną wersję oprogramowania.

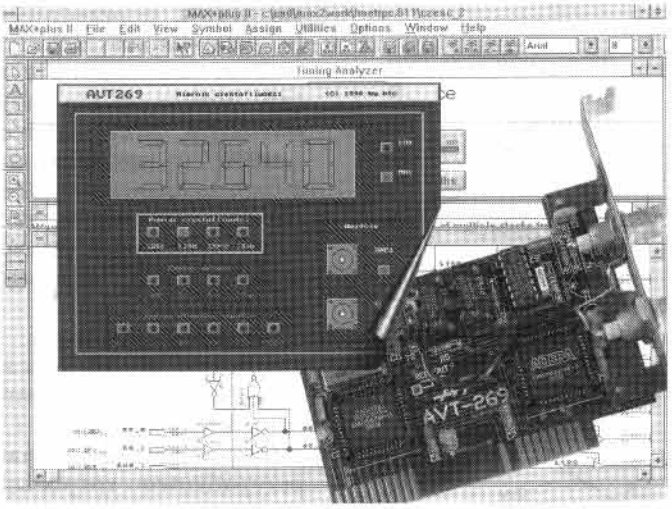


Tabela 1. Dopuszczalne wartości bitów Rejestru Sterującego.

pomiar częstotliwości

✓ pomiar z wejścia BNC1

B7	B6	B5	B4	B3	B2	B1	B0	zakres
1	0	0	-	-	0	1	0	32MHz
1	0	0	-	-	1	0	0	3.2MHz
1	0	0	-	-	1	1	0	320kHz
1	0	0	-	-	1	0	1	32kHz

✓ pomiar z wejścia BNC2

1	0	1	-	-	0	1	0	32MHz
1	0	1	-	-	1	0	0	3.2MHz
1	0	1	-	-	1	1	0	320kHz
1	0	1	-	-	1	0	1	32kHz

pomiar okresu

✓ pomiar z wejścia BNC1, aktywny stan niski

B7	B6	B5	B4	B3	B2	B1	B0	zakres
0	1	0	1	-	0	1	1	320s
0	1	0	1	-	0	1	0	32s
0	1	0	1	-	0	0	1	3,2s
0	1	0	1	-	0	0	0	0,32s

✓ pomiar z wejścia BNC2, aktywny stan niski
jak dla pomiaru z wejścia BNC1 + zmiana bitu B5 na "1"

✓ pomiar z wejścia BNC1, aktywny stan wysoki
jak dla pomiaru z wejścia BNC1 + zmiana bitu B4 na "0"

✓ pomiar z wejścia BNC2, aktywny stan wysoki
jak dla pomiaru z wejścia BNC1 + zmiana bitów: B4 na "0", B5 na "1"

pomiar czasu trwania impulsu

✓ pomiar z wejścia BNC1, aktywny stan niski

B7	B6	B5	B4	B3	B2	B1	B0	zakres
1	1	0	1	-	0	1	1	320s
1	1	0	1	-	0	1	0	32s
1	1	0	1	-	0	0	1	3,2s
1	1	0	1	-	0	0	0	0,32s

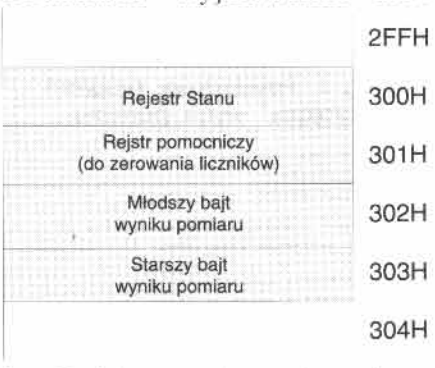
✓ pomiar z wejścia BNC2, aktywny stan niski
jak dla pomiaru z wejścia BNC1 + zmiana bitu B5 na "1"

✓ pomiar z wejścia BNC1, aktywny stan wysoki
jak dla pomiaru z wejścia BNC1 + zmiana bitu B4 na "0"

✓ pomiar z wejścia BNC2, aktywny stan wysoki
jak dla pomiaru z wejścia BNC1 + zmiana bitów: B4 na "0", B5 na "1"

Obszar adresowy

Na **rys.9** przedstawiona została mapa adresowa obszaru we/wy wykorzystywanego przez kartę miernika. Są to adresy 300H..303H z przestrzeni adresowej zarezerwowanej w komputerze PC dla kart prototypowych. Ze względu na uproszczenie konstrukcji elektrycznej i łatwość oprogramowania karty przez mniej wprawnych programistów, nie ma możliwości zmiany tych adresów. Należy więc zapewnić bezkonfliktowy dostęp do tych adresów (są one domyślnie wykorzystywane przez większość kart sieciowych Ethernet, oraz przez część kart dźwiękowych - adresy te w tego typu kartach najczęściej można zmienić na drodze programowej, bez konieczności wyjmowania kart z komputera). Poniżej opisano funkcje, jakie pełnią poszczególne porty I/O (dalej nazywane rejestrami karty).



Rys. 9. Adresy we/wy wykorzystywane przez kartę miernika częstotliwości.

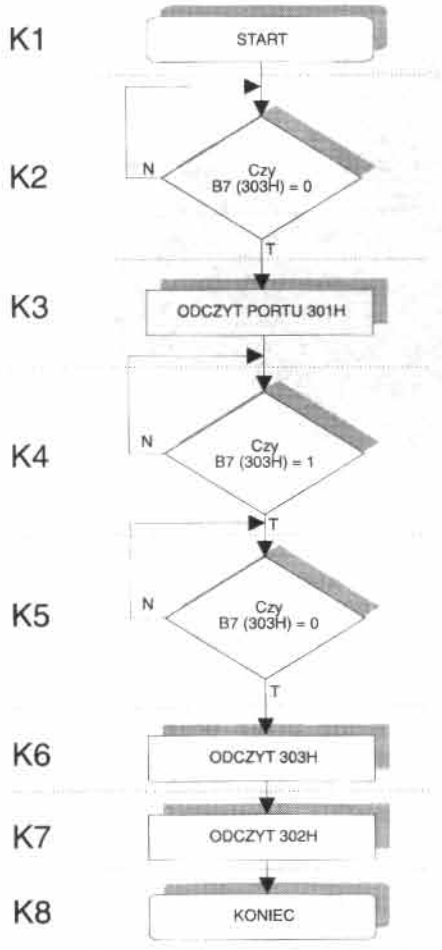
300H - port do zapisu. Rejestr Sterujący (RS) karty. Poprzez wpisywanie do tego rejestru zmienia się tryb pracy karty. Funkcje pełnione przez poszczególne bity Rejestru Sterującego przedstawiono na **rys.10**. W **tabeli 1** zawarto wszystkie dopuszczalne kombinacje, jakie można wpisać do Rejestru Sterującego.

301H - port do zapisu/odczytu. Wpisanie dowolnej wartości lub odczyt z tego portu powoduje wyzerowanie liczników. Konieczne jest wykonanie tego typu operacji przed rozpoczęciem cyklu pomiarowego.

302H - port do odczytu. Po zakończeniu cyklu pomiarowego w rejestrze tym znajduje się młodszy bajt (D7..D0) wyniku pomiaru.

303H - port do odczytu. Ustawienie przez kartę pomiarową najstarszego bitu (B7) oznacza zakończenie cyklu pomiarowego - jest to sygnał, że w rejestrach 302H i 303H znajduje się wynik pomiaru. W przeciwnym przypadku w rejestrach tych znajduje się przypadkowa wartość. Młodsze siedem bitów rejestru 303H (B6..B0) to starsze bity (D14..D8) wyniku pomiaru.

Rys. 10. Znaczenie bitów w Rejestrze Sterującym.



Rys. 11. Algorytm procedury pomiarowej.

Sposób programowania karty

Na **rys.11** przedstawiono algorytm procedury cyklu pomiarowego. W większości zastosowań, procedura ta będzie wykonywana cyklicznie np. w pętli *for*. Ze względu na zależności czasowe dodane są dwa warunki kontrolne: K2 i K5. Mają one zabezpieczyć przed kilkukrotnym odczytem wyniku pomiaru z tego samego cyklu pomiarowego na szybkich komputerach (w których obróbka wyników pomiaru, np. archiwizowanie lub wyświetlanie, zajmuje ułamki sekund). Sytuacja ta występuje przy długo trwającym pomiarze (np. impulsu kilkusekundowego), gdy przy kolejnej iteracji nie został zakończony pomiar, a w rejestrze 303H znajduje się wynik poprzedniego pomiaru (z ustawionym B7).

Przy pomiarach nieiteracyjnych (np. wykonywanie pojedynczych pomiarów przy każdym uruchomieniu swojego programu) można z tych zabezpieczeń zrezygnować, chociaż wprowadza to niezauważalne opóźnienie pracy programu w porównaniu z czasem trwania pomiaru.

W przedstawionym algorytmie należałoby dodać jeszcze warunki umożliwiające przerwanie pomiaru w krokach K2, K3 i K5. Łatwo sobie wyobrazić sytuację, kiedy przypadkowo rozpoczęliśmy pomiar np. impulsu 20 sekundowego i nie chcemy czekać tyle czasu na zakończenie tego pomiaru. Należałoby więc umożliwić przerwanie wykonywania pomiaru np. przez wciśnięcie klawisza. Zależy to już jednak od inwencji programisty.

W kroku K3 poprzez odczyt portu 301H realizowane jest kasowanie liczników.

Wynik pomiaru zazwyczaj interesuje nas jako liczba całkowita. Na **rys.12** przedstawiono graficznie konwersję z dwóch liczb 8-bitowych na liczbę 16-bitową. Przykładowy fragment programu realizujący tą konwersję pokazano na **list.1**.

Najtrudniejszą częścią programu jest obliczanie wartości Rejestru Sterującego. Najbardziej czytelną realizacją tego zadania jest obliczanie tej wartości poprzez sumowanie logiczne poszczególnych bitów (łatwo zrealizować i łatwo znaleźć ewentualne błędy). Przykładowo, jeżeli chcemy dokonać pomiaru częstotliwości z wejścia BNC1, na zakresie 32MHz wystarczy wykonać następującą operację:

RS = 100xx010b;

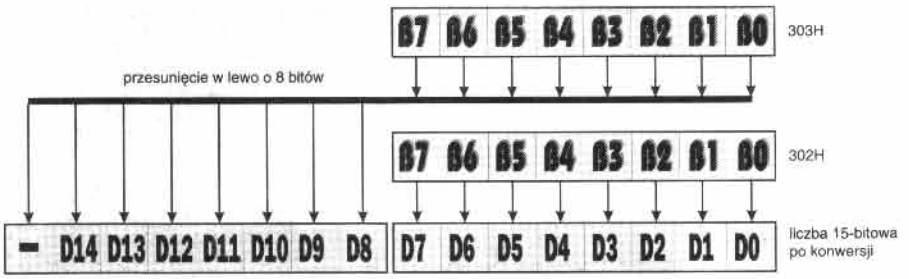
Na **list.2** przedstawiona jest procedura obliczająca wartość Re-

Listing 1. Przykładowy program "obliczający" wynik pomiaru.

```
.........
unsigned long wynik;
int bajt_mlodszy;
int bajt_starszy;
.........
/* tu jest dokonywany pomiar zależny od
wartości Rejestru Sterującego */
.........
bajt_mlodszy = inportb( 0x302 );
bajt_starszy = inportb( 0x303 );
bajt_starszy = bajt_starszy & 0x3f;      /*
zerowanie najstarszego bitu */
                /* 0x3f = 00111111 */
wynik = ( bajt_starszy << 8 );  /*
przesunięcie w lewo o ilość pozycji */
wynik = wynik | bajt_mlodszy;   /* suma
logiczna */
.........
```

Listing 2. Procedura obliczająca wartość Rejstru Sterującego.

```
.........
extern int pomiar;      /* aktualnie wykonywany
                           pomiar */
extern int pomiar_f, pomiar_t, pomiar_ti;
extern int wejscie, stan;
.........
void oblicz_RS( void )
{
  int bajt = 0;

  if( wejscie == BNC2 )
    bajt = bajt | B5;

  if( stan == STAN_L )
    bajt = bajt | B4;

  switch( pomiar )
  {
    case POM_F :
         {
           bajt = bajt | B7;

           if( pomiar_f == POM_F0 )
           {
             delay( 200 );
             bajt = bajt | B0;
           }
           else
           if( pomiar_f == POM_F1 )
           {
             delay( 200 );
             bajt = bajt | B1;
           }
           else
           if( pomiar_f == POM_F2 )
             bajt = bajt | B1 | B0;
           else
             bajt = bajt | B2 | B0;
           break;
         }

    case POM_T :
         {
           bajt = bajt | B6;

           if( pomiar_t == POM_T0 )
             bajt = bajt | B1 | B0;
           else
           if( pomiar_t == POM_T1 )
             bajt = bajt | B1;
           else
           if( pomiar_f == POM_T2 )
             bajt = bajt | B0;
           else
             bajt = bajt;
           break;
         }

    case POM_TI :
         {
           bajt = bajt | B6 | B5;

           if( pomiar_ti == POM_TI0 )
             bajt = bajt | B1 | B0;
           else
           if( pomiar_ti == POM_TI1 )
             bajt = bajt | B1;
           else
           if( pomiar_ti == POM_TI2 )
             bajt = bajt | B0;
           else
             bajt = bajt;
           break;
         }

  } /* switch pomiar */

  /* wpisanie Rejestru Sterującego */
  outportb( 0x300, bajt );
} //-------------------------- oblicz_RS
.........
```



Rys. 12. Zobrazowanie konwersji dwóch liczb 8-bitowych na liczbę 16-bitową.

jestru Sterującego z dołączanego do karty programu fmetr.exe. *Bx* (gdzie x = 0..7) oznacza 8-bitową stałą z ustawionym bitem *x* (np B4 = 0FH = 00010000). Wykorzystane są tutaj następujące zmienne i stałe (drukowanymi literami oznaczone są stałe):

- ✗ **pomiar** określa jakiego rodzaju ma być dokonany pomiar (pomiar częstotliwości, okresu, czy długości trwania impulsu). Dopuszczalne wartości:
  POM_F pomiar częstotliwości
  POM_T pomiar okresu
  POM_TI pomiar czasu trwania impulsu
- ✗ **pomiar_f** określa zakres pomiarowy dla pomiaru częstotliwości. Dopuszczalne wartości:
  POM_F0 zakres 32MHz
  POM_F1 zakres 3,2MHz
  POM_F2 zakres 320kHz
  POM_F3 zakres 32kHz
- ✗ **pomiar_t** określa zakres pomiarowy dla pomiaru okresu. Dopuszczalne wartości:
  POM_T0 zakres 320 sekund
  POM_T1 zakres 32 sekundy
  POM_T2 zakres 3,2 sekundy
  POM_T3 zakres 0,32 sekundy
- ✗ **pomiar_ti** określa zakres pomiarowy dla pomiaru długości trwania impulsu. Dopuszczalne wartości:
  POM_TI0 zakres 320 sekund
  POM_TI1 zakres 32 sekundy
  POM_TI2 zakres 3,2 sekundy
  POM_TI3 zakres 0,32 sekundy
- ✗ **wejście** zmienna określa na którym wejściu: BNC1 czy BNC2, będzie mierzony sygnał.
- ✗ **stan** zmienna określa czy mierzony ma być czas trwania stanu niskiego czy wysokiego. Dopuszczalne wartości:
  STAN_L mierzony będzie czas trwania stanu niskiego
  STAN_H mierzony będzie czas trwania stanu wysokiego

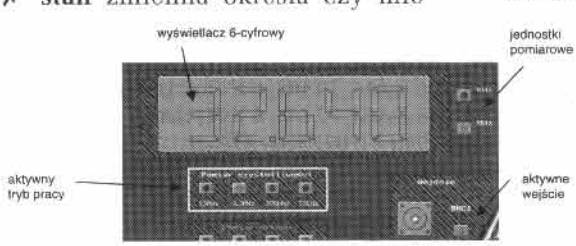
Na **list. 3** przedstawiono procedurę pomiarową napisaną w języku C, zastosowaną w dołączanym do karty programie fmetr.exe. Procedury: pomiarowa, obliczająca wartość Rejestru Sterującego i obliczająca wynik pomiaru działają zgodnie z wcześniej przytoczonymi wskazówkami.

Obsługa dołączonego oprogramowania

Dołączony do karty (w wersji A lub B) program fmetr.exe jest przykładową aplikacją umożliwiającą wykorzystanie karty jako miernika częstotliwości, okresu i czasu trwania impulsu. Program pracuje w trybie graficznym VGA (640x480/16 kolorów).

Program umożliwia pełne wykorzystanie karty pomiarowej. Wyniki pomiarów wyświetlane są na 6-cyfrowym wyświetlaczu (**rys.13**), z zaznaczeniem jednostek pomiaru (kHz/MHz, µs/s). W programie możliwa jest: zmiana trybu pracy karty (klawiszem TAB), zmiana zakresów pomiarowych (kursorami: LEWO/PRAWO), zmiana wejść pomiarowych (kursorami: GÓRA/DÓŁ), ustawienie czy mierzony jest czas trwania stanu niskiego czy wysokiego dla pomiaru długości trwania impulsu (klawisze H i L).

Aktualny tryb pracy karty sygnalizowany jest jasną obwódką wokół prostokąta z zaznaczonymi zakresami pomiarowymi (rys.13). Zakres pomiarowy pokazywany jest przez wyróżnienie jednego z pól kolorem czerwonym.



Rys. 13. Widok ekranu programu pomiarowego.

Zakończenie pracy programu następuje po wciśnięciu klawisza F10.

Opisany powyżej program jest tylko przykładem wykorzystania karty pomiarowej. Zastosowanie karty można rozszerzyć dołączając do niej różnego rodzaju przystawki, np. stosując przetwornik U/f otrzymamy dobrej jakości woltomierz. Jedynym ograniczeniem zastosowań karty jest tylko wyobraźnia programisty i konstruktora przystawek...

Paweł Zbysiński

Listing 3. Procedura pomiarowa dołączanego do karty programu fmetr.exe.

```
.........
unsigned long wynik;
.........
int JEST_0( void )
/* funkcja zwraca OK jeżeli na B7 w 0x303
pojawi się 0 */
{
  int bajt;

  bajt = inportb( 0x303 );
  while( (bajt&B7) != 0)
  {
    bajt = inportb( 0x303 );
  }

  return OK;
} //-------------------------JEST_0

int JEST_1( void )
/* funkcja zwraca OK jeżeli na B7 w 0x303
pojawi się 1 */
{
  int bajt;

  bajt = inportb( 0x303 );
  while( (bajt&B7) == 0)
  {
    bajt = inportb( 0x303 );
  }

  return OK;
} //-------------------------JEST_1


void pomiar( void )
/* funkcja realizuje algorytm pomiarowy z rys.
11 w zmiennej globalnej wynik zapisywany jest
wynik pomiaru wykorzystana jest funkcja
oblicz_RS z listingu 2 */
{
  int bajt1, bajt2;

  oblicz_RE();

  while( !JEST_0() )  // czekaj na 0 na B7
    ;
  // zerowanie liczników
  bajt1 = inportb( 0x391 );
  // wartość bajt1 jest tu nieistotna

  while( JEST_1() )  // czekaj na 0 na B7
    ;

  while( !JEST_0() )
    ;
  // odczytaj wyniki
  bajt1 = inportb( 0x303 );
  bajt2 = inportb( 0x302 );

  // zerowanie najstarszego bitu
  bajt1 = bajt1 & (~B7);

  // przesunięcie w lewo o osiem pozycji
  wynik = ( bajt1 << 8);
  // suma logiczna
  wynik = wynik | bajt2;

} //-------------------------pomiar
.........
```