dr inż. Stanisław Bocian
mgr inż. Jarosław Iwanowski
Instytut Pojazdów Szynowych „TABOR”

Układ przetwarzania sygnałów analogowych z czujników reluktancyjnych i program do pomiaru prędkości w pojazdach szynowych

W artykule opisano układ przetwarzania sygnałów analogowych z reluktancyjnych czujników prędkości z wykorzystaniem mikrokontrolera PSoC firmy CYPRESS. Mikrokontrolery PSoC są jedynymi układami wyposażonymi w konfigurowane peryferia analogowe i cyfrowe. Mikrokontrolery PSoC są wyposażone w możliwość dynamicznej rekonfiguracji wewnętrznych bloków podczas pracy. Mikrokontroler zapewnia właściwy programowany poziom sygnału obrabianego w przypadku minimalnego sygnału z czujnika reluktancyjnego. W ten sposób otrzymuje się kompletnie uformowany sygnał pomiarowy pozyskany z czujników reluktancyjnych w trudnych warunkach eksploatacyjnych na taborze kolejowym. Przedstawiono program pomiaru częstotliwości w 4 kanałach wykorzystując sygnały z 4 czujników reluktancyjnych.
Artykuł powstał w wyniku realizacji projektu badawczego KBN nr. N N509 398236 „Mikrosystemy cyfrowe do inteligentnego, rozproszonego i współbieżnego sterowania pojazdami szynowymi.”

1. Wprowadzenie

Nowoczesne pojazdy szynowe wyposażone są w podzespoły na coraz wyższym poziomie technicznym wykorzystujące nowe technologie elektroniczne i informatyczne. Podsystemy pojazdów szynowych tworzą elementy rozległych systemów informatycznych i informacyjnych. Układy te zbudowane są z systemów wieloprocesorowych – sterowników programowalnych PLC (ang. Programmable Automation Controller). Sterowniki programowalne PLC stosowane obecnie w sterowaniu i diagnostyce pojazdów szynowych, jak również nowsze rozwiązanie sterowników – Programowane Systemy Sterowania PAC (ang. Programmable Automation Controller), które łączą właściwości sterownika PLC, jego wytrzymałości mechanicznej i wysokiej jakości wykonania z funkcjonalnością komputerów osobistych PC nie spełniają wszystkich wymaganych parametrów, które spełniają mikrosystemy cyfrowe.

Rozwój układów automatyki wymaga stosowania nowych rozwiązań mikroprocesorów, układów wejść i wyjść oraz układów komunikacyjnych. W procesie projektowania cyfrowych układów sterowania oprócz stosowania modeli specyfikacji formalnej bardzo ważne jest sprecyzowanie docelowej platformy realizacyjnej: sprzętowej, programowej i sprzętowo – programowej. Realizacja sprzętowa to struktury układowe małej i średniej skali integracji oraz nowoczesne matryce reprogramowalne. Realizacja programowalna to połączenie pewnego zestawu instrukcji (program) oraz odpowiednich struktur sprzętowych (np. mikroprocesor, pamięć) zdolnych do wykonania działań zapisanych w kodzie tego programu. Zalety i wady realizacji sprzętowej oraz programowej, rozpatruje się pod względem dwóch podstawowych kryteriów:

– czas reakcji – szybsze rozwiązanie sprzętowe,
– koszty realizacji – przyjmuje się niższe dla rozwiązań programowych, ze względu na niższą cenę zarówno samych układów, jak i narzędzi wspomagający proces projektowania.

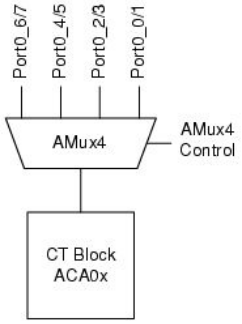
Własności te wydają się wzajemnie sprzeczne, dlatego należałoby zastosować jednocześnie obie metody stosując kompromis, łącząc zalety szybkości działania z niskimi kosztami produkcji. Funkcje takie realizują mikrosystemy cyfrowe. Układy takie buduje wiele firm, takie jak: ATMEL, CYPRESS i TRISCEND. Wspólną cechą tych układów jest posiadanie rdzenia mikroprocesorowego oraz programowalnej części sprzętowej (systemy o realizacji sprzętowo – programowej). Pod pojęciem mikrosystem cyfrowy rozumie się taki układ scalony, który w swej strukturze integruje (obok innych) właśnie te dwa elementy. Umożliwia to zmniejszenie rozmiarów gabarytowych

produktu, zużycia energii, a nawet kosztów związanych z produkcją w porównaniu do rozwiązań bazujących na łączeniu osobno struktur sprzętowych i mikroprocesorowych. W artykule opisano układ z wykorzystaniem mikrosystemu cyfrowego PSoC CY29466 firmy CYPRESS.

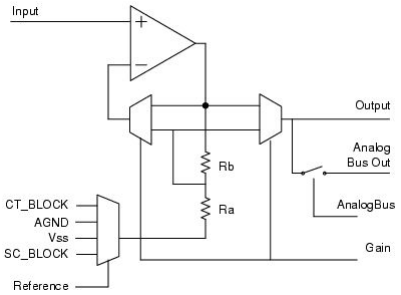
2. Sposób wykorzystania zasobów mikrokontrolera do przetworzenia sygnałów analogowych

W pracy [1] przedstawiono uniwersalne układy wejść – wyjść do zastosowania w pojazdach szynowych z wykorzystaniem mikrosystemów cyfrowych PSoC CY29466 firmy CYPRESS. W pracy [2] przedstawiono układ przetwarzania sygnałów analogowych z reluktacyjnych czujników prędkości w pojazdach szynowych. Przedstawiona praca jest poszerzeniem publikacji [2] o program do pomiaru częstotliwości układów przetwarzania. sygnałów analogowych z reluktacyjnych czujników prędkości w pojazdach szynowych. Mikrokontrolery PSoC są jedynymi układami wyposażonymi w konfigurowalne peryferia analogowe i cyfrowe. Dzięki takiemu wyposażeniu jeden typ mikrokontrolera może być stosowany w różnorodnych aplikacjach, do wymogów których będą dostosowywane uniwersalne zasoby konfigurowalnych bloków Analog PSoC Array oraz Digital PsoC Array.Bloki cyfrowe są ułożone w wiersze, a analogowe w kolumny. Każdy wiersz składa się z 4 bloków. Dwa bloki DBBXX mogą pełnić prostsze funkcje cyfrowe, a dwa DCBXX mogą służyć jeszcze jako bloki komunikacyjne. Jedna kolumna analogowa jest zbudowana z trzech bloków. Jeden jest blokiem liniowym ciągłym, dwa następne są zbudowane na przełączanych pojemnościach. Z wyżej wymienionych bloków można budować różne układy analogowe, cyfrowe i mieszane. Począwszy od prostych timerów, poprzez wzmacniacze, kończąc na szerokiej gamie przetworników analogowo – cyfrowych. Mikrokontrolery PSoC są wyposażone w możliwość dynamicznej rekonfiguracji wewnętrznych bloków podczas pracy.

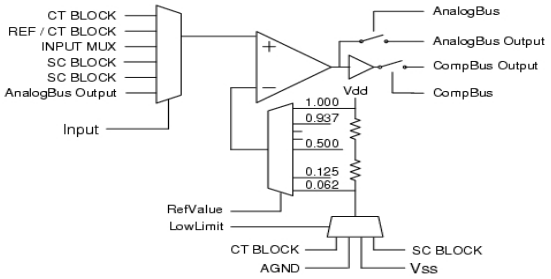
Schemat blokowy układu pomiarowego konfiguracji podstawowej przedstawiono na rys.5. Cztery sygnały analogowe zostały podzielone tak, by wartość maksymalna napięcia z czujnika reluktancyjnego nie przekroczyła wartości dopuszczalnej dla wejść mikrokontrolera. Następnie sygnały te wprowadzono poprzez porty wejściowe zadekretowane jako analogowe na wejście multipleksera analogowego (rys.1.). Jest to sterowany przez program moduł, który przyłącza kolejne wejścia do wspólnego toru pomiarowego.
Pierwszym elementem tego toru jest wzmacniacz analogowy (rys. 2.) o wzmocnieniu zaprogramowanym na k=24.
Zapewnia on właściwy poziom sygnału obrabianego dla minimalnych wartości sygnału z czujnika reluktancyjnego. Tak obrobiony sygnał jest przekazywany do modułu komparatora analogowego (rys. 3.), na wyjściu którego otrzymujemy sygnał prostokątny o prawidłowo uformowanych zboczach.



Rys. 1. Schemat blokowy modułu multipleksera analogowego

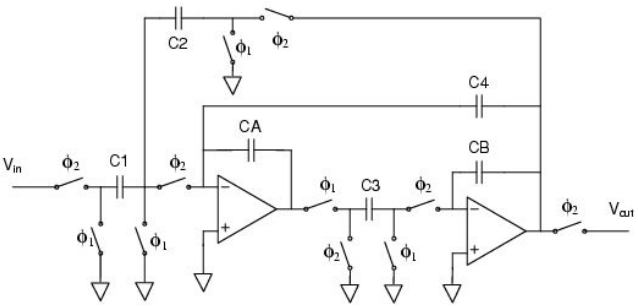


Rys. 2. Schemat blokowy modułu PGA – programowalny wzmacniacz wejściowy

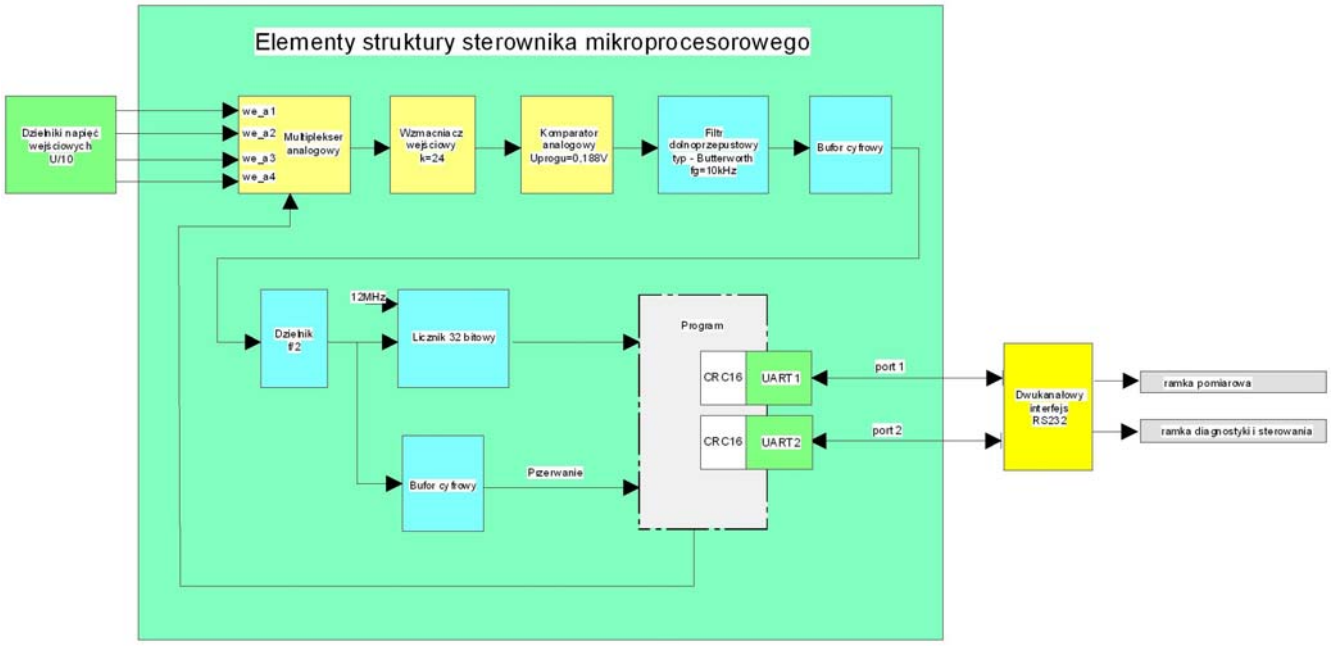


Rys. 3. Schemat blokowy modułu komparatora analogowego

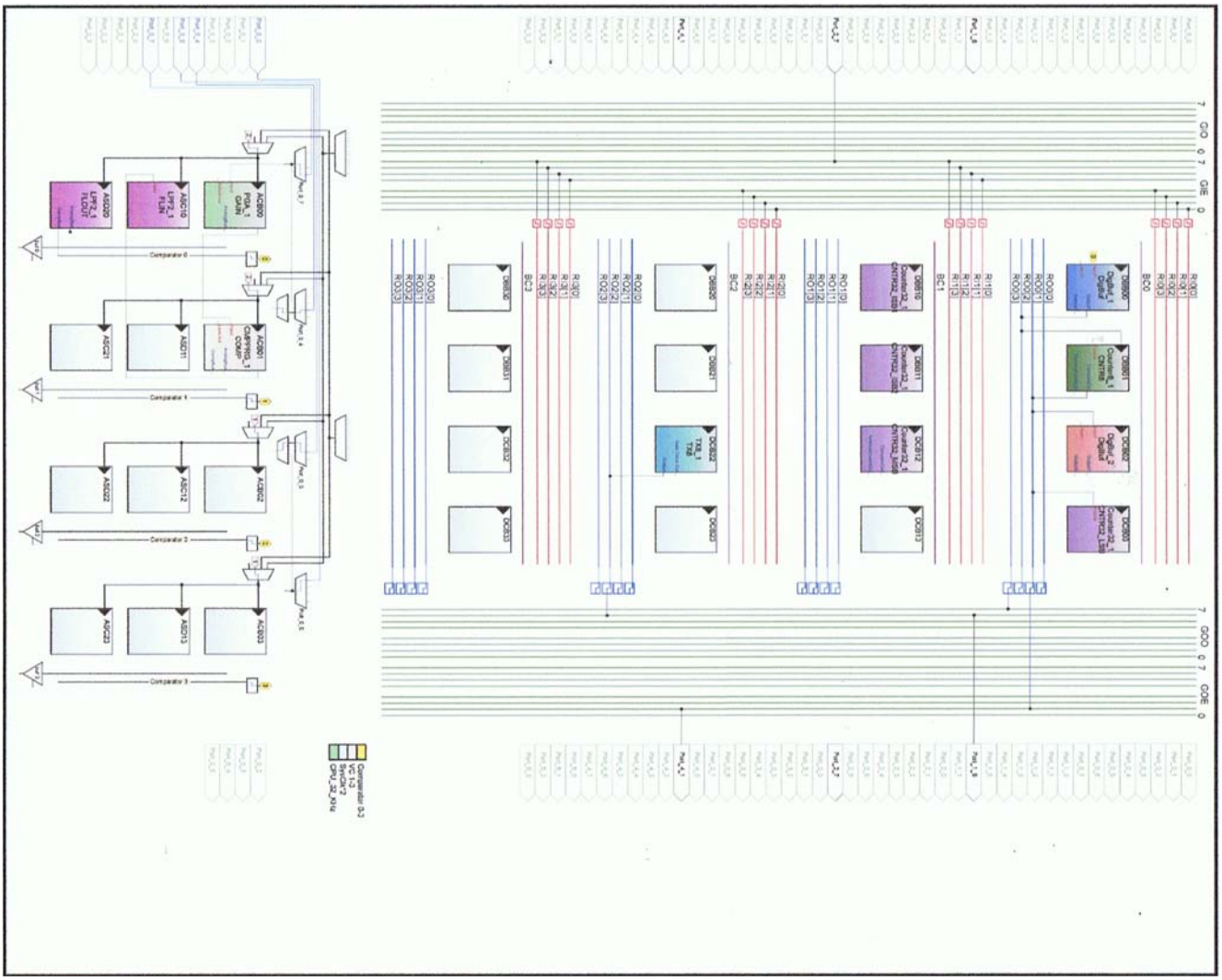
Następnie w celu wyeliminowania zakłóceń zastosowano filtr dolnoprzepustowy (rys. 4.) typu Butterworth o Fg=10Hz. W ten sposób otrzymano kompletnie uformowany sygnał pomiarowy, który poprzez moduł bufora cyfrowego przekazywany jest na tor pomiaru częstotliwości.



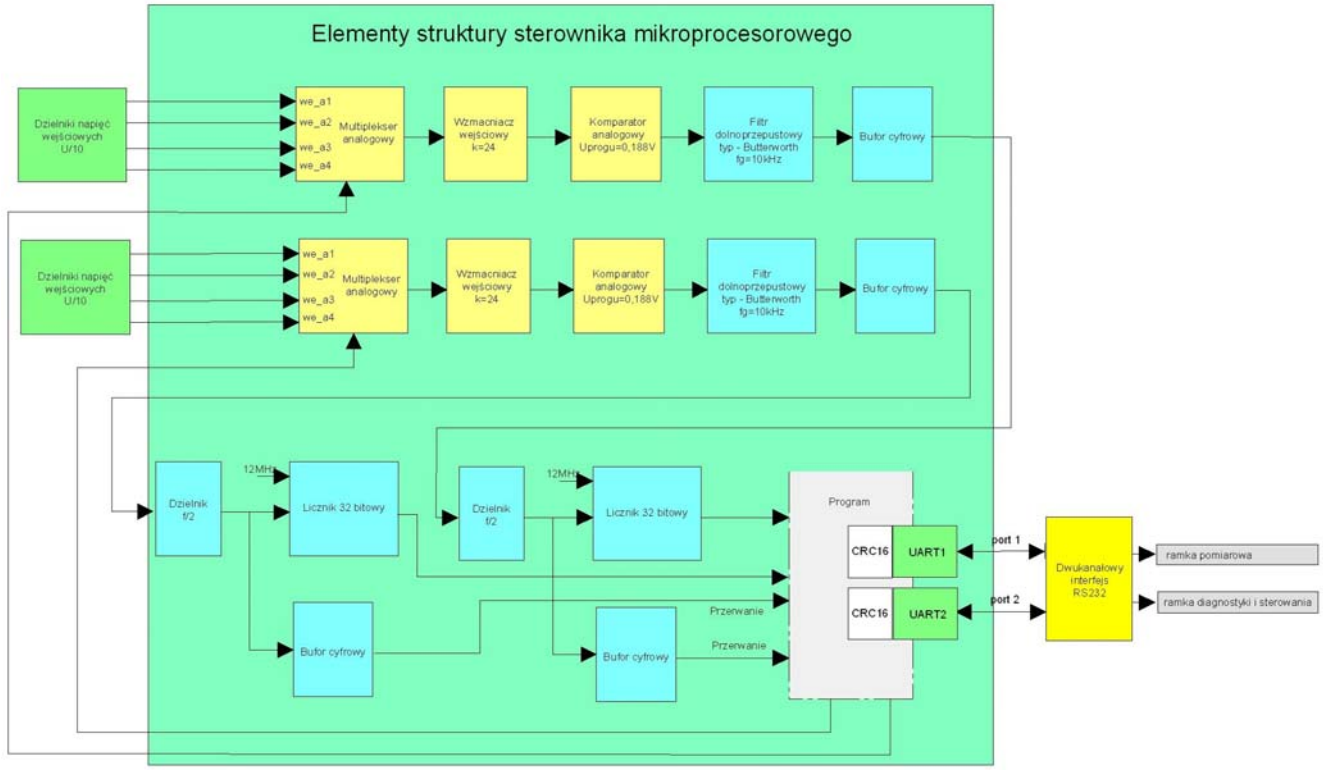
Rys. 4. Schemat blokowy modułu filtru



Rys. 5. Schemat pomiarowy w konfiguracji podstawowej.



Rys. 6. Mapa obsadzeń zasobów sprzętowych mikrokontrolera

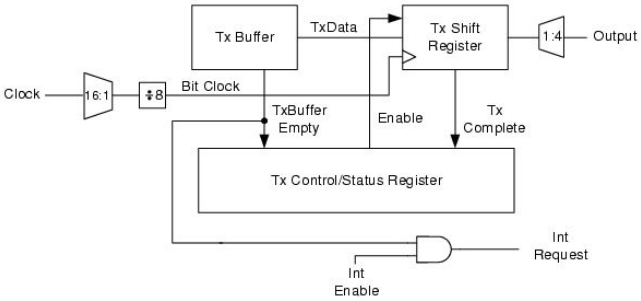


Rys. 7. Schemat pomiarowy w konfiguracji zdublowanej.

Jest on realizowany metodą zliczania impulsów częstotliwości wzorcowej w czasie trwania bramki pomiarowej. Ze względu na możliwość pojawienia się sygnału z czujnika reluktancyjnego o niesymetrycznym wypełnieniu, bramka pomiarowa obejmuje cały okres pomiarowy. Jest to zrealizowane poprzez podział sygnału pomiarowego przez 2. Przez czas trwania bramki pomiarowej następuje zliczanie impulsów o częstotliwości 12 MHz w liczniku 32 bitowym. Dalej programowo zawartość licznika jest odczytywana i przetwarzana na liczbę odpowiadającą częstotliwości. Następnie cykl powtarza się dla kolejnego wejścia multipleksera aż do wyczerpania liczby wejść. Liczbę obsługiwanych wejść analogowych można zwiększyć do 12 poprzez zastosowanie zespołu multiplekserów. To rozwiązanie posiada jednak wadę polegającą na spowolnieniu pomiaru pojedynczego wejścia. Można temu zaradzić poprzez zdublowanie toru pomiarowego sterowanego ze wspólnego programu głównego. Pozwalają na to zasoby sprzętowe mikrokontrolera przedstawione na mapie obsadzeń na rys. 6. Po zrealizowaniu tego zadania otrzymamy prawie pełne wykorzystanie zasobów cyfrowych i w 80% analogowych. Schemat blokowy układu pomiarowego konfiguracji zdublowanej przedstawiono na rys. 7.

3. Funkcje programu

Program główny ustala parametry i steruje zasobami sprzętowymi mikrokontrolera, odczytuje stany liczników pomiarowych, oblicza wartości liczbowe częstotliwości, formuje wyjściową ramkę transmisji RS232, oblicza sumę kontrolną CRC16 i za pośrednictwem sprzętowych modułów transmisji (rys.8.) wysyła ramkę wyjściową na porty mikrokontrolera. Następnie za pośrednictwem zewnętrznych modułów konwersji poziomów RS232 następuje połączenie z zewnętrzną linią transmisyjną.



Rys. 8. Schemat blokowy modułu transmisji RS232 – TX8

Sumę kontrolną wyznaczona jest dla zawartości przesyłanej ramki i umieszczona w ramce po części informacyjnej. Węzeł odbiorczy oblicza sumę kontrolną dla odebranej ramki i porównuje jej wartość z wartością otrzymaną. Niezgodność sum świadczy o wystąpieniu błędu.
Obliczanie sumy CRC 16 odbywa się według następującego algorytmu:

1. nadanie CRC wartości początkowej równej FFFFH,

2. pobranie bajtu komunikatu i wykonanie operacji XOR z młodszym bajtem CRC
3. przesunięcie CRC w prawo o jedną pozycję połączone z wpisaniem 0 na bit najbardziej znaczący,
4. jeżeli wysunięty bit był równy 1, wykonanie operacji XOR CRC ze stałą A001H,
5. ośmiokrotne powtórzenie sekwencji kroków 3...4, co odpowiada przetworzeniu całego bajtu,
6. powtórzenie sekwencji kroków 2...5 dla kolejnych bajtów ramki. Kontynuacja tego procesu do przetworzenia całego komunikatu.

Po wykonaniu wymienionych operacji CRC zawiera wymaganą wartość.
Do zrealizowania tego algorytmu służy następująca procedura:

```
void OblCRC(BYTE zn)
{
        CR = CR^zn;
        for (j=0;j<8;j ++)
        {
                if ((CR & 0x01)==1)
                {CR =(CR>>1)^POLYNOMIAL;}
                else
                {CR = CR>>1;}
        }
}
```

Zmienna POLYNOMIAL może zawierać różne wartości w zależności od typu CRC:

1. A001h dla CRC wg Modbus,
2. 8810h dla CRC CCITT,
3. C002 dla CRC16.

Procedurę weryfikacji zrealizowano w oparciu następującą procedurę:

```
// sprawdzenie zgodności CRC
                CRa =CR;
                cyfCRC1 = CharToByte (cyfCRC1);
                cyfCRC2 = CharToByte (cyfCRC2);
                cyfCRC3 = CharToByte (cyfCRC3);
                cyfCRC4 = CharToByte (cyfCRC4);
                cyfCRC4a =CRa&0x000F;
                cyfCRC3a =(CRa&0x00F0)>>4;
                cyfCRC2a =(CRa&0x0F00)>>8;
                cyfCRC1a =(CRa&0xF000)>>12;

        if((cyfCRC1!=cyfCRC1a)||(cyfCRC2!=cyfCR
C2a)||(cyfCRC3!=cyfCRC3a)||(cyfCRC4!=cyfCRC4a)
)
{goto end;}// CRC odebrane i obliczone niezgodne
// CRC odebrane i obliczone zgodne
```

4. WNIOSKI

Jest to jedyny z nielicznych mikrokontrolerów, który może mieć cztery układy UART (realizujące dwukierunkową, asynchroniczną transmisję szeregową). Niewykorzystany drugi port RS232 można użyć do pracy w sieci sterowników. Nieużywany kanał odbiorczy pierwszego portu RS232 może posłużyć do sterowania przez jednostkę centralną. Wobec łatwości rekonfigurowania programu w trakcie jego działania można np. na polecenie jednostki centralnej zacząć przesyłać zamiast częstotliwości torów pomiarowych próbki amplitudy celem zarejestrowania kształtu przebiegu wejściowego.
Można również dołączać do ramki wyjściowej wartość temperatury procesora z wykorzystaniem jego zasobu sprzętowego do pomiaru temperatury. Ma to znaczenie w przypadku oddalenia od jednostki centralnej i pracy w innych warunkach otoczenia.
W pracy przedstawiono uniwersalny program w języku C do pomiaru częstotliwości w 4 kanałach (z wykorzystaniem przerwań).

LITERATURA

[1] Bocian S., Iwanowski J.: Uniwersalne układy wejść – wyjść do zastosowania w pojazdach szynowych z wykorzystaniem mikrosystemu cyfrowego PSoC CY29466 firmy CYPRESS. Pojazdy Szynowe, Nr 3/2008.

[2] Bocian S., Iwanowski J.: Układ przetwarzania sygnałów analogowych z reluktacyjnych czujników prędkości w pojazdach szynowych. XIII MIĘDZYNARODOWA KONFERENCJA KOMPUTEROWE SYSTEMY WSPOMAGANIA NAUKI, PRZEMYSŁU I TRANSPORTU TRANSCOMP 2009, 30.XI-3.XII 2009, Zakopane.

Program do pomiaru częstotliwości w czterech kanałach

```
//-------------------------------------------------------------------------
// C main line
//  Program pomiaru czestotliwości w 4 kanalach (z wykorzystaniem przerwan)
//  i odczyt 6 wejsc cyfrowych
//  kodowanie wejsc cyfrowych:
//  nr wejscia        weC1 weC2 weC3 weC4        weC5 weC6 0 0
//  nazwa zmiennej          bajt1                   bajt2
//  kolejnosc transmisji: P0(7), P0(5), P0(3), P0(1), bajt1, bajt2
//  predkosc transmisji RS232 = 38400
//-------------------------------------------------------------------------

#include <m8c.h>      // part specific constants and macros
#include "PSoCAPI.h"    // PSoC API definitions for all User Modules

#include "driverdecl.h"
#include "CMXSystem.h"
#include "CMXSystemExtern.h"
#include "TransferFunction.h"

#include "cmx.h"
#include "ProjectProperties.h"
#include "Custom.h"
#include "stdlib.h"

// Channel includes
extern BYTE  Przerw1;
extern void Wyslodcz(void);


    int licznik = 0;
    int flagaStart=0;

    char temp;
    char *string;

    DWORD * pdwCount;

    BYTE by = 0;

void main()
{
    // Initialize Project

//.................inicjalizacja zasobów poczatek
    CMPPRG_1_Start(CMPPRG_1_MEDPOWER);      //inicjalizacja komparatora o programowanym progu
    DigBuf_1_Start();                       //inicjalizacja bufora cyfrowego DigBuf_1
    DigBuf_2_Start();                       //inicjalizacja bufora cyfrowego DigBuf_2
    DigBuf_2_EnableInt();                   //inicjalizacja przerwan bufora cyfrowego DigBuf_2
    Counter32_1_Start();                    //inicjalizacja pomiarowego licznika 32-bitowego
    TX8_1_Start(0x00);                      //inicjalizacja 8-bitowej transmisji
    PGA_1_Start(PGA_1_MEDPOWER);            //inicjalizacja wzmacniacza o programowanym wzmocnieniu
    AMUX4_1_Start();                        //inicjalizacja multipleksera 4-kanalowego
    AMUX4_1_InputSelect(AMUX4_1_PORT0_7);   // wtepne zaadresowanie portu multipleksera
    LPF2_1_Start(LPF2_1_MEDPOWER);          //inicjalizacja filtru dolnoprzepustowego
    Counter8_2_EnableInt();                 //inicjalizacja przerwan licznika 8-bitowego
    Counter8_2_Start();                     //inicjalizacja  8-bitowego licznika przerwan
//.................inicjalizacja zasobów koniec


    M8C_EnableGInt;
    I2C_CFG &= 0xFC;
    SystemTimer_Start();
    SystemTimer_SetInterval(SystemTimer_64_HZ);
    SystemTimer_EnableInt();
```

```
//.................inicjalizacja zmiennych poczatek
    SystemVars.ReadOnlyVars.pse_Output1 = 2;
    SystemVars.ReadOnlyVars.pse_gen1 = 0;
    SystemVars.ReadOnlyVars.pse_stangen1_state = ID_pse_stangen1_state_State0;
    SystemVars.ReadOnlyVars.pse_stangen1_transition = ID_pse_stangen1_transition_NoTransition;
    SystemVars.ReadOnlyVars.pse_weC1 = 0;
    SystemVars.ReadOnlyVars.pse_weC2 = 0;
    SystemVars.ReadOnlyVars.pse_weC3 = 0;
    SystemVars.ReadOnlyVars.pse_weC4 = 0;
    SystemVars.ReadOnlyVars.pse_weC5 = 0;
    SystemVars.ReadOnlyVars.pse_weC6 = 0;

    // Driver instantiations
CMX_DIO_Instantiate(&pse_weC4);
CMX_BLINKINGLED_Instantiate(&pse_Output1);
CMX_BLINKINGLED_SetValue(&pse_Output1, (BYTE)SystemVars.ReadOnlyVars.pse_Output1);
CMX_DIO_Instantiate(&pse_weC3);
CMX_DIO_Instantiate(&pse_weC2);
CMX_DIO_Instantiate(&pse_weC1);
CMX_DIO_Instantiate(&pse_weC6);
CMX_DIO_Instantiate(&pse_weC5);
CMX_INTERVAL_Instantiate(&pse_gen1);

    // Custom initization code.
    CustomInit();
    // End Initialize Project

    while(1)
    {
        // Sync loop sample rate
#if ( SAMPLE_DIVIDER )
        SystemTimer_SyncWait(SAMPLE_DIVIDER, SystemTimer_WAIT_RELOAD);
#endif
end1:       // aktualizacja wejsc
        SystemVars.ReadOnlyVars.pse_gen1 = CMX_INTERVAL_GetValue(&pse_gen1);
        SystemVars.ReadOnlyVars.pse_weC1 = CMX_DIO_GetValue(&pse_weC1);
        SystemVars.ReadOnlyVars.pse_weC2 = CMX_DIO_GetValue(&pse_weC2);
        SystemVars.ReadOnlyVars.pse_weC3 = CMX_DIO_GetValue(&pse_weC3);
        SystemVars.ReadOnlyVars.pse_weC4 = CMX_DIO_GetValue(&pse_weC4);
        SystemVars.ReadOnlyVars.pse_weC5 = CMX_DIO_GetValue(&pse_weC5);
        SystemVars.ReadOnlyVars.pse_weC6 = CMX_DIO_GetValue(&pse_weC6);

        // Custom Post Input function
        CustomPostInputUpdate();

        // run transfer function and update output variables
        TransferFunction();

        // CustomPreOutputUpdate();
        CustomPreOutputUpdate();

        // set outputs
        //sterowanie wyjsciem do sterowania diody LED
        CMX_BLINKINGLED_SetValue(&pse_Output1, (BYTE)SystemVars.ReadOnlyVars.pse_Output1);
        if(SystemVars.ReadOnlyVars.pse_stangen1_transition == pse_stangen1_transition__tr1)
            {

            if ( Przerw1==0x01)
                {
                if (licznik==0x00)      //odczyt licznika portu nr7
                    {
                    Wyslodcz();
                    //zaadresowanie portu nr5 multipleksera
                    AMUX4_1_InputSelect(AMUX4_1_PORT0_5);
                    AMUX4_1_Start();
                    licznik++;
```

```
                goto end1;
                }
            if (licznik==0x01)      //odczyt licznika portu nr5
                {
                Wyslodcz();
                //zaadresowanie portu nr3 multipleksera
                AMUX4_1_InputSelect(AMUX4_1_PORT0_3);
                AMUX4_1_Start();
                licznik++;
                goto end1;
                }
            if (licznik==0x02)      //odczyt licznika portu nr3
                {
                Wyslodcz();
                //zaadresowanie portu nr1 multipleksera
                AMUX4_1_InputSelect(AMUX4_1_PORT0_1);
                AMUX4_1_Start();
                licznik++;
                goto end1;
                }
            if (licznik==0x03)      //odczyt licznika portu nr1
                {
                Wyslodcz();
                //zaadresowanie portu nr7 multipleksera
                AMUX4_1_InputSelect(AMUX4_1_PORT0_7);
                AMUX4_1_Start();
                licznik=0x00;
                //formowanie i wyslanie 1-go bajtu jako znaku
                    by = (SystemVars.ReadOnlyVars.pse_weC1<<3)|(SystemVars.ReadOnlyVars.pse_weC2<<2)
                        |(SystemVars.ReadOnlyVars.pse_weC3<<1)|SystemVars.ReadOnlyVars.pse_weC4;
                    //konwersja na znak
                    if((by>=0x00)&&(by<=0x09)){(by=by+0x30);}
                    if((by>=0x0A)&&(by<=0x0F)){(by=by+0x37);}
                    //wyslanie 1-go znaku na port RS232 TX8_1
                    TX8_1_PutChar(by);
                    //wyslanie spacji jako znaku separujacego na port RS232 TX8_1
                    TX8_1_PutChar(0x20);
                //formowanie i wyslanie 2-go bajtu jako znaku
                    by = 0x00;
                    by = (SystemVars.ReadOnlyVars.pse_weC5<<3)|(SystemVars.ReadOnlyVars.pse_weC6<<2);
                    //konwersja na znak
                    if((by>=0x00)&&(by<=0x09)){(by=by+0x30);}
                    if((by>=0x0A)&&(by<=0x0F)){(by=by+0x37);}
                    //wyslanie 2-go znaku na port RS232 TX8_1
                    TX8_1_PutChar(by);
                    by = 0;
                //wyslanie  znakow konca transmisji na port RS232 TX8_1
                TX8_1_PutCRLF();
                goto end1;
                }
            }
        }
}}

//odczyt licznika, konwersja wyniku, wyslanie winiku do portu RS232'odblokowanie przerwan
void Wyslodcz(void)
{
        //odczyt zawartosci licznika
        Counter32_1_ReadCounter(pdwCount);
        //zamiana wartości zliczonej w liczniku Counter32_1
        //na uzupelnienie dwojkowe a nastepnie czestotliwosc
        *pdwCount=(1214574898/(0xFFFFFFFF-*pdwCount));
        //konwersja czestotliwosci na string
        ultoa(string, *pdwCount,10);
        //wyslanie wartosci na port RS232 TX8_1
        TX8_1_PutString(string);
        TX8_1_PutChar(0x20);
        //inicjalizacja licznikow
        Counter8_2_WritePeriod(0x01);
        Counter8_2_Start();
        Counter32_1_WritePeriod(0xFFFFFFFF);
        Counter32_1_Start();
        DigBuf_1_Start();
        //odblokowanie przerwan
        Przerw1=0x00;

}
```