W bascomowym „kąciku" będziemy się starać w miarę przystępnie przedstawiać rozwiązania problemów napotykanych przez naszych Czytelników podczas pisania programów w Bascomie. Rubryka ta powstała z myślą o rozwiązywaniu problemów, jakie najczęściej napotykają programiści, zatem zachęcamy wszystkich Czytelników do zgłaszania problemów, na jakie się natknęli podczas tworzenia własnych programów.

Obsługa RS485 w Bascomie, część 1

Interfejs RS485 cieszy się coraz większą popularnością, którą zawdzięcza głównie możliwości przesyłania danych na dość duże odległości z dość dużymi prędkościami. W artykule przedstawiamy przykłady w Bascomie ilustrujące wymianę danych za pomocą tego interfejsu i specjalnie opracowanego protokołu.

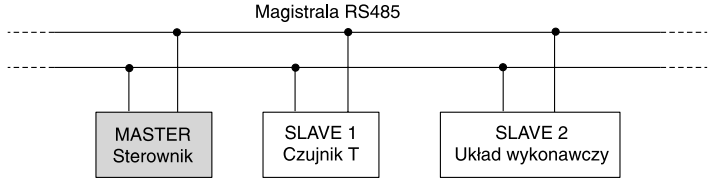
W przypadku interfejsu RS232 przesyłanie danych jest możliwe jedynie na odległość kilkunastu metrów, ale RS485 umożliwia przesyłanie danych aż na odległość do 1200 metrów! Ważne jest także to, że do jednej magistrali RS485 można dołączyć wiele (do 32) urządzeń nadających i odbierających, a nie jak w przypadku RS232 tylko jedno urządzenie. Wiele systemów opartych o RS485 używa architektury *Master-Slave*. Urządzenia w tak wykonanym systemie mają unikalne identyfikujące je adresy. Urządzenia *Slave* będą reagowały jedynie na dane zaadresowane do nich, wysyłane przez *Mastera*, który okresowo komunikuje się z urządzeniami *Slave*. Układy *Slave* nigdy same nie inicjują wymiany danych.

Interfejs RS485 występuje w dwóch wersjach: z pojedynczą oraz podwójną linią transmisyjną. W przypadku pojedynczej linii, komunikacja w tym samym czasie może być przeprowadzana tylko w jednym kierunku – w tym przypadku wszystkie urządzenia dołączone do magistrali muszą posiadać możliwość zmiany kierunku transmisji (nadawanie lub odbiór). W przypadku podwójnej linii komunikacyjnej jedna para linii używana jest do nadawania, a druga do odbierania danych. Interfejs RS485 używa do komunikacji linii różnicowych, co daje dużą odporność magistrali na zakłócenia.

Po krótkim przedstawieniu kilku podstawowych informacji o RS485, czas przejść do praktyki. Jako przykład przedstawiona zostanie budowa prostego systemu składającego się z jednego *Mastera* oraz dwóch układów *Slave* komunikujących się tylko po jednej parze przewodów – czyli komunikacja w danym czasie odbywać się może tylko w jednym kierunku, co dla tego systemu jest wystarczające i nie sprawia kłopotów. W danym czasie wszystkie układy dołączone do systemu są w trybie odbioru, natomiast jako pierwszy może nadawać jedynie *Master*.

Na **rys. 1** przedstawiono schemat blokowy przykładowego systemu. Układ *Slave 1* działa jako zdalny czujnik temperatury, natomiast układ *Slave 2* jest układem wykonawczym, który może sterować dwoma przekaźnikami oraz prędkością obrotową dołączonego do niego wentylatora. Całością zarządza *Master*, który może zdalnie sterować prędkością obrotową wentylatora oraz przekaźników dołączonych do *Slave 2* oraz może odczytywać temperaturę z układu *Slave 1*.

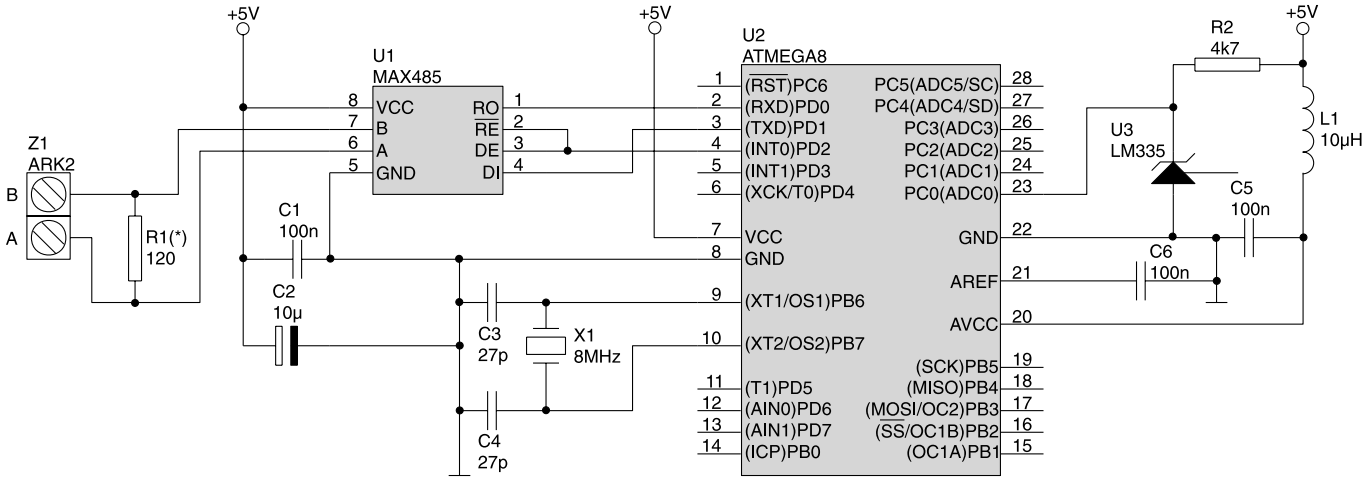
Oprogramowanie sterujące *Mastera* napisano w taki sposób, by z wykorzystaniem układów *Slave 1* oraz *Slave 2* realizował funkcję prostego termostatu. W zależności od odczytanej z układu *Slave 1* temperatury, steruje stanem jednego z przekaźników dołączonych do układu *Slave 2*. Tak więc *Master* jest sterownikiem systemu, układ *Slave 1* – czujnikiem, a *Slave 2* – układem wykonawczym. W tym systemie podanie poprawnej komendy dla danego układu *Slave* może aktywować go do nadawania, co eliminuje możliwość powstania konfliktów w przypadku jednoczesnego wysyłania danych przez kilka układów dołączonych do magistrali RS485. *Master* w danym momencie może aktywować do nadawania tylko jeden układ podrzędny. Komunikacja *Mastera* z układami *Slave* odbywa się w określonym porządku. Aby *Master* mógł skomunikować się z danym układem *Slave*, musi wysłać znak początku transmisji, adres układu *Slave* (0 do 99, choć do jednej magistrali można podłączyć do 32 urządzeń) oraz przeznaczone dla niego dane. Każdy układ *Slave* posiada inny adres i jeśli wysyłany adres nie zgadza się z adresem danego układu *Slave*, to będzie on ignorował kolejne dane wysyłane przez *Mastera*. W opisywanym systemie z układu *Slave 1* można tylko odczytywać dane, natomiast w przypadku układu *Slave 2* można zarówno z niego odczytywać, jak i zapisywać do niego dane. Na wyświetlaczu układu *Master* wyświetlane są informa-



Rys. 1. Schemat blokowy przykładowego systemu transmisyjnego z magistralą RS485

Tajemniczy RS485

Normy RS485 opisują wyłącznie elektryczny standard połączenia, a nie protokół lub złącza. W związku z tym, wystarczy dołączyć do klasycznego UART-a układy interfejsowe RS485, żeby uzyskać duży zasięg i wysoką szybkość transmisji.

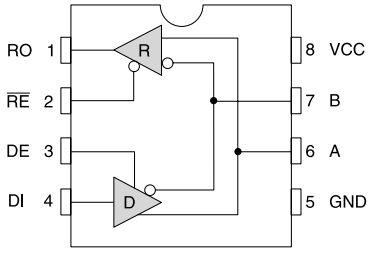


Rys. 2. Schemat ideowy układu *Slave 1* (czujnik temperatury)

cje o stanie układów *Slave 1* i *Slave 2*. Transmisję danych w systemie oparto o standardowe znaki ASCII, dzięki czemu obsługa systemu jest także możliwa z poziomu komputerowego terminala, przy czym należy do komputera dołączyć odpowiedni konwerter RS232/RS485. Szybkość transmisji układów w całym przedstawionym systemie wynosi 9600 bd, a ramka transmisyjna ma następujące parametry: 1 bit stopu, brak bitu parzystości oraz 8 bitów danych. Opis systemu rozpocznę od najprostszego urządzenia, którym jest czujnik temperatury (*Slave 1*). Na **rys. 2** przedstawiono schemat elektryczny układu *Slave 1* (czujnik temperatury).

Jako czujnik temperatury zastosowano układ LM335 (można także zastosować dowolny inny czujnik), z którego temperatura mierzona jest przez wewnętrzny przetwornik A/C mikrokontrolera ATmega8. Do przetworzenia sygnałów TTL na sygnały zgodne z interfejsem RS485 służy układ MAX485, którego schemat blokowy przedstawiono na **rys. 3**.

W RS485 stan logiczny jest określany na podstawie napięcia różnicowego pomiędzy liniami A i B magistrali. Jeśli napięcie na linii A jest większe od napięcia na linii B o co



Rys. 3. Schemat blokowy konwertera MAX485

najmniej 200 mV, to jest to stan logicznej „1”, a jeżeli jest odwrotnie, to jest to stan zera logicznego. Konwerter posiada dodatkowo linie sygnałowe przełączające go pomiędzy nadawaniem a odbiorem. Konwerter będzie przełączany w tryb nadawania tylko, gdy dany układ *Slave* lub *Master* będzie wysyłać dane. W spoczynku konwertery w przedstawionym systemie pracują jako odbiorniki, by nie była zajęta magistrala RS485. Liniami przełączającymi konwerter na nadawanie lub odbiór są linie !RE oraz DE, które można połączyć razem i sterować jednym sygnałem. Stan niski na tych liniach będzie powodował przełączenie konwertera MAX485 w tryb odbioru, a stan wysoki przełączenie go w tryb nadawania. Kierunkiem transmisji konwertera MAX485 steruje mikrokontroler za pomocą linii PD2. Układy interfejsowe stosowane w systemie z magistralą RS485 powinny posiadać swoje linie A dołączone do linii A magistrali, a linie B dołączone do linii magistrali B. Odwrotne podłączenie któregoś z układów do magistrali RS485 uniemożliwi pracę tego układu. Konwerter RS485 dołączono do linii UART-u mikrokontrolera, tak więc szeregowa wymiana danych na poziomie bitowym jest identyczna jak w przypadku RS232 (występują bity startu, stopu itp.). Elementy L1, C5 i C6 odpowiedzialne są za filtrację napięć zasilających przetwornik A/C mikrokontrolera. Dodatkowy rezystor R1 odpowiada za dopasowanie linii magistrali i powinien być zamontowany tylko w jednym układzie dołączonym do magistrali RS485 i to w układzie najdalej oddalonym od nadajnika (w tym przypadku nadajnikiem jest *Master*). Opisywany układ czujnika temperatury (*Slave 1*) reaguje tylko na jedną komendę, która jest prośbą o wysłanie przez *Slave 1* zmierzonej temperatury. Adres układu czujnika został ustalony na wartość „5”. Aby układ czujnika wysłał zmierzoną temperaturę, należy do niego wysłać komendę:

```
BS 5 r enter
```

gdzie:

- BS – to znak „BACK SPACE” (o kodzie ASCII „8”),
- 5 – adres układu *Slave 1* (adres może być z zakresu od 0 do 99),
- r – „read” – wskazuje, że dane będą odczytywane z układu *Slave 1*,
- enter – to odpowiednik znaku CR w ASCII kod 13.

Jak widać, komenda wysyłana do *Slave 1* z prośbą wysłania temperatury jest prosta. Należy ją rozpocząć od wysłania znaku BS, potem podać adres układu *Slave 1*, znak „r” i potwierdzić *enterem*. Jeżeli wszystko jest w porządku, układ *Slave 1* powinien wysłać wartość zmierzonej temperatury (potwierdzaną znakiem CR), którą mierzy z dokładnością ±1°C. W przypadku temperatury 25°C, układ *Slave 1* wyśle przez magistralę RS485 wartość 25. Gdy zawarty w komendzie adres jest niezgodny z adresem układu *Slave 1*, układ ten w ogóle nie zareaguje, gdyż komenda nie jest przeznaczona dla niego. Program obsługujący czujnik temperatury przedstawiono na **list. 1**.

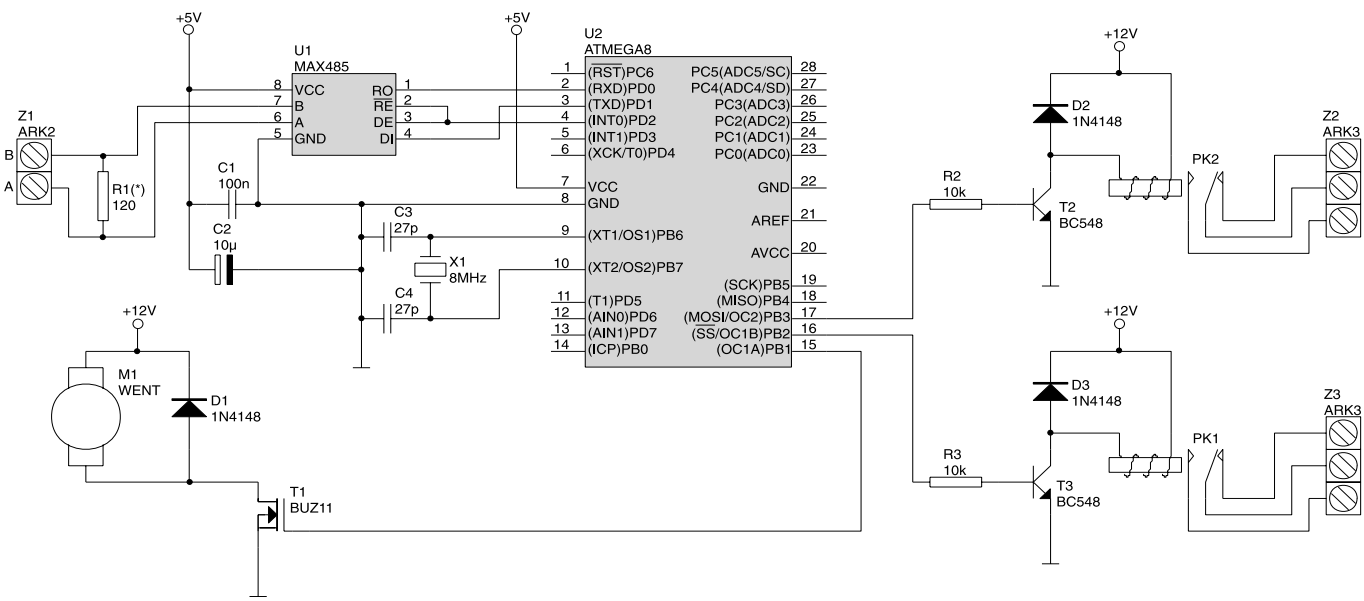
Program sterujący układem *Slave1* jest w miarę prosty, a większą jego część zajmują instrukcję odpowiedzialne za interpretację otrzymywanych komend. Układ ten na dane

List. 1. Program sterujący czujnikiem temperatury (*Slave 1*)

```
'Przykład pierwszego urzadzenia slave sieci RS485.
'Urzadzenie mierzy temerature i na zadanie mastera wysyla ja do niego
'dokladnosc pomiaru temperatury +/- 1 stopien C
'Adres tego urzadzenia wynosi 5

$regfile = "m8def.dat"              'informuje kompilator o pliku dyrektyw wykorzystywanego
                                    'mikrontrolera
$crystal = 8000000                  'informuje kompilator o czestotliwosci rezonatora kwarcowego
$baud = 9600                        'informuje kompilator o predkosci transmisji RS232

Config Pind.2 = Output              'linia pd.2 jako wyjscie
Config Adc = Single , Prescaler = Auto , Reference = Avcc
                                    'konfiguracja wewnetrznego przetwornika

Config Serialin = Buffered , Size = 15
                                    'konfiguracja by interfejs rs232 uzywal przy odbiorze transmisji
                                    'buforowej (bufor o wielkosci 15 znakow)

Const Adr = 5                       'adres ukladu - od 0 do 99

Dim Wart_zm As Word                 'zmienna przechowujaca wartosc zmierzona przez przetwornik A/C
Dim Temp As Integer                 'zmienna ktora przechowuje obliczona temperature
Dim Zn As String * 1                'zmienna przecowujaca odebrany znak z rs232
Dim Adr_s As String * 5             'zmienna w ktorej skladany jest w calosc otrzymany adres
Dim Adres As Byte                   'zmienna ktora przechowuje przekonwertowany
                                    'na dziesietnie otrzymany adres
Dim Il_zn As Byte                   'zmienna liczy ilosc otrzymanych znakow przy skladaniu adresu

Re_de Alias Portd.2                 'przypisanie aliasu linii pd.2, re_de steruje kierunkiem
                                    'transmisji konwertera

Start Adc                           'uruchamia wbudowany przetwornik A/C
Enable Interrupts                   'odblokowanie globalnych przerwan
Reset Re_de                         'przelaczenie konwertera rs485 na odbiornik

Do                                  'poczatek petli programu
  Wart_zm = Getadc(0)               'pomiar napiecia na wejściu PC0 mikrokontrolera
  Temp = Wart_zm - 560              'odjecie od zmierzonej wartosci 560 (przeliczenie kelwinow
                                    'na stopnie)
  Temp = Temp / 2                   'dalsze obliczenie zmierzonej temperatury
                                    '(wartosc dzielona przez 2)
  Zn = Inkey()                      'odczyt znaku z portu rs232
  If Zn = Chr(8) Then               'jesli otrzymany znak to BS (backspace - kod ascii 8) to
    Adr_s = ""                      'czysc zmienna adr_s
    Il_zn = 0                       'oraz zeruj zmienna il_zn
    Do                              'poczatek petli do-loop
      Zn = Inkey()                  'odczyt znaku z portu rs232
      If Zn >= "0" And Zn <= "9" Then     'jesli otrzynany znak jest cyfra (0..9) to
        Incr Il_zn                  'zwieksz o jeden il_zn
        Adr_s = Adr_s + Zn          'dodaj otrzymany znak cyfry do zmiennej adr_s
      Else                          'w przeciwnym razie
        If Zn > Chr(0) Then         'jesli otrzymany znak ma kod ascii wiekszy od 0 to
          Exit Do                   'opusc petle do-loop
        End If
      End If
    Loop                            'koniec wewnetrznej petli do-loop
    If Zn = "r" Then                'jesli otrzymany znak to "r" oraz
      If Il_zn > 0 And Il_zn < 3 Then     'jesli il_zn wynosi 1 lub 2 to
        Adres = Val(adr_s)          'zamien znakowa wartosc otrzymanego adresu
                                    'na postac liczbowa i umiesc ja w adres
        If Adres = Adr Then
          Do                        'poczatek wewnetrznej petli do-loop
            Zn = Inkey()            'odczyt znaku z portu rs232
          Loop Until Zn = Chr(13)   'jesli otrzymany znak to cr - enter (kod ascii 13) to
          Set Re_de                 'przelaczenie konwertera rs485 na nadajnik
          Waitus 200                'czekaj 200 us
          Print Temp                'wyslij zmierzona wartosc temperatury
          Waitms 1                  'czekaj 1ms
          Reset Re_de               'przelaczenie konwertera rs485 na odbiornik
        End If
      End If
    End If
  End If
Loop                                'koniec glownej petli do-loop
End                                 'koniec programu
```

odbierane z magistrali RS485 posiada 15-znakowy bufor, dzięki któremu nie „zgubi" żadnego nadanego przez *Mastera* znaku. Mikrokontroler układu *Slave 1* dane z magistrali RS485 odbiera w przerwaniu, w tle działania programu głównego. Na początku programu linia sterująca kierunkiem transmisji układu *Slave 1* jest ustawiana w stan niski, przez co *Slave1* będzie odbiornikiem. Stała *adr* zawiera adres układu *Slave 1*. Na początku pętli głównej programu dokonywany jest pomiar oraz przeliczenie otrzymanej temperatury. Temperatura przechowywana jest w zmiennej *temp*. Pozostałe instrukcje w pętli odpowiedzialne są za poprawną interpretację otrzymywanej komendy, które dla lepszego zrozumienia dokładniej przedstawię. Na podobnej zasadzie można interpretować także inne otrzymywane komendy.

Jeśli otrzymano z magistrali znak o kodzie ASCII „8" (znak BS), program oczekuje w wewnętrznej pętli *do-loop* na otrzymanie adresu, który będzie liczbą od 0 do 99. Jeśli otrzymany zostanie jakiś nieodpowiedni znak np. litera lub otrzymana wartość adresu będzie większa niż 99, program powróci na początek pętli głównej i będzie oczekiwał ponownie na znak „BS" rozpoczynający nową komendę. Jeśli otrzymany adres zawiera się w podanym zakresie oraz otrzymano po nim znak „r", następuje sprawdzenie otrzymanego adresu z adresem układu *Slave 1*. Po otrzymaniu potwierdzającego znaku komendy „CR" oraz jeśli otrzymany adres jest równy adresowi układu *Slave 1*, następuje przełączenie tego



Rys. 4. Schemat ideowy układu wykonawczego (*Slave 2*)

układu w tryb nadajnika, po czym zostaje wysłana wartość zmierzonej temperatury (zawartość zmiennej temp) potwierdzona znakiem „CR". Po wysłaniu do *Mastera* temperatury i odczekaniu ok. 1 ms na nadanie ostatniego znaku, układ *Slave 1* jest przełączany z powrotem w tryb odbiornika. W programie zmieniając stałą *adr*, można zmienić adres układu *Slave 1*. Trochę bardziej rozbudowany jest układ wykonawczy (*Slave 2*). Interpretuje on zarówno komendy odczytu (jak *Slave 1*), jak i komendy zapisu do niego danych. Na **rys. 4** przedstawiono schemat ideowy układu wykonawczego (*Slave 2*).

Interfejs RS485 tego układu jest identyczny jak układu *Slave 1* – wykorzystany został do jego realizacji także konwerter MAX485. Linie PB2 i PB3 mikrokontrolera sterują poprzez tranzystory dwoma układami wykonawczymi, którymi w tym przypadku są przekaźniki. Poprzez tranzystor T1 sterowany jest wentylator M1. Prędkość wentylatora można regulować za pomocą sygnału PWM, który generowany jest na wyjściu OC1A mikrokontrolera. Układ ten reaguje na dwie komendy odczytu oraz dwie komendy zapisu do niego danych. Adres tego układu *Slave* został ustalony na wartość „8". Komendy dla układu *Slave 2* mają podobną budowę jak dla układu *Slave 1*, choć są trochę bardziej złożone. Aby układ *Slave* zwrócił wartość wypełnienia sygnału sterującego wentylatorem (jego prędkością), należy wysłać do niego następującą komendę:

```
BS 8 r p enter
```

gdzie:

– parametry BS, 8 oraz r mają takie samo znaczenie jak w przypadku układu *Slave 1*,
– p – wskazuje żądanie zwrotu wartości wypełnienia sygnału PWM.

Po wysłaniu tejże komendy, układ *Slave 2* zwróci wartość wypełnienia generowanego sygnału PWM, który ma rozdzielczość 8 bitów. Tak więc zwrócona wartość będzie wartością od 0 do 255. Przy wartości 0 wentylator będzie wyłączony, a przy 255 będzie pracował z maksymalną prędkością. Do odczytu stanu dwóch przekaźników służy komenda:

```
BS 8 r o enter
```

gdzie:

– parametry *BS*, *8* oraz *r* mają takie samo znaczenie jak w przypadku układu *Slave 1*,
– o – wskazuje żądanie zwrotu wartości stanu dwóch przekaźników (PK1 i PK2).

Komenda ta jest podobna do poprzedniej, lecz zwraca stan przekaźników w formie x,x, gdzie stan drugiego przekaźnika jest oddzielony przecinkiem. Tak więc zwrócenie przez *Slave 2* wartości „0,1" będzie oznaczać wyłączony PK1 oraz załączony PK2. Wartość „1" – oznacza przekaźnik załączony, a „0" przekaźnik wyłączony. Jak wspomniałem, układ *Slave 2* obsługuje także komendy umożliwiające zapis do niego parametrów, czyli w tym przypadku wypełnienia przebiegu PWM oraz stanu przekaźników PK1 oraz PK2. Do zapisania wartości wypełnienia PWM służy komenda:

```
BS 8 w pxxx enter
```

gdzie:

– parametry *BS*, *8* mają takie samo znaczenie jak w przypadku układu *Slave 1*,
– w – „write" – wskazuje, że dane będą zapisywane do układu *Slave 2*,
– p – wskazuje, że zapisywane dane będą dotyczyć wypełnienia przebiegu PWM,
– xxx – to podawana wartość wypełnienia PWM z zakresu od 0 do 255.

Aby zapisać do *Slave 2* wypełnienie sygnału PWM równe 127, należy wysłać następującą komendę:

```
BS8p127 (CR - enter)
```

Po wysłaniu takiej wartości, *Slave 2* będzie generował przebieg PWM o wypełnieniu bliskim 50%. Pozostała jeszcze komenda zapisu stanów przekaźników PK1 oraz PK2. Komenda ta jest podobna do komendy zapisu wypełnienia PWM i ma następującą postać:

```
BS 8 w ox,x enter
```

gdzie:

– parametry BS, 8 oraz „w" mają takie samo znaczenie jak w przypadku komendy zapisu wypełnienia sygnału PWM,
– o – wskazuje, że zapisywany będzie stan dwóch przekaźników (PK1 i PK2),
– x,x to podawana wartość stanu odpowiednio przekaźnika PK1 i PK2. Po przecinku podawany jest stan przekaźnika PK2. Wartości „x" mogą być tylko z zakresu „0" lub „1".

Aby zapisać do *Slave 2* załączenie PK1 i wyłączenie PK2, należy wysłać następującą komendę:

```
BS8o1,0 (CR - enter)
```

Po jej wykonaniu przekaźnik PK1 będzie włączony, a PK2 wyłączony. Należy zauważyć, że stan PK2 podawany jest zawsze po przecinku, nie tylko przy zapisie, ale i odczycie stanów przekaźników z układu *Slave 2*. Przy niezgodnym adresie komendy z adresem układu *Slave 2* (ma adres 8) układ ten w ogóle nie będzie reagował na wysyłane przez *Mastera* komendy.

Marcin Wiązania, EP
marcin.wiazania@ep.com.pl