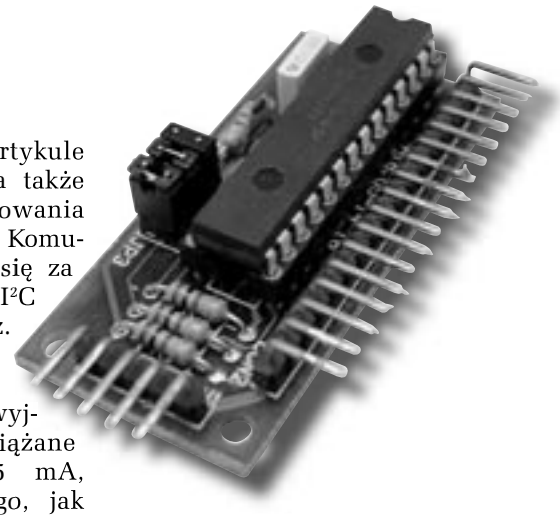
16-bitowy port I²C

AVT-579

Projektując układy z mikrokontrolerami, często natrafiamy na problem niewystarczającej liczby portów, jaką dysponuje upatrzony procesor. Wyjściem z kłopotliwej sytuacji może być zastosowanie dodatkowego ekspandera z interfejsem szeregowym. Czytelnikom dobrze jest znany układ PCF8574, posiadający port 8-bitowy z interfejsem I²C. Tym razem proponujemy rozwiązanie z układem ***MCP23016.***

Rekomendacje: *proponowany układ rozwiązuje w bardzo prosty sposób często spotykany problem braku portów we/wy w systemie mikroprocesorowym, ponadto robi to nie zmniejszając (w większości przypadków) zasobów mikrokontrolera.*



Przedstawiony w artykule układ posiada dwa porty, a także większe możliwości dopasowania pracy portów do wymagań. Komunikacja z układem odbywa się za pośrednictwem magistrali I²C z prędkością 0 do 400kHz. Każda linia portów może pracować jako wejściowa lub wyjściowa. W trybie wyjścia linie mogą być obciążane prądem maksymalnym 25 mA, zarówno dla stanu niskiego, jak i wysokiego. Przy odczycie stanu portów układ MCP23016 może podać prosty lub zanegowany stan występujący na danym wyprowadzeniu układu. Zmiana stanu portu w trybie odczytu generuje przerwanie informujące układ nadrzędny o tym fakcie. Ponadto można wybrać konfigurację portów typu „otwarty kolektor".

Budowa

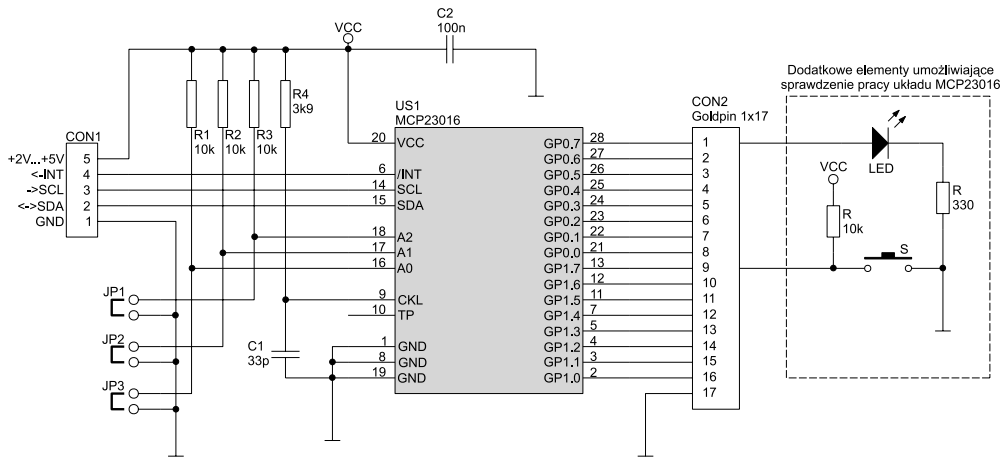
Schemat elektryczny dodatkowego portu jest przedstawiony na **rys. 1**. Do pracy układu MCP23016 wymagany jest zewnętrzny sygnał zegarowy, który jest wytwarzany za pomocą układu RC (rezystor R4, kondensator C1). Podstawowy adres, pod którym zgłasza się układ US1 na magistrali I²C, jest równy 0100[A0][A1][A2]0b, gdzie A0, A1 i A2 są zależne od stanów linii wejściowych układu. Do zmiany adresu służą zworki JP1, JP2 i JP3. Rezystory R1, R2, R3 podciągają wejścia układu US1 do plusa zasilania, w przypadku rozwarcia danej zworki wymuszają jedynkę logiczną. Na złącze CON1 zostały wyprowadzone sygnały magistrali I²C, przerwanie !INT oraz zasilanie, natomiast na złącze CON2 wyprowadzenia portu GP0 i GP1.

Montaż

Układ dodatkowego portu zawiera niewiele elementów, dlatego montaż nie sprawi problemów. Należy go rozpocząć od wlutowania rezystorów, podstawki pod układ US1, następnie należy wlutować kondensatory, a na końcu złącza CON1 i CON2. Podłączenie układu do mikrokontrolera sterującego należy wykonać zgodnie z opisem sygnałów na złączu CON1.

Obsługa układu MCP23016

Ponieważ układ MCP23016 posiada duże możliwości konfiguracji, do ustalenia wymaganych



Rys. 1. Schemat elektryczny poru I²C

parametrów pracy zastosowano 12 wewnętrznych rejestrów, umożliwiających określenie trybu pracy portów, zapisu i odczytu danych. Wykaz wszystkich rejestrów oraz przyporządkowane im adresy znajdują się w **tab. 1**.

Funkcje rejestrów

1. Rejestry **GP0** i **GP1** służą do odczytu danych z portów odpowiednio GP0 i GP1. Poprzez odczyt tych rejestrów można sprawdzić stan linii portów GP0 i GP1. Dodatkowo zapis do tych rejestrów powoduje modyfikację rejestrów (OLAT0, OLAT1), a wpisane dane pojawiają się na wyjściach portów GP0 i GP1.
2. Rejestry **OLAT0** i **OLAT1** służą do zapisu danych do wyjściowych rejestrów (Latch) portów GP0 i GP1. Odczyt z tych rejestrów zwraca wartość tych rejestrów – nie odpowiada aktualnym stanom panującym na portach GP0 i GP1. Zapis danych do rejestrów OLAT0 i OLAT1 powoduje wystawienie wpisanych danych na porty GP0 i GP1.
3. Rejestry **IPOL0** i **IPOL1** umożliwiają odwrócenie polaryzacji danych wejściowych z portów GP0 i GP1. Jeśli bit rejestru jest wyzerowany, to odczyt z odpowiadającego mu wejścia portu GP0 (GP1) będzie odpowiadał jego faktycznemu stanowi. Jeżeli bit rejestru IPOL0 (IPOL1) będzie ustawiony, to odczyt odpowiadającego bitu z portu GP0 (GP1) będzie zanegowany.
4. Rejestry **IODIR0** i **IODIR1** służą do ustawienia portów GP0 i GP1 w tryb wejściowy lub wyjściowy. Jeśli bit rejestru IODIR0 (IODIR1) jest ustawiony, to odpowiadająca mu linia portu GP0 (GP1) jest skonfigurowana jako wejściowa, jeśli bit będzie wyzerowany, to linia portu będzie skonfigurowana do pracy jako wyjście.

List.1 Procedury do obsługi układu MCP23016

```
#define address 0x40 //adres ukladu I2C
char temp1,temp2;
//*************************************************************//
// Procedura zapisu dwoch bajtow od podanego adresu           //
//*************************************************************//
void WriteToMCP(char cmd, char data1, char data2)
{ delay_us(30); //pauza 30us
  i2c_start();  //I2C START
  delay_us(30);
  i2c_write(address);  //wyslij adres ukladu I2C
  delay_us(30);
  i2c_write(cmd);   //wyślij adres komendy
  delay_us(30);
  i2c_write(data1); //wyslij pierwszy bajt danych
  delay_us(30);
  i2c_write(data2); //wyslij drugi bajt danych
  delay_us(30);
  i2c_stop();//I2C STOP
}
//*************************************************************//

//*************************************************************//
// Procedura odczytu dwoch bajtow od podanego adresu          //
//*************************************************************//
void ReadFromMCP(char cmd)
{
  delay_us(30); //pauza 30us
  i2c_start();  //I2C START
  delay_us(30);
  i2c_write(address);  //wyslij adres ukladu I2C
  delay_us(30);
  i2c_write(cmd);   //wyslij adres komendy
  delay_us(30);
  i2c_start();  //ponowny I2C START
  delay_us(30);
  i2c_write(address|1);//wysli adres ukladu I2C START
  delay_us(30); //i przelacz na odczyt
  temp1=i2c_read(); //odbierz pierwszy bajt +ACK(potwierdzenie)
  delay_us(30); //i zapisz do temp1
  temp2=i2c_read(0);   //odbierz drugi bajt bez potwierdzenia
  delay_us(30); //i zapisz do temp1
  i2c_stop();// I2C STOP
}
//*************************************************************//

void main()
{
delay_ms(750);  //pauza potrzebna do inicjalizacji MCP23016
WriteToMCP(0x02,0x00,0x00);   //wyzeruj porty GP0, GP1
WriteToMCP(0x06,0x00,0xFF);   //port GP0 jako wyjscie, GP1 jako wejscie

while(1)
  {
   ReadFromMCP(0x00);  //odczytaj stan portow GP0,GP1
   delay_ms(40);
   WriteToMCP(0x02,temp2,temp1); //i zapisz stan GP0 do GP1, GP1 do GP0
  }

}
//*************************************************************//
```

Tab. 1. Adresy rejestrów układu MCP23016

Adres	Rejestr
00h	GP0
01h	GP1
02h	OLAT0
03h	OLAT1
04h	IPOL0
05h	IPOL1
06h	IODIR0
07h	IODIR1
08h	INTCAP0 (tylko do odczytu}
09h	INTAP1 (tylko do odczytu)
0Ah	IOCON0
0Bh	IOCON1

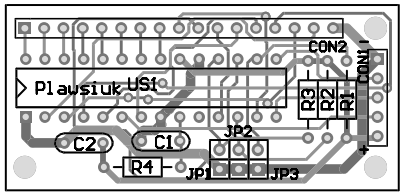
5. Rejestry **INTCAP0** i **INCAP1** służą do wykrywania źródła przerwania wygenerowanego na wyjściu !INT. Odczyt tych rejestrów informuje o tym, które linie portu GP0 (GP1) uległy zmianie i wywołały przerwanie.
6. Rejestr **IOCON0** zawiera tylko jeden znaczący bit (IOCON0.0), służący do kontroli szybkości detekcji zmian na wejściu portów GP0 i GP1. Jeśli bit ten jest wyzerowany, to maksymalny czas wykrycia zmiany na portach wynosi 32 ms. Jeżeli bit zostanie ustawiony, to maksymalny czas wykrycia zmian wyniesie 200 ns.

Procedury

Komunikacja z układem MCP23016 odbywa się według specyfikacji I²C z jednym ograniczeniem – po każdej komendzie należy odczekać około 30 µs, gdyż po odebraniu komendy układ interpretuje ją i wysłanie kolejnych danych przed tym czasem może spowodować błędną pracę układu. Procedury komunikacji z układem MCP23016 przedstawione są na **list. 1**. Wszystkie rejestry grupowane są w dwa bajty, z których pierwszy zawsze dotyczy portu GP0, drugi portu GP1. Funkcja o nazwie

`WriteToMCP(char cmd, char data1, char data2)`

umożliwia zapis do układu dwóch bajtów danych, począwszy od adresu podanego jako parametr. Funkcja `ReadFromMCP(char cmd)` odczytuje natomiast dwa bajty danych z układu, począwszy od podanego adresu i zapisuje je w zmiennych `temp1` i `temp2`.



Rys. 2. Rozmieszczenie elementów na płytce drukowanej

Do sprawdzenia pracy układu służą polecenia zawarte w funkcji `main()`, w której dla przedstawionego przypadku odczytywany jest stan portu GP1 i zapisywany do portu GP0, do sprawdzenia poprawności działania można zastosować diodę świecącą i mikrowyłącznik, przedstawione na rys. 1. Po włączeniu zasilania należy odczekać około 750 ms, aż wewnętrzny układ zerujący układu MCP23016 uruchomi prace oscylatora, dopiero po tym czasie można wysyłać komendy do układu. Jako pierwsze zostanie wydane polecenie wyzerowania portów. Jest to zrealizowane przez wpisanie do rejestrów OLAT0 i OLAT1 wartości 00h. Wpis ten nie spowoduje zmiany stanów na portach GP0 i GP1, gdyż po włączeniu zasilania są one ustawione jako wejścia. Następna komenda ustala dopiero tryb pracy portów, i tak do rejestru IODIR0 zostaje wpisana wartość 00h (cały port GP0 pracuje jako wyjście), a do rejestru IODIR1 wartość FFh (cały port GP1 pracuje jako wejście). Po skonfigurowaniu portów następuje cykliczne odczytywanie stanu portu GP1 i zapis tego stanu do portu GP0. Zarówno odczyt, jak i zapis wykonywany jest na rejestrach obu portów, jednak odczyt stanu portu GP0 jest ignorowany, natomiast zapis danych do portu GP1 nie powoduje na nim żadnych zmian, gdyż znajduje się on w trybie wejściowym. Przedstawione procedury można zmodyfikować według potrzeb, na przykład wykonując jednocześnie operacje tylko na jednym rejestrze. W tym celu zamiast wysyłać i odbierać jednocześnie dane dotyczące obydwu portów, można wysłać tylko adres rejestru i bajt danych, jaki ma być do niego wpisany. W ten sposób skróci się czas przesyłania danych w przypadku, gdy modyfikacji wymaga rejestr tylko jednego portu.

WYKAZ ELEMENTÓW

Rezystory
R1...R3: 10kΩ
R4: 3,9kΩ
Kondensatory
C1: 33pF
C2: 100nF
Półprzewodniki
US1: MCP23016
Inne
JP1...JP3: goldpin 1x2 + zworka
CON1: goldpin 1x5
CON2: goldpin 1x17
Podstawka DIP28 300mils

Krzysztof Pławsiuk, EP
krzysztof.plawsiuk@ep.com.pl

Wzory płytek drukowanych w formacie PDF są dostępne w Internecie pod adresem: ***pcb.ep.com.pl*** *oraz na płycie CD-EP6/2004B w katalogu* ***PCB****.*