Sterowanie drukarkami za pomocą mikrokontrolerów część 2

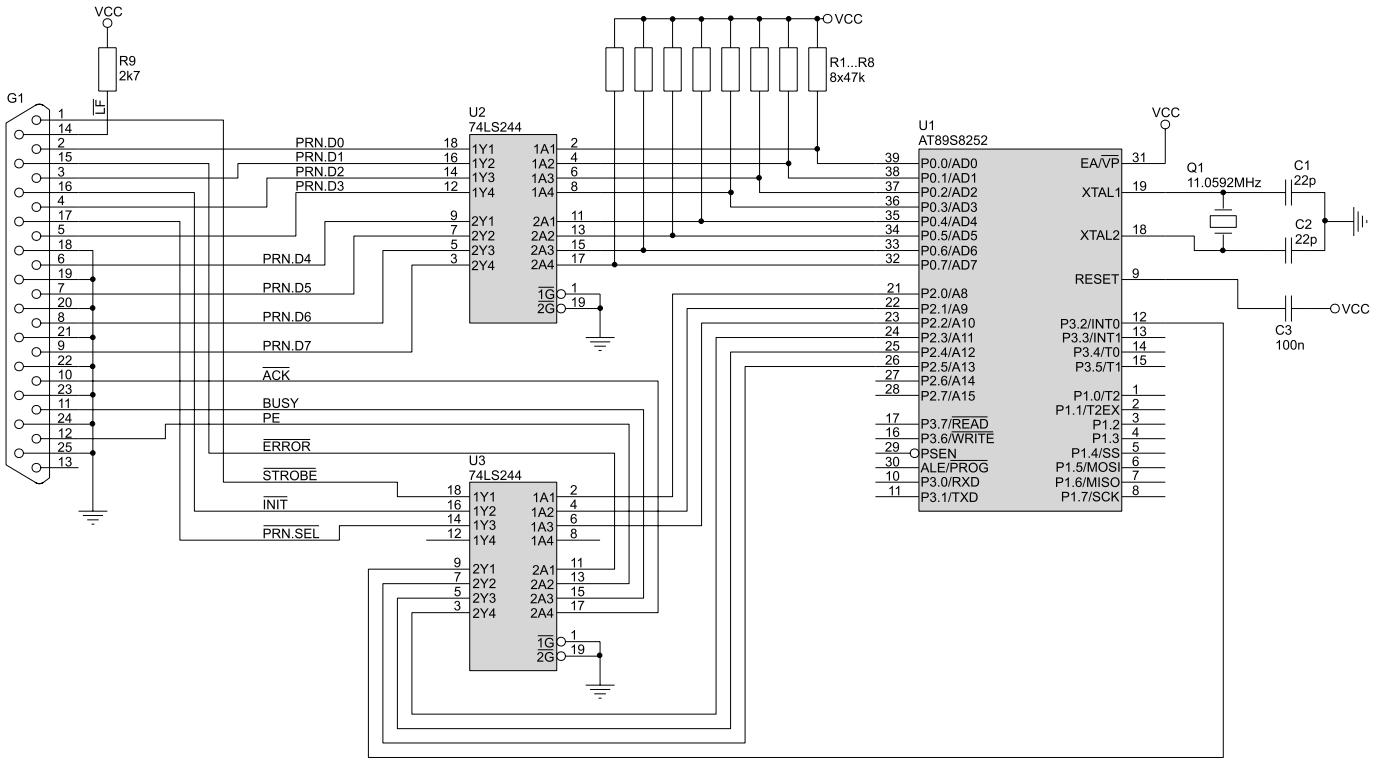
W drugiej części artykułu przedstawiamy przykłady prostych aplikacji, ilustrujących sposoby obsługi drukarek za pomocą mikrokontrolerów. Opis został wzbogacony przykładowymi programami dla mikrokontrolerów '51.

Na potrzeby demonstracji sposobów dołączenia drukarki do systemu z mikrokontrolerem AT89S8252 wykonałem dwie aplikacje. Pierwsza z nich, której schemat umieszczono na **rys. 2**, steruje drukarką z wykorzystaniem dwóch portów mikrokontrolera. Rzadko można sobie pozwolić na taki luksus (ze względu na zaangażowanie dużej liczby linii I/O), toteż mikrokontroler w aplikacji pokazanej na **rys. 3** steruje drukarką włączoną w obszar adresowy mikrokontrolera. Zapis danych do drukarki odbywa się w taki sam sposób jak zapis lub odczyt komórek pamięci w obszarze adresowania 8-bitowego.

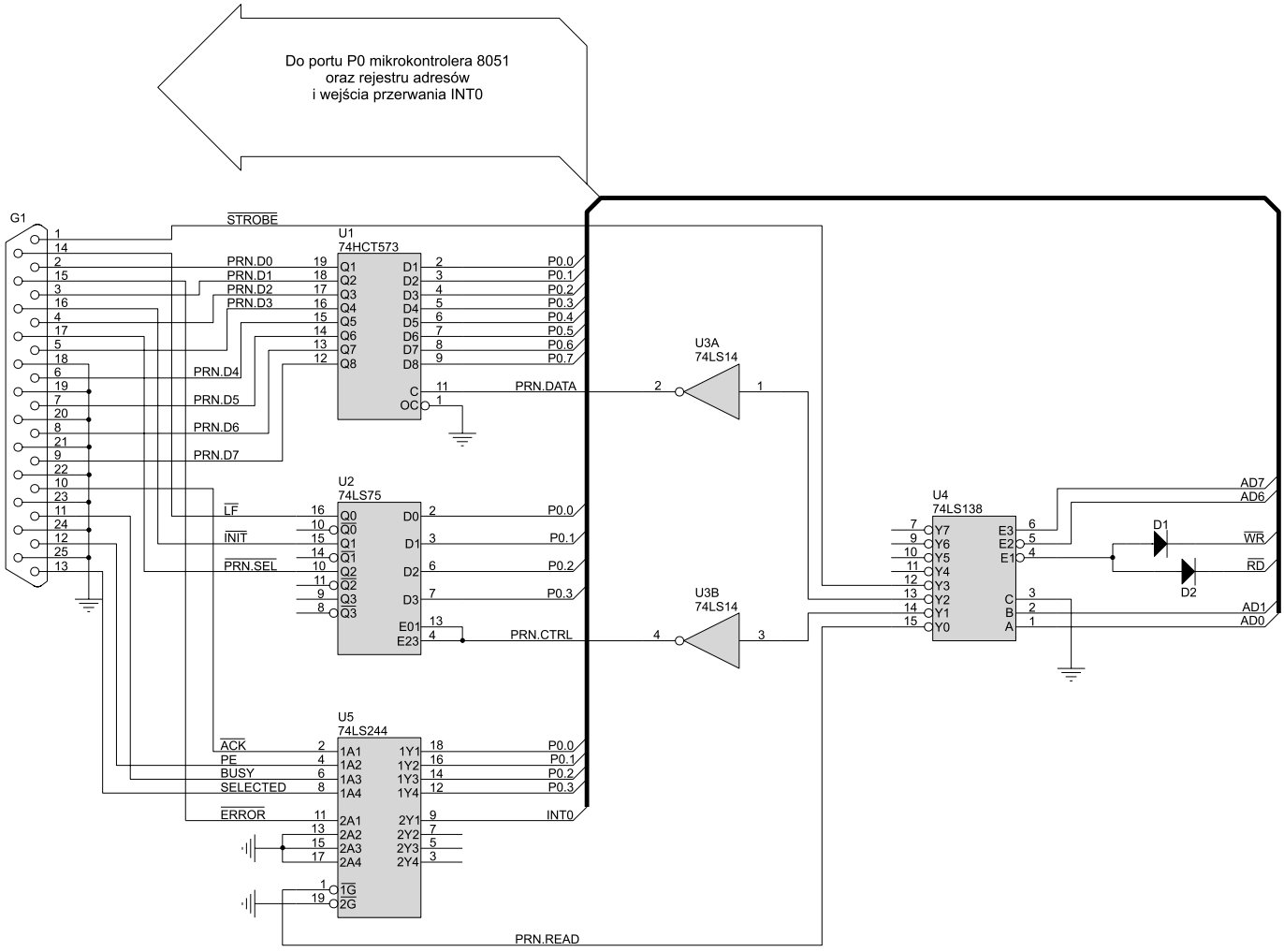
Programy sterujące, pokazane na **list. 1** i **2**, napisano w języku C. Obydwa programy umożliwiają sterowanie drukarką w trybie tekstowym i nie zaimplementowano w nich funkcji związanych z sygnalizacją błędów wydruku lub transmisji danych.

Algorytm działania jest wspólny dla obu aplikacji, aczkolwiek różnią się one pomiędzy sobą szczegółami implementacji. W obu dokonano wymiany standardowej funkcji biblioteki *stdio.h* o nazwie *putchar()* tak, że funkcja *printf()* korzystająca z *putchar()* podczas przesyłania danych, wysyła dane do drukarki zamiast przez interfejs UART mikrokontrolera. Na **rys. 4** pokazano algorytm działania funkcji *putchar()*, po jej wymianie na nową implementację. Oczywiście można napisać własne funkcje obsługi, jednak użycie standardowej funkcji *printf()* umożliwia dostęp do licznych opcji formatowania danych wyjściowych.

Jak można zorientować się na podstawie rysunku 4, jako pierwsze wyprowadzane są dane. Następnie badany jest stan linii BUSY i jeśli ta znajduje się w stanie niskim oznacza to, że kontroler drukarki uwikłany jest w realizację jakiegoś procesu i funkcja oczekuje na



Rys. 2. Sterowanie pracą drukarki za pomocą wyprowadzeń portów



Rys. 3. Drukarka pracująca jako komórka pamięci w obszarze adresowania 8-bitowego

zmianę stanu na wysoki, czyli na zakończenie bieżących zadań i zgłoszenie możliwości przyjmowania danych. Następnie mikrokontroler sterujący przesyła krótki, ujemny impuls STROBE, o czasie trwania większym niż 1μs. Impuls ten zapisuje dane do pamięci drukarki. Teraz funkcja oczekuje na stan niski linii ACK sygnalizujący odbiór bajtu danych. Cykl ten powtarzany jest dla każdego, wysyłanego do drukarki, kodu znaku.

Sterowanie przy pomocy portów.

Na listingu 1 umieszczono program sterujący drukarką podłączoną jak na schemacie z rysunku 2. Zaletą jest prostota wykonania układu, natomiast wadą ilość wykorzystanych doprowadzeń mikrokontrolera. Praktycznie zajęte są prawie wszystkie linie dwóch portów. Dla uproszczenia pominięto sterowanie sygnałem /LF oraz testowanie stanu linii SELECTED, która to sygnalizacja i tak nie jest wykorzystywana w drukarce EPSON LX400 (wyprowadzenie jest na stałe dołączone do +5V przez rezystor 3,3kΩ).

Program napisano w języku C dla mikrokontrolera z rodziny 8051/8052. Funkcja *main()* zawiera inicjalizację drukarki a następnie, w przypadku pomyślnego jej przebiegu, wysyła do drukarki napisy. Funkcja *putchar()* wysyłająca znaki jest bardzo prosta i praktycznie polega na odpowiednim sterowaniu wyprowadzeniami portów mikrokontrolera (zgodnym z opisanymi wcześniej sekwencjami czasowymi) oraz testowaniu stanów linii statusu. Linia sygnalizacji błędu dołączona jest do wejścia przerwania INT0 tak, że opadające zbocze sygnału powoduje przejście do funkcji obsługi przerwania. Zarówno program główny jak i funkcja obsługi przerwania, nie zawierają procedur obsługi błędów. Jak wspomniano wcześniej, szczegółową implementację pozostawiono Czytelnikowi.

Drukarka jako komórka pamięci

Drugi przykład aplikacji jest znacznie bardziej rozbudowany od strony sprzętowej. Konieczne jest zarówno zbudowanie rejestru adresów jak i dekodera adresów oraz układów buforujących dane, sygnały kontrolne i umożliwiających odczyt linii statusu drukarki.

Mikrokontroler z rodziny 8051/8052 żądając dostępu do zewnętrznej komórki pamięci, przesyła przez port P0 na przemian adres oraz dane. Rodzaj przesyłanego słowa (adresowe czy danych) sygnalizowany jest przez wyprowadzenie o nazwie ALE (*Address Latch Enable*). Opadające zbocze ALE powoduje zapis młodszego bajtu adresu do rejestru adresowego (nie ma go na schemacie) oraz sygnalizuje pojawienie się bajtu danych. Mikrokontroler posiada na swojej liście rozkazy umożliwiające dostęp zarówno dostęp do danych leżących w obszarze adresowania 8-bitowe-

List. 1. Program zapewniający sterowanie drukarką dołączonej bezpośrednio do portów mikrokontrolera

```
/*********************************************
**************
 DOŁĄCZENIE DRUKARKI 9-IGŁOWEJ W TRYBIE: EP-
SON_STANDARD_TEXT
 DO MIKROKONTROLERA Z RODZINY INTEL 8051. LI-
NIE DANYCH DRUKARKI
 DOŁĄCZONE DO P0 PRZEZ DRIVER, LINIE KONTROLNE
DO P2 (WEJŚCIA
 I WYJŚCIA), LINIA ZGŁOSZENIA BŁĘDU /ERROR DO
/INT0 MIKROKON-
 TROLERA. KWARC 11,0592 MHZ.
 *********************************************
**************/

#pragma SMALL
#include <reg8252.h>
#include <stdio.h>

//SPOSÓB DOŁĄCZENIA DRUKARKI
//linie danych: P0^0 .. P0^7
sbit ACK = P2^3;
sbit BUSY = P2^4;
sbit ERROR = P3^2; //INT0
sbit STROBE = P2^0;
sbit INIT = P2^1;
sbit PRNSEL = P2^2;

//obsługa przerwania INT0,to jest stanu sygna-
lizacji błędu
void isr_INT0() interrupt 0 using 1
{
   BYTE status = P2;    //odczyt statusu dru-
karki
            //<-- tu obsługa sygnalizacji
błędu
}

//funkcja realizuje opóźnienie k*1ms dla rezo-
natora f=11.0592 MHz
void delay(WORD k)
{
   WORD i,j;
   for ( j = 0; j < k; ++j)
       for (i = 0; i <= 451; ++i);
}

//funkcja putchar wysyła dane do drukarki,
linia STROBE musi być w
//stanie wysokim!
int putchar (const int c)
{
   while (BUSY);    //testowanie stanu BUSY,o-
czekiwanie na stan niski
   P0 = c;          //wyprowadzenie danych
   STROBE = 0;      //ujemny impuls na linii
STROBE
   delay(1);     //to jest zapis danych do dru-
karki
   STROBE = 1;
   while (!ACK);    //oczekiwanie na stan ni-
ski linii ACK
}

//inicjalizacja drukarki (zerowanie bufora i
pozycjonowanie głowicy)
bit initprinter()
{
   BYTE cnt = 30;       //30 sekund oczekiwania
na ustąpienie BUSY
            //po wysłaniu sygnału INIT
   PRNSEL = 0;      //aktywny sygnał wyboru
drukarki
   INIT = 0;     //zerowanie bufora drukarki,
pozycjonowanie głowicy
   delay(1);     //(impuls o czas trwania ~1ms
na linii INIT)
   INIT = 1;
   while (BUSY & cnt--) delay(1000); //dopóki
BUSY=1 i cnt>0
   return (ERROR);  //funkcja zwraca stan li-
nii sygnalizacji błędu
            //jeśli błąd, to stan linii = 0
}

//program główny
void main()
{
   P0 = P2 = 0xFF;
   EA = EX0 IT0 = 1;    //załączenie obsługi
przerwań zewnętrznych (opadające zbocze
            //zbocze sygnału na INT0)
   if (initprinter())
   {
      printf("%s\n\r", "tO jEST tEST dRUKAR-
KI!");
      printf("%s\n\r", "ABCDEFGHIJKLMNOPQRSTU-
VWXYZ 01234567890");
      printf("%s\n\r", "abcdefghijklmnopqrstu-
vwxyz 01234567890");
   }
      else
   {
      //tu funkcja sygnalizacji błędu
   }
   while(1);
}
```

List. 2. Program obsługi drukarki włączonej w obszar adresowania 8-bitowego

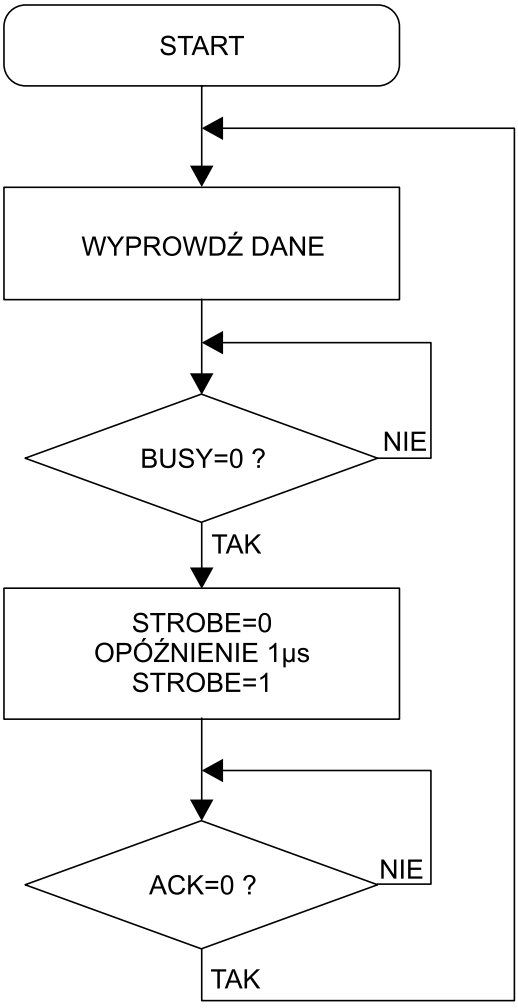
```
/*********************************************
**************
 DOŁĄCZENIE DRUKARKI 9-IGŁOWEJ W TRYBIE: EP-
SON_STANDARD_TEXT
 DO MIKROKONTROLERA Z RODZINY INTEL 8051. DRU-
KARKA DOŁĄCZONA
 W OBSZARZE ADRESOWANIA 8-BITOWEGO.
 *********************************************
**************/
#pragma SMALL
#include <reg8252.h>
#include <stdio.h>
//sygnały sterujące pracą interfejsu
#define LINEFEED 0x01   //stan niski powoduje,
że papier automatycznie jest wysuwany
            //o 1 linię po zakończeniu wydruku
#define INIT     0x02   //stan niski powoduje
wyzerowanie bufora drukarki
            //oraz ustawienie głowicy w pozy-
cji spoczynkowej
#define   PRNSEL     0x04   //stan wysoki powo-
duje, że można zmieniać status drukarki
            //wysyłając kody DC1/DC3
//sygnały kontrolne drukarki
#define   ACK    0x01   //sygnał potwierdzenia
odbioru danych przez drukarkę
            //aktywny jest stan niski
#define   BUSY       0x04   //sygnalizacja zaję-
tości drukarki (bufor nie gotowy,
            //drukarka off-line, błąd drukar-
ki) aktywny stan wysoki
#define   PE     0x02   //sygnalizacja braku
papieru w drukarce, aktywny stan
            //wysoki
#define   SELECTED  0x04   //zgłoszenie, że
drukarka jest on-line i gotowa do
            //pracy, aktywny jest stan wysoki
at 0x80 pdata BYTE READ_STATUS;  //rejestr
statusu drukarki (tylko odczyt)
at 0x81 pdata BYTE CTRL_SIGNALS; //rejestr sy-
gnałów LF, INIT, PRINTER SELECT (tylko zapis)
at 0x82 pdata BYTE PRINTER_DATA; //rejestr
danych drukarki (tylko zapis)
at 0x83 pdata BYTE STROBE;   //wysłanie impul-
su na linii STROBE (zapis i odczyt)
sbit ERROR = P3^2;        //linia sygnalizacji
błędu - INT0
//obsługa przerwania INT0,to jest stanu sygna-
lizacji błędu
void isr_INT0() interrupt 0 using 1
{
   BYTE status;

   status = READ_STATUS;       //odczyt statu-
su drukarki
   //tu obsługa sygnalizacji błędu
}
//funkcja realizuje opóźnienie k*1ms dla rezo-
natora f=11.0592 MHz
void delay(WORD k)
{
   WORD i,j;
   for ( j = 0; j < k; ++j)
       for (i = 0; i <= 451; ++i);
}
//inicjalizacja drukarki (zerowanie bufora i
pozycjonowanie głowicy)
bit initprinter()
{
   BYTE cnt = 30;           //30 sekund oczeki-
wania na ustąpienie BUSY
               //po wysłaniu sygnału INIT
   while ((READ_STATUS && BUSY) & cnt--)
delay(1000); //dopóki BUSY=1 i cnt>0
   return (ERROR);       //funkcja zwraca stan
linii sygnalizacji błędu
               //jeśli błąd, to stan linii =
0
}
//funkcja putchar wysyła dane do drukarki
int putchar (const int c)
{
   while (READ_STATUS && BUSY);     //testowa-
nie stanu BUSY,oczekiwanie na stan niski
   PRINTER_DATA = c;      //zapis danych do
rejestru danych drukarki
   STROBE = 0;           //ujemny impuls na
linii STROBE
   while (!(READ_STATUS && ACK)); //oczekiwanie
na stan wysoki linii ACK
}
//program główny
void main()
{
   EA = EX0 IT0 = 1;//załączenie obsługi prze-
rwań zewnętrznych (opadające zbocze
            //zbocze sygnału na INT0)
   if (initprinter())
   {
      printf(„%s\n\r", „tO jEST tEST dRUKAR-
KI!");
      printf(„%s\n\r", „ABCDEFGHIJKLMNOPQRSTU-
VWXYZ 01234567890");
      printf(„%s\n\r", „abcdefghijklmnopqrstu-
vwxyz 01234567890");
   }
      else
   {
      //tu funkcja sygnalizacji błędu
   }
   while(1);
}
```

go, jak i 16-bitowego, wykorzystując jako rejestr adresowy bądź to rejestr 8-bitowy Ri, bądź to rejestr 16-bitowy DPTR. W praktyce używanie adresu 16-bitowego skutkuje zmianą stanu P2 w czasie dostępu do danych. Ze względu na to, że P2 może być używany do innych celów, wybrano obszar adresowania 8-bitowego, który jest wystarczający dla poprawnej aplikacji. Wówczas to stan portu P2 nie zmienia się podczas dostępu do danych.

Układ U1 (74HCT573) to ośmiokrotny przerzutnik „D", który pełni rolę rejestru danych przesyłanych do drukarki. Układ U2 (74LS75) to również przerzutnik typu „D", jednak w obudowie znajdują się tylko 4 takie przerzutniki. Trzy linie wyjściowe (Q1, Q2, Q3) pełnią rolę sygnałów kontrolnych interfejsu drukarki. Układ U5 (74LS244), to bufor wejściowy umożliwiający po zaadresowaniu odczyt stanu linii statusu drukarki (BUSY, PE, ACK, SELECTED). Wyjątkiem jest linia ERROR, którą dołączono za pośrednictwem przez cały czas otwartego bufora do wyprowadzenia INT0 mikrokontrolera. Układ U4 (74LS138) pełni rolę dekodera adresów. Jest to układ demultipleksera 3/8 wyposażony dodatkowo w trzy wejścia ENABLE (E1, E2, E3).

Na pojawienie się stanu niskiego na wyjściu demultipleksera U4 zezwala następująca formuła: (!RD x !WR) x !AD6 x AD7. Wyjścia wybierane są poprzez wejścia adresowe A i B, dołączone odpowiednio do AD0 i AD1. Stan tych dwóch linii adresowych będzie wpływał na to, które wyjście będzie załączone. Spróbujmy rozszyfrować adresy poszczególnych układów:

1. Wyjście Y0 dołączone jest do U5. Wymagane jest zatem, aby AD0 = AD1 = 0, AD6 = 0 i AD7 = 1. Pozostałe bity słowa adresu nie mają żadnego znaczenia. Można stąd wywnioskować, że na magistrali adresowej musi się pojawić następująca kombinacja bitów: 10xxxx00. Zastępując „x" 0 otrzymujemy adres, który najwygodniej jest wyrazić szesnastkowo jako 0x80.
2. Wyjście Y1 dołączone jest poprzez inwertor do U2. Użycie inwertora jest konieczne, ponieważ zgodnie z zasadą działania przerzutnika typu Latch jest on „przeźroczysty", gdy na wejście taktujące doprowadzona jest „1",



Rys. 4. Algorytm działania nowej implementacji funkcji putchar()

natomiast zmiana stanu z 0 na 1 na tym wejściu, powoduje zapamiętanie informacji. Układ U2 może być zapisany jako komórka pamięci o adresie o 1 wyższym, niż adres U5. Jest to 0x81 (szesnastkowo).

3. Wyjście Y2 dołączone jest (podobnie jak Y1) przez inwertor do U1 pełniącego rolę rejestru danych drukarki. Adres, pod którym można rejestr danych drukarki zapisać, to 0x82 (szesnastkowo).
4. Wyjście Y3 wypracowuje sygnał STROBE. Zapis lub odczyt bajtu spod adresu 0x83 powoduje krótki impuls na wyjściu Y3 trwający tyle, ile sygnał RD czy WR, więc czas jego trwania będzie zależał od częstotliwości zegara mikrokontrolera. Opisywane układy pracowały z zegarem 11,0592 MHz. Aplikacja sterująca zapisuje najpierw rejestr danych a później wysyła bajt o dowolnej wartości pod adres 0x83.

Adresy zadeklarowane zostały jako zmienne leżące w obszarze PDATA mikrokontrolera. W związku z tym, że jest to obszar rozciągający się w pamięci zewnętrznej od adres 0 do 0xFF, mikrokontroler adresuje go pośrednio przy pomocy rejestrów 8-bitowych. W ten sposób stan portu P2 nie zmienia się podczas dostępu do rejsetrów związanych ze sterowaniem drukarki.

Zapis rejestru polega na przypisaniu odpowiedniej zmiennej wartości a odczyt, na pobraniu wartości zmiennej przez jej użycie w operacjach logicznych lub przypisania. Kompilator sam „wie" na podstawie deklaracji, że przy odczycie czy zapisie tego rodzaju zmiennych, należy odwołać się do pamięci zewnętrznej. Tak dla przykładu może wyglądać zapis rejestru danych drukarki: PRINTER_DATA = 'A'. A tak pobranie wartości z rejestru statusu: *unsigned char status = CTRL_SIGNALS* lub *CTRL_SIGNALS && 0x01*.

Podobnie jak w poprzednim programie (patrz list. 1) tak i tu wymieniona została funkcja *putchar()*. Jednak, mimo, iż algorytmy działania funkcji z list. 1 i 2 są zgodne, to jednak implementacja jest zupełnie inna. Funkcja pokazana na list. 2 nie ustawia bezpośrednio stanów linii portów a jedynie zapisuje do zmiennych wartości. Otoczenie sprzętowe mikrokontrolera samo wypracowuje niezbędne sygnały sterujące. Można powiedzieć, że kosztem dodatkowych układów TTL i miejsca na płytce można znacznie uprościć program sterujący.

Mam nadzieję, że przedstawione wyżej przykłady aplikacji będą wystarczającymi wskazówkami do samodzielnego eksperymentowania. Okiełznanie drukarki pracującej w trybie tekstowym nie jest trudne. Znacznie gorzej jest w trybie graficznym. Dodatkowo drukarki mają tę nieprzyjemną cechę, że każdy producent stosuje jakieś własne, charakterystyczne rozwiązania i nie zawsze można drukarką sterować za pomocą tych samych poleceń. Jest to cecha dobrze znana twórcom starszego oprogramowanie pracującego pod kontrolą systemu DOS. Można powiedzieć, że pewne rozkazy są wspólne dla wszystkich drukarek, ale chcąc zmienić czcionkę czy zagęścić wydruk, można napotkać na duże różnice w formacie poleceń pomiędzy drukarkami. Konstruując interfejs przeznaczony do pracy z konkretną drukarką należy bacznie prześledzić informacje zawarte w instrukcji użytkownika.

Jacek Bogusz, EP
jacek.bogusz@ep.com.pl