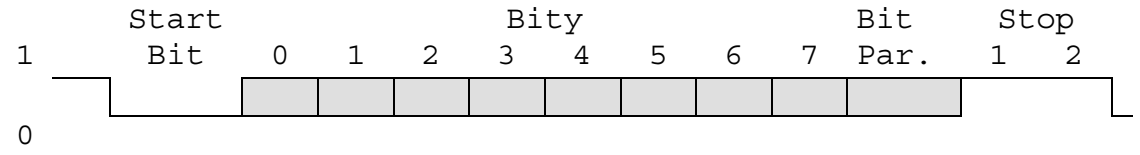
Temat: Obsługa portu komunikacji szeregowej RS232 w systemie STRC51. Ćwiczenie 2. (sd)

1.Wprowadzenie do komunikacji szeregowej RS232

Systemy bazujące na procesorach C51 mogą komunikować się za pomocą standardu RS232. W najprostszym przypadku może być to komunikacja z komputerami PC. Rysunek 1. pokazuje format wymiany danych - najniższa warstwa komunikacji.



Rys. 1

Każda ramka rozpoczyna się Bitem startu zawsze równym 0, po którym przesyłane jest 5,6,7,8 lub 9 bitów informacji. Dla celów weryfikacji poprawności transmisji może być używane kontrolowanie parzystości / nie parzystości. Algorytm ten sprzętowo wspomagany, generuje wartość (bitu parzystości): 0 lub 1, w taki sposób aby przy ustawionym generowaniu parzystości w całości transmitowanej informacji była parzysta liczba jedynek, lub dla generowanej nieparzystości liczba jedynek była nie parzysta. Dla programisty ważne jest tylko czy C51 jest ustawiony w taki sam tryb co strona przeciwna (np. PC). Podobnie z punktu widzenia programisty jest z bitem stopu. Standard zakłada, że obszar stopu może mieć długość: 1, 1½, 2 elementarnej informacji (równe długości bitu). Po nadanej ramce danych może zostać nadana następna, bez żadnego opóźnienia czy potwierdzenia. Należy zaznaczyć, że istnieje system potwierdzania lub bardziej precyzyjnie kontroli przepływu wymiany danych - zagadnienie to jednak wykracza poza ramy ćwiczenia.
Użycie pewnych trybów komunikacji w STRC51 jest niemożliwe z powodów nie wspierania ich w samym procesorze (np. długość bitu stopu 1½), inne są nie dostępne, brak sprzętowego wsparcia w systemie STRC51 - niewykonane niektóre połączenia na płytce, dokładniejsza analiza możliwości procesora AT89C51 i schematu STRC51 może być pomocna w ustaleniu możliwości komunikacyjnych.

2.Zasada pracy portu transmisji szeregowej w STRC51

W mikrokontrolerze 80C51 mamy do dyspozycji rejestry specjalne obsługujące komunikację szeregową: SCON, TMOD, TCON, PCON, SMOD, SBUF.

Rejestr SCON umieszczony pod adresem 0x98 w przestrzeni pamięci wewnętrznej rejestrów specjalnych (deklaracja: `sfr at 0x98 SCON;`)
Poniższa tabela przedstawia organizacje i symboliczne oznaczenie bitów:

7	6	5	4	3	2	1	0
SM1	SM0	SM2	REN	TR8	RB8	TI	RI

Gdzie bity oznaczają:

SM0, SM1 - Określenie trybu pracy łącza

SM1	SM0	
0	0	- tryb 0, transmisja synchroniczna,
0	1	- tryb 1, transmisja asynchroniczna, z regulacją prędkości transmisji, 8 bitowy transfer,

1 0 - tryb 2, transmisja asynchroniczna bez możliwości regulacji prędkości, 9 bitowy transfer,
1 1 - tryb 3, transmisja asynchroniczna z możliwością regulacji prędkości, 9 bitowy transfer,

SM2 - Bit sterujący pracą tryb: punkt-punkt (0), wieloprocesorowa (1),[1].
REN - Bit zezwolenia na odbiór danych przez łącze szeregowe (1),
TR8 - Dodatkowy dziewiąty bit transmisji wieloprocesorowej,
RB8 - Dodatkowy dziewiąty bit transmisji wieloprocesorowej,
TI - Wskaźnik przerwania (opróżnienia) od bufora nadawczego (1), należy go kasować programowo,
RI - Wskaźnik przerwania (napełnienia się) bufora odbiorczego (1), należy go kasować programowo.

Rejestr TMOD opisany dokładnie w ćwiczeniu 3 – ustawia pracę Timera niezbędnego do prawidłowego funkcjonowania łącza szeregowego. Nie wchodzą w szczegóły dla poprawnej pracy, należy wykonać następującą sekwencję (ustawienie licznika w tryb samo przeładowywania):

```
TMOD &= 0x0F;
TMOD |= 0x20;
```

Sprawi ona że licznik nr.1 będzie dostarczał niezbędny dla układów UART sygnał zegarowy (zgodnie z ustawieniami TH1).

Rejestr TCON umieszczony od adresem 0x88 w przestrzeni pamięci wewnętrznej rejestrów specjalnych (deklaracja: `sfr at 0x88 TCON;`).
Poniższa tabela przedstawia organizacje i symboliczne oznaczenie bitów:

7	6	5	4	3	2	1	0
TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

TF1 - Wskaźnik przerwania od licznika 1, zerowany sprzętowo podczas wykonywania procedury przerwania,
TR1 - Bit włączenia(1)/wyłączenia(0) licznika 1,
TF0 - Wskaźnik przerwania od licznika 0, zerowany sprzętowo podczas wykonywania procedury przerwania,
TR0 - Bit włączenia(1)/wyłączenia(0) licznika 0,
IE1 - Wskaźnik przerwania od wejścia /INT1, zerowany sprzętowo podczas wykonywania procedury przerwania,
IT1 - Bit określający sposób reakcji na sygnał na wejściu /INT1:
0 - przerwanie wyzwalane niskim poziomem na wejściu,
1 - przerwania wyzwalane zboczem na wejściu,
IE0 - Wskaźnik przerwania od wejścia /INT0, zerowany sprzętowo podczas wykonywania procedury przerwania,
IT0 - Bit określający sposób reakcji na sygnał na wejściu /INT0:
0 - przerwanie wyzwalane niskim poziomem na wejściu,
1 - przerwania wyzwalane zboczem na wejściu,

[1] Standardowo komunikacja wieloprocesorowa nie jest dostępna w komputerach PC.

Temat: Obsługa portu komunikacji szeregowej RS232 w systemie STRC51. Ćwiczenie 2. (sd)

Rejestr PCON umieszczony pod adresem `0x87` w przestrzeni pamięci wewnętrznej rejestrów specjalnych (deklaracja: `sfr at 0x87 PCON;`).

7	6	5	4	3	2	1	0
SMOD	-	-	-	-	-	-	-

`SMOD` - Bit sterujący prędkością pracy łącza szeregowego (patrz wzór w dalszej części dokumentu).

Rejestr `SBUF` umieszczony pod adresem 0x99 w przestrzeni pamięci wewnętrznej rejestrów specjalnych (deklaracja: `sfr at 0x99 SBUF;`).
Jest on rejestrem nadawczo/odbiorczym, transmisji szeregowej. Czytana wartość z tego rejestru jest wartością ostatnio odebranego znaku (jeżeli został takowy odebrany), wartość wpisana do tego rejestru będzie transmitowana przez nadajnik.
Uwaga nie można odczytać wartości wpisanej w poprzednim wpisie – wartość tak odczytana jest dokładnie wartością ostatnio odebraną a nie nadaną).

3.Użycie przerwań związanych z transmisją szeregową
Przed przystąpieniem do obsługiwania łączności szeregowej z udziałem przerwań, należy prawidłowo zainicjować odpowiednie rejestry.
- ustawiamy tryb transmisji łącza (`SCON`),
- zezwolenie na obsługę odbierania danych z łącza (`SCON`),
- ustawienie trybu pracy licznika 1 (`TMOD, TCON`) z pobudzaniem z wewnętrznego oscylatora,
- ustawienie bitu `SMOD` (`PCON`),
- ustalenie – wpisanie prędkości transmisji (`TH1 i TL1`),
- zezwolenie na generowanie przerwań od łącza szeregowego (`ES`),
- zezwolenie na obsługę wszelkich przerwań (`EA`).

Na uwagę zasługuje sposób wyliczania wartości wpisywanej do `TH1` w zależności od częstotliwości oscylatora zasilającego procesor. Najwygodniej jest używać w tym liczniku tryb z auto-przeładowywaniem, ustawiając licznik `T1` w tryb 2.
Prędkość transmisji wyznaczana jest ze wzoru:

```
Baud = fosc * 2^SMOD / (32*12*(256-TH1))
```

`fosc` - częstotliwości oscylatora w `STRC51` równa `11059200` (co odpowiada częstotliwości rezonatora kwarcowego wmontowanego w `STRC51`),

Po tak pracochłonnych czynnościach można uważać port transmisji szeregowej za skonfigurowany.
Należy pamiętać, że:
-procedura przerwania jest wywoływana raz dla każdego odebranego/nadanego znaku,
-bufor `SBUF` może być tylko raz odczytany dla każdego znaku odbieranego i raz zapisany dla nadanego znaku,
-sprawdzenie czy przerwanie wygenerował sygnał opróżnienia bufora nadawczego czy zapełnienie bufora odbiorczego może być przeprowadzone przez badanie bitu `TI` lub `RI` (odpowiednio).
-po wykonaniu odpowiednich operacji związanych z `SBUF` należy wyzerować bity `TI/RI`.
Poniższy pseudokod prezentuje szablon uruchamiania połączenia szeregowego z nadawaniem w poolingu jednego znaku jak i w podobny sposób jego odbieraniem.

Temat: Obsługa portu komunikacji szeregowej RS232 w systemie STRC51. Ćwiczenie 2. (sd)

`<sekcja include>`	Dołączenie wszelkich plików nagłówkowych – jeżeli potrzebne	

`Funkcja-getchar{`
` petla-az-znak-gotowy;`
` znak<-SBUF;`
` return znak;`
`}`
	Funkcja getchar	Czekanie na znak (sprawdzanie bitu `RI`)
		Odczytanie znaku z `SBUF`
		Zwrócenie znaku

`Funkca-putchar{`
` SBUF<-znak-do-wyslania;`
` czekaj-az-znak-sie-wysle;`
`}`
	Funkcja putchar	Wpisanie do `SBUF` nowego znaku
		Czekamy aż znak zostanie nadany (`TI`)

`main(){`
` SCON=?`
` TMOD &= 0x0f;`
` TMOD |= 0x20;`
` TCON=?`
` PCON=?`
` TH1=TL1=?`
` TI=?`
` petla-nieskonczona{`
`  c<-funkcja-getchar();`
`  c<-zmodyfikuj-znak(c);`
`  funkca-putchar(c);`
` }`
`}`
	Funkcja main	Konfiguracja pracy łącza szeregowego
		„Wciągniecie” znaku z łącza
		Zmodyfikowanie znaku
		wysłanie znaku

Powyższy pseudokod w tym przykładzie przetwarza znaki w nieskończonej pętli. Rozwiązanie takie jest nie optymalne, może zdarzyć się, że przy bardziej zaawansowanym modyfikowaniu znaku, lub całych bloków znaki będą gubione – nadejście nowego znaku nadpisuje poprzedni. W takich sytuacjach zaleca się stosowanie buforów okrężnych (zwanych też cyklicznymi, kołowymi) oraz używanie przerwań związanych z łączem szeregowym.

4.Wsparcie transmisji szeregowej w pakiecie `SDCC`

Najlepiej zadeklarować funkcję obsługi przerwania od łącza szeregowego w następujący sposób:

```
void serial_isr(void) interrupt 4{
  if(TI){
   TI=0;              //ten fragment będzie wykonany po wysyłaniu znaków przez C51
   ...
  }
  if(RI){
   RI=0;              //ten fragment będzie wykonany przy odbieraniu znaków przez C51
   ...
  }
}
```

W prezentacji pod adresem: `http://cygnus.tele.pw.edu.pl/olek/doc/np/rs232_cyc_a.pps`, pokazano scenariusz działania buforów i obsługi z wykorzystaniem przerwań.

5.Zmienne i stałe w `SDCC`

Twórcy pakietu `SDCC` przewidzieli następujące (najważniejsze) rozszerzenia deklaracji języka C:

```
{code | xdata | data | idata} <typ zmiennej> <nazwa zmiennej>
```

Temat: Obsługa portu komunikacji szeregowej RS232 w systemie STRC51. Ćwiczenie 2. (sd)

Gdzie symbole mają następujące znaczenie:

`code` -deklaracja obiektu tworzonego w stałej pamięci programu przestrzeni adresowej procesora `C51`, z tego typu przedrostkiem nie wolno tworzyć obiektów zmiennych (podczas pracy programu), ten typ najlepiej stosować w przypadku deklaracji dużych tablic danych (tablica wartości dla funkcji matematycznych),

`xdata` -deklaracja obiektu tworzonego w pamięci danych przestrzeni adresowej procesora `C51`, z tego typu przedrostkiem tworzy się obiekty w pamięci zewnętrznej procesora (która może nie być fizycznie dołączona, w przypadku `STRC51` jest jej `32KB`), w przypadku STRC51 w przestrzeni tej umieszczone są także urządzenia wejścia/ wyjścia.

`data` -deklaracja obiektu tworzonego w pamięci wewnętrznej procesora `C51`, w przypadku całej rodziny `C51` jest jej `128B` (włączając w to stos procesora),

`idata` -deklaracja obiektu tworzonego w pamięci wewnętrznej procesora `C51`, w klasycznych procesorach `80C51` pamięci tej nie ma! (układ AT89C51 też tej pamięci nie ma!)

Dodatkowo (w stosunku do klasycznej wersji `C`) można definiować zmienne następujących typów:

`bit` -najmniejsza (najmniej miejsca zajmuje) zmienna w `C51`, deklaruje zmienną która może przyjąć tylko dwie wartości `0` | `1`, ilość takich zmiennych jest ograniczona do `160` bitów (ten obszar jest alokowany w maksymalnym rozmiarze `20B`), pakiet `SDCC`[2] nie wspiera zwracania wyniku funkcji w takim typie,

`sfr, sbit` -zmienne zasadniczo związane z architekturą procesora (lub jego odmianą), `sfr` - jest zmienną 8 bitową umieszczaną w przestrzeni rejestrów specjalnych (patrz:), `sbit` – zminne 1 bitowe podobnie jak `sfr` umieszczaną w przestrzeni rejestrów specjalnych (patrz:), w wielu przypadkach składowe rejestrów `sfr` np.:

deklaracja `IE`:

```
sfr at 0xA8 IE;
```

deklaracje bitów w rejestrze `IE`:

```
sbit at 0xA8 EX0;
sbit at 0xA9 ET0;
sbit at 0xAA EX1;
...
```

Modyfikator ‘`at`’, używany jest dla adresowania absolutnego zmiennej lub stałej. Przykład wykorzystania (znany z poprzednich ćwiczeń):

```
xdata at 0x8000 unsigned char U15;      //obiekt umieszczony pod adresem
                                        //0x8000 w przestrzeni xdata,
xdata at 0xffff unsigned char U10;      //obiekt umieszczony pod adresem
                                        //0xFFFF w przestrzeni xdata,
```

lub:

```
sbit at 0xB5 T1;                        //deklaracja głośnika w STRC51
```

Więcej informacji co do budowy pakietu `SDCC` można znaleźć w pliku:
`\sdcc\doc\index.html`

[2] Dotyczy wersji `2.3.0` pakietu `SDCC`