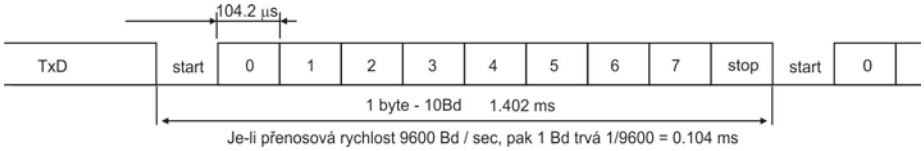
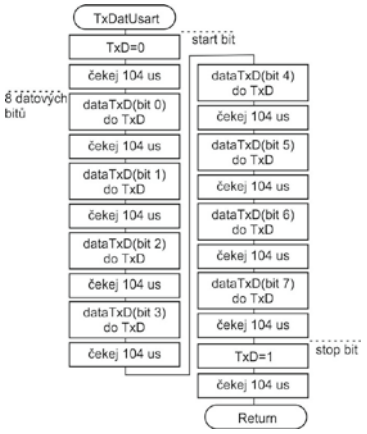
Programování MP3 modulu v Assembleru

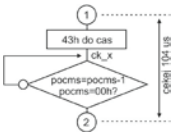
Tento způsob přenosu dat patří mezi zcela základní a už velmi dlouho dobu využívané způsoby. V klidovem stavu je na sběrnici vysoká úroveň nebo také jednička. Vyslání bytu dat je zahájeno vysláním start bitu v nule. Za ním následuje postupné vyslání 8 datových bitů od bitu 0 po bit 7. Jako poslední je v sekvenci vysílán stop bit v jedničce. Start bit, dalové bity i stop bit mají stejnou délku, která se vypočte z požadované přenosové rychlosti. Přenosová rychlost se udává v Baudech (Bd) za sekundu.



Poznámka: *Vyslané sekvenci - start bit, datové bity a stop bit (bity) se říká rámec (frame). Vyslaný rámec se může podle potřeby skládat z jednoho start bitu, 5 až 8 datových bitů, jednoho paritního bitu a jesnoho, nebo dvou stop bitů. Nejčastější tvar je však jeden start bit, 8 datových bitů a jeden stop bit, bez paritního bitu. Pokud v budoucnu bude, třeba popíšeme si i jiný typ rámce.*

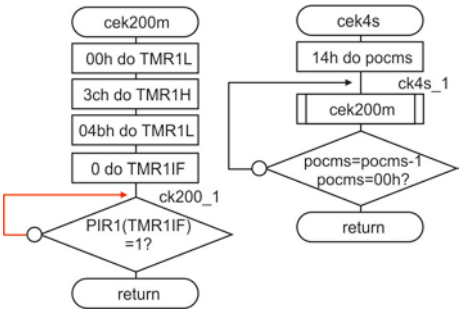
Vyslání dat





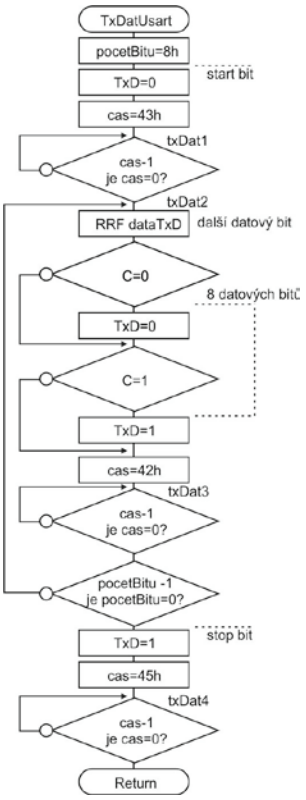

```
; vyslani bytu dat USARTem  9600 Bd, 8 bitů 1 stop - delka Bd = 104.2 us
; celý byte = start,data,stop = 1.146 ms
; cas urcuje delku bodu, dataTxD - byte dat k odeslani
;
txDatUsart:
        bcf     PORTB,TxD       ;vysli START bit
        nop                     ;prodlouzi start bit na 104 us
        movlw   43h
        movwf   cas
txDat1:
        decfsz  cas,regF
        goto    txDat1
;
        nop
        nop
        btfss   dataTxD,0
        bcf     PORTB,TxD       ;vysli datový bit = 0
        btfsc   dataTxD,0
        bsf     PORTB,TxD       ;vysli datový bit = 1
        movlw   43h
        movwf   cas
txDat2:
        decfsz  cas,regF
        goto    txDat2
;
        nop
        btfss   dataTxD,1
        bcf     PORTB,TxD       ;vysli datový bit = 0
        btfsc   dataTxD,1
        bsf     PORTB,TxD       ;vysli datový bit = 1
        movlw   43h
        movwf   cas
txDat3:
        decfsz  cas,regF
        goto    txDat3
;
        nop
        nop
        btfss   dataTxD,2
        bcf     PORTB,TxD       ;vysli datový bit = 0
        btfsc   dataTxD,2
        bsf     PORTB,TxD       ;vysli datový bit = 1
        movlw   43h
        movwf   cas
txDat4:
        decfsz  cas,regF
        goto    txDat4
;
        nop
        nop
        btfss   dataTxD,3
        bcf     PORTB,TxD       ;vysli datový bit = 0
        btfsc   dataTxD,3
        bsf     PORTB,TxD       ;vysli datový bit = 1
        movlw   43h
        movwf   cas
txDat5:
        decfsz  cas,regF
        goto    txDat5
;
        nop
        nop
        btfss   dataTxD,4
        bcf     PORTB,TxD       ;vysli datový bit = 0
        btfsc   dataTxD,4
        bsf     PORTB,TxD       ;vysli datový bit = 1
        movlw   43h
        movwf   cas
txDat6:
        decfsz  cas,regF
        goto    txDat6
;
        nop
        btfss   dataTxD,5
        bcf     PORTB,TxD       ;vysli datový bit = 0
        btfsc   dataTxD,5
        bsf     PORTB,TxD       ;vysli datový bit = 1
        movlw   43h
        movwf   cas
txDat7:
        decfsz  cas,regF
        goto    txDat7
;
        nop
        nop
        btfss   dataTxD,6
        bcf     PORTB,TxD       ;vysli datový bit = 0
        btfsc   dataTxD,6
        bsf     PORTB,TxD       ;vysli datový bit = 1
        movlw   44h
        movwf   cas
txDat8:
        decfsz  cas,regF
        goto    txDat8
;
        nop
        nop
        btfss   dataTxD,7
        bcf     PORTB,TxD       ;vysli datový bit = 0
        btfsc   dataTxD,7
        bsf     PORTB,TxD       ;vysli datový bit = 1
        movlw   43h
        movwf   cas
txDat9:
        decfsz  cas,regF
        goto    txDat9
;
        nop
        nop
        bsf     PORTB,TxD       ;vysli STOP bit
        movlw   45h
        movwf   cas
txDat10:
        decfsz  cas,regF
        goto    txDat10
        return
```

Podprogram vyslání jednoho rámce je jednoduchý a jeho vývojový diagram vidíme na obrázku. Nastavíme příslušnou úroveň na výstup TxD a počkáme odpovídající dobu po kterou má trvat vysílání bitu. Pro rychlost přenosu dat 9600 bodů je tato doba 104,2 mikrosekundy. Část programu, která zajišťuje čekání 104 μs pracuje podobně, jako když počítáme při hře na schovku. Odečítáme tak dlouho, až se dostaneme na nulu. Od jakého čísla odečítáme, to určuje dobu čekání. Přesnou velikost určíme pomocí simulace činnosti procesoru. Zvýšení/snížení čísla od kterého počítáme o 1 nezmění čas čekání o 0.5 μs, ale o 1.5 μs. Při kmitočtu hodin 8 MHz je instrukční cyklus 0.5 μs. Pokud bychom potřebovali například prodloužit čekání o 1.0 μs, stačilo by použít dvě instrukce nop (no operation – nic nedělej) za sebou. Každá z nich trvá 0.5 μs. Vida a už víme, jak přesně nastavit v podprogramu čekání na 104 μs. V podprogramu tedy nesmíme zapomenout na úpravu doby čekání podle požadovaného času, který bude potřeba pro nastavení výstupu TxD do příslušné úrovně. Dobu čekání se snažíme nastavit co nejpřesněji i když se to na 100 procent nepodaří. Dobu čekání lze zvětšit použitím vícebytového čísla, i když toho lze využít pouze v případě, že po celou dobu čekání procesor nic jiného nedělá a celý systém je prakticky mrtvý. Proto delší doby čekání raději realizujeme s použitím čítačů a využitím toho, že signalizují své přetečení (přechod z maximální hodnoty do nuly). Například čítač/časovač TMR1 při přetečení nastaví příznak TMR1IF v registru PIR1. Toho lze využít k tomu, aby program mohl provádět požadované činnosti i když čeká (tyto činnosti zařadíme do červené větve), ale této problematice se budeme věnovat podrobněji později.



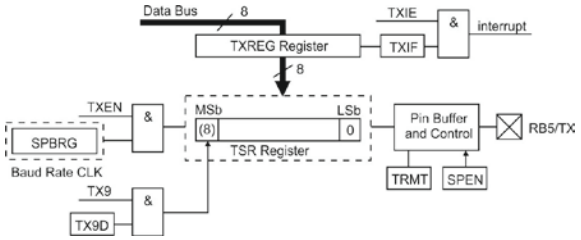
Podprogram vyslání dat můžeme také napsat například tak, že vyslání dat provedeme ve smyčce s tím, že aktuální bit, který chceme vysílat přesuneme pomocí rotace vpravo do stavového slova STATUS, C. Rotace bytu dataTxD provede následují činnost

My pak místo testování jednotlivých bitů registru dataTxD testujeme v každém cyklu bit C registru STATUS, ve kterém se požadované bity díky rotaci nacházejí. Vývojový diagram podprogramu s využití rotace dataTxD ve smyčce je na následujícím obrázku. Díky smyčce se zkrátí celý podprogram, ale doba jeho provádění je přirozeně stejná.




```
;*****************************************************************************************
; podprogramy řídicí modul
;.....................................................................
; vyslani bytu dat USARTem  9600 Bd, 8 bitů 1 stop - delka Bd = 104.2 us
; celý byte = start,data,stop = 1.146 ms ..... prgUSART
; pocetBitu      = 8 pocet vysilanych datovych bitu
; cas urcuje delku bodu, dataTxD - byte dat k odeslani
;.....................................................................
txDatUsart:
        movlw   8h
        movwf   pocetBitu
        bcf     PORTB,TxD               ;vysli START bit
        nop                             ;prodlouzi start bit na 104 us
        movlw   43h
        movwf   cas
txDat1:
        decfsz  cas,regF
        goto    txDat1
txDat2:
        nop
        nop
        rrf     dataTxD,regF
        btfss   STATUS,C
        bcf     PORTB,TxD               ;vysli datový bit = 0
        btfsc   STATUS,C
        bsf     PORTB,TxD               ;vysli datový bit = 1
        movlw   42h
        movwf   cas
txDat3:
        decfsz  cas,regF
        goto    txDat3
        decfsz  pocetBitu,regF
        goto    txDat2
        bsf     PORTB,TxD               ;vysli STOP bit
        movlw   45h
        movwf   cas
txDat4:
        decfsz  cas,regF
        goto    txDat4
        return
;*****************************************************************************************
```

Nejefektivnější je však využít modul AUSART implementovaný v mikrokontroléru. Vyslání bytu je díky modulu snadné. Zapíšeme do TXREG data k vyslání a o vše ostatní se postará sám modul. Potřebujeme-li vyslat více bytů, je to opět snadné. Po zapsání bytu dat k odeslání do registru TXREG vynulujeme příznak TXIF z registru PIR1. V okamžiku, kdy lze vyslat další byte dat, modul nastaví TXIF do H. Tento příznak testujeme a je-li v H, zapíšeme do registru TXREG další byte k vyslání a příznak TXIF opět vynulujeme.



Po dobu, co modul vysílá data, může procesor řešit jiné úkoly. Že byla data vyslána, pozná snadno pomocí příznaku TXIF.

Aby bylo data možné posílat prostřednictvím modulu, musíme modul správně nakonfigurovat. Pro vysílání 9600 Bd, 8 datových bitů, jeden stop bit bez parity je konfigurace následující.

```
;_____________________________________________________
; Inicializace AUSARTu - asynchronni Rx/Tx
;.....................................................
; TXSTA - 0,TX9=0,TXEN=1,SYNC=0, 0, BRGH =1, 00
; RCSTA - SPEN=1,00 CREN=1,00, PIR1(TXIF)
;_____________________________________________________
        BSF     STATUS,RP0              ; Go to Bank1
        MOVLW   33h                     ; Set Baud rate 9.6 kBd
        MOVWF   SPBRG
        MOVLW   0x24                    ; povolení vysílání 8-bitu dat
        MOVWF   TXSTA                   ; asynchronní vysokorychlostní režim
        BCF     STATUS,RP0              ; Go to Bank 0
        MOVLW   0x90                    ; 8-bit příjem dat
        MOVWF   RCSTA                   ; serial port povolen
;_____________________________________________________
```