Lecture 10

Serial Communication

www.elecstudy.org markalex012@gmail.com טסליצקי מרק

תקשורת טורית

- מבוא Introduction
- פסי נתונים שמשתמשים בתקשורת טורית.
- תקשורת סינכרונית ואסינכרונית.
- דיאגרמת בלוקים של UART.
- דרישות תזמון למערכת UART
- תכנון מערכות UART.
- מצבי עבודה.
- חישובי תדר עבודה לטיימר 1.
- אתחול UART בעזרת טיימר 1.
- חישובי תדר עבודה לטיימר 2.
- אתחול UART בעזרת טיימר 2.
- דגלי פסיקות במערכת UARTx בתהליך קבלת נתונים.
- דגלי פסיקות במערכת UARTx בתהליך שליחת נתונים.

www.elecstudy.org markalex012@gmail.com טסליצקי מרק

מבוא

- תקשורת מקבילית – היא העברת הנתונים (בית שלם או יותר) במקביל בעזרת מספר קווי תקשורת.

- תקשורת טורית מאפשרת שליחת נתונים סיבית לאחר סיבית בקו בודד.

- ישנם 2 סוגים של תקשורת טורית:
  - תקשורת אסינכרונית Asynchronous
  - תקשורת סינכרונית Synchronous

www.elecstudy.org markalex012@gmail.com טסליצקי מרק

פסי נתונים שמשתמשים במערכות תקשורת.

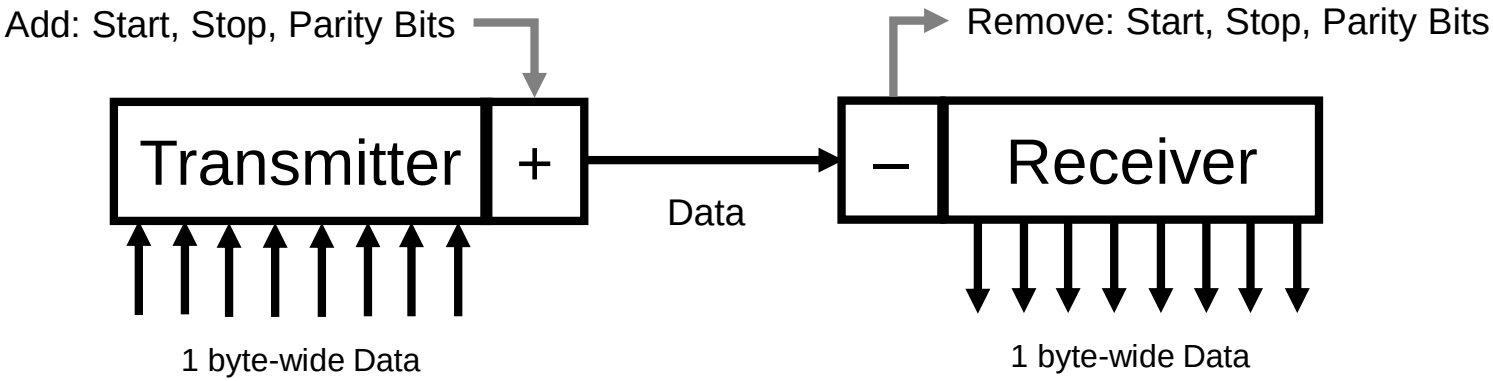
- סטנדרטים של תקשורת טורית:
  - RS-232 (using UART) סטנדרט הבסיסי ביותר בתקשורת אסינכרונית.
  - Serial peripheral interface (SPI) פרוטוקול של תקשורת סינכרונית.
  - System management bus (SMBus) שם נוסף של הפרוטוקול – I2C.
  - Serial ATA (SATA) תקשורת שדיסק קשיח מחובר למחשב.

- במיקרו בקר C8051F020 יש שני מערכות UART, מערכת SPI ומערכת I2C.
- במצגת הזאת נלמד על מערכת UART לתקשורת אסינכרונית טורית.
  UART: Universal asynchronous receiver/transmitter

www.elecstudy.org markalex012@gmail.com טסליצקי מרק

תקשורת אסינכרונית טורית.

- כאשר אנו עובדים בתקשורת אסינכרונית, למשדר ולמקלט אין תזמון משותף.



The Transmitter משדר

- ממיר נתון מקבילי (8 סיביות) לקו טורי בעזרת תזמון שלו.
- מוסיף סיביות התחלה, עצירה וסיבית זוגיות.

The Receiver מקלט

- ממיר נתון טורי למקבילי בעזרת תזמון שלו.
- מוריד סיבית עצירה, התחלה, סיבית זוגיות.

www.elecstudy.org markalex012@gmail.com טסליצקי מרק

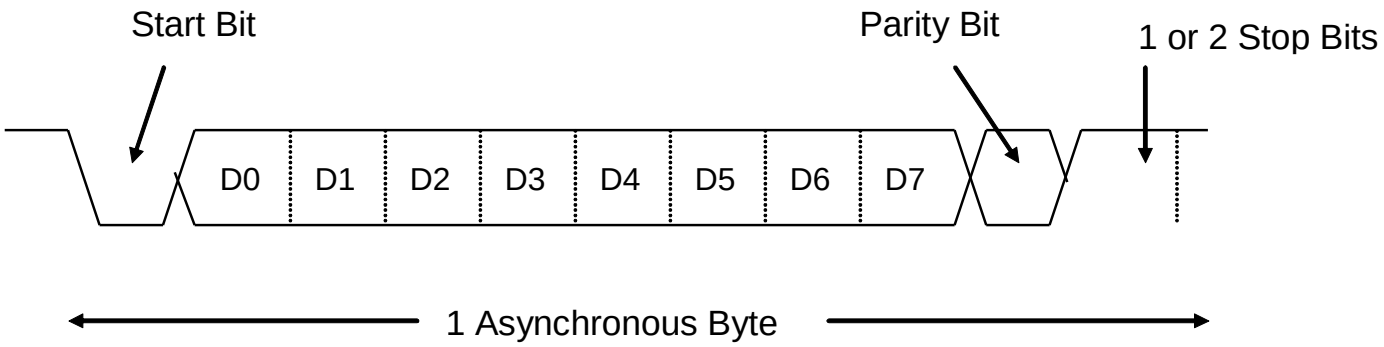
תקשורת אסינכרונית טורית. מושגים בסיסיים.

- סיבית התחלה – מסמן תחילת מילת שידור.
- סיבית עצירה – מסמן סיום מילת שידור.
- סיבית זוגיות (לא חייב) – מוסיפים אותו כדי לבדוק תקינות העברת מידע.
- סיביות נתונים – מידע שאנו רוצים לשדר. Data bits
- קצב בעברת נתונים של פורט טורי. Baud rate
- Throughput - כמות אמיתית של סיביות משודרות בשנייה

לדוגמא: 115200 baud = 115200 bits/sec . במידה ומשתמשים בסיבית התחלה, סיבית עצירה וללא סיבית זוגיות תפוקה אפקטיבית 115200 * 8 / 10 = 92160 bits/sec
(effective throughput)

www.elecstudy.org markalex012@gmail.com טסליצקי מרק

תקשורת אסינכרונית טורית.



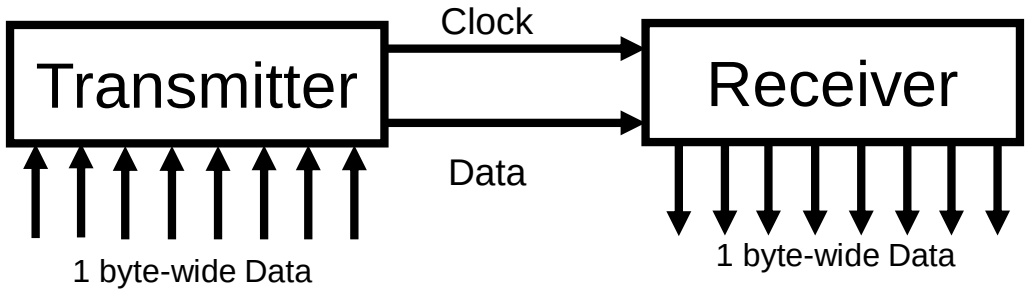
- תקשורת אסינכרונית קלה לשימוש אבל פחות אפקטיבית עקב הוספת 2-3 סיביות בקרה בנוסף ל8 סיביות נתונים.

- שיטה הזאת מקובלת לשידור של נתונים בנפח נמוך.

www.elecstudy.org markalex012@gmail.com טסליצקי מרק

תקשורת סינכרונית טורית.

- במצב סינכרוני משדר ומקלט עובדים ביחד עם שעון משותף.
- בד"כ משדר הוא זה שמייצר פולס שעון המשותף בנוסף לקו נתונים.



<u>משדר</u>

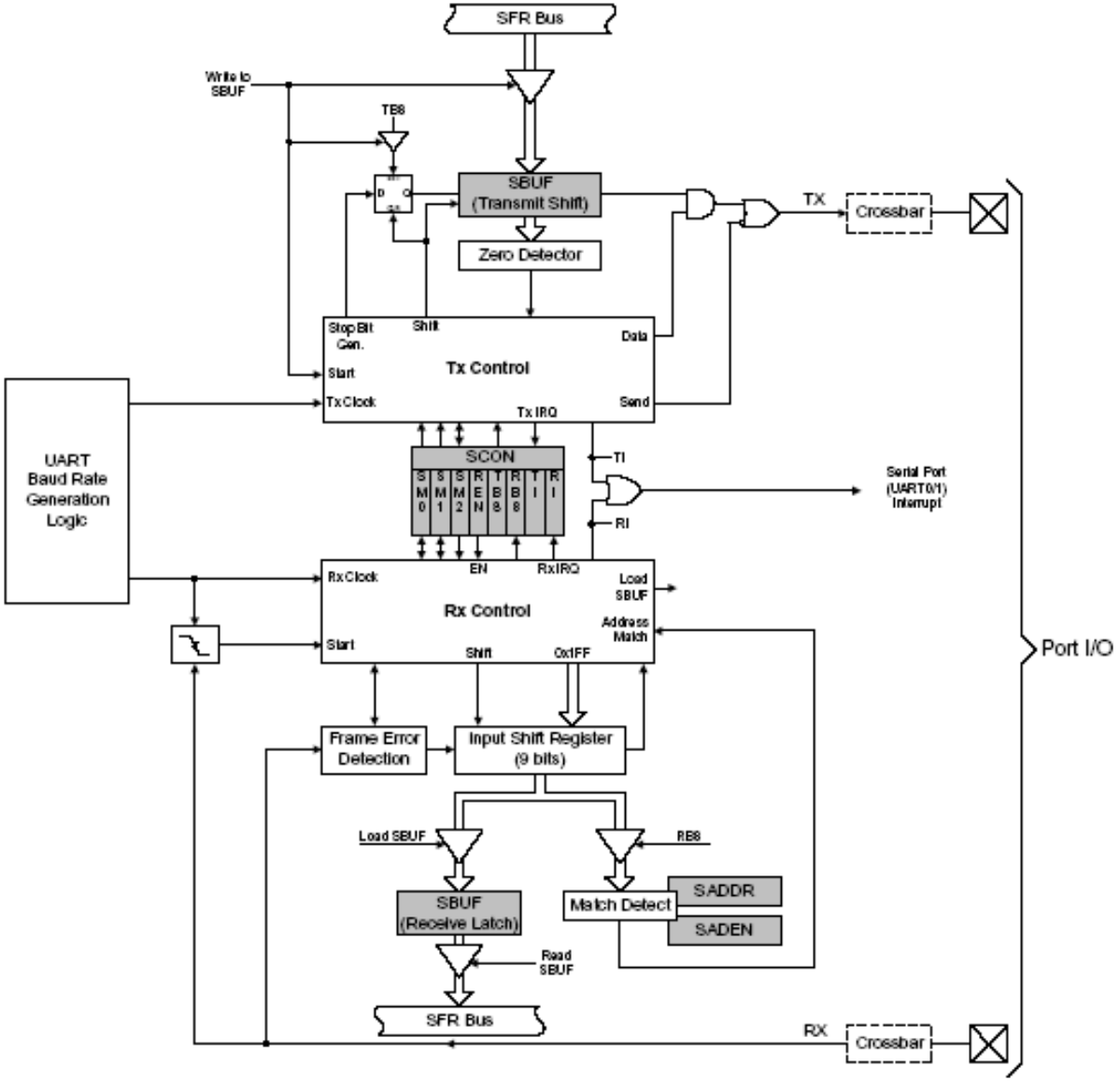
- מעביר נתונים ממקבילי לטורי בעזרת שעון שלו.
- מייצר שעון בתור אות נפרד.
- אין צורך להוסיף סיביות בקרה, התחלה ועצירה.

<u>מקלט</u>

- מקבל נתונים בעזרת פולסי שעון שמקבל ממשדר בקו נפרד..
- מעביר אותם מצורה טורית בחזרה למקבילית.

www.elecstudy.org markalex012@gmail.com טסליצקי מרק

דיאגרמת בלוקים של UART.



www.elecstudy.org markalex012@gmail.com טסליצקי מרק

UART Block

- הפעלת כל ערוץ UART מורכבת מבקרת ערכים של שני אוגרים: **SBUFx**, **SCONx**

- אוגר Serial Port Buffer (SBUFx) מכיל 2 חוצצים – חוצץ שמכיל נתונים לשידור וחוצץ שקולט נתונים שמגיעים למקלט.
- לכל חוצץ יש הגבלה משלו – חוצץ של משדר הוא לכתיבה בלבד, וחוצץ של מקלט הוא לקריאה בלבד.

- אוגר Serial Port Control register (SCONx) מכיל מצב של מערכת וסיביות בקרה של תהליך תקשורת.
  - סיביות בקרה מגדירות צורת המערכת וסיביות מצב מכילות מידע על סיום שידור או קליטה.
  - ניתן לבדוק סיביות מצב בעזרת שיטת polling (כמו שעשינו בטיימרים) או להשתמש במנגנון פסיקות.

www.elecstudy.org markalex012@gmail.com טסליצקי מרק

דרישות תזמון של UART

- UART דורש פולס שעון כדי לתזמן אירועים.
- ערך של שעון של UART (baud rate) בד"כ הרבה יותר נמוך מזה שמיקרו בקר עובד. עקב כך אי-אפשר להשתמש ישירות בתזמון של מיקרו בקר לצורך תקשורת טורית.
- ניתן לייצר תדר לתזמון המערכת בעזרת טיימרים. מחלקים תדר של הגביש ומייצרים תזמון לתקשורת טורית.
  - לדוגמא: MCU system clock—22 MHz; UART baud rate—115200
- דרוש דיוק $\pm 2\%$ או יותר גם למשדר וגם למקלט כדי שהם יהיו מסוגלים לתקשר ללא שגיאות.
  - כדי ליישם דיוק כזה גבוה, משתמשים בגביש חיצוני עם דיוק של 0.1% או יותר.

www.elecstudy.org markalex012@gmail.com טסליצקי מרק

תכנון של מערכות UART

כדי לתכנן UART יש לבצע צעדים הבאים:

- צעד 1: יש להגדיר ולאפשר crossbar (**XBR0** or **XBR2**).
  - נגדיר רגל TXx שתהיה מסוג "push-pull" - **PnMDOUT.n**.
  - נגדיר בתוך crossbar אפשור ל- TXx ו- RXx בתור פינים קלט\פלט חיצוני (**XBR0.2** for UART0 , **XBR2.2** for UART1)
  - בנוסף צריכים לשים "1" לתוך **XBARE (XBR2.6)** כדי לאפשר crossbar.
- צעד 2: יש להגדיר טיימר (או טיימרים) כדי לאפשר תדר מבוקש (baud rate).
  - ניתן להשתמש בטיימר 1 כדי לייצר תדר ל-UART0 ו- UART1.
  - ניתן להשתמש בטיימר 2 כדי לייצר תדר ל-UART0 .
  - ניתן להשתמש בטיימר 4 כדי לייצר תדר ל-UART1 .
- צעד 3: ניתן לאפשר או לנטרל מכפיל תדר (baud rate doubler **SMODx**) ע"י אוגר PCON.
- צעד 4: Step 4: select the serial port operation mode and enable/disable UART reception (**SCONx** register)
- Step 5: enable UART interrupts and set priority (if desired)

www.elecstudy.org markalex012@gmail.com טסליצקי מרק

מצבי עבודה.

- למערכת UART יש 4 אפשרויות עבודה שאפשר לקנפג (להגדיר) בעזרת סיביות **SM0x-SM1x** שנמצאות באוגר **SCONx**.
- שלושה מצבים מאפשרים תקשורת אסינכרונית (מצבים מ-1 עד 3), כאשר מצב רביעי (מצב מס' 0) עובד בתור אוגר הזזה פשוט (סינכרוני).
  - מצב 0 – אוגר הזזה 8 סיביות 8-bit shift register
    - משתמשים בתור רכיב הרכבה (משתמשים בחוצץ חיצוני).
  - מצב 1 - 8 UART סיביות עם קצב עבודה גמיש.
    - מצב נפוץ ביותר.
  - מצב 2 – 9 UART סיביות עם קצב עבודה קבוע.
    - אין צורך בטיימר.
    - יש צורך לבחור בין שעון SYSCLK/32 לבין SYSCLK/64 .
  - מצב 3 – 9 UART סיביות עם קצב עבודה גמיש.
    - משתמשים כאשר דרוש שידור 9 סיביות.

www.elecstudy.org markalex012@gmail.com טסליצקי מרק

SCONx אוגר

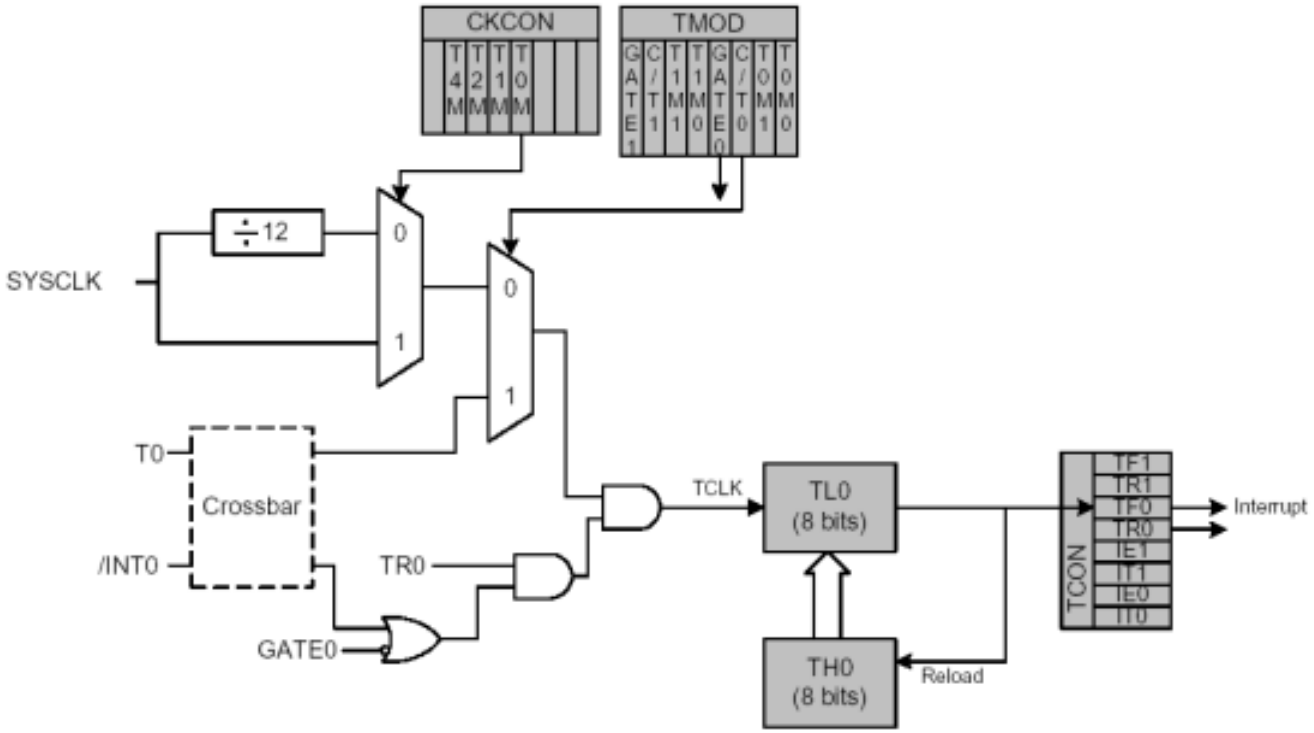
Bit	Symbol	Description
7-6	SM0x-SM1x	*Serial Port Operation Mode* 00: Mode 0: Shift Register Mode **01: Mode 1: 8 Bit UART, Variable Baud Rate** 10: Mode 2: 9 Bit UART, Fixed Baud Rate 11: Mode 3: 9 Bit UART, Variable Baud Rate
5	SM2x	***Multiprocessor Communication Enable*** The function of this bit depends on the Serial Port Operation Mode. Mode 0: No effect. Mode 1: Checks for valid stop bit. 0: Logic level of stop bit is ignored. 1: RIx will only be activated if stop bit is 1 Mode 2 & 3: Multiprocessor Communications Enable. 0: Logic level of 9th bit is ignored. 1: RIx is set and an interrupt is generated only when the 9th bit is 1 and the received address matches the UARTx address or broadcast address.
4	RENx	***Receive Enable*** 0: UARTx reception disabled 1: UARTx reception enabled
3	TB8x	***9th Transmission Bit*** The logic level of this bit will be assigned to the 9th transmission bit in Modes 2 & 3. It is not used in Modes 0 & 1. Set or cleared by software as required.
2	RB8x	***9th Receive Bit*** This bit is assigned the logic level of the 9th bit received in Modes 2 & 3. In Mode 1, if SM2x is 0, RB8x is assigned the logic level of the received stop bit. RB8 is not used in Mode 0.
1	TIx	***Transmit Interrupt Flag*** Set by hardware when a byte of data has been transmitted by UARTx (after the 8th bit in Mode 0, or at the beginning of the stop bits in other modes). When the UARTx interrupt is enabled, setting this bit causes the CPU to vector to the UARTx ISR. This bit must be cleared manually by software.
0	RIx	***Receive Interrupt Flag*** Set by hardware when a byte of data has been received by UARTx (as selected by the SM2x bit). When the UARTx interrupt is enabled, setting this bit causes the CPU to vector to the UARTx ISR. This bit must be cleared manually by software.

PCON— אוגר לבקרת הספק

Bit	Symbol	Description
7	**SMOD0**	***UART0 Baud Rate Doubler Enable*** 0: UART0 baud rate divide-by-two enabled. 1: UART0 baud rate divide-by-two disabled.
6	**SSTAT0**	***UART0 Enhanced Status Mode Select***
5	Reserved	Read is undefined. Must write 0.
4	**SMOD1**	***UART1 Baud Rate Doubler Enable*** 0: UART1 baud rate divide-by-two enabled. 1: UART1 baud rate divide-by-two disabled.
3	**SSTAT1**	***UART1 Enhanced Status Mode Select***
2	Reserved	Read is undefined. Must write 0.
1	**STOP**	***STOP Mode Select*** This bit will always read ‘0’. Writing a ‘1’ will place the microcontroller into STOP mode. (Turns off oscillator).
0	**IDLE**	***IDLE Mode Select*** This bit will always read ‘0’. Writing a ‘1’ will place the microcontroller into IDLE mode. (Shuts off clock to CPU, but clock to Timers, Interrupts, and all peripherals remain active).

שימוש בטיימר 1 כדי לייצר Baud Rate

- ניתן להגדיר baud rate ל- UART0 ו- UART1 בעזרת טיימר 1 במצב 2 (מצב מילוי אוטומטי חוזר 8 סיביות) .



Block diagram of Timer 0 in Mode 2 (8-bit Auto-reload mode)
Timer 1 is identical to Timer 0

www.elecstudy.org markalex012@gmail.com טסליצקי מרק

חישובים של זמנים לטיימר 1.

- ערך שיש להכניס לאוגר TH1 לטעינה מחזורית יש לחשב לפי נוסחה:

$$BaudRate = \left(\frac{2^{SMODx}}{32}\right) \times \left(\frac{SYSCLK \times 12^{(T1M-1)}}{256 - TH1}\right)$$

- במידה ו-**SMODx=1** (חלוקה ב-2).

$$BaudRate = \left(\frac{1}{16}\right) \times \left(\frac{SYSCLK \times 12^{(T1M-1)}}{256 - TH1}\right)$$

www.elecstudy.org markalex012@gmail.com טסליצקי מרק

חישובים של זמנים לטיימר 1. המשך.

- אם T1M=1 (טיימר 1 משתמש בשעון של מערכת לא מחולק ב-12).

$$BaudRate = \left(\frac{1}{16}\right) \times \left(\frac{SYSCLK \times 12^{(1-1)}}{256 - TH1}\right)$$

- אם תדר השעון SYSCLK=22.1184 MHz ותדר השידור Baud Rate 115200 אז:

$$115200 = \left(\frac{1}{16}\right) \times \left(\frac{22118400}{256 - TH1}\right)$$

$$256 - TH1 = \left(\frac{1}{16}\right) \times \left(\frac{22118400}{115200}\right) = 12$$

$$TH1 = 256 - 12 = 244$$

$$TH1 = 0xF4$$

www.elecstudy.org markalex012@gmail.com טסליצקי מרק

תכנות UART בעזרת טיימר 1. דוגמא

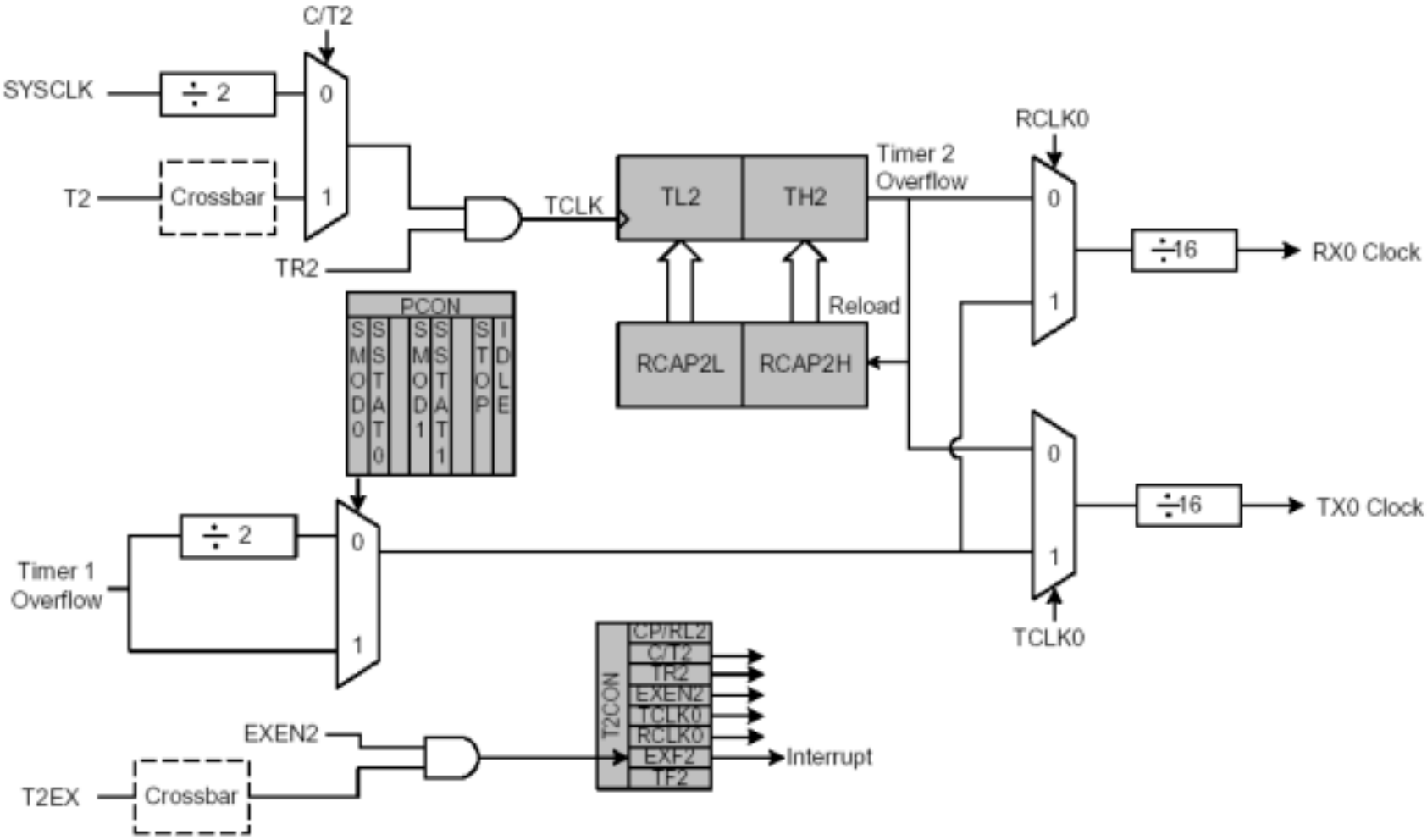
```
void Init_UART0(void)
{
    //-- Set up Timer 1 to generate the baud rate (115200)for UART0 -------
    CKCON |= 0x10;   //-- T1M=1; Timer 1 uses the SYSCLK 22.11845 MHz
    TMOD = 0x20;     //-- Timer 1 in Mode 2 (8-bit auto-reload)
    TH1 = 0xF4;      //-- Baudrate = 115200
    TR1 = 1;         //-- Start Timer 1 (TCON.6 = 1)

    T2CON &= 0xCF;   //-- Timer 1 overflows are used for receive
                     //   and transmit clock. RCLK0=0 and TCLK0=0

    //-- Set up UART0 ----------------------------------------------------
    PCON |= 0x80;    //-- SMOD0=1 (UART0 baud rate divide-by-2 disabled)
    SCON0 = 0x50;    //-- UART0 Mode 1, Logic level of stop bit ignored
                     //   and Receive enabled
    //-- Enable UART0 interrupt ------------------------------------------
    IE |= 0x10;
    IP |= 0x10;      //-- Set to high priority level
    RI0 = 0;         //-- Clear the receive interrupt flag;
                     //   ready to receive more
}
```

שימוש בטיימר 2 לייצור תדר העבודה.

- אם משתמשים בטיימר 2 או טיימר 4 כדי ליצור תדר Baud, צריכים להגדיר אותם במצב 2 (מיולי חוזר).



חישובים של זמנים לטיימר 2. מצב 2

- ניתן לחשב ערך שצריכים להכניס לאוגר **RCAP2** בהתאם לתדר הרצוי לפי משוואה הבאה:

$$BaudRate = \frac{SYSCLK}{32 \times \left(65536 - \left[RCAP2H, RCAP2L\right]\right)}$$

- אם תדר השעון **SYSCLK=22.1184 MHz** ותדר רצוי : **BaudRate=115200**, אז לפי משוואה נקבל חישוב הבא:

$$115200 = \frac{22118400}{32 \times \left(65536 - \left[RCAP2\right]\right)}$$

$$65536 - RCAP2 = \frac{22118400}{32 \times \left(115200\right)} = 6$$

$$RCAP2 = 65536 - 6 = 65530$$

$$RCAP2 = 0xFFFA$$

תכנות **UART** בעזרת טיימר 2. דוגמא

```
void Init_UART0_T2(void)
{
    //-- Set up Timer 2 to generate the Baudrate (115200) for UART0 ---
    CKCON |= 0x20;          //-- T2M=1; Timer 2 uses the SYSCLK 22.11845 MHz
    T2CON = 0x30;           //-- Timer 2 in Mode 2 (Baudrate Generation Mode)
                            //   RCLK0=1 and TCLK0=1
    RCAP2 = 0xFFFA;         //-- Capture Register value for Baudrate = 115200
    TR2 = 1;                //-- Start Timer 2 (T2CON.2 = 1)

    //-- Set up the UART0 ----------------------------------------
    PCON |= 0x80;           //-- SMOD0=1 (UART0 BaudRate divide-by-2 disabled)
    SCON0 = 0x50;           //-- UART0 Mode 1, Logic level of stop bit ignored
                            //   and Receive enabled

    //-- Enable UART0 interrupt ----------------------------------
    IE |= 0x10;
    IP |= 0x10;             //-- Set to high priority level

    RI0 = 0;                //-- Clear the receive interrupt flag;
                            //   ready to receive more
}
```


www.elecstudy.org markalex012@gmail.com טסליצקי מרק

חישובים של זמנים לטיימר 2. במצב שמשתמשים במקור תדר אחר.

- במידה ומשתמשים בשעון אחר ולא SYSCLK, צריכים לשים סיבית C/T2 ב-"1" (באוגר T2CON), וזה מאפשר לקבל פולס שעון מפורט בשם T2.

- במצב זה ניתן לחשב ערך baud של UART לפי נוסחה הבאה:

$$BaudRate = \frac{F_{CLK}}{(65536 - [RCAP2H, RCAP2L]) \times 16}$$

- F_{CLK} זה תדר שמספקים לטיימר 2 ואוגרים RCAP2H, RCAP2L מכילים ערך התחלתי בן 16 סיביות.

www.elecstudy.org markalex012@gmail.com טסליצקי מרק

UARTx דגלי פסיקות לקבלת נתונים.

- בתוך אוגר **SCONx** נמצאים דגלים לשידור וקליטה (**Tix** ו- **Rix**) שיש להם תפקיד חשוב בתהליך תקשורת טורית.
- שתי סיביות האלה "עולות" בעזרת חומרה וצריכות להתאפס בעזרת תוכנה.
- **דגל Rix** – עולה ל-"1" בסיום של קבלת תו ומסמן שחוצץ קבלת נתונים מלא. "receive buffer full"
- תנאי זה נבדק ע"י תוכנה בעזרת שיטת polling או ניתן לבדיקה ע"י פסיקה.
- במידה ואנו רוצים לקבל תו מהתקן שמחובר לפורט טורי (לדוגמא מהתקן RFID) צריכים לכתוב קוד שעושה פעולות הבאות:
  1. ממתינים עד ש- **Rix** עולה ל-1.
  2. מנקים **Rix** .
  3. קוראים תו שקיבלנו מאוגר **SBUFx.**

שימו לב: x = 0 or 1 for UART0 or UART1

www.elecstudy.org markalex012@gmail.com טסליצקי מרק

UARTx דגלי פסיקות לקבלת נתונים.

```
void UART0_ISR(void) interrupt 4
{
    //-- Pending flags RI0 (SCON0.0) and TI0(SCON0.1)
    if ( RI0 == 1)                  //-- Interrupt caused by
    {                               //   received byte
        received_byte = SBUF0; //-- Read the input buffer
        RI0 = 0;                    //-- Clear the flag
        new_cmd_received=1;
    }

    if ( TI0 == 1)                  //-- Interrupt caused by
    {                               //   transmitted byte
        TI0 = 0;                    //-- Clear the flag
    }
}
```


www.elecstudy.org markalex012@gmail.com טסליצקי מרק

UARTx דגלי פסיקות לשליחת נתונים.

- **סיבית Tix** עולה ל-"1" בסיום טו שידור ומראה שחוצץ לשליחת נתונים ריק. "transmit buffer empty"

- אם קוד רוצה לשלוח תו להתקן שמחובר לפורט טורי, הוא צריך לבדוק קודם כל שפורט מוגדר וקיים.

- אם תו הקודם נשלח אנו צריכים לחקות עד ששידור יסתיים לפני שליחת תו הבא.

www.elecstudy.org markalex012@gmail.com טסליצקי מרק

UARTx דגלי פסיקות לשליחת נתונים.

```
void Init_UART0(void)
{
    //-- Set up Timer 1 to generate the baud rate (115200) for UART0
    CKCON |= 0x10;                  //-- T1M=1; Timer 1 uses the
                                    //   system clock 22.11845 MHz
    TMOD = 0x20;                    //-- Timer 1 in Mode 2 (8-bit auto-reload)
    TH1 = 0xF4;                     //-- Baudrate = 115200
    TR1 = 1;                        //-- Start Timer 1 (TCON.6 = 1)
    T2CON &= 0xCF;                  //-- Timer 1 overflows are used for receive
                                    //   and transmit clock. RCLK0=0 and TCLK0=0

    //-- Set up the UART0
    PCON |= 0x80;                   //-- SMOD0=1 (UART0 baud rate divide-by-2
                                    //   disabled)
    SCON0 = 0x50;                   //-- UART0 Mode 1, Logic level of stop bit
                                    //   ignored and Receive enabled

    //-- Enable UART0 interrupt
    IE |= 0x10;

    RI0 = 0;                        //-- Clear the receive interrupt flag;
                                    //   Ready to receive more
    TI0 = 1;                        //-- TX0 ready to transmit
}
```

UARTx דגלי פסיקות לשליחת נתונים.

- תחילת שידור ע"י כתיבת תו לאוגר SBUFx.
- מעלים סיבית Tix ל-"1" בתוך אוגר SCONx **בתחילת סיבית עצירה.**
- **חייבים לנקות סיבית Tix ("0") באופן ידני ע"י תוכנה.**

```
int  i,n;
char sendbuf[20];              //-- Buffer to hold string for
                               //   transmission
n = sprintf(sendbuf, "Hello! %c", '\0');
for (i=0; i<n; i++)
{
    while (TI0 == 0);          //-- Wait while the transmission is
                               //   going on
    TI0 = 0;                   //-- Clear TI0
    SBUF0 = sendbuf[i];        //-- Load the serial buffer
                               //   with the char to send
}
```

www.silabs.com/MCU