התקשורת הטורית - UART - בפיקו R2040

1. מבוא קצר לתקשורת

תקשורת טורית היא תהליך של שליחת מידע של ביט אחד אחרי השני . ביט אחד בכל פרק זמן נתון. הביטים מועברים ברצף אחד אחרי השני עד שמסתיים הביית ולאחריו מתחיל שידור הביטים של הביית הבא וכך הלאה עד לסיום ההודעה הרצויה.

התקשורת הטורית משמשת במקרים שצריך תקשורת לטווחים ארוכים או ברשתות מחשבים שבהם עלות הכבלים וקשיי הסנכרון של תקשורת מקבילית אינם מעשיים וגורמים העדפה של תקשורת טורית. היום מאד נפוץ להעביר נתונים בצורה טורית אפילו במרחקים קצרים.

ישנם מספר סוגי תקשורת טורית נפוצים:

א. תקשורת טורית "רגילה" הנקראת גם UART – Universal Asynchronous Receiver Transmitter - משדר מקלט אסינכרוני אוניברסלי.

ב. תקשורת SPI (Serial Peripheral Interface – ממשק טורי היקפי).

ג. I2C - (Inter-Integrated Circuit - מעגל בין משולב)הנקראת גם i^2c וגם IIC .

ד. I2S - Inter Ic Sound – שהוא תקשורת טורית לשמע דיגיטאלי.

ה. One wire - תקשורת שתוכננה על ידי חברת דאלאס Dallas , הנותנת מהירות נמוכה (16.3 Kbps) . התקשורת היא על קו אחד . היא דומה ל I2C אבל בקצב נמוך יותר ובטווח גדול יותר. משמשת בדרך כלל לתקשורת עם רכיבים לא יקרים כמו טרמומטר דיגיטאלי ומכשירי מזג אוויר. כאשר בתקשורת יש מספר רכיבים המתחברים אל רכיב מסטר היא נקראת MicroLAN .

השוואה בין סוגי התקשורת הטורית ניתן למצוא בקישור:

http://www.arikporat.com/arduino1/serial%20communication%20comparison.pdf

2. תקשורת טורית "רגילה" - UART

א. תקשורת טורית רגילה/סטנדרטית היא תקשורת שבה מתחברים רכיבים עם הדק TXD של צד אחד המתחבר להדק RXD של השני והדק TXD של השני מתחבר להדק RXD של הראשון (חיבור כזה נקרא חיבור מוצלב). בתוך המיקרו בקר יש מעגל חומרה הנקרא UART – Universal A Synchronous Receiver Transmitter – משדר מקלט א-סינכרוני אוניברסלי המבצע את השידור והקליטה הטוריים.

ב. בצורה כזו ה UART מבצע את פעולת השידור והקליטה של הנתונים והמיקרו פנוי לטפל בדברים נוספים ולא צריך לנהל את השידור או הקליטה הטורית.

ג. התקשורת הטורית הרגילה נקראת UART עם רמות מתח של TTL (0 ו 5 וולט). ניתן למצוא גם את השם RS232 לתקשורת הזו אם כי שם זה לא מדויק. ה RS232 הוא תקשורת טורית "רגילה" אבל עם רמות מתח שונות (0 לוגי יכול להיות מתח של בין 7 וולט ל 15 וולט ו 1 לוגי יכול להיות מינוס 7וולט עד מינוס 15 וולט. כאשר רמת ה '0' גבוהה מרמת ה '1' - לוגיקה שלילית). מתחים אלו מאפשרים שידור לטווח גדול יותר מאשר 0 ו 5 וולט .

ד. התקשורת הטורית עם UART היא תקשורת א-סינכרונית. כלומר אין שעון מערכת System Clock או שעון ראשי המסנכרן את התקשורת.

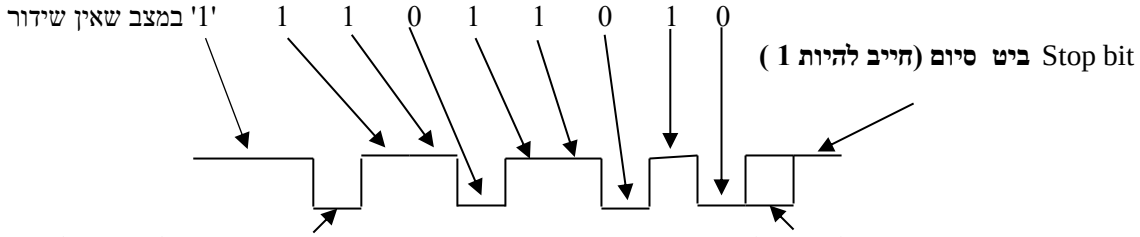
ה. בתקשורת טורית כזו ניתן גם לשדר וגם לקלוט בו זמנית. זה נקרא **Full Duplex** .

3. כיצד שולחים נתון בקו תקשורת טורית ?

א. מערכת תקשורת טורית רגילה – UART - יודעת לקבל ביית מהמיקרו ולשדר אותו בהדק TxD של המיקרו, כמקובל בתקשורת טורית : קודם את **ביט ההתחלה – Start Bit** , אח"כ את סיביות הנתון ולבסוף את ביט הזוגיות - **PARITY** (אם הוחלט שעובדים עם זוגיות) ואת **ביט הסיום Stop Bit** . בסיום שידור התו ה UART מודיע למיקרו (גם בעזרת פסיקה וגם על ידי דגל) על סיום שידור התו. השידור מתבצע בקצב שהמתכנת קבע ל UART .

ב. בקליטה טורית קורה תהליך הפוך. המיקרו מקבל בהדק RxD של המיקרו את האות הטורי. ה UART מזהה את ביט ההתחלה, את הביטים של הנתון , את הביט של הזוגיות (אם הוחלט לעבוד עם זוגיות) ואת ביט הסיום ומודיע למיקרו על ידי פסיקה וגם על ידי דגל על סיום קליטה טורית.

ג. נניח שרוצים לשדר את התו 5BH כלומר 01011011 בינארי. התו משודר **מהביט הנמוך אל הגבוה (מה LSB אל ה MSB).**



איור מספר 1 : שידור תו טורי

ד. כמות הביטים הנשלחת בשנייה נקראת **קצב התקשורת או קצב הביטים**. היחידות הן **סיביות לשנייה או סל"ש. באנגלית bps – Bits Per Second** . קצבי שידור מקובלים בתקשורת טורית : 4800 ,9600 ו 19200 ביטים בשנייה. קיים גם מושג שנקרא **באוד – BAUD** , על שם מהנדס צרפתי שהמציא את קוד הטלטייפ בן 5 הביטים. המושג מתייחס למספר שינויי האות או הסמל המיוצרים בשנייה אחת. סמל הוא אחד משינויי המתח (ניתן לשלוח על הקו 0 וולט והוא יסומן 00 , 1 וולט שיסומן 01 , 2 וולט יסומן 10 ו 3 וולט יסומן 11 ואז הקצב גבוה יותר). אצלנו יש ל '0' מתח של כ 0 וולט ול '1' יש מתח של כ 5 וולט ולכן קצב באוד וקצב הביטים שווה.

ה. ביט הזוגיות הוא ביט שמציין את כמות ה '1' (האחדים) שיש בנתון. בתחילת ימי התקשורת הטורית היו משתמשים בביט הזוגיות כדי שהצד הקולט יידע האם הוא קלט את המידע נכון. קיימים 2 סוגי זוגיות : **א. זוגיות זוגית – Even Parity ב. זוגיות אי זוגית Odd Parity**. בזוגיות זוגית כמות ה '1' של הנתון פלוס ביט הזוגיות צריך להיות זוגי. בזוגיות אי זוגית כמות ה '1' בנתון פלוס ביט הזוגיות הוא אי זוגי. לדוגמא : נניח שמשדרים את התו 5bH ועובדים בזוגיות אי-זוגית. בתו יש 5 פעמים 1 . היות וכמות ה 1 איננה זוגית אז בביט הזוגיות נשלח '0' . הצד הקולט סופר את כמות ה 1 שקלט. היות והוא קלט 5 פעמים 1 הוא יודע שביט הזוגיות שהוא

צריך לקלוט הוא '0'. אם אחד הביטים היה מתהפך בדרך אז הצד הקולט היה קולט מספר זוגי של 1 ואז היה רואה שביט הזוגיות הוא 0 והיה מבחין שקלט נתון שגוי. במקרה כזה הוא יכול לבקש מהצד המשדר לשלוח את הנתון פעם נוספת. הוספת ביט הזוגיות גורם לביט טורי תשיעי בנוסף ל 8 הביטים של המידע, דבר שמאט את קצב התקשורת הטורית, הנמוך ממילא. היום יש שיטות חדשות לאיתור שגיאות ואפילו לתיקון שגיאות.

4. תקשורת UART ב RP2040 ובכרטיס ראספברי פיי פיקו

ב RP2040 יש שני UARTs שנקראים UART0 ו UART1.

ניתן למפות את UART0 להדקים - GPIO 0/1 (רגליים 1 ו2 בכרטיס) , ל GPIO 12/13 (רגליים 16 ו 17 בכרטיס) ו 16/17 GPIO (רגלים 21 ו 22 בכרטיס).

את UART1 ניתן למפות ל GPIO 4/5 (רגלים 6 ו 7 בכרטיס) ו GPIO 8/9 (רגלים 11 ו 12 בכרטיס) .

לא נרחיב לגבי הרגיסטרים בתוך המיקרו בקר. ניתן לקרא עליהם בקישור :

5. המחלקה UART במיקרו פייתון

המחלקה UART מיישמת את פרוטוקול התקשורת הטורית הסטנדרטי UART/USART. ברמה הפיזית הוא מורכב משני קווים: RX ו- TX. יחידת התקשורת היא תו (לא להתבלבל עם תו מחרוזת) שיכול להיות ברוחב 8 או 9 סיביות.

ניתן ליצור ולאתחל אובייקטים של UART באמצעות השורות הבאות :

```
from machine import UART

uart = UART(1, 9600)                         # יצירת האובייקט עם uart1 בקצב 9600 ביטים לשנייה

uart.init(9600, bits=8, parity=None, stop=1) # אתחול עם הפרמטרים הרצויים
```

הפרמטרים הנתמכים שונים בלוח (כרטיס המיקרו בקר) . בכרטיסי PII הסיביות יכולות להיות 7, 8 או 9. העצירה יכולה להיות 1 או 2. כשעובדים ללא זוגיות ניתן לעבוד רק 8 או 9 סיביות . כאשר עובדים עם זוגיות ניתן לעבוד רק עם 7 או 8 סיביות.

אובייקט UART פועל כמו אובייקט stream וקריאה וכתיבה מתבצעות במתודות ה stream הסטנדרטיות.

```
uart.read(10)         # קריאה של 10 תווים
uart.read( )          # קורא את כל התווים שהתקבלו
uart.readline( )      # קריאה של שורה
uart.readinto(buf)    # קריאה ושמירה בתוך מחרוזת
uart.write('abc')     # כתיבה של 3 תווים
```

5.א הבנאי - constructor

***class*machine.UART(*id, ...*)**

בנה אובייקט UART של ה id הנתון. המספר 0 או 1 לפי ה UART הרצוי.

5.ב המתודות במחלקה

הדוגמאות לכל המתודות בסוף הפרק.

1.ב.5 אתחול

UART.init(baudrate=9600, bits=8, parity=None, stop=1, *, ...)

אתחל את אפיק UART עם הפרמטרים הנתונים:

1. *baudrate* הוא קצב השעון (כמות הביטים לשנייה).
2. *bits* הוא מספר הסיביות לכל תו : 7, 8 או 9.
3. *Parity* היא זוגיות : None ללא זוגיות , 0 (זוגי) או 1 (אי זוגי).
4. *Stop* מספר סיביות העצירה : 1 או 2.

פרמטרים נוספים למילת מפתח בלבד שעשויים להיות נתמכים על-ידי יציאה הם:

- *tx* מציין את הדק ה- TX לשימוש.
- *rx* מציין את הדק RX לשימוש.
- rts מציין את פין RTS (פלט) לשימוש עבור בקרת זרימה של קבלת חומרה.
- *cts* מציין את סיכת CTS (קלט) לשימוש עבור בקרת זרימת שידור חומרה.
- *txbuf* מציין את האורך בתווים של מאגר ה- TX.
- *rxbuf* מציין את האורך בתווים של מאגר RX.
- *timeout* מציין את הזמן להמתנה לתו הראשון (ב- ms).
- *timeout_char* מציין את זמן ההמתנה בין תווים (ב- ms).
- *invert* מציין אילו שורות הפוך.
- *flow* מציין באיזה אותות בקרת זרימת חומרה להשתמש. הערך הוא מסיכת סיביות.

0 - יתעלם מאותות בקרת זרימת חומרה.

UART.RTS יאפשר בקרת זרימה של קבלה באמצעות פין הפלט של RTS כדי לאותת אם ל- FIFO של הקליטה יש מספיק מקום כדי לקבל נתונים נוספים.

UART.CTS יאפשר בקרת זרימת שידור על-ידי הפסקת השידור כאשר פין הקלט של CTS מאותת ששטח החוצץ של המקלט אוזל.

UART.RTS | UART.CTS יהפוך את שניהם לזמינים עבור בקרת זרימת חומרה מלאה.

2.ב.5 סיום פעולת ה UART

UART.deinit()

כבה את ערוץ ה UART .

3.ב.5

UART.any()

החזרת מספר שלם שסופר את מספר התווים שניתן לקרוא מבלי לחסום. הוא יחזיר 0 אם אין תווים זמינים ומספר חיובי אם קיימים תווים. השיטה עלולה להחזיר 1 גם אם יש יותר מתו אחד זמין לקריאה.

לקבלת שאילתות מתוחכמות יותר של תווים זמינים, השתמש ב- select.poll:

```
poll = select.poll()
poll.register(uart, select.POLLIN)
poll.poll(timeout)
```

4.ב.5 קריאה של כמות בתים רצויה

UART.read([nbytes])

קריאה של תווים. אם צוין **nbytes** (מספר התווים המקסימלי שרוצים לקרא) אז קרא לכל היותר את כמות הבתים . אם אין **nbytes** אז קרא נתונים רבים ככל האפשר. ייתכן שלא נספיק לקרא את כל הבתים ונחזור מוקדם יותר אם יגיע פסק הזמן timeout - שהוא הזמן הקצוב וניתן להגדרה בבנאי.

הערך המוחזר : אובייקט בתים המכיל את הבתים שנקראו בו. החזרת None אם היה timeout.

5.ב.5 קריאה של כמות בתים רצויה לתוך חוצץ

UART.readinto(buf, [nbytes])

קרא בתים לתוך buf (שם של מחרוזת). nbytes הוא אופציונאלי. אם הוא רשום אז הוא מראה את כמות הבתים המקסימאלית שיש לקלוט ולהעביר ל buf - למחרוזת . אם nbytes לא רשום אז קוראים את כל הבתים שאפשר עד לאורך של ה buf . אם מגיע timeout (הזמן הקצוב) אז מסיימים את הקליטה עוד לפני שקלטנו את כמות הבתים הרשומה. ה timeout - הזמן הקצוב - ניתן להגדרה בבנאי.

הערך המוחזר: מספר הבתים שנקראו ומאוחסנים ב buf או None אם יש timeout .

6.ב.5 קריאה של שורה

UART.readline( )

קרא תווים עד לתו '\n' שורה חדשה newline. אפשר לחזור מוקדם יותר אם מגיע timeout שניתן להגדרה בבנאי.

הערך המוחזר: השורה שנקראה או None אם היה timeout.

7.ב.5 כתיבה של מחרוזת

UART.write(buf)

כתיבה- שידור של הבתים שב buf – מחרוזת .

הערך המוחזר: מספר הבתים שנכתבו או None אם היה timeout .

8.ב.5 שליחה של הודעת הפסקה

UART.sendbreak( )

שלח מצב "הפסקה" בקו. פעולה זו דוחפת את קו השידור TX למצב נמוך למשך זמן ארוך מהנדרש עבור שידור רגיל של תו.

9.ב.5 יצירת פסיקה כאשר יש קליטה (זמין בכרטיסי WiPy של Espressif ESP32)

UART.irq(trigger, priority = 1, handler = None, wake = machine.IDLE)

- **trigger** - יכול להיות רק UART.RX_ANY
- **priority** - הרמה של עדיפות הפסיקה . יכולה לקבל ערך בין 1 ל 7 . מספר גבוה יותר – עדיפות גבוהה יותר.
- **handler** - פונקציה אופציונאלית שתופעל כאשר תווים חדשים מגיעים.
- **wake** – יכול להיות רק machine.IDLE .

הערה : הפסיקה תבוצע כאשר מתמלאים אחד מ 2 התנאים הבאים:

- נקלטו 8 תווים חדשים .
- לפחות תו חדש מחכה בחוצץ של Rx וקו ה Rx היה שקט למשך מסגרת אחת מלאה.

משמעות הדבר היא שכאשר עוברים לפונקציית הפסיקה יש בין 1 ל 8 תווים המחכים בחוצץ של Rx .

הפונקציה מחזירה אובייקט irq .

6. דוגמאות של שליחת נתונים מפורט UART0 לפורט UART1.

6.א חומרה:

נחבר 2 פורטים של הפיקו יחד. לשם כך נחבר בעזרת מגשר קצר בין UART0 TX אל UART1 RX . בכרטיס הפיקו יש לקצר את הרגליים : רגל 1 - GP0 אל רגל 12 - GP9 .

6.ב תוכנה:

מבוא: על str ו bytes

בדוגמאות שבהמשך נשדר ונקלוט מחרוזות ובתים. בפייתון 3.x יש הבחנה ברורה בין 2 הטיפוסים האלו:

str - ליטרלים המופיעים בין גרשים או גרשיים ('....' או ".....") הם רצף של תווי Unicode

bytes - ליטרלים המופיעים עם התו b כמו b'....' הם רצף של שמיניות בינאריות (מספרים שלמים בין 0 ל 255).

1.ב.6 אתחול 2 ערוצי ה UART :

```
from machine import UART, Pin
```

```
import utime
# GP1 והדק הקליטה GP0 אתחול uart0 לקצב תקשורת של 9600 ביטים בשנייה. הדק השידור
uart0 = UART(0, baudrate=9600, tx=Pin(0), rx=Pin(1))
# GP9 והדק הקליטה GP8 אתחול uart1 לקצב תקשורת של 9600 ביטים בשנייה. הדק השידור
uart1=  UART(1, baudrate=9600, tx=Pin(8), rx=Pin(9))
```

2.ב.6 שידור וקליטה של מחרוזת תווי אסקי

```
#----  שידור וקליטה של מחרוזת בת 5 תווי אסקי -----
uart0.write('Hello')  # שידור של מחרוזת בת 5 בתים בקוד אסקי
utime.sleep(0.01)
x=uart1.read(5)       # קליטה של 5 בתי המחרוזת
print(x)
```

ההדפסה שנקבל : b'Hello'

3.ב.6 שידור וקליטה של כל התווים המשודרים :

```
uart0.write('Hello MicroPython World')  # שידור של מחרוזת בתים באסקי
utime.sleep(0.01)
x=uart1.read()       # קליטה של כל הבתים
print(x)
```

ההדפסה שנקבל : b'Hello MicroPython World'

4.ב.6 שידור וקליטה של 2 בתים :

```
# --- בתים 2 של מחרוזת של וקליטה שידור -----
uart0.write(b'\x98\xa7')  # שידור של 2 בתים
utime.sleep(0.01)
x=uart1.read(2)       # קליטה של 2 בתים
print(x)
```

ההדפסה שנקבל : b'\x98\xa7'

5.ב.6 שידור וקליטה של 8 בתים :

```
a = b'\xff\xf8\xee\xdc\xcb\xba\x7f\x00'
uart0.write(a)
utime.sleep(0.01)
x=uart1.read(8)       # קליטת 8 בתים
```

```
print(x)
```

ההדפסה שנקבל : b'\xff\xf8\xee\xdc\xcb\xba\x7f\x00'

6.ב.6 הדפסת כמות התווים שנקלטו :

```
a = b'\x65\x66\x67\x68\x69\x70\x71\x72'
uart0.write(a)
utime.sleep(0.01)
הדפסת מספר התווים שנקלטו #
print("Number of characters that were received :",uart1.any())
```

ההדפסה שנקבל : Number of characters that were received : 8

7.ב.6 שידור מ uart1 אל uart0 והדפסת המחרוזת הנקלטת

```
txData = b'hello world\nHi everyone'
uart1.write(txData)
utime.sleep(0.1)
rxData = bytes()
while uart0.any() > 0:
    rxData += uart0.read(1)
print(rxData.decode('utf-8'))
```

ההדפסה שנקבל :

```
hello world
Hi everyone
```

8.ב.6 הדפסה בעברית

```
txData = 'אריק פורת'
uart1.write(txData)
utime.sleep(0.1)
rxData = bytes()
while uart0.any() > 0:
    rxData += uart0.read(1)
print(rxData)
print(rxData.decode('utf-8'))
```

ההדפסה שנקבל:

```
b'\xd7\x90\xd7\xa8\xd7\x99\xd7\xa7 \xd7\xa4\xd7\x95\xd7\xa8\xd7\xaa'
אריק פורת
```

9.ב.6 הדפסה של שורות

```
txData= 'printing line after line\nHello MicroPython World\nHi everyone'
uart1.write(txData)
utime.sleep(0.1)
rxData = bytes()
while uart0.any() > 0:
    rxData=uart0.readline( )
    print(rxData.decode('utf-8'))
```

ההדפסה שנקבל:

```
printing line after line

Hello MicroPython World

Hi everyone
```