**פסיקות**

1. **פסיקה – הסבר כללי**

המילה interrupt באנגלית היא "הפרעה" בעברית. דוגמא לפסיקה בחיי היום יום היא המצב שאנחנו עסוקים בקריאת ספר וצלצול הטלפון מפריע לנו . הטיפול בהפרעה יתבצע כך : **א.** נשים סימניה בשורה אליה הגענו בספר. **ב**. נענה לטלפון ונטפל בהפרעה. **ג**. בסיום הטיפול נחזור לקרוא את הספר , ובעזרת הסימנייה , נחזור בדיוק לשורה שבה היינו לפני שענינו לטלפון.

הפסיקה במערכות מחשבים מתארת מצב שבו המחשב מבצע תכנית מסוימת ובזמן כלשהו , אחת המערכות או הרכיבים המתחברים למחשב מפריעות למהלך התוכנית ומבקשות טיפול. על המחשב לטפל במערכת/רכיב שהפריעו וביקשו טיפול ובסיום הטיפול צריך לחזור לאותה הפקודה בתוכנית שהמחשב היה.

**הערה חשובה**: פסיקה –הפרעה – נחשבת בחיי היום יום כפעולה שלילית. במעבדים ומיקרו בקרים היא דרך עבודה מקובלת ולא מתייחסים אליה בצורה שלילית . התקני הקלט והפלט במחשב האישי מתחברים בעזרת פסיקות. במחשב נקראות בקשות הפסיקה IRQ קיצור של Interrupt ReQuest או דרישת (בקשת) פסיקה.

מתכנני המעבדים והמיקרו בקרים התייחסו למעבדים ולמיקרו בקרים בהקבלה למנהלים במפעל. במחשבתם הם תיארו 2 סוגי מנהלים :

1. **מנהל העובד עם פסיקות**.

זהו מנהל אשר יושב במשרדו ועושה את עבודתו. נניח שהמנהל קורא מסמך המתאר את מצבה הפיננסי של החברה. אחד העובדים רוצה לדבר או להתייעץ עם המנהל. העובד בא לחדרו של המנהל , דופק על הדלת ומפריע למנהל בעבודתו . תהליך הטיפול של המנהל בהפרעה יהיה לפי השלבים הבאים :

1. המנהל מסיים לקרא את המשפט האחרון שאליו הגיע .
2. המנהל מסמן בצורה כלשהי (אולי בעזרת סימנייה ...) את הנקודה שבה הפסיק את עבודתו, כלומר הוא מסמן את המשפט הבא במכתב אותו יצטרך לקרא אחרי הטיפול בהפרעה.
3. המנהל מטפל בעובד המפריע.
4. בסיום הטיפול בעובד , המנהל חוזר למכתב שקרא , לאותו המשפט שצריך להמשיך ממנו בעזרת הסימון שעשה מקודם.
5. **מנהל העובד עם שאילתות Polling .**

מנהל זה איננו עובד עם פסיקות . מנהל זה איננו מוכן לקבל הפרעות. הוא עושה את עבודתו וכל פרק זמן מסוים הוא עוצר את העבודה ובודק האם מישהו חיפש אותו. במידה וכן הוא מטפל בעובד שחיפש אותו. במידה ולא הוא ממשיך בעבודתו.

כמובן שיש מנהלים המשלבים בין 2 השיטות . גם במיקרו ניתן לשלב בין 2 השיטות.

1. **חסימה / אפשור ועדיפות**

מנהל העובד עם פסיקות, יכול לקבוע את האפשרויות הבאות :

1. כאשר הוא עסוק בנושא מהותי , הוא יכול לקבוע שהוא איננו נענה להפרעות. אנשים יכולים לדפוק על הדלת , לצלצל אליו לטלפון או לנסות כל פעולה אחרת. המנהל נועל את הדלת ומנתק את הטלפון. הוא איננו מטפל באף הפרעה. בצורה כזו הוא חסם את כל הפסיקות.
2. המנהל יכול לקבוע באיזו הפרעה הוא מוכן לטפל ואיזו לא. לדוגמא : אם מדובר באחד המנהלים הבכירים הוא מוכן לטפל ואם מדובר בעובד זוטר הוא איננו מוכן לטפל בו כרגע. למעשה הוא קובע איזו הפרעה הוא מאפשר ואיזו הוא חוסם.
3. המנהל גם יכול לקבוע סדר עדיפות – Priority . במידה ומגיעות מספר הפרעות יחד – כמה עובדים מגיעים יחד – המנהל קובע באיזו הפרעה הוא יטפל ראשונה , מי אחריה וכך הלאה. כיצד יכולים להגיע כמה הפרעות בו זמנית ? ניקח דוגמא שבה המנהל חסם את כל ההפרעות והגיעו מספר עובדים וכולם רוצים טיפול והם ממתינים ליד חדר המנהל. ברגע שהמנהל יאפשר פסיקות הוא רואה שיש מספר פסיקות בו זמנית והוא יטפל בהן לפי הסדר שקבע מראש.

נתאר בעזרת טבלה מספר 1 תהליך הענות לפסיקה במיקרו בקר. תיארנו מקודם את השלבים בטיפול בפסיקה של מנהל מפעל. נראה את הפעולות המקבילות במיקרו :

נצא מתוך ההנחות הבאות :

המנהל קורא את משפט מספר 100 במסמך. המשפט הבא הוא משפט מספר 101 .

המיקרו מבצע פקודה בכתובת 100 בתוכנית. הפקודה הבאה נמצאת בכתובת 101 .

|  |  |  |  |
| --- | --- | --- | --- |
| **השלב בהיענות לפסיקה** | **מנהל המפעל** | **מיקרו בקר** | **הערות** |
| 1 | המנהל מסיים את המשפט שבו הוא נמצא | המיקרו מסיים את הפקודה שבה נמצא | המנהל מסיים לקרא את משפט 100 והמיקרו מסיים לבצע את הפקודה שבכתובת 100  |
| 2 | המנהל שומר בעזרת סימנייה את מיקום המשפט אליו הגיע.  | המיקרו שומר בזיכרון במקום שנקרא "מחסנית" את הכתובת של הפקודה אליה הגיע ("כתובת החזרה" ). | המנהל שם את הסימנייה על כתובת 101 . המיקרו מכניס למחסנית את כתובת 101 . |
| 3 | המנהל מכניס את העובד ומטפל בו. בזמן הזה הוא איננו מקבל פסיקות נוספות ומקדיש את עצמו לעובד בלבד. | המיקרו עובר לתוכנית שמטפלת במפריע. המיקרו שם בביט EA ברגיסטר IE את הערך 0 כדי לחסום את כל הפסיקות ולא לקבל פסיקות נוספות. |  |
| 4 | בסיום הטיפול הוא אומר לעובד "שלום" וחוזר בעזרת הסימנייה אל המשפט אליו הגיע. עכשיו הוא יכול לקבל פסיקות נוספות. | בסיום תוכנית הטיפול מופיעה פקודת reti (קיצור של RETurn from Interrupt- חזור מפסיקה ) . המיקרו מושך מהמחסנית את כתובת החזרה וחוזר אל הפקודה אליו הגיע. הוא מחזיר לביט EA את הערך 1 כדי לאפשר קבלת פסיקות נוספות. | המנהל חוזר למשפט מספר 101 (בעזרת הסימנייה ) וממשיך לקרא את המסמך ממשפט זה. המיקרו מושך מהמחסנית את כתובת 101 וממשיך את התוכנית מכתובת זו. |

טבלה מספר 1 – תהליך השוואה בין הענות לבקשת פסיקה במיקרו לבקשת טיפול של עובד

1. **שימוש בפסיקות**

הפסיקות מועילות כאשר רוצים שפעולות יתבצעו בצורה אוטומטית במיקרו בקר, והן עוזרות לפתור בעיות של תזמון. דוגמאות מתאימות לפסיקות יכולות להיות קריאת rotary encoder למדידת מהירות סיבוב של מנוע או ניטור (פיקוח) על כניסות ממקלדת או מפסקים. במקום לדגום באופן קבוע את כניסות הrotary encoder או את כניסות המפסקים, המיקרו חופשי לטפל בנושאים אחרים וכאשר ה rotary encoder או המפסקים יילחצו נקבל פולס שיפעיל פסיקה ונעבור לתכנית שמטפלת בהם.

פונקציות שמטפלות בפסיקה נקראות ISR – Interrupt Service Routine -. הן לא מקבלות פרמטרים והן אינן מחזירות ערך. רצוי שפונקציה כזו תהיה קצרה ומהירה ככל האפשר. כאשר פונקציית כלשהי רצה , שאר בקשות הפסיקה אינן נענות עד שהפסיקה הנוכחית מסתיימת. הפונקציות delay() ו millis() נעזרות בפסיקה ולכן הן לא יפעלו כאשר פסיקה מסוימת רצה !! . לעומתן הפונקציה delayMicroseconds() לא עובדת עם פסיקה וניתן לקרא לה בתוך פונקציית פסיקה. בדרך כלל משתמשים במשתנים גלובאליים כדי לבצע העברת נתונים בין פונקציה המטפלת בפסיקה והפונקציות האחרות. כדי לוודא שמשתנים יעודכנו נכון בתכנית פסיקה כדאי להגדיר אותם כ **volatile** . המושג volatile אומר שערך המשתנה יכול להשתנות בפונקציה אחרת.

לדוגמא :

volatile int x; // הגדרת משתנה גלובאלי

void func1( ) // הגדרת פונקציה

{

 x=0;

while(x<1); // 0 שווה x לולאה החוזרת על עצמה כל עוד

// 0 איננו x משפטים לביצוע כאשר המשתנה

…..

}

קומפיילר "חכם" לא יתרגם את התכנית ( או ייתן אזהרה או יגיד שהאופטימייזר שלו שינה את הפקודה ) כי x=0 ומיד לאחר מכן נכנסים כאילו ללולאה אין סופית כי לא רואים איפה x יכול להשתנות. המילה volatile אומרת לקומפיילר שהמשתנה x יכול להשתנות במקום אחר כמו תוכנית פסיקה שבה נרשום נרשום את השורה x=1; . לתכנית הפסיקה נגיע על ידי לחיצה על מפסק.

**כיצד כותבים ומפעילים פונקציית פסיקה ?**

* 1. כותבים פונקציה "רגילה" רק שהפונקציה איננה מקבלת ערכים ואיננה מחזירה ערך , כלומר רושמים :

void שם הפונקציה ( )

{

 // שיתבצע שנקבל את הפסיקה תכנית המבצעת מה שרוצים

}

* 1. מקשרים בתוכנה בעזרת הפונקציה attachInterrupt() את הפסיקה המתאימה לפונקציה שרשמנו.
1. **הפונקציה attachInterrupt( )**

הפונקציה מציינת שם של ISR שתופעל כאשר תקרה פסיקה. היא מחליפה כל פונקציה קודמת שהייתה קשורה לפסיקה. לרוב כרטיסי הארדואינו יש 2 פסיקות חיצוניות, מספר 0 ( בהדק דיגיטאלי 2 ) ומספר 1 (בהדק דיגיטאלי 3). הטבלה הבאה מראה את הפסיקות בלוחות הארדואינו השונים.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Board** | **int.0** | **int.1** | **int.2** | **int.3** | **int.4** | **int.5** |
| **Uno, Ethernet** | 2 | 3 |   |   |   |   |
| **Mega2560** | 2 | 3 | 21 | 20 | 19 | 18 |
| **Leonardo** | 3 | 2 | 0 | 1 | 7 |   |
| **Due** | ראה הסבר בהמשך |

ללוח ארדואינו Due יש יכולות פסיקה המאפשרות לקשר פונקציות פסיקה לכל הדק אפשרי. ניתן לציין את מספר ההדק בהגדרת הפונקציה attachInterrupt( ) .

הערה : בתוך הפונקציה המקושרת לפסיקה לא תעבוד הפונקציה delay( ) והערך המוחזר מהפונקציה millis ( ) לא יתקדם. נתון טורי הנקלט בזמן הפונקציה יכול ללכת לאיבוד. יש להגדיר את כל המשתנים שמשנים בפונקצית הפסיקה כ volatile .

**התחביר** :

|  |
| --- |
| attachInterrupt(interrupt, ISR, mode)  |
| attachInterrupt(pin, ISR, mode) // Due לארדואינו  |   |

**interrupt** - מספר הפסיקה . **ISR** – הפונקצייה ( ISR ) שתתבצע בזמן הפסיקה. ( הפונקציה לא תקבל ערכים ולא תחזיר ערך !) . **mode** - מגדירה את אופן ההפעלה – התנעה - של הפסיקה . יש 4 אופנים : א. **LOW** - הפסיקה מופעלת על ידי 0 . ב. **CHANGE** - הפסיקה מופעלת כאשר יש שינוי בהדק הפסיקה ( מ 0 ל 1 או מ 1 ל 0 ). ג. **RISING** - כאשר בהדק יש מעבר מ 0 ל 1 . ד. **FALLING** - כשבהדק יש מעבר מ 1 ל 0 . (הערה : בארדואינו Due בלבד יש גם מצב של HIGH ). **pin** - רק בארדואינו Due - מספר ההדק .

דוגמא : התכנית הבאה מחליפה את מצב הלד בכל לחיצה על מפסק. המפסק מחובר בהדק 2 של הכרטיס המתאים לפסיקה חיצונית 0 . ההדק מוגדר כ INPUT\_PULLUP . כאשר המפסק פתוח יש 1 בכניסה וכאשר הוא סגור תועבר אדמה ( 0) לכניסת ההדק.

int pin = 13; // הגדרת משתנה ואתחול ערכו ל 13
volatile int state = LOW; // הגדרת משתנה

void **setup**()
{
  pinMode(2,INPUT\_PULLUP); // הגדרת הדק מספר 2 כקלט עם נגד משיכה למעלה

 pinMode(pin, OUTPUT); // הגדרת ההדק 13 כיציאה
  attachInterrupt(0, blink, FALLING);

/\* blink היא פסיקה מספר 0 , הפונקציה שתתבצע בזמן הפסיקה

. הפסיקה תתבצע כשיש שינוי בהדק מ 1 ל 0

\*/
}

void **loop**()
{
  digitalWrite(pin, state); // state העברה להדק 13 את מצב המשתנה
}

void blink() // פונקצית הפסיקה
{
  state = !state; // הפוך את מצב המשתנה

 for(int i=0;i<50000;i++); // השהייה להתגבר על ניתור המגעות
}

1. **הפונקציה detachInterrupt( )**

הפונקציה מפסיקה ( חוסמת) את הפסיקה הרשומה בין הסוגריים.

התחביר :

|  |
| --- |
| detachInterrupt(interrupt)  |
| detachInterrupt(pin) // Due רק בארדואינו  |   |
|  |  |

#### interrupt – מספר הפסיקה אותה רוצים לחסום ( לא לאפשר). pin – מספר ההדק של הפסיקה שאותו יש לחסום ( רק בארדואינו Due ) .

1. **הפונקציה interrupts( )**

הפונקציה מאפשרת שוב את הפסיקות , אחרי שנחסמו על ידי הפונקציה ( )noInterrupts (שתוסבר בסעיף הבא ) . הפסיקות מאפשרות למשימות מסוימות לרוץ ברקע והן מאופשרות ב default . חלק מהפונקציות לא יעבדו כאשר פסיקות נחסמות ותקשורת שהתקבלה לא תקבל התייחסות.

התחביר : interrupts( )

הפונקציה איננה מקבלת ערכים ואיננה מחזירה ערך.

דוגמא :

void setup() {} // פונקצית האתחול

void loop()

{

 noInterrupts(); // חסימת כל הפסיקות

 // כאן נרשום קטע קוד שבו לא רוצים שפסיקות יפריעו

 interrupts();

 // קטע קוד שהפסיקות יכולות להשפיע

}

1. **הפונקציה noInterrupts( )**

#### הפונקציה חוסמת פסיקות ( ניתן לאפשר אותן שוב עם הפונקציה interrupts( ) שבסעיף הקודם).

#### תחביר : **noInterrupts()**

#### הפונקציה איננה מקבלת ערכים ואיננה מחזירה ערך.

**כללים לכתיבת פונקציית פסיקה (ISR ) :**

1. קצרה ככל הניתן
2. לא להשתמש בפונקציית delay() .
3. לא לבצע הדפסות למוניטור.
4. להגדיר משתנים גלובאליים המתעדכנים בפונקציית הפסיקה כ volatile .
5. רצוי להימנע מאפשור או חסימה במהלך ריצת התכנית (פונקציית האפשור והחסימה הוסברו בסעיפים הקודמים ).

**טבלאות של מספרי ווקטור הפסיקות עבור ארדואינו אונו ועבור ארדואינו מגה בעמודים הבאים :**

**טבלת וקטור הפסיקות בארדואינו מגה שבו מיקרו בקר ATmega 2560**

****

**המשך טבלת מספרי ווקטור הפסיקות של ATmega2560**

****

**טבלת מספרי וקטור הפסיקות עבור ארדואינו אונו/ארדואינו נאנו עם מיקרו בקר ATmega328**

****