**שעון זמן אמת Real Time Clock (RTC)**

 כאמור, ה-RTC הוא רכיב שעון עם סוללה משלו כך שתוכנו אינו הולך לעיבוד עם כיבוי המחשב. מנקודת ראות פיזית הוא חלק מצ'יפ הנקרא CMOS-RAM או NVRAM זיכרון הניתן לשינוי שיש לו מקור כוח משלו כך שהוא אינו הולך לעיבוד כאשר המחשב נכבה. חלק מההגדרות של מערכת ההפעלה נעשים כאן. משתמשי מחשב מתקדמים מכירים את הבעיה שרכיב כלשהוא של המחשב אינו מתפקד נכון וכיבוי הדלקה של ההמחשה לא פותר זאת, אבל הסרה והתקנה מחדש של ה-driver של רכיב חומרה כן פותר את הבעיה. ראשי התיבות של CMOS הוא Complementary Metal Oxide Semiconductor כאשר Metal Oxide זה החומר שממנו הוא עשוי, והמתפקד תוך ניצול מינימלי של חשמל. המשמעות של זה הוא בערך "רכיב מוליך למחצה חסכוני מאד בחשמל". צ'יפ ה-CMOS הראשון היה ה-MC146818 של חברת מוטורולה ומה שיש במחשבים האישיים עכשיו זה או הצ'יפ הזה או תואם לו.

 כפי שנאמר קודם, ה-RTC יכול לשמש לקריאת הזמן האמיתי, לכתיבת זמן חדש, והוא גם גורם לפסיקות בתדירות בלתי תלויה ב-PIT וגם ניתן לתכנת אותו ליזום פסיקה בתאריך ושעה מסוימים וגם גורם לפסיקה כל אימת שהזמן האמיתי משתנה (בפועל כל שנייה). כל הפסיקות שהצ'יפ הזה גורם מנותבים ל-IRQ8/INT70h, אך התוכנה מסוגלת לכל פסיקה מאיזה משלושת הסוגים היא מהווה. שלושת הסוגים נקראים:

* פסיקות מחזוריות periodic interrupts
* פסיקות יזומות alarm interrupts
* פסיקות עדכון update interrupts.

כל אחד מסוגי הפסיקות הללו אפשר להפעיל ואפשר שלא, באופן בלתי תלוי. רוטינת טיפול בפסיקה 70h יכול לבדוק איזה משלושת הפסיקות התרחשה.

על פי הספרות ניתן לתכנת את ה- periodic interruptsלתדירויות של 1k, 2k, 4k, 8k, 16k, 32k. לפחות בעבר אם לא גם היות הניסיון של מתכנתים שלא כל מחשב יכול לעמוד בתדירויות הגבוהות ורק תדירויות עד 4k נחשבות אמינות.

ה- update interrupts אם מפעילים אותם מתרחשות כל שנייה.

ה-alarm interrupts במידה ומפעילים אותו מתרחש כאשר מזמינים אותו, ואפשר להזמין שיתרחש בזמן עד לרזולוציה של שנייה. במידה ויותר מאחד מסוגי הפסיקות פעיל, לפסיקה מסוימת יכולה להיות משניים או שלושת הסוגים, ורוטינת הטיפול בפסיקה יכולה לדעת זאת.

בנוסף לכך ה-RTC יכול לקבל ערך של זמן ותאריך אמיתי, יכולה לשמר אותו ללא צורך במעורבות מצד המשתמש כביכול. כהערת אגב אומר, שבמשך זמן רב שה-RTC בשימוש נתקלו לעיתים קרובות, בעבר אולי אפילו רוב הזמן, שה-RTC אינו מדויק: הוא נטה לפגר אחר הזמן האמיתי בקנה מידה של דקה לשבוע או אפילו כמה ימים. אני חושב שהמצב עכשיו טוב יותר.

ה-RTC נוסף למחשב האישי בשלב ה-XT. עד אז ברגע הדלקת המחשב התאריך של המחשב היה באופן פיקטיבי 1.1.1980 (בפועל כל המחשבים האישיים סופקו לאחר התאריך הזה) והמשתמש היה צריך בכל פעם שהדליק את המחשב להכניס את התאריך והשעה הנכונים. אם התעצל או טעה, הדבר יצר בעיות במערכות הקבצים למשל, שנוצרו "בעתיד".

**תכנות ה-RTC**

 לכמעט כל CMOS יש לפחות 64 בתים של זיכרון (למעשה אוגרים) שהמעבד יכול לגשת אליהם. יש צ'פים של CMOS שיש להם יותר מ-64 בתים של אוגרים, אבל השימוש בהם הוא לא סטנדרטי, ולכן התוכניות שנראה נזהרות לא לגעת בהם, ולא להיות מושפעות מקיומן האפשרי.

 מתוך 64 בתי האוגרים, 14 הבתים הראשונים קשורים ל-RTC. היתר קשורים לרכיבים אחרים כמו כמות הזיכרון האלקטרוני של המחשב או קיומם של דיסקים קשיחים ראשי ומשני וכו'. לעניינינו רלוונטיים רק 14 הבתים הראשונים.

 העיקרון של תכנות הגישה לאוגרים הללו הוא פשוט. ישנו צמד port –ים מספרים 70h ו-71h המשמשים לגישה לאוגרים הללו. כמו צמדי port –ים אחרים במחשב, אחד מהם, port 70h, משמש נתב: על ידי כתיבה לתוכו, החיבור port 71h מתחבר לאוגר המתאים.

 14 מספרי הניתוב הראשונים של port 70h הם כלהלן:

1. - זמן אמיתי בשניות
2. - alarm זמן אמיתי בשניות
3. - זמן אמיתי דקות
4. - alarm זמן אמיתי דקות
5. - זמן אמיתי שעות
6. - alarm זמן אמיתי שעות
7. - זמן אמיתי יום בשבוע (מ-1 עד 7)
8. - זמן אמיתי יום בחודש(מ-1 עד 31)
9. - זמן אמיתי חודש (מ-1 עד 12)
10. - זמן אמיתי שנה (מ-0 עד 99)
11. - Aאיגר סטטוס
12. - B אוגר סטטוס
13. - C אוגר סטטוס
14. - D אוגר סטטוס

 **כל פעם שרוצים לקרוא מ-port 71h חייבים לכתוב ל-port70h את אחד הערכים הללו.**

**כל פעם שרוצים לכתוב ל-port 71h חייבי לכתוב ל-port70h את אחד הערכים הללו, וצריך לכתוב אותו מחדש עם הביט המשמעותי ביותר דלוק.**

**בכל פעם שכותבים ל-port 71h חייבים מיד לאחר מכן לקרוא את port 71h.**

**גם חייבים לקרוא את אוגר סטאטוס C ו-D אם משנים של אוגרי סטאטוס A או B, גם אם לא עושים עם זה שום דבר. את C יש לקרוא עם כתיבה כפולה של הקוד, D רק כתיבה בודדת.**

לדוגמא, אם אני רוצה לקרוא את השעה הנוכחית,נניח לתוך משתנה hour, הקוד הבא יעשה זאת:

 MOV AL,4

 OUT 70h,AL

 IN AL,71h

 MOV BYTE PTR hour,AL

בכדי לכתוב לשדה השעה ערך חדש, נניח מתוך משתנה hour, הקוד הבא יעשה זאת:

 MOV AL,4

 OUT 70h,AL

 MOV AL,84h

 OUT 70h,AL

 MOV AL,BYTE PTR hour

 OUT 71h,AL

 IN AL,71h

הפורמט של הנתונים הוא בדרך כלל ב-Binary Coded Decimal (BCD) פורמט דומה למחרוזות שבו מיוצגים ספרות עשרוניות, כל סיפרה עשרונית ב-4 סיביות, שני ספרות עשרוניות בכל בית. לדוגמא, אם העשה עכשיו 23:45 אז תוכן x71h בבינארי יהיה

----2-- ---3---

0 0 1 0 0 0 1 1

אפשר לשנות את הפורמט לבינארי אבל אין טעם לעשות זאת כי קוד פנימי של המערכות הנפוצות מסתמכות על כך שהתוכן של השדות הללו בפורמט BCD.

שדות ה-alarm מאפשרים לנו להזמין פסיקה מהסוג הזה בשעה ביום שבו מדובר, לכל היותר 24 שעות מראש.

כרגיל פקודתh 71AL, IN קוראת את הנתון ופקודת OUT 71h,AL כותבת לשם.

אוגרי הסטאטוס מאפשרים לנו לקרוא או לקבוע את המאפיינים של השעון, להפעיל או להשבית את פסיקה 70h ובמקרה של טיפול בפסיקה 70h, לגלות איזה סוג של פסיקה התרחשה.

**מבנה אוגרי הסטאטוס:**

**אוגר סטאטוס A:**

 7 6 5 4 3 2 1 0

|  |  |  |
| --- | --- | --- |
| UIP |  Base |  Rate |

UIP: Update In Progress

 1= date/time is updated 0= date/time can be accessed

Base: time Base

 Standard value 010 = 32768 Hz

Rate: Rate select 65536/2Rate

 Standard value 0110 = 1024 Hz (65536/26 = 65536/64)

**אוגר סטאטוס B:**

 7 6 5 4 3 2 1 0

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| SET | PI | AI | UI | SQU | DM | 24h | DLS |

SET: Set Update Cycle

 0= disabled 1= enabled (standard)

PI: Periodic Interrupt

 0= disabled 1= enabled (standard)

AI: Alarm Interrupt

 0= disabled 1= enabled (standard)

UI : Update Interrupt

 0= disabled 1= enabled (standard)

SQU: Square-wave signal

 0= disabled 1= enabled (standard)

DM: Data Mode

 0= date/time BCD encoded 1= binary coded

24h:

 0=24-hour-clock 1=12-hour-clock

DLS: DayLight Saving

 0= disabled 1= enabled (standard)

**אוגר סטאטוס C:**

 7 6 5 4 3 2 1 0

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| IRQ | PS | AS | US | Reserved |

IRQ: Interrupt Request

 1= Interrupt requested 0 = No interrupt requested

 PS: Source of IRQ is Periodic interrupt

 1=yes 0=no

 AS: Source of IRQ is Alarm interrupt

 1=yes 0=no

 US: Source of IRQ is Update interrupt

 1=yes 0=no

**אוגר סטאטוס D:**

 7 6 5 4 3 2 1 0

|  |  |
| --- | --- |
| Val | Reserved |

val: Data value in CMOS RAM

 1=valid (battery ok) 0=invalid

**הערות:**

 פורמאלית, הערכים של Base יכולים לרוץ מ-0 עד 7 אך המשמעות של הערכים לא מוגדרים רשמית (למעט ערך 2), והערך של Rate בין 3 ל-15. ערך 0 ל-Rate משתק כנראה את הצ'יפ.

התדירות של ה-periodic interrupt נא מ-2Hz ללא ברור עד כמה, אבל המקסימום הוא לפחות 1024.

**מרשם לתוכנית המשתמשת בפסיקה 70h**

 תכנות פסיקה 70h הוא יותר מורכב מ-8.

סידרת הפעולות הבאות תספיק.

ככלל צריך לבצע את הדברים הבאים:

 על מנת להשתלט על פסיקה 70h יש לעשות את הפקודות הבאות:

* תוך חסימת פסיקות:
* יש להגדיר רוטינה משלך כרוטינת הטיפול בפסיקה 70h. את זה עושים באותה צורה שמשתלטים על כל פסיקה אחרת, בדרך כלל תוך שימור המטפל הקודם, למשל ע"י רוטינות getvect ו-setvect של טורבו C, קריאה כתיבי ל-4 הבתים 448-451 (4x70h – 4x70h+3) בסגמנט האפס וכו'.
* יש לוודא שביט מספר 0 של ה-Mask register של ה-PIC **המשני** הינו אפס. כתיבה/קריאה של האוגר הזה הוא דרך port 0A1h. מומלץ לשמור את הערך הקודם שלו בשביל לשחזר אותו עם סיום האפליקציה.

בהנחה שאנחנו רוצים לשמר את ה-Mask register של ה-PIC המשני במשתנה בשם old\_0A1h\_mask הפקודות הבאות, למשל, יעשו זאת:

 IN AL,0A1h

 MOV old\_0A1h\_mask,AL

 AND AL,0FEh

 OUT 0A1h,AL

לאחר מכן יש לבחור תדירות רצויה ולהגדיר אותו בשדה Rate באוגר הסטאטוס A של ה-RTC דרך ports 70h-71h. למשל הקוד הבא עושה קובע את התדירות לערך הסטנדרטי של 1024Hz, תוך שהוא נזהר לא לשנות את השדות UIP ו-Base:

 MOV AL,0Ah

 OUT 70h,AL

 MOV AL,8Ah

 OUT 70h,AL

 IN AL,71h

 AND AL,11110000b

 OR AL,0110b

 OUT 71h,AL

 IN AL,71h

לאחר מכן יש להפעיל את סוג הפסיקה המתאימה באוגר סטאטוס B של ה-RTC דרך ports 70h-71h. למשל הקוד הבא מפעיל את ה-periodic interrupts ע"י הדלקת ביו ה-PI תוך שהוא נזהר לא לשנות את השדות האחרים באוגר. מאחר והוא משנה את אוגרי הסטאטוס A –B, הוא חייב לאחר מכן לקרוא את אוגר סטאטוס C עם כתיבה כפולה, D עם כתיבה בודדת:

 IN AL,70h

 MOV AL,0Bh

 OUT 70h,AL

 MOV AL,8Bh

 OUT 70h,AL

 IN AL,71h

 OR AL,40h

 OUT 71h,AL

 IN AL,71h

 IN AL,70h

 MOV AL,0Ch

 OUT 70h,AL

 IN AL,70h

 MOV AL,8Ch

 OUT 70h,AL

 IN AL,71h

 IN AL,70h

 MOV AL,0Dh

 OUT 70h,AL

 IN AL,70h

 MOV AL,8Dh

 OUT 70h,AL

 IN AL,71h

אשר לרוטינת הטיפול בפסיקה 70h עצמה, מלבד לבצע את הקוד שלמענה הייתה ההשתלטות על הפסיקה, היא צריכה להודיע לחומרה שהפסיקה טופלה. הדבר נעשה ע"י קריאת אוגר סטאטוס C של ה-RTC וכן לכתוב את הערך 20h ל-ports של ה-PIC הראשי (port 20h) והמשני (port 0A0). מאחר ומדובר בפסיקת חומרה, יש לשמר את תוכן port 70h לפני שמשנים אותו ולשחזר אותו בתום התהליך. אחרת יש סכנה (קלושה אך קיימת) שתוכנית שהייתה בעיצומו של קריאה מה-CMOS ולא חסם פסיקות יוטעה בשל כך. הקוד הבא עושה את הדברים הללו:

 IN AL,70h

 MOV BX,AX

 MOV AL,0Ch

 OUT 70h,AL

 MOV AL,8Ch

 OUT 70h,AL

 IN AL,71h

 MOV AX,BX

 OUT 70h,AL

 MOV AL,20h

 OUT 0A0h,AL

 OUT 020h,AL

**תדירות פסיקות תחת האמולציה**

 תוכנית מדידות שעשיתי (ספירת מספר הפסיקות ה-pereiodic בי שני פסיקות update) תחת ה-cmd.exe נותן את התוצאות הבאות:

base = 0, rate = 1, frequency = 251 Hz

base = 0, rate = 2, frequency = 126 Hz

base = 0, rate = 3, frequency = 1001 Hz

base = 0, rate = 4, frequency = 1064 Hz

base = 0, rate = 5, frequency = 1004 Hz

base = 0, rate = 6, frequency = 999 Hz

base = 0, rate = 7, frequency = 502 Hz

base = 0, rate = 8, frequency = 252 Hz

base = 0, rate = 9, frequency = 133 Hz

base = 0, rate = 10, frequency = 63 Hz

base = 0, rate = 11, frequency = 33 Hz

base = 0, rate = 12, frequency = 16 Hz

base = 0, rate = 13, frequency = 8 Hz

base = 0, rate = 14, frequency = 4 Hz

base = 0, rate = 15, frequency = 2 Hz

base = 1, rate = 1, frequency = 253 Hz

base = 1, rate = 2, frequency = 126 Hz

base = 1, rate = 3, frequency = 1008 Hz

base = 1, rate = 4, frequency = 1002 Hz

base = 1, rate = 5, frequency = 1012 Hz

base = 1, rate = 6, frequency = 1056 Hz

base = 1, rate = 7, frequency = 503 Hz

base = 1, rate = 8, frequency = 253 Hz

base = 1, rate = 9, frequency = 127 Hz

base = 1, rate = 10, frequency = 63 Hz

base = 1, rate = 11, frequency = 33 Hz

base = 1, rate = 12, frequency = 16 Hz

base = 1, rate = 13, frequency = 9 Hz

base = 1, rate = 14, frequency = 4 Hz

base = 1, rate = 15, frequency = 2 Hz

base = 2, rate = 1, frequency = 222 Hz

base = 2, rate = 2, frequency = 126 Hz

base = 2, rate = 3, frequency = 1067 Hz

base = 2, rate = 4, frequency = 1008 Hz

base = 2, rate = 5, frequency = 1009 Hz

base = 2, rate = 6, frequency = 1002 Hz

base = 2, rate = 7, frequency = 500 Hz

base = 2, rate = 8, frequency = 265 Hz

base = 2, rate = 9, frequency = 126 Hz

base = 2, rate = 10, frequency = 64 Hz

base = 2, rate = 11, frequency = 32 Hz

base = 2, rate = 12, frequency = 16 Hz

base = 2, rate = 13, frequency = 8 Hz

base = 2, rate = 14, frequency = 4 Hz

base = 2, rate = 15, frequency = 2 Hz

base = 3, rate = 1, frequency = 250 Hz

base = 3, rate = 2, frequency = 126 Hz

base = 3, rate = 3, frequency = 1005 Hz

base = 3, rate = 4, frequency = 1012 Hz

base = 3, rate = 5, frequency = 1066 Hz

base = 3, rate = 6, frequency = 1006 Hz

base = 3, rate = 7, frequency = 502 Hz

base = 3, rate = 8, frequency = 251 Hz

base = 3, rate = 9, frequency = 125 Hz

base = 3, rate = 10, frequency = 67 Hz

base = 3, rate = 11, frequency = 31 Hz

base = 3, rate = 12, frequency = 16 Hz

base = 3, rate = 13, frequency = 8 Hz

base = 3, rate = 14, frequency = 4 Hz

base = 3, rate = 15, frequency = 2 Hz

base = 4, rate = 1, frequency = 222 Hz

base = 4, rate = 2, frequency = 132 Hz

base = 4, rate = 3, frequency = 1011 Hz

base = 4, rate = 4, frequency = 1005 Hz

base = 4, rate = 5, frequency = 1008 Hz

base = 4, rate = 6, frequency = 1008 Hz

base = 4, rate = 7, frequency = 499 Hz

base = 4, rate = 8, frequency = 253 Hz

base = 4, rate = 9, frequency = 126 Hz

base = 4, rate = 10, frequency = 67 Hz

base = 4, rate = 11, frequency = 32 Hz

base = 4, rate = 12, frequency = 16 Hz

base = 4, rate = 13, frequency = 8 Hz

base = 4, rate = 14, frequency = 4 Hz

base = 4, rate = 15, frequency = 1 Hz

base = 5, rate = 1, frequency = 248 Hz

base = 5, rate = 2, frequency = 134 Hz

base = 5, rate = 3, frequency = 1012 Hz

base = 5, rate = 4, frequency = 1003 Hz

base = 5, rate = 5, frequency = 1010 Hz

base = 5, rate = 6, frequency = 1000 Hz

base = 5, rate = 7, frequency = 530 Hz

base = 5, rate = 8, frequency = 251 Hz

base = 5, rate = 9, frequency = 126 Hz

base = 5, rate = 10, frequency = 63 Hz

base = 5, rate = 11, frequency = 32 Hz

base = 5, rate = 12, frequency = 17 Hz

base = 5, rate = 13, frequency = 8 Hz

base = 5, rate = 14, frequency = 4 Hz

base = 5, rate = 15, frequency = 2 Hz

base = 6, rate = 1, frequency = 229 Hz

base = 6, rate = 2, frequency = 126 Hz

base = 6, rate = 3, frequency = 1003 Hz

base = 6, rate = 4, frequency = 1064 Hz

base = 6, rate = 5, frequency = 1008 Hz

base = 6, rate = 6, frequency = 1007 Hz

base = 6, rate = 7, frequency = 503 Hz

base = 6, rate = 8, frequency = 252 Hz

base = 6, rate = 9, frequency = 133 Hz

base = 6, rate = 10, frequency = 63 Hz

base = 6, rate = 11, frequency = 33 Hz

base = 6, rate = 12, frequency = 16 Hz

base = 6, rate = 13, frequency = 8 Hz

base = 6, rate = 14, frequency = 4 Hz

base = 6, rate = 15, frequency = 2 Hz

base = 7, rate = 1, frequency = 253 Hz

base = 7, rate = 2, frequency = 126 Hz

base = 7, rate = 3, frequency = 1005 Hz

base = 7, rate = 4, frequency = 1012 Hz

base = 7, rate = 5, frequency = 1007 Hz

base = 7, rate = 6, frequency = 1067 Hz

base = 7, rate = 7, frequency = 498 Hz

base = 7, rate = 8, frequency = 252 Hz

base = 7, rate = 9, frequency = 126 Hz

base = 7, rate = 10, frequency = 63 Hz

base = 7, rate = 11, frequency = 31 Hz

base = 7, rate = 12, frequency = 16 Hz

base = 7, rate = 13, frequency = 9 Hz

base = 7, rate = 14, frequency = 4 Hz

base = 7, rate = 15, frequency = 2 Hz