תוכנת ה – Bootloader

לעומת משפחת הכרטיסים הישנה (משפחת HC11) , למשפחת ה- HC(S)12, אין bootstrap ROM המכיל תוכנת בסיס לשכתוב התוכנות הקיימות על זיכרון הכרטיס. על מנת לתכנת את הכרטיס מחדש, ללא ROM כזה, יש צורך באחד משני הבאים:

- חומרה מיוחדת בשם BDM - זהו כבל מיוחד המכניס את הכרטיס למצב Background Debug Mode.
- Serial Bootloader

ישנם חלקים מוגנים בזיכרון ה- FLASH, וה- Serial Bootloader בדרך כלל יהיה בכתובות $ C000 עד $FFFF (מוגן מפני מחיקה).
זיכרון ה- FLASH עצמו ב- 9S12DP256 מורכב מ- 4 בלוקים בגודל 65,536 בייטים האחד. כל בלוק מאורגן כ- 32,768 מילים בנות 16 ביט. התייחסות אליו נעשית כבייטים או כמילים. זמן הגישה אליו, הוא מחזור שעון אחד לבייט ומילים מאורגנות (Aligned) ו- 2 מחזורי שעון למילים לא מאורגנות (misaligned words). כתיבה/מחיקה של הזיכרון היא רק עבור מילים מאורגנות.
כל k-64 בלוק, מאורגן כ – 1024 שורות של 32 מילים. בלוק מחיקה מכיל 8 שורות של 512 בייט.
פעולות מחיקה על ה- FLASH יעשו על בלוק לפחות בגודל k-512, או על כל 65,536 הבייטים.

תוכנה זו היא הבסיסית ביותר הקיימת על הכרטיס, בעזרתה ניתן להעלות תוכנה על הזיכרון,ללא כרטיס חיצוני מיוחד הנקרא BDM (Background Debug Mode). בעזרת 2 מתגי קונפיגורציה (שלהם ארבעה מצבים, שרק 2 מהם רלוונטיים – 00 או 11, מפני של – 2אחרים מובילים לזיכרון שאינו גדול מ 4 קילו בייט)
אסביר ואנתח את ה- Bootloader של MC9S12DP256, מכיוון שהוא המתוחכם ביותר.
הגרסאות של הצ'יפים הקודמים מאד דומים , אך פחות מתוחכמים.
תוכנת ה- Bootloader, ממוקמת בזיכרון ה – FLASH בכתובות $FFFF - $F000 .
ראשית אסביר על הוקטורים של ה- HC(S)12. הוקטורים הם סדרת כתובות המכילות סדרת כתובות אליהן התוכנה צריכה לגשת בזמן פסיקה. לדוגמא, פסיקת RESET. בתוך כתובת הזיכרון של וקטור Reset, יש לשים את כתובת ההתחלה של התוכנה, באופן הבא:

```
ORG     ResetVec
fdb     ProgStrt     ;RESET - where execution begins after a
                     ;hard reset to boot pesuado vector
```

כאשר מספקים מתח , או לוחצים על כפתור ה – Reset, כתובת הווקטור ResetVec נבדקת, ונלקח ערך הכתובת ProgStrt. אני הגדרתי את ProgStrt כ - $4000, ו- ResetVec מוגדר כ - $ EFFE.
כתובת הווקטור המקורי של ה- Reset, מוגדר כ- $FFFE, אך כשנלחץ RESET, אם קונפיגורציית הבורד מכוונת ל- Bootloader, בתוך וקטור זה יש את כתובת תחילת תוכנת ה- Bootloader, שהיא $F000. כאשר מכוון הכרטיס לעבודה מה- FLASH, דואג ה- Bootloader, במקום לקפוץ לכתובת שערכה שמור ב- $FFFE, אלא לכתובת השמורה כ- 1000 כתובות מתחת - $EFFE.
למעשה, כל טבלת הוקטורים המקוריים הועתקה אל מתחת ל- $F000, ואלו נקראים פסאודו וקטורים.

התוכנה עצמה נמצאת בעמוד הבא.
ראשית נראה כי הפקודה הראשונה היא ORG $F000. ORG משמעה ORIGIN, כלומר שורה זו מוגדרת להיות $F000.

לאחר מכן, נבדק הערך בשני הביטים הימניים בפורט PortAD0 (שהוא בייט שלם). 2 ביטים אלו יכולים להיות 00, 01 , 10 או 11 (כלומר, 0 , 1 , 2 או 3).
אם הוא שווה ל- 01, התוכנה קופצת לתווית בשם JumpEE – כלומר לזיכרון eeprom - .
אם הוא שווה ל – 11 (הערך 3) , הוא קופץ לתווית בשם Boot.

בעלי אתר הרובוטיקה הישראלי לא ישאו באחריות כלשהי לכל נזק, כספי או אחר שייגרם במישרין או בעקיפין משימוש במידע המצוי באתר זה

© כל הזכויות שמורות לאסף פוניס, גיא יונה ואלי קולברג
אין להעתיק תכנים מאתר זה ללא רשות בכתב ממנהלי האתר

ואם לא, הוא ממשיך לשורה הבאה –

```
jmp          [Reset-BootBlkSize,pcr]
```

זוהי שיטת מיעון. ראשית, נלקח הערך של Reset (שהוא $FFFE) , ומוחסר ממנו BootBlkSize, גודל שהוגדר כ- $1000. כעת יש לנו $EFFE. כעת, pcr הוגדר, אך לא הושם בו ערך כלשהו, לכן יש בו אפס (default) . כעת, האפס הזה מוסף לערך $EFFE, ומתוך הכתובת $EFFE נלקח מספר (המאוחסן בתוך הכתובת הזו), ולשם קופצת התוכנה.
כלומר אם, בכתובת $EFFE יש את הערך $4000, התוכנה תגיע לכתובת התחלתית 4000 (ב- 9S12, זהו תחילת ה- FLASH).

תחילת תוכנת ה – Bootloader:

```
*****************************************************************************************
;
;
            org    $f000
;
;
BootStart:  bra      OverGlobals
;
SRecLow:    dc.b  SRecLow256
NumFBlocks: dc.b  3                  ; number of Flash blocks to perform a mass erase on.
PPAGEBase:  dc.b  $30
;
OverGlobals:
            ldaa  #$03               ; enable the digital input buffer for bit 0 & 1 of
            staa  ATDDIEN0; A2D #0 port.
            nop
            nop
            ldaa  PortAD0  ; get PortAD0 value.
            anda  #$03               ; mask off all but bit 0.
            cmpa  #$01               ; jump to internal EEPROM?
            beq    JumpEE  ; yes.
;
            cmpa  #$03               ; should we execute the bootloader?
            beq    Boot              ; yes.
DEBUG12:    jmp    [Reset-BootBlkSize,pcr]
```

נעבור על רוטינת ה- EEPROM:

```
JumpEE:       ldab  PARTID
              cmpb  #Dx128ID
              beq            JumpEE128
              jmp            EEStart
JumpEE128:   ldab   #$20     ; move the RAM to $2000 (so the EEPROM becomes visible).
              stab  INITRM
              Jmp             EEStart  ; then jump to the start of internal EEPROM
                             ;(above the I/O registers).
```

בעלי אתר הרובוטיקה הישראלי לא ישאו באחריות כלשהי לכל נזק, כספי או אחר שייגרם במישרין או בעקיפין משימוש במידע המצוי באתר זה

© כל הזכויות שמורות לאסף פוניס, גיא יונה ואלי קולברג
אין להעתיק תכנים מאתר זה ללא רשות בכתב ממנהלי האתר

PARTID היא כתובת פנימית הבודקת את מצב ה- EEPROM.
לאחר מכן, נקבעת כתובת 2000 ככתובת ההתחלה עבור התחלה מ – EEPROM.

כעת נעבור על מחיקת ה- EEPROM (למרות, כפי שצוין, שזו אינה אפשרות שנרצה).
ראשית, נלקח ערך לצובר B והושם בכתובת $400. כתובת זו אחראית על מחיקה של ה – EEPROM והערך הזה אומר כי מוחקים את ה – EEPROM ומוחקים בבלוקים.

הפקודות שלאחר מכן, אומרות להתחיל במחיקה – בכתובת הנקראת ESTAT – שהיא מעידה על סטטוס ה- EEPROM. במקרה והיה ERROR במחיקה, נדע על כך וכל התהליך יסתיים – ויחזור למוניטור, עם הודעה שגיאה. אם לא, המחיקה ממשיכה.
במקרה והמחיקה בסדר, ישנה לולאה, שתיתקע, עד שתסתיים המחיקה.
לאחר מכן, נראה אם הגענו לסוף ה- $1000 (RAM) . אם כן – יציאה מהלולאה, חזרה מהמוניטור ואם לא, חזרה על המחיקה, מעבר לבלוק הבא של הזיכרון.

```
;*****************************************************************************************
EraseEE:
        ldab    #ERASE+MASS         ; perform a bulk erase.
        std     EEStart             ; latch address for erase command.
        stab    ECMD
        ldab    #CBEIF
        stab    ESTAT               ; initiate the erase command.
        brclr   ESTAT,#PVIOL+ACCERR,EEErCmdOK        ; continue if the privliage
                                    ;violation & Access error flags are clear.
        ldaa    #EEraseError
        bra     SaveEEError
EEErCmdOK:
        brclr   ESTAT,#CCIF,*       ; wait until the command has completed.
        ldab    #ERVER+MASS         ; perform an erase verify.
        std     EEStart             ; latch address for erase verify command.
        stab    ECMD
        ldab    #CBEIF
        stab    ESTAT               ; initiate the erase command.
        brclr   ESTAT,#PVIOL+ACCERR,EEVerfCmdOK      ; indicate failure if the privliage
                                    ;violation or Access error flags are set.
EENotErased:
        ldaa    #EEraseError
        bra     SaveEEError
EEVerfCmdOK:
        brclr   ESTAT,#CCIF,*       ; wait until the command has completed.
        brclr   ESTAT,#BLANK,EENotErased    ; flag a not erased error if the BLANK bit
                                            ;did not set.
        ldab    #BLANK              ; clear the BLANK status bit.
        stab    ESTAT
        clra
SaveEEError:
        staa    ErrorFlag,pcr       ; put error code where pod can access it.
        rts                         ; if we fall through, we automatically return a non-
                                    ;zero condition.
;*****************************************************************************************
```

בעלי אתר הרובוטיקה הישראלי לא ישאו באחריות כלשהי לכל נזק, כספי או אחר שייגרם במישרין או בעקיפין משימוש במידע המצוי באתר זה

© כל הזכויות שמורות לאסף פוניס, גיא יונה ואלי קולברג
אין להעתיק תכנים מאתר זה ללא רשות בכתב ממנהלי האתר

מחיקה ותכנות מחדש של ה- FLASH נעשה בתהליך זהה למדי, רק עם שמות רגיסטרים שונים.
בדומה, אם הייתה שגיאה במהלך המחיקה/תכנות של זיכרון זה, הודעה תופיע על המוניטור על השגיאה.
נשאר אם כן, שינוי ה- Baud Rate:

```
;*******************************************************************************
;
SetBaud:
          leax      BaudPrompt,pcr      ; get the bootloader prompt
          jsr       PromptResp,pcr      ; go display the prompt & get a 1 character response.
          cmpb      #$31                ; do a range check. less than '1'?
          blo       SetBaud             ; yes. just re-display the prompt.
          cmpb      #$34                ; greater than '4'?
          bhi       SetBaud             ; yes. just re-display the prompt.
          andb      #$0f                ; no. mask off the upper nybble.
          decb
          lslb
          leax      BaudTable,pcr
          ldd       b,x
          pshd
          leax      BaudChgPrompt,pcr
          jsr       OutStr,pcr          ; send it to the terminal.
          brclr     SCI0SR1,#TC,*       ; wait until the last character is sent until we change
                                        ;the baud rate.
          puld
          std       SCI0BD
          jsr       getchar,pcr         ; go get the user's choice.
          leax      CrLfStr,pcr         ; go to the next line.
          jsr       OutStr,pcr
          rts
;
BaudTable:
          dc.w      Baud9600
          dc.w      Baud38400
          dc.w      Baud57600
          dc.w      Baud115200
;
;*******************************************************************************
```

לפי הערך ב – ASCII של '1', '2', '3' או '4' , נלקח הערך מהטבלה שלמאה (BaudTable) ונקבע ערך חדש, ומעדכן את כל הרגיסטרים של מערך ה- SCI. לבסוף , נשלחת הודעה למוניטור , האומרת לקבוע ערך ASCII חדש. לאחר שליחת ההודעה, מעודכן סופית ערך ה- BR במערכת ה- SCI של ה- HC(S)12.

שאר הרוטינות הן לולאות זמן (הסופרות מילי ומיקרו שניות), רוטינות שליחת/קבלת ערכי ASCII, רוטינות בקרה של מחיקה ותכנות של הזיכרון וטבלאות ערכים וכתובות.

בעלי אתר הרובוטיקה הישראלי לא ישאו באחריות כלשהי לכל נזק, כספי או אחר שייגרם במישרין או בעקיפין משימוש במידע המצוי באתר זה

© כל הזכויות שמורות לאסף פוניס, גיא יונה ואלי קולברג
אין להעתיק תכנים מאתר זה ללא רשות בכתב ממנהלי האתר