פרויקטון מבוא להנדסת מחשבים

28BYJ-48 – 5V Stepper Motor

קבוצה מספר 6

מאור אסייג 318550746

רפאל שטרית 204654891

תוכן עניינים

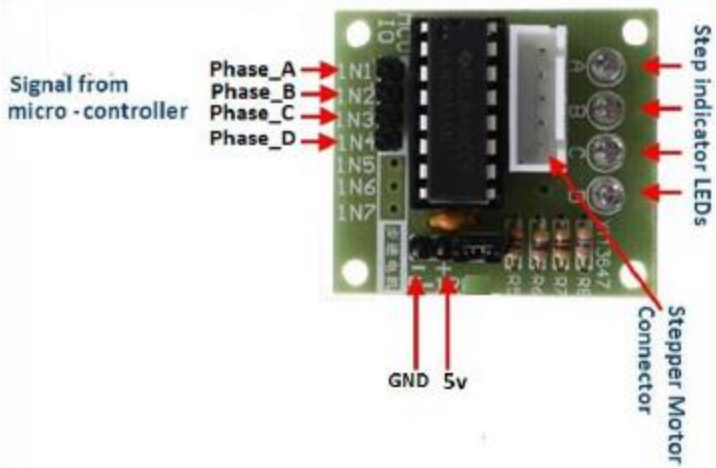
תיאור כללי

ממשק

תיאור כללי

בפרויקט זה נתבקשנו לממש מחוגה דיגיטלית ומונה שניות אנלוגי תוך שימוש בשפת הקוד assembly, בקר MSP-430 (והציוד הנלווה אליו – מקלדת, נורות, כפתורים פיזיים, מסך LCD) ומנוע צעדים מסוג 28BYJ-48 – 5V Stepper Motor.

מנוע צעד, זהו מנוע שניתן להפעילו כך שמוט הסיבוב שלו, יסתובב בכל צעד(סדרה של פקודות) בזווית קבועה. מנוע צעד מהווה עומס , המחובר לכרטיס ממשק המתווך בין הבקר לעומס. מצד אחד נחבר לכרטיס הממשק את ה - MCU המספק מידע בהספק נמוך, מצד שני נחבר את המנוע המהווה עומס וצורך הספק גבוה. יש צורך לחבר לכרטיס מתח הפעלה של v5.



את האינטראקציה עם המשתמש ביססנו דרך מסך ה LCD המובנה עם בקר ה Msp שבמעבדה, תוך כדי לחיצות על כפתורי המקלדת ומעבר בין תפריטי המצבים (Modes) שיפורטו בהמשך.

LCD dislay

מבנה הממשק

ממשק המשתמש, להלן "התפריט" מוצג תוך כדי הדפסה ישירות אל מסך ה LCD ושליטה בתוכן המודפס ובמצבי העבודה על ידי הכפתורים הפיזיים בבקר (P1.0, P1.1) ומקשי המקלדת (KeyBoard) כדלהלן :

- לחיצה על P1.0 תעיר את הבקר ממצב שינה - הודעת כניסה לתפריט, Welcome
- פירוט השימוש בתפריט, לחיצה על A במקלדת תגלול את התפריט כלפי מטה, לחיצה על B במקלדת תגלול את התפריט כלפי מעלה. התפריט בנוי בצורה טורית.
- לאחר לחיצה ראשונית על B או A - נגלה תפריט מצבי העבודה השונים, אותו ניתן לגלול כמפורט לעיל. מצבי העבודה :

1. Spin By Angle – המשתמש יכניס מספר [0,360] המייצג זווית במעלות והמחוגה המחוברת למנוע הצעדים תסתובב בהתאם לזווית ולציר הייחוס. במידה והמשתמש יקיש זווית שאינה בטווח המותר, או תו שאינו ספרה – תופיע הודעת שגיאה.
2. Spin By Voltage – המשתמש יכניס לבקר מתח [0v,3v] דרך כניסת ה ADC ובעזרת דגימה מנוע הצעדים ינוע לזווית המתאימה באופן ייחסי [0,360]~[0v,3v].
3. Clock Mode – מונה שניות אנלוגי, מדמה את מחוג השניות בשעון המוכר לנו.

- המשתמש אינו יכול לבחור מצב עבודה שאינו מופיע על גבי מסך ה LCD, ובמידה ובחר בצורה זו – תופיע הודעת שגיאה לזמן קצר והתפריט יופיע מחדש במצבו הקודם.
- במידה והמשתמש מעוניין להשתמש במצבי העבודה הנוספים לאחר שימוש במצב עבודה מסויים, ברגע לחיצתו על מקש מסויים על מנת לחזור לתפריט – מנוע הצעדים יכייל את עצמו לציר הייחוס.
- בזמן הסיבוב של מנוע הצעדים (במשך הגעתו לזווית מסויימת במצב 1 או 2, או במשך הידמותו למונה שניות אנולוגי) תופיע על מסך ה LCD הודעה המתארת את המתרחש (..'Spining etc).
- לחיצה על P1.1 בכל עת תפסיק את פעולותו של מנוע הצעדים, מנוע הצעדים יכייל את עצמו חזרה לציר הייחוס והבקר יכנס למצב שינה.

אלגוריתמיקה

תיאור כללי

לחיצה על P1.0 – תעיר את הבקר ממצב שינה ותאפשר אינטראקציה כמפורט בעמודים הקודמים עם המשתמש דרך המקלדת.

לחיצה על P1.1 – הפסקת מצב העבודה של מנוע הצעדים, מחיקת התפריט מהמסך, כניסה למצב שינה.

גלילה בתפריט – את התפריט בנינו בצורה טורית, כך שישנו אוגר מצב עבור התפריט המתאר את 2 השורות המודפסות כרגע על המסך, פירוט על האוגרים השונים יהיה בחלק 3. בעת לחיצת מקש היה עלינו לזהות ראשית האם כוונת המשתמש לגלול את המסך, או לבחור את מצב העבודה (כשכל התנאים תקינים). בבחירת מצב העבודה יש לקרוא לפונקציות חיצוניות מתאימות המחלות את תהליך האינטראקציה עם המשתמש (לדוגמא - מתן–זווית סיבוב רצויה), הדפסה למסך והחלת תהליך עבודת מנוע הצעדים.

מנוע הצעדים מסוגל לנוע צעד מלא או חצי צעד. את התוכנית שלנו ביססנו על צעדים מלאים, תוך כדי מתן תשומת לב רבה לתיקון סטיות במצבי העבודה השונים. את מנוע הצעדים חיברנו לכניסות 4~9.0, כפי שיפורט בהמשך.

❖ כדי להפעיל את המנוע **בצעד בודד** (סיבוב אחד בזווית $\varphi = 0.088°$) **בכיוון קדמי**, נדרש להכתיב מהבקר (MCU) מתח לארבע רגליים Phase_A, Phase_B, Phase_C, Phase_D לפי הטבלה הבאה:
כדי לבצע הזזה רציפה כרצוננו, נצטרך לבצע המתואר בטבלה בצורה מחזורית.
קצב השינוי יקבע את מהירות הסיבוב של מוט המנוע.

רגל בכרטיס הממשק המחוברת לבקר	t_0	t_1	t_2	t_3
Phase_A	1	0	0	0
Phase_B	0	0	0	1
Phase_C	0	0	1	0
Phase_D	0	1	0	0

→ t

את זווית הסיבוב הרצוי (בין אם נקלטה כמספר תלת ספרתי או כערך ייחסי למתח המתקבל מדגימת ADC12) תרגמנו למספר הצעדים הרצוי עבור המנוע תוך כדי התחשבות בסטיות הנורמאליות הנובעות מדיוק עשרוני. במצב עבודה 3, השעון האנולוגי – נעזרנו בשעון פנימי Basic

Timer עם תדירות 1 סיבובים לשנייה (כך שכל סיבוב המנוע ינוע בקירוב 6 מעלות). הרחבה על שיטות הפעולה, שיטות החישוב, התמרון בין הפונקציות, והשימוש הנרחב באוגרי המצבים נפרט בחלק 3.

רכיבי החומרה ואופני העבודה

רכיבי החומרה

- Keypad - לכרטיס הבקר (לוח עליו יושב הבקר) מחובר לוח מקשים כמתואר בתמונה הבאה ומשמש כ input - לבקר.
- LCD - תוכן תצוגת ה - LCD, בעזרת כתיבה לשלושת רגלי הבקרה ושמונה רגלי ה - DATA של ה - LCD.
- P1.0,P1.1 – מקשים פיזיים המחוברים לבקר.
- ADC12 - תפקידו לקשר בין העולם האנלוגי מחוץ לבקר (מתח רציף) לעולם הדיגיטלי בתוך הבקר (על ידי עיבוד נתונים) – על ידי דגימת המתח והמרתו לערך מספרי הנאגר.
- BasicTimer – שעון פנימי בבקר, תדירותו נקבעת על ידי בחירת מקור הכניסה ובמידת הצורך פעולות מתמטיות נוספות עליו (כגון חלוקה וכדומה).
- 5V Stepper Motor – 28BYJ-48 – מנוע צעד המחובר לבקר דרך 4 כניסות שונות, צעד מלא(0.7 מעלות) יתבצע תוך כדי מתן רצף של פקודות קבוע.

תיאור התוכנית

פירוט כללי

קבצים

main	התוכנית הראשית, ISR
F1	מצב עבודה 1 – ע"פ זווית
F2	מצב עבודה 2 – ע"פ מתח
F3	מצב עבודה 3 – שעון אנלוגי
KeyBoard	שיטת 0 הרץ – זיהוי מקש כתו
LCDActivate	אתחול למסך LCD
Port1Function	מקשי התחלה\שינה
Port2Function	תפריט, כניסה למצבי עבודה

פונקציות Macro

Lcd_Data	מדפיס תו למסך ה LCD
Delay	מקבל מספר מחזורי שעון
StepNum	זז כמספר הצעדים
BackStepNum	זז אחורה כמספר הצעדים
NumOfSteps	מספר הצעדים הנכנסים בזווית
AngleINVoltage	מספר הצעדים הנכנסים ברמת מתח
Lcd_Cmd	מקרו של הLCD
ExtraDelay	מקבל מספר מחזורי שעון (מורחב)
LineLcd	מקבל שורה ומדפיס אותה

פונקציות Routines

מגדירה מצב עבודה 1	Function 1
הופכת תווים לזווית תלת ספרתית	Multi
בודקת תקינות הספרה המוקשת	CheckValid
בודקת תקינות הזווית הכוללת	CheckAngle
מבצעת צעד מלא של המנוע	Step
מבצעת צעד מלא אחורנית של המנוע	BackStep
משתמשת בזווית(אוגר) ומסובבת את המנוע לפי הזווית	Spin
משתמשת בזווית(אוגר) ומסובבת את המנוע אחרונית לפי הזווית	BackSpin
מכפילה את R4 ב 10	Multi10
מגדירה את מצב עבודה 2	Function2
דוגם מתח כניסה ומסובב את המנוע ביחס למתח	SampleADC
תיקוני סטיות הזווית במצב עבודה 2	Correction
מגדירה מצב עבודה 3	Function3
מכיילת את המנוע לפי סיבובי המנוע במצב עבודה 3	AfterF3
מכיילת את המנוע לפי מספר הצעדים שעשה	BackSpinClock
מדפיסה את מסך ההתחלה	Starting
אחראית לעדכון התפריט בהתאם לגלילה	MovingMenu
מדפיסה מסך שגיאה	Error

מדפיסה על המסך בהתאם למצבי העבודה השונים	FuncPrint
בודק את המקש הנלחץ ופועל בהתאם	CheckKey
במצב עבודה 1,בודק את תקינות הקלט ופועל בהתאם	CheckF1
אפס הרץ על המקלדת, שומר את התו שהוקש	KeyBoard
אתחול מסך ה LCD	ActivateLCD
אתחול מסך ה LCD	Lcd_Strobe

פסיקות ISR

פסיקת הכפתורים הפיזיים P1.0,P1.1	PORT1_ISR
פסיקת המקלדת,תפריט,מצבי עבודה	PORT2_ISR
שעון פנימי ACLK\128	Basic_Timer_ISR
תוכנית ראשית, קנפוג החומרה	RESET

פירוט נרחב (ראה קוד בעמודים המפורטים)

קובץ main – עמודים 23-26

- **PORT1_ISR :**

מזהה לחיצה על P1.0 או P1.1, P1.0 – מדפיס את מסך ההתחלה, אחראי לכיול המנוע במידה ושומש לפני שנכנס למצב שינה.
P1.1 – מכניס את הבקר למצב שינה, מאתחל את המסך.

- **PORT2_ISR : (מסודר לפי LABLES)**

F3 - ראשית נזהה אם היינו במצב עבודה 3, במידה וכן – נכייל את המנוע לפי מספר הסיבובים שעשינו בעבר (כמובן שהספירה תהיה מ 0 ל 360 מעלות וחוזר חלילה) על ידי קריאה לפונקציה AfterF3
BackSpinAngle – בודק האם המנוע הסתובב בעבר בעזרת R9 (שומר את מספר הצעדים שהמנוע ביצע לסיבוב בזווית מסויימת). במידה וכן – נקרא לפונקציית הדפסה של "נא המתן", וקורא לסיבוב המנוע אחרונית BackSpin.
F1 – בודק האם אנחנו במצב עבודה 1 ובהתאם קורא לCheckF1 האחראית לזיהוי הזווית המוכנסת במקלדת וסיבוב בהתאם לתנאים התקינים.
StartCheckKey – במידה ועד כה לא פעלנו, נבדוק את המקשים האחרים (גלילה\מצבי עבודה חדשים\תקן לא חוקי).

- **Basic_Timer_ISR** :

בכל כניסה לפי תדר המקור הפנימי (128\32768) המנוע ינוע ב 6 מעלות, בצורה זו נדמה מונה שניות אנלוגי. בכל כניסה נעזר במונה כך שנוכל לעקוב אחר מספר הכניסות הכולל, בהשלמת 360 מעלות נאפס את מונה הכניסות (כל 57 כניסות, כאשר כל 6 מעלות זה בעצם 6.3 מעלות במנוע הצעד).

קובץ F1 – עמודים 27-31

- **Function1** :
R10 מקבל 1 (מצב עבודה 1),R12 מקבל 3(יוסבר ב Multi – הכוונה היא שהספרה הראשונה הינה ספרת המאות אזי תוכפל ב100).
קריאה להדפסה מתאימה.
- **Multi :**
נקראת לאחר הקשת תו במקלדת במצב עבודה 1. בסופה תשמר הזווית כמספר תלת ספרתי(זיהוי הכפל התקין בעזרת אוגר R12) תקין באוגר R4.
- **CheckValid :**
בודקת אם התו שהוקש (R5) אכן בטווח התקין (0~9) ומדפיסה אותו(בהתאם לתקינותו), מעדכנת את אוגר התקינות R11 שבעזרתו תודפס הודעת שגיאה בהתאמה.
- **CheckAngle :**
לאחר השלמת הזווית ב R4 כמספר תלת ספרתי, פונקציה זו בודקת את תקינות גודל הזווית ומשנה בהתאמה את אוגר התקינות R11.
- **Step :**
מבצעת את סט הפקודות המתאים לביצוע אוריינטציה לביצוע צעד מלא(0.7 מעלות) במנוע הצעדים, כשבין פקודה לפקודה 1500 מחזורי שעון.
- **BackStep :**
מבצעת את סט הפקודות המתאים לביצוע אוריינטציה לביצוע צעד מלא(0.7 מעלות) אחרונית במנוע הצעדים, כשבין פקודה לפקודה 1500 מחזורי שעון.
- **Spin :**
ראשית מכפילה את R4 (זווית) ב 10, קוראת למקרו שמחשב את מספר הצעדים הנדרש(NumOfSteps) כדי להגיע לזווית זו(תוך ההתחשבות בהכפלה ב 10 – למניעת שברים), מעבירה ל R9(אוגר מצב של הצעדים שבוצעו) ומסובבת את המנוע בהתאם בעזרת מקרו (StepNum).

- **BackSpin :**
  בכדי לכייל את המנוע לאחר ביצוע מצב עבודה כלשהו, בודקת הפונקציה באופן יחסי היכן נמצאת המחוג ביחס לציר האפס (בעזרת מספר הצעדים שבוצעו R9), ובהתאם מסובבת את ההפרש אחרונית או צעד רגיל בדרך הקצרה ביותר.
- **Multi10 :**
  מכפילה ב 10 את מה שיש ב R4.

קובץ F2 – עמודים 32-33

- **Function2 :**
מעדכנת את אוגר המצב R10 למצב 2,מדפיסה למסך את השורות המתאימות, קוראת לפונקציית דגימת המתח והסיבוב בהתאם.
- **SampleADC :**
דוגמת את המתח בעזרת חומרת ה ADC12, ממירה בעזרת מקרו (AngleInVoltage) את המתח היחסי לזווית יחסית, מדפיסה את המסך המתאים, מסובבת את הזווית(Spin) בהתאם לחישוב.
- **Correction :**
מתקן את הזווית היחסית למתח הנכנס ב ADC תוך כדי התחשבות בעיגול המספרים הנעשה בחישוב מספר הצעדים הדרוש להשלמת הזווית.

קובץ F3 – עמודים 34-35

- **Function3 :**
מעדכנת את אוגר המצב R10 למצב 3,מדפיסה למסך את השורות המתאימות, מאפשרת את פסיקת השעון הפנימי התדמה מחוג שניות.
- **AfterF3 :**
תקרא לאחר חזרה לתפריט בסיום מצב עבודה 3, תכייל את המחוג בהתאם למספר הסיבובים שהוא עשה חזרה לציר הייחוס.
- **BackSpinClock :**
בודק את R15 (שומר את מספר הכניסות לשעון הפנימי) כך שלפי מספר הכניסות נדע לאיזה כיוון יותר מהיר לכייל את הציר הייחוס (צעדים קדימה\אחורה של המנוע).

קובץ Port1Function – עמודים 36-39

- **Starting :**
מדפיסה לLCD את שורות ההתחלה. מאתחלת את R8 לתחילת התפריט.
- **MovingMenu :**
אחראי להדפסת השורות (באופן טורי) המתאימות בהתאם לאוגר מצב התפריט R8.
- **Error :**
הדפסת מסך שגיאה למשך זמן קצר אל ה LCD ועדכון חזרה למצב התפריט שהיה.
- **FuncPrint :**
בהתאם למצבים השונים בתוכנית אנו נקרא לפונקציה זו תוך כדי עדכון האוגרים הנבדקים המתאימים, בכל שלב בתוכנית תוך כדי מתן תשומת לב אלו אוגרים נוכל לדרוס. פונקציה זו תדפיס שורות המתארות את המתרחש במצבי העבודה השונים.

קובץ Port2Function – עמודים 40-42

- **CheckKey :**
קולט את התו הנלחץ במקלדת השמור ב R5, במידה ונלחצו A או B נגלול את המסך בהתאם (למטה\למעלה). במידה ו (1\2\3) נפעיל את מצבי העבודה השונים בהתאמה תוך כדי התייחסות למצב התפריט התקיון (כלומר לא ניתן להיכנס למצב 1 בתפריט שמציג את 2 ו 3). במידה והקלט לא תקין תקפוץ הודעת שגיאה.
- **CheckF1 :**
אם אנו במצב עבודה 1 הפונקציה תקרא בכל לחיצה על המקלדת, פונקציה זו אחראית לקליטת הזווית באופן תקין וסיבוב המנוע לזווית הרצויה בהתאמה.

קובץ KeyBoard – עמודים 42-44

- **CheckPress :**
פונקציה הבודקת את התו שהוקש ב KeyPad החומרתי על ידי שיטת "אפס הרץ", במידה ונקלט מקש נשמור את ערכו בAscii.

קובץ LCDActivate - עמודים 21-22

- **Lcd_Strobe ,ActivateLcd :**

קנפוג חומרתי של המקלדת על פי דפי ההסבר המצורפים למודל.

Macro's

- Lcd_Data :

מקבל תו ומדפיס אותו בצורה החומרתית הנכונה על מסך הLCD.

```
; Lcd_Data MACRO
Lcd_Data     MACRO Char
             Delay #0005000
             bic.b #0xff,&P5OUT
             bis.b #0x20,&P3OUT
             mov.b Char,&P5OUT
             call  #Lcd_strobe
             bic.b #0x20,&P3OUT
             ENDM
```

- **Delay** :

מקבלת את מספר מחזורי השעון שהבקר צריך לבצע ומבצעת אותם.

```
;Delay  MACRO
Delay        MACRO  Num
             LOCAL  L1
             mov.w  Num,R13
L1           dec    R13
             jnz    L1
             ENDM
```

- **StepNum** :

מקבלת את מספר הצעדים המלאים שהמנוע צריך לבצע.

```
;StepNum   MACRO
StepNum     MACRO   Num
            LOCAL   L1
            mov.w   Num,R14
L1          call    #Step
            dec     R14
            jnz     L1
            ENDM
```

- **BackStepNum** :

מקבלת את מספר הצעדים אחורנית המלאים שהמנוע צריך לבצע.

```
;BackStepNum   MACRO
BackStepNum     MACRO   Num
            LOCAL   L1
            mov.w   Num,R14
L1          call    #BackStep
            dec     R14
            jnz     L1
            ENDM
```

- **NumOfSteps** :

מקבלת את הזווית*10 בR4 ומחשבת את מספר הצעדים הנדרש בכדי להגיע אליה (כל צעד=0.7 מעלות) ושומרת ב R13 את מספר הצעדים הסופי.

```
;NumOfSteps   MACRO
NumOfSteps MACRO   Num ;get R4*10, Return the numOfSteps in R13
           LOCAL   L1
           mov.w   #7,R11
           mov.w   Num,R14
L1         add.w  #1,R13
           sub.w  R11,R14
           cmp.w  #0,R14
           jge   L1
           ENDM
```

- **AngleInVoltage** :
מקבלת מספר רמת המתח הנקלט במודל החומרתי ADC12
וממירה אותו באופן יחסי לזווית בתחום המתאים [0,360] ושומרת
אותה ב R13.

```
;AngleInVoltage  MACRO
AngleInVoltage MACRO  Num ;get Voltage, Return the Angle ratio to Volta
               LOCAL  L1
           mov.w  #10,R11
           mov.w  Num,R14
L1         add.w #1,R13
           sub.w R11,R14
           cmp.w #0,R14
           jge  L1
           ENDM
```

- **Lcd_Cmd** :
מקרו חומרתי לאתחול מסך ה LCD.

```
; Lcd_cmd MACRO
Lcd_cmd    MACRO command
           Delay #5000   ; 5msec Delay
           mov.b command,&P5OUT
           call #Lcd_strobe
           ENDM
```

- **ExtraDelay** :
ביצוע מספר מחזורי שעון מורחב למען אתחול ה LCD.

```
; ExtraDelay MACRO
ExtraDelay MACRO  Num1
           LOCAL  L2,L3
           clr R7
           mov.w  Num1,R7
L2         mov.w  #35,R13      ;1msec for each 1 value, for dutycycle
L3         dec    R13
           jnz    L3
           dec    R7
           cmp  #0,R7
           jnz L2
           ENDM
```

- **LineLcd :**

מקבל מערך של תווים ומדפיס אותו באופן תקין על מסך ה LCD.

```
; LineLcd   MACRO
LineLcd     MACRO   Line
            LOCAL   newChar,exitLineLcd
            clr R7
            mov     #Line,R7
newChar     mov.b    @R7,R6
            cmp.b   #0,R6
            jz      exitLineLcd
            Lcd_Data R6
            incd R7
            jmp newChar
exitLineLcd
            ENDM
```

אוגרים ושימושים עיקריים

R9	Multi10, BackSpin
R15	Mode 3 – AfterF3,BT_ISR שומר את מספר הסיבובים במונה השניות
R14	StepNum Macro
R13	אוגר זמני לחישוב עיכוב, אוגר לשמירת מספר צעדים באופן זמני
R8	אוגר מצב תפריט
R10	אוגר מצב עבודה

נספח – הקוד כמכלול

ניתן לזהות את שמות הקבצים על פי הכותרות, או לחלופין :

קובץ **main** – עמודים 23-26

קובץ **F1** – עמודים 27-31

קובץ **F2** – עמודים 32-33

קובץ **F3** – עמודים 34-35

קובץ **Port1Function** – עמודים 36-39

קובץ **Port2Function** – עמודים 40-42

קובץ **KeyBoard** – עמודים 42-44

קובץ **LCDActivate** - עמודים 21-22

```
#include  <msp430xG46x.h>

;*********MACRO FUNCTIONS*********************************************
;*************************************************************************
; Lcd_cmd MACRO
Lcd_cmd    MACRO command
           Delay #5000   ; 5msec Delay
           mov.b command,&P5OUT
           call #Lcd_strobe
           ENDM

 ;Delay  MACRO
Delay       MACRO  Num
            LOCAL  L1
            mov.w  Num,R13
L1          dec    R13
            jnz    L1
            ENDM

;=========================================================================
; LCD Active
;=========================================================================
            MODULE    ActivateLCD
            PUBLIC    ActivateLCD
            RSEG CODE

ActivateLCD   mov.b #0x0F,&P10DIR        ;4 LSB OF P10 OUTPUT
              mov.b #0x00,&P10OUT
              bis.b #0xE0,&P3DIR         ; Set P3.5-P3.7 as Output
              bis.b #0xff,&P5DIR         ; Set P5 as Output
              bic.b #0xE0,&P3OUT         ; Set P3.5-P3.7 with 0v
              Delay #15000               ; Delay of 15msec
              mov.b #0x3F,&P5OUT
              call  #Lcd_strobe
              Delay #5000                ; Delay of 5msec
              mov.b #0x3F,&P5OUT
              call  #Lcd_strobe
              Delay #200                 ; Delay of 200usec
              mov.b #0x3F,&P5OUT
              call  #Lcd_strobe
              Lcd_cmd #0x3C
              Lcd_cmd #0x0f
              Lcd_cmd #0x01
              Lcd_cmd #0x06
```

```
            Lcd_cmd #0x80
            Lcd_cmd #0x02
            ret
;==============================================================================
; LCD Strobe
;==============================================================================
          PUBLIC    Lcd_strobe
          RSEG CODE

Lcd_strobe  bis.b   #0x80,&P3OUT
            NOP
            NOP
            bic.b   #0x80,&P3OUT
            ret

            ENDMOD
;==============================================================================
;==============================================================================
            END
```

```
#include  <msp430xG46x.h>
;*******************************************************************
;*******************************************************************
        NAME   RESET
        PUBLIC RESET
        EXTERN KeyBoard
        EXTERN Starting
        EXTERN ActivateLCD
        EXTERN Spin
        EXTERN BackSpin
        EXTERN Function2
        EXTERN FuncPrint
        EXTERN BackSpinClock
        EXTERN Function3
        EXTERN AfterF3
        EXTERN CheckF1
        EXTERN CheckKey
        RSEG    CSTACK
        RSEG    CODE
;*******************************************************************
;*******************************************************************
RESET       mov.w   #SFE(CSTACK),SP
StopWDT     mov.w   #WDTPW+WDTHOLD,&WDTCTL

Main        clr R7  ;ExtraDelay
            clr R9   ;BackSpin Register, Multi10
            clr R6
            clr R10
            clr R11
            clr R15 ;for clock correction
            clr R4
            clr R14 ;for stepNum macro
            clr R13 ;for  Delay macro
            mov #1,R8   ; Move Menu

SetupBT     mov.b #BTDIV+BT_fCLK2_DIV128,&BTCTL ; ACLK/(256*128)

SetupP10    bis.b #0x0f,&P10DIR ;set 10.0 - 10.3 to Output //setup kryb
            bic.b #0xf0,&P10DIR ;set 10.4 - 10.7 to Input
            bic.b #0xff,&P10OUT ;clear P10 out

SetupP1     bic.b   #3,&P1SEL
            bic.b   #3,&P1DIR
            bis.b   #3,&P1IES
```

```
            bis.b   #3,&P1IE
            bic.b   #0x03,&P1IFG  ; reset of interrupt flag

SetupP2     bic.b   #1,&P2SEL
            bic.b   #1,&P2DIR
            bis.b   #1,&P2IES
            bis.b   #1,&P2IE
            bic.b   #0x01,&P2IFG  ; reset of interrupt flag

BuffEn      bis.b #0x40,&P7DIR ; P7.6 output = #0x40 -->M(P7DIR)
            bis.b #0x40,&P7OUT ; P7.6=1 - #0x40 -->M(P7OUT)
            bis.b #0xff,&P9DIR ; Set P9 as Output
            bic.b #0xff,&P9OUT

SetupADC12  mov.w #SHT0_2+MSC+ADC12ON,&ADC12CTL0
            mov.w #SHP+CSTARTADD_3,&ADC12CTL1
            mov.b #INCH_3,&ADC12MCTL3 ;Analog input is A3, VR+=3.3v VR-
            mov.w #0x08,&ADC12IE ; Enable interrupt
            bis.w #ENC,&ADC12CTL0
            bis.b #0x08,&P6SEL ; P6.3 ADC option select
            bis.w #ENC,&ADC12CTL0

            bis.w   #CPUOFF+GIE,SR ; enter sleep mode
            NOP
;*******************************************************************
;******************Interrupts***************************************
; PORT1 Interrupt
PORT1_ISR  bit.b #0x01,&P1IFG ;check if p1.0 is pressed
           jz CheckP11
           bic.w   #CPUOFF+GIE,SR
           call #ActivateLCD

func3      cmp.b #3,R10
           jnz BackSpinA
           call  #AfterF3
           jmp Continue
BackSpinA  cmp.w #0,R9       ;return the angle to 0
           jz  Continue
           mov.w  #2,R14
           call #FuncPrint ;print Please Wait
           call #BackSpin

Continue   call #Starting    ;starting menu
           jmp  exitP1
```

```
CheckP11    bit.b #0x02,&P1IFG
            jz exitP1
            call #ActivateLCD
            bis.w   #CPUOFF+GIE,SR

exitP1      bic.b   #0x03,&P1IFG
            reti
;*******************************************************
;*******************************************************
; PORT2 Interrupt
PORT2_ISR   call #KeyBoard
            bic.b #0xff,&P10OUT

F3          cmp.b #3,R10
            jnz BackSpinAngle
            call  #AfterF3
            jmp StartCheckKey

BackSpinAngle   cmp.w #0,R9      ;return the angle to 0
                jz  F1
                mov.w  #2,R14
                call #FuncPrint ;print Please Wait
                call #BackSpin

F1          cmp.b #1,R10
            jnz StartCheckKey
            call  #CheckF1
            jmp ExitP2

StartCheckKey   call  #CheckKey
ExitP2          bic.b #0x01,&P2IFG
                reti
;*******************************************************
;*******************************************************
; Basic Timer Interrupt
Basic_Timer_ISR ; Basic Timer Interrupt Service Routine
                mov.w #6,R4
                add.w #1,R15

                cmp.w #57,R15 ;each angle is 6.3
                jn Cont
                mov.w #0,R15
```

```
Cont            call  #Spin
                reti
;*****************************************************
;*****************************************************
;-------------------------------------------------------------------------
            COMMON  INTVEC                  ; Interrupt Vectors-Begins
;-------------------------------------------------------------------------
            ORG     PORT1_VECTOR            ;PORT1 Interrupt Vector
            DW      PORT1_ISR
            ORG     PORT2_VECTOR            ;PORT1 Interrupt Vector
            DW      PORT2_ISR
            ORG     RESET_VECTOR            ; MSP430 RESET Vector
            DW      RESET
            ORG BASICTIMER_VECTOR
            DW Basic_Timer_ISR
            END
```

```
#include  <msp430xG46x.h>
; Lcd_Data MACRO
Lcd_Data     MACRO Char
             Delay #0005000
             bic.b #0xff,&P5OUT
             bis.b #0x20,&P3OUT
             mov.b Char,&P5OUT
             call  #Lcd_strobe
             bic.b #0x20,&P3OUT
             ENDM

;Delay  MACRO
Delay        MACRO  Num
             LOCAL  L1
             mov.w  Num,R13
L1           dec    R13
             jnz    L1
             ENDM

;StepNum  MACRO
StepNum      MACRO  Num
             LOCAL  L1
             mov.w  Num,R14
L1           call   #Step
             dec    R14
             jnz    L1
             ENDM

;BackStepNum  MACRO
BackStepNum     MACRO  Num
             LOCAL  L1
             mov.w  Num,R14
L1           call   #BackStep
             dec    R14
             jnz    L1
             ENDM

;NumOfSteps  MACRO
NumOfSteps MACRO  Num ;get R4*10, Return the numOfSteps in R13
             LOCAL  L1
             mov.w  #7,R11
             mov.w  Num,R14
L1           add.w #1,R13
             sub.w R11,R14
```

```
            cmp.w #0,R14
            jge  L1
            ENDM

;==========================================================================
; Function 1
;==========================================================================
             MODULE    Function1
             PUBLIC    Function1
             EXTERN Lcd_strobe
             EXTERN FuncPrint
             RSEG CODE
Function1
            mov #1,R10
            mov #3,R12
            call #FuncPrint
            ret
;==========================================================================
; Multi - for angle calculate
;==========================================================================
             PUBLIC    Multi
             RSEG CODE

Multi       cmp.b #3,R12
            jnz  second
            mov.w  #100,R14
            JMP    count
second      cmp.b #2,R12
            jnz  third
            mov.w  #10,R14
            JMP    count
third       cmp.b #1,R12
            jnz  exitMulti
            mov.w  #1,R14
            JMP    count
count       add  R5,R4
            dec  R14
            jnz  count
            dec    R12
exitMulti   ret
;==========================================================================
; CheckValid - for angle calculate
;==========================================================================
             PUBLIC    CheckValid
```

```
            RSEG CODE

CheckValid        cmp.b  #'A',R5
                  jge NotValid
                  mov #1,R11
                  jmp   exitCheckValid
NotValid          mov.w #0,R11
exitCheckValid  Lcd_Data R5
                ret
;==================================================================
; CheckAngle - for angle calculate
;==================================================================
            PUBLIC    CheckAngle
            RSEG CODE

CheckAngle
             cmp.w  #361,R4
             jge NotValid2
             mov #1,R11
             jmp  exitCheckAngle
NotValid2    mov.w #0,R11

exitCheckAngle ret
;==================================================================
; Step in positive direction
;==================================================================
Step
           mov.b #0x01,&P9OUT
           Delay  #2000
           mov.b #0x08,&P9OUT
           Delay  #2000
           mov.b #0x04,&P9OUT
           Delay  #2000
           mov.b #0x02,&P9OUT
           Delay  #2000
           ret

;==================================================================
; BackStep in positive direction
;==================================================================
BackStep
           mov.b #0x08,&P9OUT
           Delay  #1500
           mov.b #0x01,&P9OUT
```

```
            Delay  #1500
            mov.b #0x02,&P9OUT
            Delay  #1500
            mov.b #0x04,&P9OUT
            Delay  #1500
            ret
;==========================================================================
; BackSpin
;==========================================================================
             PUBLIC    BackSpin
             RSEG      CODE

BackSpin     cmp.w #256,R9
             jge  ClockWise
             BackStepNum R9
             jmp exitBackSpin

ClockWise    mov.w #516,R13
             sub.w R9,R13
             sub.w #2,R13
             jn exitBackSpin
             StepNum R13

exitBackSpin   clr R9
               clr R13
               ret

;==========================================================================
; Spin
;==========================================================================
             PUBLIC    Spin
             RSEG CODE

Spin        clr R13
            call #Multi10  ;make R4*10
            NumOfSteps R4  ;Return the num of steps in R13
            mov.w  R13,R9
            StepNum R13 ;spin with [R13] steps
            ret

;==========================================================================
; Multi10 - for angle calculate
;==========================================================================
             PUBLIC    Multi10
```

```
            RSEG CODE
Multi10   mov.w  R4,R9
          mov.w  #9,R14
count10   add  R9,R4
          dec  R14
          jnz  count10
          ret
;==================================================================
;==================================================================
               ENDMOD
               END
```

```
#include  <msp430xG46x.h>

;Delay  MACRO
Delay       MACRO  Num
            LOCAL  L1
            mov.w  Num,R13
L1          dec    R13
            jnz    L1
            ENDM

;AngleInVoltage  MACRO
AngleInVoltage MACRO  Num ;get Voltage, Return the Angle ratio to Volta
               LOCAL  L1
            mov.w  #10,R11
            mov.w  Num,R14
L1          add.w #1,R13
            sub.w R11,R14
            cmp.w #0,R14
            jge  L1
            ENDM

;=================================================================
; Function 2
;=================================================================
             MODULE    Function2
             PUBLIC    Function2
             EXTERN FuncPrint
             RSEG CODE
Function2
            mov #2,R10
            call #FuncPrint
            call #SampleADC
            ret
;=================================================================
; SampleADC
;=================================================================
             PUBLIC    SampleADC
             EXTERN Spin
             RSEG CODE
SampleADC
            bis.w #ADC12SC,&ADC12CTL0 ; Start sampling/conversion
            Delay #30000
            mov.w &ADC12MEM3,R9
            AngleInVoltage R9    ;return angle in R13
```

```
            call  #Correction     ;correct the angle~Voltage Ratio
            mov.w  R13,R4         ;mov the angle to R4
            mov.w  #1,R14
            call #FuncPrint ;print Spining
            call #Spin
            mov.w #5,R7       ;print done :)!
            call #FuncPrint
            Delay  #60000     ;the user can see "done"
            Delay  #60000
            ret
 ;=====================================================================
; Correction the angle~voltage Ratio
;=====================================================================
Correction
            cmp.w  #100,R13
            jn Continue
            sub.w #3,R13
            cmp.w  #175,R13
            jn Continue
            sub.w #4,R13
            cmp.w  #230,R13
            jn Continue
            sub.w #7,R13
            cmp.w  #270,R13
            jn Continue
            sub.w #10,R13
            cmp.w  #305,R13
            jn Continue
            add.w #6,R13
            cmp.w  #340,R13
            jn Continue
            add.w #2,R13
Continue    ret
;=====================================================================
;=====================================================================
                ENDMOD
                END
```

```
#include  <msp430xG46x.h>
;NumOfSteps  MACRO
NumOfSteps MACRO  Num ;get R4*10, Return the numOfSteps in R13
           LOCAL  L1
           mov.w  #7,R11
           mov.w  Num,R14
L1         add.w #1,R13
           sub.w R11,R14
           cmp.w #0,R14
           jge  L1
           ENDM

;=================================================================
; Function 3
;=================================================================
            MODULE    Function3
            PUBLIC    Function3
            EXTERN FuncPrint
            RSEG CODE

Function3  mov #3,R10
           call #FuncPrint
           clr  R15
           bis.b #BTIE,&IE2
           ret

;=================================================================
; AfterF3
;=================================================================
            PUBLIC    AfterF3
            RSEG CODE

AfterF3     bic.b #BTIE,&IE2
            clr R10

            mov.w  #2,R14
            call #FuncPrint ;print  Please Wait

            call #BackSpinClock
            clr R9  ;the last spin save 6 angle in R9
            clr R4
            ret
;=================================================================
; BackSpinClock
```

```
;=================================================================
            PUBLIC    BackSpinClock
            EXTERN BackSpin
            EXTERN Spin
            RSEG CODE

BackSpinClock  cmp.w  #0,R15
               jz DontSpin
               cmp.w  #28,R15
               jge  LFoward

L1             mov.w  #9,R9
               call  #BackSpin  ;backspin get the num of steps
               dec  R15
               jnz  L1
               jmp DontSpin

LFoward        mov.w  R15,R9  ;save the num of spins
               mov.w  #57,R15
               sub.w  R9,R15  ;R15 is now the num of 6* angles to compl

L2             mov.w #6,R4
               call  #Spin    ;spin get the num of angle
               dec  R15
               jnz  L2

DontSpin       ret
;=================================================================
;=================================================================
              ENDMOD
              END
```

```
#include  <msp430xG46x.h>
         ORG 1100h
Line1    DC16 'W','e','l','l','c','o','m','e',' ',':',')',0 ; wellcome
Line2    DC16 'P','r','e','s','s',' ','A',' ','t','o',' ','D','o','w','
Line3    DC16 'P','r','e','s','s',' ','B',' ','t','o',' ','U','p',0; A
Line4    DC16 '1','.',' ','S','p','i','n',' ','b','y',' ','a','n','g','
Line5    DC16 '2','.',' ','S','p','i','n',' ','b','y',' ','v','o','l','
Line6    DC16 '3','.',' ','C','l','o','c','k',' ','m','o','d','e',0 ;3.
Line7    DC16 'E','r','r','o','r','!',0 ;Error
Line8    DC16 'P','u','t',' ','a','n','g','l','e',':',0 ;put angle F1
Line9    DC16 'P','u','t',' ','V','o','l','t','a','g','e',':',0 ;put Vo
Line10   DC16 'D','o','n','e',' ',':',')','!',0 ;put Voltage F2
Line11   DC16 'C','l','o','c','k',' ','E','n','a','b''l','e',0 ;Clock E
Line12   DC16 'T','i','k','.','.','T','o','k','.','.',0 ;Clock Enable
Line13   DC16 'S','p','i','n','n','i','n','g',' ','^','_','^',0 ;Clock
Line14   DC16 'P','l','e','a','s','e',' ','W','a','i','t',0 ;Clock Enab
;*********MACRO FUNCTIONS*********************************************
;*************************************************************************
; Lcd_cmd MACRO
Lcd_cmd   MACRO command
          Delay #5000  ; 5msec Delay
          mov.b command,&P5OUT
          call #Lcd_strobe
          ENDM

; Lcd_Data MACRO
Lcd_Data    MACRO Char
            Delay #0005000
            bic.b #0xff,&P5OUT
            bis.b #0x20,&P3OUT
            mov.b Char,&P5OUT
            call  #Lcd_strobe
            bic.b #0x20,&P3OUT
            ENDM

; ExtraDelay MACRO
ExtraDelay MACRO  Num1
           LOCAL  L2,L3
           clr R7
           mov.w  Num1,R7
L2         mov.w  #35,R13      ;1msec for each 1 value, for dutycycle
L3         dec    R13
           jnz    L3
           dec    R7
```

```
            cmp  #0,R7
            jnz L2
            ENDM

; LineLcd  MACRO
LineLcd     MACRO  Line
            LOCAL  newChar,exitLineLcd
            clr R7
            mov    #Line,R7
newChar     mov.b   @R7,R6
            cmp.b  #0,R6
            jz     exitLineLcd
            Lcd_Data R6
            incd R7
            jmp newChar
exitLineLcd
            ENDM

;Delay  MACRO
Delay       MACRO  Num
            LOCAL  L1
            mov.w  Num,R13
L1          dec    R13
            jnz    L1
            ENDM
 ;=================================================================
; P1.0 Starting Menu
;=================================================================
             MODULE    Starting
             PUBLIC    Starting
             EXTERN Lcd_strobe
             RSEG CODE

Starting    LineLcd Line1
            ExtraDelay #10000 ; wait 2 sec to show wellcome
            Lcd_cmd #0x01
            LineLcd Line2
            Lcd_cmd #0xC0
            LineLcd Line3
            Lcd_cmd #0x0C
            mov #1,R8    ; Move Menu
            ret
;=================================================================
; P1.0 Moving Menu
```

```
;=================================================================
            PUBLIC    MovingMenu
            RSEG CODE

MovingMenu Lcd_cmd #0x01
           cmp.b #2,R8
           jnz  Lines23
           LineLcd Line4
           Lcd_cmd #0xC0
           LineLcd Line5
           jmp exitMoving

Lines23    cmp.b #3,R8
           jnz  Lines31
           LineLcd Line5
           Lcd_cmd #0xC0
           LineLcd Line6
           jmp exitMoving
Lines31    LineLcd Line6
           Lcd_cmd #0xC0
           LineLcd Line4

exitMoving   Lcd_cmd #0x0C
             ret
 ;=================================================================
; P1.0 Error
;=================================================================
            PUBLIC    Error
            RSEG CODE

Error      Lcd_cmd #0x01
           LineLcd Line7
           ExtraDelay #10000
           Lcd_cmd #0x01
           call #MovingMenu
           clr R4
           ret

  ;=================================================================
; FuncPrint
;=================================================================
            PUBLIC    FuncPrint
            RSEG CODE
```

```
FuncPrint
            cmp.w  #1,R14
            jnz  Wait
            Lcd_cmd #0xC0
            LineLcd Line13
            jmp exitFuncPrint

Wait        cmp.w  #2,R14
            jnz  Func1
            Lcd_cmd #0x01
            LineLcd Line14
            jmp exitFuncPrint

Func1       cmp.b #1,R10
            jnz  Func2
            Lcd_cmd #0x01
            LineLcd Line8
            jmp exitFuncPrint

Func2       cmp.b #2,R10
            jnz  Func2Extra
            Lcd_cmd #0x01
            LineLcd Line9
            jmp exitFuncPrint

Func2Extra  cmp.b #5,R7
            jnz  Func3
            Lcd_cmd #0x01
            LineLcd Line10
            jmp exitFuncPrint

Func3       cmp.b #3,R10
            jnz  exitFuncPrint
            Lcd_cmd #0x01
            LineLcd Line11
            Lcd_cmd #0xC0
            LineLcd Line12

exitFuncPrint
                ret
;==================================================================
;==================================================================
             END
```

```
#include  <msp430xG46x.h>

 ;=================================================================
;  CheckKey
;=================================================================
            MODULE    CheckKey
            PUBLIC    CheckKey
            EXTERN  Function1
            EXTERN  Function2
            EXTERN  Function3
            EXTERN  Error
            EXTERN  MovingMenu
            RSEG CODE

CheckKey
CheckA      cmp.b #'A',R5 ;A is Pressed
            jnz  CheckB
            add #1,R8
            cmp.b #5,R8
            jnz exitP2  ;update menu
            mov #2,R8
            jmp exitP2  ;update menu

CheckB      cmp.b #'B',R5 ;A is Pressed
            jnz Check1
            sub #1,R8
            cmp.b #2,R8
            jge exitP2
            mov #4,R8
            jmp exitP2

Check1      cmp.b #'1',R5 ;1 is Pressed
            jnz  Check2
            cmp.b #2,R8
            jz StartF1
            cmp.b #4,R8
            jz StartF1
            call #Error
            jmp exitP2F ;dont update the menu
StartF1     call #Function1      ;start Function 1
            jmp exitP2F

Check2      cmp.b #'2',R5 ;2 is Pressed
            jnz  Check3      ;start Function 2
```

```
            clr R4  ;clrz  Check3
            cmp.b #2,R8
            jz StartF2
            cmp.b #3,R8
            jz StartF2
            call #Error
            jmp exitP2F ;dont update the menu
StartF2     call #Function2
            clr R10 ;clr the status regeister
            jmp exitP2

Check3      cmp.b #'3',R5 ;3 is Pressed
            jnz  exitWithError
            cmp.b #3,R8
            jz StartF3
            cmp.b #4,R8
            jz StartF3
            call #Error
            jmp exitP2F ;dont update the menu
StartF3     call #Function3    ;start Function 3
            jmp exitP2F

exitWithError  call #Error
               jmp  exitP2F
exitP2         call #MovingMenu
exitP2F        ret
;==================================================================
; CheckF1
;==================================================================
            PUBLIC    CheckF1
            EXTERN CheckValid
            EXTERN Multi
            EXTERN CheckAngle
            EXTERN Spin
            EXTERN Error
            EXTERN MovingMenu
            EXTERN FuncPrint
            RSEG CODE
CheckF1
            call #CheckValid ; check if R5 is 0~9
            bit.b #1,R11
            jz exitFromF1

            sub.w #'0',R5   ; convert ascii to binary
```

```
            call  #Multi
            cmp.w #0,R12  ;R12-0
            jnz exitF1
                          ;after the 3rd digit
            call  #CheckAngle
            bit.b #1,R11
            jz exitFromF1
            mov.w  #1,R14
            call #FuncPrint ;print Spining
            call #Spin    ;R4 hold the angle in binary
            clr R4  ;clr the angle in binary
            clr R10 ;clr the status regeister
            jmp exitCheckP2

exitFromF1      clr R10
                call #Error
                jmp exitF1
exitCheckP2    call #MovingMenu
exitF1         ret
;=================================================================
;=================================================================
              ENDMOD
              END
```

```
#include  <msp430xG46x.h>

;*********MACRO FUNCTIONS*********************************************
;*************************************************************************
;Delay  MACRO
Delay       MACRO   Num
            LOCAL   L1
            mov.w   Num,R13
L1          dec     R13
            jnz     L1
            ENDM
;=========================================================================
; KeyBoard - find the ascii value of the key that been pressed;
;=========================================================================
            MODULE    KeyBoard
            PUBLIC    KeyBoard
            RSEG CODE
KeyBoard
            clr   R5
            mov.b #'Z',R5 ; for comparition later on
CheckPress

            mov.b #0x07,&P10OUT
            bit.b #0x80,&P10IN
            jnz N1
            mov.b #'F',R5
N1          bit.b #0x40,&P10IN
            jnz N2
            mov.b #'E',R5
N2          bit.b #0x20,&P10IN
            jnz N3
            mov.b #'D',R5
N3          bit.b #0x10,&P10IN
            jnz N4
            mov.w #'C',R5

N4          mov.b #0x0b,&P10OUT
            bit.b #0x80,&P10IN
            jnz N5
            mov.b #'B',R5
N5          bit.b #0x40,&P10IN
            jnz N6
            mov.b #'9',R5
```

```
N6          bit.b #0x20,&P10IN
            jnz N7
            mov.b #'6',R5
N7          bit.b #0x10,&P10IN
            jnz N8
            mov.b #'3',R5

N8          mov.b #0x0d,&P10OUT
            bit.b #0x80,&P10IN
            jnz N9
            mov.b #'0',R5
N9          bit.b #0x40,&P10IN
            jnz N10
            mov.b #'8',R5
N10         bit.b #0x20,&P10IN
            jnz N11
            mov.b #'5',R5
N11         bit.b #0x10,&P10IN
            jnz N12
            mov.b #'2',R5

N12         mov.b #0x0e,&P10OUT
            bit.b #0x80,&P10IN
            jnz N13
            mov.b #'A',R5
N13         bit.b #0x40,&P10IN
            jnz N14
            mov.b #'7',R5
N14         bit.b #0x20,&P10IN
            jnz N15
            mov.b #'4',R5
N15         bit.b #0x10,&P10IN
            jnz Check
            mov.b #'1',R5

Check       cmp #'Z',R5
            jz CheckPress
            Delay #50000
            Delay #50000
            ret
            ENDMOD
;==================================================================
;==================================================================
            END
```