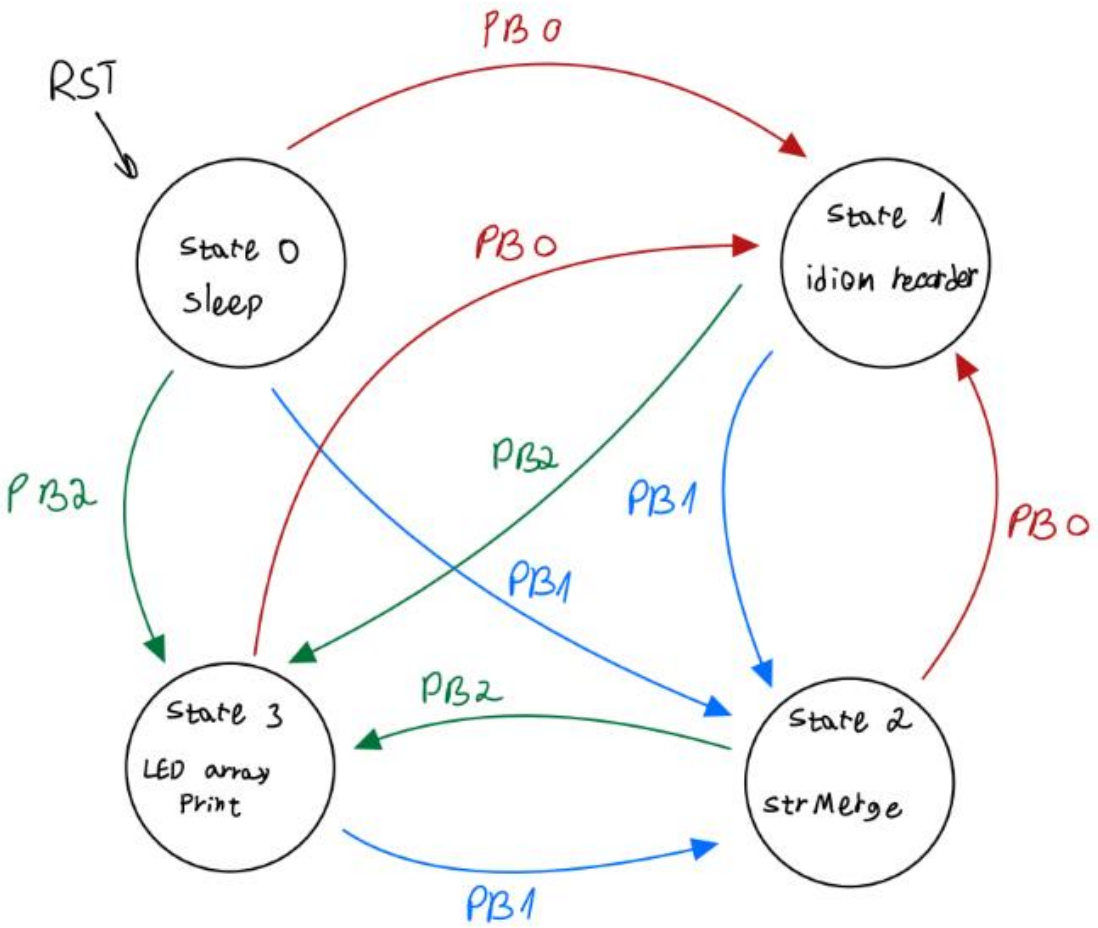
Digital Computer Structure – Preparation Report Lab 3



By:

Itamar Meir, 208536888

Michael Leib, 319095832

C. שאלות הכנה – MSP430 DMA module:

1) הסבר/י בפירוט את המוטיבציה לשימוש ב DMA ביחס ל- CPU לצורך העברת נתונים (מהו ה Tradeoff).

ה-CPU הוא הרכיב הכי חשוב בבקר. הוא אחראי על ביצוע פעולות חישוביות ועל ביצוע התוכנית. לכן, אם יש צורך בביצוע פעולה מסויימת שלוקחת הרבה זמן ומעסיקה את ה-CPU, עדיף לעשות שימוש בחומרה נוספת, ייעודית לאותה הפעולה, כדי שהמעבד יוכל להתפנות ולבצע פעולות נוספות. היתרון של חומרה ייעודית שכזו הוא שהיא תוכננה במיוחד לביצוע פעולות מסוג מסויים כך שהיא בד"כ יותר יעילה ויותר מהירה מהמעבד בביצוע פעולה זו. כך גם הרווחנו מהירות וגם המעבד פנוי לביצוע פעולות אחרות במקביל. רכיב ה-DMA מסוגל להעתיק בלוקי מידע מהזיכרון לרכיבי חומרה אחרים בבקר (ולהיפך) ביעילות ובמהירות רבה יותר מהמעבד וכך הרווחנו זמן והספק.

החיסרון של בקר בעל DMA הוא בראש ובראשונה התווספות של חומרה נוספת המייקרת את המעבד. בנוסף, מורכבות השימוש בבקר עולה כי כדי להשתמש ב-DMA יש להכיר את החומרה טוב ולדעת איך להשתמש בו (ואיך לקנפג אותו). בנוסף, ה-DMA עושה שימוש ב-BUS של המעבד ולכן נוצרת "תחרות" על משאב זה בין הרכיב למעבד. דבר זה עלול להשפיע על ביצועי המערכת. חיסרון נוסף הוא שלמרות ה-DMA עוזר למעבד בהעברת נתונים ממקום למקום, המעבד צריך לנהל וקטורי פסיקה נוספים, בנוסף לאלו הקיימים באופן רגיל. כך אם ה-DMA פוסק באופן תדיר המעבד מועמס בריבוי פסיקות.

2) הסבר/י בפירוט את ארבע השיטות המעון, רשמו דוגמה מתאימה עבור כל אחת מהשיטות.

ישנן ארבע שיטות מיעון להעברת מידע:

א. מכתובת קבועה לכתובת קבועה. דוגמא: רכיב INPUT מסוים שולח מידע לכתובת ספציפית, ואנו רוצים לשמור מידע זה בכתובת ספציפית בזיכרון.
ב. מכתובת קבועה לבלוק של כתובות. דוגמא: רכיב תקשורת סריאלית שולח מידע רב לכתובת קבועה ואנו רוצים להעביר את המידע הזה לבלוק בזיכרון.
ג. העברת מידע מבלוק של כתובות לכתובת קבועה. דוגמא: הכיוון השני של סעיף ב'.
ד. העברת מידע מבלוק של כתובות לבלוק של כתובות: נניח העברת כמות מידע ממקום אחד בזיכרון למקום אחר בזיכרון.

3) הסבר/י בפירוט את שש השיטות להעברת מידע בשימוש DMA, רשמו דוגמה מתאימה עבור כל אחת מהשיטות.

א. העברה בודדת – כל העברה של בייט או מילה בודדת שדורשת טריגר נפרד ולאחר ההעברה DMAEN מתאפס (הביט שמפעיל את הערוץ הספציפי). דוגמא: תקשורת סיריאלית – העברת מידע מזיכרון ה-UART לבלוק בזיכרון.
ב. העברת בלוק – בלוק שלם מועבר בעת טריגר. דוגמא: העברת מידע מהזיכרון לרכיב ה-DAC.
ג. העברת BURST – בהעברה זו ה-DMA מעביר 4 בתים או מילים ולאחר מכן ה-BUS פנוי לשימוש ה-CPU למשך 2 מחזורי שעון. דוגמא: נשתמש במצב זה כאשר ה-DMA מעביר כמויות גדולות של מידע ולכן ה-CPU לא יוכל להשתמש ב-BUS להרבה זמן. לכן במצב כזה נעדיף להשתמש ב-BURST.
ד. העברה בודדת חוזרת – כמו א' רק ש-DMAEN לא מתאפס וכך מתאפשרת העברה רצופה שלמידע כשמגיע טריגר (נניח כשהערוץ מקבל בלוק). דוגמא: קריאת מידע מחיישן.
ה. העברת בלוק חוזרת – כמו ב' רק ש- DMAEN לא מתאפס. דוגמא: העברת מידע מה-ADC נניח כשמדובר במידע של שמע.
ו. העברת BURST חוזרת – כמו ג' רק ש- DMAEN לא מתאפס. דוגמא: שידור וקבלת מידע מרכיב UART שדורש תקשורת רציפה.

4) הסבר/י כיצד ניתן להשתמש ב DMA לצורך העברת מידע מהמודולים TimerB, ADC12, DAC12, רשמו דוגמה מתאימה עבור כל אחת מהשיטות.

ניתן להשתמש כך:

```
#include <msp430.h>

#define BUFFER_SIZE 128
volatile uint16_t timerBuffer[BUFFER_SIZE];

void setupDMA_TimerB(void) {
    // Configure TimerB to generate an overflow event
```

```
    TB0CTL = TBSSEL_2 + MC_2 + TBCLR; // SMCLK, continuous mode, clear timer

    // Configure DMA channel 0
    DMACTL0 = DMA0TSEL_1; // TimerB overflow as trigger source
    __data16_write_addr((unsigned short)&DMA0SA, (unsigned long)&TB0R); // Source address: TimerB
register
    __data16_write_addr((unsigned short)&DMA0DA, (unsigned long)timerBuffer); // Destination address:
buffer
    DMA0SZ = BUFFER_SIZE; // Transfer size
    DMA0CTL = DMADT_0 + DMADSTINCR_3 + DMAEN; // Single transfer, increment destination, enable DMA
}

void setupDMA_DAC12(void) {
    // Configure DAC12
    DAC12_0CTL = DAC12IR + DAC12AMP_5 + DAC12ENC; // Internal reference, enable DAC

    // Configure DMA channel 1
    DMACTL0 = DMA1TSEL_7; // DAC12IFG as trigger source
    __data16_write_addr((unsigned short)&DMA1SA, (unsigned long)&DAC12_0DAT); // Source address: DAC12
data register
    __data16_write_addr((unsigned short)&DMA1DA, (unsigned long)dacBuffer); // Destination address:
buffer
    DMA1SZ = BUFFER_SIZE; // Transfer size
    DMA1CTL = DMADT_0 + DMADSTINCR_3 + DMAEN; // Single transfer, increment destination, enable DMA
}

void setupDMA_ADC12(void) {
    // Configure ADC12
    ADC12CTL0 = SHT0_2 + MSC + ADC12ON; // Sampling time, multiple sample conversion, ADC12 on
    ADC12CTL1 = SHP + CONSEQ_2; // Use sampling timer, repeated single channel
    ADC12MCTL0 = INCH_0; // Channel 0 (A0)
    ADC12CTL0 |= ENC + ADC12SC; // Enable conversion and start conversion

    // Configure DMA channel 2
    DMACTL1 = DMA2TSEL_24; // ADC12IFGx as trigger source
    __data16_write_addr((unsigned short)&DMA2SA, (unsigned long)&ADC12MEM0); // Source address: ADC12
memory register
    __data16_write_addr((unsigned short)&DMA2DA, (unsigned long)adcBuffer); // Destination address:
buffer
    DMA2SZ = BUFFER_SIZE; // Transfer size
    DMA2CTL = DMADT_0 + DMADSTINCR_3 + DMAEN; // Single transfer, increment destination, enable DMA
}

int main(void) {
    //-------------------- TIMER_B --------------------
    WDTCTL = WDTPW | WDTHOLD; // Stop watchdog timer

    setupDMA_TimerB();
```

```
    __bis_SR_register(LPM0_bits + GIE); // Enter low-power mode and enable global interrupts
    // -------------------------------------------------------

    //-------------------- DAC12 --------------------
     WDTCTL = WDTPW | WDTHOLD; // Stop watchdog timer

    setupDMA_DAC12();

    __bis_SR_register(LPM0_bits + GIE); // Enter low-power mode and enable global interrupts
    // -------------------------------------------------------

    //-------------------- ADC12 --------------------
     WDTCTL = WDTPW | WDTHOLD; // Stop watchdog timer

    setupDMA_ADC12();

    __bis_SR_register(LPM0_bits + GIE); // Enter low-power mode and enable global interrupts
}
```

5) הסבר/י את המושג *DMA Channel Priorities* ומדוע יש צורך בו.

ה-DMA לא יכול להעביר מידע בו זמנית עבור יותר מערוץ אחד. לכן, אם שניים או שלושה ערוצים דורשים העברה בו זמנית, יש צורך לטפל בעדיפויות. כמו כן, יש רכיבים שדחופים יותר לטיפול מאשר רכיבים אחרים, בהתאם לדרישות המערכת. סדר העדיפויות הדיפולטיבי הוא: DMA2 -- DMA1 -- DMA0 (DMA0 הכי גבוה). אם מתרחשת העברה ובאמצע ערוץ עם סדר עדיפות גבוה יותר מבקש טיפול, ה-DMA קודם יסיים את ההעברה הנוכחית ורק אז יטפל בערוץ הגבוה.

ישנה אפשרות נוספת לשינוי סדר העדיפויות ע"י ביט ROUNDROBIN. כאשר ביט זה דולק, הערוץ שפעל לאחרונה עובר לעדיפות האחרונה.

6) הסבירו כיצד מורכב ה- *DMA Transfer Cycle Time* (זמן העברה בפועל + תקורה) במקרים הבאים:

- <u>Case 1:</u>
  CPU Operating Mode - Active mode
  CPU Clock Source MCLK - DCOCLK
- <u>Case 2:</u>
  CPU Operating Mode - Low-power mode LPM0/1
  CPU Clock Source MCLK - DCOCLK

בכל המקרים:

1-2 מחזורים לסנכרון לפני העברה (תלוי מצב CPU). כול בית או מילה לוקחת 2 מחזורים. עוד מחזור נוסף של המתנה אחרי ההעברה. כלומר 4-5 מחזורים (תלוי מצב CPU).

מקרה 1: 4 מחזורים. מקרה 2: 5 מחזורים.

7) הסבר/י באילו תנאים תתבצע בקשת פסיקה של מודול DMA.

ה-DMA יבצע בקשת פסיקה כאשר העברה של ערוץ מסויים התבצעה (כאשר רגיסטר DMAxSZ מגיע ל-0). אם בנוסף ה-GIE וה-DMAIE דולקים תתבצע הפסיקה.

8) הסבר/י האם מודול פריפריאלי יכול לבצע בקשת DMA ובקשת פסיקה בו זמנית.

לא ניתן גם לבצע העברה של DMA וגם לבצע פסיקה. בזמן ביצוע ההעברה של ה-DMA הפסיקות האחרות יישארו כPENDING וכשיסיים יוכל ה-CPU לטפל בהן. מקרה מיוחד הוא NMI שהן פסיקות שכן יכולות לקטוע פעולת DMA. בנוסף, פעולות פסיקה של מודולים אחרים יכולות להיקטע על ידי פסיקה של ה-DMA ולכן יש לכבות את בקשת הפסיקה מהDMA אם ברצוננו שהפעולה הרצויה לא תיקטע באמצע.