**Digital Computer Structure – Final Report Lab 3**

**By:**

**Nachman Mimoun 321730558**

**Yarden Levi, 206212276**

במטלת זמן האמת נתבקשנו להוסיף STATE 4 ל-FSM הקיים שלנו שבו עלינו לבצע מיזוג בין שתי מחרוזות באמצעות .DMA המיזוג מתבצע על ידי שימוש בשני ערוצי DMA באופן עקבי, כאשר המחרוזת השנייה מתווספת בסוף המחרוזת הראשונה.

הגדרנו בקוד הראשי (main.c) את שתי המחרוזות הנדרשות:

// string vars for state4

char str1[] = "An apple a day keeps the doctor away ";

char str2[] = "Strike while the iron's hot";

לצורך מימוש המיזוג, הגדרנו פונקציה בשם merge() ב:API-

void merge(void) {

char strMerge[80]; // Buffer for merged string

char sliced[33]; // Buffer for display (32 chars + null)

int i, max\_idx;

int len1 = strlen(str1);

int len2 = strlen(str2);

int strLen;

int displayed = 0;

unsigned int keyInput;

// Clear display and show initial message

lcd\_clear();

lcd\_puts("Merging strings...");

// Clear output buffer

memset(strMerge, 0, 80);

// Configure DMA for string merging

startDMA();

// Set up DMA0 (higher priority, for first string)

DMA0\_Src\_Add = (unsigned int)str1;

DMA0\_Dst\_Add = (unsigned int)strMerge;

DMA0\_Size = len1;

// Set up DMA1 (for second string)

DMA1\_Src\_Add = (unsigned int)str2;

DMA1\_Dst\_Add = (unsigned int)(strMerge + len1); // Start at the end of first string

DMA1\_Size = len2;

// Start DMA0 transfer (higher priority)

triggerDMA0();

*// Enter low power mode, wait for DMA0 interrupt*

enterLPM(mode0);

// Start DMA1 transfer

triggerDMA1();

*// Enter low power mode, wait for DMA1 interrupt*

enterLPM(mode0);

// Enter low power mode, wake up by other interrupts

enterLPM(mode0);

// Ensure null termination

strMerge[len1 + len2] = '\0';

// Get total string length

strLen = len1 + len2;

// Display the first 32 characters initially

lcd\_clear();

strncpy(sliced, &strMerge[0], 32);

sliced[32] = '\0'; // Ensure null termination

lcd\_puts(sliced);

displayed = 32;

// Main operation loop for displaying string with pagination

while (state == state4) {

// Wait for keypad input

Key = 16; // Reset key value

en\_keypad\_interrupts();

enterLPM(mode0);

disable\_keypad\_interrupts();

// Check if state changed

if (state != state4) {

break;

}

keyInput = Key;

Key = 16; // Reset key

// Check if \* key pressed for scrolling

if (keyInput == 12) { // \* key

// Handle long strings with scrolling

if (strLen > displayed) { // If there are more characters to display

lcd\_clear();

max\_idx = MIN(32, strLen - displayed);

strncpy(sliced, &strMerge[displayed], max\_idx);

sliced[max\_idx] = '\0'; // Ensure null termination

// Display next screen

lcd\_puts(sliced);

displayed += max\_idx;

} else {

// No more characters to display, return to idle state

state = state0;

break;

}

}

}

// Return to idle state if we exit the loop

state = state0;

}

בתוך פונקציית startDMA() בשכבת ,HAL הוספנו את הקונפיגורציה הנדרשת עבור :state4

else if(state==state4) { // state 4 - String Merge

// Configure DMA0 for first string transfer

DMA0CTL = DMADT\_1 + DMASBDB + DMASRCINCR\_3 + DMADSTINCR\_3 + DMAIE; // block, byte-to-byte, src inc, dst inc, interrupt enabled

// Configure DMA1 for second string transfer

DMA1CTL = DMADT\_1 + DMASBDB + DMASRCINCR\_3 + DMADSTINCR\_3 + DMAIE; // block, byte-to-byte, src inc, dst inc, interrupt enabled

}

כמו כן, הוספנו פונקציות חדשות לטריגור הערוצים השונים:

void triggerDMA0(void) {

DMA0CTL |= DMAEN + DMAREQ; // Enable DMA0 and trigger transfer

}

void triggerDMA1(void) {

DMA1CTL |= DMAEN + DMAREQ; // Enable DMA1 and trigger transfer

}

והוספנו טיפול בפסיקות ה-DMA בפונקציית ה-ISR:

#pragma vector=DMA\_VECTOR

\_\_interrupt void DMA\_ISR(void) {

// Check which DMA channel triggered the interrupt

if (DMA0CTL & DMAIFG) {

DMA0CTL &= ~DMAIFG; // Clear DMA0 interrupt flag

// Exit low power mode if it was waiting

switch (lpm\_mode) {

case mode0: LPM0\_EXIT; break;

case mode1: LPM1\_EXIT; break;

case mode2: LPM2\_EXIT; break;

case mode3: LPM3\_EXIT; break;

case mode4: LPM4\_EXIT; break;

}

}

else if (DMA1CTL & DMAIFG) {

DMA1CTL &= ~DMAIFG; // Clear DMA1 interrupt flag

// Exit low power mode if it was waiting

switch (lpm\_mode) {

case mode0: LPM0\_EXIT; break;

case mode1: LPM1\_EXIT; break;

case mode2: LPM2\_EXIT; break;

case mode3: LPM3\_EXIT; break;

case mode4: LPM4\_EXIT; break;

}

}

}

התהליך כולו עובד כך:

1. כאשר המשתמש לוחץ על לחצן PB0, המערכת עוברת ל.state4 -
2. בפונקציית ,merge()המערכת מקנפגת את ה DMA -לביצוע המיזוג:
   * DMA0 מעתיק את המחרוזת הראשונה (str1) למערך היעד .(strMerge)
   * DMA1 מעתיק את המחרוזת השנייה (str2) לסוף המחרוזת הראשונה במערך היעד .(strMerge + len1)
3. לאחר סיום המיזוג, המערכת מציגה את ה-32 התווים הראשונים של המחרוזת הממוזגת על גבי ה-LCD.
4. המשתמש יכול ללחוץ על מקש \* כדי לגלול ולראות את ה-32 התווים הבאים, וכן הלאה עד סוף המחרוזת.
5. בסיום הצגת כל המחרוזת, או אם המשתמש משנה את ה state -המערכת חוזרת למצב idle .(state0)

שיטה זו מאפשרת מיזוג יעיל של מחרוזות באמצעות ,DMA ללא צורך בהתערבות ה CPU -ומציגה את התוצאה באופן נוח למשתמש עם תמיכה בדפדוף.