Teknisk Rapport Digitala System - Memory

Viktor.A, Tobias.C, Joachim.M.

Sammanfattning

Projektet är ett memoryspel som är byggt på Atmega1284 processorn. Projektet går ut på att man ska se vilken lampa som tänds sedan ska man klicka på den knapp som är rätt associerad med lampan. Om man klickar rätt kommer samma lampa eller lampor att lysa i samma sekvens och sedan en ny, och så fortsätter man tills man förlorar, klickar på fel knapp.

Nyckelord

Atmega-1284, Memoryspel, lampor, sekvens, Push-Down-Button

Abstract

This project is a memory game that is built on the processor Atmega1284. The project works as the one who plays the game is supposed to press the right corresponding button to a light in the right order. As the order has been pressed correctly, another light gets pushed to the sequence and the game continues. This continues until the gamer presses the wrong button or does not press a button in time.

Keywords

Atmega-1284, Memory-Game, lights, sequence, Push-Down-Button

Inledning

Detta projekt görs då kunskaper inom Atmega1284 ska bevisas. Dels genom programmering, problemlösning och hur man kopplar komponenter direkt till processorn med hänvisningar från dess dokumentation.

Målet med projektet är att processorn ska kunna se vilken lampa som lyser, känna vilken knapp som trycks och få det registrerat hos processorn samt kunna känna av rätt sekvens och exekvera olika uppgifter beroende på hur spelet spelas.

Val av arbetet gjordes då ett memory spel är tillräckligt komplicerat med den tid som gavs för projektet. Samt att det kan lätt byggas ut då det finns tidsrum för det.

Teknisk bakgrund

Atmega 1284

Atmega1284 är en 8-bitars mikrokontroller som har funktioner som fungerar till inbäddade system med bland annat en klocka på 20 MHz, 32 GPIO pinnar samt 32 registrar. Den har även en SPI serial port som ger möjlighet att skicka data till en dator via usb med serialinterface.

2x16 LED display

2x16 LED display är en skärm som tillåter dig att både skriva ut totalt 32 tecken men även läsa av vad som står på skärmen. För att skärmen ska fungera måste man då först berätta för den vad det är den ska göra, tillsammans med hur många rader den ska hantera/använda samt även starta själva skärmen. Dessa skärmarna är inte aktiva som standard och skrivbara bara för att den har kopplats in till ström utan de måste även startas för att kunna se vad skärmen visar.

Push Down Button

Vid användning av en tvåpinnad knapp kan man göra antingen en pull-up eller push-down resistor vilket fungerar på ungefär samma sätt. Beroende på vad man vill ha då knappen inte är nedtryckt så väljer man mellan de olika. Vill man ha en etta då knappen inte är nedtryckt ska man använda en pull-up resistor medan push-down resistor då man vill ha en nolla. För en bild på hur kretsen för en pull-down resistor ser ut, kolla under appendix kretsschema.

Metod

Projektet genomfördes genom att det först ritades ett kretsschema på alla komponenter som ska inkluderas i MVP, minimum viable product. Efter att kretsen blivit godkänd för fortsättning plockades alla komponenter och sattes på ett THB, Through Hole breadboard. När de första knapparna hade fäst, testades de, vilket då gjorde att det upptäcktes att kopplingsschemat inte var fullständigt. Då kopplades alla knappar om efter ett fungerande kretsschema och alla komponenter fungerarade. När alla lampor och knappar hade testats, och såg att de fungerade, började programmet att programmeras.

För att programmet skulle fungera behövdes det en RNG, Random Number Generator, men då atmega1284 inte hade ett paket för slumpmässiga tal utan att behöva ett seed[1], användes då timer som börjar räkna när projektet startas efter att den inte haft ström för att sedan räkna tills en knapp blir nedtryckt och den tiden kommer sen att användas som en seed.

Efter att programmet testats och fått de största buggarna fixade, lödades sladdarna om och fick den längd som de minst behövde.

Analys

Tekniska dokument för komponenter hade kunnat läsas igenom mer grundligt innan virning, lödning och programmering hade påbörjats. Tid hade då kunnat sparats och istället använts för att lägga till andra komponenter som gör det lättare och roligare för den som spelar att spela spelet. Varav kunde vara flera spelvariationer av memory.

Det största problemet med projektet var att göra kopplingsschemat för knapparna då det tog flera dagar för det att bli rätt. Det som hade kunnat göras är att man följt schemat för knappar enligt andra projekt.

Ett problem med displayen var timingen för att skriva och programmera den. Det tog flera timmar innan problemet blev löst. Det hade kunnat vara till hjälp om vi hade kollat på andra projekt för att se hur det hanterade problemet. Problemet löstes genom att lägga in en fördröjning på 1 mikrosekund efter varje registerskrivning för att skärmen ska kunna registrera att den har fått ny data.

Resultat

Slutversionen på memory spelet fungerar som avsett med rätt lampa associerad med rätt knapp och en display som visar hur mycket poäng spelaren har lyckats få eller fick under spelets omgång. Förinställd skärmlayout skrivs ut efter varje spelrunda beroende på om man har vunnit eller förlorat. Skärmen förklarar tydligt hur mycket poäng man får samt även berättar hur man startar nästa spelomgång.

Har man förlorat kommer den lampan som är associerad med knapp ett att lysa för att förtydliga förlusten, vilket är en röd LED. Däremot om man lyckades vinna spelet kommer fjärde knappens LED lampa istället att lysa, som lyser grönt.

För att “vinna” spelet i slutversionen är du tvungen att få tusen knappar i korrekt sekvens. Efter varje runda läggs det till en ny lampa på samma sekvens och fortsätter tills spelet är slut. Efter ett spel är slut kommer nästa spel inte att följa samma sekvens som det förra spelet hade.

Slutsats

MVP målet som sattes i början av projektet blev gjort i slutet. Skärmen, knapparna och programmet blev gjort så att spelet fungerade som det planerades. Om dokumentation och research hade gjorts bättre hade det med sparad tid kunnat bli större och mer roligt genom att det skulle finnas fler lägen på

[1] Nummer som bestämmer vad sekvensen med nummer som används.

spelet med lampor som hade kunnat lysa i flera färger istället för enbart en. Men då det enbart är saker som gjort det bättre, är det inget viktigt och projektet fungerade som det ska.

Källförteckning

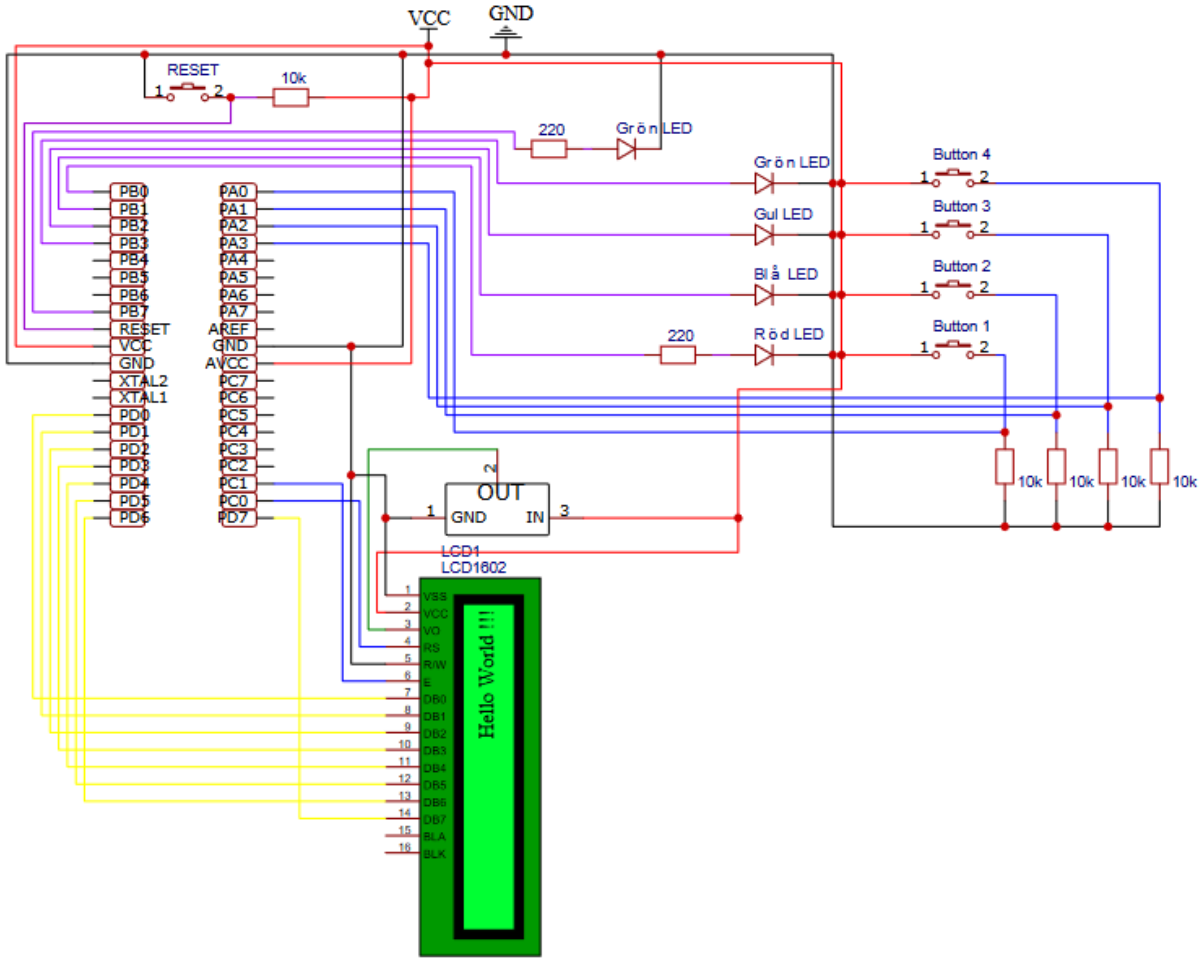
1. Atmel Corporation (2023), “ATmega1284”. https://www.microchip.com/en-us/product/ATmega1284 [2023-05-18]

2. Xiamen Ocular (2001) “GDM1602K”. https://www.sparkfun.com/datasheets/LCD/GDM1602K-Extended.pdf [2023-05-18]

3. Electronics Tutorials (2023) “Pull-up Resistors and Pull-down Resistors. https://www.electronics-tutorials.ws/logic/pull-up-resistor.html [2023-05-18]

Appendix

Kretsschema



Atmega1284 kod

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>

#define F_CPU 20000000UL

#include <avr/io.h>
#include <util/delay.h>
#include <stdlib.h>

void start_sec();
int rnd_number();
int simon_game();
void not_win();
void win_win();
void start_new_game();
void show_led();
void new_game_seq();
int get_button();
void display_set_char();
void display_setup();
void display_clear();
void display_set_number();

#define MAX_SEQUENCE_LENGTH 100
int game_vector[MAX_SEQUENCE_LENGTH];

int round_game_vector[MAX_SEQUENCE_LENGTH];
int pressed_vector[MAX_SEQUENCE_LENGTH];

int time_to_get = 0;

int main(void){
    DDRA = 0x00; // Set PA as input
    DDRB = 0xff; // Set PB as output
    DDRC |= 0b00000011;
    DDRD = 0xff; //Screen programing pins

    start_sec();

    display_setup();

    char arr[] = "BTN 1 to start";
    int size = sizeof(arr) / sizeof(arr[0]);
    display_set_char(arr, size, 1);
```

```
    int wat = 1;
    while(wat){
        if(PINA & 0b00000001){
            wat = 0;
        }
        time_to_get++;
    }
    srand(time_to_get);

    new_game_seq();
    display_clear();
    simon_game();
    while(1){}
}
void display_set_number(int num, int row){
    int num2 = num;
    int count = 0;
    while (num2 != 0) {
        num2 /= 10;
        ++count;
    }

    char arr2[count+1];
    int size2 = sizeof(arr2) / sizeof(arr2[0]);
    sprintf(arr2, "%d", num);
    display_set_char(arr2, size2, row);
}
void check_game_vector_values(){
    for (int i = 0; i<MAX_SEQUENCE_LENGTH; i++){
        show_led(8);
        _delay_ms(10);
        show_led(game_vector[i]);
        _delay_ms(10);
    }
}
void show_led(int choosen){
    switch(choosen){
        case 0:
        PORTB = 0b00000000;
        break;
        case 1:
        PORTB = 0b00000001;
        break;
        case 2:
        PORTB = 0b00000010;
        break;
        case 3:
        PORTB = 0b00000100;
```

```
            break;
            case 4:
            PORTB = 0b00001000;
            break;
            default:
            PORTB = 0b10000000;
            break;

    }
}
int get_button(){
    int waiting = 0;
    int noPressTime = 250;
    int loop = 0;

    while (waiting != 1){
            if (PINA & 0b00000001){ // Check if the first button on PA is pressed
                    waiting = 1;
                    while(PINA & 0b00000001){}
                    return 1;
            }

            if (PINA & 0b00000010){ // Check if the first button on PA is pressed
                    waiting = 1;
                    while(PINA & 0b00000010){}
                    return 2;
            }

            if (PINA & 0b00000100){ // Check if the first button on PA is pressed
                    waiting = 1;
                    while(PINA & 0b00000100){}
                    return 3;
            }

            if (PINA & 0b00001000){ // Check if the first button on PA is pressed
                    waiting = 1;
                    while(PINA & 0b00001000){}
                    return 4;
            }

            if(loop >= noPressTime){
                    waiting = 1;
                    return 0;
                    }else{
                    loop++;
            }

            time_to_get++;
```

```
            _delay_ms(1);
    }
    return 0;
}
void start_sec(){
    int delay_amount = 15;
    _delay_ms(delay_amount);
    PORTB = 0b00000001;
    _delay_ms(delay_amount);
    PORTB = 0b00000010;
    _delay_ms(delay_amount);
    PORTB = 0b00000100;
    _delay_ms(delay_amount);
    PORTB = 0b00001000;
    _delay_ms(delay_amount);
    PORTB = 0b10000000;
    _delay_ms(delay_amount);
    PORTB = 0b11111111;
    _delay_ms(delay_amount);
    PORTB = 0x00;
}
int rnd_number(){
    uint8_t random_number = (rand() % 4) + 1;
    return random_number;
}
void new_game_seq(){
    for (int i = 0; i<MAX_SEQUENCE_LENGTH; i++){
            game_vector[i] = rnd_number();
    }
}
int simon_game(){
    int round = 1;
    int win_amount = 1000;

    while(round < win_amount+1){
            display_clear();
            char arr[] = "Score: ";
            int size = sizeof(arr) / sizeof(arr[0]);
            display_set_char(arr, size, 2, 1);

            display_set_number(round-1, 2);

            for (int i = 0 ; i<round; i++){
                    _delay_ms(20);
                    show_led(game_vector[i]);
                    _delay_ms(20);
                    show_led(0);
            }
```

```
            show_led(8);

            for (int j = 0; j<round; j++){
                int pressed_button = get_button();
                if(pressed_button != game_vector[j]){
                    not_win(round-1);
                    return 0;
                }
            }
            round++;

            show_led(0);
        }
        win_win(round-1);
        return 1;
    }
    void not_win(int round){
        int loss = 1;

        display_clear();
        char arr[] = "You Lost: ";
        int size = sizeof(arr) / sizeof(arr[0]);
        display_set_char(arr, size, 1, 1);

        display_set_number(round, 1);

        char arr2[] = "BTN 1 to Start!";
        int size2 = sizeof(arr2) / sizeof(arr2[0]);
        display_set_char(arr2, size2, 2, 1);

        PORTB = 0b00000001;
        while(loss){
            if (PINA & 0b00000001){ // Check if the first button on PA is pressed
                loss = 0; // Turn on the first LED on PB
            }
        }
        PORTB = 0b00000000;
        _delay_ms(50);
        start_new_game();
    }
    void win_win(int round){
        int loss = 1;

        display_clear();
        char arr[] = "You Won: ";
        int size = sizeof(arr) / sizeof(arr[0]);
        display_set_char(arr, size, 1, 1);
```

```
    display_set_number(round, 1);

    char arr2[] = "BTN 1 to Start!";
    int size2 = sizeof(arr2) / sizeof(arr2[0]);
    display_set_char(arr2, size2, 2, 1);

    while(loss){
        PORTB = 0b00001000;
        if (PINA & 0b10000001){ // Check if the first button on PA is pressed
            loss = 0; // Turn on the first LED on PB
        }
    }
    start_new_game();
}
void start_new_game(){
    display_clear();
    new_game_seq();
    simon_game();
}
void display_setup(){
    PORTC = 0b00000010;
    _delay_us(10);
    PORTD = 0b00111100;    //Function set
    _delay_us(10);
    PORTC = 0b00000000;
    _delay_us(10);

    PORTC = 0b00000010;
    _delay_us(10);
    PORTD = 0b00001100;
    _delay_us(10);
    PORTC = 0b00000000;
    _delay_us(10);

    display_clear();
}
void display_clear(){
    PORTC = 0b00000010;
    _delay_us(1);
    PORTD = 0b00000001;
    _delay_us(1);
    PORTC = 0b00000000;
    _delay_us(1);
    PORTC = 0b00000011;
    _delay_us(1);
}
void display_set_char(char str[], int size, int row, int cur_start){
```

```
int _cur_start = 0;

if(cur_start == 1){
	_cur_start = 1;
}

if(row == 0){
	display_clear();
}

if(_cur_start == 1){
	PORTC = 0b00000010;
	_delay_us(1);
	PORTD = 0b10000000;
	_delay_us(1);
	PORTC = 0b00000000;
}

if(row == 0){
	for (int i = 0; i < size-1; i++){
		if(i==16){
			PORTC = 0b00000010;
			_delay_us(1);
			PORTD = 0b11000000;
			_delay_us(1);
			PORTC = 0b00000000;
		}
		PORTC = 0b00000011;
		_delay_us(1);
		PORTD = str[i];
		_delay_us(1);
		PORTC = 0b00000001;
		_delay_us(1);
	}
}else if(row == 1){
	for (int i = 0; i < size-1; i++){
		PORTC = 0b00000011;
		_delay_us(1);
		PORTD = str[i];
		_delay_us(1);
		PORTC = 0b00000001;
		_delay_us(1);
	}
}else{
	if(_cur_start == 1){
		PORTC = 0b00000010;
		_delay_us(1);
		PORTD = 0b11000000;
```

```
                _delay_us(1);
                PORTC = 0b00000000;
        }

        for (int i = 0; i < size-1; i++){
                PORTC = 0b00000011;
                _delay_us(1);
                PORTD = str[i];
                _delay_us(1);
                PORTC = 0b00000001;
                _delay_us(1);
        }
    }
}
```