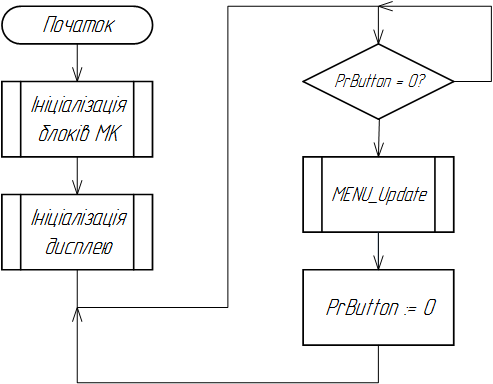
**РОЗДІЛ 3**

**Створення програмної частини**

***3.1. Загальна структура програми***

З метою організації роботи стенду написано програму в середовищі програмування Keil [10], що дає можливість керувати DC-двигуном безпосередньо зі стенду та за допомогою комп’ютера. Блок-схема даної програми наведена на рисунку 3.1.



*Рис. 3.1 – Блок-схема роботи програми*

Програма працює наступним чином: спочатку відбувається ініціалізація всіх необхідних блоків МК (порти вводу/виводу, таймери, послідовний інтерфейс, переривання тощо) та дисплею (встановлення налаштувань для роботи та відображення даних). Після цього очікуємо поки глобальна змінна PrButton не прийме значення відмінного від нуля. Це означатиме, що натиснута одна із чотирьох користувацьких кнопок або відбувся прийом даних через послідовний інтерфейс. Далі йде обробка відповідної події (процедура MENU\_Update), присвоєння змінній PrButton нульового значення та очікування нової події.

***3.2. Ініціалізація блоків МК***

Ініціалізація блоків МК відбувається наступним чином. Спочатку визначаємо, які блоки будуть тактуватись, а саме: таймери TIM2, TIM3, TIM6; порти вводу/виводу GPIOA, GPIOB, GPIOC; блок альтернативних функцій AFIO; пристрій послідовний інтерфейсу USART1.

#define RCC\_APB1ENR\_INIT (RCC\_APB1ENR\_TIM2EN | RCC\_APB1ENR\_TIM3EN | RCC\_APB1ENR\_TIM6EN)

#define RCC\_APB2ENR\_INIT (RCC\_APB2ENR\_IOPAEN | RCC\_APB2ENR\_IOPBEN | RCC\_APB2ENR\_IOPCEN | RCC\_APB2ENR\_AFIOEN | RCC\_APB2ENR\_USART1EN)

…

RCC->APB1ENR |= RCC\_APB1ENR\_INIT;

RCC->APB2ENR |= RCC\_APB2ENR\_INIT;

...

Після цього налаштовуємо порти вводу/виводу:

#define GPIOA\_CRH\_INIT 0x44442222

#define GPIOB\_CRL\_INIT 0x4a444444

#define GPIOB\_CRH\_INIT 0x4444a244

#define GPIOC\_CRL\_INIT 0x44424444

#define GPIOC\_CRH\_INIT 0x444222a2

#define AFIO\_MAPR\_INIT…(AFIO\_MAPR\_TIM2\_REMAP | AFIO\_MAPR\_TIM3\_REMAP | AFIO\_MAPR\_USART1\_REMAP)

…

GPIOA->CRH = GPIOA\_CRH\_INIT;

GPIOB->CRL = GPIOB\_CRL\_INIT;

GPIOB->CRH = GPIOB\_CRH\_INIT;

GPIOC->CRL = GPIOC\_CRL\_INIT;

GPIOC->CRH = GPIOC\_CRH\_INIT;

AFIO->MAPR = AFIO\_MAPR\_INIT;

Маємо наступні налаштування портів:

* PA8..11 – вихідні порти із максимальною частотою 2 МГц;
* PB6 – TxD USART1, PB7 – RxD USART1;
* PB10 – вихідний порт, PB11 – 4-ий канал таймера TIM2;
* PC4, PC8 – вихідний порт із максимальною частотою 2 МГц;
* PC9 – 4-ий канал таймера TIM3;
* PC10..12 – вихідні порти із максимальною частотою 2 МГц;

Налаштовуємо приймач/передавач USART1 (8 біт даних, 1 стоп біт, 9600 бод):

#define USART1\_BRR\_INIT (0x9c4)

#define USART1\_CR1\_INIT (USART\_CR1\_UE | USART\_CR1\_TE | USART\_CR1\_RE | USART\_CR1\_TCIE | USART\_CR1\_RXNEIE)

…

USART1->BRR = USART1\_BRR\_INIT;

USART1->CR1 = USART1\_CR1\_INIT;

Дозволяємо переривання від кнопок (на передній фронт) та приймача/передавача:

#define EXTI\_IMR\_INIT 0x0000000f

#define EXTI\_RTSR\_INIT 0x0000000f

#define NVIC\_ISER0\_INIT (NVIC\_ISER\_SETENA\_6 | NVIC\_ISER\_SETENA\_7 | NVIC\_ISER\_SETENA\_8 | NVIC\_ISER\_SETENA\_9)

#define NVIC\_ISER1\_INIT (NVIC\_ISER\_SETENA\_5)

…

EXTI->IMR = EXTI\_IMR\_INIT;

EXTI->RTSR = EXTI\_RTSR\_INIT;

NVIC->ISER[0] = NVIC\_ISER0\_INIT;

NVIC->ISER[1] = NVIC\_ISER1\_INIT;

Налаштовуємо таймери TIM2 та TIM3 на частоту генерування ШІМ сигналу 1 кГц, скважність 5% для 4-го каналу TIM2 та 95% для 4-го каналу TIM3 (обернена полярність):

TIM2->CR1 |= TIM\_CR1\_ARPE;

TIM2->CCMR2 |= (TIM\_CCMR2\_OC4PE | TIM\_CCMR2\_OC4M\_2 | TIM\_CCMR2\_OC4M\_1);

TIM2->ARR = 24000;

TIM2->CCR4 = 1200;

TIM2->CCER |= (TIM\_CCER\_CC4E);

TIM3->CR1 |= TIM\_CR1\_ARPE;

TIM3->CCMR2 |= (TIM\_CCMR2\_OC4PE | TIM\_CCMR2\_OC4M\_2 | TIM\_CCMR2\_OC4M\_1);

TIM3->ARR = 24000;

TIM3->CCR4 = 22800;

TIM3->CCER |= (TIM\_CCER\_CC4E | TIM\_CCER\_CC4P);

***3.3. Ініціалізація дисплею***

Налаштування дисплею починається із затримки в 50 мс для того, щоб впевнитись у включенні живлення дисплею. Після цього посилаємо команду 0x30 (робота із 4 бітним інтерфейсом) та очікуємо 50 мкс:

WH0802\_SendDC(0x30, 0);

WH0802\_DelayUS(50);

Встановлюємо параметри відображення даних (2 рядки, символ 5\*8 пікселів):

WH0802\_SendDC(0x28, 0);

WH0802\_DelayUS(50);

WH0802\_SendDC(0x28, 0);

WH0802\_DelayUS(50);

Відключаємо вивід курсору на дисплей та режим мигання:

WH0802\_SendDC(0x0c, 0);

WH0802\_DelayUS(50);

Очищаємо дисплей та проводимо останні налаштування дисплею:

WH0802\_SendDC(0x01, 0);

WH0802\_DelayMS(2);

WH0802\_SendDC(0x06, 0);

***3.4. Обробка подій***

При роботі стенду можливе виникнення 8-ми подій:

* натиснення кнопок SW1-SW4 (PrButton = 1..4);
* прийом команди запуску двигуна від ПК (PrButton = 5);
* прийом команди зупинки двигуна від ПК (PrButton = 6);
* прийом команди зміни напрямку від ПК (PrButton = 7);
* прийом команди встановлення швидкості від ПК (PrButton = 8).

Для обробки даних подій написана процедура MENU\_Update, після виконання якої відбувається оновлення даних на дисплеї та обнуління змінної PrButton. Розглянемо детальніше роботу процедури.

Якщо натиснута кнопка SW1 («вліво»), то виконуємо зміну параметру або його значення в залежності від попередніх дій:

if (Button == 1) {

if (!CONTROL.ItemValue) {

if (CONTROL.CurItem) CONTROL.CurItem = 0;

else CONTROL.CurItem = 1;

}

else {

if (CONTROL.CurItem == 0) {

if (CONTROL.Speed > 5) CONTROL.Speed -= 5;

else CONTROL.Speed = 5;

MENU\_UpdateSpeed(CONTROL.Speed);

}

else {

if (CONTROL.Direction) CONTROL.Direction = 0;

else CONTROL.Direction = 1;

MENU\_UpdateDirection(CONTROL.Direction);

}

}

}

Якщо натиснута кнопка SW2 («підтвердити»), то визначаємо, що ми далі будемо змінювати при натисненні кнопок SW1, SW3 – параметри чи їх значення:

if (Button == 2) {

if (!CONTROL.ItemValue) {

CONTROL.ItemValue = 1;

CONTROL.ITEM[CONTROL.CurItem][0] = ' ';

CONTROL.VALUE[CONTROL.CurItem][0] = '>';

}

else {

CONTROL.ItemValue = 0;

CONTROL.ITEM[CONTROL.CurItem][0] = '>';

CONTROL.VALUE[CONTROL.CurItem][0] = ' ';

}

}

Якщо натиснута кнопка SW3 («вправо»), то виконуємо зміну параметру або його значення в залежності від попередніх дій:

if (Button == 3) {

if (!CONTROL.ItemValue) {

if (!CONTROL.CurItem) CONTROL.CurItem = 1;

else CONTROL.CurItem = 0;

}

else {

if (CONTROL.CurItem == 0) {

if (CONTROL.Speed < 95) CONTROL.Speed += 5;

else CONTROL.Speed = 95;

MENU\_UpdateSpeed(CONTROL.Speed);

}

else {

if (CONTROL.TurnOn) MOTOR\_Stop();

if (CONTROL.Direction) CONTROL.Direction = 0;

else CONTROL.Direction = 1;

MENU\_UpdateDirection(CONTROL.Direction);

if (CONTROL.TurnOn) MOTOR\_Start();

}

}

}

Якщо натиснута кнопка SW4 («старт/стоп»), то виконуємо включення/виключення двигуна:

if (Button == 4) {

if (CONTROL.TurnOn) {

CONTROL.TurnOn = 0;

MOTOR\_Stop();

}

else {

CONTROL.TurnOn = 1;

MOTOR\_Start();

}

}

Якщо відбувся прийом команди запуску двигуна, то виконуємо початкові налаштування швидкості та напряму руху, відключаємо кнопки та запускаємо двигун:

if (Button == 5) {

EXTI->IMR &= ~EXTI\_IMR\_INIT;

if (!CONTROL.TurnOn) {

CONTROL.Direction = 0;

MENU\_UpdateDirection(CONTROL.Direction);

CONTROL.Speed = 5;

MENU\_UpdateSpeed(CONTROL.Speed);

CONTROL.TurnOn = 1;

MOTOR\_Start();

}

}

Якщо відбувся прийом команди зупинки двигуна, то включаємо можливість керування за допомогою кнопок та зупиняємо двигун:

if (Button == 6) {

EXTI->IMR |= EXTI\_IMR\_INIT;

if (CONTROL.TurnOn) {

CONTROL.TurnOn = 0;

MOTOR\_Stop();

}

}

Якщо відбувся прийом команди зміни напряму, то в разі необхідності зупиняємо двигун, змінюємо значення відповідної змінної та запускаємо знову двигун:

if (Button == 7) {

if (CONTROL.TurnOn) MOTOR\_Stop();

if (CONTROL.Direction) CONTROL.Direction = 0;

else CONTROL.Direction = 1;

MENU\_UpdateDirection(CONTROL.Direction);

if (CONTROL.TurnOn) MOTOR\_Start();}

Якщо відбувся прийом команди зміни швидкості двигуна, то виконуємо необхідні дії:

if (Button == 8) {

if ((CONTROL.Speed >= 5) && (CONTROL.Speed <= 95))

MENU\_UpdateSpeed(CONTROL.Speed);

}

**Висновки**

Для організації роботи стенду була написана програма (див. додатки Б, В, Г, Д), що дає змогу керувати двигуном безпосередньо зі стенду, а також за допомогою ПК. Алгоритм роботи програми полягає в ініціалізації необхідних блоків МК (порти вводу/виводу, таймери, приймач/передавач тощо), ініціалізації дисплею та в подальшому очікуванні подій: натиснення кнопок SW1..SW4 та прийом команд від ПК. При виконанні однієї із умов, викликається відповідний обробник, що встановлює змінну PrButton у необхідне значення, після чого керування передається процедурі MENU\_Update. Дана процедура та допоміжні написані таким чином, щоб забезпечити плавне керування двигуном, тобто запуск, зупинка та зміни напряму і швидкості відбувається не миттєво, а поступово. Це необхідно для того, щоб уникнути зростання струму, який протікає через ключові транзистори та двигун під час перемикання, що, в свою чергу, попереджає перегрівання та руйнування двигуна.