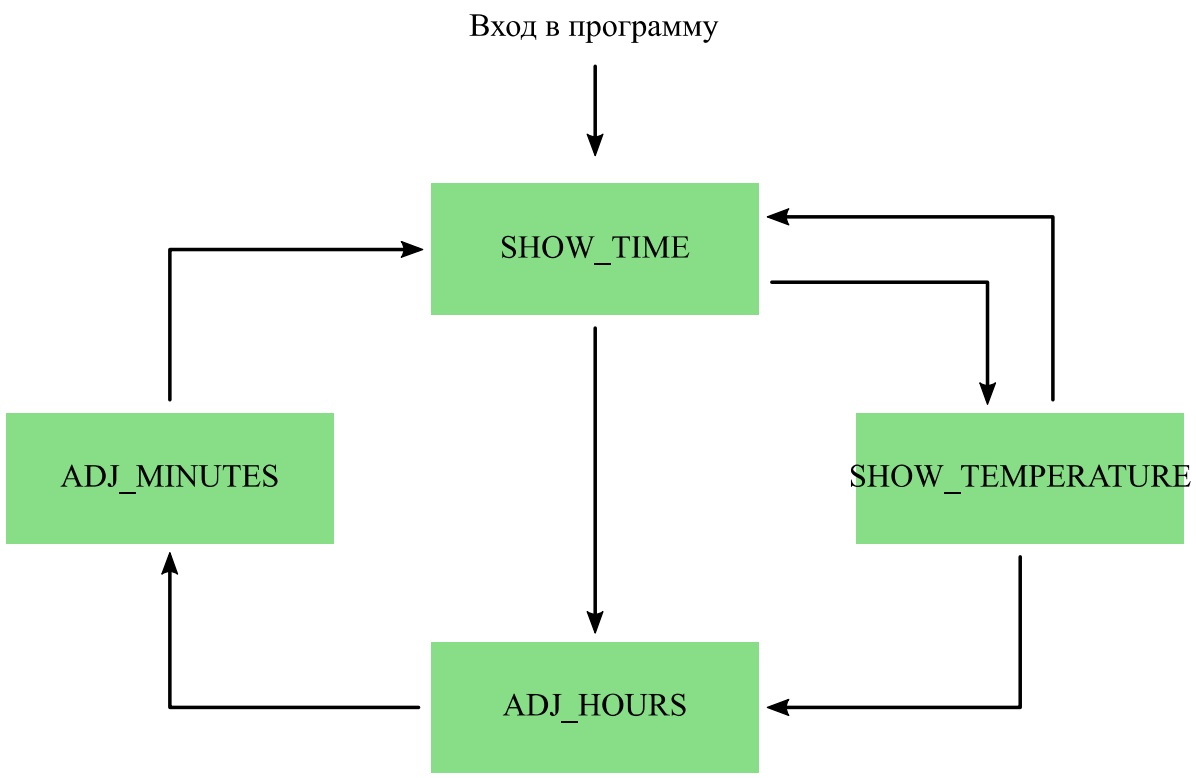
Машина состояний

Большая часть прошивки уже написана, осталось только организовать все модули и объединить их в одну целостную программу. Самый широкоиспользуемый метод — это написание машины состояний (англ. state machine).

Давайте поразмышляем. Устройство в любом случае должно отображать время (`ST_SHOW_TIME`). Так как имеется датчик температуры, то логично что нужно отображать его (`ST_SHOW_TEMPERATURE`). Плюс ко всему нужно иметь возможность настраивать время, а это ещё два состояния — настройка часов (`ST_SET_HOURS`) и минут (`ST_SET_MINUTES`). Пусть при этом переход между состояниями осуществляет нажатие кнопки, а при запуске устройство отображает время. Если нажать на кнопку, то состояние переключится на отображение температуры. Ещё одно нажатие возвращает к отображению времени. При длительном удержании кнопки (скажем больше 3 секунд), устройство переходит в режим настройки часов, после выставления нужного значения энкодером кратковременное нажатие переводит к состоянию настройки минут. Ещё одно нажатие возвращает в режим отображения времени. Граф состояний и переходов изображён ниже.



Создадим переменную для хранения текущего состояния `state` типа перечисления `enum` .

```
// state.h
typedef enum {
    ST_SHOW_TIME,
    ST_SHOW_TEMPERATURE,
    ST_SET_HOURS,
    ST_SET_MINUTES,
} STATE_t;

// state.c
volatile STATE_t state = ST_SHOW_TIME;
```

Для того чтобы обращаться к этой переменной, мы напишем две функции: так называемый геттер (англ. getter) и сеттер (англ. setter):

```
// state.h
STATES_t state_get(void);
void state_set(STATES_t new_state);

// state.c
STATES_t state_get(void) {
    return state;
}

void state_set(STATES_t new_state) {
    state = new_state;
}
```

Таким образом, подключив `state.h` к нужному модулю, мы получим доступ к состоянию через данные функции.

Реализация машины состояний может быть разной: простой и сложной; с отслеживанием переходов или без; с использованием таблицы переходов или без неё. В нашем случае не так много состояний и событий, не имеет смысла как-то усложнять программу.

> Обратитесь к книге «**Си для встраиваемых систем**», там описаны другие подходы.

```
int main(void) {
    mcu_init();

    while(1) {
        switch(state_get()) {
            case ST_SHOW_TIME:
                show_time();
                break;
            case ST_SHOW_TEMPERATURE:
                // update temperature data here
                show_temperature();
                break;
            case ST_SET_HOURS:
                show_set_hours();
                break;
            case ST_SET_MINUTES:
                show_set_minutes();
```

```
                break;
        }
        // get light sensor data and process it here
    }
}
```

Попробуйте самостоятельно реализовать функции отображения для каждого режима и задать поведение энкодера.

Осталось только задать поведение кнопок. Создадим конструкцию `switch` / `case` по аналогии с главным циклом в прерывании кнопки:

```
void EXTI9_5_IRQHandler(void) {
    switch(state_get()) {
        case ST_SHOW_TIME:
            // if short press
            state_set(ST_SHOW_TEMPERATURE);
            // if long press
            state_set(ST_SET_HOURS);
            break;
        case ST_SHOW_TEMPERATURE:
            // if short press
            state_set(ST_SHOW_TIME);
            // if long press
            state_set(ST_SET_HOURS);
            break;
        case ST_SET_HOURS:
            state_set(ST_SHOW_MINUTES);
            break;
        case ST_SET_MINUTES:
            state_set(ST_SHOW_TIME);
            break;
    }
}
```

Здесь мы опустили тонкости реализации обработки удержания кнопки, но мы рассматривали это несколькими параграфами ранее. Вернитесь к ним и попробуйте дописать прошивку.

Если у вас не получается по какой-то причине дописать минимальную прошивку, то его можно подсмотреть в репозитории на **github**.

На этом основная часть курса закончена. В следующем разделе представлены дополнительные темы, которые будут полезны вам, если вы захотите реализовать что-то на подобии **стоковой прошивки**. Её исходный код не будет опубликован, с той целью, что бы вы самостоятельно попробовали расширить функционал устройства.

Назад | **Оглавление** | **Дальше**