

Часы на светодиодных матрицах

Программные модули и функции

main.c

Функция main(), инициализация периферии и функции для основного режима(не меню).

void main(void)

загрузка настроек из EEPROM, машина состояний. Машина состояний для основного режима(часы, бег.строка, секунды) реализована через switch-case, меню реализовано через косвенный переход по массиву указателей на функции.

void temperature_read(void)

Читает температуру с датчика каждые TEMPERATURE_READ_INTERVAL секунд.

void alarm_stop(void)

Останавливает будильник по нажатию на любую кнопку.

uint8_t is_alarm_active(void)

Возвращает TRUE когда активирован будильник(играет мелодия).

void alarm_process(void)

Раз в минуту, когда приходит сигнал rtc_min_pulse, просматривает все включённые будильники и запускает один из них если совпадает время.

uint8_t text_start(void)

Подготавливает к выводу бегущую строку с датой, температурой и т.д. Возвращает TEXT_ENABLED если бегущая строка разрешена и будет воспроизводиться.

void sm_change(uint8_t sta)

Интерфейсная функция для безопасного изменения машины состояний. Аргумент: новое состояние. Если устройство находится в режиме сна(DEV_STA_SLEEP), то не изменяет состояние.

void hw_init(void)

Начальная инициализация периферии.

mem_copy.c

Функции копирования память-память.

void copymem_flash2ram(flash uint8_t *src, uint8_t *dest, uint8_t len)

Копировать flash в буфер ОЗУ.

uint8_t copymem_load_fstr(flash uint8_t *str, uint8_t *dest)

Загрузить строку из flash в ОЗУ.

Возвращает: число символов в строке.

uint8_t copymem_load_eestr(eeprom uint8_t *str, uint8_t *dest)

Прочитать строку из EEPROM.

Возвращает: число символов в строке.

void copymem_store_eestr(uint8_t *str, eeprom uint8_t *dest)

Сохранить строку в EEPROM.

void copymem_ram2ram(uint8_t *src, uint8_t *dest, uint8_t len)

Копировать буфер ОЗУ-ОЗУ.

void copymem_ramex(uint8_t *src, uint8_t *dest, uint8_t len)

Копировать буфер ОЗУ-ОЗУ. Использует исключительный доступ, чтобы недопустить искажения данных из подпрограмм прерываний.

scr_hal.c

Взаимодействие с экраном на низком уровне.

scr_buf[SCR_BUF_LEN] - буфер экрана для работы с высокоуровневыми функциями.

scr_buf_raw[SCR_BUF_LEN] - буфер регенерации экрана. Содержит данные в том виде, в каком они непосредственно отправляются на матрицы.

scr_row_shift_tabl[SCR_ROWS] - массив с номерами строк для конкретного подключения матриц, значения определяются пользователем в режиме настройки.

scr_col_shift_tabl[SCR_COLUMNS] - таблица номеров столбцов светодиодных матриц.

volatile uint8_t scr_busy - флаг, показывающий что сейчас идёт регенерация кадра, запрещает изменение буфера scr_buf_raw, чтобы не было мерцаний и прочих артефактов.

void scr_cls(void)

Очистить буфер экрана(=0). CLS - кто программировал на QBASIC, тот поймёт.

void scr_send_next_row(void)

Собственно регенерация экрана, отправляет новую строку на матрицы. Вызывается из обработчика прерывания.

void scr_led_off(void)

Выключает экран, используется для управления яркостью(программная ШИМ).

void scr_led_disable(void)

Отключает экран и таймер0 регенерации для перехода в спящий режим.

void scr_led_enable(void)

Включает экран по выходу из спящего режима.

void scr_bright_set(uint8_t br)

Устанавливает яркость экрана.

30=минимум

255=максимум.

Из-за того, что ШИМ программная, надо ставить мин>=30.

void scr_update(void)

Передаёт пользовательский буфер экрана scr_buf на регенерацию(в scr_buf_raw) с учётом подключения экрана по строкам и столбцам.

scr_api.c

Высокоуровневые функции взаимодействия с экраном. Работает с пользовательским буфером экрана scr_buf[SCR_BUF_LEN].

scr_str_buf[STRING_MAX_LEN] - буфер текстовой строки для надписей и бегущих строк.

scr_fnt - шрифт, которым печатаются буквы и цифры.

uint16_t scr_chr_get_adress(uint8_t chr, uint8_t fnt)

Распаковывает шрифт и возвращает адрес первого байта символа.

Аргументы: номер символа по ASCII/CP1251; номер шрифта.

void scr_shift(uint8_t x_start, uint8_t x_stop, uint8_t direction, uint8_t step)

Сдвинуть экран(или его часть) в указанном направлении на указанное число пикселей.

Аргументы: x_start, x_stop - выделяем область экрана для сдвига; direction - направление сдвига; step - на какое число пикселей сдвинуть экран.

uint8_t scr_chr_draw(uint8_t chr, uint8_t x_start)

Рисует изображение символа в буфер экрана.

Аргументы: номер символа, начальная X0-координата символа на экране.

CAPTION_STRUCT *scr_caption(void)

Рисует статичную(не двигающуюся) надпись.

Возвращаемый указатель на структуру может использоваться в бегущей строке для отрисовки оставшейся части надписи.

uint8_t scr_scroll_text(uint8_t control)

Бегущая строка.

Аргумент: управляющее сообщение.

Возвращает: SCRL_TXT_FINISH если строка выведена до конца.

void scr_show_time(RTC_STRUCT *time, uint8_t control)

Показывает время.

Аргумент: указатель на структуру времени; управляющее сообщение-задаёт режим показа.

void scr_str_clear(void)

Очищает буфер текстовой строки(=0).

void scr_set_font(uint8_t fnt)

Устанавливает новый шрифт для отображения текстов и цифр.

Аргумент: номер шрифта.

uint8_t scr_val2str(uint8_t val, uint8_t lead_zero, uint8_t *buf)

Переводит байтовое число U8 в текстовый вид(строку), сильно упрощённый аналог printf.

Аргументы: значение/число; кол-во ведущих нулей; указатель на буфер строки.

Возвращает: кол-во символов в числе.

font.c

Шрифт.

fnt_adress_tabl[] - таблица начальных адресов для каждого шрифта.

MyFont[] - сам шрифт. Данные запакованы в следующем виде:

[кол-во байт/столбцов символа][байты этого символа].

Для получения первого байта символа надо пройти следующую процедуру (см.

scr_chr_get_adress):

Получить адрес первого символа из fnt_adress_tabl, затем пройти в MyFont до нужного символа.

buttons.c

Работа с кнопками.

btn1_click

btn2_click - была нажата первая или вторая кнопка (TRUE).

void buttons_update(void)

Вызывается из прерывания TIM0 для получения статуса кнопок.

rtc.c

Часы реального времени.

volatile RTC_STRUCT rtc - структура с текущим временем

volatile RTC_CORR rtc_correction - коррекция времени.

uint8_t is_leap_year(uint8_t year)

Возвращает TRUE если год високосный.

uint8_t day_of_week(RTC_STRUCT * time)

День недели.

inline void rtc_update(void)

Вызывается из TIM2 для обновления/счёта текущего времени.

void rtc_read(RTC_STRUCT *dest)

Копирует значение часов в буфер. Используется атомарный доступ чтобы исключить порчу данных.

irq.c

Обработчики прерываний.

beeper.c

Реализация бипера/секвенсора.

flash uint8_t note2freq[] - таблица значений OCR1, соответствующая нотам разных октав.

flash uint16_t note_delay_tabl[] - длительность нот.

volatile uint8_t seq_work - флаг, показывает работает ли секвенсор(TRUE).

void beeper_on(void)

Включить звук "БИП" при нажатии на кнопку.

void beeper_delay(void)

Задержка/длительность звука "БИП". Вызывается из прерывания TIM0.

void beeper_off(void)

Выключить бипер.

void sequencer_process(void)

Собственно секвенсор, вызывается из прерывания TIM0 для отработки длительности нот и пауз.

void sequencer_start(uint8_t melody_id)

Пуск секвенсора.

Аргумент: номер мелодии.

void sequencer_stop(void)

Остановить секвенсор.

melody.c

Мелодии.

flash NOTE_STRUCT *melody_dat_ptr[] - массив указателей на мелодии.

flash uint8_t melody_len[] - массив длин мелодий в нотах.

flash FLASH_STR_PTR melody_name[] - массив указателей на строки/названия мелодий.

Мелодия состоит из нот, ноты определяются структурой:

typedef struct{

uint8_t note; // нота + октава

uint8_t len; // длительность ноты в тактах(музыкальных).

} NOTE_STRUCT;

Ноты записываются следующим образом:

В музыкальном редакторе	В исходном коде Си
C	NC
C#	NC_
D	ND
D#	ND_
E	NE
F	NF
F#	NF_
G	NG
G#	NG_
A	NA
A#	NA_
B	NB

Нота NPAUSE = пауза.

Запись ноты в мелодии

```
flash NOTE_STRUCT melody_v_lesu[]={  
{OCT4+NC, L_1_2},
```

Означает следующее:

Нота C четвёртой октавы, длительностью 1/2.

Ноты имеют фиксированную длительность по секундам. Длина целой(1/1) = 0.75 сек.

Октавы воспроизводятся со 2-й по 6-ю. Но 6-я из-за ошибок деления частоты сильно фальшивит, поэтому нормально работают со 2-й по 5-ю.

OCT2 - вторая октава по научной нотации

OCT6 - шестая.

1wire.c

Низкоуровневые функции для работы с шиной 1Wire.

unsigned char ow_reset(void)

Сброс.

Возвращает импульс PRESENCE.

unsigned char ow_read(void)

Читать байт.

void ow_write(unsigned char data)

Записать байт на шину.

unsigned char ow_crc(unsigned char *source, unsigned char len)

Посчитать CRC8.

ds1820.c

Работа с термометром.

void ds1820_start_convert(void)

Запуск преобразования.

signed char ds1820_get_temperature(void)

Прочитать температуру.

Возвращает целочисленное знаковое значение. -99 если была ошибка на шине.

menu.c

Функции меню.

msg01[EEPROM_STRING_MAX_LEN]="С новым годом! Янги йилингиз муборак!";

...

msg10[EEPROM_STRING_MAX_LEN]="Сообщение десятое"; - тексты сообщений.

flash MNU_FUNC_PTR mnu_func_ptr[] - массив указателей на функции меню для косвенного перехода в автомате состояний.

uint8_t load_scr_tables(void)

Прочитать таблицу настроек(строки/столбцы) из EEPROM.

Возвращает: SCR_TABL_ERR если данные содержат ошибки.

void scr_rows_reset(void)

void scr_columns_reset(void)

Сбросить таблицу строк и столбцов экрана в состояние по умолчанию.

void load_alarm_data(void)

Прочитать время будильников из EEPROM.

void load_text_date(void)

Прочитать даты показа текстов из EEPROM.

void menu_reset(void)

Сбросить механизм меню при переходе между подпунктами.

void btn1_reset(void)

void btn2_reset(void)

Сбросить состояние кнопок в режиме меню.

void menu_main(void)

Главное меню.

void menu_time(void)

Меню установки времени.

void menu_date(void)

Меню установки даты.

void set_alarm(void)

Установить время и мелодию будильников.

void menu_txt_select(void)

Выбор/просмотр текстов.

void editor_write(void)

Редактор текста.

void text_timetable(void)

Расписание показа текстов.

void calibr_rows(void)

void calibr_columns(void)

Меню настройки строк и столбцов.

void menu_system(void)

Системное меню.

void menu_correction(void)

Установка коррекции времени.

void menu_bright(void)

Установка яркости.

void menu_timefont(void)

Шрифт времени.

void menu_text_interval(void)

Интервал бег. строки.

Константы.

ALARM_LEN длина будильника в минутах.

TEMPERATURE_READ_INTERVAL интервал чтения температуры.

STRING_MAX_LEN максимальный размер буфера бегущей строки

EEPROM_STRING_MAX_LEN максимальная длина текста-напоминалки в eeprom.

MELODY_TOTAL_CNT количество мелодий.

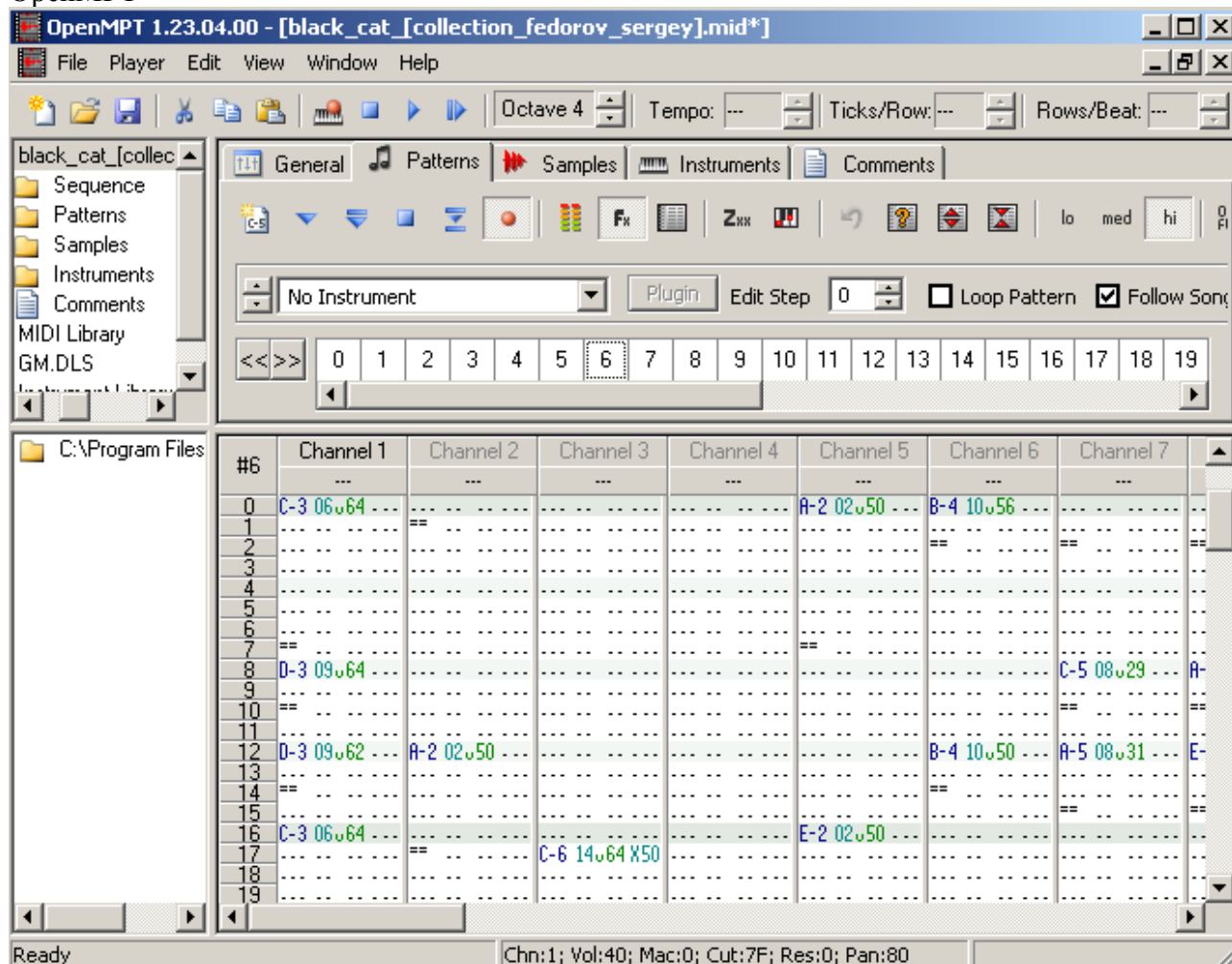
FNT_TOTAL_CNT количество шрифтов.

Добавление мелодии.

Не всегда мелодия есть в приемлемом виде. Тогда ищем в интернете её в формате MIDI.

MIDI можно распотрошить многими инструментами для музыкантов. Среди бесплатных:

OpenMPT



В Patterns выбираем канал и срисовываем его ноты. Недостаток: в сложной музыке один инструмент может занимать несколько каналов.

LMMS

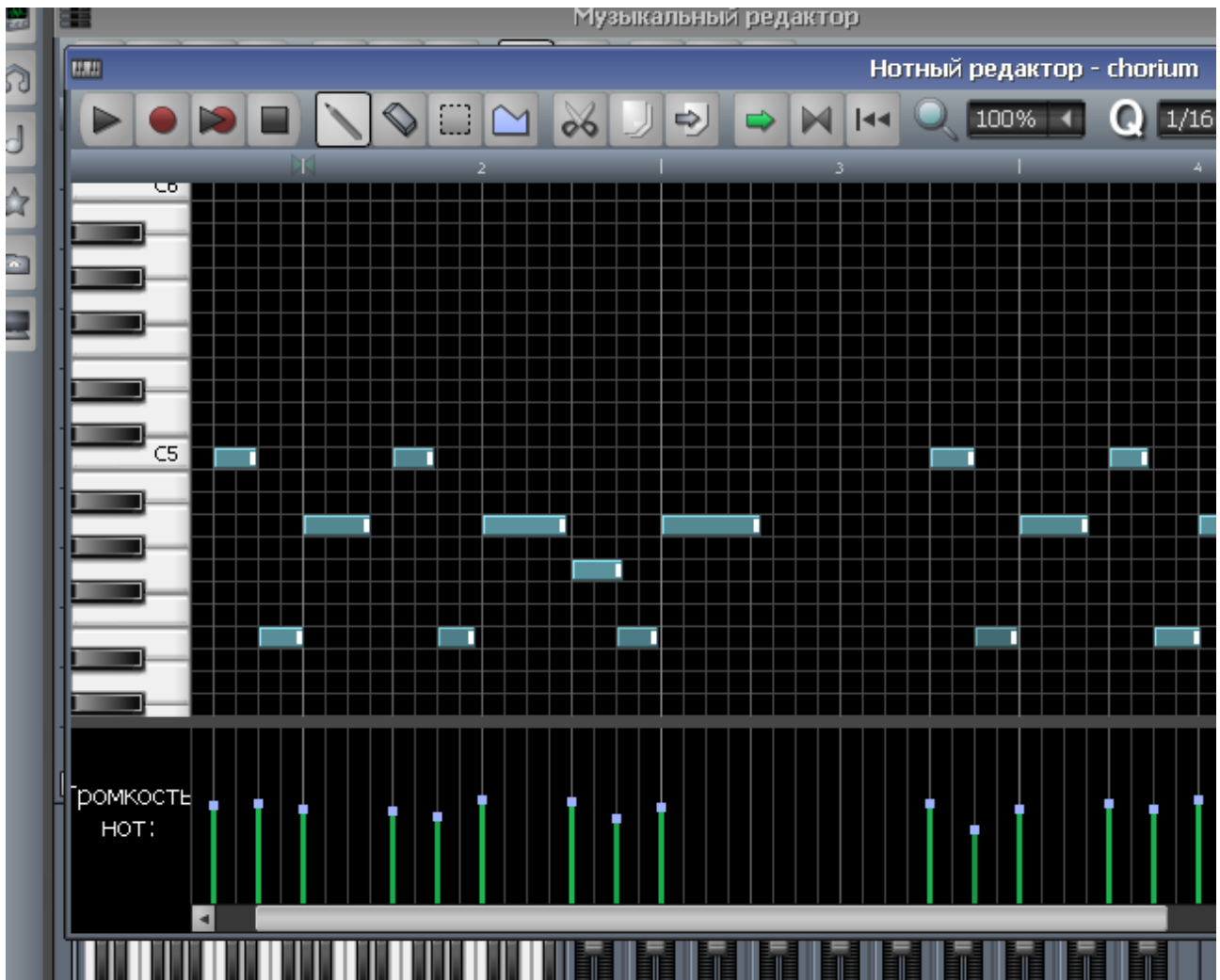
Для каждого инструмента показывает свою дорожку.

Полноценный инструмент для министудии, но и более сложный в обращении.

Пример для картинки с OpenMPT:

В первом канале нота С третьей октавы занимает 7 условных(внутренних) тактов. В музыке понятие длины ноты относительно и не привязано жёстко ко времени в секундах.

Поэтому когда мы переписали все ноты и их длительность, надо определиться какая из них будет целая, где 1/2, 1/3 и т.д. После этого можно приступить к переводу мелодии в массив на языке Си.



Затем открываем файл melody.c. Для расшифровки типов данных см. описание файла выше.

Создаём строку с названием и массив с нотами:

```
flash uint8_t new_melody_name[]="Новая мелодия..."; // имя мелодии
flash NOTE_STRUCT new_melody[]={ // массив нот
{OCT4+NC, L_1_2}, // октава+нота, длительность
{OCT4+NA, L_1_2},
{NPAUSE, L_1_48},
```

Добавим в массивы указатель на мелодию и её длину:

```
flash NOTE_STRUCT *melody_dat_ptr[]={ // указатель на мелодию
new_melody };
flash uint8_t melody_len[]={ // длина мелодии в нотах
sizeof(new_melody)/2 };
flash FLASH_STR_PTR melody_name[]={ // указатель на строку с названием
new_melody_name };
```

В файле melody.h пропишем новое число мелодий:

```
#define MELODY_TOTAL_CNT 10+1
```