

Пример работы со светодиодным модулем P10(1R)-V701B-3

Пример основан на исходном коде бегущей строки STX-4P10 и может быть загружен в неё без каких-либо доработок. С некоторыми изменениями можно запустить и на других контроллерах, например ATmega16, ATmega32.

При выводе изображений используется двойная буферизация. Такой подход позволил изолировать медленный процесс построения картинки от довольно быстрого процесса регенерации табло.

Задействованная периферия контроллера ATmega64:

SPI, режим мастера, MSB, Mode 3. Работает только на передачу. Режим 3 т.к. в режиме Mode 0 на длинных табло, после 3-4 модулей, почему-то начинается подёргивание картинки.

TIMER0. Формирует задержки для регенерации строк изображения.

Светодиодный модуль P10 имеет 16 строк и одновременно светятся 4 из них. Поэтому период регенерации кадра=(OCR0+1)/4. Таймер работает в режиме CTC.

Прерывания:

TIM0_COMP.

Прерывание по сравнению инициирует начало передачи следующей горизонтальной линии изображения в регистры 74HC595 через SPI.

Картинка считывается из буфера регенерации video_raw_out[].

Флаг scr_busy устанавливается и говорит, что сейчас идёт регенерация кадра.

Нельзя изменять буфер регенерации до окончания текущего кадра.

SPI_STC.

Окончание передачи байта через SPI. Загружает следующий байт из буфера регенерации video_raw_out[] в SPI.

Когда передача линии завершена, в порт выставляется код следующей линии scr_line_cnt и на регистры подаётся сигнал стробирования SCR_STROB = ON;.

Когда передача кадра завершена, флаг scr_busy очищается и говорит, что теперь можно изменить буфер регенерации. Таким образом изменение буферов синхронизировано с механизмом регенерации.

Счётчик scroll_frm_cnt считает кадры и служит для создания разных задержек.

Функции:

```
void scr_message(flash char *str)
```

Печатает сообщение/строку из флеш на экран с паузой.

```
void cls()
```

Очистить буфер экрана.

```
char scr_putchar(char chr, unsigned char fnt, unsigned char param)
```

Напечатать символ в буфер экрана.

Аргументы:

chr - код символа.

fnt - номер шрифта. В системе определены константы SCR_FNT1, SCR_FNT2, SCR_FNT3 и т.д.

param - дополнительный параметр:

ALIGN_RIGHT - выравнивание по правому краю.

ALIGN_LEFT - выравнивание слева.

CHR_UNDERLINE - подчеркнутый символ.

Печатаемый символ автоматически изменяет позицию курсора.

Возвращает:

0 - если напечатанная буква находится в пределах экрана.

>1 - число напечатанных пикселей буквы по горизонтали, когда она выходит за пределы экрана.

`void screen_prepare(void)`

Обновляет буфер регенерации из буфера экрана, передаёт картинку строки на регенерацию.

Синхронизировано с механизмом регенерации кадра.

Шрифты.

В системе используются растровые пропорциональные шрифты. Для экономии места структура каждого символа имеет переменную длину. Шрифты подготовлены в самописной программе Binary Font Editor V2 путём захвата экранных шрифтов и последующей их ручной доводкой для лучшего восприятия на светодиодном табло.

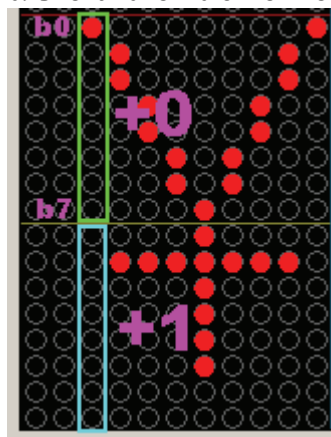
FontLUT[] - таблица смещений/адресов символов. Содержит по 256 индексов на каждый шрифт. Общее число индексов=число шрифтов*256.

MyFont[] - таблица данных символов шрифтов. Каждый символ имеет следующую структуру:

+0 - ширина символа в столбцах.

+1...+N - картинка символа. На каждый столбец приходится 2 байта, поэтому $\text{размер_данных_символа} = \text{его_ширина} * 2$.

Байты расположены вертикально. Упаковка битов показана на картинке.



Чтобы считать символ надо сначала получить его смещение:

$\text{offset} = \text{FontLUT}[\text{номер_символа} * \text{номер_шрифта}]$

Только потом обратиться к таблице данных:

$\text{ширина} = \text{MyFont}[\text{offset}]$

$\text{offset} = \text{offset} + 1$

$\text{for}(i=0; i < \text{ширина} * 2; i++)$

{

$\text{данные} = \text{MyFont}[\text{offset}]$

$\text{offset} = \text{offset} + 1$

}