Программирование ЖК-дисплея

Южанин Виктор Владимирович

Каф. Автоматизации технологических процессов

Характеристики дисплея

Контроллер дисплея	ST7735
Аппаратный интерфейс управления	Четырехпроводный интерфейс Serial-Peripheral Interface (SPI)
Разрешение	128x160 либо 160x120 пикселей
Размер текстового символа	5x7 пикселей
Портретный режим	разрешение 160x128 20 текстовых строк 21 символов на строку
Альбомный режим	разрешение 128x160 16 текстовых строк 26 символов на строку

Функции инициализации и очистки

```
void InitTFT();
// Инциализация дисплея

void ClearScreen()
// Очистка экрана

void SetOrientation(int degrees);
// Задает поворот дисплея на 0, 90, 180, 270 градусов
```

Функции отрисовки линий

```
void HLine(byte x0, byte x1, byte y, int color);
// горизонтальная линия заданного цвета

void VLine(byte x, byte y0, byte y1, int color);
// вертикальная линия заданного цвета

void Line(int x0, int y0, int x1, int y1, int color);
// линия произвольного положения на основе алгоритма Брезенхэма
// https://ru.wikipedia.org/wiki/Алгоритм_Брезенхэма
```

Функции отрисовки прямоугольников

```
void DrawRect(byte x0, byte y0, byte x1, byte y1, int color);
// прямоугольник заданного цвета

void FillRect(byte x0, byte y0, byte x1, byte y1, int color);
// закрашенный прямоугольник заданного цвета

void RoundRect(byte x0, byte y0, byte x1, byte y1, byte r, int color);
// прямоугольник со скругленными углами (r - радиус скругления)
// координаты: левый верхний угол = x0,y0; нижний правый угол = x1,y1
```

Функции отрисовки окружностей и эллипсов

```
void Circle(byte xPos, byte yPos, byte radius, int color);
// окружность с центром в заданной точке, с заданным радиусом и цветом

void FillCircle(byte xPos, byte yPos, byte radius, int color);
// круг с центром в заданной точке, с заданным радиусом и цветом

void CircleQuadrant(byte xPos, byte yPos, byte radius, byte quad, int color);
// отрисовывает окружность в заданном квадранте с заданным радиусом и цветом
// квадрант кодируется битами в quad, причем квадранты нумеруются сверху вниз
// бит 0: квадрант I (нижний правый)
// бит 1: квадрант IV (верхний правый)
// бит 2: квадрант II (нижний левый)
// бит 3: квадрант III (верхний левый)

void Ellipse(int x0, int y0, int width, int height, int color);
// эллипс заданной ширины и высоты
// Алгоритм Брезенхэма для окружностей
// https://ru.wikipedia.org/wiki/Алгоритм_Брезенхэма
// Особенность: небольой разрыв между частями узких эллипсов

void FillEllipse(int xPos, int yPos, int width, int height, int color);
// закрашенный эллипс заданной ширины и высоты
```

Функции вывода текста

```
void GotoXY(byte x, byte y);
// задает позицию курсора (не отображается), где 0<x<20, 0<y<19.

void GotoLine(byte y);
// задает позицию курсора в начало заданной строки, где 0<y<19.

void WriteChar(char ch, int color);
// выводит символ заданного цвета в текущей позиции курсора

void WriteInt(int i);
// выводит число заданного цвета в текущей позиции курсора

void WriteHex(int i);
// выводит число заданного цвета в шестнадцатеричном коде в
// текущей позиции курсора

void WriteString(char *text, int color);
// выводит строку заданного цвета в текущей позиции курсора
```

Кодировка ASCII

	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
00	NUL 0000	STX 0001	SOT 0002	ETX 0003	EOT 0004	ENQ 0005	ACK 0006	BEL 0007	BS 0008	HT 0009	LF 000A	VT 000B	FF 000C	CR 000D	SO 000E	SI 000F
10	DLE 0010	DC1 0011	DC2 0012	DC3 0013	DC4 0014	NAK 0015	SYN 0016	ETB 0017	CAN 0018	EM 0019	SUB 001A	ESC 001B	FS 001C	GS 001D	RS 001E	US 001F
20	SP 0020	! 0021	" 0022	# 0023	$ 0024	% 0025	& 0026	' 0027	(0028	) 0029	* 002A	+ 002B	, 002C	- 002D	. 002E	/ 002F
30	0 0030	1 0031	2 0032	3 0033	4 0034	5 0035	6 0036	7 0037	8 0038	9 0039	: 003A	; 003B	< 003C	= 003D	> 003E	? 003F
40	@ 0040	A 0041	B 0042	C 0043	D 0044	E 0045	F 0046	G 0047	H 0048	I 0049	J 004A	K 004B	L 004C	M 004D	N 004E	O 004F
50	P 0050	Q 0051	R 0052	S 0053	T 0054	U 0055	V 0056	W 0057	X 0058	Y 0059	Z 005A	[005B	\ 005C	] 005D	^ 005E	_ 005F
60	` 0060	a 0061	b 0062	c 0063	d 0064	e 0065	f 0066	g 0067	h 0068	i 0069	j 006A	k 006B	l 006C	m 006D	n 006E	o 006F
70	p 0070	q 0071	r 0072	s 0073	t 0074	u 0075	v 0076	w 0077	x 0078	y 0079	z 007A	{ 007B	| 007C	} 007D	~ 007E	DEL 007F

Кодировка Windows-1251

	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
00	NUL 0000	STX 0001	SOT 0002	ETX 0003	EOT 0004	ENQ 0005	ACK 0006	BEL 0007	BS 0008	HT 0009	LF 000A	VT 000B	FF 000C	CR 000D	SO 000E	SI 000F
10	DLE 0010	DC1 0011	DC2 0012	DC3 0013	DC4 0014	NAK 0015	SYN 0016	ETB 0017	CAN 0018	EM 0019	SUB 001A	ESC 001B	FS 001C	GS 001D	RS 001E	US 001F
20	SP 0020	! 0021	" 0022	# 0023	$ 0024	% 0025	& 0026	' 0027	(0028	) 0029	* 002A	+ 002B	, 002C	- 002D	. 002E	/ 002F
30	0 0030	1 0031	2 0032	3 0033	4 0034	5 0035	6 0036	7 0037	8 0038	9 0039	: 003A	; 003B	< 003C	= 003D	> 003E	? 003F
40	@ 0040	A 0041	B 0042	C 0043	D 0044	E 0045	F 0046	G 0047	H 0048	I 0049	J 004A	K 004B	L 004C	M 004D	N 004E	O 004F
50	P 0050	Q 0051	R 0052	S 0053	T 0054	U 0055	V 0056	W 0057	X 0058	Y 0059	Z 005A	[005B	\ 005C	] 005D	^ 005E	_ 005F
60	` 0060	a 0061	b 0062	c 0063	d 0064	e 0065	f 0066	g 0067	h 0068	i 0069	j 006A	k 006B	l 006C	m 006D	n 006E	o 006F
70	p 0070	q 0071	r 0072	s 0073	t 0074	u 0075	v 0076	w 0077	x 0078	y 0079	z 007A	{ 007B	| 007C	} 007D	~ 007E	DEL 007F
80	Ђ 0402	Ѓ 0403	‚ 201A	ѓ 0453	„ 201E	… 2026	† 2020	‡ 2021	€ 20AC	‰ 2030	Љ 0409	‹ 2039	Њ 040A	Ќ 040C	Ћ 040B	Џ 040F
90	ђ 0452	‘ 2018	’ 2019	“ 201C	” 201D	• 2022	– 2013	— 2014		™ 2122	љ 0459	› 203A	њ 045A	ќ 045C	ћ 045B	џ 045F
A0	NBSP 00A0	Ў 040E	ў 045E	Ј 0408	¤ 00A4	Ґ 0490	¦ 00A6	§ 00A7	Ё 0401	© 00A9	Є 0404	« 00AB	¬ 00AC	- 00AD	® 00AE	Ї 0407
B0	° 00B0	± 00B1	І 0406	і 0456	ґ 0491	µ 00B5	¶ 00B6	· 00B7	ё 0451	№ 2116	є 0454	» 00BB	ј 0458	Ѕ 0405	ѕ 0455	ї 0457
C0	А 0410	Б 0411	В 0412	Г 0413	Д 0414	Е 0415	Ж 0416	З 0417	И 0418	Й 0419	К 041A	Л 041B	М 041C	Н 041D	О 041E	П 041F
D0	Р 0420	С 0421	Т 0422	У 0423	Ф 0424	Х 0425	Ц 0426	Ч 0427	Ш 0428	Щ 0429	Ъ 042A	Ы 042B	Ь 042C	Э 042D	Ю 042E	Я 042F
E0	а 0430	б 0431	в 0432	г 0433	д 0434	е 0435	ж 0436	з 0437	и 0438	й 0439	к 043A	л 043B	м 043C	н 043D	о 043E	п 043F
F0	р 0440	с 0441	т 0442	у 0443	ф 0444	х 0445	ц 0446	ч 0447	ш 0448	щ 0449	ъ 044A	ы 044B	ь 044C	э 044D	ю 044E	я 044F

Строки – массив символов, заканчивающийся нулем

```
int main() {
    // массив символов, заканчивающийся нулем
    char string1[] = {0x41, 0x54, 0x2D, 0x30, 0x38, 0x2D, 0x32, 0x00};
}
```

Что за строка записана в string1?

	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
00	NUL 0000	STX 0001	SOT 0002	ETX 0003	EOT 0004	ENQ 0005	ACK 0006	BEL 0007	BS 0008	HT 0009	LF 000A	VT 000B	FF 000C	CR 000D	SO 000E	SI 000F
10	DLE 0010	DC1 0011	DC2 0012	DC3 0013	DC4 0014	NAK 0015	SYN 0016	ETB 0017	CAN 0018	EM 0019	SUB 001A	ESC 001B	FS 001C	GS 001D	RS 001E	US 001F
20	SP 0020	! 0021	" 0022	# 0023	$ 0024	% 0025	& 0026	' 0027	(0028	) 0029	* 002A	+ 002B	, 002C	- 002D	. 002E	/ 002F
30	0 0030	1 0031	2 0032	3 0033	4 0034	5 0035	6 0036	7 0037	8 0038	9 0039	: 003A	; 003B	< 003C	= 003D	> 003E	? 003F
40	@ 0040	A 0041	B 0042	C 0043	D 0044	E 0045	F 0046	G 0047	H 0048	I 0049	J 004A	K 004B	L 004C	M 004D	N 004E	O 004F
50	P 0050	Q 0051	R 0052	S 0053	T 0054	U 0055	V 0056	W 0057	X 0058	Y 0059	Z 005A	[005B	\ 005C	] 005D	^ 005E	_ 005F
60	` 0060	a 0061	b 0062	c 0063	d 0064	e 0065	f 0066	g 0067	h 0068	i 0069	j 006A	k 006B	l 006C	m 006D	n 006E	o 006F
70	p 0070	q 0071	r 0072	s 0073	t 0074	u 0075	v 0076	w 0077	x 0078	y 0079	z 007A	{ 007B	| 007C	} 007D	~ 007E	DEL 007F

Более удобный способ задания строк

Вместо кода символа можно писать сам символ в одинарных кавычках

```
int main() {
      // массив символов, заканчивающийся нулем
      char string2[] = {‘A’, ‘T’, ‘-’, ‘0’, ‘8’, ‘-’, ‘2’, 0x00};
}
```

Нормальный способ задания строк

Последовательность символов-строка записывается в двойных кавычках

```
int main() {
        char string3[] = “AT-08-2”;
}
```

- Задан ли здесь ноль на конце?

- И все-таки, зачем он нужен?

Зачем нужен ноль в конце строки

Как передать строку в функцию, которая, например, считает длину строки?

```
int main() {
        char string3[] = "AT-08-2";
        char* first_symbol = &string3[0]; // указатель на первый символ строки
        int leng = string_length(first_symbol);
}
```

- Как понять длину строки по указателю на первый символ?
- А что, если продвигаться вперед по указателю, пока не поймем, что он указывает на 0?

```
int string_length(char* string) {
        int length = 0;
        for (char* current_char = string; *current_char != 0; ++current_char)
        {
                length++;
        }
        return length;
}
```

Стандартная библиотека C

- Для строки полезны модули: <string.h> <stdio.h>
- Определение длины строки: функция strlen()
- Форматированный вывод строк
- Много других полезных функций
- Описание http://www.nongnu.org/avr-libc/user-manual/modules.html

Форматированный вывод (функция printf)

```
#include <avr/io.h>
#include <stdio.h>
// собирать проект с опциями:
// -Wl,-u,vfprintf -lprintf_flt -lm
int main() {
	char str[20];
	sprintf(str, "%u", -1); // беззнаковое число
	sprintf(str, "%7d", 15); // знаковое целое на 7 знакомест
	sprintf(str, "%10.2f", -24.3); // вещественное на 10 целых и 2 дробных знакоместа
	sprintf(str, "%x", 255); // 16-разрядное представление целого числа
}
```

Что теперь делать со строками?

- Можно вывести на дисплей
- Это делается с помощью специальной библиотеки, будут рассказано на лабораторных занятиях