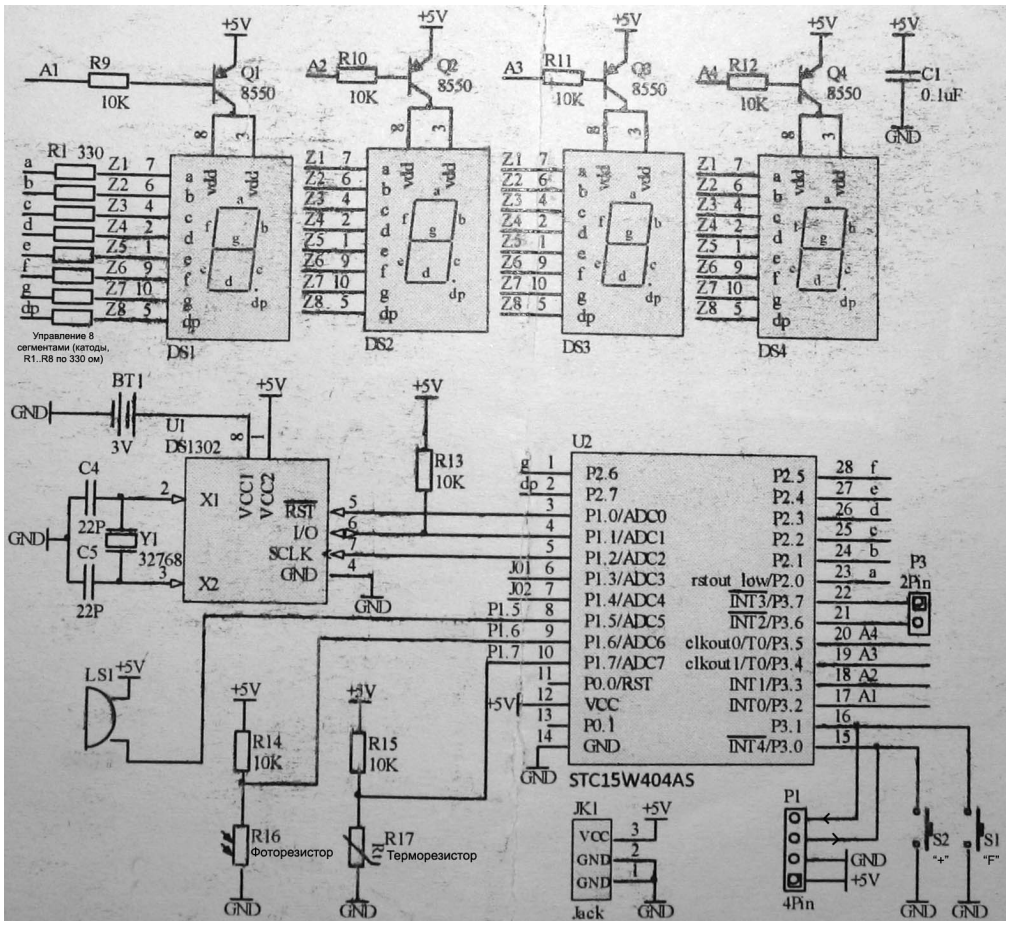
Цифровые часы на микроконтроллере STC15W404AS

Данные цифровые часы легко собираются в домашних условиях и при этом красиво выглядят. Также приятный бонус - схема открыта, и в интернете можно легко найти исходный код микроконтроллера часов, так что при желании можно разобраться как все работает и внести свои изменения в алгоритм. Я купил себе такие часы, собрал за 1 вечер, и здесь решил выложить перевод инструкции к часам, которая в оригинале поставляется на английском языке.



Часы работают на дешевом микроконтроллере **STC15W404AS**, который можно купить на сайте aliexpress по цене 0.6..1.5 доллара.

Микроконтроллер STC15W404AS

Здесь приведена краткая информация по микроконтроллеру STC15W404AS. Подробные данные см. в даташите STC15-English.pdf из архива [3].

STC15W404AS относится к серии STC15W401AS семейства STC15 микроконтроллеров компании STC MCU Limited. Микроконтроллер имеет усовершенствованное ядро MSC51 (система команд и архитектура популярного некогда семейства микроконтроллеров Intel 8051). Это новое ядро, отличающееся высоким быстродействием (скорость работы в 8..12 раз быстрее традиционного ядра 8051 на той же тактовой частоте), широким рабочим диапазоном напряжений питания, низким энергопотреблением и устойчивостью к помехам. Код программы может быть защищен от несанкционированного доступа при перепрошивке с помощью специальной технологии шифрования STC. Поставляется в различных вариантах корпусов на 16, 20 и 28 выводов. STC15W404AS полностью совместим по системе команд с традиционным ядром 8051, и реализует его все аппаратные функции. Дополнительно в нем имеется два указателя DPTR вместо одного, порт UART можно использовать как 3 последовательных порта путем сдвига его данных на 3 группы выводов. Также имеется интерфейс SPI, и 8-канальный АЦП, которых нет в традиционных микроконтроллерах Intel 8051/8052/8751. Порты GPIO могут работать точно так же, как и оригинальные, и их также можно использовать в расширенных режимах (есть 4 режима работы: квази-двунаправленный со слабым pull-up, мощный двухтактный с усиленным pull-up, только вход с высоким сопротивлением и открытый сток). Каждый выход может коммутировать ток до 20 мА, однако общий коммутируемый ток не должен превышать 120 мА на корпус для 40-выводного корпуса и 90 мА для 16-выводного корпуса. В таблице ниже сведены основные параметры микроконтроллера.

Параметр	**Описание**
Рабочее напряжение	2.4V..5V
Память программ (FLASH)	4 килобайта, с поддержкой ISP/IAP (IAP расшифровывается как In-Application Programming, т. е. перепрограммирование в программе), количество перезаписей не менее 100 тысяч раз.

ОЗУ (SRAM)	512 байт: 256 байт традиционное регистровое ОЗУ (scratch-pad RAM) и 256 байт расширенное ОЗУ (auxiliary RAM).
UART	1 шт., его можно по выбору использовать на 3 группах выводов (P3.0/P3.1, или P3.6/P3.7 или P1.6/P1.7).
SPI	1 шт.
Таймеры	3 таймера разрядностью 16 бит T0, T1, T2 (T0 и T1 совместимы с традиционными Timer/Counter 0 и Timer/Counter 1 архитектуры 8051).
Захват и генерация сигнала	CCP, PCA, PWM: 3 канала, которые можно использовать как 3 таймера или 3 ЦАП.
Специальные режимы энергопотребления, таймер пробуждения	Есть
Стандартные внешние прерывания	5 каналов: INT0, INT1, ~INT2, ~INT3, ~INT4.
АЦП	8 каналов, 10 бит
Компаратор	Есть
EEPROM	9 килобайт с поддержкой IAP (In-Application Programming), количество перезаписей не менее 100 тысяч раз.
Прерывание при детектировании низкого напряжения	Есть
Сторожевой таймер (WDT)	Есть
Внутренняя система сброса	Есть, порог напряжения сброса опционально настраивается.
Внутренний точный тактовый генератор	Есть
Выходные сигналы тактирования и сброса	Есть
Шифрование загружаемого кода	Есть
Управление по RS485	Есть
Варианты корпусов	SOP28, TSSOP28, SKDIP28, QFN28, SOP20, DIP20, TSSOP20, SOP16, DIP16

Система шифрования кода. С применением ключа шифрования, прошитого в MCU, имеется возможность обновлять программное обеспечение с помощью кнопки update. Для этого в системе программирования выбираются опции "encryption" download и "release project", когда требуется обновить программное обеспечение микроконтроллера. Из-за того, что в памяти программ последние 7 байт используются для хранения глобального идентификатора (global ID), то пространство памяти FLASH, доступное для программы пользователя, уменьшается на эти 7 байт.

Система сброса. В микроконтроллер встроена очень удобная система сброса, так что можно полностью исключить внешние цепочки, формирующие сигнал RESET. По умолчанию вывод P5.4/RST/MCLKO используется как порт ввода/вывода (GPIO), но его можно переконфигурировать как ножку сброса RST с активным уровнем лог. 1, это делается программатором STC-ISP. Порог сброса можно запрограммировать по 16 различным уровням.

Тактирование. В кристалл встроен точный R/C генератор (точность установки частоты ±0.3%). Уход частоты в зависимости от температуры в диапазоне -40..+80°C составляет 1%, в диапазоне -20..+65°C составляет 0.6%. Это позволяет отказаться от подключения дорогого внешнего кварцевого резонатора. Тактовая частота может быть установлена в диапазоне 5..35 МГц (предпочтительные частоты 5.5296, 11.0592, 22.1184, 33.1776 МГц).

Цоколевка и назначение выводов микроконтроллера для 28-выводного корпуса показана в таблице ниже.

№	Мнемоника	Описание/функция	
1	P2.6	Порт ввода/вывода (GPIO).	
2	P2.7		
3	P1.0		
4	P1.1		
5	P1.2/SS/CMPO	P1.2	Порт ввода/вывода (GPIO).
		SS	Slave Selection, сигнал выборки подчиненного устройства интерфейса SPI.
		CMPO	Выходной порт результата сравнения компаратора.
6	P1.3/MOSI	P1.3	Порт ввода/вывода (GPIO).
		MOSI	Master Output Slave Input, сигнал данных интерфейса SPI.
7	P1.4/MISO	P1.4	Порт ввода/вывода (GPIO).
		MISO	Master Input Slave Output, сигнал данных интерфейса SPI.
8	P1.5/SCLK	P1.5	Порт ввода/вывода (GPIO).

		SCLK	Тактовый сигнал интерфейса SPI.
		P1.6	Порт ввода/вывода (GPIO).
		RXD_3	Вход данных UART1
9	P1.6/RXD_3/MCLKO_2	MCLKO_3	Выход инвертирующего усилителя внутренней схемы тактирования. Когда используется внешний генератор тактов, этот вывод должен оставаться не подключенным.
		P1.7	Порт ввода/вывода (GPIO).
10	P1.7/TXD_3	TXD_3	Выход данных UART1.
		P5.4	Порт ввода/вывода (GPIO).
		RST	Вход для сигнала сброса. Уровень лог. 1 на этом выводе длительностью не менее 2 машинных такта приводит к сбросу устройства.
11	P5.4/RST/MCLKO/CMP-	MCLKO	Master clock output, главный выход тактов. Выходная частота может быть равна MCLK, MCLK/2 и MCLK/4. Эта частота может вырабатываться от внутреннего R/C генератора, или от внешнего генератора, или с помощью использования внешнего кварцевого резонатора.
		CMP-	Инверсный вход компаратора.
12	VCC	Плюс напряжения питания.	
		P5.5	Порт ввода/вывода (GPIO).
13	P5.5/CMP+	CMP+	Прямой (без инверсии) вход компаратора.
14	GND	Минус напряжения питания, общий провод.	
		P3.0	Порт ввода/вывода (GPIO).
		RXD	Вход данных UART1.
15	P3.0/RXD/~INT4/T2CLKO	~INT4	Внешнее прерывание 4, которое может сработать только по спаду логического уровня (от лог. 1 к лог. 0). Этот сигнал поддерживает функцию пробуждения из режима пониженного энергопотребления/выключения (power-down wake-up).
		T2CLKO	T2 Clock Output, выход тактов T2. Эта ножка может быть сконфигурирована для вывода тактовой частоты путем установки бита INT_CLKO[2] /T2CLKO.
		P3.1	Порт ввода/вывода (GPIO).
16	P3.1/TXD/T2	TXD	Выход данных UART1.
		T2	Внешний вход тактов для Timer/Counter 2.
		P3.2	Порт ввода/вывода (GPIO).
17	P3.2/INT0	INT0	Внешнее прерывание 0, которое может сработать как по нарастанию, так и по спаду уровня, что определяется настройкой бита IT0 (TCON.0).
		P3.3	Порт ввода/вывода (GPIO).
18	P3.3/INT1	INT1	Внешнее прерывание 1, которое может сработать как по нарастанию, так и по спаду уровня, что определяется настройкой бита IT1 (TCON.2). INT1 поддерживает функцию пробуждения из режима пониженного энергопотребления/выключения (power-down wake-up).
		P3.4	Порт ввода/вывода (GPIO).
		T0	Внешний вход тактов для Timer/Counter 0.
19	P3.4/T0/T1CLKO	T1CLKO	T1 Clock Output, выход тактов T1. Эта ножка может быть сконфигурирована для вывода тактовой частоты путем установки бита INT_CLKO[1] /T1CLKO.
		P3.5	Порт ввода/вывода (GPIO).
		T1	Внешний вход тактов для Timer/Counter 1.
20	P3.5/T1/T0CLKO	T0CLKO	T0 Clock Output, выход тактов T0. Эта ножка может быть сконфигурирована для вывода тактовой частоты путем установки бита INT_CLKO[0] /T0CLKO.
		P3.6	Порт ввода/вывода (GPIO).
21	P3.6/~INT2/RXD_2	~INT2	Внешнее прерывание 2, которое может срабатывать только по спаду уровня (при переходе от лог. 1 к лог. 0). ~INT2 поддерживает функцию пробуждения из режима пониженного энергопотребления/выключения (power-down wake-up).
		RXD_2	Вход данных UART1.
22	P3.7/~INT3/TXD_2	P3.7	Порт ввода/вывода (GPIO).

		~INT3	Внешнее прерывание 2, которое может срабатывать только по спаду уровня (при переходе от лог. 1 к лог. 0). ~INT3 поддерживает функцию пробуждения из режима пониженного энергопотребления/выключения (power-down wake-up).
		TXD_2	Выход данных UART1.
		P2.0	Порт ввода/вывода (GPIO).
23	P2.0/RSTOUT_LOW	RSTOUT_LOW	Выход, который аппаратно генерирует лог. 0 после включения питания и во время сброса, и его можно программно установить в лог. 1.
24	P2.1/SCLK_2	P2.1	Порт ввода/вывода (GPIO).
		SCLK_2	Сигнал тактов интерфейса SPI.
25	P2.2/MISO_2	P2.2	Порт ввода/вывода (GPIO).
		MISO_2	Master Input Slave Output, сигнал данных интерфейса SPI.
26	P2.3/MOSI_2	P2.3	Порт ввода/вывода (GPIO).
		MOSI_2	Master Output Slave Input, сигнал данных интерфейса SPI.
27	P2.4/SS_2	P2.4	Порт ввода/вывода (GPIO).
		SS_2	Slave Selection, сигнал выборки подчиненного устройства интерфейса SPI.
28	P2.5	P2.5	Порт ввода/вывода (GPIO).

Таблица выбора микроконтроллеров из серии STC15W408AS (STC15W401AS, STC15W402AS, STC15W404AS, STC15W408AS, IAP15W413AS, IRC15W415AS):

Параметр	**401AS**	**402AS**	**404AS**	**408AS**	**413AS**	**415AS**
Рабочее напряжение 2.4V..5V	+	+	+	+	+	+
Память программ FLASH, байт	1k	2k	4k	8k	13k	15.5k
Память EEPROM, байт	5k	5k	9k	5k	IAP	
SRAM, байт	512					
UART	1 шт., с возможностью пробуждения MCU					
SPI	Есть					
Таймеры/счетчики	T0, T2					
Поддержка RS485	+	+	+	+	+	+
Компаратор	+	+	+	+	+	+
Аппаратура CCP, PWM, PCA	3 канала, с возможностью пробуждения MCU					
Поддержка RTC	Есть					
Внутренний высокоточный генератор	+	+	+	+	+	+
АЦП	10 бит					
Сторожевой таймер (WDT)	+	+	+	+	+	+
Поддержка DPTR	+	+	+	+	+	+
Детектор низкого напряжения питания	Есть, с генерацией прерывания					
Источники внутреннего надежного сброса	8, по уровню					
Поддержка внешнего прерывания при подаче питания	Более 5V					
Варианты корпусов	SKDIP28, SOP28, TSSOP28, QFN28, PDIP20, SOP20, TSSOP20, PDIP16, SOP16					

На чем писать программы

Можно использовать компилятор Keil C или любой другой компилятор, рассчитанный на семейство MCS-51. Также можно использовать бесплатный инструментарий SDCC (см. папку compiler архива [3]). В опциях проекта следует выбрать ядро Intel 8052, и в качестве заголочного файла определения регистров нужно подключить хедер reg51.h (можете в качестве него использовать файл stc_diyclock-master\src\stc15.h из исходного кода архива [3]).

Чем прошивать

Программа может быть загружена с помощью порта UART через ножки GPIO P3.0 и P3.1. Для этого нужен простой переходничок USB-TTL который можно купить на aliexpress (прогуглите запрос stc-isp programmer

site:aliexpress.com). Также можно купить специальный программатор STC-ISP40PIN, U8 или U8-mini (прогуглите STC-ISP40PIN site:aliexpress.com). Но можно ничего не покупать, для перепрошивки достаточно иметь любой переходик USB-TTL-UART. Итак, процесс перепрошивки по шагам.

1. Вбейте ключевые слова для поиска 6, откройте страничку загрузки на сайте STC, и выберите там последнюю версию утилиты программирования (я скачал stc-isp6.85.rar). На иероглифы не обращайте внимания. Распакуйте из-архива exe-файл, запустите.

2. Из выпадающего списка MCU Type выберите тип Вашего микроконтроллера. Этот список представляет собой дерево, в котором на верхнем уровне перечислены не сами типы микроконтроллеров, а их серии (разделы, список которых можно дополнительно развернуть). Например, чтобы выбрать микроконтроллер STC15W1K24S, нужно сначала в списке выбрать раздел STC15W1K16S Series, и уже в этом разделе выбрать микроконтроллер STC15W1K24S.

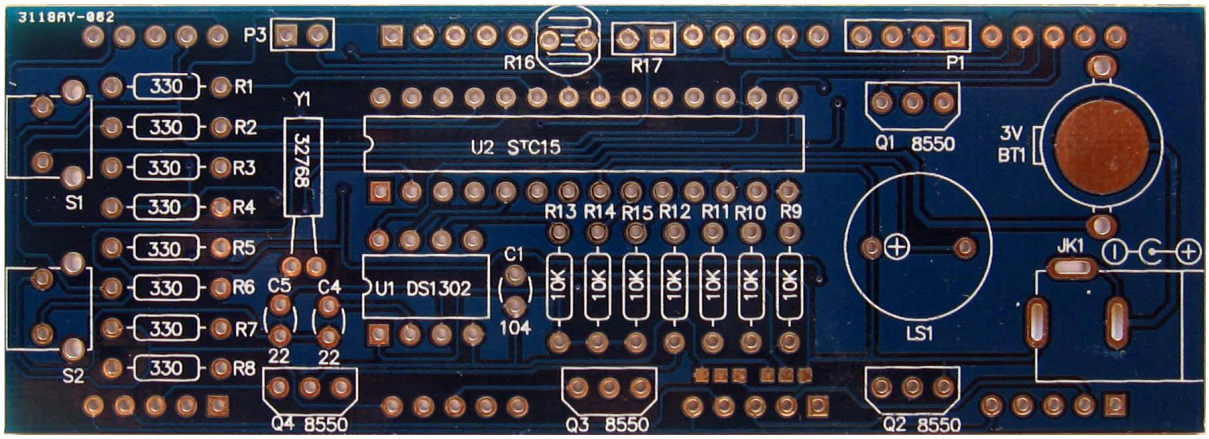
3. Подключите через USB Ваш переходничок USB-TTL-UART (я использовал дешевый USB-SERIAL CH340), выберите его по номеру COM-порта в выпадающем списке COM Port.

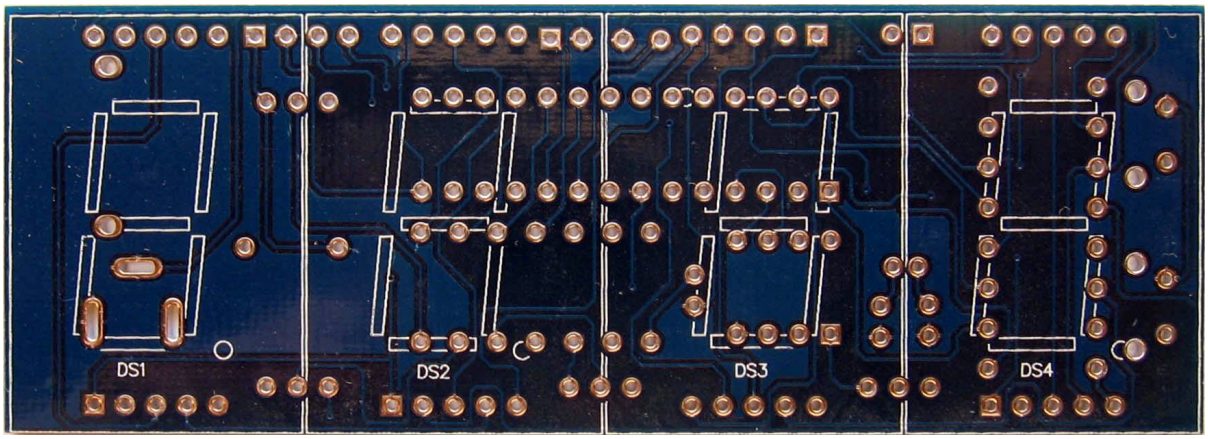
4. Отключите питание от платы, где установлен прошиваемый Вами микроконтроллер STC (микроконтроллер STC должен быть обесточен). Соедините провода GND, TXD и RXD переходничка соответственно с ножками GND, P3.0, P3.1 микроконтроллера.

5. Теперь проверьте, работает ли соединение с программируемым микроконтроллером, следующим образом: нажмите кнопку Check MCU, после чего подайте питание на программируемый микроконтроллер STC. В результате этой операции в консоль утилиты stc-isp будет выдано текст наподобие следующего (это пример проверки STC15W1K24S):

```
Checking target MCU ...
  MCU type: STC15W1K24S
  F/W version: 7.2.5T

Current H/W Option:
  . Current system clock source is internal IRC oscillator
  . IRC frequency: 18.443MHz
  . Wakeup Timer frequency: 36.727KHz
  . Do not detect the level of P3.2 and P3.3 next download
  . Power-on reset, use the extra power-on delay
  . RESET pin behaves as I/O pin
  . Reset while detect a Low-Voltage
  . Thresh voltage level of the built-in LVD : 2.62 V
  . Inhibit EEPROM operation under Low-Voltage
  . CPU-Core supply level : 2.81 V
  . Hardware do not enable Watch-Dog-Timer
  . Watch-Dog-Timer pre-scalar : 256
  . Watch-Dog-Timer stop count in idle mode
  . Program can modify the Watch-Dog-Timer scalar
  . Do not erase user EEPROM area at next download
  . Do not control 485 at next download
  . Do not check user password next download
  . TXD is independent IO
  . TXD pin as quasi-bidirectional mode after reset
  . P2.0 output HIGH level after reset
  . MCU type: STC15W1K24S

  F/W version: 7.2.5T

  Complete !(2017-03-11 15:53:59)
```

Если увидели этот текст, значит программирование работает, и Вы можете прошить в микроконтроллер свою программу. Если же нет, то где-то в подключении допустили ошибку, проверяйте все соединения.

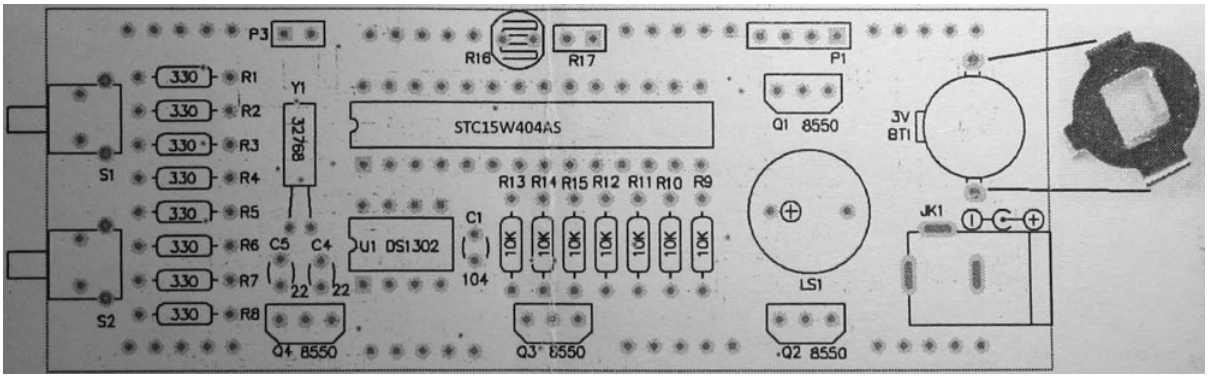




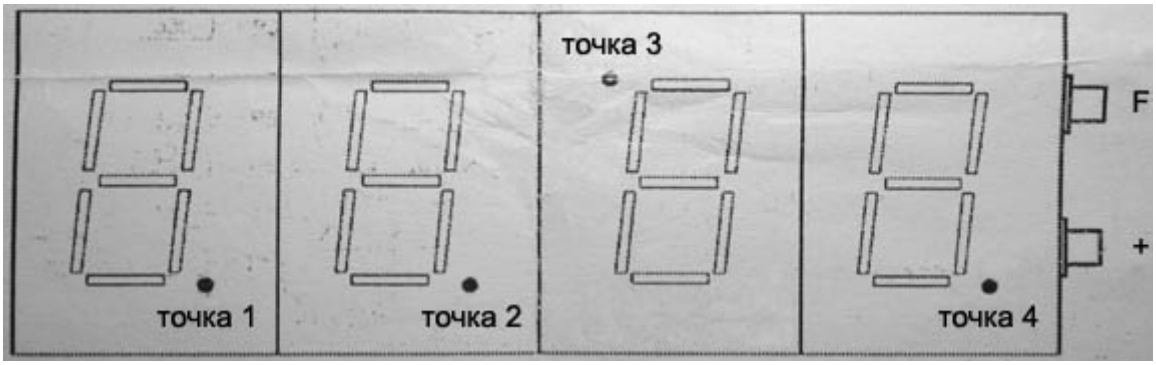
Рекомендуется сначала запаивать короткие, невысокие компоненты - резисторы и конденсаторы. Если начнете с высоких компонентов, то в принципе ничего страшного, однако высокие компоненты будут слегка мешать запаивать низкие. Я сначала запаял резисторы, потом конденсаторы, потом транзисторы, потом коннекторы, потом панельки для микроконтроллера и микросхемы RTC, потом фоторезистор и терморезистор, потом кнопки и держатель батарейки. Само собой, 7-сегментные индикаторы следует запаивать последними, предварительно аккуратно промыв плату спиртом. Промывка нужна не только для эстетики, а чтобы тщательно визуально просмотреть качество монтажа. Также советую перед запайкой индикаторов прозвонить цепи питания на отсутствие коротких замыканий. После того, как запаяли индикаторы и еще раз смыли остатки флюса, можно установить в кроватки микроконтроллер и чип RTC.

Внимание: держатель батарейки советую запаивать только после того, как найдете подходящую батарейку на 3V, потому что возможно его придется немного приподнять над платой (в зависимости от толщины батарейки). Подойдут батарейки типов CR1216 (5034LC), CR1220 (5012LC) толщиной 1.6 и 2 мм соответственно и диаметром 12.5 мм (в комплекте батарейки нет). В крайнем случае можно поставить батарейку CR1025 (5033LC), но она толстовата (2.5 мм), так что держатель батарейки немного выгнется, если его не приподнять над платой при пайке.

Внимание: перед тем, как запаивать семисегментные индикаторы часов, установите и припаяйте все другие компоненты схемы, и тщательно проверьте монтаж. Понятно почему - индикаторы закрывают точки пайки, так что если допустили ошибку, то при смонтированных индикаторах трудно будет что-то исправить.



Внимание: индикаторы следует устанавливать с определенной ориентировкой в разных разрядах, см. картинку ниже.



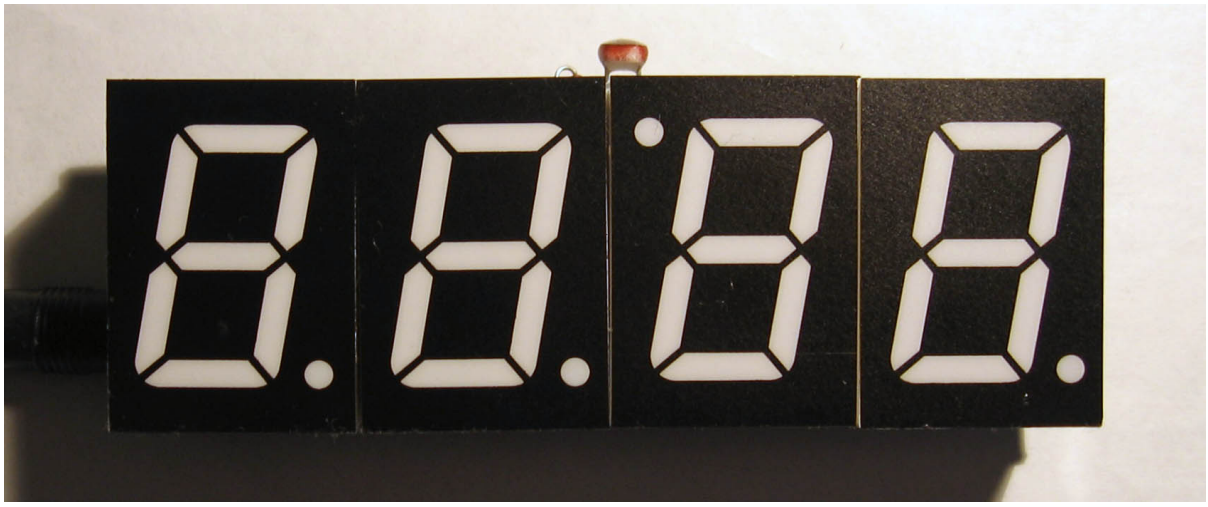
Резисторы, конденсаторы и часовой кварц не имеют полярности, поэтому их установка проста. Но будьте внимательны с полярностью выводов, когда запаиваете транзисторы и бипер. При установке микросхем в кроватки будьте внимательны с ориентировкой корпусов микросхем.

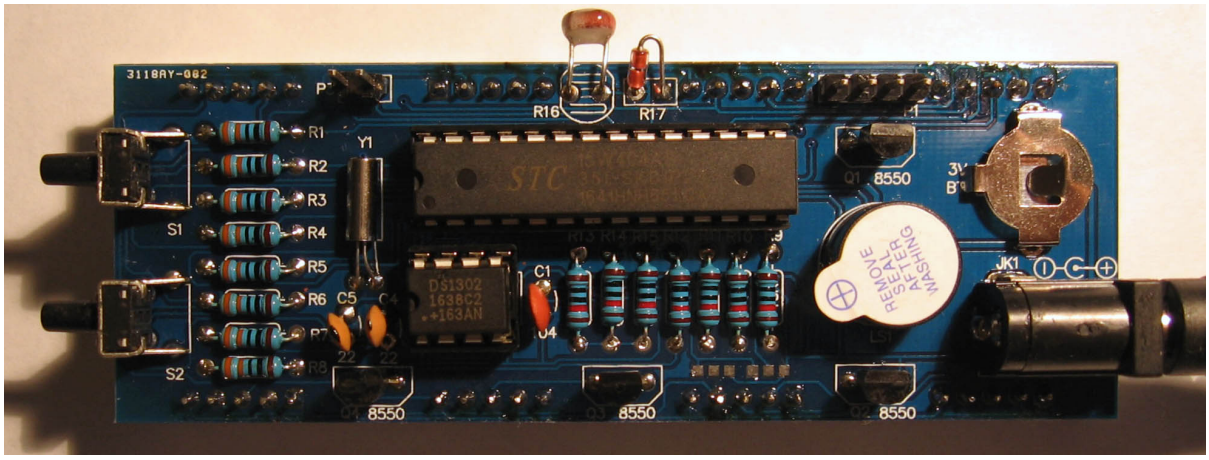
Последней следует установить литиевую батарейку 3V. Без батарейки часы не могут хранить время и настройки при выключении питания.

Ниже в таблице приведен список деталей в том порядке, в каком их следует запаивать на плату.

REFDES	Номинал, тип	Описание
R1..R8	330 ом	Токоограничительные резисторы цепей катодов индикаторов.

R9..R12	10 ком	Токоограничительные резисторы цепей управления ключами анодов индикаторов.
R13	10 ком	Верхний подтягивающий резистор сигнала данных микросхемы RTC DS1302.
R14, R15	10 ком	Верхние подтягивающие резисторы для фоторезистора и терморезистора.
C1	0.1 мкф	Фильтрующий конденсатор по питанию.
C4, C5	22 пф	Корректирующие конденсаторы для правильной работы кварцевого генератора микросхемы RTC.
Y1	32768 Гц	Низкочастотный кварц для микросхемы RTC.
Q1..Q4	SS8550	Биполярные транзисторы P-N-P, анодные ключи индикаторов.
LS1		Бипер на 5V со встроенным генератором.
R16		Фоторезистор, датчик освещенности.
R17		Терморезистор, датчик температуры.
S1, S2		Кнопки управления "F" и "+".
JK1		Гнездо для джека питания 5.5 мм.
P1		
P3		
U1	DS1302	Микросхема RTC (Real Time Clock, часы реального времени).
U2	STC15W404AS	Управляющий микроконтроллер.
BT1		Держатель для 3V литиевой батарейки.
DS1..DS4		Светодиодные 7-сегментные индикаторы.





В комплекте набора идет светофильтр, стенки пластикового корпуса и винты с гайками для его сборки. Для улучшения отображения наклейте на дисплей светофильтр. Поместите конструкцию в корпус.

[**Настройка времени и основных функций**]

После первого включения часы не идут, их необходимо сбросить длительным удержанием обоих кнопок управления в в течение 5 секунд. После этого часы покажут время 7:59. Через 5 секунд сброс завершится, время станет равным 8:00, на некоторое время запищит будильник. Дальше часы покажут текущую температуру в градусах Цельсия и текущую дату. Настроить время, будильник и работу часов можно в нижеуказанном порядке с помощью кнопок "F" ("функция", верхняя кнопка) и "+" (нижняя кнопка).

1. Установка часов. Для того, чтобы установить разряды часов, коротко нажмите на кнопку F. Цифры часов начнут мигать. Нажимайте кнопку + нужное количество раз для установки часов.

2. Установка минут. После установки часов снова коротко нажмите на кнопку F. Начнут мигать цифры минут. Точно так же, как устанавливали часы, установите кнопкой + количество минут. После завершения установки секунды установленной минуты будут отсчитываться от 0.

3. Установка часов будильника. Еще раз нажмите на кнопку F. Будут выведены и начнут мигать цифры часов настройки будильника, но точки в такт секундам при этом мигать не будут. Установите час будильника кнопкой +.

4. Установка минут будильника. Еще раз нажмите на кнопку F. Будут выведены и начнут мигать цифры минут настройки будильника, но точки в такт секундам при этом мигать не будут. Установите разряды минут будильника кнопкой +.

5. Проверка, активен ли будильник. Еще раз нажмите на кнопку F. Цифры на дисплее при этом не поменяются, но светящаяся точка в последнем разряде покажет активность будильника. Нажатия на кнопку + будут переключать работу будильника: если в последнем разряде точка горит, значит будильник активен и сработает в установленное время, а если не горит, то будильник отключен.

6. Настройка почасового бикания. Нажмите кнопку F, разряды часов начнут мигать. Нажимайте кнопку + для изменения начального времени будильника. Например, если установили на 9, то бикания начнутся с 9 часов. Снова нажмите на кнопку F, начнут мигать разряды минут. Кнопка + будет устанавливать время часов, когда ежечасное пищание прекращается. Например, если Вы установили начальное время 9, и конечное время 23, то часы будут бикать каждый час днем, и не будут беспокоить во время сна.

7. Проверка, активна ли функция почасового бикания. Еще раз нажмите на кнопку F. Цифры на дисплее при этом не поменяются, но светящаяся точка в разряде 3 (точка сверху) покажет активность этой функции. Нажатия на кнопку + будут циклически переключать работу функции: если точка в разряде 3 горит, то функция почасового бикания работает, а если не горит, то не работает.

8. Завершение настройки. Нажмите кнопку F последний раз, это завершит настройку часов.

[**Подстройка показаний температуры и настройка даты**]

1. Подстройка температуры. Нажмите кнопку +, часы покажут температуру. Нажимайте кнопку F для коррекции температуры, и для завершения коррекции температуры нажмите кнопку +.

2. Установка даты производится после подстройки температуры. Нажмите кнопку F, будут мигать цифры месяца, настраивайте их кнопкой +. Нажмите кнопку F еще раз, это подтвердит настройку месяца. После этого будут мигать цифры дней, настройте их кнопкой +. Нажатие кнопки F еще раз подтвердит настройку дней и переведет часы в настройку дня недели. Будет мигать цифра дня недели, кнопкой + её можно настроить. Нажмите кнопку F еще раз, чтобы подтвердить настройку недели, и нажмите кнопку + для завершения настройки.

[**Работа часов**]

45 секунд отображается текущее время, 5 секунд отображается температура, 5 секунд отображается дата, и еще 5 секунд отображается день недели. Далее по циклу процесс повторяется.

Ночью, когда освещение падает, яркость часов автоматически уменьшается.

Порт P1.4 может использоваться для управления реле (в зависимости от залитой прошивки).

Коннектор P1 может использоваться для подключения адаптера USB-UART с целью загрузки программного обеспечения. Назначение коннектора P3 мне неизвестно.

[**Недостатки часов**]

Китайцы постарались максимально удешевить конструкцию. По этой причине имеются недостатки, которые можно исправить либо корректировкой прошивки, либо с небольшой модификацией схемы.

1. При затемнении датчика освещенности (фоторезистор R16) яркость регулируется ступенчато, не плавно.

2. Немного удивило то, что по питанию стоит единственный фильтрующий конденсатор C1 емкостью 0.1 мкф. Это означает, что для питания можно применять стабилизированный источник питания, желательно с низкими пульсациями по питанию. В комплекте прилагается USB-кабель с джеком, которым можно подать питание на часы от компьютера или ноутбука.

3. В качестве датчика применяется терморезистор. Вместо терморезистора можно подключить датчик температуры типа DS18S20 или DS18B20 с интерфейсом 1-Wire, после этого коррекция измерения температуры не потребуется.

4. Нет возможности отключения будильника в определенные дни недели. Например, чтобы он не срабатывал в субботу и воскресенье.

[**Исходный код (автор Jens Jensen, github.com/zerog2k/stc_diyclock)**]

[**adc.h**]

```
/*---------------------------------------------------------------*/
/* --- STC MCU International Limited ---------------------------*/
/* --- STC 15 Series MCU A/D Conversion Demo -------------------*/
/* --- Mobile: (86)13922805190 ---------------------------------*/
/* --- Fax: 86-755-82944243 ------------------------------------*/
/* --- Tel: 86-755-82948412 ------------------------------------*/
/* --- Web: www.STCMCU.com -------------------------------------*/
/*---------------------------------------------------------------*/
#include "stc15.h"
```

```
#include < stdint.h >

#define _nop_ __asm nop __endasm;

/* Определение констант операций ADC для ADC_CONTR */
#define ADC_POWER   0x80         //бит управления питанием ADC
#define ADC_FLAG    0x10         //бит завершения преобразования ADC
#define ADC_START   0x08         //бит управления запуском ADC
#define ADC_SPEEDLL 0x00         //540 тактов
#define ADC_SPEEDL  0x20         //360 тактов
#define ADC_SPEEDH  0x40         //180 тактов
#define ADC_SPEEDHH 0x60         //90 тактов

/*----------------------------
Инициализация регистров ADC
----------------------------*/
void InitADC(uint8_t chan);

/*----------------------------------
Получение 10-битного результата ADC
----------------------------------*/
uint16_t getADCResult(uint8_t chan);

/*----------------------------------
Получение 8-битного результата ADC
----------------------------------*/
uint8_t getADCResult8(uint8_t chan);
```

[**adc.c**]

```
/*---------------------------------------------------------------------*/
/* --- STC MCU International Limited ---------------------------------*/
/* --- STC 15 Series MCU A/D Conversion Demo -------------------------*/
/* --- Mobile: (86)13922805190 ---------------------------------------*/
/* --- Fax: 86-755-82944243 ------------------------------------------*/
/* --- Tel: 86-755-82948412 ------------------------------------------*/
/* --- Web: www.STCMCU.com -------------------------------------------*/
/*---------------------------------------------------------------------*/
#include "stc15.h"
#include "adc.h"

/*----------------------------
Инициализация регистров ADC
----------------------------*/
void InitADC(uint8_t chan)
{
   P1ASF |= 1 << chan;                  //разрешение функции канала ADC
   ADC_RES = 0;                         //очистка предыдущего результата
   ADC_CONTR = ADC_POWER | ADC_SPEEDLL;//включение ADC
   //Delay(2);                          // и задержка
}

/*----------------------------------
Получение 10-битного результата ADC
----------------------------------*/
uint16_t getADCResult(uint8_t chan)
{
   ADC_CONTR = ADC_POWER | ADC_SPEEDHH | ADC_START | chan;
   _nop_;                             //нужно подождать перед запросом
   while (!(ADC_CONTR & ADC_FLAG)); //ожидание флага завершения преобразования
   ADC_CONTR &= ~ADC_FLAG;          //закрытие ADC
   return  ADC_RES << 2 | (ADC_RESL & 0b11) ;   //возврат значения ADC
}

/*----------------------------------
Получение 8-битного результата ADC
----------------------------------*/
uint8_t getADCResult8(uint8_t chan)
{
   ADC_CONTR = ADC_POWER | ADC_SPEEDHH | ADC_START | chan;
   _nop_;                             //нужно подождать перед запросом
   while (!(ADC_CONTR & ADC_FLAG)); //ожидание флага завершения преобразования
   ADC_CONTR &= ~ADC_FLAG;          //закрытие ADC
   return ADC_RES;                  //возврат значения ADC
}
```

[**ds1302.h**]

```
// DS1302 RTC IC
```

```
// http://datasheets.maximintegrated.com/en/ds/DS1302.pdf
#include "stc15.h"
#include < stdint.h >

#define _nop_ __asm nop __endasm;

#define DS_CE    P1_0
#define DS_IO    P1_1
#define DS_SCLK  P1_2

#define DS_CMD        1 << 7
#define DS_CMD_READ   1
#define DS_CMD_WRITE  0
#define DS_CMD_RAM    1 << 6
#define DS_CMD_CLOCK  0 << 6

#define DS_ADDR_SECONDS     0
#define DS_ADDR_MINUTES     1
#define DS_ADDR_HOUR        2
#define DS_ADDR_DAY         3
#define DS_ADDR_MONTH       4
#define DS_ADDR_WEEKDAY     5
#define DS_ADDR_YEAR        6
#define DS_ADDR_WP          7
#define DS_ADDR_TCSDS       8

#define DS_BURST_MODE       31

#define DS_MASK_SECONDS       0b01111111
#define DS_MASK_SECONDS_TENS  0b01110000
#define DS_MASK_SECONDS_UNITS 0b00001111
#define DS_MASK_MINUTES       0b01111111
#define DS_MASK_MINUTES_TENS  0b01110000
#define DS_MASK_MINUTES_UNITS 0b00001111
#define DS_MASK_AMPM_MODE     0b10000000
#define DS_MASK_PM            0b00100000
#define DS_MASK_HOUR12        0b00011111
#define DS_MASK_HOUR12_TENS   0b00010000
#define DS_MASK_HOUR24        0b00111111
#define DS_MASK_HOUR24_TENS   0b00110000
#define DS_MASK_HOUR_UNITS    0b00001111
#define DS_MASK_DAY           0b00111111
#define DS_MASK_DAY_TENS      0b00110000
#define DS_MASK_DAY_UNITS     0b00001111
#define DS_MASK_MONTH         0b00011111
#define DS_MASK_MONTH_TENS    0b00010000
#define DS_MASK_MONTH_UNITS   0b00001111
#define DS_MASK_WEEKDAY       0b00000111
#define DS_MASK_YEAR          0b11111111
#define DS_MASK_YEAR_TENS     0b11110000
#define DS_MASK_YEAR_UNITS    0b00001111

/*
  Примечание: биты RTC и конфигурации изначально находились
  в структурах/объединениях (structs/unions), но по некоторой
  причине выходной код sdcc слишком разросся. Это прямая попытка
  заново создать функциональность структуры/объединения путем
  явного выбора адресации бита/байта iram масками, заданными выше.
  Из-за того, что эта текущая реализация более сложная, чем
  с помощью структур/объединений, есть некая надежда вернуться
  к доступу через struct/union, если это позволит размер кода
  (может быть, это даст оптимизация sdcc?).*/

// 8051 область памяти 0x20 .. 0x2F в IRAM может быть доступна
// как биты (по адресам 0x00 .. 0x7F)
// 0x20, бит 0 это бит по адресу __bit __at (0x00),
// бит 7 это бит по адресу __bit __at (0x07), и так далее...

uint8_t __at (0x24) rtc_table[8];

// h12.tenhour в RTC по адресу 0x26, бит 4 -> => 0x26-0x20 => 0x6*8+4 => 52 => 0x34
__bit __at (0x34) H12_TH;
// h12.pm в RTC по адресу 0x26, бит 5 -> => 0x26-0x20 => 0x6*8+5 => 53 => 0x35
__bit __at (0x35) H12_PM;
// hour_12_24 в RTC по адресу 0x26, бит 7 -> => 0x26-0x20 => 0x6*8+7 => 55 => 0x37
__bit __at (0x37) H12_24;

// Конфигурация в DS1302 RAM
uint8_t __at (0x2c) cfg_table[4];
```

```
#define CFG_ALARM_HOURS_BYTE   0
#define CFG_ALARM_MINUTES_BYTE 1
#define CFG_TEMP_BYTE          2

#define CFG_ALARM_HOURS_MASK   0b11111000
#define CFG_ALARM_MINUTES_MASK 0b00111111
#define CFG_TEMP_MASK          0b00000111

// Смещение 0 => alarm_hour (7..3) / chime_on (2) / alarm_on (1) / temp_C_F (0)
// Смещение 1 => (7) не используется / (6) sw_mmdd / alarm_minute (5..0)
// Смещение 2 => chime_hour_start (7..3) / temp_offset (2..0), signed -4 / +3
// Смещение 3 => (7),(6)&(5) не используется / chime_hour_stop (4..0)

// temp_C_F в конфигурации находится по адресу 0x2c,
// бит 0 => 0x2c-0x20 => 0xc*8+0 => 96 => 0x60
__bit __at (0x60) CONF_C_F;
__bit __at (0x61) CONF_ALARM_ON;
__bit __at (0x62) CONF_CHIME_ON;
__bit __at (0x6E) CONF_SW_MMDD;

// Функции DS1302
void ds_ram_config_init();
void ds_ram_config_write();

// Чтение одного байта ds1302
uint8_t ds_readbyte(uint8_t addr);

// Чтение ds1302 пакетом 8 байт в структуру
void ds_readburst();

// Запись одного байта ds1302
void ds_writebyte(uint8_t addr, uint8_t data);

// Очистка WP, CH
void ds_init();

// Сброс даты/времени в 01/01 00:00
void ds_reset_clock();

// переключение режима часов 12/24
void ds_hours_12_24_toggle();

// инкремент часов
void ds_hours_incr();

// инкремент минут
void ds_minutes_incr();

// инкремент месяца
void ds_month_incr();

// инкремент дней
void ds_day_incr();

void ds_weekday_incr();
void ds_sec_zero();

// разбиение bcd на int
uint8_t ds_split2int(uint8_t tens_ones);

// возврат байта в кодировке bcd, полученного из целого числа
uint8_t ds_int2bcd(uint8_t integer);

// преобразование целого числа на части bcd (старшая тетрада десятки,
// младшая тетрада единицы)
uint8_t ds_int2bcd_tens(uint8_t integer);
uint8_t ds_int2bcd_ones(uint8_t integer);
```

[**dc1302.c**]

```
// Модуль для работы с микросхемой DS1302 часов реального времени (RTC)
// http://datasheets.maximintegrated.com/en/ds/DS1302.pdf
#pragma callee_saves sendbyte,readbyte
#pragma callee_saves ds_writebyte,ds_readbyte

#include "ds1302.h"

#define MAGIC_HI  0x5A
#define MAGIC_LO  0xA5
```

```
void ds_ram_config_init()
{
   uint8_t i,j;
   // проверка magic-байтов, чтобы определить, была ли ранее записана ram
   if ( (ds_readbyte( DS_CMD_RAM >> 1 | 0x00) != MAGIC_LO
      || ds_readbyte( DS_CMD_RAM >> 1 | 0x01) != MAGIC_HI) )
   {
      // если нет, то нужно инициализировать конфигурацию ram значениями
      // по умолчанию
      ds_writebyte( DS_CMD_RAM >> 1 | 0x00, MAGIC_LO);
      ds_writebyte( DS_CMD_RAM >> 1 | 0x01, MAGIC_HI);
      // ОПТИМИЗАЦИЯ: генерировать ljmp в ds_ram_config_write
      ds_ram_config_write();
      return;
   }

   // чтение конфигурации ram
   // ОПТИМИЗАЦИЯ: условие завершения цикла !=4 даст код меньше, чем < 4
   // ОПТИМИЗАЦИЯ: было cfg_table[i] = ds_readbyte(DS_CMD_RAM >> 1 | (i+2));
   //              использование второй переменной вместо DS_CMD_RAM>>1 | (i+2)
   //              дает код меньшего размера
   j=DS_CMD_RAM>>1|2;
   for (i=0; i!=4; i++)
      cfg_table[i] = ds_readbyte(j++);
}

void ds_ram_config_write()
{
   uint8_t i,j;
   j=DS_CMD_RAM>>1|2;
   for (i=0; i!=4; i++)
      ds_writebyte(j++, cfg_table[i]);
}

void sendbyte(uint8_t b)
{
   b;
   __asm
   push  ar7
   mov   a,dpl
   mov   r7,#8
00001$:
   nop
   nop
   rrc   a
   mov   _P1_1,c
   setb  _P1_2
   nop
   nop
   clr   _P1_2
   djnz  r7,00001$
   pop   ar7
   __endasm;
}

uint8_t readbyte()
{
   __asm
   push  ar7
   mov   a,#0
   mov   r7,#8
00002$:
   nop
   nop
   mov   c,_P1_1
   rrc   a
   setb  _P1_2
   nop
   nop
   clr   _P1_2
   djnz  r7,00002$
   mov   dpl,a
   pop   ar7
   __endasm;
}

uint8_t ds_readbyte(uint8_t addr)
{
   // чтение одного байта из ds1302
   uint8_t b;
```

```
   b = DS_CMD | DS_CMD_CLOCK | addr << 1 | DS_CMD_READ;
   DS_CE = 0;
   DS_SCLK = 0;
   DS_CE = 1;
   // отправка байта команды
   sendbyte(b);
   // чтение байта
   b=readbyte();
   DS_CE = 0;
   return b;
}

void ds_readburst()
{
   // чтение ds1302 пакетом 8 байт
   uint8_t j, b;
   b = DS_CMD | DS_CMD_CLOCK | DS_BURST_MODE << 1 | DS_CMD_READ;
   DS_CE = 0;
   DS_SCLK = 0;
   DS_CE = 1;
   // отправка байта команды
   sendbyte(b);
   // чтение байт
   for (j=0; j!=8; j++)
      rtc_table[j] = readbyte();
   DS_CE = 0;
}

void ds_writebyte(uint8_t addr, uint8_t data)
{
   // запись одного байта в ds1302
   uint8_t b = 0;
   b = DS_CMD | DS_CMD_CLOCK | addr << 1 | DS_CMD_WRITE;
   DS_CE = 0;
   DS_SCLK = 0;
   DS_CE = 1;
   // отправка байта команды
   sendbyte(b);
   // отправка байта данных
   sendbyte(data);
   DS_CE = 0;
}

void ds_init()
{
   uint8_t b = ds_readbyte(DS_ADDR_SECONDS);
   ds_writebyte(DS_ADDR_WP, 0);        // очистка WP
   b &= ~(0b10000000);
   ds_writebyte(DS_ADDR_SECONDS, b);   // очистка CH
}

// Сброс даты и времени
void ds_reset_clock()
{
   ds_writebyte(DS_ADDR_MINUTES, 0x00);
   ds_writebyte(DS_ADDR_HOUR,  DS_MASK_AMPM_MODE|0x07);
   ds_writebyte(DS_ADDR_MONTH, 0x01);
   ds_writebyte(DS_ADDR_DAY,   0x01);
}

void ds_hours_12_24_toggle()
{
   uint8_t hours,b;
   if (H12_24)
   {  // 12h->24h
      // часы в формате 12h (1-11am 12pm 1-11pm 12am)
      hours=ds_split2int(rtc_table[DS_ADDR_HOUR]&DS_MASK_HOUR12);
      if (hours==12)
      {
         if (!H12_PM) hours=0;
      }
      else
      {
         if (H12_PM) hours+=12;
      }// в 24h формат
      b = ds_int2bcd(hours);  // очистка бита hour_12_24 bit
   }
   else
   {  // 24h->12h
      // часы в формате 24h (0-23, 0-11=>am , 12-23=>pm)
```

```
      hours = ds_split2int(rtc_table[DS_ADDR_HOUR]&DS_MASK_HOUR24);
      b = DS_MASK_AMPM_MODE;
      if (hours >= 12) { hours-=12; b|=0x20; }  // pm
      if (hours == 0) { hours=12; }             //12am
      b |= ds_int2bcd(hours);
   }
   ds_writebyte(DS_ADDR_HOUR,b);
}

// инкремент часов
void ds_hours_incr()
{
   uint8_t hours, b = 0;
   if (!H12_24)
   {
      // 24-часовой формат
      hours = ds_split2int(rtc_table[DS_ADDR_HOUR]&DS_MASK_HOUR24);
      if (hours < 23)
         hours++;
      else
         hours = 00;
      b = ds_int2bcd(hours);  // бит 7 = 0
   }
   else
   {
      // 12-часовой формат
      hours = ds_split2int(rtc_table[DS_ADDR_HOUR]&DS_MASK_HOUR12);
      if (hours < 12)
         hours++;
      else
      {
         hours = 1;
         H12_PM=!H12_PM;
      }
      b = (H12_PM?(DS_MASK_AMPM_MODE|DS_MASK_PM):DS_MASK_AMPM_MODE)|ds_int2bcd(hours);
   }
   ds_writebyte(DS_ADDR_HOUR, b);
}

// инкремент минут
void ds_minutes_incr()
{
   uint8_t minutes = ds_split2int(rtc_table[DS_ADDR_MINUTES]&DS_MASK_MINUTES);
   if (minutes < 59)
      minutes++;
   else
      minutes = 1;
   ds_writebyte(DS_ADDR_MINUTES, ds_int2bcd(minutes));
}

// инкремент месяца
void ds_month_incr()
{
   uint8_t month = ds_split2int(rtc_table[DS_ADDR_MONTH]&DS_MASK_MONTH);
   if (month < 12)
      month++;
   else
      month = 1;
   ds_writebyte(DS_ADDR_MONTH, ds_int2bcd(month));
}

// инкремент дней
void ds_day_incr()
{
   uint8_t day = ds_split2int(rtc_table[DS_ADDR_DAY]&DS_MASK_DAY);
   if (day < 31)
      day++;
   else
      day = 1;
   ds_writebyte(DS_ADDR_DAY, ds_int2bcd(day));
}

void ds_weekday_incr()
{
   uint8_t day = rtc_table[DS_ADDR_WEEKDAY];
   if (day < 7)
      day++;
   else
      day=1;
   ds_writebyte(DS_ADDR_WEEKDAY, day);
```

```
   rtc_table[DS_ADDR_WEEKDAY] = day;   // полезно ?
}

void ds_sec_zero()
{
   rtc_table[DS_ADDR_SECONDS]=0;
   ds_writebyte(DS_ADDR_SECONDS,0);
}

uint8_t ds_split2int(uint8_t tens_ones)
{
   return (tens_ones>>4) * 10 + (tens_ones&0xF);
}

// вернет байт в кодировке bcd, полученный из целого числа
uint8_t ds_int2bcd(uint8_t integer)
{
   return integer / 10 << 4 | integer % 10;
}

uint8_t ds_int2bcd_tens(uint8_t integer)
{
   return integer / 10 % 10;
}

uint8_t ds_int2bcd_ones(uint8_t integer)
{
   return integer % 10;
}
```

[**led.h**]

```
// Функции для управления 4-разрядным цифровым
// семисегментным светодиодным (LED) индикатором
#include < stdint.h >

// индексы для ledtable[]
#define LED_a   0xa
#define LED_b   0xb
#define LED_c   0xc
#define LED_d   0xd
#define LED_e   0xe
#define LED_f   0xf
#define LED_BLANK  0x10
#define LED_DASH   0x11
#define LED_h   0x12
#define LED_dp  0x13

const uint8_t __at (0x1000) ledtable[] =
{
    // таблица для генерации цифр
    // dp,g,f,e,d,c,b,a
    0b11000000, // was 0b00111111, // 0
    0b11111001, //     0b00000110, // 1
    0b10100100, //     0b01011011, // 2
    0b10110000, //     0b01001111, // 3
    0b10011001, //     0b01100110, // 4
    0b10010010, //     0b01101101, // 5
    0b10000010, //     0b01111101, // 6
    0b11111000, //     0b00000111, // 7
    0b10000000, //     0b01111111, // 8
    0b10011000, //     0b01100111, // 9
    0b10001000, //     0b01110111, // A
    0b10000011, //     0b01111100, // b
    0b11000110, //     0b00111001, // C
    0b10100001, //     0b01011110, // d
    0b10000110, //     0b01111001, // E
    0b10001110, //     0b01110001, // F
    0b11111111, //     0b00000000, // 0x10 - ' '
    0b10111111, //     0b01000000, // 0x11 - '-'
    0b10001011, //     0b01110100, // 0x12 - 'h'
    0b01111111, //     0b10000000, // 0x13 - '.'
};

// Некоторые буквенные символы abc .. def
const uint8_t __at (0x1100) ledtable2[]  =
{
    0b11000000, // was 0b00111111, // 0
    0b11001111, //     0b00000110, // 1
    0b10100100, //     0b01011011, // 2
```

```
    0b10000110, //     0b01001111, // 3
    0b10001011, //     0b01100110, // 4
    0b10010010, //     0b01101101, // 5
    0b10010000, //     0b01111101, // 6
    0b11000111, //     0b00000111, // 7
    0b10000000, //     0b01111111, // 8
    0b10000011, //     0b01100111, // 9
    0b10000001, //     0b01110111, // A
    0b10011000, //     0b01111100, // b
    0b11110000, //     0b00111001, // C
    0b10001100, //     0b01011110, // d
    0b10110000, //     0b01111001, // E
    0b10110001, //     0b01110001, // F
    0b11111111, //     0b00000000, // 0x10 - ' '
    0b10111111, //     0b01000000, // 0x11 - '-'
    0b10011001, //     0b01110100, // 0x12 - 'h'
    0b01111111, //     0b10000000, // 0x13 - '.'
};

uint8_t tmpbuf[4];
__bit   dot0;
__bit   dot1;
__bit   dot2;
__bit   dot3;

uint8_t dbuf[4];

#define clearTmpDisplay() { dot0=0; dot1=0; dot2=0; dot3=0; tmpbuf[0]=tmpbuf[1]=tmpbuf[2]=tmpbuf[3]=LED_BLANK; }

#define filldisplay(pos,val,dp) { tmpbuf[pos]=(uint8_t)(val); if (dp) dot##pos=1;}
#define dotdisplay(pos,dp) { if (dp) dot##pos=1;}

#define updateTmpDisplay() { uint8_t tmp; \
                        tmp=ledtable[tmpbuf[0]]; if (dot0) tmp&=0x7F; dbuf[0]=tmp; \
                        tmp=ledtable[tmpbuf[1]]; if (dot1) tmp&=0x7F; dbuf[1]=tmp; \
                        tmp=ledtable[tmpbuf[3]]; if (dot3) tmp&=0x7F; dbuf[3]=tmp; \
                        tmp=ledtable2[tmpbuf[2]]; if (dot2) tmp&=0x7F; dbuf[2]=tmp; }
```

[**main.c**]

```
// STC15F204EA DIY LED Clock
// Copyright 2016, Jens Jensen
// Можно применить STC15F404AS
#include "stc15.h"
#include < stdint.h >
#include < stdio.h >
#include "adc.h"
#include "ds1302.h"
#include "led.h"

#define FOSC    11059200

// Очистка сторожевого таймера (WDT, Watch Dog Timer)
#define WDT_CLEAR()    (WDT_CONTR |= 1 << 4)

// Псевдонимы для выходов управления реле и пищалки.
// Реле использовалось для инидикации состояния
// главного цикла main.
#define RELAY   P1_4
#define BUZZER  P1_5

// Каналы АЦП для сенсоров:
#define ADC_LIGHT 6
#define ADC_TEMP  7

// Псевдонимы для кнопок:
#define SW2     P3_0
#define S2      1
#define SW1     P3_1
#define S1      0

// Состояния режима клавиатуры:
enum keyboard_mode {
   K_NORMAL,
   K_WAIT_S1,
   K_WAIT_S2,
   K_SET_HOUR,
   K_SET_MINUTE,
   K_SET_HOUR_12_24,
   K_SEC_DISP,
```

```
   K_TEMP_DISP,
   K_DATE_DISP,
   K_DATE_SWDISP,
   K_SET_MONTH,
   K_SET_DAY,
   K_WEEKDAY_DISP,
   K_DEBUG
};

// Состояния режима дисплея:
enum display_mode {
   M_NORMAL,
   M_SET_HOUR_12_24,
   M_SEC_DISP,
   M_TEMP_DISP,
   M_DATE_DISP,
   M_WEEKDAY_DISP,
   M_DEBUG
};

/* ------------------------------------------------------------------------- */

void _delay_ms(uint8_t ms)
{
   // i,j выбраны для тактовой частоты fosc=11.0592 МГц с помощью осциллографа.
   // Инструментарий stc-isp дает неточные результаты (может быть, для C51
   // в отличие sdcc?).
   // max 255 мс
   uint8_t i, j;
   do
   {
      i = 4;
      j = 240;
      do
      {
         while (--j);
      } while (--i);
   } while (--ms);
}

// Глобальные переменные:
uint8_t  count;      // раньше было uint16, но 8 бит достаточно
uint16_t temp;       // значение сенсора температуры
uint8_t  lightval;   // значение сенсора освещенности

volatile uint8_t displaycounter;
volatile uint8_t _100us_count;
volatile uint8_t _10ms_count;

uint8_t dmode = M_NORMAL;        // состояние режима дисплея
uint8_t kmode = K_NORMAL;        // состояние режима клавиатуры
uint8_t smode,lmode;

volatile __bit  display_colon;   // мигание двоеточия
__bit  flash_01;
__bit  flash_23;
__bit  beep = 1;

volatile __bit  S1_LONG;
volatile __bit  S1_PRESSED;
volatile __bit  S2_LONG;
volatile __bit  S2_PRESSED;

volatile uint8_t debounce[2];    // буфер подавления дребезга контактов
volatile uint8_t switchcount[2];
#define SW_CNTMAX 80

void timer0_isr() __interrupt 1 __using 1
{
   // Обработчик прерывания (ISR) обновления дисплея.
   // За 1 проход обрабатывается одна цифра.
   uint8_t digit = displaycounter % 4;
   // Выключение всех цифр, установка уровней лог. 1.
   P3 |= 0x3C;
   // Автоподстройка яркости: пропускается несколько тактов
   if (displaycounter % lightval < 4 )
   {
      // заполнение цифр
      P2 = dbuf[digit];
      // включение выбранной цифры установкой лог. 0
```

```
      P3 &= ~(0x4 << digit);
   }
   displaycounter++;
   // делитель: для получения интервалов 10 мс
   if (++_100us_count == 100)
   {
      _100us_count = 0;
      _10ms_count++;
      // код для мигания двоеточия, 500 мс
      if (_10ms_count == 50)
      {
         display_colon = !display_colon;
         _10ms_count = 0;
      }
      // чтение кнопок с подавлением дребезга:
      // инкремент счетчика, если контакты замкнуты
      if ((debounce[0]) == 0x00)
      {
         // удержание как минимум на 8 тиков
         S1_PRESSED = 1;
         switchcount[0]++;
      }
      else
      {
         // кнопка отпущена, или дребезг, сброс состояния
         S1_PRESSED = 0;
         switchcount[0] = 0;
      }
      if ((debounce[1]) == 0x00)
      {
         // удержание как минимум на 8 тиков
         S2_PRESSED = 1;
         switchcount[1]++;
      }
      else
      {
         // кнопка отпущена, или дребезг, сброс состояния
         S2_PRESSED = 0;
         switchcount[1] = 0;
      }
      // код подавления дребезга
      if (switchcount[0] > SW_CNTMAX)
         {switchcount[0] = SW_CNTMAX; S1_LONG=1;}
      if (switchcount[1] > SW_CNTMAX)
         {switchcount[1] = SW_CNTMAX; S2_LONG=1;}
      // чтение состояния кнопок в смещающееся 8-битное окно
      debounce[0] = (debounce[0] << 1) | SW1;
      debounce[1] = (debounce[1] << 1) | SW2;
   }
}

//Интервал таймера 100 мкс на частоте 11.0592 МГц
void Timer0Init(void)
{
   TL0 = 0xA4;TH0 = 0xFF;  //Начальное значение таймера
   TF0 = 0;                //Очистка флага TF0
   TR0 = 1;                //Запуск в работу Timer0
   ET0 = 1;                //Разрешение прерывания от Timer0
   EA = 1;                 //Глобальное разрешение прерываний
}

#define getkeypress(a) a##_PRESSED

int8_t gettemp(uint16_t raw) {
   // Формула для преобразования значения АЦП, подключенного
   // к терморезистору ntc (Negative Temperature Coeff,
   // терморезистор с отрицательным коэффициентом)
   // в приблизительное значение градусов Цельсия:
   return 76 - raw * 64 / 637;
}

/*****************************************/
int main()
{
   // НАСТРОЙКА
   // установка выводов фоторезистора и терморезистора в режим выхода
   // с открытым коллектором:
   P1M1 |= (1 << 6) | (1 << 7);
   P1M0 |= (1 << 6) | (1 << 7);
```

```
// инициализация RTC (микросхема часов реального времени DS1302):
ds_init();
// инициализация/чтение конфигурации ram:
ds_ram_config_init();

// Раскомментируйте, чтобы сбросить минуты и часы в 0.
// Использовалось для отладки, сейчас это не нужно.
//ds_reset_clock();

Timer0Init();  // настройка таймера для обновления дисплея и чтения кнопок

// ГЛАВНЫЙ ЦИКЛ
while(1)
{
   RELAY = 0;
   _delay_ms(100);
   RELAY = 1;
   // Часто запускаемый опрос АЦП:
   if ((count % 4) == 0)
   {
      // автоподстройка яркости цифр путем деления диапазона АЦП на 8 шагов:
      lightval = getADCResult8(ADC_LIGHT) >> 3;
      temp = gettemp(getADCResult(ADC_TEMP))
           + (cfg_table[CFG_TEMP_BYTE]&CFG_TEMP_MASK)
           - 4;
      // установка нижнего порога подстройки яркости индикатора:
      if (lightval < 4)
         lightval = 4;
   }
   ds_readburst();   // чтение RTC
   // Дерево принятия решений при опросе клавиатуры:
   switch (kmode)
   {
   case K_SET_HOUR:
      flash_01 = !flash_01;
      if (! flash_01)
      {
         if (getkeypress(S2)) ds_hours_incr();
         if (getkeypress(S1)) kmode = K_SET_MINUTE;
      }
      break;
   case K_SET_MINUTE:
      flash_01 = 0;
      flash_23 = !flash_23;
      if (! flash_23)
      {
         if (getkeypress(S2)) ds_minutes_incr();
         if (getkeypress(S1)) kmode = K_SET_HOUR_12_24;
      }
      break;
   case K_SET_HOUR_12_24:
      dmode=M_SET_HOUR_12_24;
      if (getkeypress(S2)) ds_hours_12_24_toggle();
      if (getkeypress(S1)) kmode = K_NORMAL;
       break;

   case K_TEMP_DISP:
      dmode=M_TEMP_DISP;
      if (getkeypress(S1))
      {
         uint8_t offset=cfg_table[CFG_TEMP_BYTE]&CFG_TEMP_MASK;
         offset++; offset&=CFG_TEMP_MASK;
         cfg_table[CFG_TEMP_BYTE]=(cfg_table[CFG_TEMP_BYTE]&~CFG_TEMP_MASK)|offset;
      }
      if (getkeypress(S2)) kmode = K_DATE_DISP;
      break;

   case K_DATE_DISP:
      dmode=M_DATE_DISP;
      if (getkeypress(S1))
      {
         kmode=K_WAIT_S1;
         lmode=CONF_SW_MMDD?K_SET_DAY:K_SET_MONTH;
         smode=K_DATE_SWDISP;
      }
      if (getkeypress(S2)) kmode = K_WEEKDAY_DISP;
      break;
   case K_DATE_SWDISP:
      CONF_SW_MMDD=!CONF_SW_MMDD;
      kmode=K_DATE_DISP;
```

```
      break;
   case K_SET_MONTH:
      flash_01 = !flash_01;
      if (! flash_01)
      {
         if (getkeypress(S2))
         {
            ds_month_incr();
         }
         if (getkeypress(S1))
         {
            flash_01 = 0; kmode = CONF_SW_MMDD?K_DATE_DISP:K_SET_DAY;
         }
      }
      break;
   case K_SET_DAY:
      flash_23 = !flash_23;
      if (! flash_23)
      {
         if (getkeypress(S2))
         {
            ds_day_incr();
         }
         if (getkeypress(S1))
         {
            flash_23 = 0; kmode = CONF_SW_MMDD?K_SET_MONTH:K_DATE_DISP;
         }
      }
      break;

   case K_WEEKDAY_DISP:
      dmode=M_WEEKDAY_DISP;
      if (getkeypress(S1)) ds_weekday_incr();
      if (getkeypress(S2)) kmode = K_NORMAL;
      break;
   case K_DEBUG:
      dmode=M_DEBUG;
      if (count>100) kmode = K_NORMAL;
      if (S1_PRESSED||S2_PRESSED) count=0;
      break;
   case K_SEC_DISP:
      dmode=M_SEC_DISP;
      if (getkeypress(S1) || (count>100)) { kmode = K_NORMAL; }
      if (getkeypress(S2)) { ds_sec_zero(); }
      break;
   case K_WAIT_S1:
      count=0;
      if (!S1_PRESSED)
      {
         if (S1_LONG) {S1_LONG=0; kmode=lmode;}
         else  {kmode=smode;}
      }
      break;
   case K_WAIT_S2:
      count=0;
      if (!S2_PRESSED)
      {
         if (S2_LONG) {S2_LONG=0; kmode=lmode;}
         else  {kmode=smode;}
      }
      break;
   case K_NORMAL:
   default:
      flash_01 = 0;
      flash_23 = 0;
      dmode=M_NORMAL;
      if (S1_PRESSED)
      {
         kmode = K_WAIT_S1; lmode=K_SET_HOUR; smode=K_SEC_DISP;
      }
      //if (S2_PRESSED)
      //{
      //   kmode = K_WAIT_S2; lmode=K_DEBUG;smode=K_TEMP_DISP;
      //}
      if (S2_PRESSED) { kmode = K_TEMP_DISP; }
   };
   // Выполнение дерева отображения:
   clearTmpDisplay();
   switch (dmode)
   {
```

```
case M_NORMAL:
   if (flash_01)
   {
      dotdisplay(1,display_colon);
   }
   else
   {
      if (!H12_24)
      {
         // десятки часов
         filldisplay(0,
                     (rtc_table[DS_ADDR_HOUR]>>4)&(DS_MASK_HOUR24_TENS>>4),
                     0);
      }
      else
      {
         // десятки часов для случая включенного режима AMPM,
         // когда '1' только когда включено H12_TH
         if (H12_TH)
            filldisplay(0, 1, 0);
      }
      filldisplay(1,
                  rtc_table[DS_ADDR_HOUR]&DS_MASK_HOUR_UNITS,
                  display_colon);
   }
   if (flash_23)
   {
      // точка в разряде 3, если режим AMPM и PM=1
      dotdisplay(2,display_colon);
      dotdisplay(3,H12_24&H12_PM);
   }
   else
   {
      // десятки минут
      filldisplay(2,
                  (rtc_table[DS_ADDR_MINUTES]>>4)&(DS_MASK_MINUTES_TENS>>4),
                  display_colon);
      // минуты
      filldisplay(3,
                  rtc_table[DS_ADDR_MINUTES]&DS_MASK_MINUTES_UNITS,
                  H12_24 & H12_PM);
   }
   break;
case M_SET_HOUR_12_24:
   if (!H12_24)
   {
      filldisplay(1, 2, 0);
      filldisplay(2, 4, 0);
   }
   else
   {
      filldisplay(1, 1, 0);
      filldisplay( 2, 2, 0);
   }
   filldisplay(3, LED_h, 0);
   break;
case M_SEC_DISP:
   dotdisplay(0,display_colon);
   dotdisplay(1,display_colon);
   filldisplay(2,(rtc_table[DS_ADDR_SECONDS]>>4)&(DS_MASK_SECONDS_TENS>>4),0);
   filldisplay(3,rtc_table[DS_ADDR_SECONDS]&DS_MASK_SECONDS_UNITS,0);
   break;
case M_DATE_DISP:
   if (flash_01)
   {
      dotdisplay(1,1);
   }
   else
   {
      if (!CONF_SW_MMDD)
      {
         // десятки месяца (операция &MASK_TENS бесполезна, поскольку
         // старшие биты читаются как '0')
         filldisplay(0, rtc_table[DS_ADDR_MONTH]>>4, 0);
         filldisplay(1, rtc_table[DS_ADDR_MONTH]&DS_MASK_MONTH_UNITS, 1);
      }
      else
      {
         // десятки месяца (операция &MASK_TENS бесполезна, поскольку
         // старшие биты читаются как '0')
```

```
                filldisplay(2, rtc_table[DS_ADDR_MONTH]>>4, 0);
                filldisplay( 3, rtc_table[DS_ADDR_MONTH]&DS_MASK_MONTH_UNITS, 0);
             }
          }
          if (!flash_23)
          {
             if (!CONF_SW_MMDD)
             {
                // десятки дней (операция &MASK_TENS бесполезна)
                filldisplay(2, rtc_table[DS_ADDR_DAY]>>4, 0);
                // дни
                filldisplay(3, rtc_table[DS_ADDR_DAY]&DS_MASK_DAY_UNITS, 0);
             }
             else
             {
                // десятки дней (операция &MASK_TENS бесполезна)
                filldisplay( 0, rtc_table[DS_ADDR_DAY]>>4, 0);
                // дни
                filldisplay( 1, rtc_table[DS_ADDR_DAY]&DS_MASK_DAY_UNITS, 1);
             }
          }
          break;
       case M_WEEKDAY_DISP:
          filldisplay(1, LED_DASH, 0);
          // день недели (операция &MASK_UNITS бесполезна, потому что
          // старшие биты читаются как '0')
          filldisplay(2, rtc_table[DS_ADDR_WEEKDAY], 0);
          filldisplay(3, LED_DASH, 0);
          break;
       case M_TEMP_DISP:
          filldisplay(0, ds_int2bcd_tens(temp), 0);
          filldisplay(1, ds_int2bcd_ones(temp), 0);
          filldisplay(2, CONF_C_F ? LED_f : LED_c, 1);
          // -- температура определена как uint16, она не может быть < 0
          // if (temp < 0) filldisplay( 3, LED_DASH, 0);
          break;
       case M_DEBUG:
          filldisplay( 0, switchcount[0]>>4, S1_LONG);
          filldisplay( 1, switchcount[0]&15, S1_PRESSED);
          filldisplay( 2, switchcount[1]>>4, S2_LONG);
          filldisplay( 3, switchcount[1]&15, S2_PRESSED);
          break;
       }
       __critical { updateTmpDisplay(); }
       // сохранение конфигурации ram
       ds_ram_config_write();

       if (S1_PRESSED || S2_PRESSED && ! (S1_LONG || S2_LONG))
       {
          // попытка ослабить чрезмерно длинное нажатие кнопок
          _delay_ms(100);
       }
       // сброс при долгом нажатии кнопок, когда кнопка отпущена
       if (! S1_PRESSED && S1_LONG)
       {
          S1_LONG = 0;
       }
       if (! S2_PRESSED && S2_LONG)
       {
          S2_LONG = 0;
       }
       count++;
       WDT_CLEAR();
    }
}
/* ------------------------------------------------------------------------ */
```

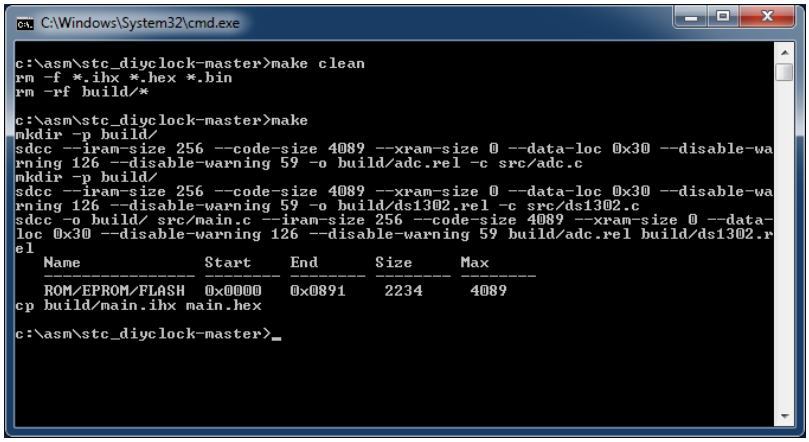
Как компилировать исходный код [2] под Windows, процесс по шагам:

1. Установите SDCC в папку, которую инсталлятор предлагает по умолчанию (C:\Program Files). Компилятор SDCC можно скачать из архива 170301stc15w404as-clock.zip по ссылке [3] из статьи "Цифровые часы на микроконтроллере STC15W404AS" (см. в архиве папку compiler, там 2 запускаемых инсталлятора, один для 32-битной Windows, другой для 64-битной Windows).

2. Распакуйте архив stc_diyclock-master.zip, он находится в папке srccode архива 170301stc15w404as-clock.zip (см. пункт 1), или можно скачать с сайта автора.

3. Перейдите в корневую папку распакованного проекта stc_diyclock-master (где находится файл Makefile), откройте интерпретатор командной строки cmd.exe, и выполните в этой папке 2 команды, одну за другой:

```
make clean
make
```



После успешного выполнения этих команд в корневой папке проекта окажется файл прошивки main.hex.

[**UPD180317**]

Внимание, исходный код по ссылке [3] использует не только память FLASH, но и EEPROM, поэтому прошивки просто кода main.hex недостаточно, нужно прошить еще и файл eeprom.hex. Если eeprom.hex не прошить, то цифры отображаться не будут, будут только мигать точки.

К сожалению, у меня не получилось получить файл eeprom.hex из-за непонятной проблемы с sed:

```
c:\temp\stc_diyclock-master>make eeprom
sed -ne '/:..1/ { s/1/0/2; p }' main.hex > eeprom.hex
sed: -e expression #2, char 21: Extra characters after command
```

Было лень разбираться, что тут неправильно в командной строке sed, поэтому я поступил проще: по ссылке [6] нашел готовый исходный код и готовую прошику, которая не использует EEPROM. Кстати, там же можно найти и готовый файл eeprom.hex.