Why I am in no hurry to continue the conversation about the timing of the execution of instructions by the STM8 computing core

Blog named Deer

A long time ago I wanted to start a conversation about the subject. At that time, my oscilloscope was being repaired, and I was armed only with a frequency meter. With the return of the oscilloscope, I thought that everything would become clearer. As you may have guessed, it did not! :)

So, our task has not changed yet: to pull the pin with the LED and count how many cycles it takes. It is clear that we can only see the execution phase, and the processes of filling the input buffer of the pipeline are hidden from us.The STM8 data sheets even provide an explanation on this matter, but it did not provide a complete understanding.

The experimental controller will be **the STM8L152C6T6** , soldered on the STM8L Discovery board, clocked by an internal oscillator at four megahertz.

Listing 1 The commands for writing zero and one to the corresponding digit follow one after another. Nothing more superfluous. At the end of the cycle, there is a JPF
command - an unconditional transition

```
stm8/
        #include "stm8l152c6.inc"
        #include "mapping.inc"

        segment 'rom'
stack_start.w EQU $stack_segment_start
stack_end.w EQU $stack_segment_end
        ; initialize SP
        ldw X,#stack_end
        ldw SP,X

;Переключаемся на тактирование 4 МГц
        mov CLK_CKDIVR,#2

        bset PC_DDR,#7
        bset PC_CR1,#7
```

Live

Comments Publications

penzet → Sprint Layout in OS X 18 → Software for electronics engineer

Vga → EmBitz 6 → Software for electronics engineer

Vga → Rail-to-rail: ideal OU or a clever marketing ploy? 1 → Theory, measurements and calculations

Flint → A little more about 1-wire + UART 56 → Hardware connection to the computer.

penzet → Cross-platform terminal - SerIO 3.x 25 → Software for electronics engineer

Gornist → PIP regulator 2 → Algorithms and software solutions

whoim → CC1101, Treatise on Tracing 89 → Blog im. khomin

OlegG → Clock on Bluetooth LE module 8 → Cypress PSoC

Technicum505SU → Nuances of PWM control of a DC motor by a microcontroller 3 → Theory, measurements and calculations

sunjob → DDS synthesizer AD9833 88 → Blog named after grand1987

DIHALT → W801, LCD screen and fly in the ointment 2 → Detail

nictrace → New Arduino-compatible board. 20 → FPGA

sunjob → Connecting the ARM GCC compiler to CLion 1 → Software for electronics engineers

Vga → CRC32: on STM32 as on PC or on PC as on STM32. 58 → STM32

dmitrij999 → Capturing images from a USB camera using STM32 6 → STM32

Gilaks → Expanding the capabilities of a simple MC up to an ADC on 2 or 1 pin. 8 → Theory, measurements and calculations

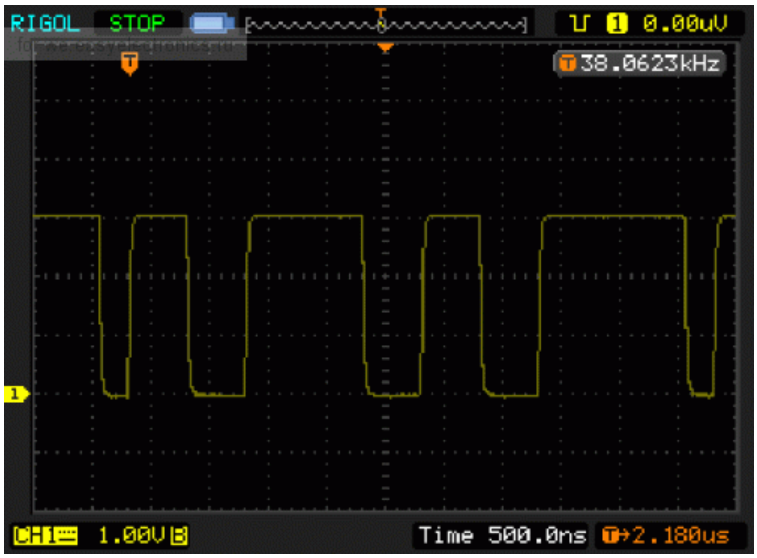
sva_omsk → Lithium ECAD - Russian PCB CAD 40 → Software for electronics engineer

x893 → W801 - budget controller with Wi-Fi 4 → Details

podkassetnik → Changing the standard instrument cluster lighting of Logan-like cars 5 → Automotive electronics

```
        bset PC_CR2,#7          ;Output, Push-Pull, Fast Mode
        nop

loop:
        MOV PC_ODR,#0
        MOV PC_ODR,#128         ;1 макт
        MOV PC_ODR,#0           ;2 макта
        MOV PC_ODR,#128         ;2 макта
        MOV PC_ODR,#128         ;2 (?) макта
        MOV PC_ODR,#0           ;2 макта
        MOV PC_ODR,#128         ;2 макта
        MOV PC_ODR,#0           ;2 макта
        MOV PC_ODR,#128         ;2 (?) макта
        jpf loop                ;3 (?) такта

        end
```

And the result of execution:



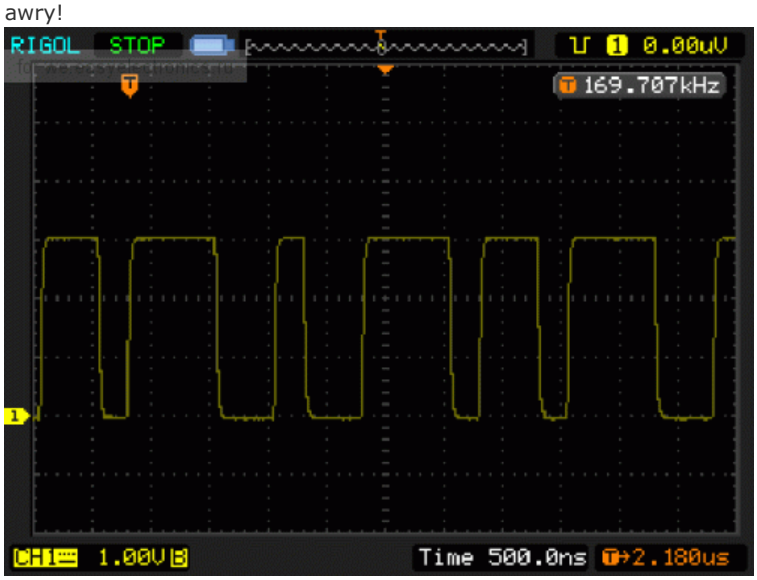
Already interesting! But the most interesting is yet to come!

Listing 2

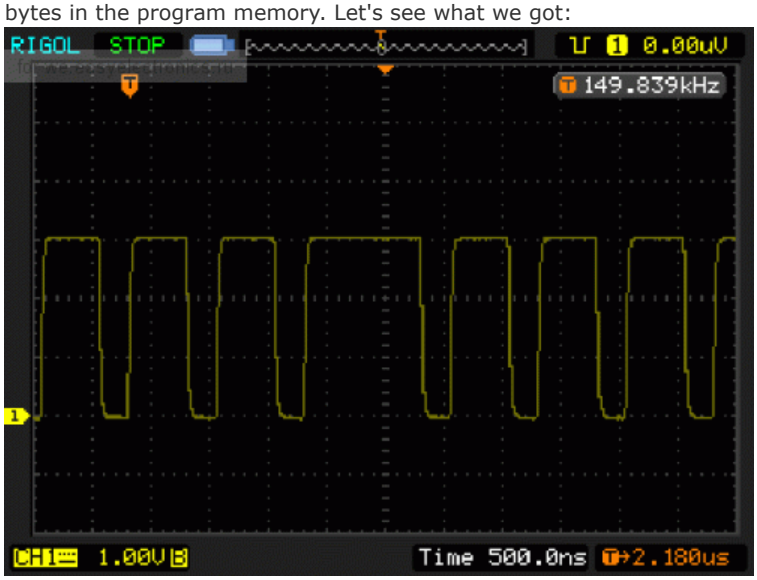
The listing differs only in that **the NOP** was removed before the start and in the middle two consecutive installations to "1" were reduced to one

```
stm8/
;...
;Здесь всё то же самое
;...
        bset PC_DDR,#7
        bset PC_CR1,#7
        bset PC_CR2,#7          ;Output, Push-Pull, Fast Mode

        ;nop    ;Убрали NOP

loop:
        MOV     PC_ODR,#0       ;2(?) макта
        MOV     PC_ODR,#128     ;1 макт
        MOV     PC_ODR,#0       ;2 макта
        MOV     PC_ODR,#128     ;1 макт
        ;MOV    PC_ODR,#128
        MOV     PC_ODR,#0       ;2 макта
        MOV     PC_ODR,#128     ;1 макт
        MOV     PC_ODR,#0       ;2 макта
        MOV     PC_ODR,#128     ;1 макт
        jpf     loop            ;3 (?) такта
```

Execution result. As we can see, the first "column" unexpectedly changed its width (we fully expected this from the second one, since the command was duplicated there before). And it would seem that only one NOP was removed, which stood long before this place and any consequences should have become unnoticeable N iterations of the loop back! But here everything is crooked and awry!



Next. Let's finish the program with **NOPs** before the loop so that the first instruction of the loop is located at the nearest address multiple of four (according to the bit depth of the instruction register at the pipeline input). We do not change anything else. Each instruction from the loop body takes up 4 bytes in the program memory. Let's see what we got:

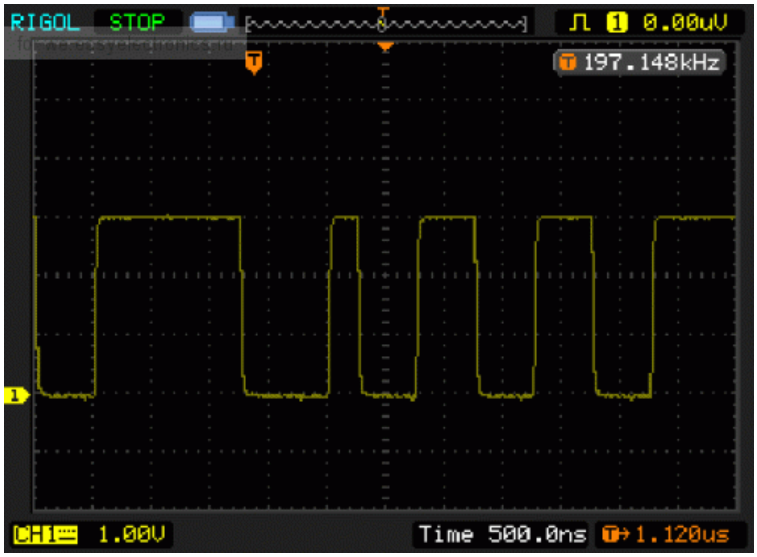


Listing 3

Let's try to expand the pads with **NOPs** when the voltage on the pin of interest is zero

```
;nop

loop:
        MOV     PC_ODR,#0
        nop
        MOV     PC_ODR,#128
        MOV     PC_ODR,#0
        nop
```

```
        MOV     PC_ODR,#128
        MOV     PC_ODR,#0
        nop
        MOV     PC_ODR,#128
        MOV     PC_ODR,#0
        nop
        MOV     PC_ODR,#128
        jpf     loop
```

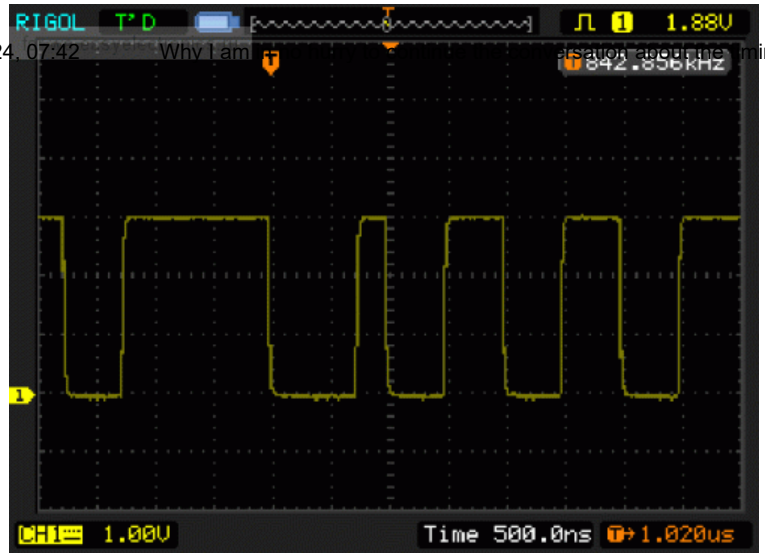


We see something "ugly" here - the first pad has a length of three cycles, and the rest - two. Let's try to remove **the NOP** after the first **MOV PC_ODR,#0**

Listing 3a

```
        ;nop

loop:
        MOV     PC_ODR,#0
        MOV     PC_ODR,#128
        MOV     PC_ODR,#0
        nop
        MOV     PC_ODR,#128
        MOV     PC_ODR,#0
        nop
        MOV     PC_ODR,#128
        MOV     PC_ODR,#0
        nop
        MOV     PC_ODR,#128
        jpf     loop
```

Result:



The differences are minimal!

So now I don't even know how possible it is to use STM8 to send pulses with a duration measured in fractions and units of microseconds! If the slightest change in another place of the program (okay, in a high-level language, but in assembler!) entails such consequences in other places of the program!

STM8 , assembler

+1 January 14, 2012, 4:24 PM **Deer**

Comments (35)

RSS Collapse / Expand

Good review.
Hmm, they seem to have the same problem as with ARMs.
I wonder how to stick software protocols on them??
On AVR and PIC it's a piece of cake. +1

a9d
January 14, 2012, 4:47 PM

It seems to me that on STM8, if you really need to implement a software implementation of the protocol, you should implement only synchronous protocols so that all slave devices wait for the required clock signal edge. Because here, uncertainties of 1-2 clocks will be almost guaranteed. And catching them with an oscilloscope and trying to correct them with NOPs or something else is almost useless work, imho.

With longer pulses and time delays, I believe, timers will help work more stably! 0

Deer
January 14, 2012, 4:53 PM

In general, how many software timing protocols are needed? Dallas, iButton, what else? Can't we do without them? What is needed is widespread and present in the hardware, or there are adapters for a more standard interface. 0

angel5a
January 16, 2012, 4:26 PM

The most popular is 1-wire.
Also, small delays are important when working with some sensors. And here I see that the stmok has small problems. +1

a9d
January 16, 2012, 6:26 PM

The stmok has some minor problems

Theoretically...
In practice, it borders on the wrong choice of stone for the speed of the noddyga in certain applications... "Each processor - for its own task"...)))

0

HIMERA
January 16, 2012, 19:04

Here are most likely the peculiarities of either the computing core or the specific implementation of STM. I am absolutely not familiar with the STM32 core, except that I heard that STM32 and STM8 have extremely similar peripherals => extremely similar behavior can be expected.

Maybe someone can do similar measurements on STM32 and on ARM from another manufacturer? It is desirable, of course, to write the "measuring" cycle in assembler "for the purity of the experiment"

0

Deer
January 16, 2012, 21:27

It doesn't depend on the periphery but on the core. In ARMs it is much worse.
There it is almost impossible to count the number of cycles before compilation.

0

a9d
January 16, 2012, 21:36

Well, at least you can do several times in a row the same conversions to zero and to one and estimate how different the width of these "columns" will be. Then experiment with adding commands both in the cycle and before and after it...

0

Deer
January 16, 2012, 21:39

And what's the point if the result is known in advance???))))
Once again... everything is according to the task... if you need to take into account each cycle, then it's better to take PIC24... you can calculate each cycle there, but... even there PLL will give jitter, and there will be an application where this jitter is unacceptable...
To be honest... I don't understand this panic about the +- cycle... there aren't that many such applications... and if there's no other way - the wrong choice of stone at the design stage... the rest is from the evil one...)))))
"Forewarned is forearmed"...

0

HIMERA
January 16, 2012, 10:00 PM

"Can't we do without them?" The element base is now quite wide. The same thermometers can be taken with the I2C interface, which is implemented in hardware and does not require strict timings (interface) and kernel time (which is a much bigger plus).
If you want to kick your legs - use AVR to your heart's content. Only then you will hardly be able to calmly use C/C++, assembler rates will be mandatory - which is ugly, incomprehensible and not portable.
So I agree. Stmok has some small problems.

0

angel5a
January 17, 2012, 09:55

Yes, it seems like this topic has already been sucked out and more than once.
Due to the three-level conveyor, such things are obtained.

+1

ZiB
January 15, 2012, 05:44

0

Add another link to the doc (the pipeline operation is described):
zblog.ru/wp-content/plugins/download-monitor/download.php?id=48
p.s. I couldn't find the original link

ZiB
January 15, 2012, 06:18

I asked yesterday in the forum

0

A simple program that also twitches the leg:

```
                    ; C Compiler for STM8 (COSMIC Software)
                    ; Parser V4.9.10 - 10 Feb 2011
                    ; Generator (Limited) V4.3.6 - 15 Feb 2011
                    ; Optimizer V4.3.5 - 15 Feb 2011
                    ; 35 void main(void)
                    ; 36 {
                          scross  off
0000                _main:
                    ; 37   CLK->CKDIVR = 0; // HSI 16Mhz
0000 725f50c6             clr     20678
                    ; 38   PORT->DDR |= (1 << PIN); // Output
0004 72185011             bset    20497,#4
                    ; 39   PORT->CR1 |= (1 << PIN); // Push-pull
0008 72185012             bset    20498,#4
                    ; 40   PORT->CR2 |= (1 << PIN); // Output speed u
000c 72185013             bset    20499,#4
0010                L3:
                    ; 44           PORT->ODR |= (1 << PIN);
0010 7218500f             bset    20495,#4
                    ; 45           PORT->ODR &= (uint8_t)~(1 << PIN);
0014 7219500f             bres    20495,#4
0018 20f6                 jra     L3
                          xdef    _main
                          end
```

Regardless of the presence of the line PORT->CR2 |= (1 << PIN); the frequency meter in the multimeter shows 2.6 MHz (in theory, without it it should be less, with it - more).

Is this the same effect?

artvolk
January 15, 2012, 11:19

" Output speed up to 10 MHz" affects the duration of transient processes (signal fronts), but not the operating speed.
I think we can attribute the low switching frequency to the "specified" effect :)
That is, it is logical that depending on the sequence and length of commands, the number of cycles for executing commands will be different. Conveyor mother of it.

0

ZiB
January 15, 2012, 15:15

I ran it in the simulator - the clock cycles that System clock ___ cpu ticks shows coincide with the number of clock cycles per command. The frequency is too low, this whole piece is executed in 5 clock cycles:

0

```
                    ; 44           PORT->ODR |= (1 << PIN);
0010 7218500f             bset    20495,#4
                    ; 45           PORT->ODR &= (uint8_t)~(1
0014 7219500f             bres    20495,#4
0018 20f6                 jra     L3
```

artvolk

January 16, 2012, 00:00

In my case, the problem was different - a fundamental misunderstanding of what would be 1 period of the signal in this case, I assembled a demo on AVR, then found an explanation for AVR here: www.arduino.cc/cgi-bin/yabb2/YaBB.pl?num=1230286016/18#18
For STM8, the principle is the same.

0

artvolk
January 16, 2012, 14:45

Проверил аналогичный код в зависимости от предыдущих команд перед циклом (от выравнивания) получал от пяти до шести тактов, т.е. при 16 МГц
3,2 МГц и 2,66 МГц.

0

ZiB
16 января 2012, 16:14

Да, так и должно быть :) Мне привидилось, что должно быть по-другому :)

0

artvolk
16 января 2012, 17:36

По теме статьи:
В комплекте с STVD идёт документ PM0044 STM8 CPU programming manual старой ревизии. В новом документе:
www.st.com/internet/com/TECHNICAL_RESOURCES/TECHNICAL_LITERATURE/PROGRAMMING_MAI
(ревизия 3)

Есть раздел Pipelined execution, там есть несколько примеров с растактовкой + в разделе 5.4 есть импирическая формула для расчёта количества тактов…

Although the decode and/or execute stage of some instructions may take a different number
of cycles, a simplified convention providing a good match with reality, has been used in this
section:
• The decode stage of each instruction takes one cycle only
• The execution stage takes a number of cycles equal to

Cy = DecCy + ExeCy – 1

Where
Cy is the number of execution cycles. In case of decode and execute cycles, It corresponds to the minimum number of cycles needed by the instruction itself, and does not take into account the impact of the instruction sequence.
DecCy is the exact number of decode cycles.
ExeCy is the exact number of execute cycles.

0

artvolk
15 января 2012, 23:44

Подозреваю, что из-за этого единственный надежный способ генирировать задержки — таймером, например TIM4. В библиотеке для STM8L есть пример, а для STM8S — нет :)

0

artvolk
15 января 2012, 23:48

то что примера нет, не значит что нет таймера :) а если нет именно TIM4, то есть TIM1, TIM2,…

0

angel5a
16 января 2012, 16:34

Видел. То ли лыжи не едут, то ли я очень глупый. Но до конца не въехал.

Потом решил, что занимаюсь фигнёй. Потом подумал, что кой-чего намеренного есть. Что рано или поздно кто-нибудь ещё задумается на ту же тему. Поэтому в блог для всякой фигни таки выложил свои наблюдения. Чтобы следующим

0

облегчить первые шаги забега по граблям и, может, слегка сориентировать в направлении.)

А для более-менее больших выдержек уже можно мутить хотя бы на таймерах различные выдержки.

Deer
16 января 2012, 00:19

0

В целом в рассмотренном примере разнича во времени выполнения команд идет из-за кеша, я не наблюдаю команд с разным временем выполнения. Тот же единственный переход безусловный — выполнение так же за определённое время. (То ли дело условные переходы, предсказание и сброс конвеера).
У меня просьба к автору, как к владельцу осциллографа (и возможно желания эксперементировать). Сможете ли вы провести подобное исследование с выполнением кода из оперативки?
— Шина там 8 бит, следовательно время выборки не зависит от положения инструкции (на границе 4байт или нет), зависит от размера инструкции.
— Время декодирования так же величина постоянная (на сколько мне известно в стм декодер не использует повторно декодированные инструкции, не интел всё же).
— Время исполнения так же постоянное (по доке).
— Сбросов конвеера так же нет (условных переходов нет).
Все шансы на постоянное время выполнения кода. Это возможно будет интересно любителям ногодрыгства, мне это только проверить понимание архетиктуры. Сам такой эксперимент провести не смогу, не имею осциллографа, но и на проведение не настаиваю.

angel5a
17 января 2012, 10:07

0

ОК, померим через несколько дней…

Deer
17 января 2012, 16:38

0

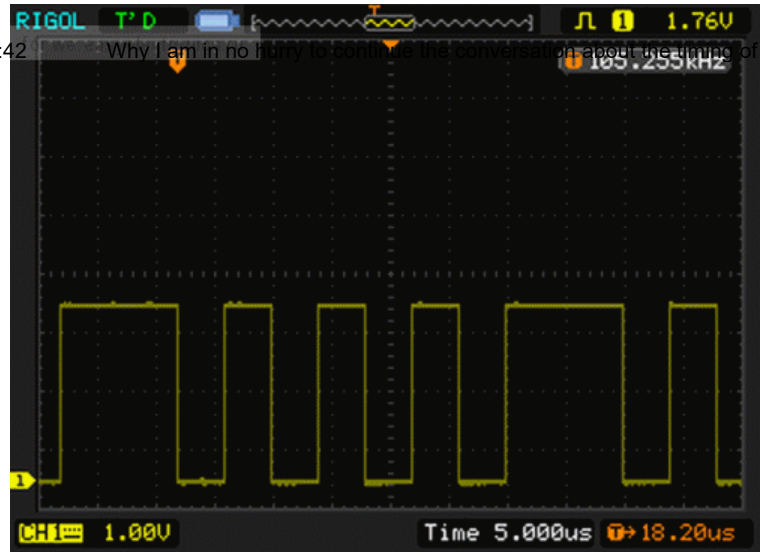
Программа написана и из SRAM даже выполняется! Завтра сдам экзамен и для Вас померю, коллега! :)

Deer
19 января 2012, 21:33

0

Переселил в RAM цикл (см. листинг), сделал тактирование 4 МГц

```
Main:
        MOV PC_ODR,#0
        MOV PC_ODR,#128
        MOV PC_ODR,#0
        MOV PC_ODR,#128
        MOV PC_ODR,#0
        MOV PC_ODR,#128
        MOV PC_ODR,#0
        MOV PC_ODR,#128
        jra Main
```



Идеально ровно, но безобразно медленно!

Deer
22 января 2012, 17:17

Гм, порядка 16 тактов на команду? Да, довольно невкусно. 0

Vga
22 января 2012, 18:27

Да, из RAM команда MOV (long),(imm) исполняется ровно 16 тактов. 0
Хм… А ведь при таком быстродействии таким образом 1-wire реализовывать — в самый раз! :)

Тем более, что в RAM можно перезаписывать кусок кода, критичный ко времени исполнения, исполнять его, потом на то же место поселять любой другой кусок кода!

Deer
22 января 2012, 18:41

Примного благодарен. 0
По скорости исполнения в даташите сразу отмечено, что шина используется для кода и для данных одна, да к тому же и 8 бит вместо 32 (флешевских). Да, USB софтверный не реализуешь, да и думаю это мы переживем, можно взять арм с хардверным — разница в цене не столь высока, исли устройству нужен USB.

А Вместо MOV наверно лучше будет использовать битовые команды в данном случае (bset или как-то так). они генерятся если делать через битовые поля структур для портов IO. Там команда 4 байта, но зато работа только с указанным битом и операция атомарная.

Еще раз благодарю за проверку.

angel5a
23 января 2012, 11:55

Well, after all, according to that datasheet, the command execution should not stretch out three times, but for several cycles. After all, 4 bytes (and the **BSET** , **BRES** , **BCPL** , and **MOV** instructions occupy 4 bytes in memory) are loaded from flash in 1 cycle, and from RAM - in 4. And the decoding and execution time, in theory, should remain per cycle. A total of 6 cycles. And here all 16 are collected... 0

Deer
January 23, 2012, 19:43

By the way, if my memory serves me right, these MCs have flash and accelerator settings, 0

which, as I understand it, affect the clocking. Maybe if I tweak their settings, it will be better?

Vga
January 23, 2012, 20:26

It won't happen... Take it as inevitable... 0

HIMERA
January 23, 2012, 20:36

Maybe, although who knows. +1

I don't see any point in poking around too much. I've come to the conclusion that if I need to generate rectangular signals with a clearly defined geometry, I won't be doing it on STM8. And that the same commands can be executed under the same conditions for different periods of time.

Now I want to make one Project on this little stone and do other things. And there are plenty of them! Some of them will definitely be accompanied by more interesting articles than staring at oscillograms and drawing conclusions from them! :)

Deer
January 23, 2012, 21:05

And still, run a couple of experiments with different flash settings. It's interesting. 0

Vga
January 23, 2012, 21:19

Okay, we'll do it! But not in the near future... [I'll try my best within a week!] 0

Deer
January 23, 2012, 21:31

Only registered and authorized users can leave comments.