РЕФЕРАТ

РПЗ страниц, рисунок, таблиц, источников, приложений

МИКРОКОНТРОЛЛЕР, ХЭШИРОВАНИЕ, ХЭШ-ФУНКЦИЯ, АЛГОРИТМЫ, SHA256, CRC16/CCITT-FALSE, MD5

Объектом разработки является МК-система для хэширования данных.

Цель работы - создание системы хэширования с возможностью выбора алгоритма хэширования, модель устройства и разработка необходимой документации.

Поставленная цель достигается при помощи Proteus 8.13 и STM32CubeIDE.

В процессе работы над курсовым проектом решаются задачи: выбор МК и драйвера обмена данных, создание функциональной и принципиальной схем системы, расчет потребляемой мощности устройства, разработка алгоритма управления и соответствующей программы МК, а также написание сопутствующей документации.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В данной курсовой работе производится разработка МК-системы хэширования данных.

В процессе выполнения работы проведён анализ технического задания, создана концепция устройства, разработаны электрические схемы, построен алгоритм и управляющая программа для МК, выполнено интерактивное моделирование устройства.

Система состоит из МК, виртуального терминала, необходимого для ввода данных для последующего хэширования, ЖК-дисплея для вывода результатов выполнения операций, ошибок, возникших во время выполнения программы, 3 кнопок, с помощью которых производится управление выбором алгоритма хэширования, 1 кнопки, с помощью которой определяется, осуществляется ввод с телефона или ПЭВМ, преобразователя UART-USB, коннектора USB.

Актуальность разрабатываемой модели хэширования данных состоит в

1 Конструкторская часть

1.1. Анализ требований и принцип работы системы

Исходя из требований, изложенных в техническом задании, необходимо осуществить разработку системы, хэширующих данные, вводимые оператором, с возможностью выбора алгоритма хэширования и выбором места осуществления ввода - ПЭВМ или телефон.

Ну и что еще сюда

1.2. Проектирование функциональной схемы

В этом разделе приведено функциональное описание работы системы и проектирование функциональной схемы.

1.2.1. Микроконтроллер STM32F103C8T6

1.2.1.1 Используемые элементы

1.2.1.2 Распределение портов

1.2.1.3 Организация памяти

1.2.2 ЖК-дисплей

Для вывода отладочной информации и информации об ошибках был выбран ЖК-дисплей LM016L размером 16x2. У этого дисплея установлен контроллер HD44780.

На рисунке изображен LM016L.

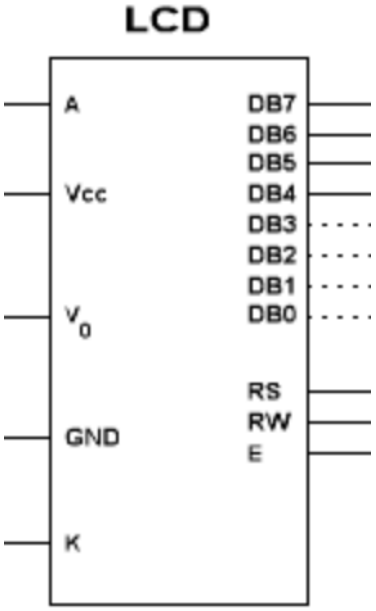

Рисунок - ЖК-дисплей LM016L. (заменить фотку, а то шакальная)

В таблице указаны пины индикатора LM016L.

Таблица - пины индикатора LM016L.

№ пина	Название	Описание
1	VSS	Земля
2	VDD	Питание, +5В
3	V_0	Контрастность дисплея
4	RS	Выбор регистра
5	R/W	L - MPU к LCM, H - LCM к MPU
6	E	Enable
7, 8, 9, 10, 11, 12, 13, 14	DB0..DB7	Биты данных от 0 до 7

15	A	Анод LED
16	K	Катод LED

ЖК-дисплей подключен к микроконтроллеру черезу порты: PB-10 - DB7, PB-11 - DB6, PB-12 - DB5, PB-13 - DB4, PB-14 - E, PB-15 - RS.

1.2.3. Выбор вида ввода

Выбор вида ввода представляет собой кнопку (COMP/PHONE), которая отвечает за то, какой порт будет слушаться при состоянии ввода данных. Кнопка подключена к микроконтроллеру через порт PA-14. Нажатая кнопка означает, что ввод должен производится через устройство ПЭВМ с помощью UART, а если кнопка не нажата, то чтение производится с телефона через USB-порт.

1.2.4. Прием данных от ПЭВМ

Прием данных от ПЭВМ на микроконтроллер осуществляется при помощи модуля UART. В реализации микроконтроллера STM32F103C8T6 имеется 3 интерфейса UART - они одинаковы. На данном микроконтроллере скорость передачи составляет 115200 бод, длина слова составляет 8 бит, включая бит четности и 1 стоповый бит. Передача полнодуплексная. В модуле UART от STM32 существует 7 регистров, которые его контролируют: USART_SR, USART_DR, USART_BRR, USART_CR1, USART_CR2, USART_GTPR и USART_CR3.

Первый из регистров, USART_SR - регистр статуса. На рисунке изображены его биты.

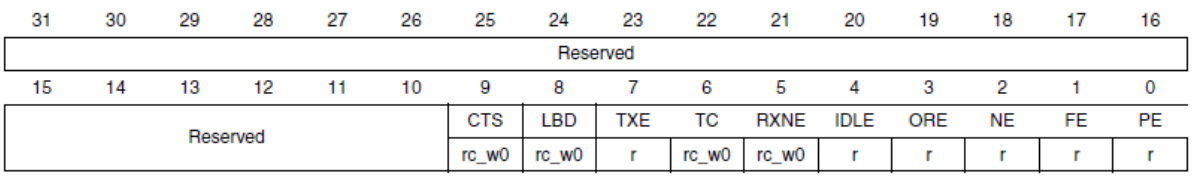

Рисунок - регистр USART_SR

Регистр USART_SR имеет следующие биты:

- TXE. Буферный регистр передатчика пуст. Флаг становится равным 1 после того, как данные перегружаются в регистр сдвига. Флаг устанавливается аппаратно и сбрасывается после записи байта в буферный регистр USART_DR;
- TC. Флаг устанавливается аппаратно и сообщает о том, что передача данных закончена, сдвиговый регистр пуст. Сбрасывается флаг последовательным чтением регистров USART_SR и USART_DR;
- RXNE. Буферный регистр приема не пуст. Флаг сообщает, что в буферном регистре приемника есть данные. Сбрасывается флаг при чтении регистра USART_DR;
- LBD. Выставляется при обнаружении брейка при использовании LIN
- ORE. Ошибка переполнения. Флаг устанавливается в случае, если в приемный буферный регистр поступило новые данные, а предыдущие считаны не были;
- NE. Флаг устанавливается при выделении шума во входном сигнале. Наличие шума определяется как слишком частое переключение входного сигнала;
- FE. Ошибка приема фрейма (кадра). Возникает, когда не был выделен стоп-бит;
- PE. Ошибка паритета. Сигнализирует об ошибке при включенном контроле четности.

На рисунке изображена структура регистра USART_DR.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved							DR[8:0]								
							rw	rw	rw	rw	rw	rw	rw	rw	rw

Рисунок - регистр USART_DR.

USART_DR - регистр данных, используется для передачи и чтения данных из буферных регистров передатчика и приемника. На самом деле состоит из 2 регистров: TDR и RDR - регистры данных передатчика и приемника соответственно.

На рисунке изображена структура регистра USART_BRR.

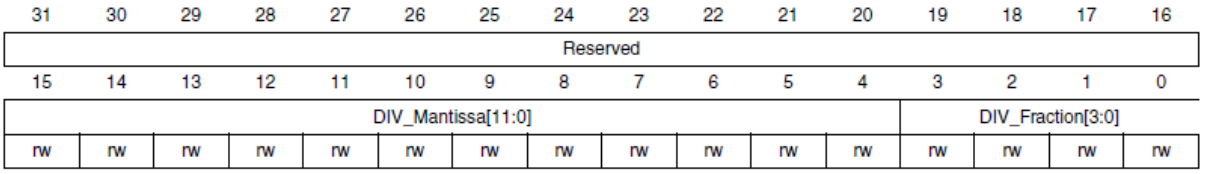

Рисунок - регистр USART_BRR.

USART_BRR - это регистр скорости передачи данных USART. В регистре содержится значение делителя частоты, который определяет скорость обмена данными.

$$BAUD = \frac{F_{ck}}{16 * BRR},$$

где BAUD - скорость обмена данными, в бодах;

F_{ck} - входная частота тактирования UART: PCLK2 для UART1 (используется в приеме данных от ПЭВМ), на шине APB2 и PCLK1 и PCLK3 для UART2 (используется в приеме данных от телефона);

BRR - значение регистра USART_BRR.

На рисунке изображена структура регистра USART_CR1.

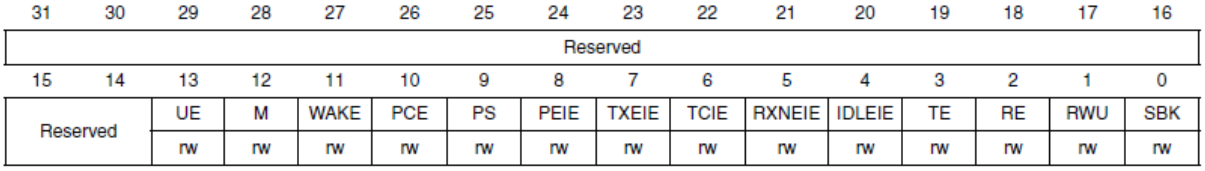

Рисунок - структура USART_CR1.

USART_CR1 - регистр контроля. Его биты:

- UE: включить USART - 0 USART выключен, 1 - включен;
- M: длина слова данных. Этот бит определяет длину передаваемых данных. Устанавливается и очищается программно;

- PCE: разрешить контроль четности. Устанавливается и очищается программно;
- PS: выбор типа контроля четности. Этот бит выбирает вариант контроля четности, если установлен бит PCE. Устанавливается и очищается программно;
- TXEIE: разрешить прерывание при опустошении буфера передатчика. Если установлен в 1, то генерируется запрос прерывания USART при установке бита TXE регистра USART_SR;
- TCIE: разрешить прерывания окончания передачи. Если 1, то генерируется запрос прерывания USART при установке флага TC в регистре USART_SR;
- RXNEIE: разрешить прерывание при появлении данных в регистре приемника. Если 1, то генерируется запрос прерывания USART при установке флага RXNE или ORE в регистре USART_SR;
- TE: включить передатчик USART;
- RE: включить приемник USART.

На рисунке изображена структура регистра USART_CR2.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	LINEN	STOP[1:0]		CLK EN	CPOL	CPHA	LBCL	Res.	LBDIE	LBDL	Res.	ADD[3:0]			
	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw

Рисунок - структура USART_CR2.

USART_CR2 - второй регистр конфигурации. Из важных битов можно выделить STOP - отвечает за количество стоп-бит (00 - 1 стоп-бит, 01 - 0.5 стоп-бит, 10 - 2 стоп-бит, 11 - 1.5 стоп-бит).

Также, у USART еще есть регистры USART_GTPR - регистр, отвечающий за предделитель и USART_CR3 - третий конфигурационный регистр, в котором содержится информация о DMA и SmartCard.

Для приема данных от ПЭВМ используется интерфейс USART1, которому соответствуют порты: PA-9 - TX, PA-10 - RX. Интерфейс находится в асинхронном режиме. Также, на USART1 включены глобальные прерывания.

1.2.5. Прием данных от телефона

Прием данных от телефона осуществляется по USB-коннектору, подключенному к интерфейсу USART2 микроконтроллера STM32F103C8T6 через преобразователь USB-UART FT232RQ(L?).

На рисунках изображен преобразователь USB-UART FT232RL и его УГО.

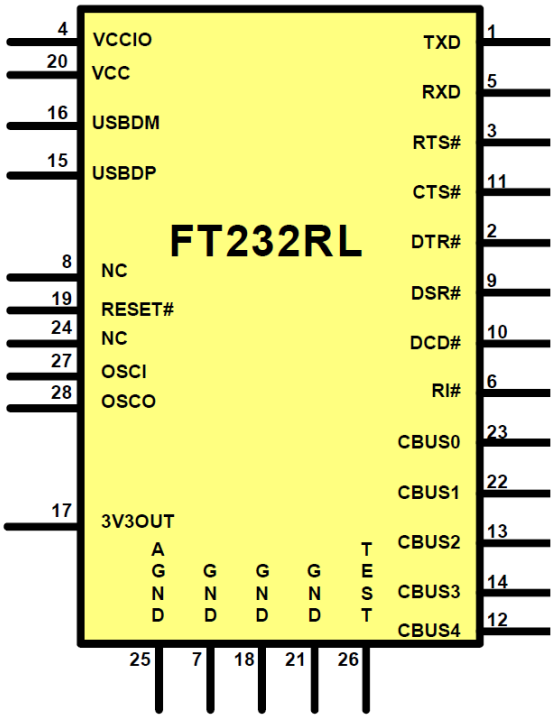

Рисунок - FT232RL.

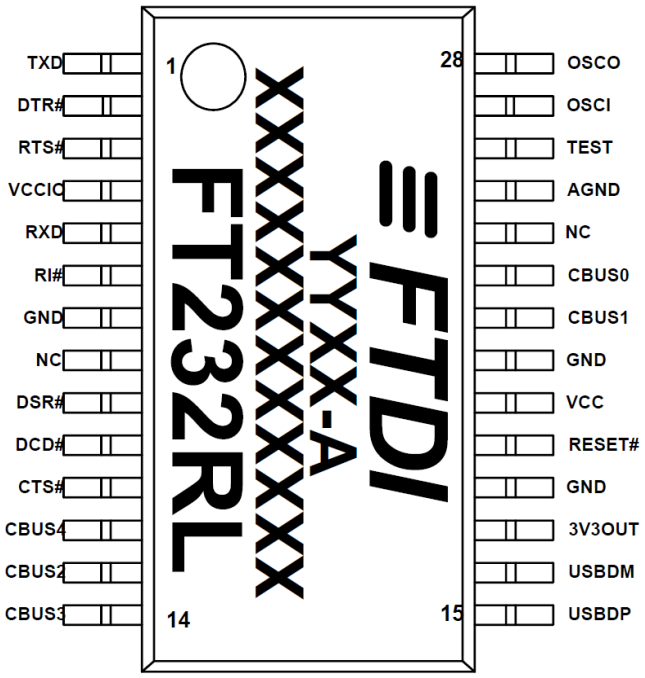

Рисунок - УГО FT232RL.

В таблице дано описание пинов преобразователя FT232RL.

Таблица - пины FT232RL.

№ пина	Название	Тип	Описание
15	USBDP	I/O	положительный сигнал USB данных
16	USBDM	I/O	отрицательный сигнал USB данных
4	VCCIO	PWR	напряжение питание, подключено к источнику питания 5V
7, 18, 21	GND	PWR	земля
17	3V3OUT	O	вывод 3V из интегрированного LDO-регулятора

20	VCC	PWR	напряжение питание, подключено к источнику питания 5V
25	AGND	PWR	Аналог земли устройства для внутреннего счетчика
8, 24	NC	NC	Без внутренней связи
19	RESET	I	RESET
26	TEST	I	Переводит устройство в IC тестовый режим
27	OSCI	I	Ввод осциллографа
28	OSCO	O	Вывод осциллографа
1	TXD	O	Вывод UART
2, 3, 9, 11	DTR#, RTS#, DSR#, CTS#	O	Сигналы рукопожатия
5	RXD	I	Ввод UART
6	RI#	I	Ввод кольцевого управляющего индикатора
10	DCD#	I	Ввод детектора переноса данных
12	CBUS4	O	CBUS только на вывод
13, 14, 22, 23	CBUS2, CBUS3,	I/O	CBUS на ввод/вывод

	CBUS1, CBUS0		

FT232RL подключен к микроконтроллеру STM32F103C8T6 через интерфейс USART2 через порты: PA-2 - TX, PA-3 - RX. Интерфейс находится в асинхронном режиме. Также, на USART1 включены глобальные прерывания.

В качестве USB-коннектора выбран AU-Y1005-R, тип коннектора: USB-A.

1.2.6. Выбор алгоритма хэширования

Выбор используемого алгоритма хэширования осуществляется при помощи 3 кнопок. Первая кнопка (CHOOSE), подключена через порт PA-1, выбирает текущий алгоритм в качестве исполняемого. Вторая кнопка (NEXT), подключенная по порту PA-2, увеличивает id текущего алгоритма на 1. id принимает значения от 0 до 2, если при нажатии второй кнопки id станет превышать 2, то он принудительно устанавливается в 0. Третья кнопка (PREV), подключена через порт PA-3, уменьшает id текущего алгоритма на 1. Если при нажатии на третью кнопку id станет меньше 0, то он устанавливается в 2.

1.2.7. Построение функциональной схемы

Схема + водичка

1.3. Проектирование принципиальной схемы

В этом разделе приведено принципиальное описание работы системы и проектирование принципиальной схемы.

1.3.1. Подключение цепи питания

1.3.2. Расчет потребляемой мощности

Здесь наверн схема + мб еще что

1.4. Алгоритмы работы системы

1.4.1. Функция main

Программа начинает работу с функции main. Управление программой осуществляется с помощью состояний, за которое отвечает extern переменная state.

Первым состоянием является нулевое, состояние ожидания ввода данных. Программа ожидает ввод с UART или USB. После того, как были получены входные данные, состояние переключается в “1”, которое означает состояние ожидания выбора алгоритма хэширования. Выбор алгоритма происходит при увеличении или уменьшении переменной id, где 0 соответствует алгоритму SHA, 1 - CRC, 2 - MD. Программа находится в этом состоянии, пока пользователь не совершит действие, отвечающее за выбор алгоритма. После этого программа переходит в состояние “2”, а управление передается подпрограмме переключения между алгоритмами.

Каждое состояние сопровождается отладочным выводом о том, что требуется сделать пользователю и о введенных пользователем данных.

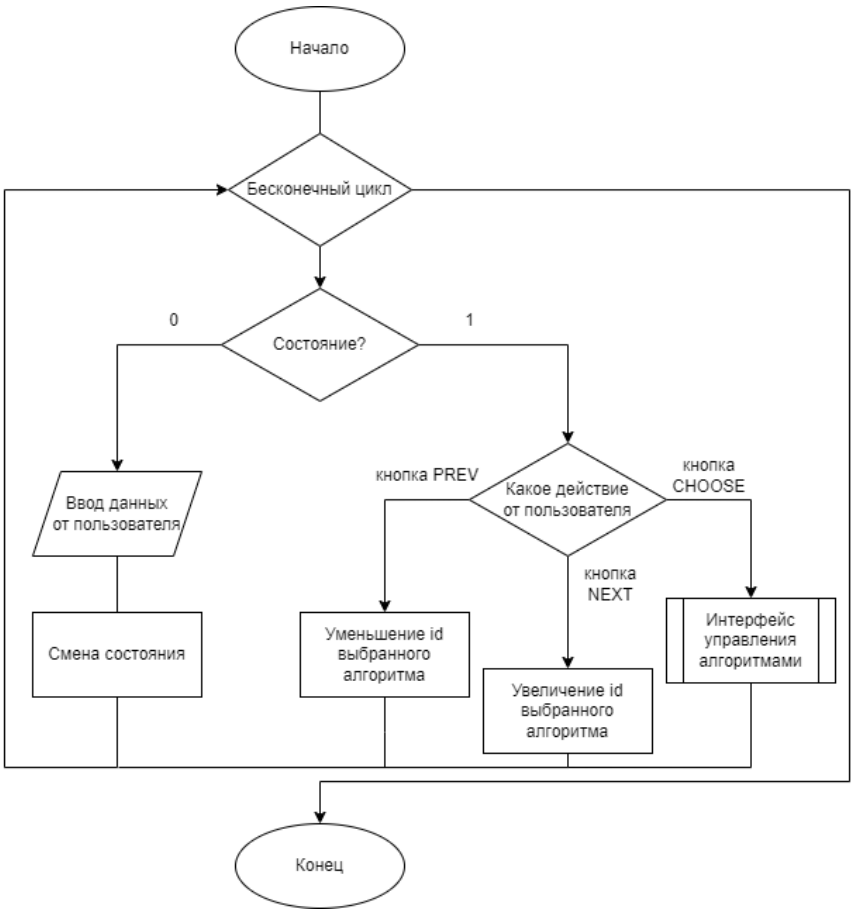

Рисунок - Схема алгоритма main

1.4.2. Подпрограмма переключения между алгоритмами

Когда управление передается подпрограмме переключения между алгоритмами, состояние программы становится “2”, означающее, что программа находится в состоянии выполнения хэширования входных данных.

Сначала пользователю на выход подается информация о выбранном алгоритме хэширования, после чего отображаются стадии выполнения алгоритма: начало хэширования, его завершение, результаты вычислений.

Так как длина хэша в каждом алгоритме разная, то и вывод организован для каждого по-разному, так как не каждый хэш полностью помещается на выбранный ЖК-дисплей размером 16x2.

Для SHA-256 длина хэша составляет 32 символа, поэтому сначала программа выводит на ЖК-дисплей первые 16 символов, 8 в верхней строке, и 8 в нижней, а после ожидания выводит оставшиеся 16. За то, будет текст отображен сверху или снизу ЖК-дисплея отвечает переменная j, равная “0” (сверху) или “1” (снизу), а за отображение переменных по горизонтали отвечает переменная i, принимающая значения от 0 до 16.

Для CRC-16 длина контрольной суммы составляет 4 символа, поэтому осуществляется простой вывод на ЖК-дисплей.

Для MD-5 длина хэша составляет 16 символов, поэтому на верхний ряд осуществляется вывод первых 8 символов хэша, а на нижний - оставшихся 8.

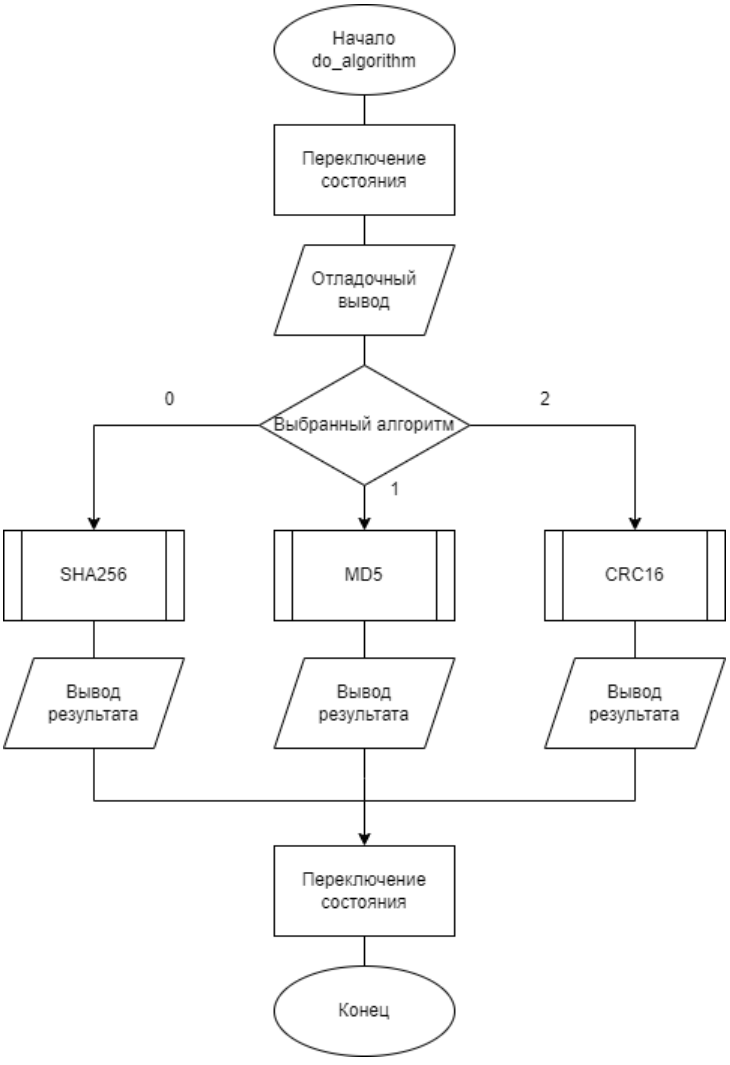

Рисунок - Схема алгоритма do_algorithm

1.4.3. Алгоритм хэширования SHA

SHA (Secure Hash Algorithm, Безопасные алгоритмы хэширования) - это семейство криптографических хэш-функций, преобразующих сообщения произвольной длины в хэш фиксированной длины.

Существуют несколько версий этого семейства:

1. SHA-0 - первая версия алгоритма, разработанная еще в 1993 году;
2. SHA-1 - является исправленной версией SHA-0, 1995 г.;
3. SHA-2 - включает в себя несколько криптографических хэш-функций, различающихся размером ключа (SHA-224, SHA-384, SHA-256, SHA-512, SHA-512/256 и SHA-512/256), более безопасен, чем SHA-1, применяется до сих пор, 2002 г.;
4. SHA-3 - последняя на данный момент версия, ранее называлась Keccak, размер хэшей на выходе равен SHA-2, но использует принципиально новый алгоритм, 2012 г..

Для реализации была выбрана хэш-функция SHA-256, принадлежащая версии SHA-2. (мб расписать, почему именно 2 версия)

Хэш-функции SHA-2 в основе используют метод Меркла-Дамгора, использующийся для построения криптографических хэш-функций, когда из сообщения произвольной длины получается хэш фиксированной длины. Для этого, входное сообщение разбивается на блоки равной длины, каждый из которых проходит через преобразования и на выходе получается хэш фиксированной длины.

На рисунке представлен схематичный принцип работы структуры Меркла-Дамгора. Функция f - односторонняя функция сжатия, принимает два входных блока, и преобразует их в один выходной. Алгоритм начинается с поданного на вход начального значения, или вектора инициализации (IV), а на выходе получаем Hash фиксированного размера.

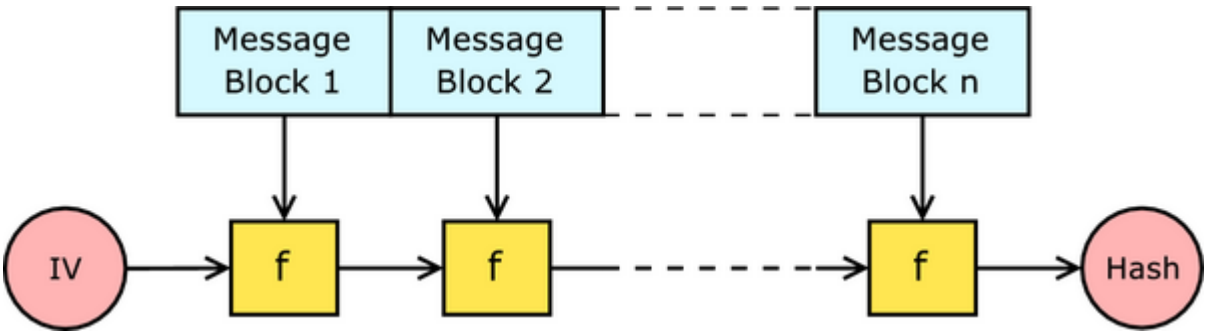

Рисунок - Структура Меркла-Дамгора.

Сам вектор инициализации состоит из 8 констант размером в 32 бита:

- H1 = 0x6A09E667;
- H2 = 0xBB67AE85;
- H3 = 0x3C6EF372;
- H4 = 0xA54FF53A;
- H5 = 0x510E527F;
- H6 = 0x9B05688C;
- H7 = 0x1F83D9AB;
- H8 = 0x5BE0CD19.

Каждая константа высчитывается как первые 32 бита дробной части корней первых 8 простых чисел. Такие числа были выбраны на основе обнаружения “Безопасности Крипто” для того, чтобы добиться высокой степени диффузии и нелинейности в процессе хэширования, так как это позволит уменьшить количество возможных коллизий.

Далее, данные разбиваются на блоки фиксированного размера. В SHA-256 блок составляет 512 бит. Если длины сообщения или его оставшейся части не хватает для заполнения блока, то он заполняется нулями. Так как существует вероятность совпадения полученных хэш-функций, если сообщения различны, но начинаются с одних и тех же символов и оканчиваются нулями, то последний бит заполнителя заменяют на “1”.

Функция сжатия работает по следующему принципу: алгоритм пропускает каждый блок через 64 операции с константами, являющимися первыми 32 битами кубических корней из первых 64 простых чисел. В каждой итерации будут изменяться значения изначальной заданных констант.

$a = H0,\ b = H1,\ c = H2,\ d = H3,\ e = H4,\ f = H5,\ g = H6,\ h = H7$

$\Sigma 0 := (a\ rotr\ 2)\ xor\ (a\ rotr\ 13)\ xor\ (a\ rotr\ 22)$

$Ma := (a\ and\ b)\ xor\ (a\ and\ c)\ xor\ (b\ and\ c)$

$t2 := \Sigma 0 + Ma$

$\Sigma 1 := (e\ rotr\ 6)\ xor\ (e\ rotr\ 11)\ xor\ (e\ rotr\ 25)$

$Ch := (e\ and\ f)\ xor\ ((not\ e)\ and\ g)$

$t1 := h + \Sigma 1 + Ch + k[i] + w[i]$

$h = g,\ g = f,\ f = e,\ e = d + t1,\ d = c,\ c = b,\ b = a,\ a = t1 + t2$

После 64 итераций цикла, сумма значений переменных в последней итерации складываются со старым вектором инициализации и образуют новый:

$h0 \mathrel{+}= a, h1 \mathrel{+}= b, h2 \mathrel{+}= c, h3 \mathrel{+}= d, h4 \mathrel{+}= e, h5 \mathrel{+}= f,$

$h6 \mathrel{+}= g, h7 \mathrel{+}= h$

Сумма элементов вектора, полученного в последнем блоке, станет итоговым хэшем:

$h1 + h2 + h3 + h4 + h5 + h6 + h7 = SHA256$

Здесь схема алгоса

мб пример

1.4.4. Алгоритм хэширования CRC

CRC (Циклический избыточный код) является алгоритмом нахождения контрольной суммы для последующей проверки целостности данных. Главные преимущества циклических кодов - простота их

реализации и обнаружение пакетных ошибок, наиболее распространенных ошибок передачи в каналах связи.

Контрольная сумма - это вычисленное значение по некоторому правилу на основе кодируемого сообщения.

Основой алгоритма CRC служат свойства деления с остатком двоичных многочленов, а само значение CRC представляет из себя остаток от деления многочлена, который является входными данными, на определенный видом CRC порождающий многочлен.

Каждой конечной последовательности битов $a_1, a_2 \dots a_n$ можно взаимно однозначно сопоставить многочлен $\sum_{n=0}^{N-1} a_n x^n$, а количество различных многочленов, меньших степени N равно 2^N. Таким образом, значение контрольной суммы равно:

$$R(x) = P(x)x^N mod\, G(x),$$

где R(x) - полином, представляющий значение CRC, P(x) - полином, соответствующий входным данным, G(x) - порождающий полином, N - степень порождающего полинома.

Основным параметром CRC является порождающий полином. У этого полинома есть степень, которая определяется как количество битов, использующихся для вычисления в алгоритме CRC. Например, 8 степень используется в 8-битном алгоритме CRC-8. Также, еще одним важным параметром для порождающего полинома является стартовое значение слова.

Общий алгоритм подсчета циклического избыточного кода следующий: из входных данных берется первого слово, размером соответствующее битности выбранного CRC. Если старший бит “1”, то выполняется сдвиг влево на один разряд, а затем производится XOR-операция с порождающим многочленом. Если же старшим битом

является “0”, то после сдвига не происходит XOR-операции. После сдвига старший бит удаляется, а на место младшего записывается следующий бит из входных данных. Алгоритм останавливается после того, как был загружен последний бит данных, а получившийся остаток и является контрольной суммой.

Существует множество видов CRC-алгоритма, которые отличаются не только размерностью (CRC-8, CRC-16, CRC-32), но и порождающим полиномом.

Для реализации был выбран алгоритм CRC-16/CCITT, где порождающий полином:

$$x^{16} + x^{12} + x^{5} + 1$$

Здесь схема алгоса + мб пример

1.4.5. Алгоритм хэширования MD

Для реализации алгоритма хэширования MD была выбрана хэш-функция MD-5.

MD-5 это 128-битная хэш-функция, которая входные данные произвольной длины преобразует в хэш длиной 128-бита. Алгоритм был разработан в 1992 году профессором Рональдом Л. Риверсом, а за основу был взят MD-4.

Алгоритм состоит из 5 шагов:

1. Выравнивание потока;
2. Добавление длины сообщения;
3. Инициализация буфера;
4. Вычисления в цикле;
5. Результат выполнения.

На этапе “Выравнивание потока” в конец сообщения дописывают “1”, а затем некоторое количество нулей, чтобы размер входных данных был сравним с 448 по модулю 512.

На следующем этапе в оставшиеся 64 бита дописывают длину сообщения до выравнивания. Если же длина превосходит $2^{64} - 1$, то записывают только младшие биты. Таким образом, длина потокового сообщения становится кратной 512, а вычисления будут проводиться с массивом данных длиной по 512 бита.

На третьем этапе инициализируются 4 32-битные переменные для хранения промежуточных результатов, а также задаются начальные их значения в 16-ричном виде:

$A = 01\ 23\ 45\ 67;\ //\ 67452301h;$

$B = 89\ AB\ CD\ EF;\ //\ EFCDAB89h;$

$C = FE\ DC\ BA\ 98;\ //\ 98BADCFEh;$

$D = 76\ 54\ 32\ 10.\ //\ 10325476h.$

Далее, производятся вычисления в 4 этапа по 16 раундов для каждого отдельного 512-битного блока данных. Каждый этап состоит из своей функции:

1. $F(X, Y, Z) = (X \wedge Y) \vee (\neg X \wedge Z)$;
2. $F(X, Y, Z) = (X \wedge Z) \vee (\neg Z \wedge Y)$;
3. $F(X, Y, Z) = X \oplus Y \oplus Z$;
4. $F(X, Y, Z) = Y \oplus (\neg Z \vee X)$.

Также, для вычислений необходима специальная таблица констант T[64], вычисляемая как:

$T[n] = int(2^{32}|sin(n)|)$

Каждый раунд на каждом этапе вычисляется по формуле:

$a = b + ((a + Func(b, c, d) + X[k] + T[i]) \ll s)$,

где k - номер 32-битного слова из текущего блока сообщения, $\ll s$ - циклический сдвиг вправо на s бит полученного 32-битного аргумента, s задается отдельно для каждого раунда, Func(b, c, d) - функция для данного этапа.

Далее, заносим в блок данных элемент *n* из массива 512-битных блоков. Сохраняются значения A, B, C и D, оставшиеся после операций над предыдущими блоками (или их начальные значения, если блок первый), а затем суммируем с результатом выполнения предыдущего цикла.

Результатом вычислений находится в буфере ABCD, выводится начиная с младшего байта A и до старшего байта D.

Здесь схема алгоса + мб пример

2 Технологическая часть

Для реализации МК-системы хэширования данных была использована среда программирования STM32CubeIDE, TerminalTMBv3 для эмуляции ввода данных с COM-порта. Симуляция проводилась в программе Proteus 8.13, программа написана на языке С.

2.1. Отладка и тестирование программы

Отладка программы производилась в среде программирования STM32CubeIDE с помощью программы Proteus 8.13, предназначенной для моделирования аналоговых и цифровых устройств и при помощи терминала TerminalTMBv3 для эмуляции ввода данных с COM-порта вместо ввода данных с телефона по USB. Также, для эмуляции работы USB было скачано расширение Virtual USB, которое подключает виртуальный COM-порт из Proteus как реальный.

При написании кода использовались библиотеки языка Си:

- string.h - библиотека предназначена для работы со строками в языке Си, оттуда были использована функция strlen(), показывающая размер строки;
- stdio.h - из библиотеки была использована функция sprintf() для преобразования вывода результатов выполнения алгоритмов в консоль;
- stdint.h - библиотека для объявления некоторых целочисленных констант и макросов;
- stdlib.h - библиотека для управления выделением памяти.

Для упрощенного взаимодействия с UART модулем микроконтроллера STM32F103C8T6 были использованы HAL-функции: HAL_UART_Receive() для чтения информации и HAL_UART_Trasmit() для вывода информации.

После сборки проекта создается “.hex” файл объемом 54 Кбайт - вес скомпилированной программы.

В результате отладки и тестирования программы была создана МК-система для хэширования вводимых данных с ПЭВМ и телефона, соответствующая требованиям ТЗ.

2.2. Симуляция работы системы

Для симуляции работы МК-системы были использованы программы Proteus 8.13 и TerminalTMBv3. На рисунке показана МК-система для хэширования данных.

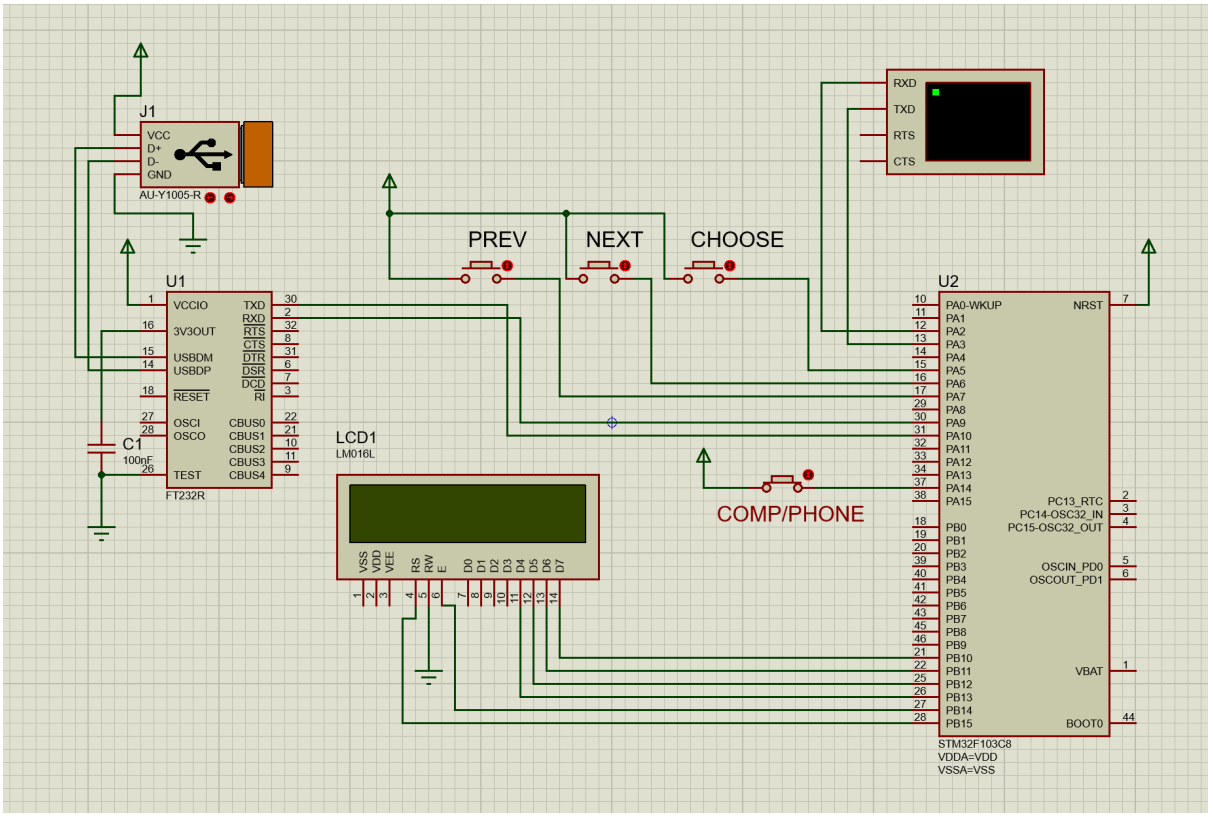

Рисунок - МК-система для хэширования данных.

Для моделирования ввода данных с ПЭВМ использован инструмент системы - Virtual Terminal. Он представляет собой эмуляцию простейшего терминала, позволяющую передавать данные на микроконтроллер по UART через порты RX и TX. На рисунке представлен ввод данных в

терминал и их отображение на ЖК-дисплее, показывающее, что данные дошли до микроконтроллера.

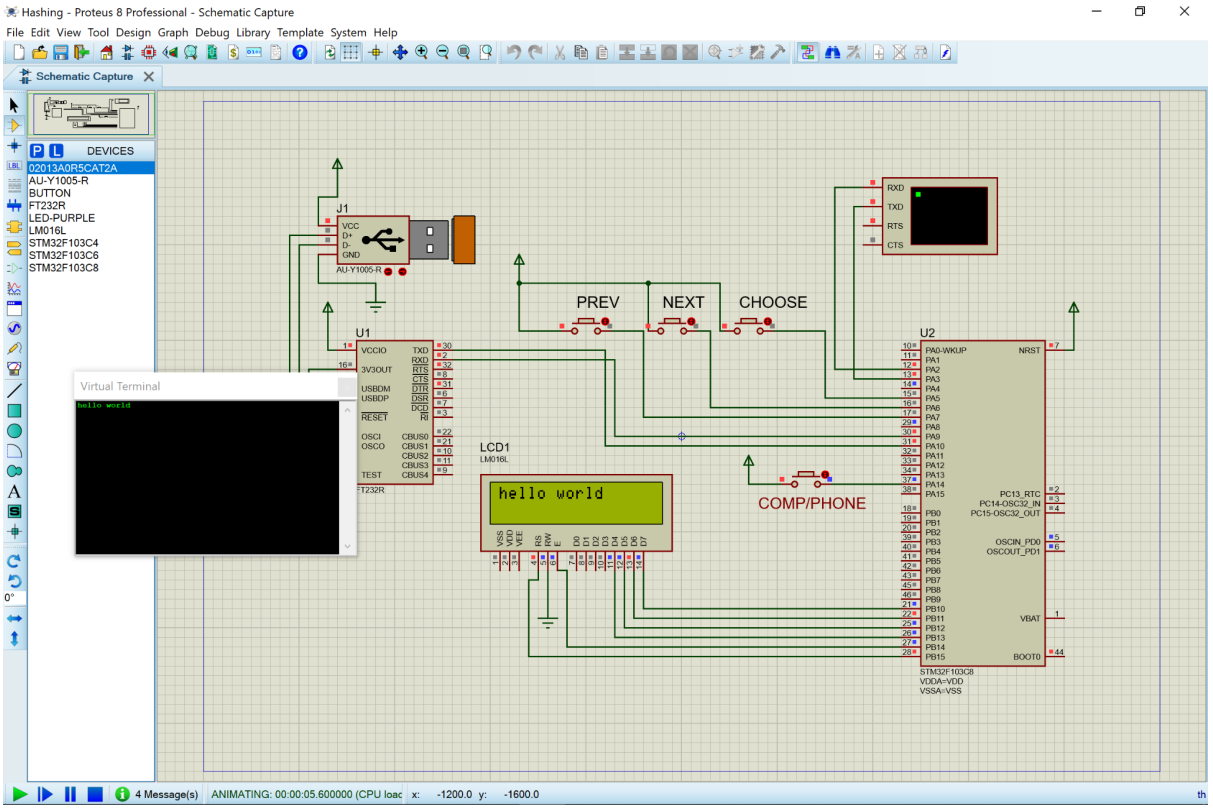

Рисунок - Ввод данных с ПЭВМ через Virtual Terminal.

Для моделирования ввода данных через телефон необходимо эмулировать подключение виртуального USB-коннектора из Proteus к компьютеру, на котором идет запуск программы. Для этого, нужно установить специальное расширение Proteus - Virtual USB Driver. В результате, при запуске программы с USB-коннектором, в Диспетчере устройств компьютера появляется COM-порт данного коннектора - изображено на рисунке.

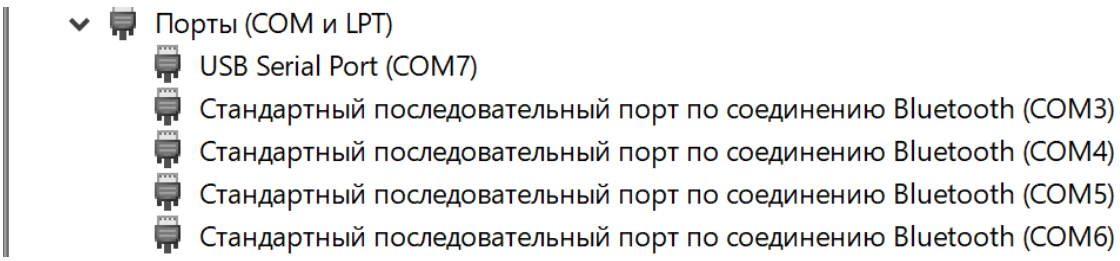

Рисунок - Наличие COM-порта в Диспетчере устройств

Далее, необходимо настроить данный COM-порт как в Диспетчере устройств, так и в программе TerminalTMB в соответствии с настройками микроконтроллера - настройки можно увидеть на рисунках.

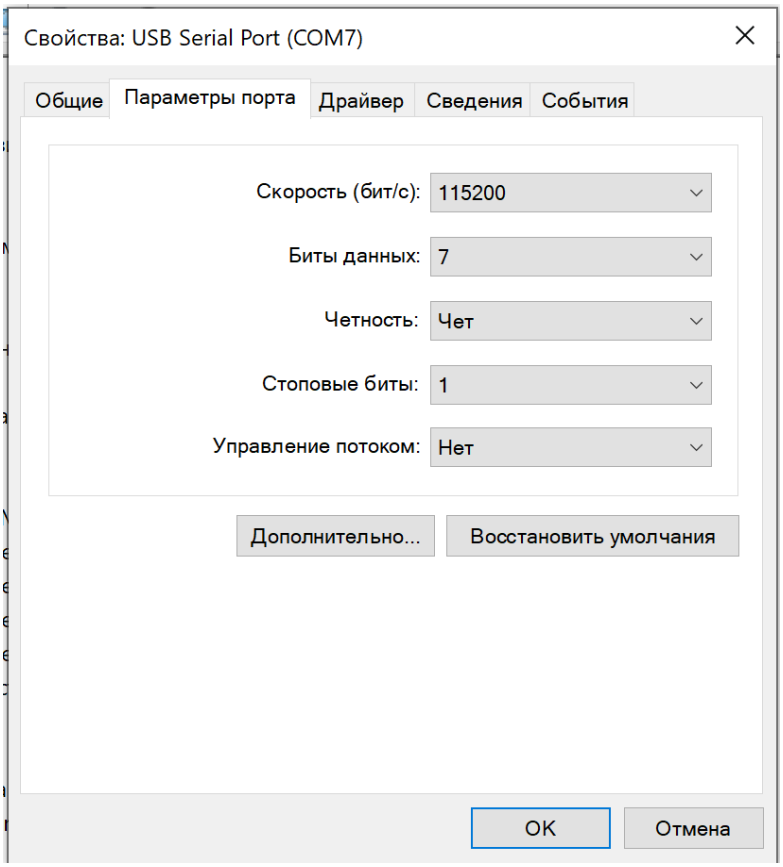

Рисунок - Настроенные параметры порта из Диспетчера устройств

На рисунках представлена эмуляция ввода данных в МК-систему с телефона.

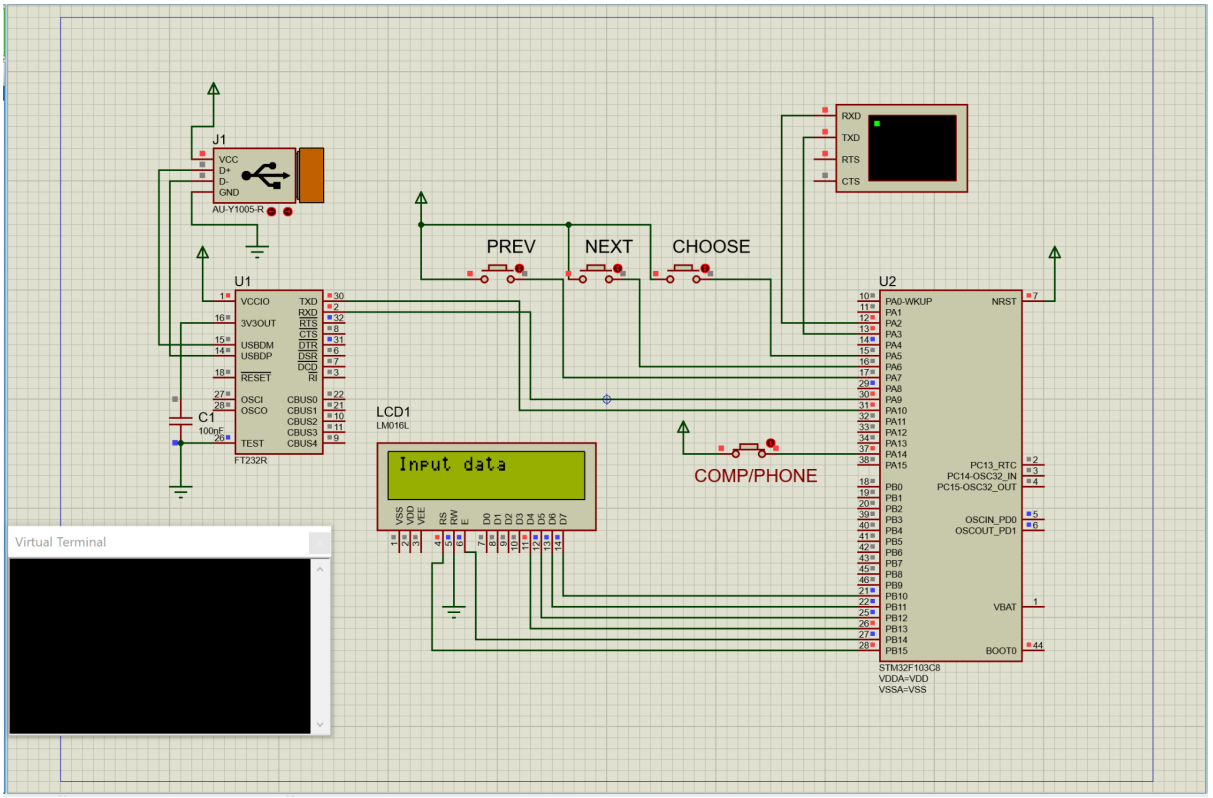

Рисунок - Модель в Proteus, ожидание ввода с USB

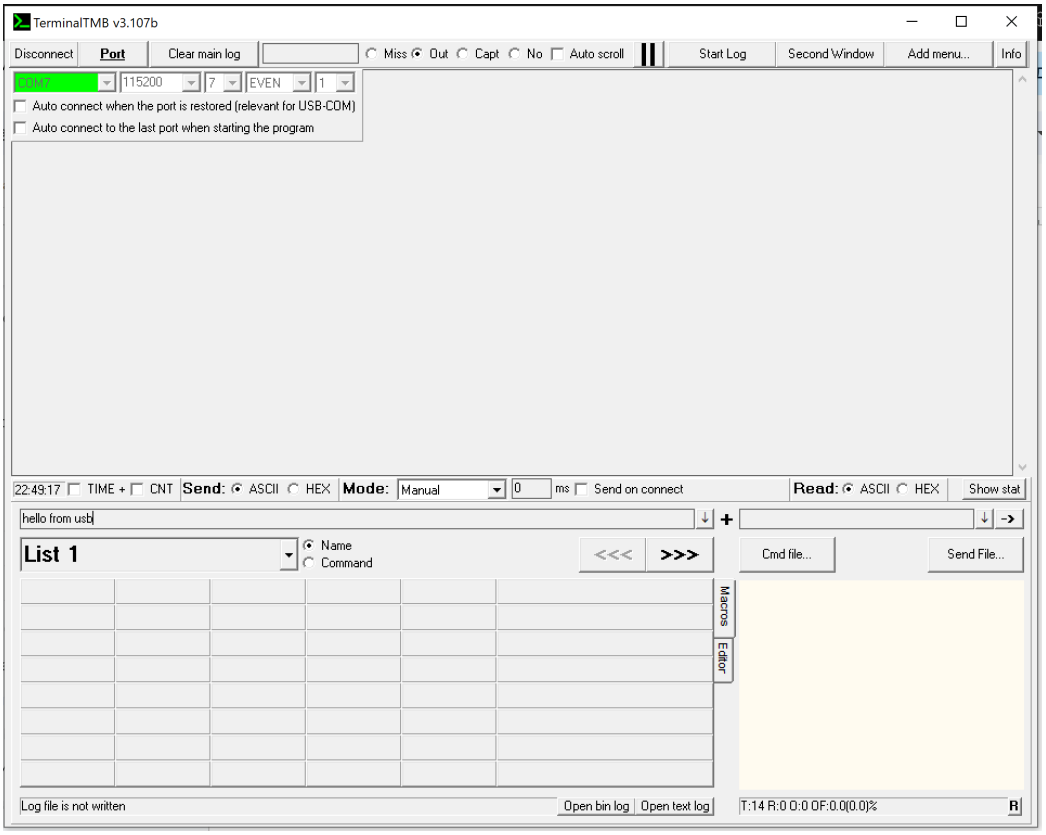

Рисунок - Ввод данных с TerminalTMB, эмулирующего ввод с COM-порта

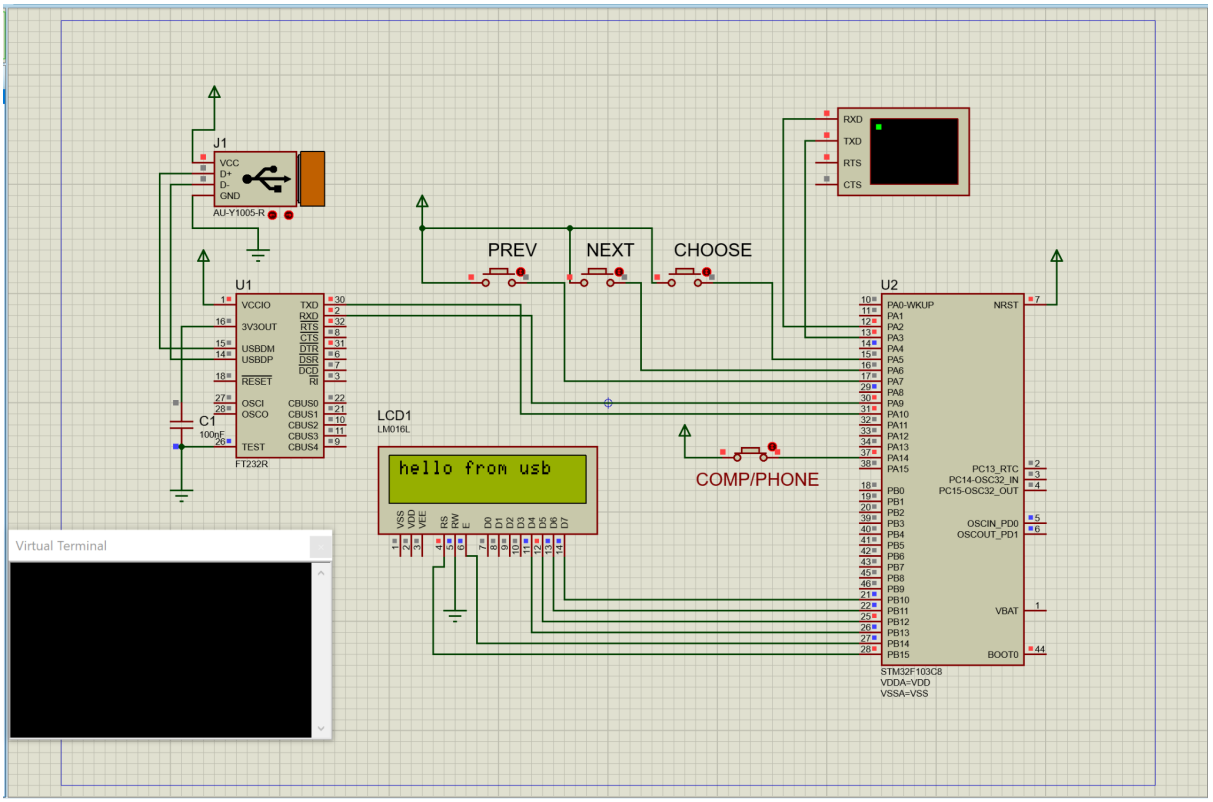

Рисунок - Модель в Proteus, данные от COM-порта получены

ЗАКЛЮЧЕНИЕ

В результате выполнения курсовой работы была разработана МК-система для хэширования данных, вводимых с ПЭВМ и телефона, тремя алгоритмами на выбор - SHA256, MD5, CRC16/CCITT-FALSE. Система работает на микроконтроллере STM32F103C8, она разработана в соответствии с условиями ТЗ.

Код программы для МК написан на языке С в среде программирования STM32CubeIDE, модель отлажена и протестирована в программе Proteus 8.13.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

Приложение А

Текст программы