Министерство образования и науки Российской Федерации
Федеральное агентство по образованию

Государственное образовательное учреждение высшего профессионального образования
«Московский Государственный Технический Университет им. Н.Э. Баумана»
(МГТУ им. Баумана)

Факультет «Информатика и системы управления»
Кафедра «Компьютерные системы и сети»

В.Я. Хартов

Лабораторный практикум для бакалавров по курсу «Микропроцессорные системы»

МГТУ им. Н.Э. Баумана, 2021

Содержание

Предисловие

Лабораторный практикум по курсу «Микропроцессорные системы» предназначен для подготовки бакалавров по направлениям 09.03.01 «Информатика и вычислительная техника» и 09.03.03 « Прикладная информатика”. Практикум предусматривает выполнение восьми лабораторных работ в течение 34 часов аудиторных занятий.

Каждая лабораторная работа предусматривает самостоятельную подготовку, включая изучение:

общих инструментальных средств разработки приложений,

базовых платформ на основе 8-разрядных микроконтроллеров с архитектурой AVR и 32-разрядных микропроцессоров ARM,

системы команд микроконтроллеров AVR,

основ языка Си для высокоуровневого программирования микроконтроллеров с архитектурой AVR и ARM,

теоретических сведений по теме выполняемой работы, представленных в описании работы.

При подготовке к лабораторной работе рекомендуется изучить базовые программы и проработать контрольные вопросы, приведенные в конце каждой темы. Выполнение работ предусматривает выполнение указанных заданий, протоколирование результатов выполнения работы. По итогам каждой работы оформляется отчет. По окончании работ предусмотрена защита, в ходе которой студент должен ответить на контрольные вопросы по теме и содержанию проведенных исследований.

Основу практикума составили материалы учебного пособия автора «Микроконтроллеры AVR. Практикум для начинающих.» 2-е издание, Издательство МГТУ им. Н. Э. Баумана, 2012 г.

Инструментальные средства AVR Studio 4 и STK500

Инструментальными средствами разработки и отладки программ для микроконтроллеров AVR, используемыми в практикуме, являются интегрированный пакет AVR Studio 4 и стартовый набор STK500. Целевыми микроконтроллерами, которые используются во всех проектах практикума, являются микроконтроллеры ATx8515 (AT90S8515, ATmega8515), поставляемые в комплекте со стартовым набором STK500. В связи с этим, прежде чем перейти непосредственно к инструментальным средствам проектирования, рассмотрим кратко основные характеристики используемых микроконтроллеров и их структуру.

1. Архитектура микроконтроллеров ATx8515

Аппаратные ресурсы

Все микроконтроллеры AVR имеют гарвардскую архитектуру, которая предполагает разделение памяти программ и данных. Используемые при этом средства адресации позволяют создавать эффективные программы с высоким быстродействием.

Упрощенная структурная схема микроконтроллера AT90S8515 представлена на рис. 1. Ядро микроконтроллера образуют блок процессора, объединяющий арифметико – логическое устройство (АЛУ) с регистром признаков (SREG) и устройство управления, память программ (FlashROM) объемом 8Кбайт, регистры общего назначения, память данных статического типа (SRAM) объемом 512 байт. Устройство управления представлено схемой синхронизации, регистром управления микроконтроллера (MCUCR), генератором, а также регистром команд с дешифратором, программным счетчиком и указателем стека.

Периферийные устройства представлены достаточно широко:

- 8- разрядные порты ввода-вывода PA, PB, PC, PD;
- последовательный асинхронный приемопередатчик UART (Universal Asynchronous Receiver-Transmitter);
- последовательный синхронный порт SPI (Serial Peripheral Interface);
- 8 - разрядный таймер T0;
- 16 – разрядный таймер T1;
- сторожевой таймер;
- широтно-импульсный модулятор PWM;
- энергонезависимая память EEPROM объемом 512 байт;
- блок прерываний;
- аналоговый компаратор.

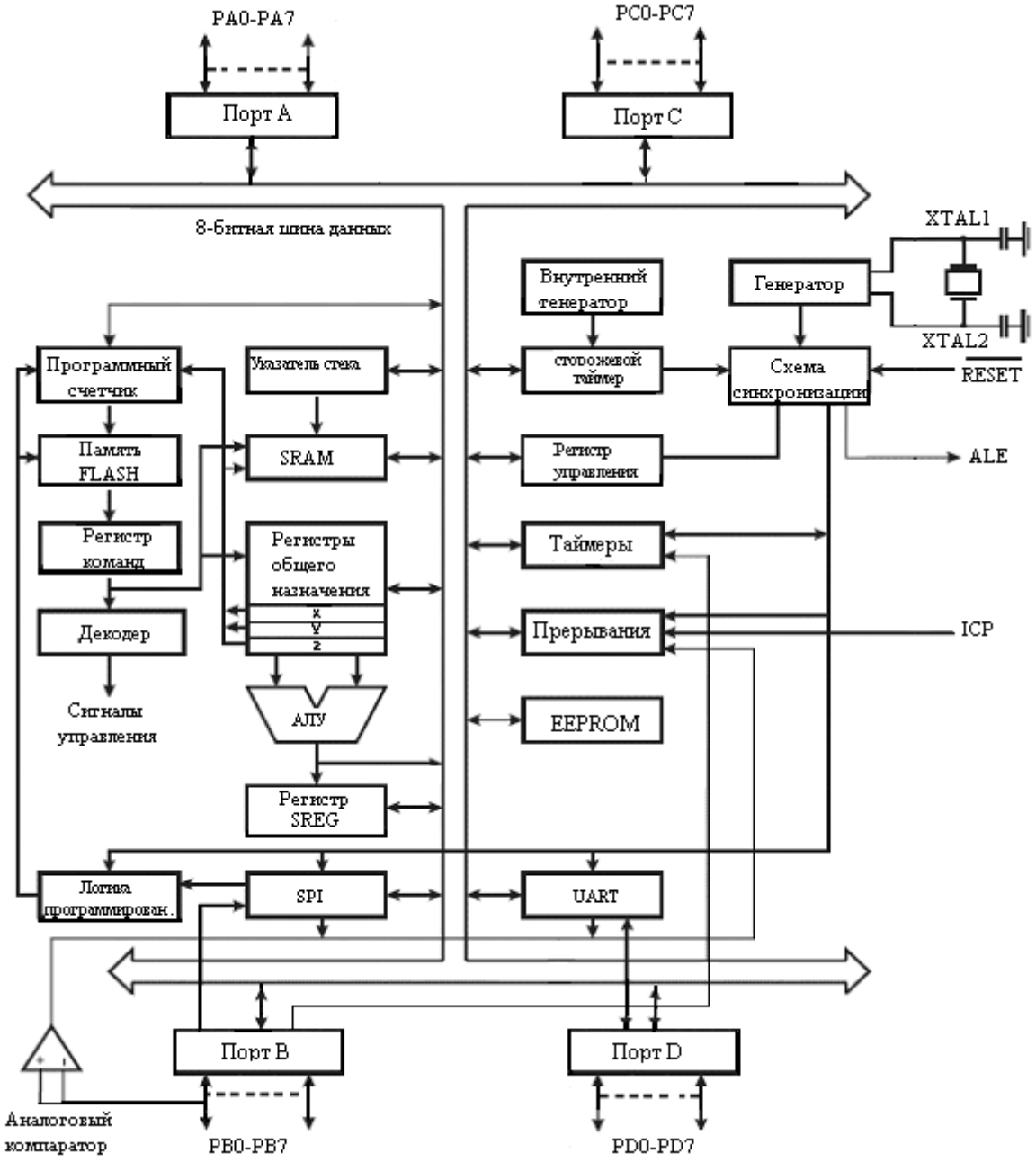


Рис. 1. Структура микроконтроллера AT90S8515

Для программирования памяти программ и энергонезависимой памяти в составе микроконтроллера имеются средства программирования через интерфейс SPI (внутрисистемное программирование).

Память микроконтроллера организована, как показано на рис. 2.

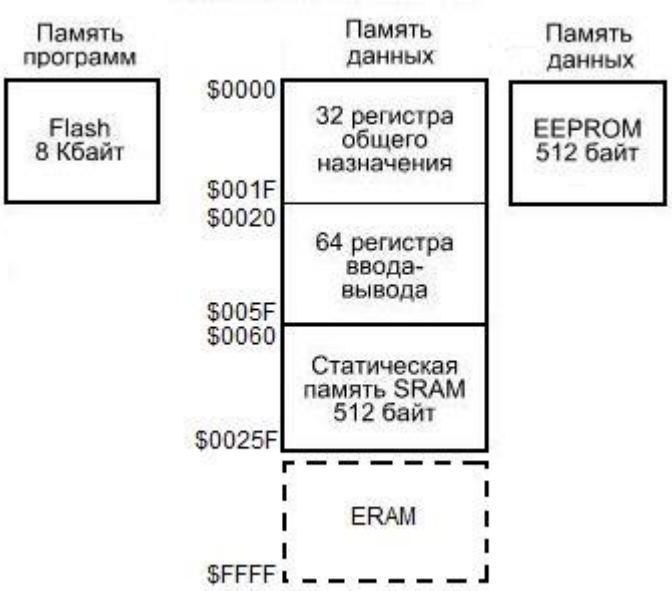


Рис. 2. Карта памяти микроконтроллера AT90S8515

Память программ Flash обособлена, ее размер составляет 8 Кбайт. Каждая ячейка Flash – памяти содержит 16 разрядов.

Память данных делится на три части: регистровая, оперативная статическая SRAM и энергонезависимая EEPROM.

Регистровую память представляют 32 регистра общего назначения и 64 регистра ввода-вывода, представляющих периферийные устройства.

Оперативная память объемом 512 байт предназначена для хранения данных при выполнении программы. Регистровая память и оперативная память образуют единое адресное пространство: регистры общего назначения занимают адреса $0000 - $001F, за ними располагаются регистры ввода-вывода $0020 - $005F, затем ячейки оперативной памяти $0060 - $025F. Расширение адресного пространства, вплоть до верхней границы $FFFF, можно осуществить за счет подключения внешнего запоминающего устройства ERAM .

Для долговременного хранения данных, которые могут изменяться в процессе работы микроконтроллера, используется память EEPROM объемом 512 байт. EEPROM имеет обособленное адресное пространство, каждая ячейка содержит 8 разрядов. Данные в EEPROM могут быть записаны при программировании микроконтроллера. При выключении питания данные сохраняются.

Регистры общего назначения разбиты на две группы: R0…R15 и R16…R31. Принадлежность регистра к той или иной группе необходимо учитывать при написании программы.

Микроконтроллер AT90S8515 содержит 44 регистра ввода-вывода, используемых в составе периферийных устройств (еще 20 регистров зарезервировано), из них 14 - это 8- разрядные регистры данных, остальные являются регистрами управления и состояния. Если учесть, что они представлены битами управления и состояния, общее количество которых 187 (!), часть которых

объединена полями с кодами управления, то нетрудно представить, насколько многообразны функциональные возможности этих устройств (к примеру, одно 3-разрядное поле управления позволяет задать 8 управляющих функций). Список регистров ввода-вывода и их адреса в адресном пространстве SRAM приведены в табл.1.

Таблица 1. Адреса регистров ввода-вывода AT90S8515

Адрес	20	30	40	50
0	Резерв	PIND	Резерв	Резерв
1	Резерв	DDRD	WDTCR	Резерв
2	Резерв	PORTD	Резерв	TCNT0
3	Резерв	PINC	Резерв	TCCR0
4	Резерв	DDRC	ICR1L	Резерв
5	Резерв	PORTC	ICR1H	MCUCR
6	Резерв	PINB	Резерв	Резерв
7	Резерв	DDRB	Резерв	Резерв
8	ACSR	PORTB	OCR1BL	TIFR
9	UBRR*	PINA	OCR1BH	TIMSK
A	UCR*	DDRA	OCR1AL	GIFR
B	USR*	PORTA	OCR1AH	GIMSK*
C	UDR	EECR	TCNT1L	Резерв
D	SPCR	EEDR	TCNT1H	SPL
E	SPSR	EEARL	TCCR1B	SPH
F	SPDR	Резерв	TCCR1A	SREG

Все регистры имеют штатные имена, которые можно использовать при написании программ. Для этого в программу на языке Ассемблера необходимо включить файл определений *8515def.inc*, в программу на языке Си – файл *90s8515.h*.

Расшифровка названий регистров, формат и назначение каждого бита в отдельности приведены в последующих разделах практикума при описании работы соответствующих периферийных устройств. Регистры, имена которых отмечены знаком «*», в модели ATmega8515 получили другие имена: по адресу $29 - UBRRL, по адресу $2A- UCSRB, по адресу $2B - UCSRA, по адресу $5B – GICR.

Список регистров ввода-вывода микроконтроллера ATmega8515 и их адреса в адресном пространстве SRAM приведен в табл. 2. Микроконтроллер ATmega8515 содержит 55 регистров ввода-

вывода. Файл определений штатных имен регистров для ATmega8515, включаемый в программы на Ассемблере, носит имя *m8515def.inc*, на языке Си - *mega8515.h.*

Таблица 2. Адреса регистров ввода-вывода ATmega8515

Адрес	**20**	**30**	**40**	**50**
0	Резерв	PIND	UBRRH	SFIOR
1	Резерв	DDRD	WDTCR	OCR0
2	Резерв	PORTD	Резерв	TCNT0
3	Резерв	PINC	Резерв	TCCR0
4	OSCCAL	DDRC	ICR1L	MCUCSR
5	PINE	PORTC	ICR1H	MCUCR
6	DDRE	PINB	Резерв	EMCUCR
7	PORTE	DDRB	Резерв	SPMCR
8	ACSR	PORTB	OCR1BL	TIFR
9	UBRRL	PINA	OCR1BH	TIMSK
A	UCSRB	DDRA	OCR1AL	GIFR
B	UCSRA	PORTA	OCR1AH	GICR
C	UDR	EECR	TCNT1L	Резерв
D	SPCR	EEDR	TCNT1H	SPL
E	SPSR	EEARL	TCCR1B	SPH
F	SPDR	EEARH	TCCR1A	SREG

Оперативная память служит для оперативного хранения данных при выполнении программы. Данные для записи в ячейку SRAM поступают из регистра общего назначения. Считываемые из ячейки памяти данные поступают в регистр общего назначения. При выключении напряжения питания данные в памяти SRAM теряются.

Стек располагается в памяти SRAM, обычно под него выделяется область адресов, начиная с адреса $025F. Стек растет в сторону убывающих адресов, контроль размера стека возлагается на программиста.

Контроллер прерываний обрабатывает внешние прерывания и прерывания от периферийных устройств микроконтроллера (таймеров, портов последовательного ввода-вывода, аналогового компаратора и др.). Все прерывания являются маскируемыми.

Аналоговый компаратор сравнивает по величине аналоговые сигналы, поступающие на входы порта PB2, PB3, и формирует запрос прерывания ANA COMP при изменении знака разности, а также сигнал захвата для таймера.

Интерфейс

На рис. 3 приведены условные графические обозначения микросхем микроконтроллеров AT90S8515 и ATmega8515, используемых в практикуме. Большинство портовых выводов имеют альтернативные функции, конфигурируемые автоматически при их использовании периферийными устройствами микроконтроллера. Их имена приведены в скобках. Микроконтроллеры ATx8515 выпускаются в различных корпусах: TQFP, PLCC, PDIP. Поставляемые со стартовым набором STK500 микроконтроллеры выполнены в 40 – выводном корпусе PDIP.

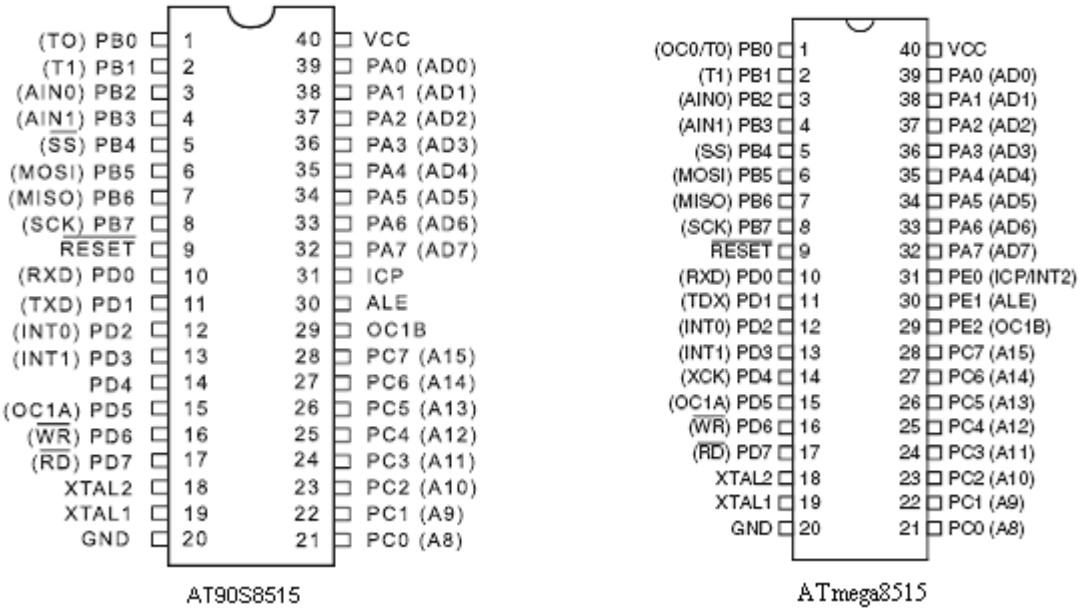


Рис. 3. Интерфейсы микроконтроллеров ATx8515

Система команд микроконтроллеров AVR

Система команд микроконтроллера AT90S8515 содержит 118 команд, ATmega8515 -130 команд. Все команды можно разбить по группам:

- команды арифметических и логических операций;
- команды пересылки и загрузки;
- команды управления;
- команды операций с битами.

В табл. 3 – 6 приведено описание базового набора команд микроконтроллеров AVR. При описании команд использованы следующие обозначения:

Rd, Rr – регистры общего назначения с номерами d и r,

Rdh:Rdl – пара регистров,

K – константа (данные),

P,b – разряд b (b=0,...7) порта P,

Rr(b)- разряд b (b=0,...7) регистра Rr,

(X), (Y), (Z) – содержимое регистровой пары X, Y, Z соответственно,

s – номер разряда в регистре SREG,

PC – содержимое программного счетчика,

k – приращение в счетчике команд,

q – 6-разрядное смещение,

STACK – область памяти SRAM, адресуемая указателем стека SP,

C, Z, N, V, S, H, T, I – биты регистра состояния SREG,

d, r = 0…31 во всех случаях, кроме специально отмеченных.

Мнемоники команд, выполняемых только в микроконтроллерах семейства Mega, помечены «*». В группу арифметических команд микроконтроллера ATmega8515 также добавлены шесть операций умножения: беззнаковых чисел MUL, чисел со знаком MULS, беззнакового числа на число со знаком MULSU, дробных беззнаковых чисел FMUL, дробных со знаком FMULS, дробного беззнакового и дробного со знаком FMULSU. Дробные сомножители имеют формат 1.7, их произведение – формат 1.15, где справа от точки указано число дробных разрядов. Во всех операциях умножения источниками операндов являются регистры Rd и Rr, произведение формируется в регистрах R1:R0.

Таблица 3. Арифметические и логические операции

Мнемоника	Описание команды	Операция	Признаки
ADD Rd, Rr	Сложение двух регистров	Rd←Rd + Rr	Z,C,N,V,H
ADC Rd, Rr	Сложение двух регистров и переноса	Rd←Rd + Rr + C	Z,C,N,V,H
ADIW Rdl, K	Сложение регистровой пары с константой	Rdh:Rdl←Rdh:Rdl + K	Z,C,N,V,S
SUB Rd, Rr	Вычитание двух регистров	Rd←Rd - Rr	Z,C,N,V,H
SUBI Rd, K	Вычитание константы из регистра	Rd←Rd – K, d=16-31	Z,C,N,V,H
SBC Rd, Rr	Вычитание двух регистров с заемом	Rd←Rd - Rr - C	Z,C,N,V,H
SBCI Rd, K	Вычитание константы из регистра с заемом	Rd←Rd - K – C, d=16-31	Z,C,N,V,H
SBIW Rdl, K	Вычитание константы из регистровой пары	Rdh:Rdl←Rdh:Rdl - K	Z,C,N,V,S
AND Rd, Rr	Логическое И двух регистров	Rd←Rd ^ Rr	Z,N,V
ANDI Rd, K	Логическое И регистра и константы	Rd←Rd ^ K, d=16-31	Z,N,V
OR Rd, Rr	Логическое ИЛИ двух регистров	Rd←Rd v Rr	Z,N,V
ORI Rd, K	Логическое ИЛИ регистра и константы	Rd←Rd v K, d=16-31	Z,N,V
EOR Rd, Rr	Логическое исключающее ИЛИ регистров	Rd←Rd ⊕ Rr	Z,N,V
LSL Rd	Логический сдвиг влево	Rd(n+1)←Rd(n),Rd(0)←0	Z,C,N,V
LSR Rd	Логический сдвиг вправо	Rd(n)←Rd(n+1),Rd(7)←0	Z,C,N,V
ROL Rd	Сдвиг влево через перенос	Rd(0)←C, Rd(n+1)←Rd(n),C←Rd(7)	Z,C,N,V
ROR Rd	Сдвиг вправо через перенос	Rd(7)←C, Rd(n)←Rd(n+1),C←Rd(0)	Z,C,N,V
ASR Rd	Арифметический сдвиг вправо	Rd(n)←Rd(n+1), n=0…6	Z,C,N,V
CP Rd, Rr	Сравнение регистров	Rd–Rr	Z,N,V,C,H
CPC Rd, Rr	Сравнение регистров с учетом заема	Rd - Rr – C	Z,N,V,C,H
CPI Rd, K	Сравнение регистра с константой	Rd – K, d=16-31	Z,N,V,C,H
COM Rd	Инверсия регистра	Rd ← $FF - Rd	Z,C,N,V
NEG Rd	Изменение знака	Rd ← $00 - Rd	Z,C,N,V,H
SBR Rd, K	Логическое ИЛИ регистра и константы	Rd←Rd v K, d=16-31	Z,N,V
CBR Rd, K	Логическое И Rd с инверсией константы	Rd←Rd ^ ($FF - K)	Z,N,V
INC Rd	Инкремент регистра	Rd ← Rd + 1	Z,N,V
DEC Rd	Декремент регистра	Rd ← Rd - 1	Z,N,V
TST Rd	Проверка регистра	Rd ← Rd ^ Rd	Z,N,V
CLR Rd	Сброс регистра в «0»	Rd ← Rd ⊕ Rd	Z,N,V
SER Rd	Установка «1» в разрядах регистра	Rd ← $FF, d=16-31	

Таблица 4. Команды пересылки

Мнемоника	**Описание команды**	**Операция**
MOV Rd, Rr	Пересылка между регистрами	Rd ← Rr
*MOVW Rd, Rr	Пересылка между парами регистров	Rd+1:Rd ← Rr+1:Rr
SWAP Rd	Обмен тетрадами	Rd(3..0)↔Rd(7..4)
LDI Rd, K	Загрузка константы в регистр	Rd ← K, d=16-31
LD Rd, X	Косвенная загрузка регистра	Rd ← (X)
LD Rd, X+	Косвенная загрузка с постинкрементом	Rd ← (X), X ← X + 1
LD Rd, -X	Косвенная загрузка с преддекрементом	X ← X - 1, Rd ← (X)
LD Rd, Y	Косвенная загрузка регистра	Rd ← (Y)
LD Rd, Y+	Косвенная загрузка с постинкрементом	Rd ← (Y), Y ← Y + 1
LD Rd, -Y	Косвенная загрузка с преддекрементом	Y ← Y - 1, Rd ← (Y)
LDD Rd,Y+q	Косвенная загрузка относительная	Rd ← (Y + q)
LD Rd, Z	Косвенная загрузка регистра	Rd ← (Z)
LD Rd, Z+	Косвенная загрузка с постинкрементом	Rd ← (Z), Z ← Z+1
LD Rd, -Z	Косвенная загрузка с преддекрементом	Z ← Z - 1, Rd ← (Z)
LDD Rd, Z+q	Косвенная загрузка относительная	Rd ← (Z + q)
LDS Rd, k	Прямая загрузка регистра	Rd ← (k)
ST X, Rr	Косвенное сохранение	(X) ← Rr
ST X+, Rr	Косвенное сохранение с постинкрементом	(X) ← Rr, X ← X + 1
ST –X, Rr	Косвенное сохранение с преддекрементом	X ← X - 1, (X) ← Rr
ST Y, Rr	Косвенное сохранение	(Y) ← Rr
ST Y+, Rr	Косвенное сохранение с постинкрементом	(Y) ← Rr, Y ← Y + 1
ST –Y, Rr	Косвенное сохранение с преддекрементом	Y ← Y - 1, (Y) ← Rr
STD Y+q, Rr	Косвенное сохранение относительное	(Y + q) ← Rr
ST Z, Rr	Косвенное сохранение	(Z) ← Rr
ST Z+, Rr	Косвенное сохранение с постинкрементом	(Z) ← Rr, Z ← Z + 1
ST -Z, Rr	Косвенное сохранение с преддекрементом	Z ← Z - 1, (Z) ← Rr
STD Z+q, Rr	Косвенное сохранение относительное	(Z + q) ← Rr
STS k, Rr	Прямое сохранение	(k) ← Rr
LPM	Загрузка из программной памяти в R0	R0 ← (Z)
*LPM Rd, Z+	Загрузка из программной памяти в регистр Rd с постинкрементом	Rd ←(Z), Z ← Z+1
*SPM	Сохранение в программной памяти	(Z)← R1:R0
IN Rd, P	Чтение регистра ввода-вывода	Rd ← P, P=0-63
OUT P, Rr	Запись в регистр ввода-вывода	P ← Rr, P=0-63

PUSH Rr	Сохранение в стеке	STACK ← Rr
POP Rd	Извлечение из стека	Rd ← STACK

Таблица 5. Команды передачи управления

Мнемоника	**Описание команды**	**Операция**
RJMP k	Переход	PC ← PC + k + 1
IJMP	Косвенный переход по (Z)	PC ← Z
RCALL k	Вызов подпрограммы	PC ← PC + k + 1
ICALL	Косвенный вызов по (Z)	PC ← Z
RET	Возврат из подпрограммы	PC ← STACK
RETI	Возврат из прерывания	PC ← STACK, I
CPSE Rd,Rr	Сравнить и пропустить команду, если равны	если (Rd = Rr), то PC←PC + 2/3 **
SBRC Rr,b	Пропустить, если бит в регистре = 0	если (Rr(b)=0), то PC←PC + 2/3
SBRS Rr,b	Пропустить, если бит в регистре = 1	если (Rr(b)=1), то PC←PC + 2/3
SBIC P,b	Пропустить, если бит порта = 0	если (P(b)=0), то PC←PC + 2/3
SBIS P,b	Пропустить, если бит порта = 1	если (P(b)=1), то PC← PC + 2/3
BRBS s,k	Перейти, если флаг в SREG = 1	если (SREG(s)=1), то PC←PC + k + 1
BRBC s,k	Перейти, если флаг в SREG = 0	если (SREG(s)=0), то PC←PC + k + 1
BREQ k	Перейти, если равно	если (Z = 1), то PC←PC + k + 1
BRNE k	Перейти, если не равно	если (Z = 0), то PC←PC + k + 1
BRCS k	Перейти, если C = 1	если (C = 1), то PC←PC + k + 1
BRCC k	Перейти, если C = 0	если (C = 0), то PC←PC + k + 1
BRSH k	Перейти, если больше или равно	если (C = 0), то PC←PC + k + 1
BRLO k	Перейти, если меньше	если (C = 1), то PC←PC + k + 1
BRMI k	Перейти, если минус	если (N = 1), то PC←PC + k + 1
BRPL k	Перейти, если плюс	если (N = 0), то PC←PC + k + 1
BRGE k	Перейти, если больше или равно (со знаком)	если (N⊕V=0), то PC←PC+k+1
BRLT k	Перейти, если меньше (со знаком)	если (N⊕V=1), то PC←PC+k+1
BRHS k	Перейти, если межтетрадный перенос H = 1	если (H = 1), то PC←PC+k+1
BRHC k	Перейти, если межтетрадный перенос H = 0	если (H = 0), то PC ← PC + k + 1
BRTS k	Перейти, если флаг T = 1	если (T = 1), то PC ← PC + k + 1
BRTC k	Перейти, если флаг T = 0	если (T = 0), то PC ← PC + k + 1
BRVS k	Перейти, если флаг переполнения V= 1	если (V = 1), то PC ← PC + k + 1
BRVC k	Перейти, если флаг переполнения V = 0	если (V = 0), то PC ← PC + k + 1
BRIE k	Перейти, если флаг прерывания I = 1	если (I = 1), то PC ← PC + k + 1

BRID k	Перейти, если флаг прерывания I = 0	если (I = 0), то PC ← PC + k + 1

**) При невыполнении условия совершается переход к очередной команде по адресу (PC+1). Если условие выполнено, происходит переход к команде, следующей за очередной. Содержимое программного счетчика увеличивается на 2 или 3 в зависимости от длины следующей команды (1 или 2 слова).

Таблица 6. Операции с битами

Мнемоника	**Операция**	**Мнемоника**	**Операция**
SBI P,b	(P,b) ← 1, P=0-31	SES	S ← 1
CBI P,b	(P,b) ← 0, P=0-31	CLS	S ← 0
BSET s	SREG(s) ← 1	SEV	V ← 1
BCLR s	SREG(s) ← 0	CLV	V ← 0
BST Rr,b	T ← Rr(b)	SET	T ← 1
BLD Rd,b	Rd(b) ← T	CLT	T ← 0
SEC	C ← 1	SEH	H ← 1
CLC	C ← 0	CLH	H ← 0
SEN	N ← 1		
CLN	N ← 0	NOP	Нет
SEZ	Z ← 1	SLEEP	Режим энерго-сбережения
CLZ	Z ← 0		
SEI	I ← 1	WDR	Сброс WDT
CLI	I ← 0		

Директивы ассемблера

При написании программ на языке ассемблера используются директивы, которые указывают компилятору положение программы в памяти, определяют макросы, инициализируют память и др. Список директив и их описание приведен в табл.7. Запись всех директив начинается с точки. Кратко перечислим выполняемые директивами функции в каждом из сегментов.

Сегмент программы открывается директивой .CSEG. Если программа начинается с этого сегмента, директива может отсутствовать. В сегменте программы с помощью директивы .ORG можно указать начало сегмента.

Директива .DB в сегменте определяет один байт или группу байтов, констант, записываемых во FlashROM. Директива .DW определяет слово или группу слов, записываемых в память в

качестве констант. Начало записи констант определяется меткой, стоящей перед соответствующей директивой. Перечисляемые константы разделяются запятыми.

Таблица 7. Список директив

Директива	Описание
.BYTE	Резервировать байты в ОЗУ
.CSEG	Сегмент программы
.DB	Определить байт - константу во флэш-памяти или EEPROM
.DEF	Назначить регистру символическое имя
.DEVICE	Определяет устройство, для которого компилируется программа
.DSEG	Сегмент данных
.DW	Определяет слово во флэш-памяти или EEPROM
.ENDM, .ENDMACRO	Конец макроса
.EQU	Установить постоянное выражение
.ESEG	Сегмент EEPROM
.EXIT	Выход из файла
.INCLUDE	Вложить другой файл
.LIST	Включить генерацию листинга
.LISTMAC	Включить разворачивание макросов в листинге
.MACRO	Начало макроса
.NOLIST	Выключить генерацию листинга
.ORG	Установить положение в сегменте
.SET	Установить для переменный эквивалентное выражение

Директива .DEF присваивает регистру символическое имя. Директивы .EQU, .SET присваивают значение имени. Имя, которому присвоено значение директивой .EQU, не может быть переназначено, и значение не может быть изменено. Имя, присвоенное директивой .SET, может быть изменено другой директивой .SET.

Директива .DEVICE определяет тип целевого микроконтроллера, который будет использован для выполнения программы. Наличие этой директивы подключает средства контроля инструкций программы по отношению к физическому устройству, предупреждая о невозможности выполнения некоторых инструкций, размеров используемой памяти и др.

Директива .INCLUDE с именем файла используется для включения в текст программы другого файла.

Директивы .MACRO и .ENDMACRO обрамляют макроопределение. Макроопределение может иметь до 10-и параметров, имеющих фиксированные имена @0,...@9. При вызове макроопределения параметры задаются в виде списка в порядке нумерации.

Сегмент данных начинается директивой .DSEG. В сегменте могут быть использованы директивы .ORG и .BYTE. Директива .BYTE определяет количество байтов, к которым будет производиться обращение при выполнении программы. Резервируемая область начинается по адресу, определяемому меткой перед директивой.

Сегмент типа EEPROM начинается директивой .ESEG. В сегменте могут быть использованы директивы .ORG, .DB, .DW. Директива .DB в сегменте определяет один или группу байтов, записываемых в EEPROM. Директива .DW определяет слово или группу слов, записываемых в память EEPROM парами по 2 байта. Начало записи байтов и слов определяется меткой, стоящей перед соответствующей директивой.

Директивы .LIST, .NOLIST, .LISTMAC используются для управления выводом листинга.

Выражения

При записи команд на ассемблере могут использоваться выражения, по которым в процессе ассемблирования программы вычисляются значения. Операндами выражений могут быть:

- числа десятичные, шестнадцатеричные и двоичные,
- метки,
- коды символов ASCII (´A´) и строки ASCII,
- символические имена, представляющие переменные, определенные директивой .SET, и константы, определенные директивой .EQU,
- текущее значение счетчика команд (PC).

Для обозначения шестнадцатеричных чисел используют указатели 0x или $ (0x1a, 0xff, $ff), для двоичных чисел – 0b (0b00001111, 0b11111111), десятичные числа не имеют указателей (255, 0).

Помимо операндов в выражения могут входить функции, например:

- LOW (выражение) – возвращает младший байт выражения,
- HIGH (выражение) - возвращает старший байт выражения,
- EXP2 (N) - возвращает 2^N,
- LOG2 (N) - возвращает целую часть $\log_2 N$.

При записи выражений можно использовать арифметические операции, логические операции и операции отношения. Группу арифметических операций образуют: сложение двух чисел или

выражений (N+M), вычитание (N-M), умножение (N*M), деление (N/M), изменение знака числа (-N). Группу логических операций образуют: инверсия (~N), побитовое И (N&M), побитовое ИЛИ (N | M), побитовое исключающее ИЛИ (N^M), сдвиг влево (N<<M - сдвинуть N влево на M разрядов), сдвиг вправо (N>>M - сдвинуть N вправо на M разрядов). Операции отношений:

- логическое отрицание (!N – возвращает «1», если выражение N=0, и «0», если N≠0);
- меньше (N<M - возвращает «1», если выражение N<M, и «0», если N≥M);
- больше (N>M - возвращает «1», если выражение N>M, и «0», если N≤M);
- меньше или равно (N<=M - возвращает «1», если выражение N<=M, и «0», если N>M);
- больше или равно (N>=M - возвращает «1», если выражение N>=M, и «0», если N<M);
- равно (N==M - возвращает «1», если выражение N=M, и «0», если N≠M);
- не равно (N!=M - возвращает «1», если выражение N≠M, и «0», если N=M);
- логическое И (N&&M - возвращает «1», если N≠0 и M≠0, иначе «0»);
- логическое ИЛИ (N||M - возвращает «1», если выражение N=0 и M=0, иначе «0»).

Для указания очередности операций можно использовать круглые скобки.

2. Интегрированная отладочная среда AVR Studio 4

Широкое применение микроконтроллеров в мире способствовало появлению на рынке программных продуктов сопровождения разработки приложений от различных фирм – производителей программного обеспечения. На смену отдельным программам (ассемблерам, компиляторам, отладчикам и др.) пришли интегрированные системы разработки приложений (IDE - Integrated Development Environment), разработанные под Windows, с удобным пользовательским интерфейсом, множеством функций, начиная от редактирования программ и заканчивая программированием микроконтроллеров.

AVR Studio – это интегрированная отладочная среда разработки приложений для 8-разрядных RISC - микроконтроллеров семейств AVR (ATtiny, AT90, ATmega, ATxmega). Версия AVR Studio 4 объединяет средства управления проектами, текстовый редактор, ассемблер и отладчик программ на языках Си и Ассемблер. Таким образом, AVR Studio 4 поддерживает проектировщика на стадиях разработки, отладки и верификации программного обеспечения. Кроме этого, AVR Studio 4 поддерживает аппаратные платформы STK500, STK600, которые позволяют программировать все устройства AVR, внутрисхемные эмуляторы ICE50, JTAG ICE и др. AVR Studio 4 распространяется бесплатно и доступна на сайте **http://www.microchip.com**.

AVR Studio 4 состоит из нескольких панелей и модулей, каждый из которых выполняет часть общей задачи. Внешний вид программы версии 4.19 в режиме редактора показан на рис. 4.

Создание программ в среде AVR Studio 4 происходит в виде проектов, каждый из которых имеет проектный файл, сохраняющий информацию о проекте и входящих в него файлах, установки ассемблера, пользовательские настройки и т.д.

Редактор служит для написания программного кода, он полнофункционален, имеет подсветку синтаксиса, которая может быть изменена и дополнена пользователем. Окно редактора также используется при отладке, при этом точки возможного программного останова могут быть размещены на левой границе поля.

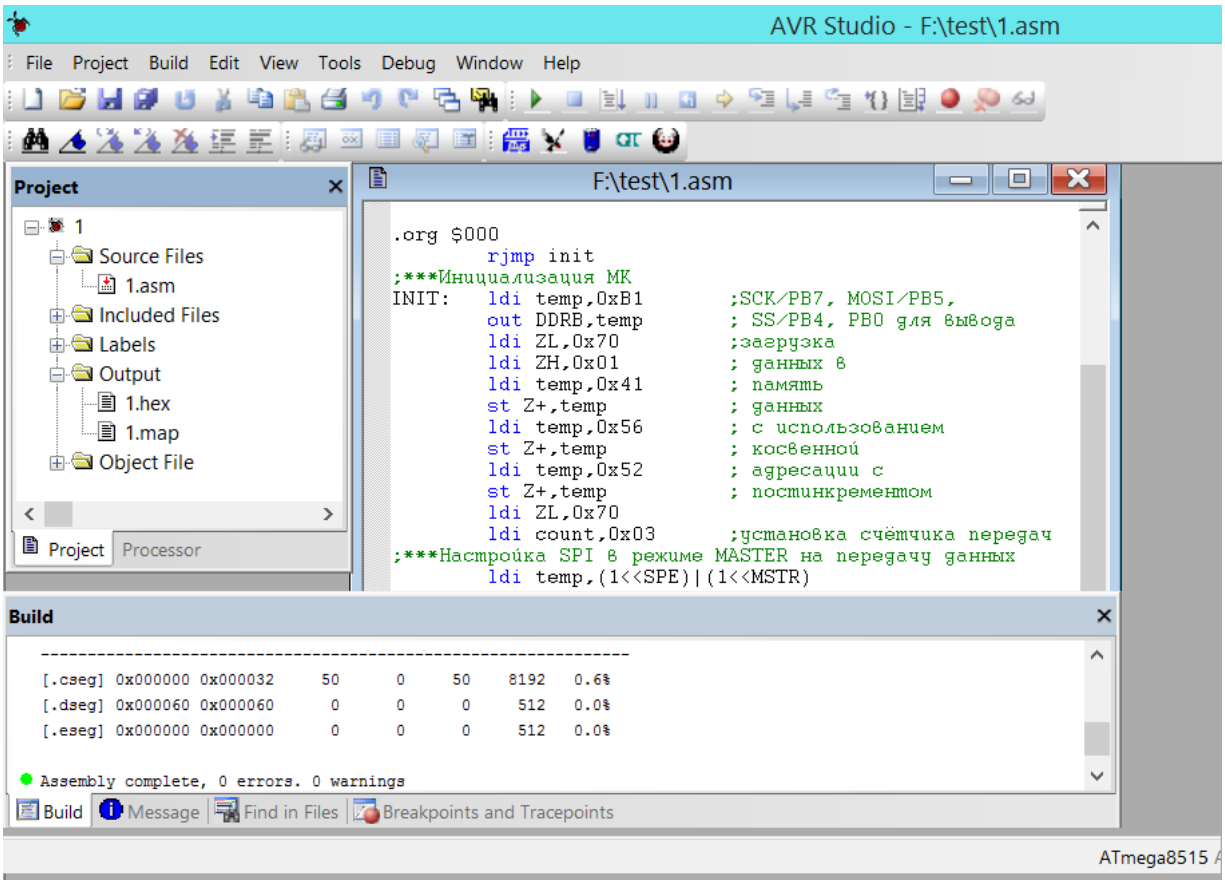


Рис. 4. Графический интерфейс AVR Studio 4

В панели вывода внизу отображается текущая и служебная информация среды разработки. Щелкнув по ярлыку, можно выбрать то или иное окно:

- *Build*. Окно сообщений о процессе и результатах компиляции/трансляции.
- *Messages*. Это общее окно предоставления сообщений пользователю от всех модулей приложения. Сообщения кодируются цветом. Большинство составляют простые сообщения без значимого приоритета. Они не выделяются цветом. Предупреждения о потенциальных проблемах выделяются жёлтым цветом, ошибки - красным. Для всех сообщений может быть записано время прихода (опция *timestamp* контекстного меню). Имеется функция фильтра, позволяющая включать/выключать сообщения разных видов.

- *Find in Files*. AVR Studio 4 имеет функцию встроенного поиска в файлах. В окне отображается информация о результатах поиска.
- *Breakpoints*. Список активных точек возможного прерывания программы во всех модулях программы пользователя. Точки могут быть включены, выключены и удалены в этом окне.

Панель рабочего пространства предназначена для помощи при отладке написанного кода. На ней в окнах *Project, Processor, I/O* можно представить информацию о проекте, регистрах процессора, регистрах ввода-вывода:

- *Project*. Окно со списком файлов, составляющих проект. Если был открыт объектный файл для отладки, то окно покажет имя загруженного файла, а также исходные файлы, с которыми связан данный.
- *Processor*. В окне представлены две секции. Одну представляют регистры управления: *Program Counter* (программный счётчик), *Stack Pointer* (указатель стека), *X-, Y-, Z-* pointer (адресные указатели), *Cycle Counter* (счётчик циклов), *Frequency* (частота работы микроконтроллера), *Stop Watch* (системные часы), *SREG* (регистр состояния программы). Содержимое этих регистров обновляется при прерывании симуляции.

Во второй секции представлены 32 регистра общего назначения (*R00…R31*), которые могут свободно использоваться программистом и также обновляются во время прерывания процесса симуляции. Если состояние регистра изменилось, он выделяется цветом (по умолчанию красным).

- *I/O*. В окне *I/O View* отображаются логически сгруппированные управляющие регистры и регистры данных периферийных устройств, что позволяет осуществить полный контроль периферийного устройства в процессе отладки. Все периферийные устройства имеют 8- или 16-разрядные регистры, образующие группу регистров ввода/вывода, которые доступны для чтения и записи. Микроконтроллеры AVR различаются по количеству и составу встроенных периферийных устройств. Список устройств, отображаемых в окне *I/O View*, соответствует модели выбранного микроконтроллера и изменяется при переходе от одной модели к другой. Например, в состав *I/O* микроконтроллера ATmega8515 (рис. 5) входят пять портов (A, B, C, D, E), два последовательных интерфейса (SPI и USART), два таймера/счетчика, модуль внешних прерываний, память EEPROM, аналоговый компаратор и ряд регистров ЦПУ (CPU).

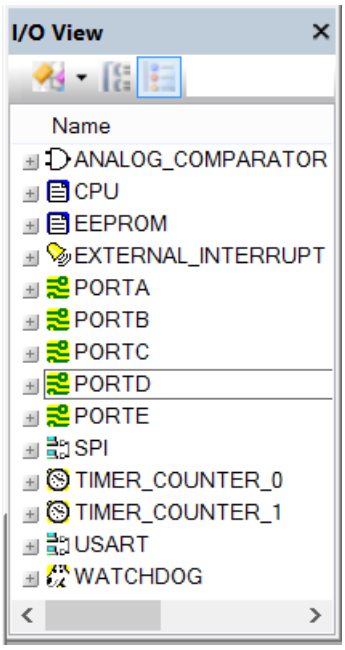


Рис. 5. Окно регистров ввода-вывода МК ATmega8515

Для контроля работы программы в процессе отладки можно открыть ряд окон в меню *View*:

- окно *Disassembler* позволяет осуществить отладку программы, представленной на языке ассемблера с указанием адресов команд в памяти программы;
- окно *Watch* используется для вывода значений переменных при отладке программ. Для этого необходимо мышью “перетащить” переменную из окна программы в данное окно. Если это массив или иная структурная переменная, то рядом появится символ “+”, раскрывающийся при щелчке;
- окно памяти *Memory* может представлять содержимое различных видов памяти микроконтроллера: памяти данных (Data), энергонезависимой памяти (Eeprom), регистров ввода/вывода (I/O), памяти программы (Program), регистров общего назначения (Register). При отладке программы можно открыть три окна памяти;
- окно *Register* служит для отображения содержимого всех регистров общего назначения.

Создание проекта

AVR Assembler - это удобный инструмент для написания небольших программ. После компиляции сразу получается выполняемый код, стадия компоновки отсутствует. Для начала работы при запуске AVR Studio нужно нажать кнопку *New project*, в проектном диалоговом окне выбрать Atmel AVR Assembler, задать имя проекта и рабочую папку для проекта, затем нажать кнопку *Finish*. Можно сразу указать модель микроконтроллера, для которого разрабатывается про-

грамма, нажав *Next* и выбрав платформу для отладки и тип устройства. Создается проектный файл, файл *.asm будет доступен в окне редактора для ввода программы.

При написании программы на языке AVR Assembler можно воспользоваться файлом помощи, где перечислены и объяснены все инструкции и директивы. Обратиться к нему можно, выполнив команды из меню команд AVR Studio 4: *Help /AVR Tools User Guide*. В открывающемся окне *AVR Tools* следует выполнить *AVR Assembler / Parts* и указать тип микроконтроллера. Подробное описание каждой из команд можно найти в разделе *AVR Assembler / Instructions*. Также доступна контекстная помощь при нажатии клавиши F1, дающая информацию о синтаксисе команды, расположенной рядом с курсором.

Предполагается включение в разрабатываемый код директивой .include файла *def.inc, по умолчанию расположенного в папке *\Program Files\Atmel\AVRTools\AVRAssembler\Appnotes*. В подобных файлах для каждого устройства AVR определены мнемоники всех внутренних регистров, битов, векторов прерываний и т.п., что упрощает процесс написания программы для конкретного микроконтроллера.

Для трансляции программы необходимо нажать клавишу F7 или выбрать пункт меню *Project / Build*. При использовании директивы .device с указанием типа микроконтроллера, для которого создается программа, транслятор выполняет проверку программы на наличие в тексте инструкций, недопустимых для выбранного микроконтроллера. При отсутствии директивы такая проверка не производится. Результаты трансляции будут показаны в окне *Build* панели вывода.

Если трансляция прошла без ошибок, выводится сообщение, помеченное зелёным кружком и указывающее, что ошибок нет и созданы файлы с расширениями .hex и .obj. В противном случае список ошибок помечается красными кружками. Для исправления ошибки необходимо дважды щелкнуть левой клавишей мыши по строке сообщения. В соответствующей строке программы появится синяя стрелка и текстовый курсор. При трансляции можно получить файл определений программы (строки с директивами .def и .equ) с расширением .map и файл листинга с расширением .lst, включающий команды в символьном и шестнадцатеричном коде. Для этого необходимо выполнить команды *Project / Assembler Options* и в открывающемся окне установить соответствующие флажки.

Сама по себе успешная трансляция говорит лишь о том, что программа не содержит синтаксических ошибок. Отладка же в симуляторе способна ответить, прежде всего, на такие вопросы: действительно ли алгоритм выполняется так, как это было задумано, и каково время выполнения той или иной процедуры. До начала отладки можно выбрать или изменить платформу для отладки, выполнив команды меню *Debug / Select Platform and Devise* и выбрав *AVR Simulator* и тип устройства.

Запускается отладчик командой меню *Debug/ Start Debugging.* Эта команда будет доступна только в случае успешной трансляции программы.

Выполнив команду меню *Debug/ AVR Simulator Options,* в окне *Device Selection* указываем частоту работы микроконтроллера, а в окне *Stimuli and Logging* - метод работы с портами микроконтроллера.

Возможен автоматический ввод данных в порт (stimuli) из файла с расширением .sti и/или протоколирование (logging) вывода. В обоих случаях данные представляются в виде:

номер цикла : данные на ввод/вывод в шестнадцатеричном коде.

Протоколируя вывод, указываем имя порта, устанавливаем флаг *To screen* для вывода на экран. Затем нажимаем кнопки *Add Entry* и *OK*.

Подготовительные операции закончены. Исходное состояние: все регистры микроконтроллера в окне *I/O* сброшены в «0», желтая стрелка в окне редактора указывает на первую команду программы. Используя опции меню *Debug*, выполняем отладку в одном из выбранных режимов: пошаговый, с контрольными точками, в автоматическом режиме.

3. Стартовый набор STK500

STK500 представляет собой универсальный стартовый набор разработчика, позволяющий выполнять разработку приложений совместно с интегрированной средой проектирования AVR Studio 4.

Набор STK500 поставляется с микроконтроллером ATx8515, но поддерживает целый ряд других микроконтроллеров AVR, имея для этого необходимые панели для установки и средства коммуникации. Исходные установки перемычек обеспечивают работу микроконтроллера совместно с тактовым генератором и стабилизатором напряжения, установленными на плате STK500. Набор имеет также широко используемые средства ввода и индикации, интерфейс RS-232, средства расширения для подключения внешних устройств.

Описание аппаратных средств

В состав отладочной платы STK500 (рис. 6) входят:

- стабилизированный источник питания с входным напряжением 10 – 15В и программно управляемым выходным напряжением;
- 8 кнопок общего назначения;
- 8 светодиодов общего назначения;
- разъемы всех портов ввода-вывода микроконтроллера;
- 8-, 20-, 28-, 40- выводные панели для установки DIP-корпусов микроконтроллеров AVR;

- интерфейс RS-232 для программирования и управления из программы AVR Studio 4, установленной на персональном компьютере;

- дополнительный порт RS-232 общего назначения;

- разъемы расширения для подключения внешних модулей для макетирования;

- встроенная флэш-память DataFlash емкостью 2 Мбит для энергонезависимого хранения данных;

- средства поддержки параллельного и последовательного программирования повышенным напряжением всех AVR-микроконтроллеров;

- средства последовательного внутрисистемного программирования (ISP) всех AVR-микроконтроллеров;

- внутрисистемный программатор для программирования микроконтроллера непосредственно в целевом приложении.

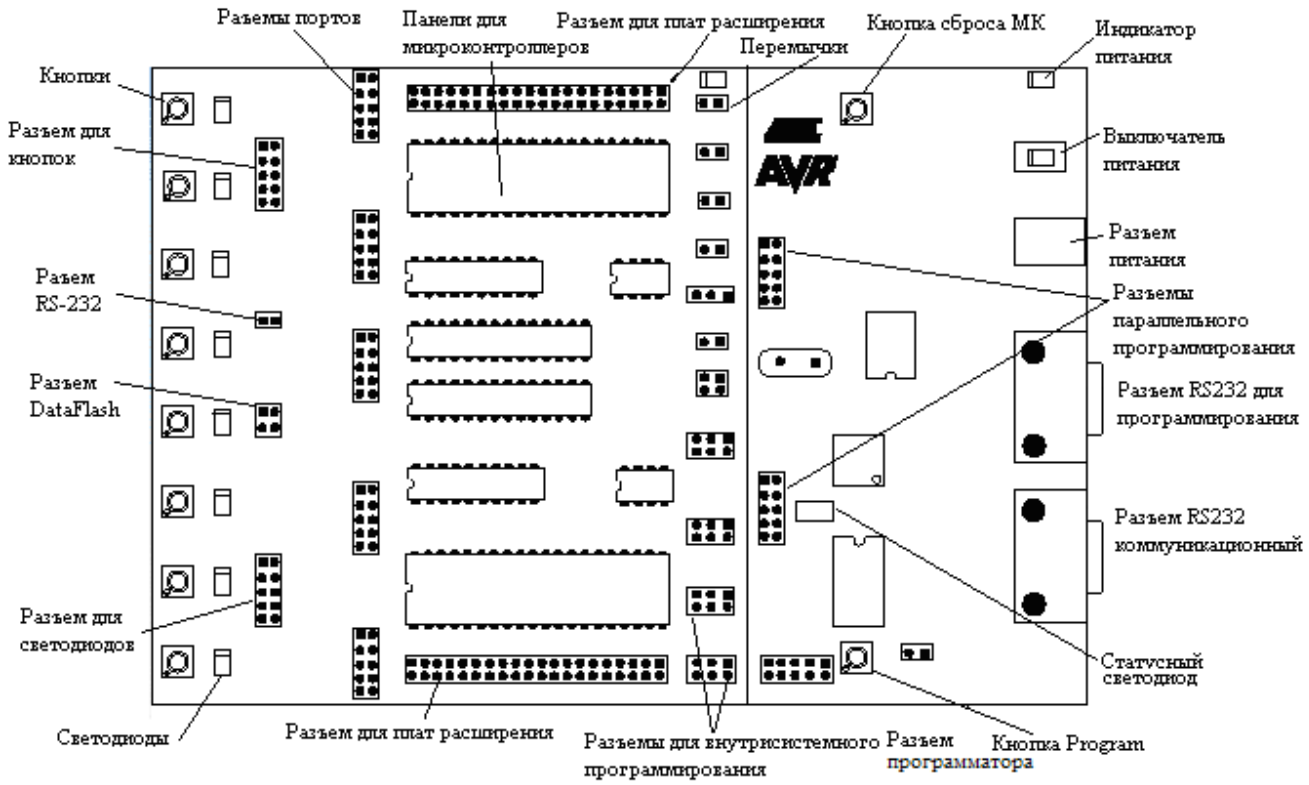
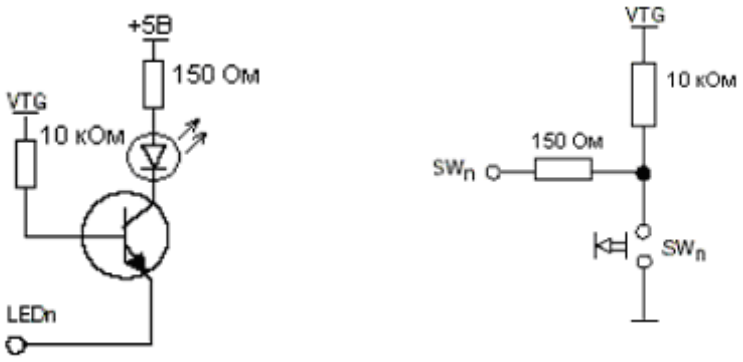


Рис. 6. Компоненты STK500

Светодиоды и кнопки общего назначения. В набор STK500 входят 8 желтых светодиодов и 8 кнопок без фиксации. Светодиоды и кнопки электрически отделены от остальной части платы и подключены к собственным разъемам. Они могут быть подключены к AVR-микроконтроллерам 10-проводными шлейфами через разъемы портов ввода-вывода.

Схема одного разряда индикации изображена на рис. 7,а. При поступлении на вывод LEDn сигнала с низким уровнем напряжения (логический «0») светодиод светится, сигнала с высоким уровнем напряжения (логическая «1») – светодиод гаснет.



а) б)

Рис.7. Схема включения светодиода а) и кнопки б)

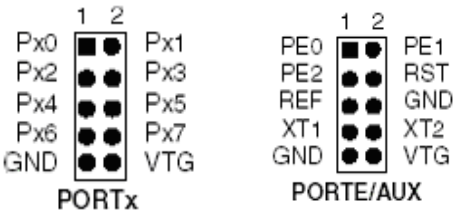
Схема подключения кнопки изображена на рис. 7,б. При нажатии на кнопку - на выводе SWn низкий уровень напряжения (GND), а при отпускании - высокий (VTG). Рабочий диапазон напряжения VTG =1.8…6.0 В.

Выводы светодиодов LEDx и кнопок SWx (x=0÷7) соединены с соответствующими контактами разъемов SWITCHS и LEDS. Следует иметь в виду, что контакты разъемов 8 и 9 использованы для сигналов GND и VTG. Поэтому необходимо соблюдать соединение шлейфами одноименных выводов разъемов индикаторов и кнопок с портами микроконтроллеров (шлейф не должен перекручиваться). С этой целью в шлейфе красным цветом выделена одна из линий, которая должна соединять одноименные выводы разъемов (например, выводы с номером 1).

Особенности работы светодиодов и кнопок необходимо учитывать при программировании портовых операций микроконтроллеров, связанных с обращением к светодиодам и кнопкам.

Разъемы портов. Любой порт ввода-вывода AVR-микроконтроллера может быть подключен к светодиодам и кнопкам с помощью 10-проводного шлейфа. На разъемы в дополнение к линиям портов выведены напряжение питания целевого микроконтроллера VTG (VCC) и общий провод GND.

Расположение выводов разъемов и их соответствие линиям портов ввода-вывода показано на рис. 8. Вывод с квадратной маркировкой указывает на вывод 1. Разъем порта E (PORTE/AUX) содержит специальные сигналы и функции в дополнение к линиям порта PE. Расположение и назначение выводов этого разъема показано на рис. 8,б.



а) б)

Рис. 8. Выводы разъемов портов Px (x=A, B, C, D) и PE микроконтроллера

Интерфейс RS-232 для пользователя. STK500 содержит два порта RS-232. Один порт RS-232 используется для связи с AVR Studio. Другой порт RS-232 можно использовать для связи микроконтроллера, установленного на плате, с компьютером через его последовательный порт RS-232 (COM-порт). Для этого два вывода канала UART микроконтроллера необходимо физически соединить с входами порта RS-232, выведенными на 2-штырьковый разъем “RS232 SPARE”. Порт RS-232 на плате STK содержит схему преобразования уровней сигналов с интерфейсом.

Flash-память данных DataFlash. В состав платы STK500 входит микросхема Flash-памяти AT45D021 емкостью 2 Мбит из семейства DataFlash, которая может быть использована для энергонезависимого хранения данных. DataFlash – флэш-память с последовательным программированием через SPI-интерфейс и может быть подключена к линиям порта PB микроконтроллера. Для этого необходимо использовать 4–штырьковый разъем с маркировкой “DATAFLASH”, который связан с SPI-интерфейсом DataFlash. Для соединения разъема “DATAFLASH” с линиями порта PB необходимо соединить PB6 - SO, PB7 - SCK, PB4 - /CS, PB5 - SI.

Секция целевых панелей. Модуль программирования состоит из 8 панелей в центре платы. В одну из них необходимо установить целевой микроконтроллер для программирования и дальнейшего использования в приложении. Для флэш-памяти микроконтроллеров гарантированная износостойкость составляет 10 000 циклов программирования.

Секции перемычек и программирования. Управляющий микроконтроллер и 8 перемычек определяют работу данного стартового набора. В обычном применении эти перемычки должны быть установлены в исходном состоянии.

После установки микроконтроллера в панель может быть выполнено программирование, для чего необходимо использовать AVR Studio 4 и один из предлагаемых методов:

- внутрисистемное программирование при нормальном напряжении питания. Этот метод используется как основной при проведении лабораторных работ;

- программирование повышенным напряжением, при котором напряжение питания всегда равно 5 В. Допускается подключение цепей VTARGET, RESET, XTAL1 и AREF к секции панелей.

Подробное описание каждого из методов программирования приведено в фирменном руководстве.

Прочие аппаратные компоненты. STK500 имеет два разъема расширения, установленные по обе стороны от секции целевых панелей. Все порты ввода-вывода микроконтроллера, сигналы программирования и управляющие сигналы присутствуют на выводах этих разъемов. Разъемы расширения позволяют легко подключить макеты приложений к STK500.

STK500 имеет 2 кнопки специального назначения и 3 светодиода для индикации состояния.

Нажатие на кнопку “RESET” приводит к сбросу целевого микроконтроллера.

Новые версии AVR Studio 4 способны обновить программу управляющего микроконтроллера STK500. При обнаружении старой версии программы STK500 AVR Studio обновит флэш-память управляющего микроконтроллера. Для выполнения этой функции пользователю необходимо нажать на кнопку PROGRAM после подачи напряжения питания на STK500.

Основной индикатор напряжения питания - красный светодиод, непосредственно подключенный к основному источнику питания STK500. Данный индикатор должен непрерывно светиться после подачи питания на STK500 переключателем POWER.

Индикатор целевого напряжения - светодиод, связанный с линией питания VCC (VTG) целевого микроконтроллера. Индикатор непрерывно светится при наличии напряжения питания на целевых панелях микроконтроллеров.

Статусный светодиод – 3-цветный. При программировании он имеет желтый цвет. После успешного завершения программирования он становится зеленым. Красный цвет показывает, что программирование было прервано. При программировании статусный светодиод последовательно изменяет свое состояние от красного через желтый к зеленому для индикации готовности целевого микроконтроллера.

4. Интерфейс STK500 в AVR Studio 4 и программирование микроконтроллера в STK500

Интерфейс STK500 в AVR Studio 4

В качестве программного приложения, используемого для связи с платой STK500, используется AVR Studio 4. Выполнение команды *Tools/Program AVR/Connect/STK500* из меню AVR Studio приводит к открытию окна пользовательского интерфейса *STK500*, показанного на рис. 9.

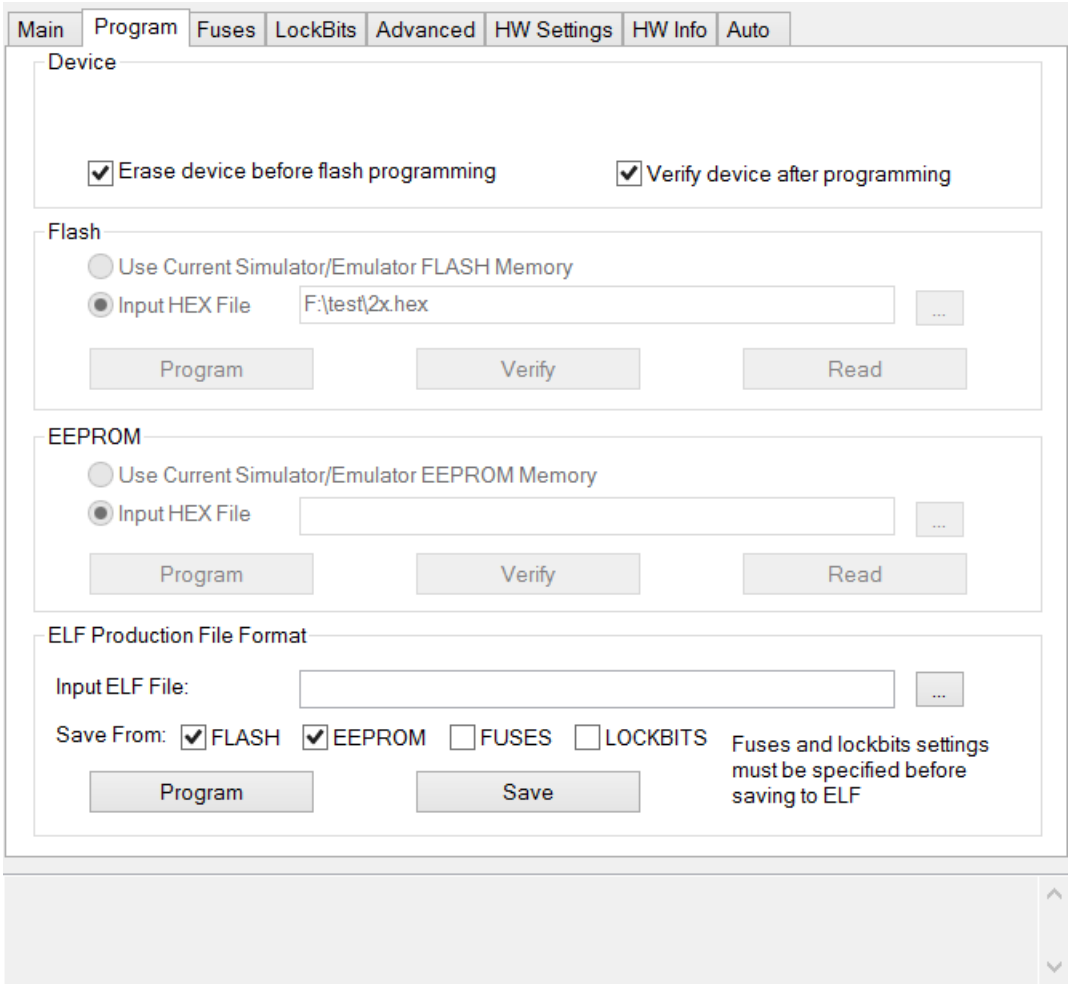


Рис. 9. Окно программирования пользовательского интерфейса STK500

Пользовательский интерфейс STK500 выполняет функции управления платой STK500. Доступные настройки разделены на восемь окон, каждое из которых открывается щелчком на соответствующей вкладке. В зависимости от выбранного типа микроконтроллера определяется доступный набор функций для выбора и установки. Недоступные функции окрашиваются серым цветом.

Установки в окне Main позволяют выбрать тип микроконтроллера, нажав кнопку стирания *Erase Device* осуществить стирание памяти микроконтроллера (Flash и EEPROM), проверить сигнатуру программируемого микроконтроллера, задать скорость передачи по интерфейсу внутрисистемного программирования ISP.

Нажатие на кнопку считывания сигнатуры *Read Signature* приводит к считыванию из микроконтроллера и отображению сигнатурных байтов. Сигнатурные байты используются для идентификации микросхемы и ее производителя. После считывания сигнатуры программа проверяет ее на соответствие выбранному типу микроконтроллера.

Кнопка *Settings* позволяет открыть окно для настройки скорости обмена информацией с микроконтроллером при последовательном внутрисхемном программировании. Необходимо, чтобы частота для ISP-программирования была минимум в 4 раза меньше тактовой частоты контроллера. Кнопка *Read* позволяет считать значение текущей частоты работы программатора, кнопка *Write* произвести запись в программатор нового значения частоты.

Установки окна Program (программирование) показаны на рис.9. В поле программируемого устройства Device помимо общего стирания памяти доступны две опции. Установка флажка *Erase device before flash programming* активизирует полезную функцию стирания памяти программ перед программированием, а при установке флажка *Verify device after programming* STK500 будет выполняться проверка правильности записанной информации не только во флэш-память, но и в EEPROM.

Выбрав в поле программирования памяти Flash опцию *Input HEX File* (входной hex-файл) и нажав кнопку справа можно указать путь к файлу и его имя. Файл должен быть создан в формате Intel-hex. Аналогично выбрав в поле программирования EEPROM опцию *Input HEX File* и нажав кнопку справа можно указать путь к файлу и его имя. Файл должен быть создан в формате Intel-hex. С помощью кнопок *Program, Verify и Read* выполняют соответствующие функции программирования, проверки и считывания.

Последняя группа опций содержит средства работы с файлами в формате .elf, которые могут содержать одновременно данные для памяти программ и для памяти данных, а так же биты защиты и конфигурации, поэтому программирование обеих областей осуществляется сразу.

Поле истории находится внизу окна пользовательского интерфейса *STK500*. В нем отображаются сообщения после выполнения каждой операции. Успешное выполнение операции заканчивается выводом *OK*.

Окно Fuses установки битов конфигурации. В окне *Fuses* представлены доступные для выбранного типа микроконтроллера конфигурационные биты. Детальная информация о доступных конфигурационных битах в различных режимах программирования, а также их назначении приведена в документации на соответствующий микроконтроллер.

Окно LockBits установки битов защиты программы. Окно *LockBits* показывает, какие режимы защиты программы доступны для выбора при заданном типе микроконтроллера. Все биты защиты доступны как в режиме ISP-программирования, так и в режиме программирования повышенным напряжением. Режим защиты задается комбинацией нескольких битов защиты.

Окно прочих установок Advanced. В окне *Advanced* представлено поле калибровочного байта генератора Oscillator Calibration Byte. Калибровочный байт записывается в микроконтроллер на стадии производства и, поэтому, доступен только для чтения. Данное значение используется в

программе для записи в регистр OSCCAL для подстройки номинальной частоты встроенного RC-генератора.

Считывание калибровочного байта. Нажатие на кнопку *Read* приводит к отображению на экране его значения в текстовом поле Value. Если данная опция выделена серым цветом, то это означает, что в выбранном микроконтроллере нет встроенного подстраиваемого RC-генератора.

Запись калибровочного байта. Поскольку значение калибровочного байта невозможно определить автоматически при выполнении программы, пользователь должен вручную записать его, предварительно указав адрес во флэш-памяти или EEPROM. Адрес задается в текстовом поле Address. С помощью переключателя *Flash , Eeprom* выбирается получатель данных, а затем нажимается кнопка *Write* для записи калибровочного байта по указанному адресу.

Окно настроек платы HW Settings. В окне *HW Settings* (рис. 10) можно изменить рабочие напряжения VTARGET, AREF и частоту генератора на плате STK500.

Информация о рекомендуемых рабочих условиях приведена в документации для каждого типа микроконтроллера. При их настройке следует иметь в виду, что выход за указанные пределы может вывести используемый микроконтроллер из строя.

Поле настроек генератора Clock Generator. Плата STK500 использует схему программируемого генератора, которая формирует широкий диапазон тактовых частот для целевых микроконтроллеров. Ввиду того, что генерировать сигнал с произвольно заданным значением частоты невозможно, пользовательский интерфейс STK500 позволяет вычислить ближайшую доступную частоту при введении желаемой, что отображается в текстовом поле. Из раскрывающегося списка можно выбрать фиксированные частоты, например 20 кГц, … 1,23 МГц, 1,84 МГц, 3,69 МГц (максимальная частота) или вообще остановить генератор (stopped). Чтение/запись частот генератора осуществляется нажатием соответствующих кнопок *Read* и *Write*.

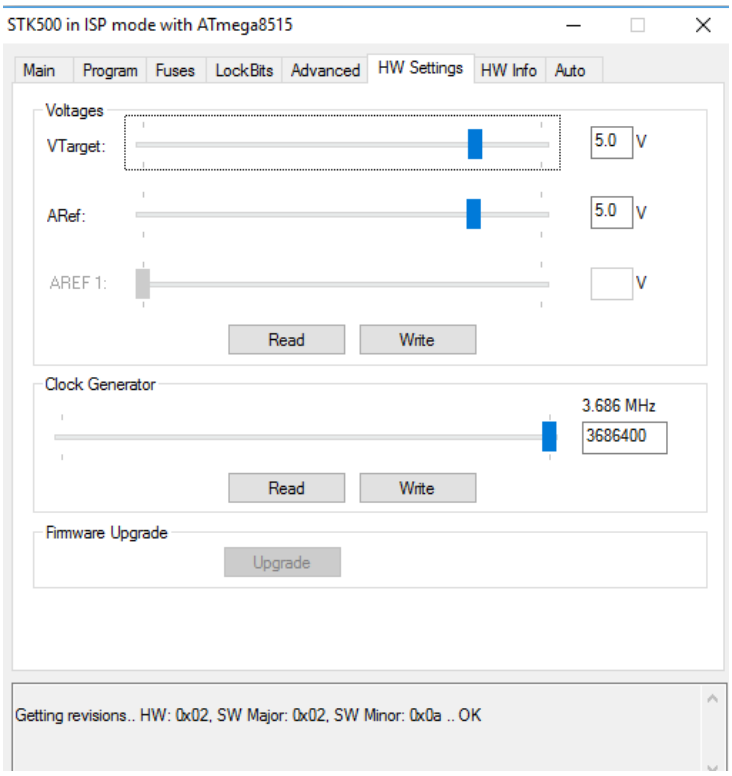


Рис. 10. Окно управления платой

Окно управления функциями автоматизации Auto. При программировании нескольких микроконтроллеров одним и тем же программным кодом функция Auto выступает в качество полезного инструмента по автоматизации последовательности действий при программировании. Действия представлены в виде списка в порядке очередности их выполнения при активизации.

Настройка функции автопрограммирования выполняется путем указания действий, желаемых для исполнения STK500. Для разрешения выполнения действия его необходимо пометить флажком. Например, если отмечено только действие Program FLASH , то после нажатия кнопки *Start* будет произведена запись в память Flash в соответствии с настройками в окне *Program*. Все действия используют настройки в пределах пользовательского интерфейса STK500.

Проверка выполненных действий может быть осуществлена благодаря использованию функции записи в журнал Log to file, которая все действия записывает в текстовый файл. Выбрать или создать файл можно нажатием кнопки *Browse* и в дальнейшем указать путь и имя имеющегося или создаваемого файла. Последовательность выполняемых действий будет фиксироваться в указанном файле, который в дальнейшем можно просмотреть.

Программирование микроконтроллера в STK500

Для программирования целевого микроконтроллера необходимо предварительно соединить плату STK500 с источником питания и компьютером.

1. Подключить источник постоянного напряжения 10 …15 В, 500 мА.

2. Выключив компьютер, соединить кабелем персональный компьютер и плату через разъем “RS232 CTRL”. (Примечание - При подключении платы STK500 к порту USB необходимо при установке AVR Studio 4 установить драйвер USB-RS232).

3. Для программирования ATx8515 соединить 6-проводным шлейфом разъемы ISP6PIN и SPROG3.

4. Переключателем питания POWER можно включить или отключить STK500. Свечение красного светодиода сигнализирует о подаче питания. Состояние статусного светодиода изменяется от красного к желтому, а затем к зеленому. Зеленый цвет светодиода сигнализирует о наличии напряжения VCC (питание микроконтроллера). Стартовый набор может настраиваться на различные частоты тактирования и источники питания.

Дальнейшие действия выполнить в окне программы AVR Studio 4.

1. Создать проект для микроконтроллера, установленного на плате STK500. Загрузив рабочую программу в окно редактора, выполнить команду *Build* из меню *Project*. После отладки программы с помощью встроенного симулятора можно перейти к программированию микроконтроллера.

2. Для того, чтобы запрограммировать hex-файл в AVR-микроконтроллер, необходимо выполнить команду *Tools/Program AVR/Connect/STK500* из меню программы AVR Studio 4. После открытия окна выбрать тип микроконтроллера из раскрывающегося списка на вкладе *Program* и указать путь к записываемому Intel-hex файлу в поле *Input HEX File*. Нажать кнопку Program. Статусный светодиод на плате STK500 во время программирования светится желтым цветом, а после успешного завершения загорается зеленым цветом. При выявлении ошибки программирования светодиод загорается красным цветом.

В случае ошибки детектирования устройства рекомендуется, выключив питание на плате, разъединить установленные проводные связи, повторить программирование, затем снова при выключенном питании аккуратно выполнить необходимые соединения. Включив питание и убедившись, что статусный светодиод горит зеленым цветом, можно приступить к работе.