ПРОГРАММИРОВАНИЕ МИКРОКОНТРОЛЛЕРОВ AVR и PIC

(www.simpleprogramming.ru)

ОГЛАВЛЕНИЕ

Введение…… 4

Глава 1. Программные и аппаратные средства программирования микроконтроллеров……... 20

1.1. Общие сведения……. 20

1.2. Операции с портами микроконтроллера……. 22

1.3. Интегрированная среда разработки Microchip Studio для AVR…… 30

1.3.1. Установка Microchip Studio……... 30

1.3.2. Создание проекта "Линейка светодиодов" в Microchip Studio…….. 35

1.3.2.1. Начало работы…….. 35

1.3.2.2. Разработка листинга программы…… 38

1.4. Интегрированная среда разработки MPLAB X IDE для PIC……. 43

1.4.1. Установка MPLAB X IDE и компиляторов…….. 43

1.4.2. Пакеты семейства устройств DFP……. 48

1.4.3. Интерфейсное окно пользователя……. 54

1.4.4. Биты конфигурации…… 65

1.4.5. Создание проекта "Линейка светодиодов" в MPLAB X IDE. 75

1.4.5.1. Начало работы……. 75

1.4.5.2. Разработка листинга программы…… 79

1.5. Программа Proteus для моделирования электронных схем……... 84

1.5.1. Интерфейсное окно Proteus……... 84

1.5.2. Моделирование электронных схем с микроконтроллерами в Proteus……. 92

1.5.2.1. Моделирование проекта с микроконтроллером AVR.. 92

1.5.2.2. Моделирование проекта с микроконтроллером PIC… 98

1.5.2.3. Моделирование проекта "Матрица светодиодов" с микроконтроллером PIC…….. 99

1.5.3. Создание листинга программы внутри Proteus без привлечения IDE……... 103
1.5.4. Использование виртуальных измерительных приборов в Proteus…… 108
1.6. Аппаратные средства программирования микроконтроллеров…… 112
Контрольные вопросы и задания…… 120
Глава 2. Примеры программирования 8-разрядных микроконтроллеров AVR…….. 122
2.1. Подключение кнопки к микроконтроллеру…… 122
2.1.1. Создание проекта…… 122
2.1.2. Моделирование проекта на макетной плате…… 124
2.2. Подключение к микроконтроллеру одноразрядного семисегментного индикатора…….. 128
2.3. Управление таймер-счетчиком…… 132
2.4. Управление внешними прерываниями…… 136
Контрольные вопросы и задания…… 142
Глава 3. Примеры программирования 8-разрядных микроконтроллеров PIC……... 144
3.1. Подключение к микроконтроллеру двух кнопок……... 144
3.2. Подключение к микроконтроллеру силовой нагрузки…….. 153
3.3. Конфигуратор кода MPLAB Code Configurator (MCC). Создание проекта "Программирование ШИМ"…….. 156
3.4. Управление вращением электродвигателя небольшой мощности... 171
3.5. Работа с последовательным интерфейсом RS-232……. 175
3.6. Работа с модулем АЦП микроконтроллера…… 182
3.6.1. Создание проекта…… 182
3.6.2. Моделирование на макетной плате……... 192
3.7. Подключение к микроконтроллеру жидкокристаллического индикатора…… 197
3.7.1. Создание проекта…… 197

3.7.2. Моделирование проекта на макетной плате…… 201
3.8. Подключение к микроконтроллеру температурного датчика…….. 203
3.9. Подключение к микроконтроллеру малогабаритного шагового электродвигателя…….. 208
3.9.1. Создание проекта…… 208
3.9.2. Моделирование проекта на макетной плате…… 219
Контрольные вопросы и задания…… 221
4. Решения и ответы……... 223
4.1. Решения и ответы к главе 1…….. 223
4.2. Решения и ответы к главе 2…….. 228
4.3. Решения и ответы к главе 3…….. 235
Литература…….. 240

Введение

Микроконтроллер (МК, англ. MCU – Microcontroller Unit) – это электронный блок для управления различными устройствами, выполненный в виде единой микросхемы, или совсем коротко – это управляющая микросхема. В состав микроконтроллера входят микропроцессор, постоянное запоминающее устройство (ПЗУ, англ. ROM), оперативное запоминающее устройство (ОЗУ, англ. RAM), порты ввода/вывода, таймеры, АЦП, последовательные и параллельные интерфейсы и др. Учитывая количество входящих в состав микроконтроллера компонентов, его иногда называют однокристальным компьютером. Вычислительная мощность у микроконтроллера неизмеримо меньше, чем у компьютера, поэтому они используются для решения узкоспециализированных задач и находят широкое применение в устройствах промышленной автоматики, в системах автоматизации зданий, в периферийных устройствах компьютерной техники, в мобильной электронике, робототехнике, автомобильной технике и многих других приложениях.

В данном учебном пособии рассмотрено программирование микроконтроллеров AVR и PIC компании Microchip. С сайта компании можно скачать бесплатное программное обеспечение для работы с этими микроконтроллерами.

Microchip Technology Inc. (www.microchip.com) – американская компания. Продукция компании включает в себя микроконтроллеры (PIC , dsPIC , AVR и SAM), а также устройства памяти EEPROM, устройства Serial SRAM, встроенные устройства безопасности, радиочастотные (RF) устройства, аналоговые управляющие устройства, линейные, интерфейсные и беспроводные решения. Микроконтроллеры компании Microchip включают в себя 8, 16 и 32-разрядные модели микроконтроллеров. Не углубляясь в детали, приведем следующее определение разрядности: под **разрядностью** понимается количество бит информации, которые обрабатываются за один такт микропроцессора. Обзор микроконтроллеров компании Microchip начнем с семейства 8-разрядных микроконтроллеров.

8-разрядные микроконтроллеры:

- 8-разрядные микроконтроллеры AVR,
- 8-разрядные микроконтроллеры PIC,

8-разрядные **микроконтроллеры AVR** подразделяются на семейства:

- **ATtiny:**
  - Флеш-память до 16 КБ; SRAM до 512 Б; EEPROM до 512 Б;
  - Число линий ввода-вывода 4-18 (общее количество выводов 6-32);
  - Ограниченный набор периферийных устройств.
- **ATmega:**
  - Флеш-память до 256 КБ; SRAM до 16 КБ; EEPROM до 4 КБ;
  - Число линий ввода-вывода 23-86 (общее количество выводов 28-100);
  - Аппаратный умножитель;
  - Расширенная система команд и периферийных устройств.
- **ATxmega:**
  - Флеш-память до 384 КБ; SRAM до 32 КБ; EEPROM до 4 КБ;
  - Четырёхканальный DMA-контроллер;
  - Инновационная система обработки событий.

Буквы в наименовании модели микроконтроллера (после дефиса) означают вариант корпуса (табл.1.1):

Таблица 1.1

ATxxx-**P** — корпус DIP	ATxxx-**MA** — корпус UDFN/USON
ATxxx-**A** — корпус TQFP	ATxxx-**C** — корпус CBGA
ATxxx-**J** — корпус PLCC	ATxxx-**CK** — корпус LGA
ATxxx-**M** — корпус MLF	ATxxx-**T** — корпус TSOP
ATxxx-**S** — корпус EIAJ SOIC	ATxxx-**X** — корпус TSSOP
ATxxx-**SS** — узкий корпус JEDEC SOIC	ATxxx-**TS** — корпус SOT-23 (ATtiny4/5/9/10)

Современная линейка микроконтроллеров AVR дополнена продуктами: AVR DD, AVR DB, AVR DA.

Продукты AVR DD имеют низкое энергопотребление и идеально подходят для сложных приложений или в качестве сопутствующего микроконтроллера в сложных конструкциях с несколькими уровнями питания. Семейство AVR DD имеет широкий набор функций, включая многовольтный ввод/вывод (MVIO), позволяющий обеспечивать двунаправленную связь с устройствами с разным уровнем питания. Гибкость семейства делает его пригодным для применения во встраиваемых системах, осуществляющих управление в режиме реального времени. Микроконтроллеры используются в промышленных системах, бытовой технике, автомобилестроении, IoT (Internet of Things – интернет вещей) и других приложениях. (Интернет вещей – это приложения, обменивающиеся информацией через интернет).

Основные характеристики:

- Встроенный генератор 24 МГц.
- До 8 КБ SRAM.
- 12-разрядный дифференциальный аналого-цифровой преобразователь (АЦП).
- 10-разрядный цифроаналоговый преобразователь (ЦАП).
- Аналоговый компаратор (АС) с масштабируемым опорным входом.
- Обнаружение сбоя часов.
- 16-разрядные часы реального времени (RTC) и таймер прерывания.
- Настраиваемое периферийное устройство с пользовательской логикой (CCL)
- Настраиваемый внутренний генератор опорного напряжения.
- USART / SPI двухрежимный двухпроводной интерфейс (TWI).
- Многовольтный ввод-вывод на порту C.

В семействе AVR DD имеются микроконтроллеры с различным числом контактов и различным объемом флэш-памяти (табл.2.1).

Таблица 2.1

Флэш-память, Кб	14 контактов	20 контактов	28 контактов	32 контакта
64	AVR64DD14	AVR64DD20	AVR64DD28	AVR64DD32
32	AVR32DD14	AVR32DD20	AVR32DD28	AVR32DD32
16	AVR16DD14	AVR16DD20	AVR16DD28	AVR16DD32

Продукты AVR DB разработаны для сложных аналоговых приложений с низким энергопотреблением. Благодаря трем встроенным операционным усилителям, не имеют аналогов по своим возможностям формирования сигнала. Продукты AVR DB имеют многовольтный ввод-вывод (MVIO) и являются идеальными вспомогательными микроконтроллерами в сложных конструкциях с несколькими уровнями питания. Семейство AVR DB можно использовать для функций управления в режиме реального времени в системах промышленного управления, бытовой технике, автомобилестроении, интернете вещей (IoT), для емкостных сенсорных пользовательских интерфейсов и в других приложениях.

Основные характеристики:

- Встроенный генератор 24 МГц.
- До 16 КБ SRAM.
- 12-разрядный дифференциальный аналого-цифровой преобразователь (АЦП).
- 10-разрядный цифроаналоговый преобразователь (ЦАП).
- Формирование аналогового сигнала (3 операционных усилителя).
- Аналоговый компаратор (АС) с масштабируемым опорным входом.
- Сканирование с циклической проверкой избыточности (CRC).
- Обнаружение сбоя часов.
- 16-разрядные часы реального времени (RTC) и таймер периодического прерывания.
- Настраиваемое периферийное устройство пользовательской логики (CCL).
- USART/SPI/двухрежимный двухпроводный интерфейс (TWI).

- Многовольтный ввод-вывод на порту C.
- Допускаемые напряжения 1,8 В могут использоваться для всех входных контактов.

В семействе AVR DB имеются микроконтроллеры с различным числом контактов и различным объемом флэш-памяти (табл. 3.1).

Таблица 3.1

Флэш-память, Кб	28 контактов	32 контакта	48 контактов	64 контакта	Статус
128	AVR128DB28	AVR128DB32	AVR128DB48	AVR128DB64	В производстве
64	AVR64DB28	AVR64DB32	AVR64DB48		Будущий проект
32	AVR32DB28	AVR32DB32	AVR32DB48		Будущий проект

Продукты AVR DA разработаны для приложений, выполняющих управление в режиме реального времени, и использующих устройства HMI с емкостной технологией сенсорного управления. Высокая плотность памяти делает микроконтроллеры подходящими как для проводной, так и беспроводной связи. В семействе используются новейшие чипы CIPs с низким энергопотреблением и напряжением 5 В для повышения помехоустойчивости. Настраиваемая пользовательская логика (CCL) периферийных устройств, наряду с интеллектуальными аналоговыми периферийными устройствами, делают семейство AVR DA идеальным решением для управления в реальном режиме времени. Микроконтроллеры используются в промышленных системах, бытовой технике, автомобилестроении, интернете вещей (IoT), в приложениях с функциями сенсорного управления.

Основные характеристики:

- Встроенный генератор с частотой 24 МГц.
- До 16 Кб SRAM.
- 12-разрядный дифференциальный аналого-цифровой преобразователь (АЦП).

- 10-разрядный цифроаналоговый преобразователь (ЦАП).
- Аналоговый компаратор с масштабируемым опорным входом.
- 16-разрядные часы реального времени (RTC) и таймер периодического прерывания.
- Настраиваемое периферийное устройство с пользовательской логикой (CCL).
- Настраиваемый внутренний генератор опорного напряжения.
- USART/SPI двухрежимный двухпроводной интерфейс (TWI).
- До трех детекторов пересечения нуля (ZCD).
- Сканирование с циклической проверкой избыточности (CRC).

В семействе AVR DA имеются микроконтроллеры с различным числом контактов и различным объемом флэш-памяти (табл. 4.1).

Таблица 4.1

Флэш-память, Кб	28 контактов	32 контакта	48 контактов	64 контакта
128	AVR128DA28	AVR128DA32	AVR128DA48	AVR128DA64
64	AVR64DA28	AVR64DA32	AVR64DA48	AVR64DA64
32	AVR32DA28	AVR32DA32	AVR32DA48	

8-разрядные микроконтроллеры PIC подразделяются на три уровня:

- базовый уровень: младшие модели PIC10, PIC12, PIC16;
- средний уровень: старшие модели PIC10, PIC12, PIC16;
- повышенный уровень PIC18.

Микроконтроллеры PIC10 имеют простой набор инструкций, обеспечивающих удобство применения и быструю разработку приложений. Они имеют самый малый форм-фактор и могут применяться в конструкциях с ограниченными размерами. Это семейство микроконтроллеров включает в себя внутренний генератор, компаратор, АЦП, память данных.

Микроконтроллеры PIC12, PIC16 подходят для решения задач, требующих более сложных алгоритмов управления и большего объема памяти. Эти микроконтроллеры оснащены различными аналоговыми и цифровыми периферийными устройствами: SPI, I2C, USART, LCD, АЦП.

Микроконтроллеры отлично подходят для устройств с малым потреблением энергии, работающих от батарей.

Микроконтроллеры с архитектурой PIC18 появились в 2000 году. Эта серия оказалась очень популярной. В отличие от более ранних устройств, которые чаще всего программировались на ассемблере, для PIC18 и более поздних устройств язык С стал преобладающим языком разработки. Микроконтроллеры PIC18 сочетают в себе максимальный уровень производительности с простотой использования 8-битной архитектуры, обладают наибольшим количеством выводов и достаточно большим объемом памяти, включая до 128 Кбайт флэш-памяти и до 8 Кбайт оперативной памяти. Микроконтроллеры имеют широкую номенклатуру периферийных устройств: CAN, USB, Ethernet, сенсорные датчики и драйверы ЖК-дисплеев.

16-разрядные микроконтроллеры PIC:

- PIC24F (16 МГц),
- PIC24H (40 МГц),
- PIC24E (70 МГц).

Семейство dsPIC с цифровым сигнальным процессором:

- dsPIC33C (100 МГц),
- dsPIC33E (70 МГц),
- dsPIC33F (16-50 МГц),
- dsPIC30F (30 МГц).

В 2001 году компания Microchip представила серию микросхем dsPIC, которая поступила в массовое производство в конце 2004 года. Это были первые 16-разрядные микроконтроллеры Microchip. Устройства dsPIC дополнительно включают возможности цифровой обработки сигналов.

В отличие от dsPIC устройства PIC24 представляют собой микроконтроллеры общего назначения.

Семейства 16-разрядных микроконтроллеров PIC24 и dsPIC обеспечивают повышенную производительность, низкое энергопотребление, развитую периферию и полный набор программных и аппаратных средств,

представляющих собой законченный комплект разработчика. Семейство dsPIC обеспечивает высокую производительность в вычислительных операциях и идеально подходит для разработки сложных приложений, работающих в режиме реального времени. Микроконтроллеры могут работать с напряжением питания 3 В и 5 В и с расширенным температурным диапазоном применения до +150°C, имеют встроенные функции, повышающие надежность и обеспечивающие безопасное отключение в случае возникновения внештатных ситуаций. Благодаря развитой периферии, 16-разрядные устройства находят широкое применение в промышленности, автомобилестроении, в бытовых приборах.

32-разрядные микроконтроллеры Microchip подразделяются на:

- 32-разрядные микроконтроллеры семейства PIC,
- 32-разрядные микроконтроллеры семейства SAM.

Семейство 32-разрядных микроконтроллеров PIC начинается с **PIC32MX**, который компания Microchip представила в 2007 году. PIC32MX выполнен на основе ядра MIPS32 M4K. Устройство программируется с помощью компилятора Microchip MPLAB C, предназначенного для микроконтроллеров PIC32. Сегодня доступен полный спектр 32-разрядных микроконтроллеров PIC24 среднего уровня, начиная с 28-контактного разъема в малогабаритных корпусах QFN и заканчивая высокопроизводительными устройствами с Ethernet, CAN и USB OTG интерфейсами. OTG (On-The-Go) – спецификация интерфейса USB, которая позволяет USB-устройству выступать в качестве хоста USB. Архитектура PIC32 привнесла в микроконтроллеры ряд новых функций, в том числе:

- максимальная скорость выполнения – 80 MIPS (MIPS – Million Instruction Per Second. Не путать с понятием MIPS-архитектура);
- самая большая флэш-память – 512 Кб;
- одна инструкция на выполнение тактового цикла;
- первый кэшированный процессор;
- полный JTAG и 2-проводное программирование и отладка;
- трассировка в реальном времени.

В 2013 году Microchip представила серию микроконтроллеров **PIC32MZ,** основанных на ядре MIPS M14K. Серия PIC32MZ имеет следующие характеристики:

- тактовая частота ядра 252 МГц;
- до 2 Мб флэш-памяти и 512 Кб ОЗУ;
- новые периферийные устройства, включая высокоскоростной USB.

В 2015 году Microchip выпустила семейство **PIC32MZ EF** с использованием обновленного процессора MIPS M5150 Warrior M-класса. В следующем 2016 году Microchip представила семейство **PIC32MM,** предназначенное для маломощных и недорогих приложений. В микроконтроллерах PIC32MM используется процессор MIPS32 M4K. Микроконтроллеры семейства PIC32MM рассчитаны на низкое энергопотребление и ограничены тактовой частотой 25 МГц. Их ключевым преимуществом является поддержка 16-битных инструкций MIPS, что делает программу намного более компактной.

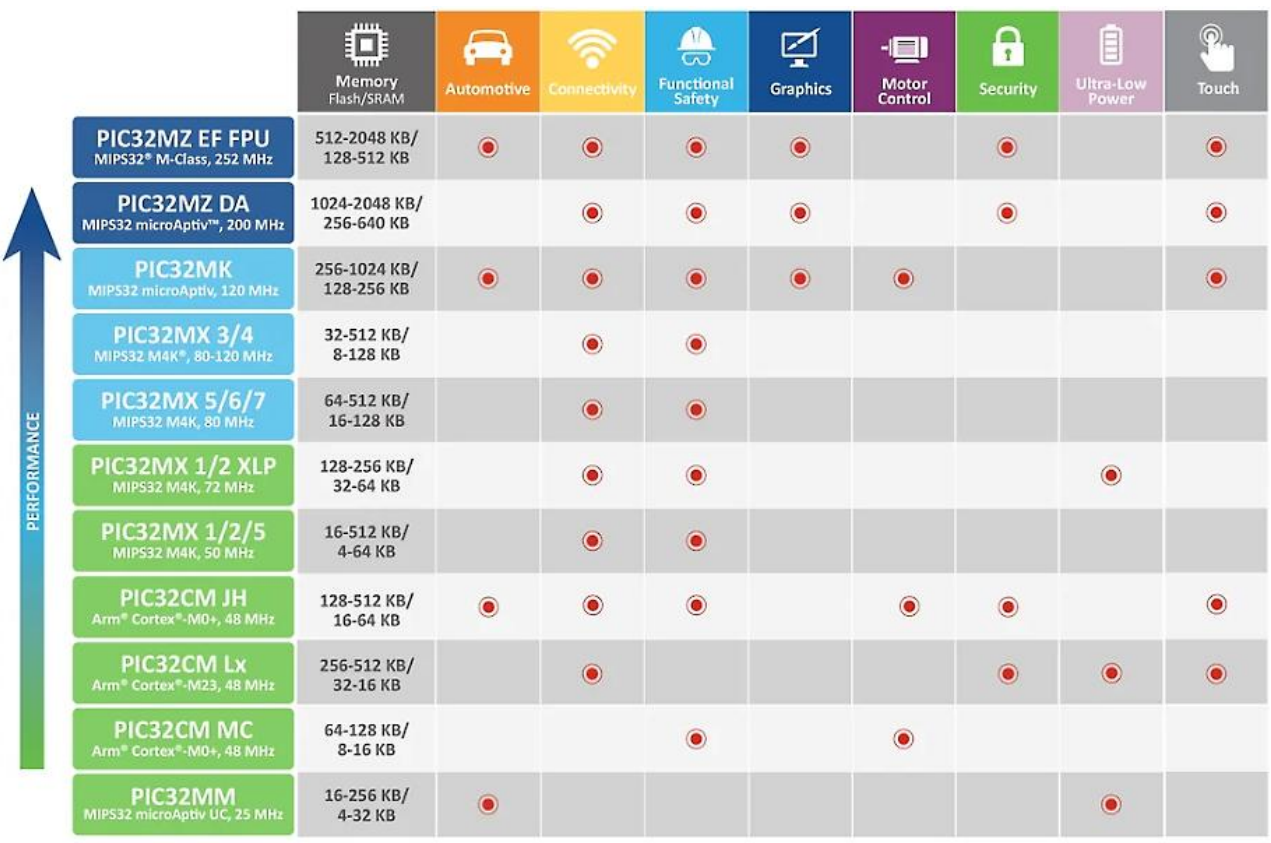


Рис. 1.1. 32-разрядные микроконтроллеры семейства PIC32

В 2017 году Microchip представила семейство **PIC32MZ DA**, включающее встроенный графический контроллер, графический процессор и 32 Мб DDR2 DRAM. В том же 2017 году Microchip представила семейство **PIC32MK**, специализирующееся на управлении двигателем и промышленном применении. Номенклатура микроконтроллеров PIC32 и их сферы применения показаны на рис. 1 (боковая стрелка показывает направление повышения производительности микроконтроллеров).

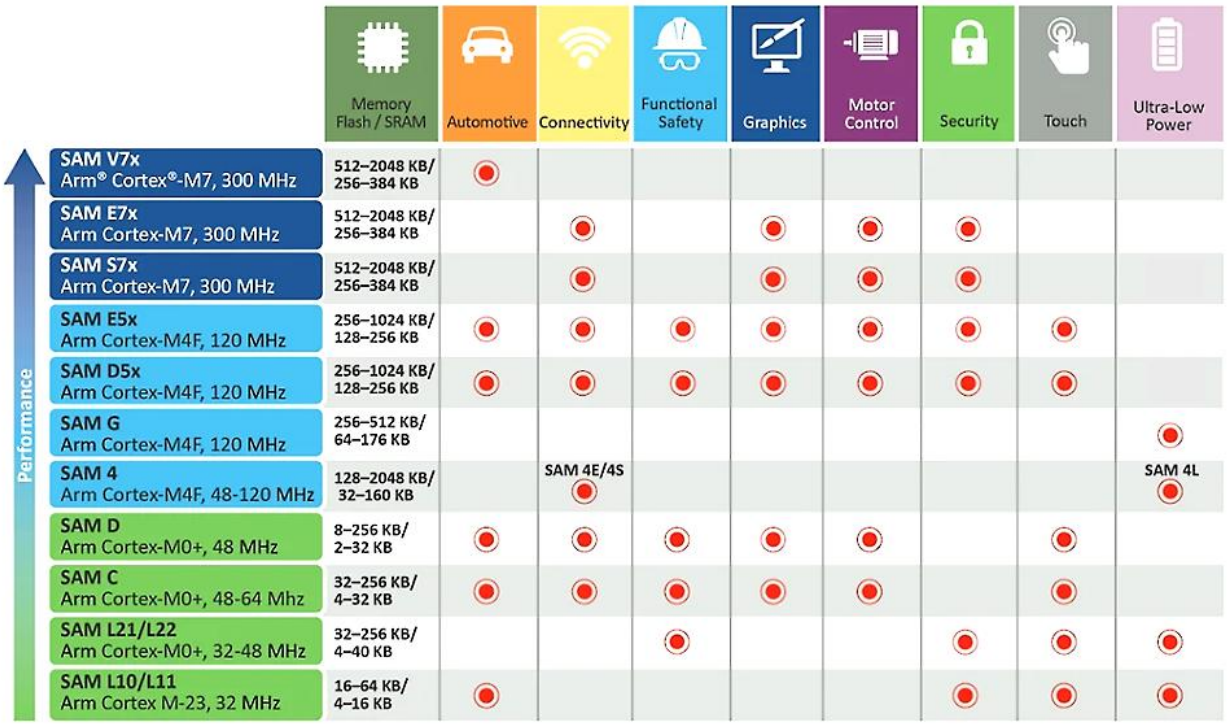


Рис. 1.2. 32-разрядные микроконтроллеры семейства SAM

32-разрядные микроконтроллеры семейства SAM. Эти микроконтроллеры оснащены процессорами с ядром ARM Cortex-M, имеют в своем составе множество передовых периферийных устройств, которые делают их отличным вариантом для разработки различных 32-разрядных приложений. Периферийные устройства имеют усовершенствованный сенсорный контроллер (PTC), низкое энергопотребление и большой набор высокопроизводительных аналоговых функций.

Номенклатура 32-разрядных микроконтроллеров SAM компании Microchip и их возможные сферы применения показаны на рис. 1.2.

Кратко рассмотрим особенности процессорных ядер серии Arm Cortex-M, применяемых в 32-разрядных микроконтроллерах SAM:

Arm® Cortex®-M3;
Arm® Cortex®-M0+;
Arm® Cortex®-M0;
Arm® Cortex®-M7;
Arm® Cortex®-M4;
Arm® Cortex®-M33.

Компания ARM (Advanced RISC Machines, www.arm.com) представила на рынок семейство Cortex-M в 2004 г., когда было анонсировано первое ядро этого семейства – Cortex-M3. В 2007 г. она представила ядро Cortex-M1, в 2009 г. – Cortex-M0, в 2010 г. – Cortex-M4, в 2012 г. – Cortex-M0+, в 2014 г. – Cortex-M7, в 2016 г. – Cortex-M23 и Cortex-M33, в 2020 г. – Cortex-M55, в 2022 г. – Cortex-M85. Разработки новых ядер Cortex-M продолжаются и новые модели появляются постоянно.

Cortex-M – это семейство физических ядер, предназначенных для дальнейшей интеграции с полупроводниковыми блоками, входящими в состав готового микроконтроллера. Состав и конструкцию готового микроконтроллера определяет и формирует производитель микроконтроллеров, в то время как состав и архитектуру ядер Cortex-M определяет проектировщик и разработчик архитектуры ядер – компания ARM.

Arm® Cortex®-M3. Ядро небольшого размера, выполненное по 90 нм технологии. Характеристики Arm Cortex®-M3:

- Архитектура Armv7-M.
- Набор команд Thumb/Thumb-2.
- Трехступенчатый конвейер.
- Динамическое энергопотребление от 10 до 150 мкВт/ МГц.
- Динамическая мощность ядра от 10 до 150 мкВт / МГц в зависимости от используемой технологии.

- Блок защиты памяти, который управляет доступом центрального процессора к памяти.

Микроконтроллеры с ядром Arm Cortex®-M3 отвечают требованиям приложений начального уровня по производительности, подходят для обработки сигналов в реальном времени, обеспечивают высокую масштабируемость.

Arm® Cortex®-M0+ является самым энергоэффективным ядром Arm®, доступным для встраиваемых приложений с ограничениями по конструкции, отличается одним из самых компактных размеров кремния и минимальным размером кода, что позволяет разработчикам достигать 32-разрядной производительности при стоимости микроконтроллера, как у 16 и 8-разрядных моделей. Характеристики Arm Cortex®-M0+:

- Архитектура Armv6-M.
- Поддержка набора команд Thumb / Thumb-2.
- Двухступенчатый конвейер.
- Дополнительный буфер микропроцессора.
- Динамическая мощность ядра от 5 до 50 мкВт/МГц в зависимости от используемой технологии.
- Блок защиты памяти.

Малое количество вентилей микропроцессора позволяет использовать его в приложениях, где требуются простые функции. Ядро Cortex®-M0+ имеет низкий уровень шума и удовлетворяет требованиям по производительности для микроконтроллеров начального уровня.

Arm® Cortex®-M0 – это самый маленький из доступных процессоров Cortex-M с малой площадью кремния, малым количеством вентилей, малым энергопотреблением и минимальным объемом кода. Подходит для устройств с аналоговым и смешанным сигналом, идеально подходит для встраиваемых приложений, имеет низкую стоимость, сравнимую со стоимостью 16- и 8-разрядных моделей микроконтроллеров. Выполнен по 90 нм технологии. Характеристики Arm® Cortex®-M0:

- Архитектура Armv6-M.

- Поддержка набора команд Thumb / Thumb-2.
- Трехступенчатый конвейер.
- Динамическая мощность ядра от 5 до 50 мкВт/МГц, в зависимости от используемой технологии.

На рис. 1.3 представлена блок-схема ядра.

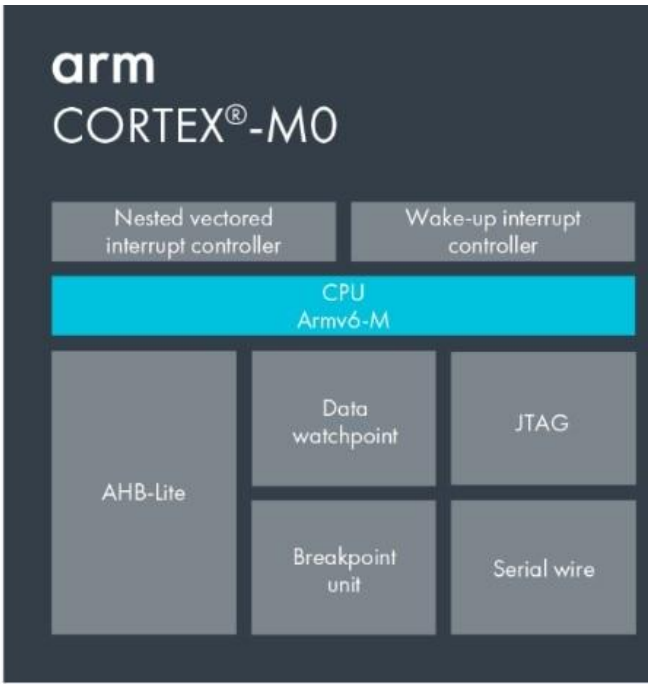


Рис. 1.3. Блок-схема Arm Cortex-M0

Arm® Cortex®-M7 обеспечивает лучшую производительность среди линейки Cortex-M. Ядро оснащено выделенными блоками цифровой обработки сигналов (DSP), включая дополнительный модуль с плавающей запятой (FPU). Высокопроизводительные функции ядра Arm Cortex-M7 идеально подходят для сложных приложений по управлению цифровыми сигналами. Типичными примерами применения являются: управление двигателем, управление питанием, алгоритмы машинного обучения и искусственного интеллекта, встроенное аудио, включая распознавание голоса, промышленная и домашняя автоматизация, интернет вещей.

На рис. 1.4 представлена блок-схема ядра.

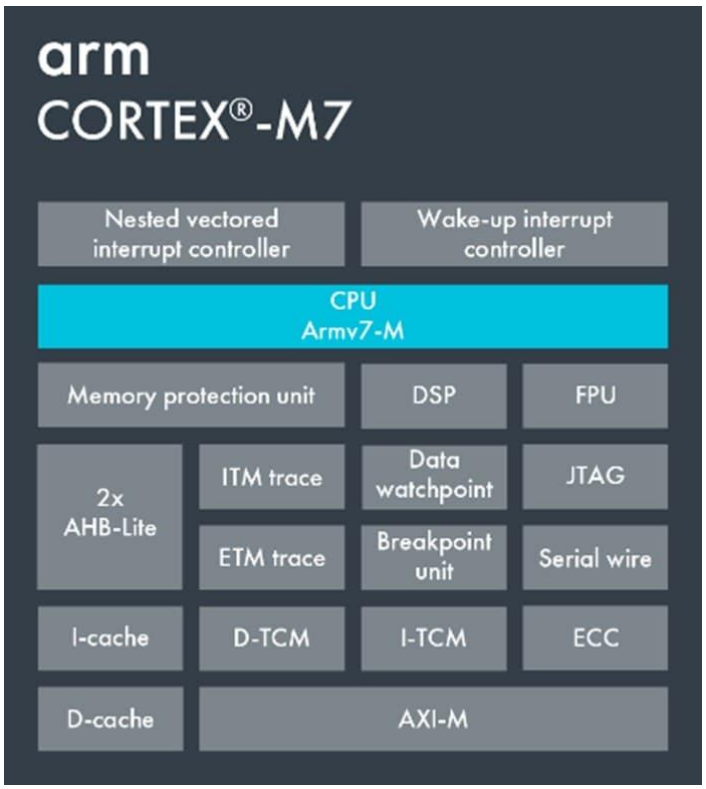


Рис. 1.4. Блок-схема Arm Cortex-M7

Характеристики Arm Cortex®-M7:

- Архитектура Armv7E-M.
- Поддержка набора команд Thumb / Thumb-2.
- Шестиступенчатый суперскалярный конвейер с предсказанием ветвлений.
- Кэш команд и кэш данных.
- Дополнительное расширение DSP для цифровой обработки сигналов с использованием математических вычислений, поддерживающее операции с плавающей запятой одинарной и двойной точности.
- Более высокая частота процессора.
- Обработка прерываний с нулевой задержкой.
- 64-разрядный интерфейс AMBA4 AXI.
- Блок защиты памяти.

Из сравнения рисунков 1.3 и 1.4 видно, насколько архитектура ядра Arm Cortex-M7 сложнее архитектуры ядра Arm Cortex-M0.

Arm® Cortex®-M4 является первым ядром линейки Cortex-M, оснащенным выделенными блоками цифровой обработки сигналов DSP, включая

дополнительный модуль с плавающей запятой FPU. Характеристики Arm Cortex®-M4:

- Архитектура Armv7E-M.
- Поддержка набора команд Thumb/Thumb-2.
- Трехступенчатый конвейер.
- Цифровые блоки DSP для обработки сигналов с использованием математических вычислений. Дополнительный модуль с плавающей запятой (FPU).

Cortex-M4 являются отличным выбором для интернета вещей, для управления батарейным питанием, для применения в бытовых портативных устройствах, для обработки выходного сигнала микрофона и сигналов обратной связи от датчиков, встроенных в систему управления двигателем.

Arm Cortex-M33 разработан для IoT и встраиваемых приложений для управления цифровыми сигналами. Процессор обеспечивает примерно на 20% большую производительность, чем Cortex-M4, имеет множество дополнительных функций, включая блок цифровой обработки сигналов (DSP), блок защиты памяти и блок с плавающей запятой (FPU). Cortex-M33 является идеальным процессором для большинства приложений, включая интеллектуальные измерительные приборы, малогабаритные портативные устройства, медицинские приложения. Характеристики ядра Arm Cortex®-M33:

- Архитектура Armv8-M.
- Поддержка набора команд Thumb / Thumb-2.
- Трехступенчатый конвейер.
- Цифровые блоки DSP для обработки сигналов с использованием математических вычислений.
- Дополнительный модуль с плавающей запятой (FPU).
- Дополнительный модуль защиты памяти (MPU).

В таблице 1.5 представлена сравнительные характеристики процессорных ядер Arm Cortex-M (Y – Yes, N – No).

Таблица 1.5

Семейство процессоров									
Архитектура		Arm v6-M		Arm v7-M			Arm v8-M		Arm v8.1-M
		M0/M1	**M0+**	**M3**	**M4**	**M7**	**M23**	**M33**	**M55**
Обработка прерываний									
	Регулярные прерывания	Y	Y	Y	Y	Y	Y	Y	Y
	Динамические прерывания	Y	Y	Y	Y	Y	Y	Y	Y
	Прямые прерывания	Y	Y	Y	Y	Y	Y	Y	Y
	Прерывания с нулевой задержкой	N	N	Y	Y	Y	Y	Y	Y
Режим холостого хода		Y	Y	Y	Y	Y	Y	Y	Y
Встроенный системный таймер (SysTick)		N	Y	Y	Y	Y	Y	Y	Y
Защита памяти									
	Пользовательский режим	N	Y	Y	Y	Y	Y	Y	Y
	Защита стека HW (MPU)	N	N	Y	Y	Y	Y	Y	Y
	Проверка предела стека с помощью HW	N	N	N	N	N	Y	Y	Y
Операции с поддержкой HW		N	N	Y	Y	Y	N	Y	Y
Выполнение функций SRAM		N	N	Y	Y	Y	N	Y	Y
Сервисы с плавающей запятой		N	N	N	Y	Y	N	Y	Y
DSP ISA		N	N	N	Y	Y	N	Y	Y
Встроенная поддержка TrustZone-M		N	N	N	N	N	Y	Y	Y

Глава 1. Программные и аппаратные средства программирования микроконтроллеров

1.1. Общие сведения

При разработке приложений для микроконтроллеров Microchip используются программные и аппаратные средства.

Программные средства необходимы для написания и отладки программного кода. Программный код для микроконтроллеров пишется либо на языке ассемблера (машинно-ориентированный язык низкого уровня), либо на языках высокого уровня C/C++. Для микроконтроллеров компании Microchip используются следующие программные средства:

- **Microchip Studio** – интегрированная среда программирования для микроконтроллеров AVR и SAM.
- **MPLAB® X IDE** – интегрированная среда программирования для микроконтроллеров PIC. Совместно с MPLAB X IDE может использоваться плагин MPLAB Code Configurator (MCC), который облегчает и ускоряет процесс написания программного кода.
- **Компилятор MPLAB® XC**. Если разрабатываемое приложение будет написано на C/C++, то потребуется установить компилятор, который преобразует программу с языка высокого уровня в машинный код. Компания Microchip предлагает компиляторы MPLAB XC8, MPLAB XC16, MPLAB XC32 для 8, 16 и 32-разрядных микроконтроллеров соответственно.
- **Proteus –** программная среда для моделирования процесса выполнения программного кода.

Аппаратные средства необходимы для прототипирования, т.е. проверки работы созданной программы на реальном устройстве. Используются следующие аппаратные средства.

- **Отладочные платы (Curiosity boards).**
- **Программатор (programmer).** Программатор используются для загрузки созданной программы в микроконтроллер. Если используемая отладочная

плата не имеет встроенного программатора, то его надо приобрести отдельно.

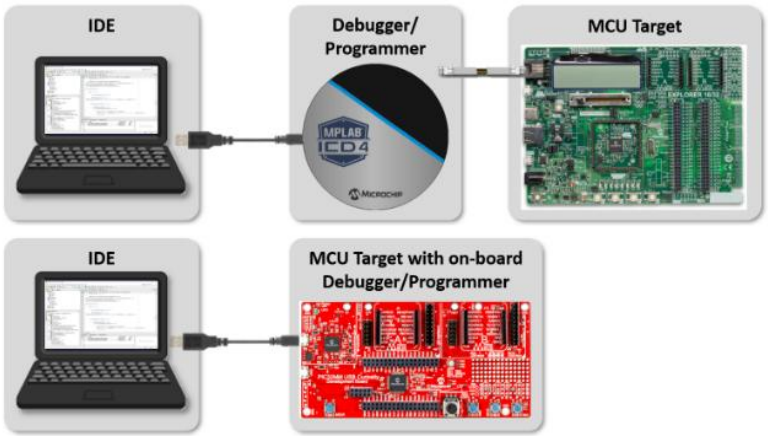


Рис. 1.5. Автономный и встроенный программаторы

На рис. 1.5 показаны два варианта применения программатора: в верхней части рисунка показан автономный программатор (требуется кабель для соединения программатора с компьютером и второй кабель – для соединения с отладочной платой); в нижней части – программатор встроен в отладочную плату (требуется один кабель для соединения компьютера с отладочной платой).

Таблица 1.6

Программатор	Поддерживаемые продукты	Поддерживаемые IDE	Промышленное применение	Цена, $
MPLAB ICE 4	PIC, AVR и SAM, dsPIC	MPLAB X IDE	Да	1799
MPLAB ICD 4	PIC, SAM, dsPIC	MPLAB X IDE	Да	259,95
MPLAB PICkit 4	PIC, AVR и SAM, dsPIC	MPLAB X IDE, Microchip Studio	Да	57,95
MPLAB SNAP	PIC, AVR и SAM, dsPIC	MPLAB X IDE	Нет	24,95
ATMEL-ICE	AVR и SAM,	Microchip Studio	Нет	140
J-32	32-разрядные PIC и SAM	MPLAB X IDE	Да	200
Power Debugger	AVR и SAM	Microchip Studio	Нет	200

В таблице 1.6. перечислены используемые в настоящее время программаторы для микроконтроллеров компании Microchip. В последнем столбце указана отпускная цена компании.

- **Платы расширения.** Эти платы позволяют нарастить функциональные возможности микроконтроллеров. Например, совместно с микроконтроллерами компании Microchip широко применяются платы расширения **click boards ™** компании MikroElektronika (www.mikroe.com).

  Click boards™ - это модульный стандарт плат расширения, которые добавляют отладочным платам новые функциональные возможности. Все click boards имеют стандартный разъем mikroBUS, что позволяет быстро создавать прототипы, не теряя времени на настройку или пайку оборудования. Эта концепция plug and play позволяет сосредоточиться не на оборудовании, а на реализации идеи проекта.

1.2. Операции с портами микроконтроллера

Микроконтроллер представляет из себя микросхему с несколькими выводами (рис. 1.6, 1.7).

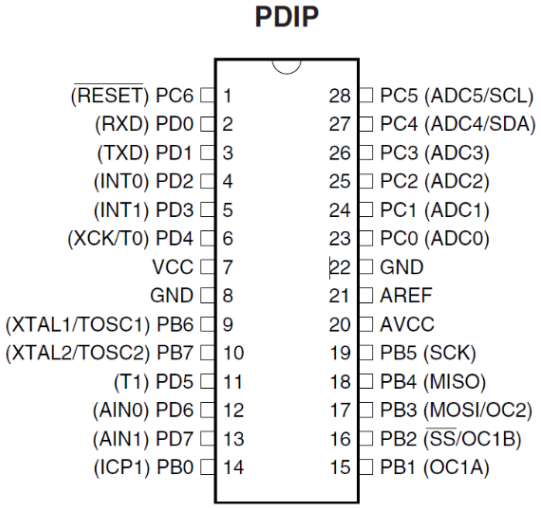


Рис. 1.6. ATMEGA8A

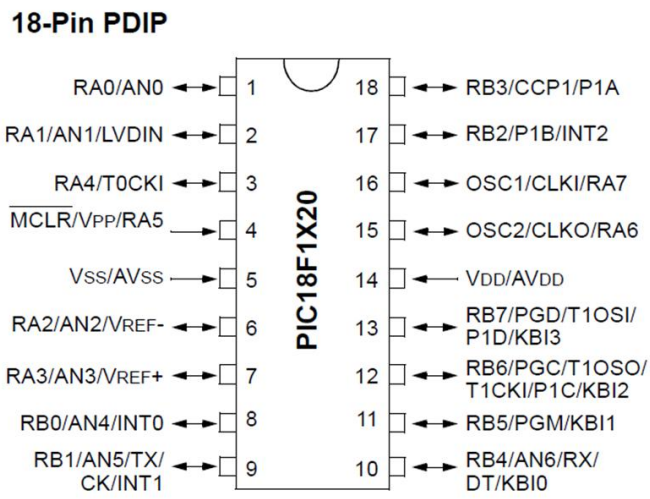


Рис. 1.7. PIC18F1X20

Выводы позволяют микроконтроллеру принимать и выводить данные. Выводы объединены в порты, которые обозначаются латинскими буквами *A, B, C, D* и т.д. Число портов зависит от модели микроконтроллера. Для 8-битных (8-разрядных) микроконтроллеров каждый порт содержит восемь выводов (восемь контактов). Неполный порт может содержать меньше выводов.

На рис. 1.6. и 1.7. показаны 28-контактная микросхема микроконтроллера AVR ATMEGA8A и 18-контактная микросхема микроконтроллера PIC18F1X20, выполненные в DIP корпусах. Нумерация **выводов микросхемы** (для данного типа корпуса) начинается с левого верхнего вывода и продолжается против часовой стрелки. Верхнюю часть микросхемы можно определить по полукруглому вырезу в корпусе микросхемы. Из рис. 1.6 следует, что микроконтроллер ATMEGA8A содержит два полных порта: *PortB, PortD* по восемь выводов, и один неполный *PortC* из семи выводов. Из рис. 1.7 следует, что микроконтроллер PIC18F1X20 содержит два полных порта: *PortA* и *PortB* по восемь выводов.

На выводы порта можно подавать логические 0 или 1, где 0 – это отсутствие напряжения, 1 – это напряжение +5 В. На самом деле напряжения занимают некоторый диапазон около этих значений, но это сути не меняет. Таким образом, на каждый вывод микроконтроллера можно записать один бит информации, принимающий значение 0 или 1. Тогда для 8-разрядного порта А можно записать 8-битное двоичное число:

PortA=0b11111111 //все биты имеют значение 1 (на все выводы порта
//подано напряжение +5 В).

PortA=0b00000000 //все биты имеют значение 0 (на всех выводах
//отсутствует напряжение).

где 0b – это префикс двоичного числа.

В первом случае в *PortA* записано двоичное число, которому соответствует десятичное число 255, во втором случае в *PortA* записано число 0.

Число в порты микроконтроллера можно записать также в шестнадцатеричном виде. Запись числа в шестнадцатеричном виде имеет префикс 0x и запишется следующим образом:

PORTA = 0xff //на всех выводах порта А установлена 1.

PORTA = 0x00 //на всех выводах порта А установлен 0.

Нумерация битов порта начинается с **крайнего правого бита, который считается нулевым,** следующий бит является первым и т.д. Нулевому биту соответствует нулевой вывод порта, первому биту — первый вывод порта и т.д. PortA = 0b00000011;

Эта запись означает, что в нулевой и первый биты записана 1, в остальные биты записан 0. Обращение к портам производится через регистры ввода-вывода микроконтроллера.

Выводы порта, как следует из рис. 1.6 и рис. 1.7 могут выполнять не только функции ввода/вывода, но и другие функции.

Порты микроконтроллера AVR имеют три регистра ввода-вывода: регистр направления данных *DDRx,* регистр данных порта *PORTx*, регистр чтения выводов порта *PINx.* Любой регистр представляет собой ячейку памяти. Названия регистров получаются подстановкой названия порта вместо символа *x,* соответственно для порта *A: PORTA, DDRA, PINA,* для порта *B: PORTB, DDRB, PINB* и т.д. Регистры *PINx* доступны только для чтения, регистры *PORTx* и *DDRx* доступны для чтения и для записи.

Любой порт микроконтроллера *AVR* можно сконфигурировать или на вход, или на выход, или частично на вход, частично на выход. Для этой цели используется регистр *DDRx*. Если некоторый разряд регистра *DDRx* содержит 0, то он сконфигурирован, как вход, если содержит 1, то как выход. Под понятием "порт сконфигурирован, как выход" подразумевается, что микроконтроллер передает информацию на внешнее устройство, под понятием "порт сконфигурирован, как вход" подразумевается, что микроконтроллер принимает информацию от внешнего устройства.

Для управления портами микроконтроллера используются побитовые операции. Перейдем к рассмотрению побитовых операций.

Сдвиг вправо (>>). С помощью данной команды все биты регистра порта сдвигаются вправо на количество позиций, указанных справа после знака >>. Пусть во все биты порта *D* микроконтроллера установлена 1:

PORTD = 0b11111111; //исходное число.

Применим операцию сдвига на три разряда вправо.

PORTD = PORTD >> 3; // команда сдвига на три разряда вправо

PORTD = 0b00011111*;* //число, получаемое в результате сдвига.

При этом освободившиеся разряды заполняются нулями, а вышедшие за пределы регистра теряются. Исходному двоичному числу соответствует десятичное число 255, полученному двоичному числу 00011111 соответствует десятичное число 31.

Сдвиг влево (<<). С помощью данной команды все биты регистра сдвигаются влево на количество позиций, указанных справа после знака << . Пусть во все биты порта *D*, как и в предыдущем случае, установлена 1:

PORTD = 0b11111111; // исходное число

PORTD = PORTD << 3; // команда сдвига на три разряда влево

PORTD = 0b11111000; // результат сдвига.

Как и в предыдущем случае, освободившиеся разряды заполняются нулями, а вышедшие за пределы регистра теряются. Двоичному числу 11111000 соответствует десятичное число 248.

Пример 1.1. Записать в порт *D* число 137 в двоичном и шестнадцатеричном виде. Для отображения числа в двоичном и шестнадцатеричном виде можно воспользоваться стандартной программой *Калькулятор* ОС Windows в режиме *Программист*, тогда получим

PORTD = 0b10001001; // число 137 в двоичном виде;

PORTD = 0x89*;* // число 137 в шестнадцатеричном виде.

Пример 1.2. В порт *D* записано число *PORTD* = 0b11001001. Сдвинуть данное число на 4 знака влево.

PORTD = PORTD << 4;

В результате получим *PORTD* = 0b10010000.

Запись 1 в произвольный бит порта. Очень часто необходимо записать 1 или 0 не на все выводы порта, а только на некоторые из них, оставляя значения на остальных выводах без изменений. Для этого используются логические

операции. Применяемые чаще других логические операции представлены в таблице 1.7.

Таблица 1.7

Условное обозначение	Наименование логической операции
|	ИЛИ (логическое сложение)
&	И (логическое умножение)
^	Исключающее ИЛИ
~	НЕ (инверсия)

К примеру, если необходимо установить 1 на второй вывод порта *D*, не меняя значения других выводов, то следует использовать команду:

PORTD |= (1<<2);

Разберем подробнее, почему в результате выполнения этой команды на второй вывод порта *D* запишется именно 1. Фрагмент вышеприведенной команды (1<<2) осуществляет сдвиг единицы влево на 2 бита, а знак «|», стоящий перед знаком равно, выполняет операцию логического сложения этой 1 и того числа (0 или 1), которое находилось до этого во втором бите. Независимо от того, 0 или 1 были во втором бите ранее, операция логического сложения 1 с этими числами даст логическую 1, которая и запишется во второй бит.

Для того, чтобы записать 1 на несколько выводов порта, например, на второй, пятый и седьмой выводы, следует применить команду

PORTD | = (1<<2) | (1<<5) | (1<<7);

Запись 0 в произвольный бит порта. Если необходимо установить 0 на второй вывод порта *D*, не меняя значения других выводов, то следует применить команду:

PORTD & = ~ (1<<2);

В этом случае результат сдвига единицы на две позиции влево инвертируется с помощью операции инвертирования, которая обозначается знаком «~». Таким образом, сдвинутая влево на две позиции единица (1<<2) превращается при

инвертировании в ноль и умножается на второй бит регистра *PORTD*. Каким бы не было состояние второго бита до этого (0 или 1), при умножении на 0 получим результат 0, и это значение записывается во второй бит.

Для записи 0 на несколько выводов порта, например, на второй, пятый и седьмой выводы, следует применить команду:

PORTD &= ~ ((1<<2) | (1<<5) | (1<<7));

Для проверки состояния, например, второго вывода порта *D* следует применить следующие условные операторы:

if (~PIND & (1<<2)) // проверка на наличие 0.

if (PIND & (1<<2)) // проверка на наличие 1.

Поясним эти условные операторы. Выражение в скобках после *if* может принимать значение «истина» или «ложь». «Истина» – это 1, «ложь» – это 0. Предположим, во втором бите порта *D* записан 0. Тогда выражение в скобках принимает значение «истина». Действительно: ~(0·1)=1. Если во втором бите будет записана 1, то выражение в скобках принимает значение «ложь». Действительно: ~(1·1)=0. Следовательно, условный оператор *if (~PIND* & (1<<2)) может служить проверкой на наличие нуля в контролируемом бите. Аналогичные рассуждения можно провести для второго условного оператора

if (PIND & (1<<2))

и убедиться, что он может использоваться для проверки 1 в контролируемом бите.

Для проверки состояния вывода можно также применить другую форму записи условного оператора. Пусть в порт *D* записана любая последовательность нулей и единиц, например, *PORTD* = 0b00001111. Затем в процессе выполнения программы на второй вывод записывается 0. Тогда

if (PIND == 0b00001011*)* // проверка на наличие 0 на втором выводе.

Если на шестой вывод записывается 1, тогда

if (PIND == 0b01001111) // проверка на наличие 1 на шестом выводе.

Напомним, нумерация битов порта начинается с крайнего правого бита, который считается нулевым, следующий бит является первым и т.д. Двойной знак равенства "==" в языке С означает "равно".

Пример 1.3. В порт *D* записано число *PORTD* = 0b11000011. Записать «1» во второй и третий биты этого числа:

PORTD |= (1<<2) | (1<<3);

в результате получим число *PORTD* = 0b11001111.

Пример 1.4. В порт *D* записано число *PORTD* = 0b11000011. Записать «0» в нулевой и первый биты этого числа:

PORTD &=~ ((1<<0) | (1<<1));

в результате получим число *PORTD* = 0b11000000.

Переключение бита. Кроме установки и сброса отдельного бита часто используется команда, которая переключает отдельный бит в противоположное состояние: единицу в ноль и наоборот. Пусть в порт *D* установлено число: *PORTD* = 0b00011111. Переключение второго бита в противоположное состояние производится командой: *PORTD* ^= (1<<2). В результате имеем: *PORTD* = 0b00011011.

Выполним переключение в противоположное состояние нулевого, второго и шестого бита исходного числа *PORTD* = 0b00011111. Для этого применим команду: *PORTD* ^= (1<<0) | (1<<2) | (1<<6). В результате получим число: *PORTD* = 0b01011010.

Если применить команду *PORTD*=~0b00011111, то все биты поменяются на противоположные и получим число: *PORTD* = 0b11100000.

Порты микроконтроллера *PIC* имеют три специальных регистра: *TRISx* (регистр направления данных), *PORTx* (регистр чтения на выводах порта), *LATx* (регистр защелка порта). Регистр *TRISx* задает направление работы выводов порта. Если все выводы порта работают на вход, то в них записываются единицы, если на выход, то в них записываются нули, то есть наоборот, нежели у микроконтроллера AVR. (Мнемоническое правило для запоминания

направления работы выводов для микроконтроллера *PIC:* 1 – Input; 0 – Output, т.е. похожее написание цифры и первой буквы английских слов вход и выход).

Для задания направления работы порта *B* можно записать:

- *TRISB* = 0; // все выводы порта *B* работают на выход.
- *TRISB* = 0xFF; // все выводы порта *B* работают на вход.
- *TRISB* | = (1<<2) // второй вывод порта *B* работает на вход.
- *TRISB* &=~ (1<<2) // второй вывод порта *B* работает на выход.

Доступ к данным на выводах порта осуществляется посредством регистра *PORTx*.

- *PORTB* = 0; // запись 0 на все выводы порта.
- *PORTB* = 0xFF; // запись 1 на все выводы порта.
- *PORTB* | = (1<<2) // запись 1 на второй вывод порта.
- *PORTB* & = ~ (1<<2*)* // запись 0 на второй вывод порта.

Среда программирования микроконтроллеров PIC – *MPLAB X IDE* позволяет напрямую обращаться к выводам порта. Для записи значений на отдельных линиях порта используется команда:

- *PORTBbits.RB2* = 1*;* // запись 1 на второй вывод порта.
- *PORTBbits.RB2* = 0*;* // запись 0 на второй вывод порта.

Для проверки состояния выводов (0 или 1) следует применять условные операторы:

if (PORTBbits.RB1== 0)*;* // проверка на наличие 0,

if (PORTBbits.RB1== 1)*;* // проверка на наличие 1.

При чтении данных регистра *PORTx* считываются значения данных, присутствующие на выводах порта, а при чтении данных регистра LATx считываются данные, хранящиеся в регистре защелке порта. Данные, записанные в защелку порта *PORTx*, и данные, физически присутствующие на выводах порта *PORTx,* могут отличаться. Для записи в защелку порта и считывании из защелки порта используется регистр *LATx.*

- *LATB* = 0*;* // запись во все биты порта значения 0.
- *LATB* = 0xFF*;* // запись во все биты порта значения 1.

– *LATB* | = (1<<2) // запись во второй бит порта значения 1.

– *LATB* & = ~ (1<<2) // запись во второй бит порта значения 0.

Или можно напрямую обратиться к отдельным битам порта *B*.

– *LATBbits.LATB2* = 1; // запись 1 во второй бит порта.

– *LATBbits.LATB2* = 0; // запись 0 во второй бит порта.

Не вдаваясь в подробности схемотехнической реализации портов, можно сформулировать такое правило: когда данные записываются в порт, то надо использовать *LATx*, когда данные считываются из порта, то надо использовать *PORTx*. Для микроконтроллеров серий *PIC16* и младше регистр *LATx* не используется, используются только регистры *TRISx* и *PORTx.* В этом случае, для записи данных в порт и чтения данных из порта используется *PORTx.*

1.3. Интегрированная среда разработки Microchip Studio для AVR

1.3.1. Установка Microchip Studio

Интегрированная среда разработки IDE (IDE – Integrated Development Environment) Microchip Studio используется для программирования микроконтроллеров AVR. Эта IDE представляет собой единую среду для записи, сборки и отладки программ на ассемблере и на C/C++. Официальную версию можно скачать бесплатно с сайта компании Microchip: https://www.microchip.com/en-us/tools-resources/develop/microchip-studio. Можно скачать либо автономный установщик (устанавливается без подключения к интернет), либо веб-установщик (при установке необходимо подключение к интернет). Среду Microchip Studio можно использовать для разработки и отладки приложений на микроконтроллерах AVR ® и SAM. Несмотря на то, что она поставляется с новым названием и внешним видом, её можно использовать с любой существующей документацией и видеороликами от предыдущей среды разработки – Atmel Studio.

Microchip Studio имеет следующие отличительные черты:

- Поддержка более 500 устройств AVR и SAM.

- Поддержка компилятора MPLAB ® XC8.
- Обширная библиотека исходного кода, включая драйверы, коммуникационные стеки, многочисленные примеры с исходным кодом, графические сервисы и сенсорные функции Advanced Software Framework (ASF).
- Расширения IDE через Microchip Gallery – онлайн-магазин приложений для инструментов разработки и встроенного программного обеспечения от Microchip и третьих сторон.
- Функционал для настройки емкостного сенсорного дизайна, проверки производительности системы, мониторинга энергопотребления и построения графиков и трассировки данных в режиме реального времени.
- Функционал для настройки и тестирования беспроводных устройств.
- Встроенный компилятор для создания и отладки кода на C/C++ и ассемблере.
- Расширенные функции отладки, включая сложные точки останова данных и поддержку трассировки (устройства SAM3 и SAM4).
- Встроенный редактор с визуальной поддержкой.
- Мастер проектов, который позволяет создавать проекты с нуля или из большой библиотеки примеров.

Установка Microchip Studio. Скачаем и запустим автономный установщик (файл as-installer-7.0.2594-full.exe). Появится окно с лицензионным соглашением (рис. 1.8). Ставим галочку *I agree* и нажимаем Next.



Рис. 1.8. Запуск установки среды программирования Microchip Studio.

Лицензионное соглашение

Появляется следующее окно (рис. 1.9)

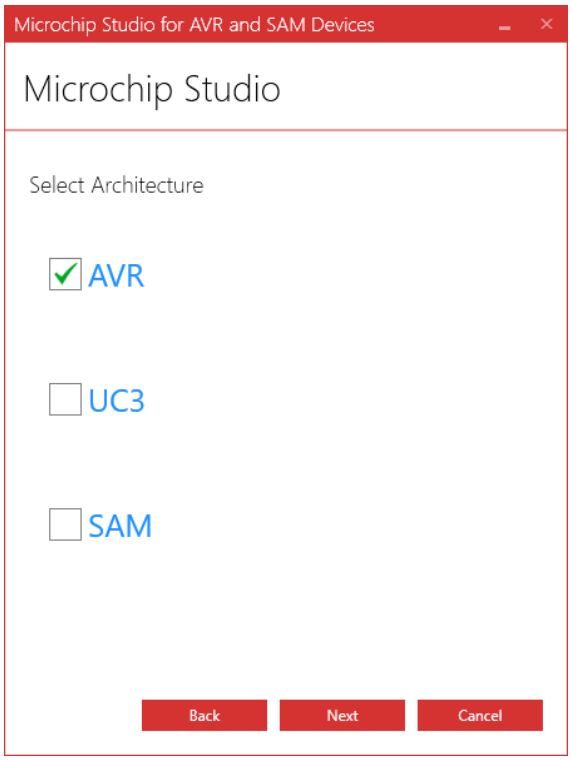


Рис. 1.9. Выбор семейства микроконтроллеров

Если вы не планируете программировать 32-разрядные микроконтроллеры, то галочки UC3 и SAM снимаем. Нажимаем Next. В следующем окне (рис. 1.10) галочку оставляем, нажимаем Next.

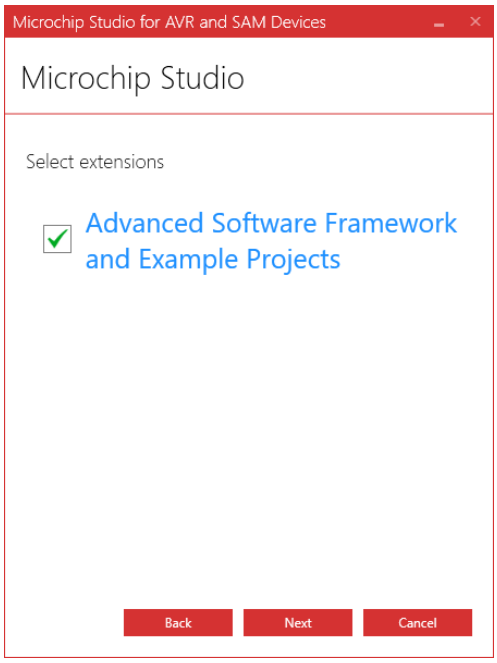


Рис. 1.10. Выбор расширений

Появляется окно (рис. 1.11, нажимаем Next

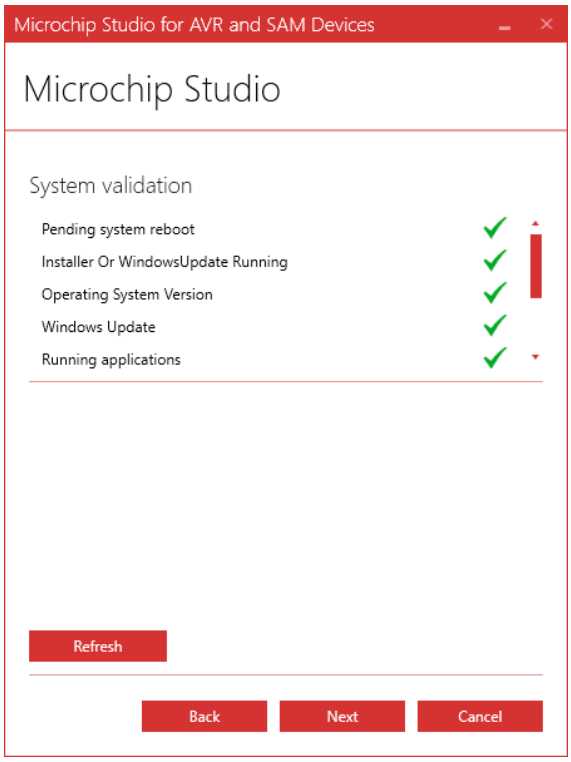


Рис. 1.11. Подтверждение выбранных опций

Появляется окно рис. 1.12, в котором предлагается сделать *Upgrade* до версии PRO. Мы делать *Upgrade* не будем и нажимаем *Install.*

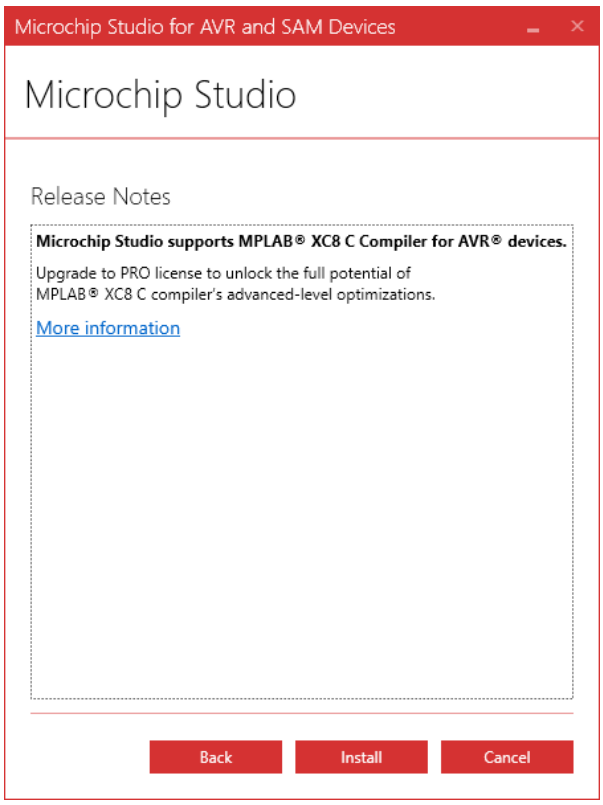


Рис. 1.12. Предложение сделать Upgrade до версии PRO

Начинается установка приложения. В конце установки появляется окно (рис. 1.13) и на рабочем столе появляется иконка Microchip Studio.

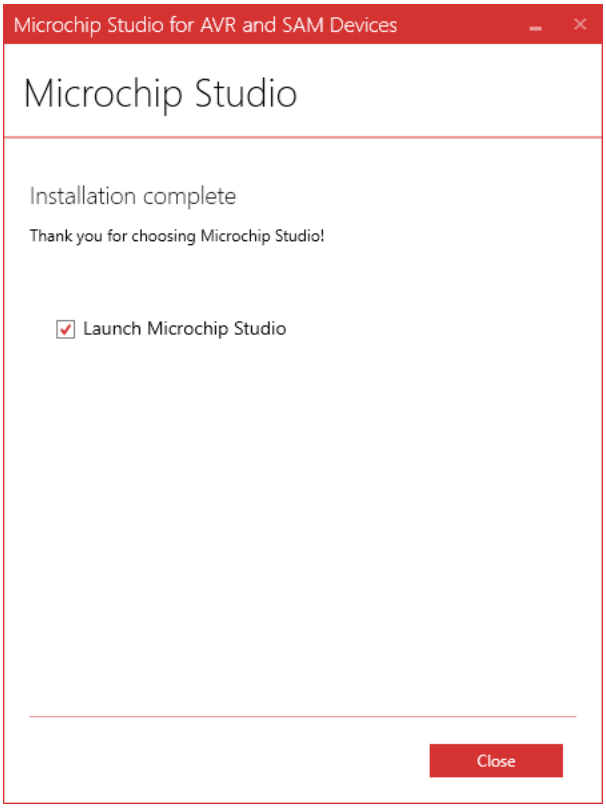


Рис. 1.13. Подтверждение окончания установки ПО

Щелкаем *Close*. Приложение запускается и появляется начальное окно Microchip Studio (рис. 1.14).

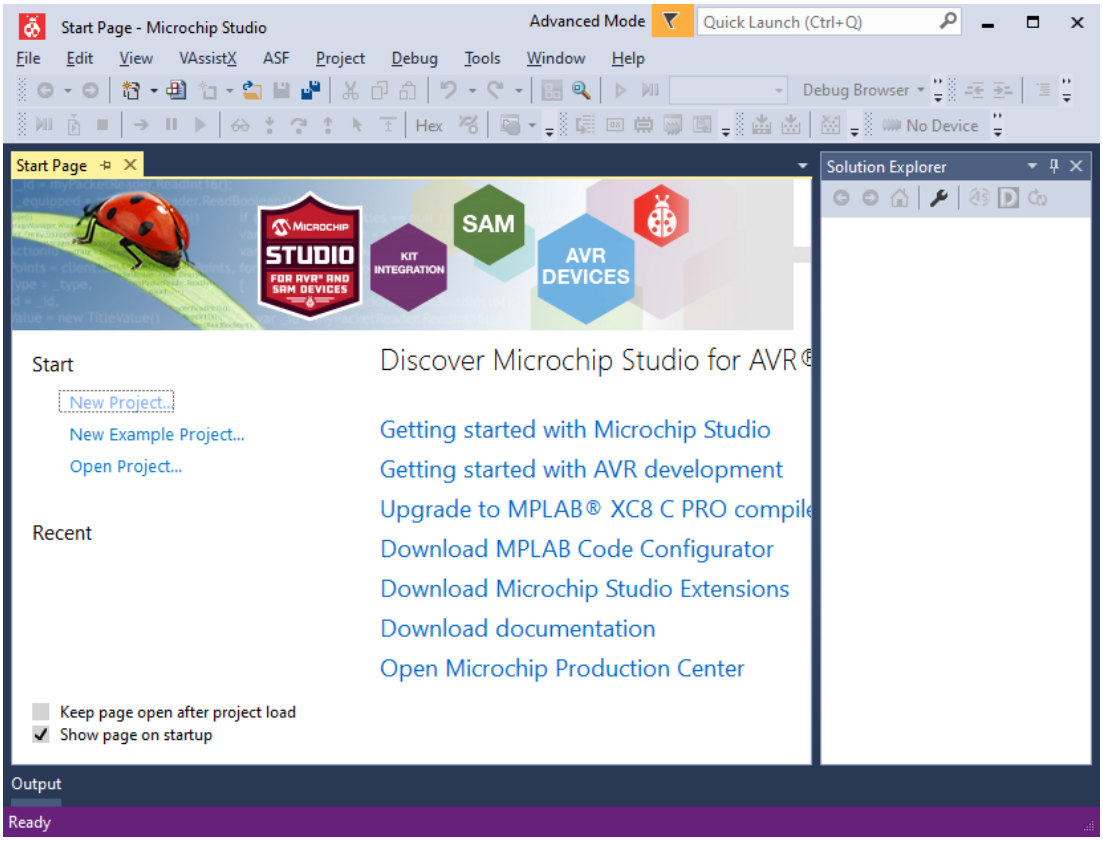


Рис. 1.14. Начальное окно Microchip Studio

1.3.2. Создание проекта "Линейка светодиодов" в Microchip Studio

1.3.2.1. Начало работы

Разработка программы начинается с создания проекта. Для начала на диске C создадим папку для хранения проектов, которую назовем *Projects_AVR*. Чтобы создать проект, надо на рис. 1.14 выбрать опции *File –> New –> Project*. Появится окно мастера создания проекта, показанное на рис. 1.15.

Выберем в левой части окна язык C/C++, в средней части выберем *AVR XC8 C Application Project*.

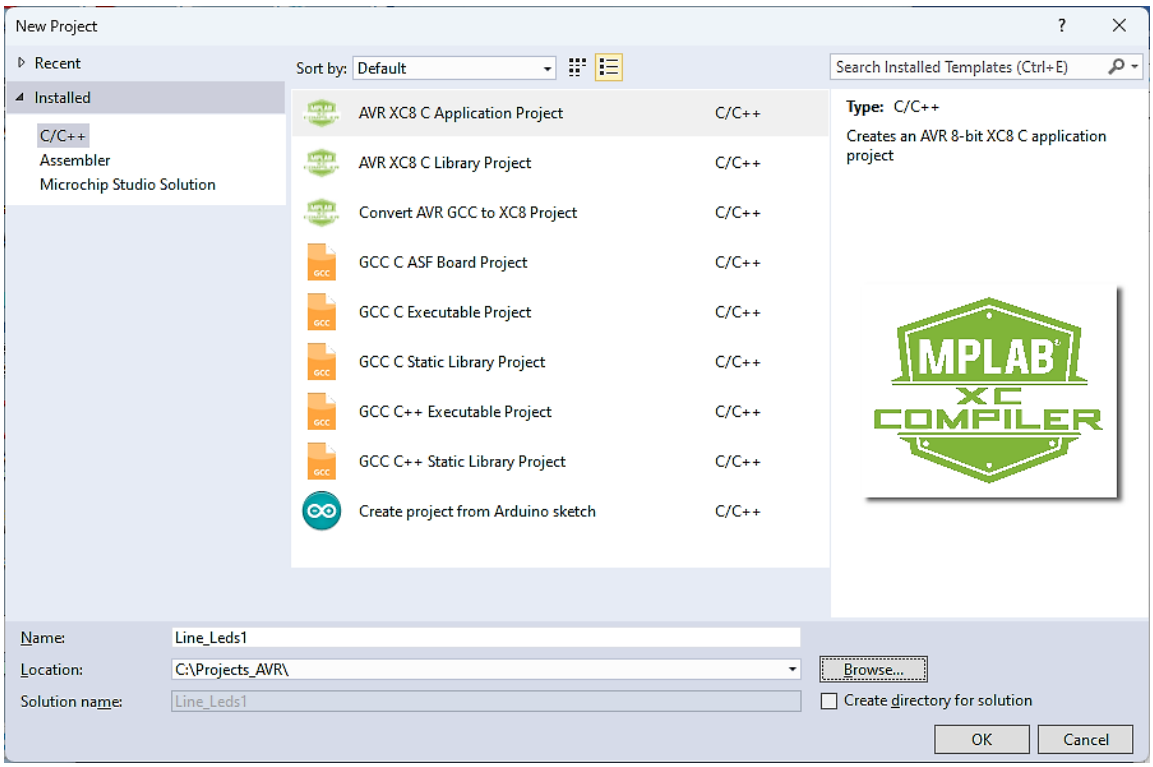


Рис. 1.15. Окно мастера создания проекта

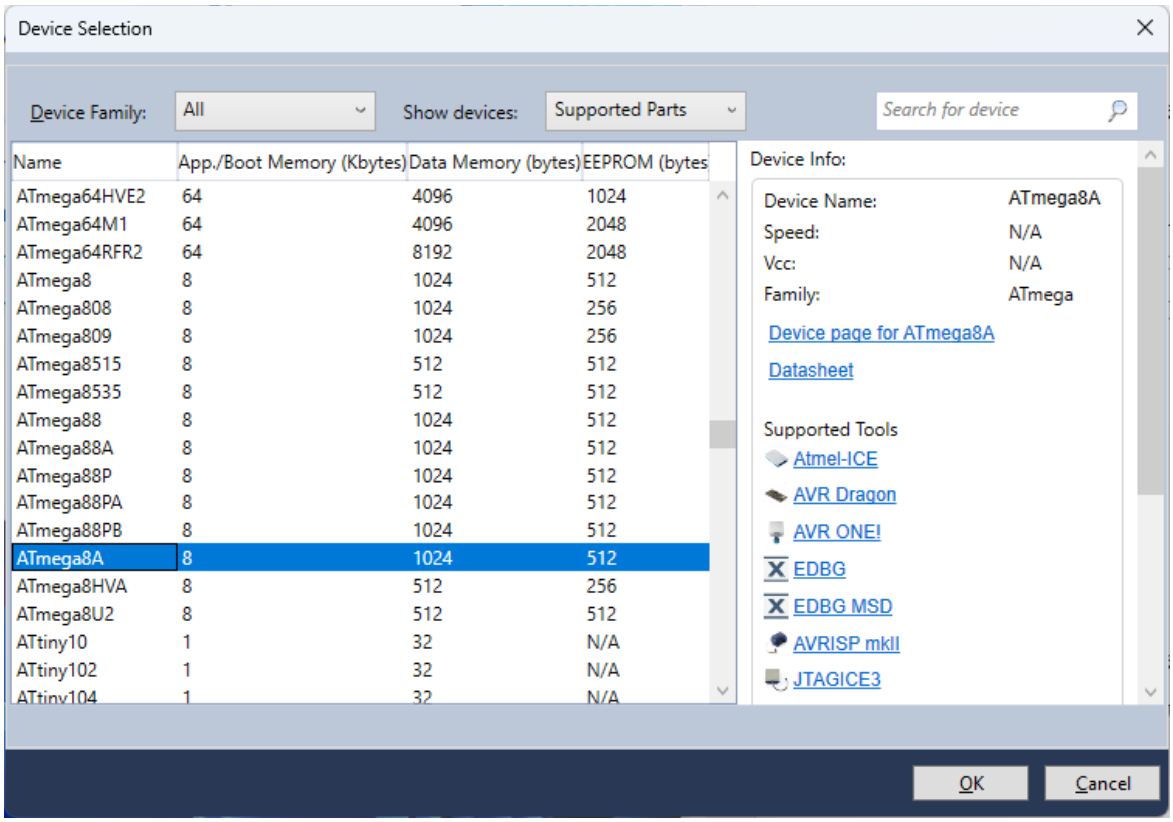


Рис. 1.16. Выбор микроконтроллера

Будем создавать проект, который представляет собой мигающую линейку из восьми светодиодов. Поэтому в нижней части этого окна в качестве имени

проекта укажем *Line_Leds1*, далее щелкнем по кнопке *Browse* и укажем созданную ранее папку *Projects_AVR*. Галочку около опции *Create directory for solution* убираем. Щелкаем OK.

Появится новое окно, в котором выбираем микроконтроллер ATmega8A, как показано на рис. 1.16. Щелкаем OK. Появляется окно, показанное на рис. 1.17 с начальным кодом на языке C. Здесь есть строки, выделенные символами:

/* … */ – начало и конец комментария.

Этими символами обозначается многострочный комментарий. Однострочный комментарий обозначатся двумя косыми чертами //. Комментарий служит для пояснения кода программы, компилятор на него не реагирует и просто пропускает. Поэтому без ущерба для программы комментарий можно удалить.

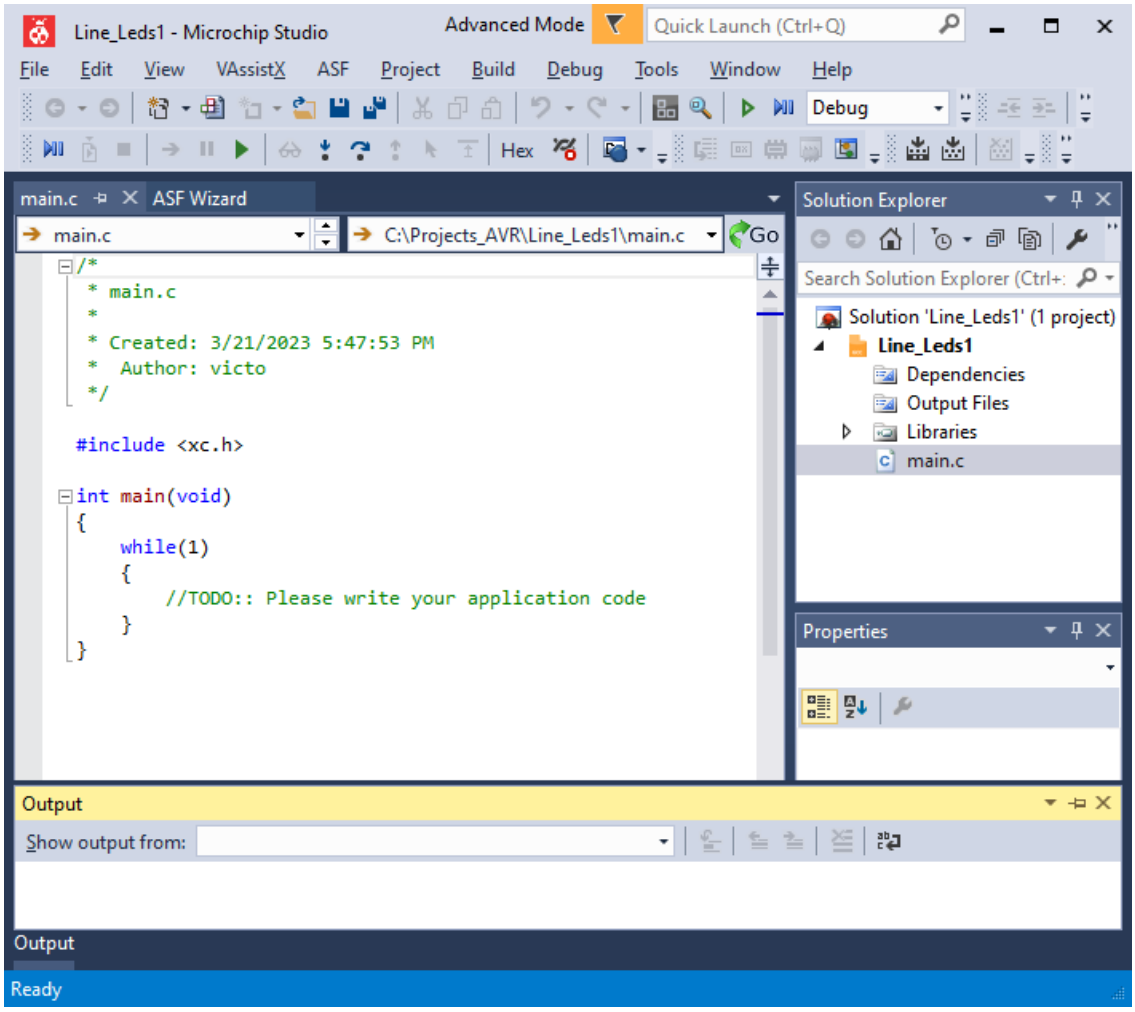


Рис. 1.17. Заготовка (начальный код) программы на языке C

1.3.2.2. Разработка листинга программы

Приступим к созданию кода программы "Линейка светодиодов". Начальный код при этом можно просто удалить. При написании программы следует обращать внимание на подсказки редактора кода. Редактор знает доступные переменные и функции. Например, при вводе директивы #include уже при наборе первых букв #in появится список, из которого можно выбрать #include, щелкнув по клавише Enter. Так меньше шансов сделать ошибку при наборе кода. При редактировании программы, если нажать *Ctrl* + *Space*, появится список автозаполнения, из которого можно выбрать доступные варианты.

В качестве примера создадим для микроконтроллера ATmega 8A проект, содержащий программу для мигания линейки из восьми светодиодов (рис. 1.18).

.

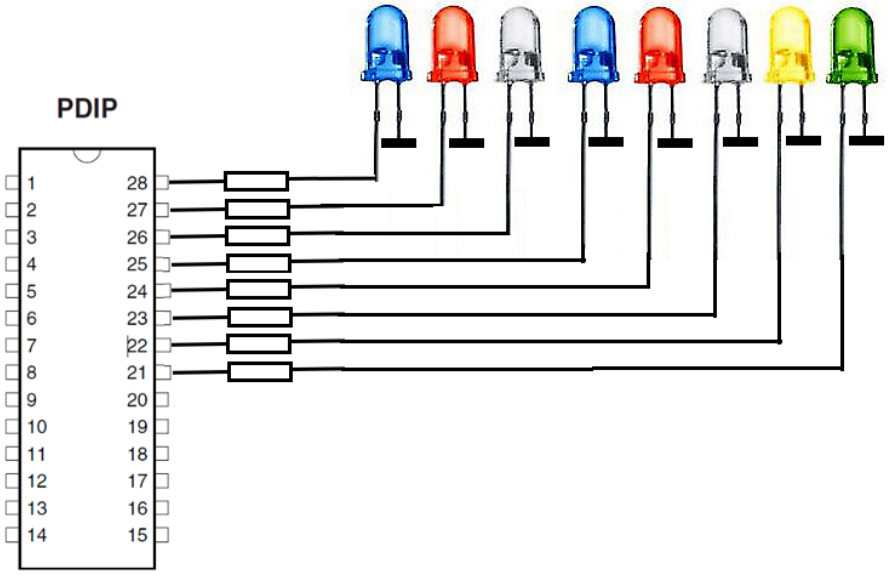


Рис. 1.18. Проект "Линейка светодиодов"

Пример 1.5. Программа **Line_Leds1.** Для написания программы задействуем восемь выводов порта D.

Примечание. Рисунок 1.18 носит иллюстративный характер и не отражает реального расположения выводов порта D микроконтроллера ATmega 8A. На этапе создания программы это не имеет принципиального значения. Это становится важным только тогда, когда мы начинаем создавать действующий образец, состоящий из реального микроконтроллера и светодиодов. Тогда уже надо смотреть, какие конкретно выводы микросхемы микроконтроллера

соответствуют выводам порта D. Пока достаточно знать, что микроконтроллер ATmega8A действительно имеет порт D, и этот порт содержит восемь выводов.

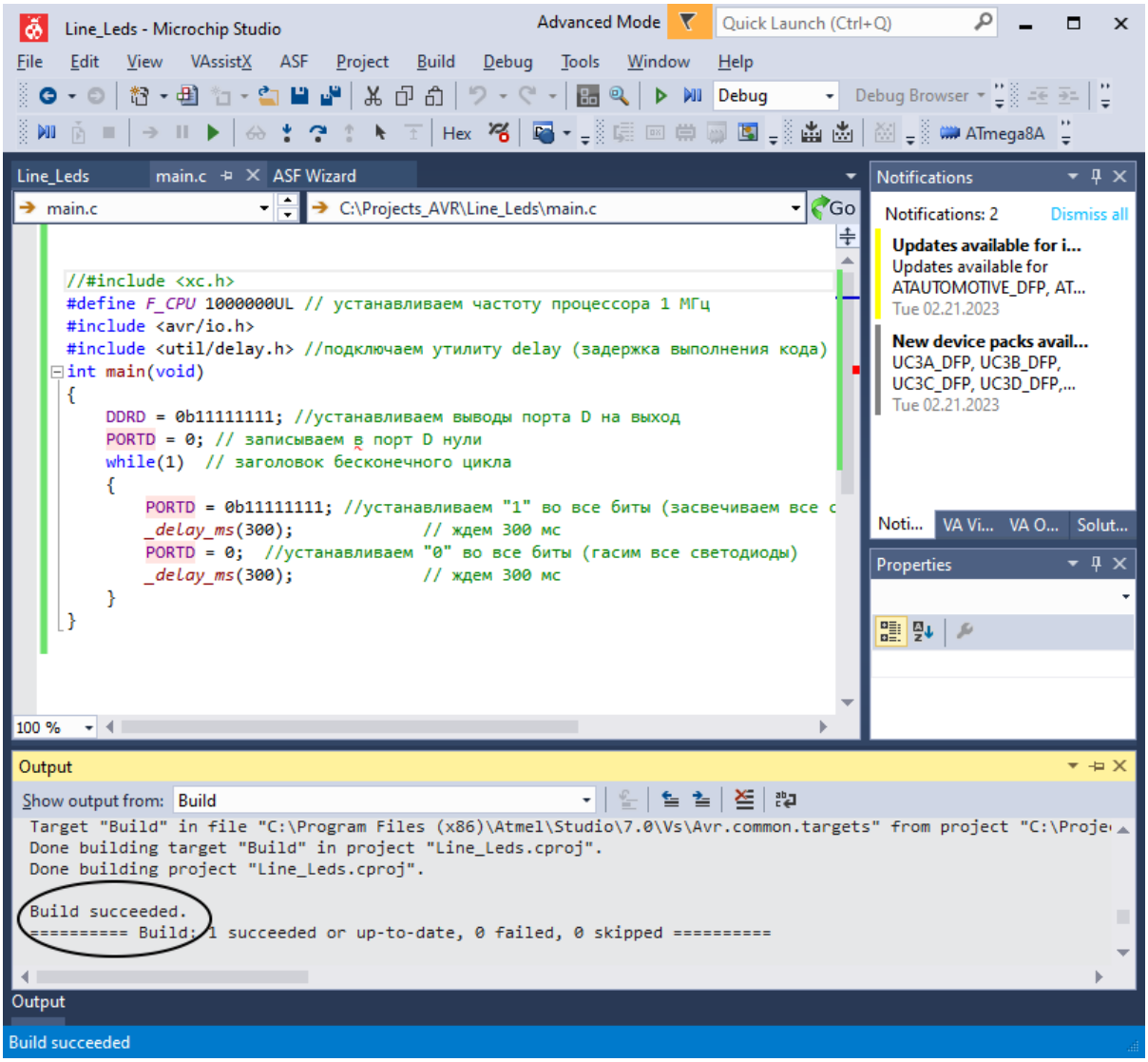


Рис. 1.19. Успешная компиляция программы

```
#define F_CPU 1000000UL // устанавливаем частоту процессора 1 МГц
#include <avr/io.h>          //подключаем библиотеку ввода/вывода
#include <util/delay.h>      //подключаем утилиту delay
int main(void)               //заголовок главной программы
{
DDRD = 0b11111111;           //устанавливаем выводы порта D на выход
PORTD = 0;                   //записываем в порт D нули
while(1)                     //заголовок бесконечного цикла
{
PORTD = 0b11111111;          //засвечиваем все светодиоды
_delay_ms(300);              // ждем 300 мс
PORTD = 0;                   //гасим светодиоды
```

```
_delay_ms(300);                // ждем 300 мс
}
}
```

После того, как программа написана, её необходимо скомпилировать. Для этого выбираем опции Build –> Build Solution (или щелкаем клавишу F7). Если код программы написан без ошибок, то в нижней части интерфейсного окна появится запись *Build succeeded* (сборка прошла успешно) (рис. 1.19).

Пример 1.6. Программа **Line_Leds2**. Давайте усложним программу. Пусть светодиоды засвечиваются не все сразу, а последовательно по одному, начиная с нулевого и заканчивая седьмым. Таким образом, по линейке светодиодов от её начала (от нулевого светодиода) к концу побежит светящийся огонек.

Создадим проект Line_Leds2 и напишем программу.

```
#define F_CPU 1000000UL // устанавливаем частоту процессора 1 МГц
#include <avr/io.h>
#include <util/delay.h>       //подключаем утилиту delay
int main(void)
{
DDRD = 0b11111111;         //устанавливаем выводы порта D на выход
PORTD = 0;                 // записываем в порт D нули
while(1)                   // заголовок бесконечного цикла
{
int i; //определяем переменную i
PORTD=0b00000001;          //засвечиваем нулевой светодиод
for (i=0; i<=7; i=i+1)     //задаем цикл
{
if (i>=1)
PORTD=PORTD<<1;          //применяем операцию сдвига влево на 1 позицию
_delay_ms(300);            // ждем 300 мс
}
PORTD=0; //гасим все светодиоды
_delay_ms(300);            // ждем 300 мс
}
}
```

Пример 1.7. Создадим проект **Line_Leds3** и напишем еще одну программу, которая работает по следующему алгоритму. Вначале светодиоды последовательно загораются с первого по восьмой. Работу этого фрагмента