AVR++

Release 1.0.0

Oct 31, 2018

Contents

AVR++ is a C++17 interface to Atmel AVR 8-bit microcontrollers. The goal of AVR++ is to provide a safe, clean, and zero-overhead interface to work with with AVR microcontrollers while still allowing for great flexibility and customizability.

CHAPTER 1

Public API

The AVR++ public API provides access to all features of the microcontroller. It abstracts away tedious bit manipulations an presents a type-safe interface that catches a wide variety of programming errors during compilation. The underlying implementation has been designed to provide zero- or near-zero overhead.

This part of the documentation describes how to use AVR++ to program microcontrollers in a safe and efficient manner.

1.1 Public API

This part of the documentation describes the user-facing public API of AVR++ and the ATL.

1.1.1 AVR Power Saving Sleep Modes

This section introduces the **AVR Power Saving Sleep API**. This API provides the necessary functions and constants to configure and manage sleep states on AVR microcontrollers. Not all AVR microcontrollers implement the same set of functionality with respect to sleep modes. Using a feature that is not supported by the targeted controller will result in a compile error.

Sleep Modes

Warning: doxygenenum: Cannot find enum "avr::sleep::mode" in doxygen xml output for project "AVR++" from directory: /home/docs/checkouts/readthedocs.org/user_builds/avrxx/checkouts/latest/doc/xml

Configuring Sleep Modes

Warning: doxygenfunction: Cannot find function "avr::sleep::select" in doxygen xml output for project "AVR++" from directory: /home/docs/checkouts/readthedocs.org/user_builds/avrxx/checkouts/latest/doc/xml

Warning: doxygenfunction: Cannot find function "avr::sleep::disable_brownout_detector" in doxygen xml output for project "AVR++" from directory: /home/docs/checkouts/readthedocs.org/user_builds/avrxx/checkouts/latest/doc/xml

Managing Sleep State

Warning: doxygenfunction: Cannot find function "avr::sleep::enter" in doxygen xml output for project "AVR++" from directory: /home/docs/checkouts/readthedocs.org/user_builds/avrxx/checkouts/latest/doc/xml

Warning: doxygenfunction: Cannot find function "avr::sleep::select_enter" in doxygen xml output for project "AVR++" from directory: /home/docs/checkouts/readthedocs.org/user_builds/avrxx/checkouts/latest/doc/xml

Warning: doxygenfunction: Cannot find function "avr::sleep::enable" in doxygen xml output for project "AVR++" from directory: /home/docs/checkouts/readthedocs.org/user_builds/avrxx/checkouts/latest/doc/xml

Warning: doxygenfunction: Cannot find function "avr::sleep::select_enable" in doxygen xml output for project "AVR++" from directory: /home/docs/checkouts/readthedocs.org/user_builds/avrxx/checkouts/latest/doc/xml

Warning: doxygenfunction: Cannot find function "avr::sleep::sleep" in doxygen xml output for project "AVR++" from directory: /home/docs/checkouts/readthedocs.org/user_builds/avrxx/checkouts/latest/doc/xml

1.1.2 Character Strings

This chapter introduces the **character string** handling infrastructure of the ATL.

String Utilities

Warning: doxygenfunction: Cannot find function "atl::string::length" in doxygen xml output for project "AVR++" from directory: /home/docs/checkouts/readthedocs.org/user_builds/avrxx/checkouts/latest/doc/xml

1.1.3 Integer Types

This chapter introduces the **integer types** library of the ATL.

Integer Type Utilities

This section presents the utility types for determining the correct integer type for a given bit-width.

Base Templates

template <int *Size*>
struct int_for_size
A template to determine the exact int type to represent the given size.

This template provides integer types that match the given size exactly. If a given size can be represented exaclty, it provides a member type called `type` that is an alias for the repective integer type.

See *avr::uint_for_size* For the equivalent helper for unsigned types

Since 1.0.0

Template Parameters

- `Size`: The desired size of the type

template <int *Size*>
struct uint_for_size : **public** make_unsigned<*int_for_size_t*<Size>>
A template to determine the exact unsigned int type to represent the given size.

This template provides unsigned integer types that match the given size exactly. If a given size can be represented exaclty, it provides a member type called `type` that is an alias for the repective integer type.

See *avr::int_for_size* For the equivalent helper for signed types

Since 1.0.0

Template Parameters

- `Size`: The desired size of the type

Convenience Accessors

using int_for_size_t = **typename** int_for_size<Size>::type
Convenience alias for the `type` member of *avr::int_for_size*.

Since 1.0.0

Template Parameters

- `Size`: The desired size of the type

using uint_for_size_t = **typename** uint_for_size<Size>::type
Convenience alias for the `type` member of *avr::uint_for_size*.

Since 1.0.0

Template Parameters

- `Size`: The desired size of the type

Fixed Width Integer Types

This section presents the fixed width integer type aliases of the ATL.

Exact Width Types

This section describes signed and unsigned integer type that match exactly the specified width.

Signed Integer Types

using int8_t = implementation_defined
A signed integer type that is exactly 8 bit wide

Since 1.0.0

using int16_t = implementation_defined
A signed integer type that is exactly 16 bit wide

Since 1.0.0

using int32_t = implementation_defined
A signed integer type that is exactly 32 bit wide

Since 1.0.0

using int64_t = implementation_defined
A signed integer type that is exactly 64 bit wide

Since 1.0.0

Unsigned Integer Types

using uint8_t = implementation_defined
An unsigned integer type that is exactly 8 bit wide

Since 1.0.0

using uint16_t = implementation_defined
An unsigned integer type that is exactly 16 bit wide

Since 1.0.0

using uint32_t = implementation_defined
An unsigned integer type that is exactly 32 bit wide

Since 1.0.0

using uint64_t = implementation_defined
An unsigned integer type that is exactly 64 bit wide

Since 1.0.0

CHAPTER 2

Library Internals

The AVR++ internals are extensively documented to make it easy to extend and improve. All subsystems have been designed to provide clean and stable interfaces, to make it easy to build derivative libraries or implement custom features as standalone libraries.

This part of the documentation describes how to work with the underlying implementation of AVR++.

2.1 Library Internals

This part of the documentation describes the library internals of AVR-wrapper, and is mainly of interest to core library developers:

2.1.1 AVR Special Function Register Infrastructure

This chapter introduces the **AVR Special Function Register API**. This API provides access to the Special Function Registers of an AVR microcontroller. These registers can be used for controlling various interfaces of an AVR controller.

Usage Example

The following code blocks demonstrates how to use the types described in this chapter to access a Special Function Register on an AVR microcontroller:

```
#include "avr/register.hpp"

struct mega328p {
  ~mega328p() = delete;

  using pinb = pin_register<0x23, 0b11111111>;
  using ddrb = ddr_register<0x24, 0b11111111>;
```

(continues on next page)

(continued from previous page)

```
  using portb = port_register<0x25, 0b11111111>;

  using pinc = pin_register<0x26, 0b01111111>;
  using ddrc = ddr_register<0x27, 0b01111111>;
  using portc = port_register<0x28, 0b01111111>;

  using pind = pin_register<0x29, 0b11111111>;
  using ddrd = ddr_register<0x2a, 0b11111111>;
  using portd = port_register<0x2b, 0b11111111>;
};

int main() {
  if(mega328p::portb::port<3>::get()) {
    mega328p::portb::port<7>::set();
  } else {
    mega328p::portb::port<7>::clear();
  }
}
```

The machine code generated by the above C++ code demonstrates that there is no overhead incurred by the wrappers. At the same time, all accesses to bits and registers are checked at compile time

```
00000080 <main>:
  80: 1b 9b           sbis    0x03, 3 ; 3
  82: 04 c0           rjmp    .+8             ; 0x8c <main+0xc>
  84: 1f 9a           sbi     0x03, 7 ; 3
  86: 90 e0           ldi     r25, 0x00       ; 0
  88: 80 e0           ldi     r24, 0x00       ; 0
  8a: 08 95           ret
  8c: 1f 98           cbi     0x03, 7 ; 3
  8e: fb cf           rjmp    .-10            ; 0x86 <main+0x6>
```

Base Type

template <avr::ptrdiff_t *Address*, *avr::uint_for_size_t*< 8 > *Bits*, *avr::uint_for_size_t*< Bits > *ValidBits*>
class special_function_register

A safe wrapper for AVR **Special Function Registers**

This class provides a safe and zero-cost abstraction for the Spcial Purpose Registers found in AVR microcontrollers. All accesses to a register as well as its bits are bound checked, and violations of these bounds will cause compilation to fail.

Since 1.0.0

Template Parameters

- `Address`: The register's address in memory
- `Base`: The address base of the register
- `Bits`: The register's size in Bits
- `ValidBits`: The available bits in the register
- `EffectiveAddress`: The actual address of the register in memory space

Subclassed by ro_io_register< Address, Bits, ValidBits >, rw_special_function_register< Address, Bits, ValidBits >

Full Register Access

static auto **get** ()
Get the full contents of the register.

This function enables the retrieval of the full contents of the underlying register.

See *avr::rw_special_function_register::set*

static auto **set** (value_type **const** *value*)
Set the value of the complete register.

Contrary to single-bit modifications via avr::bit, this functions can be used to set the value of the whole register by writing the register's full content.

See *avr::special_function_register::get*

Bitwise Register Access

Warning: doxygentypedef: Cannot find typedef "avr::rw_special_function_register::bit" in doxygen xml output for project "AVR++" from directory: /home/docs/checkouts/readthedocs.org/user_builds/avrxx/checkouts/latest/doc/xml

Warning: doxygentypedef: Cannot find typedef "avr::ro_special_function_register::bit" in doxygen xml output for project "AVR++" from directory: /home/docs/checkouts/readthedocs.org/user_builds/avrxx/checkouts/latest/doc/xml

Warning: doxygenstruct: Cannot find class "avr::bit" in doxygen xml output for project "AVR++" from directory: /home/docs/checkouts/readthedocs.org/user_builds/avrxx/checkouts/latest/doc/xml

Warning: doxygenstruct: Cannot find class "avr::rw_bit" in doxygen xml output for project "AVR++" from directory: /home/docs/checkouts/readthedocs.org/user_builds/avrxx/checkouts/latest/doc/xml

Specialized Types and Aliases

Warning: doxygenstruct: Cannot find class "avr::pin_register" in doxygen xml output for project "AVR++" from directory: /home/docs/checkouts/readthedocs.org/user_builds/avrxx/checkouts/latest/doc/xml

Warning: doxygenstruct: Cannot find class "avr::ddr_register" in doxygen xml output for project "AVR++" from directory: /home/docs/checkouts/readthedocs.org/user_builds/avrxx/checkouts/latest/doc/xml

Warning: doxygenstruct: Cannot find class “avr::port_register” in doxygen xml output for project “AVR++” from directory: /home/docs/checkouts/readthedocs.org/user_builds/avrxx/checkouts/latest/doc/xml

Index