Thermostat

0.1

Generated by Doxygen 1.10.0

Chapter 1

Themostat Firmware

1.1 Description

This embedded firmware is the thermostat mode.

1.1.1 Micro Controller Pins

ATtiny404

1. VDD(+5v)
2. PA4(ADC_LOAD1)
3. PA5(ADC_LOAD2)
4. PA6(ADC_LOAD3)
5. PA7(zerocrossing)
6. PB3(EN2)
7. PB2(EN3)
8. PB1(SDA)
9. PB0(SCL)
10. RST(NC)
11. PA1(G1)
12. PA2(EN1)
13. PA3(NC)
14. VSS(GND)

key NC:: Not Connected PBX:: Port B pin X PAX:: Port A pin X RST:: Reset pin

1.2 Project Layout

Tree -L 1, output

```
.
 avr-gcc-toolchain.cmake
 build
 CMakeLists.txt
 compile_commands.json -> ./build/compile_commands.json
 docs
 Doxyfile
 inc
 mocks
 otto.sh
 README.md
 setup.sh
 src
 tests

6 directories, 7 files
```

The source code required to run/build the project is in the `/src` directory, with the headers residing inside the `/inc` directory for most public modules.

All other directories are for supporting the development cycle and are used for testitng the code base to ensure accuracy and quality. These are contained in the `tests` and `mocks` directories.

Documentation that has been generated is inside the docs folder which contains the html output that can be browswed via your regular web browser.

PDF genreation from the doumentaiton is also possible if enabled through the `Doxyfile` inside the projects root directory.

The build directory contains the output and makefiles genrerated automatically when using CMake.

This build directory also holds the bin files genreated along with the hex and elf files.

1.3 Build Requirements

- AVR-GCC toolchain OR XC8 from microchip.
- Make OR CMAKE
- avrdude
- A AVR programmer, usbasp for older chips and UPDI for newer ones.

1.4 Dev Requirements

- ATtiny404 series micro-controller
- AVR-GCC toolchain.
- Cmake
- cpputest(Unit testing harness.)
- Doxygen(For documentation)
- Git(For version control)

1.5 Running Unit Tests

To run the cppunit tests you can use the included shell script inside a bash shell.

```
echo "1" | ./otto.sh
```

The command runs the otto script which automates parts of the development cycle such as running and building tests, compiling hex files and flashing the code to a micro-controller using arvdude.

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 3

File Index

3.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 4

Class Documentation

4.1 AC_struct Struct Reference

Public Attributes

- register8_t **CTRLA**
- register8_t **reserved_0x01**
- register8_t **MUXCTRLA**
- register8_t **reserved_0x03**
- register8_t **reserved_0x04**
- register8_t **reserved_0x05**
- register8_t **INTCTRL**
- register8_t **STATUS**

The documentation for this struct was generated from the following file:

- inc/iotn404.h

4.2 ADC_struct Struct Reference

Public Member Functions

- **_WORDREGISTER** (RES)
- **_WORDREGISTER** (WINLT)
- **_WORDREGISTER** (WINHT)

Public Attributes

- register8_t **CTRLA**
- register8_t **CTRLB**
- register8_t **CTRLC**
- register8_t **CTRLD**
- register8_t **CTRLE**
- register8_t **SAMPCTRL**
- register8_t **MUXPOS**
- register8_t **reserved_0x07**
- register8_t **COMMAND**
- register8_t **EVCTRL**
- register8_t **INTCTRL**
- register8_t **INTFLAGS**
- register8_t **DBGCTRL**
- register8_t **TEMP**
- register8_t **reserved_0x0E**
- register8_t **reserved_0x0F**
- register8_t **CALIB**
- register8_t **reserved_0x17**

The documentation for this struct was generated from the following file:

- inc/iotn404.h

4.3 BOD_struct Struct Reference

Public Attributes

- register8_t **CTRLA**
- register8_t **CTRLB**
- register8_t **reserved_0x02**
- register8_t **reserved_0x03**
- register8_t **reserved_0x04**
- register8_t **reserved_0x05**
- register8_t **reserved_0x06**
- register8_t **reserved_0x07**
- register8_t **VLMCTRLA**
- register8_t **INTCTRL**
- register8_t **INTFLAGS**
- register8_t **STATUS**
- register8_t **reserved_0x0C**
- register8_t **reserved_0x0D**
- register8_t **reserved_0x0E**
- register8_t **reserved_0x0F**

The documentation for this struct was generated from the following file:

- inc/iotn404.h

4.4 CCL_struct Struct Reference

Public Attributes

- register8_t **CTRLA**
- register8_t **SEQCTRL0**
- register8_t **reserved_0x02**
- register8_t **reserved_0x03**
- register8_t **reserved_0x04**
- register8_t **LUT0CTRLA**
- register8_t **LUT0CTRLB**
- register8_t **LUT0CTRLC**
- register8_t **TRUTH0**
- register8_t **LUT1CTRLA**
- register8_t **LUT1CTRLB**
- register8_t **LUT1CTRLC**
- register8_t **TRUTH1**
- register8_t **reserved_0x0D**
- register8_t **reserved_0x0E**
- register8_t **reserved_0x0F**

The documentation for this struct was generated from the following file:

- inc/iotn404.h

4.5 CLKCTRL_struct Struct Reference

Public Attributes

- register8_t **MCLKCTRLA**
- register8_t **MCLKCTRLB**
- register8_t **MCLKLOCK**
- register8_t **MCLKSTATUS**
- register8_t **reserved_0x04**
- register8_t **reserved_0x05**
- register8_t **reserved_0x06**
- register8_t **reserved_0x07**
- register8_t **reserved_0x08**
- register8_t **reserved_0x09**
- register8_t **reserved_0x0A**
- register8_t **reserved_0x0B**
- register8_t **reserved_0x0C**
- register8_t **reserved_0x0D**
- register8_t **reserved_0x0E**
- register8_t **reserved_0x0F**
- register8_t **OSC20MCTRLA**
- register8_t **OSC20MCALIBA**
- register8_t **OSC20MCALIBB**
- register8_t **reserved_0x13**
- register8_t **reserved_0x14**
- register8_t **reserved_0x15**

- register8_t **reserved_0x16**
- register8_t **reserved_0x17**
- register8_t **OSC32KCTRLA**
- register8_t **reserved_0x19**
- register8_t **reserved_0x1A**
- register8_t **reserved_0x1B**
- register8_t **reserved_0x1C**
- register8_t **reserved_0x1D**
- register8_t **reserved_0x1E**
- register8_t **reserved_0x1F**

The documentation for this struct was generated from the following file:

- inc/iotn404.h

4.6 CPUINT_struct Struct Reference

Public Attributes

- register8_t **CTRLA**
- register8_t **STATUS**
- register8_t **LVL0PRI**
- register8_t **LVL1VEC**

The documentation for this struct was generated from the following file:

- inc/iotn404.h

4.7 CRCSCAN_struct Struct Reference

Public Attributes

- register8_t **CTRLA**
- register8_t **CTRLB**
- register8_t **STATUS**
- register8_t **reserved_0x03**

The documentation for this struct was generated from the following file:

- inc/iotn404.h

4.8 EVSYS_struct Struct Reference

Public Attributes

- register8_t **ASYNCSTROBE**
- register8_t **SYNCSTROBE**
- register8_t **ASYNCCH0**
- register8_t **ASYNCCH1**
- register8_t **reserved_0x04**
- register8_t **reserved_0x05**
- register8_t **reserved_0x06**
- register8_t **reserved_0x07**
- register8_t **reserved_0x08**
- register8_t **reserved_0x09**
- register8_t **SYNCCH0**
- register8_t **reserved_0x0B**
- register8_t **reserved_0x0C**
- register8_t **reserved_0x0D**
- register8_t **reserved_0x0E**
- register8_t **reserved_0x0F**
- register8_t **reserved_0x10**
- register8_t **reserved_0x11**
- register8_t **ASYNCUSER0**
- register8_t **ASYNCUSER1**
- register8_t **ASYNCUSER2**
- register8_t **ASYNCUSER3**
- register8_t **ASYNCUSER4**
- register8_t **ASYNCUSER5**
- register8_t **ASYNCUSER6**
- register8_t **ASYNCUSER7**
- register8_t **ASYNCUSER8**
- register8_t **ASYNCUSER9**
- register8_t **ASYNCUSER10**
- register8_t **reserved_0x1D**
- register8_t **reserved_0x1E**
- register8_t **reserved_0x1F**
- register8_t **reserved_0x20**
- register8_t **reserved_0x21**
- register8_t **SYNCUSER0**
- register8_t **SYNCUSER1**
- register8_t **reserved_0x24**
- register8_t **reserved_0x25**
- register8_t **reserved_0x26**
- register8_t **reserved_0x27**
- register8_t **reserved_0x28**
- register8_t **reserved_0x29**
- register8_t **reserved_0x2A**
- register8_t **reserved_0x2B**
- register8_t **reserved_0x2C**
- register8_t **reserved_0x2D**
- register8_t **reserved_0x2E**
- register8_t **reserved_0x2F**
- register8_t **reserved_0x30**
- register8_t **reserved_0x31**

- register8_t **reserved_0x32**
- register8_t **reserved_0x33**
- register8_t **reserved_0x34**
- register8_t **reserved_0x35**
- register8_t **reserved_0x36**
- register8_t **reserved_0x37**
- register8_t **reserved_0x38**
- register8_t **reserved_0x39**
- register8_t **reserved_0x3A**
- register8_t **reserved_0x3B**
- register8_t **reserved_0x3C**
- register8_t **reserved_0x3D**
- register8_t **reserved_0x3E**
- register8_t **reserved_0x3F**

The documentation for this struct was generated from the following file:

- inc/iotn404.h

4.9 FUSE_struct Struct Reference

Public Attributes

- register8_t **WDTCFG**
- register8_t **BODCFG**
- register8_t **OSCCFG**
- register8_t **reserved_0x03**
- register8_t **TCD0CFG**
- register8_t **SYSCFG0**
- register8_t **SYSCFG1**
- register8_t **APPEND**
- register8_t **BOOTEND**

The documentation for this struct was generated from the following file:

- inc/iotn404.h

4.10 LOCKBIT_struct Struct Reference

Public Attributes

- register8_t **LOCKBIT**

The documentation for this struct was generated from the following file:

- inc/iotn404.h

4.11 NVMCTRL_struct Struct Reference

Public Member Functions

- **_WORDREGISTER** (DATA)
- **_WORDREGISTER** (ADDR)

Public Attributes

- register8_t **CTRLA**
- register8_t **CTRLB**
- register8_t **STATUS**
- register8_t **INTCTRL**
- register8_t **INTFLAGS**
- register8_t **reserved_0x05**
- register8_t **reserved_0x0A**
- register8_t **reserved_0x0B**
- register8_t **reserved_0x0C**
- register8_t **reserved_0x0D**
- register8_t **reserved_0x0E**
- register8_t **reserved_0x0F**

The documentation for this struct was generated from the following file:

- inc/iotn404.h

4.12 PORT_struct Struct Reference

Public Attributes

- register8_t **DIR**
- register8_t **DIRSET**
- register8_t **DIRCLR**
- register8_t **DIRTGL**
- register8_t **OUT**
- register8_t **OUTSET**
- register8_t **OUTCLR**
- register8_t **OUTTGL**
- register8_t **IN**
- register8_t **INTFLAGS**
- register8_t **reserved_0x0A**
- register8_t **reserved_0x0B**
- register8_t **reserved_0x0C**
- register8_t **reserved_0x0D**
- register8_t **reserved_0x0E**
- register8_t **reserved_0x0F**
- register8_t **PIN0CTRL**
- register8_t **PIN1CTRL**
- register8_t **PIN2CTRL**
- register8_t **PIN3CTRL**

- register8_t **PIN4CTRL**
- register8_t **PIN5CTRL**
- register8_t **PIN6CTRL**
- register8_t **PIN7CTRL**
- register8_t **reserved_0x18**
- register8_t **reserved_0x19**
- register8_t **reserved_0x1A**
- register8_t **reserved_0x1B**
- register8_t **reserved_0x1C**
- register8_t **reserved_0x1D**
- register8_t **reserved_0x1E**
- register8_t **reserved_0x1F**

The documentation for this struct was generated from the following file:

- inc/iotn404.h

4.13 PORTMUX_struct Struct Reference

Public Attributes

- register8_t **CTRLA**
- register8_t **CTRLB**
- register8_t **CTRLC**
- register8_t **CTRLD**
- register8_t **reserved_0x04**
- register8_t **reserved_0x05**
- register8_t **reserved_0x06**
- register8_t **reserved_0x07**
- register8_t **reserved_0x08**
- register8_t **reserved_0x09**
- register8_t **reserved_0x0A**
- register8_t **reserved_0x0B**
- register8_t **reserved_0x0C**
- register8_t **reserved_0x0D**
- register8_t **reserved_0x0E**
- register8_t **reserved_0x0F**

The documentation for this struct was generated from the following file:

- inc/iotn404.h

4.14 RSTCTRL_struct Struct Reference

Public Attributes

- register8_t **RSTFR**
- register8_t **SWRR**
- register8_t **reserved_0x02**
- register8_t **reserved_0x03**

The documentation for this struct was generated from the following file:

- inc/iotn404.h

4.15 RTC_struct Struct Reference

Public Member Functions

- **_WORDREGISTER** (CNT)
- **_WORDREGISTER** (PER)
- **_WORDREGISTER** (CMP)

Public Attributes

- register8_t **CTRLA**
- register8_t **STATUS**
- register8_t **INTCTRL**
- register8_t **INTFLAGS**
- register8_t **TEMP**
- register8_t **DBGCTRL**
- register8_t **reserved_0x06**
- register8_t **CLKSEL**
- register8_t **reserved_0x0E**
- register8_t **reserved_0x0F**
- register8_t **PITCTRLA**
- register8_t **PITSTATUS**
- register8_t **PITINTCTRL**
- register8_t **PITINTFLAGS**
- register8_t **reserved_0x14**
- register8_t **PITDBGCTRL**
- register8_t **reserved_0x16**
- register8_t **reserved_0x17**
- register8_t **reserved_0x18**
- register8_t **reserved_0x19**
- register8_t **reserved_0x1A**
- register8_t **reserved_0x1B**
- register8_t **reserved_0x1C**
- register8_t **reserved_0x1D**
- register8_t **reserved_0x1E**
- register8_t **reserved_0x1F**

The documentation for this struct was generated from the following file:

- inc/iotn404.h

4.16 SIGROW_struct Struct Reference

Public Attributes

- register8_t **DEVICEID0**
- register8_t **DEVICEID1**
- register8_t **DEVICEID2**
- register8_t **SERNUM0**
- register8_t **SERNUM1**
- register8_t **SERNUM2**
- register8_t **SERNUM3**
- register8_t **SERNUM4**
- register8_t **SERNUM5**
- register8_t **SERNUM6**
- register8_t **SERNUM7**
- register8_t **SERNUM8**
- register8_t **SERNUM9**
- register8_t **reserved_0x0D**
- register8_t **reserved_0x0E**
- register8_t **reserved_0x0F**
- register8_t **reserved_0x10**
- register8_t **reserved_0x11**
- register8_t **reserved_0x12**
- register8_t **reserved_0x13**
- register8_t **reserved_0x14**
- register8_t **reserved_0x15**
- register8_t **reserved_0x16**
- register8_t **reserved_0x17**
- register8_t **reserved_0x18**
- register8_t **reserved_0x19**
- register8_t **reserved_0x1A**
- register8_t **reserved_0x1B**
- register8_t **reserved_0x1C**
- register8_t **reserved_0x1D**
- register8_t **reserved_0x1E**
- register8_t **reserved_0x1F**
- register8_t **TEMPSENSE0**
- register8_t **TEMPSENSE1**
- register8_t **OSC16ERR3V**
- register8_t **OSC16ERR5V**
- register8_t **OSC20ERR3V**
- register8_t **OSC20ERR5V**
- register8_t **reserved_0x26**
- register8_t **reserved_0x27**
- register8_t **reserved_0x28**
- register8_t **reserved_0x29**
- register8_t **reserved_0x2A**
- register8_t **reserved_0x2B**
- register8_t **reserved_0x2C**
- register8_t **reserved_0x2D**
- register8_t **reserved_0x2E**
- register8_t **reserved_0x2F**
- register8_t **reserved_0x30**
- register8_t **reserved_0x31**

- register8_t **reserved_0x32**
- register8_t **reserved_0x33**
- register8_t **reserved_0x34**
- register8_t **reserved_0x35**
- register8_t **reserved_0x36**
- register8_t **reserved_0x37**
- register8_t **reserved_0x38**
- register8_t **reserved_0x39**
- register8_t **reserved_0x3A**
- register8_t **reserved_0x3B**
- register8_t **reserved_0x3C**
- register8_t **reserved_0x3D**
- register8_t **reserved_0x3E**
- register8_t **reserved_0x3F**

The documentation for this struct was generated from the following file:

- inc/iotn404.h

4.17 SLPCTRL_struct Struct Reference

Public Attributes

- register8_t **CTRLA**
- register8_t **reserved_0x01**

The documentation for this struct was generated from the following file:

- inc/iotn404.h

4.18 SPI_struct Struct Reference

Public Attributes

- register8_t **CTRLA**
- register8_t **CTRLB**
- register8_t **INTCTRL**
- register8_t **INTFLAGS**
- register8_t **DATA**
- register8_t **reserved_0x05**
- register8_t **reserved_0x06**
- register8_t **reserved_0x07**

The documentation for this struct was generated from the following file:

- inc/iotn404.h

4.19 SYSCFG_struct Struct Reference

Public Attributes

- register8_t **reserved_0x00**
- register8_t **REVID**
- register8_t **EXTBRK**
- register8_t **reserved_0x03**
- register8_t **reserved_0x04**
- register8_t **reserved_0x05**
- register8_t **reserved_0x06**
- register8_t **reserved_0x07**
- register8_t **reserved_0x08**
- register8_t **reserved_0x09**
- register8_t **reserved_0x0A**
- register8_t **reserved_0x0B**
- register8_t **reserved_0x0C**
- register8_t **reserved_0x0D**
- register8_t **reserved_0x0E**
- register8_t **reserved_0x0F**
- register8_t **reserved_0x10**
- register8_t **reserved_0x11**
- register8_t **reserved_0x12**
- register8_t **reserved_0x13**
- register8_t **reserved_0x14**
- register8_t **reserved_0x15**
- register8_t **reserved_0x16**
- register8_t **reserved_0x17**
- register8_t **reserved_0x18**
- register8_t **reserved_0x19**
- register8_t **reserved_0x1A**
- register8_t **reserved_0x1B**
- register8_t **reserved_0x1C**
- register8_t **reserved_0x1D**
- register8_t **reserved_0x1E**
- register8_t **reserved_0x1F**

The documentation for this struct was generated from the following file:

- inc/iotn404.h

4.20 TCA_SINGLE_struct Struct Reference

Public Member Functions

- **_WORDREGISTER** (CNT)
- **_WORDREGISTER** (PER)
- **_WORDREGISTER** (CMP0)
- **_WORDREGISTER** (CMP1)
- **_WORDREGISTER** (CMP2)
- **_WORDREGISTER** (PERBUF)
- **_WORDREGISTER** (CMP0BUF)
- **_WORDREGISTER** (CMP1BUF)
- **_WORDREGISTER** (CMP2BUF)

Public Attributes

- register8_t **CTRLA**
- register8_t **CTRLB**
- register8_t **CTRLC**
- register8_t **CTRLD**
- register8_t **CTRLECLR**
- register8_t **CTRLESET**
- register8_t **CTRLFCLR**
- register8_t **CTRLFSET**
- register8_t **reserved_0x08**
- register8_t **EVCTRL**
- register8_t **INTCTRL**
- register8_t **INTFLAGS**
- register8_t **reserved_0x0C**
- register8_t **reserved_0x0D**
- register8_t **DBGCTRL**
- register8_t **TEMP**
- register8_t **reserved_0x10**
- register8_t **reserved_0x11**
- register8_t **reserved_0x12**
- register8_t **reserved_0x13**
- register8_t **reserved_0x14**
- register8_t **reserved_0x15**
- register8_t **reserved_0x16**
- register8_t **reserved_0x17**
- register8_t **reserved_0x18**
- register8_t **reserved_0x19**
- register8_t **reserved_0x1A**
- register8_t **reserved_0x1B**
- register8_t **reserved_0x1C**
- register8_t **reserved_0x1D**
- register8_t **reserved_0x1E**
- register8_t **reserved_0x1F**
- register8_t **reserved_0x22**
- register8_t **reserved_0x23**
- register8_t **reserved_0x24**
- register8_t **reserved_0x25**
- register8_t **reserved_0x2E**
- register8_t **reserved_0x2F**
- register8_t **reserved_0x30**
- register8_t **reserved_0x31**
- register8_t **reserved_0x32**
- register8_t **reserved_0x33**
- register8_t **reserved_0x34**
- register8_t **reserved_0x35**
- register8_t **reserved_0x3E**
- register8_t **reserved_0x3F**

The documentation for this struct was generated from the following file:

- inc/iotn404.h

4.21 TCA_SPLIT_struct Struct Reference

Public Attributes

- register8_t **CTRLA**
- register8_t **CTRLB**
- register8_t **CTRLC**
- register8_t **CTRLD**
- register8_t **CTRLECLR**
- register8_t **CTRLESET**
- register8_t **reserved_0x06**
- register8_t **reserved_0x07**
- register8_t **reserved_0x08**
- register8_t **reserved_0x09**
- register8_t **INTCTRL**
- register8_t **INTFLAGS**
- register8_t **reserved_0x0C**
- register8_t **reserved_0x0D**
- register8_t **DBGCTRL**
- register8_t **reserved_0x0F**
- register8_t **reserved_0x10**
- register8_t **reserved_0x11**
- register8_t **reserved_0x12**
- register8_t **reserved_0x13**
- register8_t **reserved_0x14**
- register8_t **reserved_0x15**
- register8_t **reserved_0x16**
- register8_t **reserved_0x17**
- register8_t **reserved_0x18**
- register8_t **reserved_0x19**
- register8_t **reserved_0x1A**
- register8_t **reserved_0x1B**
- register8_t **reserved_0x1C**
- register8_t **reserved_0x1D**
- register8_t **reserved_0x1E**
- register8_t **reserved_0x1F**
- register8_t **LCNT**
- register8_t **HCNT**
- register8_t **reserved_0x22**
- register8_t **reserved_0x23**
- register8_t **reserved_0x24**
- register8_t **reserved_0x25**
- register8_t **LPER**
- register8_t **HPER**
- register8_t **LCMP0**
- register8_t **HCMP0**
- register8_t **LCMP1**
- register8_t **HCMP1**
- register8_t **LCMP2**
- register8_t **HCMP2**
- register8_t **reserved_0x2E**
- register8_t **reserved_0x2F**
- register8_t **reserved_0x30**
- register8_t **reserved_0x31**

- register8_t **reserved_0x32**
- register8_t **reserved_0x33**
- register8_t **reserved_0x34**
- register8_t **reserved_0x35**
- register8_t **reserved_0x36**
- register8_t **reserved_0x37**
- register8_t **reserved_0x38**
- register8_t **reserved_0x39**
- register8_t **reserved_0x3A**
- register8_t **reserved_0x3B**
- register8_t **reserved_0x3C**
- register8_t **reserved_0x3D**
- register8_t **reserved_0x3E**
- register8_t **reserved_0x3F**

The documentation for this struct was generated from the following file:

- inc/iotn404.h

4.22 TCA_union Union Reference

Public Attributes

- TCA_SINGLE_t **SINGLE**
- TCA_SPLIT_t **SPLIT**

The documentation for this union was generated from the following file:

- inc/iotn404.h

4.23 TCB_struct Struct Reference

Public Member Functions

- **_WORDREGISTER** (CNT)
- **_WORDREGISTER** (CCMP)

Public Attributes

- register8_t **CTRLA**
- register8_t **CTRLB**
- register8_t **reserved_0x02**
- register8_t **reserved_0x03**
- register8_t **EVCTRL**
- register8_t **INTCTRL**
- register8_t **INTFLAGS**
- register8_t **STATUS**
- register8_t **DBGCTRL**
- register8_t **TEMP**
- register8_t **reserved_0x0E**
- register8_t **reserved_0x0F**

The documentation for this struct was generated from the following file:

- inc/iotn404.h

4.24 TWI_struct Struct Reference

Public Attributes

- register8_t **CTRLA**
- register8_t **reserved_0x01**
- register8_t **DBGCTRL**
- register8_t **MCTRLA**
- register8_t **MCTRLB**
- register8_t **MSTATUS**
- register8_t **MBAUD**
- register8_t **MADDR**
- register8_t **MDATA**
- register8_t **SCTRLA**
- register8_t **SCTRLB**
- register8_t **SSTATUS**
- register8_t **SADDR**
- register8_t **SDATA**
- register8_t **SADDRMASK**
- register8_t **reserved_0x0F**

The documentation for this struct was generated from the following file:

- inc/iotn404.h

4.25 USART_struct Struct Reference

Public Member Functions

- **_WORDREGISTER** (BAUD)

Public Attributes

- register8_t **RXDATAL**
- register8_t **RXDATAH**
- register8_t **TXDATAL**
- register8_t **TXDATAH**
- register8_t **STATUS**
- register8_t **CTRLA**
- register8_t **CTRLB**
- register8_t **CTRLC**
- register8_t **reserved_0x0A**
- register8_t **DBGCTRL**
- register8_t **EVCTRL**
- register8_t **TXPLCTRL**
- register8_t **RXPLCTRL**
- register8_t **reserved_0x0F**

The documentation for this struct was generated from the following file:

- inc/iotn404.h

4.26 USERROW_struct Struct Reference

Public Attributes

- register8_t **USERROW0**
- register8_t **USERROW1**
- register8_t **USERROW2**
- register8_t **USERROW3**
- register8_t **USERROW4**
- register8_t **USERROW5**
- register8_t **USERROW6**
- register8_t **USERROW7**
- register8_t **USERROW8**
- register8_t **USERROW9**
- register8_t **USERROW10**
- register8_t **USERROW11**
- register8_t **USERROW12**
- register8_t **USERROW13**
- register8_t **USERROW14**
- register8_t **USERROW15**
- register8_t **USERROW16**
- register8_t **USERROW17**
- register8_t **USERROW18**
- register8_t **USERROW19**
- register8_t **USERROW20**
- register8_t **USERROW21**
- register8_t **USERROW22**
- register8_t **USERROW23**
- register8_t **USERROW24**
- register8_t **USERROW25**
- register8_t **USERROW26**
- register8_t **USERROW27**
- register8_t **USERROW28**
- register8_t **USERROW29**
- register8_t **USERROW30**
- register8_t **USERROW31**

The documentation for this struct was generated from the following file:

- inc/iotn404.h

4.27 VPORT_struct Struct Reference

Public Attributes

- register8_t **DIR**
- register8_t **OUT**
- register8_t **IN**
- register8_t **INTFLAGS**

The documentation for this struct was generated from the following file:

- inc/iotn404.h

4.28 VREF_struct Struct Reference

Public Attributes

- register8_t **CTRLA**
- register8_t **CTRLB**

The documentation for this struct was generated from the following file:

- inc/iotn404.h

4.29 WDT_struct Struct Reference

Public Attributes

- register8_t **CTRLA**
- register8_t **STATUS**

The documentation for this struct was generated from the following file:

- inc/iotn404.h

Chapter 5

File Documentation

5.1 inc/config.h File Reference

File contains the project configuration values.

```
#include <stdint.h>
```

Macros

- #define **ADC_LOAD1** 4
- #define **ADC_LOAD2** 5
- #define **ADC_LOAD3** 6
- #define HYSTERESIS_HI 900

  The upper Hystersis value.(Override) This is the upper bound of the digital/boolean hysteresis curve. It takes presidence over the value in `/src/load.h`
- #define HYSTERESIS_LO 700

  The lower Hystersis value.(Override) This is the upper bound of the digital/boolean hysteresis curve. It takes presidence over the value in `/src/load.h`

Variables

- const uint16_t TriggerValue = 512

  Positive Zero Crossing Trigger Value The 10 bit value at which the program triggers the ISR to handle the zero crossing event.
- const int GatePulsesQty = 5

  Triac Gate Pulse Quantity.
- const uint16_t GatePulses [5] = {250, 500, 750, 1000, 1250}

  Gate Pulses Array.
- const double **Tau** = 8250

  The time constant.

5.1.1 Detailed Description

File contains the project configuration values.

Author

Jake G

Date

15 June 2024

This file contains the user changable parameters. Most the values are constants that will change the behavior of the system.

For these changes to take affect you must recompile/rebuild the project after you have changed the values.

5.1.2 Macro Definition Documentation

5.1.2.1 HYSTERESIS_HI

```
#define HYSTERESIS_HI 900
```

The upper Hystersis value.(Override) This is the upper bound of the digital/boolean hysteresis curve. It takes presidence over the value in `/src/load.h`

If the input is below low it always sets the output high. If the input is between high and low with the output high it stays high. If the input is above high and the ouput is high it's set low.

5.1.2.2 HYSTERESIS_LO

```
#define HYSTERESIS_LO 700
```

The lower Hystersis value.(Override) This is the upper bound of the digital/boolean hysteresis curve. It takes presidence over the value in `/src/load.h`

If the input is below low it always sets the output high. If the input is between high and low with the output high it stays high. If the input is above high and the ouput is high it's set low.

5.1.3 Variable Documentation

5.1.3.1 GatePulses

```
const uint16_t GatePulses[5] = {250, 500, 750, 1000, 1250}
```

Gate Pulses Array.

The gate pulses array holds the duration of pulses in microseconds for the triacs gates. The length of the array must be matched with the GatePulsesQuantity parameter.

5.1.3.2 GatePulsesQty

```
const int GatePulsesQty = 5
```

Triac Gate Pulse Quantity.

Contains the number of pulses that the micro-controller will send to the gates of the triac.

This number should match the quantity of timings inside the pulse/duration array.

5.1.3.3 TriggerValue

```
const uint16_t TriggerValue = 512
```

Positive Zero Crossing Trigger Value The 10 bit value at which the program triggers the ISR to handle the zero crossing event.

You can adjust this to change when the program will start the timer.

5.2 config.h

Go to the documentation of this file.

```
00001
00014 #ifndef CONFIG_H
00015 #define CONFIG_H
00016
00017 #include <stdint.h>
00018
00019 #define ADC_LOAD1 4
00020 #define ADC_LOAD2 5
00021 #define ADC_LOAD3 6
00022
00030 const uint16_t TriggerValue = 512;
00031
00032
00042 const int GatePulsesQty = 5;
00043
00051 const uint16_t GatePulses[5] = {250, 500, 750, 1000, 1250};
00052
00056 const double Tau = 8250;
00057
00058
00059
00069 #define HYSTERESIS_HI 900
00070
00071
00081 #define HYSTERESIS_LO 700
00082
00083
00084 #endif //CONFIG_H
```

5.3 iotn404.h

```
00001
00002 /* This file is part of avr-libc.
00003
00004    Automatically created by devtools/gen-ioheader-atdf-avr8x.py
00005    DO NOT EDIT!
00006
00007    Redistribution and use in source and binary forms, with or without
00008    modification, are permitted provided that the following conditions are met:
00009
00010    * Redistributions of source code must retain the above copyright
00011      notice, this list of conditions and the following disclaimer.
00012
```

```
00013     * Redistributions in binary form must reproduce the above copyright
00014       notice, this list of conditions and the following disclaimer in
00015       the documentation and/or other materials provided with the
00016       distribution.
00017
00018     * Neither the name of the copyright holders nor the names of
00019       contributors may be used to endorse or promote products derived
00020       from this software without specific prior written permission.
00021
00022   THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
00023   AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
00024   IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
00025   ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE
00026   LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
00027   CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
00028   SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
00029   INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
00030   CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
00031   ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
00032   POSSIBILITY OF SUCH DAMAGE. */
00033
00034
00035
00036
00037 #ifndef _AVR_IO_H_
00038 #  error "Include <avr/io.h> instead of this file."
00039 #endif
00040
00041 #ifndef _AVR_IOXXX_H_
00042 #  define _AVR_IOXXX_H_ "iotn404.h"
00043 #else
00044 #  error "Attempt to include more than one <avr/ioXXX.h> file."
00045 #endif
00046
00047 #ifndef _AVR_ATTINY404_H_INCLUDED
00048 #define _AVR_ATTINY404_H_INCLUDED
00049
00050 /* Ungrouped common registers */
00051 #define CCP  _SFR_MEM8(0x0034)  /* Configuration Change Protection */
00052 #define SPH  _SFR_MEM8(0x003E)  /* Stack Pointer High */
00053 #define SPL  _SFR_MEM8(0x003D)  /* Stack Pointer Low */
00054 #define SREG  _SFR_MEM8(0x003F)  /* Status Register */
00055
00056 #define GPIOR0  _SFR_MEM8(0x001C)  /* General Purpose IO Register 0 */
00057 #define GPIOR1  _SFR_MEM8(0x001D)  /* General Purpose IO Register 1 */
00058 #define GPIOR2  _SFR_MEM8(0x001E)  /* General Purpose IO Register 2 */
00059 #define GPIOR3  _SFR_MEM8(0x001F)  /* General Purpose IO Register 3 */
00060
00061 /* Deprecated */
00062 #define GPIO0  _SFR_MEM8(0x001C)  /* General Purpose IO Register 0 */
00063 #define GPIO1  _SFR_MEM8(0x001D)  /* General Purpose IO Register 1 */
00064 #define GPIO2  _SFR_MEM8(0x001E)  /* General Purpose IO Register 2 */
00065 #define GPIO3  _SFR_MEM8(0x001F)  /* General Purpose IO Register 3 */
00066
00067 /* C Language Only */
00068 #if !defined (__ASSEMBLER__)
00069
00070 #include <stdint.h>
00071
00072 typedef volatile uint8_t register8_t;
00073 typedef volatile uint16_t register16_t;
00074 typedef volatile uint32_t register32_t;
00075
00076
00077 #ifdef _WORDREGISTER
00078 #undef _WORDREGISTER
00079 #endif
00080 #define _WORDREGISTER(regname)   \
00081     __extension__ union \
00082     { \
00083         register16_t regname; \
00084         struct \
00085         { \
00086             register8_t regname ## L; \
00087             register8_t regname ## H; \
00088         }; \
00089     }
00090
00091 #ifdef _DWORDREGISTER
00092 #undef _DWORDREGISTER
00093 #endif
00094 #define _DWORDREGISTER(regname)  \
00095     __extension__ union \
00096     { \
00097         register32_t regname; \
00098         struct \
00099         { \
```

```
00100             register8_t regname ## 0; \
00101             register8_t regname ## 1; \
00102             register8_t regname ## 2; \
00103             register8_t regname ## 3; \
00104         }; \
00105     }
00106
00107
00108 /*
00109 ==========================================================================
00110 IO Module Structures
00111 ==========================================================================
00112 */
00113
00114
00115 /*
00116 --------------------------------------------------------------------------
00117 AC - Analog Comparator
00118 --------------------------------------------------------------------------
00119 */
00120
00121 /* Analog Comparator */
00122 typedef struct AC_struct
00123 {
00124     register8_t CTRLA;  /* Control A */
00125     register8_t reserved_0x01;
00126     register8_t MUXCTRLA;  /* Mux Control A */
00127     register8_t reserved_0x03;
00128     register8_t reserved_0x04;
00129     register8_t reserved_0x05;
00130     register8_t INTCTRL;  /* Interrupt Control */
00131     register8_t STATUS;  /* Status */
00132 } AC_t;
00133
00134 /* Hysteresis Mode select */
00135 typedef enum AC_HYSMODE_enum
00136 {
00137     AC_HYSMODE_OFF_gc = (0x00<<1),  /* No hysteresis */
00138     AC_HYSMODE_10mV_gc = (0x01<<1),  /* 10mV hysteresis */
00139     AC_HYSMODE_25mV_gc = (0x02<<1),  /* 25mV hysteresis */
00140     AC_HYSMODE_50mV_gc = (0x03<<1),  /* 50mV hysteresis */
00141 } AC_HYSMODE_t;
00142
00143 /* Interrupt Mode select */
00144 typedef enum AC_INTMODE_enum
00145 {
00146     AC_INTMODE_BOTHEDGE_gc = (0x00<<4),  /* Any Edge */
00147     AC_INTMODE_NEGEDGE_gc = (0x02<<4),  /* Negative Edge */
00148     AC_INTMODE_POSEDGE_gc = (0x03<<4),  /* Positive Edge */
00149 } AC_INTMODE_t;
00150
00151 /* Negative Input MUX Selection select */
00152 typedef enum AC_MUXNEG_enum
00153 {
00154     AC_MUXNEG_PIN0_gc = (0x00<<0),  /* Negative Pin 0 */
00155     AC_MUXNEG_VREF_gc = (0x02<<0),  /* Voltage Reference */
00156 } AC_MUXNEG_t;
00157
00158 /* Positive Input MUX Selection select */
00159 typedef enum AC_MUXPOS_enum
00160 {
00161     AC_MUXPOS_PIN0_gc = (0x00<<3),  /* Positive Pin 0 */
00162 } AC_MUXPOS_t;
00163
00164 /*
00165 --------------------------------------------------------------------------
00166 ADC - Analog to Digital Converter
00167 --------------------------------------------------------------------------
00168 */
00169
00170 /* Analog to Digital Converter */
00171 typedef struct ADC_struct
00172 {
00173     register8_t CTRLA;  /* Control A */
00174     register8_t CTRLB;  /* Control B */
00175     register8_t CTRLC;  /* Control C */
00176     register8_t CTRLD;  /* Control D */
00177     register8_t CTRLE;  /* Control E */
00178     register8_t SAMPCTRL;  /* Sample Control */
00179     register8_t MUXPOS;  /* Positive mux input */
00180     register8_t reserved_0x07;
00181     register8_t COMMAND;  /* Command */
00182     register8_t EVCTRL;  /* Event Control */
00183     register8_t INTCTRL;  /* Interrupt Control */
00184     register8_t INTFLAGS;  /* Interrupt Flags */
00185     register8_t DBGCTRL;  /* Debug Control */
00186     register8_t TEMP;  /* Temporary Data */
```

```
00187     register8_t reserved_0x0E;
00188     register8_t reserved_0x0F;
00189     _WORDREGISTER(RES);  /* ADC Accumulator Result */
00190     _WORDREGISTER(WINLT);  /* Window comparator low threshold */
00191     _WORDREGISTER(WINHT);  /* Window comparator high threshold */
00192     register8_t CALIB;  /* Calibration */
00193     register8_t reserved_0x17;
00194 } ADC_t;
00195
00196 /* Automatic Sampling Delay Variation select */
00197 typedef enum ADC_ASDV_enum
00198 {
00199     ADC_ASDV_ASVOFF_gc = (0x00<<4),  /* The Automatic Sampling Delay Variation is disabled */
00200     ADC_ASDV_ASVON_gc = (0x01<<4),  /* The Automatic Sampling Delay Variation is enabled */
00201 } ADC_ASDV_t;
00202
00203 /* Duty Cycle select */
00204 typedef enum ADC_DUTYCYC_enum
00205 {
00206     ADC_DUTYCYC_DUTY50_gc = (0x00<<0),  /* 50% Duty cycle */
00207     ADC_DUTYCYC_DUTY25_gc = (0x01<<0),  /* 25% Duty cycle */
00208 } ADC_DUTYCYC_t;
00209
00210 /* Initial Delay Selection select */
00211 typedef enum ADC_INITDLY_enum
00212 {
00213     ADC_INITDLY_DLY0_gc = (0x00<<5),  /* Delay 0 CLK_ADC cycles */
00214     ADC_INITDLY_DLY16_gc = (0x01<<5),  /* Delay 16 CLK_ADC cycles */
00215     ADC_INITDLY_DLY32_gc = (0x02<<5),  /* Delay 32 CLK_ADC cycles */
00216     ADC_INITDLY_DLY64_gc = (0x03<<5),  /* Delay 64 CLK_ADC cycles */
00217     ADC_INITDLY_DLY128_gc = (0x04<<5),  /* Delay 128 CLK_ADC cycles */
00218     ADC_INITDLY_DLY256_gc = (0x05<<5),  /* Delay 256 CLK_ADC cycles */
00219 } ADC_INITDLY_t;
00220
00221 /* Analog Channel Selection Bits select */
00222 typedef enum ADC_MUXPOS_enum
00223 {
00224     ADC_MUXPOS_AIN0_gc = (0x00<<0),  /* ADC input pin 0 */
00225     ADC_MUXPOS_AIN1_gc = (0x01<<0),  /* ADC input pin 1 */
00226     ADC_MUXPOS_AIN2_gc = (0x02<<0),  /* ADC input pin 2 */
00227     ADC_MUXPOS_AIN3_gc = (0x03<<0),  /* ADC input pin 3 */
00228     ADC_MUXPOS_AIN4_gc = (0x04<<0),  /* ADC input pin 4 */
00229     ADC_MUXPOS_AIN5_gc = (0x05<<0),  /* ADC input pin 5 */
00230     ADC_MUXPOS_AIN6_gc = (0x06<<0),  /* ADC input pin 6 */
00231     ADC_MUXPOS_AIN7_gc = (0x07<<0),  /* ADC input pin 7 */
00232     ADC_MUXPOS_AIN8_gc = (0x08<<0),  /* ADC input pin 8 */
00233     ADC_MUXPOS_AIN9_gc = (0x09<<0),  /* ADC input pin 9 */
00234     ADC_MUXPOS_AIN10_gc = (0x0A<<0),  /* ADC input pin 10 */
00235     ADC_MUXPOS_AIN11_gc = (0x0B<<0),  /* ADC input pin 11 */
00236     ADC_MUXPOS_DAC0_gc = (0x1C<<0),  /* DAC0 */
00237     ADC_MUXPOS_INTREF_gc = (0x1D<<0),  /* Internal Ref */
00238     ADC_MUXPOS_TEMPSENSE_gc = (0x1E<<0),  /* Temp sensor */
00239     ADC_MUXPOS_GND_gc = (0x1F<<0),  /* GND */
00240 } ADC_MUXPOS_t;
00241
00242 /* Clock Pre-scaler select */
00243 typedef enum ADC_PRESC_enum
00244 {
00245     ADC_PRESC_DIV2_gc = (0x00<<0),  /* CLK_PER divided by 2 */
00246     ADC_PRESC_DIV4_gc = (0x01<<0),  /* CLK_PER divided by 4 */
00247     ADC_PRESC_DIV8_gc = (0x02<<0),  /* CLK_PER divided by 8 */
00248     ADC_PRESC_DIV16_gc = (0x03<<0),  /* CLK_PER divided by 16 */
00249     ADC_PRESC_DIV32_gc = (0x04<<0),  /* CLK_PER divided by 32 */
00250     ADC_PRESC_DIV64_gc = (0x05<<0),  /* CLK_PER divided by 64 */
00251     ADC_PRESC_DIV128_gc = (0x06<<0),  /* CLK_PER divided by 128 */
00252     ADC_PRESC_DIV256_gc = (0x07<<0),  /* CLK_PER divided by 256 */
00253 } ADC_PRESC_t;
00254
00255 /* Reference Selection select */
00256 typedef enum ADC_REFSEL_enum
00257 {
00258     ADC_REFSEL_INTREF_gc = (0x00<<4),  /* Internal reference */
00259     ADC_REFSEL_VDDREF_gc = (0x01<<4),  /* VDD */
00260 } ADC_REFSEL_t;
00261
00262 /* ADC Resolution select */
00263 typedef enum ADC_RESSEL_enum
00264 {
00265     ADC_RESSEL_10BIT_gc = (0x00<<2),  /* 10-bit mode */
00266     ADC_RESSEL_8BIT_gc = (0x01<<2),  /* 8-bit mode */
00267 } ADC_RESSEL_t;
00268
00269 /* Accumulation Samples select */
00270 typedef enum ADC_SAMPNUM_enum
00271 {
00272     ADC_SAMPNUM_ACC1_gc = (0x00<<0),  /* 1 ADC sample */
00273     ADC_SAMPNUM_ACC2_gc = (0x01<<0),  /* Accumulate 2 samples */
```

```
00274     ADC_SAMPNUM_ACC4_gc = (0x02<<0),  /* Accumulate 4 samples */
00275     ADC_SAMPNUM_ACC8_gc = (0x03<<0),  /* Accumulate 8 samples */
00276     ADC_SAMPNUM_ACC16_gc = (0x04<<0),  /* Accumulate 16 samples */
00277     ADC_SAMPNUM_ACC32_gc = (0x05<<0),  /* Accumulate 32 samples */
00278     ADC_SAMPNUM_ACC64_gc = (0x06<<0),  /* Accumulate 64 samples */
00279 } ADC_SAMPNUM_t;
00280
00281 /* Window Comparator Mode select */
00282 typedef enum ADC_WINCM_enum
00283 {
00284     ADC_WINCM_NONE_gc = (0x00<<0),  /* No Window Comparison */
00285     ADC_WINCM_BELOW_gc = (0x01<<0),  /* Below Window */
00286     ADC_WINCM_ABOVE_gc = (0x02<<0),  /* Above Window */
00287     ADC_WINCM_INSIDE_gc = (0x03<<0),  /* Inside Window */
00288     ADC_WINCM_OUTSIDE_gc = (0x04<<0),  /* Outside Window */
00289 } ADC_WINCM_t;
00290
00291 /*
00292 --------------------------------------------------------------------------
00293 BOD - Bod interface
00294 --------------------------------------------------------------------------
00295 */
00296
00297 /* Bod interface */
00298 typedef struct BOD_struct
00299 {
00300     register8_t CTRLA;  /* Control A */
00301     register8_t CTRLB;  /* Control B */
00302     register8_t reserved_0x02;
00303     register8_t reserved_0x03;
00304     register8_t reserved_0x04;
00305     register8_t reserved_0x05;
00306     register8_t reserved_0x06;
00307     register8_t reserved_0x07;
00308     register8_t VLMCTRLA;  /* Voltage level monitor Control */
00309     register8_t INTCTRL;  /* Voltage level monitor interrupt Control */
00310     register8_t INTFLAGS;  /* Voltage level monitor interrupt Flags */
00311     register8_t STATUS;  /* Voltage level monitor status */
00312     register8_t reserved_0x0C;
00313     register8_t reserved_0x0D;
00314     register8_t reserved_0x0E;
00315     register8_t reserved_0x0F;
00316 } BOD_t;
00317
00318 /* Operation in active mode select */
00319 typedef enum BOD_ACTIVE_enum
00320 {
00321     BOD_ACTIVE_DIS_gc = (0x00<<2),  /* Disabled */
00322     BOD_ACTIVE_ENABLED_gc = (0x01<<2),  /* Enabled */
00323     BOD_ACTIVE_SAMPLED_gc = (0x02<<2),  /* Sampled */
00324     BOD_ACTIVE_ENWAKE_gc = (0x03<<2),  /* Enabled with wakeup halt */
00325 } BOD_ACTIVE_t;
00326
00327 /* Bod level select */
00328 typedef enum BOD_LVL_enum
00329 {
00330     BOD_LVL_BODLEVEL0_gc = (0x00<<0),  /* 1.8 V */
00331     BOD_LVL_BODLEVEL1_gc = (0x01<<0),  /* 2.1 V */
00332     BOD_LVL_BODLEVEL2_gc = (0x02<<0),  /* 2.6 V */
00333     BOD_LVL_BODLEVEL3_gc = (0x03<<0),  /* 2.9 V */
00334     BOD_LVL_BODLEVEL4_gc = (0x04<<0),  /* 3.3 V */
00335     BOD_LVL_BODLEVEL5_gc = (0x05<<0),  /* 3.7 V */
00336     BOD_LVL_BODLEVEL6_gc = (0x06<<0),  /* 4.0 V */
00337     BOD_LVL_BODLEVEL7_gc = (0x07<<0),  /* 4.2 V */
00338 } BOD_LVL_t;
00339
00340 /* Sample frequency select */
00341 typedef enum BOD_SAMPFREQ_enum
00342 {
00343     BOD_SAMPFREQ_1KHZ_gc = (0x00<<4),  /* 1kHz sampling */
00344     BOD_SAMPFREQ_125Hz_gc = (0x01<<4),  /* 125Hz sampling */
00345 } BOD_SAMPFREQ_t;
00346
00347 /* Operation in sleep mode select */
00348 typedef enum BOD_SLEEP_enum
00349 {
00350     BOD_SLEEP_DIS_gc = (0x00<<0),  /* Disabled */
00351     BOD_SLEEP_ENABLED_gc = (0x01<<0),  /* Enabled */
00352     BOD_SLEEP_SAMPLED_gc = (0x02<<0),  /* Sampled */
00353 } BOD_SLEEP_t;
00354
00355 /* Configuration select */
00356 typedef enum BOD_VLMCFG_enum
00357 {
00358     BOD_VLMCFG_BELOW_gc = (0x00<<1),  /* Interrupt when supply goes below VLM level */
00359     BOD_VLMCFG_ABOVE_gc = (0x01<<1),  /* Interrupt when supply goes above VLM level */
00360     BOD_VLMCFG_CROSS_gc = (0x02<<1),  /* Interrupt when supply crosses VLM level */
```

```
00361 } BOD_VLMCFG_t;
00362
00363 /* voltage level monitor level select */
00364 typedef enum BOD_VLMLVL_enum
00365 {
00366     BOD_VLMLVL_5ABOVE_gc = (0x00<<0),  /* VLM threshold 5% above BOD level */
00367     BOD_VLMLVL_15ABOVE_gc = (0x01<<0),  /* VLM threshold 15% above BOD level */
00368     BOD_VLMLVL_25ABOVE_gc = (0x02<<0),  /* VLM threshold 25% above BOD level */
00369 } BOD_VLMLVL_t;
00370
00371 /*
00372 --------------------------------------------------------------------------
00373 CCL - Configurable Custom Logic
00374 --------------------------------------------------------------------------
00375 */
00376
00377 /* Configurable Custom Logic */
00378 typedef struct CCL_struct
00379 {
00380     register8_t CTRLA;  /* Control Register A */
00381     register8_t SEQCTRL0;  /* Sequential Control 0 */
00382     register8_t reserved_0x02;
00383     register8_t reserved_0x03;
00384     register8_t reserved_0x04;
00385     register8_t LUT0CTRLA;  /* LUT Control 0 A */
00386     register8_t LUT0CTRLB;  /* LUT Control 0 B */
00387     register8_t LUT0CTRLC;  /* LUT Control 0 C */
00388     register8_t TRUTH0;  /* Truth 0 */
00389     register8_t LUT1CTRLA;  /* LUT Control 1 A */
00390     register8_t LUT1CTRLB;  /* LUT Control 1 B */
00391     register8_t LUT1CTRLC;  /* LUT Control 1 C */
00392     register8_t TRUTH1;  /* Truth 1 */
00393     register8_t reserved_0x0D;
00394     register8_t reserved_0x0E;
00395     register8_t reserved_0x0F;
00396 } CCL_t;
00397
00398 /* Edge Detection Enable select */
00399 typedef enum CCL_EDGEDET_enum
00400 {
00401     CCL_EDGEDET_DIS_gc = (0x00<<7),  /* Edge detector is disabled */
00402     CCL_EDGEDET_EN_gc = (0x01<<7),  /* Edge detector is enabled */
00403 } CCL_EDGEDET_t;
00404
00405 /* Filter Selection select */
00406 typedef enum CCL_FILTSEL_enum
00407 {
00408     CCL_FILTSEL_DISABLE_gc = (0x00<<4),  /* Filter disabled */
00409     CCL_FILTSEL_SYNCH_gc = (0x01<<4),  /* Synchronizer enabled */
00410     CCL_FILTSEL_FILTER_gc = (0x02<<4),  /* Filter enabled */
00411 } CCL_FILTSEL_t;
00412
00413 /* LUT Input 0 Source Selection select */
00414 typedef enum CCL_INSEL0_enum
00415 {
00416     CCL_INSEL0_MASK_gc = (0x00<<0),  /* Masked input */
00417     CCL_INSEL0_FEEDBACK_gc = (0x01<<0),  /* Feedback input source */
00418     CCL_INSEL0_LINK_gc = (0x02<<0),  /* Linked LUT input source */
00419     CCL_INSEL0_EVENT0_gc = (0x03<<0),  /* Event input source 0 */
00420     CCL_INSEL0_EVENT1_gc = (0x04<<0),  /* Event input source 1 */
00421     CCL_INSEL0_IO_gc = (0x05<<0),  /* IO pin LUTn-IN0 input source */
00422     CCL_INSEL0_AC0_gc = (0x06<<0),  /* AC0 OUT input source */
00423     CCL_INSEL0_TCB0_gc = (0x07<<0),  /* TCB0 WO input source */
00424     CCL_INSEL0_TCA0_gc = (0x08<<0),  /* TCA0 WO0 input source */
00425     CCL_INSEL0_TCD0_gc = (0x09<<0),  /* TCD0 WOA input source */
00426     CCL_INSEL0_USART0_gc = (0x0A<<0),  /* USART0 XCK input source */
00427     CCL_INSEL0_SPI0_gc = (0x0B<<0),  /* SPI0 SCK source */
00428 } CCL_INSEL0_t;
00429
00430 /* LUT Input 1 Source Selection select */
00431 typedef enum CCL_INSEL1_enum
00432 {
00433     CCL_INSEL1_MASK_gc = (0x00<<4),  /* Masked input */
00434     CCL_INSEL1_FEEDBACK_gc = (0x01<<4),  /* Feedback input source */
00435     CCL_INSEL1_LINK_gc = (0x02<<4),  /* Linked LUT input source */
00436     CCL_INSEL1_EVENT0_gc = (0x03<<4),  /* Event input source 0 */
00437     CCL_INSEL1_EVENT1_gc = (0x04<<4),  /* Event input source 1 */
00438     CCL_INSEL1_IO_gc = (0x05<<4),  /* IO pin LUTn-N1 input source */
00439     CCL_INSEL1_AC0_gc = (0x06<<4),  /* AC0 OUT input source */
00440     CCL_INSEL1_TCB0_gc = (0x07<<4),  /* TCB0 WO input source */
00441     CCL_INSEL1_TCA0_gc = (0x08<<4),  /* TCA0 WO1 input source */
00442     CCL_INSEL1_TCD0_gc = (0x09<<4),  /* TCD0 WOB input source */
00443     CCL_INSEL1_USART0_gc = (0x0A<<4),  /* USART0 TXD input source */
00444     CCL_INSEL1_SPI0_gc = (0x0B<<4),  /* SPI0 MOSI input source */
00445 } CCL_INSEL1_t;
00446
00447 /* LUT Input 2 Source Selection select */
```

```
00448 typedef enum CCL_INSEL2_enum
00449 {
00450     CCL_INSEL2_MASK_gc = (0x00«0),  /* Masked input */
00451     CCL_INSEL2_FEEDBACK_gc = (0x01«0),  /* Feedback input source */
00452     CCL_INSEL2_LINK_gc = (0x02«0),  /* Linked LUT input source */
00453     CCL_INSEL2_EVENT0_gc = (0x03«0),  /* Event input source 0 */
00454     CCL_INSEL2_EVENT1_gc = (0x04«0),  /* Event input source 1 */
00455     CCL_INSEL2_IO_gc = (0x05«0),  /* IO pin LUTn-IN2 input source */
00456     CCL_INSEL2_AC0_gc = (0x06«0),  /* AC0 OUT input source */
00457     CCL_INSEL2_TCB0_gc = (0x07«0),  /* TCB0 WO input source */
00458     CCL_INSEL2_TCA0_gc = (0x08«0),  /* TCA0 WO2 input source */
00459     CCL_INSEL2_TCD0_gc = (0x09«0),  /* TCD0 WOA input source */
00460     CCL_INSEL2_SPI0_gc = (0x0B«0),  /* SPI0 MISO source */
00461 } CCL_INSEL2_t;
00462
00463 /* Sequential Selection select */
00464 typedef enum CCL_SEQSEL_enum
00465 {
00466     CCL_SEQSEL_DISABLE_gc = (0x00«0),  /* Sequential logic disabled */
00467     CCL_SEQSEL_DFF_gc = (0x01«0),  /* D FlipFlop */
00468     CCL_SEQSEL_JK_gc = (0x02«0),  /* JK FlipFlop */
00469     CCL_SEQSEL_LATCH_gc = (0x03«0),  /* D Latch */
00470     CCL_SEQSEL_RS_gc = (0x04«0),  /* RS Latch */
00471 } CCL_SEQSEL_t;
00472
00473 /*
00474 --------------------------------------------------------------------------
00475 CLKCTRL - Clock controller
00476 --------------------------------------------------------------------------
00477 */
00478
00479 /* Clock controller */
00480 typedef struct CLKCTRL_struct
00481 {
00482     register8_t MCLKCTRLA;  /* MCLK Control A */
00483     register8_t MCLKCTRLB;  /* MCLK Control B */
00484     register8_t MCLKLOCK;  /* MCLK Lock */
00485     register8_t MCLKSTATUS;  /* MCLK Status */
00486     register8_t reserved_0x04;
00487     register8_t reserved_0x05;
00488     register8_t reserved_0x06;
00489     register8_t reserved_0x07;
00490     register8_t reserved_0x08;
00491     register8_t reserved_0x09;
00492     register8_t reserved_0x0A;
00493     register8_t reserved_0x0B;
00494     register8_t reserved_0x0C;
00495     register8_t reserved_0x0D;
00496     register8_t reserved_0x0E;
00497     register8_t reserved_0x0F;
00498     register8_t OSC20MCTRLA;  /* OSC20M Control A */
00499     register8_t OSC20MCALIBA;  /* OSC20M Calibration A */
00500     register8_t OSC20MCALIBB;  /* OSC20M Calibration B */
00501     register8_t reserved_0x13;
00502     register8_t reserved_0x14;
00503     register8_t reserved_0x15;
00504     register8_t reserved_0x16;
00505     register8_t reserved_0x17;
00506     register8_t OSC32KCTRLA;  /* OSC32K Control A */
00507     register8_t reserved_0x19;
00508     register8_t reserved_0x1A;
00509     register8_t reserved_0x1B;
00510     register8_t reserved_0x1C;
00511     register8_t reserved_0x1D;
00512     register8_t reserved_0x1E;
00513     register8_t reserved_0x1F;
00514 } CLKCTRL_t;
00515
00516 /* clock select select */
00517 typedef enum CLKCTRL_CLKSEL_enum
00518 {
00519     CLKCTRL_CLKSEL_OSC20M_gc = (0x00«0),  /* 20MHz internal oscillator */
00520     CLKCTRL_CLKSEL_OSCULP32K_gc = (0x01«0),  /* 32KHz internal Ultra Low Power oscillator */
00521     CLKCTRL_CLKSEL_XOSC32K_gc = (0x02«0),  /* 32.768kHz external crystal oscillator */
00522     CLKCTRL_CLKSEL_EXTCLK_gc = (0x03«0),  /* External clock */
00523 } CLKCTRL_CLKSEL_t;
00524
00525 /* Prescaler division select */
00526 typedef enum CLKCTRL_PDIV_enum
00527 {
00528     CLKCTRL_PDIV_2X_gc = (0x00«1),  /* 2X */
00529     CLKCTRL_PDIV_4X_gc = (0x01«1),  /* 4X */
00530     CLKCTRL_PDIV_8X_gc = (0x02«1),  /* 8X */
00531     CLKCTRL_PDIV_16X_gc = (0x03«1),  /* 16X */
00532     CLKCTRL_PDIV_32X_gc = (0x04«1),  /* 32X */
00533     CLKCTRL_PDIV_64X_gc = (0x05«1),  /* 64X */
00534     CLKCTRL_PDIV_6X_gc = (0x08«1),  /* 6X */
```

```
00535     CLKCTRL_PDIV_10X_gc = (0x09<<1),  /* 10X */
00536     CLKCTRL_PDIV_12X_gc = (0x0A<<1),  /* 12X */
00537     CLKCTRL_PDIV_24X_gc = (0x0B<<1),  /* 24X */
00538     CLKCTRL_PDIV_48X_gc = (0x0C<<1),  /* 48X */
00539 } CLKCTRL_PDIV_t;
00540
00541 /*
00542 --------------------------------------------------------------------------
00543 CPU - CPU
00544 --------------------------------------------------------------------------
00545 */
00546
00547 /* CCP signature select */
00548 typedef enum CCP_enum
00549 {
00550     CCP_SPM_gc = (0x9D<<0),  /* SPM Instruction Protection */
00551     CCP_IOREG_gc = (0xD8<<0),  /* IO Register Protection */
00552 } CCP_t;
00553
00554 /*
00555 --------------------------------------------------------------------------
00556 CPUINT - Interrupt Controller
00557 --------------------------------------------------------------------------
00558 */
00559
00560 /* Interrupt Controller */
00561 typedef struct CPUINT_struct
00562 {
00563     register8_t CTRLA;  /* Control A */
00564     register8_t STATUS;  /* Status */
00565     register8_t LVL0PRI;  /* Interrupt Level 0 Priority */
00566     register8_t LVL1VEC;  /* Interrupt Level 1 Priority Vector */
00567 } CPUINT_t;
00568
00569 /*
00570 --------------------------------------------------------------------------
00571 CRCSCAN - CRCSCAN
00572 --------------------------------------------------------------------------
00573 */
00574
00575 /* CRCSCAN */
00576 typedef struct CRCSCAN_struct
00577 {
00578     register8_t CTRLA;  /* Control A */
00579     register8_t CTRLB;  /* Control B */
00580     register8_t STATUS;  /* Status */
00581     register8_t reserved_0x03;
00582 } CRCSCAN_t;
00583
00584 /* CRC Flash Access Mode select */
00585 typedef enum CRCSCAN_MODE_enum
00586 {
00587     CRCSCAN_MODE_PRIORITY_gc = (0x00<<4),  /* Priority to flash */
00588     CRCSCAN_MODE_RESERVED_gc = (0x01<<4),  /* Reserved */
00589     CRCSCAN_MODE_BACKGROUND_gc = (0x02<<4),  /* Lowest priority to flash */
00590     CRCSCAN_MODE_CONTINUOUS_gc = (0x03<<4),  /* Continuous checks in background */
00591 } CRCSCAN_MODE_t;
00592
00593 /* CRC Source select */
00594 typedef enum CRCSCAN_SRC_enum
00595 {
00596     CRCSCAN_SRC_FLASH_gc = (0x00<<0),  /* CRC on entire flash */
00597     CRCSCAN_SRC_APPLICATION_gc = (0x01<<0),  /* CRC on boot and appl section of flash */
00598     CRCSCAN_SRC_BOOT_gc = (0x02<<0),  /* CRC on boot section of flash */
00599 } CRCSCAN_SRC_t;
00600
00601 /*
00602 --------------------------------------------------------------------------
00603 EVSYS - Event System
00604 --------------------------------------------------------------------------
00605 */
00606
00607 /* Event System */
00608 typedef struct EVSYS_struct
00609 {
00610     register8_t ASYNCSTROBE;  /* Asynchronous Channel Strobe */
00611     register8_t SYNCSTROBE;  /* Synchronous Channel Strobe */
00612     register8_t ASYNCCH0;  /* Asynchronous Channel 0 Generator Selection */
00613     register8_t ASYNCCH1;  /* Asynchronous Channel 1 Generator Selection */
00614     register8_t reserved_0x04;
00615     register8_t reserved_0x05;
00616     register8_t reserved_0x06;
00617     register8_t reserved_0x07;
00618     register8_t reserved_0x08;
00619     register8_t reserved_0x09;
00620     register8_t SYNCCH0;  /* Synchronous Channel 0 Generator Selection */
00621     register8_t reserved_0x0B;
```

```
00622     register8_t reserved_0x0C;
00623     register8_t reserved_0x0D;
00624     register8_t reserved_0x0E;
00625     register8_t reserved_0x0F;
00626     register8_t reserved_0x10;
00627     register8_t reserved_0x11;
00628     register8_t ASYNCUSER0;  /* Asynchronous User Ch 0 Input Selection - TCB0 */
00629     register8_t ASYNCUSER1;  /* Asynchronous User Ch 1 Input Selection - ADC0 */
00630     register8_t ASYNCUSER2;  /* Asynchronous User Ch 2 Input Selection - CCL LUT0 Event 0 */
00631     register8_t ASYNCUSER3;  /* Asynchronous User Ch 3 Input Selection - CCL LUT1 Event 0 */
00632     register8_t ASYNCUSER4;  /* Asynchronous User Ch 4 Input Selection - CCL LUT0 Event 1 */
00633     register8_t ASYNCUSER5;  /* Asynchronous User Ch 5 Input Selection - CCL LUT1 Event 1 */
00634     register8_t ASYNCUSER6;  /* Asynchronous User Ch 6 Input Selection - TCD0 Event 0 */
00635     register8_t ASYNCUSER7;  /* Asynchronous User Ch 7 Input Selection - TCD0 Event 1 */
00636     register8_t ASYNCUSER8;  /* Asynchronous User Ch 8 Input Selection - Event Out 0 */
00637     register8_t ASYNCUSER9;  /* Asynchronous User Ch 9 Input Selection - Event Out 1 */
00638     register8_t ASYNCUSER10;  /* Asynchronous User Ch 10 Input Selection - Event Out 2 */
00639     register8_t reserved_0x1D;
00640     register8_t reserved_0x1E;
00641     register8_t reserved_0x1F;
00642     register8_t reserved_0x20;
00643     register8_t reserved_0x21;
00644     register8_t SYNCUSER0;  /* Synchronous User Ch 0 Input Selection - TCA0 */
00645     register8_t SYNCUSER1;  /* Synchronous User Ch 1 Input Selection - USART0 */
00646     register8_t reserved_0x24;
00647     register8_t reserved_0x25;
00648     register8_t reserved_0x26;
00649     register8_t reserved_0x27;
00650     register8_t reserved_0x28;
00651     register8_t reserved_0x29;
00652     register8_t reserved_0x2A;
00653     register8_t reserved_0x2B;
00654     register8_t reserved_0x2C;
00655     register8_t reserved_0x2D;
00656     register8_t reserved_0x2E;
00657     register8_t reserved_0x2F;
00658     register8_t reserved_0x30;
00659     register8_t reserved_0x31;
00660     register8_t reserved_0x32;
00661     register8_t reserved_0x33;
00662     register8_t reserved_0x34;
00663     register8_t reserved_0x35;
00664     register8_t reserved_0x36;
00665     register8_t reserved_0x37;
00666     register8_t reserved_0x38;
00667     register8_t reserved_0x39;
00668     register8_t reserved_0x3A;
00669     register8_t reserved_0x3B;
00670     register8_t reserved_0x3C;
00671     register8_t reserved_0x3D;
00672     register8_t reserved_0x3E;
00673     register8_t reserved_0x3F;
00674 } EVSYS_t;
00675
00676 /* Asynchronous Channel 0 Generator Selection select */
00677 typedef enum EVSYS_ASYNCCH0_enum
00678 {
00679     EVSYS_ASYNCCH0_OFF_gc = (0x00<<0),  /* Off */
00680     EVSYS_ASYNCCH0_CCL_LUT0_gc = (0x01<<0),  /* Configurable Custom Logic LUT0 */
00681     EVSYS_ASYNCCH0_CCL_LUT1_gc = (0x02<<0),  /* Configurable Custom Logic LUT1 */
00682     EVSYS_ASYNCCH0_AC0_OUT_gc = (0x03<<0),  /* Analog Comparator 0 out */
00683     EVSYS_ASYNCCH0_TCD0_CMPBCLR_gc = (0x04<<0),  /* Timer/Counter D0 compare B clear */
00684     EVSYS_ASYNCCH0_TCD0_CMPASET_gc = (0x05<<0),  /* Timer/Counter D0 compare A set */
00685     EVSYS_ASYNCCH0_TCD0_CMPBSET_gc = (0x06<<0),  /* Timer/Counter D0 compare B set */
00686     EVSYS_ASYNCCH0_TCD0_PROGEV_gc = (0x07<<0),  /* Timer/Counter D0 program event */
00687     EVSYS_ASYNCCH0_RTC_OVF_gc = (0x08<<0),  /* Real Time Counter overflow */
00688     EVSYS_ASYNCCH0_RTC_CMP_gc = (0x09<<0),  /* Real Time Counter compare */
00689     EVSYS_ASYNCCH0_PORTA_PIN0_gc = (0x0A<<0),  /* Asynchronous Event from Pin PA0 */
00690     EVSYS_ASYNCCH0_PORTA_PIN1_gc = (0x0B<<0),  /* Asynchronous Event from Pin PA1 */
00691     EVSYS_ASYNCCH0_PORTA_PIN2_gc = (0x0C<<0),  /* Asynchronous Event from Pin PA2 */
00692     EVSYS_ASYNCCH0_PORTA_PIN3_gc = (0x0D<<0),  /* Asynchronous Event from Pin PA3 */
00693     EVSYS_ASYNCCH0_PORTA_PIN4_gc = (0x0E<<0),  /* Asynchronous Event from Pin PA4 */
00694     EVSYS_ASYNCCH0_PORTA_PIN5_gc = (0x0F<<0),  /* Asynchronous Event from Pin PA5 */
00695     EVSYS_ASYNCCH0_PORTA_PIN6_gc = (0x10<<0),  /* Asynchronous Event from Pin PA6 */
00696     EVSYS_ASYNCCH0_PORTA_PIN7_gc = (0x11<<0),  /* Asynchronous Event from Pin PA7 */
00697     EVSYS_ASYNCCH0_UPDI_gc = (0x12<<0),  /* Unified Program and debug interface */
00698 } EVSYS_ASYNCCH0_t;
00699
00700 /* Asynchronous Channel 1 Generator Selection select */
00701 typedef enum EVSYS_ASYNCCH1_enum
00702 {
00703     EVSYS_ASYNCCH1_OFF_gc = (0x00<<0),  /* Off */
00704     EVSYS_ASYNCCH1_CCL_LUT0_gc = (0x01<<0),  /* Configurable custom logic LUT0 */
00705     EVSYS_ASYNCCH1_CCL_LUT1_gc = (0x02<<0),  /* Configurable custom logic LUT1 */
00706     EVSYS_ASYNCCH1_AC0_OUT_gc = (0x03<<0),  /* Analog Comparator 0 out */
00707     EVSYS_ASYNCCH1_TCD0_CMPBCLR_gc = (0x04<<0),  /* Timer/Counter D0 compare B clear */
00708     EVSYS_ASYNCCH1_TCD0_CMPASET_gc = (0x05<<0),  /* Timer/Counter D0 compare A set */
```

```
00709     EVSYS_ASYNCCH1_TCD0_CMPBSET_gc = (0x06<<0),  /* Timer/Counter D0 compare B set */
00710     EVSYS_ASYNCCH1_TCD0_PROGEV_gc = (0x07<<0),  /* Timer/Counter D0 program event */
00711     EVSYS_ASYNCCH1_RTC_OVF_gc = (0x08<<0),  /* Real Time Counter overflow */
00712     EVSYS_ASYNCCH1_RTC_CMP_gc = (0x09<<0),  /* Real Time Counter compare */
00713     EVSYS_ASYNCCH1_PORTB_PIN0_gc = (0x0A<<0),  /* Asynchronous Event from Pin PB0 */
00714     EVSYS_ASYNCCH1_PORTB_PIN1_gc = (0x0B<<0),  /* Asynchronous Event from Pin PB1 */
00715     EVSYS_ASYNCCH1_PORTB_PIN2_gc = (0x0C<<0),  /* Asynchronous Event from Pin PB2 */
00716     EVSYS_ASYNCCH1_PORTB_PIN3_gc = (0x0D<<0),  /* Asynchronous Event from Pin PB3 */
00717     EVSYS_ASYNCCH1_PORTB_PIN4_gc = (0x0E<<0),  /* Asynchronous Event from Pin PB4 */
00718     EVSYS_ASYNCCH1_PORTB_PIN5_gc = (0x0F<<0),  /* Asynchronous Event from Pin PB5 */
00719     EVSYS_ASYNCCH1_PORTB_PIN6_gc = (0x10<<0),  /* Asynchronous Event from Pin PB6 */
00720     EVSYS_ASYNCCH1_PORTB_PIN7_gc = (0x11<<0),  /* Asynchronous Event from Pin PB7 */
00721 } EVSYS_ASYNCCH1_t;
00722
00723 /* Asynchronous User Ch 0 Input Selection - TCB0 select */
00724 typedef enum EVSYS_ASYNCUSER0_enum
00725 {
00726     EVSYS_ASYNCUSER0_OFF_gc = (0x00<<0),  /* Off */
00727     EVSYS_ASYNCUSER0_SYNCCH0_gc = (0x01<<0),  /* Synchronous Event Channel 0 */
00728     EVSYS_ASYNCUSER0_SYNCCH1_gc = (0x02<<0),  /* Synchronous Event Channel 1 */
00729     EVSYS_ASYNCUSER0_ASYNCCH0_gc = (0x03<<0),  /* Asynchronous Event Channel 0 */
00730     EVSYS_ASYNCUSER0_ASYNCCH1_gc = (0x04<<0),  /* Asynchronous Event Channel 1 */
00731     EVSYS_ASYNCUSER0_ASYNCCH2_gc = (0x05<<0),  /* Asynchronous Event Channel 2 */
00732     EVSYS_ASYNCUSER0_ASYNCCH3_gc = (0x06<<0),  /* Asynchronous Event Channel 3 */
00733 } EVSYS_ASYNCUSER0_t;
00734
00735 /* Asynchronous User Ch 1 Input Selection - ADC0 select */
00736 typedef enum EVSYS_ASYNCUSER1_enum
00737 {
00738     EVSYS_ASYNCUSER1_OFF_gc = (0x00<<0),  /* Off */
00739     EVSYS_ASYNCUSER1_SYNCCH0_gc = (0x01<<0),  /* Synchronous Event Channel 0 */
00740     EVSYS_ASYNCUSER1_SYNCCH1_gc = (0x02<<0),  /* Synchronous Event Channel 1 */
00741     EVSYS_ASYNCUSER1_ASYNCCH0_gc = (0x03<<0),  /* Asynchronous Event Channel 0 */
00742     EVSYS_ASYNCUSER1_ASYNCCH1_gc = (0x04<<0),  /* Asynchronous Event Channel 1 */
00743     EVSYS_ASYNCUSER1_ASYNCCH2_gc = (0x05<<0),  /* Asynchronous Event Channel 2 */
00744     EVSYS_ASYNCUSER1_ASYNCCH3_gc = (0x06<<0),  /* Asynchronous Event Channel 3 */
00745 } EVSYS_ASYNCUSER1_t;
00746
00747 /* Asynchronous User Ch 10 Input Selection - Event Out 2 select */
00748 typedef enum EVSYS_ASYNCUSER10_enum
00749 {
00750     EVSYS_ASYNCUSER10_OFF_gc = (0x00<<0),  /* Off */
00751     EVSYS_ASYNCUSER10_SYNCCH0_gc = (0x01<<0),  /* Synchronous Event Channel 0 */
00752     EVSYS_ASYNCUSER10_SYNCCH1_gc = (0x02<<0),  /* Synchronous Event Channel 1 */
00753     EVSYS_ASYNCUSER10_ASYNCCH0_gc = (0x03<<0),  /* Asynchronous Event Channel 0 */
00754     EVSYS_ASYNCUSER10_ASYNCCH1_gc = (0x04<<0),  /* Asynchronous Event Channel 1 */
00755     EVSYS_ASYNCUSER10_ASYNCCH2_gc = (0x05<<0),  /* Asynchronous Event Channel 2 */
00756     EVSYS_ASYNCUSER10_ASYNCCH3_gc = (0x06<<0),  /* Asynchronous Event Channel 3 */
00757 } EVSYS_ASYNCUSER10_t;
00758
00759 /* Asynchronous User Ch 2 Input Selection - CCL LUT0 Event 0 select */
00760 typedef enum EVSYS_ASYNCUSER2_enum
00761 {
00762     EVSYS_ASYNCUSER2_OFF_gc = (0x00<<0),  /* Off */
00763     EVSYS_ASYNCUSER2_SYNCCH0_gc = (0x01<<0),  /* Synchronous Event Channel 0 */
00764     EVSYS_ASYNCUSER2_SYNCCH1_gc = (0x02<<0),  /* Synchronous Event Channel 1 */
00765     EVSYS_ASYNCUSER2_ASYNCCH0_gc = (0x03<<0),  /* Asynchronous Event Channel 0 */
00766     EVSYS_ASYNCUSER2_ASYNCCH1_gc = (0x04<<0),  /* Asynchronous Event Channel 1 */
00767     EVSYS_ASYNCUSER2_ASYNCCH2_gc = (0x05<<0),  /* Asynchronous Event Channel 2 */
00768     EVSYS_ASYNCUSER2_ASYNCCH3_gc = (0x06<<0),  /* Asynchronous Event Channel 3 */
00769 } EVSYS_ASYNCUSER2_t;
00770
00771 /* Asynchronous User Ch 3 Input Selection - CCL LUT1 Event 0 select */
00772 typedef enum EVSYS_ASYNCUSER3_enum
00773 {
00774     EVSYS_ASYNCUSER3_OFF_gc = (0x00<<0),  /* Off */
00775     EVSYS_ASYNCUSER3_SYNCCH0_gc = (0x01<<0),  /* Synchronous Event Channel 0 */
00776     EVSYS_ASYNCUSER3_SYNCCH1_gc = (0x02<<0),  /* Synchronous Event Channel 1 */
00777     EVSYS_ASYNCUSER3_ASYNCCH0_gc = (0x03<<0),  /* Asynchronous Event Channel 0 */
00778     EVSYS_ASYNCUSER3_ASYNCCH1_gc = (0x04<<0),  /* Asynchronous Event Channel 1 */
00779     EVSYS_ASYNCUSER3_ASYNCCH2_gc = (0x05<<0),  /* Asynchronous Event Channel 2 */
00780     EVSYS_ASYNCUSER3_ASYNCCH3_gc = (0x06<<0),  /* Asynchronous Event Channel 3 */
00781 } EVSYS_ASYNCUSER3_t;
00782
00783 /* Asynchronous User Ch 4 Input Selection - CCL LUT0 Event 1 select */
00784 typedef enum EVSYS_ASYNCUSER4_enum
00785 {
00786     EVSYS_ASYNCUSER4_OFF_gc = (0x00<<0),  /* Off */
00787     EVSYS_ASYNCUSER4_SYNCCH0_gc = (0x01<<0),  /* Synchronous Event Channel 0 */
00788     EVSYS_ASYNCUSER4_SYNCCH1_gc = (0x02<<0),  /* Synchronous Event Channel 1 */
00789     EVSYS_ASYNCUSER4_ASYNCCH0_gc = (0x03<<0),  /* Asynchronous Event Channel 0 */
00790     EVSYS_ASYNCUSER4_ASYNCCH1_gc = (0x04<<0),  /* Asynchronous Event Channel 1 */
00791     EVSYS_ASYNCUSER4_ASYNCCH2_gc = (0x05<<0),  /* Asynchronous Event Channel 2 */
00792     EVSYS_ASYNCUSER4_ASYNCCH3_gc = (0x06<<0),  /* Asynchronous Event Channel 3 */
00793 } EVSYS_ASYNCUSER4_t;
00794
00795 /* Asynchronous User Ch 5 Input Selection - CCL LUT1 Event 1 select */
```

```
00796 typedef enum EVSYS_ASYNCUSER5_enum
00797 {
00798     EVSYS_ASYNCUSER5_OFF_gc = (0x00<<0),  /* Off */
00799     EVSYS_ASYNCUSER5_SYNCCH0_gc = (0x01<<0),  /* Synchronous Event Channel 0 */
00800     EVSYS_ASYNCUSER5_SYNCCH1_gc = (0x02<<0),  /* Synchronous Event Channel 1 */
00801     EVSYS_ASYNCUSER5_ASYNCCH0_gc = (0x03<<0),  /* Asynchronous Event Channel 0 */
00802     EVSYS_ASYNCUSER5_ASYNCCH1_gc = (0x04<<0),  /* Asynchronous Event Channel 1 */
00803     EVSYS_ASYNCUSER5_ASYNCCH2_gc = (0x05<<0),  /* Asynchronous Event Channel 2 */
00804     EVSYS_ASYNCUSER5_ASYNCCH3_gc = (0x06<<0),  /* Asynchronous Event Channel 3 */
00805 } EVSYS_ASYNCUSER5_t;
00806
00807 /* Asynchronous User Ch 6 Input Selection - TCD0 Event 0 select */
00808 typedef enum EVSYS_ASYNCUSER6_enum
00809 {
00810     EVSYS_ASYNCUSER6_OFF_gc = (0x00<<0),  /* Off */
00811     EVSYS_ASYNCUSER6_SYNCCH0_gc = (0x01<<0),  /* Synchronous Event Channel 0 */
00812     EVSYS_ASYNCUSER6_SYNCCH1_gc = (0x02<<0),  /* Synchronous Event Channel 1 */
00813     EVSYS_ASYNCUSER6_ASYNCCH0_gc = (0x03<<0),  /* Asynchronous Event Channel 0 */
00814     EVSYS_ASYNCUSER6_ASYNCCH1_gc = (0x04<<0),  /* Asynchronous Event Channel 1 */
00815     EVSYS_ASYNCUSER6_ASYNCCH2_gc = (0x05<<0),  /* Asynchronous Event Channel 2 */
00816     EVSYS_ASYNCUSER6_ASYNCCH3_gc = (0x06<<0),  /* Asynchronous Event Channel 3 */
00817 } EVSYS_ASYNCUSER6_t;
00818
00819 /* Asynchronous User Ch 7 Input Selection - TCD0 Event 1 select */
00820 typedef enum EVSYS_ASYNCUSER7_enum
00821 {
00822     EVSYS_ASYNCUSER7_OFF_gc = (0x00<<0),  /* Off */
00823     EVSYS_ASYNCUSER7_SYNCCH0_gc = (0x01<<0),  /* Synchronous Event Channel 0 */
00824     EVSYS_ASYNCUSER7_SYNCCH1_gc = (0x02<<0),  /* Synchronous Event Channel 1 */
00825     EVSYS_ASYNCUSER7_ASYNCCH0_gc = (0x03<<0),  /* Asynchronous Event Channel 0 */
00826     EVSYS_ASYNCUSER7_ASYNCCH1_gc = (0x04<<0),  /* Asynchronous Event Channel 1 */
00827     EVSYS_ASYNCUSER7_ASYNCCH2_gc = (0x05<<0),  /* Asynchronous Event Channel 2 */
00828     EVSYS_ASYNCUSER7_ASYNCCH3_gc = (0x06<<0),  /* Asynchronous Event Channel 3 */
00829 } EVSYS_ASYNCUSER7_t;
00830
00831 /* Asynchronous User Ch 8 Input Selection - Event Out 0 select */
00832 typedef enum EVSYS_ASYNCUSER8_enum
00833 {
00834     EVSYS_ASYNCUSER8_OFF_gc = (0x00<<0),  /* Off */
00835     EVSYS_ASYNCUSER8_SYNCCH0_gc = (0x01<<0),  /* Synchronous Event Channel 0 */
00836     EVSYS_ASYNCUSER8_SYNCCH1_gc = (0x02<<0),  /* Synchronous Event Channel 1 */
00837     EVSYS_ASYNCUSER8_ASYNCCH0_gc = (0x03<<0),  /* Asynchronous Event Channel 0 */
00838     EVSYS_ASYNCUSER8_ASYNCCH1_gc = (0x04<<0),  /* Asynchronous Event Channel 1 */
00839     EVSYS_ASYNCUSER8_ASYNCCH2_gc = (0x05<<0),  /* Asynchronous Event Channel 2 */
00840     EVSYS_ASYNCUSER8_ASYNCCH3_gc = (0x06<<0),  /* Asynchronous Event Channel 3 */
00841 } EVSYS_ASYNCUSER8_t;
00842
00843 /* Asynchronous User Ch 9 Input Selection - Event Out 1 select */
00844 typedef enum EVSYS_ASYNCUSER9_enum
00845 {
00846     EVSYS_ASYNCUSER9_OFF_gc = (0x00<<0),  /* Off */
00847     EVSYS_ASYNCUSER9_SYNCCH0_gc = (0x01<<0),  /* Synchronous Event Channel 0 */
00848     EVSYS_ASYNCUSER9_SYNCCH1_gc = (0x02<<0),  /* Synchronous Event Channel 1 */
00849     EVSYS_ASYNCUSER9_ASYNCCH0_gc = (0x03<<0),  /* Asynchronous Event Channel 0 */
00850     EVSYS_ASYNCUSER9_ASYNCCH1_gc = (0x04<<0),  /* Asynchronous Event Channel 1 */
00851     EVSYS_ASYNCUSER9_ASYNCCH2_gc = (0x05<<0),  /* Asynchronous Event Channel 2 */
00852     EVSYS_ASYNCUSER9_ASYNCCH3_gc = (0x06<<0),  /* Asynchronous Event Channel 3 */
00853 } EVSYS_ASYNCUSER9_t;
00854
00855 /* Synchronous Channel 0 Generator Selection select */
00856 typedef enum EVSYS_SYNCCH0_enum
00857 {
00858     EVSYS_SYNCCH0_OFF_gc = (0x00<<0),  /* Off */
00859     EVSYS_SYNCCH0_TCB0_gc = (0x01<<0),  /* Timer/Counter B0 */
00860     EVSYS_SYNCCH0_TCA0_OVF_LUNF_gc = (0x02<<0),  /* Timer/Counter A0 overflow */
00861     EVSYS_SYNCCH0_TCA0_HUNF_gc = (0x03<<0),  /* Timer/Counter A0 underflow high byte (split mode) */
00862     EVSYS_SYNCCH0_TCA0_CMP0_gc = (0x04<<0),  /* Timer/Counter A0 compare 0 */
00863     EVSYS_SYNCCH0_TCA0_CMP1_gc = (0x05<<0),  /* Timer/Counter A0 compare 1 */
00864     EVSYS_SYNCCH0_TCA0_CMP2_gc = (0x06<<0),  /* Timer/Counter A0 compare 2 */
00865     EVSYS_SYNCCH0_PORTC_PIN0_gc = (0x07<<0),  /* Synchronous Event from Pin PC0 */
00866     EVSYS_SYNCCH0_PORTC_PIN1_gc = (0x08<<0),  /* Synchronous Event from Pin PC1 */
00867     EVSYS_SYNCCH0_PORTC_PIN2_gc = (0x09<<0),  /* Synchronous Event from Pin PC2 */
00868     EVSYS_SYNCCH0_PORTC_PIN3_gc = (0x0A<<0),  /* Synchronous Event from Pin PC3 */
00869     EVSYS_SYNCCH0_PORTC_PIN4_gc = (0x0B<<0),  /* Synchronous Event from Pin PC4 */
00870     EVSYS_SYNCCH0_PORTC_PIN5_gc = (0x0C<<0),  /* Synchronous Event from Pin PC5 */
00871     EVSYS_SYNCCH0_PORTA_PIN0_gc = (0x0D<<0),  /* Synchronous Event from Pin PA0 */
00872     EVSYS_SYNCCH0_PORTA_PIN1_gc = (0x0E<<0),  /* Synchronous Event from Pin PA1 */
00873     EVSYS_SYNCCH0_PORTA_PIN2_gc = (0x0F<<0),  /* Synchronous Event from Pin PA2 */
00874     EVSYS_SYNCCH0_PORTA_PIN3_gc = (0x10<<0),  /* Synchronous Event from Pin PA3 */
00875     EVSYS_SYNCCH0_PORTA_PIN4_gc = (0x11<<0),  /* Synchronous Event from Pin PA4 */
00876     EVSYS_SYNCCH0_PORTA_PIN5_gc = (0x12<<0),  /* Synchronous Event from Pin PA5 */
00877     EVSYS_SYNCCH0_PORTA_PIN6_gc = (0x13<<0),  /* Synchronous Event from Pin PA6 */
00878     EVSYS_SYNCCH0_PORTA_PIN7_gc = (0x14<<0),  /* Synchronous Event from Pin PA7 */
00879 } EVSYS_SYNCCH0_t;
00880
00881 /* Synchronous User Ch 0 Input Selection - TCA0 select */
00882 typedef enum EVSYS_SYNCUSER0_enum
```

```
00883 {
00884     EVSYS_SYNCUSER0_OFF_gc = (0x00<<0),  /* Off */
00885     EVSYS_SYNCUSER0_SYNCCH0_gc = (0x01<<0),  /* Synchronous Event Channel 0 */
00886     EVSYS_SYNCUSER0_SYNCCH1_gc = (0x02<<0),  /* Synchronous Event Channel 1 */
00887 } EVSYS_SYNCUSER0_t;
00888
00889 /* Synchronous User Ch 1 Input Selection - USART0 select */
00890 typedef enum EVSYS_SYNCUSER1_enum
00891 {
00892     EVSYS_SYNCUSER1_OFF_gc = (0x00<<0),  /* Off */
00893     EVSYS_SYNCUSER1_SYNCCH0_gc = (0x01<<0),  /* Synchronous Event Channel 0 */
00894     EVSYS_SYNCUSER1_SYNCCH1_gc = (0x02<<0),  /* Synchronous Event Channel 1 */
00895 } EVSYS_SYNCUSER1_t;
00896
00897 /*
00898 --------------------------------------------------------------------------
00899 FUSE - Fuses
00900 --------------------------------------------------------------------------
00901 */
00902
00903 /* Fuses */
00904 typedef struct FUSE_struct
00905 {
00906     register8_t WDTCFG;  /* Watchdog Configuration */
00907     register8_t BODCFG;  /* BOD Configuration */
00908     register8_t OSCCFG;  /* Oscillator Configuration */
00909     register8_t reserved_0x03;
00910     register8_t TCD0CFG;  /* TCD0 Configuration */
00911     register8_t SYSCFG0;  /* System Configuration 0 */
00912     register8_t SYSCFG1;  /* System Configuration 1 */
00913     register8_t APPEND;  /* Application Code Section End */
00914     register8_t BOOTEND;  /* Boot Section End */
00915 } FUSE_t;
00916
00917 /* avr-libc typedef for avr/fuse.h */
00918 typedef FUSE_t NVM_FUSES_t;
00919
00920 /* BOD Operation in Active Mode select */
00921 typedef enum ACTIVE_enum
00922 {
00923     ACTIVE_DIS_gc = (0x00<<2),  /* Disabled */
00924     ACTIVE_ENABLED_gc = (0x01<<2),  /* Enabled */
00925     ACTIVE_SAMPLED_gc = (0x02<<2),  /* Sampled */
00926     ACTIVE_ENWAKE_gc = (0x03<<2),  /* Enabled with wake-up halted until BOD is ready */
00927 } ACTIVE_t;
00928
00929 /* CRC Source select */
00930 typedef enum CRCSRC_enum
00931 {
00932     CRCSRC_FLASH_gc = (0x00<<6),  /* The CRC is performed on the entire Flash (boot, application code
    and application data section). */
00933     CRCSRC_BOOT_gc = (0x01<<6),  /* The CRC is performed on the boot section of Flash */
00934     CRCSRC_BOOTAPP_gc = (0x02<<6),  /* The CRC is performed on the boot and application code section of
    Flash */
00935     CRCSRC_NOCRC_gc = (0x03<<6),  /* Disable CRC. */
00936 } CRCSRC_t;
00937
00938 /* Frequency Select select */
00939 typedef enum FREQSEL_enum
00940 {
00941     FREQSEL_16MHZ_gc = (0x01<<0),  /* 16 MHz */
00942     FREQSEL_20MHZ_gc = (0x02<<0),  /* 20 MHz */
00943 } FREQSEL_t;
00944
00945 /* BOD Level select */
00946 typedef enum LVL_enum
00947 {
00948     LVL_BODLEVEL0_gc = (0x00<<5),  /* 1.8 V */
00949     LVL_BODLEVEL1_gc = (0x01<<5),  /* 2.1 V */
00950     LVL_BODLEVEL2_gc = (0x02<<5),  /* 2.6 V */
00951     LVL_BODLEVEL3_gc = (0x03<<5),  /* 2.9 V */
00952     LVL_BODLEVEL4_gc = (0x04<<5),  /* 3.3 V */
00953     LVL_BODLEVEL5_gc = (0x05<<5),  /* 3.7 V */
00954     LVL_BODLEVEL6_gc = (0x06<<5),  /* 4.0 V */
00955     LVL_BODLEVEL7_gc = (0x07<<5),  /* 4.2 V */
00956 } LVL_t;
00957
00958 /* Watchdog Timeout Period select */
00959 typedef enum PERIOD_enum
00960 {
00961     PERIOD_OFF_gc = (0x00<<0),  /* Watch-Dog timer Off */
00962     PERIOD_8CLK_gc = (0x01<<0),  /* 8 cycles (8ms) */
00963     PERIOD_16CLK_gc = (0x02<<0),  /* 16 cycles (16ms) */
00964     PERIOD_32CLK_gc = (0x03<<0),  /* 32 cycles (32ms) */
00965     PERIOD_64CLK_gc = (0x04<<0),  /* 64 cycles (64ms) */
00966     PERIOD_128CLK_gc = (0x05<<0),  /* 128 cycles (0.128s) */
00967     PERIOD_256CLK_gc = (0x06<<0),  /* 256 cycles (0.256s) */
```

```
00968     PERIOD_512CLK_gc = (0x07<<0),  /* 512 cycles (0.512s) */
00969     PERIOD_1KCLK_gc = (0x08<<0),  /* 1K cycles (1.0s) */
00970     PERIOD_2KCLK_gc = (0x09<<0),  /* 2K cycles (2.0s) */
00971     PERIOD_4KCLK_gc = (0x0A<<0),  /* 4K cycles (4.1s) */
00972     PERIOD_8KCLK_gc = (0x0B<<0),  /* 8K cycles (8.2s) */
00973 } PERIOD_t;
00974
00975 /* Reset Pin Configuration select */
00976 typedef enum RSTPINCFG_enum
00977 {
00978     RSTPINCFG_GPIO_gc = (0x00<<2),  /* GPIO mode */
00979     RSTPINCFG_UPDI_gc = (0x01<<2),  /* UPDI mode */
00980     RSTPINCFG_RST_gc = (0x02<<2),  /* Reset mode */
00981 } RSTPINCFG_t;
00982
00983 /* BOD Sample Frequency select */
00984 typedef enum SAMPFREQ_enum
00985 {
00986     SAMPFREQ_1KHz_gc = (0x00<<4),  /* 1kHz sampling frequency */
00987     SAMPFREQ_125Hz_gc = (0x01<<4),  /* 125Hz sampling frequency */
00988 } SAMPFREQ_t;
00989
00990 /* BOD Operation in Sleep Mode select */
00991 typedef enum SLEEP_enum
00992 {
00993     SLEEP_DIS_gc = (0x00<<0),  /* Disabled */
00994     SLEEP_ENABLED_gc = (0x01<<0),  /* Enabled */
00995     SLEEP_SAMPLED_gc = (0x02<<0),  /* Sampled */
00996 } SLEEP_t;
00997
00998 /* Startup Time select */
00999 typedef enum SUT_enum
01000 {
01001     SUT_0MS_gc = (0x00<<0),  /* 0 ms */
01002     SUT_1MS_gc = (0x01<<0),  /* 1 ms */
01003     SUT_2MS_gc = (0x02<<0),  /* 2 ms */
01004     SUT_4MS_gc = (0x03<<0),  /* 4 ms */
01005     SUT_8MS_gc = (0x04<<0),  /* 8 ms */
01006     SUT_16MS_gc = (0x05<<0),  /* 16 ms */
01007     SUT_32MS_gc = (0x06<<0),  /* 32 ms */
01008     SUT_64MS_gc = (0x07<<0),  /* 64 ms */
01009 } SUT_t;
01010
01011 /* Watchdog Window Timeout Period select */
01012 typedef enum WINDOW_enum
01013 {
01014     WINDOW_OFF_gc = (0x00<<4),  /* Window mode off */
01015     WINDOW_8CLK_gc = (0x01<<4),  /* 8 cycles (8ms) */
01016     WINDOW_16CLK_gc = (0x02<<4),  /* 16 cycles (16ms) */
01017     WINDOW_32CLK_gc = (0x03<<4),  /* 32 cycles (32ms) */
01018     WINDOW_64CLK_gc = (0x04<<4),  /* 64 cycles (64ms) */
01019     WINDOW_128CLK_gc = (0x05<<4),  /* 128 cycles (0.128s) */
01020     WINDOW_256CLK_gc = (0x06<<4),  /* 256 cycles (0.256s) */
01021     WINDOW_512CLK_gc = (0x07<<4),  /* 512 cycles (0.512s) */
01022     WINDOW_1KCLK_gc = (0x08<<4),  /* 1K cycles (1.0s) */
01023     WINDOW_2KCLK_gc = (0x09<<4),  /* 2K cycles (2.0s) */
01024     WINDOW_4KCLK_gc = (0x0A<<4),  /* 4K cycles (4.1s) */
01025     WINDOW_8KCLK_gc = (0x0B<<4),  /* 8K cycles (8.2s) */
01026 } WINDOW_t;
01027
01028 /*
01029 --------------------------------------------------------------------------
01030 LOCKBIT - Lockbit
01031 --------------------------------------------------------------------------
01032 */
01033
01034 /* Lockbit */
01035 typedef struct LOCKBIT_struct
01036 {
01037     register8_t LOCKBIT;  /* Lock bits */
01038 } LOCKBIT_t;
01039
01040 /* Lock Bits select */
01041 typedef enum LB_enum
01042 {
01043     LB_RWLOCK_gc = (0x3A<<0),  /* Read and write lock */
01044     LB_NOLOCK_gc = (0xC5<<0),  /* No locks */
01045 } LB_t;
01046
01047 /*
01048 --------------------------------------------------------------------------
01049 NVMCTRL - Non-volatile Memory Controller
01050 --------------------------------------------------------------------------
01051 */
01052
01053 /* Non-volatile Memory Controller */
01054 typedef struct NVMCTRL_struct
```

```
01055 {
01056     register8_t CTRLA;  /* Control A */
01057     register8_t CTRLB;  /* Control B */
01058     register8_t STATUS;  /* Status */
01059     register8_t INTCTRL;  /* Interrupt Control */
01060     register8_t INTFLAGS;  /* Interrupt Flags */
01061     register8_t reserved_0x05;
01062     _WORDREGISTER(DATA);  /* Data */
01063     _WORDREGISTER(ADDR);  /* Address */
01064     register8_t reserved_0x0A;
01065     register8_t reserved_0x0B;
01066     register8_t reserved_0x0C;
01067     register8_t reserved_0x0D;
01068     register8_t reserved_0x0E;
01069     register8_t reserved_0x0F;
01070 } NVMCTRL_t;
01071
01072 /* Command select */
01073 typedef enum NVMCTRL_CMD_enum
01074 {
01075     NVMCTRL_CMD_NONE_gc = (0x00«0),  /* No Command */
01076     NVMCTRL_CMD_PAGEWRITE_gc = (0x01«0),  /* Write page */
01077     NVMCTRL_CMD_PAGEERASE_gc = (0x02«0),  /* Erase page */
01078     NVMCTRL_CMD_PAGEERASEWRITE_gc = (0x03«0),  /* Erase and write page */
01079     NVMCTRL_CMD_PAGEBUFCLR_gc = (0x04«0),  /* Page buffer clear */
01080     NVMCTRL_CMD_CHIPERASE_gc = (0x05«0),  /* Chip erase */
01081     NVMCTRL_CMD_EEERASE_gc = (0x06«0),  /* EEPROM erase */
01082     NVMCTRL_CMD_FUSEWRITE_gc = (0x07«0),  /* Write fuse (PDI only) */
01083 } NVMCTRL_CMD_t;
01084
01085 /*
01086 --------------------------------------------------------------------------
01087 PORT - I/O Ports
01088 --------------------------------------------------------------------------
01089 */
01090
01091 /* I/O Ports */
01092 typedef struct PORT_struct
01093 {
01094     register8_t DIR;  /* Data Direction */
01095     register8_t DIRSET;  /* Data Direction Set */
01096     register8_t DIRCLR;  /* Data Direction Clear */
01097     register8_t DIRTGL;  /* Data Direction Toggle */
01098     register8_t OUT;  /* Output Value */
01099     register8_t OUTSET;  /* Output Value Set */
01100     register8_t OUTCLR;  /* Output Value Clear */
01101     register8_t OUTTGL;  /* Output Value Toggle */
01102     register8_t IN;  /* Input Value */
01103     register8_t INTFLAGS;  /* Interrupt Flags */
01104     register8_t reserved_0x0A;
01105     register8_t reserved_0x0B;
01106     register8_t reserved_0x0C;
01107     register8_t reserved_0x0D;
01108     register8_t reserved_0x0E;
01109     register8_t reserved_0x0F;
01110     register8_t PIN0CTRL;  /* Pin 0 Control */
01111     register8_t PIN1CTRL;  /* Pin 1 Control */
01112     register8_t PIN2CTRL;  /* Pin 2 Control */
01113     register8_t PIN3CTRL;  /* Pin 3 Control */
01114     register8_t PIN4CTRL;  /* Pin 4 Control */
01115     register8_t PIN5CTRL;  /* Pin 5 Control */
01116     register8_t PIN6CTRL;  /* Pin 6 Control */
01117     register8_t PIN7CTRL;  /* Pin 7 Control */
01118     register8_t reserved_0x18;
01119     register8_t reserved_0x19;
01120     register8_t reserved_0x1A;
01121     register8_t reserved_0x1B;
01122     register8_t reserved_0x1C;
01123     register8_t reserved_0x1D;
01124     register8_t reserved_0x1E;
01125     register8_t reserved_0x1F;
01126 } PORT_t;
01127
01128 /* Input/Sense Configuration select */
01129 typedef enum PORT_ISC_enum
01130 {
01131     PORT_ISC_INTDISABLE_gc = (0x00«0),  /* Interrupt disabled but input buffer enabled */
01132     PORT_ISC_BOTHEDGES_gc = (0x01«0),  /* Sense Both Edges */
01133     PORT_ISC_RISING_gc = (0x02«0),  /* Sense Rising Edge */
01134     PORT_ISC_FALLING_gc = (0x03«0),  /* Sense Falling Edge */
01135     PORT_ISC_INPUT_DISABLE_gc = (0x04«0),  /* Digital Input Buffer disabled */
01136     PORT_ISC_LEVEL_gc = (0x05«0),  /* Sense low Level */
01137 } PORT_ISC_t;
01138
01139 /*
01140 --------------------------------------------------------------------------
01141 PORTMUX - Port Multiplexer
```

```
01142 ------------------------------------------------------------------------
01143 */
01144
01145 /* Port Multiplexer */
01146 typedef struct PORTMUX_struct
01147 {
01148     register8_t CTRLA;  /* Port Multiplexer Control A */
01149     register8_t CTRLB;  /* Port Multiplexer Control B */
01150     register8_t CTRLC;  /* Port Multiplexer Control C */
01151     register8_t CTRLD;  /* Port Multiplexer Control D */
01152     register8_t reserved_0x04;
01153     register8_t reserved_0x05;
01154     register8_t reserved_0x06;
01155     register8_t reserved_0x07;
01156     register8_t reserved_0x08;
01157     register8_t reserved_0x09;
01158     register8_t reserved_0x0A;
01159     register8_t reserved_0x0B;
01160     register8_t reserved_0x0C;
01161     register8_t reserved_0x0D;
01162     register8_t reserved_0x0E;
01163     register8_t reserved_0x0F;
01164 } PORTMUX_t;
01165
01166 /* Configurable Custom Logic LUT0 select */
01167 typedef enum PORTMUX_LUT0_enum
01168 {
01169     PORTMUX_LUT0_DEFAULT_gc = (0x00<<4),  /* Default pin */
01170     PORTMUX_LUT0_ALTERNATE_gc = (0x01<<4),  /* Alternate pin */
01171 } PORTMUX_LUT0_t;
01172
01173 /* Configurable Custom Logic LUT1 select */
01174 typedef enum PORTMUX_LUT1_enum
01175 {
01176     PORTMUX_LUT1_DEFAULT_gc = (0x00<<5),  /* Default pin */
01177     PORTMUX_LUT1_ALTERNATE_gc = (0x01<<5),  /* Alternate pin */
01178 } PORTMUX_LUT1_t;
01179
01180 /* Port Multiplexer SPI0 select */
01181 typedef enum PORTMUX_SPI0_enum
01182 {
01183     PORTMUX_SPI0_DEFAULT_gc = (0x00<<2),  /* Default pins */
01184     PORTMUX_SPI0_ALTERNATE_gc = (0x01<<2),  /* Alternate pins */
01185 } PORTMUX_SPI0_t;
01186
01187 /* Port Multiplexer TCA0 Output 0 select */
01188 typedef enum PORTMUX_TCA00_enum
01189 {
01190     PORTMUX_TCA00_DEFAULT_gc = (0x00<<0),  /* Default pin */
01191     PORTMUX_TCA00_ALTERNATE_gc = (0x01<<0),  /* Alternate pin */
01192 } PORTMUX_TCA00_t;
01193
01194 /* Port Multiplexer TCA0 Output 1 select */
01195 typedef enum PORTMUX_TCA01_enum
01196 {
01197     PORTMUX_TCA01_DEFAULT_gc = (0x00<<1),  /* Default pin */
01198     PORTMUX_TCA01_ALTERNATE_gc = (0x01<<1),  /* Alternate pin */
01199 } PORTMUX_TCA01_t;
01200
01201 /* Port Multiplexer TCA0 Output 2 select */
01202 typedef enum PORTMUX_TCA02_enum
01203 {
01204     PORTMUX_TCA02_DEFAULT_gc = (0x00<<2),  /* Default pin */
01205     PORTMUX_TCA02_ALTERNATE_gc = (0x01<<2),  /* Alternate pin */
01206 } PORTMUX_TCA02_t;
01207
01208 /* Port Multiplexer TCA0 Output 3 select */
01209 typedef enum PORTMUX_TCA03_enum
01210 {
01211     PORTMUX_TCA03_DEFAULT_gc = (0x00<<3),  /* Default pin */
01212     PORTMUX_TCA03_ALTERNATE_gc = (0x01<<3),  /* Alternate pin */
01213 } PORTMUX_TCA03_t;
01214
01215 /* Port Multiplexer TCA0 Output 4 select */
01216 typedef enum PORTMUX_TCA04_enum
01217 {
01218     PORTMUX_TCA04_DEFAULT_gc = (0x00<<4),  /* Default pin */
01219     PORTMUX_TCA04_ALTERNATE_gc = (0x01<<4),  /* Alternate pin */
01220 } PORTMUX_TCA04_t;
01221
01222 /* Port Multiplexer TCA0 Output 5 select */
01223 typedef enum PORTMUX_TCA05_enum
01224 {
01225     PORTMUX_TCA05_DEFAULT_gc = (0x00<<5),  /* Default pin */
01226     PORTMUX_TCA05_ALTERNATE_gc = (0x01<<5),  /* Alternate pin */
01227 } PORTMUX_TCA05_t;
01228
```

```
01229 /* Port Multiplexer TCB select */
01230 typedef enum PORTMUX_TCB0_enum
01231 {
01232     PORTMUX_TCB0_DEFAULT_gc = (0x00<<0),  /* Default pin */
01233     PORTMUX_TCB0_ALTERNATE_gc = (0x01<<0),  /* Alternate pin */
01234 } PORTMUX_TCB0_t;
01235
01236 /* Port Multiplexer TWI0 select */
01237 typedef enum PORTMUX_TWI0_enum
01238 {
01239     PORTMUX_TWI0_DEFAULT_gc = (0x00<<4),  /* Default pins */
01240     PORTMUX_TWI0_ALTERNATE_gc = (0x01<<4),  /* Alternate pins */
01241 } PORTMUX_TWI0_t;
01242
01243 /* Port Multiplexer USART0 select */
01244 typedef enum PORTMUX_USART0_enum
01245 {
01246     PORTMUX_USART0_DEFAULT_gc = (0x00<<0),  /* Default pins */
01247     PORTMUX_USART0_ALTERNATE_gc = (0x01<<0),  /* Alternate pins */
01248 } PORTMUX_USART0_t;
01249
01250 /*
01251 --------------------------------------------------------------------------
01252 RSTCTRL - Reset controller
01253 --------------------------------------------------------------------------
01254 */
01255
01256 /* Reset controller */
01257 typedef struct RSTCTRL_struct
01258 {
01259     register8_t RSTFR;  /* Reset Flags */
01260     register8_t SWRR;  /* Software Reset */
01261     register8_t reserved_0x02;
01262     register8_t reserved_0x03;
01263 } RSTCTRL_t;
01264
01265 /*
01266 --------------------------------------------------------------------------
01267 RTC - Real-Time Counter
01268 --------------------------------------------------------------------------
01269 */
01270
01271 /* Real-Time Counter */
01272 typedef struct RTC_struct
01273 {
01274     register8_t CTRLA;  /* Control A */
01275     register8_t STATUS;  /* Status */
01276     register8_t INTCTRL;  /* Interrupt Control */
01277     register8_t INTFLAGS;  /* Interrupt Flags */
01278     register8_t TEMP;  /* Temporary */
01279     register8_t DBGCTRL;  /* Debug control */
01280     register8_t reserved_0x06;
01281     register8_t CLKSEL;  /* Clock Select */
01282     _WORDREGISTER(CNT);  /* Counter */
01283     _WORDREGISTER(PER);  /* Period */
01284     _WORDREGISTER(CMP);  /* Compare */
01285     register8_t reserved_0x0E;
01286     register8_t reserved_0x0F;
01287     register8_t PITCTRLA;  /* PIT Control A */
01288     register8_t PITSTATUS;  /* PIT Status */
01289     register8_t PITINTCTRL;  /* PIT Interrupt Control */
01290     register8_t PITINTFLAGS;  /* PIT Interrupt Flags */
01291     register8_t reserved_0x14;
01292     register8_t PITDBGCTRL;  /* PIT Debug control */
01293     register8_t reserved_0x16;
01294     register8_t reserved_0x17;
01295     register8_t reserved_0x18;
01296     register8_t reserved_0x19;
01297     register8_t reserved_0x1A;
01298     register8_t reserved_0x1B;
01299     register8_t reserved_0x1C;
01300     register8_t reserved_0x1D;
01301     register8_t reserved_0x1E;
01302     register8_t reserved_0x1F;
01303 } RTC_t;
01304
01305 /* Clock Select select */
01306 typedef enum RTC_CLKSEL_enum
01307 {
01308     RTC_CLKSEL_INT32K_gc = (0x00<<0),  /* Internal 32kHz OSC */
01309     RTC_CLKSEL_INT1K_gc = (0x01<<0),  /* Internal 1kHz OSC */
01310     RTC_CLKSEL_TOSC32K_gc = (0x02<<0),  /* 32KHz Crystal OSC */
01311     RTC_CLKSEL_EXTCLK_gc = (0x03<<0),  /* External Clock */
01312 } RTC_CLKSEL_t;
01313
01314 /* Period select */
01315 typedef enum RTC_PERIOD_enum
```

```
01316 {
01317     RTC_PERIOD_OFF_gc = (0x00<<3),  /* Off */
01318     RTC_PERIOD_CYC4_gc = (0x01<<3),  /* RTC Clock Cycles 4 */
01319     RTC_PERIOD_CYC8_gc = (0x02<<3),  /* RTC Clock Cycles 8 */
01320     RTC_PERIOD_CYC16_gc = (0x03<<3),  /* RTC Clock Cycles 16 */
01321     RTC_PERIOD_CYC32_gc = (0x04<<3),  /* RTC Clock Cycles 32 */
01322     RTC_PERIOD_CYC64_gc = (0x05<<3),  /* RTC Clock Cycles 64 */
01323     RTC_PERIOD_CYC128_gc = (0x06<<3),  /* RTC Clock Cycles 128 */
01324     RTC_PERIOD_CYC256_gc = (0x07<<3),  /* RTC Clock Cycles 256 */
01325     RTC_PERIOD_CYC512_gc = (0x08<<3),  /* RTC Clock Cycles 512 */
01326     RTC_PERIOD_CYC1024_gc = (0x09<<3),  /* RTC Clock Cycles 1024 */
01327     RTC_PERIOD_CYC2048_gc = (0x0A<<3),  /* RTC Clock Cycles 2048 */
01328     RTC_PERIOD_CYC4096_gc = (0x0B<<3),  /* RTC Clock Cycles 4096 */
01329     RTC_PERIOD_CYC8192_gc = (0x0C<<3),  /* RTC Clock Cycles 8192 */
01330     RTC_PERIOD_CYC16384_gc = (0x0D<<3),  /* RTC Clock Cycles 16384 */
01331     RTC_PERIOD_CYC32768_gc = (0x0E<<3),  /* RTC Clock Cycles 32768 */
01332 } RTC_PERIOD_t;
01333
01334 /* Prescaling Factor select */
01335 typedef enum RTC_PRESCALER_enum
01336 {
01337     RTC_PRESCALER_DIV1_gc = (0x00<<3),  /* RTC Clock / 1 */
01338     RTC_PRESCALER_DIV2_gc = (0x01<<3),  /* RTC Clock / 2 */
01339     RTC_PRESCALER_DIV4_gc = (0x02<<3),  /* RTC Clock / 4 */
01340     RTC_PRESCALER_DIV8_gc = (0x03<<3),  /* RTC Clock / 8 */
01341     RTC_PRESCALER_DIV16_gc = (0x04<<3),  /* RTC Clock / 16 */
01342     RTC_PRESCALER_DIV32_gc = (0x05<<3),  /* RTC Clock / 32 */
01343     RTC_PRESCALER_DIV64_gc = (0x06<<3),  /* RTC Clock / 64 */
01344     RTC_PRESCALER_DIV128_gc = (0x07<<3),  /* RTC Clock / 128 */
01345     RTC_PRESCALER_DIV256_gc = (0x08<<3),  /* RTC Clock / 256 */
01346     RTC_PRESCALER_DIV512_gc = (0x09<<3),  /* RTC Clock / 512 */
01347     RTC_PRESCALER_DIV1024_gc = (0x0A<<3),  /* RTC Clock / 1024 */
01348     RTC_PRESCALER_DIV2048_gc = (0x0B<<3),  /* RTC Clock / 2048 */
01349     RTC_PRESCALER_DIV4096_gc = (0x0C<<3),  /* RTC Clock / 4096 */
01350     RTC_PRESCALER_DIV8192_gc = (0x0D<<3),  /* RTC Clock / 8192 */
01351     RTC_PRESCALER_DIV16384_gc = (0x0E<<3),  /* RTC Clock / 16384 */
01352     RTC_PRESCALER_DIV32768_gc = (0x0F<<3),  /* RTC Clock / 32768 */
01353 } RTC_PRESCALER_t;
01354
01355 /*
01356 --------------------------------------------------------------------------
01357 SIGROW - Signature row
01358 --------------------------------------------------------------------------
01359 */
01360
01361 /* Signature row */
01362 typedef struct SIGROW_struct
01363 {
01364     register8_t DEVICEID0;  /* Device ID Byte 0 */
01365     register8_t DEVICEID1;  /* Device ID Byte 1 */
01366     register8_t DEVICEID2;  /* Device ID Byte 2 */
01367     register8_t SERNUM0;  /* Serial Number Byte 0 */
01368     register8_t SERNUM1;  /* Serial Number Byte 1 */
01369     register8_t SERNUM2;  /* Serial Number Byte 2 */
01370     register8_t SERNUM3;  /* Serial Number Byte 3 */
01371     register8_t SERNUM4;  /* Serial Number Byte 4 */
01372     register8_t SERNUM5;  /* Serial Number Byte 5 */
01373     register8_t SERNUM6;  /* Serial Number Byte 6 */
01374     register8_t SERNUM7;  /* Serial Number Byte 7 */
01375     register8_t SERNUM8;  /* Serial Number Byte 8 */
01376     register8_t SERNUM9;  /* Serial Number Byte 9 */
01377     register8_t reserved_0x0D;
01378     register8_t reserved_0x0E;
01379     register8_t reserved_0x0F;
01380     register8_t reserved_0x10;
01381     register8_t reserved_0x11;
01382     register8_t reserved_0x12;
01383     register8_t reserved_0x13;
01384     register8_t reserved_0x14;
01385     register8_t reserved_0x15;
01386     register8_t reserved_0x16;
01387     register8_t reserved_0x17;
01388     register8_t reserved_0x18;
01389     register8_t reserved_0x19;
01390     register8_t reserved_0x1A;
01391     register8_t reserved_0x1B;
01392     register8_t reserved_0x1C;
01393     register8_t reserved_0x1D;
01394     register8_t reserved_0x1E;
01395     register8_t reserved_0x1F;
01396     register8_t TEMPSENSE0;  /* Temperature Sensor Calibration Byte 0 */
01397     register8_t TEMPSENSE1;  /* Temperature Sensor Calibration Byte 1 */
01398     register8_t OSC16ERR3V;  /* OSC16 error at 3V */
01399     register8_t OSC16ERR5V;  /* OSC16 error at 5V */
01400     register8_t OSC20ERR3V;  /* OSC20 error at 3V */
01401     register8_t OSC20ERR5V;  /* OSC20 error at 5V */
01402     register8_t reserved_0x26;
```

```
01403     register8_t reserved_0x27;
01404     register8_t reserved_0x28;
01405     register8_t reserved_0x29;
01406     register8_t reserved_0x2A;
01407     register8_t reserved_0x2B;
01408     register8_t reserved_0x2C;
01409     register8_t reserved_0x2D;
01410     register8_t reserved_0x2E;
01411     register8_t reserved_0x2F;
01412     register8_t reserved_0x30;
01413     register8_t reserved_0x31;
01414     register8_t reserved_0x32;
01415     register8_t reserved_0x33;
01416     register8_t reserved_0x34;
01417     register8_t reserved_0x35;
01418     register8_t reserved_0x36;
01419     register8_t reserved_0x37;
01420     register8_t reserved_0x38;
01421     register8_t reserved_0x39;
01422     register8_t reserved_0x3A;
01423     register8_t reserved_0x3B;
01424     register8_t reserved_0x3C;
01425     register8_t reserved_0x3D;
01426     register8_t reserved_0x3E;
01427     register8_t reserved_0x3F;
01428 } SIGROW_t;
01429
01430 /*
01431 --------------------------------------------------------------------------
01432 SLPCTRL - Sleep Controller
01433 --------------------------------------------------------------------------
01434 */
01435
01436 /* Sleep Controller */
01437 typedef struct SLPCTRL_struct
01438 {
01439     register8_t CTRLA;  /* Control */
01440     register8_t reserved_0x01;
01441 } SLPCTRL_t;
01442
01443 /* Sleep mode select */
01444 typedef enum SLPCTRL_SMODE_enum
01445 {
01446     SLPCTRL_SMODE_IDLE_gc = (0x00<<1),  /* Idle mode */
01447     SLPCTRL_SMODE_STDBY_gc = (0x01<<1),  /* Standby Mode */
01448     SLPCTRL_SMODE_PDOWN_gc = (0x02<<1),  /* Power-down Mode */
01449 } SLPCTRL_SMODE_t;
01450
01451 #define SLEEP_MODE_IDLE (0x00<<1)
01452 #define SLEEP_MODE_STANDBY (0x01<<1)
01453 #define SLEEP_MODE_PWR_DOWN (0x02<<1)
01454
01455 /*
01456 --------------------------------------------------------------------------
01457 SPI - Serial Peripheral Interface
01458 --------------------------------------------------------------------------
01459 */
01460
01461 /* Serial Peripheral Interface */
01462 typedef struct SPI_struct
01463 {
01464     register8_t CTRLA;  /* Control A */
01465     register8_t CTRLB;  /* Control B */
01466     register8_t INTCTRL;  /* Interrupt Control */
01467     register8_t INTFLAGS;  /* Interrupt Flags */
01468     register8_t DATA;  /* Data */
01469     register8_t reserved_0x05;
01470     register8_t reserved_0x06;
01471     register8_t reserved_0x07;
01472 } SPI_t;
01473
01474 /* SPI Mode select */
01475 typedef enum SPI_MODE_enum
01476 {
01477     SPI_MODE_0_gc = (0x00<<0),  /* SPI Mode 0 */
01478     SPI_MODE_1_gc = (0x01<<0),  /* SPI Mode 1 */
01479     SPI_MODE_2_gc = (0x02<<0),  /* SPI Mode 2 */
01480     SPI_MODE_3_gc = (0x03<<0),  /* SPI Mode 3 */
01481 } SPI_MODE_t;
01482
01483 /* Prescaler select */
01484 typedef enum SPI_PRESC_enum
01485 {
01486     SPI_PRESC_DIV4_gc = (0x00<<1),  /* System Clock / 4 */
01487     SPI_PRESC_DIV16_gc = (0x01<<1),  /* System Clock / 16 */
01488     SPI_PRESC_DIV64_gc = (0x02<<1),  /* System Clock / 64 */
01489     SPI_PRESC_DIV128_gc = (0x03<<1),  /* System Clock / 128 */
```

```
01490 } SPI_PRESC_t;
01491
01492 /*
01493 --------------------------------------------------------------------------
01494 SYSCFG - System Configuration Registers
01495 --------------------------------------------------------------------------
01496 */
01497
01498 /* System Configuration Registers */
01499 typedef struct SYSCFG_struct
01500 {
01501     register8_t reserved_0x00;
01502     register8_t REVID;  /* Revision ID */
01503     register8_t EXTBRK;  /* External Break */
01504     register8_t reserved_0x03;
01505     register8_t reserved_0x04;
01506     register8_t reserved_0x05;
01507     register8_t reserved_0x06;
01508     register8_t reserved_0x07;
01509     register8_t reserved_0x08;
01510     register8_t reserved_0x09;
01511     register8_t reserved_0x0A;
01512     register8_t reserved_0x0B;
01513     register8_t reserved_0x0C;
01514     register8_t reserved_0x0D;
01515     register8_t reserved_0x0E;
01516     register8_t reserved_0x0F;
01517     register8_t reserved_0x10;
01518     register8_t reserved_0x11;
01519     register8_t reserved_0x12;
01520     register8_t reserved_0x13;
01521     register8_t reserved_0x14;
01522     register8_t reserved_0x15;
01523     register8_t reserved_0x16;
01524     register8_t reserved_0x17;
01525     register8_t reserved_0x18;
01526     register8_t reserved_0x19;
01527     register8_t reserved_0x1A;
01528     register8_t reserved_0x1B;
01529     register8_t reserved_0x1C;
01530     register8_t reserved_0x1D;
01531     register8_t reserved_0x1E;
01532     register8_t reserved_0x1F;
01533 } SYSCFG_t;
01534
01535 /*
01536 --------------------------------------------------------------------------
01537 TCA - 16-bit Timer/Counter Type A
01538 --------------------------------------------------------------------------
01539 */
01540
01541 /* 16-bit Timer/Counter Type A - Single Mode */
01542 typedef struct TCA_SINGLE_struct
01543 {
01544     register8_t CTRLA;  /* Control A */
01545     register8_t CTRLB;  /* Control B */
01546     register8_t CTRLC;  /* Control C */
01547     register8_t CTRLD;  /* Control D */
01548     register8_t CTRLECLR;  /* Control E Clear */
01549     register8_t CTRLESET;  /* Control E Set */
01550     register8_t CTRLFCLR;  /* Control F Clear */
01551     register8_t CTRLFSET;  /* Control F Set */
01552     register8_t reserved_0x08;
01553     register8_t EVCTRL;  /* Event Control */
01554     register8_t INTCTRL;  /* Interrupt Control */
01555     register8_t INTFLAGS;  /* Interrupt Flags */
01556     register8_t reserved_0x0C;
01557     register8_t reserved_0x0D;
01558     register8_t DBGCTRL;  /* Degbug Control */
01559     register8_t TEMP;  /* Temporary data for 16-bit Access */
01560     register8_t reserved_0x10;
01561     register8_t reserved_0x11;
01562     register8_t reserved_0x12;
01563     register8_t reserved_0x13;
01564     register8_t reserved_0x14;
01565     register8_t reserved_0x15;
01566     register8_t reserved_0x16;
01567     register8_t reserved_0x17;
01568     register8_t reserved_0x18;
01569     register8_t reserved_0x19;
01570     register8_t reserved_0x1A;
01571     register8_t reserved_0x1B;
01572     register8_t reserved_0x1C;
01573     register8_t reserved_0x1D;
01574     register8_t reserved_0x1E;
01575     register8_t reserved_0x1F;
01576     _WORDREGISTER(CNT);  /* Count */
```

```
01577     register8_t reserved_0x22;
01578     register8_t reserved_0x23;
01579     register8_t reserved_0x24;
01580     register8_t reserved_0x25;
01581     _WORDREGISTER(PER);  /* Period */
01582     _WORDREGISTER(CMP0);  /* Compare 0 */
01583     _WORDREGISTER(CMP1);  /* Compare 1 */
01584     _WORDREGISTER(CMP2);  /* Compare 2 */
01585     register8_t reserved_0x2E;
01586     register8_t reserved_0x2F;
01587     register8_t reserved_0x30;
01588     register8_t reserved_0x31;
01589     register8_t reserved_0x32;
01590     register8_t reserved_0x33;
01591     register8_t reserved_0x34;
01592     register8_t reserved_0x35;
01593     _WORDREGISTER(PERBUF);  /* Period Buffer */
01594     _WORDREGISTER(CMP0BUF);  /* Compare 0 Buffer */
01595     _WORDREGISTER(CMP1BUF);  /* Compare 1 Buffer */
01596     _WORDREGISTER(CMP2BUF);  /* Compare 2 Buffer */
01597     register8_t reserved_0x3E;
01598     register8_t reserved_0x3F;
01599 } TCA_SINGLE_t;
01600
01601 /* 16-bit Timer/Counter Type A - Split Mode */
01602 typedef struct TCA_SPLIT_struct
01603 {
01604     register8_t CTRLA;  /* Control A */
01605     register8_t CTRLB;  /* Control B */
01606     register8_t CTRLC;  /* Control C */
01607     register8_t CTRLD;  /* Control D */
01608     register8_t CTRLECLR;  /* Control E Clear */
01609     register8_t CTRLESET;  /* Control E Set */
01610     register8_t reserved_0x06;
01611     register8_t reserved_0x07;
01612     register8_t reserved_0x08;
01613     register8_t reserved_0x09;
01614     register8_t INTCTRL;  /* Interrupt Control */
01615     register8_t INTFLAGS;  /* Interrupt Flags */
01616     register8_t reserved_0x0C;
01617     register8_t reserved_0x0D;
01618     register8_t DBGCTRL;  /* Degbug Control */
01619     register8_t reserved_0x0F;
01620     register8_t reserved_0x10;
01621     register8_t reserved_0x11;
01622     register8_t reserved_0x12;
01623     register8_t reserved_0x13;
01624     register8_t reserved_0x14;
01625     register8_t reserved_0x15;
01626     register8_t reserved_0x16;
01627     register8_t reserved_0x17;
01628     register8_t reserved_0x18;
01629     register8_t reserved_0x19;
01630     register8_t reserved_0x1A;
01631     register8_t reserved_0x1B;
01632     register8_t reserved_0x1C;
01633     register8_t reserved_0x1D;
01634     register8_t reserved_0x1E;
01635     register8_t reserved_0x1F;
01636     register8_t LCNT;  /* Low Count */
01637     register8_t HCNT;  /* High Count */
01638     register8_t reserved_0x22;
01639     register8_t reserved_0x23;
01640     register8_t reserved_0x24;
01641     register8_t reserved_0x25;
01642     register8_t LPER;  /* Low Period */
01643     register8_t HPER;  /* High Period */
01644     register8_t LCMP0;  /* Low Compare */
01645     register8_t HCMP0;  /* High Compare */
01646     register8_t LCMP1;  /* Low Compare */
01647     register8_t HCMP1;  /* High Compare */
01648     register8_t LCMP2;  /* Low Compare */
01649     register8_t HCMP2;  /* High Compare */
01650     register8_t reserved_0x2E;
01651     register8_t reserved_0x2F;
01652     register8_t reserved_0x30;
01653     register8_t reserved_0x31;
01654     register8_t reserved_0x32;
01655     register8_t reserved_0x33;
01656     register8_t reserved_0x34;
01657     register8_t reserved_0x35;
01658     register8_t reserved_0x36;
01659     register8_t reserved_0x37;
01660     register8_t reserved_0x38;
01661     register8_t reserved_0x39;
01662     register8_t reserved_0x3A;
01663     register8_t reserved_0x3B;
```

```
01664     register8_t reserved_0x3C;
01665     register8_t reserved_0x3D;
01666     register8_t reserved_0x3E;
01667     register8_t reserved_0x3F;
01668 } TCA_SPLIT_t;
01669
01670 /* 16-bit Timer/Counter Type A */
01671 typedef union TCA_union
01672 {
01673     TCA_SINGLE_t SINGLE;  /* 16-bit Timer/Counter Type A - Single Mode */
01674     TCA_SPLIT_t SPLIT;  /* 16-bit Timer/Counter Type A - Split Mode */
01675 } TCA_t;
01676
01677 /* Clock Selection select */
01678 typedef enum TCA_SINGLE_CLKSEL_enum
01679 {
01680     TCA_SINGLE_CLKSEL_DIV1_gc = (0x00<<1),  /* System Clock */
01681     TCA_SINGLE_CLKSEL_DIV2_gc = (0x01<<1),  /* System Clock / 2 */
01682     TCA_SINGLE_CLKSEL_DIV4_gc = (0x02<<1),  /* System Clock / 4 */
01683     TCA_SINGLE_CLKSEL_DIV8_gc = (0x03<<1),  /* System Clock / 8 */
01684     TCA_SINGLE_CLKSEL_DIV16_gc = (0x04<<1),  /* System Clock / 16 */
01685     TCA_SINGLE_CLKSEL_DIV64_gc = (0x05<<1),  /* System Clock / 64 */
01686     TCA_SINGLE_CLKSEL_DIV256_gc = (0x06<<1),  /* System Clock / 256 */
01687     TCA_SINGLE_CLKSEL_DIV1024_gc = (0x07<<1),  /* System Clock / 1024 */
01688 } TCA_SINGLE_CLKSEL_t;
01689
01690 /* Command select */
01691 typedef enum TCA_SINGLE_CMD_enum
01692 {
01693     TCA_SINGLE_CMD_NONE_gc = (0x00<<2),  /* No Command */
01694     TCA_SINGLE_CMD_UPDATE_gc = (0x01<<2),  /* Force Update */
01695     TCA_SINGLE_CMD_RESTART_gc = (0x02<<2),  /* Force Restart */
01696     TCA_SINGLE_CMD_RESET_gc = (0x03<<2),  /* Force Hard Reset */
01697 } TCA_SINGLE_CMD_t;
01698
01699 /* Direction select */
01700 typedef enum TCA_SINGLE_DIR_enum
01701 {
01702     TCA_SINGLE_DIR_UP_gc = (0x00<<0),  /* Count up */
01703     TCA_SINGLE_DIR_DOWN_gc = (0x01<<0),  /* Count down */
01704 } TCA_SINGLE_DIR_t;
01705
01706 /* Event Action select */
01707 typedef enum TCA_SINGLE_EVACT_enum
01708 {
01709     TCA_SINGLE_EVACT_POSEDGE_gc = (0x00<<1),  /* Count on positive edge event */
01710     TCA_SINGLE_EVACT_ANYEDGE_gc = (0x01<<1),  /* Count on any edge event */
01711     TCA_SINGLE_EVACT_HIGHLVL_gc = (0x02<<1),  /* Count on prescaled clock while event line is 1. */
01712     TCA_SINGLE_EVACT_UPDOWN_gc = (0x03<<1),  /* Count on prescaled clock. Event controls count
      direction. Up-count when event line is 0, down-count when event line is 1. */
01713 } TCA_SINGLE_EVACT_t;
01714
01715 /* Waveform generation mode select */
01716 typedef enum TCA_SINGLE_WGMODE_enum
01717 {
01718     TCA_SINGLE_WGMODE_NORMAL_gc = (0x00<<0),  /* Normal Mode */
01719     TCA_SINGLE_WGMODE_FRQ_gc = (0x01<<0),  /* Frequency Generation Mode */
01720     TCA_SINGLE_WGMODE_SINGLESLOPE_gc = (0x03<<0),  /* Single Slope PWM */
01721     TCA_SINGLE_WGMODE_DSTOP_gc = (0x05<<0),  /* Dual Slope PWM, overflow on TOP */
01722     TCA_SINGLE_WGMODE_DSBOTH_gc = (0x06<<0),  /* Dual Slope PWM, overflow on TOP and BOTTOM */
01723     TCA_SINGLE_WGMODE_DSBOTTOM_gc = (0x07<<0),  /* Dual Slope PWM, overflow on BOTTOM */
01724 } TCA_SINGLE_WGMODE_t;
01725
01726 /* Clock Selection select */
01727 typedef enum TCA_SPLIT_CLKSEL_enum
01728 {
01729     TCA_SPLIT_CLKSEL_DIV1_gc = (0x00<<1),  /* System Clock */
01730     TCA_SPLIT_CLKSEL_DIV2_gc = (0x01<<1),  /* System Clock / 2 */
01731     TCA_SPLIT_CLKSEL_DIV4_gc = (0x02<<1),  /* System Clock / 4 */
01732     TCA_SPLIT_CLKSEL_DIV8_gc = (0x03<<1),  /* System Clock / 8 */
01733     TCA_SPLIT_CLKSEL_DIV16_gc = (0x04<<1),  /* System Clock / 16 */
01734     TCA_SPLIT_CLKSEL_DIV64_gc = (0x05<<1),  /* System Clock / 64 */
01735     TCA_SPLIT_CLKSEL_DIV256_gc = (0x06<<1),  /* System Clock / 256 */
01736     TCA_SPLIT_CLKSEL_DIV1024_gc = (0x07<<1),  /* System Clock / 1024 */
01737 } TCA_SPLIT_CLKSEL_t;
01738
01739 /* Command select */
01740 typedef enum TCA_SPLIT_CMD_enum
01741 {
01742     TCA_SPLIT_CMD_NONE_gc = (0x00<<2),  /* No Command */
01743     TCA_SPLIT_CMD_UPDATE_gc = (0x01<<2),  /* Force Update */
01744     TCA_SPLIT_CMD_RESTART_gc = (0x02<<2),  /* Force Restart */
01745     TCA_SPLIT_CMD_RESET_gc = (0x03<<2),  /* Force Hard Reset */
01746 } TCA_SPLIT_CMD_t;
01747
01748 /*
01749 --------------------------------------------------------------------------
```

```
01750 TCB - 16-bit Timer Type B
01751 ------------------------------------------------------------------------
01752 */
01753
01754 /* 16-bit Timer Type B */
01755 typedef struct TCB_struct
01756 {
01757     register8_t CTRLA;  /* Control A */
01758     register8_t CTRLB;  /* Control Register B */
01759     register8_t reserved_0x02;
01760     register8_t reserved_0x03;
01761     register8_t EVCTRL;  /* Event Control */
01762     register8_t INTCTRL;  /* Interrupt Control */
01763     register8_t INTFLAGS;  /* Interrupt Flags */
01764     register8_t STATUS;  /* Status */
01765     register8_t DBGCTRL;  /* Debug Control */
01766     register8_t TEMP;  /* Temporary Value */
01767     _WORDREGISTER(CNT);  /* Count */
01768     _WORDREGISTER(CCMP);  /* Compare or Capture */
01769     register8_t reserved_0x0E;
01770     register8_t reserved_0x0F;
01771 } TCB_t;
01772
01773 /* Clock Select select */
01774 typedef enum TCB_CLKSEL_enum
01775 {
01776     TCB_CLKSEL_CLKDIV1_gc = (0x00<<1),  /* CLK_PER (No Prescaling) */
01777     TCB_CLKSEL_CLKDIV2_gc = (0x01<<1),  /* CLK_PER/2 (From Prescaler) */
01778     TCB_CLKSEL_CLKTCA_gc = (0x02<<1),  /* Use Clock from TCA */
01779 } TCB_CLKSEL_t;
01780
01781 /* Timer Mode select */
01782 typedef enum TCB_CNTMODE_enum
01783 {
01784     TCB_CNTMODE_INT_gc = (0x00<<0),  /* Periodic Interrupt */
01785     TCB_CNTMODE_TIMEOUT_gc = (0x01<<0),  /* Periodic Timeout */
01786     TCB_CNTMODE_CAPT_gc = (0x02<<0),  /* Input Capture Event */
01787     TCB_CNTMODE_FRQ_gc = (0x03<<0),  /* Input Capture Frequency measurement */
01788     TCB_CNTMODE_PW_gc = (0x04<<0),  /* Input Capture Pulse-Width measurement */
01789     TCB_CNTMODE_FRQPW_gc = (0x05<<0),  /* Input Capture Frequency and Pulse-Width measurement */
01790     TCB_CNTMODE_SINGLE_gc = (0x06<<0),  /* Single Shot */
01791     TCB_CNTMODE_PWM8_gc = (0x07<<0),  /* 8-bit PWM */
01792 } TCB_CNTMODE_t;
01793
01794 /*
01795 ------------------------------------------------------------------------
01796 TWI - Two-Wire Interface
01797 ------------------------------------------------------------------------
01798 */
01799
01800 /* Two-Wire Interface */
01801 typedef struct TWI_struct
01802 {
01803     register8_t CTRLA;  /* Control A */
01804     register8_t reserved_0x01;
01805     register8_t DBGCTRL;  /* Debug Control Register */
01806     register8_t MCTRLA;  /* Master Control A */
01807     register8_t MCTRLB;  /* Master Control B */
01808     register8_t MSTATUS;  /* Master Status */
01809     register8_t MBAUD;  /* Master Baurd Rate Control */
01810     register8_t MADDR;  /* Master Address */
01811     register8_t MDATA;  /* Master Data */
01812     register8_t SCTRLA;  /* Slave Control A */
01813     register8_t SCTRLB;  /* Slave Control B */
01814     register8_t SSTATUS;  /* Slave Status */
01815     register8_t SADDR;  /* Slave Address */
01816     register8_t SDATA;  /* Slave Data */
01817     register8_t SADDRMASK;  /* Slave Address Mask */
01818     register8_t reserved_0x0F;
01819 } TWI_t;
01820
01821 /* Acknowledge Action select */
01822 typedef enum TWI_ACKACT_enum
01823 {
01824     TWI_ACKACT_ACK_gc = (0x00<<2),  /* Send ACK */
01825     TWI_ACKACT_NACK_gc = (0x01<<2),  /* Send NACK */
01826 } TWI_ACKACT_t;
01827
01828 /* Slave Address or Stop select */
01829 typedef enum TWI_AP_enum
01830 {
01831     TWI_AP_STOP_gc = (0x00<<0),  /* Stop condition generated APIF */
01832     TWI_AP_ADR_gc = (0x01<<0),  /* Address detection generated APIF */
01833 } TWI_AP_t;
01834
01835 /* Bus State select */
01836 typedef enum TWI_BUSSTATE_enum
```

```
01837 {
01838     TWI_BUSSTATE_UNKNOWN_gc = (0x00<<0),  /* Unknown Bus State */
01839     TWI_BUSSTATE_IDLE_gc = (0x01<<0),  /* Bus is Idle */
01840     TWI_BUSSTATE_OWNER_gc = (0x02<<0),  /* This Module Controls The Bus */
01841     TWI_BUSSTATE_BUSY_gc = (0x03<<0),  /* The Bus is Busy */
01842 } TWI_BUSSTATE_t;
01843
01844 /* Command select */
01845 typedef enum TWI_MCMD_enum
01846 {
01847     TWI_MCMD_NOACT_gc = (0x00<<0),  /* No Action */
01848     TWI_MCMD_REPSTART_gc = (0x01<<0),  /* Issue Repeated Start Condition */
01849     TWI_MCMD_RECVTRANS_gc = (0x02<<0),  /* Receive or Transmit Data, depending on DIR */
01850     TWI_MCMD_STOP_gc = (0x03<<0),  /* Issue Stop Condition */
01851 } TWI_MCMD_t;
01852
01853 /* Command select */
01854 typedef enum TWI_SCMD_enum
01855 {
01856     TWI_SCMD_NOACT_gc = (0x00<<0),  /* No Action */
01857     TWI_SCMD_COMPTRANS_gc = (0x02<<0),  /* Used To Complete a Transaction */
01858     TWI_SCMD_RESPONSE_gc = (0x03<<0),  /* Used in Response to Address/Data Interrupt */
01859 } TWI_SCMD_t;
01860
01861 /* SDA Hold Time select */
01862 typedef enum TWI_SDAHOLD_enum
01863 {
01864     TWI_SDAHOLD_OFF_gc = (0x00<<2),  /* SDA hold time off */
01865     TWI_SDAHOLD_50NS_gc = (0x01<<2),  /* Typical 50ns hold time */
01866     TWI_SDAHOLD_300NS_gc = (0x02<<2),  /* Typical 300ns hold time */
01867     TWI_SDAHOLD_500NS_gc = (0x03<<2),  /* Typical 500ns hold time */
01868 } TWI_SDAHOLD_t;
01869
01870 /* SDA Setup Time select */
01871 typedef enum TWI_SDASETUP_enum
01872 {
01873     TWI_SDASETUP_4CYC_gc = (0x00<<4),  /* SDA setup time is 4 clock cycles */
01874     TWI_SDASETUP_8CYC_gc = (0x01<<4),  /* SDA setup time is 8 clock cycles */
01875 } TWI_SDASETUP_t;
01876
01877 /* Inactive Bus Timeout select */
01878 typedef enum TWI_TIMEOUT_enum
01879 {
01880     TWI_TIMEOUT_DISABLED_gc = (0x00<<2),  /* Bus Timeout Disabled */
01881     TWI_TIMEOUT_50US_gc = (0x01<<2),  /* 50 Microseconds */
01882     TWI_TIMEOUT_100US_gc = (0x02<<2),  /* 100 Microseconds */
01883     TWI_TIMEOUT_200US_gc = (0x03<<2),  /* 200 Microseconds */
01884 } TWI_TIMEOUT_t;
01885
01886 /*
01887 --------------------------------------------------------------------------
01888 USART - Universal Synchronous and Asynchronous Receiver and Transmitter
01889 --------------------------------------------------------------------------
01890 */
01891
01892 /* Universal Synchronous and Asynchronous Receiver and Transmitter */
01893 typedef struct USART_struct
01894 {
01895     register8_t RXDATAL;  /* Receive Data Low Byte */
01896     register8_t RXDATAH;  /* Receive Data High Byte */
01897     register8_t TXDATAL;  /* Transmit Data Low Byte */
01898     register8_t TXDATAH;  /* Transmit Data High Byte */
01899     register8_t STATUS;  /* Status */
01900     register8_t CTRLA;  /* Control A */
01901     register8_t CTRLB;  /* Control B */
01902     register8_t CTRLC;  /* Control C */
01903     _WORDREGISTER(BAUD);  /* Baud Rate */
01904     register8_t reserved_0x0A;
01905     register8_t DBGCTRL;  /* Debug Control */
01906     register8_t EVCTRL;  /* Event Control */
01907     register8_t TXPLCTRL;  /* IRCOM Transmitter Pulse Length Control */
01908     register8_t RXPLCTRL;  /* IRCOM Receiver Pulse Length Control */
01909     register8_t reserved_0x0F;
01910 } USART_t;
01911
01912 /* Character Size select */
01913 typedef enum USART_CHSIZE_enum
01914 {
01915     USART_CHSIZE_5BIT_gc = (0x00<<0),  /* Character size: 5 bit */
01916     USART_CHSIZE_6BIT_gc = (0x01<<0),  /* Character size: 6 bit */
01917     USART_CHSIZE_7BIT_gc = (0x02<<0),  /* Character size: 7 bit */
01918     USART_CHSIZE_8BIT_gc = (0x03<<0),  /* Character size: 8 bit */
01919     USART_CHSIZE_9BITL_gc = (0x06<<0),  /* Character size: 9 bit read low byte first */
01920     USART_CHSIZE_9BITH_gc = (0x07<<0),  /* Character size: 9 bit read high byte first */
01921 } USART_CHSIZE_t;
01922
01923 /* Communication Mode select */
```

```
01924 typedef enum USART_CMODE_enum
01925 {
01926     USART_CMODE_ASYNCHRONOUS_gc = (0x00<<6),  /* Asynchronous Mode */
01927     USART_CMODE_SYNCHRONOUS_gc = (0x01<<6),  /* Synchronous Mode */
01928     USART_CMODE_IRCOM_gc = (0x02<<6),  /* Infrared Communication */
01929     USART_CMODE_MSPI_gc = (0x03<<6),  /* Master SPI Mode */
01930 } USART_CMODE_t;
01931
01932 /* Communication Mode select */
01933 typedef enum USART_MSPI_CMODE_enum
01934 {
01935     USART_MSPI_CMODE_ASYNCHRONOUS_gc = (0x00<<6),  /* Asynchronous Mode */
01936     USART_MSPI_CMODE_SYNCHRONOUS_gc = (0x01<<6),  /* Synchronous Mode */
01937     USART_MSPI_CMODE_IRCOM_gc = (0x02<<6),  /* Infrared Communication */
01938     USART_MSPI_CMODE_MSPI_gc = (0x03<<6),  /* Master SPI Mode */
01939 } USART_MSPI_CMODE_t;
01940
01941 /* Parity Mode select */
01942 typedef enum USART_PMODE_enum
01943 {
01944     USART_PMODE_DISABLED_gc = (0x00<<4),  /* No Parity */
01945     USART_PMODE_EVEN_gc = (0x02<<4),  /* Even Parity */
01946     USART_PMODE_ODD_gc = (0x03<<4),  /* Odd Parity */
01947 } USART_PMODE_t;
01948
01949 /* RS485 Mode internal transmitter select */
01950 typedef enum USART_RS485_enum
01951 {
01952     USART_RS485_OFF_gc = (0x00<<0),  /* RS485 Mode disabled */
01953     USART_RS485_EXT_gc = (0x01<<0),  /* RS485 Mode External drive */
01954     USART_RS485_INT_gc = (0x02<<0),  /* RS485 Mode Internal drive */
01955 } USART_RS485_t;
01956
01957 /* Receiver Mode select */
01958 typedef enum USART_RXMODE_enum
01959 {
01960     USART_RXMODE_NORMAL_gc = (0x00<<1),  /* Normal mode */
01961     USART_RXMODE_CLK2X_gc = (0x01<<1),  /* CLK2x mode */
01962     USART_RXMODE_GENAUTO_gc = (0x02<<1),  /* Generic autobaud mode */
01963     USART_RXMODE_LINAUTO_gc = (0x03<<1),  /* LIN constrained autobaud mode */
01964 } USART_RXMODE_t;
01965
01966 /* Stop Bit Mode select */
01967 typedef enum USART_SBMODE_enum
01968 {
01969     USART_SBMODE_1BIT_gc = (0x00<<3),  /* 1 stop bit */
01970     USART_SBMODE_2BIT_gc = (0x01<<3),  /* 2 stop bits */
01971 } USART_SBMODE_t;
01972
01973 /*
01974 --------------------------------------------------------------------------
01975 USERROW - User Row
01976 --------------------------------------------------------------------------
01977 */
01978
01979 /* User Row */
01980 typedef struct USERROW_struct
01981 {
01982     register8_t USERROW0;  /* User Row Byte 0 */
01983     register8_t USERROW1;  /* User Row Byte 1 */
01984     register8_t USERROW2;  /* User Row Byte 2 */
01985     register8_t USERROW3;  /* User Row Byte 3 */
01986     register8_t USERROW4;  /* User Row Byte 4 */
01987     register8_t USERROW5;  /* User Row Byte 5 */
01988     register8_t USERROW6;  /* User Row Byte 6 */
01989     register8_t USERROW7;  /* User Row Byte 7 */
01990     register8_t USERROW8;  /* User Row Byte 8 */
01991     register8_t USERROW9;  /* User Row Byte 9 */
01992     register8_t USERROW10;  /* User Row Byte 10 */
01993     register8_t USERROW11;  /* User Row Byte 11 */
01994     register8_t USERROW12;  /* User Row Byte 12 */
01995     register8_t USERROW13;  /* User Row Byte 13 */
01996     register8_t USERROW14;  /* User Row Byte 14 */
01997     register8_t USERROW15;  /* User Row Byte 15 */
01998     register8_t USERROW16;  /* User Row Byte 16 */
01999     register8_t USERROW17;  /* User Row Byte 17 */
02000     register8_t USERROW18;  /* User Row Byte 18 */
02001     register8_t USERROW19;  /* User Row Byte 19 */
02002     register8_t USERROW20;  /* User Row Byte 20 */
02003     register8_t USERROW21;  /* User Row Byte 21 */
02004     register8_t USERROW22;  /* User Row Byte 22 */
02005     register8_t USERROW23;  /* User Row Byte 23 */
02006     register8_t USERROW24;  /* User Row Byte 24 */
02007     register8_t USERROW25;  /* User Row Byte 25 */
02008     register8_t USERROW26;  /* User Row Byte 26 */
02009     register8_t USERROW27;  /* User Row Byte 27 */
02010     register8_t USERROW28;  /* User Row Byte 28 */
```

```
02011     register8_t USERROW29;  /* User Row Byte 29 */
02012     register8_t USERROW30;  /* User Row Byte 30 */
02013     register8_t USERROW31;  /* User Row Byte 31 */
02014 } USERROW_t;
02015
02016 /*
02017 --------------------------------------------------------------------------
02018 VPORT - Virtual Ports
02019 --------------------------------------------------------------------------
02020 */
02021
02022 /* Virtual Ports */
02023 typedef struct VPORT_struct
02024 {
02025     register8_t DIR;  /* Data Direction */
02026     register8_t OUT;  /* Output Value */
02027     register8_t IN;  /* Input Value */
02028     register8_t INTFLAGS;  /* Interrupt Flags */
02029 } VPORT_t;
02030
02031 /*
02032 --------------------------------------------------------------------------
02033 VREF - Voltage reference
02034 --------------------------------------------------------------------------
02035 */
02036
02037 /* Voltage reference */
02038 typedef struct VREF_struct
02039 {
02040     register8_t CTRLA;  /* Control A */
02041     register8_t CTRLB;  /* Control B */
02042 } VREF_t;
02043
02044 /* ADC0 reference select select */
02045 typedef enum VREF_ADC0REFSEL_enum
02046 {
02047     VREF_ADC0REFSEL_0V55_gc = (0x00<<4),  /* Voltage reference at 0.55V */
02048     VREF_ADC0REFSEL_1V1_gc = (0x01<<4),  /* Voltage reference at 1.1V */
02049     VREF_ADC0REFSEL_2V5_gc = (0x02<<4),  /* Voltage reference at 2.5V */
02050     VREF_ADC0REFSEL_4V34_gc = (0x03<<4),  /* Voltage reference at 4.34V */
02051     VREF_ADC0REFSEL_1V5_gc = (0x04<<4),  /* Voltage reference at 1.5V */
02052 } VREF_ADC0REFSEL_t;
02053
02054 /* DAC0/AC0 reference select select */
02055 typedef enum VREF_DAC0REFSEL_enum
02056 {
02057     VREF_DAC0REFSEL_0V55_gc = (0x00<<0),  /* Voltage reference at 0.55V */
02058     VREF_DAC0REFSEL_1V1_gc = (0x01<<0),  /* Voltage reference at 1.1V */
02059     VREF_DAC0REFSEL_2V5_gc = (0x02<<0),  /* Voltage reference at 2.5V */
02060     VREF_DAC0REFSEL_4V34_gc = (0x03<<0),  /* Voltage reference at 4.34V */
02061     VREF_DAC0REFSEL_1V5_gc = (0x04<<0),  /* Voltage reference at 1.5V */
02062 } VREF_DAC0REFSEL_t;
02063
02064 /*
02065 --------------------------------------------------------------------------
02066 WDT - Watch-Dog Timer
02067 --------------------------------------------------------------------------
02068 */
02069
02070 /* Watch-Dog Timer */
02071 typedef struct WDT_struct
02072 {
02073     register8_t CTRLA;  /* Control A */
02074     register8_t STATUS;  /* Status */
02075 } WDT_t;
02076
02077 /* Period select */
02078 typedef enum WDT_PERIOD_enum
02079 {
02080     WDT_PERIOD_OFF_gc = (0x00<<0),  /* Watch-Dog timer Off */
02081     WDT_PERIOD_8CLK_gc = (0x01<<0),  /* 8 cycles (8ms) */
02082     WDT_PERIOD_16CLK_gc = (0x02<<0),  /* 16 cycles (16ms) */
02083     WDT_PERIOD_32CLK_gc = (0x03<<0),  /* 32 cycles (32ms) */
02084     WDT_PERIOD_64CLK_gc = (0x04<<0),  /* 64 cycles (64ms) */
02085     WDT_PERIOD_128CLK_gc = (0x05<<0),  /* 128 cycles (0.128s) */
02086     WDT_PERIOD_256CLK_gc = (0x06<<0),  /* 256 cycles (0.256s) */
02087     WDT_PERIOD_512CLK_gc = (0x07<<0),  /* 512 cycles (0.512s) */
02088     WDT_PERIOD_1KCLK_gc = (0x08<<0),  /* 1K cycles (1.0s) */
02089     WDT_PERIOD_2KCLK_gc = (0x09<<0),  /* 2K cycles (2.0s) */
02090     WDT_PERIOD_4KCLK_gc = (0x0A<<0),  /* 4K cycles (4.1s) */
02091     WDT_PERIOD_8KCLK_gc = (0x0B<<0),  /* 8K cycles (8.2s) */
02092 } WDT_PERIOD_t;
02093
02094 /* Window select */
02095 typedef enum WDT_WINDOW_enum
02096 {
02097     WDT_WINDOW_OFF_gc = (0x00<<4),  /* Window mode off */
```

```
02098     WDT_WINDOW_8CLK_gc = (0x01<<4),  /* 8 cycles (8ms) */
02099     WDT_WINDOW_16CLK_gc = (0x02<<4),  /* 16 cycles (16ms) */
02100     WDT_WINDOW_32CLK_gc = (0x03<<4),  /* 32 cycles (32ms) */
02101     WDT_WINDOW_64CLK_gc = (0x04<<4),  /* 64 cycles (64ms) */
02102     WDT_WINDOW_128CLK_gc = (0x05<<4),  /* 128 cycles (0.128s) */
02103     WDT_WINDOW_256CLK_gc = (0x06<<4),  /* 256 cycles (0.256s) */
02104     WDT_WINDOW_512CLK_gc = (0x07<<4),  /* 512 cycles (0.512s) */
02105     WDT_WINDOW_1KCLK_gc = (0x08<<4),  /* 1K cycles (1.0s) */
02106     WDT_WINDOW_2KCLK_gc = (0x09<<4),  /* 2K cycles (2.0s) */
02107     WDT_WINDOW_4KCLK_gc = (0x0A<<4),  /* 4K cycles (4.1s) */
02108     WDT_WINDOW_8KCLK_gc = (0x0B<<4),  /* 8K cycles (8.2s) */
02109 } WDT_WINDOW_t;
02110
02111 /*
02112 ==========================================================================
02113 IO Module Instances. Mapped to memory.
02114 ==========================================================================
02115 */
02116
02117 #define VPORTA              (*(VPORT_t *) 0x0000) /* Virtual Ports */
02118 #define VPORTB              (*(VPORT_t *) 0x0004) /* Virtual Ports */
02119 #define VPORTC              (*(VPORT_t *) 0x0008) /* Virtual Ports */
02120 #define RSTCTRL           (*(RSTCTRL_t *) 0x0040) /* Reset controller */
02121 #define SLPCTRL           (*(SLPCTRL_t *) 0x0050) /* Sleep Controller */
02122 #define CLKCTRL           (*(CLKCTRL_t *) 0x0060) /* Clock controller */
02123 #define BOD                   (*(BOD_t *) 0x0080) /* Bod interface */
02124 #define VREF                 (*(VREF_t *) 0x00A0) /* Voltage reference */
02125 #define WDT                   (*(WDT_t *) 0x0100) /* Watch-Dog Timer */
02126 #define CPUINT             (*(CPUINT_t *) 0x0110) /* Interrupt Controller */
02127 #define CRCSCAN           (*(CRCSCAN_t *) 0x0120) /* CRCSCAN */
02128 #define RTC                   (*(RTC_t *) 0x0140) /* Real-Time Counter */
02129 #define EVSYS               (*(EVSYS_t *) 0x0180) /* Event System */
02130 #define CCL                   (*(CCL_t *) 0x01C0) /* Configurable Custom Logic */
02131 #define PORTMUX           (*(PORTMUX_t *) 0x0200) /* Port Multiplexer */
02132 #define PORTA                (*(PORT_t *) 0x0400) /* I/O Ports */
02133 #define PORTB                (*(PORT_t *) 0x0420) /* I/O Ports */
02134 #define ADC0                  (*(ADC_t *) 0x0600) /* Analog to Digital Converter */
02135 #define AC0                    (*(AC_t *) 0x0670) /* Analog Comparator */
02136 #define USART0              (*(USART_t *) 0x0800) /* Universal Synchronous and Asynchronous Receiver
    and Transmitter */
02137 #define TWI0                  (*(TWI_t *) 0x0810) /* Two-Wire Interface */
02138 #define SPI0                  (*(SPI_t *) 0x0820) /* Serial Peripheral Interface */
02139 #define TCA0                  (*(TCA_t *) 0x0A00) /* 16-bit Timer/Counter Type A */
02140 #define TCB0                  (*(TCB_t *) 0x0A40) /* 16-bit Timer Type B */
02141 #define SYSCFG             (*(SYSCFG_t *) 0x0F00) /* System Configuration Registers */
02142 #define NVMCTRL           (*(NVMCTRL_t *) 0x1000) /* Non-volatile Memory Controller */
02143 #define SIGROW             (*(SIGROW_t *) 0x1100) /* Signature row */
02144 #define FUSE                 (*(FUSE_t *) 0x1280) /* Fuses */
02145 #define LOCKBIT           (*(LOCKBIT_t *) 0x128A) /* Lockbit */
02146 #define USERROW           (*(USERROW_t *) 0x1300) /* User Row */
02147
02148 #endif /* !defined (__ASSEMBLER__) */
02149
02150
02151 /* ========== Flattened fully qualified IO register names ========== */
02152
02153
02154 /* VPORT (VPORTA) - Virtual Ports */
02155 #define VPORTA_DIR  _SFR_MEM8(0x0000)
02156 #define VPORTA_OUT  _SFR_MEM8(0x0001)
02157 #define VPORTA_IN  _SFR_MEM8(0x0002)
02158 #define VPORTA_INTFLAGS  _SFR_MEM8(0x0003)
02159
02160
02161 /* VPORT (VPORTB) - Virtual Ports */
02162 #define VPORTB_DIR  _SFR_MEM8(0x0004)
02163 #define VPORTB_OUT  _SFR_MEM8(0x0005)
02164 #define VPORTB_IN  _SFR_MEM8(0x0006)
02165 #define VPORTB_INTFLAGS  _SFR_MEM8(0x0007)
02166
02167
02168 /* VPORT (VPORTC) - Virtual Ports */
02169 #define VPORTC_DIR  _SFR_MEM8(0x0008)
02170 #define VPORTC_OUT  _SFR_MEM8(0x0009)
02171 #define VPORTC_IN  _SFR_MEM8(0x000A)
02172 #define VPORTC_INTFLAGS  _SFR_MEM8(0x000B)
02173
02174
02175 /* GPIO - General Purpose IO */
02176 #define GPIO_GPIOR0  _SFR_MEM8(0x001C)
02177 #define GPIO_GPIOR1  _SFR_MEM8(0x001D)
02178 #define GPIO_GPIOR2  _SFR_MEM8(0x001E)
02179 #define GPIO_GPIOR3  _SFR_MEM8(0x001F)
02180
02181
02182 /* CPU - CPU */
02183 #define CPU_CCP  _SFR_MEM8(0x0034)
```

```
02184 #define CPU_SPL  _SFR_MEM8(0x003D)
02185 #define CPU_SPH  _SFR_MEM8(0x003E)
02186 #define CPU_SREG  _SFR_MEM8(0x003F)
02187
02188
02189 /* RSTCTRL - Reset controller */
02190 #define RSTCTRL_RSTFR  _SFR_MEM8(0x0040)
02191 #define RSTCTRL_SWRR  _SFR_MEM8(0x0041)
02192
02193
02194 /* SLPCTRL - Sleep Controller */
02195 #define SLPCTRL_CTRLA  _SFR_MEM8(0x0050)
02196
02197
02198 /* CLKCTRL - Clock controller */
02199 #define CLKCTRL_MCLKCTRLA  _SFR_MEM8(0x0060)
02200 #define CLKCTRL_MCLKCTRLB  _SFR_MEM8(0x0061)
02201 #define CLKCTRL_MCLKLOCK  _SFR_MEM8(0x0062)
02202 #define CLKCTRL_MCLKSTATUS  _SFR_MEM8(0x0063)
02203 #define CLKCTRL_OSC20MCTRLA  _SFR_MEM8(0x0070)
02204 #define CLKCTRL_OSC20MCALIBA  _SFR_MEM8(0x0071)
02205 #define CLKCTRL_OSC20MCALIBB  _SFR_MEM8(0x0072)
02206 #define CLKCTRL_OSC32KCTRLA  _SFR_MEM8(0x0078)
02207
02208
02209 /* BOD - Bod interface */
02210 #define BOD_CTRLA  _SFR_MEM8(0x0080)
02211 #define BOD_CTRLB  _SFR_MEM8(0x0081)
02212 #define BOD_VLMCTRLA  _SFR_MEM8(0x0088)
02213 #define BOD_INTCTRL  _SFR_MEM8(0x0089)
02214 #define BOD_INTFLAGS  _SFR_MEM8(0x008A)
02215 #define BOD_STATUS  _SFR_MEM8(0x008B)
02216
02217
02218 /* VREF - Voltage reference */
02219 #define VREF_CTRLA  _SFR_MEM8(0x00A0)
02220 #define VREF_CTRLB  _SFR_MEM8(0x00A1)
02221
02222
02223 /* WDT - Watch-Dog Timer */
02224 #define WDT_CTRLA  _SFR_MEM8(0x0100)
02225 #define WDT_STATUS  _SFR_MEM8(0x0101)
02226
02227
02228 /* CPUINT - Interrupt Controller */
02229 #define CPUINT_CTRLA  _SFR_MEM8(0x0110)
02230 #define CPUINT_STATUS  _SFR_MEM8(0x0111)
02231 #define CPUINT_LVL0PRI  _SFR_MEM8(0x0112)
02232 #define CPUINT_LVL1VEC  _SFR_MEM8(0x0113)
02233
02234
02235 /* CRCSCAN - CRCSCAN */
02236 #define CRCSCAN_CTRLA  _SFR_MEM8(0x0120)
02237 #define CRCSCAN_CTRLB  _SFR_MEM8(0x0121)
02238 #define CRCSCAN_STATUS  _SFR_MEM8(0x0122)
02239
02240
02241 /* RTC - Real-Time Counter */
02242 #define RTC_CTRLA  _SFR_MEM8(0x0140)
02243 #define RTC_STATUS  _SFR_MEM8(0x0141)
02244 #define RTC_INTCTRL  _SFR_MEM8(0x0142)
02245 #define RTC_INTFLAGS  _SFR_MEM8(0x0143)
02246 #define RTC_TEMP  _SFR_MEM8(0x0144)
02247 #define RTC_DBGCTRL  _SFR_MEM8(0x0145)
02248 #define RTC_CLKSEL  _SFR_MEM8(0x0147)
02249 #define RTC_CNT  _SFR_MEM16(0x0148)
02250 #define RTC_CNTL  _SFR_MEM8(0x0148)
02251 #define RTC_CNTH  _SFR_MEM8(0x0149)
02252 #define RTC_PER  _SFR_MEM16(0x014A)
02253 #define RTC_PERL  _SFR_MEM8(0x014A)
02254 #define RTC_PERH  _SFR_MEM8(0x014B)
02255 #define RTC_CMP  _SFR_MEM16(0x014C)
02256 #define RTC_CMPL  _SFR_MEM8(0x014C)
02257 #define RTC_CMPH  _SFR_MEM8(0x014D)
02258 #define RTC_PITCTRLA  _SFR_MEM8(0x0150)
02259 #define RTC_PITSTATUS  _SFR_MEM8(0x0151)
02260 #define RTC_PITINTCTRL  _SFR_MEM8(0x0152)
02261 #define RTC_PITINTFLAGS  _SFR_MEM8(0x0153)
02262 #define RTC_PITDBGCTRL  _SFR_MEM8(0x0155)
02263
02264
02265 /* EVSYS - Event System */
02266 #define EVSYS_ASYNCSTROBE  _SFR_MEM8(0x0180)
02267 #define EVSYS_SYNCSTROBE  _SFR_MEM8(0x0181)
02268 #define EVSYS_ASYNCCH0  _SFR_MEM8(0x0182)
02269 #define EVSYS_ASYNCCH1  _SFR_MEM8(0x0183)
02270 #define EVSYS_SYNCCH0  _SFR_MEM8(0x018A)
```

```
02271 #define EVSYS_ASYNCUSER0  _SFR_MEM8(0x0192)
02272 #define EVSYS_ASYNCUSER1  _SFR_MEM8(0x0193)
02273 #define EVSYS_ASYNCUSER2  _SFR_MEM8(0x0194)
02274 #define EVSYS_ASYNCUSER3  _SFR_MEM8(0x0195)
02275 #define EVSYS_ASYNCUSER4  _SFR_MEM8(0x0196)
02276 #define EVSYS_ASYNCUSER5  _SFR_MEM8(0x0197)
02277 #define EVSYS_ASYNCUSER6  _SFR_MEM8(0x0198)
02278 #define EVSYS_ASYNCUSER7  _SFR_MEM8(0x0199)
02279 #define EVSYS_ASYNCUSER8  _SFR_MEM8(0x019A)
02280 #define EVSYS_ASYNCUSER9  _SFR_MEM8(0x019B)
02281 #define EVSYS_ASYNCUSER10  _SFR_MEM8(0x019C)
02282 #define EVSYS_SYNCUSER0  _SFR_MEM8(0x01A2)
02283 #define EVSYS_SYNCUSER1  _SFR_MEM8(0x01A3)
02284
02285
02286 /* CCL - Configurable Custom Logic */
02287 #define CCL_CTRLA  _SFR_MEM8(0x01C0)
02288 #define CCL_SEQCTRL0  _SFR_MEM8(0x01C1)
02289 #define CCL_LUT0CTRLA  _SFR_MEM8(0x01C5)
02290 #define CCL_LUT0CTRLB  _SFR_MEM8(0x01C6)
02291 #define CCL_LUT0CTRLC  _SFR_MEM8(0x01C7)
02292 #define CCL_TRUTH0  _SFR_MEM8(0x01C8)
02293 #define CCL_LUT1CTRLA  _SFR_MEM8(0x01C9)
02294 #define CCL_LUT1CTRLB  _SFR_MEM8(0x01CA)
02295 #define CCL_LUT1CTRLC  _SFR_MEM8(0x01CB)
02296 #define CCL_TRUTH1  _SFR_MEM8(0x01CC)
02297
02298
02299 /* PORTMUX - Port Multiplexer */
02300 #define PORTMUX_CTRLA  _SFR_MEM8(0x0200)
02301 #define PORTMUX_CTRLB  _SFR_MEM8(0x0201)
02302 #define PORTMUX_CTRLC  _SFR_MEM8(0x0202)
02303 #define PORTMUX_CTRLD  _SFR_MEM8(0x0203)
02304
02305
02306 /* PORT (PORTA) - I/O Ports */
02307 #define PORTA_DIR  _SFR_MEM8(0x0400)
02308 #define PORTA_DIRSET  _SFR_MEM8(0x0401)
02309 #define PORTA_DIRCLR  _SFR_MEM8(0x0402)
02310 #define PORTA_DIRTGL  _SFR_MEM8(0x0403)
02311 #define PORTA_OUT  _SFR_MEM8(0x0404)
02312 #define PORTA_OUTSET  _SFR_MEM8(0x0405)
02313 #define PORTA_OUTCLR  _SFR_MEM8(0x0406)
02314 #define PORTA_OUTTGL  _SFR_MEM8(0x0407)
02315 #define PORTA_IN  _SFR_MEM8(0x0408)
02316 #define PORTA_INTFLAGS  _SFR_MEM8(0x0409)
02317 #define PORTA_PIN0CTRL  _SFR_MEM8(0x0410)
02318 #define PORTA_PIN1CTRL  _SFR_MEM8(0x0411)
02319 #define PORTA_PIN2CTRL  _SFR_MEM8(0x0412)
02320 #define PORTA_PIN3CTRL  _SFR_MEM8(0x0413)
02321 #define PORTA_PIN4CTRL  _SFR_MEM8(0x0414)
02322 #define PORTA_PIN5CTRL  _SFR_MEM8(0x0415)
02323 #define PORTA_PIN6CTRL  _SFR_MEM8(0x0416)
02324 #define PORTA_PIN7CTRL  _SFR_MEM8(0x0417)
02325
02326
02327 /* PORT (PORTB) - I/O Ports */
02328 #define PORTB_DIR  _SFR_MEM8(0x0420)
02329 #define PORTB_DIRSET  _SFR_MEM8(0x0421)
02330 #define PORTB_DIRCLR  _SFR_MEM8(0x0422)
02331 #define PORTB_DIRTGL  _SFR_MEM8(0x0423)
02332 #define PORTB_OUT  _SFR_MEM8(0x0424)
02333 #define PORTB_OUTSET  _SFR_MEM8(0x0425)
02334 #define PORTB_OUTCLR  _SFR_MEM8(0x0426)
02335 #define PORTB_OUTTGL  _SFR_MEM8(0x0427)
02336 #define PORTB_IN  _SFR_MEM8(0x0428)
02337 #define PORTB_INTFLAGS  _SFR_MEM8(0x0429)
02338 #define PORTB_PIN0CTRL  _SFR_MEM8(0x0430)
02339 #define PORTB_PIN1CTRL  _SFR_MEM8(0x0431)
02340 #define PORTB_PIN2CTRL  _SFR_MEM8(0x0432)
02341 #define PORTB_PIN3CTRL  _SFR_MEM8(0x0433)
02342 #define PORTB_PIN4CTRL  _SFR_MEM8(0x0434)
02343 #define PORTB_PIN5CTRL  _SFR_MEM8(0x0435)
02344 #define PORTB_PIN6CTRL  _SFR_MEM8(0x0436)
02345 #define PORTB_PIN7CTRL  _SFR_MEM8(0x0437)
02346
02347
02348 /* ADC (ADC0) - Analog to Digital Converter */
02349 #define ADC0_CTRLA  _SFR_MEM8(0x0600)
02350 #define ADC0_CTRLB  _SFR_MEM8(0x0601)
02351 #define ADC0_CTRLC  _SFR_MEM8(0x0602)
02352 #define ADC0_CTRLD  _SFR_MEM8(0x0603)
02353 #define ADC0_CTRLE  _SFR_MEM8(0x0604)
02354 #define ADC0_SAMPCTRL  _SFR_MEM8(0x0605)
02355 #define ADC0_MUXPOS  _SFR_MEM8(0x0606)
02356 #define ADC0_COMMAND  _SFR_MEM8(0x0608)
02357 #define ADC0_EVCTRL  _SFR_MEM8(0x0609)
```

```
02358 #define ADC0_INTCTRL  _SFR_MEM8(0x060A)
02359 #define ADC0_INTFLAGS  _SFR_MEM8(0x060B)
02360 #define ADC0_DBGCTRL  _SFR_MEM8(0x060C)
02361 #define ADC0_TEMP  _SFR_MEM8(0x060D)
02362 #define ADC0_RES  _SFR_MEM16(0x0610)
02363 #define ADC0_RESL  _SFR_MEM8(0x0610)
02364 #define ADC0_RESH  _SFR_MEM8(0x0611)
02365 #define ADC0_WINLT  _SFR_MEM16(0x0612)
02366 #define ADC0_WINLTL  _SFR_MEM8(0x0612)
02367 #define ADC0_WINLTH  _SFR_MEM8(0x0613)
02368 #define ADC0_WINHT  _SFR_MEM16(0x0614)
02369 #define ADC0_WINHTL  _SFR_MEM8(0x0614)
02370 #define ADC0_WINHTH  _SFR_MEM8(0x0615)
02371 #define ADC0_CALIB  _SFR_MEM8(0x0616)
02372
02373
02374 /* AC (AC0) - Analog Comparator */
02375 #define AC0_CTRLA  _SFR_MEM8(0x0670)
02376 #define AC0_MUXCTRLA  _SFR_MEM8(0x0672)
02377 #define AC0_INTCTRL  _SFR_MEM8(0x0676)
02378 #define AC0_STATUS  _SFR_MEM8(0x0677)
02379
02380
02381 /* USART (USART0) - Universal Synchronous and Asynchronous Receiver and Transmitter */
02382 #define USART0_RXDATAL  _SFR_MEM8(0x0800)
02383 #define USART0_RXDATAH  _SFR_MEM8(0x0801)
02384 #define USART0_TXDATAL  _SFR_MEM8(0x0802)
02385 #define USART0_TXDATAH  _SFR_MEM8(0x0803)
02386 #define USART0_STATUS  _SFR_MEM8(0x0804)
02387 #define USART0_CTRLA  _SFR_MEM8(0x0805)
02388 #define USART0_CTRLB  _SFR_MEM8(0x0806)
02389 #define USART0_CTRLC  _SFR_MEM8(0x0807)
02390 #define USART0_BAUD  _SFR_MEM16(0x0808)
02391 #define USART0_BAUDL  _SFR_MEM8(0x0808)
02392 #define USART0_BAUDH  _SFR_MEM8(0x0809)
02393 #define USART0_DBGCTRL  _SFR_MEM8(0x080B)
02394 #define USART0_EVCTRL  _SFR_MEM8(0x080C)
02395 #define USART0_TXPLCTRL  _SFR_MEM8(0x080D)
02396 #define USART0_RXPLCTRL  _SFR_MEM8(0x080E)
02397
02398
02399 /* TWI (TWI0) - Two-Wire Interface */
02400 #define TWI0_CTRLA  _SFR_MEM8(0x0810)
02401 #define TWI0_DBGCTRL  _SFR_MEM8(0x0812)
02402 #define TWI0_MCTRLA  _SFR_MEM8(0x0813)
02403 #define TWI0_MCTRLB  _SFR_MEM8(0x0814)
02404 #define TWI0_MSTATUS  _SFR_MEM8(0x0815)
02405 #define TWI0_MBAUD  _SFR_MEM8(0x0816)
02406 #define TWI0_MADDR  _SFR_MEM8(0x0817)
02407 #define TWI0_MDATA  _SFR_MEM8(0x0818)
02408 #define TWI0_SCTRLA  _SFR_MEM8(0x0819)
02409 #define TWI0_SCTRLB  _SFR_MEM8(0x081A)
02410 #define TWI0_SSTATUS  _SFR_MEM8(0x081B)
02411 #define TWI0_SADDR  _SFR_MEM8(0x081C)
02412 #define TWI0_SDATA  _SFR_MEM8(0x081D)
02413 #define TWI0_SADDRMASK  _SFR_MEM8(0x081E)
02414
02415
02416 /* SPI (SPI0) - Serial Peripheral Interface */
02417 #define SPI0_CTRLA  _SFR_MEM8(0x0820)
02418 #define SPI0_CTRLB  _SFR_MEM8(0x0821)
02419 #define SPI0_INTCTRL  _SFR_MEM8(0x0822)
02420 #define SPI0_INTFLAGS  _SFR_MEM8(0x0823)
02421 #define SPI0_DATA  _SFR_MEM8(0x0824)
02422
02423
02424 /* TCA (TCA0) - 16-bit Timer/Counter Type A */
02425 #define TCA0_SINGLE_CTRLA  _SFR_MEM8(0x0A00)
02426 #define TCA0_SINGLE_CTRLB  _SFR_MEM8(0x0A01)
02427 #define TCA0_SINGLE_CTRLC  _SFR_MEM8(0x0A02)
02428 #define TCA0_SINGLE_CTRLD  _SFR_MEM8(0x0A03)
02429 #define TCA0_SINGLE_CTRLECLR  _SFR_MEM8(0x0A04)
02430 #define TCA0_SINGLE_CTRLESET  _SFR_MEM8(0x0A05)
02431 #define TCA0_SINGLE_CTRLFCLR  _SFR_MEM8(0x0A06)
02432 #define TCA0_SINGLE_CTRLFSET  _SFR_MEM8(0x0A07)
02433 #define TCA0_SINGLE_EVCTRL  _SFR_MEM8(0x0A09)
02434 #define TCA0_SINGLE_INTCTRL  _SFR_MEM8(0x0A0A)
02435 #define TCA0_SINGLE_INTFLAGS  _SFR_MEM8(0x0A0B)
02436 #define TCA0_SINGLE_DBGCTRL  _SFR_MEM8(0x0A0E)
02437 #define TCA0_SINGLE_TEMP  _SFR_MEM8(0x0A0F)
02438 #define TCA0_SINGLE_CNT  _SFR_MEM16(0x0A20)
02439 #define TCA0_SINGLE_PER  _SFR_MEM16(0x0A26)
02440 #define TCA0_SINGLE_CMP0  _SFR_MEM16(0x0A28)
02441 #define TCA0_SINGLE_CMP1  _SFR_MEM16(0x0A2A)
02442 #define TCA0_SINGLE_CMP2  _SFR_MEM16(0x0A2C)
02443 #define TCA0_SINGLE_PERBUF  _SFR_MEM16(0x0A36)
02444 #define TCA0_SINGLE_CMP0BUF  _SFR_MEM16(0x0A38)
```

```
02445 #define TCA0_SINGLE_CMP1BUF  _SFR_MEM16(0x0A3A)
02446 #define TCA0_SINGLE_CMP2BUF  _SFR_MEM16(0x0A3C)
02447 #define TCA0_SPLIT_CTRLA  _SFR_MEM8(0x0A00)
02448 #define TCA0_SPLIT_CTRLB  _SFR_MEM8(0x0A01)
02449 #define TCA0_SPLIT_CTRLC  _SFR_MEM8(0x0A02)
02450 #define TCA0_SPLIT_CTRLD  _SFR_MEM8(0x0A03)
02451 #define TCA0_SPLIT_CTRLECLR  _SFR_MEM8(0x0A04)
02452 #define TCA0_SPLIT_CTRLESET  _SFR_MEM8(0x0A05)
02453 #define TCA0_SPLIT_INTCTRL  _SFR_MEM8(0x0A0A)
02454 #define TCA0_SPLIT_INTFLAGS  _SFR_MEM8(0x0A0B)
02455 #define TCA0_SPLIT_DBGCTRL  _SFR_MEM8(0x0A0E)
02456 #define TCA0_SPLIT_LCNT  _SFR_MEM8(0x0A20)
02457 #define TCA0_SPLIT_HCNT  _SFR_MEM8(0x0A21)
02458 #define TCA0_SPLIT_LPER  _SFR_MEM8(0x0A26)
02459 #define TCA0_SPLIT_HPER  _SFR_MEM8(0x0A27)
02460 #define TCA0_SPLIT_LCMP0  _SFR_MEM8(0x0A28)
02461 #define TCA0_SPLIT_HCMP0  _SFR_MEM8(0x0A29)
02462 #define TCA0_SPLIT_LCMP1  _SFR_MEM8(0x0A2A)
02463 #define TCA0_SPLIT_HCMP1  _SFR_MEM8(0x0A2B)
02464 #define TCA0_SPLIT_LCMP2  _SFR_MEM8(0x0A2C)
02465 #define TCA0_SPLIT_HCMP2  _SFR_MEM8(0x0A2D)
02466
02467
02468 /* TCB (TCB0) - 16-bit Timer Type B */
02469 #define TCB0_CTRLA  _SFR_MEM8(0x0A40)
02470 #define TCB0_CTRLB  _SFR_MEM8(0x0A41)
02471 #define TCB0_EVCTRL  _SFR_MEM8(0x0A44)
02472 #define TCB0_INTCTRL  _SFR_MEM8(0x0A45)
02473 #define TCB0_INTFLAGS  _SFR_MEM8(0x0A46)
02474 #define TCB0_STATUS  _SFR_MEM8(0x0A47)
02475 #define TCB0_DBGCTRL  _SFR_MEM8(0x0A48)
02476 #define TCB0_TEMP  _SFR_MEM8(0x0A49)
02477 #define TCB0_CNT  _SFR_MEM16(0x0A4A)
02478 #define TCB0_CNTL  _SFR_MEM8(0x0A4A)
02479 #define TCB0_CNTH  _SFR_MEM8(0x0A4B)
02480 #define TCB0_CCMP  _SFR_MEM16(0x0A4C)
02481 #define TCB0_CCMPL  _SFR_MEM8(0x0A4C)
02482 #define TCB0_CCMPH  _SFR_MEM8(0x0A4D)
02483
02484
02485 /* SYSCFG - System Configuration Registers */
02486 #define SYSCFG_REVID  _SFR_MEM8(0x0F01)
02487 #define SYSCFG_EXTBRK  _SFR_MEM8(0x0F02)
02488
02489
02490 /* NVMCTRL - Non-volatile Memory Controller */
02491 #define NVMCTRL_CTRLA  _SFR_MEM8(0x1000)
02492 #define NVMCTRL_CTRLB  _SFR_MEM8(0x1001)
02493 #define NVMCTRL_STATUS  _SFR_MEM8(0x1002)
02494 #define NVMCTRL_INTCTRL  _SFR_MEM8(0x1003)
02495 #define NVMCTRL_INTFLAGS  _SFR_MEM8(0x1004)
02496 #define NVMCTRL_DATA  _SFR_MEM16(0x1006)
02497 #define NVMCTRL_DATAL  _SFR_MEM8(0x1006)
02498 #define NVMCTRL_DATAH  _SFR_MEM8(0x1007)
02499 #define NVMCTRL_ADDR  _SFR_MEM16(0x1008)
02500 #define NVMCTRL_ADDRL  _SFR_MEM8(0x1008)
02501 #define NVMCTRL_ADDRH  _SFR_MEM8(0x1009)
02502
02503
02504 /* SIGROW - Signature row */
02505 #define SIGROW_DEVICEID0  _SFR_MEM8(0x1100)
02506 #define SIGROW_DEVICEID1  _SFR_MEM8(0x1101)
02507 #define SIGROW_DEVICEID2  _SFR_MEM8(0x1102)
02508 #define SIGROW_SERNUM0  _SFR_MEM8(0x1103)
02509 #define SIGROW_SERNUM1  _SFR_MEM8(0x1104)
02510 #define SIGROW_SERNUM2  _SFR_MEM8(0x1105)
02511 #define SIGROW_SERNUM3  _SFR_MEM8(0x1106)
02512 #define SIGROW_SERNUM4  _SFR_MEM8(0x1107)
02513 #define SIGROW_SERNUM5  _SFR_MEM8(0x1108)
02514 #define SIGROW_SERNUM6  _SFR_MEM8(0x1109)
02515 #define SIGROW_SERNUM7  _SFR_MEM8(0x110A)
02516 #define SIGROW_SERNUM8  _SFR_MEM8(0x110B)
02517 #define SIGROW_SERNUM9  _SFR_MEM8(0x110C)
02518 #define SIGROW_TEMPSENSE0  _SFR_MEM8(0x1120)
02519 #define SIGROW_TEMPSENSE1  _SFR_MEM8(0x1121)
02520 #define SIGROW_OSC16ERR3V  _SFR_MEM8(0x1122)
02521 #define SIGROW_OSC16ERR5V  _SFR_MEM8(0x1123)
02522 #define SIGROW_OSC20ERR3V  _SFR_MEM8(0x1124)
02523 #define SIGROW_OSC20ERR5V  _SFR_MEM8(0x1125)
02524
02525
02526 /* FUSE - Fuses */
02527 #define FUSE_WDTCFG  _SFR_MEM8(0x1280)
02528 #define FUSE_BODCFG  _SFR_MEM8(0x1281)
02529 #define FUSE_OSCCFG  _SFR_MEM8(0x1282)
02530 #define FUSE_TCD0CFG  _SFR_MEM8(0x1284)
02531 #define FUSE_SYSCFG0  _SFR_MEM8(0x1285)
```

```
02532 #define FUSE_SYSCFG1  _SFR_MEM8(0x1286)
02533 #define FUSE_APPEND  _SFR_MEM8(0x1287)
02534 #define FUSE_BOOTEND  _SFR_MEM8(0x1288)
02535
02536
02537 /* LOCKBIT - Lockbit */
02538 #define LOCKBIT_LOCKBIT  _SFR_MEM8(0x128A)
02539
02540
02541 /* USERROW - User Row */
02542 #define USERROW_USERROW0  _SFR_MEM8(0x1300)
02543 #define USERROW_USERROW1  _SFR_MEM8(0x1301)
02544 #define USERROW_USERROW2  _SFR_MEM8(0x1302)
02545 #define USERROW_USERROW3  _SFR_MEM8(0x1303)
02546 #define USERROW_USERROW4  _SFR_MEM8(0x1304)
02547 #define USERROW_USERROW5  _SFR_MEM8(0x1305)
02548 #define USERROW_USERROW6  _SFR_MEM8(0x1306)
02549 #define USERROW_USERROW7  _SFR_MEM8(0x1307)
02550 #define USERROW_USERROW8  _SFR_MEM8(0x1308)
02551 #define USERROW_USERROW9  _SFR_MEM8(0x1309)
02552 #define USERROW_USERROW10  _SFR_MEM8(0x130A)
02553 #define USERROW_USERROW11  _SFR_MEM8(0x130B)
02554 #define USERROW_USERROW12  _SFR_MEM8(0x130C)
02555 #define USERROW_USERROW13  _SFR_MEM8(0x130D)
02556 #define USERROW_USERROW14  _SFR_MEM8(0x130E)
02557 #define USERROW_USERROW15  _SFR_MEM8(0x130F)
02558 #define USERROW_USERROW16  _SFR_MEM8(0x1310)
02559 #define USERROW_USERROW17  _SFR_MEM8(0x1311)
02560 #define USERROW_USERROW18  _SFR_MEM8(0x1312)
02561 #define USERROW_USERROW19  _SFR_MEM8(0x1313)
02562 #define USERROW_USERROW20  _SFR_MEM8(0x1314)
02563 #define USERROW_USERROW21  _SFR_MEM8(0x1315)
02564 #define USERROW_USERROW22  _SFR_MEM8(0x1316)
02565 #define USERROW_USERROW23  _SFR_MEM8(0x1317)
02566 #define USERROW_USERROW24  _SFR_MEM8(0x1318)
02567 #define USERROW_USERROW25  _SFR_MEM8(0x1319)
02568 #define USERROW_USERROW26  _SFR_MEM8(0x131A)
02569 #define USERROW_USERROW27  _SFR_MEM8(0x131B)
02570 #define USERROW_USERROW28  _SFR_MEM8(0x131C)
02571 #define USERROW_USERROW29  _SFR_MEM8(0x131D)
02572 #define USERROW_USERROW30  _SFR_MEM8(0x131E)
02573 #define USERROW_USERROW31  _SFR_MEM8(0x131F)
02574
02575
02576
02577 /*================== Bitfield Definitions ================== */
02578
02579 /* AC - Analog Comparator */
02580 /* AC.CTRLA  bit masks and bit positions */
02581 #define AC_ENABLE_bm  0x01  /* Enable bit mask. */
02582 #define AC_ENABLE_bp  0  /* Enable bit position. */
02583 #define AC_HYSMODE_gm  0x06  /* Hysteresis Mode group mask. */
02584 #define AC_HYSMODE_gp  1  /* Hysteresis Mode group position. */
02585 #define AC_HYSMODE0_bm  (1<<1)  /* Hysteresis Mode bit 0 mask. */
02586 #define AC_HYSMODE0_bp  1  /* Hysteresis Mode bit 0 position. */
02587 #define AC_HYSMODE1_bm  (1<<2)  /* Hysteresis Mode bit 1 mask. */
02588 #define AC_HYSMODE1_bp  2  /* Hysteresis Mode bit 1 position. */
02589 #define AC_INTMODE_gm  0x30  /* Interrupt Mode group mask. */
02590 #define AC_INTMODE_gp  4  /* Interrupt Mode group position. */
02591 #define AC_INTMODE0_bm  (1<<4)  /* Interrupt Mode bit 0 mask. */
02592 #define AC_INTMODE0_bp  4  /* Interrupt Mode bit 0 position. */
02593 #define AC_INTMODE1_bm  (1<<5)  /* Interrupt Mode bit 1 mask. */
02594 #define AC_INTMODE1_bp  5  /* Interrupt Mode bit 1 position. */
02595 #define AC_OUTEN_bm  0x40  /* Output Buffer Enable bit mask. */
02596 #define AC_OUTEN_bp  6  /* Output Buffer Enable bit position. */
02597 #define AC_RUNSTDBY_bm  0x80  /* Run in Standby Mode bit mask. */
02598 #define AC_RUNSTDBY_bp  7  /* Run in Standby Mode bit position. */
02599
02600 /* AC.MUXCTRLA  bit masks and bit positions */
02601 #define AC_MUXNEG_gm  0x03  /* Negative Input MUX Selection group mask. */
02602 #define AC_MUXNEG_gp  0  /* Negative Input MUX Selection group position. */
02603 #define AC_MUXNEG0_bm  (1<<0)  /* Negative Input MUX Selection bit 0 mask. */
02604 #define AC_MUXNEG0_bp  0  /* Negative Input MUX Selection bit 0 position. */
02605 #define AC_MUXNEG1_bm  (1<<1)  /* Negative Input MUX Selection bit 1 mask. */
02606 #define AC_MUXNEG1_bp  1  /* Negative Input MUX Selection bit 1 position. */
02607 #define AC_MUXPOS_gm  0x18  /* Positive Input MUX Selection group mask. */
02608 #define AC_MUXPOS_gp  3  /* Positive Input MUX Selection group position. */
02609 #define AC_MUXPOS0_bm  (1<<3)  /* Positive Input MUX Selection bit 0 mask. */
02610 #define AC_MUXPOS0_bp  3  /* Positive Input MUX Selection bit 0 position. */
02611 #define AC_MUXPOS1_bm  (1<<4)  /* Positive Input MUX Selection bit 1 mask. */
02612 #define AC_MUXPOS1_bp  4  /* Positive Input MUX Selection bit 1 position. */
02613 #define AC_INVERT_bm  0x80  /* Invert AC Output bit mask. */
02614 #define AC_INVERT_bp  7  /* Invert AC Output bit position. */
02615
02616 /* AC.INTCTRL  bit masks and bit positions */
02617 #define AC_CMP_bm  0x01  /* Analog Comparator 0 Interrupt Enable bit mask. */
02618 #define AC_CMP_bp  0  /* Analog Comparator 0 Interrupt Enable bit position. */
```

```
02619
02620 /* AC.STATUS  bit masks and bit positions */
02621 /* AC_CMP  is already defined. */
02622 #define AC_STATE_bm  0x10  /* Analog Comparator State bit mask. */
02623 #define AC_STATE_bp  4  /* Analog Comparator State bit position. */
02624
02625 /* ADC - Analog to Digital Converter */
02626 /* ADC.CTRLA  bit masks and bit positions */
02627 #define ADC_ENABLE_bm  0x01  /* ADC Enable bit mask. */
02628 #define ADC_ENABLE_bp  0  /* ADC Enable bit position. */
02629 #define ADC_FREERUN_bm  0x02  /* ADC Freerun mode bit mask. */
02630 #define ADC_FREERUN_bp  1  /* ADC Freerun mode bit position. */
02631 #define ADC_RESSEL_bm  0x04  /* ADC Resolution bit mask. */
02632 #define ADC_RESSEL_bp  2  /* ADC Resolution bit position. */
02633 #define ADC_RUNSTBY_bm  0x80  /* Run standby mode bit mask. */
02634 #define ADC_RUNSTBY_bp  7  /* Run standby mode bit position. */
02635
02636 /* ADC.CTRLB  bit masks and bit positions */
02637 #define ADC_SAMPNUM_gm  0x07  /* Accumulation Samples group mask. */
02638 #define ADC_SAMPNUM_gp  0  /* Accumulation Samples group position. */
02639 #define ADC_SAMPNUM0_bm  (1<<0)  /* Accumulation Samples bit 0 mask. */
02640 #define ADC_SAMPNUM0_bp  0  /* Accumulation Samples bit 0 position. */
02641 #define ADC_SAMPNUM1_bm  (1<<1)  /* Accumulation Samples bit 1 mask. */
02642 #define ADC_SAMPNUM1_bp  1  /* Accumulation Samples bit 1 position. */
02643 #define ADC_SAMPNUM2_bm  (1<<2)  /* Accumulation Samples bit 2 mask. */
02644 #define ADC_SAMPNUM2_bp  2  /* Accumulation Samples bit 2 position. */
02645
02646 /* ADC.CTRLC  bit masks and bit positions */
02647 #define ADC_PRESC_gm  0x07  /* Clock Pre-scaler group mask. */
02648 #define ADC_PRESC_gp  0  /* Clock Pre-scaler group position. */
02649 #define ADC_PRESC0_bm  (1<<0)  /* Clock Pre-scaler bit 0 mask. */
02650 #define ADC_PRESC0_bp  0  /* Clock Pre-scaler bit 0 position. */
02651 #define ADC_PRESC1_bm  (1<<1)  /* Clock Pre-scaler bit 1 mask. */
02652 #define ADC_PRESC1_bp  1  /* Clock Pre-scaler bit 1 position. */
02653 #define ADC_PRESC2_bm  (1<<2)  /* Clock Pre-scaler bit 2 mask. */
02654 #define ADC_PRESC2_bp  2  /* Clock Pre-scaler bit 2 position. */
02655 #define ADC_REFSEL_gm  0x30  /* Reference Selection group mask. */
02656 #define ADC_REFSEL_gp  4  /* Reference Selection group position. */
02657 #define ADC_REFSEL0_bm  (1<<4)  /* Reference Selection bit 0 mask. */
02658 #define ADC_REFSEL0_bp  4  /* Reference Selection bit 0 position. */
02659 #define ADC_REFSEL1_bm  (1<<5)  /* Reference Selection bit 1 mask. */
02660 #define ADC_REFSEL1_bp  5  /* Reference Selection bit 1 position. */
02661 #define ADC_SAMPCAP_bm  0x40  /* Sample Capacitance Selection bit mask. */
02662 #define ADC_SAMPCAP_bp  6  /* Sample Capacitance Selection bit position. */
02663
02664 /* ADC.CTRLD  bit masks and bit positions */
02665 #define ADC_SAMPDLY_gm  0x0F  /* Sampling Delay Selection group mask. */
02666 #define ADC_SAMPDLY_gp  0  /* Sampling Delay Selection group position. */
02667 #define ADC_SAMPDLY0_bm  (1<<0)  /* Sampling Delay Selection bit 0 mask. */
02668 #define ADC_SAMPDLY0_bp  0  /* Sampling Delay Selection bit 0 position. */
02669 #define ADC_SAMPDLY1_bm  (1<<1)  /* Sampling Delay Selection bit 1 mask. */
02670 #define ADC_SAMPDLY1_bp  1  /* Sampling Delay Selection bit 1 position. */
02671 #define ADC_SAMPDLY2_bm  (1<<2)  /* Sampling Delay Selection bit 2 mask. */
02672 #define ADC_SAMPDLY2_bp  2  /* Sampling Delay Selection bit 2 position. */
02673 #define ADC_SAMPDLY3_bm  (1<<3)  /* Sampling Delay Selection bit 3 mask. */
02674 #define ADC_SAMPDLY3_bp  3  /* Sampling Delay Selection bit 3 position. */
02675 #define ADC_ASDV_bm  0x10  /* Automatic Sampling Delay Variation bit mask. */
02676 #define ADC_ASDV_bp  4  /* Automatic Sampling Delay Variation bit position. */
02677 #define ADC_INITDLY_gm  0xE0  /* Initial Delay Selection group mask. */
02678 #define ADC_INITDLY_gp  5  /* Initial Delay Selection group position. */
02679 #define ADC_INITDLY0_bm  (1<<5)  /* Initial Delay Selection bit 0 mask. */
02680 #define ADC_INITDLY0_bp  5  /* Initial Delay Selection bit 0 position. */
02681 #define ADC_INITDLY1_bm  (1<<6)  /* Initial Delay Selection bit 1 mask. */
02682 #define ADC_INITDLY1_bp  6  /* Initial Delay Selection bit 1 position. */
02683 #define ADC_INITDLY2_bm  (1<<7)  /* Initial Delay Selection bit 2 mask. */
02684 #define ADC_INITDLY2_bp  7  /* Initial Delay Selection bit 2 position. */
02685
02686 /* ADC.CTRLE  bit masks and bit positions */
02687 #define ADC_WINCM_gm  0x07  /* Window Comparator Mode group mask. */
02688 #define ADC_WINCM_gp  0  /* Window Comparator Mode group position. */
02689 #define ADC_WINCM0_bm  (1<<0)  /* Window Comparator Mode bit 0 mask. */
02690 #define ADC_WINCM0_bp  0  /* Window Comparator Mode bit 0 position. */
02691 #define ADC_WINCM1_bm  (1<<1)  /* Window Comparator Mode bit 1 mask. */
02692 #define ADC_WINCM1_bp  1  /* Window Comparator Mode bit 1 position. */
02693 #define ADC_WINCM2_bm  (1<<2)  /* Window Comparator Mode bit 2 mask. */
02694 #define ADC_WINCM2_bp  2  /* Window Comparator Mode bit 2 position. */
02695
02696 /* ADC.SAMPCTRL  bit masks and bit positions */
02697 #define ADC_SAMPLEN_gm  0x1F  /* Sample lenght group mask. */
02698 #define ADC_SAMPLEN_gp  0  /* Sample lenght group position. */
02699 #define ADC_SAMPLEN0_bm  (1<<0)  /* Sample lenght bit 0 mask. */
02700 #define ADC_SAMPLEN0_bp  0  /* Sample lenght bit 0 position. */
02701 #define ADC_SAMPLEN1_bm  (1<<1)  /* Sample lenght bit 1 mask. */
02702 #define ADC_SAMPLEN1_bp  1  /* Sample lenght bit 1 position. */
02703 #define ADC_SAMPLEN2_bm  (1<<2)  /* Sample lenght bit 2 mask. */
02704 #define ADC_SAMPLEN2_bp  2  /* Sample lenght bit 2 position. */
02705 #define ADC_SAMPLEN3_bm  (1<<3)  /* Sample lenght bit 3 mask. */
```

```
02706 #define ADC_SAMPLEN3_bp  3  /* Sample lenght bit 3 position. */
02707 #define ADC_SAMPLEN4_bm  (1<<4)  /* Sample lenght bit 4 mask. */
02708 #define ADC_SAMPLEN4_bp  4  /* Sample lenght bit 4 position. */
02709
02710 /* ADC.MUXPOS  bit masks and bit positions */
02711 #define ADC_MUXPOS_gm  0x1F  /* Analog Channel Selection Bits group mask. */
02712 #define ADC_MUXPOS_gp  0  /* Analog Channel Selection Bits group position. */
02713 #define ADC_MUXPOS0_bm  (1<<0)  /* Analog Channel Selection Bits bit 0 mask. */
02714 #define ADC_MUXPOS0_bp  0  /* Analog Channel Selection Bits bit 0 position. */
02715 #define ADC_MUXPOS1_bm  (1<<1)  /* Analog Channel Selection Bits bit 1 mask. */
02716 #define ADC_MUXPOS1_bp  1  /* Analog Channel Selection Bits bit 1 position. */
02717 #define ADC_MUXPOS2_bm  (1<<2)  /* Analog Channel Selection Bits bit 2 mask. */
02718 #define ADC_MUXPOS2_bp  2  /* Analog Channel Selection Bits bit 2 position. */
02719 #define ADC_MUXPOS3_bm  (1<<3)  /* Analog Channel Selection Bits bit 3 mask. */
02720 #define ADC_MUXPOS3_bp  3  /* Analog Channel Selection Bits bit 3 position. */
02721 #define ADC_MUXPOS4_bm  (1<<4)  /* Analog Channel Selection Bits bit 4 mask. */
02722 #define ADC_MUXPOS4_bp  4  /* Analog Channel Selection Bits bit 4 position. */
02723
02724 /* ADC.COMMAND  bit masks and bit positions */
02725 #define ADC_STCONV_bm  0x01  /* Start Conversion Operation bit mask. */
02726 #define ADC_STCONV_bp  0  /* Start Conversion Operation bit position. */
02727
02728 /* ADC.EVCTRL  bit masks and bit positions */
02729 #define ADC_STARTEI_bm  0x01  /* Start Event Input Enable bit mask. */
02730 #define ADC_STARTEI_bp  0  /* Start Event Input Enable bit position. */
02731
02732 /* ADC.INTCTRL  bit masks and bit positions */
02733 #define ADC_RESRDY_bm  0x01  /* Result Ready Interrupt Enable bit mask. */
02734 #define ADC_RESRDY_bp  0  /* Result Ready Interrupt Enable bit position. */
02735 #define ADC_WCMP_bm  0x02  /* Window Comparator Interrupt Enable bit mask. */
02736 #define ADC_WCMP_bp  1  /* Window Comparator Interrupt Enable bit position. */
02737
02738 /* ADC.INTFLAGS  bit masks and bit positions */
02739 /* ADC_RESRDY  is already defined. */
02740 /* ADC_WCMP  is already defined. */
02741
02742 /* ADC.DBGCTRL  bit masks and bit positions */
02743 #define ADC_DBGRUN_bm  0x01  /* Debug run bit mask. */
02744 #define ADC_DBGRUN_bp  0  /* Debug run bit position. */
02745
02746 /* ADC.TEMP  bit masks and bit positions */
02747 #define ADC_TEMP_gm  0xFF  /* Temporary group mask. */
02748 #define ADC_TEMP_gp  0  /* Temporary group position. */
02749 #define ADC_TEMP0_bm  (1<<0)  /* Temporary bit 0 mask. */
02750 #define ADC_TEMP0_bp  0  /* Temporary bit 0 position. */
02751 #define ADC_TEMP1_bm  (1<<1)  /* Temporary bit 1 mask. */
02752 #define ADC_TEMP1_bp  1  /* Temporary bit 1 position. */
02753 #define ADC_TEMP2_bm  (1<<2)  /* Temporary bit 2 mask. */
02754 #define ADC_TEMP2_bp  2  /* Temporary bit 2 position. */
02755 #define ADC_TEMP3_bm  (1<<3)  /* Temporary bit 3 mask. */
02756 #define ADC_TEMP3_bp  3  /* Temporary bit 3 position. */
02757 #define ADC_TEMP4_bm  (1<<4)  /* Temporary bit 4 mask. */
02758 #define ADC_TEMP4_bp  4  /* Temporary bit 4 position. */
02759 #define ADC_TEMP5_bm  (1<<5)  /* Temporary bit 5 mask. */
02760 #define ADC_TEMP5_bp  5  /* Temporary bit 5 position. */
02761 #define ADC_TEMP6_bm  (1<<6)  /* Temporary bit 6 mask. */
02762 #define ADC_TEMP6_bp  6  /* Temporary bit 6 position. */
02763 #define ADC_TEMP7_bm  (1<<7)  /* Temporary bit 7 mask. */
02764 #define ADC_TEMP7_bp  7  /* Temporary bit 7 position. */
02765
02766 /* ADC.CALIB  bit masks and bit positions */
02767 #define ADC_DUTYCYC_bm  0x01  /* Duty Cycle bit mask. */
02768 #define ADC_DUTYCYC_bp  0  /* Duty Cycle bit position. */
02769
02770 /* BOD - Bod interface */
02771 /* BOD.CTRLA  bit masks and bit positions */
02772 #define BOD_SLEEP_gm  0x03  /* Operation in sleep mode group mask. */
02773 #define BOD_SLEEP_gp  0  /* Operation in sleep mode group position. */
02774 #define BOD_SLEEP0_bm  (1<<0)  /* Operation in sleep mode bit 0 mask. */
02775 #define BOD_SLEEP0_bp  0  /* Operation in sleep mode bit 0 position. */
02776 #define BOD_SLEEP1_bm  (1<<1)  /* Operation in sleep mode bit 1 mask. */
02777 #define BOD_SLEEP1_bp  1  /* Operation in sleep mode bit 1 position. */
02778 #define BOD_ACTIVE_gm  0x0C  /* Operation in active mode group mask. */
02779 #define BOD_ACTIVE_gp  2  /* Operation in active mode group position. */
02780 #define BOD_ACTIVE0_bm  (1<<2)  /* Operation in active mode bit 0 mask. */
02781 #define BOD_ACTIVE0_bp  2  /* Operation in active mode bit 0 position. */
02782 #define BOD_ACTIVE1_bm  (1<<3)  /* Operation in active mode bit 1 mask. */
02783 #define BOD_ACTIVE1_bp  3  /* Operation in active mode bit 1 position. */
02784 #define BOD_SAMPFREQ_bm  0x10  /* Sample frequency bit mask. */
02785 #define BOD_SAMPFREQ_bp  4  /* Sample frequency bit position. */
02786
02787 /* BOD.CTRLB  bit masks and bit positions */
02788 #define BOD_LVL_gm  0x07  /* Bod level group mask. */
02789 #define BOD_LVL_gp  0  /* Bod level group position. */
02790 #define BOD_LVL0_bm  (1<<0)  /* Bod level bit 0 mask. */
02791 #define BOD_LVL0_bp  0  /* Bod level bit 0 position. */
02792 #define BOD_LVL1_bm  (1<<1)  /* Bod level bit 1 mask. */
```

```
02793 #define BOD_LVL1_bp  1  /* Bod level bit 1 position. */
02794 #define BOD_LVL2_bm  (1<<2)  /* Bod level bit 2 mask. */
02795 #define BOD_LVL2_bp  2  /* Bod level bit 2 position. */
02796
02797 /* BOD.VLMCTRLA  bit masks and bit positions */
02798 #define BOD_VLMLVL_gm  0x03  /* voltage level monitor level group mask. */
02799 #define BOD_VLMLVL_gp  0  /* voltage level monitor level group position. */
02800 #define BOD_VLMLVL0_bm  (1<<0)  /* voltage level monitor level bit 0 mask. */
02801 #define BOD_VLMLVL0_bp  0  /* voltage level monitor level bit 0 position. */
02802 #define BOD_VLMLVL1_bm  (1<<1)  /* voltage level monitor level bit 1 mask. */
02803 #define BOD_VLMLVL1_bp  1  /* voltage level monitor level bit 1 position. */
02804
02805 /* BOD.INTCTRL  bit masks and bit positions */
02806 #define BOD_VLMIE_bm  0x01  /* voltage level monitor interrrupt enable bit mask. */
02807 #define BOD_VLMIE_bp  0  /* voltage level monitor interrrupt enable bit position. */
02808 #define BOD_VLMCFG_gm  0x06  /* Configuration group mask. */
02809 #define BOD_VLMCFG_gp  1  /* Configuration group position. */
02810 #define BOD_VLMCFG0_bm  (1<<1)  /* Configuration bit 0 mask. */
02811 #define BOD_VLMCFG0_bp  1  /* Configuration bit 0 position. */
02812 #define BOD_VLMCFG1_bm  (1<<2)  /* Configuration bit 1 mask. */
02813 #define BOD_VLMCFG1_bp  2  /* Configuration bit 1 position. */
02814
02815 /* BOD.INTFLAGS  bit masks and bit positions */
02816 #define BOD_VLMIF_bm  0x01  /* Voltage level monitor interrupt flag bit mask. */
02817 #define BOD_VLMIF_bp  0  /* Voltage level monitor interrupt flag bit position. */
02818
02819 /* BOD.STATUS  bit masks and bit positions */
02820 #define BOD_VLMS_bm  0x01  /* Voltage level monitor status bit mask. */
02821 #define BOD_VLMS_bp  0  /* Voltage level monitor status bit position. */
02822
02823 /* CCL - Configurable Custom Logic */
02824 /* CCL.CTRLA  bit masks and bit positions */
02825 #define CCL_ENABLE_bm  0x01  /* Enable bit mask. */
02826 #define CCL_ENABLE_bp  0  /* Enable bit position. */
02827 #define CCL_RUNSTDBY_bm  0x40  /* Run in Standby bit mask. */
02828 #define CCL_RUNSTDBY_bp  6  /* Run in Standby bit position. */
02829
02830 /* CCL.SEQCTRL0  bit masks and bit positions */
02831 #define CCL_SEQSEL_gm  0x07  /* Sequential Selection group mask. */
02832 #define CCL_SEQSEL_gp  0  /* Sequential Selection group position. */
02833 #define CCL_SEQSEL0_bm  (1<<0)  /* Sequential Selection bit 0 mask. */
02834 #define CCL_SEQSEL0_bp  0  /* Sequential Selection bit 0 position. */
02835 #define CCL_SEQSEL1_bm  (1<<1)  /* Sequential Selection bit 1 mask. */
02836 #define CCL_SEQSEL1_bp  1  /* Sequential Selection bit 1 position. */
02837 #define CCL_SEQSEL2_bm  (1<<2)  /* Sequential Selection bit 2 mask. */
02838 #define CCL_SEQSEL2_bp  2  /* Sequential Selection bit 2 position. */
02839
02840 /* CCL.LUT0CTRLA  bit masks and bit positions */
02841 /* CCL_ENABLE  is already defined. */
02842 #define CCL_OUTEN_bm  0x08  /* Output Enable bit mask. */
02843 #define CCL_OUTEN_bp  3  /* Output Enable bit position. */
02844 #define CCL_FILTSEL_gm  0x30  /* Filter Selection group mask. */
02845 #define CCL_FILTSEL_gp  4  /* Filter Selection group position. */
02846 #define CCL_FILTSEL0_bm  (1<<4)  /* Filter Selection bit 0 mask. */
02847 #define CCL_FILTSEL0_bp  4  /* Filter Selection bit 0 position. */
02848 #define CCL_FILTSEL1_bm  (1<<5)  /* Filter Selection bit 1 mask. */
02849 #define CCL_FILTSEL1_bp  5  /* Filter Selection bit 1 position. */
02850 #define CCL_CLKSRC_bm  0x40  /* Clock Source Selection bit mask. */
02851 #define CCL_CLKSRC_bp  6  /* Clock Source Selection bit position. */
02852 #define CCL_EDGEDET_bm  0x80  /* Edge Detection Enable bit mask. */
02853 #define CCL_EDGEDET_bp  7  /* Edge Detection Enable bit position. */
02854
02855 /* CCL.LUT0CTRLB  bit masks and bit positions */
02856 #define CCL_INSEL0_gm  0x0F  /* LUT Input 0 Source Selection group mask. */
02857 #define CCL_INSEL0_gp  0  /* LUT Input 0 Source Selection group position. */
02858 #define CCL_INSEL00_bm  (1<<0)  /* LUT Input 0 Source Selection bit 0 mask. */
02859 #define CCL_INSEL00_bp  0  /* LUT Input 0 Source Selection bit 0 position. */
02860 #define CCL_INSEL01_bm  (1<<1)  /* LUT Input 0 Source Selection bit 1 mask. */
02861 #define CCL_INSEL01_bp  1  /* LUT Input 0 Source Selection bit 1 position. */
02862 #define CCL_INSEL02_bm  (1<<2)  /* LUT Input 0 Source Selection bit 2 mask. */
02863 #define CCL_INSEL02_bp  2  /* LUT Input 0 Source Selection bit 2 position. */
02864 #define CCL_INSEL03_bm  (1<<3)  /* LUT Input 0 Source Selection bit 3 mask. */
02865 #define CCL_INSEL03_bp  3  /* LUT Input 0 Source Selection bit 3 position. */
02866 #define CCL_INSEL1_gm  0xF0  /* LUT Input 1 Source Selection group mask. */
02867 #define CCL_INSEL1_gp  4  /* LUT Input 1 Source Selection group position. */
02868 #define CCL_INSEL10_bm  (1<<4)  /* LUT Input 1 Source Selection bit 0 mask. */
02869 #define CCL_INSEL10_bp  4  /* LUT Input 1 Source Selection bit 0 position. */
02870 #define CCL_INSEL11_bm  (1<<5)  /* LUT Input 1 Source Selection bit 1 mask. */
02871 #define CCL_INSEL11_bp  5  /* LUT Input 1 Source Selection bit 1 position. */
02872 #define CCL_INSEL12_bm  (1<<6)  /* LUT Input 1 Source Selection bit 2 mask. */
02873 #define CCL_INSEL12_bp  6  /* LUT Input 1 Source Selection bit 2 position. */
02874 #define CCL_INSEL13_bm  (1<<7)  /* LUT Input 1 Source Selection bit 3 mask. */
02875 #define CCL_INSEL13_bp  7  /* LUT Input 1 Source Selection bit 3 position. */
02876
02877 /* CCL.LUT0CTRLC  bit masks and bit positions */
02878 #define CCL_INSEL2_gm  0x0F  /* LUT Input 2 Source Selection group mask. */
02879 #define CCL_INSEL2_gp  0  /* LUT Input 2 Source Selection group position. */
```

```
02880 #define CCL_INSEL20_bm  (1«0)  /* LUT Input 2 Source Selection bit 0 mask. */
02881 #define CCL_INSEL20_bp  0  /* LUT Input 2 Source Selection bit 0 position. */
02882 #define CCL_INSEL21_bm  (1«1)  /* LUT Input 2 Source Selection bit 1 mask. */
02883 #define CCL_INSEL21_bp  1  /* LUT Input 2 Source Selection bit 1 position. */
02884 #define CCL_INSEL22_bm  (1«2)  /* LUT Input 2 Source Selection bit 2 mask. */
02885 #define CCL_INSEL22_bp  2  /* LUT Input 2 Source Selection bit 2 position. */
02886 #define CCL_INSEL23_bm  (1«3)  /* LUT Input 2 Source Selection bit 3 mask. */
02887 #define CCL_INSEL23_bp  3  /* LUT Input 2 Source Selection bit 3 position. */
02888
02889 /* CCL.LUT1CTRLA  bit masks and bit positions */
02890 /* CCL_ENABLE  is already defined. */
02891 /* CCL_OUTEN  is already defined. */
02892 /* CCL_FILTSEL  is already defined. */
02893 /* CCL_CLKSRC  is already defined. */
02894 /* CCL_EDGEDET  is already defined. */
02895
02896 /* CCL.LUT1CTRLB  bit masks and bit positions */
02897 /* CCL_INSEL0  is already defined. */
02898 /* CCL_INSEL1  is already defined. */
02899
02900 /* CCL.LUT1CTRLC  bit masks and bit positions */
02901 /* CCL_INSEL2  is already defined. */
02902
02903 /* CLKCTRL - Clock controller */
02904 /* CLKCTRL.MCLKCTRLA  bit masks and bit positions */
02905 #define CLKCTRL_CLKSEL_gm  0x03  /* clock select group mask. */
02906 #define CLKCTRL_CLKSEL_gp  0  /* clock select group position. */
02907 #define CLKCTRL_CLKSEL0_bm  (1«0)  /* clock select bit 0 mask. */
02908 #define CLKCTRL_CLKSEL0_bp  0  /* clock select bit 0 position. */
02909 #define CLKCTRL_CLKSEL1_bm  (1«1)  /* clock select bit 1 mask. */
02910 #define CLKCTRL_CLKSEL1_bp  1  /* clock select bit 1 position. */
02911 #define CLKCTRL_CLKOUT_bm  0x80  /* System clock out bit mask. */
02912 #define CLKCTRL_CLKOUT_bp  7  /* System clock out bit position. */
02913
02914 /* CLKCTRL.MCLKCTRLB  bit masks and bit positions */
02915 #define CLKCTRL_PEN_bm  0x01  /* Prescaler enable bit mask. */
02916 #define CLKCTRL_PEN_bp  0  /* Prescaler enable bit position. */
02917 #define CLKCTRL_PDIV_gm  0x1E  /* Prescaler division group mask. */
02918 #define CLKCTRL_PDIV_gp  1  /* Prescaler division group position. */
02919 #define CLKCTRL_PDIV0_bm  (1«1)  /* Prescaler division bit 0 mask. */
02920 #define CLKCTRL_PDIV0_bp  1  /* Prescaler division bit 0 position. */
02921 #define CLKCTRL_PDIV1_bm  (1«2)  /* Prescaler division bit 1 mask. */
02922 #define CLKCTRL_PDIV1_bp  2  /* Prescaler division bit 1 position. */
02923 #define CLKCTRL_PDIV2_bm  (1«3)  /* Prescaler division bit 2 mask. */
02924 #define CLKCTRL_PDIV2_bp  3  /* Prescaler division bit 2 position. */
02925 #define CLKCTRL_PDIV3_bm  (1«4)  /* Prescaler division bit 3 mask. */
02926 #define CLKCTRL_PDIV3_bp  4  /* Prescaler division bit 3 position. */
02927
02928 /* CLKCTRL.MCLKLOCK  bit masks and bit positions */
02929 #define CLKCTRL_LOCKEN_bm  0x01  /* lock ebable bit mask. */
02930 #define CLKCTRL_LOCKEN_bp  0  /* lock ebable bit position. */
02931
02932 /* CLKCTRL.MCLKSTATUS  bit masks and bit positions */
02933 #define CLKCTRL_SOSC_bm  0x01  /* System Oscillator changing bit mask. */
02934 #define CLKCTRL_SOSC_bp  0  /* System Oscillator changing bit position. */
02935 #define CLKCTRL_OSC20MS_bm  0x10  /* 20MHz oscillator status bit mask. */
02936 #define CLKCTRL_OSC20MS_bp  4  /* 20MHz oscillator status bit position. */
02937 #define CLKCTRL_OSC32KS_bm  0x20  /* 32KHz oscillator status bit mask. */
02938 #define CLKCTRL_OSC32KS_bp  5  /* 32KHz oscillator status bit position. */
02939 #define CLKCTRL_XOSC32KS_bm  0x40  /* 32.768 kHz Crystal Oscillator status bit mask. */
02940 #define CLKCTRL_XOSC32KS_bp  6  /* 32.768 kHz Crystal Oscillator status bit position. */
02941 #define CLKCTRL_EXTS_bm  0x80  /* External Clock status bit mask. */
02942 #define CLKCTRL_EXTS_bp  7  /* External Clock status bit position. */
02943
02944 /* CLKCTRL.OSC20MCTRLA  bit masks and bit positions */
02945 #define CLKCTRL_RUNSTDBY_bm  0x02  /* Run standby bit mask. */
02946 #define CLKCTRL_RUNSTDBY_bp  1  /* Run standby bit position. */
02947
02948 /* CLKCTRL.OSC20MCALIBA  bit masks and bit positions */
02949 #define CLKCTRL_CAL20M_gm  0x3F  /* Calibration group mask. */
02950 #define CLKCTRL_CAL20M_gp  0  /* Calibration group position. */
02951 #define CLKCTRL_CAL20M0_bm  (1«0)  /* Calibration bit 0 mask. */
02952 #define CLKCTRL_CAL20M0_bp  0  /* Calibration bit 0 position. */
02953 #define CLKCTRL_CAL20M1_bm  (1«1)  /* Calibration bit 1 mask. */
02954 #define CLKCTRL_CAL20M1_bp  1  /* Calibration bit 1 position. */
02955 #define CLKCTRL_CAL20M2_bm  (1«2)  /* Calibration bit 2 mask. */
02956 #define CLKCTRL_CAL20M2_bp  2  /* Calibration bit 2 position. */
02957 #define CLKCTRL_CAL20M3_bm  (1«3)  /* Calibration bit 3 mask. */
02958 #define CLKCTRL_CAL20M3_bp  3  /* Calibration bit 3 position. */
02959 #define CLKCTRL_CAL20M4_bm  (1«4)  /* Calibration bit 4 mask. */
02960 #define CLKCTRL_CAL20M4_bp  4  /* Calibration bit 4 position. */
02961 #define CLKCTRL_CAL20M5_bm  (1«5)  /* Calibration bit 5 mask. */
02962 #define CLKCTRL_CAL20M5_bp  5  /* Calibration bit 5 position. */
02963
02964 /* CLKCTRL.OSC20MCALIBB  bit masks and bit positions */
02965 #define CLKCTRL_TEMPCAL20M_gm  0x0F  /* Oscillator temperature coefficient group mask. */
02966 #define CLKCTRL_TEMPCAL20M_gp  0  /* Oscillator temperature coefficient group position. */
```

```
02967 #define CLKCTRL_TEMPCAL20M0_bm  (1<<0)  /* Oscillator temperature coefficient bit 0 mask. */
02968 #define CLKCTRL_TEMPCAL20M0_bp  0  /* Oscillator temperature coefficient bit 0 position. */
02969 #define CLKCTRL_TEMPCAL20M1_bm  (1<<1)  /* Oscillator temperature coefficient bit 1 mask. */
02970 #define CLKCTRL_TEMPCAL20M1_bp  1  /* Oscillator temperature coefficient bit 1 position. */
02971 #define CLKCTRL_TEMPCAL20M2_bm  (1<<2)  /* Oscillator temperature coefficient bit 2 mask. */
02972 #define CLKCTRL_TEMPCAL20M2_bp  2  /* Oscillator temperature coefficient bit 2 position. */
02973 #define CLKCTRL_TEMPCAL20M3_bm  (1<<3)  /* Oscillator temperature coefficient bit 3 mask. */
02974 #define CLKCTRL_TEMPCAL20M3_bp  3  /* Oscillator temperature coefficient bit 3 position. */
02975 #define CLKCTRL_LOCK_bm  0x80  /* Lock bit mask. */
02976 #define CLKCTRL_LOCK_bp  7  /* Lock bit position. */
02977
02978 /* CLKCTRL.OSC32KCTRLA  bit masks and bit positions */
02979 /* CLKCTRL_RUNSTDBY  is already defined. */
02980
02981 /* CPU - CPU */
02982 /* CPU.CCP  bit masks and bit positions */
02983 #define CPU_CCP_gm  0xFF  /* CCP signature group mask. */
02984 #define CPU_CCP_gp  0  /* CCP signature group position. */
02985 #define CPU_CCP0_bm  (1<<0)  /* CCP signature bit 0 mask. */
02986 #define CPU_CCP0_bp  0  /* CCP signature bit 0 position. */
02987 #define CPU_CCP1_bm  (1<<1)  /* CCP signature bit 1 mask. */
02988 #define CPU_CCP1_bp  1  /* CCP signature bit 1 position. */
02989 #define CPU_CCP2_bm  (1<<2)  /* CCP signature bit 2 mask. */
02990 #define CPU_CCP2_bp  2  /* CCP signature bit 2 position. */
02991 #define CPU_CCP3_bm  (1<<3)  /* CCP signature bit 3 mask. */
02992 #define CPU_CCP3_bp  3  /* CCP signature bit 3 position. */
02993 #define CPU_CCP4_bm  (1<<4)  /* CCP signature bit 4 mask. */
02994 #define CPU_CCP4_bp  4  /* CCP signature bit 4 position. */
02995 #define CPU_CCP5_bm  (1<<5)  /* CCP signature bit 5 mask. */
02996 #define CPU_CCP5_bp  5  /* CCP signature bit 5 position. */
02997 #define CPU_CCP6_bm  (1<<6)  /* CCP signature bit 6 mask. */
02998 #define CPU_CCP6_bp  6  /* CCP signature bit 6 position. */
02999 #define CPU_CCP7_bm  (1<<7)  /* CCP signature bit 7 mask. */
03000 #define CPU_CCP7_bp  7  /* CCP signature bit 7 position. */
03001
03002 /* CPU.SREG  bit masks and bit positions */
03003 #define CPU_C_bm  0x01  /* Carry Flag bit mask. */
03004 #define CPU_C_bp  0  /* Carry Flag bit position. */
03005 #define CPU_Z_bm  0x02  /* Zero Flag bit mask. */
03006 #define CPU_Z_bp  1  /* Zero Flag bit position. */
03007 #define CPU_N_bm  0x04  /* Negative Flag bit mask. */
03008 #define CPU_N_bp  2  /* Negative Flag bit position. */
03009 #define CPU_V_bm  0x08  /* Two's Complement Overflow Flag bit mask. */
03010 #define CPU_V_bp  3  /* Two's Complement Overflow Flag bit position. */
03011 #define CPU_S_bm  0x10  /* N Exclusive Or V Flag bit mask. */
03012 #define CPU_S_bp  4  /* N Exclusive Or V Flag bit position. */
03013 #define CPU_H_bm  0x20  /* Half Carry Flag bit mask. */
03014 #define CPU_H_bp  5  /* Half Carry Flag bit position. */
03015 #define CPU_T_bm  0x40  /* Transfer Bit bit mask. */
03016 #define CPU_T_bp  6  /* Transfer Bit bit position. */
03017 #define CPU_I_bm  0x80  /* Global Interrupt Enable Flag bit mask. */
03018 #define CPU_I_bp  7  /* Global Interrupt Enable Flag bit position. */
03019
03020 /* CPUINT - Interrupt Controller */
03021 /* CPUINT.CTRLA  bit masks and bit positions */
03022 #define CPUINT_LVL0RR_bm  0x01  /* Round-robin Scheduling Enable bit mask. */
03023 #define CPUINT_LVL0RR_bp  0  /* Round-robin Scheduling Enable bit position. */
03024 #define CPUINT_CVT_bm  0x20  /* Compact Vector Table bit mask. */
03025 #define CPUINT_CVT_bp  5  /* Compact Vector Table bit position. */
03026 #define CPUINT_IVSEL_bm  0x40  /* Interrupt Vector Select bit mask. */
03027 #define CPUINT_IVSEL_bp  6  /* Interrupt Vector Select bit position. */
03028
03029 /* CPUINT.STATUS  bit masks and bit positions */
03030 #define CPUINT_LVL0EX_bm  0x01  /* Level 0 Interrupt Executing bit mask. */
03031 #define CPUINT_LVL0EX_bp  0  /* Level 0 Interrupt Executing bit position. */
03032 #define CPUINT_LVL1EX_bm  0x02  /* Level 1 Interrupt Executing bit mask. */
03033 #define CPUINT_LVL1EX_bp  1  /* Level 1 Interrupt Executing bit position. */
03034 #define CPUINT_NMIEX_bm  0x80  /* Non-maskable Interrupt Executing bit mask. */
03035 #define CPUINT_NMIEX_bp  7  /* Non-maskable Interrupt Executing bit position. */
03036
03037 /* CPUINT.LVL0PRI  bit masks and bit positions */
03038 #define CPUINT_LVL0PRI_gm  0xFF  /* Interrupt Level Priority group mask. */
03039 #define CPUINT_LVL0PRI_gp  0  /* Interrupt Level Priority group position. */
03040 #define CPUINT_LVL0PRI0_bm  (1<<0)  /* Interrupt Level Priority bit 0 mask. */
03041 #define CPUINT_LVL0PRI0_bp  0  /* Interrupt Level Priority bit 0 position. */
03042 #define CPUINT_LVL0PRI1_bm  (1<<1)  /* Interrupt Level Priority bit 1 mask. */
03043 #define CPUINT_LVL0PRI1_bp  1  /* Interrupt Level Priority bit 1 position. */
03044 #define CPUINT_LVL0PRI2_bm  (1<<2)  /* Interrupt Level Priority bit 2 mask. */
03045 #define CPUINT_LVL0PRI2_bp  2  /* Interrupt Level Priority bit 2 position. */
03046 #define CPUINT_LVL0PRI3_bm  (1<<3)  /* Interrupt Level Priority bit 3 mask. */
03047 #define CPUINT_LVL0PRI3_bp  3  /* Interrupt Level Priority bit 3 position. */
03048 #define CPUINT_LVL0PRI4_bm  (1<<4)  /* Interrupt Level Priority bit 4 mask. */
03049 #define CPUINT_LVL0PRI4_bp  4  /* Interrupt Level Priority bit 4 position. */
03050 #define CPUINT_LVL0PRI5_bm  (1<<5)  /* Interrupt Level Priority bit 5 mask. */
03051 #define CPUINT_LVL0PRI5_bp  5  /* Interrupt Level Priority bit 5 position. */
03052 #define CPUINT_LVL0PRI6_bm  (1<<6)  /* Interrupt Level Priority bit 6 mask. */
03053 #define CPUINT_LVL0PRI6_bp  6  /* Interrupt Level Priority bit 6 position. */
```

```
03054 #define CPUINT_LVL0PRI7_bm  (1«7)  /* Interrupt Level Priority bit 7 mask. */
03055 #define CPUINT_LVL0PRI7_bp  7  /* Interrupt Level Priority bit 7 position. */
03056
03057 /* CPUINT.LVL1VEC  bit masks and bit positions */
03058 #define CPUINT_LVL1VEC_gm  0xFF  /* Interrupt Vector with High Priority group mask. */
03059 #define CPUINT_LVL1VEC_gp  0  /* Interrupt Vector with High Priority group position. */
03060 #define CPUINT_LVL1VEC0_bm  (1«0)  /* Interrupt Vector with High Priority bit 0 mask. */
03061 #define CPUINT_LVL1VEC0_bp  0  /* Interrupt Vector with High Priority bit 0 position. */
03062 #define CPUINT_LVL1VEC1_bm  (1«1)  /* Interrupt Vector with High Priority bit 1 mask. */
03063 #define CPUINT_LVL1VEC1_bp  1  /* Interrupt Vector with High Priority bit 1 position. */
03064 #define CPUINT_LVL1VEC2_bm  (1«2)  /* Interrupt Vector with High Priority bit 2 mask. */
03065 #define CPUINT_LVL1VEC2_bp  2  /* Interrupt Vector with High Priority bit 2 position. */
03066 #define CPUINT_LVL1VEC3_bm  (1«3)  /* Interrupt Vector with High Priority bit 3 mask. */
03067 #define CPUINT_LVL1VEC3_bp  3  /* Interrupt Vector with High Priority bit 3 position. */
03068 #define CPUINT_LVL1VEC4_bm  (1«4)  /* Interrupt Vector with High Priority bit 4 mask. */
03069 #define CPUINT_LVL1VEC4_bp  4  /* Interrupt Vector with High Priority bit 4 position. */
03070 #define CPUINT_LVL1VEC5_bm  (1«5)  /* Interrupt Vector with High Priority bit 5 mask. */
03071 #define CPUINT_LVL1VEC5_bp  5  /* Interrupt Vector with High Priority bit 5 position. */
03072 #define CPUINT_LVL1VEC6_bm  (1«6)  /* Interrupt Vector with High Priority bit 6 mask. */
03073 #define CPUINT_LVL1VEC6_bp  6  /* Interrupt Vector with High Priority bit 6 position. */
03074 #define CPUINT_LVL1VEC7_bm  (1«7)  /* Interrupt Vector with High Priority bit 7 mask. */
03075 #define CPUINT_LVL1VEC7_bp  7  /* Interrupt Vector with High Priority bit 7 position. */
03076
03077 /* CRCSCAN - CRCSCAN */
03078 /* CRCSCAN.CTRLA  bit masks and bit positions */
03079 #define CRCSCAN_ENABLE_bm  0x01  /* Enable CRC scan bit mask. */
03080 #define CRCSCAN_ENABLE_bp  0  /* Enable CRC scan bit position. */
03081 #define CRCSCAN_NMIEN_bm  0x02  /* Enable NMI Trigger bit mask. */
03082 #define CRCSCAN_NMIEN_bp  1  /* Enable NMI Trigger bit position. */
03083 #define CRCSCAN_RESET_bm  0x80  /* Reset CRC scan bit mask. */
03084 #define CRCSCAN_RESET_bp  7  /* Reset CRC scan bit position. */
03085
03086 /* CRCSCAN.CTRLB  bit masks and bit positions */
03087 #define CRCSCAN_SRC_gm  0x03  /* CRC Source group mask. */
03088 #define CRCSCAN_SRC_gp  0  /* CRC Source group position. */
03089 #define CRCSCAN_SRC0_bm  (1«0)  /* CRC Source bit 0 mask. */
03090 #define CRCSCAN_SRC0_bp  0  /* CRC Source bit 0 position. */
03091 #define CRCSCAN_SRC1_bm  (1«1)  /* CRC Source bit 1 mask. */
03092 #define CRCSCAN_SRC1_bp  1  /* CRC Source bit 1 position. */
03093 #define CRCSCAN_MODE_gm  0x30  /* CRC Flash Access Mode group mask. */
03094 #define CRCSCAN_MODE_gp  4  /* CRC Flash Access Mode group position. */
03095 #define CRCSCAN_MODE0_bm  (1«4)  /* CRC Flash Access Mode bit 0 mask. */
03096 #define CRCSCAN_MODE0_bp  4  /* CRC Flash Access Mode bit 0 position. */
03097 #define CRCSCAN_MODE1_bm  (1«5)  /* CRC Flash Access Mode bit 1 mask. */
03098 #define CRCSCAN_MODE1_bp  5  /* CRC Flash Access Mode bit 1 position. */
03099
03100 /* CRCSCAN.STATUS  bit masks and bit positions */
03101 #define CRCSCAN_BUSY_bm  0x01  /* CRC Busy bit mask. */
03102 #define CRCSCAN_BUSY_bp  0  /* CRC Busy bit position. */
03103 #define CRCSCAN_OK_bm  0x02  /* CRC Ok bit mask. */
03104 #define CRCSCAN_OK_bp  1  /* CRC Ok bit position. */
03105
03106 /* EVSYS - Event System */
03107 /* EVSYS.ASYNCCH0  bit masks and bit positions */
03108 #define EVSYS_ASYNCCH0_gm  0xFF  /* Asynchronous Channel 0 Generator Selection group mask. */
03109 #define EVSYS_ASYNCCH0_gp  0  /* Asynchronous Channel 0 Generator Selection group position. */
03110 #define EVSYS_ASYNCCH00_bm  (1«0)  /* Asynchronous Channel 0 Generator Selection bit 0 mask. */
03111 #define EVSYS_ASYNCCH00_bp  0  /* Asynchronous Channel 0 Generator Selection bit 0 position. */
03112 #define EVSYS_ASYNCCH01_bm  (1«1)  /* Asynchronous Channel 0 Generator Selection bit 1 mask. */
03113 #define EVSYS_ASYNCCH01_bp  1  /* Asynchronous Channel 0 Generator Selection bit 1 position. */
03114 #define EVSYS_ASYNCCH02_bm  (1«2)  /* Asynchronous Channel 0 Generator Selection bit 2 mask. */
03115 #define EVSYS_ASYNCCH02_bp  2  /* Asynchronous Channel 0 Generator Selection bit 2 position. */
03116 #define EVSYS_ASYNCCH03_bm  (1«3)  /* Asynchronous Channel 0 Generator Selection bit 3 mask. */
03117 #define EVSYS_ASYNCCH03_bp  3  /* Asynchronous Channel 0 Generator Selection bit 3 position. */
03118 #define EVSYS_ASYNCCH04_bm  (1«4)  /* Asynchronous Channel 0 Generator Selection bit 4 mask. */
03119 #define EVSYS_ASYNCCH04_bp  4  /* Asynchronous Channel 0 Generator Selection bit 4 position. */
03120 #define EVSYS_ASYNCCH05_bm  (1«5)  /* Asynchronous Channel 0 Generator Selection bit 5 mask. */
03121 #define EVSYS_ASYNCCH05_bp  5  /* Asynchronous Channel 0 Generator Selection bit 5 position. */
03122 #define EVSYS_ASYNCCH06_bm  (1«6)  /* Asynchronous Channel 0 Generator Selection bit 6 mask. */
03123 #define EVSYS_ASYNCCH06_bp  6  /* Asynchronous Channel 0 Generator Selection bit 6 position. */
03124 #define EVSYS_ASYNCCH07_bm  (1«7)  /* Asynchronous Channel 0 Generator Selection bit 7 mask. */
03125 #define EVSYS_ASYNCCH07_bp  7  /* Asynchronous Channel 0 Generator Selection bit 7 position. */
03126
03127 /* EVSYS.ASYNCCH1  bit masks and bit positions */
03128 #define EVSYS_ASYNCCH1_gm  0xFF  /* Asynchronous Channel 1 Generator Selection group mask. */
03129 #define EVSYS_ASYNCCH1_gp  0  /* Asynchronous Channel 1 Generator Selection group position. */
03130 #define EVSYS_ASYNCCH10_bm  (1«0)  /* Asynchronous Channel 1 Generator Selection bit 0 mask. */
03131 #define EVSYS_ASYNCCH10_bp  0  /* Asynchronous Channel 1 Generator Selection bit 0 position. */
03132 #define EVSYS_ASYNCCH11_bm  (1«1)  /* Asynchronous Channel 1 Generator Selection bit 1 mask. */
03133 #define EVSYS_ASYNCCH11_bp  1  /* Asynchronous Channel 1 Generator Selection bit 1 position. */
03134 #define EVSYS_ASYNCCH12_bm  (1«2)  /* Asynchronous Channel 1 Generator Selection bit 2 mask. */
03135 #define EVSYS_ASYNCCH12_bp  2  /* Asynchronous Channel 1 Generator Selection bit 2 position. */
03136 #define EVSYS_ASYNCCH13_bm  (1«3)  /* Asynchronous Channel 1 Generator Selection bit 3 mask. */
03137 #define EVSYS_ASYNCCH13_bp  3  /* Asynchronous Channel 1 Generator Selection bit 3 position. */
03138 #define EVSYS_ASYNCCH14_bm  (1«4)  /* Asynchronous Channel 1 Generator Selection bit 4 mask. */
03139 #define EVSYS_ASYNCCH14_bp  4  /* Asynchronous Channel 1 Generator Selection bit 4 position. */
03140 #define EVSYS_ASYNCCH15_bm  (1«5)  /* Asynchronous Channel 1 Generator Selection bit 5 mask. */
```

```
03141 #define EVSYS_ASYNCCH15_bp  5  /* Asynchronous Channel 1 Generator Selection bit 5 position. */
03142 #define EVSYS_ASYNCCH16_bm  (1<<6)  /* Asynchronous Channel 1 Generator Selection bit 6 mask. */
03143 #define EVSYS_ASYNCCH16_bp  6  /* Asynchronous Channel 1 Generator Selection bit 6 position. */
03144 #define EVSYS_ASYNCCH17_bm  (1<<7)  /* Asynchronous Channel 1 Generator Selection bit 7 mask. */
03145 #define EVSYS_ASYNCCH17_bp  7  /* Asynchronous Channel 1 Generator Selection bit 7 position. */
03146
03147 /* EVSYS.SYNCCH0  bit masks and bit positions */
03148 #define EVSYS_SYNCCH0_gm  0xFF  /* Synchronous Channel 0 Generator Selection group mask. */
03149 #define EVSYS_SYNCCH0_gp  0  /* Synchronous Channel 0 Generator Selection group position. */
03150 #define EVSYS_SYNCCH00_bm  (1<<0)  /* Synchronous Channel 0 Generator Selection bit 0 mask. */
03151 #define EVSYS_SYNCCH00_bp  0  /* Synchronous Channel 0 Generator Selection bit 0 position. */
03152 #define EVSYS_SYNCCH01_bm  (1<<1)  /* Synchronous Channel 0 Generator Selection bit 1 mask. */
03153 #define EVSYS_SYNCCH01_bp  1  /* Synchronous Channel 0 Generator Selection bit 1 position. */
03154 #define EVSYS_SYNCCH02_bm  (1<<2)  /* Synchronous Channel 0 Generator Selection bit 2 mask. */
03155 #define EVSYS_SYNCCH02_bp  2  /* Synchronous Channel 0 Generator Selection bit 2 position. */
03156 #define EVSYS_SYNCCH03_bm  (1<<3)  /* Synchronous Channel 0 Generator Selection bit 3 mask. */
03157 #define EVSYS_SYNCCH03_bp  3  /* Synchronous Channel 0 Generator Selection bit 3 position. */
03158 #define EVSYS_SYNCCH04_bm  (1<<4)  /* Synchronous Channel 0 Generator Selection bit 4 mask. */
03159 #define EVSYS_SYNCCH04_bp  4  /* Synchronous Channel 0 Generator Selection bit 4 position. */
03160 #define EVSYS_SYNCCH05_bm  (1<<5)  /* Synchronous Channel 0 Generator Selection bit 5 mask. */
03161 #define EVSYS_SYNCCH05_bp  5  /* Synchronous Channel 0 Generator Selection bit 5 position. */
03162 #define EVSYS_SYNCCH06_bm  (1<<6)  /* Synchronous Channel 0 Generator Selection bit 6 mask. */
03163 #define EVSYS_SYNCCH06_bp  6  /* Synchronous Channel 0 Generator Selection bit 6 position. */
03164 #define EVSYS_SYNCCH07_bm  (1<<7)  /* Synchronous Channel 0 Generator Selection bit 7 mask. */
03165 #define EVSYS_SYNCCH07_bp  7  /* Synchronous Channel 0 Generator Selection bit 7 position. */
03166
03167 /* EVSYS.ASYNCUSER0  bit masks and bit positions */
03168 #define EVSYS_ASYNCUSER0_gm  0xFF  /* Asynchronous User Ch 0 Input Selection - TCB0 group mask. */
03169 #define EVSYS_ASYNCUSER0_gp  0  /* Asynchronous User Ch 0 Input Selection - TCB0 group position. */
03170 #define EVSYS_ASYNCUSER00_bm  (1<<0)  /* Asynchronous User Ch 0 Input Selection - TCB0 bit 0 mask. */
03171 #define EVSYS_ASYNCUSER00_bp  0  /* Asynchronous User Ch 0 Input Selection - TCB0 bit 0 position. */
03172 #define EVSYS_ASYNCUSER01_bm  (1<<1)  /* Asynchronous User Ch 0 Input Selection - TCB0 bit 1 mask. */
03173 #define EVSYS_ASYNCUSER01_bp  1  /* Asynchronous User Ch 0 Input Selection - TCB0 bit 1 position. */
03174 #define EVSYS_ASYNCUSER02_bm  (1<<2)  /* Asynchronous User Ch 0 Input Selection - TCB0 bit 2 mask. */
03175 #define EVSYS_ASYNCUSER02_bp  2  /* Asynchronous User Ch 0 Input Selection - TCB0 bit 2 position. */
03176 #define EVSYS_ASYNCUSER03_bm  (1<<3)  /* Asynchronous User Ch 0 Input Selection - TCB0 bit 3 mask. */
03177 #define EVSYS_ASYNCUSER03_bp  3  /* Asynchronous User Ch 0 Input Selection - TCB0 bit 3 position. */
03178 #define EVSYS_ASYNCUSER04_bm  (1<<4)  /* Asynchronous User Ch 0 Input Selection - TCB0 bit 4 mask. */
03179 #define EVSYS_ASYNCUSER04_bp  4  /* Asynchronous User Ch 0 Input Selection - TCB0 bit 4 position. */
03180 #define EVSYS_ASYNCUSER05_bm  (1<<5)  /* Asynchronous User Ch 0 Input Selection - TCB0 bit 5 mask. */
03181 #define EVSYS_ASYNCUSER05_bp  5  /* Asynchronous User Ch 0 Input Selection - TCB0 bit 5 position. */
03182 #define EVSYS_ASYNCUSER06_bm  (1<<6)  /* Asynchronous User Ch 0 Input Selection - TCB0 bit 6 mask. */
03183 #define EVSYS_ASYNCUSER06_bp  6  /* Asynchronous User Ch 0 Input Selection - TCB0 bit 6 position. */
03184 #define EVSYS_ASYNCUSER07_bm  (1<<7)  /* Asynchronous User Ch 0 Input Selection - TCB0 bit 7 mask. */
03185 #define EVSYS_ASYNCUSER07_bp  7  /* Asynchronous User Ch 0 Input Selection - TCB0 bit 7 position. */
03186
03187 /* EVSYS.ASYNCUSER1  bit masks and bit positions */
03188 #define EVSYS_ASYNCUSER1_gm  0xFF  /* Asynchronous User Ch 1 Input Selection - ADC0 group mask. */
03189 #define EVSYS_ASYNCUSER1_gp  0  /* Asynchronous User Ch 1 Input Selection - ADC0 group position. */
03190 #define EVSYS_ASYNCUSER10_bm  (1<<0)  /* Asynchronous User Ch 1 Input Selection - ADC0 bit 0 mask. */
03191 #define EVSYS_ASYNCUSER10_bp  0  /* Asynchronous User Ch 1 Input Selection - ADC0 bit 0 position. */
03192 #define EVSYS_ASYNCUSER11_bm  (1<<1)  /* Asynchronous User Ch 1 Input Selection - ADC0 bit 1 mask. */
03193 #define EVSYS_ASYNCUSER11_bp  1  /* Asynchronous User Ch 1 Input Selection - ADC0 bit 1 position. */
03194 #define EVSYS_ASYNCUSER12_bm  (1<<2)  /* Asynchronous User Ch 1 Input Selection - ADC0 bit 2 mask. */
03195 #define EVSYS_ASYNCUSER12_bp  2  /* Asynchronous User Ch 1 Input Selection - ADC0 bit 2 position. */
03196 #define EVSYS_ASYNCUSER13_bm  (1<<3)  /* Asynchronous User Ch 1 Input Selection - ADC0 bit 3 mask. */
03197 #define EVSYS_ASYNCUSER13_bp  3  /* Asynchronous User Ch 1 Input Selection - ADC0 bit 3 position. */
03198 #define EVSYS_ASYNCUSER14_bm  (1<<4)  /* Asynchronous User Ch 1 Input Selection - ADC0 bit 4 mask. */
03199 #define EVSYS_ASYNCUSER14_bp  4  /* Asynchronous User Ch 1 Input Selection - ADC0 bit 4 position. */
03200 #define EVSYS_ASYNCUSER15_bm  (1<<5)  /* Asynchronous User Ch 1 Input Selection - ADC0 bit 5 mask. */
03201 #define EVSYS_ASYNCUSER15_bp  5  /* Asynchronous User Ch 1 Input Selection - ADC0 bit 5 position. */
03202 #define EVSYS_ASYNCUSER16_bm  (1<<6)  /* Asynchronous User Ch 1 Input Selection - ADC0 bit 6 mask. */
03203 #define EVSYS_ASYNCUSER16_bp  6  /* Asynchronous User Ch 1 Input Selection - ADC0 bit 6 position. */
03204 #define EVSYS_ASYNCUSER17_bm  (1<<7)  /* Asynchronous User Ch 1 Input Selection - ADC0 bit 7 mask. */
03205 #define EVSYS_ASYNCUSER17_bp  7  /* Asynchronous User Ch 1 Input Selection - ADC0 bit 7 position. */
03206
03207 /* EVSYS.ASYNCUSER2  bit masks and bit positions */
03208 #define EVSYS_ASYNCUSER2_gm  0xFF  /* Asynchronous User Ch 2 Input Selection - CCL LUT0 Event 0 group
      mask. */
03209 #define EVSYS_ASYNCUSER2_gp  0  /* Asynchronous User Ch 2 Input Selection - CCL LUT0 Event 0 group
      position. */
03210 #define EVSYS_ASYNCUSER20_bm  (1<<0)  /* Asynchronous User Ch 2 Input Selection - CCL LUT0 Event 0 bit
      0 mask. */
03211 #define EVSYS_ASYNCUSER20_bp  0  /* Asynchronous User Ch 2 Input Selection - CCL LUT0 Event 0 bit 0
      position. */
03212 #define EVSYS_ASYNCUSER21_bm  (1<<1)  /* Asynchronous User Ch 2 Input Selection - CCL LUT0 Event 0 bit
      1 mask. */
03213 #define EVSYS_ASYNCUSER21_bp  1  /* Asynchronous User Ch 2 Input Selection - CCL LUT0 Event 0 bit 1
      position. */
03214 #define EVSYS_ASYNCUSER22_bm  (1<<2)  /* Asynchronous User Ch 2 Input Selection - CCL LUT0 Event 0 bit
      2 mask. */
03215 #define EVSYS_ASYNCUSER22_bp  2  /* Asynchronous User Ch 2 Input Selection - CCL LUT0 Event 0 bit 2
      position. */
03216 #define EVSYS_ASYNCUSER23_bm  (1<<3)  /* Asynchronous User Ch 2 Input Selection - CCL LUT0 Event 0 bit
      3 mask. */
03217 #define EVSYS_ASYNCUSER23_bp  3  /* Asynchronous User Ch 2 Input Selection - CCL LUT0 Event 0 bit 3
      position. */
```

```
03218 #define EVSYS_ASYNCUSER24_bm  (1«4)  /* Asynchronous User Ch 2 Input Selection - CCL LUT0 Event 0 bit
      4 mask. */
03219 #define EVSYS_ASYNCUSER24_bp  4  /* Asynchronous User Ch 2 Input Selection - CCL LUT0 Event 0 bit 4
      position. */
03220 #define EVSYS_ASYNCUSER25_bm  (1«5)  /* Asynchronous User Ch 2 Input Selection - CCL LUT0 Event 0 bit
      5 mask. */
03221 #define EVSYS_ASYNCUSER25_bp  5  /* Asynchronous User Ch 2 Input Selection - CCL LUT0 Event 0 bit 5
      position. */
03222 #define EVSYS_ASYNCUSER26_bm  (1«6)  /* Asynchronous User Ch 2 Input Selection - CCL LUT0 Event 0 bit
      6 mask. */
03223 #define EVSYS_ASYNCUSER26_bp  6  /* Asynchronous User Ch 2 Input Selection - CCL LUT0 Event 0 bit 6
      position. */
03224 #define EVSYS_ASYNCUSER27_bm  (1«7)  /* Asynchronous User Ch 2 Input Selection - CCL LUT0 Event 0 bit
      7 mask. */
03225 #define EVSYS_ASYNCUSER27_bp  7  /* Asynchronous User Ch 2 Input Selection - CCL LUT0 Event 0 bit 7
      position. */
03226
03227 /* EVSYS.ASYNCUSER3  bit masks and bit positions */
03228 #define EVSYS_ASYNCUSER3_gm  0xFF  /* Asynchronous User Ch 3 Input Selection - CCL LUT1 Event 0 group
      mask. */
03229 #define EVSYS_ASYNCUSER3_gp  0  /* Asynchronous User Ch 3 Input Selection - CCL LUT1 Event 0 group
      position. */
03230 #define EVSYS_ASYNCUSER30_bm  (1«0)  /* Asynchronous User Ch 3 Input Selection - CCL LUT1 Event 0 bit
      0 mask. */
03231 #define EVSYS_ASYNCUSER30_bp  0  /* Asynchronous User Ch 3 Input Selection - CCL LUT1 Event 0 bit 0
      position. */
03232 #define EVSYS_ASYNCUSER31_bm  (1«1)  /* Asynchronous User Ch 3 Input Selection - CCL LUT1 Event 0 bit
      1 mask. */
03233 #define EVSYS_ASYNCUSER31_bp  1  /* Asynchronous User Ch 3 Input Selection - CCL LUT1 Event 0 bit 1
      position. */
03234 #define EVSYS_ASYNCUSER32_bm  (1«2)  /* Asynchronous User Ch 3 Input Selection - CCL LUT1 Event 0 bit
      2 mask. */
03235 #define EVSYS_ASYNCUSER32_bp  2  /* Asynchronous User Ch 3 Input Selection - CCL LUT1 Event 0 bit 2
      position. */
03236 #define EVSYS_ASYNCUSER33_bm  (1«3)  /* Asynchronous User Ch 3 Input Selection - CCL LUT1 Event 0 bit
      3 mask. */
03237 #define EVSYS_ASYNCUSER33_bp  3  /* Asynchronous User Ch 3 Input Selection - CCL LUT1 Event 0 bit 3
      position. */
03238 #define EVSYS_ASYNCUSER34_bm  (1«4)  /* Asynchronous User Ch 3 Input Selection - CCL LUT1 Event 0 bit
      4 mask. */
03239 #define EVSYS_ASYNCUSER34_bp  4  /* Asynchronous User Ch 3 Input Selection - CCL LUT1 Event 0 bit 4
      position. */
03240 #define EVSYS_ASYNCUSER35_bm  (1«5)  /* Asynchronous User Ch 3 Input Selection - CCL LUT1 Event 0 bit
      5 mask. */
03241 #define EVSYS_ASYNCUSER35_bp  5  /* Asynchronous User Ch 3 Input Selection - CCL LUT1 Event 0 bit 5
      position. */
03242 #define EVSYS_ASYNCUSER36_bm  (1«6)  /* Asynchronous User Ch 3 Input Selection - CCL LUT1 Event 0 bit
      6 mask. */
03243 #define EVSYS_ASYNCUSER36_bp  6  /* Asynchronous User Ch 3 Input Selection - CCL LUT1 Event 0 bit 6
      position. */
03244 #define EVSYS_ASYNCUSER37_bm  (1«7)  /* Asynchronous User Ch 3 Input Selection - CCL LUT1 Event 0 bit
      7 mask. */
03245 #define EVSYS_ASYNCUSER37_bp  7  /* Asynchronous User Ch 3 Input Selection - CCL LUT1 Event 0 bit 7
      position. */
03246
03247 /* EVSYS.ASYNCUSER4  bit masks and bit positions */
03248 #define EVSYS_ASYNCUSER4_gm  0xFF  /* Asynchronous User Ch 4 Input Selection - CCL LUT0 Event 1 group
      mask. */
03249 #define EVSYS_ASYNCUSER4_gp  0  /* Asynchronous User Ch 4 Input Selection - CCL LUT0 Event 1 group
      position. */
03250 #define EVSYS_ASYNCUSER40_bm  (1«0)  /* Asynchronous User Ch 4 Input Selection - CCL LUT0 Event 1 bit
      0 mask. */
03251 #define EVSYS_ASYNCUSER40_bp  0  /* Asynchronous User Ch 4 Input Selection - CCL LUT0 Event 1 bit 0
      position. */
03252 #define EVSYS_ASYNCUSER41_bm  (1«1)  /* Asynchronous User Ch 4 Input Selection - CCL LUT0 Event 1 bit
      1 mask. */
03253 #define EVSYS_ASYNCUSER41_bp  1  /* Asynchronous User Ch 4 Input Selection - CCL LUT0 Event 1 bit 1
      position. */
03254 #define EVSYS_ASYNCUSER42_bm  (1«2)  /* Asynchronous User Ch 4 Input Selection - CCL LUT0 Event 1 bit
      2 mask. */
03255 #define EVSYS_ASYNCUSER42_bp  2  /* Asynchronous User Ch 4 Input Selection - CCL LUT0 Event 1 bit 2
      position. */
03256 #define EVSYS_ASYNCUSER43_bm  (1«3)  /* Asynchronous User Ch 4 Input Selection - CCL LUT0 Event 1 bit
      3 mask. */
03257 #define EVSYS_ASYNCUSER43_bp  3  /* Asynchronous User Ch 4 Input Selection - CCL LUT0 Event 1 bit 3
      position. */
03258 #define EVSYS_ASYNCUSER44_bm  (1«4)  /* Asynchronous User Ch 4 Input Selection - CCL LUT0 Event 1 bit
      4 mask. */
03259 #define EVSYS_ASYNCUSER44_bp  4  /* Asynchronous User Ch 4 Input Selection - CCL LUT0 Event 1 bit 4
      position. */
03260 #define EVSYS_ASYNCUSER45_bm  (1«5)  /* Asynchronous User Ch 4 Input Selection - CCL LUT0 Event 1 bit
      5 mask. */
03261 #define EVSYS_ASYNCUSER45_bp  5  /* Asynchronous User Ch 4 Input Selection - CCL LUT0 Event 1 bit 5
      position. */
03262 #define EVSYS_ASYNCUSER46_bm  (1«6)  /* Asynchronous User Ch 4 Input Selection - CCL LUT0 Event 1 bit
      6 mask. */
03263 #define EVSYS_ASYNCUSER46_bp  6  /* Asynchronous User Ch 4 Input Selection - CCL LUT0 Event 1 bit 6
```

```
      position. */
03264 #define EVSYS_ASYNCUSER47_bm  (1«7)  /* Asynchronous User Ch 4 Input Selection - CCL LUT0 Event 1 bit
      7 mask. */
03265 #define EVSYS_ASYNCUSER47_bp  7  /* Asynchronous User Ch 4 Input Selection - CCL LUT0 Event 1 bit 7
      position. */
03266
03267 /* EVSYS.ASYNCUSER5  bit masks and bit positions */
03268 #define EVSYS_ASYNCUSER5_gm  0xFF  /* Asynchronous User Ch 5 Input Selection - CCL LUT1 Event 1 group
      mask. */
03269 #define EVSYS_ASYNCUSER5_gp  0  /* Asynchronous User Ch 5 Input Selection - CCL LUT1 Event 1 group
      position. */
03270 #define EVSYS_ASYNCUSER50_bm  (1«0)  /* Asynchronous User Ch 5 Input Selection - CCL LUT1 Event 1 bit
      0 mask. */
03271 #define EVSYS_ASYNCUSER50_bp  0  /* Asynchronous User Ch 5 Input Selection - CCL LUT1 Event 1 bit 0
      position. */
03272 #define EVSYS_ASYNCUSER51_bm  (1«1)  /* Asynchronous User Ch 5 Input Selection - CCL LUT1 Event 1 bit
      1 mask. */
03273 #define EVSYS_ASYNCUSER51_bp  1  /* Asynchronous User Ch 5 Input Selection - CCL LUT1 Event 1 bit 1
      position. */
03274 #define EVSYS_ASYNCUSER52_bm  (1«2)  /* Asynchronous User Ch 5 Input Selection - CCL LUT1 Event 1 bit
      2 mask. */
03275 #define EVSYS_ASYNCUSER52_bp  2  /* Asynchronous User Ch 5 Input Selection - CCL LUT1 Event 1 bit 2
      position. */
03276 #define EVSYS_ASYNCUSER53_bm  (1«3)  /* Asynchronous User Ch 5 Input Selection - CCL LUT1 Event 1 bit
      3 mask. */
03277 #define EVSYS_ASYNCUSER53_bp  3  /* Asynchronous User Ch 5 Input Selection - CCL LUT1 Event 1 bit 3
      position. */
03278 #define EVSYS_ASYNCUSER54_bm  (1«4)  /* Asynchronous User Ch 5 Input Selection - CCL LUT1 Event 1 bit
      4 mask. */
03279 #define EVSYS_ASYNCUSER54_bp  4  /* Asynchronous User Ch 5 Input Selection - CCL LUT1 Event 1 bit 4
      position. */
03280 #define EVSYS_ASYNCUSER55_bm  (1«5)  /* Asynchronous User Ch 5 Input Selection - CCL LUT1 Event 1 bit
      5 mask. */
03281 #define EVSYS_ASYNCUSER55_bp  5  /* Asynchronous User Ch 5 Input Selection - CCL LUT1 Event 1 bit 5
      position. */
03282 #define EVSYS_ASYNCUSER56_bm  (1«6)  /* Asynchronous User Ch 5 Input Selection - CCL LUT1 Event 1 bit
      6 mask. */
03283 #define EVSYS_ASYNCUSER56_bp  6  /* Asynchronous User Ch 5 Input Selection - CCL LUT1 Event 1 bit 6
      position. */
03284 #define EVSYS_ASYNCUSER57_bm  (1«7)  /* Asynchronous User Ch 5 Input Selection - CCL LUT1 Event 1 bit
      7 mask. */
03285 #define EVSYS_ASYNCUSER57_bp  7  /* Asynchronous User Ch 5 Input Selection - CCL LUT1 Event 1 bit 7
      position. */
03286
03287 /* EVSYS.ASYNCUSER6  bit masks and bit positions */
03288 #define EVSYS_ASYNCUSER6_gm  0xFF  /* Asynchronous User Ch 6 Input Selection - TCD0 Event 0 group
      mask. */
03289 #define EVSYS_ASYNCUSER6_gp  0  /* Asynchronous User Ch 6 Input Selection - TCD0 Event 0 group
      position. */
03290 #define EVSYS_ASYNCUSER60_bm  (1«0)  /* Asynchronous User Ch 6 Input Selection - TCD0 Event 0 bit 0
      mask. */
03291 #define EVSYS_ASYNCUSER60_bp  0  /* Asynchronous User Ch 6 Input Selection - TCD0 Event 0 bit 0
      position. */
03292 #define EVSYS_ASYNCUSER61_bm  (1«1)  /* Asynchronous User Ch 6 Input Selection - TCD0 Event 0 bit 1
      mask. */
03293 #define EVSYS_ASYNCUSER61_bp  1  /* Asynchronous User Ch 6 Input Selection - TCD0 Event 0 bit 1
      position. */
03294 #define EVSYS_ASYNCUSER62_bm  (1«2)  /* Asynchronous User Ch 6 Input Selection - TCD0 Event 0 bit 2
      mask. */
03295 #define EVSYS_ASYNCUSER62_bp  2  /* Asynchronous User Ch 6 Input Selection - TCD0 Event 0 bit 2
      position. */
03296 #define EVSYS_ASYNCUSER63_bm  (1«3)  /* Asynchronous User Ch 6 Input Selection - TCD0 Event 0 bit 3
      mask. */
03297 #define EVSYS_ASYNCUSER63_bp  3  /* Asynchronous User Ch 6 Input Selection - TCD0 Event 0 bit 3
      position. */
03298 #define EVSYS_ASYNCUSER64_bm  (1«4)  /* Asynchronous User Ch 6 Input Selection - TCD0 Event 0 bit 4
      mask. */
03299 #define EVSYS_ASYNCUSER64_bp  4  /* Asynchronous User Ch 6 Input Selection - TCD0 Event 0 bit 4
      position. */
03300 #define EVSYS_ASYNCUSER65_bm  (1«5)  /* Asynchronous User Ch 6 Input Selection - TCD0 Event 0 bit 5
      mask. */
03301 #define EVSYS_ASYNCUSER65_bp  5  /* Asynchronous User Ch 6 Input Selection - TCD0 Event 0 bit 5
      position. */
03302 #define EVSYS_ASYNCUSER66_bm  (1«6)  /* Asynchronous User Ch 6 Input Selection - TCD0 Event 0 bit 6
      mask. */
03303 #define EVSYS_ASYNCUSER66_bp  6  /* Asynchronous User Ch 6 Input Selection - TCD0 Event 0 bit 6
      position. */
03304 #define EVSYS_ASYNCUSER67_bm  (1«7)  /* Asynchronous User Ch 6 Input Selection - TCD0 Event 0 bit 7
      mask. */
03305 #define EVSYS_ASYNCUSER67_bp  7  /* Asynchronous User Ch 6 Input Selection - TCD0 Event 0 bit 7
      position. */
03306
03307 /* EVSYS.ASYNCUSER7  bit masks and bit positions */
03308 #define EVSYS_ASYNCUSER7_gm  0xFF  /* Asynchronous User Ch 7 Input Selection - TCD0 Event 1 group
      mask. */
03309 #define EVSYS_ASYNCUSER7_gp  0  /* Asynchronous User Ch 7 Input Selection - TCD0 Event 1 group
      position. */
```

```
03310 #define EVSYS_ASYNCUSER70_bm  (1«0)  /* Asynchronous User Ch 7 Input Selection - TCD0 Event 1 bit 0
      mask. */
03311 #define EVSYS_ASYNCUSER70_bp  0  /* Asynchronous User Ch 7 Input Selection - TCD0 Event 1 bit 0
      position. */
03312 #define EVSYS_ASYNCUSER71_bm  (1«1)  /* Asynchronous User Ch 7 Input Selection - TCD0 Event 1 bit 1
      mask. */
03313 #define EVSYS_ASYNCUSER71_bp  1  /* Asynchronous User Ch 7 Input Selection - TCD0 Event 1 bit 1
      position. */
03314 #define EVSYS_ASYNCUSER72_bm  (1«2)  /* Asynchronous User Ch 7 Input Selection - TCD0 Event 1 bit 2
      mask. */
03315 #define EVSYS_ASYNCUSER72_bp  2  /* Asynchronous User Ch 7 Input Selection - TCD0 Event 1 bit 2
      position. */
03316 #define EVSYS_ASYNCUSER73_bm  (1«3)  /* Asynchronous User Ch 7 Input Selection - TCD0 Event 1 bit 3
      mask. */
03317 #define EVSYS_ASYNCUSER73_bp  3  /* Asynchronous User Ch 7 Input Selection - TCD0 Event 1 bit 3
      position. */
03318 #define EVSYS_ASYNCUSER74_bm  (1«4)  /* Asynchronous User Ch 7 Input Selection - TCD0 Event 1 bit 4
      mask. */
03319 #define EVSYS_ASYNCUSER74_bp  4  /* Asynchronous User Ch 7 Input Selection - TCD0 Event 1 bit 4
      position. */
03320 #define EVSYS_ASYNCUSER75_bm  (1«5)  /* Asynchronous User Ch 7 Input Selection - TCD0 Event 1 bit 5
      mask. */
03321 #define EVSYS_ASYNCUSER75_bp  5  /* Asynchronous User Ch 7 Input Selection - TCD0 Event 1 bit 5
      position. */
03322 #define EVSYS_ASYNCUSER76_bm  (1«6)  /* Asynchronous User Ch 7 Input Selection - TCD0 Event 1 bit 6
      mask. */
03323 #define EVSYS_ASYNCUSER76_bp  6  /* Asynchronous User Ch 7 Input Selection - TCD0 Event 1 bit 6
      position. */
03324 #define EVSYS_ASYNCUSER77_bm  (1«7)  /* Asynchronous User Ch 7 Input Selection - TCD0 Event 1 bit 7
      mask. */
03325 #define EVSYS_ASYNCUSER77_bp  7  /* Asynchronous User Ch 7 Input Selection - TCD0 Event 1 bit 7
      position. */
03326
03327 /* EVSYS.ASYNCUSER8  bit masks and bit positions */
03328 #define EVSYS_ASYNCUSER8_gm  0xFF  /* Asynchronous User Ch 8 Input Selection - Event Out 0 group mask.
      */
03329 #define EVSYS_ASYNCUSER8_gp  0  /* Asynchronous User Ch 8 Input Selection - Event Out 0 group
      position. */
03330 #define EVSYS_ASYNCUSER80_bm  (1«0)  /* Asynchronous User Ch 8 Input Selection - Event Out 0 bit 0
      mask. */
03331 #define EVSYS_ASYNCUSER80_bp  0  /* Asynchronous User Ch 8 Input Selection - Event Out 0 bit 0
      position. */
03332 #define EVSYS_ASYNCUSER81_bm  (1«1)  /* Asynchronous User Ch 8 Input Selection - Event Out 0 bit 1
      mask. */
03333 #define EVSYS_ASYNCUSER81_bp  1  /* Asynchronous User Ch 8 Input Selection - Event Out 0 bit 1
      position. */
03334 #define EVSYS_ASYNCUSER82_bm  (1«2)  /* Asynchronous User Ch 8 Input Selection - Event Out 0 bit 2
      mask. */
03335 #define EVSYS_ASYNCUSER82_bp  2  /* Asynchronous User Ch 8 Input Selection - Event Out 0 bit 2
      position. */
03336 #define EVSYS_ASYNCUSER83_bm  (1«3)  /* Asynchronous User Ch 8 Input Selection - Event Out 0 bit 3
      mask. */
03337 #define EVSYS_ASYNCUSER83_bp  3  /* Asynchronous User Ch 8 Input Selection - Event Out 0 bit 3
      position. */
03338 #define EVSYS_ASYNCUSER84_bm  (1«4)  /* Asynchronous User Ch 8 Input Selection - Event Out 0 bit 4
      mask. */
03339 #define EVSYS_ASYNCUSER84_bp  4  /* Asynchronous User Ch 8 Input Selection - Event Out 0 bit 4
      position. */
03340 #define EVSYS_ASYNCUSER85_bm  (1«5)  /* Asynchronous User Ch 8 Input Selection - Event Out 0 bit 5
      mask. */
03341 #define EVSYS_ASYNCUSER85_bp  5  /* Asynchronous User Ch 8 Input Selection - Event Out 0 bit 5
      position. */
03342 #define EVSYS_ASYNCUSER86_bm  (1«6)  /* Asynchronous User Ch 8 Input Selection - Event Out 0 bit 6
      mask. */
03343 #define EVSYS_ASYNCUSER86_bp  6  /* Asynchronous User Ch 8 Input Selection - Event Out 0 bit 6
      position. */
03344 #define EVSYS_ASYNCUSER87_bm  (1«7)  /* Asynchronous User Ch 8 Input Selection - Event Out 0 bit 7
      mask. */
03345 #define EVSYS_ASYNCUSER87_bp  7  /* Asynchronous User Ch 8 Input Selection - Event Out 0 bit 7
      position. */
03346
03347 /* EVSYS.ASYNCUSER9  bit masks and bit positions */
03348 #define EVSYS_ASYNCUSER9_gm  0xFF  /* Asynchronous User Ch 9 Input Selection - Event Out 1 group mask.
      */
03349 #define EVSYS_ASYNCUSER9_gp  0  /* Asynchronous User Ch 9 Input Selection - Event Out 1 group
      position. */
03350 #define EVSYS_ASYNCUSER90_bm  (1«0)  /* Asynchronous User Ch 9 Input Selection - Event Out 1 bit 0
      mask. */
03351 #define EVSYS_ASYNCUSER90_bp  0  /* Asynchronous User Ch 9 Input Selection - Event Out 1 bit 0
      position. */
03352 #define EVSYS_ASYNCUSER91_bm  (1«1)  /* Asynchronous User Ch 9 Input Selection - Event Out 1 bit 1
      mask. */
03353 #define EVSYS_ASYNCUSER91_bp  1  /* Asynchronous User Ch 9 Input Selection - Event Out 1 bit 1
      position. */
03354 #define EVSYS_ASYNCUSER92_bm  (1«2)  /* Asynchronous User Ch 9 Input Selection - Event Out 1 bit 2
      mask. */
03355 #define EVSYS_ASYNCUSER92_bp  2  /* Asynchronous User Ch 9 Input Selection - Event Out 1 bit 2
```

```
      position. */
03356 #define EVSYS_ASYNCUSER93_bm  (1«3)  /* Asynchronous User Ch 9 Input Selection - Event Out 1 bit 3
      mask. */
03357 #define EVSYS_ASYNCUSER93_bp  3  /* Asynchronous User Ch 9 Input Selection - Event Out 1 bit 3
      position. */
03358 #define EVSYS_ASYNCUSER94_bm  (1«4)  /* Asynchronous User Ch 9 Input Selection - Event Out 1 bit 4
      mask. */
03359 #define EVSYS_ASYNCUSER94_bp  4  /* Asynchronous User Ch 9 Input Selection - Event Out 1 bit 4
      position. */
03360 #define EVSYS_ASYNCUSER95_bm  (1«5)  /* Asynchronous User Ch 9 Input Selection - Event Out 1 bit 5
      mask. */
03361 #define EVSYS_ASYNCUSER95_bp  5  /* Asynchronous User Ch 9 Input Selection - Event Out 1 bit 5
      position. */
03362 #define EVSYS_ASYNCUSER96_bm  (1«6)  /* Asynchronous User Ch 9 Input Selection - Event Out 1 bit 6
      mask. */
03363 #define EVSYS_ASYNCUSER96_bp  6  /* Asynchronous User Ch 9 Input Selection - Event Out 1 bit 6
      position. */
03364 #define EVSYS_ASYNCUSER97_bm  (1«7)  /* Asynchronous User Ch 9 Input Selection - Event Out 1 bit 7
      mask. */
03365 #define EVSYS_ASYNCUSER97_bp  7  /* Asynchronous User Ch 9 Input Selection - Event Out 1 bit 7
      position. */
03366
03367 /* EVSYS.ASYNCUSER10  bit masks and bit positions */
03368 #define EVSYS_ASYNCUSER10_gm  0xFF  /* Asynchronous User Ch 10 Input Selection - Event Out 2 group
      mask. */
03369 #define EVSYS_ASYNCUSER10_gp  0  /* Asynchronous User Ch 10 Input Selection - Event Out 2 group
      position. */
03370 #define EVSYS_ASYNCUSER100_bm  (1«0)  /* Asynchronous User Ch 10 Input Selection - Event Out 2 bit 0
      mask. */
03371 #define EVSYS_ASYNCUSER100_bp  0  /* Asynchronous User Ch 10 Input Selection - Event Out 2 bit 0
      position. */
03372 #define EVSYS_ASYNCUSER101_bm  (1«1)  /* Asynchronous User Ch 10 Input Selection - Event Out 2 bit 1
      mask. */
03373 #define EVSYS_ASYNCUSER101_bp  1  /* Asynchronous User Ch 10 Input Selection - Event Out 2 bit 1
      position. */
03374 #define EVSYS_ASYNCUSER102_bm  (1«2)  /* Asynchronous User Ch 10 Input Selection - Event Out 2 bit 2
      mask. */
03375 #define EVSYS_ASYNCUSER102_bp  2  /* Asynchronous User Ch 10 Input Selection - Event Out 2 bit 2
      position. */
03376 #define EVSYS_ASYNCUSER103_bm  (1«3)  /* Asynchronous User Ch 10 Input Selection - Event Out 2 bit 3
      mask. */
03377 #define EVSYS_ASYNCUSER103_bp  3  /* Asynchronous User Ch 10 Input Selection - Event Out 2 bit 3
      position. */
03378 #define EVSYS_ASYNCUSER104_bm  (1«4)  /* Asynchronous User Ch 10 Input Selection - Event Out 2 bit 4
      mask. */
03379 #define EVSYS_ASYNCUSER104_bp  4  /* Asynchronous User Ch 10 Input Selection - Event Out 2 bit 4
      position. */
03380 #define EVSYS_ASYNCUSER105_bm  (1«5)  /* Asynchronous User Ch 10 Input Selection - Event Out 2 bit 5
      mask. */
03381 #define EVSYS_ASYNCUSER105_bp  5  /* Asynchronous User Ch 10 Input Selection - Event Out 2 bit 5
      position. */
03382 #define EVSYS_ASYNCUSER106_bm  (1«6)  /* Asynchronous User Ch 10 Input Selection - Event Out 2 bit 6
      mask. */
03383 #define EVSYS_ASYNCUSER106_bp  6  /* Asynchronous User Ch 10 Input Selection - Event Out 2 bit 6
      position. */
03384 #define EVSYS_ASYNCUSER107_bm  (1«7)  /* Asynchronous User Ch 10 Input Selection - Event Out 2 bit 7
      mask. */
03385 #define EVSYS_ASYNCUSER107_bp  7  /* Asynchronous User Ch 10 Input Selection - Event Out 2 bit 7
      position. */
03386
03387 /* EVSYS.SYNCUSER0  bit masks and bit positions */
03388 #define EVSYS_SYNCUSER0_gm  0xFF  /* Synchronous User Ch 0 Input Selection - TCA0 group mask. */
03389 #define EVSYS_SYNCUSER0_gp  0  /* Synchronous User Ch 0 Input Selection - TCA0 group position. */
03390 #define EVSYS_SYNCUSER00_bm  (1«0)  /* Synchronous User Ch 0 Input Selection - TCA0 bit 0 mask. */
03391 #define EVSYS_SYNCUSER00_bp  0  /* Synchronous User Ch 0 Input Selection - TCA0 bit 0 position. */
03392 #define EVSYS_SYNCUSER01_bm  (1«1)  /* Synchronous User Ch 0 Input Selection - TCA0 bit 1 mask. */
03393 #define EVSYS_SYNCUSER01_bp  1  /* Synchronous User Ch 0 Input Selection - TCA0 bit 1 position. */
03394 #define EVSYS_SYNCUSER02_bm  (1«2)  /* Synchronous User Ch 0 Input Selection - TCA0 bit 2 mask. */
03395 #define EVSYS_SYNCUSER02_bp  2  /* Synchronous User Ch 0 Input Selection - TCA0 bit 2 position. */
03396 #define EVSYS_SYNCUSER03_bm  (1«3)  /* Synchronous User Ch 0 Input Selection - TCA0 bit 3 mask. */
03397 #define EVSYS_SYNCUSER03_bp  3  /* Synchronous User Ch 0 Input Selection - TCA0 bit 3 position. */
03398 #define EVSYS_SYNCUSER04_bm  (1«4)  /* Synchronous User Ch 0 Input Selection - TCA0 bit 4 mask. */
03399 #define EVSYS_SYNCUSER04_bp  4  /* Synchronous User Ch 0 Input Selection - TCA0 bit 4 position. */
03400 #define EVSYS_SYNCUSER05_bm  (1«5)  /* Synchronous User Ch 0 Input Selection - TCA0 bit 5 mask. */
03401 #define EVSYS_SYNCUSER05_bp  5  /* Synchronous User Ch 0 Input Selection - TCA0 bit 5 position. */
03402 #define EVSYS_SYNCUSER06_bm  (1«6)  /* Synchronous User Ch 0 Input Selection - TCA0 bit 6 mask. */
03403 #define EVSYS_SYNCUSER06_bp  6  /* Synchronous User Ch 0 Input Selection - TCA0 bit 6 position. */
03404 #define EVSYS_SYNCUSER07_bm  (1«7)  /* Synchronous User Ch 0 Input Selection - TCA0 bit 7 mask. */
03405 #define EVSYS_SYNCUSER07_bp  7  /* Synchronous User Ch 0 Input Selection - TCA0 bit 7 position. */
03406
03407 /* EVSYS.SYNCUSER1  bit masks and bit positions */
03408 #define EVSYS_SYNCUSER1_gm  0xFF  /* Synchronous User Ch 1 Input Selection - USART0 group mask. */
03409 #define EVSYS_SYNCUSER1_gp  0  /* Synchronous User Ch 1 Input Selection - USART0 group position. */
03410 #define EVSYS_SYNCUSER10_bm  (1«0)  /* Synchronous User Ch 1 Input Selection - USART0 bit 0 mask. */
03411 #define EVSYS_SYNCUSER10_bp  0  /* Synchronous User Ch 1 Input Selection - USART0 bit 0 position. */
03412 #define EVSYS_SYNCUSER11_bm  (1«1)  /* Synchronous User Ch 1 Input Selection - USART0 bit 1 mask. */
03413 #define EVSYS_SYNCUSER11_bp  1  /* Synchronous User Ch 1 Input Selection - USART0 bit 1 position. */
```

```
03414 #define EVSYS_SYNCUSER12_bm  (1«2)  /* Synchronous User Ch 1 Input Selection - USART0 bit 2 mask. */
03415 #define EVSYS_SYNCUSER12_bp  2  /* Synchronous User Ch 1 Input Selection - USART0 bit 2 position. */
03416 #define EVSYS_SYNCUSER13_bm  (1«3)  /* Synchronous User Ch 1 Input Selection - USART0 bit 3 mask. */
03417 #define EVSYS_SYNCUSER13_bp  3  /* Synchronous User Ch 1 Input Selection - USART0 bit 3 position. */
03418 #define EVSYS_SYNCUSER14_bm  (1«4)  /* Synchronous User Ch 1 Input Selection - USART0 bit 4 mask. */
03419 #define EVSYS_SYNCUSER14_bp  4  /* Synchronous User Ch 1 Input Selection - USART0 bit 4 position. */
03420 #define EVSYS_SYNCUSER15_bm  (1«5)  /* Synchronous User Ch 1 Input Selection - USART0 bit 5 mask. */
03421 #define EVSYS_SYNCUSER15_bp  5  /* Synchronous User Ch 1 Input Selection - USART0 bit 5 position. */
03422 #define EVSYS_SYNCUSER16_bm  (1«6)  /* Synchronous User Ch 1 Input Selection - USART0 bit 6 mask. */
03423 #define EVSYS_SYNCUSER16_bp  6  /* Synchronous User Ch 1 Input Selection - USART0 bit 6 position. */
03424 #define EVSYS_SYNCUSER17_bm  (1«7)  /* Synchronous User Ch 1 Input Selection - USART0 bit 7 mask. */
03425 #define EVSYS_SYNCUSER17_bp  7  /* Synchronous User Ch 1 Input Selection - USART0 bit 7 position. */
03426
03427 /* FUSE - Fuses */
03428 /* FUSE.WDTCFG  bit masks and bit positions */
03429 #define FUSE_PERIOD_gm  0x0F  /* Watchdog Timeout Period group mask. */
03430 #define FUSE_PERIOD_gp  0  /* Watchdog Timeout Period group position. */
03431 #define FUSE_PERIOD0_bm  (1«0)  /* Watchdog Timeout Period bit 0 mask. */
03432 #define FUSE_PERIOD0_bp  0  /* Watchdog Timeout Period bit 0 position. */
03433 #define FUSE_PERIOD1_bm  (1«1)  /* Watchdog Timeout Period bit 1 mask. */
03434 #define FUSE_PERIOD1_bp  1  /* Watchdog Timeout Period bit 1 position. */
03435 #define FUSE_PERIOD2_bm  (1«2)  /* Watchdog Timeout Period bit 2 mask. */
03436 #define FUSE_PERIOD2_bp  2  /* Watchdog Timeout Period bit 2 position. */
03437 #define FUSE_PERIOD3_bm  (1«3)  /* Watchdog Timeout Period bit 3 mask. */
03438 #define FUSE_PERIOD3_bp  3  /* Watchdog Timeout Period bit 3 position. */
03439 #define FUSE_WINDOW_gm  0xF0  /* Watchdog Window Timeout Period group mask. */
03440 #define FUSE_WINDOW_gp  4  /* Watchdog Window Timeout Period group position. */
03441 #define FUSE_WINDOW0_bm  (1«4)  /* Watchdog Window Timeout Period bit 0 mask. */
03442 #define FUSE_WINDOW0_bp  4  /* Watchdog Window Timeout Period bit 0 position. */
03443 #define FUSE_WINDOW1_bm  (1«5)  /* Watchdog Window Timeout Period bit 1 mask. */
03444 #define FUSE_WINDOW1_bp  5  /* Watchdog Window Timeout Period bit 1 position. */
03445 #define FUSE_WINDOW2_bm  (1«6)  /* Watchdog Window Timeout Period bit 2 mask. */
03446 #define FUSE_WINDOW2_bp  6  /* Watchdog Window Timeout Period bit 2 position. */
03447 #define FUSE_WINDOW3_bm  (1«7)  /* Watchdog Window Timeout Period bit 3 mask. */
03448 #define FUSE_WINDOW3_bp  7  /* Watchdog Window Timeout Period bit 3 position. */
03449
03450 /* FUSE.BODCFG  bit masks and bit positions */
03451 #define FUSE_SLEEP_gm  0x03  /* BOD Operation in Sleep Mode group mask. */
03452 #define FUSE_SLEEP_gp  0  /* BOD Operation in Sleep Mode group position. */
03453 #define FUSE_SLEEP0_bm  (1«0)  /* BOD Operation in Sleep Mode bit 0 mask. */
03454 #define FUSE_SLEEP0_bp  0  /* BOD Operation in Sleep Mode bit 0 position. */
03455 #define FUSE_SLEEP1_bm  (1«1)  /* BOD Operation in Sleep Mode bit 1 mask. */
03456 #define FUSE_SLEEP1_bp  1  /* BOD Operation in Sleep Mode bit 1 position. */
03457 #define FUSE_ACTIVE_gm  0x0C  /* BOD Operation in Active Mode group mask. */
03458 #define FUSE_ACTIVE_gp  2  /* BOD Operation in Active Mode group position. */
03459 #define FUSE_ACTIVE0_bm  (1«2)  /* BOD Operation in Active Mode bit 0 mask. */
03460 #define FUSE_ACTIVE0_bp  2  /* BOD Operation in Active Mode bit 0 position. */
03461 #define FUSE_ACTIVE1_bm  (1«3)  /* BOD Operation in Active Mode bit 1 mask. */
03462 #define FUSE_ACTIVE1_bp  3  /* BOD Operation in Active Mode bit 1 position. */
03463 #define FUSE_SAMPFREQ_bm  0x10  /* BOD Sample Frequency bit mask. */
03464 #define FUSE_SAMPFREQ_bp  4  /* BOD Sample Frequency bit position. */
03465 #define FUSE_LVL_gm  0xE0  /* BOD Level group mask. */
03466 #define FUSE_LVL_gp  5  /* BOD Level group position. */
03467 #define FUSE_LVL0_bm  (1«5)  /* BOD Level bit 0 mask. */
03468 #define FUSE_LVL0_bp  5  /* BOD Level bit 0 position. */
03469 #define FUSE_LVL1_bm  (1«6)  /* BOD Level bit 1 mask. */
03470 #define FUSE_LVL1_bp  6  /* BOD Level bit 1 position. */
03471 #define FUSE_LVL2_bm  (1«7)  /* BOD Level bit 2 mask. */
03472 #define FUSE_LVL2_bp  7  /* BOD Level bit 2 position. */
03473
03474 /* FUSE.OSCCFG  bit masks and bit positions */
03475 #define FUSE_FREQSEL_gm  0x03  /* Frequency Select group mask. */
03476 #define FUSE_FREQSEL_gp  0  /* Frequency Select group position. */
03477 #define FUSE_FREQSEL0_bm  (1«0)  /* Frequency Select bit 0 mask. */
03478 #define FUSE_FREQSEL0_bp  0  /* Frequency Select bit 0 position. */
03479 #define FUSE_FREQSEL1_bm  (1«1)  /* Frequency Select bit 1 mask. */
03480 #define FUSE_FREQSEL1_bp  1  /* Frequency Select bit 1 position. */
03481 #define FUSE_OSCLOCK_bm  0x80  /* Oscillator Lock bit mask. */
03482 #define FUSE_OSCLOCK_bp  7  /* Oscillator Lock bit position. */
03483
03484 /* FUSE.TCD0CFG  bit masks and bit positions */
03485 #define FUSE_CMPA_bm  0x01  /* Compare A Default Output Value bit mask. */
03486 #define FUSE_CMPA_bp  0  /* Compare A Default Output Value bit position. */
03487 #define FUSE_CMPB_bm  0x02  /* Compare B Default Output Value bit mask. */
03488 #define FUSE_CMPB_bp  1  /* Compare B Default Output Value bit position. */
03489 #define FUSE_CMPC_bm  0x04  /* Compare C Default Output Value bit mask. */
03490 #define FUSE_CMPC_bp  2  /* Compare C Default Output Value bit position. */
03491 #define FUSE_CMPD_bm  0x08  /* Compare D Default Output Value bit mask. */
03492 #define FUSE_CMPD_bp  3  /* Compare D Default Output Value bit position. */
03493 #define FUSE_CMPAEN_bm  0x10  /* Compare A Output Enable bit mask. */
03494 #define FUSE_CMPAEN_bp  4  /* Compare A Output Enable bit position. */
03495 #define FUSE_CMPBEN_bm  0x20  /* Compare B Output Enable bit mask. */
03496 #define FUSE_CMPBEN_bp  5  /* Compare B Output Enable bit position. */
03497 #define FUSE_CMPCEN_bm  0x40  /* Compare C Output Enable bit mask. */
03498 #define FUSE_CMPCEN_bp  6  /* Compare C Output Enable bit position. */
03499 #define FUSE_CMPDEN_bm  0x80  /* Compare D Output Enable bit mask. */
03500 #define FUSE_CMPDEN_bp  7  /* Compare D Output Enable bit position. */
```

```
03501
03502 /* FUSE.SYSCFG0  bit masks and bit positions */
03503 #define FUSE_EESAVE_bm  0x01  /* EEPROM Save bit mask. */
03504 #define FUSE_EESAVE_bp  0  /* EEPROM Save bit position. */
03505 #define FUSE_RSTPINCFG_gm  0x0C  /* Reset Pin Configuration group mask. */
03506 #define FUSE_RSTPINCFG_gp  2  /* Reset Pin Configuration group position. */
03507 #define FUSE_RSTPINCFG0_bm  (1<<2)  /* Reset Pin Configuration bit 0 mask. */
03508 #define FUSE_RSTPINCFG0_bp  2  /* Reset Pin Configuration bit 0 position. */
03509 #define FUSE_RSTPINCFG1_bm  (1<<3)  /* Reset Pin Configuration bit 1 mask. */
03510 #define FUSE_RSTPINCFG1_bp  3  /* Reset Pin Configuration bit 1 position. */
03511 #define FUSE_CRCSRC_gm  0xC0  /* CRC Source group mask. */
03512 #define FUSE_CRCSRC_gp  6  /* CRC Source group position. */
03513 #define FUSE_CRCSRC0_bm  (1<<6)  /* CRC Source bit 0 mask. */
03514 #define FUSE_CRCSRC0_bp  6  /* CRC Source bit 0 position. */
03515 #define FUSE_CRCSRC1_bm  (1<<7)  /* CRC Source bit 1 mask. */
03516 #define FUSE_CRCSRC1_bp  7  /* CRC Source bit 1 position. */
03517
03518 /* FUSE.SYSCFG1  bit masks and bit positions */
03519 #define FUSE_SUT_gm  0x07  /* Startup Time group mask. */
03520 #define FUSE_SUT_gp  0  /* Startup Time group position. */
03521 #define FUSE_SUT0_bm  (1<<0)  /* Startup Time bit 0 mask. */
03522 #define FUSE_SUT0_bp  0  /* Startup Time bit 0 position. */
03523 #define FUSE_SUT1_bm  (1<<1)  /* Startup Time bit 1 mask. */
03524 #define FUSE_SUT1_bp  1  /* Startup Time bit 1 position. */
03525 #define FUSE_SUT2_bm  (1<<2)  /* Startup Time bit 2 mask. */
03526 #define FUSE_SUT2_bp  2  /* Startup Time bit 2 position. */
03527
03528 /* LOCKBIT - Lockbit */
03529 /* LOCKBIT.LOCKBIT  bit masks and bit positions */
03530 #define LOCKBIT_LB_gm  0xFF  /* Lock Bits group mask. */
03531 #define LOCKBIT_LB_gp  0  /* Lock Bits group position. */
03532 #define LOCKBIT_LB0_bm  (1<<0)  /* Lock Bits bit 0 mask. */
03533 #define LOCKBIT_LB0_bp  0  /* Lock Bits bit 0 position. */
03534 #define LOCKBIT_LB1_bm  (1<<1)  /* Lock Bits bit 1 mask. */
03535 #define LOCKBIT_LB1_bp  1  /* Lock Bits bit 1 position. */
03536 #define LOCKBIT_LB2_bm  (1<<2)  /* Lock Bits bit 2 mask. */
03537 #define LOCKBIT_LB2_bp  2  /* Lock Bits bit 2 position. */
03538 #define LOCKBIT_LB3_bm  (1<<3)  /* Lock Bits bit 3 mask. */
03539 #define LOCKBIT_LB3_bp  3  /* Lock Bits bit 3 position. */
03540 #define LOCKBIT_LB4_bm  (1<<4)  /* Lock Bits bit 4 mask. */
03541 #define LOCKBIT_LB4_bp  4  /* Lock Bits bit 4 position. */
03542 #define LOCKBIT_LB5_bm  (1<<5)  /* Lock Bits bit 5 mask. */
03543 #define LOCKBIT_LB5_bp  5  /* Lock Bits bit 5 position. */
03544 #define LOCKBIT_LB6_bm  (1<<6)  /* Lock Bits bit 6 mask. */
03545 #define LOCKBIT_LB6_bp  6  /* Lock Bits bit 6 position. */
03546 #define LOCKBIT_LB7_bm  (1<<7)  /* Lock Bits bit 7 mask. */
03547 #define LOCKBIT_LB7_bp  7  /* Lock Bits bit 7 position. */
03548
03549 /* NVMCTRL - Non-volatile Memory Controller */
03550 /* NVMCTRL.CTRLA  bit masks and bit positions */
03551 #define NVMCTRL_CMD_gm  0x07  /* Command group mask. */
03552 #define NVMCTRL_CMD_gp  0  /* Command group position. */
03553 #define NVMCTRL_CMD0_bm  (1<<0)  /* Command bit 0 mask. */
03554 #define NVMCTRL_CMD0_bp  0  /* Command bit 0 position. */
03555 #define NVMCTRL_CMD1_bm  (1<<1)  /* Command bit 1 mask. */
03556 #define NVMCTRL_CMD1_bp  1  /* Command bit 1 position. */
03557 #define NVMCTRL_CMD2_bm  (1<<2)  /* Command bit 2 mask. */
03558 #define NVMCTRL_CMD2_bp  2  /* Command bit 2 position. */
03559
03560 /* NVMCTRL.CTRLB  bit masks and bit positions */
03561 #define NVMCTRL_APCWP_bm  0x01  /* Application code write protect bit mask. */
03562 #define NVMCTRL_APCWP_bp  0  /* Application code write protect bit position. */
03563 #define NVMCTRL_BOOTLOCK_bm  0x02  /* Boot Lock bit mask. */
03564 #define NVMCTRL_BOOTLOCK_bp  1  /* Boot Lock bit position. */
03565
03566 /* NVMCTRL.STATUS  bit masks and bit positions */
03567 #define NVMCTRL_FBUSY_bm  0x01  /* Flash busy bit mask. */
03568 #define NVMCTRL_FBUSY_bp  0  /* Flash busy bit position. */
03569 #define NVMCTRL_EEBUSY_bm  0x02  /* EEPROM busy bit mask. */
03570 #define NVMCTRL_EEBUSY_bp  1  /* EEPROM busy bit position. */
03571 #define NVMCTRL_WRERROR_bm  0x04  /* Write error bit mask. */
03572 #define NVMCTRL_WRERROR_bp  2  /* Write error bit position. */
03573
03574 /* NVMCTRL.INTCTRL  bit masks and bit positions */
03575 #define NVMCTRL_EEREADY_bm  0x01  /* EEPROM Ready bit mask. */
03576 #define NVMCTRL_EEREADY_bp  0  /* EEPROM Ready bit position. */
03577
03578 /* NVMCTRL.INTFLAGS  bit masks and bit positions */
03579 /* NVMCTRL_EEREADY  is already defined. */
03580
03581 /* PORT - I/O Ports */
03582 /* PORT.INTFLAGS  bit masks and bit positions */
03583 #define PORT_INT_gm  0xFF  /* Pin Interrupt group mask. */
03584 #define PORT_INT_gp  0  /* Pin Interrupt group position. */
03585 #define PORT_INT0_bm  (1<<0)  /* Pin Interrupt bit 0 mask. */
03586 #define PORT_INT0_bp  0  /* Pin Interrupt bit 0 position. */
03587 #define PORT_INT1_bm  (1<<1)  /* Pin Interrupt bit 1 mask. */
```

```
03588 #define PORT_INT1_bp  1  /* Pin Interrupt bit 1 position. */
03589 #define PORT_INT2_bm  (1<<2)  /* Pin Interrupt bit 2 mask. */
03590 #define PORT_INT2_bp  2  /* Pin Interrupt bit 2 position. */
03591 #define PORT_INT3_bm  (1<<3)  /* Pin Interrupt bit 3 mask. */
03592 #define PORT_INT3_bp  3  /* Pin Interrupt bit 3 position. */
03593 #define PORT_INT4_bm  (1<<4)  /* Pin Interrupt bit 4 mask. */
03594 #define PORT_INT4_bp  4  /* Pin Interrupt bit 4 position. */
03595 #define PORT_INT5_bm  (1<<5)  /* Pin Interrupt bit 5 mask. */
03596 #define PORT_INT5_bp  5  /* Pin Interrupt bit 5 position. */
03597 #define PORT_INT6_bm  (1<<6)  /* Pin Interrupt bit 6 mask. */
03598 #define PORT_INT6_bp  6  /* Pin Interrupt bit 6 position. */
03599 #define PORT_INT7_bm  (1<<7)  /* Pin Interrupt bit 7 mask. */
03600 #define PORT_INT7_bp  7  /* Pin Interrupt bit 7 position. */
03601
03602 /* PORT.PIN0CTRL  bit masks and bit positions */
03603 #define PORT_ISC_gm  0x07  /* Input/Sense Configuration group mask. */
03604 #define PORT_ISC_gp  0  /* Input/Sense Configuration group position. */
03605 #define PORT_ISC0_bm  (1<<0)  /* Input/Sense Configuration bit 0 mask. */
03606 #define PORT_ISC0_bp  0  /* Input/Sense Configuration bit 0 position. */
03607 #define PORT_ISC1_bm  (1<<1)  /* Input/Sense Configuration bit 1 mask. */
03608 #define PORT_ISC1_bp  1  /* Input/Sense Configuration bit 1 position. */
03609 #define PORT_ISC2_bm  (1<<2)  /* Input/Sense Configuration bit 2 mask. */
03610 #define PORT_ISC2_bp  2  /* Input/Sense Configuration bit 2 position. */
03611 #define PORT_PULLUPEN_bm  0x08  /* Pullup enable bit mask. */
03612 #define PORT_PULLUPEN_bp  3  /* Pullup enable bit position. */
03613 #define PORT_INVEN_bm  0x80  /* Inverted I/O Enable bit mask. */
03614 #define PORT_INVEN_bp  7  /* Inverted I/O Enable bit position. */
03615
03616 /* PORT.PIN1CTRL  bit masks and bit positions */
03617 /* PORT_ISC  is already defined. */
03618 /* PORT_PULLUPEN  is already defined. */
03619 /* PORT_INVEN  is already defined. */
03620
03621 /* PORT.PIN2CTRL  bit masks and bit positions */
03622 /* PORT_ISC  is already defined. */
03623 /* PORT_PULLUPEN  is already defined. */
03624 /* PORT_INVEN  is already defined. */
03625
03626 /* PORT.PIN3CTRL  bit masks and bit positions */
03627 /* PORT_ISC  is already defined. */
03628 /* PORT_PULLUPEN  is already defined. */
03629 /* PORT_INVEN  is already defined. */
03630
03631 /* PORT.PIN4CTRL  bit masks and bit positions */
03632 /* PORT_ISC  is already defined. */
03633 /* PORT_PULLUPEN  is already defined. */
03634 /* PORT_INVEN  is already defined. */
03635
03636 /* PORT.PIN5CTRL  bit masks and bit positions */
03637 /* PORT_ISC  is already defined. */
03638 /* PORT_PULLUPEN  is already defined. */
03639 /* PORT_INVEN  is already defined. */
03640
03641 /* PORT.PIN6CTRL  bit masks and bit positions */
03642 /* PORT_ISC  is already defined. */
03643 /* PORT_PULLUPEN  is already defined. */
03644 /* PORT_INVEN  is already defined. */
03645
03646 /* PORT.PIN7CTRL  bit masks and bit positions */
03647 /* PORT_ISC  is already defined. */
03648 /* PORT_PULLUPEN  is already defined. */
03649 /* PORT_INVEN  is already defined. */
03650
03651 /* PORTMUX - Port Multiplexer */
03652 /* PORTMUX.CTRLA  bit masks and bit positions */
03653 #define PORTMUX_EVOUT0_bm  0x01  /* Event Output 0 bit mask. */
03654 #define PORTMUX_EVOUT0_bp  0  /* Event Output 0 bit position. */
03655 #define PORTMUX_EVOUT1_bm  0x02  /* Event Output 1 bit mask. */
03656 #define PORTMUX_EVOUT1_bp  1  /* Event Output 1 bit position. */
03657 #define PORTMUX_EVOUT2_bm  0x04  /* Event Output 2 bit mask. */
03658 #define PORTMUX_EVOUT2_bp  2  /* Event Output 2 bit position. */
03659 #define PORTMUX_LUT0_bm  0x10  /* Configurable Custom Logic LUT0 bit mask. */
03660 #define PORTMUX_LUT0_bp  4  /* Configurable Custom Logic LUT0 bit position. */
03661 #define PORTMUX_LUT1_bm  0x20  /* Configurable Custom Logic LUT1 bit mask. */
03662 #define PORTMUX_LUT1_bp  5  /* Configurable Custom Logic LUT1 bit position. */
03663
03664 /* PORTMUX.CTRLB  bit masks and bit positions */
03665 #define PORTMUX_USART0_bm  0x01  /* Port Multiplexer USART0 bit mask. */
03666 #define PORTMUX_USART0_bp  0  /* Port Multiplexer USART0 bit position. */
03667 #define PORTMUX_SPI0_bm  0x04  /* Port Multiplexer SPI0 bit mask. */
03668 #define PORTMUX_SPI0_bp  2  /* Port Multiplexer SPI0 bit position. */
03669 #define PORTMUX_TWI0_bm  0x10  /* Port Multiplexer TWI0 bit mask. */
03670 #define PORTMUX_TWI0_bp  4  /* Port Multiplexer TWI0 bit position. */
03671
03672 /* PORTMUX.CTRLC  bit masks and bit positions */
03673 #define PORTMUX_TCA00_bm  0x01  /* Port Multiplexer TCA0 Output 0 bit mask. */
03674 #define PORTMUX_TCA00_bp  0  /* Port Multiplexer TCA0 Output 0 bit position. */
```

```
03675 #define PORTMUX_TCA01_bm  0x02  /* Port Multiplexer TCA0 Output 1 bit mask. */
03676 #define PORTMUX_TCA01_bp  1  /* Port Multiplexer TCA0 Output 1 bit position. */
03677 #define PORTMUX_TCA02_bm  0x04  /* Port Multiplexer TCA0 Output 2 bit mask. */
03678 #define PORTMUX_TCA02_bp  2  /* Port Multiplexer TCA0 Output 2 bit position. */
03679 #define PORTMUX_TCA03_bm  0x08  /* Port Multiplexer TCA0 Output 3 bit mask. */
03680 #define PORTMUX_TCA03_bp  3  /* Port Multiplexer TCA0 Output 3 bit position. */
03681 #define PORTMUX_TCA04_bm  0x10  /* Port Multiplexer TCA0 Output 4 bit mask. */
03682 #define PORTMUX_TCA04_bp  4  /* Port Multiplexer TCA0 Output 4 bit position. */
03683 #define PORTMUX_TCA05_bm  0x20  /* Port Multiplexer TCA0 Output 5 bit mask. */
03684 #define PORTMUX_TCA05_bp  5  /* Port Multiplexer TCA0 Output 5 bit position. */
03685
03686 /* PORTMUX.CTRLD  bit masks and bit positions */
03687 #define PORTMUX_TCB0_bm  0x01  /* Port Multiplexer TCB bit mask. */
03688 #define PORTMUX_TCB0_bp  0  /* Port Multiplexer TCB bit position. */
03689
03690 /* RSTCTRL - Reset controller */
03691 /* RSTCTRL.RSTFR  bit masks and bit positions */
03692 #define RSTCTRL_PORF_bm  0x01  /* Power on Reset flag bit mask. */
03693 #define RSTCTRL_PORF_bp  0  /* Power on Reset flag bit position. */
03694 #define RSTCTRL_BORF_bm  0x02  /* Brown out detector Reset flag bit mask. */
03695 #define RSTCTRL_BORF_bp  1  /* Brown out detector Reset flag bit position. */
03696 #define RSTCTRL_EXTRF_bm  0x04  /* External Reset flag bit mask. */
03697 #define RSTCTRL_EXTRF_bp  2  /* External Reset flag bit position. */
03698 #define RSTCTRL_WDRF_bm  0x08  /* Watch dog Reset flag bit mask. */
03699 #define RSTCTRL_WDRF_bp  3  /* Watch dog Reset flag bit position. */
03700 #define RSTCTRL_SWRF_bm  0x10  /* Software Reset flag bit mask. */
03701 #define RSTCTRL_SWRF_bp  4  /* Software Reset flag bit position. */
03702 #define RSTCTRL_UPDIRF_bm  0x20  /* UPDI Reset flag bit mask. */
03703 #define RSTCTRL_UPDIRF_bp  5  /* UPDI Reset flag bit position. */
03704
03705 /* RSTCTRL.SWRR  bit masks and bit positions */
03706 #define RSTCTRL_SWRE_bm  0x01  /* Software reset enable bit mask. */
03707 #define RSTCTRL_SWRE_bp  0  /* Software reset enable bit position. */
03708
03709 /* RTC - Real-Time Counter */
03710 /* RTC.CTRLA  bit masks and bit positions */
03711 #define RTC_RTCEN_bm  0x01  /* Enable bit mask. */
03712 #define RTC_RTCEN_bp  0  /* Enable bit position. */
03713 #define RTC_PRESCALER_gm  0x78  /* Prescaling Factor group mask. */
03714 #define RTC_PRESCALER_gp  3  /* Prescaling Factor group position. */
03715 #define RTC_PRESCALER0_bm  (1<<3)  /* Prescaling Factor bit 0 mask. */
03716 #define RTC_PRESCALER0_bp  3  /* Prescaling Factor bit 0 position. */
03717 #define RTC_PRESCALER1_bm  (1<<4)  /* Prescaling Factor bit 1 mask. */
03718 #define RTC_PRESCALER1_bp  4  /* Prescaling Factor bit 1 position. */
03719 #define RTC_PRESCALER2_bm  (1<<5)  /* Prescaling Factor bit 2 mask. */
03720 #define RTC_PRESCALER2_bp  5  /* Prescaling Factor bit 2 position. */
03721 #define RTC_PRESCALER3_bm  (1<<6)  /* Prescaling Factor bit 3 mask. */
03722 #define RTC_PRESCALER3_bp  6  /* Prescaling Factor bit 3 position. */
03723 #define RTC_RUNSTDBY_bm  0x80  /* Run In Standby bit mask. */
03724 #define RTC_RUNSTDBY_bp  7  /* Run In Standby bit position. */
03725
03726 /* RTC.STATUS  bit masks and bit positions */
03727 #define RTC_CTRLABUSY_bm  0x01  /* CTRLA Synchronization Busy Flag bit mask. */
03728 #define RTC_CTRLABUSY_bp  0  /* CTRLA Synchronization Busy Flag bit position. */
03729 #define RTC_CNTBUSY_bm  0x02  /* Count Synchronization Busy Flag bit mask. */
03730 #define RTC_CNTBUSY_bp  1  /* Count Synchronization Busy Flag bit position. */
03731 #define RTC_PERBUSY_bm  0x04  /* Period Synchronization Busy Flag bit mask. */
03732 #define RTC_PERBUSY_bp  2  /* Period Synchronization Busy Flag bit position. */
03733 #define RTC_CMPBUSY_bm  0x08  /* Comparator Synchronization Busy Flag bit mask. */
03734 #define RTC_CMPBUSY_bp  3  /* Comparator Synchronization Busy Flag bit position. */
03735
03736 /* RTC.INTCTRL  bit masks and bit positions */
03737 #define RTC_OVF_bm  0x01  /* Overflow Interrupt enable bit mask. */
03738 #define RTC_OVF_bp  0  /* Overflow Interrupt enable bit position. */
03739 #define RTC_CMP_bm  0x02  /* Compare Match Interrupt enable bit mask. */
03740 #define RTC_CMP_bp  1  /* Compare Match Interrupt enable bit position. */
03741
03742 /* RTC.INTFLAGS  bit masks and bit positions */
03743 /* RTC_OVF  is already defined. */
03744 /* RTC_CMP  is already defined. */
03745
03746 /* RTC.DBGCTRL  bit masks and bit positions */
03747 #define RTC_DBGRUN_bm  0x01  /* Run in debug bit mask. */
03748 #define RTC_DBGRUN_bp  0  /* Run in debug bit position. */
03749
03750 /* RTC.CLKSEL  bit masks and bit positions */
03751 #define RTC_CLKSEL_gm  0x03  /* Clock Select group mask. */
03752 #define RTC_CLKSEL_gp  0  /* Clock Select group position. */
03753 #define RTC_CLKSEL0_bm  (1<<0)  /* Clock Select bit 0 mask. */
03754 #define RTC_CLKSEL0_bp  0  /* Clock Select bit 0 position. */
03755 #define RTC_CLKSEL1_bm  (1<<1)  /* Clock Select bit 1 mask. */
03756 #define RTC_CLKSEL1_bp  1  /* Clock Select bit 1 position. */
03757
03758 /* RTC.PITCTRLA  bit masks and bit positions */
03759 #define RTC_PITEN_bm  0x01  /* Enable bit mask. */
03760 #define RTC_PITEN_bp  0  /* Enable bit position. */
03761 #define RTC_PERIOD_gm  0x78  /* Period group mask. */
```

```
03762 #define RTC_PERIOD_gp  3  /* Period group position. */
03763 #define RTC_PERIOD0_bm  (1<<3)  /* Period bit 0 mask. */
03764 #define RTC_PERIOD0_bp  3  /* Period bit 0 position. */
03765 #define RTC_PERIOD1_bm  (1<<4)  /* Period bit 1 mask. */
03766 #define RTC_PERIOD1_bp  4  /* Period bit 1 position. */
03767 #define RTC_PERIOD2_bm  (1<<5)  /* Period bit 2 mask. */
03768 #define RTC_PERIOD2_bp  5  /* Period bit 2 position. */
03769 #define RTC_PERIOD3_bm  (1<<6)  /* Period bit 3 mask. */
03770 #define RTC_PERIOD3_bp  6  /* Period bit 3 position. */
03771
03772 /* RTC.PITSTATUS  bit masks and bit positions */
03773 #define RTC_CTRLBUSY_bm  0x01  /* CTRLA Synchronization Busy Flag bit mask. */
03774 #define RTC_CTRLBUSY_bp  0  /* CTRLA Synchronization Busy Flag bit position. */
03775
03776 /* RTC.PITINTCTRL  bit masks and bit positions */
03777 #define RTC_PI_bm  0x01  /* Periodic Interrupt bit mask. */
03778 #define RTC_PI_bp  0  /* Periodic Interrupt bit position. */
03779
03780 /* RTC.PITINTFLAGS  bit masks and bit positions */
03781 /* RTC_PI  is already defined. */
03782
03783 /* RTC.PITDBGCTRL  bit masks and bit positions */
03784 /* RTC_DBGRUN  is already defined. */
03785
03786 /* SLPCTRL - Sleep Controller */
03787 /* SLPCTRL.CTRLA  bit masks and bit positions */
03788 #define SLPCTRL_SEN_bm  0x01  /* Sleep enable bit mask. */
03789 #define SLPCTRL_SEN_bp  0  /* Sleep enable bit position. */
03790 #define SLPCTRL_SMODE_gm  0x06  /* Sleep mode group mask. */
03791 #define SLPCTRL_SMODE_gp  1  /* Sleep mode group position. */
03792 #define SLPCTRL_SMODE0_bm  (1<<1)  /* Sleep mode bit 0 mask. */
03793 #define SLPCTRL_SMODE0_bp  1  /* Sleep mode bit 0 position. */
03794 #define SLPCTRL_SMODE1_bm  (1<<2)  /* Sleep mode bit 1 mask. */
03795 #define SLPCTRL_SMODE1_bp  2  /* Sleep mode bit 1 position. */
03796
03797 /* SPI - Serial Peripheral Interface */
03798 /* SPI.CTRLA  bit masks and bit positions */
03799 #define SPI_ENABLE_bm  0x01  /* Enable Module bit mask. */
03800 #define SPI_ENABLE_bp  0  /* Enable Module bit position. */
03801 #define SPI_PRESC_gm  0x06  /* Prescaler group mask. */
03802 #define SPI_PRESC_gp  1  /* Prescaler group position. */
03803 #define SPI_PRESC0_bm  (1<<1)  /* Prescaler bit 0 mask. */
03804 #define SPI_PRESC0_bp  1  /* Prescaler bit 0 position. */
03805 #define SPI_PRESC1_bm  (1<<2)  /* Prescaler bit 1 mask. */
03806 #define SPI_PRESC1_bp  2  /* Prescaler bit 1 position. */
03807 #define SPI_CLK2X_bm  0x10  /* Enable Double Speed bit mask. */
03808 #define SPI_CLK2X_bp  4  /* Enable Double Speed bit position. */
03809 #define SPI_MASTER_bm  0x20  /* Master Operation Enable bit mask. */
03810 #define SPI_MASTER_bp  5  /* Master Operation Enable bit position. */
03811 #define SPI_DORD_bm  0x40  /* Data Order Setting bit mask. */
03812 #define SPI_DORD_bp  6  /* Data Order Setting bit position. */
03813
03814 /* SPI.CTRLB  bit masks and bit positions */
03815 #define SPI_MODE_gm  0x03  /* SPI Mode group mask. */
03816 #define SPI_MODE_gp  0  /* SPI Mode group position. */
03817 #define SPI_MODE0_bm  (1<<0)  /* SPI Mode bit 0 mask. */
03818 #define SPI_MODE0_bp  0  /* SPI Mode bit 0 position. */
03819 #define SPI_MODE1_bm  (1<<1)  /* SPI Mode bit 1 mask. */
03820 #define SPI_MODE1_bp  1  /* SPI Mode bit 1 position. */
03821 #define SPI_SSD_bm  0x04  /* Slave Select Disable bit mask. */
03822 #define SPI_SSD_bp  2  /* Slave Select Disable bit position. */
03823 #define SPI_BUFWR_bm  0x40  /* Buffer Write Mode bit mask. */
03824 #define SPI_BUFWR_bp  6  /* Buffer Write Mode bit position. */
03825 #define SPI_BUFEN_bm  0x80  /* Buffer Mode Enable bit mask. */
03826 #define SPI_BUFEN_bp  7  /* Buffer Mode Enable bit position. */
03827
03828 /* SPI.INTCTRL  bit masks and bit positions */
03829 #define SPI_IE_bm  0x01  /* Interrupt Enable bit mask. */
03830 #define SPI_IE_bp  0  /* Interrupt Enable bit position. */
03831 #define SPI_SSIE_bm  0x10  /* Slave Select Trigger Interrupt Enable bit mask. */
03832 #define SPI_SSIE_bp  4  /* Slave Select Trigger Interrupt Enable bit position. */
03833 #define SPI_DREIE_bm  0x20  /* Data Register Empty Interrupt Enable bit mask. */
03834 #define SPI_DREIE_bp  5  /* Data Register Empty Interrupt Enable bit position. */
03835 #define SPI_TXCIE_bm  0x40  /* Transfer Complete Interrupt Enable bit mask. */
03836 #define SPI_TXCIE_bp  6  /* Transfer Complete Interrupt Enable bit position. */
03837 #define SPI_RXCIE_bm  0x80  /* Receive Complete Interrupt Enable bit mask. */
03838 #define SPI_RXCIE_bp  7  /* Receive Complete Interrupt Enable bit position. */
03839
03840 /* SPI.INTFLAGS  bit masks and bit positions */
03841 #define SPI_BUFOVF_bm  0x01  /* Buffer Overflow bit mask. */
03842 #define SPI_BUFOVF_bp  0  /* Buffer Overflow bit position. */
03843 #define SPI_SSIF_bm  0x10  /* Slave Select Trigger Interrupt Flag bit mask. */
03844 #define SPI_SSIF_bp  4  /* Slave Select Trigger Interrupt Flag bit position. */
03845 #define SPI_DREIF_bm  0x20  /* Data Register Empty Interrupt Flag bit mask. */
03846 #define SPI_DREIF_bp  5  /* Data Register Empty Interrupt Flag bit position. */
03847 #define SPI_TXCIF_bm  0x40  /* Transfer Complete Interrupt Flag bit mask. */
03848 #define SPI_TXCIF_bp  6  /* Transfer Complete Interrupt Flag bit position. */
```

```
03849 #define SPI_WRCOL_bm  0x40  /* Write Collision bit mask. */
03850 #define SPI_WRCOL_bp  6  /* Write Collision bit position. */
03851 #define SPI_RXCIF_bm  0x80  /* Receive Complete Interrupt Flag bit mask. */
03852 #define SPI_RXCIF_bp  7  /* Receive Complete Interrupt Flag bit position. */
03853 #define SPI_IF_bm  0x80  /* Interrupt Flag bit mask. */
03854 #define SPI_IF_bp  7  /* Interrupt Flag bit position. */
03855
03856 /* SYSCFG - System Configuration Registers */
03857 /* SYSCFG.EXTBRK  bit masks and bit positions */
03858 #define SYSCFG_ENEXTBRK_bm  0x01  /* External break enable bit mask. */
03859 #define SYSCFG_ENEXTBRK_bp  0  /* External break enable bit position. */
03860
03861 /* TCA - 16-bit Timer/Counter Type A */
03862 /* TCA_SINGLE.CTRLA  bit masks and bit positions */
03863 #define TCA_SINGLE_ENABLE_bm  0x01  /* Module Enable bit mask. */
03864 #define TCA_SINGLE_ENABLE_bp  0  /* Module Enable bit position. */
03865 #define TCA_SINGLE_CLKSEL_gm  0x0E  /* Clock Selection group mask. */
03866 #define TCA_SINGLE_CLKSEL_gp  1  /* Clock Selection group position. */
03867 #define TCA_SINGLE_CLKSEL0_bm  (1«1)  /* Clock Selection bit 0 mask. */
03868 #define TCA_SINGLE_CLKSEL0_bp  1  /* Clock Selection bit 0 position. */
03869 #define TCA_SINGLE_CLKSEL1_bm  (1«2)  /* Clock Selection bit 1 mask. */
03870 #define TCA_SINGLE_CLKSEL1_bp  2  /* Clock Selection bit 1 position. */
03871 #define TCA_SINGLE_CLKSEL2_bm  (1«3)  /* Clock Selection bit 2 mask. */
03872 #define TCA_SINGLE_CLKSEL2_bp  3  /* Clock Selection bit 2 position. */
03873
03874 /* TCA_SINGLE.CTRLB  bit masks and bit positions */
03875 #define TCA_SINGLE_WGMODE_gm  0x07  /* Waveform generation mode group mask. */
03876 #define TCA_SINGLE_WGMODE_gp  0  /* Waveform generation mode group position. */
03877 #define TCA_SINGLE_WGMODE0_bm  (1«0)  /* Waveform generation mode bit 0 mask. */
03878 #define TCA_SINGLE_WGMODE0_bp  0  /* Waveform generation mode bit 0 position. */
03879 #define TCA_SINGLE_WGMODE1_bm  (1«1)  /* Waveform generation mode bit 1 mask. */
03880 #define TCA_SINGLE_WGMODE1_bp  1  /* Waveform generation mode bit 1 position. */
03881 #define TCA_SINGLE_WGMODE2_bm  (1«2)  /* Waveform generation mode bit 2 mask. */
03882 #define TCA_SINGLE_WGMODE2_bp  2  /* Waveform generation mode bit 2 position. */
03883 #define TCA_SINGLE_ALUPD_bm  0x08  /* Auto Lock Update bit mask. */
03884 #define TCA_SINGLE_ALUPD_bp  3  /* Auto Lock Update bit position. */
03885 #define TCA_SINGLE_CMP0EN_bm  0x10  /* Compare 0 Enable bit mask. */
03886 #define TCA_SINGLE_CMP0EN_bp  4  /* Compare 0 Enable bit position. */
03887 #define TCA_SINGLE_CMP1EN_bm  0x20  /* Compare 1 Enable bit mask. */
03888 #define TCA_SINGLE_CMP1EN_bp  5  /* Compare 1 Enable bit position. */
03889 #define TCA_SINGLE_CMP2EN_bm  0x40  /* Compare 2 Enable bit mask. */
03890 #define TCA_SINGLE_CMP2EN_bp  6  /* Compare 2 Enable bit position. */
03891
03892 /* TCA_SINGLE.CTRLC  bit masks and bit positions */
03893 #define TCA_SINGLE_CMP0OV_bm  0x01  /* Compare 0 Waveform Output Value bit mask. */
03894 #define TCA_SINGLE_CMP0OV_bp  0  /* Compare 0 Waveform Output Value bit position. */
03895 #define TCA_SINGLE_CMP1OV_bm  0x02  /* Compare 1 Waveform Output Value bit mask. */
03896 #define TCA_SINGLE_CMP1OV_bp  1  /* Compare 1 Waveform Output Value bit position. */
03897 #define TCA_SINGLE_CMP2OV_bm  0x04  /* Compare 2 Waveform Output Value bit mask. */
03898 #define TCA_SINGLE_CMP2OV_bp  2  /* Compare 2 Waveform Output Value bit position. */
03899
03900 /* TCA_SINGLE.CTRLD  bit masks and bit positions */
03901 #define TCA_SINGLE_SPLITM_bm  0x01  /* Split Mode Enable bit mask. */
03902 #define TCA_SINGLE_SPLITM_bp  0  /* Split Mode Enable bit position. */
03903
03904 /* TCA_SINGLE.CTRLECLR  bit masks and bit positions */
03905 #define TCA_SINGLE_DIR_bm  0x01  /* Direction bit mask. */
03906 #define TCA_SINGLE_DIR_bp  0  /* Direction bit position. */
03907 #define TCA_SINGLE_LUPD_bm  0x02  /* Lock Update bit mask. */
03908 #define TCA_SINGLE_LUPD_bp  1  /* Lock Update bit position. */
03909 #define TCA_SINGLE_CMD_gm  0x0C  /* Command group mask. */
03910 #define TCA_SINGLE_CMD_gp  2  /* Command group position. */
03911 #define TCA_SINGLE_CMD0_bm  (1«2)  /* Command bit 0 mask. */
03912 #define TCA_SINGLE_CMD0_bp  2  /* Command bit 0 position. */
03913 #define TCA_SINGLE_CMD1_bm  (1«3)  /* Command bit 1 mask. */
03914 #define TCA_SINGLE_CMD1_bp  3  /* Command bit 1 position. */
03915
03916 /* TCA_SINGLE.CTRLESET  bit masks and bit positions */
03917 /* TCA_SINGLE_DIR  is already defined. */
03918 /* TCA_SINGLE_LUPD  is already defined. */
03919 /* TCA_SINGLE_CMD  is already defined. */
03920
03921 /* TCA_SINGLE.CTRLFCLR  bit masks and bit positions */
03922 #define TCA_SINGLE_PERBV_bm  0x01  /* Period Buffer Valid bit mask. */
03923 #define TCA_SINGLE_PERBV_bp  0  /* Period Buffer Valid bit position. */
03924 #define TCA_SINGLE_CMP0BV_bm  0x02  /* Compare 0 Buffer Valid bit mask. */
03925 #define TCA_SINGLE_CMP0BV_bp  1  /* Compare 0 Buffer Valid bit position. */
03926 #define TCA_SINGLE_CMP1BV_bm  0x04  /* Compare 1 Buffer Valid bit mask. */
03927 #define TCA_SINGLE_CMP1BV_bp  2  /* Compare 1 Buffer Valid bit position. */
03928 #define TCA_SINGLE_CMP2BV_bm  0x08  /* Compare 2 Buffer Valid bit mask. */
03929 #define TCA_SINGLE_CMP2BV_bp  3  /* Compare 2 Buffer Valid bit position. */
03930
03931 /* TCA_SINGLE.CTRLFSET  bit masks and bit positions */
03932 /* TCA_SINGLE_PERBV  is already defined. */
03933 /* TCA_SINGLE_CMP0BV  is already defined. */
03934 /* TCA_SINGLE_CMP1BV  is already defined. */
03935 /* TCA_SINGLE_CMP2BV  is already defined. */
```

```
03936
03937 /* TCA_SINGLE.EVCTRL  bit masks and bit positions */
03938 #define TCA_SINGLE_CNTEI_bm  0x01  /* Count on Event Input bit mask. */
03939 #define TCA_SINGLE_CNTEI_bp  0  /* Count on Event Input bit position. */
03940 #define TCA_SINGLE_EVACT_gm  0x06  /* Event Action group mask. */
03941 #define TCA_SINGLE_EVACT_gp  1  /* Event Action group position. */
03942 #define TCA_SINGLE_EVACT0_bm  (1<<1)  /* Event Action bit 0 mask. */
03943 #define TCA_SINGLE_EVACT0_bp  1  /* Event Action bit 0 position. */
03944 #define TCA_SINGLE_EVACT1_bm  (1<<2)  /* Event Action bit 1 mask. */
03945 #define TCA_SINGLE_EVACT1_bp  2  /* Event Action bit 1 position. */
03946
03947 /* TCA_SINGLE.INTCTRL  bit masks and bit positions */
03948 #define TCA_SINGLE_OVF_bm  0x01  /* Overflow Interrupt bit mask. */
03949 #define TCA_SINGLE_OVF_bp  0  /* Overflow Interrupt bit position. */
03950 #define TCA_SINGLE_CMP0_bm  0x10  /* Compare 0 Interrupt bit mask. */
03951 #define TCA_SINGLE_CMP0_bp  4  /* Compare 0 Interrupt bit position. */
03952 #define TCA_SINGLE_CMP1_bm  0x20  /* Compare 1 Interrupt bit mask. */
03953 #define TCA_SINGLE_CMP1_bp  5  /* Compare 1 Interrupt bit position. */
03954 #define TCA_SINGLE_CMP2_bm  0x40  /* Compare 2 Interrupt bit mask. */
03955 #define TCA_SINGLE_CMP2_bp  6  /* Compare 2 Interrupt bit position. */
03956
03957 /* TCA_SINGLE.INTFLAGS  bit masks and bit positions */
03958 /* TCA_SINGLE_OVF  is already defined. */
03959 /* TCA_SINGLE_CMP0  is already defined. */
03960 /* TCA_SINGLE_CMP1  is already defined. */
03961 /* TCA_SINGLE_CMP2  is already defined. */
03962
03963 /* TCA_SINGLE.DBGCTRL  bit masks and bit positions */
03964 #define TCA_SINGLE_DBGRUN_bm  0x01  /* Debug Run bit mask. */
03965 #define TCA_SINGLE_DBGRUN_bp  0  /* Debug Run bit position. */
03966
03967 /* TCA_SPLIT.CTRLA  bit masks and bit positions */
03968 #define TCA_SPLIT_ENABLE_bm  0x01  /* Module Enable bit mask. */
03969 #define TCA_SPLIT_ENABLE_bp  0  /* Module Enable bit position. */
03970 #define TCA_SPLIT_CLKSEL_gm  0x0E  /* Clock Selection group mask. */
03971 #define TCA_SPLIT_CLKSEL_gp  1  /* Clock Selection group position. */
03972 #define TCA_SPLIT_CLKSEL0_bm  (1<<1)  /* Clock Selection bit 0 mask. */
03973 #define TCA_SPLIT_CLKSEL0_bp  1  /* Clock Selection bit 0 position. */
03974 #define TCA_SPLIT_CLKSEL1_bm  (1<<2)  /* Clock Selection bit 1 mask. */
03975 #define TCA_SPLIT_CLKSEL1_bp  2  /* Clock Selection bit 1 position. */
03976 #define TCA_SPLIT_CLKSEL2_bm  (1<<3)  /* Clock Selection bit 2 mask. */
03977 #define TCA_SPLIT_CLKSEL2_bp  3  /* Clock Selection bit 2 position. */
03978
03979 /* TCA_SPLIT.CTRLB  bit masks and bit positions */
03980 #define TCA_SPLIT_LCMP0EN_bm  0x01  /* Low Compare 0 Enable bit mask. */
03981 #define TCA_SPLIT_LCMP0EN_bp  0  /* Low Compare 0 Enable bit position. */
03982 #define TCA_SPLIT_LCMP1EN_bm  0x02  /* Low Compare 1 Enable bit mask. */
03983 #define TCA_SPLIT_LCMP1EN_bp  1  /* Low Compare 1 Enable bit position. */
03984 #define TCA_SPLIT_LCMP2EN_bm  0x04  /* Low Compare 2 Enable bit mask. */
03985 #define TCA_SPLIT_LCMP2EN_bp  2  /* Low Compare 2 Enable bit position. */
03986 #define TCA_SPLIT_HCMP0EN_bm  0x10  /* High Compare 0 Enable bit mask. */
03987 #define TCA_SPLIT_HCMP0EN_bp  4  /* High Compare 0 Enable bit position. */
03988 #define TCA_SPLIT_HCMP1EN_bm  0x20  /* High Compare 1 Enable bit mask. */
03989 #define TCA_SPLIT_HCMP1EN_bp  5  /* High Compare 1 Enable bit position. */
03990 #define TCA_SPLIT_HCMP2EN_bm  0x40  /* High Compare 2 Enable bit mask. */
03991 #define TCA_SPLIT_HCMP2EN_bp  6  /* High Compare 2 Enable bit position. */
03992
03993 /* TCA_SPLIT.CTRLC  bit masks and bit positions */
03994 #define TCA_SPLIT_LCMP0OV_bm  0x01  /* Low Compare 0 Output Value bit mask. */
03995 #define TCA_SPLIT_LCMP0OV_bp  0  /* Low Compare 0 Output Value bit position. */
03996 #define TCA_SPLIT_LCMP1OV_bm  0x02  /* Low Compare 1 Output Value bit mask. */
03997 #define TCA_SPLIT_LCMP1OV_bp  1  /* Low Compare 1 Output Value bit position. */
03998 #define TCA_SPLIT_LCMP2OV_bm  0x04  /* Low Compare 2 Output Value bit mask. */
03999 #define TCA_SPLIT_LCMP2OV_bp  2  /* Low Compare 2 Output Value bit position. */
04000 #define TCA_SPLIT_HCMP0OV_bm  0x10  /* High Compare 0 Output Value bit mask. */
04001 #define TCA_SPLIT_HCMP0OV_bp  4  /* High Compare 0 Output Value bit position. */
04002 #define TCA_SPLIT_HCMP1OV_bm  0x20  /* High Compare 1 Output Value bit mask. */
04003 #define TCA_SPLIT_HCMP1OV_bp  5  /* High Compare 1 Output Value bit position. */
04004 #define TCA_SPLIT_HCMP2OV_bm  0x40  /* High Compare 2 Output Value bit mask. */
04005 #define TCA_SPLIT_HCMP2OV_bp  6  /* High Compare 2 Output Value bit position. */
04006
04007 /* TCA_SPLIT.CTRLD  bit masks and bit positions */
04008 #define TCA_SPLIT_SPLITM_bm  0x01  /* Split Mode Enable bit mask. */
04009 #define TCA_SPLIT_SPLITM_bp  0  /* Split Mode Enable bit position. */
04010
04011 /* TCA_SPLIT.CTRLECLR  bit masks and bit positions */
04012 #define TCA_SPLIT_CMD_gm  0x0C  /* Command group mask. */
04013 #define TCA_SPLIT_CMD_gp  2  /* Command group position. */
04014 #define TCA_SPLIT_CMD0_bm  (1<<2)  /* Command bit 0 mask. */
04015 #define TCA_SPLIT_CMD0_bp  2  /* Command bit 0 position. */
04016 #define TCA_SPLIT_CMD1_bm  (1<<3)  /* Command bit 1 mask. */
04017 #define TCA_SPLIT_CMD1_bp  3  /* Command bit 1 position. */
04018
04019 /* TCA_SPLIT.CTRLESET  bit masks and bit positions */
04020 /* TCA_SPLIT_CMD  is already defined. */
04021
04022 /* TCA_SPLIT.INTCTRL  bit masks and bit positions */
```

```
04023 #define TCA_SPLIT_LUNF_bm  0x01  /* Low Underflow Interrupt Enable bit mask. */
04024 #define TCA_SPLIT_LUNF_bp  0  /* Low Underflow Interrupt Enable bit position. */
04025 #define TCA_SPLIT_HUNF_bm  0x02  /* High Underflow Interrupt Enable bit mask. */
04026 #define TCA_SPLIT_HUNF_bp  1  /* High Underflow Interrupt Enable bit position. */
04027 #define TCA_SPLIT_LCMP0_bm  0x10  /* Low Compare 0 Interrupt Enable bit mask. */
04028 #define TCA_SPLIT_LCMP0_bp  4  /* Low Compare 0 Interrupt Enable bit position. */
04029 #define TCA_SPLIT_LCMP1_bm  0x20  /* Low Compare 1 Interrupt Enable bit mask. */
04030 #define TCA_SPLIT_LCMP1_bp  5  /* Low Compare 1 Interrupt Enable bit position. */
04031 #define TCA_SPLIT_LCMP2_bm  0x40  /* Low Compare 2 Interrupt Enable bit mask. */
04032 #define TCA_SPLIT_LCMP2_bp  6  /* Low Compare 2 Interrupt Enable bit position. */
04033
04034 /* TCA_SPLIT.INTFLAGS  bit masks and bit positions */
04035 /* TCA_SPLIT_LUNF  is already defined. */
04036 /* TCA_SPLIT_HUNF  is already defined. */
04037 /* TCA_SPLIT_LCMP0  is already defined. */
04038 /* TCA_SPLIT_LCMP1  is already defined. */
04039 /* TCA_SPLIT_LCMP2  is already defined. */
04040
04041 /* TCA_SPLIT.DBGCTRL  bit masks and bit positions */
04042 #define TCA_SPLIT_DBGRUN_bm  0x01  /* Debug Run bit mask. */
04043 #define TCA_SPLIT_DBGRUN_bp  0  /* Debug Run bit position. */
04044
04045 /* TCB - 16-bit Timer Type B */
04046 /* TCB.CTRLA  bit masks and bit positions */
04047 #define TCB_ENABLE_bm  0x01  /* Enable bit mask. */
04048 #define TCB_ENABLE_bp  0  /* Enable bit position. */
04049 #define TCB_CLKSEL_gm  0x06  /* Clock Select group mask. */
04050 #define TCB_CLKSEL_gp  1  /* Clock Select group position. */
04051 #define TCB_CLKSEL0_bm  (1«1)  /* Clock Select bit 0 mask. */
04052 #define TCB_CLKSEL0_bp  1  /* Clock Select bit 0 position. */
04053 #define TCB_CLKSEL1_bm  (1«2)  /* Clock Select bit 1 mask. */
04054 #define TCB_CLKSEL1_bp  2  /* Clock Select bit 1 position. */
04055 #define TCB_SYNCUPD_bm  0x10  /* Synchronize Update bit mask. */
04056 #define TCB_SYNCUPD_bp  4  /* Synchronize Update bit position. */
04057 #define TCB_RUNSTDBY_bm  0x40  /* Run Standby bit mask. */
04058 #define TCB_RUNSTDBY_bp  6  /* Run Standby bit position. */
04059
04060 /* TCB.CTRLB  bit masks and bit positions */
04061 #define TCB_CNTMODE_gm  0x07  /* Timer Mode group mask. */
04062 #define TCB_CNTMODE_gp  0  /* Timer Mode group position. */
04063 #define TCB_CNTMODE0_bm  (1«0)  /* Timer Mode bit 0 mask. */
04064 #define TCB_CNTMODE0_bp  0  /* Timer Mode bit 0 position. */
04065 #define TCB_CNTMODE1_bm  (1«1)  /* Timer Mode bit 1 mask. */
04066 #define TCB_CNTMODE1_bp  1  /* Timer Mode bit 1 position. */
04067 #define TCB_CNTMODE2_bm  (1«2)  /* Timer Mode bit 2 mask. */
04068 #define TCB_CNTMODE2_bp  2  /* Timer Mode bit 2 position. */
04069 #define TCB_CCMPEN_bm  0x10  /* Pin Output Enable bit mask. */
04070 #define TCB_CCMPEN_bp  4  /* Pin Output Enable bit position. */
04071 #define TCB_CCMPINIT_bm  0x20  /* Pin Initial State bit mask. */
04072 #define TCB_CCMPINIT_bp  5  /* Pin Initial State bit position. */
04073 #define TCB_ASYNC_bm  0x40  /* Asynchronous Enable bit mask. */
04074 #define TCB_ASYNC_bp  6  /* Asynchronous Enable bit position. */
04075
04076 /* TCB.EVCTRL  bit masks and bit positions */
04077 #define TCB_CAPTEI_bm  0x01  /* Event Input Enable bit mask. */
04078 #define TCB_CAPTEI_bp  0  /* Event Input Enable bit position. */
04079 #define TCB_EDGE_bm  0x10  /* Event Edge bit mask. */
04080 #define TCB_EDGE_bp  4  /* Event Edge bit position. */
04081 #define TCB_FILTER_bm  0x40  /* Input Capture Noise Cancellation Filter bit mask. */
04082 #define TCB_FILTER_bp  6  /* Input Capture Noise Cancellation Filter bit position. */
04083
04084 /* TCB.INTCTRL  bit masks and bit positions */
04085 #define TCB_CAPT_bm  0x01  /* Capture or Timeout bit mask. */
04086 #define TCB_CAPT_bp  0  /* Capture or Timeout bit position. */
04087
04088 /* TCB.INTFLAGS  bit masks and bit positions */
04089 /* TCB_CAPT  is already defined. */
04090
04091 /* TCB.STATUS  bit masks and bit positions */
04092 #define TCB_RUN_bm  0x01  /* Run bit mask. */
04093 #define TCB_RUN_bp  0  /* Run bit position. */
04094
04095 /* TCB.DBGCTRL  bit masks and bit positions */
04096 #define TCB_DBGRUN_bm  0x01  /* Debug Run bit mask. */
04097 #define TCB_DBGRUN_bp  0  /* Debug Run bit position. */
04098
04099 /* TWI - Two-Wire Interface */
04100 /* TWI.CTRLA  bit masks and bit positions */
04101 #define TWI_FMPEN_bm  0x02  /* FM Plus Enable bit mask. */
04102 #define TWI_FMPEN_bp  1  /* FM Plus Enable bit position. */
04103 #define TWI_SDAHOLD_gm  0x0C  /* SDA Hold Time group mask. */
04104 #define TWI_SDAHOLD_gp  2  /* SDA Hold Time group position. */
04105 #define TWI_SDAHOLD0_bm  (1«2)  /* SDA Hold Time bit 0 mask. */
04106 #define TWI_SDAHOLD0_bp  2  /* SDA Hold Time bit 0 position. */
04107 #define TWI_SDAHOLD1_bm  (1«3)  /* SDA Hold Time bit 1 mask. */
04108 #define TWI_SDAHOLD1_bp  3  /* SDA Hold Time bit 1 position. */
04109 #define TWI_SDASETUP_bm  0x10  /* SDA Setup Time bit mask. */
```

```
04110 #define TWI_SDASETUP_bp  4  /* SDA Setup Time bit position. */
04111
04112 /* TWI.DBGCTRL  bit masks and bit positions */
04113 #define TWI_DBGRUN_bm  0x01  /* Debug Run bit mask. */
04114 #define TWI_DBGRUN_bp  0  /* Debug Run bit position. */
04115
04116 /* TWI.MCTRLA  bit masks and bit positions */
04117 #define TWI_ENABLE_bm  0x01  /* Enable TWI Master bit mask. */
04118 #define TWI_ENABLE_bp  0  /* Enable TWI Master bit position. */
04119 #define TWI_SMEN_bm  0x02  /* Smart Mode Enable bit mask. */
04120 #define TWI_SMEN_bp  1  /* Smart Mode Enable bit position. */
04121 #define TWI_TIMEOUT_gm  0x0C  /* Inactive Bus Timeout group mask. */
04122 #define TWI_TIMEOUT_gp  2  /* Inactive Bus Timeout group position. */
04123 #define TWI_TIMEOUT0_bm  (1<<2)  /* Inactive Bus Timeout bit 0 mask. */
04124 #define TWI_TIMEOUT0_bp  2  /* Inactive Bus Timeout bit 0 position. */
04125 #define TWI_TIMEOUT1_bm  (1<<3)  /* Inactive Bus Timeout bit 1 mask. */
04126 #define TWI_TIMEOUT1_bp  3  /* Inactive Bus Timeout bit 1 position. */
04127 #define TWI_QCEN_bm  0x10  /* Quick Command Enable bit mask. */
04128 #define TWI_QCEN_bp  4  /* Quick Command Enable bit position. */
04129 #define TWI_WIEN_bm  0x40  /* Write Interrupt Enable bit mask. */
04130 #define TWI_WIEN_bp  6  /* Write Interrupt Enable bit position. */
04131 #define TWI_RIEN_bm  0x80  /* Read Interrupt Enable bit mask. */
04132 #define TWI_RIEN_bp  7  /* Read Interrupt Enable bit position. */
04133
04134 /* TWI.MCTRLB  bit masks and bit positions */
04135 #define TWI_MCMD_gm  0x03  /* Command group mask. */
04136 #define TWI_MCMD_gp  0  /* Command group position. */
04137 #define TWI_MCMD0_bm  (1<<0)  /* Command bit 0 mask. */
04138 #define TWI_MCMD0_bp  0  /* Command bit 0 position. */
04139 #define TWI_MCMD1_bm  (1<<1)  /* Command bit 1 mask. */
04140 #define TWI_MCMD1_bp  1  /* Command bit 1 position. */
04141 #define TWI_ACKACT_bm  0x04  /* Acknowledge Action bit mask. */
04142 #define TWI_ACKACT_bp  2  /* Acknowledge Action bit position. */
04143 #define TWI_FLUSH_bm  0x08  /* Flush bit mask. */
04144 #define TWI_FLUSH_bp  3  /* Flush bit position. */
04145
04146 /* TWI.MSTATUS  bit masks and bit positions */
04147 #define TWI_BUSSTATE_gm  0x03  /* Bus State group mask. */
04148 #define TWI_BUSSTATE_gp  0  /* Bus State group position. */
04149 #define TWI_BUSSTATE0_bm  (1<<0)  /* Bus State bit 0 mask. */
04150 #define TWI_BUSSTATE0_bp  0  /* Bus State bit 0 position. */
04151 #define TWI_BUSSTATE1_bm  (1<<1)  /* Bus State bit 1 mask. */
04152 #define TWI_BUSSTATE1_bp  1  /* Bus State bit 1 position. */
04153 #define TWI_BUSERR_bm  0x04  /* Bus Error bit mask. */
04154 #define TWI_BUSERR_bp  2  /* Bus Error bit position. */
04155 #define TWI_ARBLOST_bm  0x08  /* Arbitration Lost bit mask. */
04156 #define TWI_ARBLOST_bp  3  /* Arbitration Lost bit position. */
04157 #define TWI_RXACK_bm  0x10  /* Received Acknowledge bit mask. */
04158 #define TWI_RXACK_bp  4  /* Received Acknowledge bit position. */
04159 #define TWI_CLKHOLD_bm  0x20  /* Clock Hold bit mask. */
04160 #define TWI_CLKHOLD_bp  5  /* Clock Hold bit position. */
04161 #define TWI_WIF_bm  0x40  /* Write Interrupt Flag bit mask. */
04162 #define TWI_WIF_bp  6  /* Write Interrupt Flag bit position. */
04163 #define TWI_RIF_bm  0x80  /* Read Interrupt Flag bit mask. */
04164 #define TWI_RIF_bp  7  /* Read Interrupt Flag bit position. */
04165
04166 /* TWI.SCTRLA  bit masks and bit positions */
04167 /* TWI_ENABLE  is already defined. */
04168 /* TWI_SMEN  is already defined. */
04169 #define TWI_PMEN_bm  0x04  /* Promiscuous Mode Enable bit mask. */
04170 #define TWI_PMEN_bp  2  /* Promiscuous Mode Enable bit position. */
04171 #define TWI_PIEN_bm  0x20  /* Stop Interrupt Enable bit mask. */
04172 #define TWI_PIEN_bp  5  /* Stop Interrupt Enable bit position. */
04173 #define TWI_APIEN_bm  0x40  /* Address/Stop Interrupt Enable bit mask. */
04174 #define TWI_APIEN_bp  6  /* Address/Stop Interrupt Enable bit position. */
04175 #define TWI_DIEN_bm  0x80  /* Data Interrupt Enable bit mask. */
04176 #define TWI_DIEN_bp  7  /* Data Interrupt Enable bit position. */
04177
04178 /* TWI.SCTRLB  bit masks and bit positions */
04179 #define TWI_SCMD_gm  0x03  /* Command group mask. */
04180 #define TWI_SCMD_gp  0  /* Command group position. */
04181 #define TWI_SCMD0_bm  (1<<0)  /* Command bit 0 mask. */
04182 #define TWI_SCMD0_bp  0  /* Command bit 0 position. */
04183 #define TWI_SCMD1_bm  (1<<1)  /* Command bit 1 mask. */
04184 #define TWI_SCMD1_bp  1  /* Command bit 1 position. */
04185 /* TWI_ACKACT  is already defined. */
04186
04187 /* TWI.SSTATUS  bit masks and bit positions */
04188 #define TWI_AP_bm  0x01  /* Slave Address or Stop bit mask. */
04189 #define TWI_AP_bp  0  /* Slave Address or Stop bit position. */
04190 #define TWI_DIR_bm  0x02  /* Read/Write Direction bit mask. */
04191 #define TWI_DIR_bp  1  /* Read/Write Direction bit position. */
04192 /* TWI_BUSERR  is already defined. */
04193 #define TWI_COLL_bm  0x08  /* Collision bit mask. */
04194 #define TWI_COLL_bp  3  /* Collision bit position. */
04195 /* TWI_RXACK  is already defined. */
04196 /* TWI_CLKHOLD  is already defined. */
```

```
04197 #define TWI_APIF_bm  0x40  /* Address/Stop Interrupt Flag bit mask. */
04198 #define TWI_APIF_bp  6  /* Address/Stop Interrupt Flag bit position. */
04199 #define TWI_DIF_bm  0x80  /* Data Interrupt Flag bit mask. */
04200 #define TWI_DIF_bp  7  /* Data Interrupt Flag bit position. */
04201
04202 /* TWI.SADDRMASK  bit masks and bit positions */
04203 #define TWI_ADDREN_bm  0x01  /* Address Enable bit mask. */
04204 #define TWI_ADDREN_bp  0  /* Address Enable bit position. */
04205 #define TWI_ADDRMASK_gm  0xFE  /* Address Mask group mask. */
04206 #define TWI_ADDRMASK_gp  1  /* Address Mask group position. */
04207 #define TWI_ADDRMASK0_bm  (1<<1)  /* Address Mask bit 0 mask. */
04208 #define TWI_ADDRMASK0_bp  1  /* Address Mask bit 0 position. */
04209 #define TWI_ADDRMASK1_bm  (1<<2)  /* Address Mask bit 1 mask. */
04210 #define TWI_ADDRMASK1_bp  2  /* Address Mask bit 1 position. */
04211 #define TWI_ADDRMASK2_bm  (1<<3)  /* Address Mask bit 2 mask. */
04212 #define TWI_ADDRMASK2_bp  3  /* Address Mask bit 2 position. */
04213 #define TWI_ADDRMASK3_bm  (1<<4)  /* Address Mask bit 3 mask. */
04214 #define TWI_ADDRMASK3_bp  4  /* Address Mask bit 3 position. */
04215 #define TWI_ADDRMASK4_bm  (1<<5)  /* Address Mask bit 4 mask. */
04216 #define TWI_ADDRMASK4_bp  5  /* Address Mask bit 4 position. */
04217 #define TWI_ADDRMASK5_bm  (1<<6)  /* Address Mask bit 5 mask. */
04218 #define TWI_ADDRMASK5_bp  6  /* Address Mask bit 5 position. */
04219 #define TWI_ADDRMASK6_bm  (1<<7)  /* Address Mask bit 6 mask. */
04220 #define TWI_ADDRMASK6_bp  7  /* Address Mask bit 6 position. */
04221
04222 /* USART - Universal Synchronous and Asynchronous Receiver and Transmitter */
04223 /* USART.RXDATAL  bit masks and bit positions */
04224 #define USART_DATA_gm  0xFF  /* RX Data group mask. */
04225 #define USART_DATA_gp  0  /* RX Data group position. */
04226 #define USART_DATA0_bm  (1<<0)  /* RX Data bit 0 mask. */
04227 #define USART_DATA0_bp  0  /* RX Data bit 0 position. */
04228 #define USART_DATA1_bm  (1<<1)  /* RX Data bit 1 mask. */
04229 #define USART_DATA1_bp  1  /* RX Data bit 1 position. */
04230 #define USART_DATA2_bm  (1<<2)  /* RX Data bit 2 mask. */
04231 #define USART_DATA2_bp  2  /* RX Data bit 2 position. */
04232 #define USART_DATA3_bm  (1<<3)  /* RX Data bit 3 mask. */
04233 #define USART_DATA3_bp  3  /* RX Data bit 3 position. */
04234 #define USART_DATA4_bm  (1<<4)  /* RX Data bit 4 mask. */
04235 #define USART_DATA4_bp  4  /* RX Data bit 4 position. */
04236 #define USART_DATA5_bm  (1<<5)  /* RX Data bit 5 mask. */
04237 #define USART_DATA5_bp  5  /* RX Data bit 5 position. */
04238 #define USART_DATA6_bm  (1<<6)  /* RX Data bit 6 mask. */
04239 #define USART_DATA6_bp  6  /* RX Data bit 6 position. */
04240 #define USART_DATA7_bm  (1<<7)  /* RX Data bit 7 mask. */
04241 #define USART_DATA7_bp  7  /* RX Data bit 7 position. */
04242
04243 /* USART.RXDATAH  bit masks and bit positions */
04244 #define USART_DATA8_bm  0x01  /* Receiver Data Register bit mask. */
04245 #define USART_DATA8_bp  0  /* Receiver Data Register bit position. */
04246 #define USART_PERR_bm  0x02  /* Parity Error bit mask. */
04247 #define USART_PERR_bp  1  /* Parity Error bit position. */
04248 #define USART_FERR_bm  0x04  /* Frame Error bit mask. */
04249 #define USART_FERR_bp  2  /* Frame Error bit position. */
04250 #define USART_BUFOVF_bm  0x40  /* Buffer Overflow bit mask. */
04251 #define USART_BUFOVF_bp  6  /* Buffer Overflow bit position. */
04252 #define USART_RXCIF_bm  0x80  /* Receive Complete Interrupt Flag bit mask. */
04253 #define USART_RXCIF_bp  7  /* Receive Complete Interrupt Flag bit position. */
04254
04255 /* USART.TXDATAL  bit masks and bit positions */
04256 /* USART_DATA  is already defined. */
04257
04258 /* USART.TXDATAH  bit masks and bit positions */
04259 /* USART_DATA8  is already defined. */
04260
04261 /* USART.STATUS  bit masks and bit positions */
04262 #define USART_WFB_bm  0x01  /* Wait For Break bit mask. */
04263 #define USART_WFB_bp  0  /* Wait For Break bit position. */
04264 #define USART_BDF_bm  0x02  /* Break Detected Flag bit mask. */
04265 #define USART_BDF_bp  1  /* Break Detected Flag bit position. */
04266 #define USART_ISFIF_bm  0x08  /* Inconsistent Sync Field Interrupt Flag bit mask. */
04267 #define USART_ISFIF_bp  3  /* Inconsistent Sync Field Interrupt Flag bit position. */
04268 #define USART_RXSIF_bm  0x10  /* Receive Start Interrupt bit mask. */
04269 #define USART_RXSIF_bp  4  /* Receive Start Interrupt bit position. */
04270 #define USART_DREIF_bm  0x20  /* Data Register Empty Flag bit mask. */
04271 #define USART_DREIF_bp  5  /* Data Register Empty Flag bit position. */
04272 #define USART_TXCIF_bm  0x40  /* Transmit Interrupt Flag bit mask. */
04273 #define USART_TXCIF_bp  6  /* Transmit Interrupt Flag bit position. */
04274 /* USART_RXCIF  is already defined. */
04275
04276 /* USART.CTRLA  bit masks and bit positions */
04277 #define USART_RS485_gm  0x03  /* RS485 Mode internal transmitter group mask. */
04278 #define USART_RS485_gp  0  /* RS485 Mode internal transmitter group position. */
04279 #define USART_RS4850_bm  (1<<0)  /* RS485 Mode internal transmitter bit 0 mask. */
04280 #define USART_RS4850_bp  0  /* RS485 Mode internal transmitter bit 0 position. */
04281 #define USART_RS4851_bm  (1<<1)  /* RS485 Mode internal transmitter bit 1 mask. */
04282 #define USART_RS4851_bp  1  /* RS485 Mode internal transmitter bit 1 position. */
04283 #define USART_ABEIE_bm  0x04  /* Auto-baud Error Interrupt Enable bit mask. */
```

```
04284 #define USART_ABEIE_bp  2  /* Auto-baud Error Interrupt Enable bit position. */
04285 #define USART_LBME_bm  0x08  /* Loop-back Mode Enable bit mask. */
04286 #define USART_LBME_bp  3  /* Loop-back Mode Enable bit position. */
04287 #define USART_RXSIE_bm  0x10  /* Receiver Start Frame Interrupt Enable bit mask. */
04288 #define USART_RXSIE_bp  4  /* Receiver Start Frame Interrupt Enable bit position. */
04289 #define USART_DREIE_bm  0x20  /* Data Register Empty Interrupt Enable bit mask. */
04290 #define USART_DREIE_bp  5  /* Data Register Empty Interrupt Enable bit position. */
04291 #define USART_TXCIE_bm  0x40  /* Transmit Complete Interrupt Enable bit mask. */
04292 #define USART_TXCIE_bp  6  /* Transmit Complete Interrupt Enable bit position. */
04293 #define USART_RXCIE_bm  0x80  /* Receive Complete Interrupt Enable bit mask. */
04294 #define USART_RXCIE_bp  7  /* Receive Complete Interrupt Enable bit position. */
04295
04296 /* USART.CTRLB  bit masks and bit positions */
04297 #define USART_MPCM_bm  0x01  /* Multi-processor Communication Mode bit mask. */
04298 #define USART_MPCM_bp  0  /* Multi-processor Communication Mode bit position. */
04299 #define USART_RXMODE_gm  0x06  /* Receiver Mode group mask. */
04300 #define USART_RXMODE_gp  1  /* Receiver Mode group position. */
04301 #define USART_RXMODE0_bm  (1<<1)  /* Receiver Mode bit 0 mask. */
04302 #define USART_RXMODE0_bp  1  /* Receiver Mode bit 0 position. */
04303 #define USART_RXMODE1_bm  (1<<2)  /* Receiver Mode bit 1 mask. */
04304 #define USART_RXMODE1_bp  2  /* Receiver Mode bit 1 position. */
04305 #define USART_ODME_bm  0x08  /* Open Drain Mode Enable bit mask. */
04306 #define USART_ODME_bp  3  /* Open Drain Mode Enable bit position. */
04307 #define USART_SFDEN_bm  0x10  /* Start Frame Detection Enable bit mask. */
04308 #define USART_SFDEN_bp  4  /* Start Frame Detection Enable bit position. */
04309 #define USART_TXEN_bm  0x40  /* Transmitter Enable bit mask. */
04310 #define USART_TXEN_bp  6  /* Transmitter Enable bit position. */
04311 #define USART_RXEN_bm  0x80  /* Reciever enable bit mask. */
04312 #define USART_RXEN_bp  7  /* Reciever enable bit position. */
04313
04314 /* USART.CTRLC  bit masks and bit positions */
04315 #define USART_UCPHA_bm  0x02  /* SPI Master Mode, Clock Phase bit mask. */
04316 #define USART_UCPHA_bp  1  /* SPI Master Mode, Clock Phase bit position. */
04317 #define USART_UDORD_bm  0x04  /* SPI Master Mode, Data Order bit mask. */
04318 #define USART_UDORD_bp  2  /* SPI Master Mode, Data Order bit position. */
04319 #define USART_CHSIZE_gm  0x07  /* Character Size group mask. */
04320 #define USART_CHSIZE_gp  0  /* Character Size group position. */
04321 #define USART_CHSIZE0_bm  (1<<0)  /* Character Size bit 0 mask. */
04322 #define USART_CHSIZE0_bp  0  /* Character Size bit 0 position. */
04323 #define USART_CHSIZE1_bm  (1<<1)  /* Character Size bit 1 mask. */
04324 #define USART_CHSIZE1_bp  1  /* Character Size bit 1 position. */
04325 #define USART_CHSIZE2_bm  (1<<2)  /* Character Size bit 2 mask. */
04326 #define USART_CHSIZE2_bp  2  /* Character Size bit 2 position. */
04327 #define USART_SBMODE_bm  0x08  /* Stop Bit Mode bit mask. */
04328 #define USART_SBMODE_bp  3  /* Stop Bit Mode bit position. */
04329 #define USART_PMODE_gm  0x30  /* Parity Mode group mask. */
04330 #define USART_PMODE_gp  4  /* Parity Mode group position. */
04331 #define USART_PMODE0_bm  (1<<4)  /* Parity Mode bit 0 mask. */
04332 #define USART_PMODE0_bp  4  /* Parity Mode bit 0 position. */
04333 #define USART_PMODE1_bm  (1<<5)  /* Parity Mode bit 1 mask. */
04334 #define USART_PMODE1_bp  5  /* Parity Mode bit 1 position. */
04335 #define USART_CMODE_gm  0xC0  /* Communication Mode group mask. */
04336 #define USART_CMODE_gp  6  /* Communication Mode group position. */
04337 #define USART_CMODE0_bm  (1<<6)  /* Communication Mode bit 0 mask. */
04338 #define USART_CMODE0_bp  6  /* Communication Mode bit 0 position. */
04339 #define USART_CMODE1_bm  (1<<7)  /* Communication Mode bit 1 mask. */
04340 #define USART_CMODE1_bp  7  /* Communication Mode bit 1 position. */
04341 /* USART_CMODE  is already defined. */
04342
04343 /* USART.DBGCTRL  bit masks and bit positions */
04344 #define USART_DBGRUN_bm  0x01  /* Debug Run bit mask. */
04345 #define USART_DBGRUN_bp  0  /* Debug Run bit position. */
04346 #define USART_ABMBP_bm  0x80  /* Autobaud majority voter bypass bit mask. */
04347 #define USART_ABMBP_bp  7  /* Autobaud majority voter bypass bit position. */
04348
04349 /* USART.EVCTRL  bit masks and bit positions */
04350 #define USART_IREI_bm  0x01  /* IrDA Event Input Enable bit mask. */
04351 #define USART_IREI_bp  0  /* IrDA Event Input Enable bit position. */
04352
04353 /* USART.TXPLCTRL  bit masks and bit positions */
04354 #define USART_TXPL_gm  0xFF  /* Transmit pulse length group mask. */
04355 #define USART_TXPL_gp  0  /* Transmit pulse length group position. */
04356 #define USART_TXPL0_bm  (1<<0)  /* Transmit pulse length bit 0 mask. */
04357 #define USART_TXPL0_bp  0  /* Transmit pulse length bit 0 position. */
04358 #define USART_TXPL1_bm  (1<<1)  /* Transmit pulse length bit 1 mask. */
04359 #define USART_TXPL1_bp  1  /* Transmit pulse length bit 1 position. */
04360 #define USART_TXPL2_bm  (1<<2)  /* Transmit pulse length bit 2 mask. */
04361 #define USART_TXPL2_bp  2  /* Transmit pulse length bit 2 position. */
04362 #define USART_TXPL3_bm  (1<<3)  /* Transmit pulse length bit 3 mask. */
04363 #define USART_TXPL3_bp  3  /* Transmit pulse length bit 3 position. */
04364 #define USART_TXPL4_bm  (1<<4)  /* Transmit pulse length bit 4 mask. */
04365 #define USART_TXPL4_bp  4  /* Transmit pulse length bit 4 position. */
04366 #define USART_TXPL5_bm  (1<<5)  /* Transmit pulse length bit 5 mask. */
04367 #define USART_TXPL5_bp  5  /* Transmit pulse length bit 5 position. */
04368 #define USART_TXPL6_bm  (1<<6)  /* Transmit pulse length bit 6 mask. */
04369 #define USART_TXPL6_bp  6  /* Transmit pulse length bit 6 position. */
04370 #define USART_TXPL7_bm  (1<<7)  /* Transmit pulse length bit 7 mask. */
```

```
04371 #define USART_TXPL7_bp  7  /* Transmit pulse length bit 7 position. */
04372
04373 /* USART.RXPLCTRL  bit masks and bit positions */
04374 #define USART_RXPL_gm  0x7F  /* Receiver Pulse Lenght group mask. */
04375 #define USART_RXPL_gp  0  /* Receiver Pulse Lenght group position. */
04376 #define USART_RXPL0_bm  (1<<0)  /* Receiver Pulse Lenght bit 0 mask. */
04377 #define USART_RXPL0_bp  0  /* Receiver Pulse Lenght bit 0 position. */
04378 #define USART_RXPL1_bm  (1<<1)  /* Receiver Pulse Lenght bit 1 mask. */
04379 #define USART_RXPL1_bp  1  /* Receiver Pulse Lenght bit 1 position. */
04380 #define USART_RXPL2_bm  (1<<2)  /* Receiver Pulse Lenght bit 2 mask. */
04381 #define USART_RXPL2_bp  2  /* Receiver Pulse Lenght bit 2 position. */
04382 #define USART_RXPL3_bm  (1<<3)  /* Receiver Pulse Lenght bit 3 mask. */
04383 #define USART_RXPL3_bp  3  /* Receiver Pulse Lenght bit 3 position. */
04384 #define USART_RXPL4_bm  (1<<4)  /* Receiver Pulse Lenght bit 4 mask. */
04385 #define USART_RXPL4_bp  4  /* Receiver Pulse Lenght bit 4 position. */
04386 #define USART_RXPL5_bm  (1<<5)  /* Receiver Pulse Lenght bit 5 mask. */
04387 #define USART_RXPL5_bp  5  /* Receiver Pulse Lenght bit 5 position. */
04388 #define USART_RXPL6_bm  (1<<6)  /* Receiver Pulse Lenght bit 6 mask. */
04389 #define USART_RXPL6_bp  6  /* Receiver Pulse Lenght bit 6 position. */
04390
04391 /* VPORT - Virtual Ports */
04392 /* VPORT.INTFLAGS  bit masks and bit positions */
04393 #define VPORT_INT_gm  0xFF  /* Pin Interrupt group mask. */
04394 #define VPORT_INT_gp  0  /* Pin Interrupt group position. */
04395 #define VPORT_INT0_bm  (1<<0)  /* Pin Interrupt bit 0 mask. */
04396 #define VPORT_INT0_bp  0  /* Pin Interrupt bit 0 position. */
04397 #define VPORT_INT1_bm  (1<<1)  /* Pin Interrupt bit 1 mask. */
04398 #define VPORT_INT1_bp  1  /* Pin Interrupt bit 1 position. */
04399 #define VPORT_INT2_bm  (1<<2)  /* Pin Interrupt bit 2 mask. */
04400 #define VPORT_INT2_bp  2  /* Pin Interrupt bit 2 position. */
04401 #define VPORT_INT3_bm  (1<<3)  /* Pin Interrupt bit 3 mask. */
04402 #define VPORT_INT3_bp  3  /* Pin Interrupt bit 3 position. */
04403 #define VPORT_INT4_bm  (1<<4)  /* Pin Interrupt bit 4 mask. */
04404 #define VPORT_INT4_bp  4  /* Pin Interrupt bit 4 position. */
04405 #define VPORT_INT5_bm  (1<<5)  /* Pin Interrupt bit 5 mask. */
04406 #define VPORT_INT5_bp  5  /* Pin Interrupt bit 5 position. */
04407 #define VPORT_INT6_bm  (1<<6)  /* Pin Interrupt bit 6 mask. */
04408 #define VPORT_INT6_bp  6  /* Pin Interrupt bit 6 position. */
04409 #define VPORT_INT7_bm  (1<<7)  /* Pin Interrupt bit 7 mask. */
04410 #define VPORT_INT7_bp  7  /* Pin Interrupt bit 7 position. */
04411
04412 /* VREF - Voltage reference */
04413 /* VREF.CTRLA  bit masks and bit positions */
04414 #define VREF_DAC0REFSEL_gm  0x07  /* DAC0/AC0 reference select group mask. */
04415 #define VREF_DAC0REFSEL_gp  0  /* DAC0/AC0 reference select group position. */
04416 #define VREF_DAC0REFSEL0_bm  (1<<0)  /* DAC0/AC0 reference select bit 0 mask. */
04417 #define VREF_DAC0REFSEL0_bp  0  /* DAC0/AC0 reference select bit 0 position. */
04418 #define VREF_DAC0REFSEL1_bm  (1<<1)  /* DAC0/AC0 reference select bit 1 mask. */
04419 #define VREF_DAC0REFSEL1_bp  1  /* DAC0/AC0 reference select bit 1 position. */
04420 #define VREF_DAC0REFSEL2_bm  (1<<2)  /* DAC0/AC0 reference select bit 2 mask. */
04421 #define VREF_DAC0REFSEL2_bp  2  /* DAC0/AC0 reference select bit 2 position. */
04422 #define VREF_ADC0REFSEL_gm  0x70  /* ADC0 reference select group mask. */
04423 #define VREF_ADC0REFSEL_gp  4  /* ADC0 reference select group position. */
04424 #define VREF_ADC0REFSEL0_bm  (1<<4)  /* ADC0 reference select bit 0 mask. */
04425 #define VREF_ADC0REFSEL0_bp  4  /* ADC0 reference select bit 0 position. */
04426 #define VREF_ADC0REFSEL1_bm  (1<<5)  /* ADC0 reference select bit 1 mask. */
04427 #define VREF_ADC0REFSEL1_bp  5  /* ADC0 reference select bit 1 position. */
04428 #define VREF_ADC0REFSEL2_bm  (1<<6)  /* ADC0 reference select bit 2 mask. */
04429 #define VREF_ADC0REFSEL2_bp  6  /* ADC0 reference select bit 2 position. */
04430
04431 /* VREF.CTRLB  bit masks and bit positions */
04432 #define VREF_DAC0REFEN_bm  0x01  /* DAC0/AC0 reference enable bit mask. */
04433 #define VREF_DAC0REFEN_bp  0  /* DAC0/AC0 reference enable bit position. */
04434 #define VREF_ADC0REFEN_bm  0x02  /* ADC0 reference enable bit mask. */
04435 #define VREF_ADC0REFEN_bp  1  /* ADC0 reference enable bit position. */
04436
04437 /* WDT - Watch-Dog Timer */
04438 /* WDT.CTRLA  bit masks and bit positions */
04439 #define WDT_PERIOD_gm  0x0F  /* Period group mask. */
04440 #define WDT_PERIOD_gp  0  /* Period group position. */
04441 #define WDT_PERIOD0_bm  (1<<0)  /* Period bit 0 mask. */
04442 #define WDT_PERIOD0_bp  0  /* Period bit 0 position. */
04443 #define WDT_PERIOD1_bm  (1<<1)  /* Period bit 1 mask. */
04444 #define WDT_PERIOD1_bp  1  /* Period bit 1 position. */
04445 #define WDT_PERIOD2_bm  (1<<2)  /* Period bit 2 mask. */
04446 #define WDT_PERIOD2_bp  2  /* Period bit 2 position. */
04447 #define WDT_PERIOD3_bm  (1<<3)  /* Period bit 3 mask. */
04448 #define WDT_PERIOD3_bp  3  /* Period bit 3 position. */
04449 #define WDT_WINDOW_gm  0xF0  /* Window group mask. */
04450 #define WDT_WINDOW_gp  4  /* Window group position. */
04451 #define WDT_WINDOW0_bm  (1<<4)  /* Window bit 0 mask. */
04452 #define WDT_WINDOW0_bp  4  /* Window bit 0 position. */
04453 #define WDT_WINDOW1_bm  (1<<5)  /* Window bit 1 mask. */
04454 #define WDT_WINDOW1_bp  5  /* Window bit 1 position. */
04455 #define WDT_WINDOW2_bm  (1<<6)  /* Window bit 2 mask. */
04456 #define WDT_WINDOW2_bp  6  /* Window bit 2 position. */
04457 #define WDT_WINDOW3_bm  (1<<7)  /* Window bit 3 mask. */
```

```
04458 #define WDT_WINDOW3_bp  7  /* Window bit 3 position. */
04459
04460 /* WDT.STATUS  bit masks and bit positions */
04461 #define WDT_SYNCBUSY_bm  0x01  /* Syncronization busy bit mask. */
04462 #define WDT_SYNCBUSY_bp  0  /* Syncronization busy bit position. */
04463 #define WDT_LOCK_bm  0x80  /* Lock enable bit mask. */
04464 #define WDT_LOCK_bp  7  /* Lock enable bit position. */
04465
04466 // Generic Port Pins
04467
04468 #define PIN0_bm 0x01
04469 #define PIN0_bp 0
04470 #define PIN1_bm 0x02
04471 #define PIN1_bp 1
04472 #define PIN2_bm 0x04
04473 #define PIN2_bp 2
04474 #define PIN3_bm 0x08
04475 #define PIN3_bp 3
04476 #define PIN4_bm 0x10
04477 #define PIN4_bp 4
04478 #define PIN5_bm 0x20
04479 #define PIN5_bp 5
04480 #define PIN6_bm 0x40
04481 #define PIN6_bp 6
04482 #define PIN7_bm 0x80
04483 #define PIN7_bp 7
04484
04485 /* ========== Interrupt Vector Definitions ========== */
04486 /* Vector 0 is the reset vector */
04487
04488 /* CRCSCAN interrupt vectors */
04489 #define CRCSCAN_NMI_vect_num  1
04490 #define CRCSCAN_NMI_vect      _VECTOR(1)  /*  */
04491
04492 /* BOD interrupt vectors */
04493 #define BOD_VLM_vect_num  2
04494 #define BOD_VLM_vect      _VECTOR(2)  /*  */
04495
04496 /* PORTA interrupt vectors */
04497 #define PORTA_PORT_vect_num  3
04498 #define PORTA_PORT_vect      _VECTOR(3)  /*  */
04499
04500 /* PORTB interrupt vectors */
04501 #define PORTB_PORT_vect_num  4
04502 #define PORTB_PORT_vect      _VECTOR(4)  /*  */
04503
04504 /* RTC interrupt vectors */
04505 #define RTC_CNT_vect_num  6
04506 #define RTC_CNT_vect      _VECTOR(6)  /*  */
04507 #define RTC_PIT_vect_num  7
04508 #define RTC_PIT_vect      _VECTOR(7)  /*  */
04509
04510 /* TCA0 interrupt vectors */
04511 #define TCA0_LUNF_vect_num  8
04512 #define TCA0_LUNF_vect      _VECTOR(8)  /*  */
04513 #define TCA0_OVF_vect_num  8
04514 #define TCA0_OVF_vect      _VECTOR(8)  /*  */
04515 #define TCA0_HUNF_vect_num  9
04516 #define TCA0_HUNF_vect      _VECTOR(9)  /*  */
04517 #define TCA0_LCMP0_vect_num  10
04518 #define TCA0_LCMP0_vect      _VECTOR(10)  /*  */
04519 #define TCA0_CMP0_vect_num  10
04520 #define TCA0_CMP0_vect      _VECTOR(10)  /*  */
04521 #define TCA0_CMP1_vect_num  11
04522 #define TCA0_CMP1_vect      _VECTOR(11)  /*  */
04523 #define TCA0_LCMP1_vect_num  11
04524 #define TCA0_LCMP1_vect      _VECTOR(11)  /*  */
04525 #define TCA0_CMP2_vect_num  12
04526 #define TCA0_CMP2_vect      _VECTOR(12)  /*  */
04527 #define TCA0_LCMP2_vect_num  12
04528 #define TCA0_LCMP2_vect      _VECTOR(12)  /*  */
04529
04530 /* TCB0 interrupt vectors */
04531 #define TCB0_INT_vect_num  13
04532 #define TCB0_INT_vect      _VECTOR(13)  /*  */
04533
04534 /* AC0 interrupt vectors */
04535 #define AC0_AC_vect_num  16
04536 #define AC0_AC_vect      _VECTOR(16)  /*  */
04537
04538 /* ADC0 interrupt vectors */
04539 #define ADC0_RESRDY_vect_num  17
04540 #define ADC0_RESRDY_vect      _VECTOR(17)  /*  */
04541 #define ADC0_WCOMP_vect_num  18
04542 #define ADC0_WCOMP_vect      _VECTOR(18)  /*  */
04543
04544 /* TWI0 interrupt vectors */
```

```
04545 #define TWI0_TWIS_vect_num  19
04546 #define TWI0_TWIS_vect      _VECTOR(19)  /*  */
04547 #define TWI0_TWIM_vect_num  20
04548 #define TWI0_TWIM_vect      _VECTOR(20)  /*  */
04549
04550 /* SPI0 interrupt vectors */
04551 #define SPI0_INT_vect_num  21
04552 #define SPI0_INT_vect      _VECTOR(21)  /*  */
04553
04554 /* USART0 interrupt vectors */
04555 #define USART0_RXC_vect_num  22
04556 #define USART0_RXC_vect      _VECTOR(22)  /*  */
04557 #define USART0_DRE_vect_num  23
04558 #define USART0_DRE_vect      _VECTOR(23)  /*  */
04559 #define USART0_TXC_vect_num  24
04560 #define USART0_TXC_vect      _VECTOR(24)  /*  */
04561
04562 /* NVMCTRL interrupt vectors */
04563 #define NVMCTRL_EE_vect_num  25
04564 #define NVMCTRL_EE_vect      _VECTOR(25)  /*  */
04565
04566 #define _VECTOR_SIZE 2 /* Size of individual vector. */
04567 #define _VECTORS_SIZE (26 * _VECTOR_SIZE)
04568
04569
04570 /* ========== Constants ========== */
04571
04572 #define DATAMEM_START     (0x0000)
04573 #define DATAMEM_SIZE      (36864)
04574 #define DATAMEM_END       (DATAMEM_START + DATAMEM_SIZE - 1)
04575
04576 #define EEPROM_START     (0x1400)
04577 #define EEPROM_SIZE      (128)
04578 #define EEPROM_PAGE_SIZE (32)
04579 #define EEPROM_END       (EEPROM_START + EEPROM_SIZE - 1)
04580
04581 /* Added MAPPED_EEPROM segment names for avr-libc */
04582 #define MAPPED_EEPROM_START     (EEPROM_START)
04583 #define MAPPED_EEPROM_SIZE      (EEPROM_SIZE)
04584 #define MAPPED_EEPROM_PAGE_SIZE (EEPROM_PAGE_SIZE)
04585 #define MAPPED_EEPROM_END       (MAPPED_EEPROM_START + MAPPED_EEPROM_SIZE - 1)
04586
04587 #define FUSES_START     (0x1280)
04588 #define FUSES_SIZE      (10)
04589 #define FUSES_PAGE_SIZE (32)
04590 #define FUSES_END       (FUSES_START + FUSES_SIZE - 1)
04591
04592 #define INTERNAL_SRAM_START     (0x3F00)
04593 #define INTERNAL_SRAM_SIZE      (256)
04594 #define INTERNAL_SRAM_PAGE_SIZE (0)
04595 #define INTERNAL_SRAM_END       (INTERNAL_SRAM_START + INTERNAL_SRAM_SIZE - 1)
04596
04597 #define IO_START     (0x0000)
04598 #define IO_SIZE      (4352)
04599 #define IO_PAGE_SIZE (0)
04600 #define IO_END       (IO_START + IO_SIZE - 1)
04601
04602 #define LOCKBITS_START     (0x128A)
04603 #define LOCKBITS_SIZE      (1)
04604 #define LOCKBITS_PAGE_SIZE (32)
04605 #define LOCKBITS_END       (LOCKBITS_START + LOCKBITS_SIZE - 1)
04606
04607 #define MAPPED_PROGMEM_START     (0x8000)
04608 #define MAPPED_PROGMEM_SIZE      (4096)
04609 #define MAPPED_PROGMEM_PAGE_SIZE (64)
04610 #define MAPPED_PROGMEM_END       (MAPPED_PROGMEM_START + MAPPED_PROGMEM_SIZE - 1)
04611
04612 #define PROD_SIGNATURES_START     (0x1103)
04613 #define PROD_SIGNATURES_SIZE      (61)
04614 #define PROD_SIGNATURES_PAGE_SIZE (64)
04615 #define PROD_SIGNATURES_END       (PROD_SIGNATURES_START + PROD_SIGNATURES_SIZE - 1)
04616
04617 #define SIGNATURES_START     (0x1100)
04618 #define SIGNATURES_SIZE      (3)
04619 #define SIGNATURES_PAGE_SIZE (64)
04620 #define SIGNATURES_END       (SIGNATURES_START + SIGNATURES_SIZE - 1)
04621
04622 #define USER_SIGNATURES_START     (0x1300)
04623 #define USER_SIGNATURES_SIZE      (32)
04624 #define USER_SIGNATURES_PAGE_SIZE (32)
04625 #define USER_SIGNATURES_END       (USER_SIGNATURES_START + USER_SIGNATURES_SIZE - 1)
04626
04627 #define PROGMEM_START     (0x0000)
04628 #define PROGMEM_SIZE      (4096)
04629 #define PROGMEM_END       (PROGMEM_START + PROGMEM_SIZE - 1)
04630
04631 #define PROGMEM_START     (0x0000)
```

```
04632 #define PROGMEM_SIZE      (4096)
04633 #define PROGMEM_PAGE_SIZE (64)
04634 #define PROGMEM_END       (PROGMEM_START + PROGMEM_SIZE - 1)
04635
04636 #define FLASHSTART   PROGMEM_START
04637 #define FLASHEND     PROGMEM_END
04638 #define RAMSTART     INTERNAL_SRAM_START
04639 #define RAMSIZE      INTERNAL_SRAM_SIZE
04640 #define RAMEND       INTERNAL_SRAM_END
04641 #define E2END        EEPROM_END
04642 #define E2PAGESIZE   EEPROM_PAGE_SIZE
04643
04644 /* ========== Fuses ========== */
04645 #define FUSE_MEMORY_SIZE 9
04646
04647 /* Fuse offset 0x00*/
04648 #define PERIOD0  (unsigned char)~_BV(0)  /* Watchdog Timeout Period Bit 0 */
04649 #define PERIOD1  (unsigned char)~_BV(1)  /* Watchdog Timeout Period Bit 1 */
04650 #define PERIOD2  (unsigned char)~_BV(2)  /* Watchdog Timeout Period Bit 2 */
04651 #define PERIOD3  (unsigned char)~_BV(3)  /* Watchdog Timeout Period Bit 3 */
04652 #define WINDOW0  (unsigned char)~_BV(4)  /* Watchdog Window Timeout Period Bit 0 */
04653 #define WINDOW1  (unsigned char)~_BV(5)  /* Watchdog Window Timeout Period Bit 1 */
04654 #define WINDOW2  (unsigned char)~_BV(6)  /* Watchdog Window Timeout Period Bit 2 */
04655 #define WINDOW3  (unsigned char)~_BV(7)  /* Watchdog Window Timeout Period Bit 3 */
04656
04657 /* Fuse offset 0x01*/
04658 #define SLEEP0  (unsigned char)~_BV(0)  /* BOD Operation in Sleep Mode Bit 0 */
04659 #define SLEEP1  (unsigned char)~_BV(1)  /* BOD Operation in Sleep Mode Bit 1 */
04660 #define ACTIVE0  (unsigned char)~_BV(2)  /* BOD Operation in Active Mode Bit 0 */
04661 #define ACTIVE1  (unsigned char)~_BV(3)  /* BOD Operation in Active Mode Bit 1 */
04662 #define SAMPFREQ  (unsigned char)~_BV(4)  /* BOD Sample Frequency */
04663 #define LVL0  (unsigned char)~_BV(5)  /* BOD Level Bit 0 */
04664 #define LVL1  (unsigned char)~_BV(6)  /* BOD Level Bit 1 */
04665 #define LVL2  (unsigned char)~_BV(7)  /* BOD Level Bit 2 */
04666
04667 /* Fuse offset 0x02*/
04668 #define FREQSEL0  (unsigned char)~_BV(0)  /* Frequency Select Bit 0 */
04669 #define FREQSEL1  (unsigned char)~_BV(1)  /* Frequency Select Bit 1 */
04670 #define OSCLOCK  (unsigned char)~_BV(7)  /* Oscillator Lock */
04671
04672 /* Fuse offset 0x03*/
04673
04674 /* Fuse offset 0x04*/
04675 #define CMPA  (unsigned char)~_BV(0)  /* Compare A Default Output Value */
04676 #define CMPB  (unsigned char)~_BV(1)  /* Compare B Default Output Value */
04677 #define CMPC  (unsigned char)~_BV(2)  /* Compare C Default Output Value */
04678 #define CMPD  (unsigned char)~_BV(3)  /* Compare D Default Output Value */
04679 #define CMPAEN  (unsigned char)~_BV(4)  /* Compare A Output Enable */
04680 #define CMPBEN  (unsigned char)~_BV(5)  /* Compare B Output Enable */
04681 #define CMPCEN  (unsigned char)~_BV(6)  /* Compare C Output Enable */
04682 #define CMPDEN  (unsigned char)~_BV(7)  /* Compare D Output Enable */
04683
04684 /* Fuse offset 0x05*/
04685 #define EESAVE  (unsigned char)~_BV(0)  /* EEPROM Save */
04686 #define RSTPINCFG0  (unsigned char)~_BV(2)  /* Reset Pin Configuration Bit 0 */
04687 #define RSTPINCFG1  (unsigned char)~_BV(3)  /* Reset Pin Configuration Bit 1 */
04688 #define CRCSRC0  (unsigned char)~_BV(6)  /* CRC Source Bit 0 */
04689 #define CRCSRC1  (unsigned char)~_BV(7)  /* CRC Source Bit 1 */
04690
04691 /* Fuse offset 0x06*/
04692 #define SUT0  (unsigned char)~_BV(0)  /* Startup Time Bit 0 */
04693 #define SUT1  (unsigned char)~_BV(1)  /* Startup Time Bit 1 */
04694 #define SUT2  (unsigned char)~_BV(2)  /* Startup Time Bit 2 */
04695
04696 /* Fuse offset 0x07*/
04697
04698 /* Fuse offset 0x08*/
04699
04700 /* Fuse offset 0x09*/
04701
04702 /* ========== Lock Bits ========== */
04703 #define __LOCK_BITS_EXIST
04704
04705 /* ========== Signature ========== */
04706 #define SIGNATURE_0 0x1E
04707 #define SIGNATURE_1 0x92
04708 #define SIGNATURE_2 0x26
04709
04710
04711 #endif /* #ifdef _AVR_ATTINY404_H_INCLUDED */
04712
04713
```

5.4 src/ADC/ADC.h File Reference

Interface to the AVR ADC hardware.

```
#include <stdint.h>
#include <stdbool.h>
```

Functions

- void ADC_Init (uint8_t pin_num)
  Initializes the AVR hardware in order to accept Input for ADC usage.
- void **ADC_Enable** (void)
  Enables the ADC.
- void **ADC_Disable** ()
  Disables the ADC.
- void ADC_Setup (void)
  Sets up the ADC.
- void ADC_SetPin (uint8_t pin_num)
  Sets the pin used in the MUX for ADC0.

Variables

- uint16_t(* ADC_ReadValue)(uint8_t pin_num)
  Reads ADC value into variable.

5.4.1 Detailed Description

Interface to the AVR ADC hardware.

This file is...

Author

Jake G

Date

2024

Copyright

None

5.4.2 Function Documentation

5.4.2.1 ADC_Init()

```
void ADC_Init (
            uint8_t pin_num )
```

Initializes the AVR hardware in order to accept Input for ADC usage.

Parameters

pin_num	The number of the pin 0-7 you are initializing.

This function only makes use of PORTA by default. It sets the direction register to input, disables the pull-up resistor and also diables interrupts alongside the input buffer(digital).

This in turn helps reduce noise when using the ADC.

5.4.2.2 ADC_SetPin()

```
void ADC_SetPin (
            uint8_t pin_num )
```

Sets the pin used in the MUX for ADC0.

Parameters

pin_num	The number of the pin in Port A.

5.4.2.3 ADC_Setup()

```
void ADC_Setup (
            void  )
```

Sets up the ADC.

This function sets up the ADC to take and accumulate 32 samples. It also sets the inital delay to 32 ADC clock cycles, and sets the VREF to VDD or VCC.

This function should only need to be called once.

5.4.3 Variable Documentation

5.4.3.1 ADC_ReadValue

```
uint16_t(* ADC_ReadValue) (uint8_t pin_num) (
            uint8_t pin_num )  [extern]
```

Reads ADC value into variable.

Parameters

pin_num	The bin number of the ADC pin being read.

This function depends on the ADC already being initialized and enabled before being called.

5.5 ADC.h

Go to the documentation of this file.

```
00001
00010 #ifndef ADC_H
00011 #define ADC_H
00012
00013 #include <stdint.h>
00014 #include <stdbool.h>
00015
00016
00017
00030 void ADC_Init(uint8_t pin_num);
00031
00032
00036 void ADC_Enable(void);
00037
00038
00042 void ADC_Disable();
00043
00044
00053 extern uint16_t (*ADC_ReadValue)(uint8_t pin_num);
00054
00064 void ADC_Setup(void);
00065
00066
00072 void ADC_SetPin(uint8_t pin_num);
00073
00074 #endif //ADC_H
```

5.6 Enable.h

```
00001
00010 #ifndef ENABLE
00011 #define ENABLE
00012
00013 #define EN1 2
00014 #define EN2 3
00015 #define EN3 2
00016
00020 void Enable_SetPinsHigh();
00021
00025 void Enable_SetPinsLow();
00026
00027
00028 #endif //ENABLE
```

5.7 src/EnOut/EnOut.h File Reference

PUT_TEXT_HERE.

```
#include <stdint.h>
```

Macros

- #define **G1** 1
- #define **G2** 3
- #define **G3** 2
- #define **G1_BM** (1<<1)
- #define **G2_BM** (1<<3)
- #define **G3_BM** (1<<2)

Functions

- void EnOut_PulsePins (uint16_t pulse_time)
- void EnOut_SetupPins (void)
- void EnOut_SetAllHigh (void)

Variables

- void(* **Delay_MicroSeconds**)(double us)

5.7.1 Detailed Description

PUT_TEXT_HERE.

This file is...

Author

username

Date

todays_date

Copyright

None

5.7.2 Function Documentation

5.7.2.1 EnOut_PulsePins()

```
void EnOut_PulsePins (
            uint16_t pulse_time )
```

A function that adds two to a number

Parameters

pulse_time	The time in micro seconds.

This function turns off G1, G2 and G3, once the passed time in micro seconds has elapsed the pins are all set to low.

5.7.2.2 EnOut_SetAllHigh()

```
void EnOut_SetAllHigh (
            void  )
```

This sets G1, G2 and G3 to high on the output.

5.7.2.3 EnOut_SetupPins()

```
void EnOut_SetupPins (
            void  )
```

This sets up the G1, G2 and the G3 pins for output.

5.8 EnOut.h

Go to the documentation of this file.

```
00001
00010 #ifndef ENOUT_H
00011 #define ENOUT_H
00012
00013 #include <stdint.h>
00014
00015
00016 #define G1 1
00017 #define G2 3
00018 #define G3 2
00019
00020 #define G1_BM (1«1)
00021 #define G2_BM (1«3)
00022 #define G3_BM (1«2)
00023
00024 extern void (*Delay_MicroSeconds)(double us);
00025
00026
00034 void EnOut_PulsePins(uint16_t pulse_time);
00035
00039 void EnOut_SetupPins(void);
00040
00044 void EnOut_SetAllHigh(void);
00045
00046
00047
00048 #endif //ENOUT_H
```

5.9 src/load/load.h File Reference

Module for handling the ADC input from the load pins.

```
#include <stdbool.h>
#include <stdint.h>
```

Macros

- #define **LOWTHRESH** 0

  Low Threshold Anything below or equal to the value will cause the pin to lock into the disabled state.
- #define **HIGHTHRESH** 1000

  High Threshold Anything equal or above the value will cause the pin to lock into the disabled state.
- #define HYSTERESIS_HI 900

  The upper Hystersis value. This is the upper bound of the digital/boolean hysteresis curve.
- #define HYSTERESIS_LO 700

  The lower Hystersis value. This is the upper bound of the digital/boolean hysteresis curve.

Functions

- void Load_HandleLoadPortA (uint8_t adc_pin, uint8_t out_pin)

  Checks if the adc pin is inbetween LOWTHRESH and HIGHTHRESH and then sets or disables the output pin on PORTA.
- void Load_HandleLoadPortB (uint8_t adc_pin, uint8_t out_pin)

  Checks if the adc pin is inbetween LOWTHRESH and HIGHTHRESH and then sets or disables the output pin on PORTB.
- void **Load_SoftResetDisabledLoads** ()

  Resets the disabled array state.

5.9.1 Detailed Description

Module for handling the ADC input from the load pins.

This file holds the functions for reading the ADC values.

Author

Jake G

Date

2024

Copyright

None

5.9.2 Macro Definition Documentation

5.9.2.1 HYSTERESIS_HI

```
#define HYSTERESIS_HI 900
```

The upper Hystersis value. This is the upper bound of the digital/boolean hysteresis curve.

If the input is below low it always sets the output high. If the input is between high and low with the output high it stays high. If the input is above high and the ouput is high it's set low.

5.9.2.2 HYSTERESIS_LO

```
#define HYSTERESIS_LO 700
```

The lower Hystersis value. This is the upper bound of the digital/boolean hysteresis curve.

If the input is below low it always sets the output high. If the input is between high and low with the output high it stays high. If the input is above high and the ouput is high it's set low.

5.9.3 Function Documentation

5.9.3.1 Load_HandleLoadPortA()

```
void Load_HandleLoadPortA (
            uint8_t adc_pin,
            uint8_t out_pin )
```

Checks if the adc pin is inbetween LOWTHRESH and HIGHTHRESH and then sets or disables the output pin on PORTA.

Parameters

adc_pin	the pin number that the load value is read from.
out_pin	The pin that is set high if the adc_pin input is valid.

5.9.3.2 Load_HandleLoadPortB()

```
void Load_HandleLoadPortB (
             uint8_t adc_pin,
             uint8_t out_pin )
```

Checks if the adc pin is inbetween LOWTHRESH and HIGHTHRESH and then sets or disables the output pin on PORTB.

Parameters

adc_pin	the pin number that the load value is read from.
out_pin	The pin that is set high if the adc_pin input is valid.

5.10 load.h

Go to the documentation of this file.

```
00001
00010 #ifndef LOAD_H
00011 #define LOAD_H
00012
00013 #include <stdbool.h>
00014 #include <stdint.h>
00015
00021 #define LOWTHRESH 0
00022
00028 #define HIGHTHRESH 1000
00029
00030
00039 #ifndef HYSTERESIS_HI
00040 #define HYSTERESIS_HI 900
00041 #endif
00042
00051 #ifndef HYSTERESIS_LO
00052 #define HYSTERESIS_LO 700
00053 #endif
00054
00055
00062 void Load_HandleLoadPortA(uint8_t adc_pin, uint8_t out_pin);
00063
00070 void Load_HandleLoadPortB(uint8_t adc_pin, uint8_t out_pin);
00071
00072
00076 void Load_SoftResetDisabledLoads();
00077
00078 #endif  /* LOAD_H */
00079
```

5.11 src/main.c File Reference

File contains the main function.

```
#include "config.h"
#include "RegEdit.h"
```

```
#include "zero_cross_detection.h"
#include "ADC.h"
#include "EnOut.h"
#include "load.h"
#include "Enable.h"
#include <avr/cpufunc.h>
#include <avr/io.h>
#include <util/delay.h>
```

Macros

- #define **F_CPU** 3333333UL
- #define **__DELAY_BACKWARD_COMPATIBLE__**

Functions

- int **main** (int argc, char ∗∗argv)

Variables

- void(∗ **Delay_MicroSeconds**)(double us) = _delay_us

5.11.1 Detailed Description

File contains the main function.

Author

Jake G

Date

18 September 2024

This file contains the main logic loop and exists to setup the values specific to the micro-controller. All other functions and configuration are extracted into separate source files and headers for configuration.

5.12 inc/RegEdit.h File Reference

Module/Interface for editing AVR registers.

```
#include <stdint.h>
#include <stdbool.h>
```

Functions

- void RegEdit_SetRegister (void *reg)
  Sets the value of the register to 0xFF.
- void RegEdit_ClearRegister (void *reg)
  Sets the value of the register to 0x00.
- void RegEdit_SetBit (void *reg, uint8_t bit_num)
  Sets a single bit in the register.
- void RegEdit_ClearBit (void *reg, uint8_t bit_num)
  Clears a single bit in the register.
- bool RegEdit_IsBitSet (void *reg, uint8_t bit_num)
  Checks if a single bit is set in the register.
- void RegEdit_OR_Num (void *reg, uint8_t num)
  Preforms logical OR Equals with the passed num.
- void RegEdit_AND_Num (void *reg, uint8_t num)
  Preforms logical AND Equals with the passed num.
- void RegEdit_SetNum (void *reg, uint8_t num)
  Sets the register to the passed number value.

5.12.1 Detailed Description

Module/Interface for editing AVR registers.

This file is an interface to AVR registers or the avr/io.h

Author

Jake G

Date

2024

Copyright

None

5.12.2 Function Documentation

5.12.2.1 RegEdit_AND_Num()

```
void RegEdit_AND_Num (
            void * reg,
            uint8_t num )
```

Preforms logical AND Equals with the passed num.

Parameters

reg	A pointer to a register
The	bit's index or number in the register
reg	The register address.
num	The bit location.

5.12.2.2 RegEdit_ClearBit()

```
void RegEdit_ClearBit (
            void * reg,
            uint8_t bit_num )
```

Clears a single bit in the register.

Parameters

reg	A pointer to a register
The	bit's index or number in the register
reg	The register address.
bit_num	The bit location.

5.12.2.3 RegEdit_ClearRegister()

```
void RegEdit_ClearRegister (
            void * reg )
```

Sets the value of the register to 0x00.

Parameters

reg	A pointer to a register
reg	The register address.

5.12.2.4 RegEdit_IsBitSet()

```
bool RegEdit_IsBitSet (
            void * reg,
            uint8_t bit_num )
```

Checks if a single bit is set in the register.

Parameters

reg	A pointer to a register
The	bit's index or number in the register
reg	The register address.
bit_num	The bit location.

Returns

5.12.2.5 RegEdit_OR_Num()

```
void RegEdit_OR_Num (
            void * reg,
            uint8_t num )
```

Preforms logical OR Equals with the passed num.

Parameters

reg	A pointer to a register
The	bit's index or number in the register
reg	The register address.
num	The bit location.

5.12.2.6 RegEdit_SetBit()

```
void RegEdit_SetBit (
            void * reg,
            uint8_t bit_num )
```

Sets a single bit in the register.

Parameters

reg	A pointer to a register
The	bit's index or number in the register
reg	The register address.
bit_num	The bit location.

5.12.2.7 RegEdit_SetNum()

```
void RegEdit_SetNum (
            void * reg,
            uint8_t num )
```

Sets the register to the passed number value.

Parameters

reg	A pointer to a register
The	bit's index or number in the register
reg	The register address.
num	The bit location.

5.12.2.8 RegEdit_SetRegister()

```
void RegEdit_SetRegister (
            void * reg )
```

Sets the value of the register to 0xFF.

Parameters

reg	A pointer to a register
reg	The register address.

5.13 RegEdit.h

Go to the documentation of this file.

```
00001
00010 #ifndef REGEDIT_H
00011 #define REGEDIT_H
00012
00013 #include <stdint.h>
00014 #include <stdbool.h>
00015
00016
00021 void RegEdit_SetRegister(void *reg);
00022
00027 void RegEdit_ClearRegister(void *reg);
00028
00034 void RegEdit_SetBit(void *reg, uint8_t bit_num);
00035
00041 void RegEdit_ClearBit(void *reg, uint8_t bit_num);
00042
00048 bool RegEdit_IsBitSet(void *reg, uint8_t bit_num);
00049
00055 void RegEdit_OR_Num(void *reg, uint8_t num);
00056
00062 void RegEdit_AND_Num(void *reg, uint8_t num);
00063
00069 void RegEdit_SetNum(void *reg, uint8_t num);
00070
00071 #endif //REGEDIT_H
```

5.14 RegEdit.h

```
00001
00010 #ifndef REGEDIT_H
00011 #define REGEDIT_H
00012
00013
00014 #include <stdint.h>
00015 #include <stdbool.h>
00016
00021 void RegEdit_SetRegister(void *reg);
00022
00027 void RegEdit_ClearRegister(void *reg);
00028
00034 void RegEdit_SetBit(void *reg, uint8_t bit_num);
00035
00041 void RegEdit_ClearBit(void *reg, uint8_t bit_num);
00042
00049 bool RegEdit_IsBitSet(void *reg, uint8_t bit_num);
00050
00056 void RegEdit_OR_Num(void *reg, uint8_t num);
00057
00063 void RegEdit_AND_Num(void *reg, uint8_t num);
00064
00070 void RegEdit_SetNum(void *reg, uint8_t num);
00071
00076 uint8_t RegEdit_ReadReg(void *reg);
00077
00078 #endif //REGEDIT_H
```

5.15 usart.h

```
00001
00010 #ifndef USART_H
00011 #define USART_H
00012
00013
00017 void USART0_init(void);
00018
00023 void USART0_sendChar(char c);
00024
00029 void USART0_sendString(char *str);
00030
00031
00032
00033 #endif //USART_H
```

5.16 inc/zero_cross_detection.h File Reference

File contains the zero cross detection functions.

```
#include <stdint.h>
#include <stdbool.h>
```

Functions

- void ZCD_Setup (void)

  Zero Cross Detection Setup.
- bool ZCD_IsTriggered (void)

  Checks if ZCD should trigger on value.
- void **ZCD_Poll** (void)

  Function blocks execution until ZCD is triggered.

5.16.1 Detailed Description

File contains the zero cross detection functions.

Author

Jake G

Date

16 June 2024

This file holds all the code/functions needed to read the value of the AC waveform. It uses the trigger value from the `config.h` file to evaluate the state.

This module depends on the ADC.h module to get readings from the ADC hardware.

5.16.2 Function Documentation

5.16.2.1 ZCD_IsTriggered()

```
bool ZCD_IsTriggered (
             void  )
```

Checks if ZCD should trigger on value.

The function checks for a positive edge first using the ZCD_IsPositiveEdge function. If a positive edge was found, then it takes an addition set of samples and averages them; only triggering(returning true) in the case that the average is higher than the previous sample.

5.16.2.2 ZCD_Setup()

```
void ZCD_Setup (
             void  )
```

Zero Cross Detection Setup.

Sets up the hardware to read the values coming from the AC input waveform.

5.17 zero_cross_detection.h

Go to the documentation of this file.

```
00001
00015 #ifndef ZERO_CROSS_DETECTION
00016 #define ZERO_CROSS_DETECTION
00017
00018 #include <stdint.h>
00019 #include <stdbool.h>
00020
00027 void ZCD_Setup(void);
00028 //int ZCD_Setup(volatile uint8_t *ddrx, uint8_t *port, uint8_t pin);
00029
00030
00039 bool ZCD_IsTriggered(void);
00040
00041
00046 void ZCD_Poll(void);
00047
00048
00049 #endif //ZERO_CROSS_DETECTION
```

5.18 src/zero_cross_detection/zero_cross_detection.h File Reference

File contains the zero cross detection functions.

```
#include <stdint.h>
#include <stdbool.h>
```

Functions

- void ZCD_Setup (void)

  Zero Cross Detection Setup.
- bool ZCD_IsTriggered (void)

  Checks if ZCD should trigger on value.
- void **ZCD_Poll** (void)

  Function blocks execution until ZCD is triggered.

5.18.1 Detailed Description

File contains the zero cross detection functions.

Author

Jake G

Date

16 June 2024

This file holds all the code/functions needed to read the value of the AC waveform. It uses the trigger value from the `config.h` file to evaluate the state.

This module depends on the ADC.h module to get readings from the ADC hardware.

5.18.2 Function Documentation

5.18.2.1 ZCD_IsTriggered()

```
bool ZCD_IsTriggered (
            void  )
```

Checks if ZCD should trigger on value.

The function checks for a positive edge first using the ZCD_IsPositiveEdge function. If a positive edge was found, then it takes an addition set of samples and averages them; only triggering(returning true) in the case that the average is higher than the previous sample.

5.18.2.2 ZCD_Setup()

```
void ZCD_Setup (
            void  )
```

Zero Cross Detection Setup.

Sets up the hardware to read the values coming from the AC input waveform.

5.19 zero_cross_detection.h

Go to the documentation of this file.

```
00001
00015 #ifndef ZERO_CROSS_DETECTION
00016 #define ZERO_CROSS_DETECTION
00017
00018 #include <stdint.h>
00019 #include <stdbool.h>
00020
00027 void ZCD_Setup(void);
00028 //int ZCD_Setup(volatile uint8_t *ddrx, uint8_t *port, uint8_t pin);
00029
00030
00039 bool ZCD_IsTriggered(void);
00040
00041
00046 void ZCD_Poll(void);
00047
00048
00049 #endif //ZERO_CROSS_DETECTION
```