

TahakomAVRLibDoc

1.0

Generated by Doxygen 1.8.17



<b>1 Introduction</b>	<b>1</b>
1.1 Library structure	1
1.2 Software setup	1
1.2.1 Linux	2
1.2.2 Windows	2
1.3 Compile and Flash programs	2
1.3.1 Linux	2
1.3.2 Windows	3
1.4 Hardware setup	3
1.5 Applications	3
1.6 Projects	3
1.7 Author	3
1.8 License	3
<b>2 Applications</b>	<b>5</b>
2.1 Blink a Led	5
2.1.1 Hardware	6
2.1.2 Circuit	6
2.1.3 Code	7
2.1.4 Author	7
2.1.5 License	8
2.2 AVR Square-Wave Organ	8
2.2.1 Hardware	8
2.2.2 Circuit	9
2.2.3 Code	9
2.2.4 Author	10
2.2.5 License	10
2.3 AVR Music Box	10
2.3.1 Hardware	11
2.3.2 Circuit	11
2.3.3 Code	12
2.3.4 Author	13
2.3.5 License	13
2.4 Light Meter	13
2.4.1 Hardware	14
2.4.2 Circuit	14
2.4.3 Code	15
2.4.4 Author	16
2.4.5 License	16
<b>3 README</b>	<b>17</b>
<b>4 Namespace Index</b>	<b>19</b>

---

4.1 Namespace List . . . . .	19
<b>5 Hierarchical Index</b>	<b>21</b>
5.1 Class Hierarchy . . . . .	21
<b>6 Class Index</b>	<b>23</b>
6.1 Class List . . . . .	23
<b>7 File Index</b>	<b>25</b>
7.1 File List . . . . .	25
<b>8 Namespace Documentation</b>	<b>27</b>
8.1 component Namespace Reference . . . . .	27
8.1.1 Detailed Description . . . . .	27
8.1.2 Enumeration Type Documentation . . . . .	27
8.1.2.1 mode . . . . .	27
8.2 core Namespace Reference . . . . .	28
8.2.1 Detailed Description . . . . .	29
8.2.2 Enumeration Type Documentation . . . . .	29
8.2.2.1 autoTriggerSource . . . . .	29
8.2.2.2 BODMode . . . . .	30
8.2.2.3 channel . . . . .	30
8.2.2.4 clockPrescaler . . . . .	31
8.2.2.5 clockSource . . . . .	31
8.2.2.6 compareOutputMode . . . . .	32
8.2.2.7 operationMode [1/2] . . . . .	32
8.2.2.8 operationMode [2/2] . . . . .	33
8.2.2.9 pinChangePort . . . . .	34
8.2.2.10 referenceVoltage . . . . .	34
8.2.2.11 resolution . . . . .	35
8.2.2.12 senseControl . . . . .	36
8.2.2.13 sleepMode . . . . .	36
8.2.2.14 timeOut . . . . .	36
8.3 io Namespace Reference . . . . .	37
8.3.1 Detailed Description . . . . .	38
8.3.2 Enumeration Type Documentation . . . . .	38
8.3.2.1 clockPrescaler . . . . .	38
8.3.2.2 communicationMode . . . . .	39
8.3.2.3 dataMode . . . . .	39
8.3.2.4 dataOrder . . . . .	40
8.3.2.5 frameSize . . . . .	40
8.3.2.6 operationMode . . . . .	41
8.3.2.7 parityMode . . . . .	41
8.3.2.8 stopBit . . . . .	42

---

8.3.2.9 transmissionMode	42
8.3.3 Variable Documentation	43
8.3.3.1 PortB	43
8.3.3.2 PortC	43
8.3.3.3 PortD	43
8.4 utils Namespace Reference	43
8.4.1 Function Documentation	43
8.4.1.1 map()	43
<b>9 Class Documentation</b>	<b>45</b>
9.1 core::ADConverter Class Reference	45
9.1.1 Detailed Description	46
9.1.2 Constructor & Destructor Documentation	46
9.1.2.1 ADConverter() [1/2]	46
9.1.2.2 ~ADConverter()	47
9.1.2.3 ADConverter() [2/2]	47
9.1.3 Member Function Documentation	47
9.1.3.1 conversionCompleteServiceRoutine()	47
9.1.3.2 enableAutoTrigger()	49
9.1.3.3 enableConversionCompleteInterrupt()	49
9.1.3.4 getConversionResult()	50
9.1.3.5 getInstance()	51
9.1.3.6 operator=()	51
9.1.3.7 selectAnalogInput()	52
9.1.3.8 selectAutoTriggerSource()	52
9.1.3.9 selectClockPrescaler()	52
9.1.3.10 selectReferenceVoltage()	53
9.1.3.11 start()	53
9.1.3.12 stop()	54
9.1.4 Member Data Documentation	54
9.1.4.1 __externally_visible__	54
9.1.4.2 __used__	54
9.1.4.3 m_conversionResult	54
9.1.4.4 m_resolution	54
9.2 core::AnalogComparator Class Reference	55
9.2.1 Detailed Description	55
9.3 component::Buzzer Class Reference	55
9.3.1 Detailed Description	56
9.3.2 Constructor & Destructor Documentation	56
9.3.2.1 Buzzer()	56
9.3.2.2 ~Buzzer()	57
9.3.3 Member Function Documentation	57

9.3.3.1 buzz() [1/2]	57
9.3.3.2 buzz() [2/2]	57
9.3.3.3 pause()	58
9.3.4 Member Data Documentation	58
9.3.4.1 m_pin	59
9.4 component::DCMotor Class Reference	59
9.4.1 Detailed Description	60
9.4.2 Constructor & Destructor Documentation	60
9.4.2.1 DCMotor()	60
9.4.2.2 ~DCMotor()	60
9.4.3 Member Function Documentation	60
9.4.3.1 connect()	60
9.4.3.2 disconnect()	61
9.4.3.3 off()	61
9.4.3.4 on()	61
9.4.3.5 spin()	61
9.4.3.6 stop()	62
9.4.3.7 toggle()	62
9.4.4 Member Data Documentation	62
9.4.4.1 m_pin	62
9.5 core::ExternInterrupt Class Reference	62
9.5.1 Detailed Description	63
9.5.2 Constructor & Destructor Documentation	63
9.5.2.1 ExternInterrupt() [1/2]	63
9.5.2.2 ~ExternInterrupt()	64
9.5.2.3 ExternInterrupt() [2/2]	64
9.5.3 Member Function Documentation	64
9.5.3.1 enableInt0()	64
9.5.3.2 enableInt1()	64
9.5.3.3 enablePinChange()	65
9.5.3.4 enablePinChangeMaskPortB()	65
9.5.3.5 enablePinChangeMaskPortC()	65
9.5.3.6 enablePinChangeMaskPortD()	66
9.5.3.7 getInstance()	66
9.5.3.8 Int0ServiceRoutine()	66
9.5.3.9 Int1ServiceRoutine()	66
9.5.3.10 operator=()	66
9.5.3.11 pinChangePortBServiceRoutine()	67
9.5.3.12 pinChangePortCServiceRoutine()	67
9.5.3.13 pinChangePortDServiceRoutine()	67
9.5.3.14 setInt0SenseControl()	67
9.5.3.15 setInt1SenseControl()	67

9.5.4 Member Data Documentation	67
9.5.4.1 __externally_visible__	68
9.5.4.2 __used__	68
9.6 component::LDR Class Reference	68
9.6.1 Detailed Description	69
9.6.2 Constructor & Destructor Documentation	69
9.6.2.1 LDR()	69
9.6.2.2 ~LDR()	70
9.6.3 Member Function Documentation	70
9.6.3.1 getLux()	70
9.6.3.2 getResistance()	70
9.6.4 Member Data Documentation	71
9.6.4.1 m_pin	71
9.6.4.2 mr_ADC	71
9.7 component::Led Class Reference	71
9.7.1 Detailed Description	72
9.7.2 Constructor & Destructor Documentation	73
9.7.2.1 Led()	73
9.7.2.2 ~Led()	73
9.7.3 Member Function Documentation	73
9.7.3.1 isOff()	74
9.7.3.2 isOn()	74
9.7.3.3 off()	74
9.7.3.4 on()	75
9.7.3.5 toggle()	75
9.7.4 Member Data Documentation	75
9.7.4.1 m_pin	75
9.8 core::MCU Class Reference	75
9.8.1 Detailed Description	76
9.8.2 Member Function Documentation	76
9.8.2.1 disableBOD()	76
9.8.2.2 enableADC()	76
9.8.2.3 enableSPI()	77
9.8.2.4 enableTimerCounter0()	77
9.8.2.5 enableTimerCounter1()	77
9.8.2.6 enableTimerCounter2()	78
9.8.2.7 enableTWI()	78
9.8.2.8 enableUSART0()	78
9.8.2.9 goToSleep()	79
9.8.2.10 init()	79
9.8.2.11 selectSleepMode()	79
9.8.2.12 sleepEnable()	80

9.9 io::Pin Class Reference . . . . .	80
9.9.1 Detailed Description . . . . .	81
9.9.2 Constructor & Destructor Documentation . . . . .	81
9.9.2.1 Pin() . . . . .	81
9.9.2.2 ~Pin() . . . . .	82
9.9.3 Member Function Documentation . . . . .	82
9.9.3.1 getPinNumber() . . . . .	82
9.9.3.2 isHigh() . . . . .	83
9.9.3.3 isLow() . . . . .	83
9.9.3.4 setHigh() . . . . .	83
9.9.3.5 setLow() . . . . .	84
9.9.3.6 toggle() . . . . .	84
9.9.3.7 toInput() . . . . .	84
9.9.3.8 toOutput() . . . . .	85
9.9.4 Member Data Documentation . . . . .	85
9.9.4.1 m_pinNumber . . . . .	85
9.9.4.2 mr_portName . . . . .	85
9.10 io::Port Struct Reference . . . . .	85
9.10.1 Detailed Description . . . . .	86
9.10.2 Member Data Documentation . . . . .	86
9.10.2.1 mp_ddrReg . . . . .	86
9.10.2.2 mp_pinReg . . . . .	86
9.10.2.3 mp_portReg . . . . .	86
9.11 component::PushButton Class Reference . . . . .	87
9.11.1 Detailed Description . . . . .	87
9.11.2 Constructor & Destructor Documentation . . . . .	88
9.11.2.1 PushButton() . . . . .	88
9.11.2.2 ~PushButton() . . . . .	88
9.11.3 Member Function Documentation . . . . .	89
9.11.3.1 getPressedCount() . . . . .	89
9.11.3.2 isPressed() . . . . .	89
9.11.3.3 resetPressedCount() . . . . .	90
9.11.4 Member Data Documentation . . . . .	90
9.11.4.1 m_buttonPressed . . . . .	90
9.11.4.2 m_pin . . . . .	90
9.11.4.3 mr_isActiveLow . . . . .	90
9.11.4.4 mr_useInternalPullUp . . . . .	91
9.12 component::ServoMotor Class Reference . . . . .	91
9.12.1 Detailed Description . . . . .	92
9.12.2 Constructor & Destructor Documentation . . . . .	92
9.12.2.1 ServoMotor() . . . . .	92
9.12.2.2 ~ServoMotor() . . . . .	92



9.12.3 Member Function Documentation	92
9.12.3.1 computePulseCycleCount()	93
9.12.3.2 computePulseWidthMaxCount()	93
9.12.3.3 computePulseWidthMidCount()	93
9.12.3.4 computePulseWidthMinCount()	93
9.12.3.5 computeRotationAngleCount()	94
9.12.3.6 connect()	94
9.12.3.7 disconnect()	94
9.12.3.8 off()	95
9.12.3.9 on()	95
9.12.3.10 rotate()	95
9.12.3.11 toggle()	95
9.12.4 Member Data Documentation	96
9.12.4.1 m_pin	96
9.12.4.2 m_pulseCycle	96
9.12.4.3 m_pulseWidthMax	96
9.12.4.4 m_pulseWidthMid	96
9.12.4.5 m_pulseWidthMin	97
9.13 io::SPI Class Reference	97
9.13.1 Detailed Description	98
9.13.2 Constructor & Destructor Documentation	98
9.13.2.1 SPI() [1/2]	99
9.13.2.2 ~SPI()	99
9.13.2.3 SPI() [2/2]	99
9.13.3 Member Function Documentation	99
9.13.3.1 enableTransferCompleteInterrupt()	100
9.13.3.2 getInstance()	100
9.13.3.3 masterReceiveByte()	100
9.13.3.4 masterSendByte()	101
9.13.3.5 operator=()	101
9.13.3.6 selectClockPrescaler()	101
9.13.3.7 selectDataMode()	101
9.13.3.8 selectDataOrder()	102
9.13.3.9 selectOperationMode()	102
9.13.3.10 selectSlave()	103
9.13.3.11 slaveReceiveByte()	103
9.13.3.12 transferComplete()	103
9.13.3.13 transferCompleteServiceRoutine()	103
9.13.3.14 writeCollision()	104
9.13.4 Member Data Documentation	104
9.13.4.1 __externally_visible__	104
9.13.4.2 __used__	104

9.13.4.3 m_data	104
9.13.4.4 m_pinMISO	104
9.13.4.5 m_pinMOSI	105
9.13.4.6 m_pinSCK	105
9.13.4.7 m_pinSS	105
9.14 component::StepperMotor Class Reference	105
9.14.1 Detailed Description	106
9.14.2 Constructor & Destructor Documentation	106
9.14.2.1 StepperMotor()	107
9.14.2.2 ~StepperMotor()	107
9.14.3 Member Function Documentation	107
9.14.3.1 computeStepDelay()	107
9.14.3.2 currentPos()	108
9.14.3.3 goalReached()	109
9.14.3.4 setCurrentPos()	109
9.14.3.5 step() [1/3]	109
9.14.3.6 step() [2/3]	110
9.14.3.7 step() [3/3]	111
9.14.3.8 stepDelay()	112
9.14.3.9 stepPulse()	112
9.14.4 Member Data Documentation	113
9.14.4.1 m_accelTime	113
9.14.4.2 m_constSpeedTime	113
9.14.4.3 m_currentPos	114
9.14.4.4 m_decelTime	114
9.14.4.5 m_goalReached	114
9.14.4.6 m_pinCoil1	114
9.14.4.7 m_pinCoil2	114
9.14.4.8 m_pinCoil3	115
9.14.4.9 m_pinCoil4	115
9.14.4.10 stepMode	115
9.15 core::TimerCounter Class Reference	115
9.15.1 Detailed Description	116
9.15.2 Member Function Documentation	116
9.15.2.1 enableOutputCompareMatchInterrupt()	116
9.15.2.2 enableOverflowInterrupt()	116
9.15.2.3 getClockPrescaler()	116
9.15.2.4 getCounter()	117
9.15.2.5 getOutputCompareRegister()	117
9.15.2.6 selectClockSource()	117
9.15.2.7 selectCompareOutputMode()	117
9.15.2.8 selectOperationMode()	117

9.15.2.9	setCounter()	117
9.15.2.10	setOutputCompareRegister()	118
9.15.2.11	start()	118
9.15.2.12	stop()	118
9.16	core::TimerCounter0 Class Reference	118
9.16.1	Detailed Description	120
9.16.2	Constructor & Destructor Documentation	120
9.16.2.1	TimerCounter0() [1/2]	120
9.16.2.2	~TimerCounter0()	120
9.16.2.3	TimerCounter0() [2/2]	121
9.16.3	Member Function Documentation	121
9.16.3.1	enableOutputCompareMatchInterrupt()	121
9.16.3.2	enableOverflowInterrupt()	122
9.16.3.3	getClockPrescaler()	122
9.16.3.4	getCounter()	122
9.16.3.5	getInstance()	123
9.16.3.6	getOutputCompareRegister()	123
9.16.3.7	operator=()	123
9.16.3.8	outputCompareMatchAServiceRoutine()	123
9.16.3.9	outputCompareMatchBServiceRoutine()	124
9.16.3.10	overflowServiceRoutine()	124
9.16.3.11	selectClockSource()	124
9.16.3.12	selectCompareOutputMode()	125
9.16.3.13	selectOperationMode()	125
9.16.3.14	setCounter()	126
9.16.3.15	setOutputCompareRegister()	126
9.16.3.16	start()	127
9.16.3.17	stop()	127
9.16.4	Member Data Documentation	127
9.16.4.1	__externally_visible__	127
9.16.4.2	__used__	127
9.16.4.3	m_clockPrescaler	128
9.16.4.4	m_clockSource	128
9.17	core::TimerCounter1 Class Reference	128
9.17.1	Detailed Description	130
9.17.2	Constructor & Destructor Documentation	130
9.17.2.1	TimerCounter1() [1/2]	130
9.17.2.2	~TimerCounter1()	130
9.17.2.3	TimerCounter1() [2/2]	130
9.17.3	Member Function Documentation	130
9.17.3.1	enableInputCaptureInterrupt()	131
9.17.3.2	enableOutputCompareMatchInterrupt()	131

9.17.3.3 enableOverflowInterrupt()	132
9.17.3.4 getClockPrescaler()	132
9.17.3.5 getCounter()	132
9.17.3.6 getInputCaptureRegister()	132
9.17.3.7 getInstance()	133
9.17.3.8 getOutputCompareRegister()	133
9.17.3.9 inputCaptureServiceRoutine()	133
9.17.3.10 operator=()	133
9.17.3.11 outputCompareMatchAServiceRoutine()	134
9.17.3.12 outputCompareMatchBServiceRoutine()	134
9.17.3.13 overflowServiceRoutine()	134
9.17.3.14 selectClockSource()	134
9.17.3.15 selectCompareOutputMode()	135
9.17.3.16 selectOperationMode()	135
9.17.3.17 setCounter()	137
9.17.3.18 setInputCaptureRegister()	137
9.17.3.19 setOutputCompareRegister()	137
9.17.3.20 start()	138
9.17.3.21 stop()	138
9.17.4 Member Data Documentation	138
9.17.4.1 __externally_visible__	138
9.17.4.2 __used__	138
9.17.4.3 m_clockPrescaler	139
9.17.4.4 m_clockSource	139
9.18 core::TimerCounter2 Class Reference	139
9.18.1 Detailed Description	140
9.18.2 Constructor & Destructor Documentation	141
9.18.2.1 TimerCounter2() [1/2]	141
9.18.2.2 ~TimerCounter2()	141
9.18.2.3 TimerCounter2() [2/2]	141
9.18.3 Member Function Documentation	141
9.18.3.1 enableOutputCompareMatchInterrupt()	142
9.18.3.2 enableOverflowInterrupt()	142
9.18.3.3 getClockPrescaler()	143
9.18.3.4 getCounter()	143
9.18.3.5 getInstance()	143
9.18.3.6 getOutputCompareRegister()	144
9.18.3.7 operator=()	144
9.18.3.8 outputCompareMatchAServiceRoutine()	144
9.18.3.9 outputCompareMatchBServiceRoutine()	144
9.18.3.10 overflowServiceRoutine()	144
9.18.3.11 selectClockSource()	145

9.18.3.12 selectCompareOutputMode()	145
9.18.3.13 selectOperationMode()	146
9.18.3.14 setCounter()	147
9.18.3.15 setOutputCompareRegister()	147
9.18.3.16 start()	147
9.18.3.17 stop()	148
9.18.4 Member Data Documentation	148
9.18.4.1 __externally_visible__	148
9.18.4.2 __used__	148
9.18.4.3 m_clockPrescaler	148
9.18.4.4 m_clockSource	148
9.19 io::USART0 Class Reference	149
9.19.1 Detailed Description	150
9.19.2 Constructor & Destructor Documentation	150
9.19.2.1 USART0() [1/2]	150
9.19.2.2 ~USART0()	151
9.19.2.3 USART0() [2/2]	151
9.19.3 Member Function Documentation	151
9.19.3.1 dataOverrun()	152
9.19.3.2 dataRegisterEmptyServiceRoutine()	152
9.19.3.3 enableDataRegisterEmptyInterrupt()	152
9.19.3.4 enableReceiveCompleteInterrupt()	153
9.19.3.5 enableTransmitCompleteInterrupt()	153
9.19.3.6 frameError()	154
9.19.3.7 getInstance()	154
9.19.3.8 getNumberBytesReceived()	155
9.19.3.9 getNumberBytesSent()	155
9.19.3.10 operator=()	155
9.19.3.11 parityError()	155
9.19.3.12 ready2Send()	156
9.19.3.13 receiveChar()	156
9.19.3.14 receiveCompleteServiceRoutine()	156
9.19.3.15 receiveFrame()	157
9.19.3.16 receiveString()	157
9.19.3.17 resetNumberBytesReceived()	158
9.19.3.18 sendByte()	158
9.19.3.19 sendChar()	158
9.19.3.20 sendFrame()	159
9.19.3.21 sendLong()	159
9.19.3.22 sendString()	160
9.19.3.23 sendWord()	160
9.19.3.24 setBaudRate()	160

9.19.3.25 setCommunicationMode()	161
9.19.3.26 setFrameSize()	161
9.19.3.27 setParityMode()	162
9.19.3.28 setStopBit()	163
9.19.3.29 setTransmissionMode()	163
9.19.3.30 transmitCompleteServiceRoutine()	164
9.19.4 Member Data Documentation	164
9.19.4.1 __externally_visible__	164
9.19.4.2 __used__	164
9.19.4.3 m_numberBytesReceived	164
9.19.4.4 m_numberBytesSent	165
9.19.4.5 m_ready2Send	165
9.19.4.6 m_sizeData2Receive	165
9.19.4.7 m_sizeData2Send	165
9.19.4.8 m_status	165
9.19.4.9 mp_data2Receive	166
9.19.4.10 mp_data2Send	166
9.20 core::WatchdogTimer Class Reference	166
9.20.1 Detailed Description	167
9.20.2 Constructor & Destructor Documentation	167
9.20.2.1 WatchdogTimer() [1/2]	167
9.20.2.2 ~WatchdogTimer()	167
9.20.2.3 WatchdogTimer() [2/2]	167
9.20.3 Member Function Documentation	167
9.20.3.1 getInstance()	168
9.20.3.2 operator=()	168
9.20.3.3 reset()	168
9.20.3.4 selectTimeOut()	168
9.20.3.5 start() [1/2]	169
9.20.3.6 start() [2/2]	169
9.20.3.7 stop()	169
9.20.3.8 timeOutServiceRoutine()	170
9.20.4 Member Data Documentation	170
9.20.4.1 __externally_visible__	170
9.20.4.2 __used__	170
9.20.4.3 m_operationMode	170
9.20.4.4 m_timeOut	170
<b>10 File Documentation</b>	<b>171</b>
10.1 ADC.cpp File Reference	171
10.2 ADC.cpp	171
10.3 ADC.h File Reference	175

10.4 ADC.h . . . . .	177
10.5 AnalogComparator.cpp File Reference . . . . .	178
10.6 AnalogComparator.cpp . . . . .	178
10.7 AnalogComparator.h File Reference . . . . .	178
10.7.1 Macro Definition Documentation . . . . .	179
10.7.1.1 ANALOG_COMARATOR_H . . . . .	179
10.8 AnalogComparator.h . . . . .	180
10.9 Buzzer.cpp File Reference . . . . .	180
10.10 Buzzer.cpp . . . . .	180
10.11 Buzzer.h File Reference . . . . .	181
10.12 Buzzer.h . . . . .	182
10.13 buzzer_pitches_16bit.h File Reference . . . . .	184
10.13.1 Macro Definition Documentation . . . . .	186
10.13.1.1 A0 . . . . .	186
10.13.1.2 A1 . . . . .	186
10.13.1.3 A2 . . . . .	186
10.13.1.4 A3 . . . . .	186
10.13.1.5 A4 . . . . .	187
10.13.1.6 A5 . . . . .	187
10.13.1.7 A6 . . . . .	187
10.13.1.8 A7 . . . . .	187
10.13.1.9 Ax0 . . . . .	187
10.13.1.10 Ax1 . . . . .	187
10.13.1.11 Ax2 . . . . .	188
10.13.1.12 Ax3 . . . . .	188
10.13.1.13 Ax4 . . . . .	188
10.13.1.14 Ax5 . . . . .	188
10.13.1.15 Ax6 . . . . .	188
10.13.1.16 Ax7 . . . . .	188
10.13.1.17 B0 . . . . .	189
10.13.1.18 B1 . . . . .	189
10.13.1.19 B2 . . . . .	189
10.13.1.20 B3 . . . . .	189
10.13.1.21 B4 . . . . .	189
10.13.1.22 B5 . . . . .	189
10.13.1.23 B6 . . . . .	190
10.13.1.24 B7 . . . . .	190
10.13.1.25 C0 . . . . .	190
10.13.1.26 C1 . . . . .	190
10.13.1.27 C2 . . . . .	190
10.13.1.28 C3 . . . . .	190
10.13.1.29 C4 . . . . .	191

10.13.1.30 C5 . . . . .	191
10.13.1.31 C6 . . . . .	191
10.13.1.32 C7 . . . . .	191
10.13.1.33 Cx0 . . . . .	191
10.13.1.34 Cx1 . . . . .	191
10.13.1.35 Cx2 . . . . .	192
10.13.1.36 Cx3 . . . . .	192
10.13.1.37 Cx4 . . . . .	192
10.13.1.38 Cx5 . . . . .	192
10.13.1.39 Cx6 . . . . .	192
10.13.1.40 Cx7 . . . . .	192
10.13.1.41 D0 . . . . .	193
10.13.1.42 D1 . . . . .	193
10.13.1.43 D2 . . . . .	193
10.13.1.44 D3 . . . . .	193
10.13.1.45 D4 . . . . .	193
10.13.1.46 D5 . . . . .	193
10.13.1.47 D6 . . . . .	194
10.13.1.48 D7 . . . . .	194
10.13.1.49 Dx0 . . . . .	194
10.13.1.50 Dx1 . . . . .	194
10.13.1.51 Dx2 . . . . .	194
10.13.1.52 Dx3 . . . . .	194
10.13.1.53 Dx4 . . . . .	195
10.13.1.54 Dx5 . . . . .	195
10.13.1.55 Dx6 . . . . .	195
10.13.1.56 Dx7 . . . . .	195
10.13.1.57 E0 . . . . .	195
10.13.1.58 E1 . . . . .	195
10.13.1.59 E2 . . . . .	196
10.13.1.60 E3 . . . . .	196
10.13.1.61 E4 . . . . .	196
10.13.1.62 E5 . . . . .	196
10.13.1.63 E6 . . . . .	196
10.13.1.64 E7 . . . . .	196
10.13.1.65 F0 . . . . .	197
10.13.1.66 F1 . . . . .	197
10.13.1.67 F2 . . . . .	197
10.13.1.68 F3 . . . . .	197
10.13.1.69 F4 . . . . .	197
10.13.1.70 F5 . . . . .	197
10.13.1.71 F6 . . . . .	198



10.13.1.72 F7 . . . . .	198
10.13.1.73 Fx0 . . . . .	198
10.13.1.74 Fx1 . . . . .	198
10.13.1.75 Fx2 . . . . .	198
10.13.1.76 Fx3 . . . . .	198
10.13.1.77 Fx4 . . . . .	199
10.13.1.78 Fx5 . . . . .	199
10.13.1.79 Fx6 . . . . .	199
10.13.1.80 Fx7 . . . . .	199
10.13.1.81 G0 . . . . .	199
10.13.1.82 G1 . . . . .	199
10.13.1.83 G2 . . . . .	200
10.13.1.84 G3 . . . . .	200
10.13.1.85 G4 . . . . .	200
10.13.1.86 G5 . . . . .	200
10.13.1.87 G6 . . . . .	200
10.13.1.88 G7 . . . . .	200
10.13.1.89 Gx0 . . . . .	201
10.13.1.90 Gx1 . . . . .	201
10.13.1.91 Gx2 . . . . .	201
10.13.1.92 Gx3 . . . . .	201
10.13.1.93 Gx4 . . . . .	201
10.13.1.94 Gx5 . . . . .	201
10.13.1.95 Gx6 . . . . .	202
10.13.1.96 Gx7 . . . . .	202
10.14 buzzer_pitches_16bit.h . . . . .	202
10.15 buzzer_pitches_8bit.h File Reference . . . . .	203
10.15.1 Macro Definition Documentation . . . . .	204
10.15.1.1 A0 . . . . .	204
10.15.1.2 A1 . . . . .	204
10.15.1.3 A2 . . . . .	204
10.15.1.4 Ax0 . . . . .	205
10.15.1.5 Ax1 . . . . .	205
10.15.1.6 Ax2 . . . . .	205
10.15.1.7 B0 . . . . .	205
10.15.1.8 B1 . . . . .	205
10.15.1.9 B2 . . . . .	205
10.15.1.10 C1 . . . . .	206
10.15.1.11 C2 . . . . .	206
10.15.1.12 C3 . . . . .	206
10.15.1.13 Cx0 . . . . .	206
10.15.1.14 Cx2 . . . . .	206

10.15.1.15 Cx3 . . . . .	206
10.15.1.16 D1 . . . . .	207
10.15.1.17 D2 . . . . .	207
10.15.1.18 D3 . . . . .	207
10.15.1.19 Dx0 . . . . .	207
10.15.1.20 Dx2 . . . . .	207
10.15.1.21 Dx3 . . . . .	207
10.15.1.22 E1 . . . . .	208
10.15.1.23 E2 . . . . .	208
10.15.1.24 E3 . . . . .	208
10.15.1.25 F1 . . . . .	208
10.15.1.26 F2 . . . . .	208
10.15.1.27 F3 . . . . .	208
10.15.1.28 Fx1 . . . . .	209
10.15.1.29 Fx2 . . . . .	209
10.15.1.30 Fx3 . . . . .	209
10.15.1.31 G1 . . . . .	209
10.15.1.32 G2 . . . . .	209
10.15.1.33 G3 . . . . .	209
10.15.1.34 Gx0 . . . . .	210
10.15.1.35 Gx1 . . . . .	210
10.15.1.36 Gx2 . . . . .	210
10.16 buzzer_pitches_8bit.h . . . . .	210
10.17 DCMotor.cpp File Reference . . . . .	211
10.18 DCMotor.cpp . . . . .	211
10.19 DCMotor.h File Reference . . . . .	211
10.19.1 Detailed Description . . . . .	212
10.20 DCMotor.h . . . . .	214
10.21 ExternInterrupt.cpp File Reference . . . . .	214
10.22 ExternInterrupt.cpp . . . . .	215
10.23 ExternInterrupt.h File Reference . . . . .	216
10.23.1 Detailed Description . . . . .	217
10.24 ExternInterrupt.h . . . . .	219
10.25 ha_base.h File Reference . . . . .	220
10.25.1 Detailed Description . . . . .	220
10.25.2 Macro Definition Documentation . . . . .	221
10.25.2.1 STR . . . . .	221
10.25.2.2 STRx . . . . .	221
10.26 ha_base.h . . . . .	221
10.27 ha_m328p.h File Reference . . . . .	222
10.27.1 Detailed Description . . . . .	225
10.27.2 Macro Definition Documentation . . . . .	226

10.27.2.1 ADC_ADJUST_RESULT_LEFT . . . . .	226
10.27.2.2 ADC_ADJUST_RESULT_RIGHT . . . . .	226
10.27.2.3 ADC_CONVERSION_COMPLETE_INTERRUPT . . . . .	226
10.27.2.4 ADC_DISABLE . . . . .	226
10.27.2.5 ADC_DISABLE_AUTOTRIGGER . . . . .	227
10.27.2.6 ADC_DISABLE_CONVERSION_COMPLETE_INTERRUPT . . . . .	227
10.27.2.7 ADC_DISABLE_DIGITAL_INPUT_REGISTER . . . . .	227
10.27.2.8 ADC_ENABLE . . . . .	227
10.27.2.9 ADC_ENABLE_AUTOTRIGGER . . . . .	227
10.27.2.10 ADC_ENABLE_CONVERSION_COMPLETE_INTERRUPT . . . . .	227
10.27.2.11 ADC_SELECT_ANALOG_INPUT . . . . .	228
10.27.2.12 ADC_SELECT_AUTO_TRIGGER_SOURCE . . . . .	228
10.27.2.13 ADC_SELECT_CLOCK_PRESCALER . . . . .	228
10.27.2.14 ADC_SELECT_REF_VOLTAGE . . . . .	228
10.27.2.15 ADC_START_CONVERSION . . . . .	228
10.27.2.16 ADC_STOP_CONVERSION . . . . .	228
10.27.2.17 EXT_INT_DISABLE_INT0 . . . . .	229
10.27.2.18 EXT_INT_DISABLE_INT1 . . . . .	229
10.27.2.19 EXT_INT_DISABLE_PIN_CHANGE_INTERRUPT . . . . .	229
10.27.2.20 EXT_INT_DISABLE_PIN_CHANGE_MASK_PORTB . . . . .	229
10.27.2.21 EXT_INT_DISABLE_PIN_CHANGE_MASK_PORTC . . . . .	229
10.27.2.22 EXT_INT_DISABLE_PIN_CHANGE_MASK_PORTD . . . . .	229
10.27.2.23 EXT_INT_ENABLE_INT0 . . . . .	230
10.27.2.24 EXT_INT_ENABLE_INT1 . . . . .	230
10.27.2.25 EXT_INT_ENABLE_PIN_CHANGE_INTERRUPT . . . . .	230
10.27.2.26 EXT_INT_ENABLE_PIN_CHANGE_MASK_PORTB . . . . .	230
10.27.2.27 EXT_INT_ENABLE_PIN_CHANGE_MASK_PORTC . . . . .	230
10.27.2.28 EXT_INT_ENABLE_PIN_CHANGE_MASK_PORTD . . . . .	230
10.27.2.29 EXT_INT_INT0_INTERRUPT . . . . .	231
10.27.2.30 EXT_INT_INT1_INTERRUPT . . . . .	231
10.27.2.31 EXT_INT_PIN_CHANGE_PORTB_INTERRUPT . . . . .	231
10.27.2.32 EXT_INT_PIN_CHANGE_PORTC_INTERRUPT . . . . .	231
10.27.2.33 EXT_INT_PIN_CHANGE_PORTD_INTERRUPT . . . . .	231
10.27.2.34 EXT_INT_SET_INT0_SENSE_CONTROL . . . . .	231
10.27.2.35 EXT_INT_SET_INT1_SENSE_CONTROL . . . . .	232
10.27.2.36 MCU_ADC_DISABLE . . . . .	232
10.27.2.37 MCU_ADC_ENABLE . . . . .	232
10.27.2.38 MCU_BOD_DISABLE . . . . .	232
10.27.2.39 MCU_SELECT_SLEEP_MODE . . . . .	232
10.27.2.40 MCU_SLEEP_DISABLE . . . . .	232
10.27.2.41 MCU_SLEEP_ENABLE . . . . .	233
10.27.2.42 MCU_SPI_DISABLE . . . . .	233

10.27.2.43 MCU_SPI_ENABLE . . . . .	233
10.27.2.44 MCU_TIMER0_DISABLE . . . . .	233
10.27.2.45 MCU_TIMER0_ENABLE . . . . .	233
10.27.2.46 MCU_TIMER1_DISABLE . . . . .	233
10.27.2.47 MCU_TIMER1_ENABLE . . . . .	234
10.27.2.48 MCU_TIMER2_DISABLE . . . . .	234
10.27.2.49 MCU_TIMER2_ENABLE . . . . .	234
10.27.2.50 MCU_TWI_DISABLE . . . . .	234
10.27.2.51 MCU_TWI_ENABLE . . . . .	234
10.27.2.52 MCU_USART0_DISABLE . . . . .	234
10.27.2.53 MCU_USART0_ENABLE . . . . .	235
10.27.2.54 PUSHBUTTON_DEBOUNCE_TIME_US . . . . .	235
10.27.2.55 PUSHBUTTON_SAMPLING . . . . .	235
10.27.2.56 SERVOMOTOR_TIMER_ANGLE_COUNT . . . . .	235
10.27.2.57 SERVOMOTOR_TIMER_PULSE_WIDTH_COUNT . . . . .	235
10.27.2.58 SPI_CONTROL_REGISTER . . . . .	236
10.27.2.59 SPI_DATA_REGISTER . . . . .	236
10.27.2.60 SPI_DISABLE . . . . .	236
10.27.2.61 SPI_DISABLE_TRANSFER_COMPLETE_INTERRUPT . . . . .	236
10.27.2.62 SPI_ENABLE . . . . .	236
10.27.2.63 SPI_ENABLE_TRANSFER_COMPLETE_INTERRUPT . . . . .	236
10.27.2.64 SPI_MASTER_MODE . . . . .	237
10.27.2.65 SPI_SELECT_CLOCK_PRESCALER . . . . .	237
10.27.2.66 SPI_SELECT_DATA_MODE . . . . .	237
10.27.2.67 SPI_SELECT_DATA_ORDER . . . . .	237
10.27.2.68 SPI_SELECT_MASTER_MODE . . . . .	237
10.27.2.69 SPI_SELECT_SLAVE_MODE . . . . .	237
10.27.2.70 SPI_STATUS_REGISTER . . . . .	238
10.27.2.71 SPI_TRANSFER_COMPLETE . . . . .	238
10.27.2.72 SPI_TRANSFER_COMPLETE_INTERRUPT . . . . .	238
10.27.2.73 SPI_WRITE_COLLISION . . . . .	238
10.27.2.74 TIMER0_COM_CHANNEL_A_INTERRUPT . . . . .	238
10.27.2.75 TIMER0_COM_CHANNEL_B_INTERRUPT . . . . .	238
10.27.2.76 TIMER0_DISABLE_COM_CHANNEL_A_INTERRUPT . . . . .	239
10.27.2.77 TIMER0_DISABLE_COM_CHANNEL_B_INTERRUPT . . . . .	239
10.27.2.78 TIMER0_DISABLE_OVERFLOW_INTERRUPT . . . . .	239
10.27.2.79 TIMER0_ENABLE_COM_CHANNEL_A_INTERRUPT . . . . .	239
10.27.2.80 TIMER0_ENABLE_COM_CHANNEL_B_INTERRUPT . . . . .	239
10.27.2.81 TIMER0_ENABLE_OVERFLOW_INTERRUPT . . . . .	239
10.27.2.82 TIMER0_OVERFLOW_INTERRUPT . . . . .	240
10.27.2.83 TIMER0_SELECT_CLOCK_SOURCE . . . . .	240
10.27.2.84 TIMER0_SELECT_COM_CHANNEL_A . . . . .	240

10.27.2.85	TIMER0_SELECT_COM_CHANNEL_B	240
10.27.2.86	TIMER0_SELECT_OPERATION_MODE	240
10.27.2.87	TIMER0_STOP	240
10.27.2.88	TIMER1_COM_CHANNEL_A_INTERRUPT	241
10.27.2.89	TIMER1_COM_CHANNEL_B_INTERRUPT	241
10.27.2.90	TIMER1_DISABLE_COM_CHANNEL_A_INTERRUPT	241
10.27.2.91	TIMER1_DISABLE_COM_CHANNEL_B_INTERRUPT	241
10.27.2.92	TIMER1_DISABLE_INPUT_CAPTURE_INTERRUPT	241
10.27.2.93	TIMER1_DISABLE_OVERFLOW_INTERRUPT	241
10.27.2.94	TIMER1_ENABLE_COM_CHANNEL_A_INTERRUPT	242
10.27.2.95	TIMER1_ENABLE_COM_CHANNEL_B_INTERRUPT	242
10.27.2.96	TIMER1_ENABLE_INPUT_CAPTURE_INTERRUPT	242
10.27.2.97	TIMER1_ENABLE_OVERFLOW_INTERRUPT	242
10.27.2.98	TIMER1_INPUT_CAPTURE_INTERRUPT	242
10.27.2.99	TIMER1_OVERFLOW_INTERRUPT	242
10.27.2.100	TIMER1_SELECT_CLOCK_SOURCE	243
10.27.2.101	TIMER1_SELECT_COM_CHANNEL_A	243
10.27.2.102	TIMER1_SELECT_COM_CHANNEL_B	243
10.27.2.103	TIMER1_SELECT_OPERATION_MODE	243
10.27.2.104	TIMER1_STOP	243
10.27.2.105	TIMER2_COM_CHANNEL_A_INTERRUPT	243
10.27.2.106	TIMER2_COM_CHANNEL_B_INTERRUPT	244
10.27.2.107	TIMER2_DISABLE_COM_CHANNEL_A_INTERRUPT	244
10.27.2.108	TIMER2_DISABLE_COM_CHANNEL_B_INTERRUPT	244
10.27.2.109	TIMER2_DISABLE_OVERFLOW_INTERRUPT	244
10.27.2.110	TIMER2_ENABLE_COM_CHANNEL_A_INTERRUPT	244
10.27.2.111	TIMER2_ENABLE_COM_CHANNEL_B_INTERRUPT	244
10.27.2.112	TIMER2_ENABLE_OVERFLOW_INTERRUPT	245
10.27.2.113	TIMER2_OVERFLOW_INTERRUPT	245
10.27.2.114	TIMER2_SELECT_CLOCK_SOURCE	245
10.27.2.115	TIMER2_SELECT_COM_CHANNEL_A	245
10.27.2.116	TIMER2_SELECT_COM_CHANNEL_B	245
10.27.2.117	TIMER2_SELECT_OPERATION_MODE	245
10.27.2.118	TIMER2_STOP	246
10.27.2.119	USART0_CONTROL_STATUS_REGISTER	246
10.27.2.120	USART0_DATA_OVERRUN	246
10.27.2.121	USART0_DATA_REGISTER	246
10.27.2.122	USART0_DATA_REGISTER_EMPTY_INTERRUPT	246
10.27.2.123	USART0_DISABLE_DATA_REGISTER_EMPTY_INTERRUPT	246
10.27.2.124	USART0_DISABLE_DOUBLE_SPEED_MODE <sup>[1/2]</sup>	247
10.27.2.125	USART0_DISABLE_DOUBLE_SPEED_MODE <sup>[2/2]</sup>	247
10.27.2.126	USART0_DISABLE_PARITY_MODE	247

10.27.2.127 USART0_DISABLE_RECEIVE_COMPLETE_INTERRUPT . . . . .	247
10.27.2.128 USART0_DISABLE_RECEIVER . . . . .	247
10.27.2.129 USART0_DISABLE_TRANSMIT_COMPLETE_INTERRUPT . . . . .	247
10.27.2.130 USART0_DISABLE_TRANSMITTER . . . . .	248
10.27.2.131 USART0_ENABLE_ASYNC_TRANSMISSION_MODE . . . . .	248
10.27.2.132 USART0_ENABLE_DATA_REGISTER_EMPTY_INTERRUPT . . . . .	248
10.27.2.133 USART0_ENABLE_DOUBLE_SPEED_MODE . . . . .	248
10.27.2.134 USART0_ENABLE_EVEN_PARITY_MODE . . . . .	248
10.27.2.135 USART0_ENABLE_MASTER_SPI_MODE . . . . .	248
10.27.2.136 USART0_ENABLE_ODD_PARITY_MODE . . . . .	249
10.27.2.137 USART0_ENABLE_RECEIVE_COMPLETE_INTERRUPT . . . . .	249
10.27.2.138 USART0_ENABLE_RECEIVER . . . . .	249
10.27.2.139 USART0_ENABLE_SYNC_TRANSMISSION_MODE . . . . .	249
10.27.2.140 USART0_ENABLE_TRANSMIT_COMPLETE_INTERRUPT . . . . .	249
10.27.2.141 USART0_ENABLE_TRANSMITTER . . . . .	249
10.27.2.142 USART0_FRAME_ERROR . . . . .	250
10.27.2.143 USART0_PARITY_ERROR . . . . .	250
10.27.2.144 USART0_RECEIVE_COMPLETE_INTERRUPT . . . . .	250
10.27.2.145 USART0_SET_5BIT_FRAME_SIZE . . . . .	250
10.27.2.146 USART0_SET_6BIT_FRAME_SIZE . . . . .	250
10.27.2.147 USART0_SET_7BIT_FRAME_SIZE . . . . .	251
10.27.2.148 USART0_SET_8BIT_FRAME_SIZE . . . . .	251
10.27.2.149 USART0_SET_9BIT_FRAME_SIZE . . . . .	251
10.27.2.150 USART0_SET_BAUDRATE_HIGH_REGISTER . . . . .	251
10.27.2.151 USART0_SET_BAUDRATE_LOW_REGISTER . . . . .	251
10.27.2.152 USART0_SET_ONE_STOP_BIT . . . . .	252
10.27.2.153 USART0_SET_TWO_STOP_BITS . . . . .	252
10.27.2.154 USART0_TRANSMIT_COMPLETE_INTERRUPT . . . . .	252
10.27.2.155 WATCHDOG_SELECT_TIMEOUT . . . . .	252
10.27.2.156 WATCHDOG_START . . . . .	252
10.27.2.157 WATCHDOG_STOP . . . . .	252
10.27.2.158 WATCHDOG_TIMEOUT_INTERRUPT . . . . .	253
10.28 ha_m328p.h . . . . .	253
10.29 LCD.cpp File Reference . . . . .	257
10.30 LCD.cpp . . . . .	257
10.31 LCD.h File Reference . . . . .	257
10.32 LCD.h . . . . .	257
10.33 LDR.cpp File Reference . . . . .	257
10.34 LDR.cpp . . . . .	257
10.35 LDR.h File Reference . . . . .	258
10.36 LDR.h . . . . .	259
10.37 Led.cpp File Reference . . . . .	259

---

10.38 Led.cpp . . . . .	260
10.39 Led.h File Reference . . . . .	260
10.40 Led.h . . . . .	261
10.41 main.cpp File Reference . . . . .	262
10.41.1 Detailed Description . . . . .	262
10.41.2 Macro Definition Documentation . . . . .	263
10.41.2.1 LED . . . . .	263
10.41.2.2 TIMEDELAY . . . . .	263
10.41.3 Function Documentation . . . . .	263
10.41.3.1 main() . . . . .	263
10.42 BlinkLed/main.cpp . . . . .	264
10.43 main.cpp File Reference . . . . .	264
10.43.1 Detailed Description . . . . .	265
10.43.2 Macro Definition Documentation . . . . .	265
10.43.2.1 LDRVD . . . . .	265
10.43.2.2 LED0 . . . . .	265
10.43.2.3 LED1 . . . . .	266
10.43.2.4 LED2 . . . . .	266
10.43.2.5 LED3 . . . . .	266
10.43.2.6 LED4 . . . . .	266
10.43.2.7 LED5 . . . . .	266
10.43.2.8 LED6 . . . . .	267
10.43.2.9 LED7 . . . . .	267
10.43.3 Function Documentation . . . . .	267
10.43.3.1 main() . . . . .	267
10.44 main.cpp . . . . .	268
10.45 main.cpp File Reference . . . . .	268
10.45.1 Detailed Description . . . . .	269
10.45.2 Macro Definition Documentation . . . . .	270
10.45.2.1 BUZZER . . . . .	270
10.45.2.2 PUSHBUTTON . . . . .	270
10.45.2.3 SONG_LENGTH . . . . .	270
10.45.3 Function Documentation . . . . .	270
10.45.3.1 main() . . . . .	271
10.46 MusicBox/main.cpp . . . . .	271
10.47 main.cpp File Reference . . . . .	272
10.47.1 Detailed Description . . . . .	273
10.47.2 Macro Definition Documentation . . . . .	274
10.47.2.1 BUZZER . . . . .	274
10.47.2.2 NOTE_DURATION . . . . .	274
10.47.3 Function Documentation . . . . .	274
10.47.3.1 main() . . . . .	274

10.48 Organ/main.cpp . . . . .	275
10.49 MCU.cpp File Reference . . . . .	276
10.50 MCU.cpp . . . . .	277
10.51 MCU.h File Reference . . . . .	278
10.51.1 Detailed Description . . . . .	279
10.52 MCU.h . . . . .	280
10.53 Pin.cpp File Reference . . . . .	281
10.54 Pin.cpp . . . . .	281
10.55 Pin.h File Reference . . . . .	282
10.56 Pin.h . . . . .	284
10.57 PushButton.cpp File Reference . . . . .	284
10.58 PushButton.cpp . . . . .	285
10.59 PushButton.h File Reference . . . . .	285
10.60 PushButton.h . . . . .	286
10.61 README.md File Reference . . . . .	287
10.62 README.md File Reference . . . . .	287
10.63 README.md File Reference . . . . .	287
10.64 README.md File Reference . . . . .	287
10.65 README.md File Reference . . . . .	287
10.66 README.md File Reference . . . . .	287
10.67 README.md File Reference . . . . .	287
10.68 ServoMotor.cpp File Reference . . . . .	287
10.69 ServoMotor.cpp . . . . .	288
10.70 ServoMotor.h File Reference . . . . .	289
10.70.1 Detailed Description . . . . .	290
10.71 ServoMotor.h . . . . .	293
10.72 SPI.cpp File Reference . . . . .	293
10.73 SPI.cpp . . . . .	294
10.74 SPI.h File Reference . . . . .	296
10.74.1 Detailed Description . . . . .	297
10.75 SPI.h . . . . .	298
10.76 StepperMotor.cpp File Reference . . . . .	299
10.77 StepperMotor.cpp . . . . .	299
10.78 StepperMotor.h File Reference . . . . .	304
10.79 StepperMotor.h . . . . .	305
10.80 TimerCounter.h File Reference . . . . .	306
10.80.1 Detailed Description . . . . .	307
10.81 TimerCounter.h . . . . .	316
10.82 TimerCounter0.cpp File Reference . . . . .	318
10.83 TimerCounter0.cpp . . . . .	318
10.84 TimerCounter0.h File Reference . . . . .	321
10.85 TimerCounter0.h . . . . .	322



10.86 TimerCounter1.cpp File Reference . . . . .	323
10.87 TimerCounter1.cpp . . . . .	324
10.88 TimerCounter1.h File Reference . . . . .	328
10.89 TimerCounter1.h . . . . .	329
10.90 TimerCounter2.cpp File Reference . . . . .	330
10.91 TimerCounter2.cpp . . . . .	330
10.92 TimerCounter2.h File Reference . . . . .	334
10.93 TimerCounter2.h . . . . .	335
10.94 USART0.cpp File Reference . . . . .	336
10.95 USART0.cpp . . . . .	336
10.96 USART0.h File Reference . . . . .	341
10.97 USART0.h . . . . .	342
10.98 utils_m328p.cpp File Reference . . . . .	344
10.99 utils_m328p.cpp . . . . .	344
10.100 utils_m328p.h File Reference . . . . .	344
10.101 utils_m328p.h . . . . .	345
10.102 WatchdogTimer.cpp File Reference . . . . .	345
10.103 WatchdogTimer.cpp . . . . .	345
10.104 WatchdogTimer.h File Reference . . . . .	346
10.104.1 Detailed Description . . . . .	348
10.105 WatchdogTimer.h . . . . .	349
<b>Index</b>	<b>351</b>



# Chapter 1

## Introduction

**TahakomAVRLib** is a C++ library to program Atmel AVR microcontrollers. The library make use of the AVR standard C library and is written for easy usage.

Currently, the classes compile on the **ATmega48P/88P/168P/328P AVR microcontrollers** family.

### 1.1 Library structure

The library is composed of several classes that abstract the internal elements of a microcontroller and some external components that when hooked up to the chip can perform some actions.

These classes implement all functionalities and are organized into the namespaces:

- core
- io (input/output)
- or components

Several [Applications](#) and [Projects](#) are implemented in order to demonstrate the library usage.

A more detailed description of the code listings can be found in the [library documentation](#) (generated by Doxygen)

### 1.2 Software setup

Before using the library and start programming and interfacing external peripherals, some software packages need to be installed in your system:

- binutils-avr: for getting tools like assembler, linker, ...
- gcc-avr : AVR GNU C cross-compiler
- avr-libc: AVR C library
- avrdude : driver program for downloading/uploading code and data from/to Atmel AVR microcontrollers

### 1.2.1 Linux

In Linux (Ubuntu in my case), the software packages can be installed as follows:

```
sudo apt-get update
sudo apt-get install gcc build-essential
sudo apt-get install gcc-avr binutils-avr avr-libc gdb-avr
sudo apt-get install avrdude
```

### 1.2.2 Windows

In Windows, the software packages can be downloaded and installed as follows:

- AVR toolchain for windows: can be downloaded from the Atmel's website [download](#)
- avrdude: can be downloaded from [download](#)
- make: can be downloaded from [download](#)
- cmake: binary distribution can be downloaded from [download](#)

An alternative, would be to download the precompiled AVR-GCC toolchain from [download](#) that also include avrdude and make utilities but not cmake.

Before using these software tools, you need to update the **PATH** environment variable with the file paths to their executables and restart the system.

## 1.3 Compile and Flash programs

To compile and flash a program code to the AVR chip via USB port, you need to:

- Install the USB driver for the programmer used
  - Adapt the CMakeLists.txt parameters to your system configuration:
    - **MCU**: AVR chip used
    - **F\_CPU**: AVR CPU frequency
    - **BAUD**: Baude rate for serial communication
    - **PROG\_TYPE**: Programmer type
    - **AVRFLASH\_PORT**: Flash port name
  - Change compiler flags if necessary
- and execute the following steps (shown for the [Blink a Led](#) application):

### 1.3.1 Linux

```
../BlinkLed$ mkdir build
../BlinkLed$ cd build
../BlinkLed/build$ cmake ..
../BlinkLed/build$ make flash
```

### 1.3.2 Windows

```
../BlinkLed$ mkdir build
../BlinkLed$ cd build
../BlinkLed/build$ cmake .. -G "Unix Makefiles"
../BlinkLed/build$ make flash
```

## 1.4 Hardware setup

- AVR chip either barebone or on a development board like an Arduino UNO
- A bunch of Leds, resistors, drivers, sensors and actuators
- A breadboard
- Jumper wires
- An AVR ISPProgrammer (optional)

## 1.5 Applications

These applications demonstrate the usage of **TahakomAVRLib** in simple examples:

- [Blink a Led](#)
- [AVR Square-Wave Organ](#)
- [AVR Music Box](#)
- [AVR Light Meter](#)

## 1.6 Projects

These are more complex projects implemented using **TahakomAVRLib**

## 1.7 Author

- Farid Oubbati
- Date: 12-May-2018
- Copyright (c) 2018

## 1.8 License

This project is licensed under the MIT License - see the [LICENSE.txt](#) file for more details



## Chapter 2

# Applications

These applications demonstrate the usage of **TahakomAVRLib** in simple examples. Some of the examples are directly inspired from the book *Make: AVR Programming*, [Elliot Williams](#):

- [Blink a Led](#)
- [AVR Square-Wave Organ](#)
- [AVR Music Box](#)
- [Light Meter](#)

### 2.1 Blink a Led

This example demonstrates the library usage in a simple blink a Led example. It also shows how to use the Pin and Led abstraction objects.

The pinout of the **ATmega48P/88P/168P/328P AVR microcontrollers** family is illustrated below:

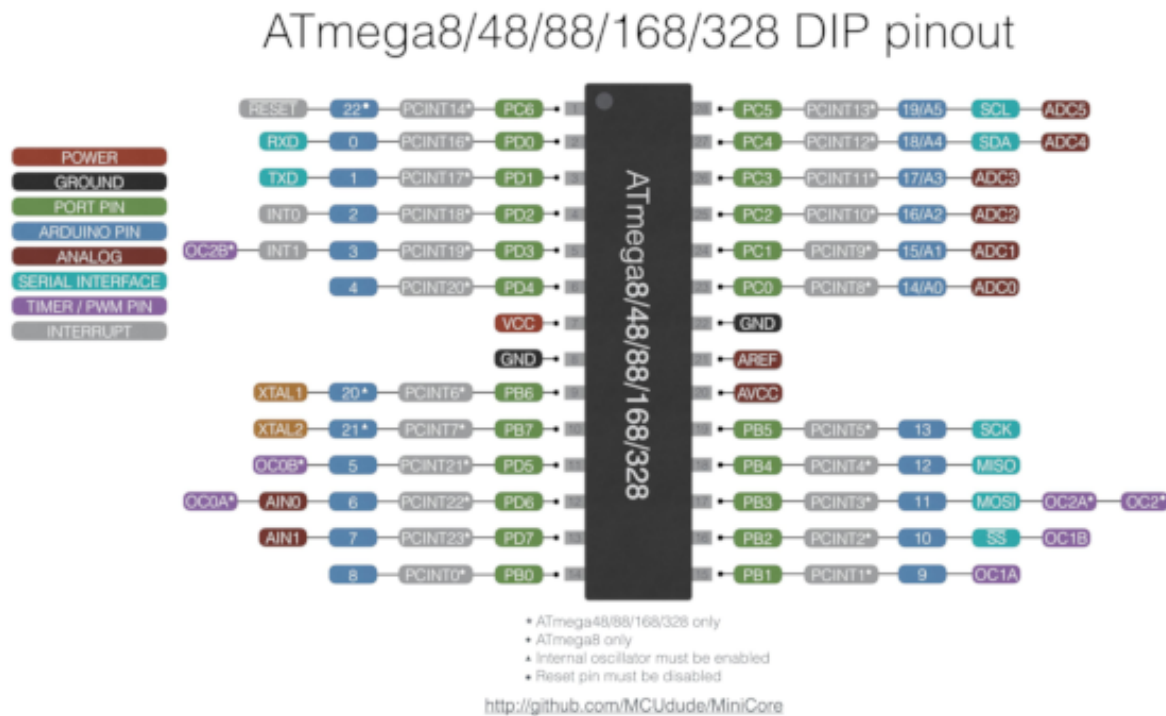


Figure 2.1 Pinout of ATmega48P/88P/168P/328P AVR microcontrollers family

### 2.1.1 Hardware

- Arduino UNO
- generic Led
- 220 ohm current limiting resistor
- A breadboard

### 2.1.2 Circuit

The Led with the current limiting resistor are connected to pin PB0 (digital pin 8 in Arduino UNO).

The circuit connection is as follows:



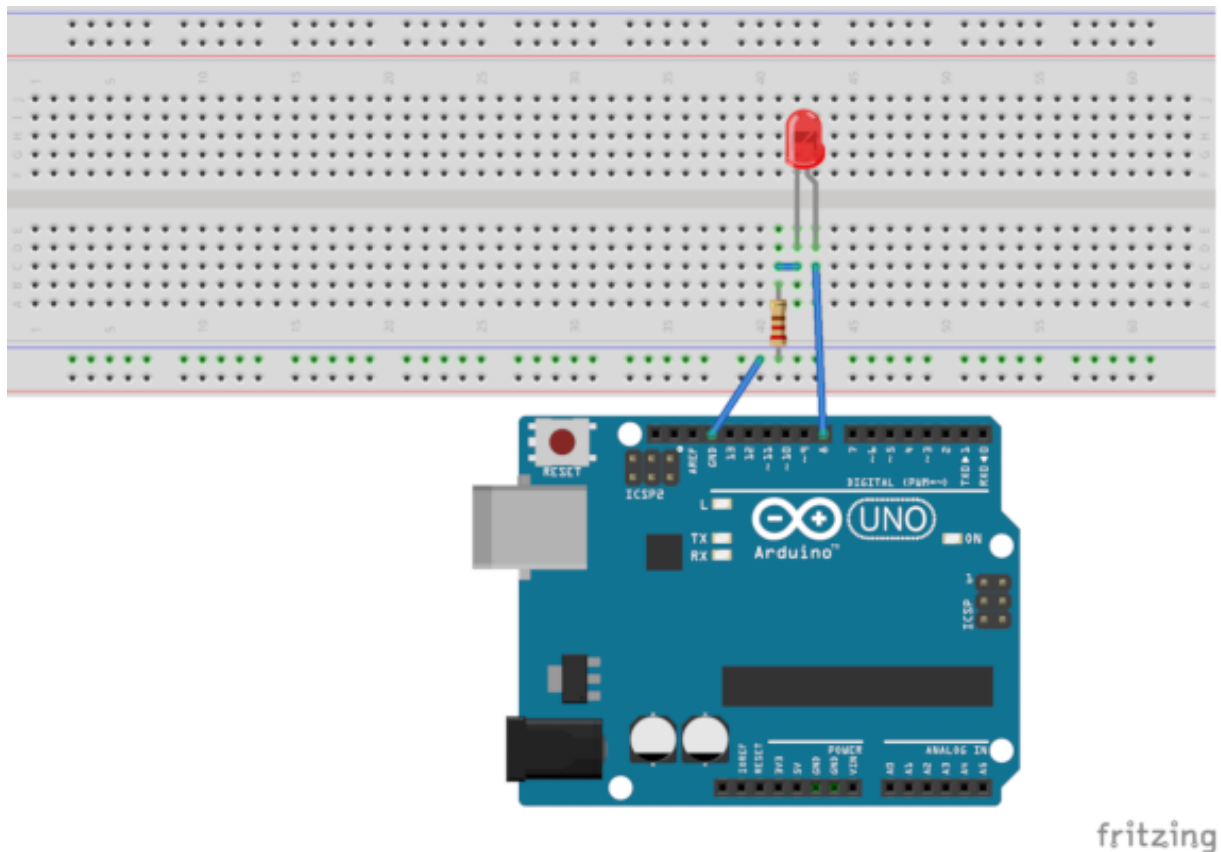


Figure 2.2 Circuit diagram

### 2.1.3 Code

The following code blinks the Led with a delay of 500 ms:

```
#include "Led.h"
#define LED 0 /**< Led pin number */
#define TIMEDELAY 500 /**< Time delay */
int main(void) {
    // Init
    // Instantiate a Led object
    component::Led Led(io::Pin(LED,io::PortB));
    // Mainloop
    while (1) {
        Led.on();
        _delay_ms(TIMEDELAY);
        Led.off();
        _delay_ms(TIMEDELAY);
    }
    return 0;
}
```

### 2.1.4 Author

- Farid Oubbati
- Date: 12-May-2018
- Copyright (c) 2018

## 2.1.5 License

This project is licensed under the MIT License - see the [LICENSE.txt](#) file for more details

## 2.2 AVR Square-Wave Organ

This example enables the user to enter a key note via a serial terminal and play the corresponding note in a Buzzer. It also shows how to use the Buzzer and USART0 abstraction objects.

The pinout of the **ATmega48P/88P/168P/328P AVR microcontrollers** family is illustrated below:

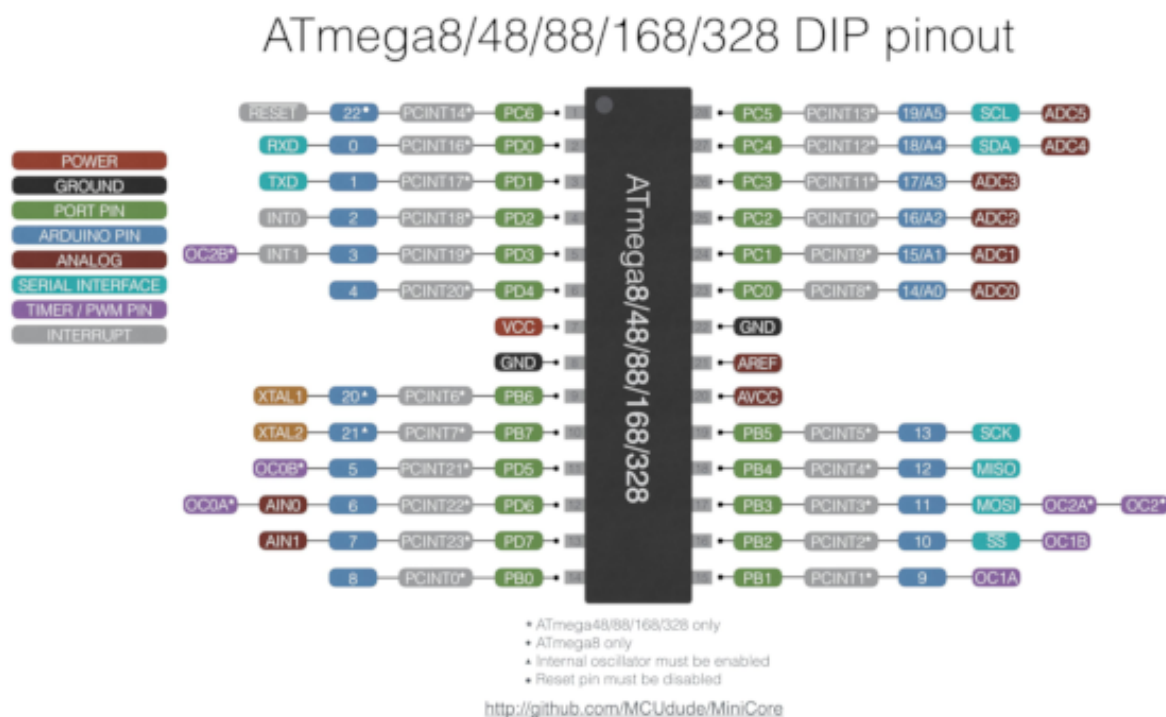


Figure 2.3 Pinout of ATmega48P/88P/168P/328P AVR microcontrollers family

### 2.2.1 Hardware

- Arduino UNO
- Piezo speaker/Buzzer
- 100 ohm current limiting resistor
- A breadboard

### 2.2.2 Circuit

The Buzzer with the current limiting resistor are connected to pin PB0 (digital pin 8 in Arduino UNO).

The circuit connection is as follows:

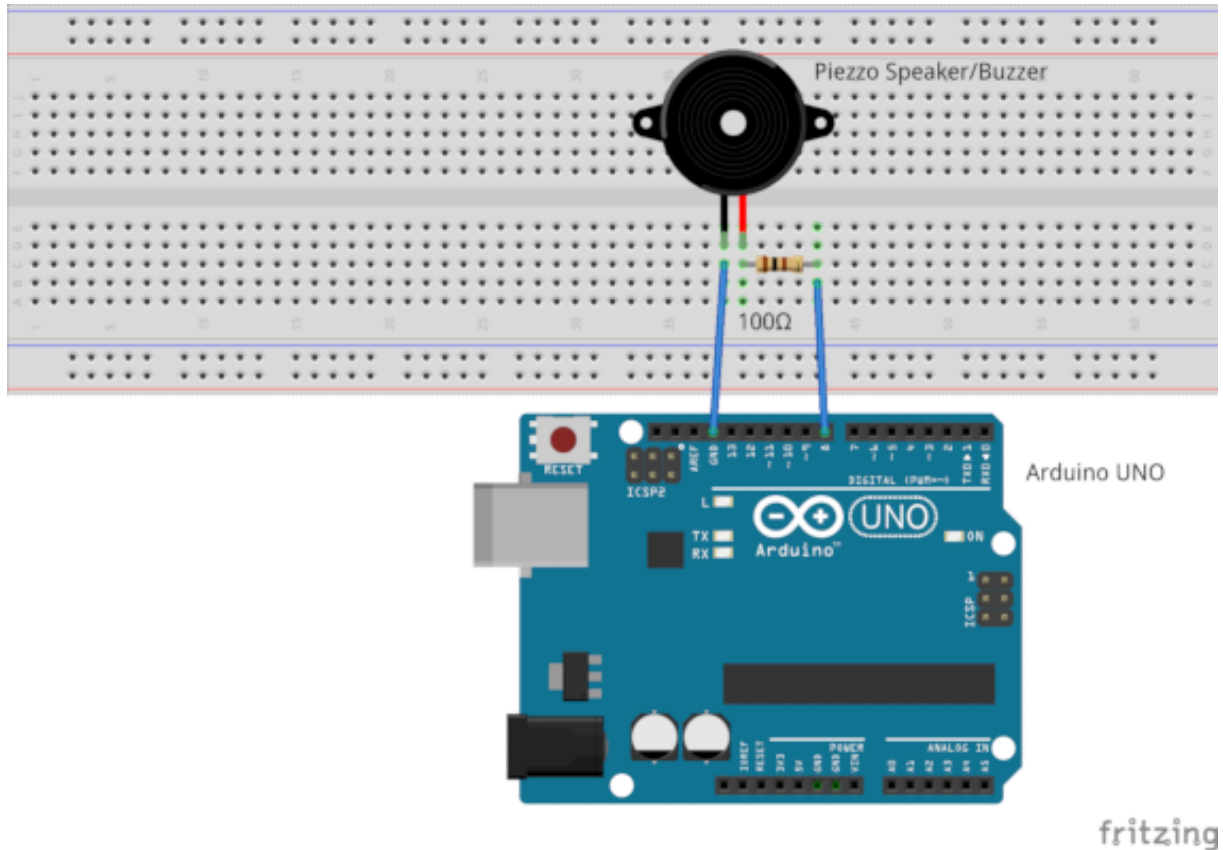


Figure 2.4 Circuit diagram

### 2.2.3 Code

The following code permits to turn a Buzzer into an Organ that plays notes entered by a user via a serial terminal (GTKTerm serial port terminal is used):

```
#include "Buzzer.h"
#include "USART0.h"
#include "buzzer_pitches_16bit.h"
#define BUZZER 0 /**< Buzzer pin number */
#define NOTE_DURATION 0xF000 /**< Note duration */
int main(void) {
    // Init
    // Instantiate a Buzzer object
    component::Buzzer Buzzer(io::Pin(BUZZER,io::PortB));
    // Instantiate a USART0 object
    io::USART0 &myUSART0 = io::USART0::getInstance();
    // List of possible keypresses
    const uint8_t l_keys[] = { 'a', 'w', 's', 'e', 'd', 'f', 't',
                              'g', 'y', 'h', 'j', 'i', 'k', 'o',
                              'l', 'p', ';', '\n'
    };
    // List of the keypresses corresponding pitches
    const uint16_t l_notes[] = { G4, Gx4, A4, Ax4, B4, C5, Cx5,
                                D5, Dx5, E5, Fx5, G5, Gx5,
                                A5, Ax5, B5, C6
    };
    // Character key from computer serial terminal
```

```

unsigned char l_key;
// Current note length
uint16_t l_currentNoteLength = NOTE_DURATION;
// Check if note key
uint8_t l_isNote;
myUSART0.sendString("----- Serial Organ -----\r\n");
// Mainloop
while (1) {
    // Send N to signal that AVR chip is ready for next key note input
    myUSART0.sendString("Enter key note\r\n");
    // Wait for key note input
    while (myUSART0.getNumberBytesReceived()==0)
    {
        myUSART0.receiveChar(l_key);
    }
    myUSART0.resetNumberBytesReceived();
    /***** Play key notes *****/
    l_isNote = 0;
    // loop through keys table
    for (uint8_t i = 0; i < sizeof(l_keys); i++) {
        // found match in lookup table
        if (l_key == l_keys[i]) {
            Buzzer.buzz(l_notes[i], l_currentNoteLength);
            // record that we've found a note
            l_isNote = 1;
            break;
        }
    }
    // Handle non-note keys: tempo changes and rests
    if (!l_isNote) {
        if (l_key == '-') {
            // code for short note
            l_currentNoteLength = l_currentNoteLength / 2;
        }
        else if (l_key == '+') {
            // code for long note
            l_currentNoteLength = l_currentNoteLength * 2;
        }
        else {
            // unrecognized, just rest
            Buzzer.pause(l_currentNoteLength);
        }
    }
}
return 0;
}

```

## 2.2.4 Author

- Farid Oubbati
- Date: 12-May-2018
- Copyright (c) 2018

## 2.2.5 License

This project is licensed under the MIT License - see the [LICENSE.txt](#) file for more details

## 2.3 AVR Music Box

This example allows the user to play a predefined song in a Buzzer and control its tempo via a PushButton It also demonstrates the use of the Buzzer and PushButton abstraction objects.

The pinout of the **ATmega48P/88P/168P/328P AVR microcontrollers** family is illustrated below:

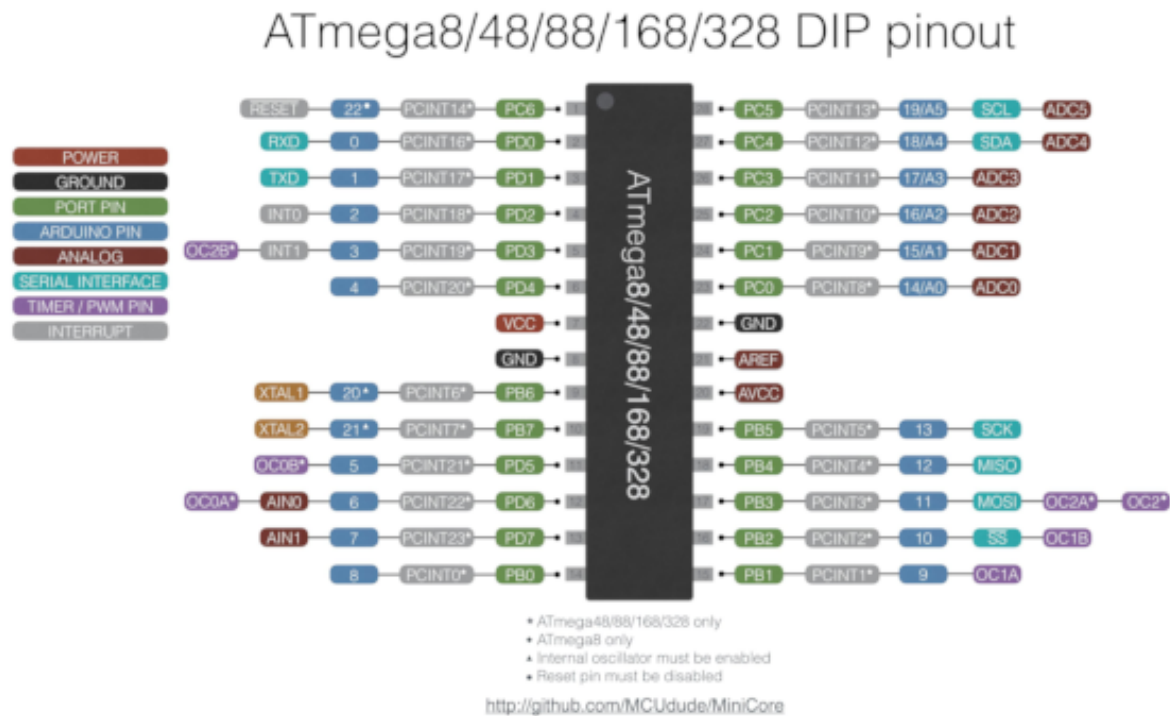


Figure 2.5 Pinout of ATmega48P/88P/168P/328P AVR microcontrollers family

### 2.3.1 Hardware

- Arduino UNO
- Piezo speaker/Buzzer
- 100 ohm current limiting resistor
- Switch/PushButton
- A breadboard

### 2.3.2 Circuit

The Buzzer with the current limiting resistor are connected to pin PB0 (digital pin 8 in Arduino UNO) and the PushButton is connected to PB1 (digital pin 9 in Arduino UNO).

The circuit connection is as follows:

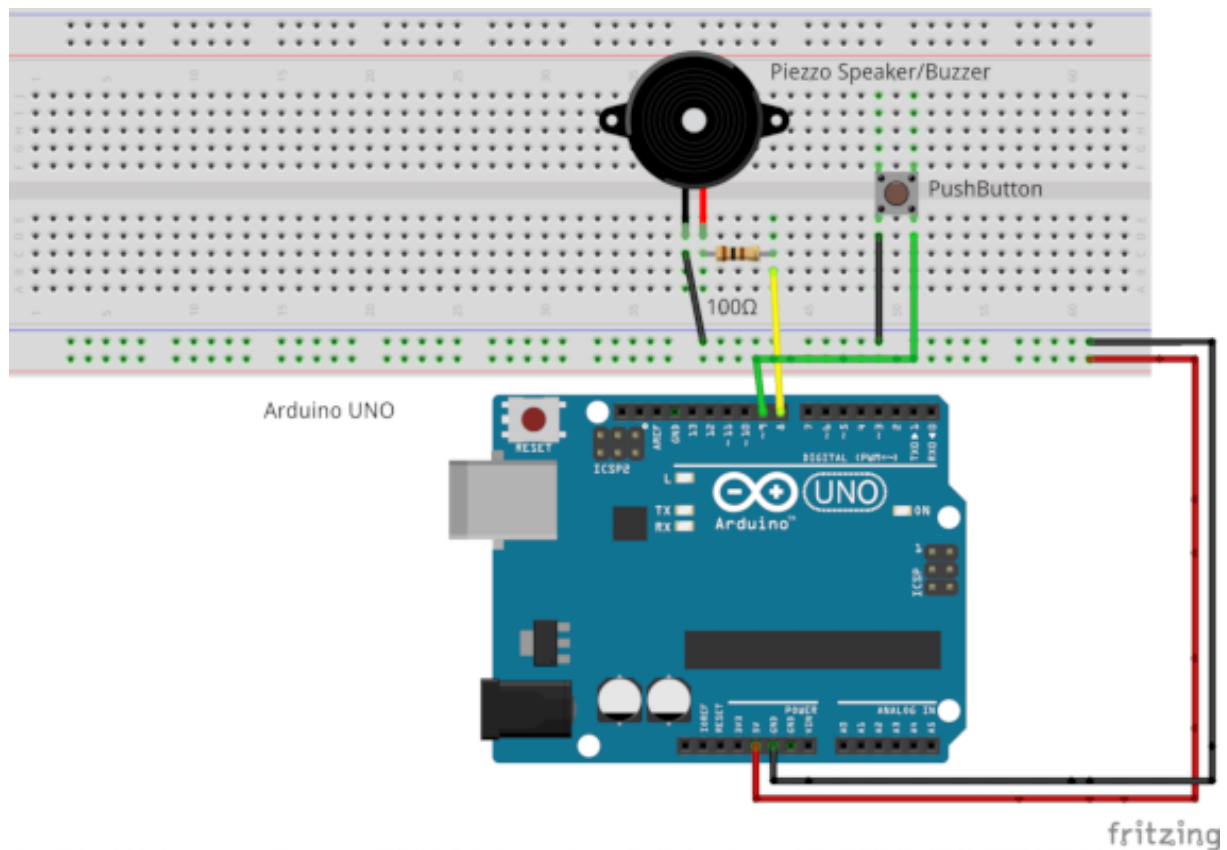


Figure 2.6 Circuit diagram

### 2.3.3 Code

The following code permits to turn a Buzzer into a Music Box that plays a predefined song which tempo is controlled by a PushButton:

```
#include "Buzzer.h"
#include "PushButton.h"
// Note pitch data
#include "buzzer_pitches_16bit.h"
#define BUZZER 0 /**< Buzzer pin number */
#define PUSHBUTTON 1 /**< PushButton pin number */
#define SONG_LENGTH (sizeof(l_song) / sizeof(uint16_t)) /**< Song length */
int main(void) {
    // Init
    // Instantiate a Buzzer object
    component::Buzzer Buzzer(io::Pin(BUZZER,io::PortB));
    // Instantiate a PushButton object
    component::PushButton PushButton(io::Pin(PUSHBUTTON,io::PortB));
    // Song notes
    const uint16_t l_song[] = {
        E6, E6, E6, C6, E6, G6, G5,
        C6, G5, E5, A5, B5, A5, A5,
        G5, E6, G6, A6, F6, G6, E6, C6, D6, B5,
        C6, G5, E5, A5, B5, A5, A5,
        G5, E6, G6, A6, F6, G6, E6, C6, D6, B5,
    };
    // Start song at start
    uint8_t l_note = SONG_LENGTH - 1;
    // Initialize PushButton state
    uint8_t l_statePushButton = 0;
    // Mainloop
    while (1) {
        // Check if PushButton is pressed
        if (PushButton.isPressed())
        {
            // Check if a new button press
            if (!l_statePushButton) {
```

```
        // play next note
        l_note++;
        // start from start
        if (l_note == SONG_LENGTH) {
            l_note = 0;
        }
        l_statePushButton = 1;
    }
    // Play note
    Buzzer.buzz(l_song[l_note], 1600);
} else {
    l_statePushButton=0;
}
}
return 0;
}
```

### 2.3.4 Author

- Farid Oubbati
- Date: 12-May-2018
- Copyright (c) 2018

### 2.3.5 License

This project is licensed under the MIT License - see the [LICENSE.txt](#) file for more details

## 2.4 Light Meter

This example presents an AVR based Light Meter that displays its Lux measurements in a Led bank It also demonstrates the use of the LED, LDR and ADC abstraction objects.

The pinout of the **ATmega48P/88P/168P/328P AVR microcontrollers** family is illustrated below:





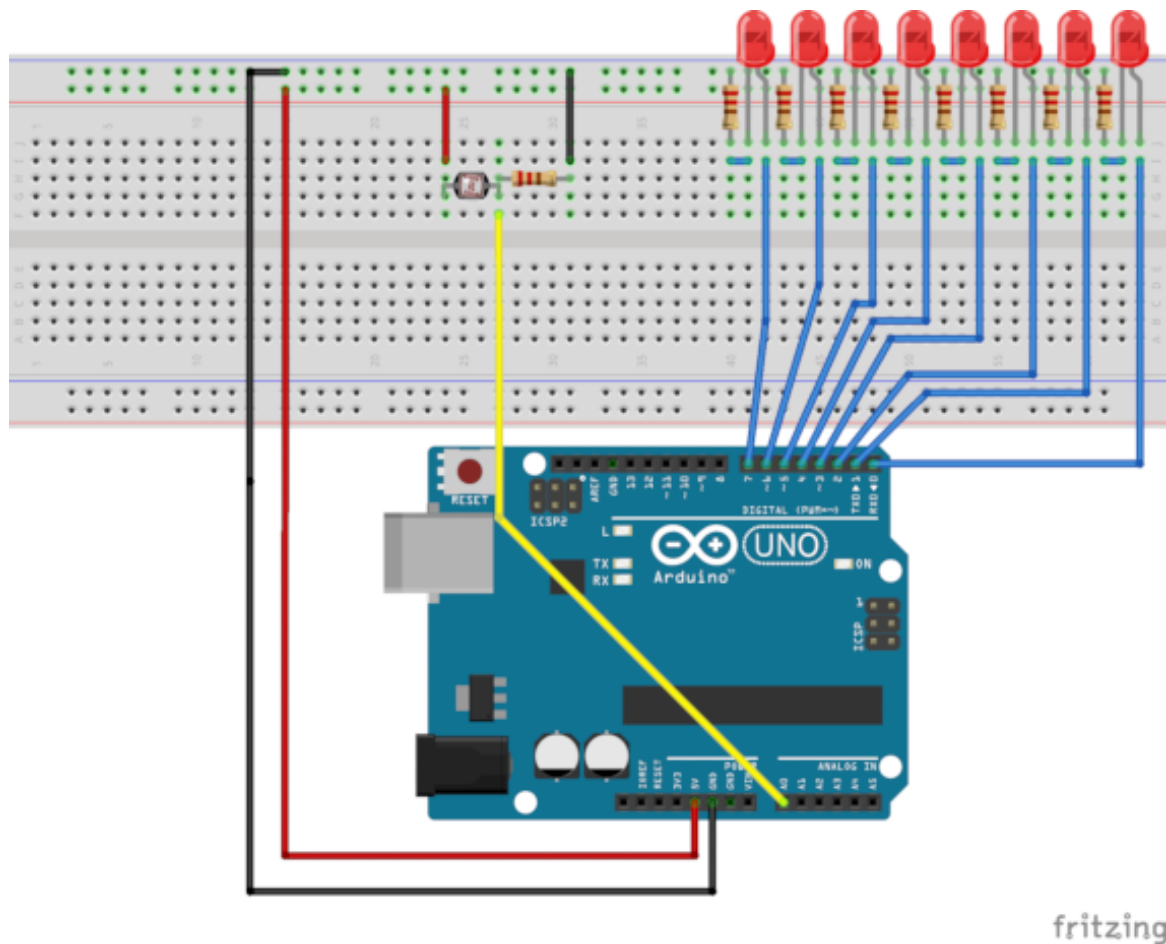


Figure 2.8 Circuit diagram

### 2.4.3 Code

The following code permits to run the AVR based Light Meter that displays its Lux measurements in a Led bank:

```
#include "Led.h"
#include "ADC.h"
#include "LDR.h"
#define LED0 0 /**< Led 0 */
#define LED1 1 /**< Led 1 */
#define LED2 2 /**< Led 2 */
#define LED3 3 /**< Led 3 */
#define LED4 4 /**< Led 4 */
#define LED5 5 /**< Led 5 */
#define LED6 6 /**< Led 6 */
#define LED7 7 /**< Led 7 */
#define LDRVD 0 /**< LDR */
int main(void) {
    // Init
    // Instantiate Led objects
    component::Led Led[8]={
        {io::Pin(LED0,io::PortD)},
        {io::Pin(LED1,io::PortD)},
        {io::Pin(LED2,io::PortD)},
        {io::Pin(LED3,io::PortD)},
        {io::Pin(LED4,io::PortD)},
        {io::Pin(LED5,io::PortD)},
        {io::Pin(LED6,io::PortD)},
        {io::Pin(LED7,io::PortD)}};
    // Instantiate ADC object
    core::ADConverter &myADC = core::ADConverter::getInstance();
    // Start conversion
    myADC.start();
    // Instantiate LDR object
    component::LDR LDR(io::Pin(LDRVD,io::PortC),myADC);
    // Mainloop
```

```
while (1) {
    for (uint8_t i=0;i<=(7-(LDR.getLux(5) & 7));i++)
    {
        Led[i].on();
    }
    for (uint8_t i=(7-(LDR.getLux(5) & 7))+1;i<8;i++)
    {
        Led[i].off();
    }
}
return 0;
}
```

#### 2.4.4 Author

- Farid Oubbati
- Date: 12-May-2018
- Copyright (c) 2018

#### 2.4.5 License

This project is licensed under the MIT License - see the [LICENSE.txt](#) file for more details

## **Chapter 3**

# **README**



## Chapter 4

# Namespace Index

### 4.1 Namespace List

Here is a list of all namespaces with brief descriptions:

<a href="#">component</a>	AVR chip external components . . . . .	<a href="#">27</a>
<a href="#">core</a>	AVR chip internal core components . . . . .	<a href="#">28</a>
<a href="#">io</a>	AVR chip internal i/o components . . . . .	<a href="#">37</a>
<a href="#">utils</a>	. . . . .	<a href="#">43</a>



## Chapter 5

# Hierarchical Index

### 5.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

core::ADConverter . . . . .	45
core::AnalogComparator . . . . .	55
component::Buzzer . . . . .	55
component::DCMotor . . . . .	59
core::ExternInterrupt . . . . .	62
component::LDR . . . . .	68
component::Led . . . . .	71
core::MCU . . . . .	75
io::Pin . . . . .	80
io::Port . . . . .	85
component::PushButton . . . . .	87
component::ServoMotor . . . . .	91
io::SPI . . . . .	97
component::StepperMotor . . . . .	105
core::TimerCounter . . . . .	115
core::TimerCounter0 . . . . .	118
core::TimerCounter1 . . . . .	128
core::TimerCounter2 . . . . .	139
io::USART0 . . . . .	149
core::WatchdogTimer . . . . .	166





## Chapter 6

# Class Index

### 6.1 Class List

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

<a href="#">core::ADConverter</a>	
Class for handling ADC	45
<a href="#">core::AnalogComparator</a>	55
<a href="#">component::Buzzer</a>	
Class for handling a <a href="#">Buzzer</a> component	55
<a href="#">component::DCMotor</a>	59
<a href="#">core::ExternInterrupt</a>	62
<a href="#">component::LDR</a>	
Class for handling an <a href="#">LDR</a> component	68
<a href="#">component::Led</a>	
Class for handling a <a href="#">Led</a> component	71
<a href="#">core::MCU</a>	75
<a href="#">io::Pin</a>	
Class for handling a <a href="#">Pin</a>	80
<a href="#">io::Port</a>	
Contains definitions of pointers to the AVR port's registers	85
<a href="#">component::PushButton</a>	
Class for handling a <a href="#">PushButton</a> component	87
<a href="#">component::ServoMotor</a>	91
<a href="#">io::SPI</a>	97
<a href="#">component::StepperMotor</a>	105
<a href="#">core::TimerCounter</a>	115
<a href="#">core::TimerCounter0</a>	118
<a href="#">core::TimerCounter1</a>	128
<a href="#">core::TimerCounter2</a>	139
<a href="#">io::USART0</a>	
Class for handling <a href="#">USART0</a>	149
<a href="#">core::WatchdogTimer</a>	166



## Chapter 7

# File Index

### 7.1 File List

Here is a list of all files with brief descriptions:

ADC.cpp	171
ADC.h	175
AnalogComparator.cpp	178
AnalogComparator.h	178
Buzzer.cpp	180
Buzzer.h	181
buzzer_pitches_16bit.h	184
buzzer_pitches_8bit.h	203
DCMotor.cpp	211
DCMotor.h	
Header file of the DCMotor class	211
ExternInterrupt.cpp	214
ExternInterrupt.h	
Header file of the ExternInterrupt class	216
ha_base.h	
Base header file for the basic hardware abstraction macros	220
ha_m328p.h	
Header file for the hardware abstraction macros of the Atmega328p	222
LCD.cpp	257
LCD.h	257
LDR.cpp	257
LDR.h	258
Led.cpp	259
Led.h	260
BlinkLed/main.cpp	262
LightMeter/main.cpp	264
MusicBox/main.cpp	268
Organ/main.cpp	272
MCU.cpp	276
MCU.h	
Header file of the MCU class	278
Pin.cpp	281
Pin.h	282
PushButton.cpp	284
PushButton.h	285

<a href="#">ServoMotor.cpp</a>	287
<a href="#">ServoMotor.h</a>	
Header file of the ServoMotor class	289
<a href="#">SPI.cpp</a>	293
<a href="#">SPI.h</a>	
Header file of the SPI class	296
<a href="#">StepperMotor.cpp</a>	299
<a href="#">StepperMotor.h</a>	304
<a href="#">TimerCounter.h</a>	
Header file of the TimerCounter class	306
<a href="#">TimerCounter0.cpp</a>	318
<a href="#">TimerCounter0.h</a>	321
<a href="#">TimerCounter1.cpp</a>	323
<a href="#">TimerCounter1.h</a>	328
<a href="#">TimerCounter2.cpp</a>	330
<a href="#">TimerCounter2.h</a>	334
<a href="#">USART0.cpp</a>	336
<a href="#">USART0.h</a>	341
<a href="#">utils_m328p.cpp</a>	344
<a href="#">utils_m328p.h</a>	344
<a href="#">WatchdogTimer.cpp</a>	345
<a href="#">WatchdogTimer.h</a>	
Header file of the WatchdogTimer class	346

## Chapter 8

# Namespace Documentation

### 8.1 component Namespace Reference

AVR chip external components.

#### Classes

- class [Buzzer](#)  
*Class for handling a [Buzzer](#) component.*
- class [DCMotor](#)
- class [LDR](#)  
*Class for handling an [LDR](#) component.*
- class [Led](#)  
*Class for handling a [Led](#) component.*
- class [PushButton](#)  
*Class for handling a [PushButton](#) component.*
- class [ServoMotor](#)
- class [StepperMotor](#)

#### Enumerations

- enum [mode](#) : `uint8_t` { [mode::fullStep](#) =0, [mode::halfStep](#) }

#### 8.1.1 Detailed Description

AVR chip external components.

The namespace englobes all external components that, when hooked up, allows the AVR chip to interact with the external World

The namespace englobes all external components that, when hooked up, allows the AVR MCU to interact with the external World

#### 8.1.2 Enumeration Type Documentation

##### 8.1.2.1 mode

```
enum component::mode : uint8_t [strong]
```

### Enumerator

fullStep	
halfStep	

Definition at line 70 of file [StepperMotor.h](#).

```
00070         : uint8_t {
00071     fullStep=0,
00072     halfStep,
00073 };
```

## 8.2 core Namespace Reference

AVR chip internal core components.

### Classes

- class [ADConverter](#)  
*Class for handling ADC.*
- class [AnalogComparator](#)
- class [ExternInterrupt](#)
- class [MCU](#)
- class [TimerCounter](#)
- class [TimerCounter0](#)
- class [TimerCounter1](#)
- class [TimerCounter2](#)
- class [WatchdogTimer](#)

### Enumerations

- enum [resolution](#) : uint8\_t {  
resolution::res\_8bit =0, resolution::res\_9bit, resolution::res\_10bit, resolution::res\_11bit,  
resolution::res\_12bit, resolution::res\_13bit, resolution::res\_14bit, resolution::res\_15bit,  
resolution::res\_16bit }  
*ADC resolution.*
- enum [referenceVoltage](#) : uint8\_t { [referenceVoltage::AREF](#) =0, [referenceVoltage::AVCC](#), [referenceVoltage::internal](#) }  
*ADC referencev voltage.*
- enum [clockPrescaler](#) : uint8\_t {  
clockPrescaler::PS\_2 = 1, clockPrescaler::PS\_4, clockPrescaler::PS\_8, clockPrescaler::PS\_16,  
clockPrescaler::PS\_32, clockPrescaler::PS\_64, clockPrescaler::PS\_128 }  
*ADC clock prescaler.*
- enum [autoTriggerSource](#) : uint8\_t {  
autoTriggerSource::freeRunning, autoTriggerSource::analogComparator, autoTriggerSource::extInterrupt,  
autoTriggerSource::timer0CompareA,  
autoTriggerSource::timer0Overflow, autoTriggerSource::timer1 CompareB, autoTriggerSource::timer1 Overflow,  
autoTriggerSource::timer1 Capture }  
*ADC autoTrigger source.*
- enum [senseControl](#) : uint8\_t { [senseControl::lowLevel](#) =0, [senseControl::logicalChange](#), [senseControl::fallingEdge](#),  
[senseControl::risingEdge](#) }

- enum `pinChangePort` : `uint8_t` { `pinChangePort::PCINTB` =0, `pinChangePort::PCINTC`, `pinChangePort::PCINTD` }
- enum `BODMode` : `uint8_t` { `BODMode::enabled` =0, `BODMode::disabled` }
- enum `sleepMode` : `uint8_t` { `sleepMode::Idle` =0, `sleepMode::ADC_NoiseReduction`, `sleepMode::powerDown`, `sleepMode::powerSave`, `sleepMode::standby` =6, `sleepMode::extendedStandby` }
- enum `channel` : `uint8_t` { `channel::A` =0, `channel::B` }
- enum `compareOutputMode` : `uint8_t` { `compareOutputMode::normal` =0, `compareOutputMode::toggle`, `compareOutputMode::clear`, `compareOutputMode::set` }
- enum `operationMode` : `uint8_t` { `operationMode::normal` =0, `operationMode::PWM_PC`, `operationMode::PWM_PC_8bit`, `operationMode::PWM_PC_9bit`, `operationMode::PWM_PC_10bit`, `operationMode::PWM_PFC_ICR`, `operationMode::PWM_PFC_OCR`, `operationMode::PWM_PC_ICR`, `operationMode::PWM_PC_OCR`, `operationMode::fast_PWM`, `operationMode::fast_PWM_8bit`, `operationMode::fast_PWM_9bit`, `operationMode::fast_PWM_10bit`, `operationMode::fast_PWM_ICR`, `operationMode::fast_PWM_OCR`, `operationMode::CTC_OCR`, `operationMode::CTC_ICR`, `operationMode::interrupt` =1, `operationMode::reset`, `operationMode::interrupt_reset` }
- enum `clockSource` : `uint16_t` { `clockSource::noClock` =0, `clockSource::PS_1`, `clockSource::PS_8`, `clockSource::PS_32`, `clockSource::PS_64`, `clockSource::PS_128`, `clockSource::PS_256`, `clockSource::PS_1024`, `clockSource::extern_Clock_T0_Falling_Edge`, `clockSource::extern_Clock_T0_Rising_Edge` }
- enum `timeOut` : `uint8_t` { `timeOut::to_16ms` =0, `timeOut::to_32ms`, `timeOut::to_64ms`, `timeOut::to_125ms`, `timeOut::to_250ms`, `timeOut::to_500ms`, `timeOut::to_1s`, `timeOut::to_2s`, `timeOut::to_4s`, `timeOut::to_8s` }
- enum `operationMode` : `uint8_t` { `operationMode::normal` =0, `operationMode::PWM_PC`, `operationMode::PWM_PC_8bit`, `operationMode::PWM_PC_9bit`, `operationMode::PWM_PC_10bit`, `operationMode::PWM_PFC_ICR`, `operationMode::PWM_PFC_OCR`, `operationMode::PWM_PC_ICR`, `operationMode::PWM_PC_OCR`, `operationMode::fast_PWM`, `operationMode::fast_PWM_8bit`, `operationMode::fast_PWM_9bit`, `operationMode::fast_PWM_10bit`, `operationMode::fast_PWM_ICR`, `operationMode::fast_PWM_OCR`, `operationMode::CTC_OCR`, `operationMode::CTC_ICR`, `operationMode::interrupt` =1, `operationMode::reset`, `operationMode::interrupt_reset` }

### 8.2.1 Detailed Description

AVR chip internal core components.

The namespace englobes all internal core components that, when configured, allows the AVR [MCU](#) to internally process data

### 8.2.2 Enumeration Type Documentation

#### 8.2.2.1 autoTriggerSource

```
enum core::autoTriggerSource : uint8_t [strong]
```

ADC autoTrigger source.

**Author**

Farid Oubbati ( [farid.oubbati@outlook.com](mailto:farid.oubbati@outlook.com))

**Date**

March 2018

**Enumerator**

freeRunning	free running trigger mode
analogComparator	analog comparator atrigger mode
extInterrupt	external interrupt trigger mode
timer0CompareA	timer0 compare A trigger mode
timer0Overflow	timer0 overflow trigger mode
timer1CompareB	timer0 compare B trigger mode
timer1Overflow	timer1 overflow trigger mode
timer1Capture	timer1 capture trigger mode

Definition at line 76 of file [ADC.h](#).

```

00076                                     : uint8_t {
00077     freeRunning,
00078     analogComparator,
00079     extInterrupt,
00080     timer0CompareA,
00081     timer0Overflow,
00082     timer1CompareB,
00083     timer1Overflow,
00084     timer1Capture
00085 };

```

**8.2.2.2 BODMode**

```
enum core::BODMode : uint8_t [strong]
```

**Enumerator**

enabled	
disabled	

Definition at line 77 of file [MCU.h](#).

```

00077                                     : uint8_t {
00078     enabled=0,
00079     disabled,
00080 };

```

**8.2.2.3 channel**

```
enum core::channel : uint8_t [strong]
```



### Enumerator

A	
B	

Definition at line 470 of file [TimerCounter.h](#).

```
00470             : uint8_t {  
00471     A=0,  
00472     B,  
00473 };
```

#### 8.2.2.4 clockPrescaler

```
enum core::clockPrescaler : uint8_t [strong]
```

ADC clock prescaler.

### Author

Farid Oubbati ( [farid.oubbati@outlook.com](mailto:farid.oubbati@outlook.com))

### Date

March 2018

### Enumerator

PS_2	prescaler /2
PS_4	prescaler /4
PS_8	prescaler /8
PS_16	prescaler /16
PS_32	prescaler /32
PS_64	prescaler /64
PS_128	prescaler /128

Definition at line 59 of file [ADC.h](#).

```
00059             : uint8_t {  
00060     PS_2 = 1,  
00061     PS_4,  
00062     PS_8,  
00063     PS_16,  
00064     PS_32,  
00065     PS_64,  
00066     PS_128  
00067 };
```

#### 8.2.2.5 clockSource

```
enum core::clockSource : uint16_t [strong]
```

## Enumerator

noClock	
PS_1	
PS_8	
PS_32	
PS_64	
PS_128	
PS_256	
PS_1024	
extern_Clock_T0_Falling_Edge	
extern_Clock_T0_Rising_Edge	

Definition at line 504 of file [TimerCounter.h](#).

```

00504                                     : uint16_t {
00505     noClock=0,
00506     PS_1,
00507     PS_8,
00508     PS_32,
00509     PS_64,
00510     PS_128,
00511     PS_256,
00512     PS_1024,
00513     extern_Clock_T0_Falling_Edge,
00514     extern_Clock_T0_Rising_Edge,
00515 };

```

## 8.2.2.6 compareOutputMode

```
enum core::compareOutputMode : uint8_t [strong]
```

## Enumerator

normal	
toggle	
clear	
set	

Definition at line 476 of file [TimerCounter.h](#).

```

00476                                     : uint8_t {
00477     normal=0,
00478     toggle,
00479     clear,
00480     set,
00481 };

```

## 8.2.2.7 operationMode [1/2]

```
enum core::operationMode : uint8_t [strong]
```

## Enumerator

normal	
--------	--

## Enumerator

PWM_PC	
PWM_PC_8bit	
PWM_PC_9bit	
PWM_PC_10bit	
PWM_PFC_ICR	
PWM_PFC_OCR	
PWM_PC_ICR	
PWM_PC_OCR	
fast_PWM	
fast_PWM_8bit	
fast_PWM_9bit	
fast_PWM_10bit	
fast_PWM_ICR	
fast_PWM_OCR	
CTC_OCR	
CTC_ICR	
interrupt	
reset	
interrupt_reset	

Definition at line 78 of file [WatchdogTimer.h](#).

```

00078                                     : uint8_t {
00079     interrupt=1,
00080     reset,
00081     interrupt_reset,
00082 };

```

## 8.2.2.8 operationMode [2/2]

```
enum core::operationMode : uint8_t [strong]
```

## Enumerator

normal	
PWM_PC	
PWM_PC_8bit	
PWM_PC_9bit	
PWM_PC_10bit	
PWM_PFC_ICR	
PWM_PFC_OCR	
PWM_PC_ICR	
PWM_PC_OCR	
fast_PWM	
fast_PWM_8bit	
fast_PWM_9bit	
fast_PWM_10bit	
fast_PWM_ICR	
fast_PWM_OCR	

## Enumerator

CTC_OCR	
CTC_ICR	
interrupt	
reset	
interrupt_reset	

Definition at line 485 of file [TimerCounter.h](#).

```

00485                                     : uint8_t {
00486     normal=0,
00487     PWM_PC,
00488     PWM_PC_8bit,
00489     PWM_PC_9bit,
00490     PWM_PC_10bit,
00491     PWM_PFC_ICR,
00492     PWM_PFC_OCR,
00493     PWM_PC_ICR,
00494     PWM_PC_OCR,
00495     fast_PWM,
00496     fast_PWM_8bit,
00497     fast_PWM_9bit,
00498     fast_PWM_10bit,
00499     fast_PWM_ICR,
00500     fast_PWM_OCR,
00501     CTC_OCR,
00502     CTC_ICR,
00503 };

```

## 8.2.2.9 pinChangePort

```
enum core::pinChangePort : uint8_t [strong]
```

## Enumerator

PCINTB	
PCINTC	
PCINTD	

Definition at line 156 of file [ExternInterrupt.h](#).

```

00156                                     : uint8_t {
00157     PCINTB=0,
00158     PCINTC,
00159     PCINTD,
00160 };

```

## 8.2.2.10 referenceVoltage

```
enum core::referenceVoltage : uint8_t [strong]
```

ADC referencev voltage.

## Author

Farid Oubbati ( [farid.oubbati@outlook.com](mailto:farid.oubbati@outlook.com))

## Date

March 2018

## Enumerator

AREF	external AREF pin voltage reference, internal 1.1V voltage reference turned off
AVCC	AVCC voltage reference with external capacitor at AREF pin
internal	internal 1.1V voltage reference with external capacitor at AREF pin

Definition at line 44 of file [ADC.h](#).

```
00044                                     : uint8_t {
00045     AREF=0,
00046     AVCC,
00047     internal
00048 };
```

## 8.2.2.11 resolution

```
enum core::resolution : uint8_t [strong]
```

ADC resolution.

## Author

Farid Oubbati ( [farid.oubbati@outlook.com](mailto:farid.oubbati@outlook.com) )

## Date

March 2018

## Enumerator

res_8bit	8 bits resolution
res_9bit	9 bits resolution
res_10bit	10 bits resolution
res_11bit	11 bits resolution
res_12bit	12 bits resolution
res_13bit	13 bits resolution
res_14bit	14 bits resolution
res_15bit	15 bits resolution
res_16bit	16 bits resolution

Definition at line 24 of file [ADC.h](#).

```
00024                                     : uint8_t {
00025     res_8bit=0,
00026     res_9bit,
00027     res_10bit,
00028     res_11bit,
00029     res_12bit,
00030     res_13bit,
00031     res_14bit,
00032     res_15bit,
00033     res_16bit
00034 };
```

### 8.2.2.12 senseControl

```
enum core::senseControl : uint8_t [strong]
```

#### Enumerator

lowLevel	
logicalChange	
fallingEdge	
risingEdge	

Definition at line 149 of file [ExternInterrupt.h](#).

```
00149         : uint8_t {
00150     lowLevel=0,
00151     logicalChange,
00152     fallingEdge,
00153     risingEdge
00154 };
```

### 8.2.2.13 sleepMode

```
enum core::sleepMode : uint8_t [strong]
```

#### Enumerator

Idle	
ADC_NoiseReduction	
powerDown	
powerSave	
standby	
extendedStandby	

Definition at line 82 of file [MCU.h](#).

```
00082         : uint8_t {
00083     Idle=0,
00084     ADC_NoiseReduction,
00085     powerDown,
00086     powerSave,
00087     standby=6,
00088     extendedStandby,
00089 };
```

### 8.2.2.14 timeOut

```
enum core::timeOut : uint8_t [strong]
```

#### Enumerator

to_16ms	
to_32ms	

## Enumerator

to_64ms	
to_125ms	
to_250ms	
to_500ms	
to_1s	
to_2s	
to_4s	
to_8s	

Definition at line 65 of file [WatchdogTimer.h](#).

```

00065         : uint8_t {
00066     to_16ms=0,
00067     to_32ms,
00068     to_64ms,
00069     to_125ms,
00070     to_250ms,
00071     to_500ms,
00072     to_1s,
00073     to_2s,
00074     to_4s,
00075     to_8s,
00076 };

```

## 8.3 io Namespace Reference

AVR chip internal i/o components.

### Classes

- class [Pin](#)  
*Class for handling a [Pin](#).*
- struct [Port](#)  
*Contains definitions of pointers to the AVR port's registers.*
- class [SPI](#)
- class [USART0](#)  
*Class for handling [USART0](#).*

### Enumerations

- enum [operationMode](#) : uint8\_t { [operationMode::master](#) =0, [operationMode::slave](#), [operationMode::submaster](#), [operationMode::disable](#) }
  - enum [clockPrescaler](#) : uint8\_t { [clockPrescaler::PS\\_4](#) = 0, [clockPrescaler::PS\\_16](#), [clockPrescaler::PS\\_64](#), [clockPrescaler::PS\\_128](#), [clockPrescaler::PS\\_2](#), [clockPrescaler::PS\\_8](#), [clockPrescaler::PS\\_32](#) }
  - enum [dataMode](#) : uint8\_t { [dataMode::mode\\_0](#) = 0, [dataMode::mode\\_1](#), [dataMode::mode\\_2](#), [dataMode::mode\\_3](#) }
  - enum [dataOrder](#) : uint8\_t { [dataOrder::first\\_MSB](#) = 0, [dataOrder::first\\_LSB](#) }
  - enum [transmissionMode](#) : uint8\_t { [transmissionMode::async](#) =0, [transmissionMode::sync](#), [transmissionMode::masterSPI](#) }
- USART0 transmission mode.*

- enum `communicationMode` : `uint8_t` { `communicationMode::duplex` =0, `communicationMode::transmit`, `communicationMode::receive` }  
*USART0 communication mode.*
- enum `parityMode` : `uint8_t` { `parityMode::noParity` =0, `parityMode::evenParity`, `parityMode::oddParity` }  
*USART0 parity mode.*
- enum `frameSize` : `uint8_t` { `frameSize::eightBits` =0, `frameSize::fiveBits`, `frameSize::sixBits`, `frameSize::sevenBits`, `frameSize::neineBits` }  
*USART0 frame size.*
- enum `stopBit` : `uint8_t` { `stopBit::oneStopBit` =0, `stopBit::twoStopBits` }  
*USART0 stop bit.*

## Variables

- static `io::Port PortB` = { &DDRB, &PORTB, &PINB }
- static `io::Port PortC` = { &DDRC, &PORTC, &PINC }
- static `io::Port PortD` = { &DDRD, &PORTD, &PIND }

### 8.3.1 Detailed Description

AVR chip internal i/o components.

The namespace englobes all internal input/output components that, when configured, allows the AVR MCU to communicate with the external World

### 8.3.2 Enumeration Type Documentation

#### 8.3.2.1 clockPrescaler

```
enum io::clockPrescaler : uint8_t [strong]
```

##### Enumerator

PS_4	
PS_16	
PS_64	
PS_128	
PS_2	
PS_8	
PS_32	

Definition at line 63 of file `SPI.h`.

```
00063                                     : uint8_t {
00064     PS_4 = 0,
00065     PS_16,
00066     PS_64,
00067     PS_128,
```



```
00068     PS_2,  
00069     PS_8,  
00070     PS_32  
00071 };
```

### 8.3.2.2 communicationMode

```
enum io::communicationMode : uint8_t [strong]
```

USART0 communication mode.

#### Author

Farid Oubbati ( [farid.oubbati@outlook.com](mailto:farid.oubbati@outlook.com))

#### Date

March 2018

#### Enumerator

duplex	full duplex mode
transmit	transmit mode
receive	receive mode

Definition at line 39 of file [USART0.h](#).

```
00039                                     : uint8_t {  
00040     duplex=0,  
00041     transmit,  
00042     receive,  
00044 };
```

### 8.3.2.3 dataMode

```
enum io::dataMode : uint8_t [strong]
```

#### Enumerator

mode↔ _0	
mode↔ _1	
mode↔ _2	
mode↔ _3	

Definition at line 73 of file [SPI.h](#).

```

00073         : uint8_t {
00074     mode_0 = 0,
00075     mode_1,
00076     mode_2,
00077     mode_3,
00078 };

```

#### 8.3.2.4 dataOrder

```
enum io::dataOrder : uint8_t [strong]
```

##### Enumerator

first_MSB	
first_LSB	

Definition at line 80 of file [SPI.h](#).

```

00080         : uint8_t {
00081     first_MSB = 0,
00082     first_LSB
00083 };

```

#### 8.3.2.5 frameSize

```
enum io::frameSize : uint8_t [strong]
```

[USART0](#) frame size.

##### Author

Farid Oubbati ( [farid.oubbati@outlook.com](mailto:farid.oubbati@outlook.com))

##### Date

March 2018

##### Enumerator

eightBits	8 bits frame size
fiveBits	5 bits frame size
sixBits	6 bits frame size
sevenBits	7 bits frame size
neineBits	9 bits frame size

Definition at line 68 of file [USART0.h](#).

```

00068         : uint8_t {
00069     eightBits=0,
00070     fiveBits,
00071     sixBits,
00072     sevenBits,

```

```
00073     neineBits
00074 };
```

### 8.3.2.6 operationMode

```
enum io::operationMode : uint8_t [strong]
```

#### Enumerator

master	
slave	
submaster	
disable	

Definition at line 56 of file [SPI.h](#).

```
00056                                     : uint8_t {
00057     master=0,
00058     slave,
00059     submaster,
00060     disable,
00061 };
```

### 8.3.2.7 parityMode

```
enum io::parityMode : uint8_t [strong]
```

[USART0](#) parity mode.

#### Author

Farid Oubbati ( [farid.oubbati@outlook.com](mailto:farid.oubbati@outlook.com))

#### Date

March 2018

#### Enumerator

noParity	no parity check mode
evenParity	even parity check mode
oddParity	odd parity check mode

Definition at line 54 of file [USART0.h](#).

```
00054                                     : uint8_t {
00055     noParity=0,
00056     evenParity,
00057     oddParity
00058 };
```

### 8.3.2.8 stopBit

```
enum io::stopBit : uint8_t [strong]
```

USART0 stop bit.

#### Author

Farid Oubbati ( [farid.oubbati@outlook.com](mailto:farid.oubbati@outlook.com) )

#### Date

March 2018

#### Enumerator

oneStopBit	1 stop bit
twoStopBits	2 stop bits

Definition at line 84 of file [USART0.h](#).

```
00084         : uint8_t {  
00085     oneStopBit=0,  
00086     twoStopBits  
00087 };
```

### 8.3.2.9 transmissionMode

```
enum io::transmissionMode : uint8_t [strong]
```

USART0 transmission mode.

#### Author

Farid Oubbati ( [farid.oubbati@outlook.com](mailto:farid.oubbati@outlook.com) )

#### Date

March 2018

#### Enumerator

async	asynchronous mode
sync	synchronous mode
masterSPI	masterSPI mode

Definition at line 25 of file [USART0.h](#).

```
00025         : uint8_t {  
00026     async=0,  
00027     sync,
```

```
00028     masterSPI
00029 };
```

### 8.3.3 Variable Documentation

#### 8.3.3.1 PortB

```
io::Port io::PortB = { &DDRB, &PORTB, &PINB } [static]
```

global static [Port](#) B object

Definition at line 100 of file [Pin.h](#).

#### 8.3.3.2 PortC

```
io::Port io::PortC = { &DDRC, &PORTC, &PINC } [static]
```

global static [Port](#) C object

Definition at line 101 of file [Pin.h](#).

#### 8.3.3.3 PortD

```
io::Port io::PortD = { &DDRD, &PORTD, &PIND } [static]
```

global static [Port](#) D object

Definition at line 102 of file [Pin.h](#).

Referenced by [main\(\)](#).

## 8.4 utils Namespace Reference

### Functions

- long [map](#) (long x, long in\_min, long in\_max, long out\_min, long out\_max)

### 8.4.1 Function Documentation

#### 8.4.1.1 map()

```
long utils::map (
    long x,
    long in_min,
    long in_max,
    long out_min,
    long out_max )
```

Definition at line 3 of file [utils\\_m328p.cpp](#).

```
00004 {
00005     return (x - in_min) * (out_max - out_min) / (in_max - in_min) + out_min;
00006 }
```



## Chapter 9

# Class Documentation

### 9.1 core::ADConverter Class Reference

Class for handling ADC.

```
#include <ADC.h>
```

#### Public Member Functions

- void [start](#) ()
- void [stop](#) ()
- void [selectReferenceVoltage](#) (const [referenceVoltage](#) &ar\_refVoltage)
- void [selectAnalogInput](#) ([io::Pin](#) a\_pin)
- void [selectClockPrescaler](#) (const [clockPrescaler](#) &ar\_clockPrescaler)
- void [enableConversionCompleteInterrupt](#) (const uint8\_t a\_enable)
- void [enableAutoTrigger](#) (const uint8\_t a\_enable)
- void [selectAutoTriggerSource](#) (const [autoTriggerSource](#) &ar\_autoTriggerSource)
- uint16\_t [getConversionResult](#) (const [resolution](#) &ar\_resolution=[resolution::res\\_10bit](#))

#### Static Public Member Functions

- static [ADConverter](#) & [getInstance](#) (const [referenceVoltage](#) &ar\_refVoltage=[referenceVoltage::AVCC](#), const [clockPrescaler](#) &ar\_clockPrescaler=[clockPrescaler::PS\\_128](#), const [autoTriggerSource](#) &ar\_autoTriggerSource=[autoTriggerSource::freeRunning](#), const [io::Pin](#) &ar\_pin=[io::Pin](#)(0, [io::PortC](#)))
- static void [conversionCompleteServiceRoutine](#) () \_\_asm\_\_(STR(ADC\_CONVERSION\_COMPLETE\_INTERRUPT))  
\_\_attribute\_\_((\_\_signal\_\_

#### Public Attributes

- static void [\\_\\_used\\_\\_](#)
- static void [\\_\\_externally\\_visible\\_\\_](#)

## Private Member Functions

- [ADConverter](#) (const [referenceVoltage](#) &ar\_refVoltage, const [clockPrescaler](#) &ar\_clockPrescaler, const [autoTriggerSource](#) &ar\_autoTriggerSource, const [io::Pin](#) &ar\_pin)
- [~ADConverter](#) ()
- [ADConverter](#) (const [ADConverter](#) &)
- const [ADConverter](#) & [operator=](#) (const [ADConverter](#) &)

## Static Private Attributes

- static volatile uint16\_t [m\\_conversionResult](#) = 0
- static uint8\_t [m\\_resolution](#) = 10

### 9.1.1 Detailed Description

Class for handling ADC.

The class implements ADC operations

#### Author

Farid Oubbati ( [farid.oubbati@outlook.com](mailto:farid.oubbati@outlook.com) )

#### Date

March 2018

Definition at line [94](#) of file [ADC.h](#).

### 9.1.2 Constructor & Destructor Documentation

#### 9.1.2.1 ADConverter() [1/2]

```
core::ADConverter::ADConverter (
    const referenceVoltage & ar_refVoltage,
    const clockPrescaler & ar_clockPrescaler,
    const autoTriggerSource & ar_autoTriggerSource,
    const io::Pin & ar_pin ) [private]
```

Constructor.

Initializes the ADC object

#### Parameters

<i>ar_refVoltage</i>	Defines reference voltage
<i>ar_clockPrescaler</i>	Defines clock prescaler
<i>ar_autoTriggerSource</i>	Defines auto trigger source
<i>ar_pin</i>	Defines a Pin object



Definition at line 22 of file [ADC.cpp](#).

```
00026 {
00027     core::MCU::enableADC(1);
00028     selectAnalogInput(ar_pin);
00029     selectReferenceVoltage(ar_refVoltage);
00030     selectClockPrescaler(ar_clockPrescaler);
00031     enableAutoTrigger(1);
00032     selectAutoTriggerSource(ar_autoTriggerSource);
00033     sei();
00034     enableConversionCompleteInterrupt(1);
00035 }
00036
00037 }
```

References [core::MCU::enableADC\(\)](#).

### 9.1.2.2 ~ADConverter()

```
core::ADConverter::~~ADConverter ( ) [private]
```

Destructor

Definition at line 40 of file [ADC.cpp](#).

```
00041 {
00042 }
00043 }
```

### 9.1.2.3 ADConverter() [2/2]

```
core::ADConverter::ADConverter (
    const ADConverter & ) [private]
```

Overried Copy constructor

## 9.1.3 Member Function Documentation

### 9.1.3.1 conversionCompleteServiceRoutine()

```
void core::ADConverter::conversionCompleteServiceRoutine ( ) [static]
```

Conversion complete ISR

Definition at line 108 of file [ADC.cpp](#).

```
00109 {
00110     static uint32_t l_resultData = 0;
00111     static uint16_t l_resultDataIndex = 0;
00112
00113     switch (m_resolution)
00114     {
00115     case 8:
00116     {
```

```
00120         m_conversionResult = ADC » 8;
00121         break;
00122     }
00123     case 9:
00124     {
00125         m_conversionResult = ADC » 7;
00126         break;
00127     }
00128     case 10:
00129     {
00130         m_conversionResult = ADC;
00131         break;
00132     }
00133     case 11:
00134     {
00135
00136         if (l_resultDataIndex < 4)
00137         {
00138             l_resultData += ADC;
00139             l_resultDataIndex++;
00140         }
00141         else
00142         {
00143             m_conversionResult = l_resultData » 1;
00144             l_resultData = 0;
00145             l_resultDataIndex = 0;
00146         }
00147
00148         break;
00149     }
00150
00151     case 12:
00152     {
00153         if (l_resultDataIndex < 16)
00154         {
00155             l_resultData += ADC;
00156             l_resultDataIndex++;
00157         }
00158         else
00159         {
00160             m_conversionResult = l_resultData » 2;
00161             l_resultData = 0;
00162             l_resultDataIndex = 0;
00163         }
00164         break;
00165     }
00166
00167     case 13:
00168     {
00169         if (l_resultDataIndex < 64)
00170         {
00171             l_resultData += ADC;
00172             l_resultDataIndex++;
00173         }
00174         else
00175         {
00176             m_conversionResult = l_resultData » 3;
00177             l_resultData = 0;
00178             l_resultDataIndex = 0;
00179         }
00180         break;
00181     }
00182
00183     case 14:
00184     {
00185         if (l_resultDataIndex < 256)
00186         {
00187             l_resultData += ADC;
00188             l_resultDataIndex++;
00189         }
00190         else
00191         {
00192             m_conversionResult = l_resultData » 4;
00193             l_resultData = 0;
00194             l_resultDataIndex = 0;
00195         }
00196         break;
00197     }
00198
00199     case 15:
00200     {
00201         if (l_resultDataIndex < 1024)
00202         {
00203             l_resultData += ADC;
00204             l_resultDataIndex++;
00205         }
00206     }
```

```

00207         else
00208         {
00209             m_conversionResult = l_resultData » 5;
00210             l_resultData = 0;
00211             l_resultDataIndex = 0;
00212         }
00213         break;
00214     }
00215     case 16:
00216     {
00217         if (l_resultDataIndex < 4096)
00218         {
00219             l_resultData += ADC;
00220             l_resultDataIndex++;
00221         }
00222     }
00223     else
00224     {
00225         m_conversionResult = l_resultData » 6;
00226         l_resultData = 0;
00227         l_resultDataIndex = 0;
00228     }
00229     break;
00230 }
00231 }
00232 }
00233 }
00234 }
00235 }
00236 }
00237 }
00238 }

```

### 9.1.3.2 enableAutoTrigger()

```

void core::ADConverter::enableAutoTrigger (
    const uint8_t a_enable )

```

Enable autotrigger mode

#### Parameters

<i>a_enable</i>	Defines enable flag
-----------------	---------------------

Definition at line 72 of file [ADC.cpp](#).

```

00073 {
00074     if (a_enable) {
00075         ADC_ENABLE_AUTOTRIGGER;
00076     } else {
00077         ADC_DISABLE_AUTOTRIGGER;
00078     }
00079 }
00080 }
00081 }

```

References [ADC\\_DISABLE\\_AUTOTRIGGER](#), and [ADC\\_ENABLE\\_AUTOTRIGGER](#).

### 9.1.3.3 enableConversionCompleteInterrupt()

```

void core::ADConverter::enableConversionCompleteInterrupt (
    const uint8_t a_enable )

```

Enable conversion complete interrupt

## Parameters

<code>a_enable</code>	Defines enable flag
-----------------------	---------------------

Definition at line 83 of file [ADC.cpp](#).

```
00084 {
00085     if (a_enable) {
00086         ADC_ENABLE_CONVERSION_COMPLETE_INTERRUPT;
00087     } else {
00088         ADC_DISABLE_CONVERSION_COMPLETE_INTERRUPT;
00089     }
00090 }
00091
00092 }
```

References [ADC\\_DISABLE\\_CONVERSION\\_COMPLETE\\_INTERRUPT](#), and [ADC\\_ENABLE\\_CONVERSION\\_COMPLETE\\_INTERRUPT](#).

#### 9.1.3.4 getConversionResult()

```
uint16_t core::ADConverter::getConversionResult (
    const resolution & ar_resolution = resolution::res_10bit )
```

Get ADC conversion result

## Returns

Conversion result

Definition at line 243 of file [ADC.cpp](#).

```
00244 {
00245
00246
00247     switch (ar_resolution)
00248     {
00249         case core::resolution::res_8bit:
00250         {
00251             ADC_ADJUST_RESULT_LEFT;
00252             m_resolution = 8;
00253             break;
00254         }
00255         case core::resolution::res_9bit:
00256         {
00257             ADC_ADJUST_RESULT_LEFT;
00258             m_resolution = 9;
00259             break;
00260         }
00261         case core::resolution::res_10bit:
00262         {
00263             ADC_ADJUST_RESULT_RIGHT;
00264             m_resolution = 10;
00265             break;
00266         }
00267         case core::resolution::res_11bit:
00268         {
00269
00270             m_resolution = 11;
00271             break;
00272         }
00273         case core::resolution::res_12bit:
00274         {
00275             m_resolution = 12;
00276             break;
00277         }
00278         case core::resolution::res_13bit:
00279         {
00280             m_resolution = 13;
00281             break;
00282         }
00283         case core::resolution::res_14bit:
```

```

00284     {
00285         m_resolution = 14;
00286         break;
00287     }
00288     case core::resolution::res_15bit:
00289     {
00290         m_resolution = 15;
00291         break;
00292     }
00293     case core::resolution::res_16bit:
00294     {
00295         m_resolution = 16;
00296         break;
00297     }
00298 }
00299
00300 return m_conversionResult;
00301
00302
00303 }
```

References [ADC\\_ADJUST\\_RESULT\\_LEFT](#), [ADC\\_ADJUST\\_RESULT\\_RIGHT](#), [core::res\\_10bit](#), [core::res\\_11bit](#), [core::res\\_12bit](#), [core::res\\_13bit](#), [core::res\\_14bit](#), [core::res\\_15bit](#), [core::res\\_16bit](#), [core::res\\_8bit](#), and [core::res\\_9bit](#).

### 9.1.3.5 getInstance()

```

core::ADConverter & core::ADConverter::getInstance (
    const referenceVoltage & ar_refVoltage = referenceVoltage::AVCC,
    const clockPrescaler & ar_clockPrescaler = clockPrescaler::PS_128,
    const autoTriggerSource & ar_autoTriggerSource = autoTriggerSource::freeRunning,
    const io::Pin & ar_pin = io::Pin(0,io::PortC) ) [static]
```

Create a single instance of the ADC object

#### Parameters

<i>ar_refVoltage</i>	Defines reference voltage
<i>ar_clockPrescaler</i>	Defines clock prescaler
<i>ar_autoTriggerSource</i>	Defines auto trigger source
<i>ar_pin</i>	Defines a Pin object

Definition at line 8 of file [ADC.cpp](#).

```

00012 {
00013
00014     static ADConverter l_instance(ar_refVoltage,
00015                                 ar_clockPrescaler,
00016                                 ar_autoTriggerSource,
00017                                 ar_pin);
00018
00019     return l_instance;
00020 }
```

### 9.1.3.6 operator=()

```

const ADConverter& core::ADConverter::operator= (
    const ADConverter & ) [private]
```

Override assign operator

### 9.1.3.7 selectAnalogInput()

```
void core::ADConverter::selectAnalogInput (
    io::Pin a_pin )
```

Select analog input

#### Parameters

<i>a_pin</i>	Defines Pin object
--------------	--------------------

Definition at line 52 of file [ADC.cpp](#).

```
00053 {
00054     a_pin.toInput(0);
00055     ADC_SELECT_ANALOG_INPUT(a_pin.getPinNumber());
00056     ADC_DISABLE_DIGITAL_INPUT_REGISTER(a_pin.getPinNumber());
00057
00058 }
```

References [ADC\\_DISABLE\\_DIGITAL\\_INPUT\\_REGISTER](#), [ADC\\_SELECT\\_ANALOG\\_INPUT](#), [io::Pin::getPinNumber\(\)](#), and [io::Pin::toInput\(\)](#).

### 9.1.3.8 selectAutoTriggerSource()

```
void core::ADConverter::selectAutoTriggerSource (
    const autoTriggerSource & ar_autoTriggerSource )
```

Select autotriggger source

#### Parameters

<i>ar_autoTriggerSource</i>	Defines autotriggger source
-----------------------------	-----------------------------

Definition at line 101 of file [ADC.cpp](#).

```
00102 {
00103     ADC_SELECT_AUTO_TRIGGER_SOURCE(static_cast<uint8_t>(ar_autoTriggerSource));
00104
00105
00106 }
```

References [ADC\\_SELECT\\_AUTO\\_TRIGGER\\_SOURCE](#).

### 9.1.3.9 selectClockPrescaler()

```
void core::ADConverter::selectClockPrescaler (
    const clockPrescaler & ar_clockPrescaler )
```

Select clock prescaler

## Parameters

<i>ar_clockPrescaler</i>	Defines clock prescaler
--------------------------	-------------------------

Definition at line 94 of file [ADC.cpp](#).

```
00095 {  
00096     ADC_SELECT_CLOCK_PRESCALER(static_cast<uint8_t>(ar_clockPrescaler));  
00097  
00098 }
```

References [ADC\\_SELECT\\_CLOCK\\_PRESCALER](#).

### 9.1.3.10 selectReferenceVoltage()

```
void core::ADConverter::selectReferenceVoltage (  
    const referenceVoltage & ar_refVoltage )
```

Select reference voltage

## Parameters

<i>ar_refVoltage</i>	Defines reference voltage
----------------------	---------------------------

Definition at line 46 of file [ADC.cpp](#).

```
00047 {  
00048     ADC_SELECT_REF_VOLTAGE(static_cast<uint8_t>(ar_refVoltage));  
00049  
00050 }
```

References [ADC\\_SELECT\\_REF\\_VOLTAGE](#).

### 9.1.3.11 start()

```
void core::ADConverter::start ( )
```

Start ADC conversion

Definition at line 60 of file [ADC.cpp](#).

```
00061 {  
00062     ADC_ENABLE;  
00063     ADC_START_CONVERSION;  
00064 }
```

References [ADC\\_ENABLE](#), and [ADC\\_START\\_CONVERSION](#).

#### 9.1.3.12 stop()

```
void core::ADConverter::stop ( )
```

Stop ADC conversion

Definition at line 66 of file [ADC.cpp](#).

```
00067 {  
00068     ADC_STOP_CONVERSION;  
00069     ADC_DISABLE;  
00070 }
```

References [ADC\\_DISABLE](#), and [ADC\\_STOP\\_CONVERSION](#).

### 9.1.4 Member Data Documentation

#### 9.1.4.1 \_\_externally\_visible\_\_

```
void core::ADConverter::__externally_visible__
```

Definition at line 153 of file [ADC.h](#).

#### 9.1.4.2 \_\_used\_\_

```
void core::ADConverter::__used__
```

Definition at line 153 of file [ADC.h](#).

#### 9.1.4.3 m\_conversionResult

```
volatile uint16_t core::ADConverter::m_conversionResult = 0 [static], [private]
```

Conversion result

Definition at line 187 of file [ADC.h](#).

#### 9.1.4.4 m\_resolution

```
uint8_t core::ADConverter::m_resolution = 10 [static], [private]
```

Conversion resolution

Definition at line 189 of file [ADC.h](#).



## 9.2 core::AnalogComparator Class Reference

```
#include <AnalogComparator.h>
```

### 9.2.1 Detailed Description

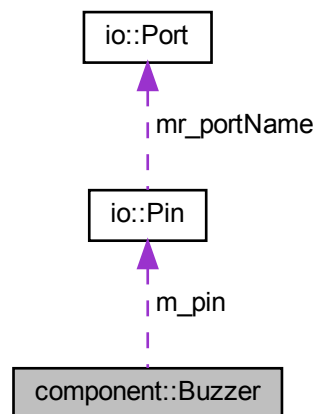
Definition at line 24 of file [AnalogComparator.h](#).

## 9.3 component::Buzzer Class Reference

Class for handling a [Buzzer](#) component.

```
#include <Buzzer.h>
```

Collaboration diagram for component::Buzzer:



### Public Member Functions

- [Buzzer](#) (const [io::Pin](#) &ar\_pin)
- [~Buzzer](#) ()
- void [buzz](#) (const uint16\_t &ar\_period\_us, const uint16\_t &ar\_duration\_us)
- void [pause](#) (uint16\_t a\_duration\_us)
- template<typename TC >  
void [buzz](#) (TC &ar\_timerCounter, const uint16\_t &ar\_period\_us, uint16\_t &ar\_duration\_ms, const [core::channel](#) &ar\_channel=[core::channel::A](#), const [core::clockSource](#) &ar\_clockSource=[core::clockSource::PS\\_64](#))

### Private Attributes

- [io::Pin](#) m\_pin

### 9.3.1 Detailed Description

Class for handling a [Buzzer](#) component.

The class implements [Buzzer](#) operations

#### Author

Farid Oubbati ( [farid.oubbati@outlook.com](mailto:farid.oubbati@outlook.com) )

#### Date

March 2018

Definition at line [74](#) of file [Buzzer.h](#).

### 9.3.2 Constructor & Destructor Documentation

#### 9.3.2.1 Buzzer()

```
component::Buzzer::Buzzer (
    const io::Pin & ar_pin )
```

#### Constructor

Initializes the [Buzzer](#) object

#### Parameters

<a href="#">ar_pin</a>	Defines a Pin object
------------------------	----------------------

#### Author

Farid Oubbati ( [farid.oubbati@outlook.com](mailto:farid.oubbati@outlook.com) )

#### Date

March 2018

- See [Buzzer.h](#) for a description of this code

Definition at line [13](#) of file [Buzzer.cpp](#).

```
00014         : m\_pin(ar_pin)
00015 {
00016     m\_pin.toOutput();
00017
00018 }
```

References [m\\_pin](#), and [io::Pin::toOutput\(\)](#).

### 9.3.2.2 ~Buzzer()

```
component::Buzzer::~Buzzer ( )
```

Destructor

Definition at line 20 of file [Buzzer.cpp](#).

```
00021 {
00022
00023 }
```

## 9.3.3 Member Function Documentation

### 9.3.3.1 buzz() [1/2]

```
void component::Buzzer::buzz (
    const uint16_t & ar_period_us,
    const uint16_t & ar_duration_us )
```

Generate a [Buzzer](#) signal

Parameters

<i>ar_period_us</i>	Defines period of <a href="#">Buzzer</a> signal
<i>ar_duration_us</i>	Defines duration of <a href="#">Buzzer</a> signal

Definition at line 25 of file [Buzzer.cpp](#).

```
00026 {
00027
00028     for (uint16_t i = 0; i < ar_duration_us; i += ar_period_us)
00029     {
00030         // for loop with variable delay selects the pitch
00031         for (uint16_t j = 0; j < ar_period_us; j++)
00032         {
00033             _delay_us(1);
00034         }
00035         m_pin.toggle();
00036     }
00037     m_pin.setLow();
00038
00039 }
```

### 9.3.3.2 buzz() [2/2]

```
template<typename TC >
void component::Buzzer::buzz (
    TC & ar_timerCounter,
    const uint16_t & ar_period_us,
    uint16_t & ar_duration_ms,
    const core::channel & ar_channel = core::channel::A,
    const core::clockSource & ar_clockSource = core::clockSource::PS_64 ) [inline]
```

Generate a [Buzzer](#) signal

## Template Parameters

<i>TC</i>	Type Defines a TimerCounter object
-----------	------------------------------------

## Parameters

<i>ar_period_us</i>	Defines period of <a href="#">Buzzer</a> signal
<i>ar_duration_ms</i>	Defines duration of <a href="#">Buzzer</a> signal
<i>ar_channel</i>	Defines TimerCounter channel
<i>ar_clockSource</i>	Defines TimerCounter clock source

Definition at line 114 of file [Buzzer.h](#).

```

00120     {
00121         ar_timerCounter.selectOperationMode(core::operationMode::CTC_OCR);
00122         ar_timerCounter.selectCompareOutputMode(ar_channel, core::compareOutputMode::toggle);
00123         ar_timerCounter.setCounter(0);
00124         ar_timerCounter.setOutputCompareRegister(ar_channel, ar_period_us);
00125         // start timer
00126         ar_timerCounter.start();
00127         // wait for the pitch duration
00128         while (ar_duration_ms) {
00129             _delay_ms(1);
00130             ar_duration_ms--;
00131         }
00132         // no buzz
00133         ar_timerCounter.selectCompareOutputMode(ar_channel, core::compareOutputMode::normal);
00134         ar_timerCounter.stop();
00135     }

```

References [core::CTC\\_OCR](#), [core::normal](#), and [core::toggle](#).

### 9.3.3.3 pause()

```

void component::Buzzer::pause (
    uint16_t a_duration_us )

```

Generate a time break

## Parameters

<i>a_duration_us</i>	Defines duration of time break
----------------------	--------------------------------

Definition at line 42 of file [Buzzer.cpp](#).

```

00043 {
00044     do {
00045         _delay_us(1);
00046     } while (--a_duration_us);
00047 }

```

## 9.3.4 Member Data Documentation

## 9.3.4.1 m\_pin

```
io::Pin component::Buzzer::m_pin [private]
```

Pin object

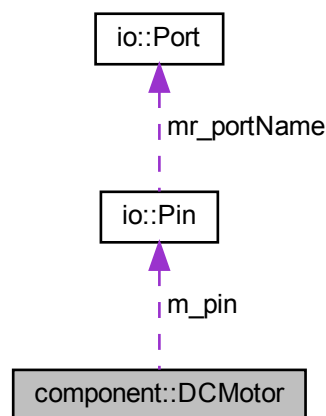
Definition at line 142 of file [Buzzer.h](#).

Referenced by [Buzzer\(\)](#).

## 9.4 component::DCMotor Class Reference

```
#include <DCMotor.h>
```

Collaboration diagram for component::DCMotor:



## Public Member Functions

- `DCMotor` (const `io::Pin` &ar\_pin)
- `~DCMotor` ()
- void `on` ()
- void `off` ()
- void `toggle` ()
- template<typename TC >  
void `spin` (TC &ar\_timerCounter, const uint16\_t &ar\_speed, const `core::channel` &ar\_channel=`core::channel::A`)
- template<typename TC >  
void `stop` (TC &ar\_timerCounter)
- template<typename TC >  
void `connect` (TC &ar\_timerCounter, const `core::channel` &ar\_channel=`core::channel::A`)
- template<typename TC >  
void `disconnect` (TC &ar\_timerCounter, const `core::channel` &ar\_channel=`core::channel::A`)

## Private Attributes

- [io::Pin m\\_pin](#)

### 9.4.1 Detailed Description

Definition at line 126 of file [DCMotor.h](#).

### 9.4.2 Constructor & Destructor Documentation

#### 9.4.2.1 DCMotor()

```
component::DCMotor::DCMotor (
    const io::Pin & ar_pin )
```

Definition at line 3 of file [DCMotor.cpp](#).

```
00004         : m_pin(ar_pin)
00005 {
00006     m_pin.toOutput();
00007
00008 }
```

References [m\\_pin](#), and [io::Pin::toOutput\(\)](#).

#### 9.4.2.2 ~DCMotor()

```
component::DCMotor::~DCMotor ( )
```

Definition at line 11 of file [DCMotor.cpp](#).

```
00012 {
00013
00014 }
```

### 9.4.3 Member Function Documentation

#### 9.4.3.1 connect()

```
template<typename TC >
void component::DCMotor::connect (
    TC & ar_timerCounter,
    const core::channel & ar_channel = core::channel::A ) [inline]
```

Definition at line 164 of file [DCMotor.h](#).

```
00166     {
00167         ar_timerCounter.selectOperationMode(core::operationMode::fast_PWM);
00168         ar_timerCounter.selectCompareOutputMode(ar_channel, core::compareOutputMode::clear);
00169         ar_timerCounter.setCounter(0);
00170
00171     }
```

References [core::clear](#), and [core::fast\\_PWM](#).

### 9.4.3.2 disconnect()

```
template<typename TC >
void component::DCMotor::disconnect (
    TC & ar_timerCounter,
    const core::channel & ar_channel = core::channel::A ) [inline]
```

Definition at line 174 of file [DCMotor.h](#).

```
00176     {
00177         ar_timerCounter.selectCompareOutputMode(ar_channel, core::compareOutputMode::normal);
00178         ar_timerCounter.stop();
00179     }
```

References [core::normal](#).

### 9.4.3.3 off()

```
void component::DCMotor::off ( )
```

Turn servo motor Off.

Definition at line 21 of file [DCMotor.cpp](#).

```
00022 {
00023     m_pin.setLow();
00024 }
```

### 9.4.3.4 on()

```
void component::DCMotor::on ( )
```

Turn servo motor On.

Definition at line 16 of file [DCMotor.cpp](#).

```
00017 {
00018     m_pin.setHigh();
00019 }
```

### 9.4.3.5 spin()

```
template<typename TC >
void component::DCMotor::spin (
    TC & ar_timerCounter,
    const uint16_t & ar_speed,
    const core::channel & ar_channel = core::channel::A ) [inline]
```

Definition at line 144 of file [DCMotor.h](#).

```
00148     {
00149
00150         ar_timerCounter.setOutputCompareRegister(ar_channel, ar_speed);
00151         ar_timerCounter.start();
00152
00153
00154     }
```

#### 9.4.3.6 stop()

```
template<typename TC >
void component::DCMotor::stop (
    TC & ar_timerCounter ) [inline]
```

Definition at line 156 of file [DCMotor.h](#).

```
00157     {
00158         ar_timerCounter.stop();
00159     }
00160
00161 }
```

#### 9.4.3.7 toggle()

```
void component::DCMotor::toggle ( )
```

Toggle servo motor state.

Definition at line 26 of file [DCMotor.cpp](#).

```
00027 {
00028     m_pin.toggle();
00029 }
00030 }
```

### 9.4.4 Member Data Documentation

#### 9.4.4.1 m\_pin

```
io::Pin component::DCMotor::m_pin [private]
```

pin object

Definition at line 185 of file [DCMotor.h](#).

Referenced by [DCMotor\(\)](#).

## 9.5 core::ExternInterrupt Class Reference

```
#include <ExternInterrupt.h>
```

### Public Member Functions

- void [setInt0SenseControl](#) (const [senseControl](#) &ar\_senseControl)
- void [setInt1SenseControl](#) (const [senseControl](#) &ar\_senseControl)
- void [enableInt0](#) (const uint8\_t a\_enable)
- void [enableInt1](#) (const uint8\_t a\_enable)
- void [enablePinChange](#) (const [pinChangePort](#) &ar\_pinChangePort, const uint8\_t a\_enable)
- void [enablePinChangeMaskPortB](#) (const uint8\_t a\_pinNumber, const uint8\_t a\_enable)
- void [enablePinChangeMaskPortC](#) (const uint8\_t a\_pinNumber, const uint8\_t a\_enable)
- void [enablePinChangeMaskPortD](#) (const uint8\_t a\_pinNumber, const uint8\_t a\_enable)



## Static Public Member Functions

- static [ExternInterrupt](#) & [getInstance](#) ()
- static void [Int0ServiceRoutine](#) () \_\_asm\_\_(STR(EXT\_INT\_INT0\_INTERRUPT)) \_\_attribute\_\_((\_\_signal\_\_
- static void [Int1ServiceRoutine](#) () \_\_asm\_\_(STR(EXT\_INT\_INT1\_INTERRUPT)) \_\_attribute\_\_((\_\_signal\_\_
- static void [pinChangePortBServiceRoutine](#) () \_\_asm\_\_(STR(EXT\_INT\_PIN\_CHANGE\_PORTB\_INTERRUPT)) \_\_attribute\_\_((\_\_signal\_\_
- static void [pinChangePortCServiceRoutine](#) () \_\_asm\_\_(STR(EXT\_INT\_PIN\_CHANGE\_PORTC\_INTERRUPT)) \_\_attribute\_\_((\_\_signal\_\_
- static void [pinChangePortDServiceRoutine](#) () \_\_asm\_\_(STR(EXT\_INT\_PIN\_CHANGE\_PORTD\_INTERRUPT)) \_\_attribute\_\_((\_\_signal\_\_

## Public Attributes

- static void [\\_\\_used\\_\\_](#)
- static void [\\_\\_externally\\_visible\\_\\_](#)

## Private Member Functions

- [ExternInterrupt](#) ()
- [~ExternInterrupt](#) ()
- [ExternInterrupt](#) (const [ExternInterrupt](#) &)
- const [ExternInterrupt](#) & [operator=](#) (const [ExternInterrupt](#) &)

### 9.5.1 Detailed Description

Definition at line 162 of file [ExternInterrupt.h](#).

### 9.5.2 Constructor & Destructor Documentation

#### 9.5.2.1 ExternInterrupt() [1/2]

```
core::ExternInterrupt::ExternInterrupt ( ) [private]
```

Definition at line 11 of file [ExternInterrupt.cpp](#).

```
00012 {
00013     sei();
00014
00015 }
```

### 9.5.2.2 ~ExternInterrupt()

```
core::ExternInterrupt::~~ExternInterrupt ( ) [private]
```

Destructor.

Definition at line 17 of file [ExternInterrupt.cpp](#).

```
00018 {  
00019  
00020 }
```

### 9.5.2.3 ExternInterrupt() [2/2]

```
core::ExternInterrupt::ExternInterrupt (  
    const ExternInterrupt & ) [private]
```

Overried Copy constructor.

## 9.5.3 Member Function Documentation

### 9.5.3.1 enableInt0()

```
void core::ExternInterrupt::enableInt0 (  
    const uint8_t a_enable )
```

Definition at line 34 of file [ExternInterrupt.cpp](#).

```
00035 {  
00036     if (a_enable) {  
00037         EXT\_INT\_ENABLE\_INT0;  
00038     } else {  
00039         EXT\_INT\_DISABLE\_INT0;  
00040     }  
00041 }  
00042  
00043 }
```

References [EXT\\_INT\\_DISABLE\\_INT0](#), and [EXT\\_INT\\_ENABLE\\_INT0](#).

### 9.5.3.2 enableInt1()

```
void core::ExternInterrupt::enableInt1 (  
    const uint8_t a_enable )
```

Definition at line 45 of file [ExternInterrupt.cpp](#).

```
00046 {  
00047     if (a_enable) {  
00048         EXT\_INT\_ENABLE\_INT1;  
00049     } else {  
00050         EXT\_INT\_DISABLE\_INT1;  
00051     }  
00052 }  
00053 }
```

References [EXT\\_INT\\_DISABLE\\_INT1](#), and [EXT\\_INT\\_ENABLE\\_INT1](#).

### 9.5.3.3 enablePinChange()

```
void core::ExternInterrupt::enablePinChange (
    const pinChangePort & ar_pinChangePort,
    const uint8_t a_enable )
```

Definition at line 56 of file [ExternInterrupt.cpp](#).

```
00057 {
00058     if (a_enable) {
00059         EXT_INT_ENABLE_PIN_CHANGE_INTERRUPT(static_cast<uint8_t>(ar_pinChangePort));
00060     } else {
00061         EXT_INT_DISABLE_PIN_CHANGE_INTERRUPT(static_cast<uint8_t>(ar_pinChangePort));
00062     }
00063 }
00064
00065 }
```

References [EXT\\_INT\\_DISABLE\\_PIN\\_CHANGE\\_INTERRUPT](#), and [EXT\\_INT\\_ENABLE\\_PIN\\_CHANGE\\_INTERRUPT](#).

### 9.5.3.4 enablePinChangeMaskPortB()

```
void core::ExternInterrupt::enablePinChangeMaskPortB (
    const uint8_t a_pinNumber,
    const uint8_t a_enable )
```

Definition at line 67 of file [ExternInterrupt.cpp](#).

```
00068 {
00069     if (a_enable) {
00070         EXT_INT_ENABLE_PIN_CHANGE_MASK_PORTB(a_pinNumber);
00071     } else {
00072         EXT_INT_ENABLE_PIN_CHANGE_MASK_PORTB(a_pinNumber);
00073     }
00074 }
00075
00076 }
```

References [EXT\\_INT\\_ENABLE\\_PIN\\_CHANGE\\_MASK\\_PORTB](#).

### 9.5.3.5 enablePinChangeMaskPortC()

```
void core::ExternInterrupt::enablePinChangeMaskPortC (
    const uint8_t a_pinNumber,
    const uint8_t a_enable )
```

Definition at line 79 of file [ExternInterrupt.cpp](#).

```
00080 {
00081     if (a_enable) {
00082         EXT_INT_ENABLE_PIN_CHANGE_MASK_PORTC(a_pinNumber);
00083     } else {
00084         EXT_INT_ENABLE_PIN_CHANGE_MASK_PORTC(a_pinNumber);
00085     }
00086 }
00087
00088 }
```

References [EXT\\_INT\\_ENABLE\\_PIN\\_CHANGE\\_MASK\\_PORTC](#).

### 9.5.3.6 enablePinChangeMaskPortD()

```
void core::ExternInterrupt::enablePinChangeMaskPortD (
    const uint8_t a_pinNumber,
    const uint8_t a_enable )
```

Definition at line 90 of file [ExternInterrupt.cpp](#).

```
00091 {
00092     if (a_enable) {
00093         EXT_INT_ENABLE_PIN_CHANGE_MASK_PORTD(a_pinNumber);
00094     } else {
00095         EXT_INT_ENABLE_PIN_CHANGE_MASK_PORTD(a_pinNumber);
00096     }
00097 }
00098 }
```

References [EXT\\_INT\\_ENABLE\\_PIN\\_CHANGE\\_MASK\\_PORTD](#).

### 9.5.3.7 getInstance()

```
core::ExternInterrupt & core::ExternInterrupt::getInstance ( ) [static]
```

Definition at line 3 of file [ExternInterrupt.cpp](#).

```
00004 {
00005     static ExternInterrupt l_instance;
00006     return l_instance;
00007 }
00008 }
```

### 9.5.3.8 Int0ServiceRoutine()

```
static void core::ExternInterrupt::Int0ServiceRoutine ( ) [static]
```

### 9.5.3.9 Int1ServiceRoutine()

```
static void core::ExternInterrupt::Int1ServiceRoutine ( ) [static]
```

### 9.5.3.10 operator=()

```
const ExternInterrupt& core::ExternInterrupt::operator= (
    const ExternInterrupt & ) [private]
```

Override assign operator.

#### 9.5.3.11 pinChangePortBServiceRoutine()

```
static void core::ExternInterrupt::pinChangePortBServiceRoutine ( ) [static]
```

#### 9.5.3.12 pinChangePortCServiceRoutine()

```
static void core::ExternInterrupt::pinChangePortCServiceRoutine ( ) [static]
```

#### 9.5.3.13 pinChangePortDServiceRoutine()

```
static void core::ExternInterrupt::pinChangePortDServiceRoutine ( ) [static]
```

#### 9.5.3.14 setInt0SenseControl()

```
void core::ExternInterrupt::setInt0SenseControl (
    const senseControl & ar_senseControl )
```

Definition at line 22 of file [ExternInterrupt.cpp](#).

```
00023 {
00024     EXT\_INT\_SET\_INT0\_SENSE\_CONTROL(static_cast<uint8_t>(ar_senseControl));
00025
00026 }
```

References [EXT\\_INT\\_SET\\_INT0\\_SENSE\\_CONTROL](#).

#### 9.5.3.15 setInt1SenseControl()

```
void core::ExternInterrupt::setInt1SenseControl (
    const senseControl & ar_senseControl )
```

Definition at line 28 of file [ExternInterrupt.cpp](#).

```
00029 {
00030     EXT\_INT\_SET\_INT1\_SENSE\_CONTROL(static_cast<uint8_t>(ar_senseControl));
00031
00032 }
```

References [EXT\\_INT\\_SET\\_INT1\\_SENSE\\_CONTROL](#).

### 9.5.4 Member Data Documentation

#### 9.5.4.1 `__externally_visible__`

```
static void core::ExternInterrupt::__externally_visible__
```

Definition at line 185 of file [ExternInterrupt.h](#).

#### 9.5.4.2 `__used__`

```
static void core::ExternInterrupt::__used__
```

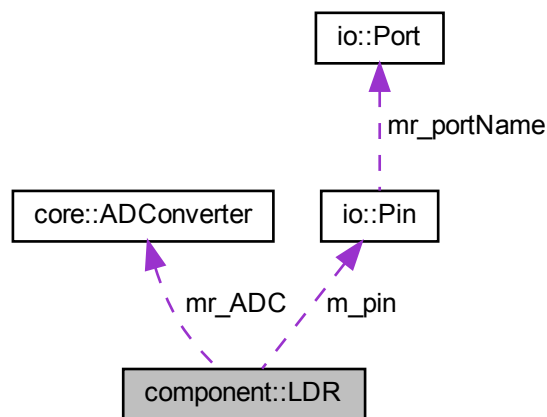
Definition at line 185 of file [ExternInterrupt.h](#).

## 9.6 `component::LDR` Class Reference

Class for handling an [LDR](#) component.

```
#include <LDR.h>
```

Collaboration diagram for `component::LDR`:



### Public Member Functions

- [LDR](#) (const [io::Pin](#) &ar\_pin, [core::ADConverter](#) &ar\_ADC)
- [~LDR](#) ()
- uint32\_t [getResistance](#) (const uint8\_t &ar\_VDResistor)
- uint16\_t [getLux](#) (const uint8\_t &ar\_VDResistor)

## Private Attributes

- [io::Pin](#) `m_pin`
- [core::ADConverter](#) & `mr_ADC`

### 9.6.1 Detailed Description

Class for handling an [LDR](#) component.

The class implements [LDR](#) operations

#### Author

Farid Oubbati ( [farid.oubbati@outlook.com](mailto:farid.oubbati@outlook.com) )

#### Date

March 2018

Definition at line [27](#) of file [LDR.h](#).

### 9.6.2 Constructor & Destructor Documentation

#### 9.6.2.1 LDR()

```
component::LDR::LDR (
    const io::Pin & ar_pin,
    core::ADConverter & ar_ADC )
```

#### Constructor

Initializes the [LDR](#) object

#### Parameters

<i>ar_pin</i>	Defines a Pin object
<i>ar_ADC</i>	Defines the ADC object

Definition at line [4](#) of file [LDR.cpp](#).

```
00005         : m\_pin(ar_pin), mr\_ADC(ar_ADC)
00006
00007 {
00008     m\_pin.toInput(0);
00009
00010 }
```

References [m\\_pin](#), and [io::Pin::toInput\(\)](#).

### 9.6.2.2 ~LDR()

```
component::LDR::~~LDR ( )
```

Destructor

Definition at line 12 of file [LDR.cpp](#).

```
00013 {
00014
00015 }
```

## 9.6.3 Member Function Documentation

### 9.6.3.1 getLux()

```
uint16_t component::LDR::getLux (
    const uint8_t & ar_VDResistor )
```

Compute amount of illuminance

Parameters

<i>ar_VDResistor</i>	Defines the voltage divider resistor in Kohm
----------------------	----------------------------------------------

Returns

Amount of illuminance

Definition at line 23 of file [LDR.cpp](#).

```
00024 {
00025     // see:
00026     https://www.digikey.com/en/maker/projects/design-a-luxmeter-with-an-ldr-and-an-arduino/623aeee0f93e427bb57e02c4592567d
00027     return 12518931 *
00028     pow(ar_VDResistor*((1000UL*1023/mr_ADC.getConversionResult(core::resolution::res_10bit))-1000UL),
00029         -1.405); ;
00027 }
```

References [core::res\\_10bit](#).

### 9.6.3.2 getResistance()

```
uint32_t component::LDR::getResistance (
    const uint8_t & ar_VDResistor )
```

Compute [LDR](#) resistance

Parameters

<i>ar_VDResistor</i>	Defines the voltage divider resistor in Kohm
----------------------	----------------------------------------------



### Returns

[LDR](#) resistance

Definition at line 17 of file [LDR.cpp](#).

```
00018 {  
00019     // 10bit resolution -> 1023  
00020     return  
        ar_VDResistor*((1000UL*1023/mr_ADC.getConversionResult(core::resolution::res_10bit))-1000UL);  
00021 }
```

References [core::res\\_10bit](#).

## 9.6.4 Member Data Documentation

### 9.6.4.1 m\_pin

[io::Pin](#) component::LDR::m\_pin [private]

Pin object

Definition at line 62 of file [LDR.h](#).

Referenced by [LDR\(\)](#).

### 9.6.4.2 mr\_ADC

[core::ADConverter&](#) component::LDR::mr\_ADC [private]

ADC object reference

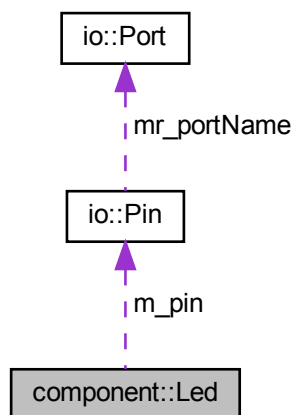
Definition at line 63 of file [LDR.h](#).

## 9.7 component::Led Class Reference

Class for handling a [Led](#) component.

```
#include <Led.h>
```

Collaboration diagram for component::Led:



## Public Member Functions

- `Led` (const `io::Pin` &ar\_pin)
- `~Led` ()
- void `on` ()
- void `off` ()
- void `toggle` ()
- uint8\_t `isOn` ()
- uint8\_t `isOff` ()

## Private Attributes

- `io::Pin` `m_pin`

### 9.7.1 Detailed Description

Class for handling a `Led` component.

The class implements `Led` operations

#### Author

Farid Oubbati ( [farid.oubbati@outlook.com](mailto:farid.oubbati@outlook.com))

#### Date

March 2018

Definition at line 26 of file `Led.h`.

## 9.7.2 Constructor & Destructor Documentation

### 9.7.2.1 Led()

```
component::Led::Led (
    const io::Pin & ar_pin )
```

Constructor

Initializes the [Led](#) object

Parameters

<code>ar_pin</code>	Defines a Pin object
---------------------	----------------------

Author

Farid Oubbati ( [farid.oubbati@outlook.com](mailto:farid.oubbati@outlook.com) )

Date

March 2018

- See [Led.h](#) for a description of this code

Definition at line 12 of file [Led.cpp](#).

```
00013         : m_pin(ar_pin)
00014 {
00015     m_pin.toOutput();
00016 }
```

References [m\\_pin](#), and [io::Pin::toOutput\(\)](#).

### 9.7.2.2 ~Led()

```
component::Led::~~Led ( )
```

Destructor

Definition at line 19 of file [Led.cpp](#).

```
00020 {
00021
00022 }
```

## 9.7.3 Member Function Documentation

#### 9.7.3.1 isOff()

```
uint8_t component::Led::isOff ( )
```

Is [Led](#) Off

##### Returns

[Led](#) status

Definition at line 51 of file [Led.cpp](#).

```
00052 {  
00053     return m_pin.isLow();  
00054  
00055 }
```

#### 9.7.3.2 isOn()

```
uint8_t component::Led::isOn ( )
```

Is [Led](#) On

##### Returns

[Led](#) status

Definition at line 44 of file [Led.cpp](#).

```
00045 {  
00046     return m_pin.isHigh();  
00047  
00048 }
```

#### 9.7.3.3 off()

```
void component::Led::off ( )
```

Turn [Led](#) Off

Definition at line 31 of file [Led.cpp](#).

```
00032 {  
00033     m_pin.setLow();  
00034  
00035 }
```

#### 9.7.3.4 on()

```
void component::Led::on ( )
```

Turn [Led](#) On

Definition at line 25 of file [Led.cpp](#).

```
00026 {  
00027     m_pin.setHigh();  
00028  
00029 }
```

#### 9.7.3.5 toggle()

```
void component::Led::toggle ( )
```

Toggle [Led](#) state

Definition at line 38 of file [Led.cpp](#).

```
00039 {  
00040     m_pin.toggle();  
00041  
00042 }
```

### 9.7.4 Member Data Documentation

#### 9.7.4.1 m\_pin

```
io::Pin component::Led::m_pin [private]
```

Pin object

Definition at line 68 of file [Led.h](#).

Referenced by [Led\(\)](#).

## 9.8 core::MCU Class Reference

```
#include <MCU.h>
```

## Static Public Member Functions

- static void [init](#) ()
- static void [selectSleepMode](#) (const [sleepMode](#) &a\_sleepMode)
- static void [goToSleep](#) (const [BODMode](#) &a\_BODMode)
- static void [sleepEnable](#) (const uint8\_t ar\_enable)
- static void [enableUSART0](#) (const uint8\_t a\_enable)
- static void [enableTimerCounter0](#) (const uint8\_t a\_enable)
- static void [enableTimerCounter1](#) (const uint8\_t a\_enable)
- static void [enableTimerCounter2](#) (const uint8\_t a\_enable)
- static void [enableTWI](#) (const uint8\_t a\_enable)
- static void [enableSPI](#) (const uint8\_t a\_enable)
- static void [enableADC](#) (const uint8\_t a\_enable)
- static void [disableBOD](#) ()

### 9.8.1 Detailed Description

Definition at line 91 of file [MCU.h](#).

### 9.8.2 Member Function Documentation

#### 9.8.2.1 [disableBOD\(\)](#)

```
void core::MCU::disableBOD ( ) [static]
```

Definition at line 126 of file [MCU.cpp](#).

```
00127 {
00128     MCU\_BOD\_DISABLE;
00129 }
```

References [MCU\\_BOD\\_DISABLE](#).

#### 9.8.2.2 [enableADC\(\)](#)

```
void core::MCU::enableADC (
    const uint8_t a_enable ) [static]
```

Definition at line 117 of file [MCU.cpp](#).

```
00118 {
00119     if (a_enable) {
00120         MCU\_ADC\_ENABLE;
00121     } else {
00122         MCU\_ADC\_DISABLE;
00123     }
00124 }
```

References [MCU\\_ADC\\_DISABLE](#), and [MCU\\_ADC\\_ENABLE](#).

Referenced by [core::ADConverter::ADConverter\(\)](#), and [init\(\)](#).

### 9.8.2.3 enableSPI()

```
void core::MCU::enableSPI (
    const uint8_t a_enable ) [static]
```

Definition at line 107 of file [MCU.cpp](#).

```
00108 {
00109     if (a_enable) {
00110         MCU_SPI_ENABLE;
00111     } else {
00112         MCU_SPI_DISABLE;
00113     }
00114 }
00115 }
```

References [MCU\\_SPI\\_DISABLE](#), and [MCU\\_SPI\\_ENABLE](#).

Referenced by [init\(\)](#), and [io::SPI::SPI\(\)](#).

### 9.8.2.4 enableTimerCounter0()

```
void core::MCU::enableTimerCounter0 (
    const uint8_t a_enable ) [static]
```

Definition at line 67 of file [MCU.cpp](#).

```
00068 {
00069     if (a_enable) {
00070         MCU_TIMER0_ENABLE;
00071     } else {
00072         MCU_TIMER0_DISABLE;
00073     }
00074 }
00075 }
```

References [MCU\\_TIMER0\\_DISABLE](#), and [MCU\\_TIMER0\\_ENABLE](#).

Referenced by [init\(\)](#), and [core::TimerCounter0::TimerCounter0\(\)](#).

### 9.8.2.5 enableTimerCounter1()

```
void core::MCU::enableTimerCounter1 (
    const uint8_t a_enable ) [static]
```

Definition at line 77 of file [MCU.cpp](#).

```
00078 {
00079     if (a_enable) {
00080         MCU_TIMER1_ENABLE;
00081     } else {
00082         MCU_TIMER1_DISABLE;
00083     }
00084 }
00085 }
```

References [MCU\\_TIMER1\\_DISABLE](#), and [MCU\\_TIMER1\\_ENABLE](#).

Referenced by [init\(\)](#), and [core::TimerCounter1::TimerCounter1\(\)](#).

### 9.8.2.6 enableTimerCounter2()

```
void core::MCU::enableTimerCounter2 (
    const uint8_t a_enable ) [static]
```

Definition at line 87 of file [MCU.cpp](#).

```
00088 {
00089     if (a_enable) {
00090         MCU_TIMER2_ENABLE;
00091     } else {
00092         MCU_TIMER2_DISABLE;
00093     }
00094
00095 }
```

References [MCU\\_TIMER2\\_DISABLE](#), and [MCU\\_TIMER2\\_ENABLE](#).

Referenced by [init\(\)](#), and [core::TimerCounter2::TimerCounter2\(\)](#).

### 9.8.2.7 enableTWI()

```
void core::MCU::enableTWI (
    const uint8_t a_enable ) [static]
```

Definition at line 97 of file [MCU.cpp](#).

```
00098 {
00099     if (a_enable) {
00100         MCU_TWI_ENABLE;
00101     } else {
00102         MCU_TWI_DISABLE;
00103     }
00104
00105 }
```

References [MCU\\_TWI\\_DISABLE](#), and [MCU\\_TWI\\_ENABLE](#).

Referenced by [init\(\)](#).

### 9.8.2.8 enableUSART0()

```
void core::MCU::enableUSART0 (
    const uint8_t a_enable ) [static]
```

Definition at line 57 of file [MCU.cpp](#).

```
00058 {
00059     if (a_enable) {
00060         MCU_USART0_ENABLE;
00061     } else {
00062         MCU_USART0_DISABLE;
00063     }
00064
00065 }
```

References [MCU\\_USART0\\_DISABLE](#), and [MCU\\_USART0\\_ENABLE](#).

Referenced by [init\(\)](#), and [io::USART0::USART0\(\)](#).



### 9.8.2.9 goToSleep()

```
void core::MCU::goToSleep (
    const BODMode & a_BODMode ) [static]
```

Definition at line 21 of file [MCU.cpp](#).

```
00022 {
00023     cli();
00024     switch (a_BODMode)
00025     {
00026         case core::BODMode::enabled:
00027         {
00028             sleepEnable(1);
00029             sei();
00030             sleep_cpu();
00031             sleepEnable(0);
00032             break;
00033         }
00034         case core::BODMode::disabled:
00035         {
00036             sleepEnable(1);
00037             disableBOD();
00038             sei();
00039             sleep_cpu();
00040             sleepEnable(0);
00041             break;
00042         }
00043     }
00044 }
00045 }
```

References [core::disabled](#), and [core::enabled](#).

### 9.8.2.10 init()

```
void core::MCU::init ( ) [static]
```

Definition at line 3 of file [MCU.cpp](#).

```
00004 {
00005     enableUSART0(0);
00006     enableTimerCounter0(0);
00007     enableTimerCounter1(0);
00008     enableTimerCounter2(0);
00009     enableTWI(0);
00010     enableSPI(0);
00011     enableADC(0);
00012 }
00013 }
```

References [enableADC\(\)](#), [enableSPI\(\)](#), [enableTimerCounter0\(\)](#), [enableTimerCounter1\(\)](#), [enableTimerCounter2\(\)](#), [enableTWI\(\)](#), and [enableUSART0\(\)](#).

### 9.8.2.11 selectSleepMode()

```
void core::MCU::selectSleepMode (
    const sleepMode & a_sleepMode ) [static]
```

Definition at line 15 of file [MCU.cpp](#).

```
00016 {
00017     MCU_SELECT_SLEEP_MODE(static_cast<uint8_t>(a_sleepMode));
00018 }
00019 }
```

References [MCU\\_SELECT\\_SLEEP\\_MODE](#).

### 9.8.2.12 sleepEnable()

```
void core::MCU::sleepEnable (
    const uint8_t ar_enable ) [static]
```

Definition at line 47 of file [MCU.cpp](#).

```
00048 {
00049     if (a_enable) {
00050         MCU_SLEEP_ENABLE;
00051     } else {
00052         MCU_SLEEP_DISABLE;
00053     }
00054 }
00055 }
```

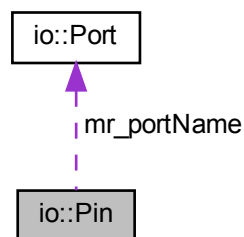
References [MCU\\_SLEEP\\_DISABLE](#), and [MCU\\_SLEEP\\_ENABLE](#).

## 9.9 io::Pin Class Reference

Class for handling a [Pin](#).

```
#include <Pin.h>
```

Collaboration diagram for io::Pin:



### Public Member Functions

- [Pin](#) (const uint8\_t a\_pinNumber, const [Port](#) &ar\_portName)
- [~Pin](#) ()
- void [toOutput](#) ()
- void [toInput](#) (const uint8\_t &ar\_useInternalPullUp)
- void [setLow](#) ()
- void [setHigh](#) ()
- void [toggle](#) ()
- uint8\_t [isHigh](#) ()
- uint8\_t [isLow](#) ()
- uint8\_t [getPinNumber](#) ()

## Private Attributes

- const [Port](#) & [mr\\_portName](#)
- const uint8\_t [m\\_pinNumber](#)

### 9.9.1 Detailed Description

Class for handling a [Pin](#).

The class implements [Pin](#) operations

#### Author

Farid Oubbati ( [farid.oubbati@outlook.com](mailto:farid.oubbati@outlook.com) )

#### Date

March 2018

Definition at line [35](#) of file [Pin.h](#).

### 9.9.2 Constructor & Destructor Documentation

#### 9.9.2.1 Pin()

```
io::Pin::Pin (
    const uint8_t a_pinNumber,
    const Port & ar_portName )
```

#### Constructor

Initializes the pin object

#### Parameters

<i>ar_portName</i>	Defines the <a href="#">Port</a> name in the AVR chip
<i>a_pinNumber</i>	Defines the <a href="#">Pin</a> number in the AVR chip

#### Author

Farid Oubbati ( [farid.oubbati@outlook.com](mailto:farid.oubbati@outlook.com) )

**Date**

March 2018

- See [Pin.h](#) for a description of this code

Definition at line 12 of file [Pin.cpp](#).

```
00013         : m_portName(ar_portName), m_pinNumber(a_pinNumber)
00014 {
00015
00016 }
```

### 9.9.2.2 ~Pin()

```
io::Pin::~~Pin ( )
```

**Destructor**

Definition at line 18 of file [Pin.cpp](#).

```
00020 {
00021
00022 }
```

## 9.9.3 Member Function Documentation

### 9.9.3.1 getPinNumber()

```
uint8_t io::Pin::getPinNumber ( )
```

Get [Pin](#) number

**Returns**

[Pin](#) number

Definition at line 80 of file [Pin.cpp](#).

```
00081 {
00082     return m_pinNumber;
00083 }
```

Referenced by [core::ADConverter::selectAnalogInput\(\)](#).

### 9.9.3.2 isHigh()

```
uint8_t io::Pin::isHigh ( )
```

Check if [Pin](#) is logic high

#### Returns

[Pin](#) status

Definition at line 70 of file [Pin.cpp](#).

```
00071 {  
00072     return *mr_portName.mp_pinReg & (1 << m_pinNumber);  
00073 }
```

### 9.9.3.3 isLow()

```
uint8_t io::Pin::isLow ( )
```

Check if [Pin](#) is logic low

#### Returns

[Pin](#) status

Definition at line 75 of file [Pin.cpp](#).

```
00076 {  
00077     return !(*mr_portName.mp_pinReg & (1 << m_pinNumber));  
00078 }
```

### 9.9.3.4 setHigh()

```
void io::Pin::setHigh ( )
```

Set [Pin](#) to logic high

Definition at line 54 of file [Pin.cpp](#).

```
00055 {  
00056     if (*mr_portName.mp_ddrReg & (1 << m_pinNumber))  
00057     {  
00058         *mr_portName.mp_portReg |= (1 << m_pinNumber);  
00059     }  
00060 }
```

### 9.9.3.5 setLow()

```
void io::Pin::setLow ( )
```

Set [Pin](#) to logic low

Definition at line 46 of file [Pin.cpp](#).

```
00047 {
00048     if (*mr_portName.mp_ddrReg & (1 « m_pinNumber))
00049     {
00050         *mr_portName.mp_portReg &= ~(1 « m_pinNumber);
00051     }
00052 }
```

### 9.9.3.6 toggle()

```
void io::Pin::toggle ( )
```

Toggle [Pin](#) state

Definition at line 62 of file [Pin.cpp](#).

```
00063 {
00064     if (*mr_portName.mp_ddrReg & (1 « m_pinNumber))
00065     {
00066         *mr_portName.mp_portReg ^= 1 « m_pinNumber;
00067     }
00068 }
```

### 9.9.3.7 toInput()

```
void io::Pin::toInput (
    const uint8_t & ar_useInternalPullUp )
```

Configures [Pin](#) to input

#### Parameters

<i>ar_useInternalPullUp</i>	Indicates if internal pull up resistor is used
-----------------------------	------------------------------------------------

Definition at line 29 of file [Pin.cpp](#).

```
00030 {
00031     if (ar_useInternalPullUp)
00032     {
00033         *mr_portName.mp_portReg |= (1 « m_pinNumber);
00034         *mr_portName.mp_ddrReg &= ~(1 « m_pinNumber);
00035     }
00036     else
00037     {
00038         *mr_portName.mp_portReg &= ~(1 « m_pinNumber);
00039         *mr_portName.mp_ddrReg &= ~(1 « m_pinNumber);
00040     }
00041 }
00042 }
00043 }
00044 }
```

Referenced by [component::LDR::LDR\(\)](#), [component::PushButton::PushButton\(\)](#), and [core::ADConverter::selectAnalogInput\(\)](#).

### 9.9.3.8 toOutput()

```
void io::Pin::toOutput ( )
```

Configures [Pin](#) to output

Definition at line 24 of file [Pin.cpp](#).

```
00025 {
00026     *mr_portName.mp_ddrReg |= (1 << m_pinNumber);
00027 }
```

Referenced by [component::Buzzer::Buzzer\(\)](#), [component::DCMotor::DCMotor\(\)](#), [component::Led::Led\(\)](#), [component::ServoMotor::ServoMotor\(\)](#) and [component::StepperMotor::StepperMotor\(\)](#).

## 9.9.4 Member Data Documentation

### 9.9.4.1 m\_pinNumber

```
const uint8_t io::Pin::m_pinNumber [private]
```

[Pin](#) number

Definition at line 93 of file [Pin.h](#).

### 9.9.4.2 mr\_portName

```
const Port& io::Pin::mr_portName [private]
```

[Port](#) object reference

Definition at line 92 of file [Pin.h](#).

## 9.10 io::Port Struct Reference

Contains definitions of pointers to the AVR port's registers.

```
#include <Pin.h>
```

### Public Attributes

- volatile uint8\_t \* [mp\\_ddrReg](#)
- volatile uint8\_t \* [mp\\_portReg](#)
- volatile uint8\_t \* [mp\\_pinReg](#)

### 9.10.1 Detailed Description

Contains definitions of pointers to the AVR port's registers.

Definition at line 19 of file [Pin.h](#).

### 9.10.2 Member Data Documentation

#### 9.10.2.1 mp\_ddrReg

```
volatile uint8_t* io::Port::mp_ddrReg
```

Pointer to the data direction register

Definition at line 21 of file [Pin.h](#).

#### 9.10.2.2 mp\_pinReg

```
volatile uint8_t* io::Port::mp_pinReg
```

Pointer to the pin register

Definition at line 25 of file [Pin.h](#).

#### 9.10.2.3 mp\_portReg

```
volatile uint8_t* io::Port::mp_portReg
```

Pointer to the port register

Definition at line 23 of file [Pin.h](#).

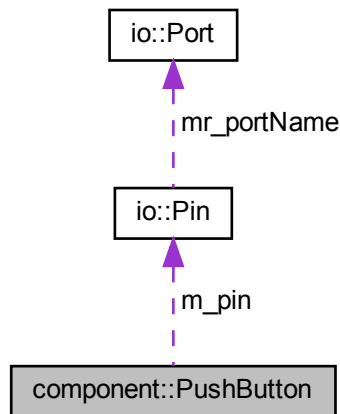


## 9.11 component::PushButton Class Reference

Class for handling a [PushButton](#) component.

```
#include <PushButton.h>
```

Collaboration diagram for component::PushButton:



### Public Member Functions

- [PushButton](#) (const [io::Pin](#) &ar\_pin, const uint8\_t &ar\_useInternalPullUp=1, const uint8\_t &ar\_isActiveLow=1)
- [~PushButton](#) ()
- uint8\_t [isPressed](#) ()
- uint8\_t [getPressedCount](#) () const
- void [resetPressedCount](#) ()

### Private Attributes

- [io::Pin](#) m\_pin
- const uint8\_t & [mr\\_isActiveLow](#)
- const uint8\_t & [mr\\_useInternalPullUp](#)
- uint8\_t [m\\_buttonPressed](#)

#### 9.11.1 Detailed Description

Class for handling a [PushButton](#) component.

The class implements [PushButton](#) operations

Author

Farid Oubbati ( [farid.oubbati@outlook.com](mailto:farid.oubbati@outlook.com) )

Date

March 2018

Definition at line 26 of file [PushButton.h](#).

## 9.11.2 Constructor & Destructor Documentation

### 9.11.2.1 PushButton()

```
component::PushButton::PushButton (
    const io::Pin & ar_pin,
    const uint8_t & ar_useInternalPullUp = 1,
    const uint8_t & ar_isActiveLow = 1 )
```

Constructor.

Initializes The [PushButton](#) object

Parameters

<i>ar_pin</i>	Defines a Pin object
<i>ar_useInternalPullUp</i>	Defines if internal pull up resistor is used
<i>ar_isActiveLow</i>	Defines if input is active low or active high

Author

Farid Oubbati ( [farid.oubbati@outlook.com](mailto:farid.oubbati@outlook.com))

Date

March 2018

- See [PushButton.h](#) for a description of this code

Definition at line 12 of file [PushButton.cpp](#).

```
00013         : m_pin(ar_pin),
00014           mr_isActiveLow(ar_isActiveLow),
00015           mr_useInternalPullUp(ar_useInternalPullUp)
00016
00017
00018 {
00019     m_pin.toInput(mr_useInternalPullUp);
00020     m_buttonPressed = 0;
00021 }
```

References [m\\_buttonPressed](#), [m\\_pin](#), [mr\\_useInternalPullUp](#), and [io::Pin::toInput\(\)](#).

### 9.11.2.2 ~PushButton()

```
component::PushButton::~~PushButton ( )
```

Destructor

Definition at line 23 of file [PushButton.cpp](#).

```
00024 {
00025
00026 }
```

### 9.11.3 Member Function Documentation

#### 9.11.3.1 getPressedCount()

```
uint8_t component::PushButton::getPressedCount ( ) const
```

Get pushbutton pressed count

##### Returns

[PushButton](#) press count

Definition at line 55 of file [PushButton.cpp](#).

```
00056 {  
00057  
00058     return m_buttonPressed;  
00059  
00060 }
```

#### 9.11.3.2 isPressed()

```
uint8_t component::PushButton::isPressed ( )
```

Is pushbutton pressed

##### Returns

[PushButton](#) status

Definition at line 29 of file [PushButton.cpp](#).

```
00030 {  
00031     if (mr_isActiveLow || mr_useInternalPullUp) {  
00032         if (m_pin.isLow()) {  
00033             _delay_us(PUSHBUTTON_DEBOUNCE_TIME_US);  
00034             if (m_pin.isLow()) {  
00035                 ++m_buttonPressed;  
00036                 return 1;  
00037             }  
00038         }  
00039     } else {  
00040  
00041         if (m_pin.isHigh()) {  
00042             _delay_us(PUSHBUTTON_DEBOUNCE_TIME_US);  
00043             if (m_pin.isHigh()) {  
00044                 ++m_buttonPressed;  
00045                 return 1;  
00046             }  
00047         }  
00048     }  
00049 }  
00050  
00051 return 0;  
00052  
00053 }
```

References [PUSHBUTTON\\_DEBOUNCE\\_TIME\\_US](#).

### 9.11.3.3 resetPressedCount()

```
void component::PushButton::resetPressedCount ( )
```

Reset pushbutton pressed count

Definition at line 62 of file [PushButton.cpp](#).

```
00063 {  
00064     m_buttonPressed = 0;  
00065  
00066 }
```

## 9.11.4 Member Data Documentation

### 9.11.4.1 m\_buttonPressed

```
uint8_t component::PushButton::m_buttonPressed [private]
```

[PushButton](#) pressed count

Definition at line 67 of file [PushButton.h](#).

Referenced by [PushButton\(\)](#).

### 9.11.4.2 m\_pin

```
io::Pin component::PushButton::m_pin [private]
```

Pin object

Definition at line 64 of file [PushButton.h](#).

Referenced by [PushButton\(\)](#).

### 9.11.4.3 mr\_isActiveLow

```
const uint8_t& component::PushButton::mr_isActiveLow [private]
```

Input active state

Definition at line 65 of file [PushButton.h](#).

#### 9.11.4.4 mr\_useInternalPullUp

```
const uint8_t& component::PushButton::mr_useInternalPullUp [private]
```

Internal pullup resistor use

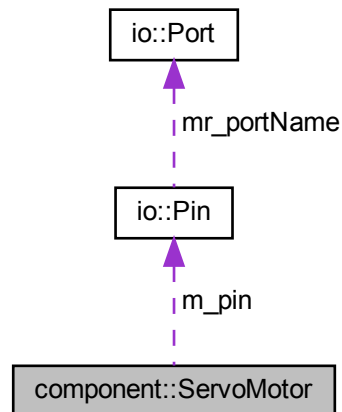
Definition at line 66 of file [PushButton.h](#).

Referenced by [PushButton\(\)](#).

## 9.12 component::ServoMotor Class Reference

```
#include <ServoMotor.h>
```

Collaboration diagram for component::ServoMotor:



### Public Member Functions

- [ServoMotor](#) (const [io::Pin](#) &ar\_pin, const uint16\_t &ar\_pulseCycle=0, const uint16\_t &ar\_pulseWidthMin=0, const uint16\_t &ar\_pulseWidthMid=0, const uint16\_t &ar\_pulseWidthMax=0)
- [~ServoMotor](#) ()
- void [on](#) ()
- void [off](#) ()
- void [toggle](#) ()
- uint16\_t [computePulseCycleCount](#) (const uint16\_t &ar\_clockPrescaler)
- uint16\_t [computePulseWidthMinCount](#) (const uint16\_t &ar\_clockPrescaler)
- uint16\_t [computePulseWidthMidCount](#) (const uint16\_t &ar\_clockPrescaler)
- uint16\_t [computePulseWidthMaxCount](#) (const uint16\_t &ar\_clockPrescaler)
- uint16\_t [computeRotationAngleCount](#) (const uint8\_t &ar\_angle\_deg, const uint16\_t &ar\_clockPrescaler)
- void [rotate](#) (core::TimerCounter1 &ar\_timerCounter1, const uint8\_t &ar\_angle\_deg, const [core::channel](#) &ar\_channel=[core::channel::A](#))
- void [connect](#) (core::TimerCounter1 &ar\_timerCounter1, const [core::channel](#) &ar\_channel=[core::channel::A](#))
- void [disconnect](#) (core::TimerCounter1 &ar\_timerCounter1, const [core::channel](#) &ar\_channel=[core::channel::A](#))

## Private Attributes

- [io::Pin](#) `m_pin`
- `uint16_t` `m_pulseCycle`
- `uint16_t` `m_pulseWidthMin`
- `uint16_t` `m_pulseWidthMid`
- `uint16_t` `m_pulseWidthMax`

### 9.12.1 Detailed Description

Definition at line 138 of file [ServoMotor.h](#).

### 9.12.2 Constructor & Destructor Documentation

#### 9.12.2.1 ServoMotor()

```
component::ServoMotor::ServoMotor (  
    const io::Pin & ar_pin,  
    const uint16_t & ar_pulseCycle = 0,  
    const uint16_t & ar_pulseWidthMin = 0,  
    const uint16_t & ar_pulseWidthMid = 0,  
    const uint16_t & ar_pulseWidthMax = 0 )
```

Definition at line 5 of file [ServoMotor.cpp](#).

```
00010         : m_pin(ar_pin),  
00011           m_pulseCycle(ar_pulseCycle),  
00012           m_pulseWidthMin(ar_pulseWidthMin),  
00013           m_pulseWidthMid(ar_pulseWidthMid),  
00014           m_pulseWidthMax(ar_pulseWidthMax)  
00015 {  
00016     m_pin.toOutput();  
00017  
00018 }
```

References [m\\_pin](#), and [io::Pin::toOutput\(\)](#).

#### 9.12.2.2 ~ServoMotor()

```
component::ServoMotor::~~ServoMotor ( )
```

Definition at line 20 of file [ServoMotor.cpp](#).

```
00021 {  
00022  
00023 }
```

### 9.12.3 Member Function Documentation

### 9.12.3.1 computePulseCycleCount()

```
uint16_t component::ServoMotor::computePulseCycleCount (
    const uint16_t & ar_clockPrescaler )
```

Definition at line 44 of file [ServoMotor.cpp](#).

```
00045 {
00046     return SERVOMOTOR_TIMER_PULSE_WIDTH_COUNT(m_pulseCycle,ar_clockPrescaler);
00047 }
```

References [SERVOMOTOR\\_TIMER\\_PULSE\\_WIDTH\\_COUNT](#).

### 9.12.3.2 computePulseWidthMaxCount()

```
uint16_t component::ServoMotor::computePulseWidthMaxCount (
    const uint16_t & ar_clockPrescaler )
```

Definition at line 54 of file [ServoMotor.cpp](#).

```
00055 {
00056     return SERVOMOTOR_TIMER_PULSE_WIDTH_COUNT(m_pulseWidthMax,ar_clockPrescaler);
00057 }
```

References [SERVOMOTOR\\_TIMER\\_PULSE\\_WIDTH\\_COUNT](#).

### 9.12.3.3 computePulseWidthMidCount()

```
uint16_t component::ServoMotor::computePulseWidthMidCount (
    const uint16_t & ar_clockPrescaler )
```

Definition at line 59 of file [ServoMotor.cpp](#).

```
00060 {
00061     return SERVOMOTOR_TIMER_PULSE_WIDTH_COUNT(m_pulseWidthMid,ar_clockPrescaler);
00062 }
```

References [SERVOMOTOR\\_TIMER\\_PULSE\\_WIDTH\\_COUNT](#).

### 9.12.3.4 computePulseWidthMinCount()

```
uint16_t component::ServoMotor::computePulseWidthMinCount (
    const uint16_t & ar_clockPrescaler )
```

Definition at line 49 of file [ServoMotor.cpp](#).

```
00050 {
00051     return SERVOMOTOR_TIMER_PULSE_WIDTH_COUNT(m_pulseWidthMin,ar_clockPrescaler);
00052 }
```

References [SERVOMOTOR\\_TIMER\\_PULSE\\_WIDTH\\_COUNT](#).

### 9.12.3.5 computeRotationAngleCount()

```
uint16_t component::ServoMotor::computeRotationAngleCount (
    const uint8_t & ar_angle_deg,
    const uint16_t & ar_clockPrescaler )
```

Definition at line 64 of file [ServoMotor.cpp](#).

```
00065 {
00066     return
00067     static_cast<uint16_t> (SERVOMOTOR_TIMER_ANGLE_COUNT (ar_angle_deg, static_cast<long> (computePulseWidthMinCount (ar_clockPr
```

References [SERVOMOTOR\\_TIMER\\_ANGLE\\_COUNT](#).

### 9.12.3.6 connect()

```
void component::ServoMotor::connect (
    core::TimerCounter1 & ar_timerCounter1,
    const core::channel & ar_channel = core::channel::A )
```

Definition at line 85 of file [ServoMotor.cpp](#).

```
00087 {
00088
00089
00090     ar_timerCounter1.setInputCaptureRegister (computePulseCycleCount (ar_timerCounter1.getClockPrescaler ()));
00091     ar_timerCounter1.selectOperationMode (core::operationMode::fast_PWM_ICR);
00092     ar_timerCounter1.selectCompareOutputMode (ar_channel, core::compareOutputMode::clear);
00093     ar_timerCounter1.setCounter (0);
00094
00095 }
```

References [core::clear](#), [core::fast\\_PWM\\_ICR](#), [core::TimerCounter1::getClockPrescaler\(\)](#), [core::TimerCounter1::selectCompareOutputMode\(\)](#), [core::TimerCounter1::selectOperationMode\(\)](#), [core::TimerCounter1::setCounter\(\)](#), and [core::TimerCounter1::setInputCaptureRegister\(\)](#).

### 9.12.3.7 disconnect()

```
void component::ServoMotor::disconnect (
    core::TimerCounter1 & ar_timerCounter1,
    const core::channel & ar_channel = core::channel::A )
```

Definition at line 97 of file [ServoMotor.cpp](#).

```
00099 {
00100     ar_timerCounter1.selectCompareOutputMode (ar_channel, core::compareOutputMode::normal);
00101     // stop timer
00102     ar_timerCounter1.stop();
00103 }
```

References [core::normal](#), [core::TimerCounter1::selectCompareOutputMode\(\)](#), and [core::TimerCounter1::stop\(\)](#).



### 9.12.3.8 off()

```
void component::ServoMotor::off ( )
```

Turn servo motor Off.

Definition at line 33 of file [ServoMotor.cpp](#).

```
00034 {  
00035     m_pin.setLow();  
00036 }
```

### 9.12.3.9 on()

```
void component::ServoMotor::on ( )
```

Turn servo motor On.

Definition at line 28 of file [ServoMotor.cpp](#).

```
00029 {  
00030     m_pin.setHigh();  
00031 }
```

### 9.12.3.10 rotate()

```
void component::ServoMotor::rotate (  
    core::TimerCounter1 & ar_timerCounter1,  
    const uint8_t & ar_angle_deg,  
    const core::channel & ar_channel = core::channel::A )
```

Definition at line 71 of file [ServoMotor.cpp](#).

```
00074 {  
00075  
00076     ar_timerCounter1.setOutputCompareRegister(ar_channel,  
        computeRotationAngleCount(ar_angle_deg, ar_timerCounter1.getClockPrescaler()));  
00077  
00078     // start timer  
00079     ar_timerCounter1.start();  
00080  
00081  
00082 }
```

References [core::TimerCounter1::getClockPrescaler\(\)](#), [core::TimerCounter1::setOutputCompareRegister\(\)](#), and [core::TimerCounter1::start\(\)](#).

### 9.12.3.11 toggle()

```
void component::ServoMotor::toggle ( )
```

Toggle servo motor state.

Definition at line 38 of file [ServoMotor.cpp](#).

```
00039 {  
00040     m_pin.toggle();  
00041  
00042 }
```

## 9.12.4 Member Data Documentation

### 9.12.4.1 m\_pin

```
io::Pin component::ServoMotor::m_pin [private]
```

pin object

Definition at line 188 of file [ServoMotor.h](#).

Referenced by [ServoMotor\(\)](#).

### 9.12.4.2 m\_pulseCycle

```
uint16_t component::ServoMotor::m_pulseCycle [private]
```

pulse cycle [us]

Definition at line 190 of file [ServoMotor.h](#).

### 9.12.4.3 m\_pulseWidthMax

```
uint16_t component::ServoMotor::m_pulseWidthMax [private]
```

pulse width max [us]

Definition at line 196 of file [ServoMotor.h](#).

### 9.12.4.4 m\_pulseWidthMid

```
uint16_t component::ServoMotor::m_pulseWidthMid [private]
```

pulse width mid [us]

Definition at line 194 of file [ServoMotor.h](#).

## 9.12.4.5 m\_pulseWidthMin

```
uint16_t component::ServoMotor::m_pulseWidthMin [private]
```

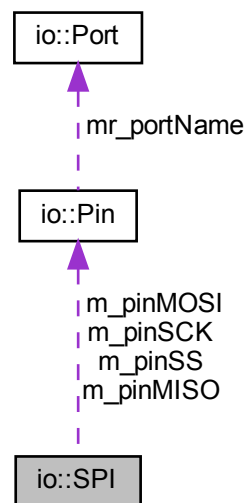
pulse width min [us]

Definition at line 192 of file [ServoMotor.h](#).

## 9.13 io::SPI Class Reference

```
#include <SPI.h>
```

Collaboration diagram for io::SPI:



## Public Member Functions

- void [selectDataMode](#) (const [dataMode](#) &ar\_dataMode)
- void [selectDataOrder](#) (const [dataOrder](#) &ar\_dataOrder)
- void [selectOperationMode](#) (const [operationMode](#) &ar\_operationMode)
- void [selectClockPrescaler](#) (const [clockPrescaler](#) &ar\_clockPrescaler)
- void [selectSlave](#) (const uint8\_t a\_select)
- uint8\_t [writeCollision](#) ()
- uint8\_t [transferComplete](#) ()
- void [masterSendByte](#) (const uint8\_t &ar\_byte)
- void [masterReceiveByte](#) (uint8\_t &ar\_byte)
- void [slaveReceiveByte](#) (uint8\_t &ar\_byte)

## Static Public Member Functions

- static [SPI](#) & [getInstance](#) (const [io::Pin](#) &ar\_pinSCK, const [io::Pin](#) &ar\_pinMISO, const [io::Pin](#) &ar\_pinMOSI, const [io::Pin](#) &ar\_pinSS)
- static void [enableTransferCompleteInterrupt](#) (const uint8\_t a\_enable)
- static void [transferCompleteServiceRoutine](#) () \_\_asm\_\_(STR(SPI\_TRANSFER\_COMPLETE\_INTERRUPT))  
\_\_attribute\_\_((\_\_signal\_\_

## Public Attributes

- static void [\\_\\_used\\_\\_](#)
- static void [\\_\\_externally\\_visible\\_\\_](#)

## Private Member Functions

- [SPI](#) (const [io::Pin](#) &ar\_pinSCK, const [io::Pin](#) &ar\_pinMISO, const [io::Pin](#) &ar\_pinMOSI, const [io::Pin](#) &ar\_pinSS)
- [~SPI](#) ()
- [SPI](#) (const [SPI](#) &)
- const [SPI](#) & [operator=](#) (const [SPI](#) &)

## Private Attributes

- [io::Pin](#) m\_pinSCK
- [io::Pin](#) m\_pinMISO
- [io::Pin](#) m\_pinMOSI
- [io::Pin](#) m\_pinSS

## Static Private Attributes

- static volatile uint8\_t [m\\_data](#) = 0

### 9.13.1 Detailed Description

Definition at line 86 of file [SPI.h](#).

### 9.13.2 Constructor & Destructor Documentation

### 9.13.2.1 SPI() [1/2]

```
io::SPI::SPI (
    const io::Pin & ar_pinSCK,
    const io::Pin & ar_pinMISO,
    const io::Pin & ar_pinMOSI,
    const io::Pin & ar_pinSS ) [private]
```

Constructor. Initializes the [USART0](#) object

@param ar\_transMode defines transmission mode  
 @param ar\_comMode defines communication mode  
 @param ar\_frameSize defines data frame size  
 @param ar\_stopBit defines number of stop bits  
 @param ar\_parityMode defines parity mode

Definition at line 22 of file [SPI.cpp](#).

```
00026         : m_pinSCK(ar_pinSCK),
00027           m_pinMISO(ar_pinMISO),
00028           m_pinMOSI(ar_pinMOSI),
00029           m_pinSS(ar_pinSS)
00030 {
00031     core::MCU::enableSPI(1);
00032     sei();
00033     enableTransferCompleteInterrupt(1);
00034 }
00035 }
```

References [core::MCU::enableSPI\(\)](#), and [enableTransferCompleteInterrupt\(\)](#).

### 9.13.2.2 ~SPI()

```
io::SPI::~~SPI ( ) [private]
```

Destructor.

Definition at line 37 of file [SPI.cpp](#).

```
00038 {
00039 }
00040 }
```

### 9.13.2.3 SPI() [2/2]

```
io::SPI::SPI (
    const SPI & ) [private]
```

Overried Copy constructor.

## 9.13.3 Member Function Documentation

### 9.13.3.1 enableTransferCompleteInterrupt()

```
void io::SPI::enableTransferCompleteInterrupt (
    const uint8_t a_enable ) [static]
```

Definition at line 110 of file [SPI.cpp](#).

```
00111 {
00112     if (a_enable) {
00113         SPI_ENABLE_TRANSFER_COMPLETE_INTERRUPT;
00114     } else {
00115         SPI_DISABLE_TRANSFER_COMPLETE_INTERRUPT;
00116     }
00117 }
```

References [SPI\\_DISABLE\\_TRANSFER\\_COMPLETE\\_INTERRUPT](#), and [SPI\\_ENABLE\\_TRANSFER\\_COMPLETE\\_INTERRUPT](#).

Referenced by [SPI\(\)](#).

### 9.13.3.2 getInstance()

```
io::SPI & io::SPI::getInstance (
    const io::Pin & ar_pinSCK,
    const io::Pin & ar_pinMISO,
    const io::Pin & ar_pinMOSI,
    const io::Pin & ar_pinSS ) [static]
```

Definition at line 8 of file [SPI.cpp](#).

```
00012 {
00013     static SPI l_instance(ar_pinSCK,
00014                          ar_pinMISO,
00015                          ar_pinMOSI,
00016                          ar_pinSS);
00017
00018     return l_instance;
00019
00020 }
```

### 9.13.3.3 masterReceiveByte()

```
void io::SPI::masterReceiveByte (
    uint8_t & ar_byte )
```

Definition at line 147 of file [SPI.cpp](#).

```
00148 {
00149     masterSendByte(0);
00150     ar_byte = m_data;
00151
00152 }
```

#### 9.13.3.4 masterSendByte()

```
void io::SPI::masterSendByte (
    const uint8_t & ar_byte )
```

Definition at line 135 of file [SPI.cpp](#).

```
00136 {
00137
00138     SPI_DATA_REGISTER = ar_byte;
00139     selectSlave(1);
00140     while(!transferComplete()){};
00141     selectSlave(0);
00142
00143
00144
00145 }
```

References [SPI\\_DATA\\_REGISTER](#).

#### 9.13.3.5 operator=()

```
const SPI& io::SPI::operator= (
    const SPI & ) [private]
```

Override assign operator.

#### 9.13.3.6 selectClockPrescaler()

```
void io::SPI::selectClockPrescaler (
    const clockPrescaler & ar_clockPrescaler )
```

Definition at line 53 of file [SPI.cpp](#).

```
00054 {
00055     SPI_SELECT_CLOCK_PRESCALER(static_cast<uint8_t>(ar_clockPrescaler));
00056
00057 }
```

References [SPI\\_SELECT\\_CLOCK\\_PRESCALER](#).

#### 9.13.3.7 selectDataMode()

```
void io::SPI::selectDataMode (
    const dataMode & ar_dataMode )
```

Definition at line 99 of file [SPI.cpp](#).

```
00100 {
00101     SPI_SELECT_DATA_MODE(static_cast<uint8_t>(ar_dataMode));
00102 }
```

References [SPI\\_SELECT\\_DATA\\_MODE](#).

### 9.13.3.8 selectDataOrder()

```
void io::SPI::selectDataOrder (
    const dataOrder & ar_dataOrder )
```

Definition at line 104 of file [SPI.cpp](#).

```
00105 {
00106     SPI\_SELECT\_DATA\_ORDER(static_cast<uint8_t>(ar_dataOrder));
00107 }
00108 }
```

References [SPI\\_SELECT\\_DATA\\_ORDER](#).

### 9.13.3.9 selectOperationMode()

```
void io::SPI::selectOperationMode (
    const operationMode & ar_operationMode )
```

Definition at line 59 of file [SPI.cpp](#).

```
00060 {
00061     switch (ar_operationMode)
00062     {
00063         case operationMode::master:
00064         {
00065             m_pinMOSI.toOutput();
00066             m_pinSCK.toOutput();
00067             m_pinMISO.toInput(1);
00068             m_pinSS.toOutput();
00069             m_pinSS.setHigh();
00070             SPI\_SELECT\_MASTER\_MODE;
00071             SPI\_ENABLE;
00072             break;
00073         }
00074         case operationMode::slave:
00075         {
00076             m_pinMISO.toOutput();
00077             SPI\_SELECT\_SLAVE\_MODE;
00078             SPI\_ENABLE;
00079             break;
00080         }
00081         case operationMode::submaster:
00082         {
00083             m_pinMOSI.toOutput();
00084             m_pinSCK.toOutput();
00085             m_pinMISO.toInput(1);
00086             m_pinSS.toInput(1);
00087             SPI\_SELECT\_MASTER\_MODE;
00088             SPI\_ENABLE;
00089             break;
00090         }
00091         case operationMode::disable:
00092         {
00093             SPI\_DISABLE;
00094             break;
00095         }
00096     }
00097 }
```

References [io::disable](#), [io::master](#), [io::slave](#), [SPI\\_DISABLE](#), [SPI\\_ENABLE](#), [SPI\\_SELECT\\_MASTER\\_MODE](#), [SPI\\_SELECT\\_SLAVE\\_MODE](#), and [io::submaster](#).



#### 9.13.3.10 selectSlave()

```
void io::SPI::selectSlave (
    const uint8_t a_select )
```

Definition at line 119 of file [SPI.cpp](#).

```
00120 {
00121     if (a_select) {
00122         m_pinSS.setLow();
00123     } else {
00124         m_pinSS.setHigh();
00125     }
00126 }
00127
00128 }
```

#### 9.13.3.11 slaveReceiveByte()

```
void io::SPI::slaveReceiveByte (
    uint8_t & ar_byte )
```

Definition at line 154 of file [SPI.cpp](#).

```
00155 {
00156     ar_byte = m_data;
00157 }
00158 }
```

#### 9.13.3.12 transferComplete()

```
uint8_t io::SPI::transferComplete ( )
```

Is serial transfer complete.

Definition at line 48 of file [SPI.cpp](#).

```
00049 {
00050     return (SPI_STATUS_REGISTER & (1 << SPI_TRANSFER_COMPLETE));
00051 }
```

References [SPI\\_STATUS\\_REGISTER](#), and [SPI\\_TRANSFER\\_COMPLETE](#).

#### 9.13.3.13 transferCompleteServiceRoutine()

```
void io::SPI::transferCompleteServiceRoutine ( ) [static]
```

Serial transfer complete ISR.

Definition at line 129 of file [SPI.cpp](#).

```
00130 {
00131
00132     m_data = SPI_DATA_REGISTER;
00133 }
```

References [SPI\\_DATA\\_REGISTER](#).

#### 9.13.3.14 writeCollision()

```
uint8_t io::SPI::writeCollision ( )
```

Is there write collision.

Definition at line 42 of file [SPI.cpp](#).

```
00043 {  
00044     return (SPI_STATUS_REGISTER & (1 « SPI_WRITE_COLLISION));  
00045 }
```

References [SPI\\_STATUS\\_REGISTER](#), and [SPI\\_WRITE\\_COLLISION](#).

### 9.13.4 Member Data Documentation

#### 9.13.4.1 \_\_externally\_visible\_\_

```
void io::SPI::__externally_visible__
```

Definition at line 124 of file [SPI.h](#).

#### 9.13.4.2 \_\_used\_\_

```
void io::SPI::__used__
```

Definition at line 124 of file [SPI.h](#).

#### 9.13.4.3 m\_data

```
volatile uint8_t io::SPI::m_data = 0 [static], [private]
```

Definition at line 154 of file [SPI.h](#).

#### 9.13.4.4 m\_pinMISO

```
io::Pin io::SPI::m_pinMISO [private]
```

pin object

Definition at line 157 of file [SPI.h](#).

#### 9.13.4.5 m\_pinMOSI

```
io::Pin io::SPI::m_pinMOSI [private]
```

pin object

Definition at line 158 of file [SPI.h](#).

#### 9.13.4.6 m\_pinSCK

```
io::Pin io::SPI::m_pinSCK [private]
```

pin object

Definition at line 156 of file [SPI.h](#).

#### 9.13.4.7 m\_pinSS

```
io::Pin io::SPI::m_pinSS [private]
```

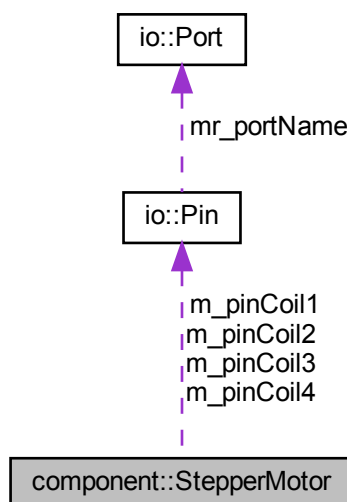
pin object

Definition at line 159 of file [SPI.h](#).

## 9.14 component::StepperMotor Class Reference

```
#include <StepperMotor.h>
```

Collaboration diagram for component::StepperMotor:



## Public Member Functions

- [StepperMotor](#) (const [mode](#) &ar\_mode, const [io::Pin](#) &ar\_pinCoil1, const [io::Pin](#) &ar\_pinCoil2, const [io::Pin](#) &ar\_pinCoil3, const [io::Pin](#) &ar\_pinCoil4)
- [~StepperMotor](#) ()
- void [step](#) (const int16\_t a\_step, const uint16\_t a\_speed)
- void [step](#) (const int16\_t a\_step, const uint16\_t a\_speed, const float a\_stepAngle)
- void [step](#) (const int16\_t a\_step, const uint16\_t a\_speed, const uint16\_t a\_accel, const uint16\_t a\_decel)
- void [stepPulse](#) (const uint8\_t a\_stepPulse)
- void [stepDelay](#) (uint8\_t a\_stepDelay)
- uint8\_t [goalReached](#) ()
- void [setCurrentPos](#) (uint16\_t a\_currentPos)
- uint16\_t [currentPos](#) ()
- uint8\_t [computeStepDelay](#) (uint16\_t a\_step, const uint16\_t a\_speed, const uint16\_t a\_accel, const uint16\_t a\_decel)

## Public Attributes

- uint16\_t [m\\_accelTime](#)
- uint16\_t [m\\_decelTime](#)
- uint16\_t [m\\_constSpeedTime](#)

## Private Attributes

- [io::Pin](#) [m\\_pinCoil1](#)
- [io::Pin](#) [m\\_pinCoil2](#)
- [io::Pin](#) [m\\_pinCoil3](#)
- [io::Pin](#) [m\\_pinCoil4](#)
- [mode](#) [stepMode](#)
- uint8\_t [m\\_goalReached](#)
- uint16\_t [m\\_currentPos](#)

### 9.14.1 Detailed Description

Definition at line 75 of file [StepperMotor.h](#).

### 9.14.2 Constructor & Destructor Documentation

### 9.14.2.1 StepperMotor()

```
component::StepperMotor::StepperMotor (
    const mode & ar_mode,
    const io::Pin & ar_pinCoil1,
    const io::Pin & ar_pinCoil2,
    const io::Pin & ar_pinCoil3,
    const io::Pin & ar_pinCoil4 )
```

Definition at line 8 of file [StepperMotor.cpp](#).

```
00013 : m_pinCoil1(ar_pinCoil1),
00014     m_pinCoil2(ar_pinCoil2),
00015     m_pinCoil3(ar_pinCoil3),
00016     m_pinCoil4(ar_pinCoil4),
00017     stepMode(ar_mode)
00018 {
00019
00020     m_pinCoil1.toOutput();
00021     m_pinCoil2.toOutput();
00022     m_pinCoil3.toOutput();
00023     m_pinCoil4.toOutput();
00024     m_goalReached = 0;
00025     m_currentPos = 0;
00026
00027
00028 }
```

References [m\\_currentPos](#), [m\\_goalReached](#), [m\\_pinCoil1](#), [m\\_pinCoil2](#), [m\\_pinCoil3](#), [m\\_pinCoil4](#), and [io::Pin::toOutput\(\)](#).

### 9.14.2.2 ~StepperMotor()

```
component::StepperMotor::~~StepperMotor ( )
```

Definition at line 30 of file [StepperMotor.cpp](#).

```
00031 {
00032
00033 }
```

## 9.14.3 Member Function Documentation

### 9.14.3.1 computeStepDelay()

```
uint8_t component::StepperMotor::computeStepDelay (
    uint16_t a_step,
    const uint16_t a_speed,
    const uint16_t a_accel,
    const uint16_t a_decel )
```

Definition at line 127 of file [StepperMotor.cpp](#).

```
00131 {
00132
00133
00134     uint32_t l_accelTime= (1000UL*a_speed/a_accel);
00135     uint16_t l_decelTime = (1000UL*a_speed/a_decel);
00136     int16_t l_constSpeedTime=((1000UL*a_step/a_speed)-(l_accelTime/2)-(l_decelTime/2));
00137     static uint32_t l_time=0;
00138     static uint32_t l_speed=0;
00139     static uint64_t l_speed_time_product=0;
```

```

00140     static uint64_t l_current_position=0;
00141
00142
00143
00144
00145
00146
00147
00148     if (l_constSpeedTime<0)
00149     {
00150         l_constSpeedTime=0;
00151         l_accelTime = static_cast<uint32_t>(1000UL*sqrtf(a_step/a_accel));
00152         a_speed=l_accelTime*a_accel/1000UL;
00153         l_decelTime = 1000UL*a_speed/a_decel;
00154
00155     }
00156
00157
00158
00159
00160     if (l_time<=l_accelTime)
00161     {
00162         l_speed=a_accel*l_time;
00163
00164     }
00165     else if ((l_time>l_accelTime) && (l_time<=(l_accelTime+l_constSpeedTime)))
00166     {
00167         l_speed=1000UL*a_speed;
00168
00169     }
00170     else if
00171     ((l_time>(l_accelTime+l_constSpeedTime)) && (l_time<=l_accelTime+l_constSpeedTime+l_decelTime))
00172     {
00173         l_speed= (1000UL*a_speed)-a_decel*(l_time-l_accelTime-l_constSpeedTime);
00174
00175     }
00176
00177
00178     l_time=l_time+1;
00179
00180
00181     l_speed_time_product = l_speed_time_product+l_speed;
00182
00183     if (l_speed_time_product - l_current_position >= 1000000UL)
00184     {
00185         l_current_position=l_current_position+1000000UL;
00186         return 1;
00187     }
00188     else
00189     {
00190         stepDelay(1);
00191         return 0;
00192     }
00193
00194
00195
00196
00197
00198 }

```

### 9.14.3.2 currentPos()

uint16\_t component::StepperMotor::currentPos ( )

Definition at line 40 of file [StepperMotor.cpp](#).

```

00041 {
00042     return m_currentPos;
00043 }

```

### 9.14.3.3 goalReached()

```
uint8_t component::StepperMotor::goalReached ( )
```

Definition at line 403 of file [StepperMotor.cpp](#).

```
00404 {
00405     return m_goalReached;
00406 }
00407 }
```

### 9.14.3.4 setCurrentPos()

```
void component::StepperMotor::setCurrentPos (
    uint16_t a_currentPos )
```

Definition at line 35 of file [StepperMotor.cpp](#).

```
00036 {
00037     m_currentPos = a_currentPos;
00038 }
```

### 9.14.3.5 step() [1/3]

```
void component::StepperMotor::step (
    const int16_t a_step,
    const uint16_t a_speed )
```

Turn servo motor On.

Definition at line 201 of file [StepperMotor.cpp](#).

```
00202 {
00203     static int16_t l_stepNumber = 0;
00204     static uint8_t l_stepDelay_ms = static_cast<uint8_t>(1000UL/a_speed);
00205
00206     if (a_step<=0)
00207     {
00208         if (l_stepNumber== -a_step)
00209         {
00210             m_goalReached = 1;
00211         } else {
00212             l_stepNumber++;
00213             m_currentPos++;
00214         }
00215     }
00216
00217     } else {
00218
00219         if (l_stepNumber == -a_step)
00220         {
00221             m_goalReached = 1;
00222         } else {
00223             l_stepNumber--;
00224             m_currentPos--;
00225         }
00226     }
00227
00228     }
00229
00230
00231
00232     if (m_goalReached == 0)
00233     {
00234
00235         switch (stepMode)
00236         {
00237             case mode::fullStep:
```

```

00238         {
00239             // equivalent to l_stepNumber % 4
00240             // 1 3 5 7 <->
00241             stepPulse(2*(l_stepNumber & 3)+1);
00242             break;
00243         }
00244
00245         case mode::halfStep:
00246         {
00247             // equivalent to l_stepNumber % 8
00248             // 0 1 2 3 4 5 6 7 <->
00249             stepPulse(l_stepNumber & 7);
00250             break;
00251         }
00252     }
00253
00254     stepDelay(l_stepDelay_ms);
00255
00256 }
00257 }

```

References [component::fullStep](#), and [component::halfStep](#).

### 9.14.3.6 step() [2/3]

```

void component::StepperMotor::step (
    const int16_t a_step,
    const uint16_t a_speed,
    const float a_stepAngle )

```

Definition at line 259 of file [StepperMotor.cpp](#).

```

00260 {
00261     static int16_t l_stepNumber = 0;
00262     static uint8_t l_stepDelay_ms = static_cast<uint8_t>(1000UL*a_stepAngle/a_speed);
00263
00264     if (a_step<=0)
00265     {
00266         if (l_stepNumber== -a_step)
00267         {
00268             m_goalReached = 1;
00269         } else {
00270             l_stepNumber++;
00271             m_currentPos++;
00272         }
00273     }
00274
00275     } else {
00276         if (l_stepNumber == -a_step)
00277         {
00278             m_goalReached = 1;
00279         } else {
00280             l_stepNumber--;
00281             m_currentPos--;
00282         }
00283     }
00284
00285     if (m_goalReached == 0)
00286     {
00287         switch (stepMode)
00288         {
00289             case mode::fullStep:
00290             {
00291                 // equivalent to l_stepNumber % 4
00292                 // 1 3 5 7 <->
00293                 stepPulse(2*(l_stepNumber & 3)+1);
00294                 break;
00295             }
00296
00297             case mode::halfStep:
00298             {
00299
00300
00301
00302
00303
00304

```



```

00305             // equivalent to l_stepNumber % 8
00306             // 0 1 2 3 4 5 6 7 <->
00307             stepPulse(l_stepNumber & 7);
00308             break;
00309         }
00310     }
00311
00312     stepDelay(l_stepDelay_ms);
00313
00314 }
00315
00316
00317 }

```

References [component::fullStep](#), and [component::halfStep](#).

### 9.14.3.7 step() [3/3]

```

void component::StepperMotor::step (
    const int16_t a_step,
    const uint16_t a_speed,
    const uint16_t a_accel,
    const uint16_t a_decel )

```

Definition at line 47 of file [StepperMotor.cpp](#).

```

00051 {
00052     static int16_t l_stepNumber = 0;
00053     uint8_t l_stepDelay_ms = 0;
00054     uint16_t l_step=0;
00055
00056     if (a_step<=0)
00057     {
00058         l_step = -a_step;
00059     } else
00060     {
00061         l_step = a_step;
00062     }
00063
00064     l_stepDelay_ms = computeStepDelay(l_step,
00065                                     a_speed,
00066                                     a_accel,
00067                                     a_decel);
00068
00069     if (l_stepDelay_ms>0)
00070     {
00071
00072         if (a_step<=0)
00073         {
00074             if (l_stepNumber== -a_step)
00075             {
00076                 m_goalReached = 1;
00077             } else {
00078                 l_stepNumber++;
00079                 m_currentPos++;
00080             }
00081         }
00082
00083     } else {
00084
00085         if (l_stepNumber == -a_step)
00086         {
00087             m_goalReached = 1;
00088         } else {
00089             l_stepNumber--;
00090             m_currentPos--;
00091         }
00092     }
00093
00094 }
00095
00096
00097
00098     if (m_goalReached == 0)
00099     {
00100
00101         switch (stepMode)

```

```

00102     {
00103         case mode::fullStep:
00104         {
00105             // equivalent to l_stepNumber % 4
00106             // 1 3 5 7 <->
00107             stepPulse(2*(l_stepNumber & 3)+1);
00108             break;
00109         }
00110
00111         case mode::halfStep:
00112         {
00113             // equivalent to l_stepNumber % 8
00114             // 0 1 2 3 4 5 6 7 <->
00115             stepPulse(l_stepNumber & 7);
00116             break;
00117         }
00118     }
00119     stepDelay(l_stepDelay_ms);
00120 }
00121 }
00122 }
00123 }
00124 }
00125 }

```

References [component::fullStep](#), and [component::halfStep](#).

#### 9.14.3.8 stepDelay()

```

void component::StepperMotor::stepDelay (
    uint8_t a_stepDelay )

```

Definition at line 395 of file [StepperMotor.cpp](#).

```

00396 {
00397     while(a_stepDelay-->0)
00398     {
00399         _delay_ms(1);
00400     }
00401 }

```

#### 9.14.3.9 stepPulse()

```

void component::StepperMotor::stepPulse (
    const uint8_t a_stepPulse )

```

Definition at line 320 of file [StepperMotor.cpp](#).

```

00321 {
00322
00323     switch (a_stepPulse) {
00324     case 0: // 1000
00325     {
00326         m_pinCoil1.setHigh();
00327         m_pinCoil2.setLow();
00328         m_pinCoil3.setLow();
00329         m_pinCoil4.setLow();
00330         break;
00331     }
00332     case 1: // 1100
00333     {
00334         m_pinCoil1.setHigh();
00335         m_pinCoil2.setHigh();
00336         m_pinCoil3.setLow();
00337         m_pinCoil4.setLow();
00338         break;
00339     }
00340     case 2: // 0100
00341     {
00342         m_pinCoil1.setLow();

```

```

00343         m_pinCoil2.setHigh();
00344         m_pinCoil3.setLow();
00345         m_pinCoil4.setLow();
00346         break;
00347     }
00348     case 3:        //0110
00349     {
00350         m_pinCoil1.setLow();
00351         m_pinCoil2.setHigh();
00352         m_pinCoil3.setHigh();
00353         m_pinCoil4.setLow();
00354         break;
00355     }
00356     case 4:        //0010
00357     {
00358         m_pinCoil1.setLow();
00359         m_pinCoil2.setLow();
00360         m_pinCoil3.setHigh();
00361         m_pinCoil4.setLow();
00362         break;
00363     }
00364     case 5:        //0011
00365     {
00366         m_pinCoil1.setLow();
00367         m_pinCoil2.setLow();
00368         m_pinCoil3.setHigh();
00369         m_pinCoil4.setHigh();
00370         break;
00371     }
00372     case 6:        //0001
00373     {
00374         m_pinCoil1.setLow();
00375         m_pinCoil2.setLow();
00376         m_pinCoil3.setLow();
00377         m_pinCoil4.setHigh();
00378         break;
00379     }
00380     case 7:        //1001
00381     {
00382         m_pinCoil1.setHigh();
00383         m_pinCoil2.setLow();
00384         m_pinCoil3.setLow();
00385         m_pinCoil4.setHigh();
00386         break;
00387     }
00388 }
00389
00390
00391 }

```

## 9.14.4 Member Data Documentation

### 9.14.4.1 m\_accelTime

uint16\_t component::StepperMotor::m\_accelTime

pulse delay in ms

Definition at line 118 of file [StepperMotor.h](#).

### 9.14.4.2 m\_constSpeedTime

uint16\_t component::StepperMotor::m\_constSpeedTime

pulse delay in ms

Definition at line 120 of file [StepperMotor.h](#).

#### 9.14.4.3 m\_currentPos

`uint16_t component::StepperMotor::m_currentPos [private]`

pulse delay in ms

Definition at line 139 of file [StepperMotor.h](#).

Referenced by [StepperMotor\(\)](#).

#### 9.14.4.4 m\_decelTime

`uint16_t component::StepperMotor::m_decelTime`

pulse delay in ms

Definition at line 119 of file [StepperMotor.h](#).

#### 9.14.4.5 m\_goalReached

`uint8_t component::StepperMotor::m_goalReached [private]`

Definition at line 138 of file [StepperMotor.h](#).

Referenced by [StepperMotor\(\)](#).

#### 9.14.4.6 m\_pinCoil1

`io::Pin component::StepperMotor::m_pinCoil1 [private]`

pin object

Definition at line 132 of file [StepperMotor.h](#).

Referenced by [StepperMotor\(\)](#).

#### 9.14.4.7 m\_pinCoil2

`io::Pin component::StepperMotor::m_pinCoil2 [private]`

pin object

Definition at line 133 of file [StepperMotor.h](#).

Referenced by [StepperMotor\(\)](#).

#### 9.14.4.8 m\_pinCoil3

`io::Pin` component::StepperMotor::m\_pinCoil3 [private]

pin object

Definition at line 134 of file [StepperMotor.h](#).

Referenced by [StepperMotor\(\)](#).

#### 9.14.4.9 m\_pinCoil4

`io::Pin` component::StepperMotor::m\_pinCoil4 [private]

pin object

Definition at line 135 of file [StepperMotor.h](#).

Referenced by [StepperMotor\(\)](#).

#### 9.14.4.10 stepMode

`mode` component::StepperMotor::stepMode [private]

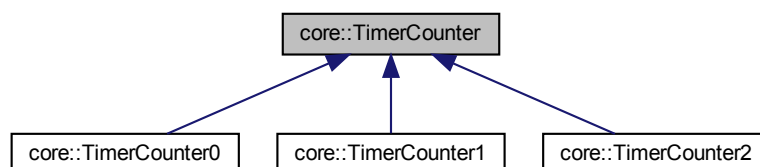
steps per revolution

Definition at line 137 of file [StepperMotor.h](#).

## 9.15 core::TimerCounter Class Reference

```
#include <TimerCounter.h>
```

Inheritance diagram for core::TimerCounter:



## Public Member Functions

- virtual void [selectOperationMode](#) (const [operationMode](#) &ar\_operationMode)=0
- virtual void [start](#) ()=0
- virtual void [stop](#) ()=0
- virtual void [selectClockSource](#) (const [clockSource](#) &ar\_clockSource)=0
- virtual void [selectCompareOutputMode](#) (const [channel](#) &ar\_channel, const [compareOutputMode](#) &ar\_↔compareOutputMode)=0
- virtual void [setCounter](#) (const uint16\_t &ar\_dataBuffer)=0
- virtual uint16\_t [getCounter](#) () const =0
- virtual void [setOutputCompareRegister](#) (const [channel](#) &ar\_channel, const uint16\_t &ar\_dataBuffer)=0
- virtual uint16\_t [getOutputCompareRegister](#) (const [channel](#) &ar\_channel) const =0
- virtual void [enableOutputCompareMatchInterrupt](#) (const [channel](#) &ar\_channel, const uint8\_t a\_enable)=0
- virtual void [enableOverflowInterrupt](#) (const uint8\_t a\_enable)=0
- virtual uint16\_t [getClockPrescaler](#) ()=0

### 9.15.1 Detailed Description

Definition at line 517 of file [TimerCounter.h](#).

### 9.15.2 Member Function Documentation

#### 9.15.2.1 [enableOutputCompareMatchInterrupt\(\)](#)

```
virtual void core::TimerCounter::enableOutputCompareMatchInterrupt (
    const channel & ar_channel,
    const uint8_t a_enable ) [pure virtual]
```

Implemented in [core::TimerCounter1](#), [core::TimerCounter0](#), and [core::TimerCounter2](#).

#### 9.15.2.2 [enableOverflowInterrupt\(\)](#)

```
virtual void core::TimerCounter::enableOverflowInterrupt (
    const uint8_t a_enable ) [pure virtual]
```

Implemented in [core::TimerCounter1](#), [core::TimerCounter0](#), and [core::TimerCounter2](#).

#### 9.15.2.3 [getClockPrescaler\(\)](#)

```
virtual uint16_t core::TimerCounter::getClockPrescaler ( ) [pure virtual]
```

Implemented in [core::TimerCounter1](#), [core::TimerCounter0](#), and [core::TimerCounter2](#).

#### 9.15.2.4 getCounter()

```
virtual uint16_t core::TimerCounter::getCounter ( ) const [pure virtual]
```

Implemented in [core::TimerCounter1](#), [core::TimerCounter0](#), and [core::TimerCounter2](#).

#### 9.15.2.5 getOutputCompareRegister()

```
virtual uint16_t core::TimerCounter::getOutputCompareRegister (
    const channel & ar_channel ) const [pure virtual]
```

Implemented in [core::TimerCounter1](#), [core::TimerCounter0](#), and [core::TimerCounter2](#).

#### 9.15.2.6 selectClockSource()

```
virtual void core::TimerCounter::selectClockSource (
    const clockSource & ar_clockSource ) [pure virtual]
```

Implemented in [core::TimerCounter0](#), [core::TimerCounter1](#), and [core::TimerCounter2](#).

#### 9.15.2.7 selectCompareOutputMode()

```
virtual void core::TimerCounter::selectCompareOutputMode (
    const channel & ar_channel,
    const compareOutputMode & ar_compareOutputMode ) [pure virtual]
```

Implemented in [core::TimerCounter1](#), [core::TimerCounter0](#), and [core::TimerCounter2](#).

#### 9.15.2.8 selectOperationMode()

```
virtual void core::TimerCounter::selectOperationMode (
    const operationMode & ar_operationMode ) [pure virtual]
```

Implemented in [core::TimerCounter0](#), [core::TimerCounter1](#), and [core::TimerCounter2](#).

#### 9.15.2.9 setCounter()

```
virtual void core::TimerCounter::setCounter (
    const uint16_t & ar_dataBuffer ) [pure virtual]
```

Implemented in [core::TimerCounter1](#), [core::TimerCounter0](#), and [core::TimerCounter2](#).

### 9.15.2.10 setOutputCompareRegister()

```
virtual void core::TimerCounter::setOutputCompareRegister (
    const channel & ar_channel,
    const uint16_t & ar_dataBuffer ) [pure virtual]
```

Implemented in [core::TimerCounter1](#), [core::TimerCounter0](#), and [core::TimerCounter2](#).

### 9.15.2.11 start()

```
virtual void core::TimerCounter::start ( ) [pure virtual]
```

Implemented in [core::TimerCounter0](#), [core::TimerCounter1](#), and [core::TimerCounter2](#).

### 9.15.2.12 stop()

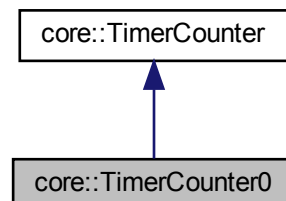
```
virtual void core::TimerCounter::stop ( ) [pure virtual]
```

Implemented in [core::TimerCounter0](#), [core::TimerCounter1](#), and [core::TimerCounter2](#).

## 9.16 core::TimerCounter0 Class Reference

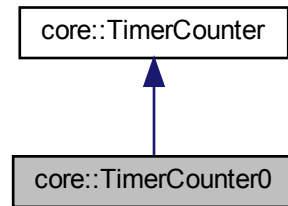
```
#include <TimerCounter0.h>
```

Inheritance diagram for core::TimerCounter0:





Collaboration diagram for core::TimerCounter0:



## Public Member Functions

- void [selectOperationMode](#) (const [operationMode](#) &ar\_operationMode) override
- void [start](#) () override
- void [stop](#) () override
- void [selectClockSource](#) (const [clockSource](#) &ar\_clockSource) override
- void [selectCompareOutputMode](#) (const [channel](#) &ar\_channel, const [compareOutputMode](#) &ar\_compareOutputMode) override
- void [setCounter](#) (const uint16\_t &ar\_dataBuffer) override
- uint16\_t [getCounter](#) () const override
- void [setOutputCompareRegister](#) (const [channel](#) &ar\_channel, const uint16\_t &ar\_dataBuffer) override
- uint16\_t [getOutputCompareRegister](#) (const [channel](#) &ar\_channel) const override
- void [enableOutputCompareMatchInterrupt](#) (const [channel](#) &ar\_channel, const uint8\_t a\_enable) override
- void [enableOverflowInterrupt](#) (const uint8\_t a\_enable) override
- uint16\_t [getClockPrescaler](#) () override

## Static Public Member Functions

- static [TimerCounter0](#) & [getInstance](#) (const [channel](#) &ar\_channel=[channel::A](#), const [operationMode](#) &ar\_operationMode=[operationMode::normal](#), const [clockSource](#) &ar\_clockSource=[clockSource::noClock](#), const [compareOutputMode](#) &ar\_compareOutputMode=[compareOutputMode::normal](#))
- static void [outputCompareMatchAServiceRoutine](#) () \_\_asm\_\_(STR(TIMERO0\_COM\_CHANNEL\_A\_INTERRUPT)) \_\_attribute\_\_((\_\_signal\_\_))
- static void [outputCompareMatchBServiceRoutine](#) () \_\_asm\_\_(STR(TIMERO0\_COM\_CHANNEL\_B\_INTERRUPT)) \_\_attribute\_\_((\_\_signal\_\_))
- static void [overflowServiceRoutine](#) () \_\_asm\_\_(STR(TIMERO0\_OVERFLOW\_INTERRUPT)) \_\_attribute\_\_((\_\_signal\_\_))

## Public Attributes

- static void [\\_\\_used\\_\\_](#)
- static void [\\_\\_externally\\_visible\\_\\_](#)

## Private Member Functions

- [TimerCounter0](#) (const [channel](#) &ar\_channel, const [operationMode](#) &ar\_operationMode, const [clockSource](#) &ar\_clockSource, const [compareOutputMode](#) &ar\_compareOutputMode)
- [~TimerCounter0](#) ()
- [TimerCounter0](#) (const [TimerCounter0](#) &)
- const [TimerCounter0](#) & [operator=](#) (const [TimerCounter0](#) &)

## Private Attributes

- uint16\_t [m\\_clockPrescaler](#)
- uint8\_t [m\\_clockSource](#)

### 9.16.1 Detailed Description

Definition at line 279 of file [TimerCounter0.h](#).

### 9.16.2 Constructor & Destructor Documentation

#### 9.16.2.1 TimerCounter0() [1/2]

```
core::TimerCounter0::TimerCounter0 (
    const channel & ar_channel,
    const operationMode & ar_operationMode,
    const clockSource & ar_clockSource,
    const compareOutputMode & ar_compareOutputMode ) [private]
```

Definition at line 20 of file [TimerCounter0.cpp](#).

```
00024 {
00025     core::MCU::enableTimerCounter0(1);
00026     stop();
00027     selectOperationMode(ar_operationMode);
00028     selectClockSource(ar_clockSource);
00029     selectCompareOutputMode(ar_channel, ar_compareOutputMode);
00030
00031
00032
00033 }
```

References [core::MCU::enableTimerCounter0\(\)](#).

#### 9.16.2.2 ~TimerCounter0()

```
core::TimerCounter0::~~TimerCounter0 ( ) [private]
```

Definition at line 34 of file [TimerCounter0.cpp](#).

```
00035 {
00036
00037 }
```

### 9.16.2.3 TimerCounter0() [2/2]

```
core::TimerCounter0::TimerCounter0 (
    const TimerCounter0 & ) [private]
```

Overried Copy constructor.

## 9.16.3 Member Function Documentation

### 9.16.3.1 enableOutputCompareMatchInterrupt()

```
void core::TimerCounter0::enableOutputCompareMatchInterrupt (
    const channel & ar_channel,
    const uint8_t a_enable ) [override], [virtual]
```

Implements [core::TimerCounter](#).

Definition at line 216 of file [TimerCounter0.cpp](#).

```
00217 {
00218     switch (ar_channel)
00219     {
00220         case core::channel::A:
00221         {
00222             if (a_enable) {
00223                 TIMER0_ENABLE_COM_CHANNEL_A_INTERRUPT;
00224             } else {
00225                 TIMER0_DISABLE_COM_CHANNEL_A_INTERRUPT;
00226             }
00227             break;
00228         }
00229         case core::channel::B:
00230         {
00231             if (a_enable) {
00232                 TIMER0_ENABLE_COM_CHANNEL_B_INTERRUPT;
00233             } else {
00234                 TIMER0_DISABLE_COM_CHANNEL_B_INTERRUPT;
00235             }
00236             break;
00237         }
00238     }
00239 }
```

References [core::A](#), [core::B](#), [TIMER0\\_DISABLE\\_COM\\_CHANNEL\\_A\\_INTERRUPT](#), [TIMER0\\_DISABLE\\_COM\\_CHANNEL\\_B\\_INTERRUPT](#), [TIMER0\\_ENABLE\\_COM\\_CHANNEL\\_A\\_INTERRUPT](#), and [TIMER0\\_ENABLE\\_COM\\_CHANNEL\\_B\\_INTERRUPT](#).

### 9.16.3.2 enableOverflowInterrupt()

```
void core::TimerCounter0::enableOverflowInterrupt (
    const uint8_t a_enable ) [override], [virtual]
```

Implements [core::TimerCounter](#).

Definition at line 251 of file [TimerCounter0.cpp](#).

```
00252 {
00253
00254     if (a_enable) {
00255
00256         TIMER0_ENABLE_OVERFLOW_INTERRUPT;
00257
00258     } else {
00259
00260         TIMER0_DISABLE_OVERFLOW_INTERRUPT;
00261     }
00262
00263 }
```

References [TIMER0\\_DISABLE\\_OVERFLOW\\_INTERRUPT](#), and [TIMER0\\_ENABLE\\_OVERFLOW\\_INTERRUPT](#).

### 9.16.3.3 getClockPrescaler()

```
uint16_t core::TimerCounter0::getClockPrescaler ( ) [override], [virtual]
```

Implements [core::TimerCounter](#).

Definition at line 141 of file [TimerCounter0.cpp](#).

```
00142 {
00143     return m_clockPrescaler;
00144 }
```

### 9.16.3.4 getCounter()

```
uint16_t core::TimerCounter0::getCounter ( ) const [override], [virtual]
```

Implements [core::TimerCounter](#).

Definition at line 176 of file [TimerCounter0.cpp](#).

```
00177 {
00178     return TCNT0;
00179 }
```

### 9.16.3.5 getInstance()

```
core::TimerCounter0 & core::TimerCounter0::getInstance (
    const channel & ar_channel = channel::A,
    const operationMode & ar_operationMode = operationMode::normal,
    const clockSource & ar_clockSource = clockSource::noClock,
    const compareOutputMode & ar_compareOutputMode = compareOutputMode::normal )
[static]
```

Definition at line 5 of file [TimerCounter0.cpp](#).

```
00009 {
00010     static TimerCounter0 l_instance(ar_channel,
00011                                     ar_operationMode,
00012                                     ar_clockSource,
00013                                     ar_compareOutputMode);
00014
00015     return l_instance;
00016
00017
00018 }
```

### 9.16.3.6 getOutputCompareRegister()

```
uint16_t core::TimerCounter0::getOutputCompareRegister (
    const channel & ar_channel ) const [override], [virtual]
```

Implements [core::TimerCounter](#).

Definition at line 201 of file [TimerCounter0.cpp](#).

```
00202 {
00203     switch (ar_channel)
00204     {
00205         case core::channel::A:
00206         {
00207             return OCR0A;
00208         }
00209         case core::channel::B:
00210         {
00211             return OCR0B;
00212         }
00213     }
00214 }
```

References [core::A](#), and [core::B](#).

### 9.16.3.7 operator=()

```
const TimerCounter0& core::TimerCounter0::operator= (
    const TimerCounter0 & ) [private]
```

Override assign operator.

### 9.16.3.8 outputCompareMatchAServiceRoutine()

```
static void core::TimerCounter0::outputCompareMatchAServiceRoutine ( ) [static]
```

### 9.16.3.9 outputCompareMatchBServiceRoutine()

```
static void core::TimerCounter0::outputCompareMatchBServiceRoutine ( ) [static]
```

### 9.16.3.10 overflowServiceRoutine()

```
static void core::TimerCounter0::overflowServiceRoutine ( ) [static]
```

### 9.16.3.11 selectClockSource()

```
void core::TimerCounter0::selectClockSource (
    const clockSource & ar_clockSource ) [override], [virtual]
```

Implements [core::TimerCounter](#).

Definition at line 39 of file [TimerCounter0.cpp](#).

```
00040 {
00041     switch (ar_clockSource)
00042     {
00043         case core::clockSource::noClock:
00044         {
00045             m_clockPrescaler=0;
00046             m_clockSource=0;
00047             break;
00048         }
00049         case core::clockSource::PS_1:
00050         {
00051             m_clockPrescaler=1;
00052             m_clockSource=1;
00053             break;
00054         }
00055         case core::clockSource::PS_8:
00056         {
00057             m_clockPrescaler=8;
00058             m_clockSource=2;
00059             break;
00060         }
00061         case core::clockSource::PS_64:
00062         {
00063             m_clockPrescaler=64;
00064             m_clockSource=3;
00065             break;
00066         }
00067         case core::clockSource::PS_256:
00068         {
00069             m_clockPrescaler=256;
00070             m_clockSource=4;
00071             break;
00072         }
00073         case core::clockSource::PS_1024:
00074         {
00075             m_clockPrescaler=1024;
00076             m_clockSource=5;
00077             break;
00078         }
00079         case core::clockSource::extern_Clock_T0_Falling_Edge:
00080         {
00081             m_clockPrescaler=0;
00082             m_clockSource=6;
00083             break;
00084         }
00085         case core::clockSource::extern_Clock_T0_Rising_Edge:
00086         {
00087             m_clockPrescaler=0;
00088             m_clockSource=7;
00089             break;
00090         }
00091     }
```

```
00092
00093
00094 }
```

References [core::extern\\_Clock\\_T0\\_Falling\\_Edge](#), [core::extern\\_Clock\\_T0\\_Rising\\_Edge](#), [core::noClock](#), [core::PS\\_1](#), [core::PS\\_1024](#), [core::PS\\_256](#), [core::PS\\_64](#), and [core::PS\\_8](#).

### 9.16.3.12 selectCompareOutputMode()

```
void core::TimerCounter0::selectCompareOutputMode (
    const channel & ar_channel,
    const compareOutputMode & ar_compareOutputMode ) [override], [virtual]
```

Implements [core::TimerCounter](#).

Definition at line 152 of file [TimerCounter0.cpp](#).

```
00153 {
00154     switch (ar_channel)
00155     {
00156         case core::channel::A:
00157         {
00158             TIMER0\_SELECT\_COM\_CHANNEL\_A(static_cast<uint8_t>(ar_compareOutputMode));
00159             break;
00160         }
00161         case core::channel::B:
00162         {
00163             TIMER0\_SELECT\_COM\_CHANNEL\_B(static_cast<uint8_t>(ar_compareOutputMode));
00164             break;
00165         }
00166     }
00167 }
```

References [core::A](#), [core::B](#), [TIMER0\\_SELECT\\_COM\\_CHANNEL\\_A](#), and [TIMER0\\_SELECT\\_COM\\_CHANNEL\\_B](#).

### 9.16.3.13 selectOperationMode()

```
void core::TimerCounter0::selectOperationMode (
    const operationMode & ar_operationMode ) [override], [virtual]
```

Implements [core::TimerCounter](#).

Definition at line 96 of file [TimerCounter0.cpp](#).

```
00097 {
00098     switch (ar_operationMode)
00099     {
00100         case core::operationMode::normal:
00101         {
00102             TIMER0\_SELECT\_OPERATION\_MODE(0);
00103             break;
00104         }
00105         case core::operationMode::PWM\_PC:
00106         {
00107             TIMER0\_SELECT\_OPERATION\_MODE(1);
00108             break;
00109         }
00110         case core::operationMode::CTC\_OCR:
00111         {
00112             TIMER0\_SELECT\_OPERATION\_MODE(2);
00113             break;
00114         }
00115         case core::operationMode::fast\_PWM:
00116         {
00117             TIMER0\_SELECT\_OPERATION\_MODE(3);
00118             break;
00119         }
00120     }
00121 }
```

```

00119     }
00120     case core::operationMode::PWM_PC_OCR:
00121     {
00122         TIMER0_SELECT_OPERATION_MODE(5);
00123         break;
00124     }
00125     case core::operationMode::fast_PWM_OCR:
00126     {
00127         TIMER0_SELECT_OPERATION_MODE(7);
00128         break;
00129     }
00130
00131 }
00132
00133 }

```

References [core::CTC\\_OCR](#), [core::fast\\_PWM](#), [core::fast\\_PWM\\_OCR](#), [core::normal](#), [core::PWM\\_PC](#), [core::PWM\\_PC\\_OCR](#), and [TIMER0\\_SELECT\\_OPERATION\\_MODE](#).

### 9.16.3.14 setCounter()

```

void core::TimerCounter0::setCounter (
    const uint16_t & ar_dataBuffer ) [override], [virtual]

```

Implements [core::TimerCounter](#).

Definition at line 171 of file [TimerCounter0.cpp](#).

```

00172 {
00173     TCNT0 = static_cast<uint8_t>(ar_dataBuffer);
00174 }

```

### 9.16.3.15 setOutputCompareRegister()

```

void core::TimerCounter0::setOutputCompareRegister (
    const channel & ar_channel,
    const uint16_t & ar_dataBuffer ) [override], [virtual]

```

Implements [core::TimerCounter](#).

Definition at line 181 of file [TimerCounter0.cpp](#).

```

00182 {
00183     switch (ar_channel)
00184     {
00185         case core::channel::A:
00186         {
00187             OCR0A = static_cast<uint8_t>(ar_dataBuffer);
00188             break;
00189         }
00190         case core::channel::B:
00191         {
00192             OCR0B = static_cast<uint8_t>(ar_dataBuffer);
00193             break;
00194         }
00195     }
00196 }
00197 }

```

References [core::A](#), and [core::B](#).



### 9.16.3.16 start()

```
void core::TimerCounter0::start ( ) [override], [virtual]
```

Implements [core::TimerCounter](#).

Definition at line 136 of file [TimerCounter0.cpp](#).

```
00137 {  
00138     TIMER0_SELECT_CLOCK_SOURCE(m_clockSource);  
00139 }
```

References [TIMER0\\_SELECT\\_CLOCK\\_SOURCE](#).

### 9.16.3.17 stop()

```
void core::TimerCounter0::stop ( ) [override], [virtual]
```

Implements [core::TimerCounter](#).

Definition at line 146 of file [TimerCounter0.cpp](#).

```
00147 {  
00148     TIMER0_STOP;  
00149 }
```

References [TIMER0\\_STOP](#).

## 9.16.4 Member Data Documentation

### 9.16.4.1 \_\_externally\_visible\_\_

```
static void core::TimerCounter0::__externally_visible__
```

Definition at line 313 of file [TimerCounter0.h](#).

### 9.16.4.2 \_\_used\_\_

```
static void core::TimerCounter0::__used__
```

Definition at line 313 of file [TimerCounter0.h](#).

#### 9.16.4.3 m\_clockPrescaler

```
uint16_t core::TimerCounter0::m_clockPrescaler [private]
```

Definition at line 340 of file [TimerCounter0.h](#).

#### 9.16.4.4 m\_clockSource

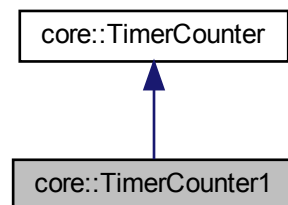
```
uint8_t core::TimerCounter0::m_clockSource [private]
```

Definition at line 342 of file [TimerCounter0.h](#).

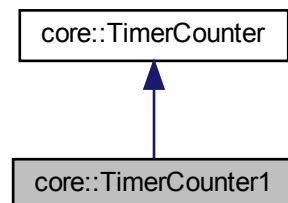
### 9.17 core::TimerCounter1 Class Reference

```
#include <TimerCounter1.h>
```

Inheritance diagram for core::TimerCounter1:



Collaboration diagram for core::TimerCounter1:



## Public Member Functions

- void [selectOperationMode](#) (const [operationMode](#) &ar\_operationMode) override
- void [start](#) () override
- void [stop](#) () override
- void [selectClockSource](#) (const [clockSource](#) &ar\_clockSource) override
- void [selectCompareOutputMode](#) (const [channel](#) &ar\_channel, const [compareOutputMode](#) &ar\_compareOutputMode) override
- void [setCounter](#) (const uint16\_t &ar\_dataBuffer) override
- uint16\_t [getCounter](#) () const override
- void [setOutputCompareRegister](#) (const [channel](#) &ar\_channel, const uint16\_t &ar\_dataBuffer) override
- uint16\_t [getOutputCompareRegister](#) (const [channel](#) &ar\_channel) const override
- void [setInputCaptureRegister](#) (const uint16\_t &ar\_dataBuffer)
- uint16\_t [getInputCaptureRegister](#) () const
- void [enableOutputCompareMatchInterrupt](#) (const [channel](#) &ar\_channel, const uint8\_t a\_enable) override
- void [enableOverflowInterrupt](#) (const uint8\_t a\_enable) override
- void [enableInputCaptureInterrupt](#) (const uint8\_t a\_enable)
- uint16\_t [getClockPrescaler](#) () override

## Static Public Member Functions

- static [TimerCounter1](#) & [getInstance](#) (const [channel](#) &ar\_channel=[channel::A](#), const [operationMode](#) &ar\_operationMode=[operationMode::normal](#), const [clockSource](#) &ar\_clockSource=[clockSource::noClock](#), const [compareOutputMode](#) &ar\_compareOutputMode=[compareOutputMode::normal](#))
- static void [outputCompareMatchAServiceRoutine](#) () \_\_asm\_\_(STR(TIMER1\_COM\_CHANNEL\_A\_INTERRUPT)) \_\_attribute\_\_((\_\_signal\_\_))
- static void [outputCompareMatchBServiceRoutine](#) () \_\_asm\_\_(STR(TIMER1\_COM\_CHANNEL\_B\_INTERRUPT)) \_\_attribute\_\_((\_\_signal\_\_))
- static void [overflowServiceRoutine](#) () \_\_asm\_\_(STR(TIMER1\_OVERFLOW\_INTERRUPT)) \_\_attribute\_\_((\_\_signal\_\_))
- static void [inputCaptureServiceRoutine](#) () \_\_asm\_\_(STR(TIMER1\_INPUT\_CAPTURE\_INTERRUPT)) \_\_attribute\_\_((\_\_signal\_\_))

## Public Attributes

- static void [\\_\\_used\\_\\_](#)
- static void [\\_\\_externally\\_visible\\_\\_](#)

## Private Member Functions

- [TimerCounter1](#) (const [channel](#) &ar\_channel, const [operationMode](#) &ar\_operationMode, const [clockSource](#) &ar\_clockSource, const [compareOutputMode](#) &ar\_compareOutputMode)
- [~TimerCounter1](#) ()
- [TimerCounter1](#) (const [TimerCounter1](#) &)
- const [TimerCounter1](#) & [operator=](#) (const [TimerCounter1](#) &)

## Private Attributes

- uint16\_t [m\\_clockPrescaler](#)
- uint8\_t [m\\_clockSource](#)

### 9.17.1 Detailed Description

Definition at line 279 of file [TimerCounter1.h](#).

### 9.17.2 Constructor & Destructor Documentation

#### 9.17.2.1 TimerCounter1() [1/2]

```
core::TimerCounter1::TimerCounter1 (
    const channel & ar_channel,
    const operationMode & ar_operationMode,
    const clockSource & ar_clockSource,
    const compareOutputMode & ar_compareOutputMode ) [private]
```

Definition at line 20 of file [TimerCounter1.cpp](#).

```
00024 {
00025     core::MCU::enableTimerCounter1(1);
00026     stop();
00027     selectOperationMode(ar_operationMode);
00028     selectClockSource(ar_clockSource);
00029     selectCompareOutputMode(ar_channel,ar_compareOutputMode);
00030
00031
00032 }
```

References [core::MCU::enableTimerCounter1\(\)](#).

#### 9.17.2.2 ~TimerCounter1()

```
core::TimerCounter1::~~TimerCounter1 ( ) [private]
```

Definition at line 33 of file [TimerCounter1.cpp](#).

```
00034 {
00035
00036 }
```

#### 9.17.2.3 TimerCounter1() [2/2]

```
core::TimerCounter1::TimerCounter1 (
    const TimerCounter1 & ) [private]
```

Overried Copy constructor.

### 9.17.3 Member Function Documentation

### 9.17.3.1 enableInputCaptureInterrupt()

```
void core::TimerCounter1::enableInputCaptureInterrupt (
    const uint8_t a_enable )
```

Definition at line 309 of file [TimerCounter1.cpp](#).

```
00310 {
00311     if (a_enable) {
00312
00313         TIMER1_ENABLE_INPUT_CAPTURE_INTERRUPT;
00314
00315     } else {
00316
00317         TIMER1_DISABLE_INPUT_CAPTURE_INTERRUPT;
00318     }
00319 }
```

References [TIMER1\\_DISABLE\\_INPUT\\_CAPTURE\\_INTERRUPT](#), and [TIMER1\\_ENABLE\\_INPUT\\_CAPTURE\\_INTERRUPT](#).

### 9.17.3.2 enableOutputCompareMatchInterrupt()

```
void core::TimerCounter1::enableOutputCompareMatchInterrupt (
    const channel & ar_channel,
    const uint8_t a_enable ) [override], [virtual]
```

Implements [core::TimerCounter](#).

Definition at line 260 of file [TimerCounter1.cpp](#).

```
00261 {
00262     switch (ar_channel)
00263     {
00264         case core::channel::A:
00265         {
00266             if (a_enable) {
00267
00268                 TIMER1_ENABLE_COM_CHANNEL_A_INTERRUPT;
00269
00270             } else {
00271
00272                 TIMER1_DISABLE_COM_CHANNEL_A_INTERRUPT;
00273             }
00274             break;
00275         }
00276         case core::channel::B:
00277         {
00278             if (a_enable) {
00279
00280                 TIMER1_ENABLE_COM_CHANNEL_B_INTERRUPT;
00281
00282             } else {
00283
00284                 TIMER1_DISABLE_COM_CHANNEL_B_INTERRUPT;
00285             }
00286             break;
00287         }
00288     }
00289 }
00290
00291 }
```

References [core::A](#), [core::B](#), [TIMER1\\_DISABLE\\_COM\\_CHANNEL\\_A\\_INTERRUPT](#), [TIMER1\\_DISABLE\\_COM\\_CHANNEL\\_B\\_INTERRUPT](#), [TIMER1\\_ENABLE\\_COM\\_CHANNEL\\_A\\_INTERRUPT](#), and [TIMER1\\_ENABLE\\_COM\\_CHANNEL\\_B\\_INTERRUPT](#).

### 9.17.3.3 enableOverflowInterrupt()

```
void core::TimerCounter1::enableOverflowInterrupt (
    const uint8_t a_enable ) [override], [virtual]
```

Implements [core::TimerCounter](#).

Definition at line 295 of file [TimerCounter1.cpp](#).

```
00296 {
00297
00298     if (a_enable) {
00299         TIMER1_ENABLE_OVERFLOW_INTERRUPT;
00300     } else {
00301         TIMER1_DISABLE_OVERFLOW_INTERRUPT;
00302     }
00303
00304     return;
00305 }
00306
00307 }
```

References [TIMER1\\_DISABLE\\_OVERFLOW\\_INTERRUPT](#), and [TIMER1\\_ENABLE\\_OVERFLOW\\_INTERRUPT](#).

### 9.17.3.4 getClockPrescaler()

```
uint16_t core::TimerCounter1::getClockPrescaler ( ) [override], [virtual]
```

Implements [core::TimerCounter](#).

Definition at line 186 of file [TimerCounter1.cpp](#).

```
00187 {
00188     return m_clockPrescaler;
00189 }
```

Referenced by [component::ServoMotor::connect\(\)](#), and [component::ServoMotor::rotate\(\)](#).

### 9.17.3.5 getCounter()

```
uint16_t core::TimerCounter1::getCounter ( ) const [override], [virtual]
```

Implements [core::TimerCounter](#).

Definition at line 221 of file [TimerCounter1.cpp](#).

```
00222 {
00223     return TCNT1;
00224 }
```

### 9.17.3.6 getInputCaptureRegister()

```
uint16_t core::TimerCounter1::getInputCaptureRegister ( ) const
```

Definition at line 327 of file [TimerCounter1.cpp](#).

```
00328 {
00329     return ICR1;
00330 }
```

### 9.17.3.7 getInstance()

```
core::TimerCounter1 & core::TimerCounter1::getInstance (
    const channel & ar_channel = channel::A,
    const operationMode & ar_operationMode = operationMode::normal,
    const clockSource & ar_clockSource = clockSource::noClock,
    const compareOutputMode & ar_compareOutputMode = compareOutputMode::normal )
[static]
```

Definition at line 5 of file [TimerCounter1.cpp](#).

```
00009 {
00010     static TimerCounter1 l_instance(ar_channel,
00011                                     ar_operationMode,
00012                                     ar_clockSource,
00013                                     ar_compareOutputMode);
00014
00015     return l_instance;
00016
00017
00018 }
```

### 9.17.3.8 getOutputCompareRegister()

```
uint16_t core::TimerCounter1::getOutputCompareRegister (
    const channel & ar_channel ) const [override], [virtual]
```

Implements [core::TimerCounter](#).

Definition at line 245 of file [TimerCounter1.cpp](#).

```
00246 {
00247     switch (ar_channel)
00248     {
00249         case core::channel::A:
00250         {
00251             return OCR1A;
00252         }
00253         case core::channel::B:
00254         {
00255             return OCR1B;
00256         }
00257     }
00258 }
```

References [core::A](#), and [core::B](#).

### 9.17.3.9 inputCaptureServiceRoutine()

```
static void core::TimerCounter1::inputCaptureServiceRoutine ( ) [static]
```

### 9.17.3.10 operator=()

```
const TimerCounter1 & core::TimerCounter1::operator= (
    const TimerCounter1 & ) [private]
```

Override assign operator.

**9.17.3.11 outputCompareMatchAServiceRoutine()**

```
static void core::TimerCounter1::outputCompareMatchAServiceRoutine ( ) [static]
```

**9.17.3.12 outputCompareMatchBServiceRoutine()**

```
static void core::TimerCounter1::outputCompareMatchBServiceRoutine ( ) [static]
```

**9.17.3.13 overflowServiceRoutine()**

```
static void core::TimerCounter1::overflowServiceRoutine ( ) [static]
```

**9.17.3.14 selectClockSource()**

```
void core::TimerCounter1::selectClockSource (
    const clockSource & ar_clockSource ) [override], [virtual]
```

Implements [core::TimerCounter](#).

Definition at line 38 of file [TimerCounter1.cpp](#).

```
00039 {
00040     switch (ar_clockSource)
00041     {
00042         case core::clockSource::noClock:
00043         {
00044             m_clockPrescaler=0;
00045             m_clockSource=0;
00046             break;
00047         }
00048         case core::clockSource::PS_1:
00049         {
00050             m_clockPrescaler=1;
00051             m_clockSource=1;
00052             break;
00053         }
00054         case core::clockSource::PS_8:
00055         {
00056             m_clockPrescaler=8;
00057             m_clockSource=2;
00058             break;
00059         }
00060         case core::clockSource::PS_64:
00061         {
00062             m_clockPrescaler=64;
00063             m_clockSource=3;
00064             break;
00065         }
00066         case core::clockSource::PS_256:
00067         {
00068             m_clockPrescaler=256;
00069             m_clockSource=4;
00070             break;
00071         }
00072         case core::clockSource::PS_1024:
00073         {
00074             m_clockPrescaler=1024;
00075             m_clockSource=5;
00076             break;
00077         }
00078         case core::clockSource::extern_Clock_T0_Falling_Edge:
```



```

00079     {
00080         m_clockPrescaler=0;
00081         m_clockSource=6;
00082         break;
00083     }
00084     case core::clockSource::extern_Clock_T0_Rising_Edge:
00085     {
00086         m_clockPrescaler=0;
00087         m_clockSource=7;
00088         break;
00089     }
00090 }
00091
00092
00093 }
```

References [core::extern\\_Clock\\_T0\\_Falling\\_Edge](#), [core::extern\\_Clock\\_T0\\_Rising\\_Edge](#), [core::noClock](#), [core::PS\\_1](#), [core::PS\\_1024](#), [core::PS\\_256](#), [core::PS\\_64](#), and [core::PS\\_8](#).

### 9.17.3.15 selectCompareOutputMode()

```

void core::TimerCounter1::selectCompareOutputMode (
    const channel & ar_channel,
    const compareOutputMode & ar_compareOutputMode ) [override], [virtual]
```

Implements [core::TimerCounter](#).

Definition at line 197 of file [TimerCounter1.cpp](#).

```

00198 {
00199     switch (ar_channel)
00200     {
00201         case core::channel::A:
00202         {
00203             TIMER1_SELECT_COM_CHANNEL_A(static_cast<uint8_t>(ar_compareOutputMode));
00204             break;
00205         }
00206         case core::channel::B:
00207         {
00208             TIMER1_SELECT_COM_CHANNEL_B(static_cast<uint8_t>(ar_compareOutputMode));
00209             break;
00210         }
00211     }
00212 }
00213 }
```

References [core::A](#), [core::B](#), [TIMER1\\_SELECT\\_COM\\_CHANNEL\\_A](#), and [TIMER1\\_SELECT\\_COM\\_CHANNEL\\_B](#).

Referenced by [component::ServoMotor::connect\(\)](#), and [component::ServoMotor::disconnect\(\)](#).

### 9.17.3.16 selectOperationMode()

```

void core::TimerCounter1::selectOperationMode (
    const operationMode & ar_operationMode ) [override], [virtual]
```

Implements [core::TimerCounter](#).

Definition at line 95 of file [TimerCounter1.cpp](#).

```

00096 {
00097     switch (ar_operationMode)
00098     {
00099         case core::operationMode::normal:
00100         {
00101             TIMER1_SELECT_OPERATION_MODE(0);
```

```

00102         break;
00103     }
00104     case core::operationMode::PWM_PC_8bit:
00105     {
00106         TIMER1_SELECT_OPERATION_MODE(1);
00107         break;
00108     }
00109     case core::operationMode::PWM_PC_9bit:
00110     {
00111         TIMER1_SELECT_OPERATION_MODE(2);
00112         break;
00113     }
00114     case core::operationMode::PWM_PC_10bit:
00115     {
00116         TIMER1_SELECT_OPERATION_MODE(3);
00117         break;
00118     }
00119     case core::operationMode::CTC_OCR:
00120     {
00121         TIMER1_SELECT_OPERATION_MODE(4);
00122         break;
00123     }
00124     case core::operationMode::fast_PWM_8bit:
00125     {
00126         TIMER1_SELECT_OPERATION_MODE(5);
00127         break;
00128     }
00129     case core::operationMode::fast_PWM_9bit:
00130     {
00131         TIMER1_SELECT_OPERATION_MODE(6);
00132         break;
00133     }
00134     case core::operationMode::fast_PWM_10bit:
00135     {
00136         TIMER1_SELECT_OPERATION_MODE(7);
00137         break;
00138     }
00139     case core::operationMode::PWM_PFC_ICR:
00140     {
00141         TIMER1_SELECT_OPERATION_MODE(8);
00142         break;
00143     }
00144     case core::operationMode::PWM_PFC_OCR:
00145     {
00146         TIMER1_SELECT_OPERATION_MODE(9);
00147         break;
00148     }
00149     case core::operationMode::PWM_PC_ICR:
00150     {
00151         TIMER1_SELECT_OPERATION_MODE(10);
00152         break;
00153     }
00154     case core::operationMode::PWM_PC_OCR:
00155     {
00156         TIMER1_SELECT_OPERATION_MODE(11);
00157         break;
00158     }
00159     case core::operationMode::CTC_ICR:
00160     {
00161         TIMER1_SELECT_OPERATION_MODE(12);
00162         break;
00163     }
00164     case core::operationMode::fast_PWM_ICR:
00165     {
00166         TIMER1_SELECT_OPERATION_MODE(14);
00167         break;
00168     }
00169     case core::operationMode::fast_PWM_OCR:
00170     {
00171         TIMER1_SELECT_OPERATION_MODE(15);
00172         break;
00173     }
00174 }
00175 }
00176
00177
00178 }

```

References [core::CTC\\_ICR](#), [core::CTC\\_OCR](#), [core::fast\\_PWM\\_10bit](#), [core::fast\\_PWM\\_8bit](#), [core::fast\\_PWM\\_9bit](#), [core::fast\\_PWM\\_ICR](#), [core::fast\\_PWM\\_OCR](#), [core::normal](#), [core::PWM\\_PC\\_10bit](#), [core::PWM\\_PC\\_8bit](#), [core::PWM\\_PC\\_9bit](#), [core::PWM\\_PC\\_ICR](#), [core::PWM\\_PC\\_OCR](#), [core::PWM\\_PFC\\_ICR](#), [core::PWM\\_PFC\\_OCR](#), and [TIMER1\\_SELECT\\_OPERATION\\_MODE](#).

Referenced by [component::ServoMotor::connect\(\)](#).

### 9.17.3.17 setCounter()

```
void core::TimerCounter1::setCounter (
    const uint16_t & ar_dataBuffer ) [override], [virtual]
```

Implements [core::TimerCounter](#).

Definition at line 216 of file [TimerCounter1.cpp](#).

```
00217 {
00218     TCNT1 = ar_dataBuffer;
00219 }
```

Referenced by [component::ServoMotor::connect\(\)](#).

### 9.17.3.18 setInputCaptureRegister()

```
void core::TimerCounter1::setInputCaptureRegister (
    const uint16_t & ar_dataBuffer )
```

Definition at line 321 of file [TimerCounter1.cpp](#).

```
00322 {
00323     ICR1 = ar_dataBuffer;
00324 }
```

Referenced by [component::ServoMotor::connect\(\)](#).

### 9.17.3.19 setOutputCompareRegister()

```
void core::TimerCounter1::setOutputCompareRegister (
    const channel & ar_channel,
    const uint16_t & ar_dataBuffer ) [override], [virtual]
```

Implements [core::TimerCounter](#).

Definition at line 226 of file [TimerCounter1.cpp](#).

```
00227 {
00228     switch (ar_channel)
00229     {
00230         case core::channel::A:
00231         {
00232             OCR1A = ar_dataBuffer;
00233             break;
00234         }
00235         case core::channel::B:
00236         {
00237             OCR1B = ar_dataBuffer;
00238             break;
00239         }
00240     }
00241 }
```

References [core::A](#), and [core::B](#).

Referenced by [component::ServoMotor::rotate\(\)](#).

### 9.17.3.20 start()

```
void core::TimerCounter1::start ( ) [override], [virtual]
```

Implements [core::TimerCounter](#).

Definition at line 181 of file [TimerCounter1.cpp](#).

```
00182 {  
00183     TIMER1\_SELECT\_CLOCK\_SOURCE(m_clockSource);  
00184 }
```

References [TIMER1\\_SELECT\\_CLOCK\\_SOURCE](#).

Referenced by [component::ServoMotor::rotate\(\)](#).

### 9.17.3.21 stop()

```
void core::TimerCounter1::stop ( ) [override], [virtual]
```

Implements [core::TimerCounter](#).

Definition at line 191 of file [TimerCounter1.cpp](#).

```
00192 {  
00193     TIMER1\_STOP;  
00194 }
```

References [TIMER1\\_STOP](#).

Referenced by [component::ServoMotor::disconnect\(\)](#).

## 9.17.4 Member Data Documentation

### 9.17.4.1 \_\_externally\_visible\_\_

```
static void core::TimerCounter1::__externally_visible__
```

Definition at line 320 of file [TimerCounter1.h](#).

### 9.17.4.2 \_\_used\_\_

```
static void core::TimerCounter1::__used__
```

Definition at line 320 of file [TimerCounter1.h](#).

#### 9.17.4.3 m\_clockPrescaler

```
uint16_t core::TimerCounter1::m_clockPrescaler [private]
```

Definition at line 349 of file [TimerCounter1.h](#).

#### 9.17.4.4 m\_clockSource

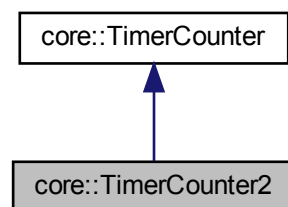
```
uint8_t core::TimerCounter1::m_clockSource [private]
```

Definition at line 351 of file [TimerCounter1.h](#).

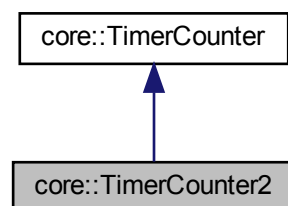
## 9.18 core::TimerCounter2 Class Reference

```
#include <TimerCounter2.h>
```

Inheritance diagram for core::TimerCounter2:



Collaboration diagram for core::TimerCounter2:



## Public Member Functions

- void [selectOperationMode](#) (const [operationMode](#) &ar\_operationMode) override
- void [start](#) () override
- void [stop](#) () override
- void [selectClockSource](#) (const [clockSource](#) &ar\_clockSource) override
- void [selectCompareOutputMode](#) (const [channel](#) &ar\_channel, const [compareOutputMode](#) &ar\_compareOutputMode) override
- void [setCounter](#) (const uint16\_t &ar\_dataBuffer) override
- uint16\_t [getCounter](#) () const override
- void [setOutputCompareRegister](#) (const [channel](#) &ar\_channel, const uint16\_t &ar\_dataBuffer) override
- uint16\_t [getOutputCompareRegister](#) (const [channel](#) &ar\_channel) const override
- void [enableOutputCompareMatchInterrupt](#) (const [channel](#) &ar\_channel, const uint8\_t a\_enable) override
- void [enableOverflowInterrupt](#) (const uint8\_t a\_enable) override
- uint16\_t [getClockPrescaler](#) () override

## Static Public Member Functions

- static [TimerCounter2](#) & [getInstance](#) (const [channel](#) &ar\_channel=[channel::A](#), const [operationMode](#) &ar\_operationMode=[operationMode::normal](#), const [clockSource](#) &ar\_clockSource=[clockSource::noClock](#), const [compareOutputMode](#) &ar\_compareOutputMode=[compareOutputMode::normal](#))
- static void [outputCompareMatchAServiceRoutine](#) () \_\_asm\_\_(STR(TIMER2\_COM\_CHANNEL\_A\_INTERRUPT)) \_\_attribute\_\_((\_\_signal\_\_))
- static void [outputCompareMatchBServiceRoutine](#) () \_\_asm\_\_(STR(TIMER2\_COM\_CHANNEL\_B\_INTERRUPT)) \_\_attribute\_\_((\_\_signal\_\_))
- static void [overflowServiceRoutine](#) () \_\_asm\_\_(STR(TIMER2\_OVERFLOW\_INTERRUPT)) \_\_attribute\_\_((\_\_signal\_\_))

## Public Attributes

- static void [\\_\\_used\\_\\_](#)
- static void [\\_\\_externally\\_visible\\_\\_](#)

## Private Member Functions

- [TimerCounter2](#) (const [channel](#) &ar\_channel, const [operationMode](#) &ar\_operationMode, const [clockSource](#) &ar\_clockSource, const [compareOutputMode](#) &ar\_compareOutputMode)
- [~TimerCounter2](#) ()
- [TimerCounter2](#) (const [TimerCounter2](#) &)
- const [TimerCounter2](#) & [operator=](#) (const [TimerCounter2](#) &)

## Private Attributes

- uint16\_t [m\\_clockPrescaler](#)
- uint8\_t [m\\_clockSource](#)

### 9.18.1 Detailed Description

Definition at line 279 of file [TimerCounter2.h](#).

## 9.18.2 Constructor & Destructor Documentation

### 9.18.2.1 TimerCounter2() [1/2]

```
core::TimerCounter2::TimerCounter2 (
    const channel & ar_channel,
    const operationMode & ar_operationMode,
    const clockSource & ar_clockSource,
    const compareOutputMode & ar_compareOutputMode ) [private]
```

Definition at line 20 of file [TimerCounter2.cpp](#).

```
00024 {
00025     core::MCU::enableTimerCounter2(1);
00026     stop();
00027     selectOperationMode(ar_operationMode);
00028     selectClockSource(ar_clockSource);
00029     selectCompareOutputMode(ar_channel, ar_compareOutputMode);
00030
00031 }
```

References [core::MCU::enableTimerCounter2\(\)](#).

### 9.18.2.2 ~TimerCounter2()

```
core::TimerCounter2::~~TimerCounter2 ( ) [private]
```

Definition at line 32 of file [TimerCounter2.cpp](#).

```
00033 {
00034
00035 }
```

### 9.18.2.3 TimerCounter2() [2/2]

```
core::TimerCounter2::TimerCounter2 (
    const TimerCounter2 & ) [private]
```

Overried Copy constructor.

## 9.18.3 Member Function Documentation

### 9.18.3.1 enableOutputCompareMatchInterrupt()

```
void core::TimerCounter2::enableOutputCompareMatchInterrupt (
    const channel & ar_channel,
    const uint8_t a_enable ) [override], [virtual]
```

Implements [core::TimerCounter](#).

Definition at line 215 of file [TimerCounter2.cpp](#).

```
00216 {
00217     switch (ar_channel)
00218     {
00219         case core::channel::A:
00220         {
00221             if (a_enable) {
00222
00223                 TIMER2_ENABLE_COM_CHANNEL_A_INTERRUPT;
00224
00225             } else {
00226
00227                 TIMER2_DISABLE_COM_CHANNEL_A_INTERRUPT;
00228
00229             }
00230             break;
00231         }
00232         case core::channel::B:
00233         {
00234             if (a_enable) {
00235
00236                 TIMER2_ENABLE_COM_CHANNEL_B_INTERRUPT;
00237
00238             } else {
00239
00240                 TIMER2_DISABLE_COM_CHANNEL_B_INTERRUPT;
00241
00242             }
00243             break;
00244         }
00245     }
00246 }
```

References [core::A](#), [core::B](#), [TIMER2\\_DISABLE\\_COM\\_CHANNEL\\_A\\_INTERRUPT](#), [TIMER2\\_DISABLE\\_COM\\_CHANNEL\\_B\\_INTERRUPT](#), [TIMER2\\_ENABLE\\_COM\\_CHANNEL\\_A\\_INTERRUPT](#), and [TIMER2\\_ENABLE\\_COM\\_CHANNEL\\_B\\_INTERRUPT](#).

### 9.18.3.2 enableOverflowInterrupt()

```
void core::TimerCounter2::enableOverflowInterrupt (
    const uint8_t a_enable ) [override], [virtual]
```

Implements [core::TimerCounter](#).

Definition at line 250 of file [TimerCounter2.cpp](#).

```
00251 {
00252
00253     if (a_enable) {
00254
00255         TIMER2_ENABLE_OVERFLOW_INTERRUPT;
00256
00257     } else {
00258
00259         TIMER2_DISABLE_OVERFLOW_INTERRUPT;
00260     }
00261 }
00262 }
```

References [TIMER2\\_DISABLE\\_OVERFLOW\\_INTERRUPT](#), and [TIMER2\\_ENABLE\\_OVERFLOW\\_INTERRUPT](#).



### 9.18.3.3 getClockPrescaler()

```
uint16_t core::TimerCounter2::getClockPrescaler ( ) [override], [virtual]
```

Implements [core::TimerCounter](#).

Definition at line 138 of file [TimerCounter2.cpp](#).

```
00139 {  
00140     return m_clockPrescaler;  
00141 }
```

### 9.18.3.4 getCounter()

```
uint16_t core::TimerCounter2::getCounter ( ) const [override], [virtual]
```

Implements [core::TimerCounter](#).

Definition at line 176 of file [TimerCounter2.cpp](#).

```
00177 {  
00178     return TCNT2;  
00179 }
```

### 9.18.3.5 getInstance()

```
core::TimerCounter2 & core::TimerCounter2::getInstance (  
    const channel & ar_channel = channel::A,  
    const operationMode & ar_operationMode = operationMode::normal,  
    const clockSource & ar_clockSource = clockSource::noClock,  
    const compareOutputMode & ar_compareOutputMode = compareOutputMode::normal )  
[static]
```

Definition at line 5 of file [TimerCounter2.cpp](#).

```
00009 {  
00010     static TimerCounter2 l_instance(ar_channel,  
00011                                     ar_operationMode,  
00012                                     ar_clockSource,  
00013                                     ar_compareOutputMode);  
00014  
00015     return l_instance;  
00016  
00017  
00018 }
```

### 9.18.3.6 getOutputCompareRegister()

```
uint16_t core::TimerCounter2::getOutputCompareRegister (
    const channel & ar_channel ) const [override], [virtual]
```

Implements [core::TimerCounter](#).

Definition at line 199 of file [TimerCounter2.cpp](#).

```
00200 {
00201     switch (ar_channel)
00202     {
00203         case core::channel::A:
00204         {
00205             return OCR2A;
00206         }
00207         case core::channel::B:
00208         {
00209             return OCR2B;
00210         }
00211     }
00212 }
00213 }
```

References [core::A](#), and [core::B](#).

### 9.18.3.7 operator=()

```
const TimerCounter2& core::TimerCounter2::operator= (
    const TimerCounter2 & ) [private]
```

Override assign operator.

### 9.18.3.8 outputCompareMatchAServiceRoutine()

```
static void core::TimerCounter2::outputCompareMatchAServiceRoutine ( ) [static]
```

### 9.18.3.9 outputCompareMatchBServiceRoutine()

```
static void core::TimerCounter2::outputCompareMatchBServiceRoutine ( ) [static]
```

### 9.18.3.10 overflowServiceRoutine()

```
static void core::TimerCounter2::overflowServiceRoutine ( ) [static]
```

### 9.18.3.11 selectClockSource()

```
void core::TimerCounter2::selectClockSource (
    const clockSource & ar_clockSource ) [override], [virtual]
```

Implements [core::TimerCounter](#).

Definition at line 37 of file [TimerCounter2.cpp](#).

```
00038 {
00039     switch (ar_clockSource)
00040     {
00041         case core::clockSource::noClock:
00042         {
00043             m_clockPrescaler=0;
00044             m_clockSource=0;
00045             break;
00046         }
00047         case core::clockSource::PS_1:
00048         {
00049             m_clockPrescaler=1;
00050             m_clockSource=1;
00051             break;
00052         }
00053         case core::clockSource::PS_8:
00054         {
00055             m_clockPrescaler=8;
00056             m_clockSource=2;
00057             break;
00058         }
00059         case core::clockSource::PS_32:
00060         {
00061             m_clockPrescaler=32;
00062             m_clockSource=3;
00063             break;
00064         }
00065         case core::clockSource::PS_64:
00066         {
00067             m_clockPrescaler=64;
00068             m_clockSource=4;
00069             break;
00070         }
00071         case core::clockSource::PS_128:
00072         {
00073             m_clockPrescaler=128;
00074             m_clockSource=5;
00075             break;
00076         }
00077         case core::clockSource::PS_256:
00078         {
00079             m_clockPrescaler=256;
00080             m_clockSource=6;
00081             break;
00082         }
00083         case core::clockSource::PS_1024:
00084         {
00085             m_clockPrescaler=1024;
00086             m_clockSource=7;
00087             break;
00088         }
00089     }
00090 }
00091 }
```

References [core::noClock](#), [core::PS\\_1](#), [core::PS\\_1024](#), [core::PS\\_128](#), [core::PS\\_256](#), [core::PS\\_32](#), [core::PS\\_64](#), and [core::PS\\_8](#).

### 9.18.3.12 selectCompareOutputMode()

```
void core::TimerCounter2::selectCompareOutputMode (
    const channel & ar_channel,
    const compareOutputMode & ar_compareOutputMode ) [override], [virtual]
```

Implements [core::TimerCounter](#).

Definition at line 150 of file [TimerCounter2.cpp](#).

```
00151 {
00152     switch (ar_channel)
00153     {
00154         case core::channel::A:
00155         {
00156             TIMER2_SELECT_COM_CHANNEL_A(static_cast<uint8_t>(ar_compareOutputMode));
00157             break;
00158         }
00159         case core::channel::B:
00160         {
00161             TIMER2_SELECT_COM_CHANNEL_B(static_cast<uint8_t>(ar_compareOutputMode));
00162             break;
00163         }
00164     }
00165 }
00166
00167 }
```

References [core::A](#), [core::B](#), [TIMER2\\_SELECT\\_COM\\_CHANNEL\\_A](#), and [TIMER2\\_SELECT\\_COM\\_CHANNEL\\_B](#).

### 9.18.3.13 selectOperationMode()

```
void core::TimerCounter2::selectOperationMode (
    const operationMode & ar_operationMode ) [override], [virtual]
```

Implements [core::TimerCounter](#).

Definition at line 93 of file [TimerCounter2.cpp](#).

```
00094 {
00095     switch (ar_operationMode)
00096     {
00097         case core::operationMode::normal:
00098         {
00099             TIMER2_SELECT_OPERATION_MODE(0);
00100             break;
00101         }
00102         case core::operationMode::PWM_PC:
00103         {
00104             TIMER2_SELECT_OPERATION_MODE(1);
00105             break;
00106         }
00107         case core::operationMode::CTC_OCR:
00108         {
00109             TIMER2_SELECT_OPERATION_MODE(2);
00110             break;
00111         }
00112         case core::operationMode::fast_PWM:
00113         {
00114             TIMER2_SELECT_OPERATION_MODE(3);
00115             break;
00116         }
00117         case core::operationMode::PWM_PC_OCR:
00118         {
00119             TIMER2_SELECT_OPERATION_MODE(5);
00120             break;
00121         }
00122         case core::operationMode::fast_PWM_OCR:
00123         {
00124             TIMER2_SELECT_OPERATION_MODE(7);
00125             break;
00126         }
00127     }
00128 }
00129
00130 }
```

References [core::CTC\\_OCR](#), [core::fast\\_PWM](#), [core::fast\\_PWM\\_OCR](#), [core::normal](#), [core::PWM\\_PC](#), [core::PWM\\_PC\\_OCR](#), and [TIMER2\\_SELECT\\_OPERATION\\_MODE](#).

### 9.18.3.14 setCounter()

```
void core::TimerCounter2::setCounter (
    const uint16_t & ar_dataBuffer ) [override], [virtual]
```

Implements [core::TimerCounter](#).

Definition at line 171 of file [TimerCounter2.cpp](#).

```
00172 {
00173     TCNT2 = static_cast<uint8_t>(ar_dataBuffer);
00174 }
```

### 9.18.3.15 setOutputCompareRegister()

```
void core::TimerCounter2::setOutputCompareRegister (
    const channel & ar_channel,
    const uint16_t & ar_dataBuffer ) [override], [virtual]
```

Implements [core::TimerCounter](#).

Definition at line 181 of file [TimerCounter2.cpp](#).

```
00182 {
00183     switch (ar_channel)
00184     {
00185         case core::channel::A:
00186         {
00187             OCR2A = static_cast<uint8_t>(ar_dataBuffer);
00188             break;
00189         }
00190         case core::channel::B:
00191         {
00192             OCR2B = static_cast<uint8_t>(ar_dataBuffer);
00193             break;
00194         }
00195     }
00196 }
```

References [core::A](#), and [core::B](#).

### 9.18.3.16 start()

```
void core::TimerCounter2::start ( ) [override], [virtual]
```

Implements [core::TimerCounter](#).

Definition at line 133 of file [TimerCounter2.cpp](#).

```
00134 {
00135     TIMER2_SELECT_CLOCK_SOURCE(m_clockSource);
00136 }
```

References [TIMER2\\_SELECT\\_CLOCK\\_SOURCE](#).

### 9.18.3.17 stop()

```
void core::TimerCounter2::stop ( ) [override], [virtual]
```

Implements [core::TimerCounter](#).

Definition at line 144 of file [TimerCounter2.cpp](#).

```
00145 {  
00146     TIMER2_STOP;  
00147 }
```

References [TIMER2\\_STOP](#).

## 9.18.4 Member Data Documentation

### 9.18.4.1 \_\_externally\_visible\_\_

```
static void core::TimerCounter2::__externally_visible__
```

Definition at line 313 of file [TimerCounter2.h](#).

### 9.18.4.2 \_\_used\_\_

```
static void core::TimerCounter2::__used__
```

Definition at line 313 of file [TimerCounter2.h](#).

### 9.18.4.3 m\_clockPrescaler

```
uint16_t core::TimerCounter2::m_clockPrescaler [private]
```

Definition at line 338 of file [TimerCounter2.h](#).

### 9.18.4.4 m\_clockSource

```
uint8_t core::TimerCounter2::m_clockSource [private]
```

Definition at line 340 of file [TimerCounter2.h](#).

## 9.19 io::USART0 Class Reference

Class for handling [USART0](#).

```
#include <USART0.h>
```

### Public Member Functions

- void [setBaudRate](#) ()
- void [setTransmissionMode](#) (const [transmissionMode](#) &ar\_transMode)
- void [setCommunicationMode](#) (const [communicationMode](#) &ar\_comMode)
- void [setParityMode](#) (const [parityMode](#) &ar\_parityMode)
- void [setFrameSize](#) (const [frameSize](#) &ar\_frameSize)
- void [setStopBit](#) (const [stopBit](#) &ar\_stopBit)
- void [sendFrame](#) (const uint8\_t \*ap\_dataBuffer, const uint8\_t a\_size)
- void [sendString](#) (const char \*ap\_string)
- void [receiveString](#) (const char \*ap\_string)
- void [sendChar](#) (const uint8\_t &ar\_char)
- void [sendByte](#) (const uint8\_t &ar\_byte)
- void [sendWord](#) (const uint16\_t &ar\_word)
- void [sendLong](#) (const uint32\_t &ar\_long)
- void [receiveChar](#) (uint8\_t &ar\_char)
- void [receiveFrame](#) (uint8\_t \*ap\_dataBuffer, const uint8\_t a\_size)
- void [enableTransmitCompleteInterrupt](#) (const uint8\_t a\_enable)
- void [enableReceiveCompleteInterrupt](#) (const uint8\_t a\_enable)
- uint8\_t [frameError](#) ()
- uint8\_t [dataOverrun](#) ()
- uint8\_t [parityError](#) ()
- uint16\_t [getNumberBytesReceived](#) ()
- uint16\_t [getNumberBytesSent](#) ()
- uint8\_t [ready2Send](#) ()
- void [resetNumberBytesReceived](#) ()

### Static Public Member Functions

- static [USART0](#) & [getInstance](#) (const [transmissionMode](#) &ar\_transMode=[transmissionMode::async](#), const [communicationMode](#) &ar\_comMode=[communicationMode::duplex](#), const [frameSize](#) &ar\_frameSize=[frameSize::eightBits](#), const [stopBit](#) &ar\_stopBit=[stopBit::oneStopBit](#), const [parityMode](#) &ar\_parityMode=[parityMode::noParity](#))
- static void [enableDataRegisterEmptyInterrupt](#) (const uint8\_t a\_enable)
- static void [receiveCompleteServiceRoutine](#) () \_\_asm\_\_(STR(USART0\_RECEIVE\_COMPLETE\_INTERRUPT)) \_\_attribute\_\_((\_\_signal\_\_))
- static void [dataRegisterEmptyServiceRoutine](#) () \_\_asm\_\_(STR(USART0\_DATA\_REGISTER\_EMPTY\_INTERRUPT)) \_\_attribute\_\_((\_\_signal\_\_))
- static void [transmitCompleteServiceRoutine](#) () \_\_asm\_\_(STR(USART0\_TRANSMIT\_COMPLETE\_INTERRUPT)) \_\_attribute\_\_((\_\_signal\_\_))

### Public Attributes

- static void [\\_\\_used\\_\\_](#)
- static void [\\_\\_externally\\_visible\\_\\_](#)

## Private Member Functions

- `USART0` (const `transmissionMode` &ar\_transMode, const `communicationMode` &ar\_comMode, const `frameSize` &ar\_frameSize, const `stopBit` &ar\_stopBit, const `parityMode` &ar\_parityMode)
- `~USART0` ()
- `USART0` (const `USART0` &)
- const `USART0` & `operator=` (const `USART0` &)

## Static Private Attributes

- static volatile uint8\_t `m_status` = 0
- static const uint8\_t \* `mp_data2Send` = nullptr
- static uint8\_t \* `mp_data2Receive` = nullptr
- static uint16\_t `m_sizeData2Send` = 0
- static uint16\_t `m_sizeData2Receive` = 0
- static volatile uint16\_t `m_numberBytesReceived` = 0
- static volatile uint16\_t `m_numberBytesSent` = 0
- static volatile uint8\_t `m_ready2Send` = 1

### 9.19.1 Detailed Description

Class for handling `USART0`.

The class implements `USART0` operations

Author

Farid Oubbati ( [farid.oubbati@outlook.com](mailto:farid.oubbati@outlook.com))

Date

March 2018

Definition at line 98 of file `USART0.h`.

### 9.19.2 Constructor & Destructor Documentation

#### 9.19.2.1 `USART0()` [1/2]

```
io::USART0::USART0 (
    const transmissionMode & ar_transMode,
    const communicationMode & ar_comMode,
    const frameSize & ar_frameSize,
    const stopBit & ar_stopBit,
    const parityMode & ar_parityMode ) [private]
```

Constructor.

Initializes the `USART0` object



## Parameters

<i>ar_transMode</i>	Defines transmission mode
<i>ar_comMode</i>	Defines communication mode
<i>ar_frameSize</i>	Defines data frame size
<i>ar_stopBit</i>	Defines number of stop bits
<i>ar_parityMode</i>	Defines parity mode

Definition at line 43 of file [USART0.cpp](#).

```
00049 {
00050     core::MCU::enableUSART0(1);
00051     setBaudRate();
00052     setTransmissionMode(ar_transMode);
00053     setCommunicationMode(ar_comMode);
00054     setParityMode(ar_parityMode);
00055     setFrameSize(ar_frameSize);
00056     setStopBit(ar_stopBit);
00057     sei();
00058     enableReceiveCompleteInterrupt(1);
00059 }
```

References [core::MCU::enableUSART0\(\)](#).

### 9.19.2.2 ~USART0()

```
io::USART0::~~USART0 ( ) [private]
```

Destructor.

Definition at line 61 of file [USART0.cpp](#).

```
00062 {
00063
00064 }
```

### 9.19.2.3 USART0() [2/2]

```
io::USART0::USART0 (
    const USART0 & ) [private]
```

Overried Copy constructor.

## 9.19.3 Member Function Documentation

### 9.19.3.1 dataOverrun()

```
uint8_t io::USART0::dataOverrun ( )
```

Is there data overrun in received data

#### Returns

Status of received data

Definition at line 204 of file [USART0.cpp](#).

```
00205 {
00206     return (m_status & (1 << USART0_DATA_OVERRUN));
00207
00208 }
```

References [USART0\\_DATA\\_OVERRUN](#).

### 9.19.3.2 dataRegisterEmptyServiceRoutine()

```
void io::USART0::dataRegisterEmptyServiceRoutine ( ) [static]
```

Data register empty ISR

Definition at line 340 of file [USART0.cpp](#).

```
00341 {
00342
00343     if (m_sizeData2Send)
00344     {
00345         m_ready2Send = 0;
00346         USART0_DATA_REGISTER = *mp_data2Send++;
00347         m_sizeData2Send--;
00348         m_numberBytesSent++;
00349     }
00350     else
00351     {
00352         enableDataRegisterEmptyInterrupt (0);
00353         m_numberBytesSent = 0;
00354         m_ready2Send = 1;
00355     }
00356
00357 }
00358
00359 }
```

References [USART0\\_DATA\\_REGISTER](#).

### 9.19.3.3 enableDataRegisterEmptyInterrupt()

```
void io::USART0::enableDataRegisterEmptyInterrupt (
    const uint8_t a_enable ) [static]
```

Enable data register empty interrupt

#### Parameters

<i>ar_enable</i>	Indicates if interrupt is enabled
------------------	-----------------------------------

Definition at line 383 of file [USART0.cpp](#).

```
00384 {
00385     if (a_enable) {
00386         USART0_ENABLE_DATA_REGISTER_EMPTY_INTERRUPT;
00387     } else {
00388         USART0_DISABLE_DATA_REGISTER_EMPTY_INTERRUPT;
00389     }
00390 }
00391
00392 }
```

References [USART0\\_DISABLE\\_DATA\\_REGISTER\\_EMPTY\\_INTERRUPT](#), and [USART0\\_ENABLE\\_DATA\\_REGISTER\\_EMPTY\\_INTERRUPT](#).

#### 9.19.3.4 enableReceiveCompleteInterrupt()

```
void io::USART0::enableReceiveCompleteInterrupt (
    const uint8_t a_enable )
```

Enable receive complete interrupt

##### Parameters

<i>ar_enable</i>	Indicates if interrupt is enabled
------------------	-----------------------------------

Definition at line 372 of file [USART0.cpp](#).

```
00373 {
00374     if (a_enable) {
00375         USART0_ENABLE_RECEIVE_COMPLETE_INTERRUPT;
00376     } else {
00377         USART0_DISABLE_RECEIVE_COMPLETE_INTERRUPT;
00378     }
00379 }
00380
00381 }
```

References [USART0\\_DISABLE\\_RECEIVE\\_COMPLETE\\_INTERRUPT](#), and [USART0\\_ENABLE\\_RECEIVE\\_COMPLETE\\_INTERRUPT](#).

#### 9.19.3.5 enableTransmitCompleteInterrupt()

```
void io::USART0::enableTransmitCompleteInterrupt (
    const uint8_t a_enable )
```

Enable transmit complete interrupt

##### Parameters

<i>ar_enable</i>	Indicates if interrupt is enabled
------------------	-----------------------------------

Definition at line 362 of file [USART0.cpp](#).

```
00363 {
00364     if (a_enable) {
00365         USART0_ENABLE_TRANSMIT_COMPLETE_INTERRUPT;
00366     } else {
00367         USART0_DISABLE_TRANSMIT_COMPLETE_INTERRUPT;
00368     }
00369 }
```

```
00370 }
```

References [USART0\\_DISABLE\\_TRANSMIT\\_COMPLETE\\_INTERRUPT](#), and [USART0\\_ENABLE\\_TRANSMIT\\_COMPLETE\\_INTERRUPT](#)

### 9.19.3.6 frameError()

```
uint8_t io::USART0::frameError ( )
```

Is there frame error in received data

#### Returns

Status of received data

Definition at line 198 of file [USART0.cpp](#).

```
00199 {
00200     return (m_status & (1 << USART0_FRAME_ERROR));
00201
00202 }
```

References [USART0\\_FRAME\\_ERROR](#).

### 9.19.3.7 getInstance()

```
io::USART0 & io::USART0::getInstance (
    const transmissionMode & ar_transMode = transmissionMode::async,
    const communicationMode & ar_comMode = communicationMode::duplex,
    const frameSize & ar_frameSize = frameSize::eightBits,
    const stopBit & ar_stopBit = stopBit::oneStopBit,
    const parityMode & ar_parityMode = parityMode::noParity ) [static]
```

Create a single instance of the [USART0](#) object

#### Parameters

<i>ar_transMode</i>	Defines transmission mode
<i>ar_comMode</i>	Defines communication mode
<i>ar_frameSize</i>	Defines data frame size
<i>ar_stopBit</i>	Defines number of stop bits
<i>ar_parityMode</i>	Defines parity mode

Definition at line 26 of file [USART0.cpp](#).

```
00031 {
00032     static USART0 l_instance(ar_transMode,
00033                             ar_comMode,
00034                             ar_frameSize,
00035                             ar_stopBit,
00036                             ar_parityMode);
00037
00038     return l_instance;
00039 }
```

### 9.19.3.8 getNumberBytesReceived()

```
uint16_t io::USART0::getNumberBytesReceived ( )
```

Get number of bytes received

#### Returns

Number of received bytes

Definition at line 404 of file [USART0.cpp](#).

```
00405 {  
00406     return m_numberBytesReceived;  
00407 }
```

### 9.19.3.9 getNumberBytesSent()

```
uint16_t io::USART0::getNumberBytesSent ( )
```

Get number of bytes sent

#### Returns

Number of transmitted bytes

Definition at line 399 of file [USART0.cpp](#).

```
00400 {  
00401     return m_numberBytesSent;  
00402 }
```

### 9.19.3.10 operator=()

```
const USART0& io::USART0::operator= (  
    const USART0 & ) [private]
```

Override assign operator.

### 9.19.3.11 parityError()

```
uint8_t io::USART0::parityError ( )
```

Is there parity error in received data

#### Returns

Status of received data

Definition at line 210 of file [USART0.cpp](#).

```
00211 {  
00212     return (m_status & (1 << USART0\_PARITY\_ERROR));  
00213  
00214 }
```

References [USART0\\_PARITY\\_ERROR](#).

### 9.19.3.12 ready2Send()

```
uint8_t io::USART0::ready2Send ( )
```

Is ready to send

#### Returns

Sender status

Definition at line 414 of file [USART0.cpp](#).

```
00415 {
00416     return m_ready2Send;
00417 }
00418 }
```

### 9.19.3.13 receiveChar()

```
void io::USART0::receiveChar (
    uint8_t & ar_char )
```

Receive character

#### Parameters

<i>ar_char</i>	Defines charcter to be received
----------------	---------------------------------

Definition at line 293 of file [USART0.cpp](#).

```
00294 {
00295     m_sizeData2Receive = 1;
00296     mp_data2Receive = &ar_char;
00297 }
```

### 9.19.3.14 receiveCompleteServiceRoutine()

```
void io::USART0::receiveCompleteServiceRoutine ( ) [static]
```

Receive complete ISR

Definition at line 315 of file [USART0.cpp](#).

```
00316 {
00317     static volatile uint8_t *lp_dataReceived = mp_data2Receive;
00318     static uint16_t l_dataSize = m_sizeData2Receive;
00319     m_status = USART0_CONTROL_STATUS_REGISTER;
00320
00321     if (l_dataSize)
00322     {
00323         *lp_dataReceived++ = USART0_DATA_REGISTER;
00324         l_dataSize--;
00325         m_numberBytesReceived++;
00326     }
00327 }
```

```

00332         else
00333         {
00334             l_dataSize = m_sizeData2Receive;
00335             lp_dataReceived = mp_data2Receive;
00336         }
00337
00338     }

```

References [USART0\\_CONTROL\\_STATUS\\_REGISTER](#), and [USART0\\_DATA\\_REGISTER](#).

### 9.19.3.15 receiveFrame()

```

void io::USART0::receiveFrame (
    uint8_t * ap_dataBuffer,
    const uint8_t a_size )

```

Receive data frame

#### Parameters

<i>ap_dataBuffer</i>	Defines pointer to receiver buffer
<i>a_size</i>	Defines size of receiver buffer
<i>a_ready2Receive</i>	Indicates if chip ready to receive data

Definition at line 300 of file [USART0.cpp](#).

```

00301 {
00302     m_sizeData2Receive = a_size;
00303     mp_data2Receive = ap_dataBuffer;
00304     // TODO: to be implemented
00305
00306
00307 }

```

### 9.19.3.16 receiveString()

```

void io::USART0::receiveString (
    const char * ap_string )

```

Receive string

#### Parameters

<i>ap_string</i>	Defines pointer to string
------------------	---------------------------

Definition at line 309 of file [USART0.cpp](#).

```

00310 {
00311     // TODO: to be implemented
00312 }

```

### 9.19.3.17 resetNumberBytesReceived()

```
void io::USART0::resetNumberBytesReceived ( )
```

Reset number of bytes received

Definition at line 409 of file [USART0.cpp](#).

```
00410 {
00411     m_numberBytesReceived = 0;
00412 }
```

### 9.19.3.18 sendByte()

```
void io::USART0::sendByte (
    const uint8_t & ar_byte )
```

Transmit byte

Parameters

<i>ar_byte</i>	Defines byte to be sent
----------------	-------------------------

Definition at line 235 of file [USART0.cpp](#).

```
00236 {
00237     static uint8_t l_byte2Send[3];
00238     l_byte2Send[0] = '0' + (ar_byte / 100);
00239     l_byte2Send[1] = '0' + ((ar_byte / 10) % 10);
00240     l_byte2Send[2] = '0' + (ar_byte % 10);
00241
00242     while (!ready2Send()){};
00243     m_sizeData2Send = 3;
00244     mp_data2Send = l_byte2Send;
00245     enableDataRegisterEmptyInterrupt(1);
00246
00247 }
```

### 9.19.3.19 sendChar()

```
void io::USART0::sendChar (
    const uint8_t & ar_char )
```

Transmit character

Parameters

<i>ar_char</i>	Defines character to be sent
----------------	------------------------------

Definition at line 284 of file [USART0.cpp](#).

```
00285 {
00286     while (!ready2Send()){};
00287     m_sizeData2Send = 1;
00288     mp_data2Send = &ar_char;
00289     enableDataRegisterEmptyInterrupt(1);
00290 }
```



```
00291 }
```

### 9.19.3.20 sendFrame()

```
void io::USART0::sendFrame (
    const uint8_t * ap_dataBuffer,
    const uint8_t a_size )
```

Transmit data frame

#### Parameters

<i>ap_dataBuffer</i>	Defines pointer to transmitter buffer
<i>a_size</i>	Defines size of transmitter buffer

Definition at line 216 of file [USART0.cpp](#).

```
00217 {
00218     while (!ready2Send()){};
00219     m_sizeData2Send = a_size;
00220     mp_data2Send = ap_dataBuffer;
00221     enableDataRegisterEmptyInterrupt(1);
00222 }
```

### 9.19.3.21 sendLong()

```
void io::USART0::sendLong (
    const uint32_t & ar_long )
```

Transmit long word

#### Parameters

<i>ar_long</i>	Defines long word to be sent
----------------	------------------------------

Definition at line 248 of file [USART0.cpp](#).

```
00249 {
00250     static uint8_t l_word2Send[10];
00251     l_word2Send[0] = '0' + (ar_long / 1000000000);
00252     l_word2Send[1] = '0' + ((ar_long / 100000000) % 10);
00253     l_word2Send[2] = '0' + ((ar_long / 10000000) % 10);
00254     l_word2Send[3] = '0' + ((ar_long / 1000000) % 10);
00255     l_word2Send[4] = '0' + ((ar_long / 100000) % 10);
00256     l_word2Send[5] = '0' + ((ar_long / 10000) % 10);
00257     l_word2Send[6] = '0' + ((ar_long / 1000) % 10);
00258     l_word2Send[7] = '0' + ((ar_long / 100) % 10);
00259     l_word2Send[8] = '0' + ((ar_long / 10) % 10);
00260     l_word2Send[9] = '0' + (ar_long % 10);
00261
00262     while (!ready2Send()){};
00263     m_sizeData2Send = 10;
00264     mp_data2Send = l_word2Send;
00265     enableDataRegisterEmptyInterrupt(1);
00266
00267 }
```

### 9.19.3.22 sendString()

```
void io::USART0::sendString (
    const char * ap_string )
```

Transmit string

#### Parameters

<i>ap_string</i>	Defines pointer to string
------------------	---------------------------

Definition at line 225 of file [USART0.cpp](#).

```
00226 {
00227     while (!ready2Send()){};
00228     m_sizeData2Send = strlen(ap_string);
00229     mp_data2Send = reinterpret_cast<const uint8_t*>(ap_string);
00230     enableDataRegisterEmptyInterrupt(1);
00231
00232 }
```

### 9.19.3.23 sendWord()

```
void io::USART0::sendWord (
    const uint16_t & ar_word )
```

Transmit word

#### Parameters

<i>ar_word</i>	Defines word to be sent
----------------	-------------------------

Definition at line 268 of file [USART0.cpp](#).

```
00269 {
00270     static uint8_t l_word2Send[5];
00271     l_word2Send[0] = '0' + (ar_word / 10000);
00272     l_word2Send[1] = '0' + ((ar_word / 1000) % 10);
00273     l_word2Send[2] = '0' + ((ar_word / 100) % 10);
00274     l_word2Send[3] = '0' + ((ar_word / 10) % 10);
00275     l_word2Send[4] = '0' + (ar_word % 10);
00276
00277     while (!ready2Send()){};
00278     m_sizeData2Send = 5;
00279     mp_data2Send = l_word2Send;
00280     enableDataRegisterEmptyInterrupt(1);
00281
00282 }
```

### 9.19.3.24 setBaudRate()

```
void io::USART0::setBaudRate ( )
```

Set baud rate.

Definition at line 66 of file [USART0.cpp](#).

```
00067 {
```

```

00068     USART0_SET_BAUDRATE_HIGH_REGISTER;
00069     USART0_SET_BAUDRATE_LOW_REGISTER;
00070
00071 }

```

References [USART0\\_SET\\_BAUDRATE\\_HIGH\\_REGISTER](#), and [USART0\\_SET\\_BAUDRATE\\_LOW\\_REGISTER](#).

### 9.19.3.25 setCommunicationMode()

```

void io::USART0::setCommunicationMode (
    const communicationMode & ar_comMode )

```

Set communication mode

#### Parameters

<i>ar_comMode</i>	Defines communication mode
-------------------	----------------------------

Definition at line 98 of file [USART0.cpp](#).

```

00099 {
00100     switch (ar_comMode)
00101     {
00102         case communicationMode::duplex:
00103         {
00104             USART0_ENABLE_TRANSMITTER;
00105             USART0_ENABLE_RECEIVER;
00106             break;
00107         }
00108         case communicationMode::receive:
00109         {
00110             USART0_ENABLE_RECEIVER;
00111             USART0_DISABLE_TRANSMITTER;
00112             break;
00113         }
00114         case communicationMode::transmit:
00115         {
00116             USART0_ENABLE_TRANSMITTER;
00117             USART0_DISABLE_RECEIVER;
00118             break;
00119         }
00120     }
00121
00122 }

```

References [io::duplex](#), [io::receive](#), [io::transmit](#), [USART0\\_DISABLE\\_RECEIVER](#), [USART0\\_DISABLE\\_TRANSMITTER](#), [USART0\\_ENABLE\\_RECEIVER](#), and [USART0\\_ENABLE\\_TRANSMITTER](#).

### 9.19.3.26 setFrameSize()

```

void io::USART0::setFrameSize (
    const frameSize & ar_frameSize )

```

Set data frame size

#### Parameters

<i>ar_frameSize</i>	Defines data frame size
---------------------	-------------------------

Definition at line 147 of file USART0.cpp.

```
00148 {
00149     switch (ar_frameSize)
00150     {
00151         case frameSize::eightBits:
00152         {
00153             USART0_SET_8BIT_FRAME_SIZE;
00154             break;
00155         }
00156         case frameSize::sevenBits:
00157         {
00158             USART0_SET_7BIT_FRAME_SIZE;
00159             break;
00160         }
00161         case frameSize::sixBits:
00162         {
00163             USART0_SET_6BIT_FRAME_SIZE;
00164             break;
00165         }
00166         case frameSize::fiveBits:
00167         {
00168             USART0_SET_5BIT_FRAME_SIZE;
00169             break;
00170         }
00171         case frameSize::neineBits:
00172         {
00173             USART0_SET_9BIT_FRAME_SIZE;
00174             break;
00175         }
00176     }
00177 }
00178 }
```

References [io::eightBits](#), [io::fiveBits](#), [io::neineBits](#), [io::sevenBits](#), [io::sixBits](#), [USART0\\_SET\\_5BIT\\_FRAME\\_SIZE](#), [USART0\\_SET\\_6BIT\\_FRAME\\_SIZE](#), [USART0\\_SET\\_7BIT\\_FRAME\\_SIZE](#), [USART0\\_SET\\_8BIT\\_FRAME\\_SIZE](#), and [USART0\\_SET\\_9BIT\\_FRAME\\_SIZE](#).

### 9.19.3.27 setParityMode()

```
void io::USART0::setParityMode (
    const parityMode & ar_parityMode )
```

Set parity mode in data frame

#### Parameters

<i>ar_parityMode</i>	Defines parity mode
----------------------	---------------------

Definition at line 124 of file USART0.cpp.

```
00125 {
00126     switch (ar_parityMode)
00127     {
00128         case parityMode::noParity:
00129         {
00130             USART0_DISABLE_PARITY_MODE;
00131             break;
00132         }
00133         case parityMode::evenParity:
00134         {
00135             USART0_ENABLE_EVEN_PARITY_MODE;
00136             break;
00137         }
00138         case parityMode::oddParity:
00139         {
00140             USART0_ENABLE_ODD_PARITY_MODE;
00141             break;
00142         }
00143     }
00144 }
00145 }
```

References [io::evenParity](#), [io::noParity](#), [io::oddParity](#), [USART0\\_DISABLE\\_PARITY\\_MODE](#), [USART0\\_ENABLE\\_EVEN\\_PARITY\\_MODE](#) and [USART0\\_ENABLE\\_ODD\\_PARITY\\_MODE](#).

### 9.19.3.28 setStopBit()

```
void io::USART0::setStopBit (
    const stopBit & ar_stopBit )
```

Set number of stop bits in data frame

#### Parameters

<i>ar_stopBit</i>	Defines number of stop bits
-------------------	-----------------------------

Definition at line 179 of file [USART0.cpp](#).

```
00180 {
00181     switch (ar_stopBit)
00182     {
00183         case stopBit::oneStopBit:
00184         {
00185             USART0_SET_ONE_STOP_BIT;
00186             break;
00187         }
00188         case stopBit::twoStopBits:
00189         {
00190             USART0_SET_TWO_STOP_BITS;
00191             break;
00192         }
00193     }
00194 }
00195 }
```

References [io::oneStopBit](#), [io::twoStopBits](#), [USART0\\_SET\\_ONE\\_STOP\\_BIT](#), and [USART0\\_SET\\_TWO\\_STOP\\_BITS](#).

### 9.19.3.29 setTransmissionMode()

```
void io::USART0::setTransmissionMode (
    const transmissionMode & ar_transMode )
```

Set transmission mode

#### Parameters

<i>ar_transMode</i>	Defines transmission mode
---------------------	---------------------------

Definition at line 73 of file [USART0.cpp](#).

```
00074 {
00075     switch (ar_transMode)
00076     {
00077         case transmissionMode::async:
00078         {
00079             USART0_ENABLE_ASYNC_TRANSMISSION_MODE;
00080             break;
00081         }
00082         case transmissionMode::sync:
00083         {
```

```

00084         USART0_DISABLE_DOUBLE_SPEED_MODE;
00085         USART0_ENABLE_SYNC_TRANSMISSION_MODE;
00086         break;
00087     }
00088     case transmissionMode::masterSPI:
00089     {
00090         USART0_ENABLE_MASTER_SPI_MODE;
00091         break;
00092     }
00093 }
00094
00095
00096 }

```

References [io::async](#), [io::masterSPI](#), [io::sync](#), [USART0\\_DISABLE\\_DOUBLE\\_SPEED\\_MODE](#), [USART0\\_ENABLE\\_ASYNC\\_TRANSMISSION\\_MODE](#), [USART0\\_ENABLE\\_MASTER\\_SPI\\_MODE](#), and [USART0\\_ENABLE\\_SYNC\\_TRANSMISSION\\_MODE](#).

### 9.19.3.30 transmitCompleteServiceRoutine()

```
void io::USART0::transmitCompleteServiceRoutine ( ) [static]
```

Transmit complete ISR

Definition at line 394 of file [USART0.cpp](#).

```

00395 {
00396
00397 }

```

## 9.19.4 Member Data Documentation

### 9.19.4.1 \_\_externally\_visible\_\_

```
static void io::USART0::__externally_visible__
```

Definition at line 283 of file [USART0.h](#).

### 9.19.4.2 \_\_used\_\_

```
static void io::USART0::__used__
```

Definition at line 283 of file [USART0.h](#).

### 9.19.4.3 m\_numberBytesReceived

```
volatile uint16_t io::USART0::m_numberBytesReceived = 0 [static], [private]
```

Number of received bytes

Definition at line 338 of file [USART0.h](#).

#### 9.19.4.4 m\_numberBytesSent

```
volatile uint16_t io::USART0::m_numberBytesSent = 0 [static], [private]
```

Number of trasnmitted bytes

Definition at line 340 of file [USART0.h](#).

#### 9.19.4.5 m\_ready2Send

```
volatile uint8_t io::USART0::m_ready2Send = 1 [static], [private]
```

Ready to send flag

Definition at line 342 of file [USART0.h](#).

#### 9.19.4.6 m\_sizeData2Receive

```
uint16_t io::USART0::m_sizeData2Receive = 0 [static], [private]
```

Size of data to be received

Definition at line 336 of file [USART0.h](#).

#### 9.19.4.7 m\_sizeData2Send

```
uint16_t io::USART0::m_sizeData2Send = 0 [static], [private]
```

Size of data to be transmitted

Definition at line 334 of file [USART0.h](#).

#### 9.19.4.8 m\_status

```
volatile uint8_t io::USART0::m_status = 0 [static], [private]
```

Status of received data

Author

Farid Oubbati ( [farid.oubbati@outlook.com](mailto:farid.oubbati@outlook.com) )

Date

March 2018

See [USART0.h](#) for a description of this code

Definition at line 328 of file [USART0.h](#).

#### 9.19.4.9 mp\_data2Receive

```
uint8_t * io::USART0::mp_data2Receive = nullptr [static], [private]
```

Pointer to receiver buffer

Definition at line 332 of file [USART0.h](#).

#### 9.19.4.10 mp\_data2Send

```
const uint8_t * io::USART0::mp_data2Send = nullptr [static], [private]
```

Pointer to transmitter buffer

Definition at line 330 of file [USART0.h](#).

## 9.20 core::WatchdogTimer Class Reference

```
#include <WatchdogTimer.h>
```

### Public Member Functions

- void [selectTimeOut](#) (const [timeOut](#) &ar\_timeOut)
- void [reset](#) ()
- void [start](#) (const [operationMode](#) &ar\_operationMode)
- void [start](#) (const [operationMode](#) &ar\_operationMode, const [timeOut](#) &ar\_timeOut)
- void [stop](#) ()

### Static Public Member Functions

- static [WatchdogTimer](#) & [getInstance](#) ()
- static void [timeOutServiceRoutine](#) () \_\_asm\_\_(STR(WATCHDOG\_TIMEOUT\_INTERRUPT)) \_\_attribute\_\_((\_\_signal\_\_

### Public Attributes

- static void [\\_\\_used\\_\\_](#)
- static void [\\_\\_externally\\_visible\\_\\_](#)

### Private Member Functions

- [WatchdogTimer](#) ()
- [~WatchdogTimer](#) ()
- [WatchdogTimer](#) (const [WatchdogTimer](#) &)
- const [WatchdogTimer](#) & [operator=](#) (const [WatchdogTimer](#) &)



## Private Attributes

- `uint8_t m_timeOut`
- `uint8_t m_operationMode`

### 9.20.1 Detailed Description

Definition at line 84 of file [WatchdogTimer.h](#).

### 9.20.2 Constructor & Destructor Documentation

#### 9.20.2.1 WatchdogTimer() [1/2]

```
core::WatchdogTimer::WatchdogTimer ( ) [private]
```

Definition at line 11 of file [WatchdogTimer.cpp](#).

```
00012 {  
00013     sei();  
00014     stop();  
00015 }
```

#### 9.20.2.2 ~WatchdogTimer()

```
core::WatchdogTimer::~~WatchdogTimer ( ) [private]
```

Destructor.

Definition at line 17 of file [WatchdogTimer.cpp](#).

```
00018 {  
00019  
00020 }
```

#### 9.20.2.3 WatchdogTimer() [2/2]

```
core::WatchdogTimer::WatchdogTimer (  
    const WatchdogTimer & ) [private]
```

Overried Copy constructor.

### 9.20.3 Member Function Documentation

### 9.20.3.1 getInstance()

```
core::WatchdogTimer & core::WatchdogTimer::getInstance ( ) [static]
```

Definition at line 4 of file [WatchdogTimer.cpp](#).

```
00005 {  
00006     static WatchdogTimer l_instance;  
00007     return l_instance;  
00008  
00009 }
```

### 9.20.3.2 operator=()

```
const WatchdogTimer& core::WatchdogTimer::operator= (  
    const WatchdogTimer & ) [private]
```

Override assign operator.

### 9.20.3.3 reset()

```
void core::WatchdogTimer::reset ( )
```

Definition at line 33 of file [WatchdogTimer.cpp](#).

```
00034 {  
00035     wdt_reset();  
00036  
00037 }
```

### 9.20.3.4 selectTimeOut()

```
void core::WatchdogTimer::selectTimeOut (  
    const timeOut & ar_timeOut )
```

Definition at line 22 of file [WatchdogTimer.cpp](#).

```
00023 {  
00024     m_timeOut = static_cast<uint8_t>(ar_timeOut);  
00025     m_timeOut = static_cast<uint8_t>((m_timeOut & 7) | ((m_timeOut & 8) << 2));  
00026     cli();  
00027     wdt_reset();  
00028     WATCHDOG_SELECT_TIMEOUT(m_timeOut);  
00029     sei();  
00030  
00031 }
```

References [WATCHDOG\\_SELECT\\_TIMEOUT](#).

**9.20.3.5 start() [1/2]**

```
void core::WatchdogTimer::start (
    const operationMode & ar_operationMode )
```

Definition at line 39 of file [WatchdogTimer.cpp](#).

```
00040 {
00041     m\_operationMode = static\_cast<uint8_t>(ar_operationMode);
00042     m\_operationMode = static\_cast<uint8_t>(((m\_operationMode & 1) << 6) | ((m\_operationMode & 2) << 3
    ));
00043     cli();
00044     wdt_reset();
00045     WATCHDOG\_START(m\_operationMode,m\_timeOut);
00046     sei();
00047 }
00048 }
```

References [WATCHDOG\\_START](#).

**9.20.3.6 start() [2/2]**

```
void core::WatchdogTimer::start (
    const operationMode & ar_operationMode,
    const timeOut & ar_timeOut )
```

Definition at line 50 of file [WatchdogTimer.cpp](#).

```
00051 {
00052     m\_timeOut = static\_cast<uint8_t>(ar_timeOut);
00053     m\_timeOut = static\_cast<uint8_t>((m\_timeOut & 7) | ((m\_timeOut & 8) << 2));
00054     m\_operationMode = static\_cast<uint8_t>(ar_operationMode);
00055     m\_operationMode = static\_cast<uint8_t>(((m\_operationMode & 1) << 6) | ((m\_operationMode & 2) << 3
    ));
00056     cli();
00057     wdt_reset();
00058     WATCHDOG\_START(m\_operationMode,m\_timeOut);
00059     sei();
00060 }
00061 }
```

References [WATCHDOG\\_START](#).

**9.20.3.7 stop()**

```
void core::WatchdogTimer::stop ( )
```

Definition at line 63 of file [WatchdogTimer.cpp](#).

```
00064 {
00065     cli();
00066     wdt_reset();
00067     WATCHDOG\_STOP;
00068     sei();
00069 }
00070 }
```

References [WATCHDOG\\_STOP](#).

### 9.20.3.8 timeOutServiceRoutine()

```
static void core::WatchdogTimer::timeOutServiceRoutine ( ) [static]
```

## 9.20.4 Member Data Documentation

### 9.20.4.1 \_\_externally\_visible\_\_

```
void core::WatchdogTimer::__externally_visible__
```

Definition at line 102 of file [WatchdogTimer.h](#).

### 9.20.4.2 \_\_used\_\_

```
void core::WatchdogTimer::__used__
```

Definition at line 102 of file [WatchdogTimer.h](#).

### 9.20.4.3 m\_operationMode

```
uint8_t core::WatchdogTimer::m_operationMode [private]
```

Definition at line 125 of file [WatchdogTimer.h](#).

### 9.20.4.4 m\_timeOut

```
uint8_t core::WatchdogTimer::m_timeOut [private]
```

Definition at line 123 of file [WatchdogTimer.h](#).

## Chapter 10

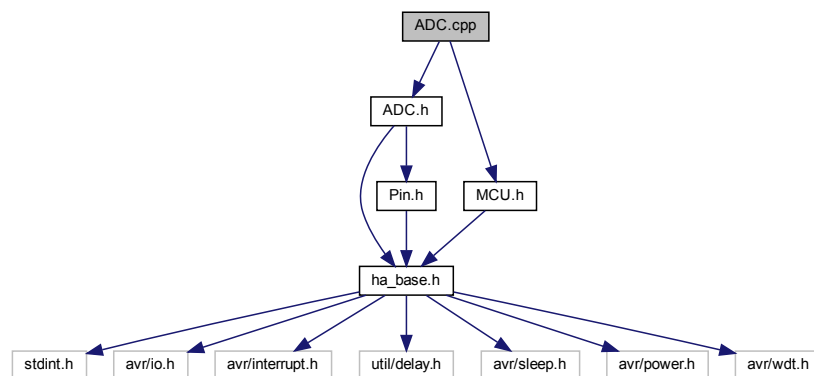
# File Documentation

### 10.1 ADC.cpp File Reference

```
#include "ADC.h"
```

```
#include "MCU.h"
```

Include dependency graph for ADC.cpp:



### 10.2 ADC.cpp

```
00001 #include "ADC.h"
00002 #include "MCU.h"
00003
00004 volatile uint16_t core::ADConverter::m_conversionResult = 0;
00005 uint8_t core::ADConverter::m_resolution = 10;
00006
00007
00008 core::ADConverter& core::ADConverter::getInstance(const referenceVoltage &ar_refVoltage,
00009                                                    const clockPrescaler& ar_clockPrescaler,
00010                                                    const autoTriggerSource& ar_autoTriggerSource,
00011                                                    const io::Pin &ar_pin)
00012 {
00013
00014     static ADConverter l_instance(ar_refVoltage,
00015                                   ar_clockPrescaler,
00016                                   ar_autoTriggerSource,
00017                                   ar_pin);
00018 }
```

```

00019     return l_instance;
00020 }
00021
00022 core::ADConverter::ADConverter(const referenceVoltage &ar_refVoltage,
00023                               const clockPrescaler& ar_clockPrescaler,
00024                               const autoTriggerSource& ar_autoTriggerSource,
00025                               const io::Pin &ar_pin)
00026 {
00027     core::MCU::enableADC(1);
00028     selectAnalogInput(ar_pin);
00029     selectReferenceVoltage(ar_refVoltage);
00030     selectClockPrescaler(ar_clockPrescaler);
00031     enableAutoTrigger(1);
00032     selectAutoTriggerSource(ar_autoTriggerSource);
00033     sei();
00034     enableConversionCompleteInterrupt(1);
00035 }
00036
00037 }
00038
00039
00040 core::ADConverter::~ADConverter()
00041 {
00042 }
00043 }
00044
00045
00046 void core::ADConverter::selectReferenceVoltage(const referenceVoltage& ar_refVoltage)
00047 {
00048     ADC_SELECT_REF_VOLTAGE(static_cast<uint8_t>(ar_refVoltage));
00049 }
00050 }
00051
00052 void core::ADConverter::selectAnalogInput(io::Pin a_pin)
00053 {
00054     a_pin.toInput(0);
00055     ADC_SELECT_ANALOG_INPUT(a_pin.getPinNumber());
00056     ADC_DISABLE_DIGITAL_INPUT_REGISTER(a_pin.getPinNumber());
00057 }
00058 }
00059
00060 void core::ADConverter::start()
00061 {
00062     ADC_ENABLE;
00063     ADC_START_CONVERSION;
00064 }
00065
00066 void core::ADConverter::stop()
00067 {
00068     ADC_STOP_CONVERSION;
00069     ADC_DISABLE;
00070 }
00071
00072 void core::ADConverter::enableAutoTrigger(const uint8_t a_enable)
00073 {
00074     if (a_enable) {
00075         ADC_ENABLE_AUTOTRIGGER;
00076     } else {
00077         ADC_DISABLE_AUTOTRIGGER;
00078     }
00079 }
00080 }
00081 }
00082
00083 void core::ADConverter::enableConversionCompleteInterrupt(const uint8_t a_enable)
00084 {
00085     if (a_enable) {
00086         ADC_ENABLE_CONVERSION_COMPLETE_INTERRUPT;
00087     } else {
00088         ADC_DISABLE_CONVERSION_COMPLETE_INTERRUPT;
00089     }
00090 }
00091 }
00092 }
00093
00094 void core::ADConverter::selectClockPrescaler(const clockPrescaler& ar_clockPrescaler)
00095 {
00096     ADC_SELECT_CLOCK_PRESCALER(static_cast<uint8_t>(ar_clockPrescaler));
00097 }
00098 }
00099
00100
00101 void core::ADConverter::selectAutoTriggerSource(const autoTriggerSource& ar_autoTriggerSource)
00102 {
00103     ADC_SELECT_AUTO_TRIGGER_SOURCE(static_cast<uint8_t>(ar_autoTriggerSource));
00104 }
00105

```

```
00106 }
00107
00108 void core::ADConverter::conversionCompleteServiceRoutine()
00109 {
00110     static uint32_t l_resultData = 0;
00111     static uint16_t l_resultDataIndex = 0;
00112
00113     switch (m_resolution)
00114     {
00115     case 8:
00116     {
00117         m_conversionResult = ADC » 8;
00118         break;
00119     }
00120     case 9:
00121     {
00122         m_conversionResult = ADC » 7;
00123         break;
00124     }
00125     case 10:
00126     {
00127         m_conversionResult = ADC;
00128         break;
00129     }
00130     case 11:
00131     {
00132         if (l_resultDataIndex < 4)
00133         {
00134             l_resultData += ADC;
00135             l_resultDataIndex++;
00136         }
00137         else
00138         {
00139             m_conversionResult = l_resultData » 1;
00140             l_resultData = 0;
00141             l_resultDataIndex = 0;
00142         }
00143         break;
00144     }
00145     case 12:
00146     {
00147         if (l_resultDataIndex < 16)
00148         {
00149             l_resultData += ADC;
00150             l_resultDataIndex++;
00151         }
00152         else
00153         {
00154             m_conversionResult = l_resultData » 2;
00155             l_resultData = 0;
00156             l_resultDataIndex = 0;
00157         }
00158         break;
00159     }
00160     case 13:
00161     {
00162         if (l_resultDataIndex < 64)
00163         {
00164             l_resultData += ADC;
00165             l_resultDataIndex++;
00166         }
00167         else
00168         {
00169             m_conversionResult = l_resultData » 3;
00170             l_resultData = 0;
00171             l_resultDataIndex = 0;
00172         }
00173         break;
00174     }
00175     case 14:
00176     {
00177         if (l_resultDataIndex < 256)
00178         {
00179             l_resultData += ADC;
00180             l_resultDataIndex++;
00181         }
00182         else
00183         {
00184             l_resultData = 0;
00185             l_resultDataIndex = 0;
00186         }
00187     }
00188 }
```

```

00193         m_conversionResult = l_resultData » 4;
00194         l_resultData = 0;
00195         l_resultDataIndex = 0;
00196     }
00197     break;
00198 }
00199 case 15:
00200 {
00201     if (l_resultDataIndex < 1024)
00202     {
00203         l_resultData += ADC;
00204         l_resultDataIndex++;
00205     }
00206     else
00207     {
00208         m_conversionResult = l_resultData » 5;
00209         l_resultData = 0;
00210         l_resultDataIndex = 0;
00211     }
00212     break;
00213 }
00214 }
00215 case 16:
00216 {
00217     if (l_resultDataIndex < 4096)
00218     {
00219
00220         l_resultData += ADC;
00221         l_resultDataIndex++;
00222     }
00223     else
00224     {
00225         m_conversionResult = l_resultData » 6;
00226         l_resultData = 0;
00227         l_resultDataIndex = 0;
00228     }
00229     break;
00230 }
00231 }
00232 }
00233 }
00234 }
00235 }
00236 }
00237 }
00238 }
00239 }
00240 }
00241 }
00242 }
00243 uint16_t core::ADConverter::getConversionResult( const resolution& ar_resolution)
00244 {
00245     switch (ar_resolution)
00246     {
00247     case core::resolution::res_8bit:
00248     {
00249         ADC_ADJUST_RESULT_LEFT;
00250         m_resolution = 8;
00251         break;
00252     }
00253     case core::resolution::res_9bit:
00254     {
00255         ADC_ADJUST_RESULT_LEFT;
00256         m_resolution = 9;
00257         break;
00258     }
00259     case core::resolution::res_10bit:
00260     {
00261         ADC_ADJUST_RESULT_RIGHT;
00262         m_resolution = 10;
00263         break;
00264     }
00265     case core::resolution::res_11bit:
00266     {
00267         m_resolution = 11;
00268         break;
00269     }
00270     case core::resolution::res_12bit:
00271     {
00272         m_resolution = 12;
00273         break;
00274     }
00275     case core::resolution::res_13bit:
00276     {

```



```

00280         m_resolution = 13;
00281         break;
00282     }
00283     case core::resolution::res_14bit:
00284     {
00285         m_resolution = 14;
00286         break;
00287     }
00288     case core::resolution::res_15bit:
00289     {
00290         m_resolution = 15;
00291         break;
00292     }
00293     case core::resolution::res_16bit:
00294     {
00295         m_resolution = 16;
00296         break;
00297     }
00298 }
00299
00300 return m_conversionResult;
00301
00302
00303 }

```

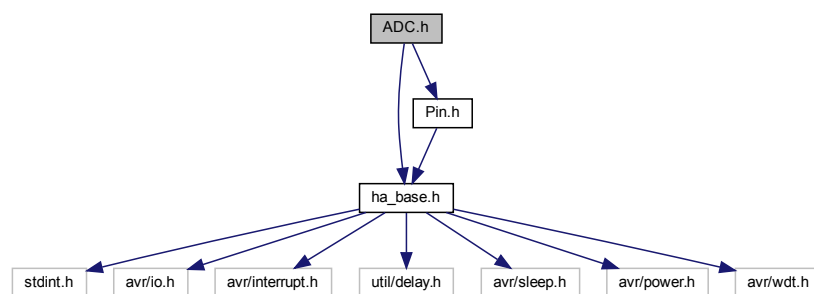
## 10.3 ADC.h File Reference

```

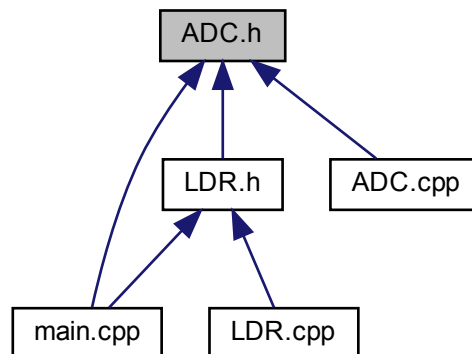
#include "ha_base.h"
#include "Pin.h"

```

Include dependency graph for ADC.h:



This graph shows which files directly or indirectly include this file:



## Classes

- class [core::ADConverter](#)  
*Class for handling ADC.*

## Namespaces

- [core](#)  
*AVR chip internal core components.*

## Enumerations

- enum [core::resolution](#) : uint8\_t {  
core::resolution::res\_8bit=0, core::resolution::res\_9bit, core::resolution::res\_10bit, core::resolution::res\_11bit,  
core::resolution::res\_12bit, core::resolution::res\_13bit, core::resolution::res\_14bit, core::resolution::res\_15bit,  
core::resolution::res\_16bit }  
*ADC resolution.*
- enum [core::referenceVoltage](#) : uint8\_t { [core::referenceVoltage::AREF](#) =0, [core::referenceVoltage::AVCC](#),  
[core::referenceVoltage::internal](#) }  
*ADC referencev voltage.*
- enum [core::clockPrescaler](#) : uint8\_t {  
core::clockPrescaler::PS\_2= 1, core::clockPrescaler::PS\_4, core::clockPrescaler::PS\_8, core::clockPrescaler::PS\_16,  
core::clockPrescaler::PS\_32, core::clockPrescaler::PS\_64, core::clockPrescaler::PS\_128 }  
*ADC clock prescaler.*
- enum [core::autoTriggerSource](#) : uint8\_t {  
core::autoTriggerSource::freeRunning, core::autoTriggerSource::analogComparator, core::autoTriggerSource::extInterrupt,  
core::autoTriggerSource::timer0CompareA,  
core::autoTriggerSource::timer0Overflow, core::autoTriggerSource::timer1CompareB, core::autoTriggerSource::timer1Overflow,  
core::autoTriggerSource::timer1Capture }  
*ADC autoTrigger source.*

## 10.4 ADC.h

```

00001 #ifndef ADC_H
00002 #define ADC_H
00003 #include "ha_base.h"
00004 #include "Pin.h"
00005
00014 namespace core
00015 {
00024 enum class resolution : uint8_t {
00025     res_8bit=0,
00026     res_9bit,
00027     res_10bit,
00028     res_11bit,
00029     res_12bit,
00030     res_13bit,
00031     res_14bit,
00032     res_15bit,
00033     res_16bit
00034 };
00035
00044 enum class referenceVoltage : uint8_t {
00045     AREF=0,
00046     AVCC,
00047     internal
00048 };
00049
00050
00059 enum class clockPrescaler : uint8_t {
00060     PS_2 = 1,
00061     PS_4,
00062     PS_8,
00063     PS_16,
00064     PS_32,
00065     PS_64,
00066     PS_128
00067 };
00076 enum class autoTriggerSource : uint8_t {
00077     freeRunning,
00078     analogComparator,
00079     extInterrupt,
00080     timer0CompareA,
00081     timer0Overflow,
00082     timer1CompareB,
00083     timer1Overflow,
00084     timer1Capture
00085 };
00094 class ADConverter
00095 {
00096 public:
00105     static ADConverter& getInstance(const referenceVoltage& ar_refVoltage = referenceVoltage::AVCC,
00106                                     const clockPrescaler& ar_clockPrescaler = clockPrescaler::PS_128,
00107                                     const autoTriggerSource& ar_autoTriggerSource =
00108                                     autoTriggerSource::freeRunning,
00109                                     const io::Pin &ar_pin = io::Pin(0,io::PortC));
00111     void start();
00114     void stop();
00119     void selectReferenceVoltage(const referenceVoltage& ar_refVoltage);
00124     void selectAnalogInput(io::Pin a_pin);
00129     void selectClockPrescaler(const clockPrescaler& ar_clockPrescaler);
00134     void enableConversionCompleteInterrupt(const uint8_t a_enable);
00139     void enableAutoTrigger(const uint8_t a_enable);
00144     void selectAutoTriggerSource(const autoTriggerSource& ar_autoTriggerSource);
00150     uint16_t getConversionResult(const resolution& ar_resolution = resolution::res_10bit);
00153     static void conversionCompleteServiceRoutine() __asm__(STR(ADC_CONVERSION_COMPLETE_INTERRUPT))
00154         __attribute__((__signal__, __used__, __externally_visible__));
00155
00156 protected:
00157
00158
00159 private:
00160     ADConverter(const referenceVoltage& ar_refVoltage,
00171                 const clockPrescaler& ar_clockPrescaler,
00172                 const autoTriggerSource &ar_autoTriggerSource,
00173                 const io::Pin &ar_pin);
00174
00177     ~ADConverter();
00178
00181     ADConverter(const ADConverter&);
00182
00185     const ADConverter& operator=(const ADConverter&);
00186
00187     static volatile uint16_t m_conversionResult;
00189     static uint8_t m_resolution;
00191 };

```

```

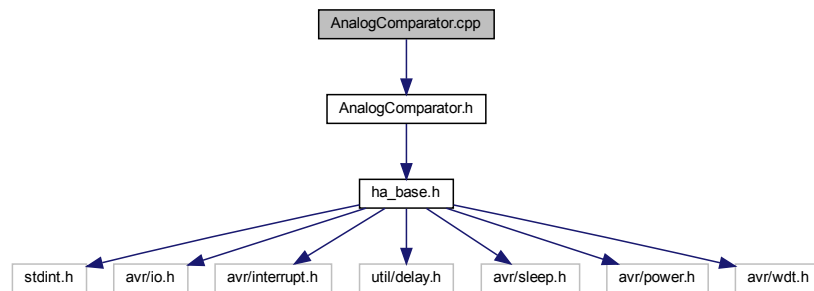
00192
00193 }
00194 #endif

```

## 10.5 AnalogComparator.cpp File Reference

```
#include "AnalogComparator.h"
```

Include dependency graph for AnalogComparator.cpp:



## 10.6 AnalogComparator.cpp

```

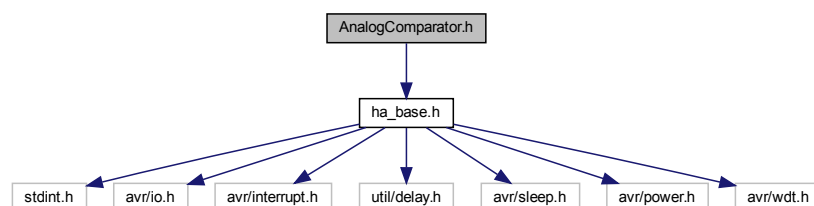
00001 #include "AnalogComparator.h"
00002
00003

```

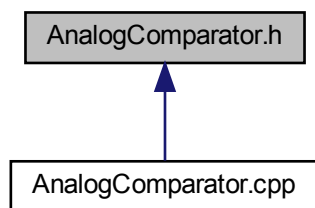
## 10.7 AnalogComparator.h File Reference

```
#include "ha_base.h"
```

Include dependency graph for AnalogComparator.h:



This graph shows which files directly or indirectly include this file:



## Classes

- class [core::AnalogComparator](#)

## Namespaces

- [core](#)  
*AVR chip internal core components.*

## Macros

- #define [ANALOG\\_COMARATOR\\_H](#)

### 10.7.1 Macro Definition Documentation

#### 10.7.1.1 ANALOG\_COMARATOR\_H

```
#define ANALOG_COMARATOR_H
```

Definition at line 13 of file [AnalogComparator.h](#).

## 10.8 AnalogComparator.h

```

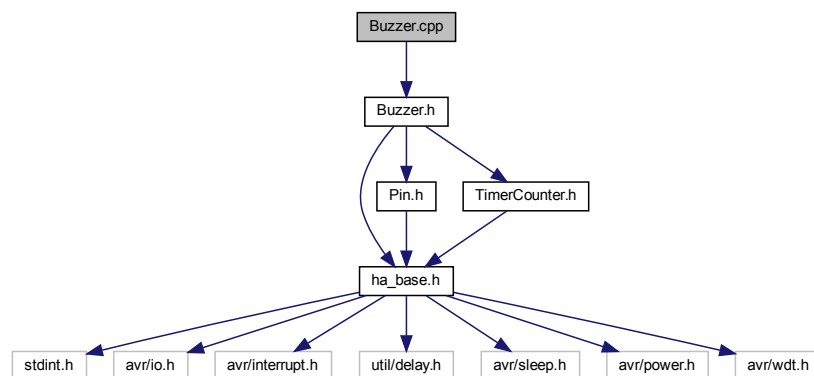
00001
00012 #ifndef ANALOG_COMPARATOR_H
00013 #define ANALOG_COMARATOR_H
00014 #include "ha_base.h"
00015
00016
00017
00018 namespace core
00019 {
00020
00021
00022
00023
00024 class AnalogComparator
00025 {
00026
00027 public:
00028
00029
00030
00031
00032 protected:
00033
00034 private:
00035
00036
00037
00038 };
00039
00040 }
00041
00042
00043 #endif

```

## 10.9 Buzzer.cpp File Reference

```
#include "Buzzer.h"
```

Include dependency graph for Buzzer.cpp:



### 10.10 Buzzer.cpp

```

00001
00010 #include "Buzzer.h"
00011
00012
00013 component::Buzzer::Buzzer(const io::Pin &ar_pin)
00014     : m_pin(ar_pin)
00015 {

```

```

00016     m_pin.toOutput();
00017
00018 }
00019
00020 component::Buzzer::~Buzzer()
00021 {
00022 }
00023 }
00024
00025 void component::Buzzer::buzz(const uint16_t &ar_period_us , const uint16_t &ar_duration_us)
00026 {
00027
00028     for (uint16_t i = 0; i < ar_duration_us; i += ar_period_us)
00029     {
00030         // for loop with variable delay selects the pitch
00031         for (uint16_t j = 0; j < ar_period_us; j++)
00032         {
00033             _delay_us(1);
00034         }
00035         m_pin.toggle();
00036     }
00037     m_pin.setLow();
00038 }
00039 }
00040
00041
00042 void component::Buzzer::pause(uint16_t a_duration_us)
00043 {
00044     do {
00045         _delay_us(1);
00046     } while (--a_duration_us);
00047 }

```

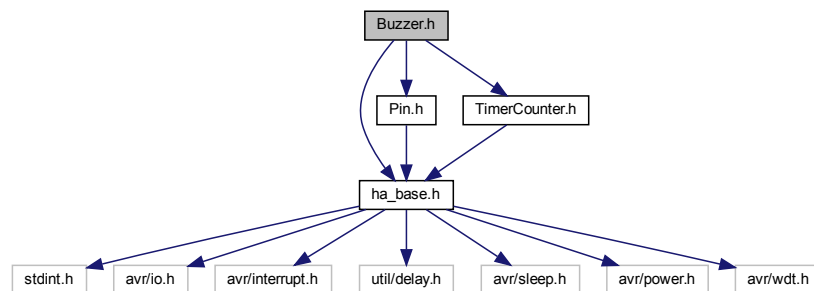
## 10.11 Buzzer.h File Reference

```

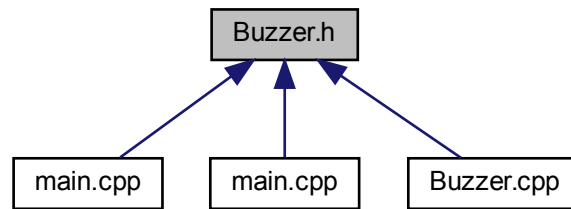
#include "ha_base.h"
#include "Pin.h"
#include "TimerCounter.h"

```

Include dependency graph for Buzzer.h:



This graph shows which files directly or indirectly include this file:



## Classes

- class `component::Buzzer`  
Class for handling a *Buzzer* component.

## Namespaces

- `component`  
AVR chip external components.

## 10.12 Buzzer.h

```

00001
00002 // #include "Buzzer.h"
00003 // #include "TimerCounter0.h"
00004 // #include "buzzer_pitches_8bit.h"
00005
00006
00007 // #define BUZZER 6
00008
00009
00010 // int main(void) {
00011
00012 //     // Init
00013 //     // instantiate the Buzzer object
00014 //     component::Buzzer Buzzer(io::Pin(BUZZER,io::PortD));
00015
00016
00017 //     // instantiate the TimerCounter0 object
00018 //     core::TimerCounter0 &myTimerCounter0 = core::TimerCounter0::getInstance();
00019
00020 //     // notes in the melody:
00021 //     const uint16_t notes[] = {C2, E2, G2, C3};
00022
00023
00024 //     for (uint8_t i = 0; i < sizeof (notes)/sizeof (uint16_t); i++)
00025 //     {
00026
00027 //         Buzzer.buzz(notes[i],200);
00028
00029 //     }
00030
00031 //     _delay_ms(1000);
00032 //     _delay_ms(1000);
00033 //     _delay_ms(1000);
00034
00035 //     for (uint8_t i = 0; i < sizeof (notes)/sizeof (uint16_t); i++)
00036 //     {
00037 //         Buzzer.buzz(myTimerCounter0,notes[i],200);

```



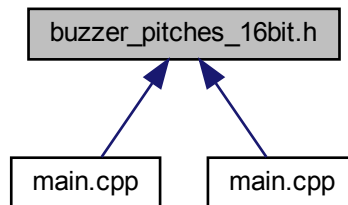
```

00038 //    }
00039
00040
00041 //    // ----- Event loop ----- //
00042 //    while (1) {
00043
00044
00045 //    }
00046 //    return 0;
00047 // }
00048
00049 #ifndef BUZZER_H
00050 #define BUZZER_H
00051 #include "ha_base.h"
00052 #include "Pin.h"
00053 #include "TimerCounter.h"
00054
00063 namespace component
00064 {
00065
00074 class Buzzer
00075 {
00076
00077 public:
00085     Buzzer(const io::Pin &ar_pin);
00089     ~Buzzer();
00090
00096     void buzz(const uint16_t &ar_period_us , const uint16_t &ar_duration_us);
00097
00102     void pause(uint16_t a_duration_us);
00103
00113     template<typename TC>
00114     void buzz(TC &ar_timerCounter,
00115               const uint16_t &ar_period_us,
00116               uint16_t &ar_duration_ms,
00117               const core::channel &ar_channel=core::channel::A,
00118               const core::clockSource &ar_clockSource= core::clockSource::PS_64
00119               )
00120     {
00121         ar_timerCounter.selectOperationMode(core::operationMode::CTC_OCR);
00122         ar_timerCounter.selectCompareOutputMode(ar_channel, core::compareOutputMode::toggle);
00123         ar_timerCounter.setCounter(0);
00124         ar_timerCounter.setOutputCompareRegister(ar_channel, ar_period_us);
00125         // start timer
00126         ar_timerCounter.start();
00127         // wait for the pitch duration
00128         while (ar_duration_ms) {
00129             _delay_ms(1);
00130             ar_duration_ms--;
00131         }
00132         // no buzz
00133         ar_timerCounter.selectCompareOutputMode(ar_channel, core::compareOutputMode::normal);
00134         ar_timerCounter.stop();
00135     }
00136
00137
00138 protected:
00139
00140 private:
00141
00142     io::Pin m_pin;
00145 };
00146 }
00147
00148
00149 #endif

```

## 10.13 buzzer\_pitches\_16bit.h File Reference

This graph shows which files directly or indirectly include this file:



### Macros

- #define C0 25000
- #define Cx0 23597
- #define D0 22272
- #define Dx0 21022
- #define E0 19843
- #define F0 18729
- #define Fx0 17678
- #define G0 16685
- #define Gx0 15749
- #define A0 14865
- #define Ax0 14031
- #define B0 13243
- #define C1 12500
- #define Cx1 11798
- #define D1 11136
- #define Dx1 10511
- #define E1 9921
- #define F1 9364
- #define Fx1 8839
- #define G1 8343
- #define Gx1 7875
- #define A1 7433
- #define Ax1 7015
- #define B1 6622
- #define C2 6250
- #define Cx2 5899
- #define D2 5568
- #define Dx2 5256
- #define E2 4961
- #define F2 4682
- #define Fx2 4419
- #define G2 4171
- #define Gx2 3937

- #define [A2](#) 3716
- #define [Ax2](#) 3508
- #define [B2](#) 3311
- #define [C3](#) 3125
- #define [Cx3](#) 2950
- #define [D3](#) 2784
- #define [Dx3](#) 2628
- #define [E3](#) 2480
- #define [F3](#) 2341
- #define [Fx3](#) 2210
- #define [G3](#) 2086
- #define [Gx3](#) 1969
- #define [A3](#) 1858
- #define [Ax3](#) 1754
- #define [B3](#) 1655
- #define [C4](#) 1562
- #define [Cx4](#) 1474
- #define [D4](#) 1392
- #define [Dx4](#) 1313
- #define [E4](#) 1240
- #define [F4](#) 1170
- #define [Fx4](#) 1105
- #define [G4](#) 1043
- #define [Gx4](#) 984
- #define [A4](#) 929
- #define [Ax4](#) 877
- #define [B4](#) 827
- #define [C5](#) 781
- #define [Cx5](#) 737
- #define [D5](#) 696
- #define [Dx5](#) 657
- #define [E5](#) 620
- #define [F5](#) 585
- #define [Fx5](#) 552
- #define [G5](#) 521
- #define [Gx5](#) 492
- #define [A5](#) 464
- #define [Ax5](#) 438
- #define [B5](#) 414
- #define [C6](#) 390
- #define [Cx6](#) 368
- #define [D6](#) 347
- #define [Dx6](#) 328
- #define [E6](#) 310
- #define [F6](#) 292
- #define [Fx6](#) 276
- #define [G6](#) 260
- #define [Gx6](#) 246
- #define [A6](#) 232
- #define [Ax6](#) 219
- #define [B6](#) 207
- #define [C7](#) 195
- #define [Cx7](#) 184
- #define [D7](#) 174
- #define [Dx7](#) 164

- `#define E7` 155
- `#define F7` 146
- `#define Fx7` 138
- `#define G7` 130
- `#define Gx7` 123
- `#define A7` 116
- `#define Ax7` 109
- `#define B7` 103

## 10.13.1 Macro Definition Documentation

### 10.13.1.1 A0

```
#define A0 14865
```

Definition at line 21 of file [buzzer\\_pitches\\_16bit.h](#).

### 10.13.1.2 A1

```
#define A1 7433
```

Definition at line 33 of file [buzzer\\_pitches\\_16bit.h](#).

### 10.13.1.3 A2

```
#define A2 3716
```

Definition at line 45 of file [buzzer\\_pitches\\_16bit.h](#).

### 10.13.1.4 A3

```
#define A3 1858
```

Definition at line 57 of file [buzzer\\_pitches\\_16bit.h](#).

#### 10.13.1.5 A4

```
#define A4 929
```

Definition at line 69 of file [buzzer\\_pitches\\_16bit.h](#).

#### 10.13.1.6 A5

```
#define A5 464
```

Definition at line 81 of file [buzzer\\_pitches\\_16bit.h](#).

#### 10.13.1.7 A6

```
#define A6 232
```

Definition at line 93 of file [buzzer\\_pitches\\_16bit.h](#).

#### 10.13.1.8 A7

```
#define A7 116
```

Definition at line 105 of file [buzzer\\_pitches\\_16bit.h](#).

#### 10.13.1.9 Ax0

```
#define Ax0 14031
```

Definition at line 22 of file [buzzer\\_pitches\\_16bit.h](#).

#### 10.13.1.10 Ax1

```
#define Ax1 7015
```

Definition at line 34 of file [buzzer\\_pitches\\_16bit.h](#).

**10.13.1.11 Ax2**

```
#define Ax2 3508
```

Definition at line [46](#) of file [buzzer\\_pitches\\_16bit.h](#).

**10.13.1.12 Ax3**

```
#define Ax3 1754
```

Definition at line [58](#) of file [buzzer\\_pitches\\_16bit.h](#).

**10.13.1.13 Ax4**

```
#define Ax4 877
```

Definition at line [70](#) of file [buzzer\\_pitches\\_16bit.h](#).

**10.13.1.14 Ax5**

```
#define Ax5 438
```

Definition at line [82](#) of file [buzzer\\_pitches\\_16bit.h](#).

**10.13.1.15 Ax6**

```
#define Ax6 219
```

Definition at line [94](#) of file [buzzer\\_pitches\\_16bit.h](#).

**10.13.1.16 Ax7**

```
#define Ax7 109
```

Definition at line [106](#) of file [buzzer\\_pitches\\_16bit.h](#).

**10.13.1.17 B0**

```
#define B0 13243
```

Definition at line 23 of file [buzzer\\_pitches\\_16bit.h](#).

**10.13.1.18 B1**

```
#define B1 6622
```

Definition at line 35 of file [buzzer\\_pitches\\_16bit.h](#).

**10.13.1.19 B2**

```
#define B2 3311
```

Definition at line 47 of file [buzzer\\_pitches\\_16bit.h](#).

**10.13.1.20 B3**

```
#define B3 1655
```

Definition at line 59 of file [buzzer\\_pitches\\_16bit.h](#).

**10.13.1.21 B4**

```
#define B4 827
```

Definition at line 71 of file [buzzer\\_pitches\\_16bit.h](#).

**10.13.1.22 B5**

```
#define B5 414
```

Definition at line 83 of file [buzzer\\_pitches\\_16bit.h](#).

**10.13.1.23 B6**

```
#define B6 207
```

Definition at line 95 of file [buzzer\\_pitches\\_16bit.h](#).

**10.13.1.24 B7**

```
#define B7 103
```

Definition at line 107 of file [buzzer\\_pitches\\_16bit.h](#).

**10.13.1.25 C0**

```
#define C0 25000
```

Definition at line 12 of file [buzzer\\_pitches\\_16bit.h](#).

**10.13.1.26 C1**

```
#define C1 12500
```

Definition at line 24 of file [buzzer\\_pitches\\_16bit.h](#).

**10.13.1.27 C2**

```
#define C2 6250
```

Definition at line 36 of file [buzzer\\_pitches\\_16bit.h](#).

**10.13.1.28 C3**

```
#define C3 3125
```

Definition at line 48 of file [buzzer\\_pitches\\_16bit.h](#).



#### 10.13.1.29 C4

```
#define C4 1562
```

Definition at line 60 of file [buzzer\\_pitches\\_16bit.h](#).

#### 10.13.1.30 C5

```
#define C5 781
```

Definition at line 72 of file [buzzer\\_pitches\\_16bit.h](#).

#### 10.13.1.31 C6

```
#define C6 390
```

Definition at line 84 of file [buzzer\\_pitches\\_16bit.h](#).

#### 10.13.1.32 C7

```
#define C7 195
```

Definition at line 96 of file [buzzer\\_pitches\\_16bit.h](#).

#### 10.13.1.33 Cx0

```
#define Cx0 23597
```

Definition at line 13 of file [buzzer\\_pitches\\_16bit.h](#).

#### 10.13.1.34 Cx1

```
#define Cx1 11798
```

Definition at line 25 of file [buzzer\\_pitches\\_16bit.h](#).

**10.13.1.35 Cx2**

```
#define Cx2 5899
```

Definition at line 37 of file [buzzer\\_pitches\\_16bit.h](#).

**10.13.1.36 Cx3**

```
#define Cx3 2950
```

Definition at line 49 of file [buzzer\\_pitches\\_16bit.h](#).

**10.13.1.37 Cx4**

```
#define Cx4 1474
```

Definition at line 61 of file [buzzer\\_pitches\\_16bit.h](#).

**10.13.1.38 Cx5**

```
#define Cx5 737
```

Definition at line 73 of file [buzzer\\_pitches\\_16bit.h](#).

**10.13.1.39 Cx6**

```
#define Cx6 368
```

Definition at line 85 of file [buzzer\\_pitches\\_16bit.h](#).

**10.13.1.40 Cx7**

```
#define Cx7 184
```

Definition at line 97 of file [buzzer\\_pitches\\_16bit.h](#).

#### 10.13.1.41 D0

```
#define D0 22272
```

Definition at line 14 of file [buzzer\\_pitches\\_16bit.h](#).

#### 10.13.1.42 D1

```
#define D1 11136
```

Definition at line 26 of file [buzzer\\_pitches\\_16bit.h](#).

#### 10.13.1.43 D2

```
#define D2 5568
```

Definition at line 38 of file [buzzer\\_pitches\\_16bit.h](#).

#### 10.13.1.44 D3

```
#define D3 2784
```

Definition at line 50 of file [buzzer\\_pitches\\_16bit.h](#).

#### 10.13.1.45 D4

```
#define D4 1392
```

Definition at line 62 of file [buzzer\\_pitches\\_16bit.h](#).

#### 10.13.1.46 D5

```
#define D5 696
```

Definition at line 74 of file [buzzer\\_pitches\\_16bit.h](#).

**10.13.1.47 D6**

```
#define D6 347
```

Definition at line 86 of file [buzzer\\_pitches\\_16bit.h](#).

**10.13.1.48 D7**

```
#define D7 174
```

Definition at line 98 of file [buzzer\\_pitches\\_16bit.h](#).

**10.13.1.49 Dx0**

```
#define Dx0 21022
```

Definition at line 15 of file [buzzer\\_pitches\\_16bit.h](#).

**10.13.1.50 Dx1**

```
#define Dx1 10511
```

Definition at line 27 of file [buzzer\\_pitches\\_16bit.h](#).

**10.13.1.51 Dx2**

```
#define Dx2 5256
```

Definition at line 39 of file [buzzer\\_pitches\\_16bit.h](#).

**10.13.1.52 Dx3**

```
#define Dx3 2628
```

Definition at line 51 of file [buzzer\\_pitches\\_16bit.h](#).

#### 10.13.1.53 Dx4

```
#define Dx4 1313
```

Definition at line 63 of file [buzzer\\_pitches\\_16bit.h](#).

#### 10.13.1.54 Dx5

```
#define Dx5 657
```

Definition at line 75 of file [buzzer\\_pitches\\_16bit.h](#).

#### 10.13.1.55 Dx6

```
#define Dx6 328
```

Definition at line 87 of file [buzzer\\_pitches\\_16bit.h](#).

#### 10.13.1.56 Dx7

```
#define Dx7 164
```

Definition at line 99 of file [buzzer\\_pitches\\_16bit.h](#).

#### 10.13.1.57 E0

```
#define E0 19843
```

Definition at line 16 of file [buzzer\\_pitches\\_16bit.h](#).

#### 10.13.1.58 E1

```
#define E1 9921
```

Definition at line 28 of file [buzzer\\_pitches\\_16bit.h](#).

**10.13.1.59 E2**

```
#define E2 4961
```

Definition at line 40 of file [buzzer\\_pitches\\_16bit.h](#).

**10.13.1.60 E3**

```
#define E3 2480
```

Definition at line 52 of file [buzzer\\_pitches\\_16bit.h](#).

**10.13.1.61 E4**

```
#define E4 1240
```

Definition at line 64 of file [buzzer\\_pitches\\_16bit.h](#).

**10.13.1.62 E5**

```
#define E5 620
```

Definition at line 76 of file [buzzer\\_pitches\\_16bit.h](#).

**10.13.1.63 E6**

```
#define E6 310
```

Definition at line 88 of file [buzzer\\_pitches\\_16bit.h](#).

**10.13.1.64 E7**

```
#define E7 155
```

Definition at line 100 of file [buzzer\\_pitches\\_16bit.h](#).

**10.13.1.65 F0**

```
#define F0 18729
```

Definition at line 17 of file [buzzer\\_pitches\\_16bit.h](#).

**10.13.1.66 F1**

```
#define F1 9364
```

Definition at line 29 of file [buzzer\\_pitches\\_16bit.h](#).

**10.13.1.67 F2**

```
#define F2 4682
```

Definition at line 41 of file [buzzer\\_pitches\\_16bit.h](#).

**10.13.1.68 F3**

```
#define F3 2341
```

Definition at line 53 of file [buzzer\\_pitches\\_16bit.h](#).

**10.13.1.69 F4**

```
#define F4 1170
```

Definition at line 65 of file [buzzer\\_pitches\\_16bit.h](#).

**10.13.1.70 F5**

```
#define F5 585
```

Definition at line 77 of file [buzzer\\_pitches\\_16bit.h](#).

**10.13.1.71 F6**

```
#define F6 292
```

Definition at line 89 of file [buzzer\\_pitches\\_16bit.h](#).

**10.13.1.72 F7**

```
#define F7 146
```

Definition at line 101 of file [buzzer\\_pitches\\_16bit.h](#).

**10.13.1.73 Fx0**

```
#define Fx0 17678
```

Definition at line 18 of file [buzzer\\_pitches\\_16bit.h](#).

**10.13.1.74 Fx1**

```
#define Fx1 8839
```

Definition at line 30 of file [buzzer\\_pitches\\_16bit.h](#).

**10.13.1.75 Fx2**

```
#define Fx2 4419
```

Definition at line 42 of file [buzzer\\_pitches\\_16bit.h](#).

**10.13.1.76 Fx3**

```
#define Fx3 2210
```

Definition at line 54 of file [buzzer\\_pitches\\_16bit.h](#).



**10.13.1.77 Fx4**

```
#define Fx4 1105
```

Definition at line 66 of file [buzzer\\_pitches\\_16bit.h](#).

**10.13.1.78 Fx5**

```
#define Fx5 552
```

Definition at line 78 of file [buzzer\\_pitches\\_16bit.h](#).

**10.13.1.79 Fx6**

```
#define Fx6 276
```

Definition at line 90 of file [buzzer\\_pitches\\_16bit.h](#).

**10.13.1.80 Fx7**

```
#define Fx7 138
```

Definition at line 102 of file [buzzer\\_pitches\\_16bit.h](#).

**10.13.1.81 G0**

```
#define G0 16685
```

Definition at line 19 of file [buzzer\\_pitches\\_16bit.h](#).

**10.13.1.82 G1**

```
#define G1 8343
```

Definition at line 31 of file [buzzer\\_pitches\\_16bit.h](#).

**10.13.1.83 G2**

```
#define G2 4171
```

Definition at line 43 of file [buzzer\\_pitches\\_16bit.h](#).

**10.13.1.84 G3**

```
#define G3 2086
```

Definition at line 55 of file [buzzer\\_pitches\\_16bit.h](#).

**10.13.1.85 G4**

```
#define G4 1043
```

Definition at line 67 of file [buzzer\\_pitches\\_16bit.h](#).

**10.13.1.86 G5**

```
#define G5 521
```

Definition at line 79 of file [buzzer\\_pitches\\_16bit.h](#).

**10.13.1.87 G6**

```
#define G6 260
```

Definition at line 91 of file [buzzer\\_pitches\\_16bit.h](#).

**10.13.1.88 G7**

```
#define G7 130
```

Definition at line 103 of file [buzzer\\_pitches\\_16bit.h](#).

**10.13.1.89 Gx0**

```
#define Gx0 15749
```

Definition at line 20 of file [buzzer\\_pitches\\_16bit.h](#).

**10.13.1.90 Gx1**

```
#define Gx1 7875
```

Definition at line 32 of file [buzzer\\_pitches\\_16bit.h](#).

**10.13.1.91 Gx2**

```
#define Gx2 3937
```

Definition at line 44 of file [buzzer\\_pitches\\_16bit.h](#).

**10.13.1.92 Gx3**

```
#define Gx3 1969
```

Definition at line 56 of file [buzzer\\_pitches\\_16bit.h](#).

**10.13.1.93 Gx4**

```
#define Gx4 984
```

Definition at line 68 of file [buzzer\\_pitches\\_16bit.h](#).

**10.13.1.94 Gx5**

```
#define Gx5 492
```

Definition at line 80 of file [buzzer\\_pitches\\_16bit.h](#).

**10.13.1.95 Gx6**

```
#define Gx6 246
```

Definition at line 92 of file [buzzer\\_pitches\\_16bit.h](#).

**10.13.1.96 Gx7**

```
#define Gx7 123
```

Definition at line 104 of file [buzzer\\_pitches\\_16bit.h](#).

**10.14 buzzer\_pitches\_16bit.h**

```
00001 /* Scale in the key of 1/25000 */
00002 /*
00003     These are 1/2 periods (us) -- if you delay this long,
00004     then toggle the speaker pin, you'll get approximate
00005     pitches.
00006
00007     This is the 16-bit version. The pitches get less accurate
00008     as they get higher.
00009
00010 */
00011
00012 #define C0      25000 // freq = 1000000/ (2*25000) = 20hz
00013 #define Cx0     23597
00014 #define D0      22272
00015 #define Dx0     21022
00016 #define E0      19843
00017 #define F0      18729
00018 #define Fx0     17678
00019 #define G0      16685
00020 #define Gx0     15749
00021 #define A0      14865
00022 #define Ax0     14031
00023 #define B0      13243
00024 #define C1      12500
00025 #define Cx1     11798
00026 #define D1      11136
00027 #define Dx1     10511
00028 #define E1      9921
00029 #define F1      9364
00030 #define Fx1     8839
00031 #define G1      8343
00032 #define Gx1     7875
00033 #define A1      7433
00034 #define Ax1     7015
00035 #define B1      6622
00036 #define C2      6250
00037 #define Cx2     5899
00038 #define D2      5568
00039 #define Dx2     5256
00040 #define E2      4961
00041 #define F2      4682
00042 #define Fx2     4419
00043 #define G2      4171
00044 #define Gx2     3937
00045 #define A2      3716
00046 #define Ax2     3508
00047 #define B2      3311
00048 #define C3      3125
00049 #define Cx3     2950
00050 #define D3      2784
00051 #define Dx3     2628
00052 #define E3      2480
00053 #define F3      2341
00054 #define Fx3     2210
00055 #define G3      2086
00056 #define Gx3     1969
00057 #define A3      1858
00058 #define Ax3     1754
```

```

00059 #define B3      1655
00060 #define C4      1562
00061 #define Cx4    1474
00062 #define D4      1392
00063 #define Dx4     1313
00064 #define E4      1240
00065 #define F4      1170
00066 #define Fx4     1105
00067 #define G4      1043
00068 #define Gx4     984
00069 #define A4      929
00070 #define Ax4     877
00071 #define B4      827
00072 #define C5      781
00073 #define Cx5     737
00074 #define D5      696
00075 #define Dx5     657
00076 #define E5      620
00077 #define F5      585
00078 #define Fx5     552
00079 #define G5      521
00080 #define Gx5     492
00081 #define A5      464
00082 #define Ax5     438
00083 #define B5      414
00084 #define C6      390
00085 #define Cx6     368
00086 #define D6      347
00087 #define Dx6     328
00088 #define E6      310
00089 #define F6      292
00090 #define Fx6     276
00091 #define G6      260
00092 #define Gx6     246
00093 #define A6      232
00094 #define Ax6     219
00095 #define B6      207
00096 #define C7      195
00097 #define Cx7     184
00098 #define D7      174
00099 #define Dx7     164
00100 #define E7      155
00101 #define F7      146
00102 #define Fx7     138
00103 #define G7      130
00104 #define Gx7     123
00105 #define A7      116
00106 #define Ax7     109
00107 #define B7      103 // freq = 1000000/ (2*103) = 4.85 khz

```

## 10.15 buzzer\_pitches\_8bit.h File Reference

### Macros

- #define [Gx0](#) 252
- #define [A0](#) 238
- #define [Ax0](#) 224
- #define [B0](#) 212
- #define [C1](#) 200
- #define [Cx0](#) 189
- #define [D1](#) 178
- #define [Dx0](#) 168
- #define [E1](#) 159
- #define [F1](#) 150
- #define [Fx1](#) 141
- #define [G1](#) 133
- #define [Gx1](#) 126
- #define [A1](#) 119
- #define [Ax1](#) 112
- #define [B1](#) 106

- `#define C2` 100
- `#define Cx2` 94
- `#define D2` 89
- `#define Dx2` 84
- `#define E2` 79
- `#define F2` 75
- `#define Fx2` 71
- `#define G2` 67
- `#define Gx2` 63
- `#define A2` 59
- `#define Ax2` 56
- `#define B2` 53
- `#define C3` 50
- `#define Cx3` 47
- `#define D3` 44
- `#define Dx3` 42
- `#define E3` 40
- `#define F3` 37
- `#define Fx3` 35
- `#define G3` 33

### 10.15.1 Macro Definition Documentation

#### 10.15.1.1 A0

```
#define A0 238
```

Definition at line 14 of file [buzzer\\_pitches\\_8bit.h](#).

#### 10.15.1.2 A1

```
#define A1 119
```

Definition at line 26 of file [buzzer\\_pitches\\_8bit.h](#).

#### 10.15.1.3 A2

```
#define A2 59
```

Definition at line 38 of file [buzzer\\_pitches\\_8bit.h](#).

#### 10.15.1.4 Ax0

```
#define Ax0 224
```

Definition at line 15 of file [buzzer\\_pitches\\_8bit.h](#).

#### 10.15.1.5 Ax1

```
#define Ax1 112
```

Definition at line 27 of file [buzzer\\_pitches\\_8bit.h](#).

#### 10.15.1.6 Ax2

```
#define Ax2 56
```

Definition at line 39 of file [buzzer\\_pitches\\_8bit.h](#).

#### 10.15.1.7 B0

```
#define B0 212
```

Definition at line 16 of file [buzzer\\_pitches\\_8bit.h](#).

#### 10.15.1.8 B1

```
#define B1 106
```

Definition at line 28 of file [buzzer\\_pitches\\_8bit.h](#).

#### 10.15.1.9 B2

```
#define B2 53
```

Definition at line 40 of file [buzzer\\_pitches\\_8bit.h](#).

**10.15.1.10 C1**

```
#define C1 200
```

Definition at line 17 of file [buzzer\\_pitches\\_8bit.h](#).

**10.15.1.11 C2**

```
#define C2 100
```

Definition at line 29 of file [buzzer\\_pitches\\_8bit.h](#).

**10.15.1.12 C3**

```
#define C3 50
```

Definition at line 41 of file [buzzer\\_pitches\\_8bit.h](#).

**10.15.1.13 Cx0**

```
#define Cx0 189
```

Definition at line 18 of file [buzzer\\_pitches\\_8bit.h](#).

**10.15.1.14 Cx2**

```
#define Cx2 94
```

Definition at line 30 of file [buzzer\\_pitches\\_8bit.h](#).

**10.15.1.15 Cx3**

```
#define Cx3 47
```

Definition at line 42 of file [buzzer\\_pitches\\_8bit.h](#).



**10.15.1.16 D1**

```
#define D1 178
```

Definition at line 19 of file [buzzer\\_pitches\\_8bit.h](#).

**10.15.1.17 D2**

```
#define D2 89
```

Definition at line 31 of file [buzzer\\_pitches\\_8bit.h](#).

**10.15.1.18 D3**

```
#define D3 44
```

Definition at line 43 of file [buzzer\\_pitches\\_8bit.h](#).

**10.15.1.19 Dx0**

```
#define Dx0 168
```

Definition at line 20 of file [buzzer\\_pitches\\_8bit.h](#).

**10.15.1.20 Dx2**

```
#define Dx2 84
```

Definition at line 32 of file [buzzer\\_pitches\\_8bit.h](#).

**10.15.1.21 Dx3**

```
#define Dx3 42
```

Definition at line 44 of file [buzzer\\_pitches\\_8bit.h](#).

**10.15.1.22 E1**

```
#define E1 159
```

Definition at line 21 of file [buzzer\\_pitches\\_8bit.h](#).

**10.15.1.23 E2**

```
#define E2 79
```

Definition at line 33 of file [buzzer\\_pitches\\_8bit.h](#).

**10.15.1.24 E3**

```
#define E3 40
```

Definition at line 45 of file [buzzer\\_pitches\\_8bit.h](#).

**10.15.1.25 F1**

```
#define F1 150
```

Definition at line 22 of file [buzzer\\_pitches\\_8bit.h](#).

**10.15.1.26 F2**

```
#define F2 75
```

Definition at line 34 of file [buzzer\\_pitches\\_8bit.h](#).

**10.15.1.27 F3**

```
#define F3 37
```

Definition at line 46 of file [buzzer\\_pitches\\_8bit.h](#).

**10.15.1.28 Fx1**

```
#define Fx1 141
```

Definition at line 23 of file [buzzer\\_pitches\\_8bit.h](#).

**10.15.1.29 Fx2**

```
#define Fx2 71
```

Definition at line 35 of file [buzzer\\_pitches\\_8bit.h](#).

**10.15.1.30 Fx3**

```
#define Fx3 35
```

Definition at line 47 of file [buzzer\\_pitches\\_8bit.h](#).

**10.15.1.31 G1**

```
#define G1 133
```

Definition at line 24 of file [buzzer\\_pitches\\_8bit.h](#).

**10.15.1.32 G2**

```
#define G2 67
```

Definition at line 36 of file [buzzer\\_pitches\\_8bit.h](#).

**10.15.1.33 G3**

```
#define G3 33
```

Definition at line 48 of file [buzzer\\_pitches\\_8bit.h](#).

#### 10.15.1.34 Gx0

```
#define Gx0 252
```

Definition at line 13 of file [buzzer\\_pitches\\_8bit.h](#).

#### 10.15.1.35 Gx1

```
#define Gx1 126
```

Definition at line 25 of file [buzzer\\_pitches\\_8bit.h](#).

#### 10.15.1.36 Gx2

```
#define Gx2 63
```

Definition at line 37 of file [buzzer\\_pitches\\_8bit.h](#).

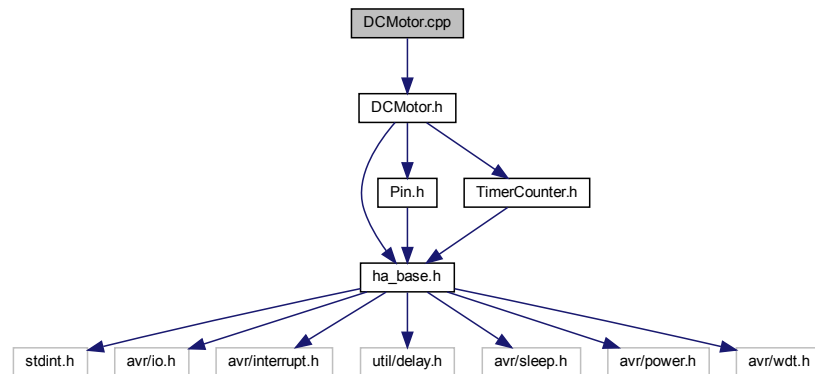
### 10.16 buzzer\_pitches\_8bit.h

```
00001 /* Scale in the key of 1/200 */
00002
00003 /*
00004     These are 1/2 periods (us) -- if you delay this long,
00005     then toggle the speaker pin, you'll get approximate
00006     pitches.
00007
00008     This is the 8-bit version. The pitches get less accurate
00009     as they get higher.
00010
00011 */
00012
00013 #define Gx0 252 // freq = 1000000/ (2*252) = 1.98 khz
00014 #define A0 238
00015 #define Ax0 224
00016 #define B0 212
00017 #define C1 200
00018 #define Cx0 189
00019 #define D1 178
00020 #define Dx0 168
00021 #define E1 159
00022 #define F1 150
00023 #define Fx1 141
00024 #define G1 133
00025 #define Gx1 126
00026 #define A1 119
00027 #define Ax1 112
00028 #define B1 106
00029 #define C2 100
00030 #define Cx2 94
00031 #define D2 89
00032 #define Dx2 84
00033 #define E2 79
00034 #define F2 75
00035 #define Fx2 71
00036 #define G2 67
00037 #define Gx2 63
00038 #define A2 59
00039 #define Ax2 56
00040 #define B2 53
00041 #define C3 50
00042 #define Cx3 47
00043 #define D3 44
00044 #define Dx3 42
00045 #define E3 40
00046 #define F3 37
00047 #define Fx3 35
00048 #define G3 33 // freq = 1000000/ (2*33) = 15.15 khz
```

## 10.17 DCMotor.cpp File Reference

```
#include "DCMotor.h"
```

Include dependency graph for DCMotor.cpp:



## 10.18 DCMotor.cpp

```

00001 #include "DCMotor.h"
00002
00003 component::DCMotor::DCMotor(const io::Pin &ar_pin)
00004     : m_pin(ar_pin)
00005 {
00006     m_pin.toOutput();
00007 }
00008
00009
00010
00011 component::DCMotor::~DCMotor()
00012 {
00013 }
00014
00015
00016 void component::DCMotor::on()
00017 {
00018     m_pin.setHigh();
00019 }
00020
00021 void component::DCMotor::off()
00022 {
00023     m_pin.setLow();
00024 }
00025
00026 void component::DCMotor::toggle()
00027 {
00028     m_pin.toggle();
00029 }
00030 }
  
```

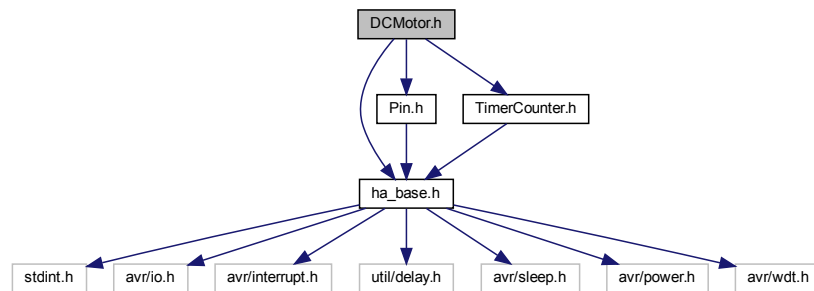
## 10.19 DCMotor.h File Reference

Header file of the DCMotor class.

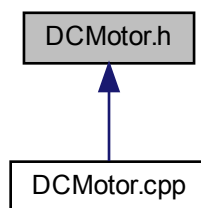
```
#include "ha_base.h"
#include "Pin.h"
```

```
#include "TimerCounter.h"
```

Include dependency graph for DCMotor.h:



This graph shows which files directly or indirectly include this file:



## Classes

- class `component::DCMotor`

## Namespaces

- `component`  
*AVR chip external components.*

### 10.19.1 Detailed Description

Header file of the DCMotor class.

Usage example (test): `#include "MCU.h" #include "TimerCounter0.h" #include "DCMotor.h"`

```
#define DCMOTOR_NUMBER 6
```

```
int main(void) {
```

Init initialize MCU `core::MCU::init();`

instantiate the TimerCounter0 object `core::TimerCounter0` &myTimerCounter0 = `core::TimerCounter0::getInstance();`  
`myTimerCounter0.selectClockSource(core::clockSource::PS_1024);`

instantiate a DCMotor object `component::DCMotor` myDCMotor(`io::Pin(DCMOTOR_NUMBER,io::PortD)`);

`myDCMotor.connect(myTimerCounter0);`

`myDCMotor.spin(myTimerCounter0,0); _delay_ms(5000);`

`myDCMotor.spin(myTimerCounter0,75); _delay_ms(5000);`

`myDCMotor.spin(myTimerCounter0,190); _delay_ms(5000);`

`myDCMotor.spin(myTimerCounter0,200); _delay_ms(5000);`

`myDCMotor.disconnect(myTimerCounter0);`

Mainloop while (1) {

} return 0; } Usage example (H Bridge):

`#include "MCU.h" #include "TimerCounter0.h" #include "DCMotor.h"`

`#define DCMOTOR_NUMBER 6 #define DCMOTOR_BACKWARD 0 #define DCMOTOR_FORWARD 1`

`int main(void) {`

Init initialize MCU `core::MCU::init();`

instantiate the TimerCounter0 object `core::TimerCounter0` &myTimerCounter0 = `core::TimerCounter0::getInstance();`  
`myTimerCounter0.selectClockSource(core::clockSource::PS_256);`

instantiate a DCMotor object `component::DCMotor` myDCMotor(`io::Pin(DCMOTOR_NUMBER,io::PortD)`);

instantiate direction pins objects

`io::Pin` myPinForward = `io::Pin(DCMOTOR_FORWARD,io::PortB)`; myPinForward.toOutput();

`io::Pin` myPinBackward = `io::Pin(DCMOTOR_BACKWARD,io::PortB)`; myPinBackward.toOutput();

`myDCMotor.connect(myTimerCounter0);`

`myPinForward.setHigh(); myPinBackward.setLow(); myDCMotor.spin(myTimerCounter0,30); _delay_ms(5000);`

`myPinForward.setLow(); myPinBackward.setHigh(); myDCMotor.spin(myTimerCounter0,30); _delay_ms(5000);`

`myDCMotor.disconnect(myTimerCounter0);`

Mainloop while (1) {

} return 0; }

Basic class for IO abstraction of Pin and Port

#### Author

Farid Oubbati ( <https://github.com/faroub>)

#### Date

March 2020

Definition in file `DCMotor.h`.

## 10.20 DCMotor.h

```

00001
00117 #ifndef DCMOTOR_H
00118 #define DCMOTOR_H
00119 #include "ha_base.h"
00120 #include "Pin.h"
00121 #include "TimerCounter.h"
00122
00123 namespace component
00124 {
00125
00126 class DCMotor
00127 {
00128 public:
00129     DCMotor(const io::Pin &ar_pin);
00130
00131     ~DCMotor();
00132
00133     void on();
00134     void off();
00135     void toggle();
00136
00137     template<typename TC>
00138     void spin(TC &ar_timerCounter,
00139              //const uint8_t &ar_dir,
00140              const uint16_t &ar_speed,
00141              const core::channel &ar_channel=core::channel::A)
00142     {
00143         ar_timerCounter.setOutputCompareRegister(ar_channel, ar_speed);
00144         ar_timerCounter.start();
00145     }
00146
00147     template<typename TC>
00148     void stop(TC &ar_timerCounter)
00149     {
00150         ar_timerCounter.stop();
00151     }
00152
00153     template<typename TC>
00154     void connect(TC &ar_timerCounter,
00155                 const core::channel &ar_channel=core::channel::A)
00156     {
00157         ar_timerCounter.selectOperationMode(core::operationMode::fast_PWM);
00158         ar_timerCounter.selectCompareOutputMode(ar_channel, core::compareOutputMode::clear);
00159         ar_timerCounter.setCounter(0);
00160     }
00161
00162     template<typename TC>
00163     void disconnect(TC &ar_timerCounter,
00164                    const core::channel &ar_channel=core::channel::A)
00165     {
00166         ar_timerCounter.selectCompareOutputMode(ar_channel, core::compareOutputMode::normal);
00167         ar_timerCounter.stop();
00168     }
00169
00170 private:
00171     io::Pin m_pin;
00172 };
00173
00174 #endif // DCMOTOR_H

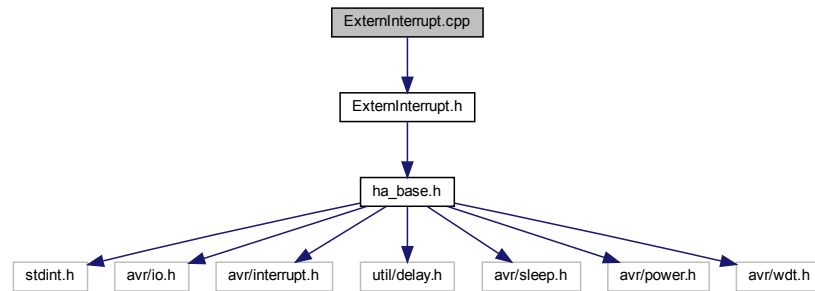
```

## 10.21 ExternInterrupt.cpp File Reference

```
#include "ExternInterrupt.h"
```



Include dependency graph for ExternInterrupt.cpp:



## 10.22 ExternInterrupt.cpp

```

00001 #include "ExternInterrupt.h"
00002
00003 core::ExternInterrupt& core::ExternInterrupt::getInstance()
00004 {
00005     static ExternInterrupt l_instance;
00006     return l_instance;
00007 }
00008
00009
00010
00011 core::ExternInterrupt::ExternInterrupt()
00012 {
00013     sei();
00014 }
00015
00016
00017 core::ExternInterrupt::~ExternInterrupt()
00018 {
00019 }
00020
00021
00022 void core::ExternInterrupt::setInt0SenseControl(const senseControl& ar_senseControl)
00023 {
00024     EXT_INT_SET_INT0_SENSE_CONTROL(static_cast<uint8_t>(ar_senseControl));
00025 }
00026
00027
00028 void core::ExternInterrupt::setInt1SenseControl(const senseControl& ar_senseControl)
00029 {
00030     EXT_INT_SET_INT1_SENSE_CONTROL(static_cast<uint8_t>(ar_senseControl));
00031 }
00032
00033
00034 void core::ExternInterrupt::enableInt0(const uint8_t a_enable)
00035 {
00036     if (a_enable) {
00037         EXT_INT_ENABLE_INT0;
00038     } else {
00039         EXT_INT_DISABLE_INT0;
00040     }
00041 }
00042
00043
00044
00045 void core::ExternInterrupt::enableInt1(const uint8_t a_enable)
00046 {
00047     if (a_enable) {
00048         EXT_INT_ENABLE_INT1;
00049     } else {
00050         EXT_INT_DISABLE_INT1;
00051     }
00052 }
00053
00054
00055
00056 void core::ExternInterrupt::enablePinChange(const pinChangePort& ar_pinChangePort, const uint8_t
a_enable)

```

```

00057 {
00058     if (a_enable) {
00059         EXT_INT_ENABLE_PIN_CHANGE_INTERRUPT(static_cast<uint8_t>(ar_pinChangePort));
00060     } else {
00061         EXT_INT_DISABLE_PIN_CHANGE_INTERRUPT(static_cast<uint8_t>(ar_pinChangePort));
00062     }
00063 }
00064
00065 }
00066
00067 void core::ExternInterrupt::enablePinChangeMaskPortB(const uint8_t a_pinNumber, const uint8_t
a_enable)
00068 {
00069     if (a_enable) {
00070         EXT_INT_ENABLE_PIN_CHANGE_MASK_PORTB(a_pinNumber);
00071     } else {
00072         EXT_INT_ENABLE_PIN_CHANGE_MASK_PORTB(a_pinNumber);
00073     }
00074 }
00075
00076 }
00077
00078
00079 void core::ExternInterrupt::enablePinChangeMaskPortC(const uint8_t a_pinNumber, const uint8_t
a_enable)
00080 {
00081     if (a_enable) {
00082         EXT_INT_ENABLE_PIN_CHANGE_MASK_PORTC(a_pinNumber);
00083     } else {
00084         EXT_INT_ENABLE_PIN_CHANGE_MASK_PORTC(a_pinNumber);
00085     }
00086 }
00087
00088 }
00089
00090 void core::ExternInterrupt::enablePinChangeMaskPortD(const uint8_t a_pinNumber, const uint8_t
a_enable)
00091 {
00092     if (a_enable) {
00093         EXT_INT_ENABLE_PIN_CHANGE_MASK_PORTD(a_pinNumber);
00094     } else {
00095         EXT_INT_ENABLE_PIN_CHANGE_MASK_PORTD(a_pinNumber);
00096     }
00097 }
00098 }

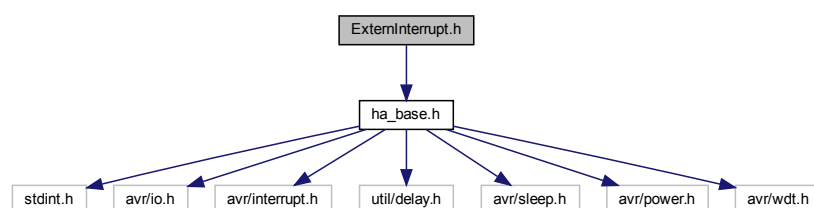
```

## 10.23 ExternInterrupt.h File Reference

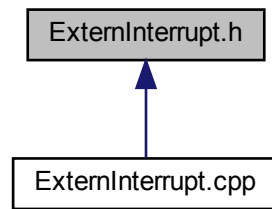
Header file of the ExternInterrupt class.

```
#include "ha_base.h"
```

Include dependency graph for ExternInterrupt.h:



This graph shows which files directly or indirectly include this file:



## Classes

- class [core::ExternInterrupt](#)

## Namespaces

- [core](#)  
*AVR chip internal core components.*

## Enumerations

- enum [core::senseControl](#) : `uint8_t` { [core::senseControl::lowLevel](#) =0, [core::senseControl::logicalChange](#), [core::senseControl::fallingEdge](#), [core::senseControl::risingEdge](#) }
- enum [core::pinChangePort](#) : `uint8_t` { [core::pinChangePort::PCINTB](#) =0, [core::pinChangePort::PCINTC](#), [core::pinChangePort::PCINTD](#) }

### 10.23.1 Detailed Description

Header file of the ExternInterrupt class.

class to handle externally triggered interrupts.

Usage example (external interrupt):

```
#include "PushButton.h" #include "Led.h" #include "ExternInterrupt.h"
```

```
#define PUSHBUTTON_NUMBER 2 #define MAIN_LED_NUMBER 0 #define INTERRUPT_LED_NUMBER 1 #define DELAYTIME 1000
```

```
instantiate a Led object declaration of global variables (shut up warning) extern component::Led MainLed; extern component::Led InterruptLed; component::Led MainLed(io::Pin(MAIN_LED_NUMBER,io::PortB)); component::Led InterruptLed(io::Pin(INTERRUPT_LED_NUMBER,io::PortB));
```

```
instantiate a Led object declaration of global variable (shut up warning) extern component::PushButton PushButton; component::PushButton PushButton(io::Pin(PUSHBUTTON_NUMBER,io::PortD));
```

```

int main(void) {

Init

    instantiate the external interrupt manager core::ExternInterrupt &myExternInterrupt = core::ExternInterrupt::getInstance();
    myExternInterrupt.enableInt0(1); myExternInterrupt.setInt0SenseControl(core::senseControl::logicalChange);

    Mainloop while (1) {

        MainLed.toggle(); _delay_ms(DELAYTIME);

    } return 0; }

void core::ExternInterrupt::Int0ServiceRoutine() { if (PushButton.isPressed()) { InterruptLed.on(); } else { InterruptLed.off(); }

}

```

Usage example (pin change interrupt):

```

#include "PushButton.h" #include "Led.h" #include "ExternInterrupt.h"

#define PUSHBUTTON_1_NUMBER 2 #define PUSHBUTTON_2_NUMBER 3 #define MAIN_LED_NUMBER 0
#define INTERRUPT_LED_1_NUMBER 1 #define INTERRUPT_LED_2_NUMBER 2 #define DELAYTIME 1000

    instantiate a Led objects declaration of global variables (shut up warning) extern component::Led MainLed; extern component::Led InterruptLed1; extern component::Led InterruptLed2; component::Led MainLed(io::Pin(MAIN_LED_NUMBER,io::PortB)); component::Led InterruptLed1(io::Pin(INTERRUPT_LED_1_NUMBER,io::PortB)); component::Led InterruptLed2(io::Pin(INTERRUPT_LED_2_NUMBER,io::PortB));

    instantiate a PushButton objects declaration of global variable (shut up warning) extern component::PushButton PushButton1; extern component::PushButton PushButton2; component::PushButton PushButton1(io::Pin(PUSHBUTTON_1_NUMBER,io::PortD)); component::PushButton PushButton2(io::Pin(PUSHBUTTON_2_NUMBER,io::PortB));

int main(void) {

Init

    instantiate the external interrupt manager core::ExternInterrupt &myExternInterrupt = core::ExternInterrupt::getInstance();
    myExternInterrupt.enablePinChange(core::pinChangePort::PCINTD,1); myExternInterrupt.enablePinChange(core::pinChangePort::PCINTB,1); myExternInterrupt.enablePinChangeMaskPortD(PUSHBUTTON_1_NUMBER,1);
    myExternInterrupt.enablePinChangeMaskPortB(PUSHBUTTON_2_NUMBER,1);

    Mainloop while (1) {

        MainLed.toggle(); _delay_ms(DELAYTIME);

    } return 0; }

void core::ExternInterrupt::pinChangePortDServiceRoutine() { if (PushButton1.isPressed()) { InterruptLed1.on(); } else { InterruptLed1.off(); }

} } void core::ExternInterrupt::pinChangePortBServiceRoutine() { if (PushButton2.isPressed()) { InterruptLed2.on(); } else { InterruptLed2.off(); }

}

```

#### Author

Farid Oubati ( <https://github.com/faroub> )

#### Date

March 2020

Definition in file [ExternInterrupt.h](#).

## 10.24 ExternInterrupt.h

```

00001
00140 #ifndef EXTERN_INTERRUPTS_H
00141 #define EXTERN_INTERRUPTS_H
00142 #include "ha_base.h"
00143
00144
00145
00146 namespace core
00147 {
00148
00149 enum class senseControl : uint8_t {
00150     lowLevel=0,
00151     logicalChange,
00152     fallingEdge,
00153     risingEdge
00154 };
00155
00156 enum class pinChangePort : uint8_t {
00157     PCINTB=0,
00158     PCINTC,
00159     PCINTD,
00160 };
00161
00162 class ExternInterrupt
00163 {
00164
00165 public:
00166
00167     static ExternInterrupt& getInstance();
00168
00169     void setInt0SenseControl(const senseControl& ar_senseControl);
00170
00171     void setInt1SenseControl(const senseControl& ar_senseControl);
00172
00173     void enableInt0(const uint8_t a_enable);
00174
00175     void enableInt1(const uint8_t a_enable);
00176
00177     void enablePinChange(const pinChangePort& ar_pinChangePort, const uint8_t a_enable);
00178
00179     void enablePinChangeMaskPortB(const uint8_t a_pinNumber, const uint8_t a_enable);
00180
00181     void enablePinChangeMaskPortC(const uint8_t a_pinNumber, const uint8_t a_enable);
00182
00183     void enablePinChangeMaskPortD(const uint8_t a_pinNumber, const uint8_t a_enable);
00184
00185     static void Int0ServiceRoutine() __asm__(STR(EXT_INT_INT0_INTERRUPT)) __attribute__((__signal__,
00186         __used__, __externally_visible__));
00187
00188     static void Int1ServiceRoutine() __asm__(STR(EXT_INT_INT1_INTERRUPT)) __attribute__((__signal__,
00189         __used__, __externally_visible__));
00190
00191     static void pinChangePortBServiceRoutine() __asm__(STR(EXT_INT_PIN_CHANGE_PORTB_INTERRUPT))
00192         __attribute__((__signal__, __used__, __externally_visible__));
00193
00194     static void pinChangePortCServiceRoutine() __asm__(STR(EXT_INT_PIN_CHANGE_PORTC_INTERRUPT))
00195         __attribute__((__signal__, __used__, __externally_visible__));
00196
00197     static void pinChangePortDServiceRoutine() __asm__(STR(EXT_INT_PIN_CHANGE_PORTD_INTERRUPT))
00198         __attribute__((__signal__, __used__, __externally_visible__));
00199
00200 protected:
00201
00202 private:
00203
00204     ExternInterrupt();
00205
00206     ~ExternInterrupt();
00207
00208     ExternInterrupt(const ExternInterrupt&);
00209
00210     const ExternInterrupt& operator=(const ExternInterrupt&);
00211
00212 };
00213
00214 #endif

```



## 10.25.2 Macro Definition Documentation

### 10.25.2.1 STR

```
#define STR(  
    s ) STRx(s)
```

Definition at line 16 of file [ha\\_base.h](#).

### 10.25.2.2 STRx

```
#define STRx(  
    s ) #s
```

Definition at line 15 of file [ha\\_base.h](#).

## 10.26 ha\_base.h

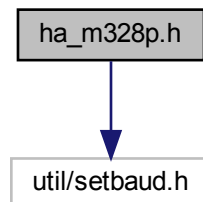
```
00001  
00009 #ifndef HABASE_H  
00010 #define HABASE_H  
00011  
00012 /*  
00013  * Helper construct to get interrupt numbers from names  
00014  */  
00015 #define STRx(s) #s  
00016 #define STR(s) STRx(s)  
00017  
00018 /*  
00019  * Any class needs these includes from avr-libc  
00020  */  
00021 #include <stdint.h>  
00022 #include <avr/io.h>  
00023 #include <avr/interrupt.h>  
00024 #include <util/delay.h>  
00025 #include <avr/sleep.h>  
00026 #include <avr/power.h>  
00027 #include <avr/wdt.h>  
00028  
00029  
00030 /*  
00031  * Include hardware abstraction the Atmega328p  
00032  */  
00033 #if defined(__AVR_ATmega328P__)  
00034  
00035     #include "ha_m328p.h"  
00036     #include "utils_m328p.h"  
00037  
00038 #endif  
00039  
00040  
00041 #endif
```

## 10.27 ha\_m328p.h File Reference

Header file for the hardware abstraction macros of the Atmega328p.

```
#include <util/setbaud.h>
```

Include dependency graph for ha\_m328p.h:



### Macros

- #define [PUSHBUTTON\\_DEBOUNCE\\_TIME\\_US](#) 1000
- #define [PUSHBUTTON\\_SAMPLING](#) 12
- #define [USART0\\_SET\\_BAUDRATE\\_HIGH\\_REGISTER](#) UBRR0H = UBRRH\_VALUE
- #define [USART0\\_SET\\_BAUDRATE\\_LOW\\_REGISTER](#) UBRR0L = UBRL\_VALUE
- #define [USART0\\_ENABLE\\_ASYNC\\_TRANSMISSION\\_MODE](#) UCSR0C &= ~((1 << UMSEL01) | (1 << UMSEL00))
- #define [USART0\\_ENABLE\\_SYNC\\_TRANSMISSION\\_MODE](#) UCSR0C = (UCSR0C & ~(1 << UMSEL01)) | (1 << UMSEL00)
- #define [USART0\\_ENABLE\\_MASTER\\_SPI\\_MODE](#) UCSR0C |= (1 << UMSEL01) | (1 << UMSEL00)
- #define [USART0\\_ENABLE\\_DOUBLE\\_SPEED\\_MODE](#)
- #define [USART0\\_DISABLE\\_DOUBLE\\_SPEED\\_MODE](#)
- #define [USART0\\_DISABLE\\_DOUBLE\\_SPEED\\_MODE](#) UCSR0A &= ~(1 << U2X0)
- #define [USART0\\_ENABLE\\_EVEN\\_PARITY\\_MODE](#) UCSR0C = (1 << UPM01) | (UCSR0C & ~(1 << UPM00))
- #define [USART0\\_ENABLE\\_ODD\\_PARITY\\_MODE](#) UCSR0C |= ((1 << UPM01) | (1 << UPM00))
- #define [USART0\\_DISABLE\\_PARITY\\_MODE](#) UCSR0C &= ~((1 << UPM01) | (1 << UPM00))
- #define [USART0\\_SET\\_ONE\\_STOP\\_BIT](#) UCSR0C &= ~(1 << USBS0)
- #define [USART0\\_SET\\_TWO\\_STOP\\_BITS](#) UCSR0C |= (1 << USBS0)
- #define [USART0\\_SET\\_9BIT\\_FRAME\\_SIZE](#)
- #define [USART0\\_SET\\_8BIT\\_FRAME\\_SIZE](#)
- #define [USART0\\_SET\\_7BIT\\_FRAME\\_SIZE](#)
- #define [USART0\\_SET\\_6BIT\\_FRAME\\_SIZE](#)
- #define [USART0\\_SET\\_5BIT\\_FRAME\\_SIZE](#)
- #define [USART0\\_ENABLE\\_TRANSMITTER](#) UCSR0B |= 1 << TXEN0
- #define [USART0\\_DISABLE\\_TRANSMITTER](#) UCSR0B &= ~(1 << TXEN0)
- #define [USART0\\_ENABLE\\_RECEIVER](#) UCSR0B |= 1 << RXEN0
- #define [USART0\\_DISABLE\\_RECEIVER](#) UCSR0B &= ~(1 << RXEN0)
- #define [USART0\\_CONTROL\\_STATUS\\_REGISTER](#) UCSR0A
- #define [USART0\\_DATA\\_REGISTER](#) UDR0
- #define [USART0\\_FRAME\\_ERROR](#) FE0



- `#define USART0_DATA_OVERRUN DOR0`
- `#define USART0_PARITY_ERROR UPE0`
- `#define USART0_ENABLE_DATA_REGISTER_EMPTY_INTERRUPT UCSR0B |= 1 << UDRIE0`
- `#define USART0_DISABLE_DATA_REGISTER_EMPTY_INTERRUPT UCSR0B &= ~(1 << UDRIE0)`
- `#define USART0_ENABLE_RECEIVE_COMPLETE_INTERRUPT UCSR0B |= 1 << RXCIE0`
- `#define USART0_DISABLE_RECEIVE_COMPLETE_INTERRUPT UCSR0B &= ~(1 << RXCIE0)`
- `#define USART0_ENABLE_TRANSMIT_COMPLETE_INTERRUPT UCSR0B |= 1 << TXCIE0`
- `#define USART0_DISABLE_TRANSMIT_COMPLETE_INTERRUPT UCSR0B &= ~(1 << TXCIE0)`
- `#define USART0_RECEIVE_COMPLETE_INTERRUPT USART_RX_vect`
- `#define USART0_TRANSMIT_COMPLETE_INTERRUPT USART_TX_vect`
- `#define USART0_DATA_REGISTER_EMPTY_INTERRUPT USART_UDRE_vect`
- `#define ADC_SELECT_REF_VOLTAGE(refVoltage) ADMUX &= 0x3F; ADMUX |= refVoltage << 6`
- `#define ADC_ADJUST_RESULT_LEFT ADMUX |= 1 << ADLAR`
- `#define ADC_ADJUST_RESULT_RIGHT ADMUX &= ~(1 << ADLAR)`
- `#define ADC_SELECT_ANALOG_INPUT(pinNumber) ADMUX &= 0xF0; ADMUX |= pinNumber`
- `#define ADC_DISABLE_DIGITAL_INPUT_REGISTER(pinNumber) DIDR0 &= 0xC0; DIDR0 |= pinNumber`
- `#define ADC_ENABLE ADCSRA |= 1 << ADEN`
- `#define ADC_DISABLE ADCSRA &= ~(1 << ADEN)`
- `#define ADC_START_CONVERSION ADCSRA |= 1 << ADSC`
- `#define ADC_STOP_CONVERSION ADCSRA &= ~(1 << ADSC)`
- `#define ADC_ENABLE_AUTOTRIGGER ADCSRA |= 1 << ADATE`
- `#define ADC_DISABLE_AUTOTRIGGER ADCSRA &= ~(1 << ADATE)`
- `#define ADC_ENABLE_CONVERSION_COMPLETE_INTERRUPT ADCSRA |= 1 << ADIE`
- `#define ADC_DISABLE_CONVERSION_COMPLETE_INTERRUPT ADCSRA &= ~(1 << ADIE)`
- `#define ADC_SELECT_CLOCK_PRESCALER(clockPrescaler) ADCSRA &= 0xF8; ADCSRA |= clockPrescaler`
- `#define ADC_SELECT_AUTO_TRIGGER_SOURCE(triggerSource) ADCSRB &= 0xF8; ADCSRB |= triggerSource`
- `#define ADC_CONVERSION_COMPLETE_INTERRUPT ADC_vect`
- `#define EXT_INT_SET_INT0_SENSE_CONTROL(senseControl) EICRA &= 0xFC; EICRA |= senseControl`
- `#define EXT_INT_SET_INT1_SENSE_CONTROL(senseControl) EICRA &= 0xF3; EICRA |= senseControl`
- `#define EXT_INT_ENABLE_INT0 EIMSK |= 1 << INT0`
- `#define EXT_INT_DISABLE_INT0 EIMSK &= ~(1 << INT0)`
- `#define EXT_INT_ENABLE_INT1 EIMSK |= 1 << INT1`
- `#define EXT_INT_DISABLE_INT1 EIMSK &= ~(1 << INT1)`
- `#define EXT_INT_ENABLE_PIN_CHANGE_INTERRUPT(pinChangePort) PCICR |= 1 << pinChangePort`
- `#define EXT_INT_DISABLE_PIN_CHANGE_INTERRUPT(pinChangePort) PCICR &= ~(1 << pinChangePort)`
- `#define EXT_INT_ENABLE_PIN_CHANGE_MASK_PORTB(pinChangePin) PCMSK0 |= 1 << pinChangePin`
- `#define EXT_INT_DISABLE_PIN_CHANGE_MASK_PORTB(pinChangePin) PCMSK0 &= ~(1 << pinChangePin)`
- `#define EXT_INT_ENABLE_PIN_CHANGE_MASK_PORTC(pinChangePin) PCMSK1 |= 1 << pinChangePin`
- `#define EXT_INT_DISABLE_PIN_CHANGE_MASK_PORTC(pinChangePin) PCMSK1 &= ~(1 << pinChangePin)`
- `#define EXT_INT_ENABLE_PIN_CHANGE_MASK_PORTD(pinChangePin) PCMSK2 |= 1 << pinChangePin`
- `#define EXT_INT_DISABLE_PIN_CHANGE_MASK_PORTD(pinChangePin) PCMSK2 &= ~(1 << pinChangePin)`
- `#define EXT_INT_INT0_INTERRUPT INT0_vect`
- `#define EXT_INT_INT1_INTERRUPT INT1_vect`
- `#define EXT_INT_PIN_CHANGE_PORTB_INTERRUPT PCINT0_vect`
- `#define EXT_INT_PIN_CHANGE_PORTC_INTERRUPT PCINT1_vect`
- `#define EXT_INT_PIN_CHANGE_PORTD_INTERRUPT PCINT2_vect`

- `#define TIMER0_STOP` TCCR0B &= 0xF8
- `#define TIMER1_STOP` TCCR1B &= 0xF8
- `#define TIMER2_STOP` TCCR2B &= 0xF8
- `#define TIMER0_SELECT_CLOCK_SOURCE`(clockSource) TCCR0B &= 0xF8; TCCR0B |= clockSource
- `#define TIMER1_SELECT_CLOCK_SOURCE`(clockSource) TCCR1B &= 0xF8; TCCR1B |= clockSource
- `#define TIMER2_SELECT_CLOCK_SOURCE`(clockSource) TCCR2B &= 0xF8; TCCR2B |= clockSource
- `#define TIMER0_SELECT_OPERATION_MODE`(operationMode) TCCR0A &= 0xFC; TCCR0A |= (operationMode & 3); TCCR0B &= 0xF7; TCCR0B |= (operationMode & 12) << 1
- `#define TIMER1_SELECT_OPERATION_MODE`(operationMode) TCCR1A &= 0xFC; TCCR1A |= (operationMode & 3); TCCR1B &= 0xE7; TCCR1B |= (operationMode & 12) << 1
- `#define TIMER2_SELECT_OPERATION_MODE`(operationMode) TCCR2A &= 0xFC; TCCR2A |= (operationMode & 3); TCCR2B &= 0xF7; TCCR2B |= (operationMode & 12) << 1
- `#define TIMER0_SELECT_COM_CHANNEL_A`(compareOutputMode) TCCR0A &= 0x3F; TCCR0A |= compareOutputMode << 6
- `#define TIMER0_SELECT_COM_CHANNEL_B`(compareOutputMode) TCCR0A &= 0xCF; TCCR0A |= compareOutputMode << 4
- `#define TIMER1_SELECT_COM_CHANNEL_A`(compareOutputMode) TCCR1A &= 0x3F; TCCR1A |= compareOutputMode << 6
- `#define TIMER1_SELECT_COM_CHANNEL_B`(compareOutputMode) TCCR1A &= 0xCF; TCCR1A |= compareOutputMode << 4
- `#define TIMER2_SELECT_COM_CHANNEL_A`(compareOutputMode) TCCR2A &= 0x3F; TCCR2A |= compareOutputMode << 6
- `#define TIMER2_SELECT_COM_CHANNEL_B`(compareOutputMode) TCCR2A &= 0xCF; TCCR2A |= compareOutputMode << 4
- `#define TIMER0_ENABLE_COM_CHANNEL_A_INTERRUPT` TIMSK0 |= 1 << OCIE0A
- `#define TIMER0_DISABLE_COM_CHANNEL_A_INTERRUPT` TIMSK0 &= ~(1 << OCIE0A)
- `#define TIMER0_ENABLE_COM_CHANNEL_B_INTERRUPT` TIMSK0 |= 1 << OCIE0B
- `#define TIMER0_DISABLE_COM_CHANNEL_B_INTERRUPT` TIMSK0 &= ~(1 << OCIE0B)
- `#define TIMER0_ENABLE_OVERFLOW_INTERRUPT` TIMSK0 |= 1 << TOIE0
- `#define TIMER0_DISABLE_OVERFLOW_INTERRUPT` TIMSK0 &= ~(1 << TOIE0)
- `#define TIMER1_ENABLE_COM_CHANNEL_A_INTERRUPT` TIMSK1 |= 1 << OCIE1A
- `#define TIMER1_DISABLE_COM_CHANNEL_A_INTERRUPT` TIMSK1 &= ~(1 << OCIE1A)
- `#define TIMER1_ENABLE_COM_CHANNEL_B_INTERRUPT` TIMSK1 |= 1 << OCIE1B
- `#define TIMER1_DISABLE_COM_CHANNEL_B_INTERRUPT` TIMSK1 &= ~(1 << OCIE1B)
- `#define TIMER1_ENABLE_OVERFLOW_INTERRUPT` TIMSK1 |= 1 << TOIE1
- `#define TIMER1_DISABLE_OVERFLOW_INTERRUPT` TIMSK1 &= ~(1 << TOIE1)
- `#define TIMER1_ENABLE_INPUT_CAPTURE_INTERRUPT` TIMSK1 |= 1 << ICIE1
- `#define TIMER1_DISABLE_INPUT_CAPTURE_INTERRUPT` TIMSK1 &= ~(1 << ICIE1)
- `#define TIMER2_ENABLE_COM_CHANNEL_A_INTERRUPT` TIMSK2 |= 1 << OCIE2A
- `#define TIMER2_DISABLE_COM_CHANNEL_A_INTERRUPT` TIMSK2 &= ~(1 << OCIE2A)
- `#define TIMER2_ENABLE_COM_CHANNEL_B_INTERRUPT` TIMSK2 |= 1 << OCIE2B
- `#define TIMER2_DISABLE_COM_CHANNEL_B_INTERRUPT` TIMSK2 &= ~(1 << OCIE2B)
- `#define TIMER2_ENABLE_OVERFLOW_INTERRUPT` TIMSK2 |= 1 << TOIE2
- `#define TIMER2_DISABLE_OVERFLOW_INTERRUPT` TIMSK2 &= ~(1 << TOIE2)
- `#define TIMER0_COM_CHANNEL_A_INTERRUPT` TIMER0\_COMPA\_vect
- `#define TIMER0_COM_CHANNEL_B_INTERRUPT` TIMER0\_COMPB\_vect
- `#define TIMER0_OVERFLOW_INTERRUPT` TIMER0\_OVF\_vect
- `#define TIMER1_COM_CHANNEL_A_INTERRUPT` TIMER1\_COMPA\_vect
- `#define TIMER1_COM_CHANNEL_B_INTERRUPT` TIMER1\_COMPB\_vect
- `#define TIMER1_OVERFLOW_INTERRUPT` TIMER1\_OVF\_vect
- `#define TIMER1_INPUT_CAPTURE_INTERRUPT` TIMER1\_CAPT\_vect
- `#define TIMER2_COM_CHANNEL_A_INTERRUPT` TIMER2\_COMPA\_vect
- `#define TIMER2_COM_CHANNEL_B_INTERRUPT` TIMER2\_COMPB\_vect
- `#define TIMER2_OVERFLOW_INTERRUPT` TIMER2\_OVF\_vect

- #define [WATCHDOG\\_SELECT\\_TIMEOUT](#)(timeOut) WDTCSR |= (1<<WDCE) | (1<<WDE); WDTCSR = (0<<WDIE) | (0<<WDE) | timeOut;
- #define [WATCHDOG\\_START](#)(operationMode, timeOut) MCUSR &= ~(1<<WDRF); WDTCSR |= (1<<WDCE) | (1<<WDE); WDTCSR = operationMode | timeOut;
- #define [WATCHDOG\\_STOP](#) MCUSR &= ~(1<<WDRF); WDTCSR |= (1<<WDCE) | (1<<WDE); WDTCSR = 0x00;
- #define [WATCHDOG\\_TIMEOUT\\_INTERRUPT](#) WDT\_vect
- #define [MCU\\_SELECT\\_SLEEP\\_MODE](#)(sleepMode) SMCR &= 0xF1; SMCR |= sleepMode << 1
- #define [MCU\\_SLEEP\\_ENABLE](#) SMCR |= 1 << SE
- #define [MCU\\_SLEEP\\_DISABLE](#) SMCR &= ~(1 << SE)
- #define [MCU\\_TWI\\_ENABLE](#) PRR &= ~(1 << PRTWI)
- #define [MCU\\_TWI\\_DISABLE](#) PRR |= 1 << PRTWI
- #define [MCU\\_TIMER2\\_ENABLE](#) PRR &= ~(1 << PRTIM2)
- #define [MCU\\_TIMER2\\_DISABLE](#) PRR |= 1 << PRTIM2
- #define [MCU\\_TIMER1\\_ENABLE](#) PRR &= ~(1 << PRTIM1)
- #define [MCU\\_TIMER1\\_DISABLE](#) PRR |= 1 << PRTIM1
- #define [MCU\\_TIMER0\\_ENABLE](#) PRR &= ~(1 << PRTIM0)
- #define [MCU\\_TIMER0\\_DISABLE](#) PRR |= 1 << PRTIM0
- #define [MCU\\_SPI\\_ENABLE](#) PRR &= ~(1 << PRSPI)
- #define [MCU\\_SPI\\_DISABLE](#) PRR |= 1 << PRSPI
- #define [MCU\\_USART0\\_ENABLE](#) PRR &= ~(1 << PRUSART0)
- #define [MCU\\_USART0\\_DISABLE](#) PRR |= 1 << PRUSART0
- #define [MCU\\_ADC\\_ENABLE](#) PRR &= ~(1 << PRADC)
- #define [MCU\\_ADC\\_DISABLE](#) PRR |= 1 << PRADC
- #define [MCU\\_BOD\\_DISABLE](#) MCUCR |= (1<<BODSE) | (1<<BODS); MCUCR &= ~(1<<BODSE);
- #define [SERVOMOTOR\\_TIMER\\_PULSE\\_WIDTH\\_COUNT](#)(pulseWidth, clockPrescaler) ((F\_CPU/1000000UL) \* (pulseWidth/ clockPrescaler))
- #define [SERVOMOTOR\\_TIMER\\_ANGLE\\_COUNT](#)(angle, out\_min, out\_mid, out\_max) (((out\_min\*(angle - 90L)\*(angle - 180L))/16200L + (angle\*out\_max\*(angle - 90L))/16200L - (angle\*out\_mid\*(angle - 180L))/8100L))
- #define [SPI\\_ENABLE](#) SPCR |= 1 << SPE
- #define [SPI\\_DISABLE](#) SPCR &= ~(1 << SPE)
- #define [SPI\\_SELECT\\_MASTER\\_MODE](#) SPCR |= 1 << MSTR
- #define [SPI\\_SELECT\\_SLAVE\\_MODE](#) SPCR &= ~(1 << MSTR)
- #define [SPI\\_SELECT\\_DATA\\_MODE](#)(dataMode) SPCR &= 0xF3; SPCR |= dataMode << 2
- #define [SPI\\_SELECT\\_CLOCK\\_PRESCALER](#)(clockPrescaler) SPCR &= 0xFC; SPCR |= (clockPrescaler & 3); SPSR &= 0xFE; SPSR |= (clockPrescaler & 4) >> 2
- #define [SPI\\_SELECT\\_DATA\\_ORDER](#)(dataOrder) SPCR &= 0xDF; SPCR |= dataOrder << 5
- #define [SPI\\_WRITE\\_COLLISION](#) WCOL
- #define [SPI\\_TRANSFER\\_COMPLETE](#) SPIF
- #define [SPI\\_MASTER\\_MODE](#) MSTR
- #define [SPI\\_CONTROL\\_REGISTER](#) SPCR
- #define [SPI\\_DATA\\_REGISTER](#) SPDR
- #define [SPI\\_STATUS\\_REGISTER](#) SPSR
- #define [SPI\\_ENABLE\\_TRANSFER\\_COMPLETE\\_INTERRUPT](#) SPCR |= 1 << SPIE
- #define [SPI\\_DISABLE\\_TRANSFER\\_COMPLETE\\_INTERRUPT](#) SPCR &= ~(1 << SPIE)
- #define [SPI\\_TRANSFER\\_COMPLETE\\_INTERRUPT](#) SPI\_STC\_vect

### 10.27.1 Detailed Description

Header file for the hardware abstraction macros of the Atmega328p.

**Author**

Farid Oubbati ( <https://github.com/faroub>)

**Date**

March 2020

Definition in file [ha\\_m328p.h](#).

## 10.27.2 Macro Definition Documentation

### 10.27.2.1 ADC\_ADJUST\_RESULT\_LEFT

```
#define ADC_ADJUST_RESULT_LEFT ADMUX |= 1 << ADLAR
```

Definition at line 126 of file [ha\\_m328p.h](#).

### 10.27.2.2 ADC\_ADJUST\_RESULT\_RIGHT

```
#define ADC_ADJUST_RESULT_RIGHT ADMUX &= ~(1 << ADLAR)
```

Definition at line 127 of file [ha\\_m328p.h](#).

### 10.27.2.3 ADC\_CONVERSION\_COMPLETE\_INTERRUPT

```
#define ADC_CONVERSION_COMPLETE_INTERRUPT ADC_vect
```

Definition at line 151 of file [ha\\_m328p.h](#).

### 10.27.2.4 ADC\_DISABLE

```
#define ADC_DISABLE ADCSRA &= ~(1 << ADEN)
```

Definition at line 134 of file [ha\\_m328p.h](#).

### 10.27.2.5 ADC\_DISABLE\_AUTOTRIGGER

```
#define ADC_DISABLE_AUTOTRIGGER ADCSRA &= ~(1 << ADATE)
```

Definition at line 140 of file [ha\\_m328p.h](#).

### 10.27.2.6 ADC\_DISABLE\_CONVERSION\_COMPLETE\_INTERRUPT

```
#define ADC_DISABLE_CONVERSION_COMPLETE_INTERRUPT ADCSRA &= ~(1 << ADIE)
```

Definition at line 143 of file [ha\\_m328p.h](#).

### 10.27.2.7 ADC\_DISABLE\_DIGITAL\_INPUT\_REGISTER

```
#define ADC_DISABLE_DIGITAL_INPUT_REGISTER(  
    pinNumber ) DIDR0 &= 0xC0; DIDR0 |= pinNumber
```

Definition at line 130 of file [ha\\_m328p.h](#).

### 10.27.2.8 ADC\_ENABLE

```
#define ADC_ENABLE ADCSRA |= 1 << ADEN
```

Definition at line 133 of file [ha\\_m328p.h](#).

### 10.27.2.9 ADC\_ENABLE\_AUTOTRIGGER

```
#define ADC_ENABLE_AUTOTRIGGER ADCSRA |= 1 << ADATE
```

Definition at line 139 of file [ha\\_m328p.h](#).

### 10.27.2.10 ADC\_ENABLE\_CONVERSION\_COMPLETE\_INTERRUPT

```
#define ADC_ENABLE_CONVERSION_COMPLETE_INTERRUPT ADCSRA |= 1 << ADIE
```

Definition at line 142 of file [ha\\_m328p.h](#).

#### 10.27.2.11 ADC\_SELECT\_ANALOG\_INPUT

```
#define ADC_SELECT_ANALOG_INPUT(  
    pinNumber ) ADMUX &= 0xF0; ADMUX |= pinNumber
```

Definition at line 129 of file [ha\\_m328p.h](#).

#### 10.27.2.12 ADC\_SELECT\_AUTO\_TRIGGER\_SOURCE

```
#define ADC_SELECT_AUTO_TRIGGER_SOURCE(  
    triggerSource ) ADCSRB &= 0xF8; ADCSRB |= triggerSource
```

Definition at line 148 of file [ha\\_m328p.h](#).

#### 10.27.2.13 ADC\_SELECT\_CLOCK\_PRESCALER

```
#define ADC_SELECT_CLOCK_PRESCALER(  
    clockPrescaler ) ADCSRA &= 0xF8; ADCSRA |= clockPrescaler
```

Definition at line 145 of file [ha\\_m328p.h](#).

#### 10.27.2.14 ADC\_SELECT\_REF\_VOLTAGE

```
#define ADC_SELECT_REF_VOLTAGE(  
    refVoltage ) ADMUX &= 0x3F; ADMUX |= refVoltage << 6
```

Definition at line 124 of file [ha\\_m328p.h](#).

#### 10.27.2.15 ADC\_START\_CONVERSION

```
#define ADC_START_CONVERSION ADCSRA |= 1 << ADSC
```

Definition at line 136 of file [ha\\_m328p.h](#).

#### 10.27.2.16 ADC\_STOP\_CONVERSION

```
#define ADC_STOP_CONVERSION ADCSRA &= ~(1 << ADSC)
```

Definition at line 137 of file [ha\\_m328p.h](#).

#### 10.27.2.17 EXT\_INT\_DISABLE\_INT0

```
#define EXT_INT_DISABLE_INT0 EIMSK &= ~(1 << INT0)
```

Definition at line 159 of file [ha\\_m328p.h](#).

#### 10.27.2.18 EXT\_INT\_DISABLE\_INT1

```
#define EXT_INT_DISABLE_INT1 EIMSK &= ~(1 << INT1)
```

Definition at line 162 of file [ha\\_m328p.h](#).

#### 10.27.2.19 EXT\_INT\_DISABLE\_PIN\_CHANGE\_INTERRUPT

```
#define EXT_INT_DISABLE_PIN_CHANGE_INTERRUPT(  
    pinChangePort ) PCICR &= ~(1 << pinChangePort)
```

Definition at line 166 of file [ha\\_m328p.h](#).

#### 10.27.2.20 EXT\_INT\_DISABLE\_PIN\_CHANGE\_MASK\_PORTB

```
#define EXT_INT_DISABLE_PIN_CHANGE_MASK_PORTB(  
    pinChangePin ) PCMSK0 &= ~(1 << pinChangePin)
```

Definition at line 170 of file [ha\\_m328p.h](#).

#### 10.27.2.21 EXT\_INT\_DISABLE\_PIN\_CHANGE\_MASK\_PORTC

```
#define EXT_INT_DISABLE_PIN_CHANGE_MASK_PORTC(  
    pinChangePin ) PCMSK1 &= ~(1 << pinChangePin)
```

Definition at line 173 of file [ha\\_m328p.h](#).

#### 10.27.2.22 EXT\_INT\_DISABLE\_PIN\_CHANGE\_MASK\_PORTD

```
#define EXT_INT_DISABLE_PIN_CHANGE_MASK_PORTD(  
    pinChangePin ) PCMSK2 &= ~(1 << pinChangePin)
```

Definition at line 176 of file [ha\\_m328p.h](#).

#### 10.27.2.23 EXT\_INT\_ENABLE\_INT0

```
#define EXT_INT_ENABLE_INT0 EIMSK |= 1 << INT0
```

Definition at line 158 of file [ha\\_m328p.h](#).

#### 10.27.2.24 EXT\_INT\_ENABLE\_INT1

```
#define EXT_INT_ENABLE_INT1 EIMSK |= 1 << INT1
```

Definition at line 161 of file [ha\\_m328p.h](#).

#### 10.27.2.25 EXT\_INT\_ENABLE\_PIN\_CHANGE\_INTERRUPT

```
#define EXT_INT_ENABLE_PIN_CHANGE_INTERRUPT(  
    pinChangePort ) PCICR |= 1 << pinChangePort
```

Definition at line 165 of file [ha\\_m328p.h](#).

#### 10.27.2.26 EXT\_INT\_ENABLE\_PIN\_CHANGE\_MASK\_PORTB

```
#define EXT_INT_ENABLE_PIN_CHANGE_MASK_PORTB(  
    pinChangePin ) PCMSK0 |= 1 << pinChangePin
```

Definition at line 169 of file [ha\\_m328p.h](#).

#### 10.27.2.27 EXT\_INT\_ENABLE\_PIN\_CHANGE\_MASK\_PORTC

```
#define EXT_INT_ENABLE_PIN_CHANGE_MASK_PORTC(  
    pinChangePin ) PCMSK1 |= 1 << pinChangePin
```

Definition at line 172 of file [ha\\_m328p.h](#).

#### 10.27.2.28 EXT\_INT\_ENABLE\_PIN\_CHANGE\_MASK\_PORTD

```
#define EXT_INT_ENABLE_PIN_CHANGE_MASK_PORTD(  
    pinChangePin ) PCMSK2 |= 1 << pinChangePin
```

Definition at line 175 of file [ha\\_m328p.h](#).



### 10.27.2.29 EXT\_INT\_INT0\_INTERRUPT

```
#define EXT_INT_INT0_INTERRUPT INT0_vect
```

Definition at line 179 of file [ha\\_m328p.h](#).

### 10.27.2.30 EXT\_INT\_INT1\_INTERRUPT

```
#define EXT_INT_INT1_INTERRUPT INT1_vect
```

Definition at line 180 of file [ha\\_m328p.h](#).

### 10.27.2.31 EXT\_INT\_PIN\_CHANGE\_PORTB\_INTERRUPT

```
#define EXT_INT_PIN_CHANGE_PORTB_INTERRUPT PCINT0_vect
```

Definition at line 182 of file [ha\\_m328p.h](#).

### 10.27.2.32 EXT\_INT\_PIN\_CHANGE\_PORTC\_INTERRUPT

```
#define EXT_INT_PIN_CHANGE_PORTC_INTERRUPT PCINT1_vect
```

Definition at line 183 of file [ha\\_m328p.h](#).

### 10.27.2.33 EXT\_INT\_PIN\_CHANGE\_PORTD\_INTERRUPT

```
#define EXT_INT_PIN_CHANGE_PORTD_INTERRUPT PCINT2_vect
```

Definition at line 184 of file [ha\\_m328p.h](#).

### 10.27.2.34 EXT\_INT\_SET\_INT0\_SENSE\_CONTROL

```
#define EXT_INT_SET_INT0_SENSE_CONTROL(  
    senseControl ) EICRA &= 0xFC; EICRA |= senseControl
```

Definition at line 155 of file [ha\\_m328p.h](#).

#### 10.27.2.35 EXT\_INT\_SET\_INT1\_SENSE\_CONTROL

```
#define EXT_INT_SET_INT1_SENSE_CONTROL(  
    senseControl ) EICRA &= 0xF3; EICRA |= senseControl
```

Definition at line 156 of file [ha\\_m328p.h](#).

#### 10.27.2.36 MCU\_ADC\_DISABLE

```
#define MCU_ADC_DISABLE PRR |= 1 << PRADC
```

Definition at line 286 of file [ha\\_m328p.h](#).

#### 10.27.2.37 MCU\_ADC\_ENABLE

```
#define MCU_ADC_ENABLE PRR &= ~(1 << PRADC)
```

Definition at line 285 of file [ha\\_m328p.h](#).

#### 10.27.2.38 MCU\_BOD\_DISABLE

```
#define MCU_BOD_DISABLE MCUCR |= (1<<BODSE) | (1<<BODS); MCUCR &= ~(1<<BODSE);
```

Definition at line 288 of file [ha\\_m328p.h](#).

#### 10.27.2.39 MCU\_SELECT\_SLEEP\_MODE

```
#define MCU_SELECT_SLEEP_MODE(  
    sleepMode ) SMCR &= 0xF1; SMCR |= sleepMode << 1
```

Definition at line 262 of file [ha\\_m328p.h](#).

#### 10.27.2.40 MCU\_SLEEP\_DISABLE

```
#define MCU_SLEEP_DISABLE SMCR &= ~(1 << SE)
```

Definition at line 265 of file [ha\\_m328p.h](#).

#### 10.27.2.41 MCU\_SLEEP\_ENABLE

```
#define MCU_SLEEP_ENABLE SMCR |= 1 << SE
```

Definition at line 264 of file [ha\\_m328p.h](#).

#### 10.27.2.42 MCU\_SPI\_DISABLE

```
#define MCU_SPI_DISABLE PRR |= 1 << PRSPI
```

Definition at line 280 of file [ha\\_m328p.h](#).

#### 10.27.2.43 MCU\_SPI\_ENABLE

```
#define MCU_SPI_ENABLE PRR &= ~(1 << PRSPI)
```

Definition at line 279 of file [ha\\_m328p.h](#).

#### 10.27.2.44 MCU\_TIMER0\_DISABLE

```
#define MCU_TIMER0_DISABLE PRR |= 1 << PRTIM0
```

Definition at line 277 of file [ha\\_m328p.h](#).

#### 10.27.2.45 MCU\_TIMER0\_ENABLE

```
#define MCU_TIMER0_ENABLE PRR &= ~(1 << PRTIM0)
```

Definition at line 276 of file [ha\\_m328p.h](#).

#### 10.27.2.46 MCU\_TIMER1\_DISABLE

```
#define MCU_TIMER1_DISABLE PRR |= 1 << PRTIM1
```

Definition at line 274 of file [ha\\_m328p.h](#).

#### 10.27.2.47 MCU\_TIMER1\_ENABLE

```
#define MCU_TIMER1_ENABLE PRR &= ~(1 << PRTIM1)
```

Definition at line 273 of file [ha\\_m328p.h](#).

#### 10.27.2.48 MCU\_TIMER2\_DISABLE

```
#define MCU_TIMER2_DISABLE PRR |= 1 << PRTIM2
```

Definition at line 271 of file [ha\\_m328p.h](#).

#### 10.27.2.49 MCU\_TIMER2\_ENABLE

```
#define MCU_TIMER2_ENABLE PRR &= ~(1 << PRTIM2)
```

Definition at line 270 of file [ha\\_m328p.h](#).

#### 10.27.2.50 MCU\_TWI\_DISABLE

```
#define MCU_TWI_DISABLE PRR |= 1 << PRTWI
```

Definition at line 268 of file [ha\\_m328p.h](#).

#### 10.27.2.51 MCU\_TWI\_ENABLE

```
#define MCU_TWI_ENABLE PRR &= ~(1 << PRTWI)
```

Definition at line 267 of file [ha\\_m328p.h](#).

#### 10.27.2.52 MCU\_USART0\_DISABLE

```
#define MCU_USART0_DISABLE PRR |= 1 << PRUSART0
```

Definition at line 283 of file [ha\\_m328p.h](#).

### 10.27.2.53 MCU\_USART0\_ENABLE

```
#define MCU_USART0_ENABLE PRR &= ~(1 << PRUSART0)
```

Definition at line 282 of file [ha\\_m328p.h](#).

### 10.27.2.54 PUSHBUTTON\_DEBOUNCE\_TIME\_US

```
#define PUSHBUTTON_DEBOUNCE_TIME_US 1000
```

Definition at line 16 of file [ha\\_m328p.h](#).

### 10.27.2.55 PUSHBUTTON\_SAMPLING

```
#define PUSHBUTTON_SAMPLING 12
```

Definition at line 17 of file [ha\\_m328p.h](#).

### 10.27.2.56 SERVOMOTOR\_TIMER\_ANGLE\_COUNT

```
#define SERVOMOTOR_TIMER_ANGLE_COUNT(  
    angle,  
    out_min,  
    out_mid,  
    out_max ) (((out_min*(angle - 90L)*(angle - 180L))/16200L + (angle*out_max*(angle  
- 90L))/16200L - (angle*out_mid*(angle - 180L))/8100L))
```

Definition at line 295 of file [ha\\_m328p.h](#).

### 10.27.2.57 SERVOMOTOR\_TIMER\_PULSE\_WIDTH\_COUNT

```
#define SERVOMOTOR_TIMER_PULSE_WIDTH_COUNT(  
    pulseWidth,  
    clockPrescaler ) ((F_CPU/1000000UL) * (pulseWidth/ clockPrescaler))
```

Definition at line 293 of file [ha\\_m328p.h](#).

#### 10.27.2.58 SPI\_CONTROL\_REGISTER

```
#define SPI_CONTROL_REGISTER SPCR
```

Definition at line 313 of file [ha\\_m328p.h](#).

#### 10.27.2.59 SPI\_DATA\_REGISTER

```
#define SPI_DATA_REGISTER SPDR
```

Definition at line 314 of file [ha\\_m328p.h](#).

#### 10.27.2.60 SPI\_DISABLE

```
#define SPI_DISABLE SPCR &= ~(1 << SPE)
```

Definition at line 300 of file [ha\\_m328p.h](#).

#### 10.27.2.61 SPI\_DISABLE\_TRANSFER\_COMPLETE\_INTERRUPT

```
#define SPI_DISABLE_TRANSFER_COMPLETE_INTERRUPT SPCR &= ~(1 << SPIE)
```

Definition at line 318 of file [ha\\_m328p.h](#).

#### 10.27.2.62 SPI\_ENABLE

```
#define SPI_ENABLE SPCR |= 1 << SPE
```

Definition at line 299 of file [ha\\_m328p.h](#).

#### 10.27.2.63 SPI\_ENABLE\_TRANSFER\_COMPLETE\_INTERRUPT

```
#define SPI_ENABLE_TRANSFER_COMPLETE_INTERRUPT SPCR |= 1 << SPIE
```

Definition at line 317 of file [ha\\_m328p.h](#).

#### 10.27.2.64 SPI\_MASTER\_MODE

```
#define SPI_MASTER_MODE MSTR
```

Definition at line 312 of file [ha\\_m328p.h](#).

#### 10.27.2.65 SPI\_SELECT\_CLOCK\_PRESCALER

```
#define SPI_SELECT_CLOCK_PRESCALER(  
    clockPrescaler ) SPCR &= 0xFC; SPCR |= (clockPrescaler & 3); SPSR &= 0xFE; SPSR |=  
(clockPrescaler & 4) >> 2
```

Definition at line 306 of file [ha\\_m328p.h](#).

#### 10.27.2.66 SPI\_SELECT\_DATA\_MODE

```
#define SPI_SELECT_DATA_MODE(  
    dataMode ) SPCR &= 0xF3; SPCR |= dataMode << 2
```

Definition at line 305 of file [ha\\_m328p.h](#).

#### 10.27.2.67 SPI\_SELECT\_DATA\_ORDER

```
#define SPI_SELECT_DATA_ORDER(  
    dataOrder ) SPCR &= 0xDF; SPCR |= dataOrder << 5
```

Definition at line 307 of file [ha\\_m328p.h](#).

#### 10.27.2.68 SPI\_SELECT\_MASTER\_MODE

```
#define SPI_SELECT_MASTER_MODE SPCR |= 1 << MSTR
```

Definition at line 301 of file [ha\\_m328p.h](#).

#### 10.27.2.69 SPI\_SELECT\_SLAVE\_MODE

```
#define SPI_SELECT_SLAVE_MODE SPCR &= ~(1 << MSTR)
```

Definition at line 302 of file [ha\\_m328p.h](#).

#### 10.27.2.70 SPI\_STATUS\_REGISTER

```
#define SPI_STATUS_REGISTER SPSR
```

Definition at line 315 of file [ha\\_m328p.h](#).

#### 10.27.2.71 SPI\_TRANSFER\_COMPLETE

```
#define SPI_TRANSFER_COMPLETE SPIF
```

Definition at line 310 of file [ha\\_m328p.h](#).

#### 10.27.2.72 SPI\_TRANSFER\_COMPLETE\_INTERRUPT

```
#define SPI_TRANSFER_COMPLETE_INTERRUPT SPI_STC_vect
```

Definition at line 321 of file [ha\\_m328p.h](#).

#### 10.27.2.73 SPI\_WRITE\_COLLISION

```
#define SPI_WRITE_COLLISION WCOL
```

Definition at line 309 of file [ha\\_m328p.h](#).

#### 10.27.2.74 TIMER0\_COM\_CHANNEL\_A\_INTERRUPT

```
#define TIMER0_COM_CHANNEL_A_INTERRUPT TIMER0_COMPA_vect
```

Definition at line 239 of file [ha\\_m328p.h](#).

#### 10.27.2.75 TIMER0\_COM\_CHANNEL\_B\_INTERRUPT

```
#define TIMER0_COM_CHANNEL_B_INTERRUPT TIMER0_COMPB_vect
```

Definition at line 240 of file [ha\\_m328p.h](#).



**10.27.2.76 TIMER0\_DISABLE\_COM\_CHANNEL\_A\_INTERRUPT**

```
#define TIMER0_DISABLE_COM_CHANNEL_A_INTERRUPT TIMSK0 &= ~(1 << OCIE0A)
```

Definition at line 210 of file [ha\\_m328p.h](#).

**10.27.2.77 TIMER0\_DISABLE\_COM\_CHANNEL\_B\_INTERRUPT**

```
#define TIMER0_DISABLE_COM_CHANNEL_B_INTERRUPT TIMSK0 &= ~(1 << OCIE0B)
```

Definition at line 213 of file [ha\\_m328p.h](#).

**10.27.2.78 TIMER0\_DISABLE\_OVERFLOW\_INTERRUPT**

```
#define TIMER0_DISABLE_OVERFLOW_INTERRUPT TIMSK0 &= ~(1 << TOIE0)
```

Definition at line 216 of file [ha\\_m328p.h](#).

**10.27.2.79 TIMER0\_ENABLE\_COM\_CHANNEL\_A\_INTERRUPT**

```
#define TIMER0_ENABLE_COM_CHANNEL_A_INTERRUPT TIMSK0 |= 1 << OCIE0A
```

Definition at line 209 of file [ha\\_m328p.h](#).

**10.27.2.80 TIMER0\_ENABLE\_COM\_CHANNEL\_B\_INTERRUPT**

```
#define TIMER0_ENABLE_COM_CHANNEL_B_INTERRUPT TIMSK0 |= 1 << OCIE0B
```

Definition at line 212 of file [ha\\_m328p.h](#).

**10.27.2.81 TIMER0\_ENABLE\_OVERFLOW\_INTERRUPT**

```
#define TIMER0_ENABLE_OVERFLOW_INTERRUPT TIMSK0 |= 1 << TOIE0
```

Definition at line 215 of file [ha\\_m328p.h](#).

#### 10.27.2.82 TIMER0\_OVERFLOW\_INTERRUPT

```
#define TIMER0_OVERFLOW_INTERRUPT TIMER0_OVF_vect
```

Definition at line 241 of file [ha\\_m328p.h](#).

#### 10.27.2.83 TIMER0\_SELECT\_CLOCK\_SOURCE

```
#define TIMER0_SELECT_CLOCK_SOURCE(  
    clockSource ) TCCR0B &= 0xF8; TCCR0B |= clockSource
```

Definition at line 192 of file [ha\\_m328p.h](#).

#### 10.27.2.84 TIMER0\_SELECT\_COM\_CHANNEL\_A

```
#define TIMER0_SELECT_COM_CHANNEL_A(  
    compareOutputMode ) TCCR0A &= 0x3F; TCCR0A |= compareOutputMode << 6
```

Definition at line 200 of file [ha\\_m328p.h](#).

#### 10.27.2.85 TIMER0\_SELECT\_COM\_CHANNEL\_B

```
#define TIMER0_SELECT_COM_CHANNEL_B(  
    compareOutputMode ) TCCR0A &= 0xCF; TCCR0A |= compareOutputMode << 4
```

Definition at line 201 of file [ha\\_m328p.h](#).

#### 10.27.2.86 TIMER0\_SELECT\_OPERATION\_MODE

```
#define TIMER0_SELECT_OPERATION_MODE(  
    operationMode ) TCCR0A &= 0xFC; TCCR0A |= (operationMode & 3); TCCR0B &= 0xF7;  
TCCR0B |= (operationMode & 12) << 1
```

Definition at line 196 of file [ha\\_m328p.h](#).

#### 10.27.2.87 TIMER0\_STOP

```
#define TIMER0_STOP TCCR0B &= 0xF8
```

Definition at line 188 of file [ha\\_m328p.h](#).

**10.27.2.88 TIMER1\_COM\_CHANNEL\_A\_INTERRUPT**

```
#define TIMER1_COM_CHANNEL_A_INTERRUPT TIMER1_COMPA_vect
```

Definition at line 243 of file [ha\\_m328p.h](#).

**10.27.2.89 TIMER1\_COM\_CHANNEL\_B\_INTERRUPT**

```
#define TIMER1_COM_CHANNEL_B_INTERRUPT TIMER1_COMPB_vect
```

Definition at line 244 of file [ha\\_m328p.h](#).

**10.27.2.90 TIMER1\_DISABLE\_COM\_CHANNEL\_A\_INTERRUPT**

```
#define TIMER1_DISABLE_COM_CHANNEL_A_INTERRUPT TIMSK1 &= ~(1 << OCIE1A)
```

Definition at line 219 of file [ha\\_m328p.h](#).

**10.27.2.91 TIMER1\_DISABLE\_COM\_CHANNEL\_B\_INTERRUPT**

```
#define TIMER1_DISABLE_COM_CHANNEL_B_INTERRUPT TIMSK1 &= ~(1 << OCIE1B)
```

Definition at line 222 of file [ha\\_m328p.h](#).

**10.27.2.92 TIMER1\_DISABLE\_INPUT\_CAPTURE\_INTERRUPT**

```
#define TIMER1_DISABLE_INPUT_CAPTURE_INTERRUPT TIMSK1 &= ~(1 << ICIE1)
```

Definition at line 228 of file [ha\\_m328p.h](#).

**10.27.2.93 TIMER1\_DISABLE\_OVERFLOW\_INTERRUPT**

```
#define TIMER1_DISABLE_OVERFLOW_INTERRUPT TIMSK1 &= ~(1 << TOIE1)
```

Definition at line 225 of file [ha\\_m328p.h](#).

#### 10.27.2.94 TIMER1\_ENABLE\_COM\_CHANNEL\_A\_INTERRUPT

```
#define TIMER1_ENABLE_COM_CHANNEL_A_INTERRUPT TIMSK1 |= 1 << OCIE1A
```

Definition at line 218 of file [ha\\_m328p.h](#).

#### 10.27.2.95 TIMER1\_ENABLE\_COM\_CHANNEL\_B\_INTERRUPT

```
#define TIMER1_ENABLE_COM_CHANNEL_B_INTERRUPT TIMSK1 |= 1 << OCIE1B
```

Definition at line 221 of file [ha\\_m328p.h](#).

#### 10.27.2.96 TIMER1\_ENABLE\_INPUT\_CAPTURE\_INTERRUPT

```
#define TIMER1_ENABLE_INPUT_CAPTURE_INTERRUPT TIMSK1 |= 1 << ICIE1
```

Definition at line 227 of file [ha\\_m328p.h](#).

#### 10.27.2.97 TIMER1\_ENABLE\_OVERFLOW\_INTERRUPT

```
#define TIMER1_ENABLE_OVERFLOW_INTERRUPT TIMSK1 |= 1 << TOIE1
```

Definition at line 224 of file [ha\\_m328p.h](#).

#### 10.27.2.98 TIMER1\_INPUT\_CAPTURE\_INTERRUPT

```
#define TIMER1_INPUT_CAPTURE_INTERRUPT TIMER1_CAPT_vect
```

Definition at line 246 of file [ha\\_m328p.h](#).

#### 10.27.2.99 TIMER1\_OVERFLOW\_INTERRUPT

```
#define TIMER1_OVERFLOW_INTERRUPT TIMER1_OVF_vect
```

Definition at line 245 of file [ha\\_m328p.h](#).

#### 10.27.2.100 TIMER1\_SELECT\_CLOCK\_SOURCE

```
#define TIMER1_SELECT_CLOCK_SOURCE(  
    clockSource ) TCCR1B &= 0xF8; TCCR1B |= clockSource
```

Definition at line 193 of file [ha\\_m328p.h](#).

#### 10.27.2.101 TIMER1\_SELECT\_COM\_CHANNEL\_A

```
#define TIMER1_SELECT_COM_CHANNEL_A(  
    compareOutputMode ) TCCR1A &= 0x3F; TCCR1A |= compareOutputMode << 6
```

Definition at line 203 of file [ha\\_m328p.h](#).

#### 10.27.2.102 TIMER1\_SELECT\_COM\_CHANNEL\_B

```
#define TIMER1_SELECT_COM_CHANNEL_B(  
    compareOutputMode ) TCCR1A &= 0xCF; TCCR1A |= compareOutputMode << 4
```

Definition at line 204 of file [ha\\_m328p.h](#).

#### 10.27.2.103 TIMER1\_SELECT\_OPERATION\_MODE

```
#define TIMER1_SELECT_OPERATION_MODE(  
    operationMode ) TCCR1A &= 0xFC; TCCR1A |= (operationMode & 3); TCCR1B &= 0xE7;  
TCCR1B |= (operationMode & 12) << 1
```

Definition at line 197 of file [ha\\_m328p.h](#).

#### 10.27.2.104 TIMER1\_STOP

```
#define TIMER1_STOP TCCR1B &= 0xF8
```

Definition at line 189 of file [ha\\_m328p.h](#).

#### 10.27.2.105 TIMER2\_COM\_CHANNEL\_A\_INTERRUPT

```
#define TIMER2_COM_CHANNEL_A_INTERRUPT TIMER2_COMPA_vect
```

Definition at line 248 of file [ha\\_m328p.h](#).

**10.27.2.106 TIMER2\_COM\_CHANNEL\_B\_INTERRUPT**

```
#define TIMER2_COM_CHANNEL_B_INTERRUPT TIMER2_COMPB_vect
```

Definition at line 249 of file [ha\\_m328p.h](#).

**10.27.2.107 TIMER2\_DISABLE\_COM\_CHANNEL\_A\_INTERRUPT**

```
#define TIMER2_DISABLE_COM_CHANNEL_A_INTERRUPT TIMSK2 &= ~(1 << OCIE2A)
```

Definition at line 231 of file [ha\\_m328p.h](#).

**10.27.2.108 TIMER2\_DISABLE\_COM\_CHANNEL\_B\_INTERRUPT**

```
#define TIMER2_DISABLE_COM_CHANNEL_B_INTERRUPT TIMSK2 &= ~(1 << OCIE2B)
```

Definition at line 234 of file [ha\\_m328p.h](#).

**10.27.2.109 TIMER2\_DISABLE\_OVERFLOW\_INTERRUPT**

```
#define TIMER2_DISABLE_OVERFLOW_INTERRUPT TIMSK2 &= ~(1 << TOIE2)
```

Definition at line 237 of file [ha\\_m328p.h](#).

**10.27.2.110 TIMER2\_ENABLE\_COM\_CHANNEL\_A\_INTERRUPT**

```
#define TIMER2_ENABLE_COM_CHANNEL_A_INTERRUPT TIMSK2 |= 1 << OCIE2A
```

Definition at line 230 of file [ha\\_m328p.h](#).

**10.27.2.111 TIMER2\_ENABLE\_COM\_CHANNEL\_B\_INTERRUPT**

```
#define TIMER2_ENABLE_COM_CHANNEL_B_INTERRUPT TIMSK2 |= 1 << OCIE2B
```

Definition at line 233 of file [ha\\_m328p.h](#).

**10.27.2.112 TIMER2\_ENABLE\_OVERFLOW\_INTERRUPT**

```
#define TIMER2_ENABLE_OVERFLOW_INTERRUPT TIMSK2 |= 1 << TOIE2
```

Definition at line 236 of file [ha\\_m328p.h](#).

**10.27.2.113 TIMER2\_OVERFLOW\_INTERRUPT**

```
#define TIMER2_OVERFLOW_INTERRUPT TIMER2_OVF_vect
```

Definition at line 250 of file [ha\\_m328p.h](#).

**10.27.2.114 TIMER2\_SELECT\_CLOCK\_SOURCE**

```
#define TIMER2_SELECT_CLOCK_SOURCE(  
    clockSource ) TCCR2B &= 0xF8; TCCR2B |= clockSource
```

Definition at line 194 of file [ha\\_m328p.h](#).

**10.27.2.115 TIMER2\_SELECT\_COM\_CHANNEL\_A**

```
#define TIMER2_SELECT_COM_CHANNEL_A(  
    compareOutputMode ) TCCR2A &= 0x3F; TCCR2A |= compareOutputMode << 6
```

Definition at line 206 of file [ha\\_m328p.h](#).

**10.27.2.116 TIMER2\_SELECT\_COM\_CHANNEL\_B**

```
#define TIMER2_SELECT_COM_CHANNEL_B(  
    compareOutputMode ) TCCR2A &= 0xCF; TCCR2A |= compareOutputMode << 4
```

Definition at line 207 of file [ha\\_m328p.h](#).

**10.27.2.117 TIMER2\_SELECT\_OPERATION\_MODE**

```
#define TIMER2_SELECT_OPERATION_MODE(  
    operationMode ) TCCR2A &= 0xFC; TCCR2A |= (operationMode & 3); TCCR2B &= 0xF7;  
TCCR2B |= (operationMode & 12) << 1
```

Definition at line 198 of file [ha\\_m328p.h](#).

**10.27.2.118 TIMER2\_STOP**

```
#define TIMER2_STOP TCCR2B &= 0xF8
```

Definition at line 190 of file [ha\\_m328p.h](#).

**10.27.2.119 USART0\_CONTROL\_STATUS\_REGISTER**

```
#define USART0_CONTROL_STATUS_REGISTER UCSR0A
```

Definition at line 91 of file [ha\\_m328p.h](#).

**10.27.2.120 USART0\_DATA\_OVERRUN**

```
#define USART0_DATA_OVERRUN DOR0
```

Definition at line 99 of file [ha\\_m328p.h](#).

**10.27.2.121 USART0\_DATA\_REGISTER**

```
#define USART0_DATA_REGISTER UDR0
```

Definition at line 92 of file [ha\\_m328p.h](#).

**10.27.2.122 USART0\_DATA\_REGISTER\_EMPTY\_INTERRUPT**

```
#define USART0_DATA_REGISTER_EMPTY_INTERRUPT USART_UDRE_vect
```

Definition at line 119 of file [ha\\_m328p.h](#).

**10.27.2.123 USART0\_DISABLE\_DATA\_REGISTER\_EMPTY\_INTERRUPT**

```
#define USART0_DISABLE_DATA_REGISTER_EMPTY_INTERRUPT UCSR0B &= ~(1 << UDRIE0)
```

Definition at line 105 of file [ha\\_m328p.h](#).



**10.27.2.124 USART0\_DISABLE\_DOUBLE\_SPEED\_MODE [1/2]**

```
#define USART0_DISABLE_DOUBLE_SPEED_MODE
```

Definition at line 40 of file [ha\\_m328p.h](#).

**10.27.2.125 USART0\_DISABLE\_DOUBLE\_SPEED\_MODE [2/2]**

```
#define USART0_DISABLE_DOUBLE_SPEED_MODE UCSRA &= ~(1 << U2X0)
```

Definition at line 40 of file [ha\\_m328p.h](#).

**10.27.2.126 USART0\_DISABLE\_PARITY\_MODE**

```
#define USART0_DISABLE_PARITY_MODE UCSRC &= ~((1 << UPM01) | (1 << UPM00))
```

Definition at line 45 of file [ha\\_m328p.h](#).

**10.27.2.127 USART0\_DISABLE\_RECEIVE\_COMPLETE\_INTERRUPT**

```
#define USART0_DISABLE_RECEIVE_COMPLETE_INTERRUPT UCSRB &= ~(1 << RXCIE0)
```

Definition at line 108 of file [ha\\_m328p.h](#).

**10.27.2.128 USART0\_DISABLE\_RECEIVER**

```
#define USART0_DISABLE_RECEIVER UCSRB &= ~(1 << RXEN0)
```

Definition at line 87 of file [ha\\_m328p.h](#).

**10.27.2.129 USART0\_DISABLE\_TRANSMIT\_COMPLETE\_INTERRUPT**

```
#define USART0_DISABLE_TRANSMIT_COMPLETE_INTERRUPT UCSRB &= ~(1 << TXCIE0)
```

Definition at line 111 of file [ha\\_m328p.h](#).

**10.27.2.130 USART0\_DISABLE\_TRANSMITTER**

```
#define USART0_DISABLE_TRANSMITTER UCSRB &= ~(1 << TXEN0)
```

Definition at line 82 of file [ha\\_m328p.h](#).

**10.27.2.131 USART0\_ENABLE\_ASYNC\_TRANSMISSION\_MODE**

```
#define USART0_ENABLE_ASYNC_TRANSMISSION_MODE UCSRC &= ~( (1 << UMSEL01) | (1 << UMSEL00) )
```

Definition at line 26 of file [ha\\_m328p.h](#).

**10.27.2.132 USART0\_ENABLE\_DATA\_REGISTER\_EMPTY\_INTERRUPT**

```
#define USART0_ENABLE_DATA_REGISTER_EMPTY_INTERRUPT UCSRB |= 1 << UDRIE0
```

Definition at line 104 of file [ha\\_m328p.h](#).

**10.27.2.133 USART0\_ENABLE\_DOUBLE\_SPEED\_MODE**

```
#define USART0_ENABLE_DOUBLE_SPEED_MODE
```

Definition at line 31 of file [ha\\_m328p.h](#).

**10.27.2.134 USART0\_ENABLE\_EVEN\_PARITY\_MODE**

```
#define USART0_ENABLE_EVEN_PARITY_MODE UCSRC = (1 << UPM01) | (UCSRC & ~(1 << UPM00))
```

Definition at line 43 of file [ha\\_m328p.h](#).

**10.27.2.135 USART0\_ENABLE\_MASTER\_SPI\_MODE**

```
#define USART0_ENABLE_MASTER_SPI_MODE UCSRC |= (1 << UMSEL01) | (1 << UMSEL00)
```

Definition at line 28 of file [ha\\_m328p.h](#).

**10.27.2.136 USART0\_ENABLE\_ODD\_PARITY\_MODE**

```
#define USART0_ENABLE_ODD_PARITY_MODE UCSR0C |= ((1 << UPM01) | (1 << UPM00))
```

Definition at line 44 of file [ha\\_m328p.h](#).

**10.27.2.137 USART0\_ENABLE\_RECEIVE\_COMPLETE\_INTERRUPT**

```
#define USART0_ENABLE_RECEIVE_COMPLETE_INTERRUPT UCSR0B |= 1 << RXCIE0
```

Definition at line 107 of file [ha\\_m328p.h](#).

**10.27.2.138 USART0\_ENABLE\_RECEIVER**

```
#define USART0_ENABLE_RECEIVER UCSR0B |= 1 << RXEN0
```

Definition at line 86 of file [ha\\_m328p.h](#).

**10.27.2.139 USART0\_ENABLE\_SYNC\_TRANSMISSION\_MODE**

```
#define USART0_ENABLE_SYNC_TRANSMISSION_MODE UCSR0C = (UCSR0C & ~(1 << UMSEL01)) | (1 << UMSEL00)
```

Definition at line 27 of file [ha\\_m328p.h](#).

**10.27.2.140 USART0\_ENABLE\_TRANSMIT\_COMPLETE\_INTERRUPT**

```
#define USART0_ENABLE_TRANSMIT_COMPLETE_INTERRUPT UCSR0B |= 1 << TXCIE0
```

Definition at line 110 of file [ha\\_m328p.h](#).

**10.27.2.141 USART0\_ENABLE\_TRANSMITTER**

```
#define USART0_ENABLE_TRANSMITTER UCSR0B |= 1 << TXEN0
```

Definition at line 81 of file [ha\\_m328p.h](#).

**10.27.2.142 USART0\_FRAME\_ERROR**

```
#define USART0_FRAME_ERROR FE0
```

Definition at line 97 of file [ha\\_m328p.h](#).

**10.27.2.143 USART0\_PARITY\_ERROR**

```
#define USART0_PARITY_ERROR UPE0
```

Definition at line 101 of file [ha\\_m328p.h](#).

**10.27.2.144 USART0\_RECEIVE\_COMPLETE\_INTERRUPT**

```
#define USART0_RECEIVE_COMPLETE_INTERRUPT USART_RX_vect
```

Definition at line 115 of file [ha\\_m328p.h](#).

**10.27.2.145 USART0\_SET\_5BIT\_FRAME\_SIZE**

```
#define USART0_SET_5BIT_FRAME_SIZE
```

**Value:**

```
do { \
    UCSR0C &= ~( (1 << UCSZ01) | (1 << UCSZ00)); \
    UCSR0B &= ~(1 << UCSZ02); \
} while (0)
```

Definition at line 74 of file [ha\\_m328p.h](#).

**10.27.2.146 USART0\_SET\_6BIT\_FRAME\_SIZE**

```
#define USART0_SET_6BIT_FRAME_SIZE
```

**Value:**

```
do { \
    UCSR0C = (UCSR0C & ~(1 << UCSZ01)) | (1 << UCSZ00); \
    UCSR0B &= ~(1 << UCSZ02); \
} while (0)
```

Definition at line 69 of file [ha\\_m328p.h](#).

**10.27.2.147 USART0\_SET\_7BIT\_FRAME\_SIZE**

```
#define USART0_SET_7BIT_FRAME_SIZE
```

**Value:**

```
do { \
    UCSR0C = (1 « UCSZ01) | (UCSR0C & ~(1 « UCSZ00)); \
    UCSR0B &= ~(1 « UCSZ02); \
} while (0)
```

Definition at line 64 of file [ha\\_m328p.h](#).

**10.27.2.148 USART0\_SET\_8BIT\_FRAME\_SIZE**

```
#define USART0_SET_8BIT_FRAME_SIZE
```

**Value:**

```
do { \
    UCSR0C |= ((1 « UCSZ01) | (1 « UCSZ00)); \
    UCSR0B &= ~(1 « UCSZ02); \
} while (0)
```

Definition at line 58 of file [ha\\_m328p.h](#).

**10.27.2.149 USART0\_SET\_9BIT\_FRAME\_SIZE**

```
#define USART0_SET_9BIT_FRAME_SIZE
```

**Value:**

```
do { \
    UCSR0C |= ((1 « UCSZ01) | (1 « UCSZ00)); \
    UCSR0B |= (1 « UCSZ02); \
} while (0)
```

Definition at line 52 of file [ha\\_m328p.h](#).

**10.27.2.150 USART0\_SET\_BAUDRATE\_HIGH\_REGISTER**

```
#define USART0_SET_BAUDRATE_HIGH_REGISTER UBRR0H = UBRRH_VALUE
```

Definition at line 23 of file [ha\\_m328p.h](#).

**10.27.2.151 USART0\_SET\_BAUDRATE\_LOW\_REGISTER**

```
#define USART0_SET_BAUDRATE_LOW_REGISTER UBRR0L = UBRL_VALUE
```

Definition at line 24 of file [ha\\_m328p.h](#).

#### 10.27.2.152 USART0\_SET\_ONE\_STOP\_BIT

```
#define USART0_SET_ONE_STOP_BIT UCSR0C &= ~(1 << USBS0)
```

Definition at line 48 of file [ha\\_m328p.h](#).

#### 10.27.2.153 USART0\_SET\_TWO\_STOP\_BITS

```
#define USART0_SET_TWO_STOP_BITS UCSR0C |= (1 << USBS0)
```

Definition at line 49 of file [ha\\_m328p.h](#).

#### 10.27.2.154 USART0\_TRANSMIT\_COMPLETE\_INTERRUPT

```
#define USART0_TRANSMIT_COMPLETE_INTERRUPT USART_TX_vect
```

Definition at line 117 of file [ha\\_m328p.h](#).

#### 10.27.2.155 WATCHDOG\_SELECT\_TIMEOUT

```
#define WATCHDOG_SELECT_TIMEOUT(  
    timeOut ) WDTCR |= (1<<WDCE) | (1<<WDE); WDTCR = (0<<WDIE) | (0<<WDE) | time↵  
Out;
```

Definition at line 254 of file [ha\\_m328p.h](#).

#### 10.27.2.156 WATCHDOG\_START

```
#define WATCHDOG_START(  
    operationMode,  
    timeOut ) MCUSR &= ~(1<<WDRF); WDTCR |= (1<<WDCE) | (1<<WDE); WDTCR = operation↵  
Mode | timeOut;
```

Definition at line 255 of file [ha\\_m328p.h](#).

#### 10.27.2.157 WATCHDOG\_STOP

```
#define WATCHDOG_STOP MCUSR &= ~(1<<WDRF); WDTCR |= (1<<WDCE) | (1<<WDE); WDTCR = 0x00;
```

Definition at line 256 of file [ha\\_m328p.h](#).

## 10.27.2.158 WATCHDOG\_TIMEOUT\_INTERRUPT

```
#define WATCHDOG_TIMEOUT_INTERRUPT WDT_vect
```

Definition at line 257 of file [ha\\_m328p.h](#).

## 10.28 ha\_m328p.h

```
00001
00008 #ifndef HAM329P_H
00009 #define HAM329P_H
00010
00011 #ifndef HABASE_H
00012     #error "don't use this file directly! Please include only ha_base.h!"
00013 #endif
00014
00015 // PushButton defines
00016 #define PUSHBUTTON_DEBOUNCE_TIME_US 1000 // microseconds to wait for next check
00017 #define PUSHBUTTON_SAMPLING 12 // number of times a push button must be pressed
00018
00019
00020 // USART defines
00021 #include <util/setbaud.h>
00022
00023 #define USART0_SET_BAUDRATE_HIGH_REGISTER UBRR0H = UBRRH_VALUE
00024 #define USART0_SET_BAUDRATE_LOW_REGISTER UBRR0L = UBRL_VALUE
00025
00026 #define USART0_ENABLE_ASYNC_TRANSMISSION_MODE UCSR0C &= ~(1 < UMSEL01) | (1 < UMSEL00)
00027 #define USART0_ENABLE_SYNC_TRANSMISSION_MODE UCSR0C = (UCSR0C & ~(1 < UMSEL01)) | (1 < UMSEL00)
00028 #define USART0_ENABLE_MASTER_SPI_MODE UCSR0C |= (1 < UMSEL01) | (1 < UMSEL00)
00029
00030
00031 #define USART0_ENABLE_DOUBLE_SPEED_MODE
00032 #define USART0_DISABLE_DOUBLE_SPEED_MODE
00033
00034 #if USE_2X
00035     #undef USART0_ENABLE_DOUBLE_SPEED_MODE
00036     #define USART0_ENABLE_DOUBLE_SPEED_MODE UCSR0A |= (1 < U2X0)
00037     #warning "double speed operation activated"
00038 #else
00039     #undef USART0_DISABLE_DOUBLE_SPEED_MODE
00040     #define USART0_DISABLE_DOUBLE_SPEED_MODE UCSR0A &= ~(1 < U2X0)
00041 #endif
00042
00043 #define USART0_ENABLE_EVEN_PARITY_MODE UCSR0C = (1 < UPM01) | (UCSR0C & ~(1 < UPM00))
00044 #define USART0_ENABLE_ODD_PARITY_MODE UCSR0C |= ((1 < UPM01) | (1 < UPM00))
00045 #define USART0_DISABLE_PARITY_MODE UCSR0C &= ~(1 < UPM01) | (1 < UPM00)
00046
00047
00048 #define USART0_SET_ONE_STOP_BIT UCSR0C &= ~(1 < USBS0)
00049 #define USART0_SET_TWO_STOP_BITS UCSR0C |= (1 < USBS0)
00050
00051
00052 #define USART0_SET_9BIT_FRAME_SIZE \
00053     do { \
00054         UCSR0C |= ((1 < UCSZ01) | (1 < UCSZ00)); \
00055         UCSR0B |= (1 < UCSZ02); \
00056     } while (0)
00057
00058 #define USART0_SET_8BIT_FRAME_SIZE \
00059     do { \
00060         UCSR0C |= ((1 < UCSZ01) | (1 < UCSZ00)); \
00061         UCSR0B &= ~(1 < UCSZ02); \
00062     } while (0)
00063
00064 #define USART0_SET_7BIT_FRAME_SIZE \
00065     do { \
00066         UCSR0C = (1 < UCSZ01) | (UCSR0C & ~(1 < UCSZ00)); \
00067         UCSR0B &= ~(1 < UCSZ02); \
00068     } while (0)
00069 #define USART0_SET_6BIT_FRAME_SIZE \
00070     do { \
00071         UCSR0C = (UCSR0C & ~(1 < UCSZ01)) | (1 < UCSZ00); \
00072         UCSR0B &= ~(1 < UCSZ02); \
00073     } while (0)
00074 #define USART0_SET_5BIT_FRAME_SIZE \
00075     do { \
00076         UCSR0C &= ~(1 < UCSZ01) | (1 < UCSZ00); \
00077         UCSR0B &= ~(1 < UCSZ02); \
00078     } while (0)
```

```

00079
00080
00081 #define USART0_ENABLE_TRANSMITTER UCSRB |= 1 < TXEN0
00082 #define USART0_DISABLE_TRANSMITTER UCSRB &= ~(1 < TXEN0)
00083
00084
00085
00086 #define USART0_ENABLE_RECEIVER UCSRB |= 1 < RXEN0
00087 #define USART0_DISABLE_RECEIVER UCSRB &= ~(1 < RXEN0)
00088
00089
00090
00091 #define USART0_CONTROL_STATUS_REGISTER UCSRA
00092 #define USART0_DATA_REGISTER UDRO
00093
00094
00095
00096
00097 #define USART0_FRAME_ERROR FE0
00098
00099 #define USART0_DATA_OVERRUN DOR0
00100
00101 #define USART0_PARITY_ERROR UPE0
00102
00103
00104 #define USART0_ENABLE_DATA_REGISTER_EMPTY_INTERRUPT UCSRB |= 1 < UDRIE0
00105 #define USART0_DISABLE_DATA_REGISTER_EMPTY_INTERRUPT UCSRB &= ~(1 < UDRIE0)
00106
00107 #define USART0_ENABLE_RECEIVE_COMPLETE_INTERRUPT UCSRB |= 1 < RXCIE0
00108 #define USART0_DISABLE_RECEIVE_COMPLETE_INTERRUPT UCSRB &= ~(1 < RXCIE0)
00109
00110 #define USART0_ENABLE_TRANSMIT_COMPLETE_INTERRUPT UCSRB |= 1 < TXCIE0
00111 #define USART0_DISABLE_TRANSMIT_COMPLETE_INTERRUPT UCSRB &= ~(1 < TXCIE0)
00112
00113
00114
00115 #define USART0_RECEIVE_COMPLETE_INTERRUPT USART_RX_vect
00116
00117 #define USART0_TRANSMIT_COMPLETE_INTERRUPT USART_TX_vect
00118
00119 #define USART0_DATA_REGISTER_EMPTY_INTERRUPT USART_UDRE_vect
00120
00121
00122 // ADC defines
00123
00124 #define ADC_SELECT_REF_VOLTAGE(refVoltage) ADMUX &= 0x3F; ADMUX |= refVoltage < 6
00125
00126 #define ADC_ADJUST_RESULT_LEFT ADMUX |= 1 < ADLAR
00127 #define ADC_ADJUST_RESULT_RIGHT ADMUX &= ~(1 < ADLAR)
00128
00129 #define ADC_SELECT_ANALOG_INPUT(pinNumber) ADMUX &= 0xF0; ADMUX |= pinNumber
00130 #define ADC_DISABLE_DIGITAL_INPUT_REGISTER(pinNumber) DIDR0 &= 0xC0; DIDR0 |= pinNumber
00131
00132
00133 #define ADC_ENABLE ADCSRA |= 1 < ADEN
00134 #define ADC_DISABLE ADCSRA &= ~(1 < ADEN)
00135
00136 #define ADC_START_CONVERSION ADCSRA |= 1 < ADSC
00137 #define ADC_STOP_CONVERSION ADCSRA &= ~(1 < ADSC)
00138
00139 #define ADC_ENABLE_AUTOTRIGGER ADCSRA |= 1 < ADIFSC
00140 #define ADC_DISABLE_AUTOTRIGGER ADCSRA &= ~(1 < ADIFSC)
00141
00142 #define ADC_ENABLE_CONVERSION_COMPLETE_INTERRUPT ADCSRA |= 1 < ADIF
00143 #define ADC_DISABLE_CONVERSION_COMPLETE_INTERRUPT ADCSRA &= ~(1 < ADIF)
00144
00145 #define ADC_SELECT_CLOCK_PRESCALER(clockPrescaler) ADCSRA &= 0xF8; ADCSRA |= clockPrescaler
00146
00147
00148 #define ADC_SELECT_AUTO_TRIGGER_SOURCE(triggerSource) ADCSRB &= 0xF8; ADCSRB |=
    triggerSource
00149
00150
00151 #define ADC_CONVERSION_COMPLETE_INTERRUPT ADC_vect
00152
00153 // Externally Triggered Interrupts defines
00154
00155 #define EXT_INT_SET_INT0_SENSE_CONTROL(senseControl) EICRA &= 0xFC; EICRA |= senseControl
00156 #define EXT_INT_SET_INT1_SENSE_CONTROL(senseControl) EICRA &= 0xF3; EICRA |= senseControl
00157
00158 #define EXT_INT_ENABLE_INT0 EIMSK |= 1 < INT0
00159 #define EXT_INT_DISABLE_INT0 EIMSK &= ~(1 < INT0)
00160
00161 #define EXT_INT_ENABLE_INT1 EIMSK |= 1 < INT1
00162 #define EXT_INT_DISABLE_INT1 EIMSK &= ~(1 < INT1)
00163
00164

```



```

00165 #define EXT_INT_ENABLE_PIN_CHANGE_INTERRUPT(pinChangePort)    PCICR |= 1 « pinChangePort
00166 #define EXT_INT_DISABLE_PIN_CHANGE_INTERRUPT(pinChangePort)    PCICR &= ~(1 « pinChangePort)
00167
00168
00169 #define EXT_INT_ENABLE_PIN_CHANGE_MASK_PORTB(pinChangePin)      PCMSK0 |= 1 « pinChangePin
00170 #define EXT_INT_DISABLE_PIN_CHANGE_MASK_PORTB(pinChangePin)    PCMSK0 &= ~(1 « pinChangePin)
00171
00172 #define EXT_INT_ENABLE_PIN_CHANGE_MASK_PORTC(pinChangePin)      PCMSK1 |= 1 « pinChangePin
00173 #define EXT_INT_DISABLE_PIN_CHANGE_MASK_PORTC(pinChangePin)    PCMSK1 &= ~(1 « pinChangePin)
00174
00175 #define EXT_INT_ENABLE_PIN_CHANGE_MASK_PORTD(pinChangePin)      PCMSK2 |= 1 « pinChangePin
00176 #define EXT_INT_DISABLE_PIN_CHANGE_MASK_PORTD(pinChangePin)    PCMSK2 &= ~(1 « pinChangePin)
00177
00178
00179 #define EXT_INT_INT0_INTERRUPT    INT0_vect
00180 #define EXT_INT_INT1_INTERRUPT    INT1_vect
00181
00182 #define EXT_INT_PIN_CHANGE_PORTB_INTERRUPT    PCINT0_vect
00183 #define EXT_INT_PIN_CHANGE_PORTC_INTERRUPT    PCINT1_vect
00184 #define EXT_INT_PIN_CHANGE_PORTD_INTERRUPT    PCINT2_vect
00185
00186 // Timers/Counters defines
00187
00188 #define TIMER0_STOP TCCR0B &= 0xF8
00189 #define TIMER1_STOP TCCR1B &= 0xF8
00190 #define TIMER2_STOP TCCR2B &= 0xF8
00191
00192 #define TIMER0_SELECT_CLOCK_SOURCE(clockSource) TCCR0B &= 0xF8; TCCR0B |= clockSource
00193 #define TIMER1_SELECT_CLOCK_SOURCE(clockSource) TCCR1B &= 0xF8; TCCR1B |= clockSource
00194 #define TIMER2_SELECT_CLOCK_SOURCE(clockSource) TCCR2B &= 0xF8; TCCR2B |= clockSource
00195
00196 #define TIMER0_SELECT_OPERATION_MODE(operationMode) TCCR0A &= 0xFC; TCCR0A |= (operationMode & 3);
00197 #define TIMER1_SELECT_OPERATION_MODE(operationMode) TCCR1A &= 0xFC; TCCR1A |= (operationMode & 3);
00198 #define TIMER2_SELECT_OPERATION_MODE(operationMode) TCCR2A &= 0xFC; TCCR2A |= (operationMode & 3);
00199 #define TCCR0B &= 0xF7; TCCR0B |= (operationMode & 12) « 1
00200 #define TCCR1B &= 0xE7; TCCR1B |= (operationMode & 12) « 1
00201 #define TCCR2B &= 0xF7; TCCR2B |= (operationMode & 12) « 1
00202
00203 #define TIMER0_SELECT_COM_CHANNEL_A(compareOutputMode) TCCR0A &= 0x3F; TCCR0A |= compareOutputMode « 6
00204 #define TIMER0_SELECT_COM_CHANNEL_B(compareOutputMode) TCCR0A &= 0xCF; TCCR0A |= compareOutputMode « 4
00205
00206 #define TIMER1_SELECT_COM_CHANNEL_A(compareOutputMode) TCCR1A &= 0x3F; TCCR1A |= compareOutputMode « 6
00207 #define TIMER1_SELECT_COM_CHANNEL_B(compareOutputMode) TCCR1A &= 0xCF; TCCR1A |= compareOutputMode « 4
00208
00209 #define TIMER2_SELECT_COM_CHANNEL_A(compareOutputMode) TCCR2A &= 0x3F; TCCR2A |= compareOutputMode « 6
00210 #define TIMER2_SELECT_COM_CHANNEL_B(compareOutputMode) TCCR2A &= 0xCF; TCCR2A |= compareOutputMode « 4
00211
00212 #define TIMER0_ENABLE_COM_CHANNEL_A_INTERRUPT TIMSK0 |= 1 « OCIE0A
00213 #define TIMER0_DISABLE_COM_CHANNEL_A_INTERRUPT TIMSK0 &= ~(1 « OCIE0A)
00214
00215 #define TIMER0_ENABLE_COM_CHANNEL_B_INTERRUPT TIMSK0 |= 1 « OCIE0B
00216 #define TIMER0_DISABLE_COM_CHANNEL_B_INTERRUPT TIMSK0 &= ~(1 « OCIE0B)
00217
00218 #define TIMER0_ENABLE_OVERFLOW_INTERRUPT TIMSK0 |= 1 « TOIE0
00219 #define TIMER0_DISABLE_OVERFLOW_INTERRUPT TIMSK0 &= ~(1 « TOIE0)
00220
00221 #define TIMER1_ENABLE_COM_CHANNEL_A_INTERRUPT TIMSK1 |= 1 « OCIE1A
00222 #define TIMER1_DISABLE_COM_CHANNEL_A_INTERRUPT TIMSK1 &= ~(1 « OCIE1A)
00223
00224 #define TIMER1_ENABLE_COM_CHANNEL_B_INTERRUPT TIMSK1 |= 1 « OCIE1B
00225 #define TIMER1_DISABLE_COM_CHANNEL_B_INTERRUPT TIMSK1 &= ~(1 « OCIE1B)
00226
00227 #define TIMER1_ENABLE_OVERFLOW_INTERRUPT TIMSK1 |= 1 « TOIE1
00228 #define TIMER1_DISABLE_OVERFLOW_INTERRUPT TIMSK1 &= ~(1 « TOIE1)
00229
00230 #define TIMER1_ENABLE_INPUT_CAPTURE_INTERRUPT TIMSK1 |= 1 « ICIE1
00231 #define TIMER1_DISABLE_INPUT_CAPTURE_INTERRUPT TIMSK1 &= ~(1 « ICIE1)
00232
00233 #define TIMER2_ENABLE_COM_CHANNEL_A_INTERRUPT TIMSK2 |= 1 « OCIE2A
00234 #define TIMER2_DISABLE_COM_CHANNEL_A_INTERRUPT TIMSK2 &= ~(1 « OCIE2A)
00235
00236 #define TIMER2_ENABLE_COM_CHANNEL_B_INTERRUPT TIMSK2 |= 1 « OCIE2B
00237 #define TIMER2_DISABLE_COM_CHANNEL_B_INTERRUPT TIMSK2 &= ~(1 « OCIE2B)
00238
00239 #define TIMER2_ENABLE_OVERFLOW_INTERRUPT TIMSK2 |= 1 « TOIE2
00240 #define TIMER2_DISABLE_OVERFLOW_INTERRUPT TIMSK2 &= ~(1 « TOIE2)
00241
00242 #define TIMER0_COM_CHANNEL_A_INTERRUPT    TIMER0_COMPA_vect
00243 #define TIMER0_COM_CHANNEL_B_INTERRUPT    TIMER0_COMPB_vect
00244 #define TIMER0_OVERFLOW_INTERRUPT         TIMER0_OVF_vect
00245 #define TIMER1_COM_CHANNEL_A_INTERRUPT    TIMER1_COMPA_vect
00246 #define TIMER1_COM_CHANNEL_B_INTERRUPT    TIMER1_COMPB_vect
00247 #define TIMER1_OVERFLOW_INTERRUPT         TIMER1_OVF_vect
00248 #define TIMER1_INPUT_CAPTURE_INTERRUPT    TIMER1_CAPT_vect
00249 #define TIMER2_COM_CHANNEL_A_INTERRUPT    TIMER2_COMPA_vect

```

```

00249 #define    TIMER2_COM_CHANNEL_B_INTERRUPT    TIMER2_COMPB_vect
00250 #define    TIMER2_OVERFLOW_INTERRUPT    TIMER2_OVF_vect
00251
00252 // Watchdog Timer defines
00253
00254 #define WATCHDOG_SELECT_TIMEOUT(timeOut) WDTCSR |= (1<WDCE) | (1<WDE); WDTCSR = (0<WDIE) | (0<WDE) |
    timeOut;
00255 #define WATCHDOG_START(operationMode,timeOut) MCUSR &= ~(1<WDRF); WDTCSR |= (1<WDCE) | (1<WDE); WDTCSR
    = operationMode | timeOut;
00256 #define WATCHDOG_STOP MCUSR &= ~(1<WDRF); WDTCSR |= (1<WDCE) | (1<WDE); WDTCSR = 0x00;
00257 #define WATCHDOG_TIMEOUT_INTERRUPT    WDT_vect
00258
00259 // MCU defines
00260
00261
00262 #define MCU_SELECT_SLEEP_MODE(sleepMode) SMCR &= 0xF1; SMCR |= sleepMode << 1
00263
00264 #define MCU_SLEEP_ENABLE SMCR |= 1 << SE
00265 #define MCU_SLEEP_DISABLE SMCR &= ~(1 << SE)
00266
00267 #define MCU_TWI_ENABLE PRR &= ~(1 << PRTWI)
00268 #define MCU_TWI_DISABLE PRR |= 1 << PRTWI
00269
00270 #define MCU_TIMER2_ENABLE PRR &= ~(1 << PRTIM2)
00271 #define MCU_TIMER2_DISABLE PRR |= 1 << PRTIM2
00272
00273 #define MCU_TIMER1_ENABLE PRR &= ~(1 << PRTIM1)
00274 #define MCU_TIMER1_DISABLE PRR |= 1 << PRTIM1
00275
00276 #define MCU_TIMER0_ENABLE PRR &= ~(1 << PRTIM0)
00277 #define MCU_TIMER0_DISABLE PRR |= 1 << PRTIM0
00278
00279 #define MCU_SPI_ENABLE PRR &= ~(1 << PRSPI)
00280 #define MCU_SPI_DISABLE PRR |= 1 << PRSPI
00281
00282 #define MCU_USART0_ENABLE PRR &= ~(1 << PRUSART0)
00283 #define MCU_USART0_DISABLE PRR |= 1 << PRUSART0
00284
00285 #define MCU_ADC_ENABLE PRR &= ~(1 << PRADC)
00286 #define MCU_ADC_DISABLE PRR |= 1 << PRADC
00287
00288 #define MCU_BOD_DISABLE MCUCR |= (1<BODSE) | (1<BODS); MCUCR &= ~(1<BODSE);
00289
00290
00291 // ServoMotor defines
00292
00293 #define SERVO MOTOR_TIMER_PULSE_WIDTH_COUNT(pulseWidth,clockPrescaler) ((F_CPU/1000000UL) *
    (pulseWidth/ clockPrescaler))
00294
00295 #define SERVO MOTOR_TIMER_ANGLE_COUNT(angle,out_min,out_mid,out_max) (((out_min*(angle - 90L)*(angle -
    180L))/16200L + (angle*out_max*(angle - 90L))/16200L - (angle*out_mid*(angle - 180L))/8100L))
00296
00297 // SPI defines
00298
00299 #define    SPI_ENABLE                SPCR |= 1 << SPE
00300 #define    SPI_DISABLE                SPCR &= ~(1 << SPE)
00301 #define    SPI_SELECT_MASTER_MODE    SPCR |= 1 << MSTR
00302 #define    SPI_SELECT_SLAVE_MODE    SPCR &= ~(1 << MSTR)
00303
00304
00305 #define    SPI_SELECT_DATA_MODE(dataMode) SPCR &= 0xF3; SPCR |= dataMode << 2
00306 #define    SPI_SELECT_CLOCK_PRESCALER(clockPrescaler) SPCR &= 0xFC; SPCR |= (clockPrescaler & 3); SPSR
    &= 0xFE; SPSR |= (clockPrescaler & 4) >> 2
00307 #define    SPI_SELECT_DATA_ORDER(dataOrder) SPCR &= 0xDF; SPCR |= dataOrder << 5
00308
00309 #define    SPI_WRITE_COLLISION    WCOL
00310 #define    SPI_TRANSFER_COMPLETE    SPIF
00311
00312 #define    SPI_MASTER_MODE    MSTR
00313 #define    SPI_CONTROL_REGISTER    SPCR
00314 #define    SPI_DATA_REGISTER    SPDR
00315 #define    SPI_STATUS_REGISTER    SPSR
00316
00317 #define    SPI_ENABLE_TRANSFER_COMPLETE_INTERRUPT    SPCR |= 1 << SPIE
00318 #define    SPI_DISABLE_TRANSFER_COMPLETE_INTERRUPT    SPCR &= ~(1 << SPIE)
00319
00320
00321 #define    SPI_TRANSFER_COMPLETE_INTERRUPT    SPI_STC_vect
00322
00323 // StepperMotor defines
00324
00325
00326
00327
00328
00329
00330 #endif

```

00331

## 10.29 LCD.cpp File Reference

### 10.30 LCD.cpp

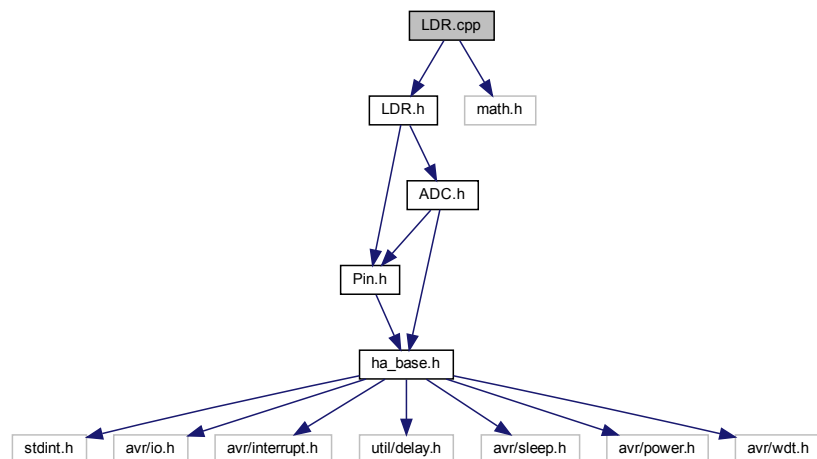
### 10.31 LCD.h File Reference

### 10.32 LCD.h

### 10.33 LDR.cpp File Reference

```
#include "LDR.h"
#include <math.h>
```

Include dependency graph for LDR.cpp:



### 10.34 LDR.cpp

```

00001 #include "LDR.h"
00002 #include <math.h>
00003
00004 component::LDR::LDR(const io::Pin &ar_pin, core::ADConverter &ar_ADC)
00005         : m_pin(ar_pin), mr_ADC(ar_ADC)
00006
00007 {
00008     m_pin.toInput(0);
00009 }
00010
00011
00012 component::LDR::~LDR()
00013 {
00014 }
00015
00016
```

```

00017 uint32_t component::LDR::getResistance(const uint8_t &ar_VDResistor)
00018 {
00019     // 10bit resolution -> 1023
00020     return
        ar_VDResistor*((1000UL*1023/mr_ADC.getConversionResult(core::resolution::res_10bit))-1000UL);
00021 }
00022
00023 uint16_t component::LDR::getLux(const uint8_t &ar_VDResistor)
00024 {
00025     // see:
00026     https://www.digikey.com/en/maker/projects/design-a-luxmeter-with-an-ldr-and-an-arduino/623aeee0f93e427bb57e02c4592567d
    return 12518931 *
        pow(ar_VDResistor*((1000UL*1023/mr_ADC.getConversionResult(core::resolution::res_10bit))-1000UL),
            -1.405); ;
00027 }

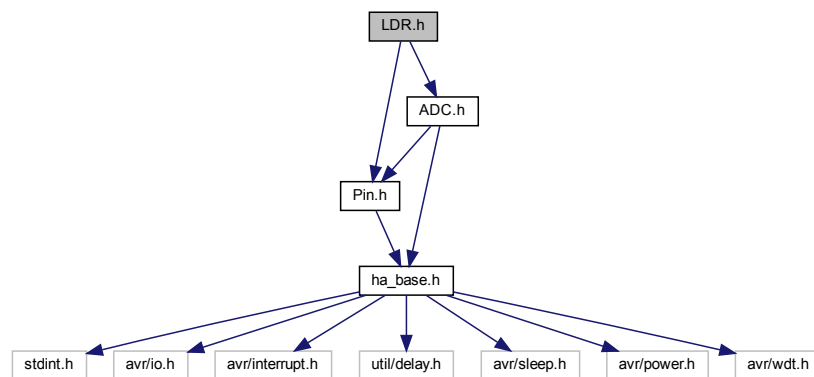
```

## 10.35 LDR.h File Reference

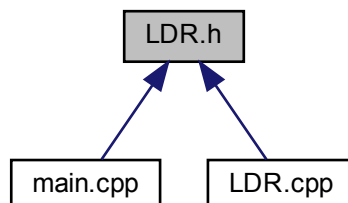
```
#include "Pin.h"
```

```
#include "ADC.h"
```

Include dependency graph for LDR.h:



This graph shows which files directly or indirectly include this file:



## Classes

- class [component::LDR](#)

*Class for handling an [LDR](#) component.*

## Namespaces

- [component](#)

*AVR chip external components.*

## 10.36 LDR.h

```

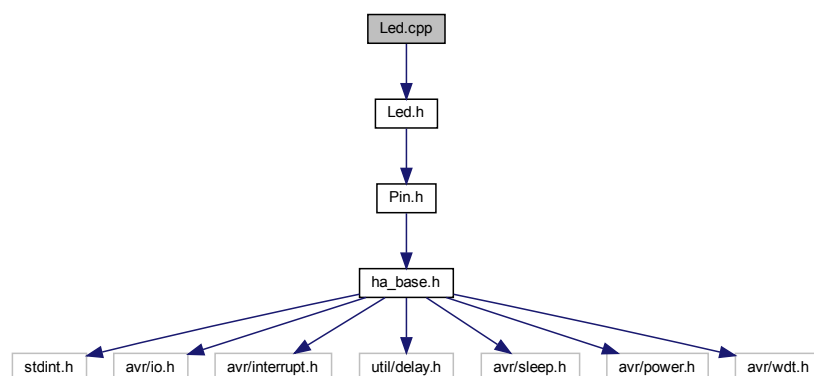
00001 #ifndef LDR_H
00002 #define LDR_H
00003 #include "Pin.h"
00004 #include "ADC.h"
00005
00015 namespace component
00016 {
00017
00027 class LDR
00028 {
00029
00030 public:
00039     LDR(const io::Pin &ar_pin, core::ADConverter &ar_ADC);
00043     ~LDR();
00049     uint32_t getResistance(const uint8_t &ar_VDResistor);
00050
00056     uint16_t getLux(const uint8_t &ar_VDResistor);
00057
00058 protected:
00059
00060 private:
00061
00062     io::Pin m_pin;
00063     core::ADConverter &mr_ADC;
00067 };
00068
00069 }
00070
00071 #endif // LDR_H

```

## 10.37 Led.cpp File Reference

```
#include "Led.h"
```

Include dependency graph for Led.cpp:



## 10.38 Led.cpp

```

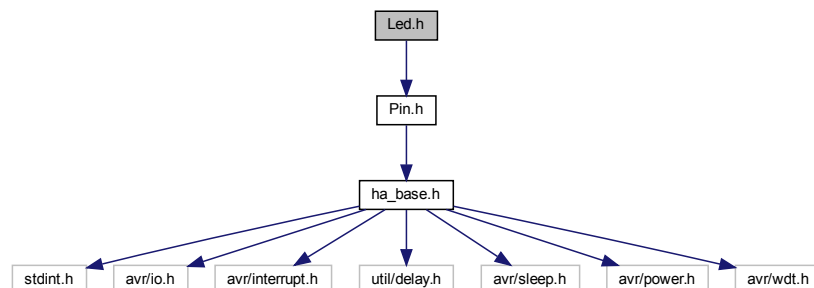
00001
00010 #include "Led.h"
00011
00012 component::Led::Led(const io::Pin &ar_pin)
00013     : m_pin(ar_pin)
00014 {
00015     m_pin.toOutput();
00016 }
00017
00018
00019 component::Led::~Led()
00020 {
00021 }
00022 }
00023
00024
00025 void component::Led::on()
00026 {
00027     m_pin.setHigh();
00028 }
00029 }
00030
00031 void component::Led::off()
00032 {
00033     m_pin.setLow();
00034 }
00035 }
00036
00037
00038 void component::Led::toggle()
00039 {
00040     m_pin.toggle();
00041 }
00042 }
00043
00044 uint8_t component::Led::isOn()
00045 {
00046     return m_pin.isHigh();
00047 }
00048 }
00049
00050
00051 uint8_t component::Led::isOff()
00052 {
00053     return m_pin.isLow();
00054 }
00055 }
00056

```

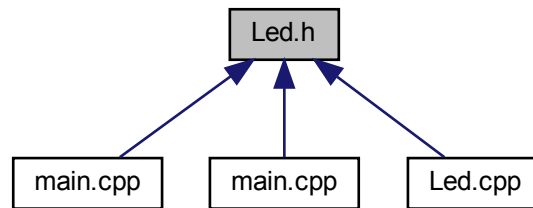
## 10.39 Led.h File Reference

```
#include "Pin.h"
```

Include dependency graph for Led.h:



This graph shows which files directly or indirectly include this file:



## Classes

- class `component::Led`

*Class for handling a `Led` component.*

## Namespaces

- `component`

*AVR chip external components.*

## 10.40 Led.h

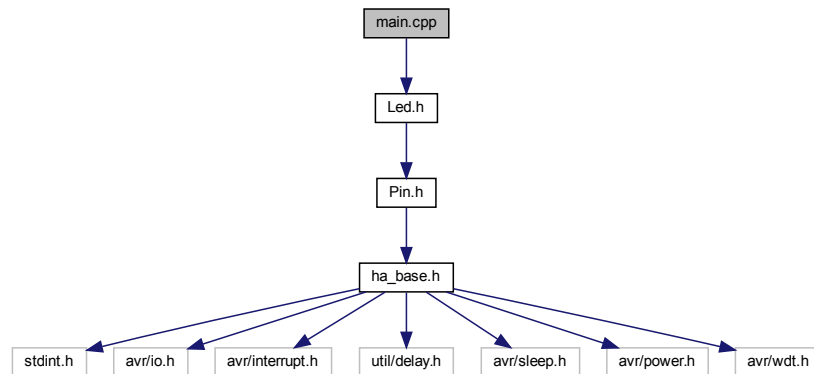
```

00001 #ifndef LED_H
00002 #define LED_H
00003 #include "Pin.h"
00004
00014 namespace component
00015 {
00016
00026 class Led
00027 {
00028
00029 public:
00037     Led(const io::Pin &ar_pin);
00041     ~Led();
00044     void on();
00047     void off();
00050     void toggle();
00056     uint8_t isOn();
00062     uint8_t isOff();
00063
00064 protected:
00065
00066 private:
00067     io::Pin m_pin;
00072 };
00073
00074 }
00075
00076
00077 #endif
  
```

## 10.41 main.cpp File Reference

```
#include "Led.h"
```

Include dependency graph for BlinkLed/main.cpp:



### Macros

- #define `LED` 0
- #define `TIMEDELAY` 500

### Functions

- int `main` (void)

#### 10.41.1 Detailed Description

Demo example: Blink a Led

Usage demonstration of the TahakomAVRLib in a simple blink a Led example. This example demonstrates the use of the `component::Led()` and `io::Pin()` abstraction objects.

- Compiler: avr-gcc (GCC) 5.4.0
- Supported devices: The example compiles on the ATmega48P/88P/168P/328P AVR family

#### Author

Farid Oubbati ( [farid.oubbati@outlook.com](mailto:farid.oubbati@outlook.com) )

#### Date

March 2018

Definition in file [BlinkLed/main.cpp](#).



## 10.41.2 Macro Definition Documentation

### 10.41.2.1 LED

```
#define LED 0
```

Led pin number

Definition at line 19 of file [BlinkLed/main.cpp](#).

### 10.41.2.2 TIMEDELAY

```
#define TIMEDELAY 500
```

Time delay

Definition at line 21 of file [BlinkLed/main.cpp](#).

## 10.41.3 Function Documentation

### 10.41.3.1 main()

```
int main (  
    void )
```

Definition at line 23 of file [BlinkLed/main.cpp](#).

```
00023     {  
00024  
00025     // Init  
00026  
00027     // Instantiate a Led object  
00028     component::Led Led(io::Pin(LED, io::PortB));  
00029  
00030     // Mainloop  
00031     while (1) {  
00032  
00033         Led.on();  
00034         _delay_ms(TIMEDELAY);  
00035  
00036         Led.off();  
00037         _delay_ms(TIMEDELAY);  
00038     }  
00039     return 0;  
00040 }
```

## 10.42 BlinkLed/main.cpp

```

00001
00017 #include "Led.h"
00018
00019 #define LED 0
00021 #define TIMEDELAY 500
00023 int main(void) {
00024
00025     // Init
00026
00027     // Instantiate a Led object
00028     component::Led Led(io::Pin(LED,io::PortB));
00029
00030     // Mainloop
00031     while (1) {
00032
00033         Led.on();
00034         _delay_ms(TIMEDELAY);
00035
00036         Led.off();
00037         _delay_ms(TIMEDELAY);
00038     }
00039     return 0;
00040 }

```

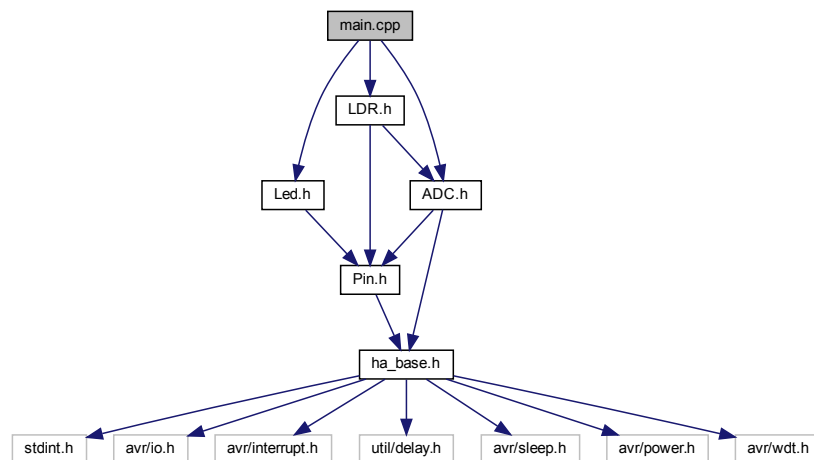
## 10.43 main.cpp File Reference

```

#include "Led.h"
#include "ADC.h"
#include "LDR.h"

```

Include dependency graph for LightMeter/main.cpp:



## Macros

- #define LED0 0
- #define LED1 1
- #define LED2 2
- #define LED3 3
- #define LED4 4
- #define LED5 5
- #define LED6 6
- #define LED7 7
- #define LDRVD 0

## Functions

- int [main](#) (void)

### 10.43.1 Detailed Description

Demo example: Light Meter

Usage demonstration of the TahakomAVRLib in an AVR based Light Meter. The example presents an AVR based Light Meter that displays its Lux measurements in a Led bank. This example demonstrates the use of the `component::LED()`, `component::LDR()` and `core::ADC()` abstraction objects.

- Compiler: avr-gcc (GCC) 5.4.0
- Supported devices: The example compiles on the ATmega48P/88P/168P/328P AVR family

#### Author

Farid Oubbati ( [farid.oubbati@outlook.com](mailto:farid.oubbati@outlook.com))

#### Date

March 2018

Definition in file [LightMeter/main.cpp](#).

### 10.43.2 Macro Definition Documentation

#### 10.43.2.1 LDRVD

```
#define LDRVD 0
```

LDR

Definition at line [31](#) of file [LightMeter/main.cpp](#).

#### 10.43.2.2 LED0

```
#define LED0 0
```

Led 0

Definition at line [22](#) of file [LightMeter/main.cpp](#).

#### 10.43.2.3 LED1

```
#define LED1 1
```

Led 1

Definition at line 23 of file [LightMeter/main.cpp](#).

#### 10.43.2.4 LED2

```
#define LED2 2
```

Led 2

Definition at line 24 of file [LightMeter/main.cpp](#).

#### 10.43.2.5 LED3

```
#define LED3 3
```

Led 3

Definition at line 25 of file [LightMeter/main.cpp](#).

#### 10.43.2.6 LED4

```
#define LED4 4
```

Led 4

Definition at line 26 of file [LightMeter/main.cpp](#).

#### 10.43.2.7 LED5

```
#define LED5 5
```

Led 5

Definition at line 27 of file [LightMeter/main.cpp](#).

### 10.43.2.8 LED6

```
#define LED6 6
```

Led 6

Definition at line 28 of file [LightMeter/main.cpp](#).

### 10.43.2.9 LED7

```
#define LED7 7
```

Led 7

Definition at line 29 of file [LightMeter/main.cpp](#).

## 10.43.3 Function Documentation

### 10.43.3.1 main()

```
int main (
    void )
```

Definition at line 34 of file [LightMeter/main.cpp](#).

```
00034     {
00035
00036     // Init
00037
00038     // Instantiate Led objects
00039     component::Led Led[8]={
00040         {io::Pin(LED0,io::PortD)},
00041         {io::Pin(LED1,io::PortD)},
00042         {io::Pin(LED2,io::PortD)},
00043         {io::Pin(LED3,io::PortD)},
00044         {io::Pin(LED4,io::PortD)},
00045         {io::Pin(LED5,io::PortD)},
00046         {io::Pin(LED6,io::PortD)},
00047         {io::Pin(LED7,io::PortD)}};
00048
00049
00050     // Instantiate ADC object
00051     core::ADConverter &myADC = core::ADConverter::getInstance();
00052
00053     // Start conversion
00054     myADC.start();
00055
00056     // Instantiate LDR object
00057     component::LDR LDR(io::Pin(LDRVD,io::PortC),myADC);
00058
00059
00060     // Mainloop
00061     while (1) {
00062
00063
00064         for (uint8_t i=0;i<=(7-(LDR.getLux(5) & 7));i++)
00065         {
00066             Led[i].on();
00067         }
00068
00069         for (uint8_t i=(7-(LDR.getLux(5) & 7))+1;i<8;i++)
00070         {
00071             Led[i].off();
00072         }
00073
00074     }
00075     return 0;
00076 }
```

References [LED0](#), [LED1](#), [LED2](#), [LED3](#), [LED4](#), [LED5](#), [LED6](#), [LED7](#), and [io::PortD](#).

## 10.44 main.cpp

```

00001
00018 #include "Led.h"
00019 #include "ADC.h"
00020 #include "LDR.h"
00021
00022 #define LED0 0
00023 #define LED1 1
00024 #define LED2 2
00025 #define LED3 3
00026 #define LED4 4
00027 #define LED5 5
00028 #define LED6 6
00029 #define LED7 7
00031 #define LDRVD 0
00034 int main(void) {
00035
00036     // Init
00037
00038     // Instantiate Led objects
00039     component::Led Led[8]={
00040         {io::Pin(LED0,io::PortD)},
00041         {io::Pin(LED1,io::PortD)},
00042         {io::Pin(LED2,io::PortD)},
00043         {io::Pin(LED3,io::PortD)},
00044         {io::Pin(LED4,io::PortD)},
00045         {io::Pin(LED5,io::PortD)},
00046         {io::Pin(LED6,io::PortD)},
00047         {io::Pin(LED7,io::PortD)}};
00048
00049
00050     // Instantiate ADC object
00051     core::ADConverter &myADC = core::ADConverter::getInstance();
00052
00053     // Start conversion
00054     myADC.start();
00055
00056     // Instantiate LDR object
00057     component::LDR LDR(io::Pin(LDRVD,io::PortC),myADC);
00058
00059
00060     // Mainloop
00061     while (1) {
00062
00063
00064         for (uint8_t i=0;i<=(7-(LDR.getLux(5) & 7));i++)
00065         {
00066             Led[i].on();
00067         }
00068
00069         for (uint8_t i=(7-(LDR.getLux(5) & 7))+1;i<8;i++)
00070         {
00071             Led[i].off();
00072         }
00073
00074     }
00075     return 0;
00076 }

```

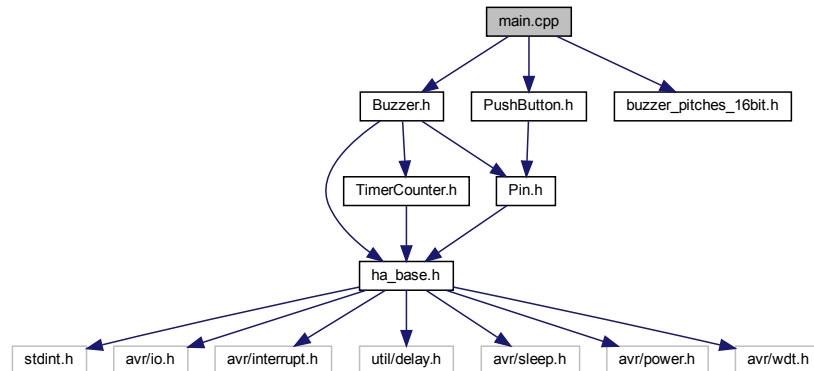
## 10.45 main.cpp File Reference

```

#include "Buzzer.h"
#include "PushButton.h"
#include "buzzer_pitches_16bit.h"

```

Include dependency graph for MusicBox/main.cpp:



## Macros

- `#define BUZZER 0`
- `#define PUSHBUTTON 1`
- `#define SONG_LENGTH (sizeof(l_song) / sizeof(uint16_t))`

## Functions

- `int main (void)`

### 10.45.1 Detailed Description

Demo example: AVR Music Box

Usage demonstration of the TahakomAVRLib in an AVR based Music Box. The example allows the user to play a predefined song in a Buzzer and control its tempo via a PushButton. This example demonstrates the use of the `component::Buzzer()` and `component::PushButton()` abstraction objects.

- Compiler: avr-gcc (GCC) 5.4.0
- Supported devices: The example compiles on the ATmega48P/88P/168P/328P AVR family

#### Author

Farid Oubbati ( [farid.oubbati@outlook.com](mailto:farid.oubbati@outlook.com) )

#### Date

March 2018

Definition in file [MusicBox/main.cpp](#).

## 10.45.2 Macro Definition Documentation

### 10.45.2.1 BUZZER

```
#define BUZZER 0
```

Buzzer pin number

Definition at line 24 of file [MusicBox/main.cpp](#).

### 10.45.2.2 PUSHBUTTON

```
#define PUSHBUTTON 1
```

PushButton pin number

Definition at line 26 of file [MusicBox/main.cpp](#).

### 10.45.2.3 SONG\_LENGTH

```
#define SONG_LENGTH (sizeof(l_song) / sizeof(uint16_t))
```

Song length

Definition at line 28 of file [MusicBox/main.cpp](#).

## 10.45.3 Function Documentation



## 10.45.3.1 main()

```
int main (
    void )
```

Definition at line 30 of file [MusicBox/main.cpp](#).

```
00030     {
00031
00032     // Init
00033
00034     // Instantiate a Buzzer object
00035     component::Buzzer Buzzer(io::Pin(BUZZER,io::PortB));
00036
00037     // Instantiate a PushButton object
00038     component::PushButton PushButton(io::Pin(PUSHBUTTON,io::PortB));
00039
00040     // Song notes
00041     const uint16_t l_song[] = {
00042     E6, E6, E6, C6, E6, G6, G5,
00043     C6, G5, E5, A5, B5, Ax5, A5,
00044     G5, E6, G6, A6, F6, G6, E6, C6, D6, B5,
00045     C6, G5, E5, A5, B5, Ax5, A5,
00046     G5, E6, G6, A6, F6, G6, E6, C6, D6, B5,
00047     };
00048
00049     // Start song at start
00050     uint8_t l_note = SONG_LENGTH - 1;
00051
00052     // Initialize PushButton state
00053     uint8_t l_statePushButton = 0;
00054
00055
00056     // Mainloop
00057     while (1) {
00058         // Check if PusgButton is pressed
00059         if (PushButton.isPressed())
00060         {
00061             // Check if a new button press
00062             if (!l_statePushButton) {
00063                 // play next note
00064                 l_note++;
00065                 // start from start
00066                 if (l_note == SONG_LENGTH) {
00067
00068                     l_note = 0;
00069
00070                 }
00071
00072                 l_statePushButton = 1;
00073
00074             }
00075             // Play note
00076             Buzzer.buzz(l_song[l_note], 1600);
00077
00078         } else {
00079
00080             l_statePushButton=0;
00081
00082         }
00083
00084     }
00085     return 0;
00086 }
```

## 10.46 MusicBox/main.cpp

```
00001
00018 #include "Buzzer.h"
00019 #include "PushButton.h"
00020 // Note pitch data
00021 #include "buzzer_pitches_16bit.h"
00022
00023
00024 #define BUZZER 0
00026 #define PUSHBUTTON 1
00028 #define SONG_LENGTH (sizeof(l_song) / sizeof(uint16_t))
00030 int main(void) {
00031
00032     // Init
00033
```

```

00034 // Instantiate a Buzzer object
00035 component::Buzzer Buzzer(io::Pin(BUZZER,io::PortB));
00036
00037 // Instantiate a PushButton object
00038 component::PushButton PushButton(io::Pin(PUSHBUTTON,io::PortB));
00039
00040 // Song notes
00041 const uint16_t l_song[] = {
00042     E6, E6, E6, C6, E6, G6, G5,
00043     C6, G5, E5, A5, B5, Ax5, A5,
00044     G5, E6, G6, A6, F6, G6, E6, C6, D6, B5,
00045     C6, G5, E5, A5, B5, Ax5, A5,
00046     G5, E6, G6, A6, F6, G6, E6, C6, D6, B5,
00047 };
00048
00049 // Start song at start
00050 uint8_t l_note = SONG_LENGTH - 1;
00051
00052 // Initialize PushButton state
00053 uint8_t l_statePushButton = 0;
00054
00055
00056 // Mainloop
00057 while (1) {
00058     // Check if PusgButton is pressed
00059     if (PushButton.isPressed())
00060     {
00061         // Check if a new button press
00062         if (!l_statePushButton) {
00063             // play next note
00064             l_note++;
00065             // start from start
00066             if (l_note == SONG_LENGTH) {
00067
00068                 l_note = 0;
00069
00070             }
00071
00072             l_statePushButton = 1;
00073
00074         }
00075         // Play note
00076         Buzzer.buzz(l_song[l_note], 1600);
00077     } else {
00078
00079         l_statePushButton=0;
00080
00081     }
00082
00083 }
00084 }
00085 return 0;
00086 }

```

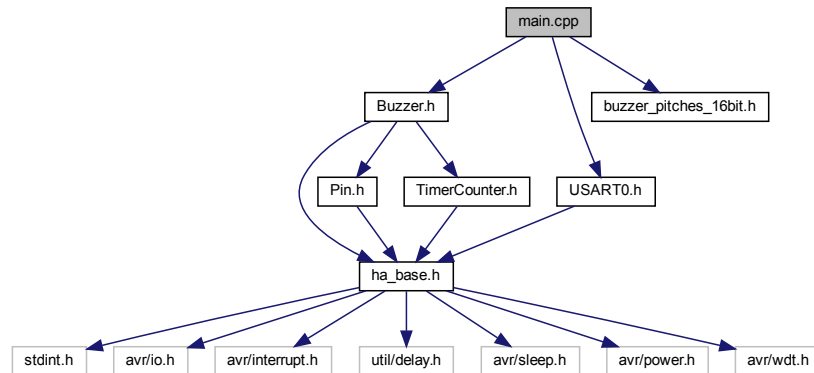
## 10.47 main.cpp File Reference

```

#include "Buzzer.h"
#include "USART0.h"
#include "buzzer_pitches_16bit.h"

```

Include dependency graph for Organ/main.cpp:



## Macros

- `#define BUZZER 0`
- `#define NOTE_DURATION 0xF000`

## Functions

- `int main (void)`

### 10.47.1 Detailed Description

Demo example: AVR Square-Wave Organ

Usage demonstration of the TahakomAVRLib in a Buzzer based and serial driven Square-Wave Organ. The example enables the user to enter a key note via serial terminal from the keyboard and play the corresponding note in a Buzzer. This example demonstrates the use of the `component::Buzzer()` and `io::USART0()` abstraction objects.

- Compiler: avr-gcc (GCC) 5.4.0
- Supported devices: The example compiles on the ATmega48P/88P/168P/328P AVR family

#### Author

Farid Oubbati ( [farid.oubbati@outlook.com](mailto:farid.oubbati@outlook.com) )

#### Date

March 2018

Definition in file [Organ/main.cpp](#).

## 10.47.2 Macro Definition Documentation

### 10.47.2.1 BUZZER

```
#define BUZZER 0
```

Buzzer pin number

Definition at line 24 of file [Organ/main.cpp](#).

### 10.47.2.2 NOTE\_DURATION

```
#define NOTE_DURATION 0xF000
```

Note duration

Definition at line 26 of file [Organ/main.cpp](#).

## 10.47.3 Function Documentation

### 10.47.3.1 main()

```
int main (
    void )
```

Definition at line 28 of file [Organ/main.cpp](#).

```
00028         {
00029
00030     // Init
00031
00032     // Instantiate a Buzzer object
00033     component::Buzzer Buzzer(io::Pin(BUZZER,io::PortB));
00034
00035     // Instantiate a USART0 object
00036     io::USART0 &myUSART0 = io::USART0::getInstance();
00037
00038     // List of possible keypresses
00039     const uint8_t l_keys[] = { 'a', 'w', 's', 'e', 'd', 'f', 't',
00040                               'g', 'y', 'h', 'j', 'i', 'k', 'o',
00041                               'l', 'p', ';', '\n'
00042     };
00043
00044     // List of the keypresses corresponding pitches
00045     const uint16_t l_notes[] = { G4, Gx4, A4, Ax4, B4, C5, Cx5,
00046                                 D5, Dx5, E5, Fx5, G5, Gx5,
00047                                 A5, Ax5, B5, C6
00048     };
00049
00050     // Character key from computer serial terminal
00051     unsigned char l_key;
00052
00053     // Current note length
00054     uint16_t l_currentNoteLength = NOTE_DURATION;
00055 }
```

```

00056 // Check if note key
00057 uint8_t l_isNote;
00058
00059 myUSART0.sendString("----- Serial Organ -----\r\n");
00060
00061 // Mainloop
00062 while (1) {
00063
00064     // Send N to signal that AVR chip is ready for next key note input
00065     myUSART0.sendString("Enter key note\r\n");
00066
00067     // Wait for key note input
00068     while (myUSART0.getNumberBytesReceived()==0)
00069     {
00070         myUSART0.receiveChar(l_key);
00071     }
00072     myUSART0.resetNumberBytesReceived();
00073
00074     /***** Play key notes *****/
00075
00076     l_isNote = 0;
00077     // Loop through keys table
00078     for (uint8_t i = 0; i < sizeof(l_keys); i++) {
00079         // Found match in lookup table
00080         if (l_key == l_keys[i]) {
00081             Buzzer.buzz(l_notes[i], l_currentNoteLength);
00082             // Record that we've found a note
00083             l_isNote = 1;
00084             break;
00085         }
00086     }
00087
00088     // Handle non-note keys: tempo changes and rests
00089     if (!l_isNote) {
00090         if (l_key == '-') {
00091             // Code for short note
00092             l_currentNoteLength = l_currentNoteLength / 2;
00093         }
00094         else if (l_key == '+') {
00095             // Code for long note
00096             l_currentNoteLength = l_currentNoteLength * 2;
00097         }
00098         else {
00099             // Unrecognized, just rest
00100             Buzzer.pause(l_currentNoteLength);
00101         }
00102     }
00103 }
00104
00105 }
00106 return 0;
00107 }

```

## 10.48 Organ/main.cpp

```

00001
00018 #include "Buzzer.h"
00019 #include "USART0.h"
00020 // Note pitch data
00021 #include "buzzer_pitches_16bit.h"
00022
00023
00024 #define BUZZER 0
00026 #define NOTE_DURATION 0xF000
00028 int main(void) {
00029
00030     // Init
00031
00032     // Instantiate a Buzzer object
00033     component::Buzzer Buzzer(io::Pin(BUZZER, io::PortB));
00034
00035     // Instantiate a USART0 object
00036     io::USART0 &myUSART0 = io::USART0::getInstance();
00037
00038     // List of possible keypresses
00039     const uint8_t l_keys[] = { 'a', 'w', 's', 'e', 'd', 'f', 't',
00040                               'g', 'y', 'h', 'j', 'i', 'k', 'o',
00041                               'l', 'p', ';', '\n'
00042     };
00043
00044     // List of the keypresses corresponding pitches
00045     const uint16_t l_notes[] = { G4, Gx4, A4, Ax4, B4, C5, Cx5,
00046                                 D5, Dx5, E5, F5, Fx5, G5, Gx5,

```

```

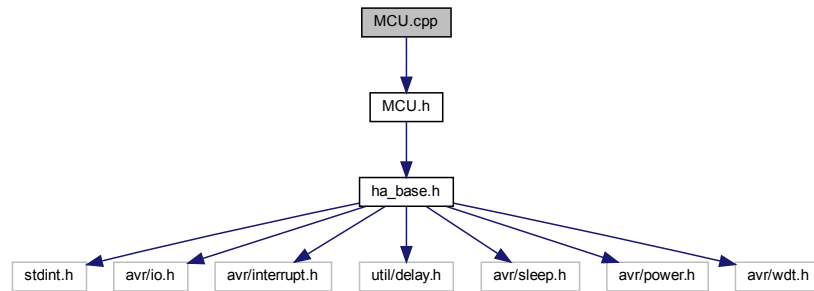
00047     A5, Ax5, B5, C6
00048 };
00049
00050 // Character key from computer serial terminal
00051 unsigned char l_key;
00052
00053 // Current note length
00054 uint16_t l_currentNoteLength = NOTE_DURATION;
00055
00056 // Check if note key
00057 uint8_t l_isNote;
00058
00059 myUSART0.sendString("----- Serial Organ -----\r\n");
00060
00061 // Mainloop
00062 while (1) {
00063
00064     // Send N to signal that AVR chip is ready for next key note input
00065     myUSART0.sendString("Enter key note\r\n");
00066
00067     // Wait for key note input
00068     while (myUSART0.getNumberBytesReceived()==0)
00069     {
00070         myUSART0.receiveChar(l_key);
00071     }
00072     myUSART0.resetNumberBytesReceived();
00073
00074     /***** Play key notes *****/
00075
00076     l_isNote = 0;
00077     // Loop through keys table
00078     for (uint8_t i = 0; i < sizeof(l_keys); i++) {
00079         // Found match in lookup table
00080         if (l_key == l_keys[i]) {
00081             Buzzer.buzz(l_notes[i], l_currentNoteLength);
00082             // Record that we've found a note
00083             l_isNote = 1;
00084             break;
00085         }
00086     }
00087
00088     // Handle non-note keys: tempo changes and rests
00089     if (!l_isNote) {
00090         if (l_key == '-' ) {
00091             // Code for short note
00092             l_currentNoteLength = l_currentNoteLength / 2;
00093         }
00094         else if (l_key == '+' ) {
00095             // Code for long note
00096             l_currentNoteLength = l_currentNoteLength * 2;
00097         }
00098         else {
00099             // Unrecognized, just rest
00100             Buzzer.pause(l_currentNoteLength);
00101         }
00102     }
00103 }
00104
00105 }
00106 return 0;
00107 }

```

## 10.49 MCU.cpp File Reference

```
#include "MCU.h"
```

Include dependency graph for MCU.cpp:



## 10.50 MCU.cpp

```

00001 #include "MCU.h"
00002
00003 void core::MCU::init()
00004 {
00005     enableUSART0(0);
00006     enableTimerCounter0(0);
00007     enableTimerCounter1(0);
00008     enableTimerCounter2(0);
00009     enableTWI(0);
00010     enableSPI(0);
00011     enableADC(0);
00012 }
00013
00014
00015 void core::MCU::selectSleepMode(const sleepMode &a_sleepMode)
00016 {
00017     MCU_SELECT_SLEEP_MODE(static_cast<uint8_t>(a_sleepMode));
00018 }
00019
00020
00021 void core::MCU::goToSleep(const BODMode &a_BODMode)
00022 {
00023     cli();
00024     switch (a_BODMode)
00025     {
00026     case core::BODMode::enabled:
00027     {
00028         sleepEnable(1);
00029         sei();
00030         sleep_cpu();
00031         sleepEnable(0);
00032         break;
00033     }
00034     case core::BODMode::disabled:
00035     {
00036         sleepEnable(1);
00037         disableBOD();
00038         sei();
00039         sleep_cpu();
00040         sleepEnable(0);
00041         break;
00042     }
00043     }
00044 }
00045
00046
00047 void core::MCU::sleepEnable(const uint8_t a_enable)
00048 {
00049     if (a_enable) {
00050         MCU_SLEEP_ENABLE;
00051     } else {
00052         MCU_SLEEP_DISABLE;
00053     }
00054 }
00055
00056
00057 void core::MCU::enableUSART0(const uint8_t a_enable)

```

```

00058 {
00059     if (a_enable) {
00060         MCU_USART0_ENABLE;
00061     } else {
00062         MCU_USART0_DISABLE;
00063     }
00064 }
00065 }
00066
00067 void core::MCU::enableTimerCounter0(const uint8_t a_enable)
00068 {
00069     if (a_enable) {
00070         MCU_TIMER0_ENABLE;
00071     } else {
00072         MCU_TIMER0_DISABLE;
00073     }
00074 }
00075 }
00076
00077 void core::MCU::enableTimerCounter1(const uint8_t a_enable)
00078 {
00079     if (a_enable) {
00080         MCU_TIMER1_ENABLE;
00081     } else {
00082         MCU_TIMER1_DISABLE;
00083     }
00084 }
00085 }
00086
00087 void core::MCU::enableTimerCounter2(const uint8_t a_enable)
00088 {
00089     if (a_enable) {
00090         MCU_TIMER2_ENABLE;
00091     } else {
00092         MCU_TIMER2_DISABLE;
00093     }
00094 }
00095 }
00096
00097 void core::MCU::enableTWI(const uint8_t a_enable)
00098 {
00099     if (a_enable) {
00100         MCU_TWI_ENABLE;
00101     } else {
00102         MCU_TWI_DISABLE;
00103     }
00104 }
00105 }
00106
00107 void core::MCU::enableSPI(const uint8_t a_enable)
00108 {
00109     if (a_enable) {
00110         MCU_SPI_ENABLE;
00111     } else {
00112         MCU_SPI_DISABLE;
00113     }
00114 }
00115 }
00116
00117 void core::MCU::enableADC(const uint8_t a_enable)
00118 {
00119     if (a_enable) {
00120         MCU_ADC_ENABLE;
00121     } else {
00122         MCU_ADC_DISABLE;
00123     }
00124 }
00125 }
00126 void core::MCU::disableBOD()
00127 {
00128     MCU_BOD_DISABLE;
00129 }

```

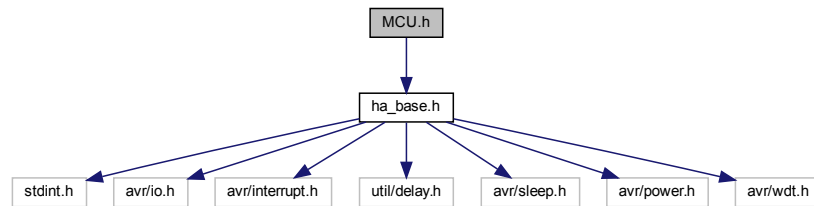
## 10.51 MCU.h File Reference

Header file of the MCU class.

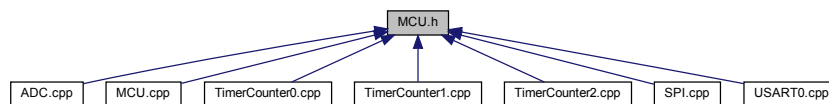


```
#include "ha_base.h"
```

Include dependency graph for MCU.h:



This graph shows which files directly or indirectly include this file:



## Classes

- class [core::MCU](#)

## Namespaces

- [core](#)

*AVR chip internal core components.*

## Enumerations

- enum [core::BODMode](#) : uint8\_t { [core::BODMode::enabled](#) =0, [core::BODMode::disabled](#) }
- enum [core::sleepMode](#) : uint8\_t { [core::sleepMode::Idle](#) =0, [core::sleepMode::ADC\\_NoiseReduction](#), [core::sleepMode::powerDown](#), [core::sleepMode::powerSave](#), [core::sleepMode::standby](#) =6, [core::sleepMode::extendedStandby](#) }

### 10.51.1 Detailed Description

Header file of the MCU class.

Basic class abstraction of the MCU.

Usage example (test):

```
#include "PushButton.h" #include "Led.h" #include "ExternInterrupt.h" #include "MCU.h"
```

```

#define PUSHBUTTON_NUMBER 2 #define LED_NUMBER 0

int main(void) {

    Init initialize MCU core::MCU::init();

    instantiate a Led object component::Led Led(io::Pin(LED_NUMBER,io::PortB));

    instantiate a Led object component::PushButton PushButton(io::Pin(PUSHBUTTON_NUMBER,io::PortD));

    instantiate the external interrupt manager core::ExternInterrupt &myExternInterrupt = core::ExternInterrupt::getInstance();
    myExternInterrupt.enableInt0(1); myExternInterrupt.setInt0SenseControl(core::senseControl::logicalChange);

    set sleep mode core::MCU::selectSleepMode(core::sleepMode::powerDown);

    Mainloop while (1) {

        flash the LED for (uint8_t i=0;i<10;i++) { Led.on(); _delay_ms(100); Led.off(); _delay_ms(100); } _delay_ms(5000);
        Led.on(); _delay_ms(100); Led.off(); core::MCU::goToSleep(core::BODMode::enabled);

    } return 0; }

    void core::ExternInterrupt::Int0ServiceRoutine() {

    }

```

#### Author

Farid Oubbati ( <https://github.com/faroub>)

#### Date

March 2020

Definition in file [MCU.h](#).

## 10.52 MCU.h

```

00001
00069 #ifndef MCU_H
00070 #define MCU_H
00071 #include "ha_base.h"
00072
00073
00074 namespace core
00075 {
00076
00077 enum class BODMode : uint8_t {
00078     enabled=0,
00079     disabled,
00080 };
00081
00082 enum class sleepMode : uint8_t {
00083     Idle=0,
00084     ADC_NoiseReduction,
00085     powerDown,
00086     powerSave,
00087     standby=6,
00088     extendedStandby,
00089 };
00090
00091 class MCU
00092 {
00093 public:
00094

```

```

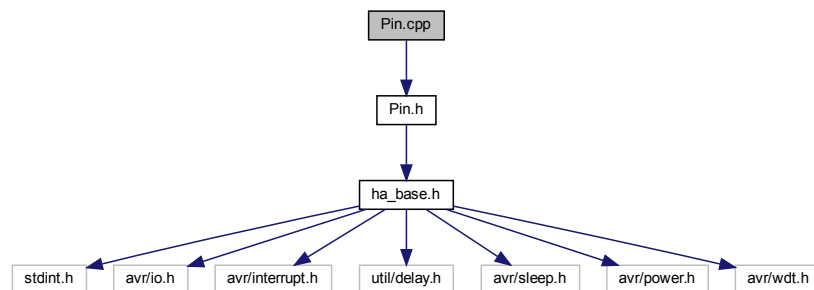
00095     static void init();
00096
00097     static void selectSleepMode(const sleepMode &a_sleepMode);
00098
00099     static void goToSleep(const BODMode &a_BODMode);
00100
00101     static void sleepEnable(const uint8_t ar_enable);
00102
00103     static void enableUSART0(const uint8_t a_enable);
00104
00105     static void enableTimerCounter0(const uint8_t a_enable);
00106
00107     static void enableTimerCounter1(const uint8_t a_enable);
00108
00109     static void enableTimerCounter2(const uint8_t a_enable);
00110
00111     static void enableTWI(const uint8_t a_enable);
00112
00113     static void enableSPI(const uint8_t a_enable);
00114
00115     static void enableADC(const uint8_t a_enable);
00116
00117     static void disableBOD();
00118
00119
00120
00121
00122
00123 protected:
00124
00125
00126
00127 private:
00128
00129
00130
00131
00132
00133
00134
00135 };
00136
00137
00138
00139
00140
00141 }
00142 #endif

```

## 10.53 Pin.cpp File Reference

```
#include "Pin.h"
```

Include dependency graph for Pin.cpp:



## 10.54 Pin.cpp

```
00001
```

```

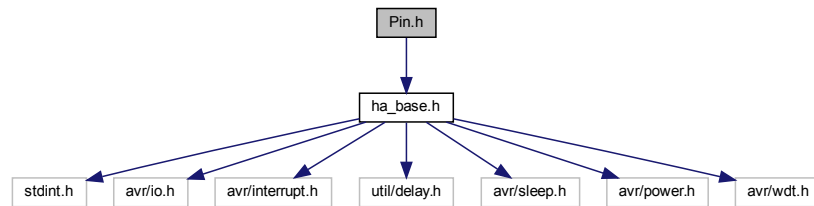
00009 #include "Pin.h"
00010
00011
00012 io::Pin::Pin(const uint8_t a_pinNumber, const Port &ar_portName)
00013     : mr_portName(ar_portName), m_pinNumber(a_pinNumber)
00014 {
00015 }
00016 }
00017
00018 io::Pin::~~Pin()
00019 {
00020 }
00021 }
00022 }
00023
00024 void io::Pin::toOutput()
00025 {
00026     *mr_portName.mp_ddrReg |= (1 « m_pinNumber);
00027 }
00028
00029 void io::Pin::toInput(const uint8_t &ar_useInternalPullUp)
00030 {
00031     if (ar_useInternalPullUp)
00032     {
00033         *mr_portName.mp_portReg |= (1 « m_pinNumber);
00034         *mr_portName.mp_ddrReg &= ~(1 « m_pinNumber);
00035     }
00036 }
00037 }
00038 else
00039 {
00040     *mr_portName.mp_portReg &= ~(1 « m_pinNumber);
00041     *mr_portName.mp_ddrReg &= ~(1 « m_pinNumber);
00042 }
00043 }
00044 }
00045
00046 void io::Pin::setLow()
00047 {
00048     if (*mr_portName.mp_ddrReg & (1 « m_pinNumber))
00049     {
00050         *mr_portName.mp_portReg &= ~(1 « m_pinNumber);
00051     }
00052 }
00053
00054 void io::Pin::setHigh()
00055 {
00056     if (*mr_portName.mp_ddrReg & (1 « m_pinNumber))
00057     {
00058         *mr_portName.mp_portReg |= (1 « m_pinNumber);
00059     }
00060 }
00061
00062 void io::Pin::toggle()
00063 {
00064     if (*mr_portName.mp_ddrReg & (1 « m_pinNumber))
00065     {
00066         *mr_portName.mp_portReg ^= 1 « m_pinNumber;
00067     }
00068 }
00069
00070 uint8_t io::Pin::isHigh()
00071 {
00072     return *mr_portName.mp_pinReg & (1 « m_pinNumber);
00073 }
00074
00075 uint8_t io::Pin::isLow()
00076 {
00077     return !(*mr_portName.mp_pinReg & (1 « m_pinNumber));
00078 }
00079
00080 uint8_t io::Pin::getPinNumber()
00081 {
00082     return m_pinNumber;
00083 }

```

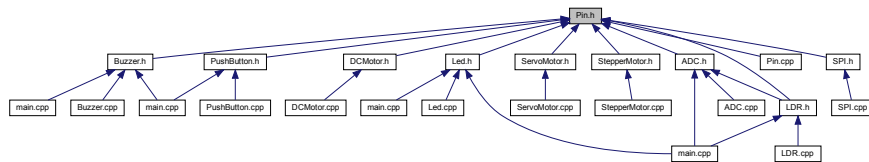
## 10.55 Pin.h File Reference

```
#include "ha_base.h"
```

Include dependency graph for Pin.h:



This graph shows which files directly or indirectly include this file:



## Classes

- struct [io::Port](#)  
*Contains definitions of pointers to the AVR port's registers.*
- class [io::Pin](#)  
*Class for handling a [Pin](#).*

## Namespaces

- [io](#)  
*AVR chip internal i/o components.*

## Variables

- static [io::Port](#) [io::PortB](#) = { &DDRB, &PORTB, &PINB }
- static [io::Port](#) [io::PortC](#) = { &DDRC, &PORTC, &PINC }
- static [io::Port](#) [io::PortD](#) = { &DDRD, &PORTD, &PIND }

## 10.56 Pin.h

```

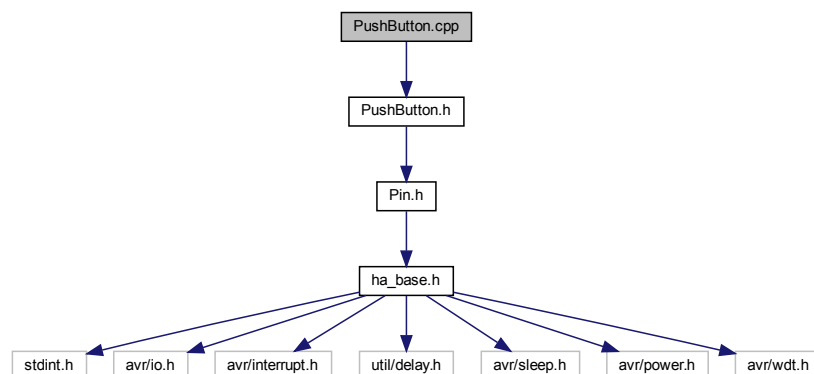
00001 #ifndef PIN_H
00002 #define PIN_H
00003 #include "ha_base.h"
00012 namespace io
00013 {
00014
00019 struct Port
00020 {
00021     volatile uint8_t* mp_ddrReg;
00023     volatile uint8_t* mp_portReg;
00025     volatile uint8_t* mp_pinReg;
00026 };
00035 class Pin
00036 {
00037 public:
00046     Pin(const uint8_t a_pinNumber, const Port &ar_portName);
00049     ~Pin();
00052     void toOutput();
00057     void toInput(const uint8_t &ar_useInternalPullUp);
00060     void setLow();
00063     void setHigh();
00064
00067     void toggle();
00073     uint8_t isHigh();
00079     uint8_t isLow();
00085     uint8_t getPinNumber();
00086
00087 protected:
00088
00089 private:
00090     const Port &mr_portName;
00093     const uint8_t m_pinNumber;
00098 };
00099
00100 static io::Port PortB = { &DDRB, &PORTB, &PINB };
00101 static io::Port PortC = { &DDRC, &PORTC, &PINC };
00102 static io::Port PortD = { &DDRD, &PORTD, &PIND };
00104 }
00105
00106
00107 #endif

```

## 10.57 PushButton.cpp File Reference

```
#include "PushButton.h"
```

Include dependency graph for PushButton.cpp:



## 10.58 PushButton.cpp

```

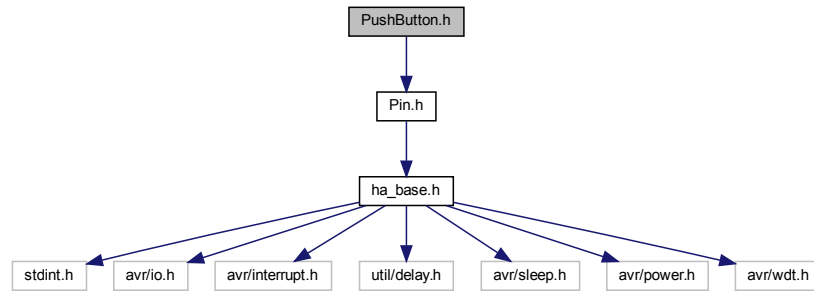
00001
00010 #include "PushButton.h"
00011
00012 component::PushButton::PushButton(const io::Pin& ar_pin, const uint8_t &ar_useInternalPullUp, const
    uint8_t &ar_isActiveLow)
00013                                     : m_pin(ar_pin),
00014                                     mr_isActiveLow(ar_isActiveLow),
00015                                     mr_useInternalPullUp(ar_useInternalPullUp)
00016
00017
00018 {
00019     m_pin.toInput(mr_useInternalPullUp);
00020     m_buttonPressed = 0;
00021 }
00022
00023 component::PushButton::~PushButton()
00024 {
00025
00026 }
00027
00028
00029 uint8_t component::PushButton::isPressed()
00030 {
00031     if (mr_isActiveLow || mr_useInternalPullUp) {
00032         if (m_pin.isLow()) {
00033             _delay_us(PUSHBUTTON_DEBOUNCE_TIME_US);
00034             if (m_pin.isLow()) {
00035                 ++m_buttonPressed;
00036                 return 1;
00037             }
00038         }
00039     } else {
00040
00041         if (m_pin.isHigh()) {
00042             _delay_us(PUSHBUTTON_DEBOUNCE_TIME_US);
00043             if (m_pin.isHigh()) {
00044                 ++m_buttonPressed;
00045                 return 1;
00046             }
00047         }
00048     }
00049 }
00050
00051 return 0;
00052
00053 }
00054
00055 uint8_t component::PushButton::getPressedCount() const
00056 {
00057     return m_buttonPressed;
00058
00059 }
00060
00061
00062 void component::PushButton::resetPressedCount()
00063 {
00064     m_buttonPressed = 0;
00065
00066 }

```

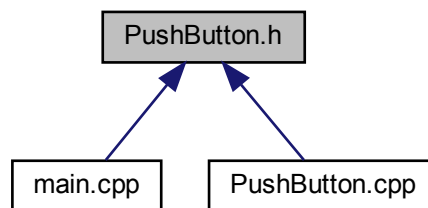
## 10.59 PushButton.h File Reference

```
#include "Pin.h"
```

Include dependency graph for PushButton.h:



This graph shows which files directly or indirectly include this file:



## Classes

- class `component::PushButton`  
Class for handling a *PushButton* component.

## Namespaces

- `component`  
AVR chip external components.

## 10.60 PushButton.h

```

00001 #ifndef PUSHBUTTON_H
00002 #define PUSHBUTTON_H
00003 #include "Pin.h"
00004
00014 namespace component
00015 {
00016
00026 class PushButton
00027 {
00028 public:
  
```



```
00037     PushButton(const io::Pin &ar_pin, const uint8_t &ar_useInternalPullUp=1, const uint8_t
        &ar_isActiveLow=1);
00041     ~PushButton();
00047     uint8_t isPressed();
00053     uint8_t getPressedCount() const;
00058     void resetPressedCount();
00059
00060
00061 protected:
00062
00063 private:
00064     io::Pin m_pin;
00065     const uint8_t &mr_isActiveLow;
00066     const uint8_t &mr_useInternalPullUp;
00067     uint8_t m_buttonPressed;
00071 };
00072 }
00073
00074
00075
00076
00077 #endif
```

## 10.61 README.md File Reference

## 10.62 README.md File Reference

## 10.63 README.md File Reference

## 10.64 README.md File Reference

## 10.65 README.md File Reference

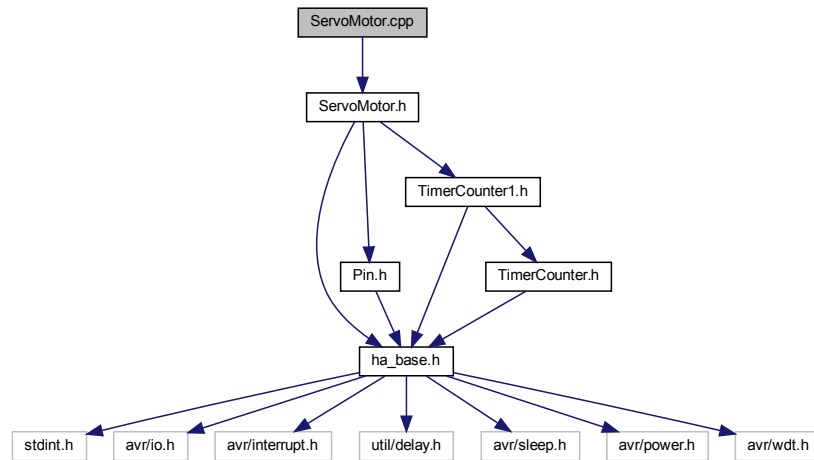
## 10.66 README.md File Reference

## 10.67 README.md File Reference

## 10.68 ServoMotor.cpp File Reference

```
#include "ServoMotor.h"
```

Include dependency graph for ServoMotor.cpp:



## 10.69 ServoMotor.cpp

```

00001 #include "ServoMotor.h"
00002
00003
00004
00005 component::ServoMotor::ServoMotor(const io::Pin &ar_pin,
00006                                   const uint16_t &ar_pulseCycle,
00007                                   const uint16_t &ar_pulseWidthMin,
00008                                   const uint16_t &ar_pulseWidthMid,
00009                                   const uint16_t &ar_pulseWidthMax)
00010     : m_pin(ar_pin),
00011       m_pulseCycle(ar_pulseCycle),
00012       m_pulseWidthMin(ar_pulseWidthMin),
00013       m_pulseWidthMid(ar_pulseWidthMid),
00014       m_pulseWidthMax(ar_pulseWidthMax)
00015 {
00016     m_pin.toOutput();
00017 }
00018 }
00019
00020 component::ServoMotor::~ServoMotor()
00021 {
00022 }
00023 }
00024
00025
00026
00027
00028 void component::ServoMotor::on()
00029 {
00030     m_pin.setHigh();
00031 }
00032
00033 void component::ServoMotor::off()
00034 {
00035     m_pin.setLow();
00036 }
00037
00038 void component::ServoMotor::toggle()
00039 {
00040     m_pin.toggle();
00041 }
00042 }
00043
00044 uint16_t component::ServoMotor::computePulseCycleCount(const uint16_t &ar_clockPrescaler)
00045 {
00046     return SERVOMOTOR_TIMER_PULSE_WIDTH_COUNT(m_pulseCycle, ar_clockPrescaler);
00047 }
00048
00049 uint16_t component::ServoMotor::computePulseWidthMinCount(const uint16_t &ar_clockPrescaler)

```

```

00050 {
00051     return SERVOMOTOR_TIMER_PULSE_WIDTH_COUNT(m_pulseWidthMin, ar_clockPrescaler);
00052 }
00053
00054 uint16_t component::ServoMotor::computePulseWidthMaxCount(const uint16_t &ar_clockPrescaler)
00055 {
00056     return SERVOMOTOR_TIMER_PULSE_WIDTH_COUNT(m_pulseWidthMax, ar_clockPrescaler);
00057 }
00058
00059 uint16_t component::ServoMotor::computePulseWidthMidCount(const uint16_t &ar_clockPrescaler)
00060 {
00061     return SERVOMOTOR_TIMER_PULSE_WIDTH_COUNT(m_pulseWidthMid, ar_clockPrescaler);
00062 }
00063
00064 uint16_t component::ServoMotor::computeRotationAngleCount(const uint8_t &ar_angle_deg, const uint16_t
&ar_clockPrescaler)
00065 {
00066     return
static_cast<uint16_t>(SERVOMOTOR_TIMER_ANGLE_COUNT(ar_angle_deg, static_cast<long>(computePulseWidthMinCount(ar_clockPr
00067 }
00068
00069
00070
00071 void component::ServoMotor::rotate(core::TimerCounter1 &ar_timerCounter1,
00072     const uint8_t &ar_angle_deg,
00073     const core::channel &ar_channel)
00074 {
00075
00076     ar_timerCounter1.setOutputCompareRegister(ar_channel,
computeRotationAngleCount(ar_angle_deg, ar_timerCounter1.getClockPrescaler()));
00077
00078     // start timer
00079     ar_timerCounter1.start();
00080
00081
00082 }
00083
00084
00085 void component::ServoMotor::connect(core::TimerCounter1 &ar_timerCounter1,
00086     const core::channel &ar_channel)
00087 {
00088
00089
00090     ar_timerCounter1.setInputCaptureRegister(computePulseCycleCount(ar_timerCounter1.getClockPrescaler()));
00091     ar_timerCounter1.selectOperationMode(core::operationMode::fast_PWM_ICR);
00092     ar_timerCounter1.selectCompareOutputMode(ar_channel, core::compareOutputMode::clear);
00093     ar_timerCounter1.setCounter(0);
00094
00095 }
00096
00097 void component::ServoMotor::disconnect(core::TimerCounter1 &ar_timerCounter1,
00098     const core::channel &ar_channel)
00099 {
00100     ar_timerCounter1.selectCompareOutputMode(ar_channel, core::compareOutputMode::normal);
00101     // stop timer
00102     ar_timerCounter1.stop();
00103 }

```

## 10.70 ServoMotor.h File Reference

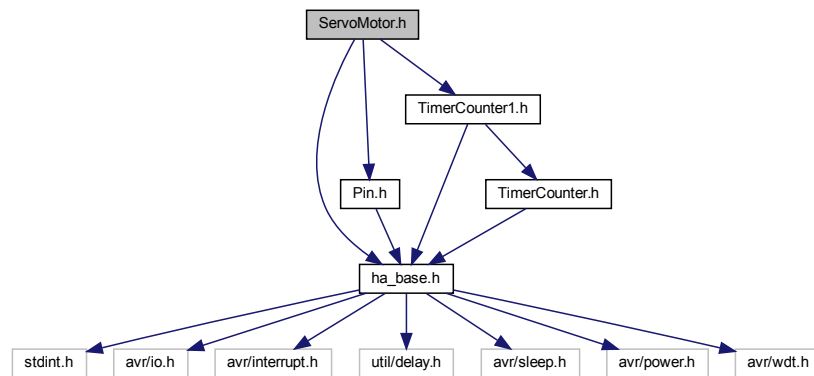
Header file of the ServoMotor class.

```

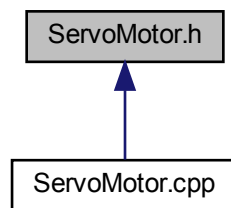
#include "ha_base.h"
#include "Pin.h"
#include "TimerCounter1.h"

```

Include dependency graph for ServoMotor.h:



This graph shows which files directly or indirectly include this file:



## Classes

- class [component::ServoMotor](#)

## Namespaces

- [component](#)  
*AVR chip external components.*

### 10.70.1 Detailed Description

Header file of the ServoMotor class.

Header file of the StepperMotor class.

Usage example (separate): `#include "MCU.h" #include "ServoMotor.h" #include "TimerCounter1.h"`

```

#define SERVOMOTOR_NUMBER 1 #define SERVOMOTOR_PULSE_CYCLE 30000 // pulse cycle [us] #define
SERVOMOTOR_PULSE_WIDTH_MIN 500// pulse width min [us] #define SERVOMOTOR_PULSE_WIDTH_MID
1520 // pulse width mid [us] #define SERVOMOTOR_PULSE_WIDTH_MAX 3000 // pulse width max [us]

int main(void) {

Init initialize MCU core::MCU::init();

instantiate the TimerCounter0 object core::TimerCounter1 &myTimerCounter1 = core::TimerCounter1::getInstance();
myTimerCounter1.selectClockSource(core::clockSource::PS_8); myTimerCounter1.selectOperationMode(core::
::operationMode::fast_PWM_ICR); myTimerCounter1.selectCompareOutputMode(core::channel::A, core::
::compareOutputMode::clear);

instantiate the Buzzer object component::ServoMotor myServoMotor(io::Pin(SERVOMOTOR_NUMBER,io::PortB),
SERVOMOTOR_PULSE_CYCLE, SERVOMOTOR_PULSE_WIDTH_MIN, SERVOMOTOR_PULSE_WIDTH_MID, SERVOMOTOR_PULSE_WIDTH_MAX);

myTimerCounter1.setInputCaptureRegister(myServoMotor.computePulseCycleCount(myTimerCounter1.getClockPrescaler())); myTimerCounter1.setOutputCompareRegister(core::channel::A, myServoMotor.computeRotationAngleCount(0,myTimerCounter1.getClockPrescaler())); myTimerCounter1.start(); _delay_ms(2000); myTimerCounter1.setOutputCompareRegister(core::channel::A, myServoMotor.computeRotationAngleCount(45,myTimerCounter1.getClockPrescaler())); myTimerCounter1.start(); _delay_ms(2000); myTimerCounter1.setOutputCompareRegister(core::channel::A, myServoMotor.computeRotationAngleCount(90,myTimerCounter1.getClockPrescaler())); myTimerCounter1.start(); _delay_ms(2000); myTimerCounter1.setOutputCompareRegister(core::channel::A, myServoMotor.computeRotationAngleCount(135,myTimerCounter1.getClockPrescaler())); myTimerCounter1.start(); _delay_ms(2000); myTimerCounter1.setOutputCompareRegister(core::channel::A, myServoMotor.computeRotationAngleCount(180,myTimerCounter1.getClockPrescaler())); myTimerCounter1.start(); _delay_ms(2000); myTimerCounter1.setOutputCompareRegister(core::channel::A, myServoMotor.computeRotationAngleCount(0,myTimerCounter1.getClockPrescaler())); myTimerCounter1.start(); _delay_ms(2000);

Mainloop while (1) {

} return 0; } Usage example (TimerCounter1): #include "MCU.h" #include "ServoMotor.h" #include "TimerCounter1.h"

#define SERVOMOTOR_NUMBER 1 #define SERVOMOTOR_PULSE_CYCLE 30000 // pulse cycle [us] #define
SERVOMOTOR_PULSE_WIDTH_MIN 500// pulse width min [us] #define SERVOMOTOR_PULSE_WIDTH_MID
1520 // pulse width mid [us] #define SERVOMOTOR_PULSE_WIDTH_MAX 3000 // pulse width max [us]

int main(void) {

Init initialize MCU core::MCU::init();

instantiate the TimerCounter1 object core::TimerCounter1 &myTimerCounter1 = core::TimerCounter1::getInstance();
myTimerCounter1.selectClockSource(core::clockSource::PS_8);

instantiate the Buzzer object component::ServoMotor myServoMotor(io::Pin(SERVOMOTOR_NUMBER,io::PortB),
SERVOMOTOR_PULSE_CYCLE, SERVOMOTOR_PULSE_WIDTH_MIN, SERVOMOTOR_PULSE_WIDTH_MID, SERVOMOTOR_PULSE_WIDTH_MAX);

myServoMotor.connect(myTimerCounter1);

myServoMotor.rotate(myTimerCounter1,0); _delay_ms(2000); myServoMotor.rotate(myTimerCounter1,45); _delay_ms(2000); myServoMotor.rotate(myTimerCounter1,90); _delay_ms(2000); myServoMotor.rotate(myTimerCounter1,135); _delay_ms(2000); myServoMotor.rotate(myTimerCounter1,180); _delay_ms(2000); myServoMotor.rotate(myTimerCounter1,0); _delay_ms(2000);

myServoMotor.disconnect(myTimerCounter1);

Mainloop while (1) {

}
return 0;

}

class to control a servo motor

```

**Author**

Farid Oubbati ( <https://github.com/faroub>)

**Date**

March 2020

The StepperMotor class is implemented and tested for the 28BYJ-48 Stepper Motor Usage example (basic):

```
#include "StepperMotor.h"
```

```
#define StepperMotor_Pin_1 0 #define StepperMotor_Pin_2 1 #define StepperMotor_Pin_3 2 #define StepperMotor_Pin_4 3
```

```
#define STEP_ANGLE_FULL_STEP 0.176 #define STEP_ANGLE_HALF_STEP 0.0879
```

```
int main(void) {
```

```
    instantiate the StepperMotor object component::StepperMotor myStepperMotor(component::mode::halfStep,
    io::Pin(StepperMotor_Pin_1,io::PortB), io::Pin(StepperMotor_Pin_2,io::PortB), io::Pin(StepperMotor_Pin_3,io::PortB),
    io::Pin(StepperMotor_Pin_4,io::PortB));
```

```
    int16_t l_angle = -180; int16_t l_step = static_cast<int16_t>(l_angle / STEP_ANGLE_HALF_STEP);
```

```
    set speed [steps/s] full step: max speed: 500p/s half step: max speed: 1000p/s
```

```
    set speed [%/s] full step: max speed: 40%/s half step: max speed: 80%/s
```

```
    Mainloop while (1) {
```

```
        while (!myStepperMotor.goalReached())
        {
```

```
            set motor steps myStepperMotor.step(l_step,1000); myStepperMotor.step(l_step,80,STEP_ANGLE_HALF_STEP);
```

```
        }
```

```
    } return 0; }
```

```
class to control a servo motor
```

**Author**

Farid Oubbati ( <https://github.com/faroub>)

**Date**

March 2020

Definition in file [ServoMotor.h](#).

## 10.71 ServoMotor.h

```

00001
00126 #ifndef SERVOMOTOR_H
00127 #define SERVOMOTOR_H
00128 #include "ha_base.h"
00129 #include "Pin.h"
00130 #include "TimerCounter1.h"
00131
00132
00133
00134 namespace component
00135 {
00136
00137
00138 class ServoMotor
00139 {
00140 public:
00141
00142     ServoMotor(const io::Pin &ar_pin,
00143                const uint16_t &ar_pulseCycle=0,
00144                const uint16_t &ar_pulseWidthMin=0,
00145                const uint16_t &ar_pulseWidthMid=0,
00146                const uint16_t &ar_pulseWidthMax=0);
00147
00148     ~ServoMotor();
00149
00150     void on();
00151     void off();
00152     void toggle();
00153
00154     uint16_t computePulseCycleCount(const uint16_t &ar_clockPrescaler);
00155     uint16_t computePulseWidthMinCount(const uint16_t &ar_clockPrescaler);
00156     uint16_t computePulseWidthMidCount(const uint16_t &ar_clockPrescaler);
00157     uint16_t computePulseWidthMaxCount(const uint16_t &ar_clockPrescaler);
00158     uint16_t computeRotationAngleCount(const uint8_t &ar_angle_deg, const uint16_t
&ar_clockPrescaler);
00159
00160     void rotate(core::TimerCounter1 &ar_timerCounter1,
00161                const uint8_t &ar_angle_deg,
00162                const core::channel &ar_channel=core::channel::A);
00163
00164     void connect(core::TimerCounter1 &ar_timerCounter1,
00165                 const core::channel &ar_channel=core::channel::A);
00166
00167     void disconnect(core::TimerCounter1 &ar_timerCounter1,
00168                    const core::channel &ar_channel=core::channel::A);
00169
00170
00171
00172
00173
00174
00175
00176
00177
00178
00179
00180
00181
00182
00183 protected:
00184
00185 private:
00186
00187
00188     io::Pin m_pin;
00189     uint16_t m_pulseCycle;
00190     uint16_t m_pulseWidthMin;
00191     uint16_t m_pulseWidthMid;
00192     uint16_t m_pulseWidthMax;
00193 };
00194
00195
00196
00197
00198
00199
00200
00201
00202
00203
00204
00205
00206
00207
00208
00209
00210
00211
00212 #endif

```

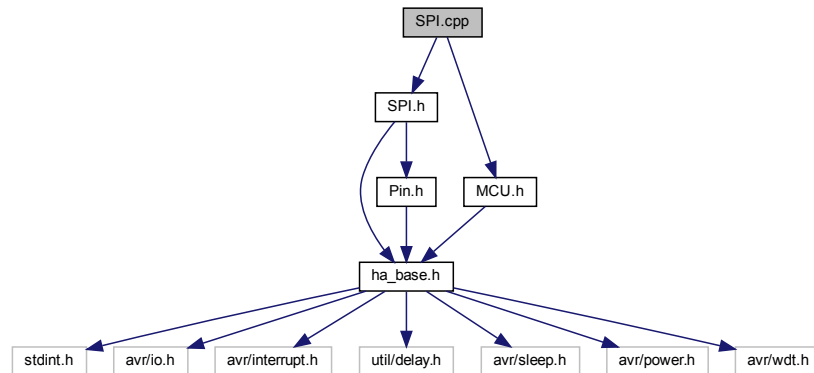
## 10.72 SPI.cpp File Reference

```

#include "SPI.h"
#include "MCU.h"

```

Include dependency graph for SPI.cpp:



## 10.73 SPI.cpp

```

00001 #include "SPI.h"
00002 #include "MCU.h"
00003
00004 volatile uint8_t io::SPI::m_data = 0;
00005
00006
00007
00008 io::SPI& io::SPI::getInstance(const Pin &ar_pinSCK,
00009                               const Pin &ar_pinMISO,
00010                               const Pin &ar_pinMOSI,
00011                               const Pin &ar_pinSS)
00012 {
00013     static SPI l_instance(ar_pinSCK,
00014                           ar_pinMISO,
00015                           ar_pinMOSI,
00016                           ar_pinSS);
00017
00018     return l_instance;
00019 }
00020
00021
00022 io::SPI::SPI(const Pin &ar_pinSCK,
00023              const Pin &ar_pinMISO,
00024              const Pin &ar_pinMOSI,
00025              const Pin &ar_pinSS)
00026 : m_pinSCK(ar_pinSCK),
00027   m_pinMISO(ar_pinMISO),
00028   m_pinMOSI(ar_pinMOSI),
00029   m_pinSS(ar_pinSS)
00030 {
00031     core::MCU::enableSPI(1);
00032     sei();
00033     enableTransferCompleteInterrupt(1);
00034 }
00035
00036
00037 io::SPI::~SPI()
00038 {
00039
00040 }
00041
00042 uint8_t io::SPI::writeCollision()
00043 {
00044     return (SPI_STATUS_REGISTER & (1 << SPI_WRITE_COLLISION));
00045 }
00046
00047
00048 uint8_t io::SPI::transferComplete()
00049 {
00050     return (SPI_STATUS_REGISTER & (1 << SPI_TRANSFER_COMPLETE));
00051 }
00052
00053 void io::SPI::selectClockPrescaler(const clockPrescaler& ar_clockPrescaler)

```



```

00054 {
00055     SPI_SELECT_CLOCK_PRESCALER(static_cast<uint8_t>(ar_clockPrescaler));
00056
00057 }
00058
00059 void io::SPI::selectOperationMode(const operationMode& ar_operationMode)
00060 {
00061     switch (ar_operationMode)
00062     {
00063         case operationMode::master:
00064         {
00065             m_pinMOSI.toOutput();
00066             m_pinSCK.toOutput();
00067             m_pinMISO.toInput(1);
00068             m_pinSS.toOutput();
00069             m_pinSS.setHigh();
00070             SPI_SELECT_MASTER_MODE;
00071             SPI_ENABLE;
00072             break;
00073         }
00074         case operationMode::slave:
00075         {
00076             m_pinMISO.toOutput();
00077             SPI_SELECT_SLAVE_MODE;
00078             SPI_ENABLE;
00079             break;
00080         }
00081         case operationMode::submaster:
00082         {
00083             m_pinMOSI.toOutput();
00084             m_pinSCK.toOutput();
00085             m_pinMISO.toInput(1);
00086             m_pinSS.toInput(1);
00087             SPI_SELECT_MASTER_MODE;
00088             SPI_ENABLE;
00089             break;
00090         }
00091         case operationMode::disable:
00092         {
00093             SPI_DISABLE;
00094             break;
00095         }
00096     }
00097 }
00098
00099 void io::SPI::selectDataMode(const dataMode& ar_dataMode)
00100 {
00101     SPI_SELECT_DATA_MODE(static_cast<uint8_t>(ar_dataMode));
00102 }
00103
00104 void io::SPI::selectDataOrder(const dataOrder& ar_dataOrder)
00105 {
00106     SPI_SELECT_DATA_ORDER(static_cast<uint8_t>(ar_dataOrder));
00107 }
00108 }
00109
00110 void io::SPI::enableTransferCompleteInterrupt(const uint8_t a_enable)
00111 {
00112     if (a_enable) {
00113         SPI_ENABLE_TRANSFER_COMPLETE_INTERRUPT;
00114     } else {
00115         SPI_DISABLE_TRANSFER_COMPLETE_INTERRUPT;
00116     }
00117 }
00118
00119 void io::SPI::selectSlave(const uint8_t a_select)
00120 {
00121     if (a_select) {
00122         m_pinSS.setLow();
00123     } else {
00124         m_pinSS.setHigh();
00125     }
00126 }
00127
00128 }
00129 void io::SPI::transferCompleteServiceRoutine()
00130 {
00131
00132     m_data = SPI_DATA_REGISTER;
00133 }
00134
00135 void io::SPI::masterSendByte(const uint8_t &ar_byte)
00136 {
00137
00138     SPI_DATA_REGISTER = ar_byte;
00139     selectSlave(1);
00140     while(!transferComplete()){};

```

```

00141     selectSlave(0);
00142
00143
00144
00145 }
00146
00147 void io::SPI::masterReceiveByte(uint8_t &ar_byte)
00148 {
00149     masterSendByte(0);
00150     ar_byte = m_data;
00151 }
00152 }
00153
00154 void io::SPI::slaveReceiveByte(uint8_t &ar_byte)
00155 {
00156     ar_byte = m_data;
00157 }
00158 }

```

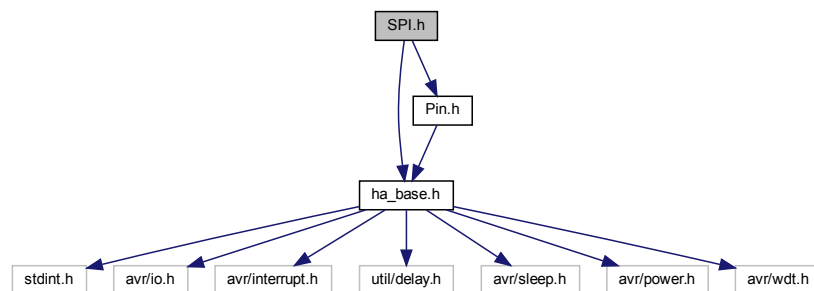
## 10.74 SPI.h File Reference

Header file of the SPI class.

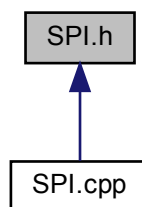
```
#include "ha_base.h"
```

```
#include "Pin.h"
```

Include dependency graph for SPI.h:



This graph shows which files directly or indirectly include this file:



## Classes

- class `io::SPI`

## Namespaces

- [io](#)

*AVR chip internal i/o components.*

## Enumerations

- enum [io::operationMode](#) : uint8\_t { [io::operationMode::master](#) =0, [io::operationMode::slave](#), [io::operationMode::submaster](#), [io::operationMode::disable](#) }
- enum [io::clockPrescaler](#) : uint8\_t { [io::clockPrescaler::PS\\_4](#) = 0, [io::clockPrescaler::PS\\_16](#), [io::clockPrescaler::PS\\_64](#), [io::clockPrescaler::PS\\_128](#), [io::clockPrescaler::PS\\_2](#), [io::clockPrescaler::PS\\_8](#), [io::clockPrescaler::PS\\_32](#) }
- enum [io::dataMode](#) : uint8\_t { [io::dataMode::mode\\_0](#) = 0, [io::dataMode::mode\\_1](#), [io::dataMode::mode\\_2](#), [io::dataMode::mode\\_3](#) }
- enum [io::dataOrder](#) : uint8\_t { [io::dataOrder::first\\_MSB](#) = 0, [io::dataOrder::first\\_LSB](#) }

### 10.74.1 Detailed Description

Header file of the SPI class.

Usage example (test): #include "MCU.h" #include "SPI.h"

```
#define SPI_SCK 5 #define SPI_MISO 4 #define SPI_MOSI 3 #define SPI_SS 2
```

```
int main(void) {
```

```
    Init initialize MCU core::MCU::init\(\); instantiate a SPI object io::SPI &myISP = io::SPI::getInstance(io::Pin(SPI_SCK,io::PortB),  
io::Pin(SPI_MISO,io::PortB), io::Pin(SPI_MOSI,io::PortB), io::Pin(SPI_SS,io::PortB));
```

```
    myISP.selectClockPrescaler(io::clockPrescaler::PS\_128); myISP.selectDataMode(io::dataMode::mode\_0); myISP.  
P.selectDataOrder(io::dataOrder::first\_LSB); myISP.selectOperationMode(io::operationMode::master);
```

```
    myISP.masterSendByte(0x03);
```

```
    Mainloop while (1) {
```

```
    } return 0; }
```

Basic class for IO abstraction of Pin and Port

#### Author

Farid Oubbati ( <https://github.com/faroub>)

#### Date

March 2020

Definition in file [SPI.h](#).

## 10.75 SPI.h

```

00001
00048 #ifndef SPI_H
00049 #define SPI_H
00050 #include "ha_base.h"
00051 #include "Pin.h"
00052
00053 namespace io
00054 {
00055
00056 enum class operationMode : uint8_t {
00057     master=0,
00058     slave,
00059     submaster,
00060     disable,
00061 };
00062
00063 enum class clockPrescaler : uint8_t {
00064     PS_4 = 0,
00065     PS_16,
00066     PS_64,
00067     PS_128,
00068     PS_2,
00069     PS_8,
00070     PS_32
00071 };
00072
00073 enum class dataMode : uint8_t {
00074     mode_0 = 0,
00075     mode_1,
00076     mode_2,
00077     mode_3,
00078 };
00079
00080 enum class dataOrder : uint8_t {
00081     first_MSB = 0,
00082     first_LSB
00083 };
00084
00085
00086 class SPI
00087 {
00088 public:
00089     static SPI& getInstance(const io::Pin &ar_pinSCK,
00090                             const io::Pin &ar_pinMISO,
00091                             const io::Pin &ar_pinMOSI,
00092                             const io::Pin &ar_pinSS);
00093
00094     void selectDataMode(const dataMode& ar_dataMode);
00095
00096     void selectDataOrder(const dataOrder& ar_dataOrder);
00097
00098     void selectOperationMode(const operationMode& ar_operationMode);
00099
00100     void selectClockPrescaler(const clockPrescaler& ar_clockPrescaler);
00101
00102     void selectSlave(const uint8_t a_select);
00103
00104     uint8_t writeCollision();
00107
00108     uint8_t transferComplete();
00112
00113     static void enableTransferCompleteInterrupt(const uint8_t a_enable);
00115
00116     void masterSendByte(const uint8_t &ar_byte);
00117
00118     void masterReceiveByte(uint8_t &ar_byte);
00119
00120     void slaveReceiveByte(uint8_t &ar_byte);
00121
00124     static void transferCompleteServiceRoutine() __asm__(STR(SPI_TRANSFER_COMPLETE_INTERRUPT))
__attribute__((__signal__, __used__, __externally_visible__));
00125
00126
00127 private:
00128     SPI(const io::Pin &ar_pinSCK,
00137         const io::Pin &ar_pinMISO,
00138         const io::Pin &ar_pinMOSI,
00139         const io::Pin &ar_pinSS);
00140
00141     ~SPI();
00144
00145     SPI(const SPI&);
00148

```

```

00149
00152     const SPI& operator=(const SPI&);
00153
00154     static volatile uint8_t m_data;
00155
00156     io::Pin m_pinSCK;
00157     io::Pin m_pinMISO;
00158     io::Pin m_pinMOSI;
00159     io::Pin m_pinSS;
00167 };
00168
00169 }
00170 #endif // SPI_H

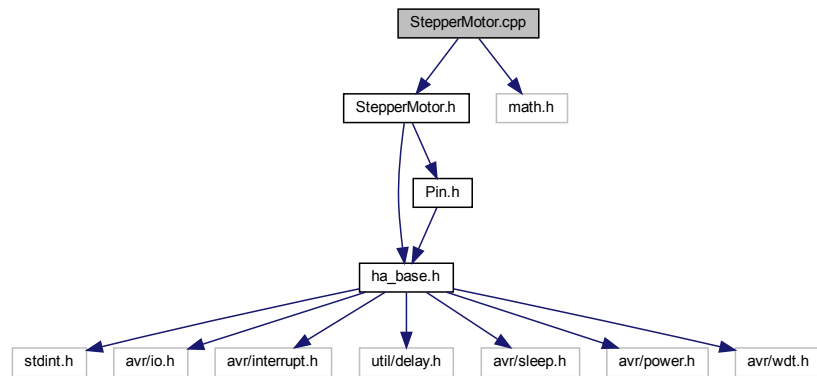
```

## 10.76 StepperMotor.cpp File Reference

```
#include "StepperMotor.h"
```

```
#include <math.h>
```

Include dependency graph for StepperMotor.cpp:



## 10.77 StepperMotor.cpp

```

00001 #include "StepperMotor.h"
00002 #include <math.h>
00003
00004 // TODO: steps to angle
00005 // TODO: maximum speed
00006 // TODO: speed and acceleration profiles
00007
00008 component::StepperMotor::StepperMotor(const mode &ar_mode,
00009                                         const io::Pin &ar_pinCoil1,
00010                                         const io::Pin &ar_pinCoil2,
00011                                         const io::Pin &ar_pinCoil3,
00012                                         const io::Pin &ar_pinCoil4)
00013 : m_pinCoil1(ar_pinCoil1),
00014   m_pinCoil2(ar_pinCoil2),
00015   m_pinCoil3(ar_pinCoil3),
00016   m_pinCoil4(ar_pinCoil4),
00017   stepMode(ar_mode)
00018 {
00019
00020     m_pinCoil1.toOutput();
00021     m_pinCoil2.toOutput();
00022     m_pinCoil3.toOutput();
00023     m_pinCoil4.toOutput();
00024     m_goalReached = 0;
00025     m_currentPos = 0;
00026
00027 }
00028 }

```

```

00029
00030 component::StepperMotor::~StepperMotor()
00031 {
00032 }
00033 }
00034
00035 void component::StepperMotor::setCurrentPos(uint16_t a_currentPos)
00036 {
00037     m_currentPos = a_currentPos;
00038 }
00039
00040 uint16_t component::StepperMotor::currentPos()
00041 {
00042     return m_currentPos;
00043 }
00044
00045
00046
00047 void component::StepperMotor::step(const int16_t a_step,
00048     const uint16_t a_speed,
00049     const uint16_t a_accel,
00050     const uint16_t a_decel)
00051 {
00052     static int16_t l_stepNumber = 0;
00053     uint8_t l_stepDelay_ms = 0;
00054     uint16_t l_step=0;
00055
00056     if (a_step<=0)
00057     {
00058         l_step = -a_step;
00059     } else
00060     {
00061         l_step = a_step;
00062     }
00063
00064     l_stepDelay_ms = computeStepDelay(l_step,
00065                                     a_speed,
00066                                     a_accel,
00067                                     a_decel);
00068
00069     if (l_stepDelay_ms>0)
00070     {
00071
00072         if (a_step<=0)
00073         {
00074             if (l_stepNumber== -a_step)
00075             {
00076                 m_goalReached = 1;
00077
00078             } else {
00079                 l_stepNumber++;
00080                 m_currentPos++;
00081             }
00082
00083         } else {
00084
00085             if (l_stepNumber == -a_step)
00086             {
00087                 m_goalReached = 1;
00088
00089             } else {
00090                 l_stepNumber--;
00091                 m_currentPos--;
00092             }
00093
00094         }
00095     }
00096
00097
00098     if (m_goalReached == 0)
00099     {
00100
00101         switch (stepMode)
00102         {
00103             case mode::fullStep:
00104             {
00105                 // equivalent to l_stepNumber % 4
00106                 // 1 3 5 7 <->
00107                 stepPulse(2*(l_stepNumber & 3)+1);
00108                 break;
00109             }
00110
00111             case mode::halfStep:
00112             {
00113                 // equivalent to l_stepNumber % 8
00114                 // 0 1 2 3 4 5 6 7 <->
00115                 stepPulse(l_stepNumber & 7);

```

```

00116             break;
00117         }
00118     }
00119
00120     stepDelay(l_stepDelay_ms);
00121
00122 }
00123 }
00124
00125 }
00126
00127 uint8_t component::StepperMotor::computeStepDelay(uint16_t a_step,
00128             uint16_t a_speed,
00129             uint16_t a_accel,
00130             uint16_t a_decel)
00131 {
00132
00133
00134     uint32_t l_accelTime= (1000UL*a_speed/a_accel);
00135     uint16_t l_decelTime = (1000UL*a_speed/a_decel);
00136     int16_t l_constSpeedTime=((1000UL*a_step/a_speed)-(l_accelTime/2)-(l_decelTime/2));
00137     static uint32_t l_time=0;
00138     static uint32_t l_speed=0;
00139     static uint64_t l_speed_time_product=0;
00140     static uint64_t l_current_position=0;
00141
00142
00143
00144
00145
00146
00147
00148     if (l_constSpeedTime<0)
00149     {
00150         l_constSpeedTime=0;
00151         l_accelTime = static_cast<uint32_t>(1000UL*sqrtf(a_step/a_accel));
00152         a_speed=l_accelTime*a_accel/1000UL;
00153         l_decelTime = 1000UL*a_speed/a_decel;
00154     }
00155
00156
00157
00158
00159
00160     if (l_time<=l_accelTime)
00161     {
00162         l_speed=a_accel*l_time;
00163     }
00164     else if ((l_time>l_accelTime) && (l_time<=(l_accelTime+l_constSpeedTime)))
00165     {
00166         l_speed=1000UL*a_speed;
00167     }
00168     else if
00169     ((l_time>(l_accelTime+l_constSpeedTime)) && (l_time<=l_accelTime+l_constSpeedTime+l_decelTime))
00170     {
00171         l_speed= (1000UL*a_speed)-a_decel*(l_time-l_accelTime-l_constSpeedTime);
00172     }
00173
00174
00175
00176
00177     l_time=l_time+1;
00178
00179
00180     l_speed_time_product = l_speed_time_product+l_speed;
00181
00182
00183     if (l_speed_time_product - l_current_position >= 1000000UL)
00184     {
00185         l_current_position=l_current_position+1000000UL;
00186         return 1;
00187     }
00188     else
00189     {
00190         stepDelay(1);
00191         return 0;
00192     }
00193
00194
00195
00196
00197
00198 }
00199
00200
00201 void component::StepperMotor::step(const int16_t a_step, const uint16_t a_speed)

```

```

00202 {
00203     static int16_t l_stepNumber = 0;
00204     static uint8_t l_stepDelay_ms = static_cast<uint8_t>(1000UL/a_speed);
00205
00206     if (a_step<=0)
00207     {
00208         if (l_stepNumber== -a_step)
00209         {
00210             m_goalReached = 1;
00211         } else {
00212             l_stepNumber++;
00213             m_currentPos++;
00214         }
00215     }
00216
00217 } else {
00218     if (l_stepNumber == -a_step)
00219     {
00220         m_goalReached = 1;
00221     } else {
00222         l_stepNumber--;
00223         m_currentPos--;
00224     }
00225 }
00226
00227 if (m_goalReached == 0)
00228 {
00229     switch (stepMode)
00230     {
00231     case mode::fullStep:
00232     {
00233         // equivalent to l_stepNumber % 4
00234         // 1 3 5 7 <->
00235         stepPulse(2*(l_stepNumber & 3)+1);
00236         break;
00237     }
00238     case mode::halfStep:
00239     {
00240         // equivalent to l_stepNumber % 8
00241         // 0 1 2 3 4 5 6 7 <->
00242         stepPulse(l_stepNumber & 7);
00243         break;
00244     }
00245     }
00246     stepDelay(l_stepDelay_ms);
00247 }
00248 }
00249
00250 void component::StepperMotor::step(const int16_t a_step, const uint16_t a_speed, const float
a_stepAngle)
00251 {
00252     static int16_t l_stepNumber = 0;
00253     static uint8_t l_stepDelay_ms = static_cast<uint8_t>(1000UL*a_stepAngle/a_speed);
00254
00255     if (a_step<=0)
00256     {
00257         if (l_stepNumber== -a_step)
00258         {
00259             m_goalReached = 1;
00260         } else {
00261             l_stepNumber++;
00262             m_currentPos++;
00263         }
00264     }
00265
00266 } else {
00267     if (l_stepNumber == -a_step)
00268     {
00269         m_goalReached = 1;
00270     } else {
00271         l_stepNumber--;
00272         m_currentPos--;
00273     }
00274 }
00275
00276 if (m_goalReached == 0)
00277 {
00278     switch (stepMode)
00279     {
00280     case mode::fullStep:
00281     {
00282         // equivalent to l_stepNumber % 4
00283         // 1 3 5 7 <->
00284         stepPulse(2*(l_stepNumber & 3)+1);
00285         break;
00286     }
00287     case mode::halfStep:
00288     {
00289         // equivalent to l_stepNumber % 8
00290         // 0 1 2 3 4 5 6 7 <->
00291         stepPulse(l_stepNumber & 7);
00292         break;
00293     }
00294     }
00295     stepDelay(l_stepDelay_ms);
00296 }
00297 }

```



```

00288
00289
00290     if (m_goalReached == 0)
00291     {
00292
00293         switch (stepMode)
00294         {
00295             case mode::fullStep:
00296             {
00297                 // equivalent to l_stepNumber % 4
00298                 // 1 3 5 7 <->
00299                 stepPulse(2*(l_stepNumber & 3)+1);
00300                 break;
00301             }
00302
00303             case mode::halfStep:
00304             {
00305                 // equivalent to l_stepNumber % 8
00306                 // 0 1 2 3 4 5 6 7 <->
00307                 stepPulse(l_stepNumber & 7);
00308                 break;
00309             }
00310         }
00311
00312         stepDelay(l_stepDelay_ms);
00313     }
00314 }
00315
00316
00317 }
00318
00319
00320 void component::StepperMotor::stepPulse(const uint8_t a_stepPulse)
00321 {
00322
00323     switch (a_stepPulse) {
00324         case 0: // 1000
00325         {
00326             m_pinCoil1.setHigh();
00327             m_pinCoil2.setLow();
00328             m_pinCoil3.setLow();
00329             m_pinCoil4.setLow();
00330             break;
00331         }
00332         case 1: // 1100
00333         {
00334             m_pinCoil1.setHigh();
00335             m_pinCoil2.setHigh();
00336             m_pinCoil3.setLow();
00337             m_pinCoil4.setLow();
00338             break;
00339         }
00340         case 2: //0100
00341         {
00342             m_pinCoil1.setLow();
00343             m_pinCoil2.setHigh();
00344             m_pinCoil3.setLow();
00345             m_pinCoil4.setLow();
00346             break;
00347         }
00348         case 3: //0110
00349         {
00350             m_pinCoil1.setLow();
00351             m_pinCoil2.setHigh();
00352             m_pinCoil3.setHigh();
00353             m_pinCoil4.setLow();
00354             break;
00355         }
00356         case 4: //0010
00357         {
00358             m_pinCoil1.setLow();
00359             m_pinCoil2.setLow();
00360             m_pinCoil3.setHigh();
00361             m_pinCoil4.setLow();
00362             break;
00363         }
00364         case 5: //0011
00365         {
00366             m_pinCoil1.setLow();
00367             m_pinCoil2.setLow();
00368             m_pinCoil3.setHigh();
00369             m_pinCoil4.setHigh();
00370             break;
00371         }
00372         case 6: //0001
00373         {
00374             m_pinCoil1.setLow();

```

```

00375         m_pinCoil2.setLow();
00376         m_pinCoil3.setLow();
00377         m_pinCoil4.setHigh();
00378         break;
00379     }
00380     case 7:    //1001
00381     {
00382         m_pinCoil1.setHigh();
00383         m_pinCoil2.setLow();
00384         m_pinCoil3.setLow();
00385         m_pinCoil4.setHigh();
00386         break;
00387     }
00388 }
00389
00390
00391 }
00392
00393
00394
00395 void component::StepperMotor::stepDelay(uint8_t a_stepDelay)
00396 {
00397     while(a_stepDelay-->0)
00398     {
00399         _delay_ms(1);
00400     }
00401 }
00402
00403 uint8_t component::StepperMotor::goalReached()
00404 {
00405     return m_goalReached;
00406 }
00407 }

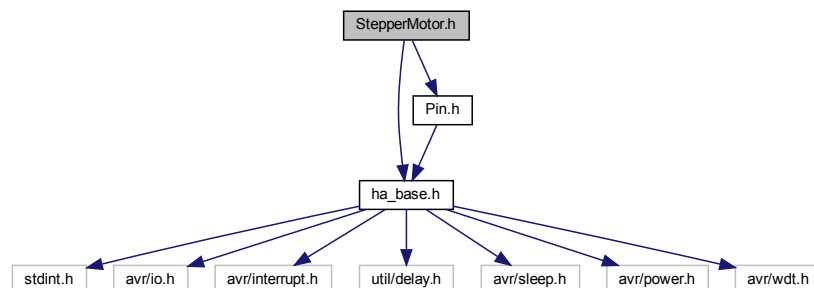
```

## 10.78 StepperMotor.h File Reference

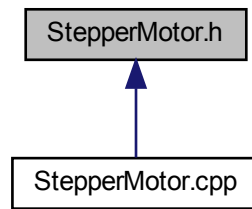
```
#include "ha_base.h"
```

```
#include "Pin.h"
```

Include dependency graph for StepperMotor.h:



This graph shows which files directly or indirectly include this file:



## Classes

- class `component::StepperMotor`

## Namespaces

- `component`  
*AVR chip external components.*

## Enumerations

- enum `component::mode` : `uint8_t` { `component::mode::fullStep` =0, `component::mode::halfStep` }

## 10.79 StepperMotor.h

```

00001
00062 #ifndef STEPPERMOTOR_H
00063 #define STEPPERMOTOR_H
00064 #include "ha_base.h"
00065 #include "Pin.h"
00066
00067 namespace component
00068 {
00069
00070 enum class mode : uint8_t {
00071     fullStep=0,
00072     halfStep,
00073 };
00074
00075 class StepperMotor
00076 {
00077 public:
00078
00079     StepperMotor(const mode &ar_mode,
00080                 const io::Pin &ar_pinCoil1,
00081                 const io::Pin &ar_pinCoil2,
00082                 const io::Pin &ar_pinCoil3,
00083                 const io::Pin &ar_pinCoil4);
00084
00085     ~StepperMotor();
00086
00090     void step(const int16_t a_step,
00091              const uint16_t a_speed);
00092
00093     void step(const int16_t a_step,
00094              const uint16_t a_speed,
  
```

```

00095         const float a_stepAngle);
00096
00097 void step(const int16_t a_step,
00098          const uint16_t a_speed,
00099          const uint16_t a_accel,
00100          const uint16_t a_decel);
00101
00102 void stepPulse(const uint8_t a_stepPulse);
00103
00104 void stepDelay(uint8_t a_stepDelay);
00105
00106 uint8_t goalReached();
00107
00108 void setCurrentPos(uint16_t a_currentPos);
00109
00110 uint16_t currentPos();
00111
00112
00113 uint8_t computeStepDelay(uint16_t a_step,
00114                          const uint16_t a_speed,
00115                          const uint16_t a_accel,
00116                          const uint16_t a_decel);
00117
00118 uint16_t m_accelTime;
00119 uint16_t m_decelTime;
00120 uint16_t m_constSpeedTime;
00124 private:
00125
00126 //      uint8_t computeStepDelay(int16_t a_step,
00127 //                              const uint16_t a_speed,
00128 //                              const uint16_t a_accel,
00129 //                              const uint16_t a_decel);
00130
00131
00132 io::Pin m_pinCoil1;
00133 io::Pin m_pinCoil2;
00134 io::Pin m_pinCoil3;
00135 io::Pin m_pinCoil4;
00137 mode stepMode;
00138 uint8_t m_goalReached;
00139 uint16_t m_currentPos;
00144 };
00145
00146 }
00147 #endif // STEPPERMOTOR_H

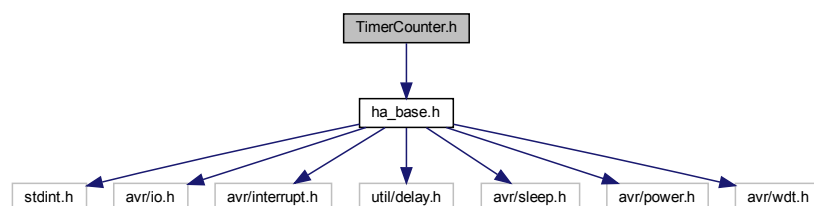
```

## 10.80 TimerCounter.h File Reference

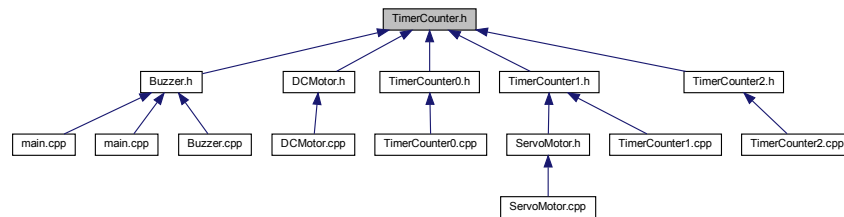
Header file of the TimerCounter class.

```
#include "ha_base.h"
```

Include dependency graph for TimerCounter.h:



This graph shows which files directly or indirectly include this file:



## Classes

- class [core::TimerCounter](#)

## Namespaces

- [core](#)  
*AVR chip internal core components.*

## Enumerations

- enum [core::channel](#) : `uint8_t` { [core::channel::A](#) =0, [core::channel::B](#) }
- enum [core::compareOutputMode](#) : `uint8_t` { [core::compareOutputMode::normal](#) =0, [core::compareOutputMode::toggle](#), [core::compareOutputMode::clear](#), [core::compareOutputMode::set](#) }
- enum [core::operationMode](#) : `uint8_t` {  
[core::operationMode::normal](#) =0, [core::operationMode::PWM\\_PC](#), [core::operationMode::PWM\\_PC\\_8bit](#),  
[core::operationMode::PWM\\_PC\\_9bit](#),  
[core::operationMode::PWM\\_PC\\_10bit](#), [core::operationMode::PWM\\_PFC\\_ICR](#), [core::operationMode::PWM\\_PFC\\_OCR](#),  
[core::operationMode::PWM\\_PC\\_ICR](#),  
[core::operationMode::PWM\\_PC\\_OCR](#), [core::operationMode::fast\\_PWM](#), [core::operationMode::fast\\_PWM\\_8bit](#),  
[core::operationMode::fast\\_PWM\\_9bit](#),  
[core::operationMode::fast\\_PWM\\_10bit](#), [core::operationMode::fast\\_PWM\\_ICR](#), [core::operationMode::fast\\_PWM\\_OCR](#),  
[core::operationMode::CTC\\_OCR](#),  
[core::operationMode::CTC\\_ICR](#), [core::operationMode::interrupt](#) =1, [core::operationMode::reset](#), [core::operationMode::interrupt](#)  
}
- enum [core::clockSource](#) : `uint16_t` {  
[core::clockSource::noClock](#) =0, [core::clockSource::PS\\_1](#), [core::clockSource::PS\\_8](#), [core::clockSource::PS\\_32](#),  
[core::clockSource::PS\\_64](#), [core::clockSource::PS\\_128](#), [core::clockSource::PS\\_256](#), [core::clockSource::PS\\_1024](#),  
[core::clockSource::extern\\_Clock\\_T0\\_Falling\\_Edge](#), [core::clockSource::extern\\_Clock\\_T0\\_Rising\\_Edge](#) }

### 10.80.1 Detailed Description

Header file of the TimerCounter class.

Basic class for abstraction of the TimerCounter peripherals.

**Author**

Farid Oubbati ( <https://github.com/faroub>)

**Date**

March 2020

Basic class for abstraction of the TimerCounter peripherals.

Usage example (Reaction Timer): #include "TimerCounter.h" #include "USART0.h" #include "PushButton.h" #include "Led.h"

```
void randomDelay(void); void printWord(uint16_t word);
```

```
instantiate the Counter object extern core::TimerCounter core::TimerCounter &myCounter = core::TimerCounter<
::getInstanceTimerCounter0();
```

```
instantiate the USART0 object extern io::USART0 io::USART0 &myUSART0 = io::USART0::getInstance();
```

```
#define BUFFER_SIZE 1 #define LED_NUMBER 0 #define PUSHBUTTON_NUMBER 2
```

```
int main(void) {
```

```
    Init uint16_t timerValue;
```

```
    char l_receiverBuffer[BUFFER_SIZE];
```

```
    myCounter.selectClockSource(core::timerCounter::timerCounter1, core::clockSource::PS_1024);
```

```
    / instantiate a Led object component::Led Led(io::Pin(LED_NUMBER,io::PortB));
```

```
    instantiate a Led object component::PushButton PushButton(io::Pin(PUSHBUTTON_NUMBER,io::PortD));
```

```
    ready to send flag uint8_t l_ready2Send = 1;
```

```
    if (myUSART0.ready2Send()) { myUSART0.sendString("Reaction Timer:\r\n"); }
```

```
    wait to send next string while (!myUSART0.ready2Send());
```

```
    if (myUSART0.ready2Send()) { myUSART0.sendString("-----\r\n"); }
```

```
    wait to send next string while (!myUSART0.ready2Send());
```

```
    if (myUSART0.ready2Send()) { myUSART0.sendString("Press any key to start.\r\n"); }
```

```
    Mainloop while (1) {
```

```
        myUSART0.receiveFrame(reinterpret_cast<uint8_t*>(l_receiverBuffer),BUFFER_SIZE);
```

```
        wait for a key press while (!myUSART0.getNumberBytesReceived()) {
```

```
    }
```

```
    reset number of bytes after extracting the received data myUSART0.resetNumberBytesReceived();
```

```
    wait to send next string while (!myUSART0.ready2Send());
```

```

if (myUSART0.ready2Send()) { myUSART0.sendString("\r\nGet ready..."); }

randomDelay();

wait to send next string while (!myUSART0.ready2Send());

if (myUSART0.ready2Send()) { myUSART0.sendString("\r\nGo!\r\n"); }

Led.on();

myCounter.setCounter(core::timerCounter::timerCounter1,0);

if (PushButton.isPressed()) { wait to send next string while (!myUSART0.ready2Send()); }

if (myUSART0.ready2Send()) { myUSART0.sendString("You're only cheating yourself.\r\n"); }

} else {

wait for a button press while (!PushButton.isPressed()) {

} get counter value myCounter.getCounter(core::timerCounter::timerCounter1, &timerValue); bit shift divide by 16 =
2^4 to convert from micro to milliseconds timerValue = timerValue >> 4; print response time printWord(timerValue);
}

Led.off();

while (!myUSART0.ready2Send());

if (myUSART0.ready2Send()) { myUSART0.sendString("\r\nPress any key to try again.\r\n"); }

} return 0; }

void randomDelay(void) {

Waits for a "random" delay from 1 - 3.5 sec Requires timer 1 initialized and running It's not really random, but very
hard to control –like coin-flipping.

uint16_t counter;
_delay_ms(1000);

myCounter.getCounter(core::timerCounter::timerCounter1, &counter);
uint8_t randomTime = static_cast<uint8_t>(counter);

```

type-casting the 16-bit TCNT1 as an 8-bit number keeps only the 8 least-significant (fastest-changing) bits

```

while (--randomTime) {
    _delay_ms(10);
}

void printWord(uint16_t word) {

while (!myUSART0.ready2Send());

if (myUSART0.ready2Send())
{
    myUSART0.sendByte('0' + (word / 10000));
}

while (!myUSART0.ready2Send());

```

```
if (myUSART0.ready2Send())
{
    myUSART0.sendByte('0' + ((word / 1000) % 10));
}

while (!myUSART0.ready2Send()){};

if (myUSART0.ready2Send())
{
    myUSART0.sendByte('0' + ((word / 100) % 10));
}

while (!myUSART0.ready2Send()){};

if (myUSART0.ready2Send())
{
    myUSART0.sendByte('0' + ((word / 10) % 10));
}

while (!myUSART0.ready2Send()){};

if (myUSART0.ready2Send())
{
    myUSART0.sendByte('0' + (word % 10));
}

}

void printWord(uint16_t word) {

char timerValue = '0' + (word / 10000);

while (!myUSART0.ready2Send()){};

if (myUSART0.ready2Send())
{
    myUSART0.sendFrame(reinterpret_cast<uint8_t*>(&timerValue),BUFFER_SIZE);
}

timerValue = '0' + ((word / 1000) % 10);

while (!myUSART0.ready2Send()){};

if (myUSART0.ready2Send())
{
    myUSART0.sendFrame(reinterpret_cast<uint8_t*>(&timerValue),BUFFER_SIZE);
}

timerValue = '0' + ((word / 100) % 10);

while (!myUSART0.ready2Send()){};

if (myUSART0.ready2Send())
{
    myUSART0.sendFrame(reinterpret_cast<uint8_t*>(&timerValue),BUFFER_SIZE);
}

timerValue = '0' + ((word / 10) % 10);

while (!myUSART0.ready2Send()){};

if (myUSART0.ready2Send())
{
    myUSART0.sendFrame(reinterpret_cast<uint8_t*>(&timerValue),BUFFER_SIZE);
}

}
```



```

timerValue = '0' + (word % 10);

while (!myUSART0.ready2Send()){};

if (myUSART0.ready2Send())
{
    myUSART0.sendFrame(reinterpret_cast<uint8_t*>(&timerValue), BUFFER_SIZE);
}

}

```

Usage example (PWM):

```

#include "Led.h" #include "USART0.h" #include "TimerCounter1.h" #include "TimerCounter2.h"

uint8_t getNumber(const char *ap_receive);

#define LED_1 1 #define LED_2 2 #define LED_3 3 #define BUFFER_SIZE 4

int main(void) {

    Init receiver buffer char l_receiverBuffer[BUFFER_SIZE];

    instantiate Led objects component::Led Led1(io::Pin(LED_1,io::PortB)); component::Led Led2(io::Pin(LED_2,io::PortB)); component::Led Led3(io::Pin(LED_3,io::PortB));

    instantiate USART0 object io::USART0 &myUSART0 = io::USART0::getInstance();

    instantiate Timer1 object core::TimerCounter1 &myTimerCounter1 = core::TimerCounter1::getInstance(); myTimerCounter1.selectOperationMode(core::operationMode::Fast_PWM_8bit); myTimerCounter1.selectCompareOutputMode(core::channel::A,core::compareOutputMode::Clear); myTimerCounter1.selectCompareOutputMode(core::channel::B,core::compareOutputMode::Clear); instantiate Timer2 object core::TimerCounter2 &myTimerCounter2 = core::TimerCounter2::getInstance(); myTimerCounter2.selectOperationMode(core::operationMode::Fast_PWM); myTimerCounter2.selectCompareOutputMode(core::channel::A,core::compareOutputMode::Clear);

    if (myUSART0.ready2Send()) { myUSART0.sendString("-- LED PWM Demo --\r\n"); } wait to send next string while (!myUSART0.ready2Send()){}

    myTimerCounter1.start(core::clockSource::PS_64); myTimerCounter2.start(core::clockSource::PS_64);

    ----- Event loop ----- // while (1) {

        if (myUSART0.ready2Send())
        {
            myUSART0.sendString("\r\nEnter (0-255) for PWM duty cycle: ");
        }

        myUSART0.receiveFrame(reinterpret_cast<uint8_t*>(l_receiverBuffer), BUFFER_SIZE);

        wait for a key press while (myUSART0.getNumberBytesReceived()<4){}

        reset number of bytes after extracting the received data myUSART0.resetNumberBytesReceived();

        send back the received character if (myUSART0.ready2Send()) { myUSART0.sendFrame(reinterpret_cast<uint8_t*>(l_receiverBuffer), BUFFER_SIZE); }

        while (!myUSART0.ready2Send()){} myTimerCounter2.setOutputCompareRegister(core::channel::A,myTimerCounter1.getOutputCompareRegister(core::channel::B)); myTimerCounter1.setOutputCompareRegister(core::channel::B,myTimerCounter1.getOutputCompareRegister(core::channel::A)); myTimerCounter1.setOutputCompareRegister(core::channel::A,getNumber(l_receiverBuffer));
    }
}

```

```

}
return 0;

```

```

}

```

```

uint8_t getNumber(const char *ap_receive) { Gets a numerical 0-255 from the serial port. Converts from string to
number. char hundreds = '0'; char tens = '0'; char ones = '0'; char thisChar = '0'; do { hundreds = tens; tens = ones;
ones = thisChar; thisChar = *ap_receive; ap_receive++;

```

```

} while (*ap_receive != '\r'); return (100 * (hundreds - '0') + 10 * (tens - '0') + ones - '0'); }

```

Usage example (PWM on Any Pin):

```

#include "TimerCounter0.h" #include "Led.h"

```

```

#define LED_0 0 #define LED_1 1 #define LED_2 2 #define LED_3 3

```

```

instantiate Timer0 object extern core::TimerCounter0 core::TimerCounter0 &myTimerCounter0 = core::TimerCounter0::getInstance();

```

```

instantiate Led objects extern component::Led Led0; component::Led Led0(io::Pin(LED_0,io::PortB)); ex-
tern component::Led Led1; component::Led Led1(io::Pin(LED_1,io::PortB)); extern component::Led Led2;
component::Led Led2(io::Pin(LED_2,io::PortB)); extern component::Led Led3; component::Led Led3(io::Pin(LED_3,io::PortB));

```

```

#define DELAYTIME 3

```

```

volatile uint8_t l_brightnessA; volatile uint8_t l_brightnessB;

```

```

int main(void) {

```

```

    Init uint8_t i;

```

```

    sei();

```

```

    myTimerCounter0.enableOutputCompareMatchInterrupt(core::channel::A,1); myTimerCounter0.enableOutputCompareMatchInterrupt(core::channel::B,1); myTimerCounter0.enableOverflowInterrupt(1); myTimerCounter0.start(core::clockSource::PS_1024);

```

```

    ----- Event loop ----- // while (1) {

```

```

        for (i = 0; i < 255; i++) {
            _delay_ms(DELAYTIME);
            l_brightnessA = i;
            l_brightnessB = 255 - i;
        }
        for (i = 254; i > 0; i--) {
            _delay_ms(DELAYTIME);
            l_brightnessA = i;
            l_brightnessB = 255 - i;
        }

```

```

    } return 0; }

```

```

void core::TimerCounter0::overflowServiceRoutine()
{
    Led0.on();
    Led1.on();
    Led2.on();
    Led3.on();
    myTimerCounter0.setOutputCompareRegister(core::channel::A, l_brightnessA);
    myTimerCounter0.setOutputCompareRegister(core::channel::B, l_brightnessB);
}

void core::TimerCounter0::outputCompareMatchAServiceRoutine()
{
    Led0.off();
    Led1.off();
    Led2.on();
    Led3.on();
}
void core::TimerCounter0::outputCompareMatchBServiceRoutine()
{
    Led0.on();
    Led1.on();
    Led2.off();
    Led3.off();
}

```

**Author**

Farid Oubbati ( <https://github.com/faroub>)

**Date**

March 2020

Basic class for abstraction of the TimerCounter peripherals.

Usage example (Reaction Timer): #include "TimerCounter.h" #include "USART0.h" #include "PushButton.h" #include "Led.h"

```
void randomDelay(void); void printWord(uint16_t word);
```

instantiate the Counter object extern `core::TimerCounter core::TimerCounter` &myCounter = core::TimerCounter↵  
::getInstanceTimerCounter0();

instantiate the USART0 object extern `io::USART0 io::USART0` &myUSART0 = io::USART0::getInstance();

```
#define BUFFER_SIZE 1 #define LED_NUMBER 0 #define PUSHBUTTON_NUMBER 2
```

```
int main(void) {
```

```
    Init uint16_t timerValue;
```

```
    char l_receiverBuffer[BUFFER_SIZE];
```

```
    myCounter.selectClockSource(core::timerCounter::timerCounter1, core::clockSource::PS_1024);
```

```

/ instantiate a Led object component::Led Led(io::Pin(LED_NUMBER,io::PortB));

instantiate a Led object component::PushButton PushButton(io::Pin(PUSHBUTTON_NUMBER,io::PortD));

ready to send flag uint8_t I_ready2Send = 1;

if (myUSART0.ready2Send()) { myUSART0.sendString("Reaction Timer:\r\n"); }

wait to send next string while (!myUSART0.ready2Send());

if (myUSART0.ready2Send()) { myUSART0.sendString("-----\r\n"); }

wait to send next string while (!myUSART0.ready2Send());

if (myUSART0.ready2Send()) { myUSART0.sendString("Press any key to start.\r\n"); }

Mainloop while (1) {

myUSART0.receiveFrame(reinterpret_cast<uint8_t*>(&I_receiverBuffer),BUFFER_SIZE);

wait for a key press while (!myUSART0.getNumberBytesReceived()) {

}

reset number of bytes after extracting the received data myUSART0.resetNumberBytesReceived();

wait to send next string while (!myUSART0.ready2Send());

if (myUSART0.ready2Send()) { myUSART0.sendString("\r\nGet ready..."); }

randomDelay();

wait to send next string while (!myUSART0.ready2Send());

if (myUSART0.ready2Send()) { myUSART0.sendString("\r\nGo!\r\n"); }

Led.on();

myCounter.setCounter(core::timerCounter::timerCounter1,0);

if (PushButton.isPressed()) { wait to send next string while (!myUSART0.ready2Send()); }

if (myUSART0.ready2Send()) { myUSART0.sendString("You're only cheating yourself.\r\n"); }

} else {

wait for a button press while (!PushButton.isPressed()) {

} get counter value myCounter.getCounter(core::timerCounter::timerCounter1, &timerValue); bit shift divide by 16 =
2^4 to convert from micro to milliseconds timerValue = timerValue >> 4; print response time printWord(timerValue);
}

Led.off();

while (!myUSART0.ready2Send());

if (myUSART0.ready2Send()) { myUSART0.sendString("\r\nPress any key to try again.\r\n"); }

} return 0; }

void randomDelay(void) {

Waits for a "random" delay from 1 - 3.5 sec Requires timer 1 initialized and running It's not really random, but very
hard to control –like coin-flipping.

```

```
uint16_t counter;
_delay_ms(1000);

myCounter.getCounter(core::timerCounter::timerCounter1, &counter);
uint8_t randomTime = static_cast<uint8_t>(counter);
```

type-casting the 16-bit TCNT1 as an 8-bit number keeps only the 8 least-significant (fastest-changing) bits

```
while (--randomTime) {
    _delay_ms(10);
}

void printWord(uint16_t word) {

    while (!myUSART0.ready2Send()){};

    if (myUSART0.ready2Send())
    {
        myUSART0.sendByte('0' + (word / 10000));

    }

    while (!myUSART0.ready2Send()){};

    if (myUSART0.ready2Send())
    {
        myUSART0.sendByte('0' + ((word / 1000) % 10));
    }

    while (!myUSART0.ready2Send()){};

    if (myUSART0.ready2Send())
    {
        myUSART0.sendByte('0' + ((word / 100) % 10));
    }

    while (!myUSART0.ready2Send()){};

    if (myUSART0.ready2Send())
    {
        myUSART0.sendByte('0' + ((word / 10) % 10));
    }

    while (!myUSART0.ready2Send()){};

    if (myUSART0.ready2Send())
    {
        myUSART0.sendByte('0' + (word % 10));
    }

}

void printWord(uint16_t word) {

    char timerValue = '0' + (word / 10000);

    while (!myUSART0.ready2Send()){};

    if (myUSART0.ready2Send())
    {
        myUSART0.sendFrame(reinterpret_cast<uint8_t*>(&timerValue), BUFFER_SIZE);
    }

    timerValue = '0' + ((word / 1000) % 10);

    while (!myUSART0.ready2Send()){};
```

```

if (myUSART0.ready2Send())
{
    myUSART0.sendFrame(reinterpret_cast<uint8_t*>(&timerValue), BUFFER_SIZE);
}

timerValue = '0' + ((word / 100) % 10);

while (!myUSART0.ready2Send()){};

if (myUSART0.ready2Send())
{
    myUSART0.sendFrame(reinterpret_cast<uint8_t*>(&timerValue), BUFFER_SIZE);
}

timerValue = '0' + ((word / 10) % 10);

while (!myUSART0.ready2Send()){};

if (myUSART0.ready2Send())
{
    myUSART0.sendFrame(reinterpret_cast<uint8_t*>(&timerValue), BUFFER_SIZE);
}

timerValue = '0' + (word % 10);

while (!myUSART0.ready2Send()){};

if (myUSART0.ready2Send())
{
    myUSART0.sendFrame(reinterpret_cast<uint8_t*>(&timerValue), BUFFER_SIZE);
}

}

```

Usage example (Reaction Timer):

#### Author

Farid Oubbati ( <https://github.com/faroub>)

#### Date

March 2020

Definition in file [TimerCounter.h](#).

## 10.81 TimerCounter.h

```

00001
00039
00461 #ifndef TIMER_COUNTER_H
00462 #define TIMER_COUNTER_H
00463 #include "ha_base.h"
00464
00465
00466 namespace core
00467 {
00468
00469
00470 enum class channel : uint8_t {
00471     A=0,

```

```

00472     B,
00473 };
00474
00475
00476 enum class compareOutputMode : uint8_t {
00477     normal=0,
00478     toggle,
00479     clear,
00480     set,
00481 };
00482
00483
00484
00485 enum class operationMode : uint8_t {
00486     normal=0,
00487     PWM_PC,
00488     PWM_PC_8bit,
00489     PWM_PC_9bit,
00490     PWM_PC_10bit,
00491     PWM_PFC_ICR,
00492     PWM_PFC_OCR,
00493     PWM_PC_ICR,
00494     PWM_PC_OCR,
00495     fast_PWM,
00496     fast_PWM_8bit,
00497     fast_PWM_9bit,
00498     fast_PWM_10bit,
00499     fast_PWM_ICR,
00500     fast_PWM_OCR,
00501     CTC_OCR,
00502     CTC_ICR,
00503 };
00504 enum class clockSource : uint16_t {
00505     noClock=0,
00506     PS_1,
00507     PS_8,
00508     PS_32,
00509     PS_64,
00510     PS_128,
00511     PS_256,
00512     PS_1024,
00513     extern_Clock_T0_Falling_Edge,
00514     extern_Clock_T0_Rising_Edge,
00515 };
00516
00517 class TimerCounter
00518 {
00519
00520 public:
00521
00522
00523     virtual void selectOperationMode(const operationMode &ar_operationMode) = 0;
00524
00525     virtual void start() = 0;
00526
00527     virtual void stop() = 0;
00528
00529     virtual void selectClockSource(const clockSource &ar_clockSource) = 0;
00530
00531     virtual void selectCompareOutputMode(const channel &ar_channel, const compareOutputMode
&ar_compareOutputMode) = 0;
00532
00533     virtual void setCounter(const uint16_t &ar_dataBuffer) = 0;
00534
00535     virtual uint16_t getCounter() const = 0;
00536
00537     virtual void setOutputCompareRegister(const channel &ar_channel, const uint16_t &ar_dataBuffer) =
0;
00538
00539     virtual uint16_t getOutputCompareRegister(const channel &ar_channel) const = 0;
00540
00541     virtual void enableOutputCompareMatchInterrupt(const channel &ar_channel, const uint8_t a_enable)
= 0;
00542
00543     virtual void enableOverflowInterrupt(const uint8_t a_enable) = 0;
00544
00545     virtual uint16_t getClockPrescaler() = 0;
00546
00547 protected:
00548
00549 private:
00550
00551
00552
00553
00554 };
00555

```

```

00556 }
00557
00558
00559 #endif

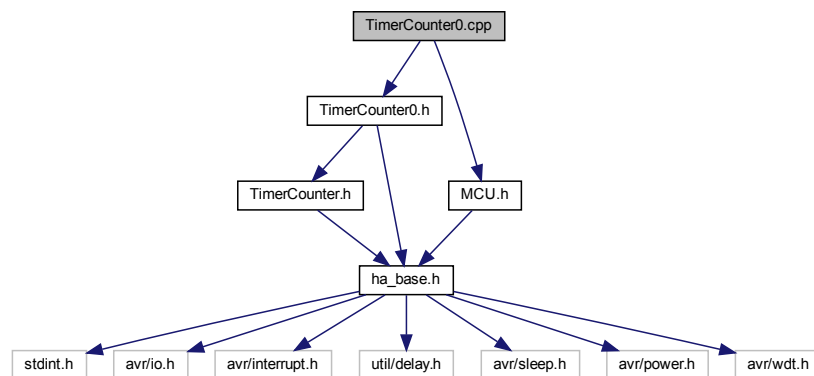
```

## 10.82 TimerCounter0.cpp File Reference

```
#include "TimerCounter0.h"
```

```
#include "MCU.h"
```

Include dependency graph for TimerCounter0.cpp:



## 10.83 TimerCounter0.cpp

```

00001 #include "TimerCounter0.h"
00002 #include "MCU.h"
00003
00004
00005 core::TimerCounter0& core::TimerCounter0::getInstance(const channel &ar_channel,
00006                                                         const operationMode &ar_operationMode,
00007                                                         const clockSource &ar_clockSource,
00008                                                         const compareOutputMode &ar_compareOutputMode)
00009 {
00010     static TimerCounter0 l_instance(ar_channel,
00011                                     ar_operationMode,
00012                                     ar_clockSource,
00013                                     ar_compareOutputMode);
00014
00015     return l_instance;
00016 }
00017
00018 }
00019
00020 core::TimerCounter0::TimerCounter0(const channel &ar_channel,
00021                                     const operationMode &ar_operationMode,
00022                                     const clockSource &ar_clockSource,
00023                                     const compareOutputMode& ar_compareOutputMode)
00024 {
00025     core::MCU::enableTimerCounter0(1);
00026     stop();
00027     selectOperationMode(ar_operationMode);
00028     selectClockSource(ar_clockSource);
00029     selectCompareOutputMode(ar_channel, ar_compareOutputMode);
00030 }
00031
00032
00033 }
00034 core::TimerCounter0::~TimerCounter0()
00035 {
00036 }
00037 }

```



```
00038
00039 void core::TimerCounter0::selectClockSource(const clockSource &ar_clockSource)
00040 {
00041     switch (ar_clockSource)
00042     {
00043         case core::clockSource::noClock:
00044         {
00045             m_clockPrescaler=0;
00046             m_clockSource=0;
00047             break;
00048         }
00049         case core::clockSource::PS_1:
00050         {
00051             m_clockPrescaler=1;
00052             m_clockSource=1;
00053             break;
00054         }
00055         case core::clockSource::PS_8:
00056         {
00057             m_clockPrescaler=8;
00058             m_clockSource=2;
00059             break;
00060         }
00061         case core::clockSource::PS_64:
00062         {
00063             m_clockPrescaler=64;
00064             m_clockSource=3;
00065             break;
00066         }
00067         case core::clockSource::PS_256:
00068         {
00069             m_clockPrescaler=256;
00070             m_clockSource=4;
00071             break;
00072         }
00073         case core::clockSource::PS_1024:
00074         {
00075             m_clockPrescaler=1024;
00076             m_clockSource=5;
00077             break;
00078         }
00079         case core::clockSource::extern_Clock_T0_Falling_Edge:
00080         {
00081             m_clockPrescaler=0;
00082             m_clockSource=6;
00083             break;
00084         }
00085         case core::clockSource::extern_Clock_T0_Rising_Edge:
00086         {
00087             m_clockPrescaler=0;
00088             m_clockSource=7;
00089             break;
00090         }
00091     }
00092 }
00093
00094 }
00095
00096 void core::TimerCounter0::selectOperationMode(const operationMode &ar_operationMode)
00097 {
00098     switch (ar_operationMode)
00099     {
00100         case core::operationMode::normal:
00101         {
00102             TIMER0_SELECT_OPERATION_MODE(0);
00103             break;
00104         }
00105         case core::operationMode::PWM_PC:
00106         {
00107             TIMER0_SELECT_OPERATION_MODE(1);
00108             break;
00109         }
00110         case core::operationMode::CTC_OCR:
00111         {
00112             TIMER0_SELECT_OPERATION_MODE(2);
00113             break;
00114         }
00115         case core::operationMode::fast_PWM:
00116         {
00117             TIMER0_SELECT_OPERATION_MODE(3);
00118             break;
00119         }
00120         case core::operationMode::PWM_PC_OCR:
00121         {
00122             TIMER0_SELECT_OPERATION_MODE(5);
00123             break;
00124         }
00125     }
00126 }
```

```

00125         case core::operationMode::fast_PWM_OCR:
00126         {
00127             TIMER0_SELECT_OPERATION_MODE(7);
00128             break;
00129         }
00130     }
00131 }
00132
00133 }
00134
00135
00136 void core::TimerCounter0::start()
00137 {
00138     TIMER0_SELECT_CLOCK_SOURCE(m_clockSource);
00139 }
00140
00141 uint16_t core::TimerCounter0::getClockPrescaler()
00142 {
00143     return m_clockPrescaler;
00144 }
00145
00146 void core::TimerCounter0::stop()
00147 {
00148     TIMER0_STOP;
00149 }
00150
00151
00152 void core::TimerCounter0::selectCompareOutputMode(const channel &ar_channel, const compareOutputMode
&ar_compareOutputMode)
00153 {
00154     switch (ar_channel)
00155     {
00156         case core::channel::A:
00157         {
00158             TIMER0_SELECT_COM_CHANNEL_A(static_cast<uint8_t>(ar_compareOutputMode));
00159             break;
00160         }
00161         case core::channel::B:
00162         {
00163             TIMER0_SELECT_COM_CHANNEL_B(static_cast<uint8_t>(ar_compareOutputMode));
00164             break;
00165         }
00166     }
00167 }
00168
00169
00170
00171 void core::TimerCounter0::setCounter(const uint16_t &ar_dataBuffer)
00172 {
00173     TCNT0 = static_cast<uint8_t>(ar_dataBuffer);
00174 }
00175
00176 uint16_t core::TimerCounter0::getCounter() const
00177 {
00178     return TCNT0;
00179 }
00180
00181 void core::TimerCounter0::setOutputCompareRegister(const channel &ar_channel, const uint16_t
&ar_dataBuffer)
00182 {
00183     switch (ar_channel)
00184     {
00185         case core::channel::A:
00186         {
00187             OCR0A = static_cast<uint8_t>(ar_dataBuffer);
00188             break;
00189         }
00190         case core::channel::B:
00191         {
00192             OCR0B = static_cast<uint8_t>(ar_dataBuffer);
00193             break;
00194         }
00195     }
00196 }
00197 }
00198
00199
00200
00201 uint16_t core::TimerCounter0::getOutputCompareRegister(const channel &ar_channel) const
00202 {
00203     switch (ar_channel)
00204     {
00205         case core::channel::A:
00206         {
00207             return OCR0A;
00208         }
00209         case core::channel::B:

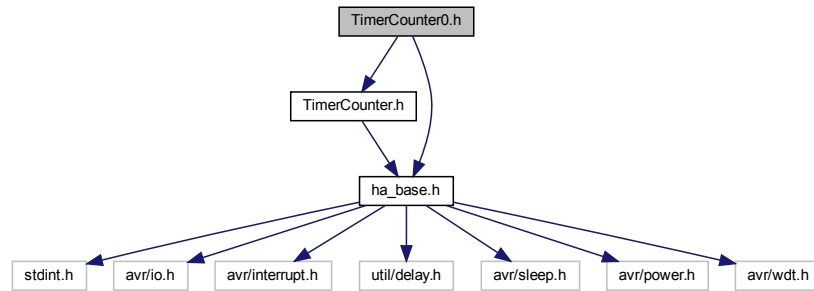
```

```
00210     {
00211         return OCR0B;
00212     }
00213 }
00214 }
00215
00216 void core::TimerCounter0::enableOutputCompareMatchInterrupt(const channel &ar_channel, const uint8_t
a_enable)
00217 {
00218     switch (ar_channel)
00219     {
00220         case core::channel::A:
00221         {
00222             if (a_enable) {
00223
00224                 TIMER0_ENABLE_COM_CHANNEL_A_INTERRUPT;
00225
00226             } else {
00227
00228                 TIMER0_DISABLE_COM_CHANNEL_A_INTERRUPT;
00229             }
00230             break;
00231         }
00232         case core::channel::B:
00233         {
00234             if (a_enable) {
00235
00236                 TIMER0_ENABLE_COM_CHANNEL_B_INTERRUPT;
00237
00238             } else {
00239
00240                 TIMER0_DISABLE_COM_CHANNEL_B_INTERRUPT;
00241             }
00242             break;
00243         }
00244     }
00245 }
00246
00247 }
00248
00249
00250
00251 void core::TimerCounter0::enableOverflowInterrupt(const uint8_t a_enable)
00252 {
00253
00254     if (a_enable) {
00255
00256         TIMER0_ENABLE_OVERFLOW_INTERRUPT;
00257
00258     } else {
00259
00260         TIMER0_DISABLE_OVERFLOW_INTERRUPT;
00261     }
00262 }
00263 }
```

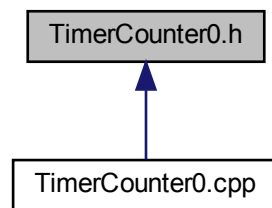
## 10.84 TimerCounter0.h File Reference

```
#include "TimerCounter.h"
#include "ha_base.h"
```

Include dependency graph for TimerCounter0.h:



This graph shows which files directly or indirectly include this file:



## Classes

- class `core::TimerCounter0`

## Namespaces

- `core`

*AVR chip internal core components.*

## 10.85 TimerCounter0.h

```

00001
00039
00268 #ifndef TIMER_COUNTER0_H
00269 #define TIMER_COUNTER0_H
00270 #include "TimerCounter.h"
00271 #include "ha_base.h"
00272
00273
00274
00275 namespace core
00276 {
00277

```

```

00278
00279 class TimerCounter0 : public TimerCounter
00280 {
00281 public:
00282     static TimerCounter0& getInstance(const channel &ar_channel = channel::A,
00283                                     const operationMode &ar_operationMode = operationMode::normal,
00284                                     const clockSource &ar_clockSource= clockSource::noClock,
00285                                     const compareOutputMode& ar_compareOutputMode =
00286                                     compareOutputMode::normal);
00287
00288     void selectOperationMode(const operationMode &ar_operationMode) override;
00289     void start() override;
00290     void stop() override;
00291     void selectClockSource(const clockSource &ar_clockSource) override;
00292     void selectCompareOutputMode(const channel &ar_channel, const compareOutputMode
00293                                &ar_compareOutputMode) override;
00294     void setCounter(const uint16_t &ar_dataBuffer) override;
00295     uint16_t getCounter() const override;
00296     void setOutputCompareRegister(const channel &ar_channel, const uint16_t &ar_dataBuffer) override;
00297     uint16_t getOutputCompareRegister(const channel &ar_channel) const override;
00298     void enableOutputCompareMatchInterrupt(const channel &ar_channel, const uint8_t a_enable)
00299     override;
00300     void enableOverflowInterrupt(const uint8_t a_enable) override;
00301     uint16_t getClockPrescaler() override;
00302
00303     static void outputCompareMatchAServiceRoutine() __asm__(STR(TIMERO_COM_CHANNEL_A_INTERRUPT))
00304     __attribute__((__signal__, __used__, __externally_visible__));
00305     static void outputCompareMatchBServiceRoutine() __asm__(STR(TIMERO_COM_CHANNEL_B_INTERRUPT))
00306     __attribute__((__signal__, __used__, __externally_visible__));
00307     static void overflowServiceRoutine() __asm__(STR(TIMERO_OVERFLOW_INTERRUPT))
00308     __attribute__((__signal__, __used__, __externally_visible__));
00309
00310 protected:
00311 private:
00312     TimerCounter0(const channel &ar_channel,
00313                  const operationMode &ar_operationMode,
00314                  const clockSource &ar_clockSource,
00315                  const compareOutputMode& ar_compareOutputMode);
00316     ~TimerCounter0();
00317     TimerCounter0(const TimerCounter0&);
00318     const TimerCounter0& operator=(const TimerCounter0&);
00319     uint16_t m_clockPrescaler;
00320     uint8_t m_clockSource;
00321
00322 };
00323
00324 #endif

```

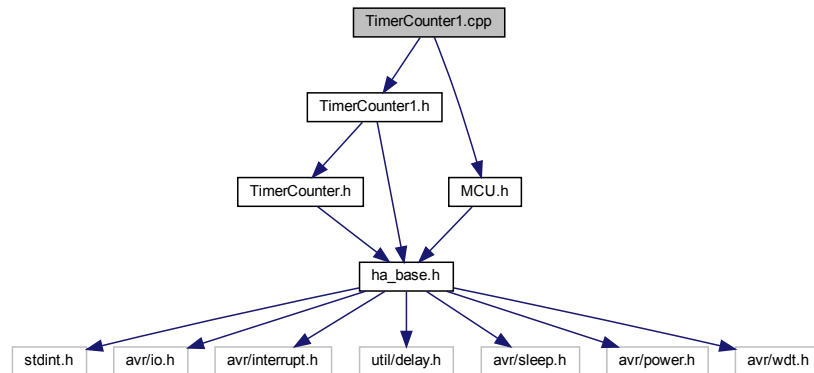
## 10.86 TimerCounter1.cpp File Reference

```

#include "TimerCounter1.h"
#include "MCU.h"

```

Include dependency graph for TimerCounter1.cpp:



## 10.87 TimerCounter1.cpp

```

00001 #include "TimerCounter1.h"
00002 #include "MCU.h"
00003
00004
00005 core::TimerCounter1& core::TimerCounter1::getInstance(const channel &ar_channel,
00006                                                         const operationMode &ar_operationMode,
00007                                                         const clockSource &ar_clockSource,
00008                                                         const compareOutputMode& ar_compareOutputMode)
00009 {
00010     static TimerCounter1 l_instance(ar_channel,
00011                                     ar_operationMode,
00012                                     ar_clockSource,
00013                                     ar_compareOutputMode);
00014
00015     return l_instance;
00016 }
00017
00018 }
00019
00020 core::TimerCounter1::TimerCounter1(const channel &ar_channel,
00021                                     const operationMode &ar_operationMode,
00022                                     const clockSource &ar_clockSource,
00023                                     const compareOutputMode& ar_compareOutputMode)
00024 {
00025     core::MCU::enableTimerCounter1(1);
00026     stop();
00027     selectOperationMode(ar_operationMode);
00028     selectClockSource(ar_clockSource);
00029     selectCompareOutputMode(ar_channel, ar_compareOutputMode);
00030 }
00031
00032 }
00033 core::TimerCounter1::~TimerCounter1()
00034 {
00035 }
00036 }
00037
00038 void core::TimerCounter1::selectClockSource(const clockSource &ar_clockSource)
00039 {
00040     switch (ar_clockSource)
00041     {
00042     case core::clockSource::noClock:
00043     {
00044         m_clockPrescaler=0;
00045         m_clockSource=0;
00046         break;
00047     }
00048     case core::clockSource::PS_1:
00049     {
00050         m_clockPrescaler=1;
00051         m_clockSource=1;
00052         break;
00053     }
  
```

```

00054         case core::clockSource::PS_8:
00055         {
00056             m_clockPrescaler=8;
00057             m_clockSource=2;
00058             break;
00059         }
00060         case core::clockSource::PS_64:
00061         {
00062             m_clockPrescaler=64;
00063             m_clockSource=3;
00064             break;
00065         }
00066         case core::clockSource::PS_256:
00067         {
00068             m_clockPrescaler=256;
00069             m_clockSource=4;
00070             break;
00071         }
00072         case core::clockSource::PS_1024:
00073         {
00074             m_clockPrescaler=1024;
00075             m_clockSource=5;
00076             break;
00077         }
00078         case core::clockSource::extern_Clock_T0_Falling_Edge:
00079         {
00080             m_clockPrescaler=0;
00081             m_clockSource=6;
00082             break;
00083         }
00084         case core::clockSource::extern_Clock_T0_Rising_Edge:
00085         {
00086             m_clockPrescaler=0;
00087             m_clockSource=7;
00088             break;
00089         }
00090     }
00091 }
00092
00093 }
00094
00095 void core::TimerCounter1::selectOperationMode(const operationMode &ar_operationMode)
00096 {
00097     switch (ar_operationMode)
00098     {
00099         case core::operationMode::normal:
00100         {
00101             TIMER1_SELECT_OPERATION_MODE(0);
00102             break;
00103         }
00104         case core::operationMode::PWM_PC_8bit:
00105         {
00106             TIMER1_SELECT_OPERATION_MODE(1);
00107             break;
00108         }
00109         case core::operationMode::PWM_PC_9bit:
00110         {
00111             TIMER1_SELECT_OPERATION_MODE(2);
00112             break;
00113         }
00114         case core::operationMode::PWM_PC_10bit:
00115         {
00116             TIMER1_SELECT_OPERATION_MODE(3);
00117             break;
00118         }
00119         case core::operationMode::CTC_OCR:
00120         {
00121             TIMER1_SELECT_OPERATION_MODE(4);
00122             break;
00123         }
00124         case core::operationMode::fast_PWM_8bit:
00125         {
00126             TIMER1_SELECT_OPERATION_MODE(5);
00127             break;
00128         }
00129         case core::operationMode::fast_PWM_9bit:
00130         {
00131             TIMER1_SELECT_OPERATION_MODE(6);
00132             break;
00133         }
00134         case core::operationMode::fast_PWM_10bit:
00135         {
00136             TIMER1_SELECT_OPERATION_MODE(7);
00137             break;
00138         }
00139         case core::operationMode::PWM_PFC_ICR:
00140         {

```

```

00141         TIMER1_SELECT_OPERATION_MODE(8);
00142         break;
00143     }
00144     case core::operationMode::PWM_PFC_OCR:
00145     {
00146         TIMER1_SELECT_OPERATION_MODE(9);
00147         break;
00148     }
00149     case core::operationMode::PWM_PC_ICR:
00150     {
00151         TIMER1_SELECT_OPERATION_MODE(10);
00152         break;
00153     }
00154     case core::operationMode::PWM_PC_OCR:
00155     {
00156         TIMER1_SELECT_OPERATION_MODE(11);
00157         break;
00158     }
00159     case core::operationMode::CTC_ICR:
00160     {
00161         TIMER1_SELECT_OPERATION_MODE(12);
00162         break;
00163     }
00164     case core::operationMode::fast_PWM_ICR:
00165     {
00166         TIMER1_SELECT_OPERATION_MODE(14);
00167         break;
00168     }
00169     case core::operationMode::fast_PWM_OCR:
00170     {
00171         TIMER1_SELECT_OPERATION_MODE(15);
00172         break;
00173     }
00174 }
00175 }
00176
00177
00178 }
00179
00180
00181 void core::TimerCounter1::start()
00182 {
00183     TIMER1_SELECT_CLOCK_SOURCE(m_clockSource);
00184 }
00185
00186 uint16_t core::TimerCounter1::getClockPrescaler()
00187 {
00188     return m_clockPrescaler;
00189 }
00190
00191 void core::TimerCounter1::stop()
00192 {
00193     TIMER1_STOP;
00194 }
00195
00196
00197 void core::TimerCounter1::selectCompareOutputMode(const channel &ar_channel, const compareOutputMode
&ar_compareOutputMode)
00198 {
00199     switch (ar_channel)
00200     {
00201         case core::channel::A:
00202         {
00203             TIMER1_SELECT_COM_CHANNEL_A(static_cast<uint8_t>(ar_compareOutputMode));
00204             break;
00205         }
00206         case core::channel::B:
00207         {
00208             TIMER1_SELECT_COM_CHANNEL_B(static_cast<uint8_t>(ar_compareOutputMode));
00209             break;
00210         }
00211     }
00212 }
00213 }
00214
00215
00216 void core::TimerCounter1::setCounter(const uint16_t &ar_dataBuffer)
00217 {
00218     TCNT1 = ar_dataBuffer;
00219 }
00220
00221 uint16_t core::TimerCounter1::getCounter() const
00222 {
00223     return TCNT1;
00224 }
00225
00226 void core::TimerCounter1::setOutputCompareRegister(const channel &ar_channel, const uint16_t

```



```

        &ar_dataBuffer)
00227 {
00228     switch (ar_channel)
00229     {
00230         case core::channel::A:
00231         {
00232             OCR1A = ar_dataBuffer;
00233             break;
00234         }
00235         case core::channel::B:
00236         {
00237             OCR1B = ar_dataBuffer;
00238             break;
00239         }
00240     }
00241 }
00242
00243
00244
00245 uint16_t core::TimerCounter1::getOutputCompareRegister(const channel &ar_channel) const
00246 {
00247     switch (ar_channel)
00248     {
00249         case core::channel::A:
00250         {
00251             return OCR1A;
00252         }
00253         case core::channel::B:
00254         {
00255             return OCR1B;
00256         }
00257     }
00258 }
00259
00260 void core::TimerCounter1::enableOutputCompareMatchInterrupt(const channel &ar_channel, const uint8_t
a_enable)
00261 {
00262     switch (ar_channel)
00263     {
00264         case core::channel::A:
00265         {
00266             if (a_enable) {
00267
00268                 TIMER1_ENABLE_COM_CHANNEL_A_INTERRUPT;
00269
00270             } else {
00271
00272                 TIMER1_DISABLE_COM_CHANNEL_A_INTERRUPT;
00273             }
00274             break;
00275         }
00276         case core::channel::B:
00277         {
00278             if (a_enable) {
00279
00280                 TIMER1_ENABLE_COM_CHANNEL_B_INTERRUPT;
00281
00282             } else {
00283
00284                 TIMER1_DISABLE_COM_CHANNEL_B_INTERRUPT;
00285             }
00286             break;
00287         }
00288     }
00289 }
00290
00291 }
00292
00293
00294
00295 void core::TimerCounter1::enableOverflowInterrupt(const uint8_t a_enable)
00296 {
00297     if (a_enable) {
00298
00299         TIMER1_ENABLE_OVERFLOW_INTERRUPT;
00300
00301     } else {
00302
00303         TIMER1_DISABLE_OVERFLOW_INTERRUPT;
00304     }
00305 }
00306
00307 }
00308
00309 void core::TimerCounter1::enableInputCaptureInterrupt(const uint8_t a_enable)
00310 {
00311     if (a_enable) {

```

```

00312
00313         TIMER1_ENABLE_INPUT_CAPTURE_INTERRUPT;
00314
00315     } else {
00316
00317         TIMER1_DISABLE_INPUT_CAPTURE_INTERRUPT;
00318     }
00319 }
00320
00321 void core::TimerCounter1::setInputCaptureRegister(const uint16_t &ar_dataBuffer)
00322 {
00323     ICR1 = ar_dataBuffer;
00324 }
00325
00326
00327 uint16_t core::TimerCounter1::getInputCaptureRegister() const
00328 {
00329     return ICR1;
00330 }

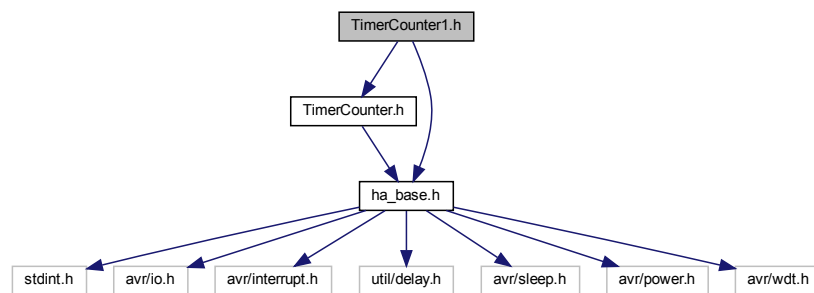
```

## 10.88 TimerCounter1.h File Reference

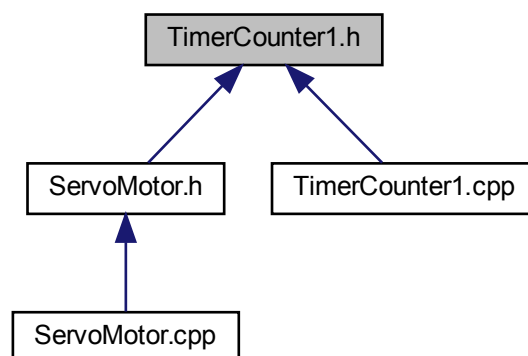
```
#include "TimerCounter.h"
```

```
#include "ha_base.h"
```

Include dependency graph for TimerCounter1.h:



This graph shows which files directly or indirectly include this file:



## Classes

- class `core::TimerCounter1`

## Namespaces

- `core`

*AVR chip internal core components.*

## 10.89 TimerCounter1.h

```

00001
00039
00268 #ifndef TIMER_COUNTER1_H
00269 #define TIMER_COUNTER1_H
00270 #include "TimerCounter.h"
00271 #include "ha_base.h"
00272
00273
00274
00275 namespace core
00276 {
00277
00278
00279 class TimerCounter1 : public TimerCounter
00280 {
00281
00282 public:
00283
00284     static TimerCounter1& getInstance(const channel &ar_channel = channel::A,
00285                                     const operationMode &ar_operationMode = operationMode::normal,
00286                                     const clockSource &ar_clockSource= clockSource::noClock,
00287                                     const compareOutputMode& ar_compareOutputMode =
compareOutputMode::normal);
00288
00289     void selectOperationMode(const operationMode &ar_operationMode) override;
00290
00291     void start() override;
00292
00293     void stop() override;
00294
00295     void selectClockSource(const clockSource &ar_clockSource) override;
00296
00297
00298     void selectCompareOutputMode(const channel &ar_channel, const compareOutputMode
&ar_compareOutputMode) override;
00299
00300     void setCounter(const uint16_t &ar_dataBuffer) override;
00301
00302     uint16_t getCounter() const override;
00303
00304     void setOutputCompareRegister(const channel &ar_channel, const uint16_t &ar_dataBuffer) override;
00305
00306     uint16_t getOutputCompareRegister(const channel &ar_channel) const override;
00307
00308     void setInputCaptureRegister(const uint16_t &ar_dataBuffer);
00309
00310     uint16_t getInputCaptureRegister() const;
00311
00312     void enableOutputCompareMatchInterrupt(const channel &ar_channel, const uint8_t a_enable)
override;
00313
00314     void enableOverflowInterrupt(const uint8_t a_enable) override;
00315
00316     void enableInputCaptureInterrupt(const uint8_t a_enable);
00317
00318     uint16_t getClockPrescaler() override;
00319
00320     static void outputCompareMatchAServiceRoutine() __asm__(STR(TIMER1_COM_CHANNEL_A_INTERRUPT))
__attribute__((__signal__, __used__, __externally_visible__));
00321
00322     static void outputCompareMatchBServiceRoutine() __asm__(STR(TIMER1_COM_CHANNEL_B_INTERRUPT))
__attribute__((__signal__, __used__, __externally_visible__));
00323
00324     static void overflowServiceRoutine() __asm__(STR(TIMER1_OVERFLOW_INTERRUPT))
__attribute__((__signal__, __used__, __externally_visible__));
00325

```

```

00326     static void inputCaptureServiceRoutine() __asm__(STR(TIMER1_INPUT_CAPTURE_INTERRUPT))
    __attribute__((__signal__, __used__, __externally_visible__));
00327
00328
00329
00330 protected:
00331
00332 private:
00333
00334     TimerCounter1(const channel &ar_channel,
00335                  const operationMode &ar_operationMode,
00336                  const clockSource &ar_clockSource,
00337                  const compareOutputMode& ar_compareOutputMode);
00338
00339     ~TimerCounter1();
00340
00341     TimerCounter1(const TimerCounter1&);
00342
00343     const TimerCounter1& operator=(const TimerCounter1&);
00344
00345     uint16_t m_clockPrescaler;
00346
00347     uint8_t m_clockSource;
00348
00349 };
00350
00351 #endif

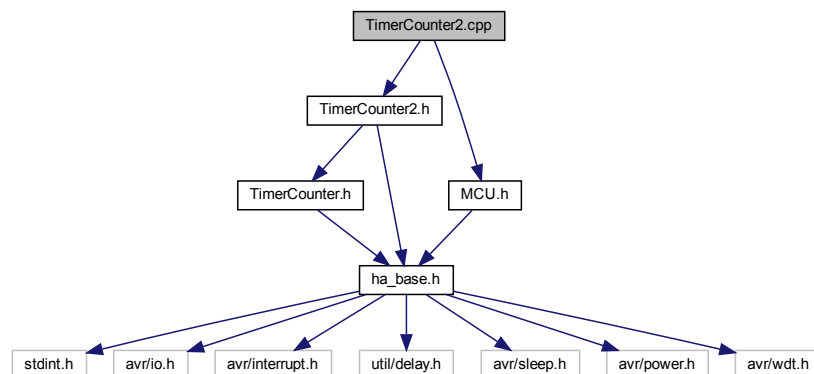
```

## 10.90 TimerCounter2.cpp File Reference

```
#include "TimerCounter2.h"
```

```
#include "MCU.h"
```

Include dependency graph for TimerCounter2.cpp:



## 10.91 TimerCounter2.cpp

```

00001 #include "TimerCounter2.h"
00002 #include "MCU.h"
00003
00004
00005 core::TimerCounter2& core::TimerCounter2::getInstance(const channel &ar_channel,
00006                                                         const operationMode &ar_operationMode,
00007                                                         const clockSource &ar_clockSource,
00008                                                         const compareOutputMode& ar_compareOutputMode)
00009 {
00010     static TimerCounter2 l_instance(ar_channel,
00011                                     ar_operationMode,

```

```
00012             ar_clockSource,
00013             ar_compareOutputMode);
00014
00015     return l_instance;
00016
00017
00018 }
00019
00020 core::TimerCounter2::TimerCounter2(const channel &ar_channel,
00021                                     const operationMode &ar_operationMode,
00022                                     const clockSource &ar_clockSource,
00023                                     const compareOutputMode& ar_compareOutputMode)
00024 {
00025     core::MCU::enableTimerCounter2(1);
00026     stop();
00027     selectOperationMode(ar_operationMode);
00028     selectClockSource(ar_clockSource);
00029     selectCompareOutputMode(ar_channel, ar_compareOutputMode);
00030 }
00031
00032 core::TimerCounter2::~TimerCounter2()
00033 {
00034 }
00035
00036
00037 void core::TimerCounter2::selectClockSource(const clockSource &ar_clockSource)
00038 {
00039     switch (ar_clockSource)
00040     {
00041     case core::clockSource::noClock:
00042     {
00043         m_clockPrescaler=0;
00044         m_clockSource=0;
00045         break;
00046     }
00047     case core::clockSource::PS_1:
00048     {
00049         m_clockPrescaler=1;
00050         m_clockSource=1;
00051         break;
00052     }
00053     case core::clockSource::PS_8:
00054     {
00055         m_clockPrescaler=8;
00056         m_clockSource=2;
00057         break;
00058     }
00059     case core::clockSource::PS_32:
00060     {
00061         m_clockPrescaler=32;
00062         m_clockSource=3;
00063         break;
00064     }
00065     case core::clockSource::PS_64:
00066     {
00067         m_clockPrescaler=64;
00068         m_clockSource=4;
00069         break;
00070     }
00071     case core::clockSource::PS_128:
00072     {
00073         m_clockPrescaler=128;
00074         m_clockSource=5;
00075         break;
00076     }
00077     case core::clockSource::PS_256:
00078     {
00079         m_clockPrescaler=256;
00080         m_clockSource=6;
00081         break;
00082     }
00083     case core::clockSource::PS_1024:
00084     {
00085         m_clockPrescaler=1024;
00086         m_clockSource=7;
00087         break;
00088     }
00089     }
00090 }
00091
00092
00093 void core::TimerCounter2::selectOperationMode(const operationMode &ar_operationMode)
00094 {
00095     switch (ar_operationMode)
00096     {
00097     case core::operationMode::normal:
00098     {
```

```

00099         TIMER2_SELECT_OPERATION_MODE(0);
00100         break;
00101     }
00102     case core::operationMode::PWM_PC:
00103     {
00104         TIMER2_SELECT_OPERATION_MODE(1);
00105         break;
00106     }
00107     case core::operationMode::CTC_OCR:
00108     {
00109         TIMER2_SELECT_OPERATION_MODE(2);
00110         break;
00111     }
00112     case core::operationMode::fast_PWM:
00113     {
00114         TIMER2_SELECT_OPERATION_MODE(3);
00115         break;
00116     }
00117     case core::operationMode::PWM_PC_OCR:
00118     {
00119         TIMER2_SELECT_OPERATION_MODE(5);
00120         break;
00121     }
00122     case core::operationMode::fast_PWM_OCR:
00123     {
00124         TIMER2_SELECT_OPERATION_MODE(7);
00125         break;
00126     }
00127 }
00128 }
00129
00130 }
00131
00132
00133 void core::TimerCounter2::start()
00134 {
00135     TIMER2_SELECT_CLOCK_SOURCE(m_clockSource);
00136 }
00137
00138 uint16_t core::TimerCounter2::getClockPrescaler()
00139 {
00140     return m_clockPrescaler;
00141 }
00142
00143
00144 void core::TimerCounter2::stop()
00145 {
00146     TIMER2_STOP;
00147 }
00148
00149
00150 void core::TimerCounter2::selectCompareOutputMode(const channel &ar_channel, const compareOutputMode
&ar_compareOutputMode)
00151 {
00152     switch (ar_channel)
00153     {
00154         case core::channel::A:
00155         {
00156             TIMER2_SELECT_COM_CHANNEL_A(static_cast<uint8_t>(ar_compareOutputMode));
00157             break;
00158         }
00159         case core::channel::B:
00160         {
00161             TIMER2_SELECT_COM_CHANNEL_B(static_cast<uint8_t>(ar_compareOutputMode));
00162             break;
00163         }
00164     }
00165 }
00166
00167 }
00168
00169
00170
00171 void core::TimerCounter2::setCounter(const uint16_t &ar_dataBuffer)
00172 {
00173     TCNT2 = static_cast<uint8_t>(ar_dataBuffer);
00174 }
00175
00176 uint16_t core::TimerCounter2::getCounter() const
00177 {
00178     return TCNT2;
00179 }
00180
00181 void core::TimerCounter2::setOutputCompareRegister(const channel &ar_channel, const uint16_t
&ar_dataBuffer)
00182 {
00183     switch (ar_channel)

```

```

00184     {
00185         case core::channel::A:
00186         {
00187             OCR2A = static_cast<uint8_t>(ar_dataBuffer);
00188             break;
00189         }
00190         case core::channel::B:
00191         {
00192             OCR2B = static_cast<uint8_t>(ar_dataBuffer);
00193             break;
00194         }
00195     }
00196 }
00197
00198
00199 uint16_t core::TimerCounter2::getOutputCompareRegister(const channel &ar_channel) const
00200 {
00201     switch (ar_channel)
00202     {
00203         case core::channel::A:
00204         {
00205             return OCR2A;
00206         }
00207         case core::channel::B:
00208         {
00209             return OCR2B;
00210         }
00211     }
00212 }
00213 }
00214
00215 void core::TimerCounter2::enableOutputCompareMatchInterrupt(const channel &ar_channel, const uint8_t
a_enable)
00216 {
00217     switch (ar_channel)
00218     {
00219         case core::channel::A:
00220         {
00221             if (a_enable) {
00222                 TIMER2_ENABLE_COM_CHANNEL_A_INTERRUPT;
00223             } else {
00224                 TIMER2_DISABLE_COM_CHANNEL_A_INTERRUPT;
00225             }
00226             break;
00227         }
00228         case core::channel::B:
00229         {
00230             if (a_enable) {
00231                 TIMER2_ENABLE_COM_CHANNEL_B_INTERRUPT;
00232             } else {
00233                 TIMER2_DISABLE_COM_CHANNEL_B_INTERRUPT;
00234             }
00235             break;
00236         }
00237     }
00238 }
00239
00240
00241 void core::TimerCounter2::enableOverflowInterrupt(const uint8_t a_enable)
00242 {
00243     if (a_enable) {
00244         TIMER2_ENABLE_OVERFLOW_INTERRUPT;
00245     } else {
00246         TIMER2_DISABLE_OVERFLOW_INTERRUPT;
00247     }
00248 }
00249
00250
00251
00252
00253
00254
00255
00256
00257
00258
00259
00260
00261
00262

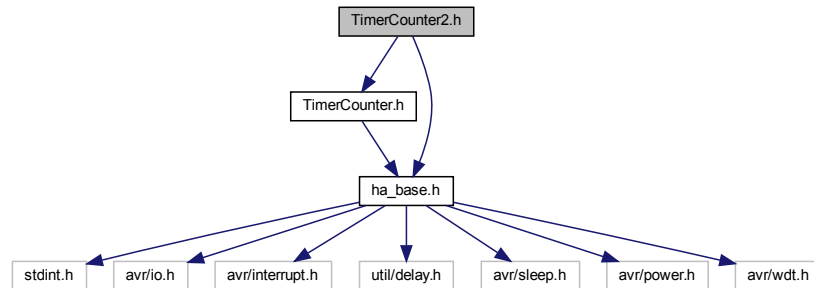
```

## 10.92 TimerCounter2.h File Reference

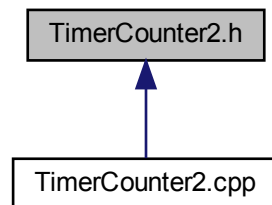
```
#include "TimerCounter.h"
```

```
#include "ha_base.h"
```

Include dependency graph for TimerCounter2.h:



This graph shows which files directly or indirectly include this file:



### Classes

- class `core::TimerCounter2`

### Namespaces

- `core`

*AVR chip internal core components.*



## 10.93 TimerCounter2.h

```

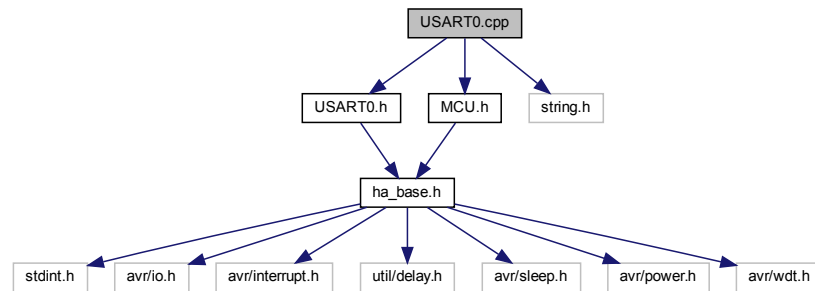
00001
00039
00268 #ifndef TIMER_COUNTER2_H
00269 #define TIMER_COUNTER2_H
00270 #include "TimerCounter.h"
00271 #include "ha_base.h"
00272
00273
00274
00275 namespace core
00276 {
00277
00278
00279 class TimerCounter2 : public TimerCounter
00280 {
00281
00282 public:
00283
00284     static TimerCounter2& getInstance(const channel &ar_channel = channel::A,
00285                                     const operationMode &ar_operationMode = operationMode::normal,
00286                                     const clockSource &ar_clockSource= clockSource::noClock,
00287                                     const compareOutputMode& ar_compareOutputMode =
compareOutputMode::normal);
00288
00289     void selectOperationMode(const operationMode &ar_operationMode) override;
00290
00291     void start() override;
00292
00293     void stop() override;
00294
00295     void selectClockSource(const clockSource &ar_clockSource) override;
00296
00297     void selectCompareOutputMode(const channel &ar_channel, const compareOutputMode
&ar_compareOutputMode) override;
00298
00299     void setCounter(const uint16_t &ar_dataBuffer) override;
00300
00301     uint16_t getCounter() const override;
00302
00303     void setOutputCompareRegister(const channel &ar_channel, const uint16_t &ar_dataBuffer) override;
00304
00305     uint16_t getOutputCompareRegister(const channel &ar_channel) const override;
00306
00307     void enableOutputCompareMatchInterrupt(const channel &ar_channel, const uint8_t a_enable)
override;
00308
00309     void enableOverflowInterrupt(const uint8_t a_enable) override;
00310
00311     uint16_t getClockPrescaler() override;
00312
00313     static void outputCompareMatchAServiceRoutine() __asm__(STR(TIMER2_COM_CHANNEL_A_INTERRUPT))
__attribute__((__signal__, __used__, __externally_visible__));
00314
00315     static void outputCompareMatchBServiceRoutine() __asm__(STR(TIMER2_COM_CHANNEL_B_INTERRUPT))
__attribute__((__signal__, __used__, __externally_visible__));
00316
00317     static void overflowServiceRoutine() __asm__(STR(TIMER2_OVERFLOW_INTERRUPT))
__attribute__((__signal__, __used__, __externally_visible__));
00318
00319 protected:
00320
00321 private:
00322
00323     TimerCounter2(const channel &ar_channel,
00324                 const operationMode &ar_operationMode,
00325                 const clockSource &ar_clockSource,
00326                 const compareOutputMode& ar_compareOutputMode);
00327
00328     ~TimerCounter2();
00329
00330     TimerCounter2(const TimerCounter2&);
00331
00332     const TimerCounter2& operator=(const TimerCounter2&);
00333
00334     uint16_t m_clockPrescaler;
00335
00336     uint8_t m_clockSource;
00337
00338 };
00339
00340
00341
00342 };
00343
00344 }
00345
00346
00347 #endif

```

## 10.94 USART0.cpp File Reference

```
#include "USART0.h"
#include "MCU.h"
#include <string.h>
```

Include dependency graph for USART0.cpp:



## 10.95 USART0.cpp

```

00001
00010 #include "USART0.h"
00011 #include "MCU.h"
00012 #include <string.h>
00013
00014 volatile uint8_t io::USART0::m_status = 0;
00015 const uint8_t* io::USART0::mp_data2Send = nullptr;
00016 uint8_t* io::USART0::mp_data2Receive = nullptr;
00017 uint16_t io::USART0::m_sizeData2Send = 0;
00018 uint16_t io::USART0::m_sizeData2Receive = 0;
00019 volatile uint16_t io::USART0::m_numberBytesReceived = 0;
00020 volatile uint16_t io::USART0::m_numberBytesSent = 0;
00021 volatile uint8_t io::USART0::m_ready2Send = 1;
00022
00023
00024
00025
00026 io::USART0& io::USART0::getInstance(const transmissionMode& ar_transMode,
00027                                     const communicationMode& ar_comMode,
00028                                     const frameSize& ar_frameSize,
00029                                     const stopBit& ar_stopBit,
00030                                     const parityMode& ar_parityMode)
00031 {
00032     static USART0 l_instance(ar_transMode,
00033                             ar_comMode,
00034                             ar_frameSize,
00035                             ar_stopBit,
00036                             ar_parityMode);
00037
00038     return l_instance;
00039 }
00040
00041
00042
00043 io::USART0::USART0(const transmissionMode& ar_transMode,
00044                   const communicationMode& ar_comMode,
00045                   const frameSize& ar_frameSize,
00046                   const stopBit& ar_stopBit,
00047                   const parityMode& ar_parityMode)
00048 {
00049     {
00050         core::MCU::enableUSART0(1);
00051         setBaudRate();
00052         setTransmissionMode(ar_transMode);
00053         setCommunicationMode(ar_comMode);
00054         setParityMode(ar_parityMode);
00055         setFrameSize(ar_frameSize);
00056         setStopBit(ar_stopBit);
00057     }
00058 }
  
```

```
00057     sei();
00058     enableReceiveCompleteInterrupt(1);
00059 }
00060
00061 io::USART0::~USART0()
00062 {
00063 }
00064 }
00065
00066 void io::USART0::setBaudRate()
00067 {
00068     USART0_SET_BAUDRATE_HIGH_REGISTER;
00069     USART0_SET_BAUDRATE_LOW_REGISTER;
00070 }
00071 }
00072
00073 void io::USART0::setTransmissionMode(const transmissionMode& ar_transMode)
00074 {
00075     switch (ar_transMode)
00076     {
00077     case transmissionMode::async:
00078     {
00079         USART0_ENABLE_ASYNC_TRANSMISSION_MODE;
00080         break;
00081     }
00082     case transmissionMode::sync:
00083     {
00084         USART0_DISABLE_DOUBLE_SPEED_MODE;
00085         USART0_ENABLE_SYNC_TRANSMISSION_MODE;
00086         break;
00087     }
00088     case transmissionMode::masterSPI:
00089     {
00090         USART0_ENABLE_MASTER_SPI_MODE;
00091         break;
00092     }
00093     }
00094 }
00095
00096 }
00097
00098 void io::USART0::setCommunicationMode(const communicationMode &ar_comMode)
00099 {
00100     switch (ar_comMode)
00101     {
00102     case communicationMode::duplex:
00103     {
00104         USART0_ENABLE_TRANSMITTER;
00105         USART0_ENABLE_RECEIVER;
00106         break;
00107     }
00108     case communicationMode::receive:
00109     {
00110         USART0_ENABLE_RECEIVER;
00111         USART0_DISABLE_TRANSMITTER;
00112         break;
00113     }
00114     case communicationMode::transmit:
00115     {
00116         USART0_ENABLE_TRANSMITTER;
00117         USART0_DISABLE_RECEIVER;
00118         break;
00119     }
00120     }
00121 }
00122 }
00123
00124 void io::USART0::setParityMode(const parityMode& ar_parityMode)
00125 {
00126     switch (ar_parityMode)
00127     {
00128     case parityMode::noParity:
00129     {
00130         USART0_DISABLE_PARITY_MODE;
00131         break;
00132     }
00133     case parityMode::evenParity:
00134     {
00135         USART0_ENABLE_EVEN_PARITY_MODE;
00136         break;
00137     }
00138     case parityMode::oddParity:
00139     {
00140         USART0_ENABLE_ODD_PARITY_MODE;
00141         break;
00142     }
00143     }
```

```
00144
00145 }
00146
00147 void io::USART0::setFrameSize(const frameSize& ar_frameSize)
00148 {
00149     switch (ar_frameSize)
00150     {
00151         case frameSize::eightBits:
00152         {
00153             USART0_SET_8BIT_FRAME_SIZE;
00154             break;
00155         }
00156         case frameSize::sevenBits:
00157         {
00158             USART0_SET_7BIT_FRAME_SIZE;
00159             break;
00160         }
00161         case frameSize::sixBits:
00162         {
00163             USART0_SET_6BIT_FRAME_SIZE;
00164             break;
00165         }
00166         case frameSize::fiveBits:
00167         {
00168             USART0_SET_5BIT_FRAME_SIZE;
00169             break;
00170         }
00171         case frameSize::neineBits:
00172         {
00173             USART0_SET_9BIT_FRAME_SIZE;
00174             break;
00175         }
00176     }
00177 }
00178 }
00179 void io::USART0::setStopBit(const stopBit& ar_stopBit)
00180 {
00181     switch (ar_stopBit)
00182     {
00183         case stopBit::oneStopBit:
00184         {
00185             USART0_SET_ONE_STOP_BIT;
00186             break;
00187         }
00188         case stopBit::twoStopBits:
00189         {
00190             USART0_SET_TWO_STOP_BITS;
00191             break;
00192         }
00193     }
00194 }
00195 }
00196
00197
00198 uint8_t io::USART0::frameError()
00199 {
00200     return (m_status & (1 << USART0_FRAME_ERROR));
00201 }
00202 }
00203
00204 uint8_t io::USART0::dataOverrun()
00205 {
00206     return (m_status & (1 << USART0_DATA_OVERRUN));
00207 }
00208 }
00209
00210 uint8_t io::USART0::parityError()
00211 {
00212     return (m_status & (1 << USART0_PARITY_ERROR));
00213 }
00214 }
00215
00216 void io::USART0::sendFrame(const uint8_t* ap_dataBuffer, const uint8_t a_size)
00217 {
00218     while (!ready2Send()){};
00219     m_sizeData2Send = a_size;
00220     mp_data2Send = ap_dataBuffer;
00221     enableDataRegisterEmptyInterrupt(1);
00222 }
00223
00224
00225 void io::USART0::sendString(const char* ap_string)
00226 {
00227     while (!ready2Send()){};
00228     m_sizeData2Send = strlen(ap_string);
00229     mp_data2Send = reinterpret_cast<const uint8_t*>(ap_string);
00230     enableDataRegisterEmptyInterrupt(1);
00231 }
```

```

00231
00232 }
00233
00234
00235 void io::USART0::sendByte(const uint8_t &ar_byte)
00236 {
00237     static uint8_t l_byte2Send[3];
00238     l_byte2Send[0] = '0' + (ar_byte / 100);
00239     l_byte2Send[1] = '0' + ((ar_byte / 10) % 10);
00240     l_byte2Send[2] = '0' + (ar_byte % 10);
00241
00242     while (!ready2Send()){};
00243     m_sizeData2Send = 3;
00244     mp_data2Send = l_byte2Send;
00245     enableDataRegisterEmptyInterrupt(1);
00246 }
00247
00248 void io::USART0::sendLong(const uint32_t &ar_long)
00249 {
00250     static uint8_t l_word2Send[10];
00251     l_word2Send[0] = '0' + (ar_long / 1000000000);
00252     l_word2Send[1] = '0' + ((ar_long / 100000000) % 10);
00253     l_word2Send[2] = '0' + ((ar_long / 10000000) % 10);
00254     l_word2Send[3] = '0' + ((ar_long / 1000000) % 10);
00255     l_word2Send[4] = '0' + ((ar_long / 100000) % 10);
00256     l_word2Send[5] = '0' + ((ar_long / 10000) % 10);
00257     l_word2Send[6] = '0' + ((ar_long / 1000) % 10);
00258     l_word2Send[7] = '0' + ((ar_long / 100) % 10);
00259     l_word2Send[8] = '0' + ((ar_long / 10) % 10);
00260     l_word2Send[9] = '0' + (ar_long % 10);
00261
00262     while (!ready2Send()){};
00263     m_sizeData2Send = 10;
00264     mp_data2Send = l_word2Send;
00265     enableDataRegisterEmptyInterrupt(1);
00266 }
00267
00268 void io::USART0::sendWord(const uint16_t &ar_word)
00269 {
00270     static uint8_t l_word2Send[5];
00271     l_word2Send[0] = '0' + (ar_word / 10000);
00272     l_word2Send[1] = '0' + ((ar_word / 1000) % 10);
00273     l_word2Send[2] = '0' + ((ar_word / 100) % 10);
00274     l_word2Send[3] = '0' + ((ar_word / 10) % 10);
00275     l_word2Send[4] = '0' + (ar_word % 10);
00276
00277     while (!ready2Send()){};
00278     m_sizeData2Send = 5;
00279     mp_data2Send = l_word2Send;
00280     enableDataRegisterEmptyInterrupt(1);
00281 }
00282
00283
00284 void io::USART0::sendChar(const uint8_t &ar_char)
00285 {
00286     while (!ready2Send()){};
00287     m_sizeData2Send = 1;
00288     mp_data2Send = &ar_char;
00289     enableDataRegisterEmptyInterrupt(1);
00290 }
00291
00292
00293 void io::USART0::receiveChar(uint8_t &ar_char)
00294 {
00295     m_sizeData2Receive = 1;
00296     mp_data2Receive = &ar_char;
00297 }
00298
00299
00300 void io::USART0::receiveFrame(uint8_t *ap_dataBuffer, const uint8_t a_size)
00301 {
00302     m_sizeData2Receive = a_size;
00303     mp_data2Receive = ap_dataBuffer;
00304     // TODO: to be implemented
00305 }
00306
00307
00308
00309 void io::USART0::receiveString(const char *ap_string)
00310 {
00311     // TODO: to be implemented
00312 }
00313
00314
00315 void io::USART0::receiveCompleteServiceRoutine()
00316 {
00317     static volatile uint8_t *lp_dataReceived = mp_data2Receive;

```

```
00318     static uint16_t l_dataSize = m_sizeData2Receive;
00319
00320     m_status = USART0_CONTROL_STATUS_REGISTER;
00321
00322
00323
00324     if (l_dataSize)
00325     {
00326
00327         *lp_dataReceived++ = USART0_DATA_REGISTER;
00328         l_dataSize--;
00329         m_numberBytesReceived++;
00330
00331     }
00332     else
00333     {
00334         l_dataSize = m_sizeData2Receive;
00335         lp_dataReceived = mp_data2Receive;
00336     }
00337 }
00338 }
00339
00340 void io::USART0::dataRegisterEmptyServiceRoutine()
00341 {
00342
00343     if (m_sizeData2Send)
00344     {
00345         m_ready2Send = 0;
00346         USART0_DATA_REGISTER = *mp_data2Send++;
00347         m_sizeData2Send--;
00348         m_numberBytesSent++;
00349     }
00350     else
00351     {
00352         enableDataRegisterEmptyInterrupt(0);
00353         m_numberBytesSent = 0;
00354         m_ready2Send = 1;
00355     }
00356 }
00357 }
00358 }
00359 }
00360
00361
00362 void io::USART0::enableTransmitCompleteInterrupt(const uint8_t a_enable)
00363 {
00364     if (a_enable) {
00365         USART0_ENABLE_TRANSMIT_COMPLETE_INTERRUPT;
00366     } else {
00367         USART0_DISABLE_TRANSMIT_COMPLETE_INTERRUPT;
00368     }
00369 }
00370 }
00371
00372 void io::USART0::enableReceiveCompleteInterrupt(const uint8_t a_enable)
00373 {
00374     if (a_enable) {
00375         USART0_ENABLE_RECEIVE_COMPLETE_INTERRUPT;
00376     } else {
00377         USART0_DISABLE_RECEIVE_COMPLETE_INTERRUPT;
00378     }
00379 }
00380 }
00381 }
00382
00383 void io::USART0::enableDataRegisterEmptyInterrupt(const uint8_t a_enable)
00384 {
00385     if (a_enable) {
00386         USART0_ENABLE_DATA_REGISTER_EMPTY_INTERRUPT;
00387     } else {
00388         USART0_DISABLE_DATA_REGISTER_EMPTY_INTERRUPT;
00389     }
00390 }
00391 }
00392 }
00393
00394 void io::USART0::transmitCompleteServiceRoutine()
00395 {
00396
00397 }
00398
00399 uint16_t io::USART0::getNumberBytesSent()
00400 {
00401     return m_numberBytesSent;
00402 }
00403
00404 uint16_t io::USART0::getNumberBytesReceived()
```

```

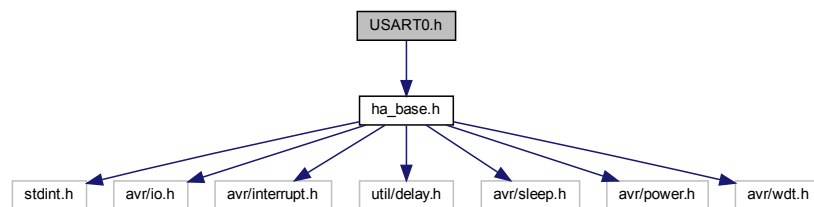
00405 {
00406     return m_numberBytesReceived;
00407 }
00408
00409 void io::USART0::resetNumberBytesReceived()
00410 {
00411     m_numberBytesReceived = 0;
00412 }
00413
00414 uint8_t io::USART0::ready2Send()
00415 {
00416     return m_ready2Send;
00417 }
00418 }
00419

```

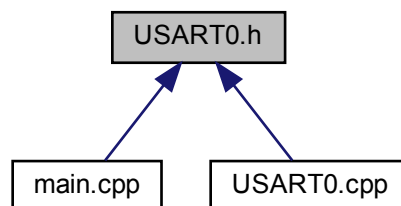
## 10.96 USART0.h File Reference

```
#include "ha_base.h"
```

Include dependency graph for USART0.h:



This graph shows which files directly or indirectly include this file:



### Classes

- class `io::USART0`  
Class for handling `USART0`.

### Namespaces

- `io`  
AVR chip internal i/o components.

## Enumerations

- enum `io::transmissionMode` : `uint8_t` { `io::transmissionMode::async` =0, `io::transmissionMode::sync`, `io::transmissionMode::masterSPI` }  
*USART0 transmission mode.*
- enum `io::communicationMode` : `uint8_t` { `io::communicationMode::duplex` =0, `io::communicationMode::transmit`, `io::communicationMode::receive` }  
*USART0 communication mode.*
- enum `io::parityMode` : `uint8_t` { `io::parityMode::noParity` =0, `io::parityMode::evenParity`, `io::parityMode::oddParity` }  
*USART0 parity mode.*
- enum `io::frameSize` : `uint8_t` { `io::frameSize::eightBits` =0, `io::frameSize::fiveBits`, `io::frameSize::sixBits`, `io::frameSize::sevenBits`, `io::frameSize::neineBits` }  
*USART0 frame size.*
- enum `io::stopBit` : `uint8_t` { `io::stopBit::oneStopBit` =0, `io::stopBit::twoStopBits` }  
*USART0 stop bit.*

## 10.97 USART0.h

```

00001 #ifndef USART_H
00002 #define USART_H
00003 #include "ha_base.h"
00004
00013 namespace io
00014 {
00015
00016
00025 enum class transmissionMode : uint8_t {
00026     async=0,
00027     sync,
00028     masterSPI
00029 };
00030
00039 enum class communicationMode : uint8_t {
00040     duplex=0,
00041     transmit,
00042     receive,
00044 };
00045
00054 enum class parityMode : uint8_t {
00055     noParity=0,
00056     evenParity,
00057     oddParity
00058 };
00059
00068 enum class frameSize : uint8_t {
00069     eightBits=0,
00070     fiveBits,
00071     sixBits,
00072     sevenBits,
00073     neineBits
00074 };
00075
00084 enum class stopBit : uint8_t{
00085     oneStopBit=0,
00086     twoStopBits
00087 };
00088
00098 class USART0
00099 {
00100
00101 public:
00102
00103
00113     static USART0& getInstance(const transmissionMode& ar_transMode = transmissionMode::async,
00114                               const communicationMode& ar_comMode = communicationMode::duplex,
00115                               const frameSize& ar_frameSize = frameSize::eightBits,
00116                               const stopBit& ar_stopBit = stopBit::oneStopBit,
00117                               const parityMode& ar_parityMode = parityMode::noParity);
00122     void setBaudRate();
00128     void setTransmissionMode(const transmissionMode& ar_transMode);
00134     void setCommunicationMode(const communicationMode& ar_comMode);

```



```

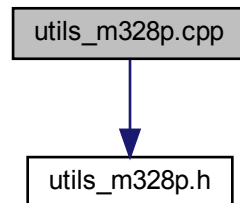
00140     void setParityMode(const parityMode& ar_parityMode);
00146     void setFrameSize(const frameSize& ar_frameSize);
00152     void setStopBit(const stopBit& ar_stopBit);
00159     void sendFrame(const uint8_t *ap_dataBuffer, const uint8_t a_size);
00160
00166     void sendString(const char *ap_string);
00167
00173     void receiveString(const char *ap_string);
00174
00180     void sendChar(const uint8_t &ar_char);
00181
00187     void sendByte(const uint8_t &ar_byte);
00188
00194     void sendWord(const uint16_t &ar_word);
00195
00201     void sendLong(const uint32_t &ar_long);
00202
00208     void receiveChar(uint8_t &ar_char);
00209
00217     void receiveFrame(uint8_t *ap_dataBuffer, const uint8_t a_size);
00218
00224     void enableTransmitCompleteInterrupt(const uint8_t a_enable);
00225
00231     void enableReceiveCompleteInterrupt(const uint8_t a_enable);
00232
00238     static void enableDataRegisterEmptyInterrupt(const uint8_t a_enable);
00239
00245     uint8_t frameError();
00246
00252     uint8_t dataOverrun();
00253
00259     uint8_t parityError();
00265     uint16_t getNumberBytesReceived();
00271     uint16_t getNumberBytesSent();
00277     uint8_t ready2Send();
00280     void resetNumberBytesReceived();
00283     static void receiveCompleteServiceRoutine() __asm__(STR(USART0_RECEIVE_COMPLETE_INTERRUPT))
__attribute__((__signal__, __used__, __externally_visible__));
00286     static void dataRegisterEmptyServiceRoutine() __asm__(STR(USART0_DATA_REGISTER_EMPTY_INTERRUPT))
__attribute__((__signal__, __used__, __externally_visible__));
00289     static void transmitCompleteServiceRoutine() __asm__(STR(USART0_TRANSMIT_COMPLETE_INTERRUPT))
__attribute__((__signal__, __used__, __externally_visible__));
00290
00291 protected:
00292
00293
00294
00295
00296 private:
00297     USART0(const transmissionMode& ar_transMode,
00308             const communicationMode& ar_comMode,
00309             const frameSize& ar_frameSize,
00310             const stopBit& ar_stopBit,
00311             const parityMode& ar_parityMode);
00312
00313     ~USART0();
00316
00317     USART0(const USART0&);
00321
00322     const USART0& operator=(const USART0&);
00326
00327
00328     static volatile uint8_t m_status;
00330     static const uint8_t *mp_data2Send;
00332     static uint8_t *mp_data2Receive;
00334     static uint16_t m_sizeData2Send;
00336     static uint16_t m_sizeData2Receive;
00338     static volatile uint16_t m_numberBytesReceived;
00340     static volatile uint16_t m_numberBytesSent;
00342     static volatile uint8_t m_ready2Send;
00345 };
00346
00347
00348
00349 }
00350
00351
00352
00353
00354
00355
00356
00357
00358
00359
00360
00361 #endif

```

## 10.98 utils\_m328p.cpp File Reference

```
#include "utils_m328p.h"
```

Include dependency graph for utils\_m328p.cpp:

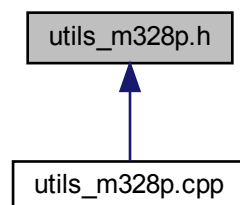


## 10.99 utils\_m328p.cpp

```
00001 #include "utils_m328p.h"
00002
00003 long utils::map(long x, long in_min, long in_max, long out_min, long out_max)
00004 {
00005     return (x - in_min) * (out_max - out_min) / (in_max - in_min) + out_min;
00006 }
```

## 10.100 utils\_m328p.h File Reference

This graph shows which files directly or indirectly include this file:



## Namespaces

- [utils](#)

## Functions

- long `utils::map` (long x, long in\_min, long in\_max, long out\_min, long out\_max)

## 10.101 utils\_m328p.h

```

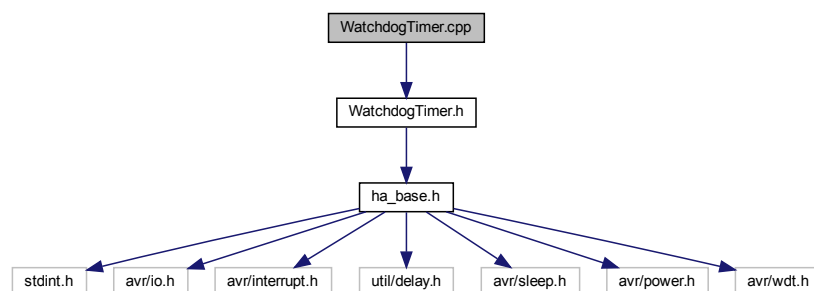
00001 #ifndef UTILSM329P_H
00002 #define UTILSM329P_H
00003 namespace utils
00004 {
00005
00006
00007 // delay functions
00008
00009
00010
00011 // multiplication / division operations bit-shift divide multiply
00012
00013
00014 // clock division (set prescaler)
00015
00016 // clock power saving through macros
00017
00018
00019 // mapping values
00020
00021 long map(long x, long in_min, long in_max, long out_min, long out_max);
00022
00023
00024 }
00025 #endif

```

## 10.102 WatchdogTimer.cpp File Reference

```
#include "WatchdogTimer.h"
```

Include dependency graph for WatchdogTimer.cpp:



## 10.103 WatchdogTimer.cpp

```

00001 #include "WatchdogTimer.h"
00002
00003
00004 core::WatchdogTimer& core::WatchdogTimer::getInstance()
00005 {
00006     static WatchdogTimer l_instance;
00007     return l_instance;
00008 }
00009
00010

```

```

00011 core::WatchdogTimer::WatchdogTimer()
00012 {
00013     sei();
00014     stop();
00015 }
00016
00017 core::WatchdogTimer::~WatchdogTimer()
00018 {
00019
00020 }
00021
00022 void core::WatchdogTimer::selectTimeOut(const timeOut &ar_timeOut)
00023 {
00024     m_timeOut = static_cast<uint8_t>(ar_timeOut);
00025     m_timeOut = static_cast<uint8_t>((m_timeOut & 7) | ((m_timeOut & 8) << 2));
00026     cli();
00027     wdt_reset();
00028     WATCHDOG_SELECT_TIMEOUT(m_timeOut);
00029     sei();
00030 }
00031
00032
00033 void core::WatchdogTimer::reset()
00034 {
00035     wdt_reset();
00036 }
00037
00038
00039 void core::WatchdogTimer::start(const operationMode &ar_operationMode)
00040 {
00041     m_operationMode = static_cast<uint8_t>(ar_operationMode);
00042     m_operationMode = static_cast<uint8_t>((m_operationMode & 1) << 6) | ((m_operationMode & 2) << 3
00043 ));
00044     cli();
00045     wdt_reset();
00046     WATCHDOG_START(m_operationMode,m_timeOut);
00047     sei();
00048 }
00049
00050 void core::WatchdogTimer::start(const operationMode &ar_operationMode, const timeOut &ar_timeOut)
00051 {
00052     m_timeOut = static_cast<uint8_t>(ar_timeOut);
00053     m_timeOut = static_cast<uint8_t>((m_timeOut & 7) | ((m_timeOut & 8) << 2));
00054     m_operationMode = static_cast<uint8_t>(ar_operationMode);
00055     m_operationMode = static_cast<uint8_t>((m_operationMode & 1) << 6) | ((m_operationMode & 2) << 3
00056 ));
00057     cli();
00058     wdt_reset();
00059     WATCHDOG_START(m_operationMode,m_timeOut);
00060     sei();
00061 }
00062
00063 void core::WatchdogTimer::stop()
00064 {
00065     cli();
00066     wdt_reset();
00067     WATCHDOG_STOP;
00068     sei();
00069 }
00070
00071

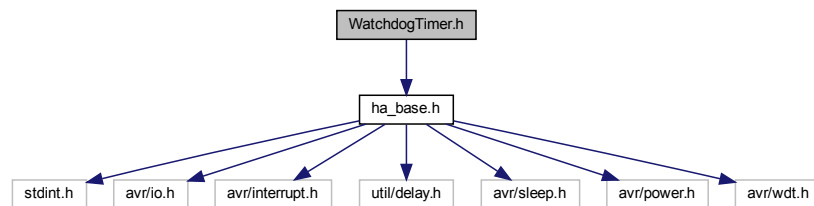
```

## 10.104 WatchdogTimer.h File Reference

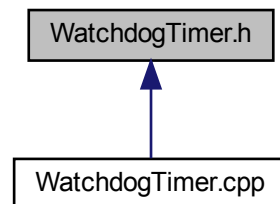
Header file of the WatchdogTimer class.

```
#include "ha_base.h"
```

Include dependency graph for WatchdogTimer.h:



This graph shows which files directly or indirectly include this file:



## Classes

- class [core::WatchdogTimer](#)

## Namespaces

- [core](#)  
*AVR chip internal core components.*

## Enumerations

- enum [core::timeOut](#) : `uint8_t` {  
[core::timeOut::to\\_16ms](#) =0, [core::timeOut::to\\_32ms](#), [core::timeOut::to\\_64ms](#), [core::timeOut::to\\_125ms](#),  
[core::timeOut::to\\_250ms](#), [core::timeOut::to\\_500ms](#), [core::timeOut::to\\_1s](#), [core::timeOut::to\\_2s](#),  
[core::timeOut::to\\_4s](#), [core::timeOut::to\\_8s](#) }
- enum [core::operationMode](#) : `uint8_t` {  
[core::operationMode::normal](#) =0, [core::operationMode::PWM\\_PC](#), [core::operationMode::PWM\\_PC\\_8bit](#),  
[core::operationMode::PWM\\_PC\\_9bit](#),  
[core::operationMode::PWM\\_PC\\_10bit](#), [core::operationMode::PWM\\_PFC\\_ICR](#), [core::operationMode::PWM\\_PFC\\_OCR](#),  
[core::operationMode::PWM\\_PC\\_ICR](#),  
[core::operationMode::PWM\\_PC\\_OCR](#), [core::operationMode::fast\\_PWM](#), [core::operationMode::fast\\_PWM\\_8bit](#),  
[core::operationMode::fast\\_PWM\\_9bit](#),  
[core::operationMode::fast\\_PWM\\_10bit](#), [core::operationMode::fast\\_PWM\\_ICR](#), [core::operationMode::fast\\_PWM\\_OCR](#),  
[core::operationMode::CTC\\_OCR](#),  
[core::operationMode::CTC\\_ICR](#), [core::operationMode::interrupt](#) =1, [core::operationMode::reset](#), [core::operationMode::interrupt](#),  
}

### 10.104.1 Detailed Description

Header file of the WatchdogTimer class.

Basic class abstraction of the WatchdogTimer peripheral

Usage example (test):

```
#include "WatchdogTimer.h" #include "Led.h" instantiate a Led object extern component::Led Led; component::Led
Led(io::Pin(1,io::PortB)); extern component::Led LedStart; component::Led LedStart(io::Pin(2,io::PortB));
```

```
int main(void) {
```

```
Init instantiate a Watchdog object core::WatchdogTimer &myWatchdog = core::WatchdogTimer::getInstance(); my
Watchdog.selectTimeOut(core::timeOut::to_8s); LedStart.on(); _delay_ms(5000); LedStart.off(); myWatchdog.
start(core::operationMode::reset); ----- Event loop ----- // while (1) {
```

```
Led.on();
_delay_ms(1000);
Led.off();
_delay_ms(1000);
```

```
} return 0; }
```

```
void core::WatchdogTimer::timeOutServiceRoutine() { for (uint8_t i=0;i<10;i++) { Led.on(); _delay_ms(100); Led.
off(); _delay_ms(100);
```

```
}
```

```
}
```

#### Author

Farid Oubbati ( <https://github.com/faroub>)

#### Date

March 2020

Definition in file [WatchdogTimer.h](#).

## 10.105 WatchdogTimer.h

```

00001
00057 #ifndef WATCHDOG_TIMER_H
00058 #define WATCHDOG_TIMER_H
00059 #include "ha_base.h"
00060
00061
00062
00063 namespace core
00064 {
00065     enum class timeOut : uint8_t {
00066         to_16ms=0,
00067         to_32ms,
00068         to_64ms,
00069         to_125ms,
00070         to_250ms,
00071         to_500ms,
00072         to_1s,
00073         to_2s,
00074         to_4s,
00075         to_8s,
00076     };
00077
00078     enum class operationMode : uint8_t {
00079         interrupt=1,
00080         reset,
00081         interrupt_reset,
00082     };
00083
00084     class WatchdogTimer
00085     {
00086     public:
00087         static WatchdogTimer& getInstance();
00088
00089         void selectTimeOut(const timeOut &ar_timeOut);
00090
00091         void reset();
00092
00093         void start(const operationMode &ar_operationMode);
00094
00095         void start(const operationMode &ar_operationMode, const timeOut &ar_timeOut);
00096
00097         void stop();
00098
00099         static void timeOutServiceRoutine() __asm__(STR(WATCHDOG_TIMEOUT_INTERRUPT))
00100         __attribute__((__signal__, __used__, __externally_visible__));
00101
00102     protected:
00103
00104     private:
00105         WatchdogTimer();
00106         ~WatchdogTimer();
00107         WatchdogTimer(const WatchdogTimer&);
00108         const WatchdogTimer& operator=(const WatchdogTimer&);
00109         uint8_t m_timeOut;
00110         uint8_t m_operationMode;
00111     };
00112 }
00113
00114 #endif

```





# Index

- `__externally_visible__`
  - `core::ADConverter`, [54](#)
  - `core::ExternInterrupt`, [67](#)
  - `core::TimerCounter0`, [127](#)
  - `core::TimerCounter1`, [138](#)
  - `core::TimerCounter2`, [148](#)
  - `core::WatchdogTimer`, [170](#)
  - `io::SPI`, [104](#)
  - `io::USART0`, [164](#)
- `__used__`
  - `core::ADConverter`, [54](#)
  - `core::ExternInterrupt`, [68](#)
  - `core::TimerCounter0`, [127](#)
  - `core::TimerCounter1`, [138](#)
  - `core::TimerCounter2`, [148](#)
  - `core::WatchdogTimer`, [170](#)
  - `io::SPI`, [104](#)
  - `io::USART0`, [164](#)
- `~ADConverter`
  - `core::ADConverter`, [47](#)
- `~Buzzer`
  - `component::Buzzer`, [56](#)
- `~DCMotor`
  - `component::DCMotor`, [60](#)
- `~ExternInterrupt`
  - `core::ExternInterrupt`, [63](#)
- `~LDR`
  - `component::LDR`, [69](#)
- `~Led`
  - `component::Led`, [73](#)
- `~Pin`
  - `io::Pin`, [82](#)
- `~PushButton`
  - `component::PushButton`, [88](#)
- `~SPI`
  - `io::SPI`, [99](#)
- `~ServoMotor`
  - `component::ServoMotor`, [92](#)
- `~StepperMotor`
  - `component::StepperMotor`, [107](#)
- `~TimerCounter0`
  - `core::TimerCounter0`, [120](#)
- `~TimerCounter1`
  - `core::TimerCounter1`, [130](#)
- `~TimerCounter2`
  - `core::TimerCounter2`, [141](#)
- `~USART0`
  - `io::USART0`, [151](#)
- `~WatchdogTimer`
  - `core::WatchdogTimer`, [167](#)
- A
  - `core`, [31](#)
- A0
  - `buzzer_pitches_16bit.h`, [186](#)
  - `buzzer_pitches_8bit.h`, [204](#)
- A1
  - `buzzer_pitches_16bit.h`, [186](#)
  - `buzzer_pitches_8bit.h`, [204](#)
- A2
  - `buzzer_pitches_16bit.h`, [186](#)
  - `buzzer_pitches_8bit.h`, [204](#)
- A3
  - `buzzer_pitches_16bit.h`, [186](#)
- A4
  - `buzzer_pitches_16bit.h`, [186](#)
- A5
  - `buzzer_pitches_16bit.h`, [187](#)
- A6
  - `buzzer_pitches_16bit.h`, [187](#)
- A7
  - `buzzer_pitches_16bit.h`, [187](#)
- `ADC.cpp`, [171](#)
- `ADC.h`, [175](#), [177](#)
- `ADC_ADJUST_RESULT_LEFT`
  - `ha_m328p.h`, [226](#)
- `ADC_ADJUST_RESULT_RIGHT`
  - `ha_m328p.h`, [226](#)
- `ADC_CONVERSION_COMPLETE_INTERRUPT`
  - `ha_m328p.h`, [226](#)
- `ADC_DISABLE`
  - `ha_m328p.h`, [226](#)
- `ADC_DISABLE_AUTOTRIGGER`
  - `ha_m328p.h`, [226](#)
- `ADC_DISABLE_CONVERSION_COMPLETE_INTERRUPT`
  - `ha_m328p.h`, [227](#)
- `ADC_DISABLE_DIGITAL_INPUT_REGISTER`
  - `ha_m328p.h`, [227](#)
- `ADC_ENABLE`
  - `ha_m328p.h`, [227](#)
- `ADC_ENABLE_AUTOTRIGGER`
  - `ha_m328p.h`, [227](#)
- `ADC_ENABLE_CONVERSION_COMPLETE_INTERRUPT`
  - `ha_m328p.h`, [227](#)
- `ADC_NoiseReduction`
  - `core`, [36](#)
- `ADC_SELECT_ANALOG_INPUT`
  - `ha_m328p.h`, [227](#)
- `ADC_SELECT_AUTO_TRIGGER_SOURCE`

- ha\_m328p.h, [228](#)
- ADC\_SELECT\_CLOCK\_PRESCALER
  - ha\_m328p.h, [228](#)
- ADC\_SELECT\_REF\_VOLTAGE
  - ha\_m328p.h, [228](#)
- ADC\_START\_CONVERSION
  - ha\_m328p.h, [228](#)
- ADC\_STOP\_CONVERSION
  - ha\_m328p.h, [228](#)
- ADConverter
  - core::ADConverter, [46](#), [47](#)
- ANALOG\_COMARATOR\_H
  - AnalogComparator.h, [179](#)
- analogComparator
  - core, [30](#)
- AnalogComparator.cpp, [178](#)
- AnalogComparator.h, [178](#), [180](#)
  - ANALOG\_COMARATOR\_H, [179](#)
- AREF
  - core, [35](#)
- async
  - io, [42](#)
- autoTriggerSource
  - core, [29](#)
- AVCC
  - core, [35](#)
- Ax0
  - buzzer\_pitches\_16bit.h, [187](#)
  - buzzer\_pitches\_8bit.h, [204](#)
- Ax1
  - buzzer\_pitches\_16bit.h, [187](#)
  - buzzer\_pitches\_8bit.h, [205](#)
- Ax2
  - buzzer\_pitches\_16bit.h, [187](#)
  - buzzer\_pitches\_8bit.h, [205](#)
- Ax3
  - buzzer\_pitches\_16bit.h, [188](#)
- Ax4
  - buzzer\_pitches\_16bit.h, [188](#)
- Ax5
  - buzzer\_pitches\_16bit.h, [188](#)
- Ax6
  - buzzer\_pitches\_16bit.h, [188](#)
- Ax7
  - buzzer\_pitches\_16bit.h, [188](#)
- B
  - core, [31](#)
- B0
  - buzzer\_pitches\_16bit.h, [188](#)
  - buzzer\_pitches\_8bit.h, [205](#)
- B1
  - buzzer\_pitches\_16bit.h, [189](#)
  - buzzer\_pitches\_8bit.h, [205](#)
- B2
  - buzzer\_pitches\_16bit.h, [189](#)
  - buzzer\_pitches\_8bit.h, [205](#)
- B3
  - buzzer\_pitches\_16bit.h, [189](#)
- B4
  - buzzer\_pitches\_16bit.h, [189](#)
- B5
  - buzzer\_pitches\_16bit.h, [189](#)
- B6
  - buzzer\_pitches\_16bit.h, [189](#)
- B7
  - buzzer\_pitches\_16bit.h, [190](#)
- BlinkLed/main.cpp
  - LED, [263](#)
  - main, [263](#)
  - TIMEDELAY, [263](#)
- BODMode
  - core, [30](#)
- buzz
  - component::Buzzer, [57](#)
- BUZZER
  - MusicBox/main.cpp, [270](#)
  - Organ/main.cpp, [274](#)
- Buzzer
  - component::Buzzer, [56](#)
- Buzzer.cpp, [180](#)
- Buzzer.h, [181](#), [182](#)
- buzzer\_pitches\_16bit.h, [184](#), [202](#)
  - A0, [186](#)
  - A1, [186](#)
  - A2, [186](#)
  - A3, [186](#)
  - A4, [186](#)
  - A5, [187](#)
  - A6, [187](#)
  - A7, [187](#)
  - Ax0, [187](#)
  - Ax1, [187](#)
  - Ax2, [187](#)
  - Ax3, [188](#)
  - Ax4, [188](#)
  - Ax5, [188](#)
  - Ax6, [188](#)
  - Ax7, [188](#)
  - B0, [188](#)
  - B1, [189](#)
  - B2, [189](#)
  - B3, [189](#)
  - B4, [189](#)
  - B5, [189](#)
  - B6, [189](#)
  - B7, [190](#)
  - C0, [190](#)
  - C1, [190](#)
  - C2, [190](#)
  - C3, [190](#)
  - C4, [190](#)
  - C5, [191](#)
  - C6, [191](#)
  - C7, [191](#)
  - Cx0, [191](#)
  - Cx1, [191](#)

- Cx2, [191](#)
- Cx3, [192](#)
- Cx4, [192](#)
- Cx5, [192](#)
- Cx6, [192](#)
- Cx7, [192](#)
- D0, [192](#)
- D1, [193](#)
- D2, [193](#)
- D3, [193](#)
- D4, [193](#)
- D5, [193](#)
- D6, [193](#)
- D7, [194](#)
- Dx0, [194](#)
- Dx1, [194](#)
- Dx2, [194](#)
- Dx3, [194](#)
- Dx4, [194](#)
- Dx5, [195](#)
- Dx6, [195](#)
- Dx7, [195](#)
- E0, [195](#)
- E1, [195](#)
- E2, [195](#)
- E3, [196](#)
- E4, [196](#)
- E5, [196](#)
- E6, [196](#)
- E7, [196](#)
- F0, [196](#)
- F1, [197](#)
- F2, [197](#)
- F3, [197](#)
- F4, [197](#)
- F5, [197](#)
- F6, [197](#)
- F7, [198](#)
- Fx0, [198](#)
- Fx1, [198](#)
- Fx2, [198](#)
- Fx3, [198](#)
- Fx4, [198](#)
- Fx5, [199](#)
- Fx6, [199](#)
- Fx7, [199](#)
- G0, [199](#)
- G1, [199](#)
- G2, [199](#)
- G3, [200](#)
- G4, [200](#)
- G5, [200](#)
- G6, [200](#)
- G7, [200](#)
- Gx0, [200](#)
- Gx1, [201](#)
- Gx2, [201](#)
- Gx3, [201](#)
- Gx4, [201](#)
- Gx5, [201](#)
- Gx6, [201](#)
- Gx7, [202](#)
- buzzer\_pitches\_8bit.h, [203](#), [210](#)
  - A0, [204](#)
  - A1, [204](#)
  - A2, [204](#)
  - Ax0, [204](#)
  - Ax1, [205](#)
  - Ax2, [205](#)
  - B0, [205](#)
  - B1, [205](#)
  - B2, [205](#)
  - C1, [205](#)
  - C2, [206](#)
  - C3, [206](#)
  - Cx0, [206](#)
  - Cx2, [206](#)
  - Cx3, [206](#)
  - D1, [206](#)
  - D2, [207](#)
  - D3, [207](#)
  - Dx0, [207](#)
  - Dx2, [207](#)
  - Dx3, [207](#)
  - E1, [207](#)
  - E2, [208](#)
  - E3, [208](#)
  - F1, [208](#)
  - F2, [208](#)
  - F3, [208](#)
  - Fx1, [208](#)
  - Fx2, [209](#)
  - Fx3, [209](#)
  - G1, [209](#)
  - G2, [209](#)
  - G3, [209](#)
  - Gx0, [209](#)
  - Gx1, [210](#)
  - Gx2, [210](#)
- C0
  - buzzer\_pitches\_16bit.h, [190](#)
- C1
  - buzzer\_pitches\_16bit.h, [190](#)
  - buzzer\_pitches\_8bit.h, [205](#)
- C2
  - buzzer\_pitches\_16bit.h, [190](#)
  - buzzer\_pitches\_8bit.h, [206](#)
- C3
  - buzzer\_pitches\_16bit.h, [190](#)
  - buzzer\_pitches\_8bit.h, [206](#)
- C4
  - buzzer\_pitches\_16bit.h, [190](#)
- C5
  - buzzer\_pitches\_16bit.h, [191](#)
- C6
  - buzzer\_pitches\_16bit.h, [191](#)

- C7
  - buzzer\_pitches\_16bit.h, 191
- channel
  - core, 30
- clear
  - core, 32
- clockPrescaler
  - core, 31
  - io, 38
- clockSource
  - core, 31
- communicationMode
  - io, 39
- compareOutputMode
  - core, 32
- component, 27
  - fullStep, 28
  - halfStep, 28
  - mode, 27
- component::Buzzer, 55
  - ~Buzzer, 56
  - buzz, 57
  - Buzzer, 56
  - m\_pin, 58
  - pause, 58
- component::DCMotor, 59
  - ~DCMotor, 60
  - connect, 60
  - DCMotor, 60
  - disconnect, 60
  - m\_pin, 62
  - off, 61
  - on, 61
  - spin, 61
  - stop, 61
  - toggle, 62
- component::LDR, 68
  - ~LDR, 69
  - getLux, 70
  - getResistance, 70
  - LDR, 69
  - m\_pin, 71
  - mr\_ADC, 71
- component::Led, 71
  - ~Led, 73
  - isOff, 73
  - isOn, 74
  - Led, 73
  - m\_pin, 75
  - off, 74
  - on, 74
  - toggle, 75
- component::PushButton, 87
  - ~PushButton, 88
  - getPressedCount, 89
  - isPressed, 89
  - m\_buttonPressed, 90
  - m\_pin, 90
  - mr\_isActiveLow, 90
  - mr\_useInternalPullUp, 90
  - PushButton, 88
  - resetPressedCount, 89
- component::ServoMotor, 91
  - ~ServoMotor, 92
  - computePulseCycleCount, 92
  - computePulseWidthMaxCount, 93
  - computePulseWidthMidCount, 93
  - computePulseWidthMinCount, 93
  - computeRotationAngleCount, 93
  - connect, 94
  - disconnect, 94
  - m\_pin, 96
  - m\_pulseCycle, 96
  - m\_pulseWidthMax, 96
  - m\_pulseWidthMid, 96
  - m\_pulseWidthMin, 96
  - off, 94
  - on, 95
  - rotate, 95
  - ServoMotor, 92
  - toggle, 95
- component::StepperMotor, 105
  - ~StepperMotor, 107
  - computeStepDelay, 107
  - currentPos, 108
  - goalReached, 108
  - m\_accelTime, 113
  - m\_constSpeedTime, 113
  - m\_currentPos, 113
  - m\_decelTime, 114
  - m\_goalReached, 114
  - m\_pinCoil1, 114
  - m\_pinCoil2, 114
  - m\_pinCoil3, 114
  - m\_pinCoil4, 115
  - setCurrentPos, 109
  - step, 109–111
  - stepDelay, 112
  - stepMode, 115
  - StepperMotor, 106
  - stepPulse, 112
- computePulseCycleCount
  - component::ServoMotor, 92
- computePulseWidthMaxCount
  - component::ServoMotor, 93
- computePulseWidthMidCount
  - component::ServoMotor, 93
- computePulseWidthMinCount
  - component::ServoMotor, 93
- computeRotationAngleCount
  - component::ServoMotor, 93
- computeStepDelay
  - component::StepperMotor, 107
- connect
  - component::DCMotor, 60
  - component::ServoMotor, 94

- conversionCompleteServiceRoutine
  - core::ADConverter, 47
- core, 28
  - A, 31
  - ADC\_NoiseReduction, 36
  - analogComparator, 30
  - AREF, 35
  - autoTriggerSource, 29
  - AVCC, 35
  - B, 31
  - BODMode, 30
  - channel, 30
  - clear, 32
  - clockPrescaler, 31
  - clockSource, 31
  - compareOutputMode, 32
  - CTC\_ICR, 33, 34
  - CTC\_OCR, 33, 34
  - disabled, 30
  - enabled, 30
  - extendedStandby, 36
  - extern\_Clock\_T0\_Falling\_Edge, 32
  - extern\_Clock\_T0\_Rising\_Edge, 32
  - extInterrupt, 30
  - fallingEdge, 36
  - fast\_PWM, 33
  - fast\_PWM\_10bit, 33
  - fast\_PWM\_8bit, 33
  - fast\_PWM\_9bit, 33
  - fast\_PWM\_ICR, 33
  - fast\_PWM\_OCR, 33
  - freeRunning, 30
  - Idle, 36
  - internal, 35
  - interrupt, 33, 34
  - interrupt\_reset, 33, 34
  - logicalChange, 36
  - lowLevel, 36
  - noClock, 32
  - normal, 32, 33
  - operationMode, 32, 33
  - PCINTB, 34
  - PCINTC, 34
  - PCINTD, 34
  - pinChangePort, 34
  - powerDown, 36
  - powerSave, 36
  - PS\_1, 32
  - PS\_1024, 32
  - PS\_128, 31, 32
  - PS\_16, 31
  - PS\_2, 31
  - PS\_256, 32
  - PS\_32, 31, 32
  - PS\_4, 31
  - PS\_64, 31, 32
  - PS\_8, 31, 32
  - PWM\_PC, 33
  - PWM\_PC\_10bit, 33
  - PWM\_PC\_8bit, 33
  - PWM\_PC\_9bit, 33
  - PWM\_PC\_ICR, 33
  - PWM\_PC\_OCR, 33
  - PWM\_PFC\_ICR, 33
  - PWM\_PFC\_OCR, 33
  - referenceVoltage, 34
  - res\_10bit, 35
  - res\_11bit, 35
  - res\_12bit, 35
  - res\_13bit, 35
  - res\_14bit, 35
  - res\_15bit, 35
  - res\_16bit, 35
  - res\_8bit, 35
  - res\_9bit, 35
  - reset, 33, 34
  - resolution, 35
  - risingEdge, 36
  - senseControl, 35
  - set, 32
  - sleepMode, 36
  - standby, 36
  - timeOut, 36
  - timer0CompareA, 30
  - timer0Overflow, 30
  - timer1Capture, 30
  - timer1CompareB, 30
  - timer1Overflow, 30
  - to\_125ms, 37
  - to\_16ms, 36
  - to\_1s, 37
  - to\_250ms, 37
  - to\_2s, 37
  - to\_32ms, 36
  - to\_4s, 37
  - to\_500ms, 37
  - to\_64ms, 37
  - to\_8s, 37
  - toggle, 32
- core::ADConverter, 45
  - \_\_externally\_visible\_\_, 54
  - \_\_used\_\_, 54
  - ~ADConverter, 47
  - ADConverter, 46, 47
  - conversionCompleteServiceRoutine, 47
  - enableAutoTrigger, 49
  - enableConversionCompleteInterrupt, 49
  - getConversionResult, 50
  - getInstance, 51
  - m\_conversionResult, 54
  - m\_resolution, 54
  - operator=, 51
  - selectAnalogInput, 51
  - selectAutoTriggerSource, 52
  - selectClockPrescaler, 52
  - selectReferenceVoltage, 53

- start, 53
- stop, 53
- core::AnalogComparator, 55
- core::ExternInterrupt, 62
  - \_\_externally\_visible\_\_, 67
  - \_\_used\_\_, 68
  - ~ExternInterrupt, 63
  - enableInt0, 64
  - enableInt1, 64
  - enablePinChange, 64
  - enablePinChangeMaskPortB, 65
  - enablePinChangeMaskPortC, 65
  - enablePinChangeMaskPortD, 65
  - ExternInterrupt, 63, 64
  - getInstance, 66
  - Int0ServiceRoutine, 66
  - Int1ServiceRoutine, 66
  - operator=, 66
  - pinChangePortBServiceRoutine, 66
  - pinChangePortCServiceRoutine, 67
  - pinChangePortDServiceRoutine, 67
  - setInt0SenseControl, 67
  - setInt1SenseControl, 67
- core::MCU, 75
  - disableBOD, 76
  - enableADC, 76
  - enableSPI, 76
  - enableTimerCounter0, 77
  - enableTimerCounter1, 77
  - enableTimerCounter2, 77
  - enableTWI, 78
  - enableUSART0, 78
  - goToSleep, 78
  - init, 79
  - selectSleepMode, 79
  - sleepEnable, 79
- core::TimerCounter, 115
  - enableOutputCompareMatchInterrupt, 116
  - enableOverflowInterrupt, 116
  - getClockPrescaler, 116
  - getCounter, 116
  - getOutputCompareRegister, 117
  - selectClockSource, 117
  - selectCompareOutputMode, 117
  - selectOperationMode, 117
  - setCounter, 117
  - setOutputCompareRegister, 117
  - start, 118
  - stop, 118
- core::TimerCounter0, 118
  - \_\_externally\_visible\_\_, 127
  - \_\_used\_\_, 127
  - ~TimerCounter0, 120
  - enableOutputCompareMatchInterrupt, 121
  - enableOverflowInterrupt, 121
  - getClockPrescaler, 122
  - getCounter, 122
  - getInstance, 122
  - getOutputCompareRegister, 123
  - m\_clockPrescaler, 127
  - m\_clockSource, 128
  - operator=, 123
  - outputCompareMatchAServiceRoutine, 123
  - outputCompareMatchBServiceRoutine, 123
  - overflowServiceRoutine, 124
  - selectClockSource, 124
  - selectCompareOutputMode, 125
  - selectOperationMode, 125
  - setCounter, 126
  - setOutputCompareRegister, 126
  - start, 126
  - stop, 127
  - TimerCounter0, 120
- core::TimerCounter1, 128
  - \_\_externally\_visible\_\_, 138
  - \_\_used\_\_, 138
  - ~TimerCounter1, 130
  - enableInputCaptureInterrupt, 130
  - enableOutputCompareMatchInterrupt, 131
  - enableOverflowInterrupt, 131
  - getClockPrescaler, 132
  - getCounter, 132
  - getInputCaptureRegister, 132
  - getInstance, 132
  - getOutputCompareRegister, 133
  - inputCaptureServiceRoutine, 133
  - m\_clockPrescaler, 138
  - m\_clockSource, 139
  - operator=, 133
  - outputCompareMatchAServiceRoutine, 133
  - outputCompareMatchBServiceRoutine, 134
  - overflowServiceRoutine, 134
  - selectClockSource, 134
  - selectCompareOutputMode, 135
  - selectOperationMode, 135
  - setCounter, 137
  - setInputCaptureRegister, 137
  - setOutputCompareRegister, 137
  - start, 137
  - stop, 138
  - TimerCounter1, 130
- core::TimerCounter2, 139
  - \_\_externally\_visible\_\_, 148
  - \_\_used\_\_, 148
  - ~TimerCounter2, 141
  - enableOutputCompareMatchInterrupt, 141
  - enableOverflowInterrupt, 142
  - getClockPrescaler, 142
  - getCounter, 143
  - getInstance, 143
  - getOutputCompareRegister, 143
  - m\_clockPrescaler, 148
  - m\_clockSource, 148
  - operator=, 144
  - outputCompareMatchAServiceRoutine, 144
  - outputCompareMatchBServiceRoutine, 144

- overflowServiceRoutine, [144](#)
- selectClockSource, [144](#)
- selectCompareOutputMode, [145](#)
- selectOperationMode, [146](#)
- setCounter, [146](#)
- setOutputCompareRegister, [147](#)
- start, [147](#)
- stop, [147](#)
- TimerCounter2, [141](#)
- core::WatchdogTimer, [166](#)
  - \_\_externally\_visible\_\_, [170](#)
  - \_\_used\_\_, [170](#)
  - ~WatchdogTimer, [167](#)
  - getInstance, [167](#)
  - m\_operationMode, [170](#)
  - m\_timeOut, [170](#)
  - operator=, [168](#)
  - reset, [168](#)
  - selectTimeOut, [168](#)
  - start, [168](#), [169](#)
  - stop, [169](#)
  - timeOutServiceRoutine, [169](#)
  - WatchdogTimer, [167](#)
- CTC\_ICR
  - core, [33](#), [34](#)
- CTC\_OCR
  - core, [33](#), [34](#)
- currentPos
  - component::StepperMotor, [108](#)
- Cx0
  - buzzer\_pitches\_16bit.h, [191](#)
  - buzzer\_pitches\_8bit.h, [206](#)
- Cx1
  - buzzer\_pitches\_16bit.h, [191](#)
- Cx2
  - buzzer\_pitches\_16bit.h, [191](#)
  - buzzer\_pitches\_8bit.h, [206](#)
- Cx3
  - buzzer\_pitches\_16bit.h, [192](#)
  - buzzer\_pitches\_8bit.h, [206](#)
- Cx4
  - buzzer\_pitches\_16bit.h, [192](#)
- Cx5
  - buzzer\_pitches\_16bit.h, [192](#)
- Cx6
  - buzzer\_pitches\_16bit.h, [192](#)
- Cx7
  - buzzer\_pitches\_16bit.h, [192](#)
- D0
  - buzzer\_pitches\_16bit.h, [192](#)
- D1
  - buzzer\_pitches\_16bit.h, [193](#)
  - buzzer\_pitches\_8bit.h, [206](#)
- D2
  - buzzer\_pitches\_16bit.h, [193](#)
  - buzzer\_pitches\_8bit.h, [207](#)
- D3
  - buzzer\_pitches\_16bit.h, [193](#)
- buzzer\_pitches\_8bit.h, [207](#)
- D4
  - buzzer\_pitches\_16bit.h, [193](#)
- D5
  - buzzer\_pitches\_16bit.h, [193](#)
- D6
  - buzzer\_pitches\_16bit.h, [193](#)
- D7
  - buzzer\_pitches\_16bit.h, [194](#)
- dataMode
  - io, [39](#)
- dataOrder
  - io, [40](#)
- dataOverrun
  - io::USART0, [151](#)
- dataRegisterEmptyServiceRoutine
  - io::USART0, [152](#)
- DCMotor
  - component::DCMotor, [60](#)
- DCMotor.cpp, [211](#)
- DCMotor.h, [211](#), [214](#)
- disable
  - io, [41](#)
- disableBOD
  - core::MCU, [76](#)
- disabled
  - core, [30](#)
- disconnect
  - component::DCMotor, [60](#)
  - component::ServoMotor, [94](#)
- duplex
  - io, [39](#)
- Dx0
  - buzzer\_pitches\_16bit.h, [194](#)
  - buzzer\_pitches\_8bit.h, [207](#)
- Dx1
  - buzzer\_pitches\_16bit.h, [194](#)
- Dx2
  - buzzer\_pitches\_16bit.h, [194](#)
  - buzzer\_pitches\_8bit.h, [207](#)
- Dx3
  - buzzer\_pitches\_16bit.h, [194](#)
  - buzzer\_pitches\_8bit.h, [207](#)
- Dx4
  - buzzer\_pitches\_16bit.h, [194](#)
- Dx5
  - buzzer\_pitches\_16bit.h, [195](#)
- Dx6
  - buzzer\_pitches\_16bit.h, [195](#)
- Dx7
  - buzzer\_pitches\_16bit.h, [195](#)
- E0
  - buzzer\_pitches\_16bit.h, [195](#)
- E1
  - buzzer\_pitches\_16bit.h, [195](#)
  - buzzer\_pitches\_8bit.h, [207](#)
- E2
  - buzzer\_pitches\_16bit.h, [195](#)

- buzzer\_pitches\_8bit.h, [208](#)
- E3
  - buzzer\_pitches\_16bit.h, [196](#)
  - buzzer\_pitches\_8bit.h, [208](#)
- E4
  - buzzer\_pitches\_16bit.h, [196](#)
- E5
  - buzzer\_pitches\_16bit.h, [196](#)
- E6
  - buzzer\_pitches\_16bit.h, [196](#)
- E7
  - buzzer\_pitches\_16bit.h, [196](#)
- eightBits
  - io, [40](#)
- enableADC
  - core::MCU, [76](#)
- enableAutoTrigger
  - core::ADConverter, [49](#)
- enableConversionCompleteInterrupt
  - core::ADConverter, [49](#)
- enabled
  - core, [30](#)
- enableDataRegisterEmptyInterrupt
  - io::USART0, [152](#)
- enableInputCaptureInterrupt
  - core::TimerCounter1, [130](#)
- enableInt0
  - core::ExternInterrupt, [64](#)
- enableInt1
  - core::ExternInterrupt, [64](#)
- enableOutputCompareMatchInterrupt
  - core::TimerCounter, [116](#)
  - core::TimerCounter0, [121](#)
  - core::TimerCounter1, [131](#)
  - core::TimerCounter2, [141](#)
- enableOverflowInterrupt
  - core::TimerCounter, [116](#)
  - core::TimerCounter0, [121](#)
  - core::TimerCounter1, [131](#)
  - core::TimerCounter2, [142](#)
- enablePinChange
  - core::ExternInterrupt, [64](#)
- enablePinChangeMaskPortB
  - core::ExternInterrupt, [65](#)
- enablePinChangeMaskPortC
  - core::ExternInterrupt, [65](#)
- enablePinChangeMaskPortD
  - core::ExternInterrupt, [65](#)
- enableReceiveCompleteInterrupt
  - io::USART0, [153](#)
- enableSPI
  - core::MCU, [76](#)
- enableTimerCounter0
  - core::MCU, [77](#)
- enableTimerCounter1
  - core::MCU, [77](#)
- enableTimerCounter2
  - core::MCU, [77](#)
- enableTransferCompleteInterrupt
  - io::SPI, [99](#)
- enableTransmitCompleteInterrupt
  - io::USART0, [153](#)
- enableTWI
  - core::MCU, [78](#)
- enableUSART0
  - core::MCU, [78](#)
- evenParity
  - io, [41](#)
- EXT\_INT\_DISABLE\_INT0
  - ha\_m328p.h, [228](#)
- EXT\_INT\_DISABLE\_INT1
  - ha\_m328p.h, [229](#)
- EXT\_INT\_DISABLE\_PIN\_CHANGE\_INTERRUPT
  - ha\_m328p.h, [229](#)
- EXT\_INT\_DISABLE\_PIN\_CHANGE\_MASK\_PORTB
  - ha\_m328p.h, [229](#)
- EXT\_INT\_DISABLE\_PIN\_CHANGE\_MASK\_PORTC
  - ha\_m328p.h, [229](#)
- EXT\_INT\_DISABLE\_PIN\_CHANGE\_MASK\_PORTD
  - ha\_m328p.h, [229](#)
- EXT\_INT\_ENABLE\_INT0
  - ha\_m328p.h, [229](#)
- EXT\_INT\_ENABLE\_INT1
  - ha\_m328p.h, [230](#)
- EXT\_INT\_ENABLE\_PIN\_CHANGE\_INTERRUPT
  - ha\_m328p.h, [230](#)
- EXT\_INT\_ENABLE\_PIN\_CHANGE\_MASK\_PORTB
  - ha\_m328p.h, [230](#)
- EXT\_INT\_ENABLE\_PIN\_CHANGE\_MASK\_PORTC
  - ha\_m328p.h, [230](#)
- EXT\_INT\_ENABLE\_PIN\_CHANGE\_MASK\_PORTD
  - ha\_m328p.h, [230](#)
- EXT\_INT\_INT0\_INTERRUPT
  - ha\_m328p.h, [230](#)
- EXT\_INT\_INT1\_INTERRUPT
  - ha\_m328p.h, [231](#)
- EXT\_INT\_PIN\_CHANGE\_PORTB\_INTERRUPT
  - ha\_m328p.h, [231](#)
- EXT\_INT\_PIN\_CHANGE\_PORTC\_INTERRUPT
  - ha\_m328p.h, [231](#)
- EXT\_INT\_PIN\_CHANGE\_PORTD\_INTERRUPT
  - ha\_m328p.h, [231](#)
- EXT\_INT\_SET\_INT0\_SENSE\_CONTROL
  - ha\_m328p.h, [231](#)
- EXT\_INT\_SET\_INT1\_SENSE\_CONTROL
  - ha\_m328p.h, [231](#)
- extendedStandby
  - core, [36](#)
- extern\_Clock\_T0\_Falling\_Edge
  - core, [32](#)
- extern\_Clock\_T0\_Rising\_Edge
  - core, [32](#)
- ExternInterrupt
  - core::ExternInterrupt, [63](#), [64](#)
- ExternInterrupt.cpp, [214](#), [215](#)
- ExternInterrupt.h, [216](#), [219](#)



extInterrupt  
     core, 30

F0  
     buzzer\_pitches\_16bit.h, 196

F1  
     buzzer\_pitches\_16bit.h, 197  
     buzzer\_pitches\_8bit.h, 208

F2  
     buzzer\_pitches\_16bit.h, 197  
     buzzer\_pitches\_8bit.h, 208

F3  
     buzzer\_pitches\_16bit.h, 197  
     buzzer\_pitches\_8bit.h, 208

F4  
     buzzer\_pitches\_16bit.h, 197

F5  
     buzzer\_pitches\_16bit.h, 197

F6  
     buzzer\_pitches\_16bit.h, 197

F7  
     buzzer\_pitches\_16bit.h, 198

fallingEdge  
     core, 36

fast\_PWM  
     core, 33

fast\_PWM\_10bit  
     core, 33

fast\_PWM\_8bit  
     core, 33

fast\_PWM\_9bit  
     core, 33

fast\_PWM\_ICR  
     core, 33

fast\_PWM\_OCR  
     core, 33

first\_LSB  
     io, 40

first\_MSB  
     io, 40

fiveBits  
     io, 40

frameError  
     io::USART0, 154

frameSize  
     io, 40

freeRunning  
     core, 30

fullStep  
     component, 28

Fx0  
     buzzer\_pitches\_16bit.h, 198

Fx1  
     buzzer\_pitches\_16bit.h, 198  
     buzzer\_pitches\_8bit.h, 208

Fx2  
     buzzer\_pitches\_16bit.h, 198  
     buzzer\_pitches\_8bit.h, 209

Fx3  
     buzzer\_pitches\_16bit.h, 198  
     buzzer\_pitches\_8bit.h, 209

Fx4  
     buzzer\_pitches\_16bit.h, 198

Fx5  
     buzzer\_pitches\_16bit.h, 199

Fx6  
     buzzer\_pitches\_16bit.h, 199

Fx7  
     buzzer\_pitches\_16bit.h, 199

G0  
     buzzer\_pitches\_16bit.h, 199

G1  
     buzzer\_pitches\_16bit.h, 199  
     buzzer\_pitches\_8bit.h, 209

G2  
     buzzer\_pitches\_16bit.h, 199  
     buzzer\_pitches\_8bit.h, 209

G3  
     buzzer\_pitches\_16bit.h, 200  
     buzzer\_pitches\_8bit.h, 209

G4  
     buzzer\_pitches\_16bit.h, 200

G5  
     buzzer\_pitches\_16bit.h, 200

G6  
     buzzer\_pitches\_16bit.h, 200

G7  
     buzzer\_pitches\_16bit.h, 200

getClockPrescaler  
     core::TimerCounter, 116  
     core::TimerCounter0, 122  
     core::TimerCounter1, 132  
     core::TimerCounter2, 142

getConversionResult  
     core::ADConverter, 50

getCounter  
     core::TimerCounter, 116  
     core::TimerCounter0, 122  
     core::TimerCounter1, 132  
     core::TimerCounter2, 143

getInputCaptureRegister  
     core::TimerCounter1, 132

getInstance  
     core::ADConverter, 51  
     core::ExternInterrupt, 66  
     core::TimerCounter0, 122  
     core::TimerCounter1, 132  
     core::TimerCounter2, 143  
     core::WatchdogTimer, 167  
     io::SPI, 100  
     io::USART0, 154

getLux  
     component::LDR, 70

getNumberBytesReceived  
     io::USART0, 154

getNumberBytesSent  
     io::USART0, 155

- getOutputCompareRegister
  - core::TimerCounter, [117](#)
  - core::TimerCounter0, [123](#)
  - core::TimerCounter1, [133](#)
  - core::TimerCounter2, [143](#)
- getPinNumber
  - io::Pin, [82](#)
- getPressedCount
  - component::PushButton, [89](#)
- getResistance
  - component::LDR, [70](#)
- goalReached
  - component::StepperMotor, [108](#)
- goToSleep
  - core::MCU, [78](#)
- Gx0
  - buzzer\_pitches\_16bit.h, [200](#)
  - buzzer\_pitches\_8bit.h, [209](#)
- Gx1
  - buzzer\_pitches\_16bit.h, [201](#)
  - buzzer\_pitches\_8bit.h, [210](#)
- Gx2
  - buzzer\_pitches\_16bit.h, [201](#)
  - buzzer\_pitches\_8bit.h, [210](#)
- Gx3
  - buzzer\_pitches\_16bit.h, [201](#)
- Gx4
  - buzzer\_pitches\_16bit.h, [201](#)
- Gx5
  - buzzer\_pitches\_16bit.h, [201](#)
- Gx6
  - buzzer\_pitches\_16bit.h, [201](#)
- Gx7
  - buzzer\_pitches\_16bit.h, [202](#)
- ha\_base.h, [220](#), [221](#)
  - STR, [221](#)
  - STRx, [221](#)
- ha\_m328p.h, [222](#), [253](#)
  - ADC\_ADJUST\_RESULT\_LEFT, [226](#)
  - ADC\_ADJUST\_RESULT\_RIGHT, [226](#)
  - ADC\_CONVERSION\_COMPLETE\_INTERRUPT, [226](#)
  - ADC\_DISABLE, [226](#)
  - ADC\_DISABLE\_AUTOTRIGGER, [226](#)
  - ADC\_DISABLE\_CONVERSION\_COMPLETE\_INTERRUPT, [227](#)
  - ADC\_DISABLE\_DIGITAL\_INPUT\_REGISTER, [227](#)
  - ADC\_ENABLE, [227](#)
  - ADC\_ENABLE\_AUTOTRIGGER, [227](#)
  - ADC\_ENABLE\_CONVERSION\_COMPLETE\_INTERRUPT, [227](#)
  - ADC\_SELECT\_ANALOG\_INPUT, [227](#)
  - ADC\_SELECT\_AUTO\_TRIGGER\_SOURCE, [228](#)
  - ADC\_SELECT\_CLOCK\_PRESCALER, [228](#)
  - ADC\_SELECT\_REF\_VOLTAGE, [228](#)
  - ADC\_START\_CONVERSION, [228](#)
  - ADC\_STOP\_CONVERSION, [228](#)
  - EXT\_INT\_DISABLE\_INT0, [228](#)
  - EXT\_INT\_DISABLE\_INT1, [229](#)
  - EXT\_INT\_DISABLE\_PIN\_CHANGE\_INTERRUPT, [229](#)
  - EXT\_INT\_DISABLE\_PIN\_CHANGE\_MASK\_PORTB, [229](#)
  - EXT\_INT\_DISABLE\_PIN\_CHANGE\_MASK\_PORTC, [229](#)
  - EXT\_INT\_DISABLE\_PIN\_CHANGE\_MASK\_PORTD, [229](#)
  - EXT\_INT\_ENABLE\_INT0, [229](#)
  - EXT\_INT\_ENABLE\_INT1, [230](#)
  - EXT\_INT\_ENABLE\_PIN\_CHANGE\_INTERRUPT, [230](#)
  - EXT\_INT\_ENABLE\_PIN\_CHANGE\_MASK\_PORTB, [230](#)
  - EXT\_INT\_ENABLE\_PIN\_CHANGE\_MASK\_PORTC, [230](#)
  - EXT\_INT\_ENABLE\_PIN\_CHANGE\_MASK\_PORTD, [230](#)
  - EXT\_INT\_INT0\_INTERRUPT, [230](#)
  - EXT\_INT\_INT1\_INTERRUPT, [231](#)
  - EXT\_INT\_PIN\_CHANGE\_PORTB\_INTERRUPT, [231](#)
  - EXT\_INT\_PIN\_CHANGE\_PORTC\_INTERRUPT, [231](#)
  - EXT\_INT\_PIN\_CHANGE\_PORTD\_INTERRUPT, [231](#)
  - EXT\_INT\_SET\_INT0\_SENSE\_CONTROL, [231](#)
  - EXT\_INT\_SET\_INT1\_SENSE\_CONTROL, [231](#)
  - MCU\_ADC\_DISABLE, [232](#)
  - MCU\_ADC\_ENABLE, [232](#)
  - MCU\_BOD\_DISABLE, [232](#)
  - MCU\_SELECT\_SLEEP\_MODE, [232](#)
  - MCU\_SLEEP\_DISABLE, [232](#)
  - MCU\_SLEEP\_ENABLE, [232](#)
  - MCU\_SPI\_DISABLE, [233](#)
  - MCU\_SPI\_ENABLE, [233](#)
  - MCU\_TIMER0\_DISABLE, [233](#)
  - MCU\_TIMER0\_ENABLE, [233](#)
  - MCU\_TIMER1\_DISABLE, [233](#)
  - MCU\_TIMER1\_ENABLE, [233](#)
  - MCU\_TIMER2\_DISABLE, [234](#)
  - MCU\_TIMER2\_ENABLE, [234](#)
  - MCU\_TWI\_DISABLE, [234](#)
  - MCU\_TWI\_ENABLE, [234](#)
  - MCU\_USART0\_DISABLE, [234](#)
  - MCU\_USART0\_ENABLE, [234](#)
  - PUSHBUTTON\_DEBOUNCE\_TIME\_US, [235](#)
  - PUSHBUTTON\_SAMPLING, [235](#)
  - SERVOMOTOR\_TIMER\_ANGLE\_COUNT, [235](#)
  - SERVOMOTOR\_TIMER\_PULSE\_WIDTH\_COUNT, [235](#)
  - SPI\_CONTROL\_REGISTER, [235](#)
  - SPI\_DATA\_REGISTER, [236](#)
  - SPI\_DISABLE, [236](#)
  - SPI\_DISABLE\_TRANSFER\_COMPLETE\_INTERRUPT, [236](#)

- SPI\_ENABLE, [236](#)
- SPI\_ENABLE\_TRANSFER\_COMPLETE\_INTERRUPT, [236](#)
- SPI\_MASTER\_MODE, [236](#)
- SPI\_SELECT\_CLOCK\_PRESCALER, [237](#)
- SPI\_SELECT\_DATA\_MODE, [237](#)
- SPI\_SELECT\_DATA\_ORDER, [237](#)
- SPI\_SELECT\_MASTER\_MODE, [237](#)
- SPI\_SELECT\_SLAVE\_MODE, [237](#)
- SPI\_STATUS\_REGISTER, [237](#)
- SPI\_TRANSFER\_COMPLETE, [238](#)
- SPI\_TRANSFER\_COMPLETE\_INTERRUPT, [238](#)
- SPI\_WRITE\_COLLISION, [238](#)
- TIMER0\_COM\_CHANNEL\_A\_INTERRUPT, [238](#)
- TIMER0\_COM\_CHANNEL\_B\_INTERRUPT, [238](#)
- TIMER0\_DISABLE\_COM\_CHANNEL\_A\_INTERRUPT, [238](#)
- TIMER0\_DISABLE\_COM\_CHANNEL\_B\_INTERRUPT, [239](#)
- TIMER0\_DISABLE\_OVERFLOW\_INTERRUPT, [239](#)
- TIMER0\_ENABLE\_COM\_CHANNEL\_A\_INTERRUPT, [239](#)
- TIMER0\_ENABLE\_COM\_CHANNEL\_B\_INTERRUPT, [239](#)
- TIMER0\_ENABLE\_OVERFLOW\_INTERRUPT, [239](#)
- TIMER0\_OVERFLOW\_INTERRUPT, [239](#)
- TIMER0\_SELECT\_CLOCK\_SOURCE, [240](#)
- TIMER0\_SELECT\_COM\_CHANNEL\_A, [240](#)
- TIMER0\_SELECT\_COM\_CHANNEL\_B, [240](#)
- TIMER0\_SELECT\_OPERATION\_MODE, [240](#)
- TIMER0\_STOP, [240](#)
- TIMER1\_COM\_CHANNEL\_A\_INTERRUPT, [240](#)
- TIMER1\_COM\_CHANNEL\_B\_INTERRUPT, [241](#)
- TIMER1\_DISABLE\_COM\_CHANNEL\_A\_INTERRUPT, [241](#)
- TIMER1\_DISABLE\_COM\_CHANNEL\_B\_INTERRUPT, [241](#)
- TIMER1\_DISABLE\_INPUT\_CAPTURE\_INTERRUPT, [241](#)
- TIMER1\_DISABLE\_OVERFLOW\_INTERRUPT, [241](#)
- TIMER1\_ENABLE\_COM\_CHANNEL\_A\_INTERRUPT, [241](#)
- TIMER1\_ENABLE\_COM\_CHANNEL\_B\_INTERRUPT, [242](#)
- TIMER1\_ENABLE\_INPUT\_CAPTURE\_INTERRUPT, [242](#)
- TIMER1\_ENABLE\_OVERFLOW\_INTERRUPT, [242](#)
- TIMER1\_INPUT\_CAPTURE\_INTERRUPT, [242](#)
- TIMER1\_OVERFLOW\_INTERRUPT, [242](#)
- TIMER1\_SELECT\_CLOCK\_SOURCE, [242](#)
- TIMER1\_SELECT\_COM\_CHANNEL\_A, [243](#)
- TIMER1\_SELECT\_COM\_CHANNEL\_B, [243](#)
- TIMER1\_SELECT\_OPERATION\_MODE, [243](#)
- TIMER1\_STOP, [243](#)
- TIMER2\_COM\_CHANNEL\_A\_INTERRUPT, [243](#)
- TIMER2\_COM\_CHANNEL\_B\_INTERRUPT, [243](#)
- TIMER2\_DISABLE\_COM\_CHANNEL\_A\_INTERRUPT, [244](#)
- TIMER2\_DISABLE\_COM\_CHANNEL\_B\_INTERRUPT, [244](#)
- TIMER2\_DISABLE\_OVERFLOW\_INTERRUPT, [244](#)
- TIMER2\_ENABLE\_COM\_CHANNEL\_A\_INTERRUPT, [244](#)
- TIMER2\_ENABLE\_COM\_CHANNEL\_B\_INTERRUPT, [244](#)
- TIMER2\_ENABLE\_OVERFLOW\_INTERRUPT, [244](#)
- TIMER2\_OVERFLOW\_INTERRUPT, [245](#)
- TIMER2\_SELECT\_CLOCK\_SOURCE, [245](#)
- TIMER2\_SELECT\_COM\_CHANNEL\_A, [245](#)
- TIMER2\_SELECT\_COM\_CHANNEL\_B, [245](#)
- TIMER2\_SELECT\_OPERATION\_MODE, [245](#)
- TIMER2\_STOP, [245](#)
- USART0\_CONTROL\_STATUS\_REGISTER, [246](#)
- USART0\_DATA\_OVERRUN, [246](#)
- USART0\_DATA\_REGISTER, [246](#)
- USART0\_DATA\_REGISTER\_EMPTY\_INTERRUPT, [246](#)
- USART0\_DISABLE\_DATA\_REGISTER\_EMPTY\_INTERRUPT, [246](#)
- USART0\_DISABLE\_DOUBLE\_SPEED\_MODE, [246](#), [247](#)
- USART0\_DISABLE\_PARITY\_MODE, [247](#)
- USART0\_DISABLE\_RECEIVE\_COMPLETE\_INTERRUPT, [247](#)
- USART0\_DISABLE\_RECEIVER, [247](#)
- USART0\_DISABLE\_TRANSMIT\_COMPLETE\_INTERRUPT, [247](#)
- USART0\_DISABLE\_TRANSMITTER, [247](#)
- USART0\_ENABLE\_ASYNC\_TRANSMISSION\_MODE, [248](#)
- USART0\_ENABLE\_DATA\_REGISTER\_EMPTY\_INTERRUPT, [248](#)
- USART0\_ENABLE\_DOUBLE\_SPEED\_MODE, [248](#)
- USART0\_ENABLE\_EVEN\_PARITY\_MODE, [248](#)
- USART0\_ENABLE\_MASTER\_SPI\_MODE, [248](#)
- USART0\_ENABLE\_ODD\_PARITY\_MODE, [248](#)
- USART0\_ENABLE\_RECEIVE\_COMPLETE\_INTERRUPT, [249](#)
- USART0\_ENABLE\_RECEIVER, [249](#)
- USART0\_ENABLE\_SYNC\_TRANSMISSION\_MODE, [249](#)
- USART0\_ENABLE\_TRANSMIT\_COMPLETE\_INTERRUPT, [249](#)
- USART0\_ENABLE\_TRANSMITTER, [249](#)
- USART0\_FRAME\_ERROR, [249](#)
- USART0\_PARITY\_ERROR, [250](#)
- USART0\_RECEIVE\_COMPLETE\_INTERRUPT, [250](#)
- USART0\_SET\_5BIT\_FRAME\_SIZE, [250](#)

- USART0\_SET\_6BIT\_FRAME\_SIZE, [250](#)
- USART0\_SET\_7BIT\_FRAME\_SIZE, [250](#)
- USART0\_SET\_8BIT\_FRAME\_SIZE, [251](#)
- USART0\_SET\_9BIT\_FRAME\_SIZE, [251](#)
- USART0\_SET\_BAUDRATE\_HIGH\_REGISTER, [251](#)
- USART0\_SET\_BAUDRATE\_LOW\_REGISTER, [251](#)
- USART0\_SET\_ONE\_STOP\_BIT, [251](#)
- USART0\_SET\_TWO\_STOP\_BITS, [252](#)
- USART0\_TRANSMIT\_COMPLETE\_INTERRUPT, [252](#)
- WATCHDOG\_SELECT\_TIMEOUT, [252](#)
- WATCHDOG\_START, [252](#)
- WATCHDOG\_STOP, [252](#)
- WATCHDOG\_TIMEOUT\_INTERRUPT, [252](#)
- halfStep
  - component, [28](#)
- Idle
  - core, [36](#)
- init
  - core::MCU, [79](#)
- inputCaptureServiceRoutine
  - core::TimerCounter1, [133](#)
- Int0ServiceRoutine
  - core::ExternInterrupt, [66](#)
- Int1ServiceRoutine
  - core::ExternInterrupt, [66](#)
- internal
  - core, [35](#)
- interrupt
  - core, [33](#), [34](#)
- interrupt\_reset
  - core, [33](#), [34](#)
- io, [37](#)
  - async, [42](#)
  - clockPrescaler, [38](#)
  - communicationMode, [39](#)
  - dataMode, [39](#)
  - dataOrder, [40](#)
  - disable, [41](#)
  - duplex, [39](#)
  - eightBits, [40](#)
  - evenParity, [41](#)
  - first\_LSB, [40](#)
  - first\_MSB, [40](#)
  - fiveBits, [40](#)
  - frameSize, [40](#)
  - master, [41](#)
  - masterSPI, [42](#)
  - mode\_0, [39](#)
  - mode\_1, [39](#)
  - mode\_2, [39](#)
  - mode\_3, [39](#)
  - neineBits, [40](#)
  - noParity, [41](#)
  - oddParity, [41](#)
  - oneStopBit, [42](#)
  - operationMode, [41](#)
  - parityMode, [41](#)
  - PortB, [43](#)
  - PortC, [43](#)
  - PortD, [43](#)
  - PS\_128, [38](#)
  - PS\_16, [38](#)
  - PS\_2, [38](#)
  - PS\_32, [38](#)
  - PS\_4, [38](#)
  - PS\_64, [38](#)
  - PS\_8, [38](#)
  - receive, [39](#)
  - sevenBits, [40](#)
  - sixBits, [40](#)
  - slave, [41](#)
  - stopBit, [41](#)
  - submaster, [41](#)
  - sync, [42](#)
  - transmissionMode, [42](#)
  - transmit, [39](#)
  - twoStopBits, [42](#)
- io::Pin, [80](#)
  - ~Pin, [82](#)
  - getPinNumber, [82](#)
  - isHigh, [82](#)
  - isLow, [83](#)
  - m\_pinNumber, [85](#)
  - mr\_portName, [85](#)
  - Pin, [81](#)
  - setHigh, [83](#)
  - setLow, [83](#)
  - toggle, [84](#)
  - toInput, [84](#)
  - toOutput, [84](#)
- io::Port, [85](#)
  - mp\_ddrReg, [86](#)
  - mp\_pinReg, [86](#)
  - mp\_portReg, [86](#)
- io::SPI, [97](#)
  - \_\_externally\_visible\_\_, [104](#)
  - \_\_used\_\_, [104](#)
  - ~SPI, [99](#)
  - enableTransferCompleteInterrupt, [99](#)
  - getInstance, [100](#)
  - m\_data, [104](#)
  - m\_pinMISO, [104](#)
  - m\_pinMOSI, [104](#)
  - m\_pinSCK, [105](#)
  - m\_pinSS, [105](#)
  - masterReceiveByte, [100](#)
  - masterSendByte, [100](#)
  - operator=, [101](#)
  - selectClockPrescaler, [101](#)
  - selectDataMode, [101](#)
  - selectDataOrder, [101](#)
  - selectOperationMode, [102](#)
  - selectSlave, [102](#)

- slaveReceiveByte, [103](#)
- SPI, [98](#), [99](#)
- transferComplete, [103](#)
- transferCompleteServiceRoutine, [103](#)
- writeCollision, [103](#)
- io::USART0, [149](#)
  - \_\_externally\_visible\_\_, [164](#)
  - \_\_used\_\_, [164](#)
  - ~USART0, [151](#)
  - dataOverrun, [151](#)
  - dataRegisterEmptyServiceRoutine, [152](#)
  - enableDataRegisterEmptyInterrupt, [152](#)
  - enableReceiveCompleteInterrupt, [153](#)
  - enableTransmitCompleteInterrupt, [153](#)
  - frameError, [154](#)
  - getInstance, [154](#)
  - getNumberBytesReceived, [154](#)
  - getNumberBytesSent, [155](#)
  - m\_numberBytesReceived, [164](#)
  - m\_numberBytesSent, [164](#)
  - m\_ready2Send, [165](#)
  - m\_sizeData2Receive, [165](#)
  - m\_sizeData2Send, [165](#)
  - m\_status, [165](#)
  - mp\_data2Receive, [165](#)
  - mp\_data2Send, [166](#)
  - operator=, [155](#)
  - parityError, [155](#)
  - ready2Send, [155](#)
  - receiveChar, [156](#)
  - receiveCompleteServiceRoutine, [156](#)
  - receiveFrame, [157](#)
  - receiveString, [157](#)
  - resetNumberBytesReceived, [157](#)
  - sendByte, [158](#)
  - sendChar, [158](#)
  - sendFrame, [159](#)
  - sendLong, [159](#)
  - sendString, [159](#)
  - sendWord, [160](#)
  - setBaudRate, [160](#)
  - setCommunicationMode, [161](#)
  - setFrameSize, [161](#)
  - setParityMode, [162](#)
  - setStopBit, [163](#)
  - setTransmissionMode, [163](#)
  - transmitCompleteServiceRoutine, [164](#)
  - USART0, [150](#), [151](#)
- isHigh
  - io::Pin, [82](#)
- isLow
  - io::Pin, [83](#)
- isOff
  - component::Led, [73](#)
- isOn
  - component::Led, [74](#)
- isPressed
  - component::PushButton, [89](#)
- LCD.cpp, [257](#)
- LCD.h, [257](#)
- LDR
  - component::LDR, [69](#)
- LDR.cpp, [257](#)
- LDR.h, [258](#), [259](#)
- LDRVD
  - LightMeter/main.cpp, [265](#)
- LED
  - BlinkLed/main.cpp, [263](#)
- Led
  - component::Led, [73](#)
- Led.cpp, [259](#), [260](#)
- Led.h, [260](#), [261](#)
- LED0
  - LightMeter/main.cpp, [265](#)
- LED1
  - LightMeter/main.cpp, [265](#)
- LED2
  - LightMeter/main.cpp, [266](#)
- LED3
  - LightMeter/main.cpp, [266](#)
- LED4
  - LightMeter/main.cpp, [266](#)
- LED5
  - LightMeter/main.cpp, [266](#)
- LED6
  - LightMeter/main.cpp, [266](#)
- LED7
  - LightMeter/main.cpp, [267](#)
- LightMeter/main.cpp
  - LDRVD, [265](#)
  - LED0, [265](#)
  - LED1, [265](#)
  - LED2, [266](#)
  - LED3, [266](#)
  - LED4, [266](#)
  - LED5, [266](#)
  - LED6, [266](#)
  - LED7, [267](#)
  - main, [267](#)
- logicalChange
  - core, [36](#)
- lowLevel
  - core, [36](#)
- m\_accelTime
  - component::StepperMotor, [113](#)
- m\_buttonPressed
  - component::PushButton, [90](#)
- m\_clockPrescaler
  - core::TimerCounter0, [127](#)
  - core::TimerCounter1, [138](#)
  - core::TimerCounter2, [148](#)
- m\_clockSource
  - core::TimerCounter0, [128](#)
  - core::TimerCounter1, [139](#)
  - core::TimerCounter2, [148](#)
- m\_constSpeedTime

- component::StepperMotor, [113](#)
- m\_conversionResult
  - core::ADConverter, [54](#)
- m\_currentPos
  - component::StepperMotor, [113](#)
- m\_data
  - io::SPI, [104](#)
- m\_decelTime
  - component::StepperMotor, [114](#)
- m\_goalReached
  - component::StepperMotor, [114](#)
- m\_numberBytesReceived
  - io::USART0, [164](#)
- m\_numberBytesSent
  - io::USART0, [164](#)
- m\_operationMode
  - core::WatchdogTimer, [170](#)
- m\_pin
  - component::Buzzer, [58](#)
  - component::DCMotor, [62](#)
  - component::LDR, [71](#)
  - component::Led, [75](#)
  - component::PushButton, [90](#)
  - component::ServoMotor, [96](#)
- m\_pinCoil1
  - component::StepperMotor, [114](#)
- m\_pinCoil2
  - component::StepperMotor, [114](#)
- m\_pinCoil3
  - component::StepperMotor, [114](#)
- m\_pinCoil4
  - component::StepperMotor, [115](#)
- m\_pinMISO
  - io::SPI, [104](#)
- m\_pinMOSI
  - io::SPI, [104](#)
- m\_pinNumber
  - io::Pin, [85](#)
- m\_pinSCK
  - io::SPI, [105](#)
- m\_pinSS
  - io::SPI, [105](#)
- m\_pulseCycle
  - component::ServoMotor, [96](#)
- m\_pulseWidthMax
  - component::ServoMotor, [96](#)
- m\_pulseWidthMid
  - component::ServoMotor, [96](#)
- m\_pulseWidthMin
  - component::ServoMotor, [96](#)
- m\_ready2Send
  - io::USART0, [165](#)
- m\_resolution
  - core::ADConverter, [54](#)
- m\_sizeData2Receive
  - io::USART0, [165](#)
- m\_sizeData2Send
  - io::USART0, [165](#)
- m\_status
  - io::USART0, [165](#)
- m\_timeOut
  - core::WatchdogTimer, [170](#)
- main
  - BlinkLed/main.cpp, [263](#)
  - LightMeter/main.cpp, [267](#)
  - MusicBox/main.cpp, [270](#)
  - Organ/main.cpp, [274](#)
- main.cpp, [262](#), [264](#), [268](#), [271](#), [272](#), [275](#)
- map
  - utils, [43](#)
- master
  - io, [41](#)
- masterReceiveByte
  - io::SPI, [100](#)
- masterSendByte
  - io::SPI, [100](#)
- masterSPI
  - io, [42](#)
- MCU.cpp, [276](#), [277](#)
- MCU.h, [278](#), [280](#)
- MCU\_ADC\_DISABLE
  - ha\_m328p.h, [232](#)
- MCU\_ADC\_ENABLE
  - ha\_m328p.h, [232](#)
- MCU\_BOD\_DISABLE
  - ha\_m328p.h, [232](#)
- MCU\_SELECT\_SLEEP\_MODE
  - ha\_m328p.h, [232](#)
- MCU\_SLEEP\_DISABLE
  - ha\_m328p.h, [232](#)
- MCU\_SLEEP\_ENABLE
  - ha\_m328p.h, [232](#)
- MCU\_SPI\_DISABLE
  - ha\_m328p.h, [233](#)
- MCU\_SPI\_ENABLE
  - ha\_m328p.h, [233](#)
- MCU\_TIMER0\_DISABLE
  - ha\_m328p.h, [233](#)
- MCU\_TIMER0\_ENABLE
  - ha\_m328p.h, [233](#)
- MCU\_TIMER1\_DISABLE
  - ha\_m328p.h, [233](#)
- MCU\_TIMER1\_ENABLE
  - ha\_m328p.h, [233](#)
- MCU\_TIMER2\_DISABLE
  - ha\_m328p.h, [234](#)
- MCU\_TIMER2\_ENABLE
  - ha\_m328p.h, [234](#)
- MCU\_TWI\_DISABLE
  - ha\_m328p.h, [234](#)
- MCU\_TWI\_ENABLE
  - ha\_m328p.h, [234](#)
- MCU\_USART0\_DISABLE
  - ha\_m328p.h, [234](#)
- MCU\_USART0\_ENABLE
  - ha\_m328p.h, [234](#)

- mode
  - component, 27
- mode\_0
  - io, 39
- mode\_1
  - io, 39
- mode\_2
  - io, 39
- mode\_3
  - io, 39
- mp\_data2Receive
  - io::USART0, 165
- mp\_data2Send
  - io::USART0, 166
- mp\_ddrReg
  - io::Port, 86
- mp\_pinReg
  - io::Port, 86
- mp\_portReg
  - io::Port, 86
- mr\_ADC
  - component::LDR, 71
- mr\_isActiveLow
  - component::PushButton, 90
- mr\_portName
  - io::Pin, 85
- mr\_useInternalPullUp
  - component::PushButton, 90
- MusicBox/main.cpp
  - BUZZER, 270
  - main, 270
  - PUSHBUTTON, 270
  - SONG\_LENGTH, 270
- neineBits
  - io, 40
- noClock
  - core, 32
- noParity
  - io, 41
- normal
  - core, 32, 33
- NOTE\_DURATION
  - Organ/main.cpp, 274
- oddParity
  - io, 41
- off
  - component::DCMotor, 61
  - component::Led, 74
  - component::ServoMotor, 94
- on
  - component::DCMotor, 61
  - component::Led, 74
  - component::ServoMotor, 95
- oneStopBit
  - io, 42
- operationMode
  - core, 32, 33
- io, 41
- operator=
  - core::ADConverter, 51
  - core::ExternInterrupt, 66
  - core::TimerCounter0, 123
  - core::TimerCounter1, 133
  - core::TimerCounter2, 144
  - core::WatchdogTimer, 168
  - io::SPI, 101
  - io::USART0, 155
- Organ/main.cpp
  - BUZZER, 274
  - main, 274
  - NOTE\_DURATION, 274
- outputCompareMatchAServiceRoutine
  - core::TimerCounter0, 123
  - core::TimerCounter1, 133
  - core::TimerCounter2, 144
- outputCompareMatchBServiceRoutine
  - core::TimerCounter0, 123
  - core::TimerCounter1, 134
  - core::TimerCounter2, 144
- overflowServiceRoutine
  - core::TimerCounter0, 124
  - core::TimerCounter1, 134
  - core::TimerCounter2, 144
- parityError
  - io::USART0, 155
- parityMode
  - io, 41
- pause
  - component::Buzzer, 58
- PCINTB
  - core, 34
- PCINTC
  - core, 34
- PCINTD
  - core, 34
- Pin
  - io::Pin, 81
- Pin.cpp, 281
- Pin.h, 282, 284
- pinChangePort
  - core, 34
- pinChangePortBServiceRoutine
  - core::ExternInterrupt, 66
- pinChangePortCServiceRoutine
  - core::ExternInterrupt, 67
- pinChangePortDServiceRoutine
  - core::ExternInterrupt, 67
- PortB
  - io, 43
- PortC
  - io, 43
- PortD
  - io, 43
- powerDown
  - core, 36

- powerSave
  - core, [36](#)
- PS\_1
  - core, [32](#)
- PS\_1024
  - core, [32](#)
- PS\_128
  - core, [31](#), [32](#)
  - io, [38](#)
- PS\_16
  - core, [31](#)
  - io, [38](#)
- PS\_2
  - core, [31](#)
  - io, [38](#)
- PS\_256
  - core, [32](#)
- PS\_32
  - core, [31](#), [32](#)
  - io, [38](#)
- PS\_4
  - core, [31](#)
  - io, [38](#)
- PS\_64
  - core, [31](#), [32](#)
  - io, [38](#)
- PS\_8
  - core, [31](#), [32](#)
  - io, [38](#)
- PUSHBUTTON
  - MusicBox/main.cpp, [270](#)
- PushButton
  - component::PushButton, [88](#)
- PushButton.cpp, [284](#), [285](#)
- PushButton.h, [285](#), [286](#)
- PUSHBUTTON\_DEBOUNCE\_TIME\_US
  - ha\_m328p.h, [235](#)
- PUSHBUTTON\_SAMPLING
  - ha\_m328p.h, [235](#)
- PWM\_PC
  - core, [33](#)
- PWM\_PC\_10bit
  - core, [33](#)
- PWM\_PC\_8bit
  - core, [33](#)
- PWM\_PC\_9bit
  - core, [33](#)
- PWM\_PC\_ICR
  - core, [33](#)
- PWM\_PC\_OCR
  - core, [33](#)
- PWM\_PFC\_ICR
  - core, [33](#)
- PWM\_PFC\_OCR
  - core, [33](#)
- README.md, [287](#)
- ready2Send
  - io::USART0, [155](#)
- receive
  - io, [39](#)
- receiveChar
  - io::USART0, [156](#)
- receiveCompleteServiceRoutine
  - io::USART0, [156](#)
- receiveFrame
  - io::USART0, [157](#)
- receiveString
  - io::USART0, [157](#)
- referenceVoltage
  - core, [34](#)
- res\_10bit
  - core, [35](#)
- res\_11bit
  - core, [35](#)
- res\_12bit
  - core, [35](#)
- res\_13bit
  - core, [35](#)
- res\_14bit
  - core, [35](#)
- res\_15bit
  - core, [35](#)
- res\_16bit
  - core, [35](#)
- res\_8bit
  - core, [35](#)
- res\_9bit
  - core, [35](#)
- reset
  - core, [33](#), [34](#)
  - core::WatchdogTimer, [168](#)
- resetNumberBytesReceived
  - io::USART0, [157](#)
- resetPressedCount
  - component::PushButton, [89](#)
- resolution
  - core, [35](#)
- risingEdge
  - core, [36](#)
- rotate
  - component::ServoMotor, [95](#)
- selectAnalogInput
  - core::ADConverter, [51](#)
- selectAutoTriggerSource
  - core::ADConverter, [52](#)
- selectClockPrescaler
  - core::ADConverter, [52](#)
  - io::SPI, [101](#)
- selectClockSource
  - core::TimerCounter, [117](#)
  - core::TimerCounter0, [124](#)
  - core::TimerCounter1, [134](#)
  - core::TimerCounter2, [144](#)
- selectCompareOutputMode
  - core::TimerCounter, [117](#)
  - core::TimerCounter0, [125](#)



- core::TimerCounter1, [135](#)
- core::TimerCounter2, [145](#)
- selectDataMode
  - io::SPI, [101](#)
- selectDataOrder
  - io::SPI, [101](#)
- selectOperationMode
  - core::TimerCounter, [117](#)
  - core::TimerCounter0, [125](#)
  - core::TimerCounter1, [135](#)
  - core::TimerCounter2, [146](#)
  - io::SPI, [102](#)
- selectReferenceVoltage
  - core::ADConverter, [53](#)
- selectSlave
  - io::SPI, [102](#)
- selectSleepMode
  - core::MCU, [79](#)
- selectTimeOut
  - core::WatchdogTimer, [168](#)
- sendByte
  - io::USART0, [158](#)
- sendChar
  - io::USART0, [158](#)
- sendFrame
  - io::USART0, [159](#)
- sendLong
  - io::USART0, [159](#)
- sendString
  - io::USART0, [159](#)
- sendWord
  - io::USART0, [160](#)
- senseControl
  - core, [35](#)
- ServoMotor
  - component::ServoMotor, [92](#)
- ServoMotor.cpp, [287](#), [288](#)
- ServoMotor.h, [289](#), [293](#)
- SERVOMOTOR\_TIMER\_ANGLE\_COUNT
  - ha\_m328p.h, [235](#)
- SERVOMOTOR\_TIMER\_PULSE\_WIDTH\_COUNT
  - ha\_m328p.h, [235](#)
- set
  - core, [32](#)
- setBaudRate
  - io::USART0, [160](#)
- setCommunicationMode
  - io::USART0, [161](#)
- setCounter
  - core::TimerCounter, [117](#)
  - core::TimerCounter0, [126](#)
  - core::TimerCounter1, [137](#)
  - core::TimerCounter2, [146](#)
- setCurrentPos
  - component::StepperMotor, [109](#)
- setFrameSize
  - io::USART0, [161](#)
- setHigh
  - io::Pin, [83](#)
- setInputCaptureRegister
  - core::TimerCounter1, [137](#)
- setInt0SenseControl
  - core::ExternInterrupt, [67](#)
- setInt1SenseControl
  - core::ExternInterrupt, [67](#)
- setLow
  - io::Pin, [83](#)
- setOutputCompareRegister
  - core::TimerCounter, [117](#)
  - core::TimerCounter0, [126](#)
  - core::TimerCounter1, [137](#)
  - core::TimerCounter2, [147](#)
- setParityMode
  - io::USART0, [162](#)
- setStopBit
  - io::USART0, [163](#)
- setTransmissionMode
  - io::USART0, [163](#)
- sevenBits
  - io, [40](#)
- sixBits
  - io, [40](#)
- slave
  - io, [41](#)
- slaveReceiveByte
  - io::SPI, [103](#)
- sleepEnable
  - core::MCU, [79](#)
- sleepMode
  - core, [36](#)
- SONG\_LENGTH
  - MusicBox/main.cpp, [270](#)
- SPI
  - io::SPI, [98](#), [99](#)
- SPI.cpp, [293](#), [294](#)
- SPI.h, [296](#), [298](#)
- SPI\_CONTROL\_REGISTER
  - ha\_m328p.h, [235](#)
- SPI\_DATA\_REGISTER
  - ha\_m328p.h, [236](#)
- SPI\_DISABLE
  - ha\_m328p.h, [236](#)
- SPI\_DISABLE\_TRANSFER\_COMPLETE\_INTERRUPT
  - ha\_m328p.h, [236](#)
- SPI\_ENABLE
  - ha\_m328p.h, [236](#)
- SPI\_ENABLE\_TRANSFER\_COMPLETE\_INTERRUPT
  - ha\_m328p.h, [236](#)
- SPI\_MASTER\_MODE
  - ha\_m328p.h, [236](#)
- SPI\_SELECT\_CLOCK\_PRESCALER
  - ha\_m328p.h, [237](#)
- SPI\_SELECT\_DATA\_MODE
  - ha\_m328p.h, [237](#)
- SPI\_SELECT\_DATA\_ORDER
  - ha\_m328p.h, [237](#)

- SPI\_SELECT\_MASTER\_MODE
  - ha\_m328p.h, [237](#)
- SPI\_SELECT\_SLAVE\_MODE
  - ha\_m328p.h, [237](#)
- SPI\_STATUS\_REGISTER
  - ha\_m328p.h, [237](#)
- SPI\_TRANSFER\_COMPLETE
  - ha\_m328p.h, [238](#)
- SPI\_TRANSFER\_COMPLETE\_INTERRUPT
  - ha\_m328p.h, [238](#)
- SPI\_WRITE\_COLLISION
  - ha\_m328p.h, [238](#)
- spin
  - component::DCMotor, [61](#)
- standby
  - core, [36](#)
- start
  - core::ADConverter, [53](#)
  - core::TimerCounter, [118](#)
  - core::TimerCounter0, [126](#)
  - core::TimerCounter1, [137](#)
  - core::TimerCounter2, [147](#)
  - core::WatchdogTimer, [168](#), [169](#)
- step
  - component::StepperMotor, [109–111](#)
- stepDelay
  - component::StepperMotor, [112](#)
- stepMode
  - component::StepperMotor, [115](#)
- StepperMotor
  - component::StepperMotor, [106](#)
- StepperMotor.cpp, [299](#)
- StepperMotor.h, [304](#), [305](#)
- stepPulse
  - component::StepperMotor, [112](#)
- stop
  - component::DCMotor, [61](#)
  - core::ADConverter, [53](#)
  - core::TimerCounter, [118](#)
  - core::TimerCounter0, [127](#)
  - core::TimerCounter1, [138](#)
  - core::TimerCounter2, [147](#)
  - core::WatchdogTimer, [169](#)
- stopBit
  - io, [41](#)
- STR
  - ha\_base.h, [221](#)
- STRx
  - ha\_base.h, [221](#)
- submaster
  - io, [41](#)
- sync
  - io, [42](#)
- TIMEDELAY
  - BlinkLed/main.cpp, [263](#)
- timeOut
  - core, [36](#)
- timeOutServiceRoutine
  - core::WatchdogTimer, [169](#)
- TIMER0\_COM\_CHANNEL\_A\_INTERRUPT
  - ha\_m328p.h, [238](#)
- TIMER0\_COM\_CHANNEL\_B\_INTERRUPT
  - ha\_m328p.h, [238](#)
- TIMER0\_DISABLE\_COM\_CHANNEL\_A\_INTERRUPT
  - ha\_m328p.h, [238](#)
- TIMER0\_DISABLE\_COM\_CHANNEL\_B\_INTERRUPT
  - ha\_m328p.h, [239](#)
- TIMER0\_DISABLE\_OVERFLOW\_INTERRUPT
  - ha\_m328p.h, [239](#)
- TIMER0\_ENABLE\_COM\_CHANNEL\_A\_INTERRUPT
  - ha\_m328p.h, [239](#)
- TIMER0\_ENABLE\_COM\_CHANNEL\_B\_INTERRUPT
  - ha\_m328p.h, [239](#)
- TIMER0\_ENABLE\_OVERFLOW\_INTERRUPT
  - ha\_m328p.h, [239](#)
- TIMER0\_OVERFLOW\_INTERRUPT
  - ha\_m328p.h, [239](#)
- TIMER0\_SELECT\_CLOCK\_SOURCE
  - ha\_m328p.h, [240](#)
- TIMER0\_SELECT\_COM\_CHANNEL\_A
  - ha\_m328p.h, [240](#)
- TIMER0\_SELECT\_COM\_CHANNEL\_B
  - ha\_m328p.h, [240](#)
- TIMER0\_SELECT\_OPERATION\_MODE
  - ha\_m328p.h, [240](#)
- TIMER0\_STOP
  - ha\_m328p.h, [240](#)
- timer0CompareA
  - core, [30](#)
- timer0Overflow
  - core, [30](#)
- TIMER1\_COM\_CHANNEL\_A\_INTERRUPT
  - ha\_m328p.h, [240](#)
- TIMER1\_COM\_CHANNEL\_B\_INTERRUPT
  - ha\_m328p.h, [241](#)
- TIMER1\_DISABLE\_COM\_CHANNEL\_A\_INTERRUPT
  - ha\_m328p.h, [241](#)
- TIMER1\_DISABLE\_COM\_CHANNEL\_B\_INTERRUPT
  - ha\_m328p.h, [241](#)
- TIMER1\_DISABLE\_INPUT\_CAPTURE\_INTERRUPT
  - ha\_m328p.h, [241](#)
- TIMER1\_DISABLE\_OVERFLOW\_INTERRUPT
  - ha\_m328p.h, [241](#)
- TIMER1\_ENABLE\_COM\_CHANNEL\_A\_INTERRUPT
  - ha\_m328p.h, [241](#)
- TIMER1\_ENABLE\_COM\_CHANNEL\_B\_INTERRUPT
  - ha\_m328p.h, [242](#)
- TIMER1\_ENABLE\_INPUT\_CAPTURE\_INTERRUPT
  - ha\_m328p.h, [242](#)
- TIMER1\_ENABLE\_OVERFLOW\_INTERRUPT
  - ha\_m328p.h, [242](#)
- TIMER1\_INPUT\_CAPTURE\_INTERRUPT
  - ha\_m328p.h, [242](#)
- TIMER1\_OVERFLOW\_INTERRUPT
  - ha\_m328p.h, [242](#)
- TIMER1\_SELECT\_CLOCK\_SOURCE

- ha\_m328p.h, [242](#)
- TIMER1\_SELECT\_COM\_CHANNEL\_A
  - ha\_m328p.h, [243](#)
- TIMER1\_SELECT\_COM\_CHANNEL\_B
  - ha\_m328p.h, [243](#)
- TIMER1\_SELECT\_OPERATION\_MODE
  - ha\_m328p.h, [243](#)
- TIMER1\_STOP
  - ha\_m328p.h, [243](#)
- timer1Capture
  - core, [30](#)
- timer1CompareB
  - core, [30](#)
- timer1Overflow
  - core, [30](#)
- TIMER2\_COM\_CHANNEL\_A\_INTERRUPT
  - ha\_m328p.h, [243](#)
- TIMER2\_COM\_CHANNEL\_B\_INTERRUPT
  - ha\_m328p.h, [243](#)
- TIMER2\_DISABLE\_COM\_CHANNEL\_A\_INTERRUPT
  - ha\_m328p.h, [244](#)
- TIMER2\_DISABLE\_COM\_CHANNEL\_B\_INTERRUPT
  - ha\_m328p.h, [244](#)
- TIMER2\_DISABLE\_OVERFLOW\_INTERRUPT
  - ha\_m328p.h, [244](#)
- TIMER2\_ENABLE\_COM\_CHANNEL\_A\_INTERRUPT
  - ha\_m328p.h, [244](#)
- TIMER2\_ENABLE\_COM\_CHANNEL\_B\_INTERRUPT
  - ha\_m328p.h, [244](#)
- TIMER2\_ENABLE\_OVERFLOW\_INTERRUPT
  - ha\_m328p.h, [244](#)
- TIMER2\_OVERFLOW\_INTERRUPT
  - ha\_m328p.h, [245](#)
- TIMER2\_SELECT\_CLOCK\_SOURCE
  - ha\_m328p.h, [245](#)
- TIMER2\_SELECT\_COM\_CHANNEL\_A
  - ha\_m328p.h, [245](#)
- TIMER2\_SELECT\_COM\_CHANNEL\_B
  - ha\_m328p.h, [245](#)
- TIMER2\_SELECT\_OPERATION\_MODE
  - ha\_m328p.h, [245](#)
- TIMER2\_STOP
  - ha\_m328p.h, [245](#)
- TimerCounter.h, [306](#), [316](#)
- TimerCounter0
  - core::TimerCounter0, [120](#)
- TimerCounter0.cpp, [318](#)
- TimerCounter0.h, [321](#), [322](#)
- TimerCounter1
  - core::TimerCounter1, [130](#)
- TimerCounter1.cpp, [323](#), [324](#)
- TimerCounter1.h, [328](#), [329](#)
- TimerCounter2
  - core::TimerCounter2, [141](#)
- TimerCounter2.cpp, [330](#)
- TimerCounter2.h, [334](#), [335](#)
- to\_125ms
  - core, [37](#)
- to\_16ms
  - core, [36](#)
- to\_1s
  - core, [37](#)
- to\_250ms
  - core, [37](#)
- to\_2s
  - core, [37](#)
- to\_32ms
  - core, [36](#)
- to\_4s
  - core, [37](#)
- to\_500ms
  - core, [37](#)
- to\_64ms
  - core, [37](#)
- to\_8s
  - core, [37](#)
- toggle
  - component::DCMotor, [62](#)
  - component::Led, [75](#)
  - component::ServoMotor, [95](#)
  - core, [32](#)
  - io::Pin, [84](#)
- toInput
  - io::Pin, [84](#)
- toOutput
  - io::Pin, [84](#)
- transferComplete
  - io::SPI, [103](#)
- transferCompleteServiceRoutine
  - io::SPI, [103](#)
- transmissionMode
  - io, [42](#)
- transmit
  - io, [39](#)
- transmitCompleteServiceRoutine
  - io::USART0, [164](#)
- twoStopBits
  - io, [42](#)
- USART0
  - io::USART0, [150](#), [151](#)
- USART0.cpp, [336](#)
- USART0.h, [341](#), [342](#)
- USART0\_CONTROL\_STATUS\_REGISTER
  - ha\_m328p.h, [246](#)
- USART0\_DATA\_OVERRUN
  - ha\_m328p.h, [246](#)
- USART0\_DATA\_REGISTER
  - ha\_m328p.h, [246](#)
- USART0\_DATA\_REGISTER\_EMPTY\_INTERRUPT
  - ha\_m328p.h, [246](#)
- USART0\_DISABLE\_DATA\_REGISTER\_EMPTY\_INTERRUPT
  - ha\_m328p.h, [246](#)
- USART0\_DISABLE\_DOUBLE\_SPEED\_MODE
  - ha\_m328p.h, [246](#), [247](#)
- USART0\_DISABLE\_PARITY\_MODE
  - ha\_m328p.h, [247](#)

USART0\_DISABLE\_RECEIVE\_COMPLETE\_INTERRUPT [Utils\\_m328p.cpp, 344](#)  
     [ha\\_m328p.h, 247](#)                      [utils\\_m328p.h, 344, 345](#)  
 USART0\_DISABLE\_RECEIVER  
     [ha\\_m328p.h, 247](#)                      WATCHDOG\_SELECT\_TIMEOUT  
                                             [ha\\_m328p.h, 252](#)  
 USART0\_DISABLE\_TRANSMIT\_COMPLETE\_INTERRUPT                      WATCHDOG\_START  
     [ha\\_m328p.h, 247](#)                      [ha\\_m328p.h, 252](#)  
 USART0\_DISABLE\_TRANSMITTER                      WATCHDOG\_STOP  
     [ha\\_m328p.h, 247](#)                      [ha\\_m328p.h, 252](#)  
 USART0\_ENABLE\_ASYNC\_TRANSMISSION\_MODE                      WATCHDOG\_TIMEOUT\_INTERRUPT  
     [ha\\_m328p.h, 248](#)                      [ha\\_m328p.h, 252](#)  
 USART0\_ENABLE\_DATA\_REGISTER\_EMPTY\_INTERRUPT                      WatchdogTimer  
     [ha\\_m328p.h, 248](#)                      [core::WatchdogTimer, 167](#)  
 USART0\_ENABLE\_DOUBLE\_SPEED\_MODE                      WatchdogTimer.cpp, [345](#)  
     [ha\\_m328p.h, 248](#)                      WatchdogTimer.h, [346, 349](#)  
 USART0\_ENABLE\_EVEN\_PARITY\_MODE                      writeCollision  
     [ha\\_m328p.h, 248](#)                      [io::SPI, 103](#)  
 USART0\_ENABLE\_MASTER\_SPI\_MODE  
     [ha\\_m328p.h, 248](#)  
 USART0\_ENABLE\_ODD\_PARITY\_MODE  
     [ha\\_m328p.h, 248](#)  
 USART0\_ENABLE\_RECEIVE\_COMPLETE\_INTERRUPT  
     [ha\\_m328p.h, 249](#)  
 USART0\_ENABLE\_RECEIVER  
     [ha\\_m328p.h, 249](#)  
 USART0\_ENABLE\_SYNC\_TRANSMISSION\_MODE  
     [ha\\_m328p.h, 249](#)  
 USART0\_ENABLE\_TRANSMIT\_COMPLETE\_INTERRUPT  
     [ha\\_m328p.h, 249](#)  
 USART0\_ENABLE\_TRANSMITTER  
     [ha\\_m328p.h, 249](#)  
 USART0\_FRAME\_ERROR  
     [ha\\_m328p.h, 249](#)  
 USART0\_PARITY\_ERROR  
     [ha\\_m328p.h, 250](#)  
 USART0\_RECEIVE\_COMPLETE\_INTERRUPT  
     [ha\\_m328p.h, 250](#)  
 USART0\_SET\_5BIT\_FRAME\_SIZE  
     [ha\\_m328p.h, 250](#)  
 USART0\_SET\_6BIT\_FRAME\_SIZE  
     [ha\\_m328p.h, 250](#)  
 USART0\_SET\_7BIT\_FRAME\_SIZE  
     [ha\\_m328p.h, 250](#)  
 USART0\_SET\_8BIT\_FRAME\_SIZE  
     [ha\\_m328p.h, 251](#)  
 USART0\_SET\_9BIT\_FRAME\_SIZE  
     [ha\\_m328p.h, 251](#)  
 USART0\_SET\_BAUDRATE\_HIGH\_REGISTER  
     [ha\\_m328p.h, 251](#)  
 USART0\_SET\_BAUDRATE\_LOW\_REGISTER  
     [ha\\_m328p.h, 251](#)  
 USART0\_SET\_ONE\_STOP\_BIT  
     [ha\\_m328p.h, 251](#)  
 USART0\_SET\_TWO\_STOP\_BITS  
     [ha\\_m328p.h, 252](#)  
 USART0\_TRANSMIT\_COMPLETE\_INTERRUPT  
     [ha\\_m328p.h, 252](#)  
 utils, [43](#)  
     map, [43](#)