G8R

Generated by Doxygen 1.9.1

1 Hierarchical Index **1**

1.1 Class Hierarchy 1

2 Class Index **3**

2.1 Class List 3

3 File Index **5**

3.1 File List 5

4 Class Documentation **7**

4.1 AnalogInputPin Class Reference 7

4.1.1 Constructor & Destructor Documentation 8

4.1.1.1 AnalogInputPin() 8

4.1.2 Member Function Documentation 8

4.1.2.1 begin() 9

4.1.2.2 read() 9

4.2 ClockState Struct Reference 9

4.2.1 Constructor & Destructor Documentation 9

4.2.1.1 ClockState() 10

4.2.2 Member Data Documentation 10

4.2.2.1 isRunning 10

4.2.2.2 lastTickTime 10

4.2.2.3 tickInterval 10

4.3 Debug Class Reference 10

4.3.1 Member Function Documentation 11

4.3.1.1 print() 11

4.3.2 Member Data Documentation 11

4.3.2.1 isEnabled 11

4.4 Encoder Class Reference 12

4.4.1 Member Enumeration Documentation 13

4.4.1.1 ButtonState 13

4.4.1.2 Direction 13

4.4.2 Constructor & Destructor Documentation 14

4.4.2.1 Encoder() 14

4.4.3 Member Function Documentation 14

4.4.3.1 begin() 14

4.4.3.2 handleEncoderDirection() 14

4.4.3.3 isButtonDoublePressed() 14

4.4.3.4 isButtonLongPressed() 15

4.4.3.5 isButtonSinglePressed() 15

4.4.3.6 readButton() 15

4.4.3.7 readEncoder() 15

4.4.3.8 readSpeed() 15

4.4.4 Member Data Documentation 15
4.4.4.1 buttonState 15
4.4.4.2 DOUBLE_PRESS_INTERVAL 15
4.4.4.3 encButton 16
4.4.4.4 encCLK 16
4.4.4.5 encDT 16
4.4.4.6 lastButtonPress 16
4.4.4.7 lastTurnTime 16
4.4.4.8 LONG_PRESS_INTERVAL 16
4.4.4.9 pressCount 16
4.4.4.10 prevStateCLK 16
4.4.4.11 speed 17
4.5 EurorackClock Class Reference 17
4.5.1 Constructor & Destructor Documentation 20
4.5.1.1 EurorackClock() 20
4.5.2 Member Function Documentation 20
4.5.2.1 decideFlash() 20
4.5.2.2 flashTempoLed() 20
4.5.2.3 getTempo() 20
4.5.2.4 handleExternalClock() 21
4.5.2.5 handleMidiClock() 21
4.5.2.6 handleResetTrigger() 21
4.5.2.7 handleTempoLed() 21
4.5.2.8 interruptHandler() 21
4.5.2.9 reset() [1/2] 21
4.5.2.10 reset() [2/2] 21
4.5.2.11 resetInterruptHandler() 22
4.5.2.12 setExternalTempo() 22
4.5.2.13 setPPQN() 22
4.5.2.14 setTempo() 22
4.5.2.15 setup() 22
4.5.2.16 shouldTriggerClockPulse() 22
4.5.2.17 start() 22
4.5.2.18 stop() 23
4.5.2.19 tick() 23
4.5.2.20 toggleLedOnDuration() 23
4.5.2.21 triggerClockPulse() 23
4.5.2.22 triggerGates() 23
4.5.2.23 triggerTempoLed() 23
4.5.2.24 updateFlashPulseCount() 23
4.5.2.25 updateTempoLed() 24
4.5.3 Member Data Documentation 24

4.5.3.1 clockPin 24
4.5.3.2 clockState 24
4.5.3.3 externalClock 24
4.5.3.4 externalTempo 24
4.5.3.5 flashPulseCount 24
4.5.3.6 gates 24
4.5.3.7 instance 25
4.5.3.8 isExternalTempo 25
4.5.3.9 isMidiClock 25
4.5.3.10 isRunning 25
4.5.3.11 lastClockState 25
4.5.3.12 lastClockTime 25
4.5.3.13 lastExternalTickTime 25
4.5.3.14 lastFlashTime 25
4.5.3.15 lastMidiClockTime 26
4.5.3.16 lastTickTime 26
4.5.3.17 LED_ON_DURATION 26
4.5.3.18 ledOnDuration 26
4.5.3.19 ledOnTime 26
4.5.3.20 leds 26
4.5.3.21 LONG_LED_ON_DURATION 26
4.5.3.22 MIDI_CLOCK_PULSE_COUNT 26
4.5.3.23 MIDI_CLOCK_TIMEOUT 27
4.5.3.24 ppqn 27
4.5.3.25 resetButton 27
4.5.3.26 resetPin 27
4.5.3.27 resetTriggered 27
4.5.3.28 tempo 27
4.5.3.29 tempoLed 27
4.5.3.30 tickCount 27
4.5.3.31 tickInterval 28
4.5.3.32 timer 28
4.5.3.33 timeToFlash 28
4.6 Gate Class Reference 28
4.6.1 Constructor & Destructor Documentation 31
4.6.1.1 Gate() 31
4.6.1.2 ∼Gate() 31
4.6.2 Member Function Documentation 31
4.6.2.1 getDivision() 31
4.6.2.2 setDivision() 31
4.6.2.3 setGateOnDuration() 31
4.6.2.4 trigger() 32

4.6.2.5 update() 32
4.6.3 Member Data Documentation 32
4.6.3.1 division 32
4.6.3.2 gateOnDuration 32
4.6.3.3 triggeredTime 32
4.7 Gates Class Reference 33
4.7.1 Constructor & Destructor Documentation 34
4.7.1.1 Gates() 34
4.7.1.2 ∼Gates() 34
4.7.2 Member Function Documentation 34
4.7.2.1 begin() 35
4.7.2.2 getDivision() 35
4.7.2.3 getSelectedGate() 35
4.7.2.4 getState() 35
4.7.2.5 setALLGates() 35
4.7.2.6 setDivision() 35
4.7.2.7 setGateOnDuration() 35
4.7.2.8 setSelectedGate() 36
4.7.2.9 setState() 36
4.7.2.10 trigger() 36
4.7.2.11 turnOffGate() 36
4.7.2.12 turnOnGate() 36
4.7.2.13 update() 36
4.7.3 Member Data Documentation 36
4.7.3.1 gateArray 37
4.7.3.2 gateCounters 37
4.7.3.3 numGates 37
4.7.3.4 selectedGate 37
4.8 InputHandler Class Reference 37
4.8.1 Constructor & Destructor Documentation 38
4.8.1.1 InputHandler() 39
4.8.2 Member Function Documentation 39
4.8.2.1 begin() 39
4.8.2.2 readCVA() 39
4.8.2.3 readCVB() 39
4.8.3 Member Data Documentation 39
4.8.3.1 cvA 39
4.8.3.2 cvB 39
4.9 InputPin Class Reference 40
4.9.1 Constructor & Destructor Documentation 41
4.9.1.1 InputPin() [1/2] 41
4.9.1.2 InputPin() [2/2] 42

4.9.2 Member Function Documentation 42
4.9.2.1 begin() 42
4.9.2.2 getState() 42
4.9.3 Member Data Documentation 42
4.9.3.1 useInternalPulldown 42
4.9.3.2 useInternalPullup 42
4.10 LED Class Reference 43
4.10.1 Constructor & Destructor Documentation 45
4.10.1.1 LED() 45
4.10.1.2 ∼LED() 45
4.10.2 Member Function Documentation 45
4.10.2.1 resetIvernted() 45
4.10.2.2 setIntensity() 46
4.10.2.3 setLedOnDuration() 46
4.10.2.4 startBlinking() 46
4.10.2.5 stopBlinking() 46
4.10.2.6 trigger() 46
4.10.2.7 update() 46
4.10.2.8 updateBlinking() 46
4.10.3 Member Data Documentation 47
4.10.3.1 blinkInterval 47
4.10.3.2 blinkStartTime 47
4.10.3.3 intensity 47
4.10.3.4 inverted 47
4.10.3.5 invertedLedOnDuration 47
4.10.3.6 isBlinking 47
4.10.3.7 ledOnDuration 47
4.10.3.8 triggeredTime 48
4.11 LEDController Class Reference 48
4.11.1 Constructor & Destructor Documentation 50
4.11.1.1 LEDController() 50
4.11.2 Member Function Documentation 50
4.11.2.1 blinkFast() 50
4.11.2.2 blinkFaster() 50
4.11.2.3 blinkSlow() 51
4.11.2.4 clearAndResetLEDs() 51
4.11.2.5 clearLEDs() 51
4.11.2.6 getNumLeds() 51
4.11.2.7 resetInverted() [1/2] 51
4.11.2.8 resetInverted() [2/2] 51
4.11.2.9 setState() 51
4.11.2.10 stopAllBlinking() 52

4.11.2.11 stopBlinking() 52
4.11.2.12 turnAllOff() 52
4.11.2.13 turnAllOn() 52
4.11.2.14 update() 52
4.11.2.15 updateBlinking() 52
4.11.3 Member Data Documentation 52
4.11.3.1 leds 52
4.11.3.2 numLeds 53
4.12 LEDs Class Reference 53
4.12.1 Constructor & Destructor Documentation 55
4.12.1.1 LEDs() 55
4.12.1.2 ∼LEDs() 55
4.12.2 Member Function Documentation 55
4.12.2.1 begin() 55
4.12.2.2 getState() 56
4.12.2.3 resetInverted() 56
4.12.2.4 setAllintensity() 56
4.12.2.5 setAllLeds() 56
4.12.2.6 setIntensity() 56
4.12.2.7 setState() [1/2] 56
4.12.2.8 setState() [2/2] 57
4.12.2.9 startBlinking() 57
4.12.2.10 stopAllBlinking() 57
4.12.2.11 stopBlinking() 57
4.12.2.12 trigger() 57
4.12.2.13 update() 57
4.12.2.14 updateBlinking() 57
4.12.3 Member Data Documentation 58
4.12.3.1 leds 58
4.12.3.2 numLeds 58
4.13 MIDIHandler Class Reference 58
4.13.1 Constructor & Destructor Documentation 60
4.13.1.1 MIDIHandler() 60
4.13.2 Member Function Documentation 60
4.13.2.1 begin() 60
4.13.2.2 handleClock() 61
4.13.2.3 handleContinue() 61
4.13.2.4 handleMidiMessage() 61
4.13.2.5 handleMode0NoteOff() 61
4.13.2.6 handleMode0NoteOn() 61
4.13.2.7 handleMode1NoteOff() 61
4.13.2.8 handleMode1NoteOn() 62

4.13.2.9 handleMode2NoteOff() 62
4.13.2.10 handleMode2NoteOn() 62
4.13.2.11 handleStart() 62
4.13.2.12 handleStop() 62
4.13.2.13 setChannel() 62
4.13.2.14 setMode() 63
4.13.3 Member Data Documentation 63
4.13.3.1 channel 63
4.13.3.2 clock 63
4.13.3.3 confirmedChannel 63
4.13.3.4 gates 63
4.13.3.5 instance 63
4.13.3.6 leds 63
4.13.3.7 midi 64
4.13.3.8 midiSerial 64
4.14 Mode Class Reference 64
4.14.1 Member Function Documentation 65
4.14.1.1 handleDoublePress() 66
4.14.1.2 handleLongPress() 66
4.14.1.3 handlePressReleased() 66
4.14.1.4 handleResetDoublePress() 66
4.14.1.5 handleResetLongPress() 66
4.14.1.6 handleResetPressReleased() 66
4.14.1.7 handleResetSinglePress() 67
4.14.1.8 handleSelectionStates() 67
4.14.1.9 handleSinglePress() 67
4.14.1.10 setup() 67
4.14.1.11 teardown() 67
4.14.1.12 update() 67
4.15 Mode0 Class Reference 68
4.15.1 Constructor & Destructor Documentation 70
4.15.1.1 Mode0() 71
4.15.2 Member Function Documentation 71
4.15.2.1 handleButton() 71
4.15.2.2 handleCVInput() 71
4.15.2.3 handleDoublePress() 71
4.15.2.4 handleEncoder() 71
4.15.2.5 handleLongPress() 71
4.15.2.6 handlePressReleased() 72
4.15.2.7 handleResetButton() 72
4.15.2.8 handleResetDoublePress() 72
4.15.2.9 handleResetLongPress() 72

4.15.2.10 handleResetPressReleased() 72
4.15.2.11 handleResetSinglePress() 72
4.15.2.12 handleSelectionStates() 73
4.15.2.13 handleSinglePress() 73
4.15.2.14 handleTempoSelection() 73
4.15.2.15 setDefaultDivisionIndex() 73
4.15.2.16 setDivisionPPQN() 73
4.15.2.17 setup() 73
4.15.2.18 teardown() 73
4.15.2.19 update() 74
4.15.3 Member Data Documentation 74
4.15.3.1 clock 74
4.15.3.2 divisionIndex 74
4.15.3.3 doublePressHandled 74
4.15.3.4 doubleResetPressHandled 74
4.15.3.5 encoder 74
4.15.3.6 externalTempo 74
4.15.3.7 gates 75
4.15.3.8 inDivisionSelection 75
4.15.3.9 inputHandler 75
4.15.3.10 ledController 75
4.15.3.11 maxTempo 75
4.15.3.12 midiHandler 75
4.15.3.13 minTempo 75
4.15.3.14 resetButton 75
4.15.3.15 selectedGate 76
4.15.3.16 selectingTempo 76
4.15.3.17 singlePressHandled 76
4.15.3.18 singleResetPressHandled 76
4.15.3.19 tempoIncrement 76
4.16 Mode1 Class Reference 77
4.16.1 Constructor & Destructor Documentation 79
4.16.1.1 Mode1() 79
4.16.2 Member Function Documentation 80
4.16.2.1 handleButton() 80
4.16.2.2 handleChannelSelection() 80
4.16.2.3 handleChannelSelectionPress() 80
4.16.2.4 handleDoublePress() 80
4.16.2.5 handleEncoder() 80
4.16.2.6 handleLongPress() 80
4.16.2.7 handlePressReleased() 81
4.16.2.8 handleResetDoublePress() 81

4.16.2.9 handleResetLongPress() 81
4.16.2.10 handleResetPressReleased() 81
4.16.2.11 handleResetSinglePress() 81
4.16.2.12 handleSelectionStates() 81
4.16.2.13 handleSinglePress() 82
4.16.2.14 setup() 82
4.16.2.15 teardown() 82
4.16.2.16 update() 82
4.16.3 Member Data Documentation 82
4.16.3.1 confirmedChannel 82
4.16.3.2 doublePressHandled 82
4.16.3.3 encoder 83
4.16.3.4 gates 83
4.16.3.5 inChannelSelection 83
4.16.3.6 inputHandler 83
4.16.3.7 isInSelection 83
4.16.3.8 ledController 83
4.16.3.9 midiHandler 83
4.16.3.10 NUM_MIDI_CHANNELS 83
4.16.3.11 numLeds 84
4.16.3.12 previousChannel 84
4.16.3.13 resetButton 84
4.16.3.14 selectedChannel 84
4.16.3.15 singlePressHandled 84
4.17 Mode2 Class Reference 85
4.17.1 Constructor & Destructor Documentation 87
4.17.1.1 Mode2() 87
4.17.2 Member Function Documentation 87
4.17.2.1 handleButton() 88
4.17.2.2 handleDoublePress() 88
4.17.2.3 handleEncoder() 88
4.17.2.4 handleLongPress() 88
4.17.2.5 handlePressReleased() 88
4.17.2.6 handleResetDoublePress() 88
4.17.2.7 handleResetLongPress() 89
4.17.2.8 handleResetPressReleased() 89
4.17.2.9 handleResetSinglePress() 89
4.17.2.10 handleSelectionStates() 89
4.17.2.11 handleSinglePress() 89
4.17.2.12 setup() 89
4.17.2.13 teardown() 90
4.17.2.14 update() 90

4.17.3 Member Data Documentation 90
4.17.3.1 doublePressHandled 90
4.17.3.2 encoder 90
4.17.3.3 gates 90
4.17.3.4 inputHandler 90
4.17.3.5 isInSelection 90
4.17.3.6 ledController 91
4.17.3.7 midiHandler 91
4.17.3.8 numLeds 91
4.17.3.9 resetButton 91
4.17.3.10 singlePressHandled 91
4.18 ModeSelector Class Reference 92
4.18.1 Constructor & Destructor Documentation 93
4.18.1.1 ModeSelector() [1/2] 93
4.18.1.2 ModeSelector() [2/2] 94
4.18.2 Member Function Documentation 94
4.18.2.1 addMode() 94
4.18.2.2 getCurrentMode() 94
4.18.2.3 getInstance() 94
4.18.2.4 getMode() 94
4.18.2.5 handleButtonPress() 94
4.18.2.6 handleEncoderRotation() 94
4.18.2.7 handleLongPress() 95
4.18.2.8 handleModeSelectionPress() 95
4.18.2.9 isInModeSelection() 95
4.18.2.10 operator=() 95
4.18.2.11 setCurrentMode() 95
4.18.2.12 setEncoder() 95
4.18.2.13 setLedController() 95
4.18.2.14 setMode() 96
4.18.2.15 update() 96
4.18.3 Member Data Documentation 96
4.18.3.1 currentMode 96
4.18.3.2 doublePressHandled 96
4.18.3.3 encoder 96
4.18.3.4 inChannelSelection 96
4.18.3.5 inModeSelection 96
4.18.3.6 isInSelection 97
4.18.3.7 ledController 97
4.18.3.8 longPressHandled 97
4.18.3.9 mode 97
4.18.3.10 modes 97

4.18.3.11 nullMode . . . 97
4.18.3.12 numLeds . . . 97
4.18.3.13 previousMode . . . 97
4.18.3.14 resetButton . . . 98
4.18.3.15 singlePressHandled . . . 98
4.18.3.16 totalModes . . . 98
4.19 OutputPin Class Reference . . . 98
4.19.1 Constructor & Destructor Documentation . . . 100
4.19.1.1 OutputPin() . . . 100
4.19.2 Member Function Documentation . . . 100
4.19.2.1 begin() . . . 101
4.19.2.2 getState() . . . 101
4.19.2.3 setState() . . . 101
4.20 Pin Class Reference . . . 101
4.20.1 Constructor & Destructor Documentation . . . 102
4.20.1.1 Pin() . . . 102
4.20.1.2 ∼Pin() . . . 102
4.20.2 Member Data Documentation . . . 102
4.20.2.1 pin . . . 103
4.20.2.2 state . . . 103
4.21 PWMPin Class Reference . . . 103
4.21.1 Constructor & Destructor Documentation . . . 106
4.21.1.1 PWMPin() . . . 106
4.21.2 Member Function Documentation . . . 106
4.21.2.1 begin() . . . 106
4.21.2.2 getDutyCycle() . . . 106
4.21.2.3 setDutyCycle() . . . 106
4.21.3 Member Data Documentation . . . 106
4.21.3.1 dutyCycle . . . 106
4.21.3.2 timer . . . 107
4.22 ResetButton Class Reference . . . 107
4.22.1 Member Enumeration Documentation . . . 109
4.22.1.1 ButtonState . . . 109
4.22.2 Constructor & Destructor Documentation . . . 109
4.22.2.1 ResetButton() . . . 109
4.22.3 Member Function Documentation . . . 109
4.22.3.1 begin() . . . 110
4.22.3.2 isButtonDoublePressed() . . . 110
4.22.3.3 isButtonLongPressed() . . . 110
4.22.3.4 isButtonSinglePressed() . . . 110
4.22.3.5 readButton() . . . 110
4.22.4 Member Data Documentation . . . 110

4.22.4.1 buttonState 110
4.22.4.2 DOUBLE_PRESS_INTERVAL 110
4.22.4.3 lastButtonPress 111
4.22.4.4 LONG_PRESS_INTERVAL 111
4.22.4.5 pin 111
4.22.4.6 pressCount 111
4.22.4.7 resetButton 111
4.23 SPDTSwitch Class Reference 112
4.23.1 Constructor & Destructor Documentation 113
4.23.1.1 SPDTSwitch() 113
4.23.2 Member Function Documentation 113
4.23.2.1 begin() 113
4.23.2.2 read() 113
4.23.3 Member Data Documentation 113
4.23.3.1 pinA 113
4.23.3.2 pinB 113

5 File Documentation **115**

5.1 include/Constants.h File Reference 115
5.1.1 Variable Documentation 116
5.1.1.1 internalPPQN 116
5.1.1.2 musicalIntervals 116
5.1.1.3 musicalIntervalsSize 116
5.2 include/Debug.h File Reference 116
5.3 include/Encoder.h File Reference 117
5.4 include/EurorackClock.h File Reference 117
5.5 include/Gate.h File Reference 118
5.6 include/Gates.h File Reference 119
5.7 include/InputHandler.h File Reference 120
5.8 include/LED.h File Reference 121
5.9 include/LEDController.h File Reference 122
5.10 include/LEDs.h File Reference 123
5.11 include/MIDIHandler.h File Reference 124
5.12 include/Mode.h File Reference 125
5.13 include/Mode0.h File Reference 125
5.14 include/Mode1.h File Reference 126
5.15 include/Mode2.h File Reference 127
5.16 include/ModeSelector.h File Reference 128
5.17 include/Pin.h File Reference 129
5.18 include/ResetButton.h File Reference 129
5.19 include/SPDTSwitch.h File Reference 130
5.19.1 Enumeration Type Documentation 131

5.19.1.1 SwitchState . . . 131
5.20 src/Debug.cpp File Reference . . . 131
5.21 src/Encoder.cpp File Reference . . . 132
5.21.1 Macro Definition Documentation . . . 133
5.21.1.1 DEBUG_PRINT . . . 133
5.22 src/EurorackClock.cpp File Reference . . . 133
5.22.1 Macro Definition Documentation . . . 133
5.22.1.1 DEBUG_PRINT . . . 133
5.23 src/Gate.cpp File Reference . . . 134
5.23.1 Macro Definition Documentation . . . 134
5.23.1.1 DEBUG_PRINT . . . 134
5.24 src/Gates.cpp File Reference . . . 135
5.24.1 Macro Definition Documentation . . . 135
5.24.1.1 DEBUG_PRINT . . . 135
5.25 src/InputHandler.cpp File Reference . . . 136
5.25.1 Macro Definition Documentation . . . 136
5.25.1.1 DEBUG_PRINT . . . 136
5.26 src/LED.cpp File Reference . . . 136
5.26.1 Macro Definition Documentation . . . 137
5.26.1.1 DEBUG_PRINT . . . 137
5.27 src/LEDController.cpp File Reference . . . 138
5.28 src/LEDs.cpp File Reference . . . 138
5.28.1 Macro Definition Documentation . . . 139
5.28.1.1 DEBUG_PRINT . . . 139
5.29 src/main.cpp File Reference . . . 139
5.29.1 Macro Definition Documentation . . . 141
5.29.1.1 CLOCK_PIN . . . 141
5.29.1.2 CV_A_PIN . . . 141
5.29.1.3 CV_B_PIN . . . 141
5.29.1.4 DEBUG_PRINT . . . 142
5.29.1.5 ENCODER_BUTTON . . . 142
5.29.1.6 ENCODER_PINA . . . 142
5.29.1.7 ENCODER_PINB . . . 142
5.29.1.8 RESET_BUTTON . . . 142
5.29.1.9 RESET_PIN . . . 142
5.29.1.10 RX_PIN . . . 142
5.29.1.11 TEMPO_LED . . . 143
5.29.1.12 TX_PIN . . . 143
5.29.2 Function Documentation . . . 143
5.29.2.1 loop() . . . 143
5.29.2.2 setup() . . . 143
5.29.3 Variable Documentation . . . 143

5.29.3.1 clock 143
5.29.3.2 currentMode 143
5.29.3.3 encButtonPin 144
5.29.3.4 encCLKPin 144
5.29.3.5 encDTPin 144
5.29.3.6 encoder 144
5.29.3.7 gates 144
5.29.3.8 inModeSelection 144
5.29.3.9 inputHandler 144
5.29.3.10 intensity 144
5.29.3.11 intensity_step 145
5.29.3.12 internalPPQN 145
5.29.3.13 isInSelection 145
5.29.3.14 lastFlashTime 145
5.29.3.15 ledController 145
5.29.3.16 ledPins 145
5.29.3.17 leds 145
5.29.3.18 midiHandler 146
5.29.3.19 min_intensity 146
5.29.3.20 mode0 146
5.29.3.21 mode1 146
5.29.3.22 mode2 146
5.29.3.23 modeSelector 147
5.29.3.24 musicalIntervals 147
5.29.3.25 musicalIntervalsSize 147
5.29.3.26 numLedPins 147
5.29.3.27 numPins 147
5.29.3.28 pins 147
5.29.3.29 resetButton 147
5.29.3.30 total_pages 147
5.30 src/MIDIHandler.cpp File Reference 148
5.30.1 Macro Definition Documentation 148
5.30.1.1 DEBUG_PRINT 148
5.30.2 Variable Documentation 148
5.30.2.1 isInSelection 149
5.31 src/Mode.cpp File Reference 149
5.32 src/Mode0.cpp File Reference 149
5.32.1 Macro Definition Documentation 149
5.32.1.1 DEBUG_PRINT 149
5.33 src/Mode1.cpp File Reference 150
5.33.1 Macro Definition Documentation 150
5.33.1.1 DEBUG_PRINT 150

5.34 src/Mode2.cpp File Reference 150
5.34.1 Macro Definition Documentation 151
5.34.1.1 DEBUG_PRINT 151
5.35 src/ModeSelector.cpp File Reference 151
5.35.1 Macro Definition Documentation 152
5.35.1.1 DEBUG_PRINT 152
5.36 src/Pin.cpp File Reference 152
5.36.1 Macro Definition Documentation 153
5.36.1.1 DEBUG_PRINT 153
5.37 src/ResetButton.cpp File Reference 154
5.38 src/SPDTSwitch.cpp File Reference 154

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

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

- ClockState 9
- Debug 10
- Encoder 12
- EurorackClock 17
- Gates 33
- InputHandler 37
- LEDController 48
- LEDs 53
- MIDIHandler 58
- Mode 64
  - Mode0 68
  - Mode1 77
  - Mode2 85
- ModeSelector 92
- Pin 101
  - AnalogInputPin 7
  - InputPin 40
  - OutputPin 98
    - Gate 28
    - LED 43
    - PWMPin 103
- ResetButton 107
- SPDTSwitch 112

Chapter 2

Class Index

2.1 Class List

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

AnalogInputPin 7
ClockState 9
Debug 10
Encoder 12
EurorackClock 17
Gate 28
Gates 33
InputHandler 37
InputPin 40
LED 43
LEDController 48
LEDs 53
MIDIHandler 58
Mode 64
Mode0 68
Mode1 77
Mode2 85
ModeSelector 92
OutputPin 98
Pin 101
PWMPin 103
ResetButton 107
SPDTSwitch 112

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

include/Constants.h . . . 115
include/Debug.h . . . 116
include/Encoder.h . . . 117
include/EurorackClock.h . . . 117
include/Gate.h . . . 118
include/Gates.h . . . 119
include/InputHandler.h . . . 120
include/LED.h . . . 121
include/LEDController.h . . . 122
include/LEDs.h . . . 123
include/MIDIHandler.h . . . 124
include/Mode.h . . . 125
include/Mode0.h . . . 125
include/Mode1.h . . . 126
include/Mode2.h . . . 127
include/ModeSelector.h . . . 128
include/Pin.h . . . 129
include/ResetButton.h . . . 129
include/SPDTSwitch.h . . . 130
src/Debug.cpp . . . 131
src/Encoder.cpp . . . 132
src/EurorackClock.cpp . . . 133
src/Gate.cpp . . . 134
src/Gates.cpp . . . 135
src/InputHandler.cpp . . . 136
src/LED.cpp . . . 136
src/LEDController.cpp . . . 138
src/LEDs.cpp . . . 138
src/main.cpp . . . 139
src/MIDIHandler.cpp . . . 148
src/Mode.cpp . . . 149
src/Mode0.cpp . . . 149
src/Mode1.cpp . . . 150
src/Mode2.cpp . . . 150
src/ModeSelector.cpp . . . 151
src/Pin.cpp . . . 152
src/ResetButton.cpp . . . 154
src/SPDTSwitch.cpp . . . 154

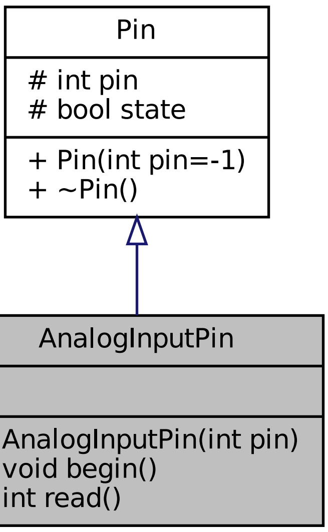
Chapter 4

Class Documentation

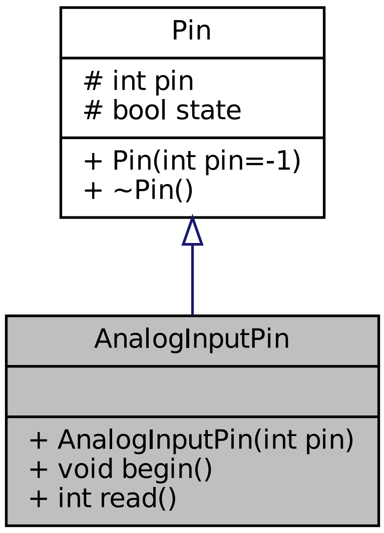
4.1 AnalogInputPin Class Reference

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

Inheritance diagram for AnalogInputPin:



Collaboration diagram for AnalogInputPin:



Public Member Functions

- AnalogInputPin (int pin)
- void begin ()
- int read ()

Additional Inherited Members

4.1.1 Constructor & Destructor Documentation

4.1.1.1 AnalogInputPin()

```
AnalogInputPin::AnalogInputPin (
              int pin )
```

4.1.2 Member Function Documentation

4.1.2.1 begin()

```
void AnalogInputPin::begin ( )
```

4.1.2.2 read()

```
int AnalogInputPin::read ( )
```

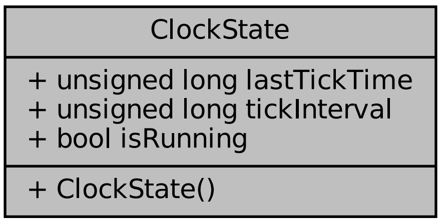
The documentation for this class was generated from the following files:

- include/Pin.h
- src/Pin.cpp

4.2 ClockState Struct Reference

```
#include <EurorackClock.h>
```

Collaboration diagram for ClockState:



Public Member Functions

- ClockState ()

Public Attributes

- unsigned long lastTickTime
- unsigned long tickInterval
- bool isRunning

4.2.1 Constructor & Destructor Documentation

4.2.1.1 ClockState()

```
ClockState::ClockState ( )  [inline]
```

4.2.2 Member Data Documentation

4.2.2.1 isRunning

```
bool ClockState::isRunning
```

4.2.2.2 lastTickTime

```
unsigned long ClockState::lastTickTime
```

4.2.2.3 tickInterval

```
unsigned long ClockState::tickInterval
```

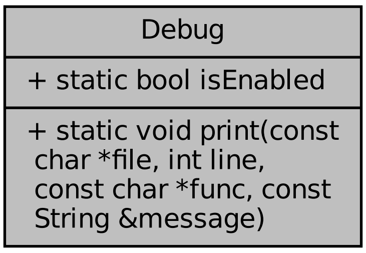
The documentation for this struct was generated from the following file:

- include/EurorackClock.h

4.3 Debug Class Reference

```
#include <Debug.h>
```

Collaboration diagram for Debug:


Debug
+ static bool isEnabled
+ static void print(const char *file, int line, const char *func, const String &message)

Static Public Member Functions

- static void print (const char ∗file, int line, const char ∗func, const String &message)

Static Public Attributes

- static bool isEnabled = false

4.3.1 Member Function Documentation

4.3.1.1 print()

```
void Debug::print (
             const char * file,
             int line,
             const char * func,
             const String & message )  [static]
```

4.3.2 Member Data Documentation

4.3.2.1 isEnabled

```
bool Debug::isEnabled = false  [static]
```

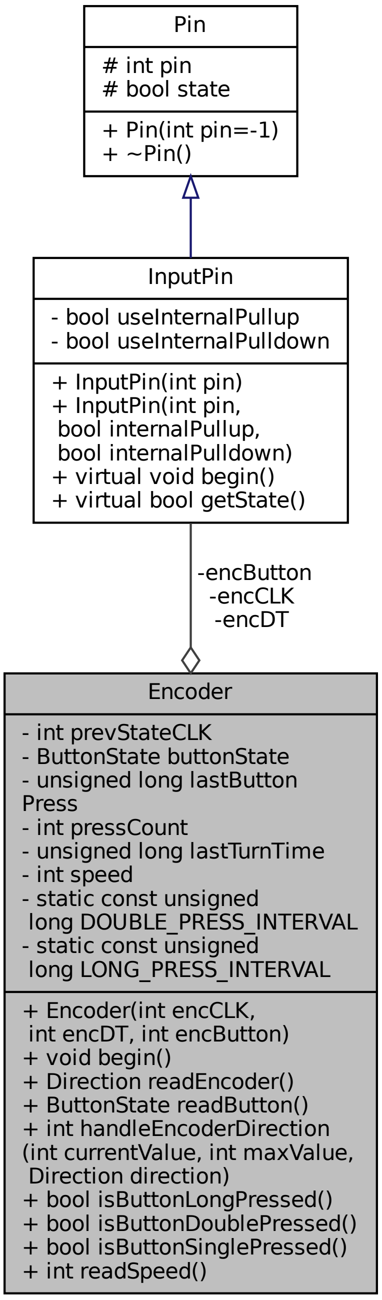
The documentation for this class was generated from the following files:

- include/Debug.h
- src/Debug.cpp

4.4 Encoder Class Reference

```
#include <Encoder.h>
```

Collaboration diagram for Encoder:


Pin
# int pin
# bool state
+ Pin(int pin=-1)
+ ~Pin()
InputPin
- bool useInternalPullup
- bool useInternalPulldown
+ InputPin(int pin)
+ InputPin(int pin, bool internalPullup, bool internalPulldown)
+ virtual void begin()
+ virtual bool getState()
-encButton
-encCLK
-encDT
Encoder
- int prevStateCLK
- ButtonState buttonState
- unsigned long lastButtonPress
- int pressCount
- unsigned long lastTurnTime
- int speed
- static const unsigned long DOUBLE_PRESS_INTERVAL
- static const unsigned long LONG_PRESS_INTERVAL
+ Encoder(int encCLK, int encDT, int encButton)
+ void begin()
+ Direction readEncoder()
+ ButtonState readButton()
+ int handleEncoderDirection(int currentValue, int maxValue, Direction direction)
+ bool isButtonLongPressed()
+ bool isButtonDoublePressed()
+ bool isButtonSinglePressed()
+ int readSpeed()

Public Types

- enum Direction { NONE , CW , CCW }
- enum ButtonState { OPEN , PRESSED }

Public Member Functions

- Encoder (int encCLK, int encDT, int encButton)
- void begin ()
- Direction readEncoder ()
- ButtonState readButton ()
- int handleEncoderDirection (int currentValue, int maxValue, Direction direction)
- bool isButtonLongPressed ()
- bool isButtonDoublePressed ()
- bool isButtonSinglePressed ()
- int readSpeed ()

Private Attributes

- InputPin encCLK
- InputPin encDT
- InputPin encButton
- int prevStateCLK
- ButtonState buttonState
- unsigned long lastButtonPress
- int pressCount
- unsigned long lastTurnTime
- int speed

Static Private Attributes

- static const unsigned long DOUBLE_PRESS_INTERVAL = 500
- static const unsigned long LONG_PRESS_INTERVAL = 1000

4.4.1 Member Enumeration Documentation

4.4.1.1 ButtonState

`enum Encoder::ButtonState`

Enumerator

OPEN	
PRESSED	

4.4.1.2 Direction

`enum Encoder::Direction`

Enumerator

NONE	
CW	
CCW	

4.4.2 Constructor & Destructor Documentation

4.4.2.1 Encoder()

```
Encoder::Encoder (
             int encCLK,
             int encDT,
             int encButton )
```

4.4.3 Member Function Documentation

4.4.3.1 begin()

```
void Encoder::begin ( )
```

4.4.3.2 handleEncoderDirection()

```
int Encoder::handleEncoderDirection (
             int currentValue,
             int maxValue,
             Direction direction )
```

4.4.3.3 isButtonDoublePressed()

```
bool Encoder::isButtonDoublePressed ( )
```

4.4.3.4 isButtonLongPressed()

```
bool Encoder::isButtonLongPressed ( )
```

4.4.3.5 isButtonSinglePressed()

```
bool Encoder::isButtonSinglePressed ( )
```

4.4.3.6 readButton()

```
Encoder::ButtonState Encoder::readButton ( )
```

4.4.3.7 readEncoder()

```
Encoder::Direction Encoder::readEncoder ( )
```

4.4.3.8 readSpeed()

```
int Encoder::readSpeed ( )
```

4.4.4 Member Data Documentation

4.4.4.1 buttonState

```
ButtonState Encoder::buttonState  [private]
```

4.4.4.2 DOUBLE_PRESS_INTERVAL

```
const unsigned long Encoder::DOUBLE_PRESS_INTERVAL = 500  [static], [private]
```

4.4.4.3 encButton

```
InputPin Encoder::encButton  [private]
```

4.4.4.4 encCLK

```
InputPin Encoder::encCLK  [private]
```

4.4.4.5 encDT

```
InputPin Encoder::encDT  [private]
```

4.4.4.6 lastButtonPress

```
unsigned long Encoder::lastButtonPress  [private]
```

4.4.4.7 lastTurnTime

```
unsigned long Encoder::lastTurnTime  [private]
```

4.4.4.8 LONG_PRESS_INTERVAL

```
const unsigned long Encoder::LONG_PRESS_INTERVAL = 1000  [static], [private]
```

4.4.4.9 pressCount

```
int Encoder::pressCount  [private]
```

4.4.4.10 prevStateCLK

```
int Encoder::prevStateCLK  [private]
```

4.4.4.11 speed

```
int Encoder::speed  [private]
```

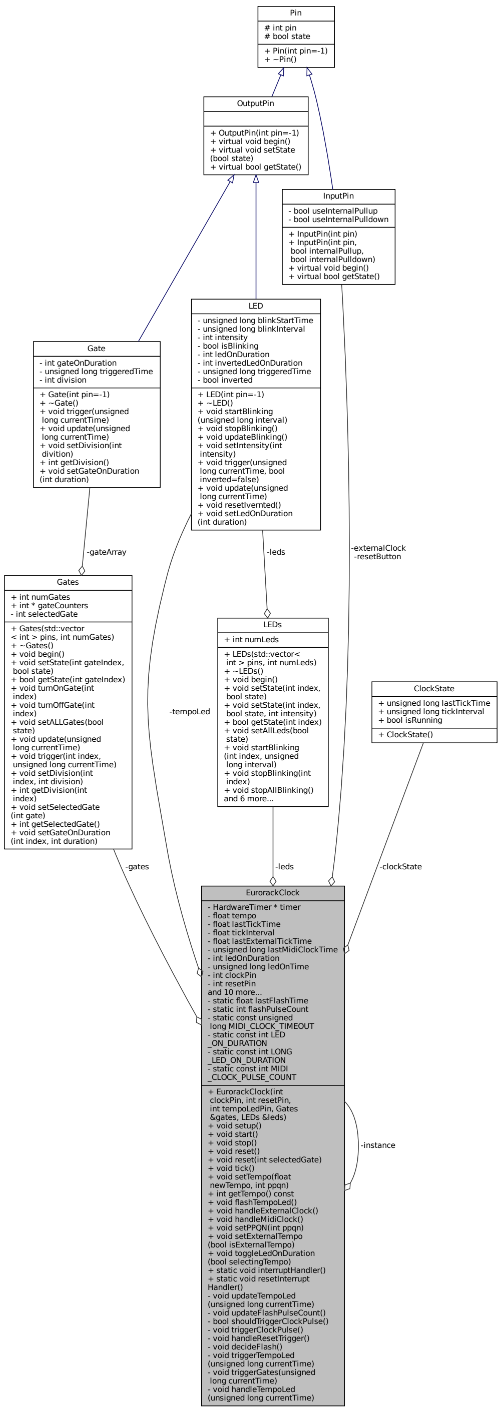
The documentation for this class was generated from the following files:

- include/Encoder.h
- src/Encoder.cpp

4.5 EurorackClock Class Reference

```
#include <EurorackClock.h>
```

Collaboration diagram for EurorackClock:



Public Member Functions

- EurorackClock (int clockPin, int resetPin, int tempoLedPin, Gates &gates, LEDs &leds)
- void setup ()
- void start ()
- void stop ()
- void reset ()

- void reset (int selectedGate)
- void tick ()
- void setTempo (float newTempo, int ppqn)
- int getTempo () const
- void flashTempoLed ()
- void handleExternalClock ()
- void handleMidiClock ()
- void setPPQN (int ppqn)
- void setExternalTempo (bool isExternalTempo)
- void toggleLedOnDuration (bool selectingTempo)

Static Public Member Functions

- static void interruptHandler ()
- static void resetInterruptHandler ()

Private Member Functions

- void updateTempoLed (unsigned long currentTime)
- void updateFlashPulseCount ()
- bool shouldTriggerClockPulse ()
- void triggerClockPulse ()
- void handleResetTrigger ()
- void decideFlash ()
- void triggerTempoLed (unsigned long currentTime)
- void triggerGates (unsigned long currentTime)
- void handleTempoLed (unsigned long currentTime)

Private Attributes

- ClockState clockState
- HardwareTimer ∗ timer
- LED tempoLed
- InputPin externalClock
- InputPin resetButton
- Gates & gates
- LEDs & leds
- float tempo
- float lastTickTime
- float tickInterval
- float lastExternalTickTime
- unsigned long lastMidiClockTime
- int ledOnDuration = LONG_LED_ON_DURATION
- unsigned long ledOnTime = 0
- int clockPin
- int resetPin
- int ppqn
- bool isRunning
- bool isExternalTempo
- bool isMidiClock
- bool timeToFlash
- bool resetTriggered
- float externalTempo
- int lastClockState
- unsigned long lastClockTime
- int tickCount

Static Private Attributes

- static EurorackClock ∗ instance = nullptr
- static float lastFlashTime = 0
- static int flashPulseCount = 0
- static const unsigned long MIDI_CLOCK_TIMEOUT = 1000
- static const int LED_ON_DURATION = 10
- static const int LONG_LED_ON_DURATION = 50
- static const int MIDI_CLOCK_PULSE_COUNT = 24

4.5.1 Constructor & Destructor Documentation

4.5.1.1 EurorackClock()

```
EurorackClock::EurorackClock (
            int clockPin,
            int resetPin,
            int tempoLedPin,
            Gates & gates,
            LEDs & leds )
```

4.5.2 Member Function Documentation

4.5.2.1 decideFlash()

```
void EurorackClock::decideFlash ( )  [private]
```

4.5.2.2 flashTempoLed()

```
void EurorackClock::flashTempoLed ( )
```

4.5.2.3 getTempo()

```
int EurorackClock::getTempo ( ) const
```

4.5.2.4 handleExternalClock()

```
void EurorackClock::handleExternalClock ( )
```

4.5.2.5 handleMidiClock()

```
void EurorackClock::handleMidiClock ( )
```

4.5.2.6 handleResetTrigger()

```
void EurorackClock::handleResetTrigger ( )  [private]
```

4.5.2.7 handleTempoLed()

```
void EurorackClock::handleTempoLed (
             unsigned long currentTime )  [private]
```

4.5.2.8 interruptHandler()

```
static void EurorackClock::interruptHandler ( )  [inline], [static]
```

4.5.2.9 reset() [1/2]

```
void EurorackClock::reset ( )
```

4.5.2.10 reset() [2/2]

```
void EurorackClock::reset (
             int selectedGate )
```

4.5.2.11 resetInterruptHandler()

```
static void EurorackClock::resetInterruptHandler ( )  [inline], [static]
```

4.5.2.12 setExternalTempo()

```
void EurorackClock::setExternalTempo (
             bool isExternalTempo )
```

4.5.2.13 setPPQN()

```
void EurorackClock::setPPQN (
             int ppqn )
```

4.5.2.14 setTempo()

```
void EurorackClock::setTempo (
             float newTempo,
             int ppqn )
```

4.5.2.15 setup()

```
void EurorackClock::setup ( )
```

4.5.2.16 shouldTriggerClockPulse()

```
bool EurorackClock::shouldTriggerClockPulse ( )  [private]
```

4.5.2.17 start()

```
void EurorackClock::start ( )
```

4.5.2.18 stop()

```
void EurorackClock::stop ( )
```

4.5.2.19 tick()

```
void EurorackClock::tick ( )
```

4.5.2.20 toggleLedOnDuration()

```
void EurorackClock::toggleLedOnDuration (
             bool selectingTempo )
```

4.5.2.21 triggerClockPulse()

```
void EurorackClock::triggerClockPulse ( )  [private]
```

4.5.2.22 triggerGates()

```
void EurorackClock::triggerGates (
             unsigned long currentTime )  [private]
```

4.5.2.23 triggerTempoLed()

```
void EurorackClock::triggerTempoLed (
             unsigned long currentTime )  [private]
```

4.5.2.24 updateFlashPulseCount()

```
void EurorackClock::updateFlashPulseCount ( )  [private]
```

4.5.2.25 updateTempoLed()

```
void EurorackClock::updateTempoLed (
             unsigned long currentTime )  [private]
```

4.5.3 Member Data Documentation

4.5.3.1 clockPin

```
int EurorackClock::clockPin  [private]
```

4.5.3.2 clockState

```
ClockState EurorackClock::clockState  [private]
```

4.5.3.3 externalClock

```
InputPin EurorackClock::externalClock  [private]
```

4.5.3.4 externalTempo

```
float EurorackClock::externalTempo  [private]
```

4.5.3.5 flashPulseCount

```
int EurorackClock::flashPulseCount = 0  [static], [private]
```

4.5.3.6 gates

```
Gates& EurorackClock::gates  [private]
```

4.5.3.7 instance

```
EurorackClock * EurorackClock::instance = nullptr  [static], [private]
```

4.5.3.8 isExternalTempo

```
bool EurorackClock::isExternalTempo  [private]
```

4.5.3.9 isMidiClock

```
bool EurorackClock::isMidiClock  [private]
```

4.5.3.10 isRunning

```
bool EurorackClock::isRunning  [private]
```

4.5.3.11 lastClockState

```
int EurorackClock::lastClockState  [private]
```

4.5.3.12 lastClockTime

```
unsigned long EurorackClock::lastClockTime  [private]
```

4.5.3.13 lastExternalTickTime

```
float EurorackClock::lastExternalTickTime  [private]
```

4.5.3.14 lastFlashTime

```
float EurorackClock::lastFlashTime = 0  [static], [private]
```

4.5.3.15 lastMidiClockTime

```
unsigned long EurorackClock::lastMidiClockTime  [private]
```

4.5.3.16 lastTickTime

```
float EurorackClock::lastTickTime  [private]
```

4.5.3.17 LED_ON_DURATION

```
const int EurorackClock::LED_ON_DURATION = 10  [static], [private]
```

4.5.3.18 ledOnDuration

```
int EurorackClock::ledOnDuration = LONG_LED_ON_DURATION  [private]
```

4.5.3.19 ledOnTime

```
unsigned long EurorackClock::ledOnTime = 0  [private]
```

4.5.3.20 leds

```
LEDs& EurorackClock::leds  [private]
```

4.5.3.21 LONG_LED_ON_DURATION

```
const int EurorackClock::LONG_LED_ON_DURATION = 50  [static], [private]
```

4.5.3.22 MIDI_CLOCK_PULSE_COUNT

```
const int EurorackClock::MIDI_CLOCK_PULSE_COUNT = 24  [static], [private]
```

4.5.3.23 MIDI_CLOCK_TIMEOUT

```
const unsigned long EurorackClock::MIDI_CLOCK_TIMEOUT = 1000  [static], [private]
```

4.5.3.24 ppqn

```
int EurorackClock::ppqn  [private]
```

4.5.3.25 resetButton

```
InputPin EurorackClock::resetButton  [private]
```

4.5.3.26 resetPin

```
int EurorackClock::resetPin  [private]
```

4.5.3.27 resetTriggered

```
bool EurorackClock::resetTriggered  [private]
```

4.5.3.28 tempo

```
float EurorackClock::tempo  [private]
```

4.5.3.29 tempoLed

```
LED EurorackClock::tempoLed  [private]
```

4.5.3.30 tickCount

```
int EurorackClock::tickCount  [private]
```

4.5.3.31 tickInterval

```
float EurorackClock::tickInterval  [private]
```

4.5.3.32 timer

```
HardwareTimer* EurorackClock::timer  [private]
```

4.5.3.33 timeToFlash

```
bool EurorackClock::timeToFlash  [private]
```

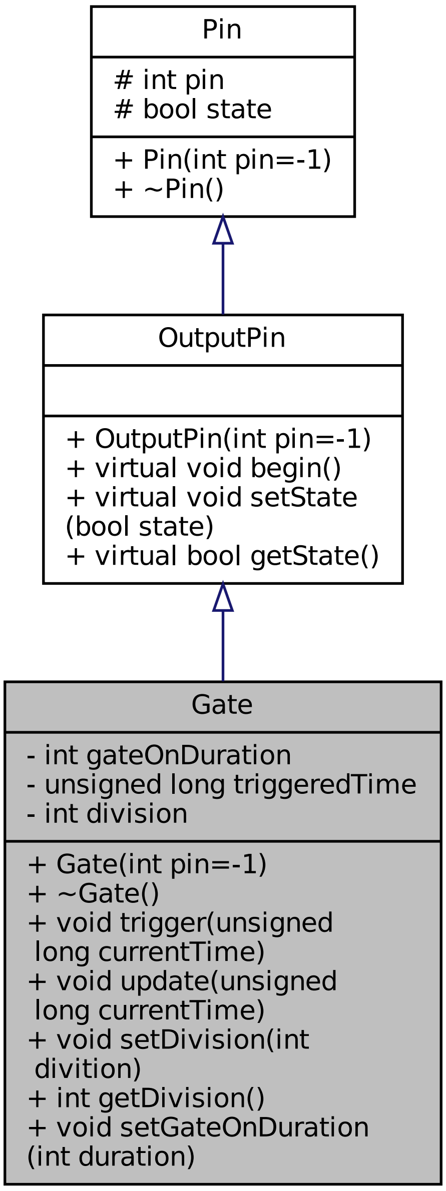
The documentation for this class was generated from the following files:

- include/EurorackClock.h
- src/EurorackClock.cpp

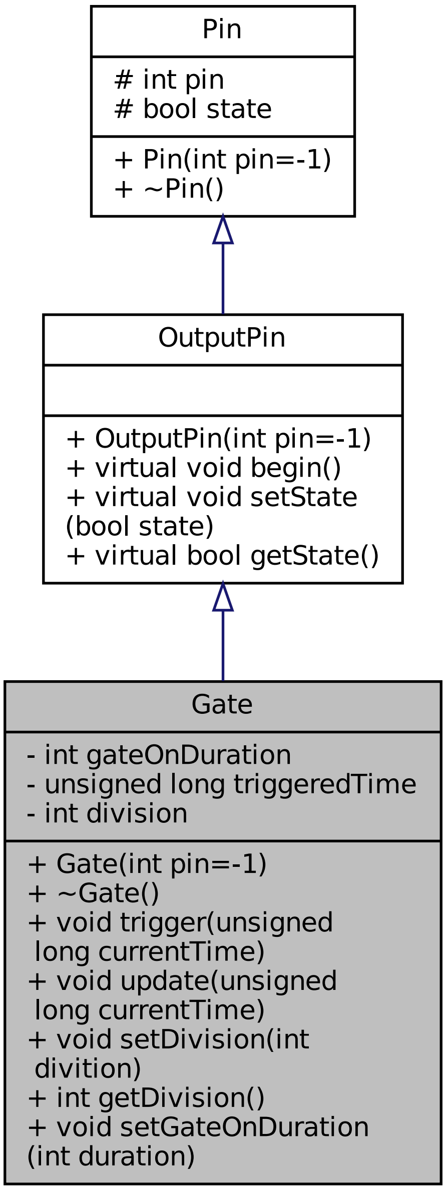
4.6 Gate Class Reference

```
#include <Gate.h>
```

Inheritance diagram for Gate:


Pin
# int pin
# bool state
+ Pin(int pin=-1)
+ ~Pin()
OutputPin
+ OutputPin(int pin=-1)
+ virtual void begin()
+ virtual void setState (bool state)
+ virtual bool getState()
Gate
- int gateOnDuration
- unsigned long triggeredTime
- int division
+ Gate(int pin=-1)
+ ~Gate()
+ void trigger(unsigned long currentTime)
+ void update(unsigned long currentTime)
+ void setDivision(int divition)
+ int getDivision()
+ void setGateOnDuration (int duration)

Collaboration diagram for Gate:



Public Member Functions

- Gate (int pin=-1)
- ∼Gate ()
- void trigger (unsigned long currentTime)
- void update (unsigned long currentTime)
- void setDivision (int divition)
- int getDivision ()
- void setGateOnDuration (int duration)

Private Attributes

- int gateOnDuration = 10
- unsigned long triggeredTime = 0
- int division = internalPPQN

Additional Inherited Members

4.6.1 Constructor & Destructor Documentation

4.6.1.1 Gate()

```
Gate::Gate (
             int pin = -1 )
```

4.6.1.2 ∼Gate()

```
Gate::∼Gate ( )
```

4.6.2 Member Function Documentation

4.6.2.1 getDivision()

```
int Gate::getDivision ( )
```

4.6.2.2 setDivision()

```
void Gate::setDivision (
             int divition )
```

4.6.2.3 setGateOnDuration()

```
void Gate::setGateOnDuration (
             int duration )
```

4.6.2.4 trigger()

```
void Gate::trigger (
             unsigned long currentTime )
```

4.6.2.5 update()

```
void Gate::update (
             unsigned long currentTime )
```

4.6.3 Member Data Documentation

4.6.3.1 division

```
int Gate::division = internalPPQN  [private]
```

4.6.3.2 gateOnDuration

```
int Gate::gateOnDuration = 10  [private]
```

4.6.3.3 triggeredTime

```
unsigned long Gate::triggeredTime = 0  [private]
```

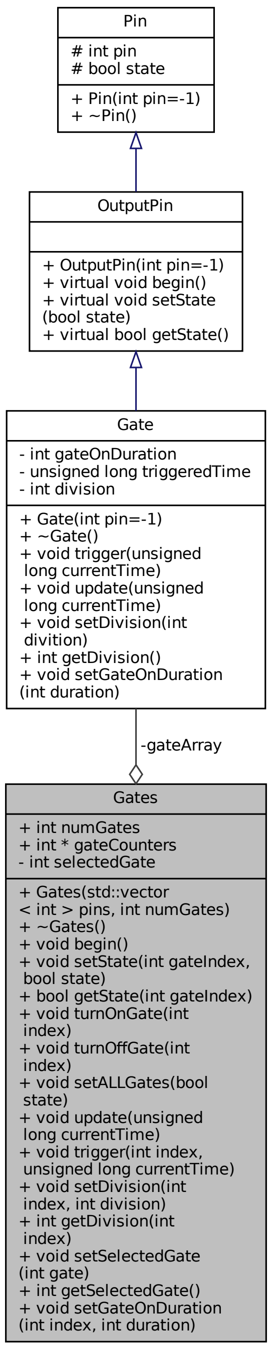
The documentation for this class was generated from the following files:

- include/Gate.h
- src/Gate.cpp

4.7 Gates Class Reference

```
#include <Gates.h>
```

Collaboration diagram for Gates:



Pin
# int pin
# bool state
+ Pin(int pin=-1)
+ ~Pin()
OutputPin
+ OutputPin(int pin=-1)
+ virtual void begin()
+ virtual void setState (bool state)
+ virtual bool getState()
Gate
- int gateOnDuration
- unsigned long triggeredTime
- int division
+ Gate(int pin=-1)
+ ~Gate()
+ void trigger(unsigned long currentTime)
+ void update(unsigned long currentTime)
+ void setDivision(int divition)
+ int getDivision()
+ void setGateOnDuration (int duration)
-gateArray
Gates
+ int numGates
+ int * gateCounters
- int selectedGate
+ Gates(std::vector < int > pins, int numGates)
+ ~Gates()
+ void begin()
+ void setState(int gateIndex, bool state)
+ bool getState(int gateIndex)
+ void turnOnGate(int index)
+ void turnOffGate(int index)
+ void setALLGates(bool state)
+ void update(unsigned long currentTime)
+ void trigger(int index, unsigned long currentTime)
+ void setDivision(int index, int division)
+ int getDivision(int index)
+ void setSelectedGate (int gate)
+ int getSelectedGate()
+ void setGateOnDuration (int index, int duration)

Public Member Functions

- Gates (std::vector< int > pins, int numGates)

- ∼Gates ()
- void begin ()
- void setState (int gateIndex, bool state)
- bool getState (int gateIndex)
- void turnOnGate (int index)
- void turnOffGate (int index)
- void setALLGates (bool state)
- void update (unsigned long currentTime)
- void trigger (int index, unsigned long currentTime)
- void setDivision (int index, int division)
- int getDivision (int index)
- void setSelectedGate (int gate)
- int getSelectedGate ()
- void setGateOnDuration (int index, int duration)

Public Attributes

- int numGates
- int ∗ gateCounters

Private Attributes

- Gate ∗ gateArray
- int selectedGate

4.7.1 Constructor & Destructor Documentation

4.7.1.1 Gates()

```
Gates::Gates (
          std::vector< int > pins,
          int numGates )
```

4.7.1.2 ∼Gates()

```
Gates::∼Gates ( )
```

4.7.2 Member Function Documentation

4.7.2.1 begin()

```
void Gates::begin ( )
```

4.7.2.2 getDivision()

```
int Gates::getDivision (
             int index )
```

4.7.2.3 getSelectedGate()

```
int Gates::getSelectedGate ( )
```

4.7.2.4 getState()

```
bool Gates::getState (
             int gateIndex )
```

4.7.2.5 setALLGates()

```
void Gates::setALLGates (
             bool state )
```

4.7.2.6 setDivision()

```
void Gates::setDivision (
             int index,
             int division )
```

4.7.2.7 setGateOnDuration()

```
void Gates::setGateOnDuration (
             int index,
             int duration )
```

4.7.2.8 setSelectedGate()

```
void Gates::setSelectedGate (
            int gate )
```

4.7.2.9 setState()

```
void Gates::setState (
            int gateIndex,
            bool state )
```

4.7.2.10 trigger()

```
void Gates::trigger (
            int index,
            unsigned long currentTime )
```

4.7.2.11 turnOffGate()

```
void Gates::turnOffGate (
            int index )
```

4.7.2.12 turnOnGate()

```
void Gates::turnOnGate (
            int index )
```

4.7.2.13 update()

```
void Gates::update (
            unsigned long currentTime )
```

4.7.3 Member Data Documentation

4.7.3.1 gateArray

```
Gate* Gates::gateArray  [private]
```

4.7.3.2 gateCounters

```
int* Gates::gateCounters
```

4.7.3.3 numGates

```
int Gates::numGates
```

4.7.3.4 selectedGate

```
int Gates::selectedGate  [private]
```

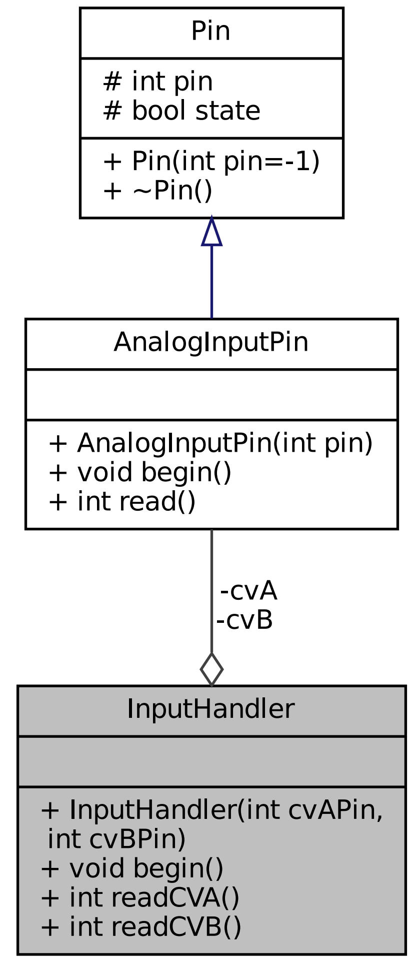
The documentation for this class was generated from the following files:

- include/Gates.h
- src/Gates.cpp

4.8 InputHandler Class Reference

```
#include <InputHandler.h>
```

Collaboration diagram for InputHandler:



Public Member Functions

- InputHandler (int cvAPin, int cvBPin)
- void begin ()
- int readCVA ()
- int readCVB ()

Private Attributes

- AnalogInputPin cvA
- AnalogInputPin cvB

4.8.1 Constructor & Destructor Documentation

4.8.1.1 InputHandler()

```
InputHandler::InputHandler (
             int cvAPin,
             int cvBPin )
```

4.8.2 Member Function Documentation

4.8.2.1 begin()

```
void InputHandler::begin ( )
```

4.8.2.2 readCVA()

```
int InputHandler::readCVA ( )
```

4.8.2.3 readCVB()

```
int InputHandler::readCVB ( )
```

4.8.3 Member Data Documentation

4.8.3.1 cvA

```
AnalogInputPin InputHandler::cvA  [private]
```

4.8.3.2 cvB

```
AnalogInputPin InputHandler::cvB  [private]
```

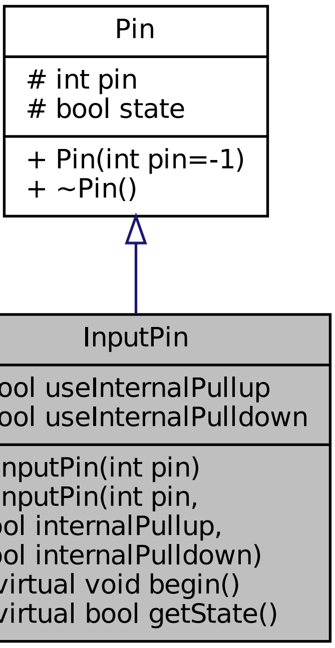
The documentation for this class was generated from the following files:

- include/InputHandler.h
- src/InputHandler.cpp

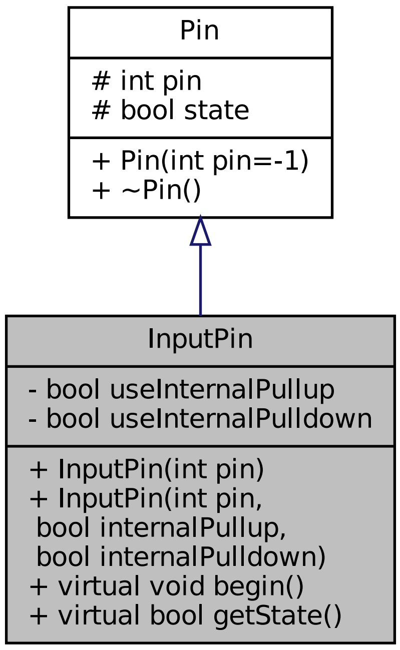
4.9 InputPin Class Reference

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

Inheritance diagram for InputPin:



Collaboration diagram for InputPin:



Public Member Functions

- InputPin (int pin)
- InputPin (int pin, bool internalPullup, bool internalPulldown)
- virtual void begin ()
- virtual bool getState ()

Private Attributes

- bool useInternalPullup
- bool useInternalPulldown

Additional Inherited Members

4.9.1 Constructor & Destructor Documentation

4.9.1.1 InputPin() [1/2]

```
InputPin::InputPin (
             int pin )
```

4.9.1.2 InputPin() [2/2]

```
InputPin::InputPin (
             int pin,
             bool internalPullup,
             bool internalPulldown )
```

4.9.2 Member Function Documentation

4.9.2.1 begin()

```
void InputPin::begin ( )  [virtual]
```

4.9.2.2 getState()

```
bool InputPin::getState ( )  [virtual]
```

4.9.3 Member Data Documentation

4.9.3.1 useInternalPulldown

```
bool InputPin::useInternalPulldown  [private]
```

4.9.3.2 useInternalPullup

```
bool InputPin::useInternalPullup  [private]
```

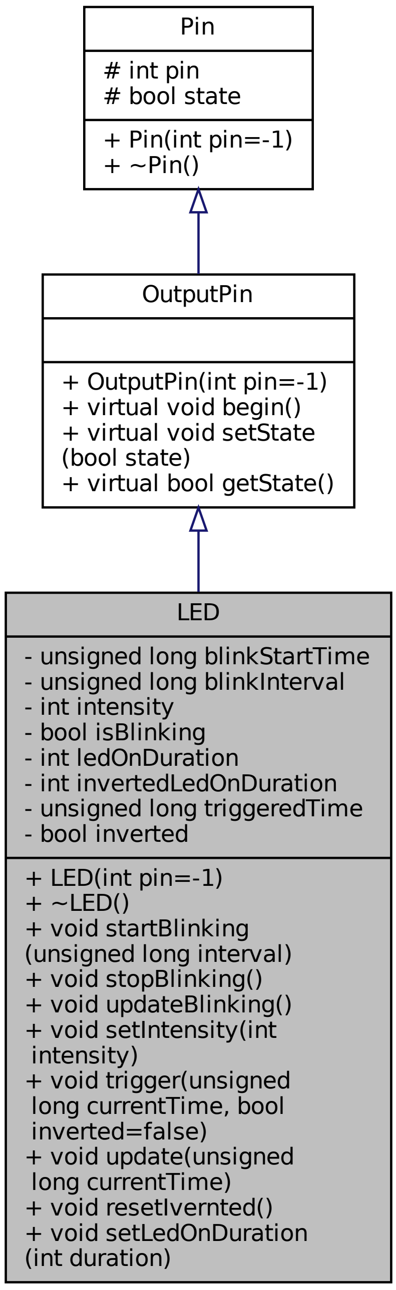
The documentation for this class was generated from the following files:

- include/Pin.h
- src/Pin.cpp

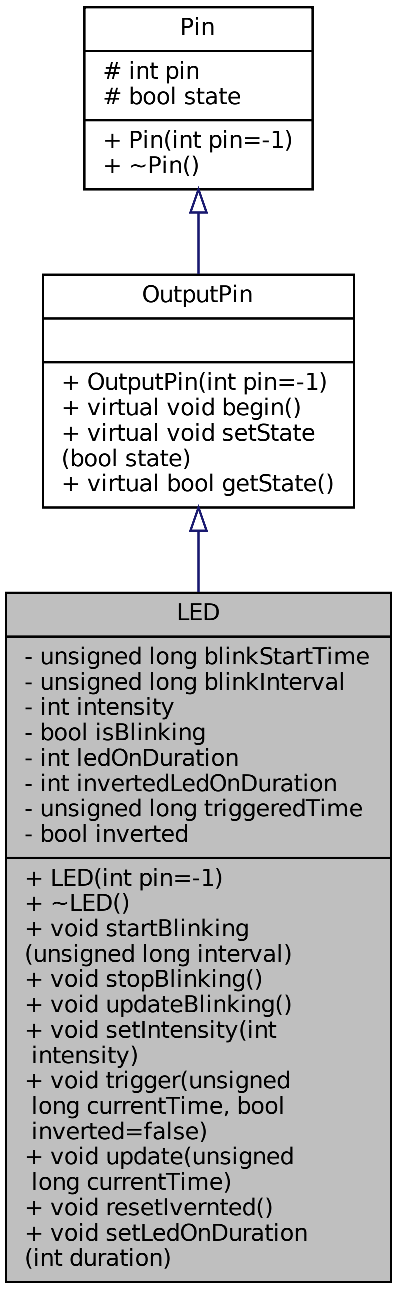
4.10 LED Class Reference

```
#include <LED.h>
```

Inheritance diagram for LED:



Collaboration diagram for LED:


Pin
# int pin
# bool state
+ Pin(int pin=-1)
+ ~Pin()
OutputPin
+ OutputPin(int pin=-1)
+ virtual void begin()
+ virtual void setState (bool state)
+ virtual bool getState()
LED
- unsigned long blinkStartTime
- unsigned long blinkInterval
- int intensity
- bool isBlinking
- int ledOnDuration
- int invertedLedOnDuration
- unsigned long triggeredTime
- bool inverted
+ LED(int pin=-1)
+ ~LED()
+ void startBlinking (unsigned long interval)
+ void stopBlinking()
+ void updateBlinking()
+ void setIntensity(int intensity)
+ void trigger(unsigned long currentTime, bool inverted=false)
+ void update(unsigned long currentTime)
+ void resetIvernted()
+ void setLedOnDuration (int duration)

Public Member Functions

- LED (int pin=-1)
- ∼LED ()
- void startBlinking (unsigned long interval)
- void stopBlinking ()
- void updateBlinking ()

- void setIntensity (int intensity)
- void trigger (unsigned long currentTime, bool inverted=false)
- void update (unsigned long currentTime)
- void resetIvernted ()
- void setLedOnDuration (int duration)

Private Attributes

- unsigned long blinkStartTime
- unsigned long blinkInterval
- int intensity = 255
- bool isBlinking
- int ledOnDuration = 25
- int invertedLedOnDuration = 40
- unsigned long triggeredTime = 0
- bool inverted = false

Additional Inherited Members

4.10.1 Constructor & Destructor Documentation

4.10.1.1 LED()

```
LED::LED (
            int pin = -1 )
```

4.10.1.2 ∼LED()

```
LED::∼LED ( )
```

4.10.2 Member Function Documentation

4.10.2.1 resetIvernted()

```
void LED::resetIvernted ( )
```

4.10.2.2 setIntensity()

```
void LED::setIntensity (
             int intensity )
```

4.10.2.3 setLedOnDuration()

```
void LED::setLedOnDuration (
             int duration )
```

4.10.2.4 startBlinking()

```
void LED::startBlinking (
             unsigned long interval )
```

4.10.2.5 stopBlinking()

```
void LED::stopBlinking ( )
```

4.10.2.6 trigger()

```
void LED::trigger (
             unsigned long currentTime,
             bool inverted = false )
```

4.10.2.7 update()

```
void LED::update (
             unsigned long currentTime )
```

4.10.2.8 updateBlinking()

```
void LED::updateBlinking ( )
```

4.10.3 Member Data Documentation

4.10.3.1 blinkInterval

```
unsigned long LED::blinkInterval  [private]
```

4.10.3.2 blinkStartTime

```
unsigned long LED::blinkStartTime  [private]
```

4.10.3.3 intensity

```
int LED::intensity = 255  [private]
```

4.10.3.4 inverted

```
bool LED::inverted = false  [private]
```

4.10.3.5 invertedLedOnDuration

```
int LED::invertedLedOnDuration = 40  [private]
```

4.10.3.6 isBlinking

```
bool LED::isBlinking  [private]
```

4.10.3.7 ledOnDuration

```
int LED::ledOnDuration = 25  [private]
```

4.10.3.8 triggeredTime

```
unsigned long LED::triggeredTime = 0  [private]
```

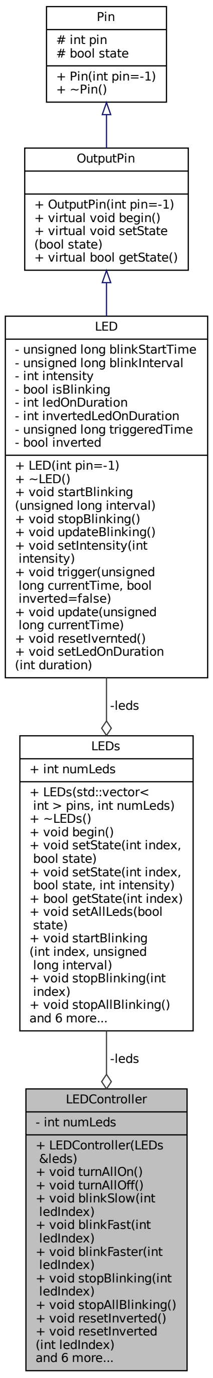
The documentation for this class was generated from the following files:

- include/LED.h
- src/LED.cpp

4.11 LEDController Class Reference

```
#include <LEDController.h>
```

Collaboration diagram for LEDController:


Pin
# int pin
# bool state
+ Pin(int pin=-1)
+ ~Pin()
OutputPin
+ OutputPin(int pin=-1)
+ virtual void begin()
+ virtual void setState (bool state)
+ virtual bool getState()
LED
- unsigned long blinkStartTime
- unsigned long blinkInterval
- int intensity
- bool isBlinking
- int ledOnDuration
- int invertedLedOnDuration
- unsigned long triggeredTime
- bool inverted
+ LED(int pin=-1)
+ ~LED()
+ void startBlinking (unsigned long interval)
+ void stopBlinking()
+ void updateBlinking()
+ void setIntensity(int intensity)
+ void trigger(unsigned long currentTime, bool inverted=false)
+ void update(unsigned long currentTime)
+ void resetIvernted()
+ void setLedOnDuration (int duration)
-leds
LEDs
+ int numLeds
+ LEDs(std::vector< int > pins, int numLeds)
+ ~LEDs()
+ void begin()
+ void setState(int index, bool state)
+ void setState(int index, bool state, int intensity)
+ bool getState(int index)
+ void setAllLeds(bool state)
+ void startBlinking (int index, unsigned long interval)
+ void stopBlinking(int index)
+ void stopAllBlinking()
and 6 more...
-leds
LEDController
- int numLeds
+ LEDController(LEDs &leds)
+ void turnAllOn()
+ void turnAllOff()
+ void blinkSlow(int ledIndex)
+ void blinkFast(int ledIndex)
+ void blinkFaster(int ledIndex)
+ void stopBlinking(int ledIndex)
+ void stopAllBlinking()
+ void resetInverted()
+ void resetInverted (int ledIndex)
and 6 more...

Public Member Functions

- LEDController (LEDs &leds)
- void turnAllOn ()
- void turnAllOff ()
- void blinkSlow (int ledIndex)
- void blinkFast (int ledIndex)

- void blinkFaster (int ledIndex)
- void stopBlinking (int ledIndex)
- void stopAllBlinking ()
- void resetInverted ()
- void resetInverted (int ledIndex)
- int getNumLeds ()
- void update ()
- void clearAndResetLEDs ()
- void clearLEDs ()
- void updateBlinking ()
- void setState (int ledIndex, bool state)

Private Attributes

- LEDs & leds
- int numLeds

4.11.1 Constructor & Destructor Documentation

4.11.1.1 LEDController()

```
LEDController::LEDController (
             LEDs & leds )
```

4.11.2 Member Function Documentation

4.11.2.1 blinkFast()

```
void LEDController::blinkFast (
             int ledIndex )
```

4.11.2.2 blinkFaster()

```
void LEDController::blinkFaster (
             int ledIndex )
```

4.11.2.3 blinkSlow()

```
void LEDController::blinkSlow (
             int ledIndex )
```

4.11.2.4 clearAndResetLEDs()

```
void LEDController::clearAndResetLEDs ( )
```

4.11.2.5 clearLEDs()

```
void LEDController::clearLEDs ( )
```

4.11.2.6 getNumLeds()

```
int LEDController::getNumLeds ( )
```

4.11.2.7 resetInverted() [1/2]

```
void LEDController::resetInverted ( )
```

4.11.2.8 resetInverted() [2/2]

```
void LEDController::resetInverted (
             int ledIndex )
```

4.11.2.9 setState()

```
void LEDController::setState (
             int ledIndex,
             bool state )
```

4.11.2.10 stopAllBlinking()

```
void LEDController::stopAllBlinking ( )
```

4.11.2.11 stopBlinking()

```
void LEDController::stopBlinking (
             int ledIndex )
```

4.11.2.12 turnAllOff()

```
void LEDController::turnAllOff ( )
```

4.11.2.13 turnAllOn()

```
void LEDController::turnAllOn ( )
```

4.11.2.14 update()

```
void LEDController::update ( )
```

4.11.2.15 updateBlinking()

```
void LEDController::updateBlinking ( )
```

4.11.3 Member Data Documentation

4.11.3.1 leds

```
LEDs& LEDController::leds  [private]
```

4.11.3.2 numLeds

```
int LEDController::numLeds  [private]
```

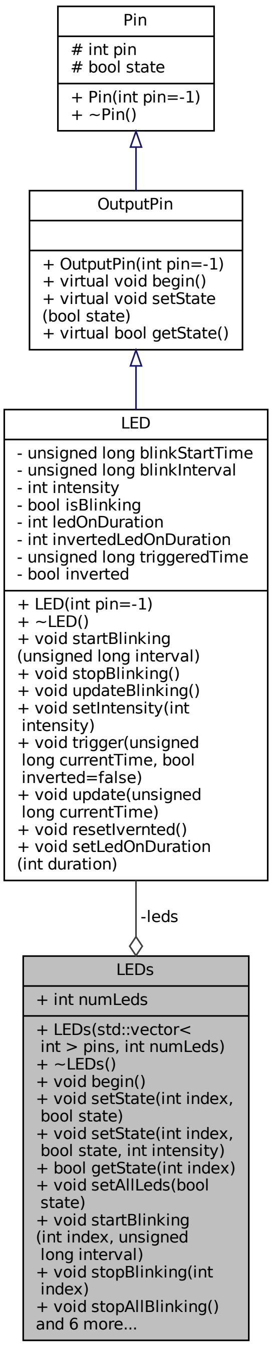
The documentation for this class was generated from the following files:

- include/LEDController.h
- src/LEDController.cpp

4.12 LEDs Class Reference

```
#include <LEDs.h>
```

Collaboration diagram for LEDs:


Pin
# int pin
# bool state
+ Pin(int pin=-1)
+ ~Pin()
OutputPin
+ OutputPin(int pin=-1)
+ virtual void begin()
+ virtual void setState(bool state)
+ virtual bool getState()
LED
- unsigned long blinkStartTime
- unsigned long blinkInterval
- int intensity
- bool isBlinking
- int ledOnDuration
- int invertedLedOnDuration
- unsigned long triggeredTime
- bool inverted
+ LED(int pin=-1)
+ ~LED()
+ void startBlinking(unsigned long interval)
+ void stopBlinking()
+ void updateBlinking()
+ void setIntensity(int intensity)
+ void trigger(unsigned long currentTime, bool inverted=false)
+ void update(unsigned long currentTime)
+ void resetIverted()
+ void setLedOnDuration(int duration)
-leds
LEDs
+ int numLeds
+ LEDs(std::vector< int > pins, int numLeds)
+ ~LEDs()
+ void begin()
+ void setState(int index, bool state)
+ void setState(int index, bool state, int intensity)
+ bool getState(int index)
+ void setAllLeds(bool state)
+ void startBlinking(int index, unsigned long interval)
+ void stopBlinking(int index)
+ void stopAllBlinking()
and 6 more...

Public Member Functions

- LEDs (std::vector< int > pins, int numLeds)
- ∼LEDs ()
- void begin ()
- void setState (int index, bool state)
- void setState (int index, bool state, int intensity)

- bool getState (int index)
- void setAllLeds (bool state)
- void startBlinking (int index, unsigned long interval)
- void stopBlinking (int index)
- void stopAllBlinking ()
- void updateBlinking ()
- void setIntensity (int index, int intensity)
- void setAllintensity (int intensity)
- void update (unsigned long currentTime)
- void trigger (int index, unsigned long currentTime, bool inverted=false)
- void resetInverted (int index)

Public Attributes

- int numLeds

Private Attributes

- LED ∗ leds

4.12.1 Constructor & Destructor Documentation

4.12.1.1 LEDs()

```
LEDs::LEDs (
            std::vector< int > pins,
            int numLeds )
```

4.12.1.2 ∼LEDs()

```
LEDs::∼LEDs ( )
```

4.12.2 Member Function Documentation

4.12.2.1 begin()

```
void LEDs::begin ( )
```

4.12.2.2 getState()

```
bool LEDs::getState (
             int index )
```

4.12.2.3 resetInverted()

```
void LEDs::resetInverted (
             int index )
```

4.12.2.4 setAllintensity()

```
void LEDs::setAllintensity (
             int intensity )
```

4.12.2.5 setAllLeds()

```
void LEDs::setAllLeds (
             bool state )
```

4.12.2.6 setIntensity()

```
void LEDs::setIntensity (
             int index,
             int intensity )
```

4.12.2.7 setState() [1/2]

```
void LEDs::setState (
             int index,
             bool state )
```

4.12.2.8 setState() [2/2]

```
void LEDs::setState (
            int index,
            bool state,
            int intensity )
```

4.12.2.9 startBlinking()

```
void LEDs::startBlinking (
            int index,
            unsigned long interval )
```

4.12.2.10 stopAllBlinking()

```
void LEDs::stopAllBlinking ( )
```

4.12.2.11 stopBlinking()

```
void LEDs::stopBlinking (
            int index )
```

4.12.2.12 trigger()

```
void LEDs::trigger (
            int index,
            unsigned long currentTime,
            bool inverted = false )
```

4.12.2.13 update()

```
void LEDs::update (
            unsigned long currentTime )
```

4.12.2.14 updateBlinking()

```
void LEDs::updateBlinking ( )
```

4.12.3 Member Data Documentation

4.12.3.1 leds

```
LED* LEDs::leds  [private]
```

4.12.3.2 numLeds

```
int LEDs::numLeds
```

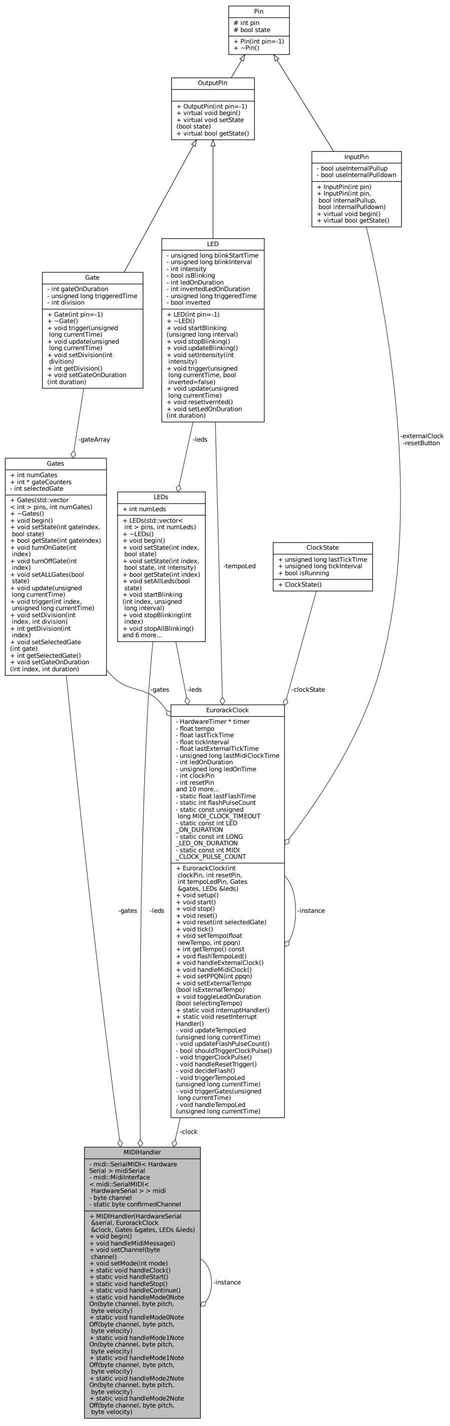
The documentation for this class was generated from the following files:

- include/LEDs.h
- src/LEDs.cpp

4.13 MIDIHandler Class Reference

```
#include <MIDIHandler.h>
```

Collaboration diagram for MIDIHandler:



Public Member Functions

- MIDIHandler (HardwareSerial &serial, EurorackClock &clock, Gates &gates, LEDs &leds)
- void begin ()
- void handleMidiMessage ()
- void setChannel (byte channel)
- void setMode (int mode)

Static Public Member Functions

- static void handleClock ()
- static void handleStart ()
- static void handleStop ()
- static void handleContinue ()
- static void handleMode0NoteOn (byte channel, byte pitch, byte velocity)
- static void handleMode0NoteOff (byte channel, byte pitch, byte velocity)
- static void handleMode1NoteOn (byte channel, byte pitch, byte velocity)
- static void handleMode1NoteOff (byte channel, byte pitch, byte velocity)
- static void handleMode2NoteOn (byte channel, byte pitch, byte velocity)
- static void handleMode2NoteOff (byte channel, byte pitch, byte velocity)

Private Attributes

- midi::SerialMIDI< HardwareSerial > midiSerial
- midi::MidiInterface< midi::SerialMIDI< HardwareSerial > > midi
- EurorackClock & clock
- byte channel = 10
- Gates & gates
- LEDs & leds

Static Private Attributes

- static MIDIHandler ∗ instance = nullptr
- static byte confirmedChannel = 9

4.13.1 Constructor & Destructor Documentation

4.13.1.1 MIDIHandler()

```
MIDIHandler::MIDIHandler (
             HardwareSerial & serial,
             EurorackClock & clock,
             Gates & gates,
             LEDs & leds )
```

4.13.2 Member Function Documentation

4.13.2.1 begin()

```
void MIDIHandler::begin ( )
```

4.13.2.2 handleClock()

```
void MIDIHandler::handleClock ( )  [static]
```

4.13.2.3 handleContinue()

```
void MIDIHandler::handleContinue ( )  [static]
```

4.13.2.4 handleMidiMessage()

```
void MIDIHandler::handleMidiMessage ( )
```

4.13.2.5 handleMode0NoteOff()

```
void MIDIHandler::handleMode0NoteOff (
            byte channel,
            byte pitch,
            byte velocity )  [static]
```

4.13.2.6 handleMode0NoteOn()

```
void MIDIHandler::handleMode0NoteOn (
            byte channel,
            byte pitch,
            byte velocity )  [static]
```

4.13.2.7 handleMode1NoteOff()

```
void MIDIHandler::handleMode1NoteOff (
            byte channel,
            byte pitch,
            byte velocity )  [static]
```

4.13.2.8 handleMode1NoteOn()

```
void MIDIHandler::handleMode1NoteOn (
            byte channel,
            byte pitch,
            byte velocity )  [static]
```

4.13.2.9 handleMode2NoteOff()

```
void MIDIHandler::handleMode2NoteOff (
            byte channel,
            byte pitch,
            byte velocity )  [static]
```

4.13.2.10 handleMode2NoteOn()

```
void MIDIHandler::handleMode2NoteOn (
            byte channel,
            byte pitch,
            byte velocity )  [static]
```

4.13.2.11 handleStart()

```
void MIDIHandler::handleStart ( )  [static]
```

4.13.2.12 handleStop()

```
void MIDIHandler::handleStop ( )  [static]
```

4.13.2.13 setChannel()

```
void MIDIHandler::setChannel (
            byte channel )
```

4.13.2.14 setMode()

```
void MIDIHandler::setMode (
             int mode )
```

4.13.3 Member Data Documentation

4.13.3.1 channel

```
byte MIDIHandler::channel = 10  [private]
```

4.13.3.2 clock

```
EurorackClock& MIDIHandler::clock  [private]
```

4.13.3.3 confirmedChannel

```
byte MIDIHandler::confirmedChannel = 9  [static], [private]
```

4.13.3.4 gates

```
Gates& MIDIHandler::gates  [private]
```

4.13.3.5 instance

```
MIDIHandler * MIDIHandler::instance = nullptr  [static], [private]
```

4.13.3.6 leds

```
LEDs& MIDIHandler::leds  [private]
```

4.13.3.7 midi

```
midi::MidiInterface<midi::SerialMIDI<HardwareSerial> > MIDIHandler::midi  [private]
```

4.13.3.8 midiSerial

```
midi::SerialMIDI<HardwareSerial> MIDIHandler::midiSerial  [private]
```

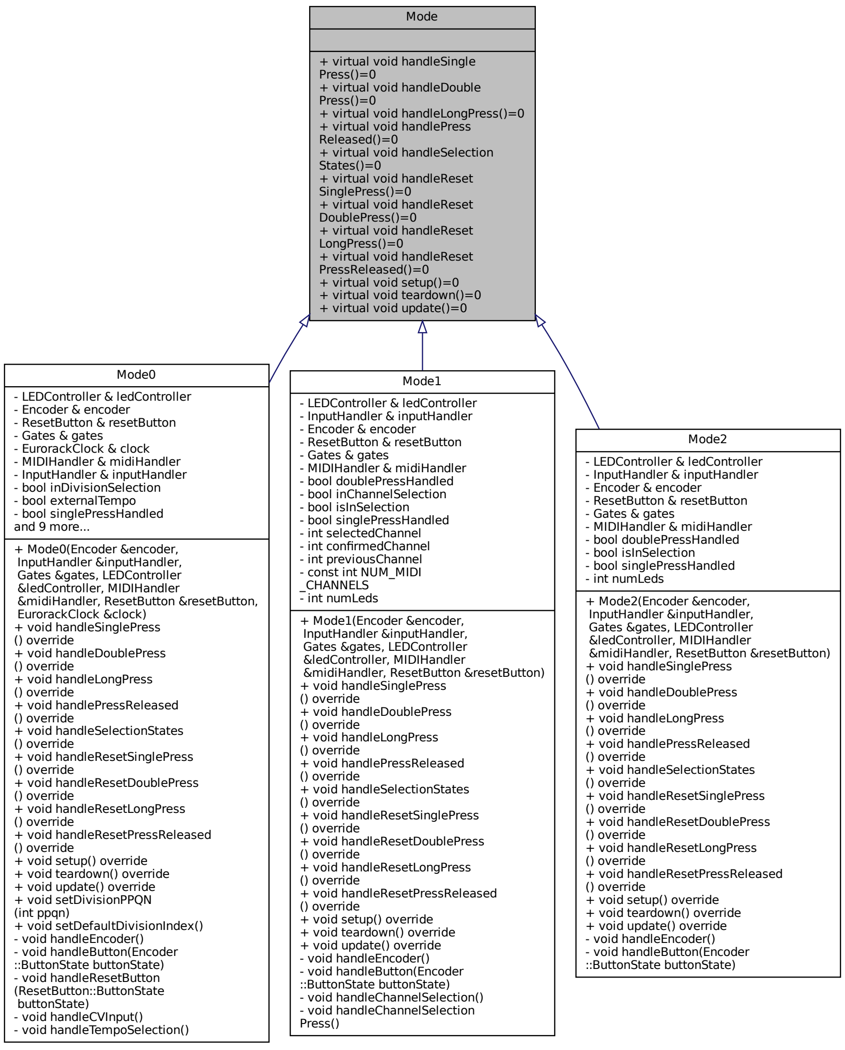
The documentation for this class was generated from the following files:

- include/MIDIHandler.h
- src/MIDIHandler.cpp

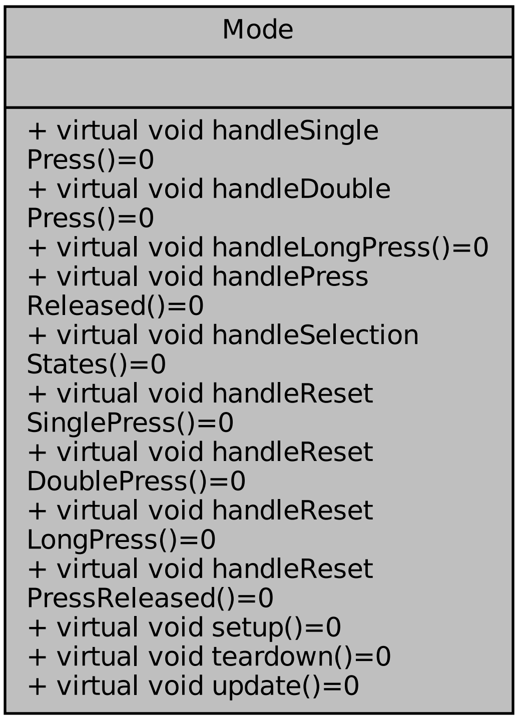
4.14 Mode Class Reference

```
#include <Mode.h>
```

Inheritance diagram for Mode:


Mode
+ virtual void handleSingle Press()=0
+ virtual void handleDouble Press()=0
+ virtual void handleLongPress()=0
+ virtual void handlePress Released()=0
+ virtual void handleSelection States()=0
+ virtual void handleReset SinglePress()=0
+ virtual void handleReset DoublePress()=0
+ virtual void handleReset LongPress()=0
+ virtual void handleReset PressReleased()=0
+ virtual void setup()=0
+ virtual void teardown()=0
+ virtual void update()=0
Mode0
- LEDController & ledController
- Encoder & encoder
- ResetButton & resetButton
- Gates & gates
- EurorackClock & clock
- MIDIHandler & midiHandler
- InputHandler & inputHandler
- bool inDivisionSelection
- bool externalTempo
- bool singlePressHandled
and 9 more...
+ Mode0(Encoder &encoder, InputHandler &inputHandler, Gates &gates, LEDController &ledController, MIDIHandler &midiHandler, ResetButton &resetButton, EurorackClock &clock)
+ void handleSinglePress () override
+ void handleDoublePress () override
+ void handleLongPress () override
+ void handlePressReleased () override
+ void handleSelectionStates () override
+ void handleResetSinglePress () override
+ void handleResetDoublePress () override
+ void handleResetLongPress () override
+ void handleResetPressReleased () override
+ void setup() override
+ void teardown() override
+ void update() override
+ void setDivisionPPQN (int ppqn)
+ void setDefaultDivisionIndex()
- void handleEncoder()
- void handleButton(Encoder ::ButtonState buttonState)
- void handleResetButton (ResetButton::ButtonState buttonState)
- void handleCVInput()
- void handleTempoSelection()
Mode1
- LEDController & ledController
- InputHandler & inputHandler
- Encoder & encoder
- ResetButton & resetButton
- Gates & gates
- MIDIHandler & midiHandler
- bool doublePressHandled
- bool inChannelSelection
- bool isInSelection
- bool singlePressHandled
- int selectedChannel
- int confirmedChannel
- int previousChannel
- const int NUM_MIDI _CHANNELS
- int numLeds
+ Mode1(Encoder &encoder, InputHandler &inputHandler, Gates &gates, LEDController &ledController, MIDIHandler &midiHandler, ResetButton &resetButton)
+ void handleSinglePress () override
+ void handleDoublePress () override
+ void handleLongPress () override
+ void handlePressReleased () override
+ void handleSelectionStates () override
+ void handleResetSinglePress () override
+ void handleResetDoublePress () override
+ void handleResetLongPress () override
+ void handleResetPressReleased () override
+ void setup() override
+ void teardown() override
+ void update() override
- void handleEncoder()
- void handleButton(Encoder ::ButtonState buttonState)
- void handleChannelSelection()
- void handleChannelSelection Press()
Mode2
- LEDController & ledController
- InputHandler & inputHandler
- Encoder & encoder
- ResetButton & resetButton
- Gates & gates
- MIDIHandler & midiHandler
- bool doublePressHandled
- bool isInSelection
- bool singlePressHandled
- int numLeds
+ Mode2(Encoder &encoder, InputHandler &inputHandler, Gates &gates, LEDController &ledController, MIDIHandler &midiHandler, ResetButton &resetButton)
+ void handleSinglePress () override
+ void handleDoublePress () override
+ void handleLongPress () override
+ void handlePressReleased () override
+ void handleSelectionStates () override
+ void handleResetSinglePress () override
+ void handleResetDoublePress () override
+ void handleResetLongPress () override
+ void handleResetPressReleased () override
+ void setup() override
+ void teardown() override
+ void update() override
- void handleEncoder()
- void handleButton(Encoder ::ButtonState buttonState)

Collaboration diagram for Mode:


Mode
+ virtual void handleSingle
Press()=0
+ virtual void handleDouble
Press()=0
+ virtual void handleLongPress()=0
+ virtual void handlePress
Released()=0
+ virtual void handleSelection
States()=0
+ virtual void handleReset
SinglePress()=0
+ virtual void handleReset
DoublePress()=0
+ virtual void handleReset
LongPress()=0
+ virtual void handleReset
PressReleased()=0
+ virtual void setup()=0
+ virtual void teardown()=0
+ virtual void update()=0

Public Member Functions

- virtual void handleSinglePress ()=0
- virtual void handleDoublePress ()=0
- virtual void handleLongPress ()=0
- virtual void handlePressReleased ()=0
- virtual void handleSelectionStates ()=0
- virtual void handleResetSinglePress ()=0
- virtual void handleResetDoublePress ()=0
- virtual void handleResetLongPress ()=0
- virtual void handleResetPressReleased ()=0
- virtual void setup ()=0
- virtual void teardown ()=0
- virtual void update ()=0

4.14.1 Member Function Documentation

4.14.1.1 handleDoublePress()

```
virtual void Mode::handleDoublePress ( )  [pure virtual]
```

Implemented in Mode2, Mode1, and Mode0.

4.14.1.2 handleLongPress()

```
virtual void Mode::handleLongPress ( )  [pure virtual]
```

Implemented in Mode2, Mode1, and Mode0.

4.14.1.3 handlePressReleased()

```
virtual void Mode::handlePressReleased ( )  [pure virtual]
```

Implemented in Mode2, Mode1, and Mode0.

4.14.1.4 handleResetDoublePress()

```
virtual void Mode::handleResetDoublePress ( )  [pure virtual]
```

Implemented in Mode2, Mode1, and Mode0.

4.14.1.5 handleResetLongPress()

```
virtual void Mode::handleResetLongPress ( )  [pure virtual]
```

Implemented in Mode2, Mode1, and Mode0.

4.14.1.6 handleResetPressReleased()

```
virtual void Mode::handleResetPressReleased ( )  [pure virtual]
```

Implemented in Mode2, Mode1, and Mode0.

4.14.1.7 handleResetSinglePress()

```
virtual void Mode::handleResetSinglePress ( ) [pure virtual]
```

Implemented in Mode2, Mode1, and Mode0.

4.14.1.8 handleSelectionStates()

```
virtual void Mode::handleSelectionStates ( ) [pure virtual]
```

Implemented in Mode2, Mode1, and Mode0.

4.14.1.9 handleSinglePress()

```
virtual void Mode::handleSinglePress ( ) [pure virtual]
```

Implemented in Mode2, Mode1, and Mode0.

4.14.1.10 setup()

```
virtual void Mode::setup ( ) [pure virtual]
```

Implemented in Mode2, Mode1, and Mode0.

4.14.1.11 teardown()

```
virtual void Mode::teardown ( ) [pure virtual]
```

Implemented in Mode2, Mode1, and Mode0.

4.14.1.12 update()

```
virtual void Mode::update ( ) [pure virtual]
```

Implemented in Mode2, Mode1, and Mode0.

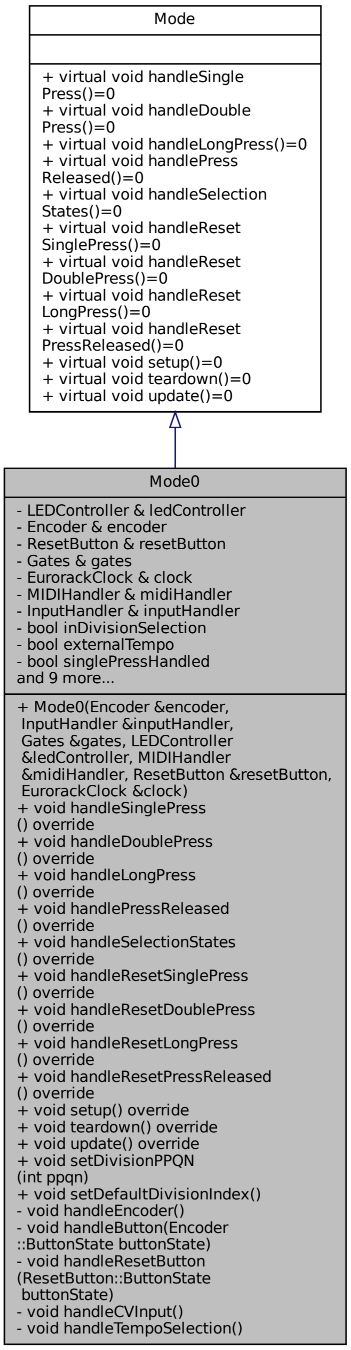
The documentation for this class was generated from the following file:

- include/Mode.h

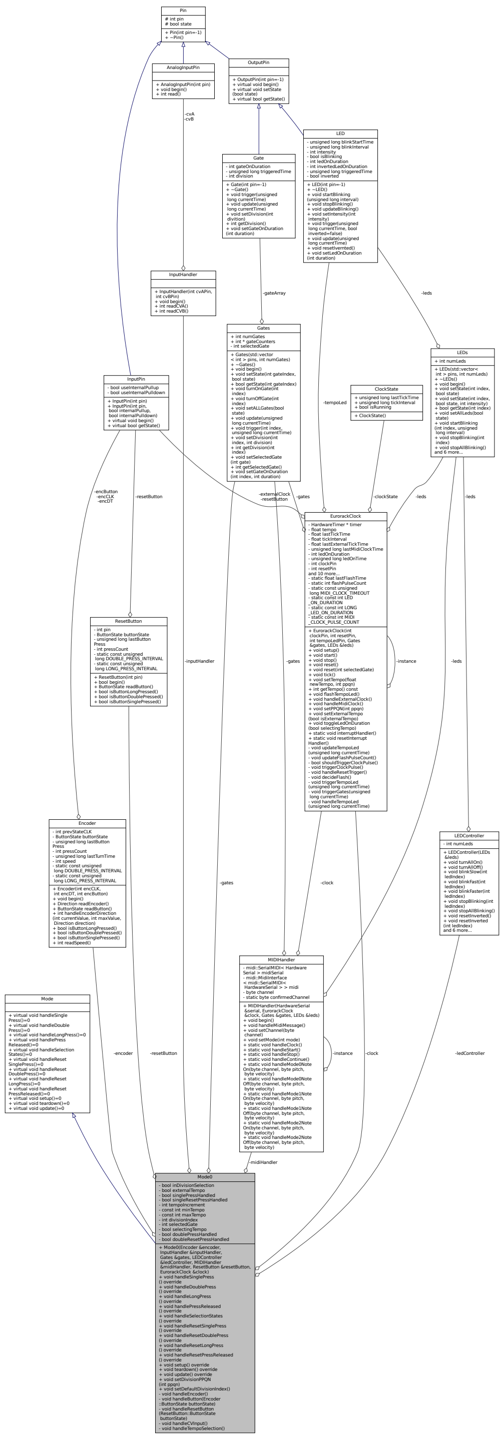
4.15 Mode0 Class Reference

```
#include <Mode0.h>
```

Inheritance diagram for Mode0:


Mode
+ virtual void handleSingle Press()=0
+ virtual void handleDouble Press()=0
+ virtual void handleLongPress()=0
+ virtual void handlePress Released()=0
+ virtual void handleSelection States()=0
+ virtual void handleReset SinglePress()=0
+ virtual void handleReset DoublePress()=0
+ virtual void handleReset LongPress()=0
+ virtual void handleReset PressReleased()=0
+ virtual void setup()=0
+ virtual void teardown()=0
+ virtual void update()=0
Mode0
- LEDController & ledController
- Encoder & encoder
- ResetButton & resetButton
- Gates & gates
- EurorackClock & clock
- MIDIHandler & midiHandler
- InputHandler & inputHandler
- bool inDivisionSelection
- bool externalTempo
- bool singlePressHandled
and 9 more...
+ Mode0(Encoder &encoder, InputHandler &inputHandler, Gates &gates, LEDController &ledController, MIDIHandler &midiHandler, ResetButton &resetButton, EurorackClock &clock)
+ void handleSinglePress () override
+ void handleDoublePress () override
+ void handleLongPress () override
+ void handlePressReleased () override
+ void handleSelectionStates () override
+ void handleResetSinglePress () override
+ void handleResetDoublePress () override
+ void handleResetLongPress () override
+ void handleResetPressReleased () override
+ void setup() override
+ void teardown() override
+ void update() override
+ void setDivisionPPQN (int ppqn)
+ void setDefaultDivisionIndex()
- void handleEncoder()
- void handleButton(Encoder ::ButtonState buttonState)
- void handleResetButton (ResetButton::ButtonState buttonState)
- void handleCVInput()
- void handleTempoSelection()

Collaboration diagram for Mode0:



Public Member Functions

- Mode0 (Encoder &encoder, InputHandler &inputHandler, Gates &gates, LEDController &ledController, MIDIHandler &midiHandler, ResetButton &resetButton, EurorackClock &clock)
- void handleSinglePress () override
- void handleDoublePress () override
- void handleLongPress () override

- void handlePressReleased () override
- void handleSelectionStates () override
- void handleResetSinglePress () override
- void handleResetDoublePress () override
- void handleResetLongPress () override
- void handleResetPressReleased () override
- void setup () override
- void teardown () override
- void update () override
- void setDivisionPPQN (int ppqn)
- void setDefaultDivisionIndex ()

Private Member Functions

- void handleEncoder ()
- void handleButton (Encoder::ButtonState buttonState)
- void handleResetButton (ResetButton::ButtonState buttonState)
- void handleCVInput ()
- void handleTempoSelection ()

Private Attributes

- LEDController & ledController
- Encoder & encoder
- ResetButton & resetButton
- Gates & gates
- EurorackClock & clock
- MIDIHandler & midiHandler
- InputHandler & inputHandler
- bool inDivisionSelection = false
- bool externalTempo = false
- bool singlePressHandled = false
- bool singleResetPressHandled = false
- int tempoIncrement = 1
- const int minTempo = 20
- const int maxTempo = 340
- int divisionIndex = 24
- int selectedGate = 0
- bool selectingTempo = false
- bool doublePressHandled = false
- bool doubleResetPressHandled = false

4.15.1 Constructor & Destructor Documentation

4.15.1.1 Mode0()

```
Mode0::Mode0 (
             Encoder & encoder,
             InputHandler & inputHandler,
             Gates & gates,
             LEDController & ledController,
             MIDIHandler & midiHandler,
             ResetButton & resetButton,
             EurorackClock & clock )
```

4.15.2 Member Function Documentation

4.15.2.1 handleButton()

```
void Mode0::handleButton (
             Encoder::ButtonState buttonState ) [private]
```

4.15.2.2 handleCVInput()

```
void Mode0::handleCVInput ( ) [private]
```

4.15.2.3 handleDoublePress()

```
void Mode0::handleDoublePress ( ) [override], [virtual]
```

Implements Mode.

4.15.2.4 handleEncoder()

```
void Mode0::handleEncoder ( ) [private]
```

4.15.2.5 handleLongPress()

```
void Mode0::handleLongPress ( ) [override], [virtual]
```

Implements Mode.

4.15.2.6 handlePressReleased()

```
void Mode0::handlePressReleased ( )  [override], [virtual]
```

Implements Mode.

4.15.2.7 handleResetButton()

```
void Mode0::handleResetButton (
            ResetButton::ButtonState buttonState )  [private]
```

4.15.2.8 handleResetDoublePress()

```
void Mode0::handleResetDoublePress ( )  [override], [virtual]
```

Implements Mode.

4.15.2.9 handleResetLongPress()

```
void Mode0::handleResetLongPress ( )  [override], [virtual]
```

Implements Mode.

4.15.2.10 handleResetPressReleased()

```
void Mode0::handleResetPressReleased ( )  [override], [virtual]
```

Implements Mode.

4.15.2.11 handleResetSinglePress()

```
void Mode0::handleResetSinglePress ( )  [override], [virtual]
```

Implements Mode.

4.15.2.12 handleSelectionStates()

```
void Mode0::handleSelectionStates ( )  [override], [virtual]
```

Implements Mode.

4.15.2.13 handleSinglePress()

```
void Mode0::handleSinglePress ( )  [override], [virtual]
```

Implements Mode.

4.15.2.14 handleTempoSelection()

```
void Mode0::handleTempoSelection ( )  [private]
```

4.15.2.15 setDefaultDivisionIndex()

```
void Mode0::setDefaultDivisionIndex ( )
```

4.15.2.16 setDivisionPPQN()

```
void Mode0::setDivisionPPQN (
             int ppqn )
```

4.15.2.17 setup()

```
void Mode0::setup ( )  [override], [virtual]
```

Implements Mode.

4.15.2.18 teardown()

```
void Mode0::teardown ( )  [override], [virtual]
```

Implements Mode.

4.15.2.19 update()

```
void Mode0::update ( )  [override], [virtual]
```

Implements Mode.

4.15.3 Member Data Documentation

4.15.3.1 clock

```
EurorackClock& Mode0::clock  [private]
```

4.15.3.2 divisionIndex

```
int Mode0::divisionIndex = 24  [private]
```

4.15.3.3 doublePressHandled

```
bool Mode0::doublePressHandled = false  [private]
```

4.15.3.4 doubleResetPressHandled

```
bool Mode0::doubleResetPressHandled = false  [private]
```

4.15.3.5 encoder

```
Encoder& Mode0::encoder  [private]
```

4.15.3.6 externalTempo

```
bool Mode0::externalTempo = false  [private]
```

4.15.3.7 gates

```
Gates& Mode0::gates [private]
```

4.15.3.8 inDivisionSelection

```
bool Mode0::inDivisionSelection = false [private]
```

4.15.3.9 inputHandler

```
InputHandler& Mode0::inputHandler [private]
```

4.15.3.10 ledController

```
LEDController& Mode0::ledController [private]
```

4.15.3.11 maxTempo

```
const int Mode0::maxTempo = 340 [private]
```

4.15.3.12 midiHandler

```
MIDIHandler& Mode0::midiHandler [private]
```

4.15.3.13 minTempo

```
const int Mode0::minTempo = 20 [private]
```

4.15.3.14 resetButton

```
ResetButton& Mode0::resetButton [private]
```

4.15.3.15 selectedGate

```
int Mode0::selectedGate = 0  [private]
```

4.15.3.16 selectingTempo

```
bool Mode0::selectingTempo = false  [private]
```

4.15.3.17 singlePressHandled

```
bool Mode0::singlePressHandled = false  [private]
```

4.15.3.18 singleResetPressHandled

```
bool Mode0::singleResetPressHandled = false  [private]
```

4.15.3.19 tempoIncrement

```
int Mode0::tempoIncrement = 1  [private]
```

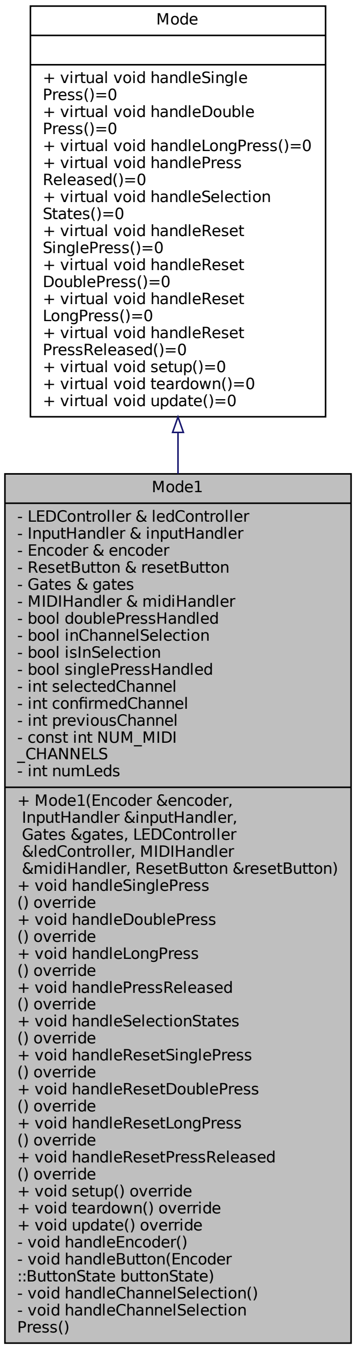
The documentation for this class was generated from the following files:

- include/Mode0.h
- src/Mode0.cpp

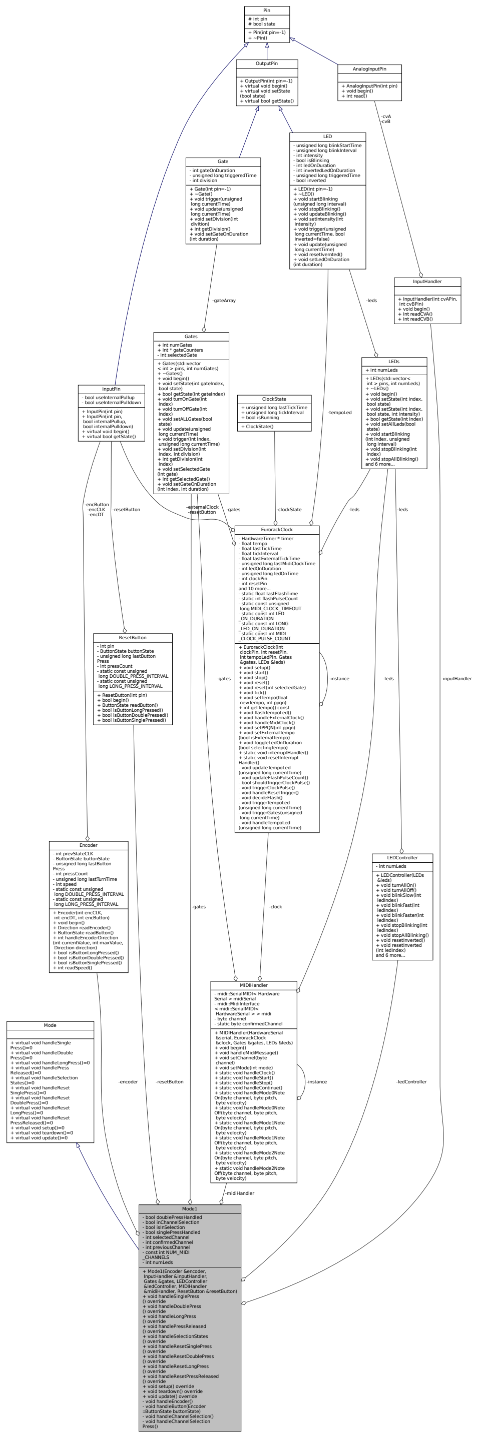
4.16 Mode1 Class Reference

```
#include <Mode1.h>
```

Inheritance diagram for Mode1:


Mode
+ virtual void handleSingle
Press()=0
+ virtual void handleDouble
Press()=0
+ virtual void handleLongPress()=0
+ virtual void handlePress
Released()=0
+ virtual void handleSelection
States()=0
+ virtual void handleReset
SinglePress()=0
+ virtual void handleReset
DoublePress()=0
+ virtual void handleReset
LongPress()=0
+ virtual void handleReset
PressReleased()=0
+ virtual void setup()=0
+ virtual void teardown()=0
+ virtual void update()=0
Mode1
- LEDController & ledController
- InputHandler & inputHandler
- Encoder & encoder
- ResetButton & resetButton
- Gates & gates
- MIDIHandler & midiHandler
- bool doublePressHandled
- bool inChannelSelection
- bool isInSelection
- bool singlePressHandled
- int selectedChannel
- int confirmedChannel
- int previousChannel
- const int NUM_MIDI
_CHANNELS
- int numLeds
+ Mode1(Encoder &encoder,
InputHandler &inputHandler,
Gates &gates, LEDController
&ledController, MIDIHandler
&midiHandler, ResetButton &resetButton)
+ void handleSinglePress
() override
+ void handleDoublePress
() override
+ void handleLongPress
() override
+ void handlePressReleased
() override
+ void handleSelectionStates
() override
+ void handleResetSinglePress
() override
+ void handleResetDoublePress
() override
+ void handleResetLongPress
() override
+ void handleResetPressReleased
() override
+ void setup() override
+ void teardown() override
+ void update() override
- void handleEncoder()
- void handleButton(Encoder
::ButtonState buttonState)
- void handleChannelSelection()
- void handleChannelSelection
Press()

Collaboration diagram for Mode1:



Public Member Functions

- Mode1 (Encoder &encoder, InputHandler &inputHandler, Gates &gates, LEDController &ledController, MIDIHandler &midiHandler, ResetButton &resetButton)
- void handleSinglePress () override
- void handleDoublePress () override
- void handleLongPress () override

- void handlePressReleased () override
- void handleSelectionStates () override
- void handleResetSinglePress () override
- void handleResetDoublePress () override
- void handleResetLongPress () override
- void handleResetPressReleased () override
- void setup () override
- void teardown () override
- void update () override

Private Member Functions

- void handleEncoder ()
- void handleButton (Encoder::ButtonState buttonState)
- void handleChannelSelection ()
- void handleChannelSelectionPress ()

Private Attributes

- LEDController & ledController
- InputHandler & inputHandler
- Encoder & encoder
- ResetButton & resetButton
- Gates & gates
- MIDIHandler & midiHandler
- bool doublePressHandled = false
- bool inChannelSelection = false
- bool isInSelection = false
- bool singlePressHandled = false
- int selectedChannel = 9
- int confirmedChannel = 9
- int previousChannel = -1
- const int NUM_MIDI_CHANNELS = 16
- int numLeds = 8

4.16.1 Constructor & Destructor Documentation

4.16.1.1 Mode1()

```
Mode1::Mode1 (
             Encoder & encoder,
             InputHandler & inputHandler,
             Gates & gates,
             LEDController & ledController,
             MIDIHandler & midiHandler,
             ResetButton & resetButton )
```

4.16.2 Member Function Documentation

4.16.2.1 handleButton()

```
void Mode1::handleButton (
            Encoder::ButtonState buttonState )  [private]
```

4.16.2.2 handleChannelSelection()

```
void Mode1::handleChannelSelection ( )  [private]
```

4.16.2.3 handleChannelSelectionPress()

```
void Mode1::handleChannelSelectionPress ( )  [private]
```

4.16.2.4 handleDoublePress()

```
void Mode1::handleDoublePress ( )  [override], [virtual]
```

Implements Mode.

4.16.2.5 handleEncoder()

```
void Mode1::handleEncoder ( )  [private]
```

4.16.2.6 handleLongPress()

```
void Mode1::handleLongPress ( )  [override], [virtual]
```

Implements Mode.

4.16.2.7 handlePressReleased()

```
void Mode1::handlePressReleased ( )  [override], [virtual]
```

Implements Mode.

4.16.2.8 handleResetDoublePress()

```
void Mode1::handleResetDoublePress ( )  [override], [virtual]
```

Implements Mode.

4.16.2.9 handleResetLongPress()

```
void Mode1::handleResetLongPress ( )  [override], [virtual]
```

Implements Mode.

4.16.2.10 handleResetPressReleased()

```
void Mode1::handleResetPressReleased ( )  [override], [virtual]
```

Implements Mode.

4.16.2.11 handleResetSinglePress()

```
void Mode1::handleResetSinglePress ( )  [override], [virtual]
```

Implements Mode.

4.16.2.12 handleSelectionStates()

```
void Mode1::handleSelectionStates ( )  [override], [virtual]
```

Implements Mode.

4.16.2.13 handleSinglePress()

```
void Mode1::handleSinglePress ( )  [override], [virtual]
```

Implements Mode.

4.16.2.14 setup()

```
void Mode1::setup ( )  [override], [virtual]
```

Implements Mode.

4.16.2.15 teardown()

```
void Mode1::teardown ( )  [override], [virtual]
```

Implements Mode.

4.16.2.16 update()

```
void Mode1::update ( )  [override], [virtual]
```

Implements Mode.

4.16.3 Member Data Documentation

4.16.3.1 confirmedChannel

```
int Mode1::confirmedChannel = 9  [private]
```

4.16.3.2 doublePressHandled

```
bool Mode1::doublePressHandled = false  [private]
```

4.16.3.3 encoder

```
Encoder& Mode1::encoder  [private]
```

4.16.3.4 gates

```
Gates& Mode1::gates  [private]
```

4.16.3.5 inChannelSelection

```
bool Mode1::inChannelSelection = false  [private]
```

4.16.3.6 inputHandler

```
InputHandler& Mode1::inputHandler  [private]
```

4.16.3.7 isInSelection

```
bool Mode1::isInSelection = false  [private]
```

4.16.3.8 ledController

```
LEDController& Mode1::ledController  [private]
```

4.16.3.9 midiHandler

```
MIDIHandler& Mode1::midiHandler  [private]
```

4.16.3.10 NUM_MIDI_CHANNELS

```
const int Mode1::NUM_MIDI_CHANNELS = 16  [private]
```

4.16.3.11 numLeds

```
int Mode1::numLeds = 8  [private]
```

4.16.3.12 previousChannel

```
int Mode1::previousChannel = -1  [private]
```

4.16.3.13 resetButton

```
ResetButton& Mode1::resetButton  [private]
```

4.16.3.14 selectedChannel

```
int Mode1::selectedChannel = 9  [private]
```

4.16.3.15 singlePressHandled

```
bool Mode1::singlePressHandled = false  [private]
```

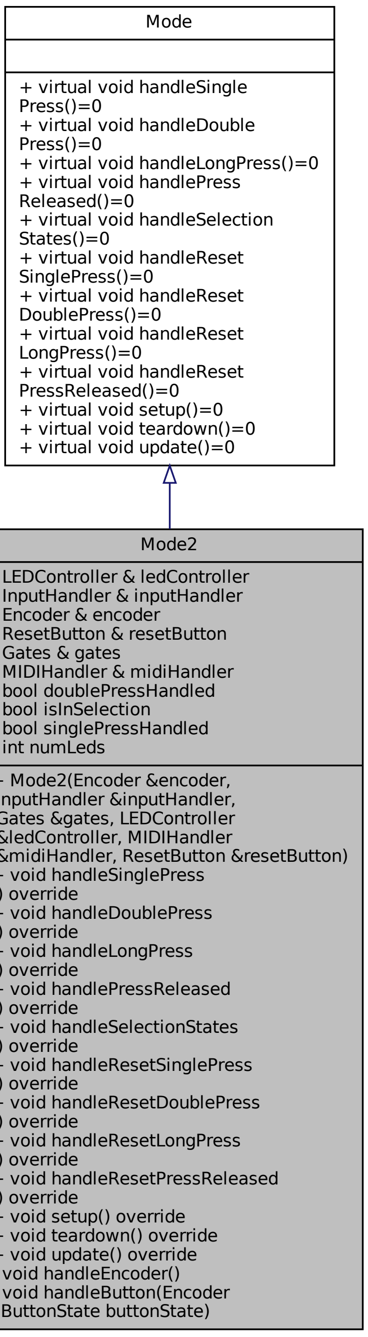
The documentation for this class was generated from the following files:

- include/Mode1.h
- src/Mode1.cpp

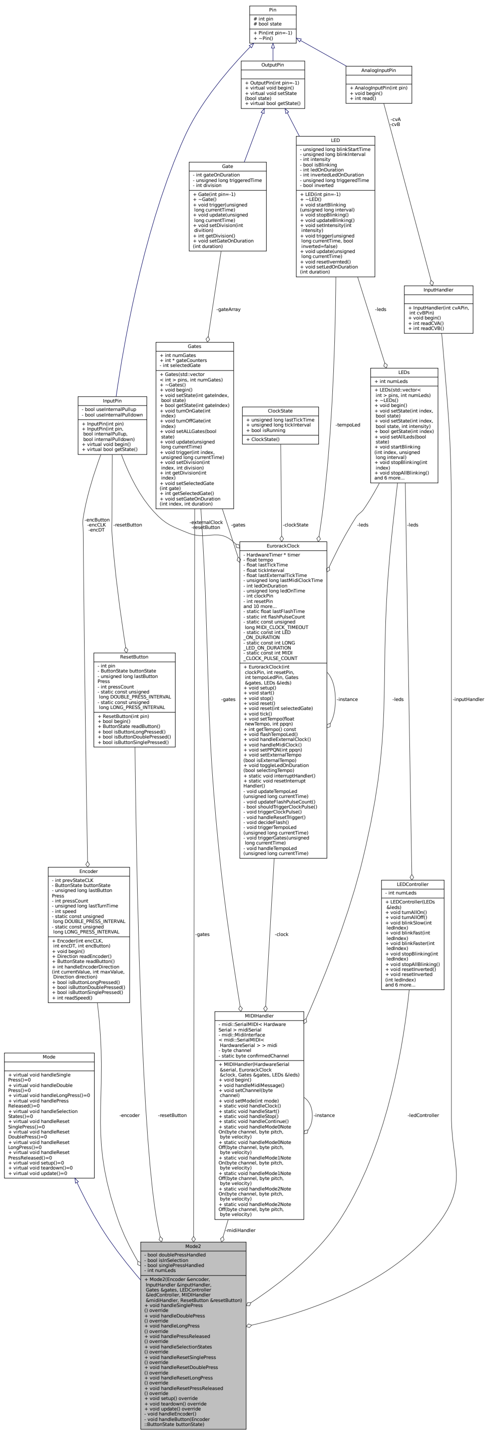
4.17 Mode2 Class Reference

```
#include <Mode2.h>
```

Inheritance diagram for Mode2:


Mode
+ virtual void handleSingle Press()=0
+ virtual void handleDouble Press()=0
+ virtual void handleLongPress()=0
+ virtual void handlePress Released()=0
+ virtual void handleSelection States()=0
+ virtual void handleReset SinglePress()=0
+ virtual void handleReset DoublePress()=0
+ virtual void handleReset LongPress()=0
+ virtual void handleReset PressReleased()=0
+ virtual void setup()=0
+ virtual void teardown()=0
+ virtual void update()=0
Mode2
- LEDController & ledController
- InputHandler & inputHandler
- Encoder & encoder
- ResetButton & resetButton
- Gates & gates
- MIDIHandler & midiHandler
- bool doublePressHandled
- bool isInSelection
- bool singlePressHandled
- int numLeds
+ Mode2(Encoder &encoder, InputHandler &inputHandler, Gates &gates, LEDController &ledController, MIDIHandler &midiHandler, ResetButton &resetButton)
+ void handleSinglePress () override
+ void handleDoublePress () override
+ void handleLongPress () override
+ void handlePressReleased () override
+ void handleSelectionStates () override
+ void handleResetSinglePress () override
+ void handleResetDoublePress () override
+ void handleResetLongPress () override
+ void handleResetPressReleased () override
+ void setup() override
+ void teardown() override
+ void update() override
- void handleEncoder()
- void handleButton(Encoder ::ButtonState buttonState)

Collaboration diagram for Mode2:



Public Member Functions

- Mode2 (Encoder &encoder, InputHandler &inputHandler, Gates &gates, LEDController &ledController, MIDIHandler &midiHandler, ResetButton &resetButton)
- void handleSinglePress () override
- void handleDoublePress () override
- void handleLongPress () override

- void handlePressReleased () override
- void handleSelectionStates () override
- void handleResetSinglePress () override
- void handleResetDoublePress () override
- void handleResetLongPress () override
- void handleResetPressReleased () override
- void setup () override
- void teardown () override
- void update () override

Private Member Functions

- void handleEncoder ()
- void handleButton (Encoder::ButtonState buttonState)

Private Attributes

- LEDController & ledController
- InputHandler & inputHandler
- Encoder & encoder
- ResetButton & resetButton
- Gates & gates
- MIDIHandler & midiHandler
- bool doublePressHandled = false
- bool isInSelection = false
- bool singlePressHandled = false
- int numLeds = 8

4.17.1 Constructor & Destructor Documentation

4.17.1.1 Mode2()

```
Mode2::Mode2 (
            Encoder & encoder,
            InputHandler & inputHandler,
            Gates & gates,
            LEDController & ledController,
            MIDIHandler & midiHandler,
            ResetButton & resetButton )
```

4.17.2 Member Function Documentation

4.17.2.1 handleButton()

```
void Mode2::handleButton (
            Encoder::ButtonState buttonState )  [private]
```

4.17.2.2 handleDoublePress()

```
void Mode2::handleDoublePress ( )  [override], [virtual]
```

Implements Mode.

4.17.2.3 handleEncoder()

```
void Mode2::handleEncoder ( )  [private]
```

4.17.2.4 handleLongPress()

```
void Mode2::handleLongPress ( )  [override], [virtual]
```

Implements Mode.

4.17.2.5 handlePressReleased()

```
void Mode2::handlePressReleased ( )  [override], [virtual]
```

Implements Mode.

4.17.2.6 handleResetDoublePress()

```
void Mode2::handleResetDoublePress ( )  [override], [virtual]
```

Implements Mode.

4.17.2.7 handleResetLongPress()

```
void Mode2::handleResetLongPress ( )  [override], [virtual]
```

Implements Mode.

4.17.2.8 handleResetPressReleased()

```
void Mode2::handleResetPressReleased ( )  [override], [virtual]
```

Implements Mode.

4.17.2.9 handleResetSinglePress()

```
void Mode2::handleResetSinglePress ( )  [override], [virtual]
```

Implements Mode.

4.17.2.10 handleSelectionStates()

```
void Mode2::handleSelectionStates ( )  [override], [virtual]
```

Implements Mode.

4.17.2.11 handleSinglePress()

```
void Mode2::handleSinglePress ( )  [override], [virtual]
```

Implements Mode.

4.17.2.12 setup()

```
void Mode2::setup ( )  [override], [virtual]
```

Implements Mode.

4.17.2.13 teardown()

```
void Mode2::teardown ( )  [override], [virtual]
```

Implements Mode.

4.17.2.14 update()

```
void Mode2::update ( )  [override], [virtual]
```

Implements Mode.

4.17.3 Member Data Documentation

4.17.3.1 doublePressHandled

```
bool Mode2::doublePressHandled = false  [private]
```

4.17.3.2 encoder

```
Encoder& Mode2::encoder  [private]
```

4.17.3.3 gates

```
Gates& Mode2::gates  [private]
```

4.17.3.4 inputHandler

```
InputHandler& Mode2::inputHandler  [private]
```

4.17.3.5 isInSelection

```
bool Mode2::isInSelection = false  [private]
```

4.17.3.6 ledController

```
LEDController& Mode2::ledController  [private]
```

4.17.3.7 midiHandler

```
MIDIHandler& Mode2::midiHandler  [private]
```

4.17.3.8 numLeds

```
int Mode2::numLeds = 8  [private]
```

4.17.3.9 resetButton

```
ResetButton& Mode2::resetButton  [private]
```

4.17.3.10 singlePressHandled

```
bool Mode2::singlePressHandled = false  [private]
```

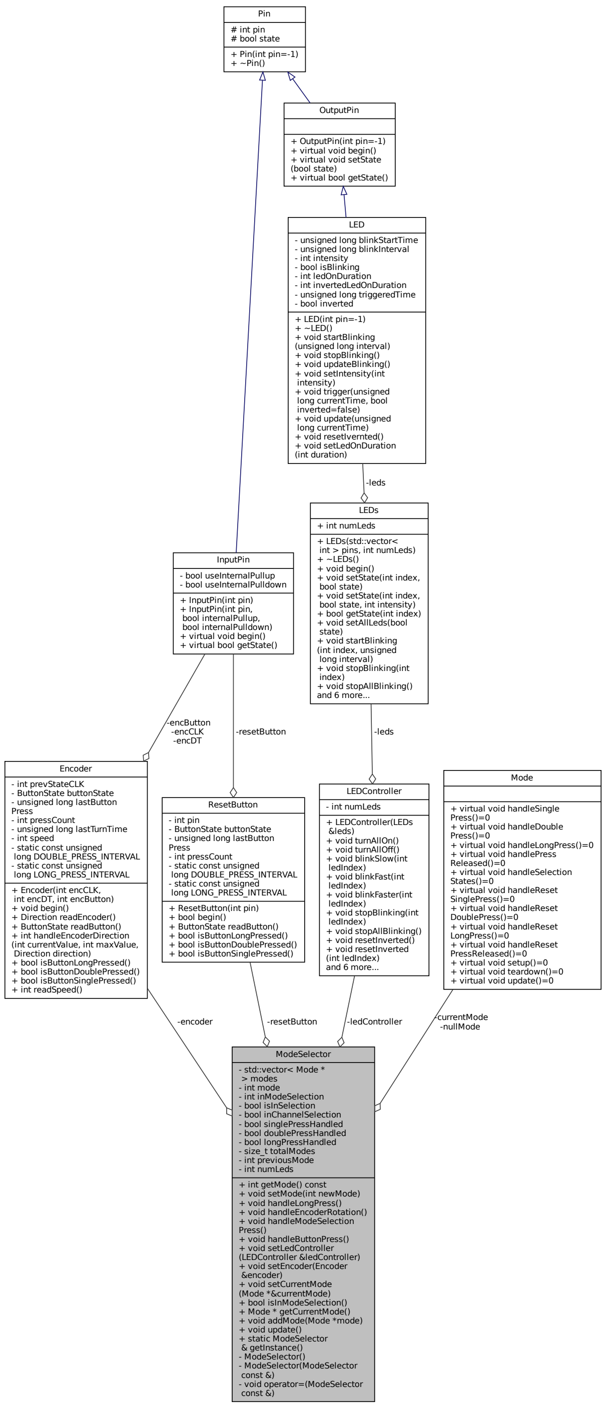
The documentation for this class was generated from the following files:

- include/Mode2.h
- src/Mode2.cpp

4.18 ModeSelector Class Reference

```
#include <ModeSelector.h>
```

Collaboration diagram for ModeSelector:


Pin
# int pin
# bool state
+ Pin(int pin=-1)
+ ~Pin()
OutputPin
+ OutputPin(int pin=-1)
+ virtual void begin()
+ virtual void setState (bool state)
+ virtual bool getState()
LED
- unsigned long blinkStartTime
- unsigned long blinkInterval
- int intensity
- bool isBlinking
- int ledOnDuration
- int invertedLedOnDuration
- unsigned long triggeredTime
- bool inverted
+ LED(int pin=-1)
+ ~LED()
+ void startBlinking (unsigned long interval)
+ void stopBlinking()
+ void updateBlinking()
+ void setIntensity(int intensity)
+ void trigger(unsigned long currentTime, bool inverted=false)
+ void update(unsigned long currentTime)
+ void resetIvernted()
+ void setLedOnDuration (int duration)
-leds
LEDs
+ int numLeds
+ LEDs(std::vector< int > pins, int numLeds)
+ ~LEDs()
+ void begin()
+ void setState(int index, bool state)
+ void setState(int index, bool state, int intensity)
+ bool getState(int index)
+ void setAllLeds(bool state)
+ void startBlinking (int index, unsigned long interval)
+ void stopBlinking(int index)
+ void stopAllBlinking()
and 6 more...
InputPin
- bool useInternalPullup
- bool useInternalPulldown
+ InputPin(int pin)
+ InputPin(int pin, bool internalPullup, bool internalPulldown)
+ virtual void begin()
+ virtual bool getState()
-encButton
-encCLK
-encDT
-resetButton
-leds
Encoder
- int prevStateCLK
- ButtonState buttonState
- unsigned long lastButtonPress
- int pressCount
- unsigned long lastTurnTime
- int speed
- static const unsigned long DOUBLE_PRESS_INTERVAL
- static const unsigned long LONG_PRESS_INTERVAL
+ Encoder(int encCLK, int encDT, int encButton)
+ void begin()
+ Direction readEncoder()
+ ButtonState readButton()
+ int handleEncoderDirection (int currentValue, int maxValue, Direction direction)
+ bool isButtonLongPressed()
+ bool isButtonDoublePressed()
+ bool isButtonSinglePressed()
+ int readSpeed()
ResetButton
- int pin
- ButtonState buttonState
- unsigned long lastButtonPress
- int pressCount
- static const unsigned long DOUBLE_PRESS_INTERVAL
- static const unsigned long LONG_PRESS_INTERVAL
+ ResetButton(int pin)
+ bool begin()
+ ButtonState readButton()
+ bool isButtonLongPressed()
+ bool isButtonDoublePressed()
+ bool isButtonSinglePressed()
LEDController
- int numLeds
+ LEDController(LEDs &leds)
+ void turnAllOn()
+ void turnAllOff()
+ void blinkSlow(int ledIndex)
+ void blinkFast(int ledIndex)
+ void blinkFaster(int ledIndex)
+ void stopBlinking(int ledIndex)
+ void stopAllBlinking()
+ void resetInverted()
+ void resetInverted (int ledIndex)
and 6 more...
Mode
+ virtual void handleSingle Press()=0
+ virtual void handleDouble Press()=0
+ virtual void handleLongPress()=0
+ virtual void handlePress Released()=0
+ virtual void handleSelection States()=0
+ virtual void handleReset SinglePress()=0
+ virtual void handleReset DoublePress()=0
+ virtual void handleReset LongPress()=0
+ virtual void handleReset PressReleased()=0
+ virtual void setup()=0
+ virtual void teardown()=0
+ virtual void update()=0
-encoder
-resetButton
-ledController
-currentMode
-nullMode
ModeSelector
- std::vector< Mode * > modes
- int mode
- int inModeSelection
- bool isInSelection
- bool inChannelSelection
- bool singlePressHandled
- bool doublePressHandled
- bool longPressHandled
- size_t totalModes
- int previousMode
- int numLeds
+ int getMode() const
+ void setMode(int newMode)
+ void handleLongPress()
+ void handleEncoderRotation()
+ void handleModeSelection Press()
+ void handleButtonPress()
+ void setLedController (LEDController &ledController)
+ void setEncoder(Encoder &encoder)
+ void setCurrentMode (Mode *&currentMode)
+ bool isInModeSelection()
+ Mode * getCurrentMode()
+ void addMode(Mode *mode)
+ void update()
+ static ModeSelector & getInstance()
- ModeSelector()
- ModeSelector(ModeSelector const &)
- void operator=(ModeSelector const &)

Public Member Functions

- int getMode () const

- void setMode (int newMode)
- void handleLongPress ()
- void handleEncoderRotation ()
- void handleModeSelectionPress ()
- void handleButtonPress ()
- void setLedController (LEDController &ledController)
- void setEncoder (Encoder &encoder)
- void setCurrentMode (Mode ∗&currentMode)
- bool isInModeSelection ()
- Mode ∗ getCurrentMode ()
- void addMode (Mode ∗mode)
- void update ()

Static Public Member Functions

- static ModeSelector & getInstance ()

Private Member Functions

- ModeSelector ()
- ModeSelector (ModeSelector const &)
- void operator= (ModeSelector const &)

Private Attributes

- std::vector< Mode ∗ > modes
- Mode ∗ nullMode = nullptr
- Mode ∗& currentMode
- int mode
- int inModeSelection = false
- LEDController ∗ ledController
- Encoder ∗ encoder
- ResetButton ∗ resetButton
- bool isInSelection
- bool inChannelSelection
- bool singlePressHandled
- bool doublePressHandled
- bool longPressHandled
- size_t totalModes = modes.size()
- int previousMode = -1
- int numLeds

4.18.1 Constructor & Destructor Documentation

4.18.1.1 ModeSelector() [1/2]

```
ModeSelector::ModeSelector ( )  [private]
```

4.18.1.2 ModeSelector() [2/2]

```
ModeSelector::ModeSelector (
            ModeSelector const & )  [private]
```

4.18.2 Member Function Documentation

4.18.2.1 addMode()

```
void ModeSelector::addMode (
            Mode * mode )
```

4.18.2.2 getCurrentMode()

```
Mode * ModeSelector::getCurrentMode ( )
```

4.18.2.3 getInstance()

```
ModeSelector & ModeSelector::getInstance ( )  [static]
```

4.18.2.4 getMode()

```
int ModeSelector::getMode ( ) const
```

4.18.2.5 handleButtonPress()

```
void ModeSelector::handleButtonPress ( )
```

4.18.2.6 handleEncoderRotation()

```
void ModeSelector::handleEncoderRotation ( )
```

4.18.2.7 handleLongPress()

```
void ModeSelector::handleLongPress ( )
```

4.18.2.8 handleModeSelectionPress()

```
void ModeSelector::handleModeSelectionPress ( )
```

4.18.2.9 isInModeSelection()

```
bool ModeSelector::isInModeSelection ( )
```

4.18.2.10 operator=()

```
void ModeSelector::operator= (
             ModeSelector const &  )  [private]
```

4.18.2.11 setCurrentMode()

```
void ModeSelector::setCurrentMode (
             Mode *& currentMode )
```

4.18.2.12 setEncoder()

```
void ModeSelector::setEncoder (
             Encoder & encoder )
```

4.18.2.13 setLedController()

```
void ModeSelector::setLedController (
             LEDController & ledController )
```

4.18.2.14 setMode()

```
void ModeSelector::setMode (
             int newMode )
```

4.18.2.15 update()

```
void ModeSelector::update ( )
```

4.18.3 Member Data Documentation

4.18.3.1 currentMode

```
Mode*& ModeSelector::currentMode  [private]
```

4.18.3.2 doublePressHandled

```
bool ModeSelector::doublePressHandled  [private]
```

4.18.3.3 encoder

```
Encoder* ModeSelector::encoder  [private]
```

4.18.3.4 inChannelSelection

```
bool ModeSelector::inChannelSelection  [private]
```

4.18.3.5 inModeSelection

```
int ModeSelector::inModeSelection = false  [private]
```

4.18.3.6 isInSelection

```
bool ModeSelector::isInSelection  [private]
```

4.18.3.7 ledController

```
LEDController* ModeSelector::ledController  [private]
```

4.18.3.8 longPressHandled

```
bool ModeSelector::longPressHandled  [private]
```

4.18.3.9 mode

```
int ModeSelector::mode  [private]
```

4.18.3.10 modes

```
std::vector<Mode*> ModeSelector::modes  [private]
```

4.18.3.11 nullMode

```
Mode* ModeSelector::nullMode = nullptr  [private]
```

4.18.3.12 numLeds

```
int ModeSelector::numLeds  [private]
```

4.18.3.13 previousMode

```
int ModeSelector::previousMode = -1  [private]
```

4.18.3.14 resetButton

```
ResetButton* ModeSelector::resetButton  [private]
```

4.18.3.15 singlePressHandled

```
bool ModeSelector::singlePressHandled  [private]
```

4.18.3.16 totalModes

```
size_t ModeSelector::totalModes = modes.size()  [private]
```

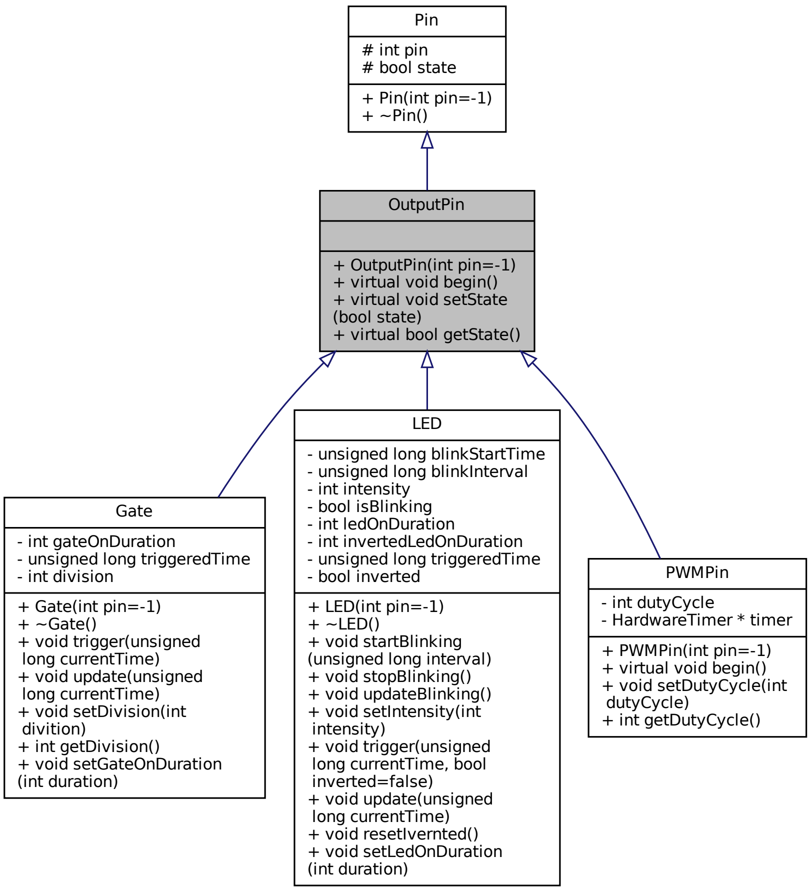
The documentation for this class was generated from the following files:

- include/ModeSelector.h
- src/ModeSelector.cpp

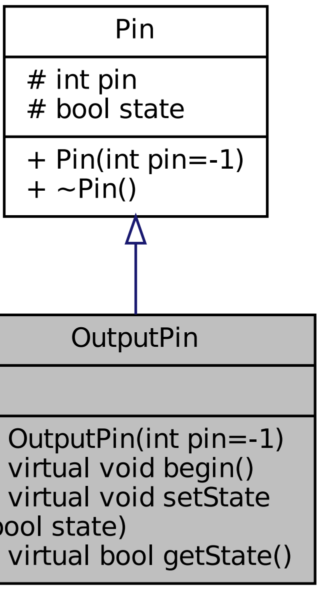
4.19 OutputPin Class Reference

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

Inheritance diagram for OutputPin:


Pin
# int pin
# bool state
+ Pin(int pin=-1)
+ ~Pin()
OutputPin
+ OutputPin(int pin=-1)
+ virtual void begin()
+ virtual void setState (bool state)
+ virtual bool getState()
Gate
- int gateOnDuration
- unsigned long triggeredTime
- int division
+ Gate(int pin=-1)
+ ~Gate()
+ void trigger(unsigned long currentTime)
+ void update(unsigned long currentTime)
+ void setDivision(int divition)
+ int getDivision()
+ void setGateOnDuration (int duration)
LED
- unsigned long blinkStartTime
- unsigned long blinkInterval
- int intensity
- bool isBlinking
- int ledOnDuration
- int invertedLedOnDuration
- unsigned long triggeredTime
- bool inverted
+ LED(int pin=-1)
+ ~LED()
+ void startBlinking (unsigned long interval)
+ void stopBlinking()
+ void updateBlinking()
+ void setIntensity(int intensity)
+ void trigger(unsigned long currentTime, bool inverted=false)
+ void update(unsigned long currentTime)
+ void resetIvernted()
+ void setLedOnDuration (int duration)
PWMPin
- int dutyCycle
- HardwareTimer * timer
+ PWMPin(int pin=-1)
+ virtual void begin()
+ void setDutyCycle(int dutyCycle)
+ int getDutyCycle()

Collaboration diagram for OutputPin:



Public Member Functions

- OutputPin (int pin=-1)
- virtual void begin ()
- virtual void setState (bool state)
- virtual bool getState ()

Additional Inherited Members

4.19.1 Constructor & Destructor Documentation

4.19.1.1 OutputPin()

```
OutputPin::OutputPin (
            int pin = -1 )
```

4.19.2 Member Function Documentation

4.19.2.1 begin()

```
void OutputPin::begin ( )  [virtual]
```

Reimplemented in PWMPin.

4.19.2.2 getState()

```
bool OutputPin::getState ( )  [virtual]
```

4.19.2.3 setState()

```
void OutputPin::setState (
             bool state )  [virtual]
```

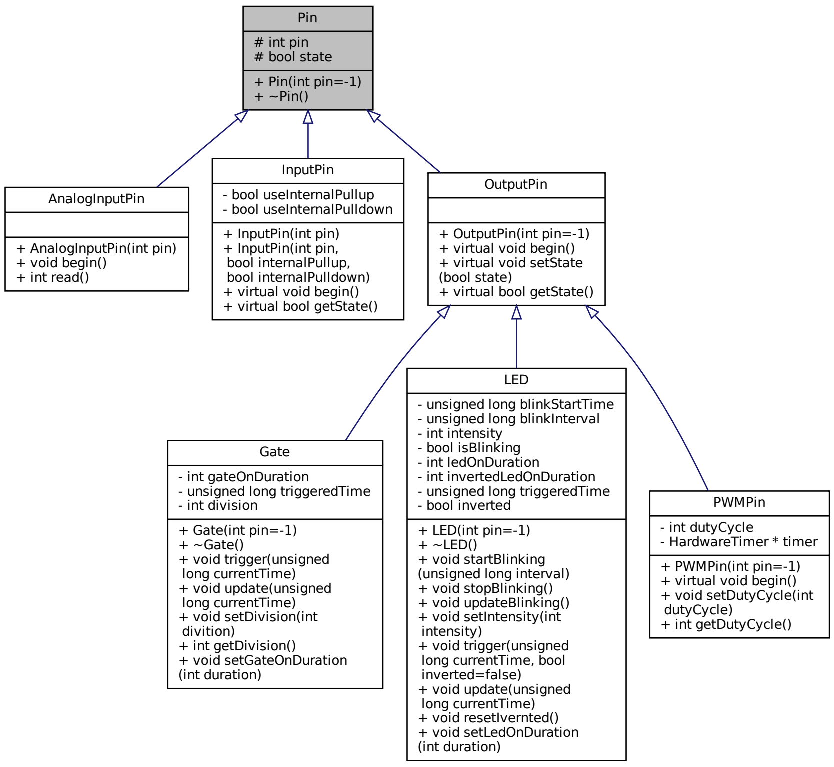
The documentation for this class was generated from the following files:

- include/Pin.h
- src/Pin.cpp

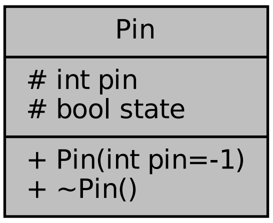
4.20 Pin Class Reference

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

Inheritance diagram for Pin:


Pin
# int pin
# bool state
+ Pin(int pin=-1)
+ ~Pin()
AnalogInputPin
+ AnalogInputPin(int pin)
+ void begin()
+ int read()
InputPin
- bool useInternalPullup
- bool useInternalPulldown
+ InputPin(int pin)
+ InputPin(int pin, bool internalPullup, bool internalPulldown)
+ virtual void begin()
+ virtual bool getState()
OutputPin
+ OutputPin(int pin=-1)
+ virtual void begin()
+ virtual void setState (bool state)
+ virtual bool getState()
Gate
- int gateOnDuration
- unsigned long triggeredTime
- int division
+ Gate(int pin=-1)
+ ~Gate()
+ void trigger(unsigned long currentTime)
+ void update(unsigned long currentTime)
+ void setDivision(int divition)
+ int getDivision()
+ void setGateOnDuration (int duration)
LED
- unsigned long blinkStartTime
- unsigned long blinkInterval
- int intensity
- bool isBlinking
- int ledOnDuration
- int invertedLedOnDuration
- unsigned long triggeredTime
- bool inverted
+ LED(int pin=-1)
+ ~LED()
+ void startBlinking (unsigned long interval)
+ void stopBlinking()
+ void updateBlinking()
+ void setIntensity(int intensity)
+ void trigger(unsigned long currentTime, bool inverted=false)
+ void update(unsigned long currentTime)
+ void resetIverted()
+ void setLedOnDuration (int duration)
PWMPin
- int dutyCycle
- HardwareTimer * timer
+ PWMPin(int pin=-1)
+ virtual void begin()
+ void setDutyCycle(int dutyCycle)
+ int getDutyCycle()

Collaboration diagram for Pin:



Public Member Functions

- Pin (int pin=-1)
- ∼Pin ()

Protected Attributes

- int pin
- bool state

4.20.1 Constructor & Destructor Documentation

4.20.1.1 Pin()

```
Pin::Pin (
            int pin = -1 )
```

4.20.1.2 ∼Pin()

```
Pin::∼Pin ( )
```

4.20.2 Member Data Documentation

4.20.2.1 pin

```
int Pin::pin  [protected]
```

4.20.2.2 state

```
bool Pin::state  [protected]
```

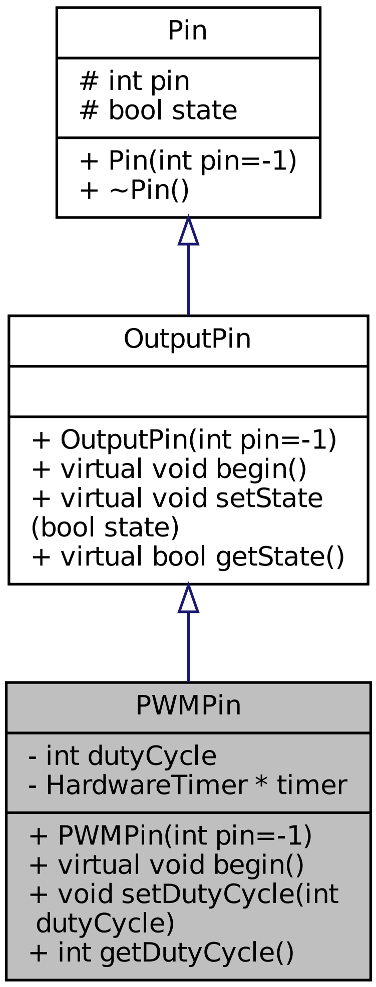
The documentation for this class was generated from the following files:

- include/Pin.h
- src/Pin.cpp

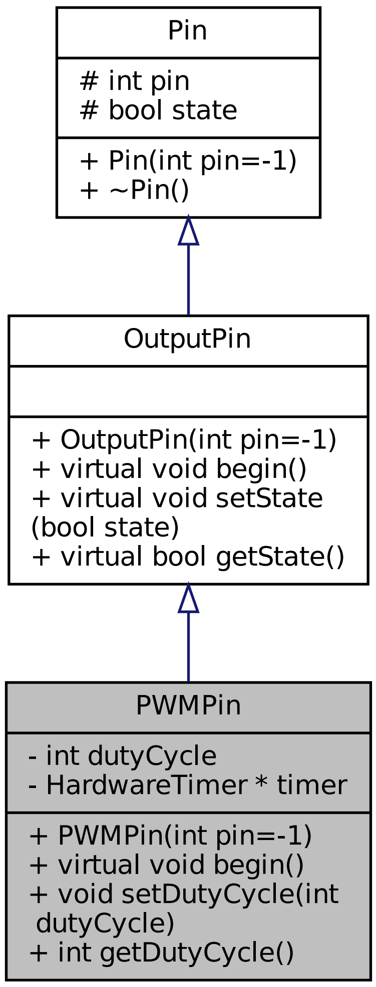
4.21 PWMPin Class Reference

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

Inheritance diagram for PWMPin:



Collaboration diagram for PWMPin:



Public Member Functions

- PWMPin (int pin=-1)
- virtual void begin ()
- void setDutyCycle (int dutyCycle)
- int getDutyCycle ()

Private Attributes

- int dutyCycle
- HardwareTimer ∗ timer

Additional Inherited Members

4.21.1 Constructor & Destructor Documentation

4.21.1.1 PWMPin()

```
PWMPin::PWMPin (
             int pin = -1 )
```

4.21.2 Member Function Documentation

4.21.2.1 begin()

```
void PWMPin::begin ( )  [virtual]
```

Reimplemented from OutputPin.

4.21.2.2 getDutyCycle()

```
int PWMPin::getDutyCycle ( )
```

4.21.2.3 setDutyCycle()

```
void PWMPin::setDutyCycle (
             int dutyCycle )
```

4.21.3 Member Data Documentation

4.21.3.1 dutyCycle

```
int PWMPin::dutyCycle  [private]
```

4.21.3.2 timer

```
HardwareTimer* PWMPin::timer  [private]
```

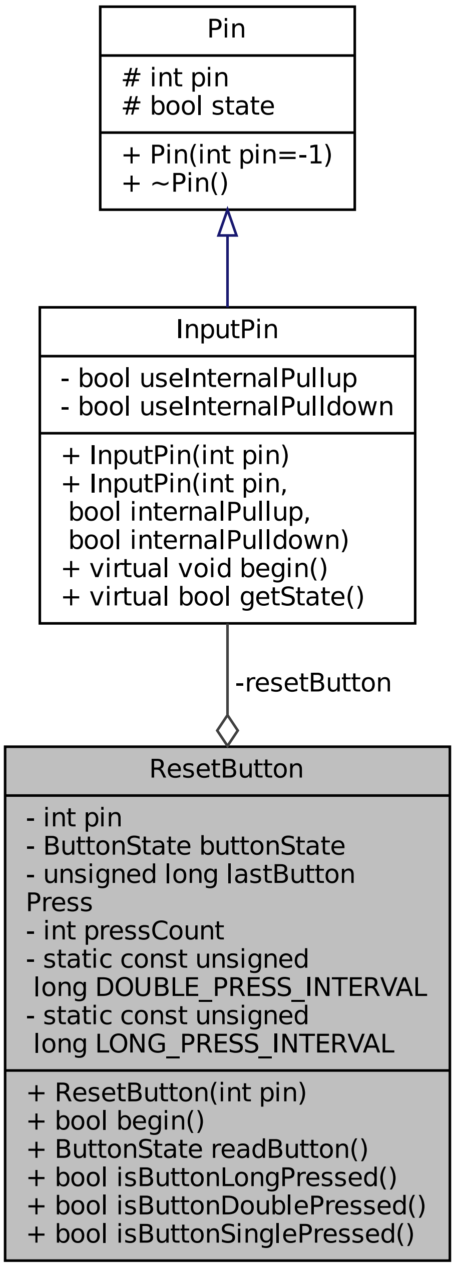
The documentation for this class was generated from the following files:

- include/Pin.h
- src/Pin.cpp

4.22 ResetButton Class Reference

```
#include <ResetButton.h>
```

Collaboration diagram for ResetButton:


Pin
# int pin
# bool state
+ Pin(int pin=-1)
+ ~Pin()
InputPin
- bool useInternalPullup
- bool useInternalPulldown
+ InputPin(int pin)
+ InputPin(int pin, bool internalPullup, bool internalPulldown)
+ virtual void begin()
+ virtual bool getState()
-resetButton
ResetButton
- int pin
- ButtonState buttonState
- unsigned long lastButtonPress
- int pressCount
- static const unsigned long DOUBLE_PRESS_INTERVAL
- static const unsigned long LONG_PRESS_INTERVAL
+ ResetButton(int pin)
+ bool begin()
+ ButtonState readButton()
+ bool isButtonLongPressed()
+ bool isButtonDoublePressed()
+ bool isButtonSinglePressed()

Public Types

- enum ButtonState { OPEN , PRESSED }

Public Member Functions

- ResetButton (int pin)

- bool begin ()
- ButtonState readButton ()
- bool isButtonLongPressed ()
- bool isButtonDoublePressed ()
- bool isButtonSinglePressed ()

Private Attributes

- int pin
- InputPin resetButton
- ButtonState buttonState
- unsigned long lastButtonPress
- int pressCount

Static Private Attributes

- static const unsigned long DOUBLE_PRESS_INTERVAL = 500
- static const unsigned long LONG_PRESS_INTERVAL = 1000

4.22.1 Member Enumeration Documentation

4.22.1.1 ButtonState

```
enum ResetButton::ButtonState
```

Enumerator

OPEN	
PRESSED	

4.22.2 Constructor & Destructor Documentation

4.22.2.1 ResetButton()

```
ResetButton::ResetButton (
            int pin )
```

4.22.3 Member Function Documentation

4.22.3.1 begin()

```
bool ResetButton::begin ( )
```

4.22.3.2 isButtonDoublePressed()

```
bool ResetButton::isButtonDoublePressed ( )
```

4.22.3.3 isButtonLongPressed()

```
bool ResetButton::isButtonLongPressed ( )
```

4.22.3.4 isButtonSinglePressed()

```
bool ResetButton::isButtonSinglePressed ( )
```

4.22.3.5 readButton()

```
ResetButton::ButtonState ResetButton::readButton ( )
```

4.22.4 Member Data Documentation

4.22.4.1 buttonState

```
ButtonState ResetButton::buttonState  [private]
```

4.22.4.2 DOUBLE_PRESS_INTERVAL

```
const unsigned long ResetButton::DOUBLE_PRESS_INTERVAL = 500  [static], [private]
```

4.22.4.3 lastButtonPress

```
unsigned long ResetButton::lastButtonPress  [private]
```

4.22.4.4 LONG_PRESS_INTERVAL

```
const unsigned long ResetButton::LONG_PRESS_INTERVAL = 1000  [static], [private]
```

4.22.4.5 pin

```
int ResetButton::pin  [private]
```

4.22.4.6 pressCount

```
int ResetButton::pressCount  [private]
```

4.22.4.7 resetButton

```
InputPin ResetButton::resetButton  [private]
```

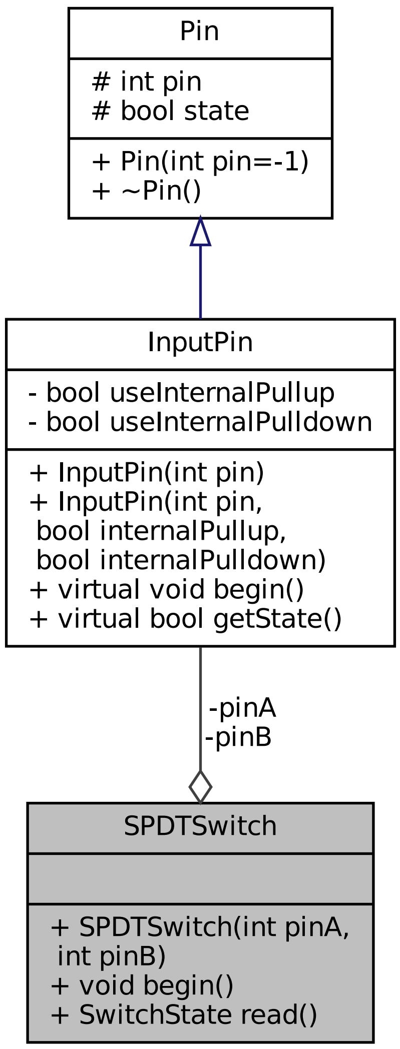
The documentation for this class was generated from the following files:

- include/ResetButton.h
- src/ResetButton.cpp

4.23 SPDTSwitch Class Reference

```
#include <SPDTSwitch.h>
```

Collaboration diagram for SPDTSwitch:



Public Member Functions

- SPDTSwitch (int pinA, int pinB)
- void begin ()
- SwitchState read ()

Private Attributes

- InputPin pinA
- InputPin pinB

4.23.1 Constructor & Destructor Documentation

4.23.1.1 SPDTSwitch()

```
SPDTSwitch::SPDTSwitch (
             int pinA,
             int pinB )
```

4.23.2 Member Function Documentation

4.23.2.1 begin()

```
void SPDTSwitch::begin ( )
```

4.23.2.2 read()

```
SwitchState SPDTSwitch::read ( )
```

4.23.3 Member Data Documentation

4.23.3.1 pinA

```
InputPin SPDTSwitch::pinA  [private]
```

4.23.3.2 pinB

```
InputPin SPDTSwitch::pinB  [private]
```

The documentation for this class was generated from the following files:

- include/SPDTSwitch.h
- src/SPDTSwitch.cpp

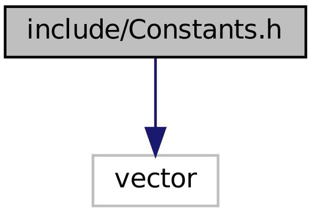
Chapter 5

File Documentation

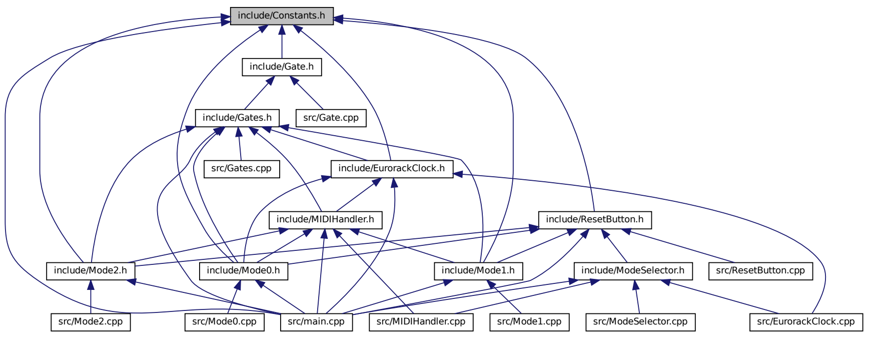
5.1 include/Constants.h File Reference

```
#include <vector>
```

Include dependency graph for Constants.h:



This graph shows which files directly or indirectly include this file:



Variables

- std::vector< int > musicalIntervals
- const int musicalIntervalsSize
- unsigned char internalPPQN

5.1.1 Variable Documentation

5.1.1.1 internalPPQN

```
unsigned char internalPPQN  [extern]
```

5.1.1.2 musicalIntervals

```
std::vector<int> musicalIntervals  [extern]
```

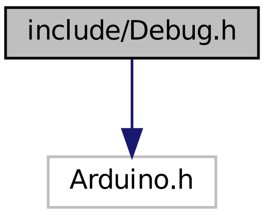
5.1.1.3 musicalIntervalsSize

```
const int musicalIntervalsSize  [extern]
```

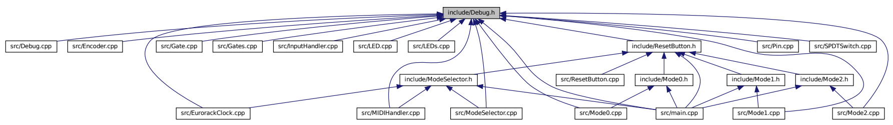
5.2 include/Debug.h File Reference

```
#include <Arduino.h>
```

Include dependency graph for Debug.h:



This graph shows which files directly or indirectly include this file:



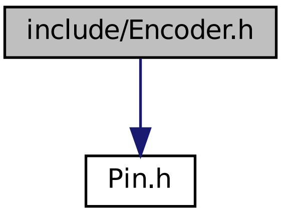
Classes

- class Debug

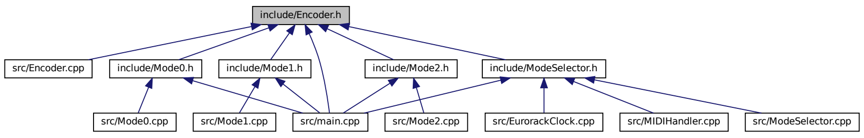
5.3 include/Encoder.h File Reference

```
#include "Pin.h"
```

Include dependency graph for Encoder.h:



This graph shows which files directly or indirectly include this file:



Classes

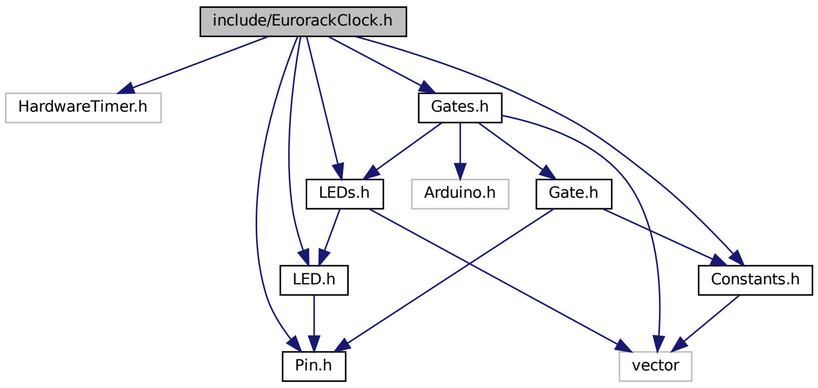
- class Encoder

5.4 include/EurorackClock.h File Reference

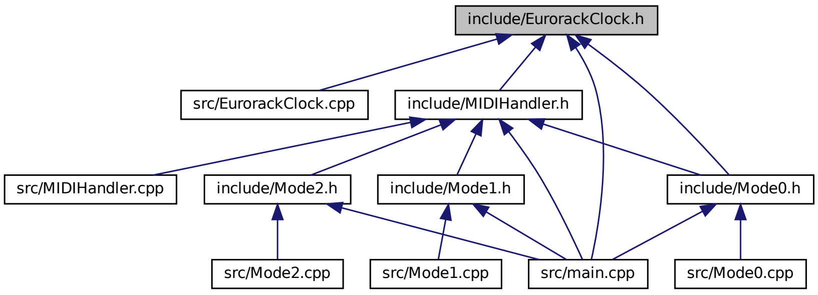
```
#include <HardwareTimer.h>
#include "LED.h"
#include "Pin.h"
#include "Gates.h"
#include "LEDs.h"
```

```
#include "Constants.h"
```

Include dependency graph for EurorackClock.h:



This graph shows which files directly or indirectly include this file:



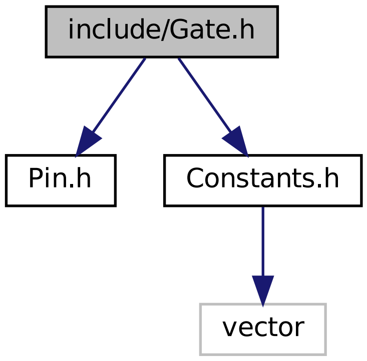
Classes

- struct ClockState
- class EurorackClock

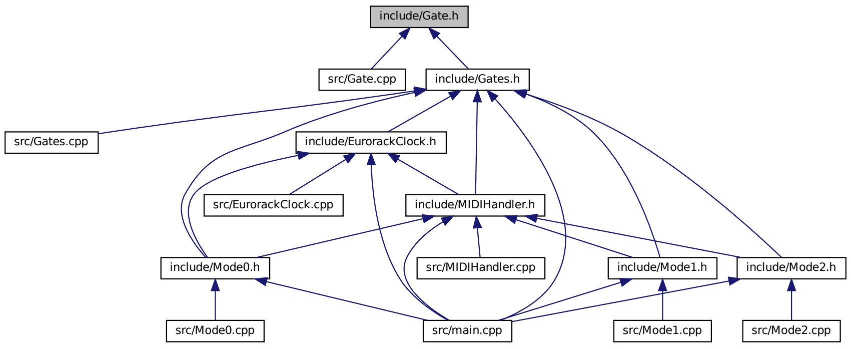
5.5 include/Gate.h File Reference

```
#include "Pin.h"
#include "Constants.h"
```

Include dependency graph for Gate.h:



This graph shows which files directly or indirectly include this file:



Classes

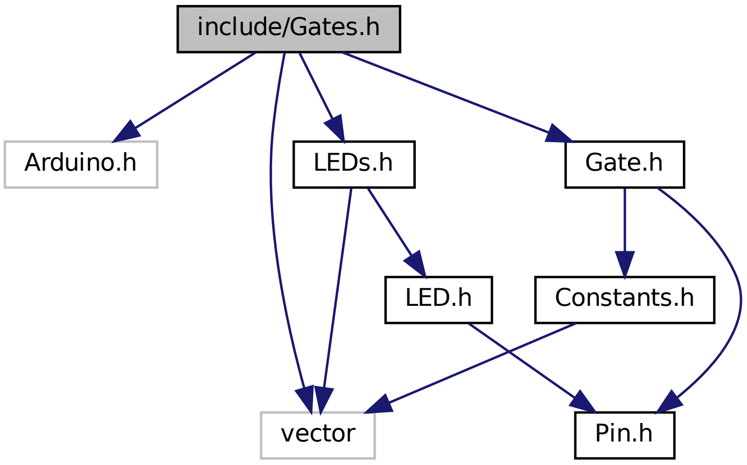
- class Gate

5.6 include/Gates.h File Reference

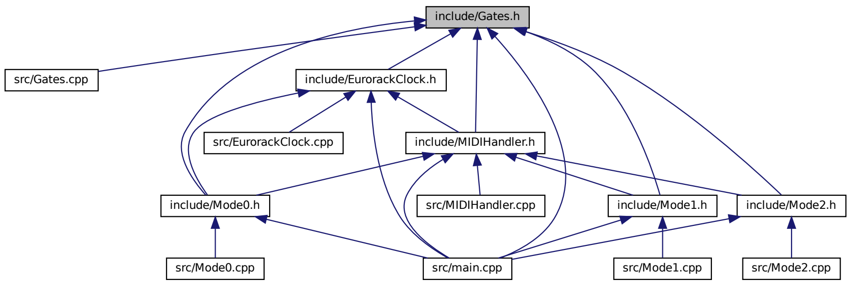
```
#include <Arduino.h>
#include "Gate.h"
#include "LEDs.h"
```

```
#include <vector>
```

Include dependency graph for Gates.h:



This graph shows which files directly or indirectly include this file:



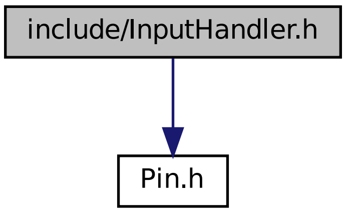
Classes

- class Gates

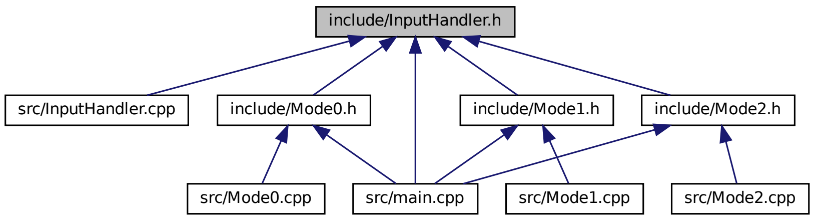
5.7 include/InputHandler.h File Reference

```
#include "Pin.h"
```

Include dependency graph for InputHandler.h:



This graph shows which files directly or indirectly include this file:



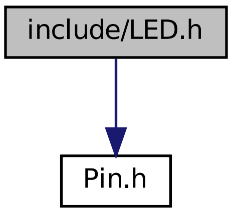
Classes

- class InputHandler

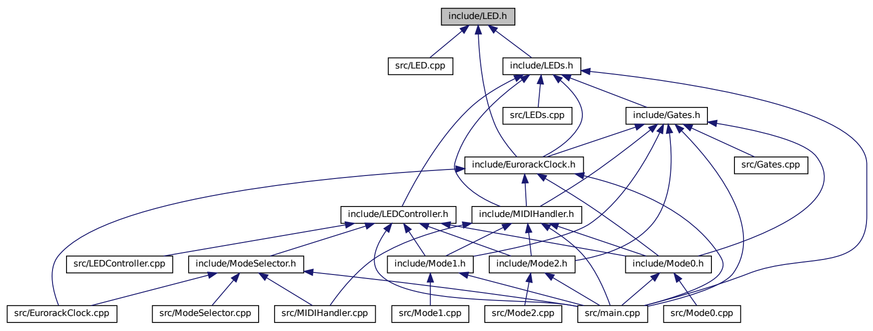
5.8 include/LED.h File Reference

```
#include "Pin.h"
```

Include dependency graph for LED.h:



This graph shows which files directly or indirectly include this file:



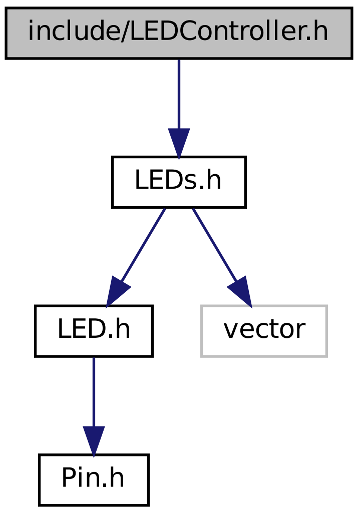
Classes

- class LED

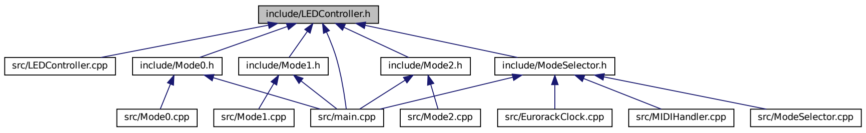
5.9 include/LEDController.h File Reference

```
#include "LEDs.h"
```

Include dependency graph for LEDController.h:



This graph shows which files directly or indirectly include this file:



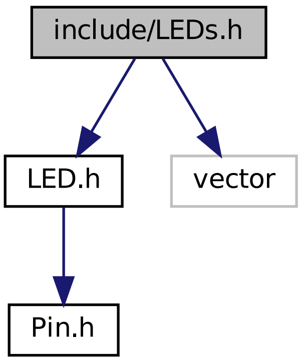
Classes

- class LEDController

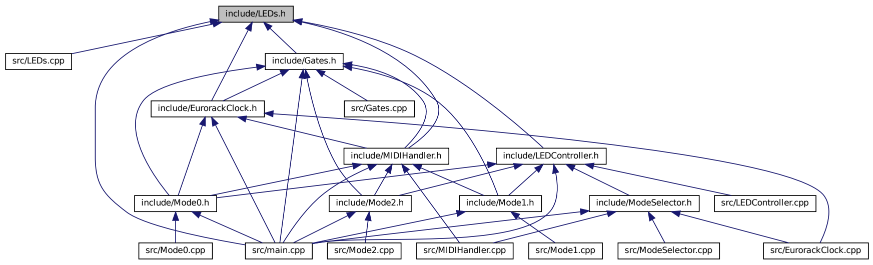
5.10 include/LEDs.h File Reference

```
#include "LED.h"
#include <vector>
```

Include dependency graph for LEDs.h:



This graph shows which files directly or indirectly include this file:



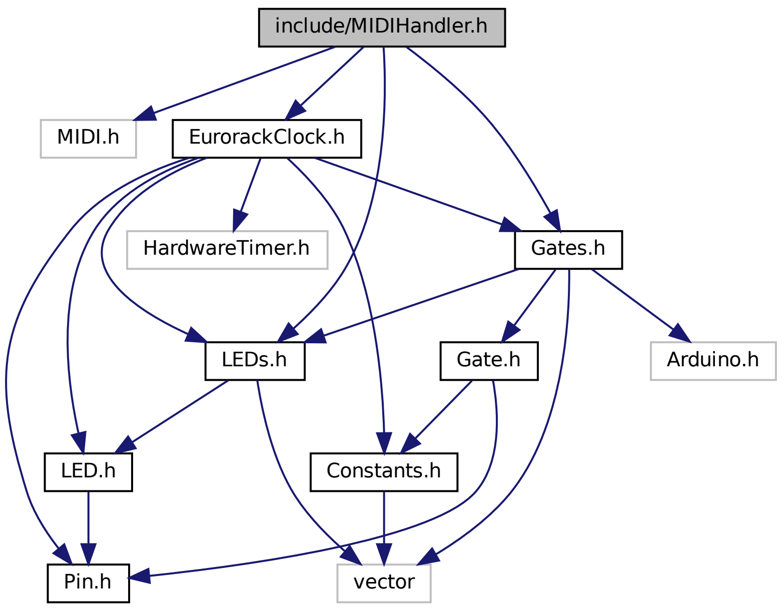
Classes

- class LEDs

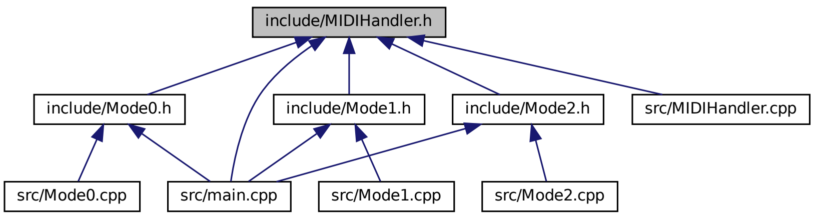
5.11 include/MIDIHandler.h File Reference

```
#include <MIDI.h>
#include "EurorackClock.h"
#include "Gates.h"
#include "LEDs.h"
```

Include dependency graph for MIDIHandler.h:



This graph shows which files directly or indirectly include this file:

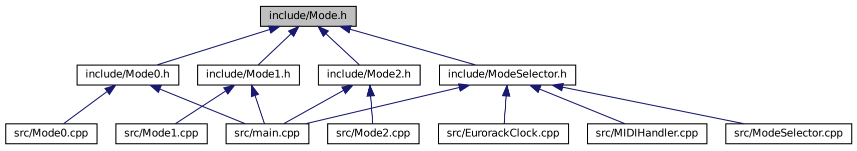


Classes

- class MIDIHandler

5.12 include/Mode.h File Reference

This graph shows which files directly or indirectly include this file:



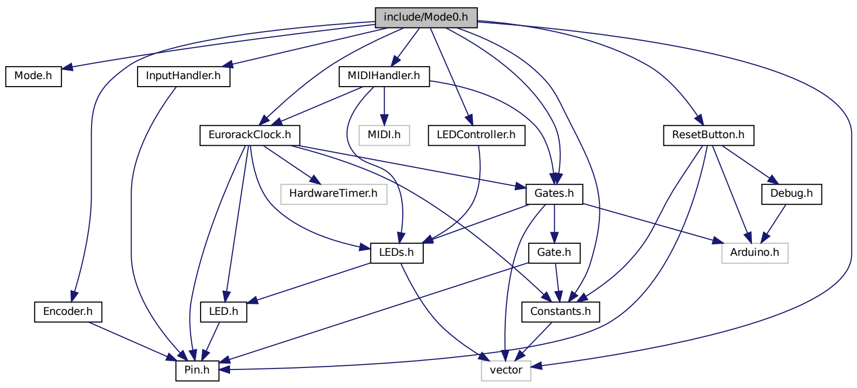
Classes

- class Mode

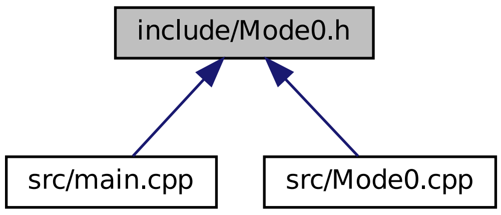
5.13 include/Mode0.h File Reference

```
#include "Mode.h"
#include "Encoder.h"
#include "Gates.h"
#include "LEDController.h"
#include "EurorackClock.h"
#include "MIDIHandler.h"
#include "Constants.h"
#include "ResetButton.h"
#include "InputHandler.h"
#include <vector>
```

Include dependency graph for Mode0.h:



This graph shows which files directly or indirectly include this file:



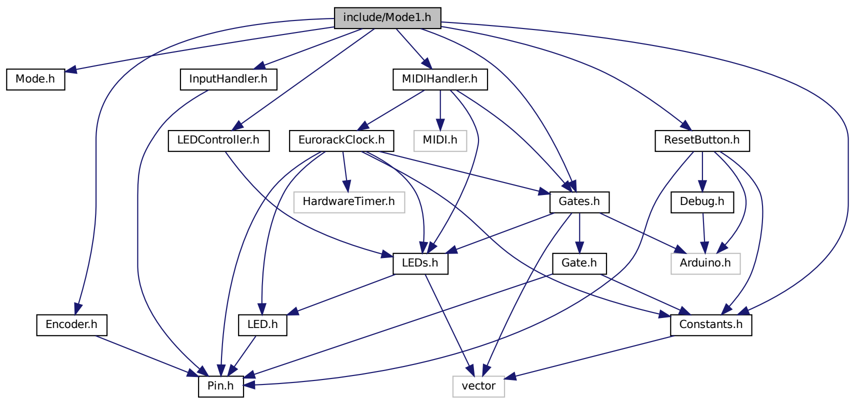
Classes

- class Mode0

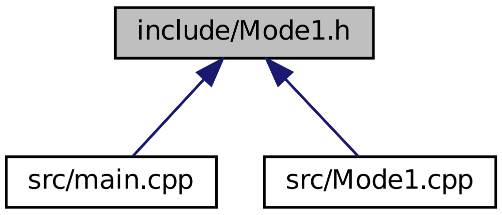
5.14 include/Mode1.h File Reference

```
#include "Mode.h"
#include "Encoder.h"
#include "Gates.h"
#include "LEDController.h"
#include "MIDIHandler.h"
#include "Constants.h"
#include "ResetButton.h"
#include "InputHandler.h"
```

Include dependency graph for Mode1.h:



This graph shows which files directly or indirectly include this file:



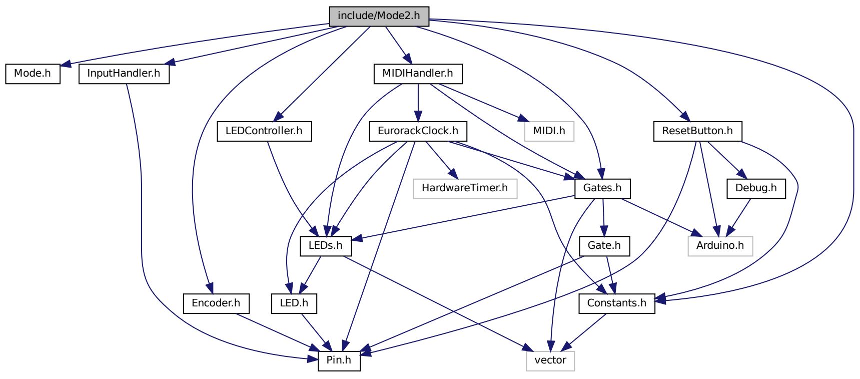
Classes

- class Mode1

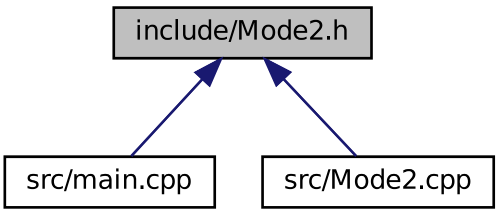
5.15 include/Mode2.h File Reference

```
#include "Mode.h"
#include "LEDController.h"
#include "Encoder.h"
#include "Gates.h"
#include "MIDIHandler.h"
#include "Constants.h"
#include "InputHandler.h"
#include "ResetButton.h"
```

Include dependency graph for Mode2.h:



This graph shows which files directly or indirectly include this file:



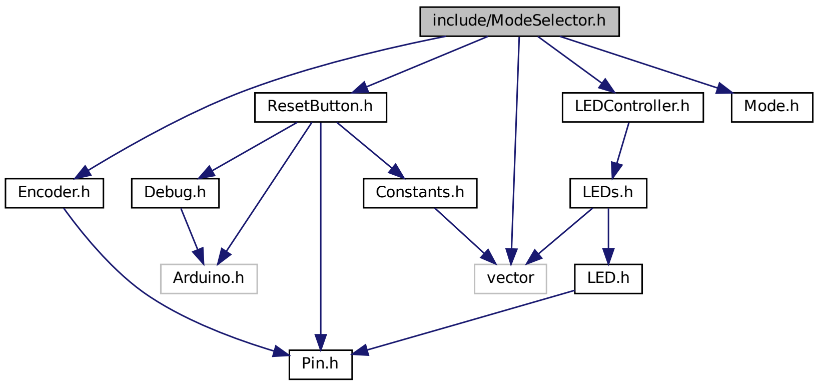
Classes

- class Mode2

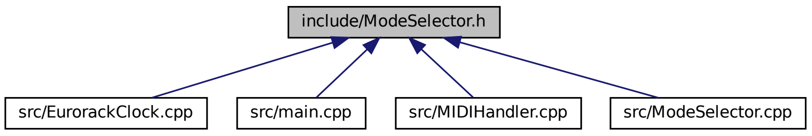
5.16 include/ModeSelector.h File Reference

```
#include <vector>
#include "LEDController.h"
#include "Encoder.h"
#include "Mode.h"
#include "ResetButton.h"
```

Include dependency graph for ModeSelector.h:



This graph shows which files directly or indirectly include this file:

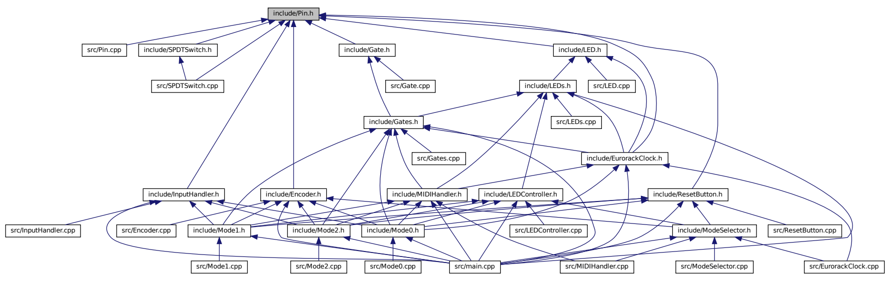


Classes

- class ModeSelector

5.17 include/Pin.h File Reference

This graph shows which files directly or indirectly include this file:



Classes

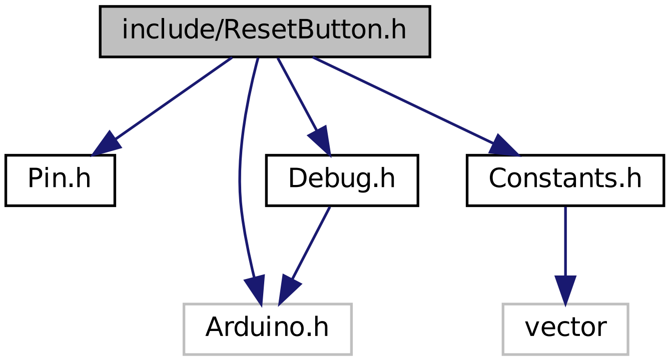
- class Pin
- class InputPin
- class AnalogInputPin
- class OutputPin
- class PWMPin

5.18 include/ResetButton.h File Reference

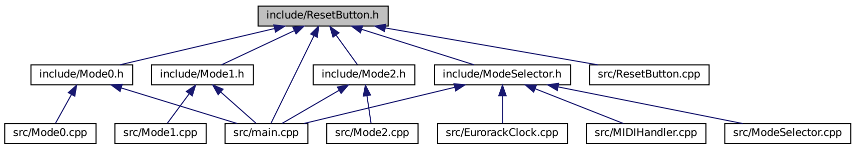
```
#include "Pin.h"
#include <Arduino.h>
#include "Debug.h"
```

```
#include "Constants.h"
```

Include dependency graph for ResetButton.h:



This graph shows which files directly or indirectly include this file:



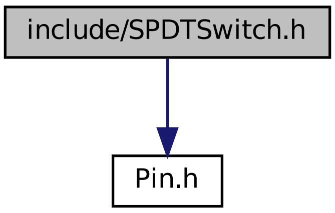
Classes

- class ResetButton

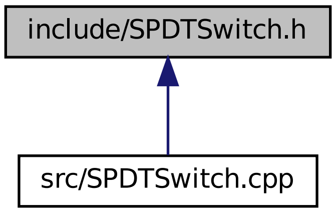
5.19 include/SPDTSwitch.h File Reference

```
#include "Pin.h"
```

Include dependency graph for SPDTSwitch.h:



This graph shows which files directly or indirectly include this file:



Classes

- class SPDTSwitch

Enumerations

- enum SwitchState { NEUTRAL , STATE_A , STATE_B }

5.19.1 Enumeration Type Documentation

5.19.1.1 SwitchState

```
enum SwitchState
```

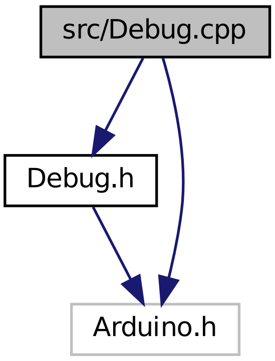
Enumerator

NEUTRAL	
STATE_A	
STATE_B	

5.20 src/Debug.cpp File Reference

```
#include "Debug.h"
#include <Arduino.h>
```

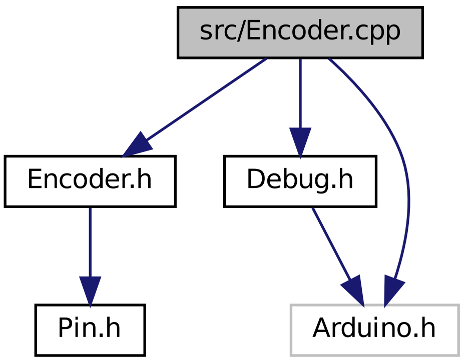
Include dependency graph for Debug.cpp:



5.21 src/Encoder.cpp File Reference

```
#include "Encoder.h"
#include "Debug.h"
#include <Arduino.h>
```

Include dependency graph for Encoder.cpp:



Macros

- #define DEBUG_PRINT(message) Debug::print(__FILE__, __LINE__, __func__, String(message))

5.21.1 Macro Definition Documentation

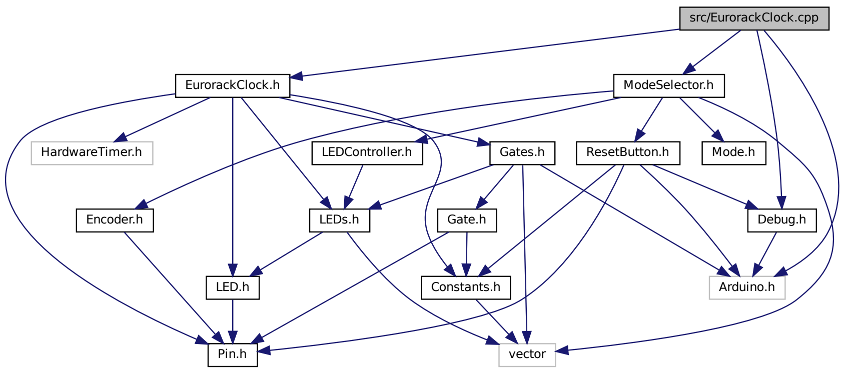
5.21.1.1 DEBUG_PRINT

```
#define DEBUG_PRINT(
              message ) Debug::print(__FILE__, __LINE__, __func__, String(message))
```

5.22 src/EurorackClock.cpp File Reference

```
#include "EurorackClock.h"
#include "Debug.h"
#include <Arduino.h>
#include "ModeSelector.h"
```

Include dependency graph for EurorackClock.cpp:



Macros

- #define DEBUG_PRINT(message) Debug::print(__FILE__, __LINE__, __func__, String(message))

5.22.1 Macro Definition Documentation

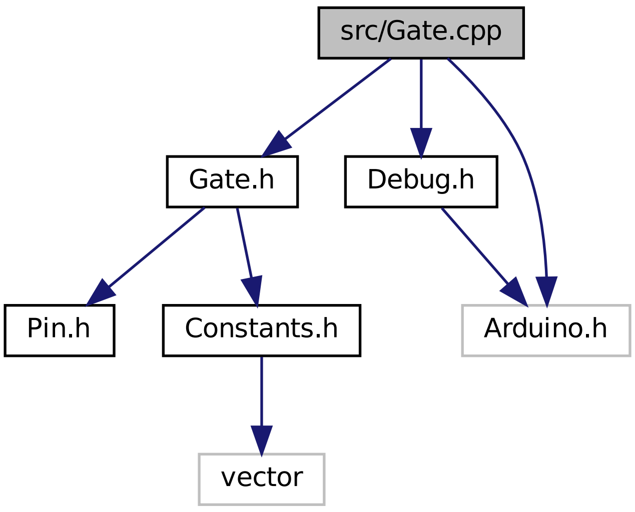
5.22.1.1 DEBUG_PRINT

```
#define DEBUG_PRINT(
              message ) Debug::print(__FILE__, __LINE__, __func__, String(message))
```

5.23 src/Gate.cpp File Reference

```
#include "Gate.h"
#include "Debug.h"
#include <Arduino.h>
```

Include dependency graph for Gate.cpp:



Macros

- #define DEBUG_PRINT(message) Debug::print(__FILE__, __LINE__, __func__, String(message))

5.23.1 Macro Definition Documentation

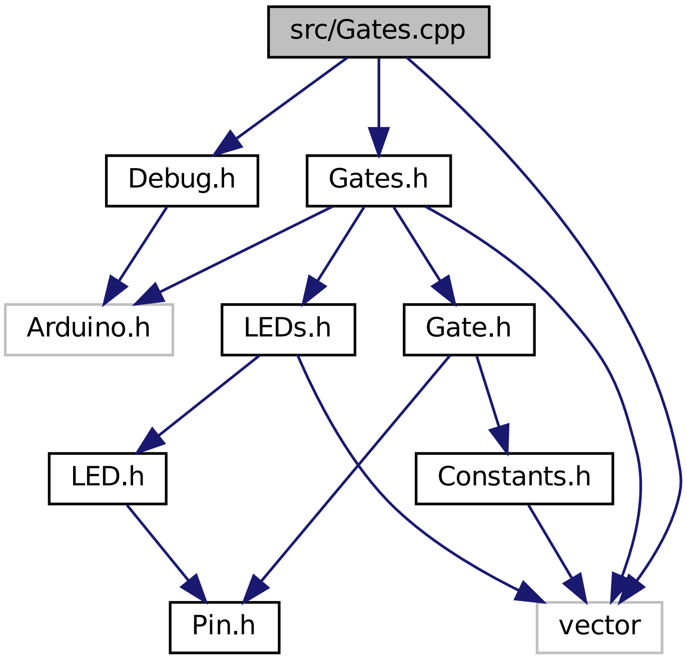
5.23.1.1 DEBUG_PRINT

```
#define DEBUG_PRINT(
            message ) Debug::print(__FILE__, __LINE__, __func__, String(message))
```

5.24 src/Gates.cpp File Reference

```
#include "Gates.h"
#include "Debug.h"
#include <vector>
```

Include dependency graph for Gates.cpp:



Macros

- #define DEBUG_PRINT(message) Debug::print(__FILE__, __LINE__, __func__, String(message))

5.24.1 Macro Definition Documentation

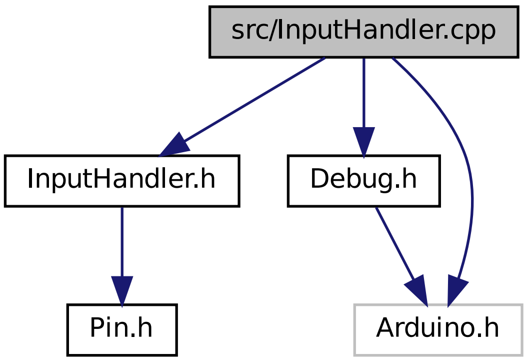
5.24.1.1 DEBUG_PRINT

```
#define DEBUG_PRINT(
            message ) Debug::print(__FILE__, __LINE__, __func__, String(message))
```

5.25 src/InputHandler.cpp File Reference

```
#include "InputHandler.h"
#include "Debug.h"
#include <Arduino.h>
```

Include dependency graph for InputHandler.cpp:



Macros

- #define DEBUG_PRINT(message) Debug::print(__FILE__, __LINE__, __func__, String(message))

5.25.1 Macro Definition Documentation

5.25.1.1 DEBUG_PRINT

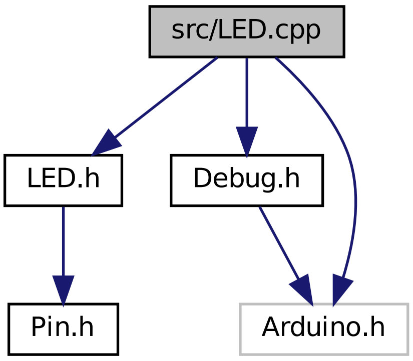
```
#define DEBUG_PRINT(
             message ) Debug::print(__FILE__, __LINE__, __func__, String(message))
```

5.26 src/LED.cpp File Reference

```
#include "LED.h"
#include "Debug.h"
```

```
#include <Arduino.h>
```

Include dependency graph for LED.cpp:



Macros

- #define DEBUG_PRINT(message) Debug::print(__FILE__, __LINE__, __func__, String(message))

5.26.1 Macro Definition Documentation

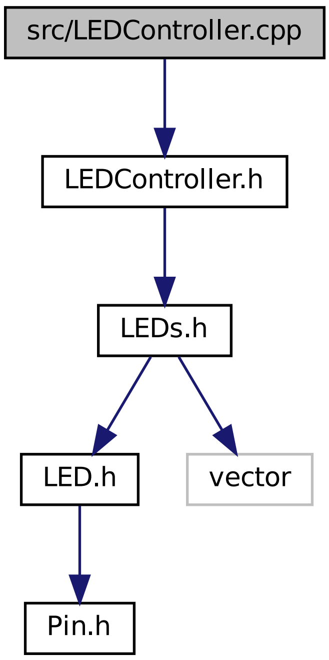
5.26.1.1 DEBUG_PRINT

```
#define DEBUG_PRINT(
            message ) Debug::print(__FILE__, __LINE__, __func__, String(message))
```

5.27 src/LEDController.cpp File Reference

```
#include "LEDController.h"
```

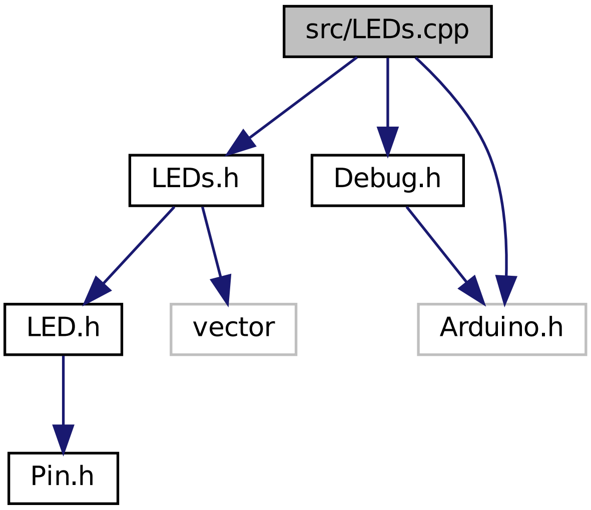
Include dependency graph for LEDController.cpp:



5.28 src/LEDs.cpp File Reference

```
#include "LEDs.h"
#include "Debug.h"
#include <Arduino.h>
```

Include dependency graph for LEDs.cpp:



Macros

- #define DEBUG_PRINT(message) Debug::print(__FILE__, __LINE__, __func__, String(message))

5.28.1 Macro Definition Documentation

5.28.1.1 DEBUG_PRINT

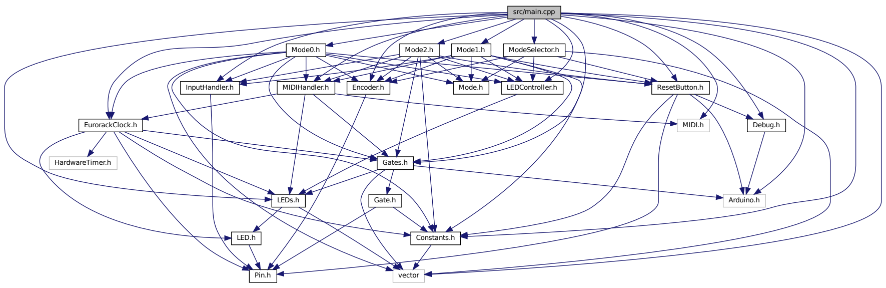
```
#define DEBUG_PRINT(
            message ) Debug::print(__FILE__, __LINE__, __func__, String(message))
```

5.29 src/main.cpp File Reference

```
#include <Arduino.h>
#include <MIDI.h>
#include "Gates.h"
#include "ModeSelector.h"
#include "LEDs.h"
#include "Debug.h"
#include "Encoder.h"
#include "MIDIHandler.h"
#include "EurorackClock.h"
```

```
#include "Constants.h"
#include "Mode0.h"
#include "Mode1.h"
#include "Mode2.h"
#include "LEDController.h"
#include "ResetButton.h"
#include "InputHandler.h"
#include <vector>
```

Include dependency graph for main.cpp:



Macros

- #define DEBUG_PRINT(message) Debug::print(__FILE__, __LINE__, __func__, String(message))
- #define RX_PIN PA3
- #define TX_PIN PA2
- #define ENCODER_PINA PB13
- #define ENCODER_PINB PB14
- #define ENCODER_BUTTON PB12
- #define CLOCK_PIN PB10
- #define RESET_PIN PB11
- #define RESET_BUTTON PB15
- #define TEMPO_LED PA8
- #define CV_A_PIN PA4
- #define CV_B_PIN PA5

Functions

- void setup ()
- void loop ()

Variables

- std::vector< int > pins = {PA15, PB3, PB4, PB5, PB6, PB7, PB8, PB9}
- const int numPins = pins.size()
- Gates gates = Gates(pins, numPins)
- std::vector< int > ledPins = {PA12, PA11, PB1, PB0, PA7, PA6, PA1, PA0}
- int numLedPins = ledPins.size()
- LEDs leds = LEDs(ledPins, numLedPins)
- int encCLKPin = ENCODER_PINA
- int encDTPin = ENCODER_PINB

- int encButtonPin = ENCODER_BUTTON
- bool inModeSelection = false
- int intensity = 255
- bool isInSelection = false
- unsigned long lastFlashTime = 0
- unsigned char internalPPQN = 24
- std::vector$<$ int $>$ musicalIntervals = {1, 2, 3, 4, 6, 8, 12, 16, 24, 32, 48, 64, 72, 96, 128, 144, 192, 288, 384, 576, 768, 1152, 1536}
- const int musicalIntervalsSize = musicalIntervals.size()
- int total_pages = 16 / leds.numLeds
- int min_intensity = 64
- int intensity_step = (255 - min_intensity) / (total_pages - 1)
- Encoder encoder = Encoder(encCLKPin, encDTPin, encButtonPin)
- ResetButton resetButton = ResetButton(RESET_BUTTON)
- LEDController ledController (leds)
- EurorackClock clock (CLOCK_PIN, RESET_PIN, TEMPO_LED, gates, leds)
- MIDIHandler midiHandler (Serial2, clock, gates, leds)
- InputHandler inputHandler = InputHandler(CV_A_PIN, CV_B_PIN)
- ModeSelector & modeSelector = ModeSelector::getInstance()
- Mode $*$ currentMode = nullptr
- Mode0 mode0 (encoder, inputHandler, gates, ledController, midiHandler, resetButton, clock)
- Mode1 mode1 (encoder, inputHandler, gates, ledController, midiHandler, resetButton)
- Mode2 mode2 (encoder, inputHandler, gates, ledController, midiHandler, resetButton)

5.29.1 Macro Definition Documentation

5.29.1.1 CLOCK_PIN

```
#define CLOCK_PIN PB10
```

5.29.1.2 CV_A_PIN

```
#define CV_A_PIN PA4
```

5.29.1.3 CV_B_PIN

```
#define CV_B_PIN PA5
```

5.29.1.4 DEBUG_PRINT

```
#define DEBUG_PRINT(
              message ) Debug::print(__FILE__, __LINE__, __func__, String(message))
```

5.29.1.5 ENCODER_BUTTON

```
#define ENCODER_BUTTON PB12
```

5.29.1.6 ENCODER_PINA

```
#define ENCODER_PINA PB13
```

5.29.1.7 ENCODER_PINB

```
#define ENCODER_PINB PB14
```

5.29.1.8 RESET_BUTTON

```
#define RESET_BUTTON PB15
```

5.29.1.9 RESET_PIN

```
#define RESET_PIN PB11
```

5.29.1.10 RX_PIN

```
#define RX_PIN PA3
```

5.29.1.11 TEMPO_LED

```
#define TEMPO_LED PA8
```

5.29.1.12 TX_PIN

```
#define TX_PIN PA2
```

5.29.2 Function Documentation

5.29.2.1 loop()

```
void loop ( )
```

5.29.2.2 setup()

```
void setup ( )
```

5.29.3 Variable Documentation

5.29.3.1 clock

```
EurorackClock clock(CLOCK_PIN, RESET_PIN, TEMPO_LED, gates, leds) (
            CLOCK_PIN ,
            RESET_PIN ,
            TEMPO_LED ,
            gates ,
            leds )
```

5.29.3.2 currentMode

```
Mode* currentMode = nullptr
```

5.29.3.3 encButtonPin

```
int encButtonPin = ENCODER_BUTTON
```

5.29.3.4 encCLKPin

```
int encCLKPin = ENCODER_PINA
```

5.29.3.5 encDTPin

```
int encDTPin = ENCODER_PINB
```

5.29.3.6 encoder

```
Encoder encoder = Encoder(encCLKPin, encDTPin, encButtonPin)
```

5.29.3.7 gates

```
Gates gates = Gates(pins, numPins)
```

5.29.3.8 inModeSelection

```
bool inModeSelection = false
```

5.29.3.9 inputHandler

```
InputHandler inputHandler = InputHandler(CV_A_PIN, CV_B_PIN)
```

5.29.3.10 intensity

```
int intensity = 255
```

5.29.3.11 intensity_step

```
int intensity_step = (255 - min_intensity) / (total_pages - 1)
```

5.29.3.12 internalPPQN

```
unsigned char internalPPQN = 24
```

5.29.3.13 isInSelection

```
bool isInSelection = false
```

5.29.3.14 lastFlashTime

```
unsigned long lastFlashTime = 0
```

5.29.3.15 ledController

```
LEDController ledController(leds) (
             leds   )
```

5.29.3.16 ledPins

```
std::vector<int> ledPins = {PA12, PA11, PB1, PB0, PA7, PA6, PA1, PA0}
```

5.29.3.17 leds

```
LEDs leds = LEDs(ledPins, numLedPins)
```

5.29.3.18 midiHandler

```
MIDIHandler midiHandler(Serial2, clock, gates, leds) (
             Serial2 ,
             clock ,
             gates ,
             leds  )
```

5.29.3.19 min_intensity

```
int min_intensity = 64
```

5.29.3.20 mode0

```
Mode0 mode0(encoder, inputHandler, gates, ledController, midiHandler, resetButton, clock) (
             encoder ,
             inputHandler ,
             gates ,
             ledController ,
             midiHandler ,
             resetButton ,
             clock  )
```

5.29.3.21 mode1

```
Mode1 mode1(encoder, inputHandler, gates, ledController, midiHandler, resetButton) (
             encoder ,
             inputHandler ,
             gates ,
             ledController ,
             midiHandler ,
             resetButton  )
```

5.29.3.22 mode2

```
Mode2 mode2(encoder, inputHandler, gates, ledController, midiHandler, resetButton) (
             encoder ,
             inputHandler ,
             gates ,
             ledController ,
             midiHandler ,
             resetButton  )
```

5.29.3.23 modeSelector

```
ModeSelector& modeSelector = ModeSelector::getInstance()
```

5.29.3.24 musicalIntervals

```
std::vector<int> musicalIntervals = {1, 2, 3, 4, 6, 8, 12, 16, 24, 32, 48, 64, 72, 96, 128,
144, 192, 288, 384, 576, 768, 1152, 1536}
```

5.29.3.25 musicalIntervalsSize

```
const int musicalIntervalsSize = musicalIntervals.size()
```

5.29.3.26 numLedPins

```
int numLedPins = ledPins.size()
```

5.29.3.27 numPins

```
const int numPins = pins.size()
```

5.29.3.28 pins

```
std::vector<int> pins = {PA15, PB3, PB4, PB5, PB6, PB7, PB8, PB9}
```

5.29.3.29 resetButton

```
ResetButton resetButton = ResetButton(RESET_BUTTON)
```

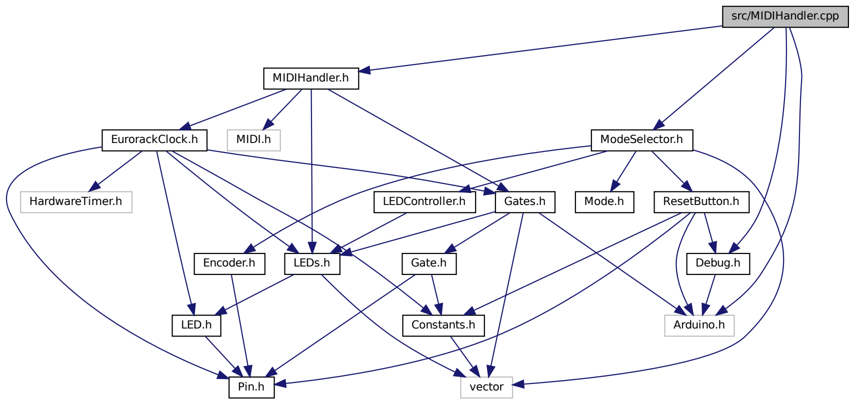
5.29.3.30 total_pages

```
int total_pages = 16 / leds.numLeds
```

5.30 src/MIDIHandler.cpp File Reference

```
#include "MIDIHandler.h"
#include "Debug.h"
#include <Arduino.h>
#include "ModeSelector.h"
```

Include dependency graph for MIDIHandler.cpp:



Macros

- #define DEBUG_PRINT(message)

Variables

- bool isInSelection

5.30.1 Macro Definition Documentation

5.30.1.1 DEBUG_PRINT

```
#define DEBUG_PRINT(
                message )
```

Value:

```
    { \
    Debug::print(__FILE__, __LINE__, __func__, String(message)); \
    Serial.flush(); \
}
```

5.30.2 Variable Documentation

5.30.2.1 isInSelection

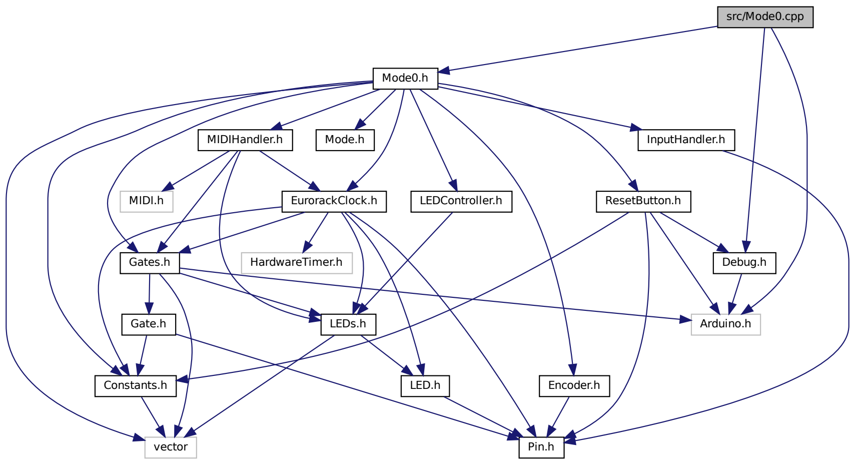
```
bool isInSelection  [extern]
```

5.31 src/Mode.cpp File Reference

5.32 src/Mode0.cpp File Reference

```
#include "Mode0.h"
#include "Debug.h"
#include <Arduino.h>
```

Include dependency graph for Mode0.cpp:



Macros

- #define DEBUG_PRINT(message) Debug::print(__FILE__, __LINE__, __func__, String(message))

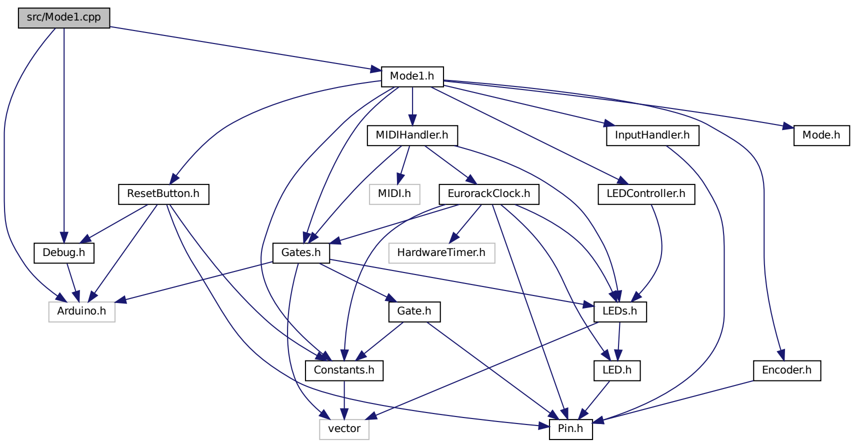
5.32.1 Macro Definition Documentation

5.32.1.1 DEBUG_PRINT

```
#define DEBUG_PRINT(
            message ) Debug::print(__FILE__, __LINE__, __func__, String(message))
```

5.33 src/Mode1.cpp File Reference

```
#include "Mode1.h"
#include "Debug.h"
#include <Arduino.h>
```
Include dependency graph for Mode1.cpp:



Macros

- #define DEBUG_PRINT(message) Debug::print(__FILE__, __LINE__, __func__, String(message))

5.33.1 Macro Definition Documentation

5.33.1.1 DEBUG_PRINT

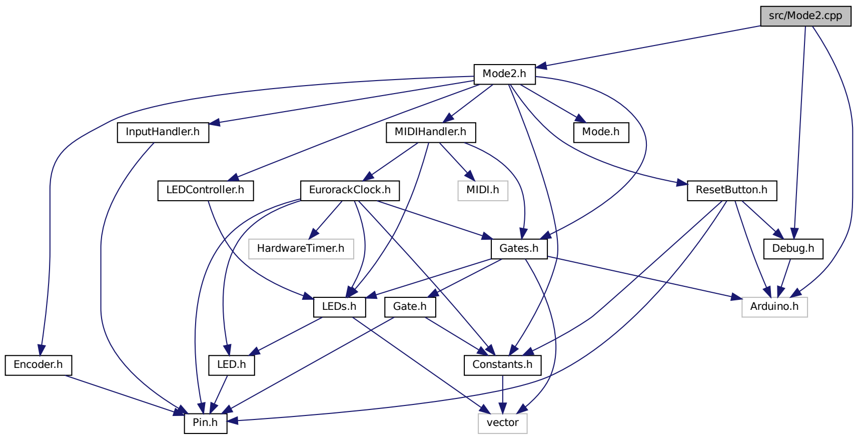
```
#define DEBUG_PRINT(
            message ) Debug::print(__FILE__, __LINE__, __func__, String(message))
```

5.34 src/Mode2.cpp File Reference

```
#include "Mode2.h"
#include "Debug.h"
```

```
#include <Arduino.h>
```

Include dependency graph for Mode2.cpp:



Macros

- #define DEBUG_PRINT(message)

5.34.1 Macro Definition Documentation

5.34.1.1 DEBUG_PRINT

```
#define DEBUG_PRINT(
                message )
```

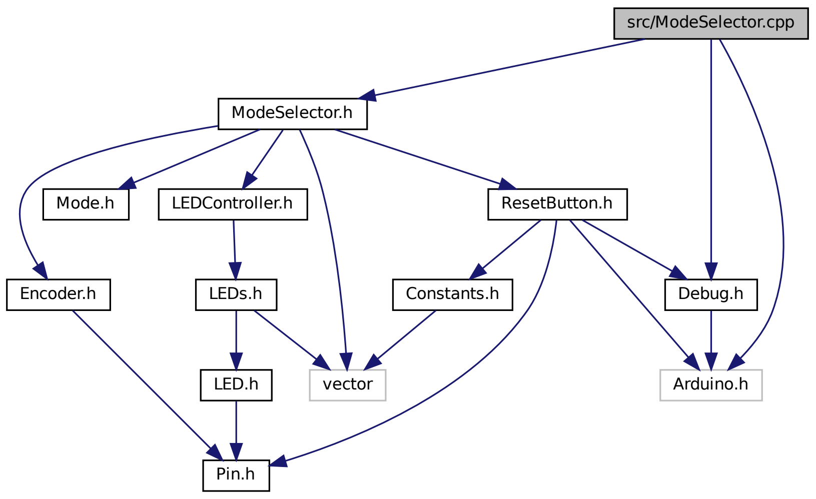
Value:
```
    { \
    Debug::print(__FILE__, __LINE__, __func__, String(message)); \
    Serial.flush(); \
}
```

5.35 src/ModeSelector.cpp File Reference

```
#include "ModeSelector.h"
#include <Arduino.h>
```

```
#include "Debug.h"
```

Include dependency graph for ModeSelector.cpp:



Macros

- #define DEBUG_PRINT(message) Debug::print(__FILE__, __LINE__, __func__, String(message))

5.35.1 Macro Definition Documentation

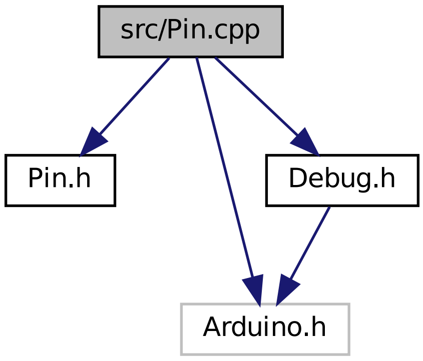
5.35.1.1 DEBUG_PRINT

```
#define DEBUG_PRINT(
            message ) Debug::print(__FILE__, __LINE__, __func__, String(message))
```

5.36 src/Pin.cpp File Reference

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

```
#include "Debug.h"
```
Include dependency graph for Pin.cpp:



Macros

- #define DEBUG_PRINT(message) Debug::print(__FILE__, __LINE__, __func__, String(message))

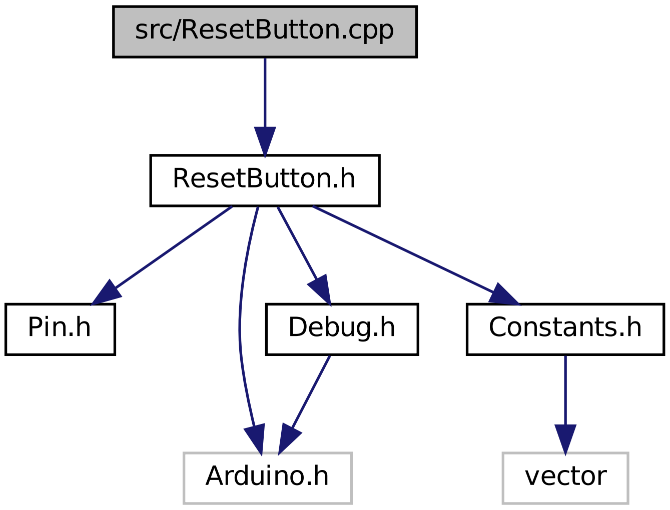
5.36.1 Macro Definition Documentation

5.36.1.1 DEBUG_PRINT

```
#define DEBUG_PRINT(
              message ) Debug::print(__FILE__, __LINE__, __func__, String(message))
```

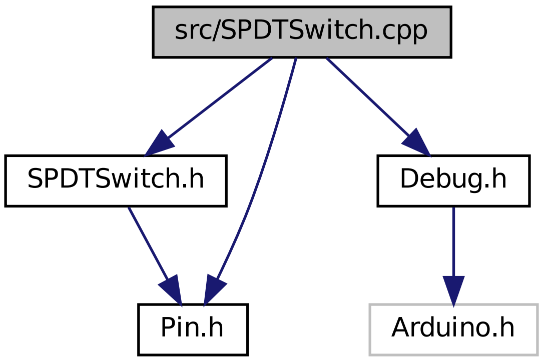
5.37 src/ResetButton.cpp File Reference

```
#include "ResetButton.h"
```
Include dependency graph for ResetButton.cpp:



5.38 src/SPDTSwitch.cpp File Reference

```
#include "SPDTSwitch.h"
#include "Pin.h"
#include "Debug.h"
```
Include dependency graph for SPDTSwitch.cpp:


src/SPDTSwitch.cpp
SPDTSwitch.h
Debug.h
Pin.h
Arduino.h