G8R

Generated by Doxygen 1.9.1

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

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

Chapter 2

Class Index

2.1 Class List

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

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

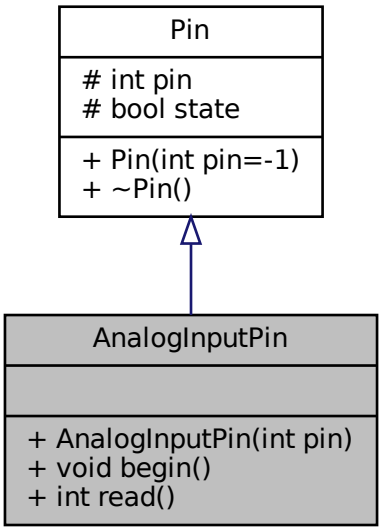
Chapter 4

Class Documentation

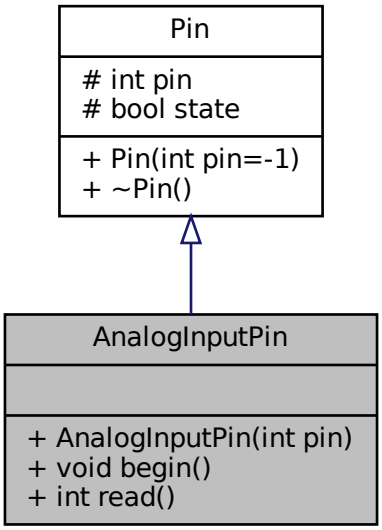
4.1 AnalogInputPin Class Reference

`#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 ( )
```

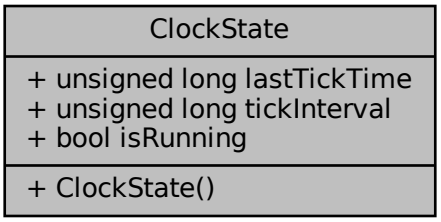
The documentation for this class was generated from the following files:

- 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
```

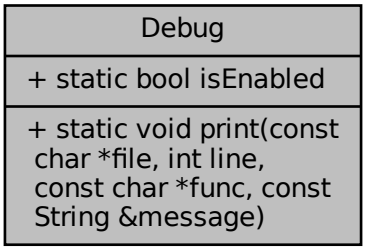
The documentation for this struct was generated from the following file:

- include/EurorackClock.h

4.3 Debug Class Reference

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

Collaboration diagram for Debug:



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]
```

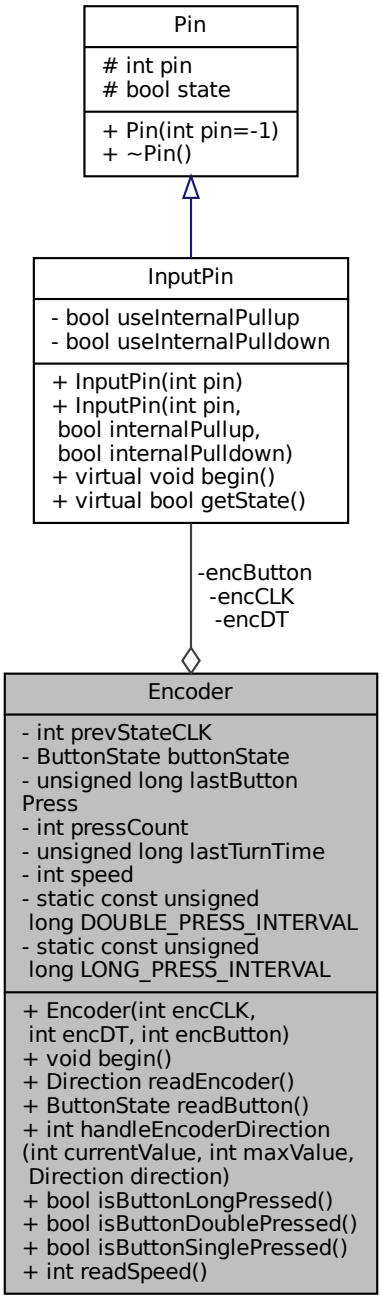
The documentation for this class was generated from the following files:

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

4.4 Encoder Class Reference

`#include <Encoder.h>`

Collaboration diagram for Encoder:



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]
```

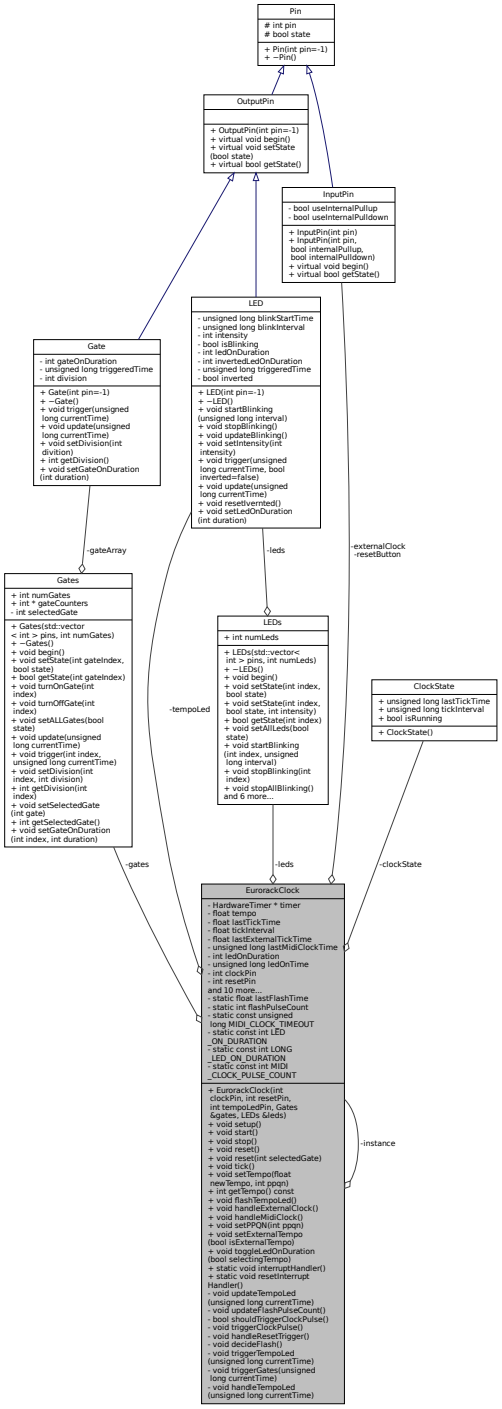
The documentation for this class was generated from the following files:

- 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]
```

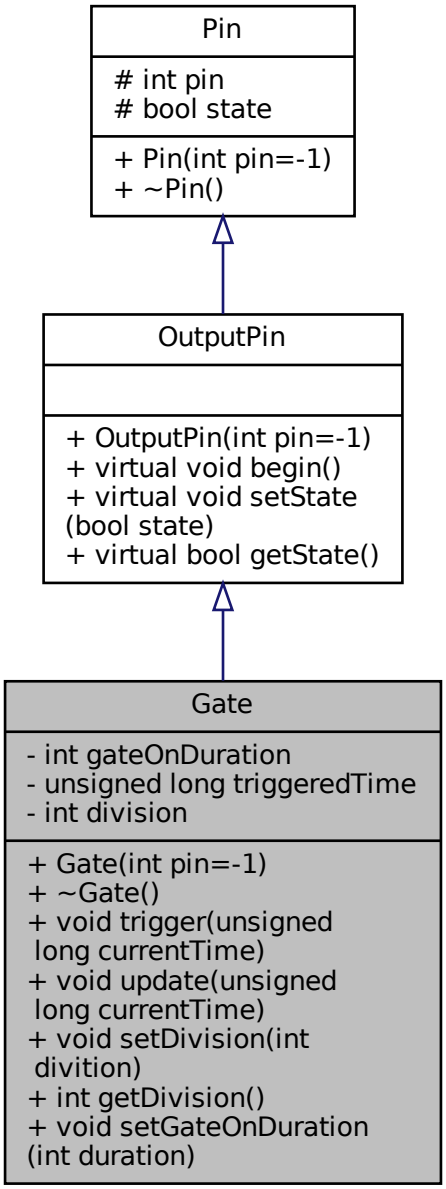
The documentation for this class was generated from the following files:

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

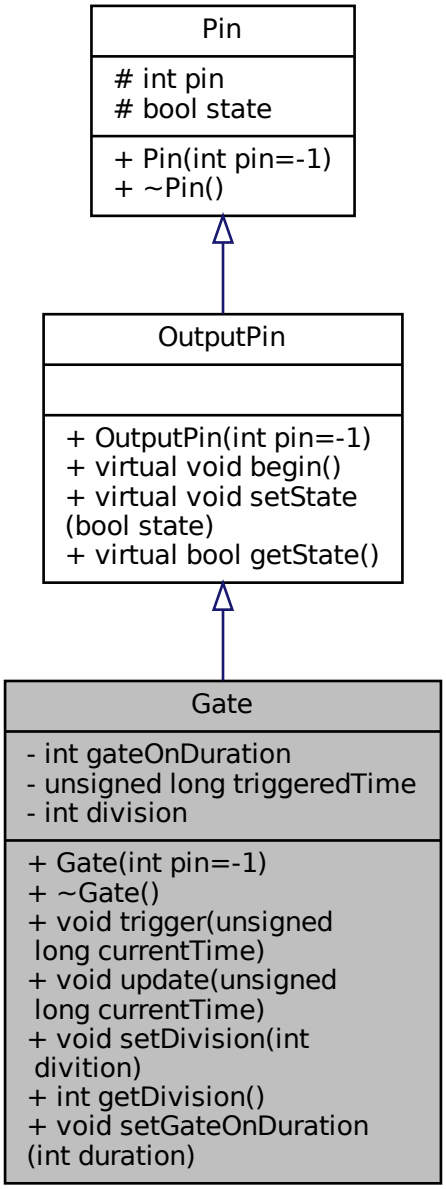
4.6 Gate Class Reference

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

Inheritance diagram for Gate:



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]
```

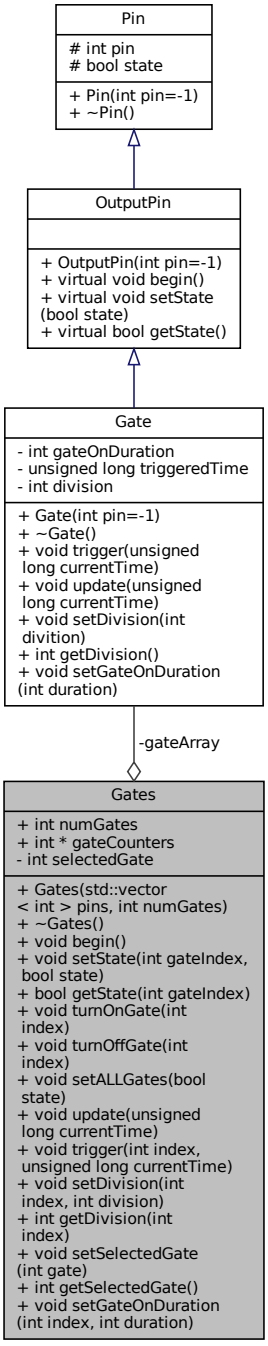
The documentation for this class was generated from the following files:

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

4.7 Gates Class Reference

`#include <Gates.h>`

Collaboration diagram for Gates:



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]
```

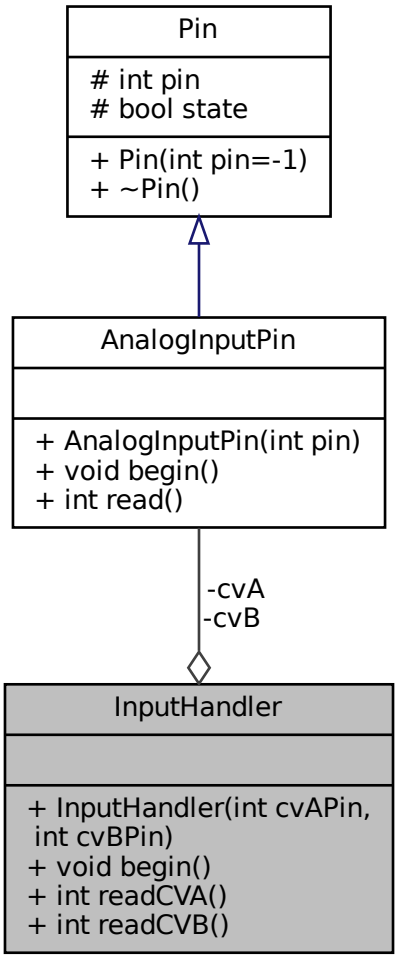
The documentation for this class was generated from the following files:

- 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]
```

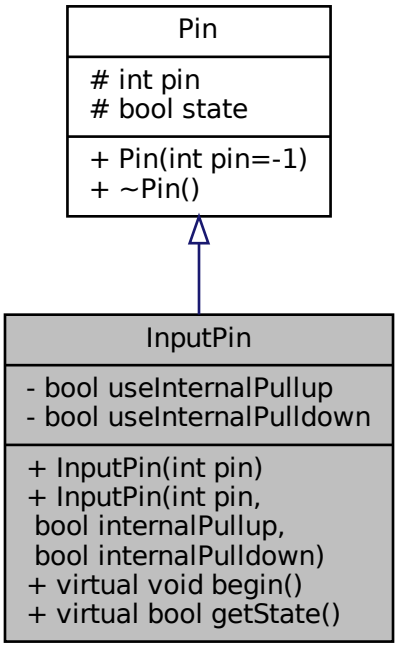
The documentation for this class was generated from the following files:

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

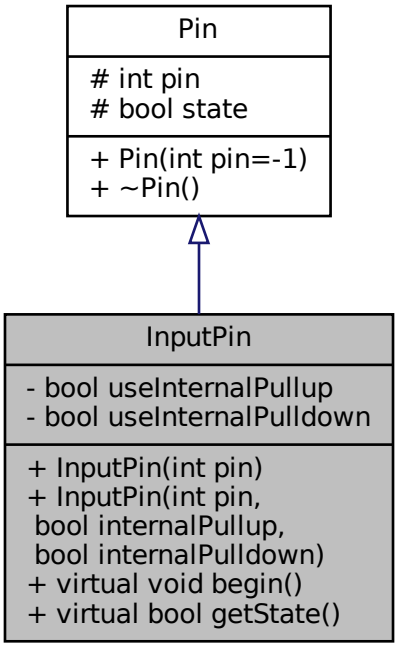
4.9 InputPin Class Reference

`#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]
```

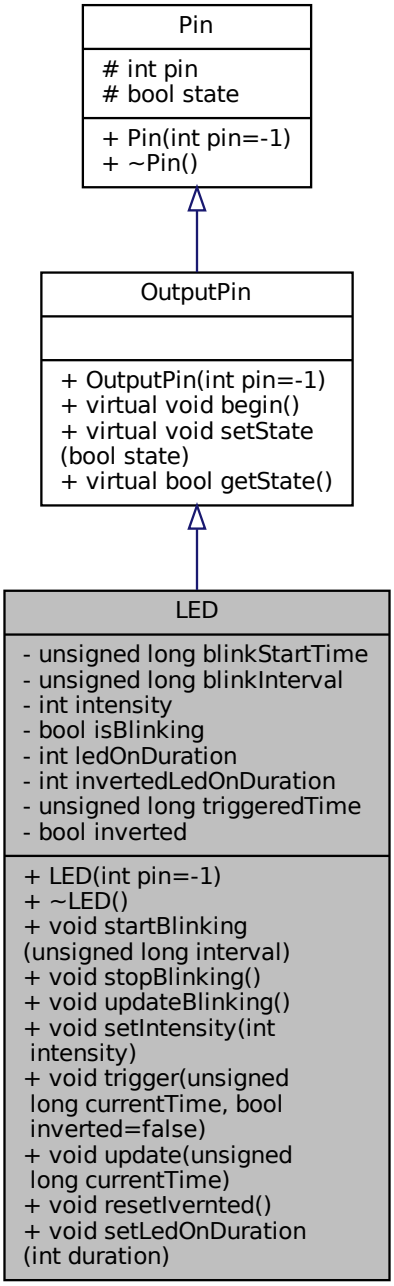
The documentation for this class was generated from the following files:

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

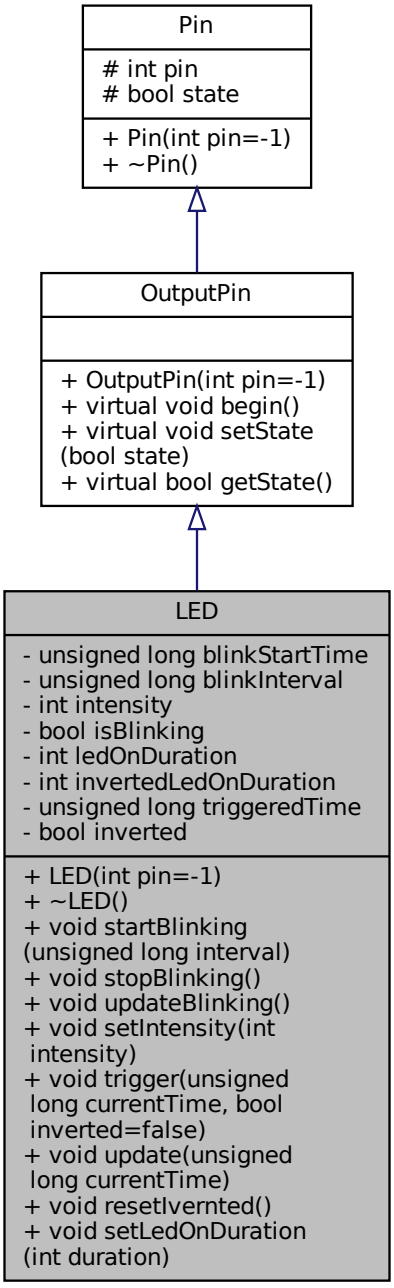
4.10 LED Class Reference

`#include <LED.h>`

Inheritance diagram for LED:



Collaboration diagram for LED:



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]
```

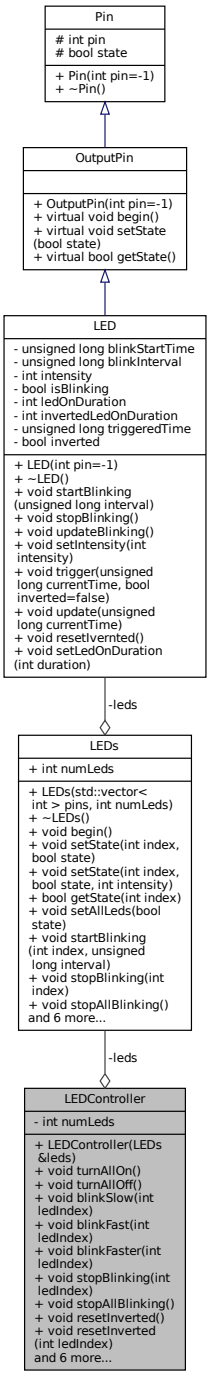
The documentation for this class was generated from the following files:

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

4.11 LEDController Class Reference

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

Collaboration diagram for LEDController:



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]
```

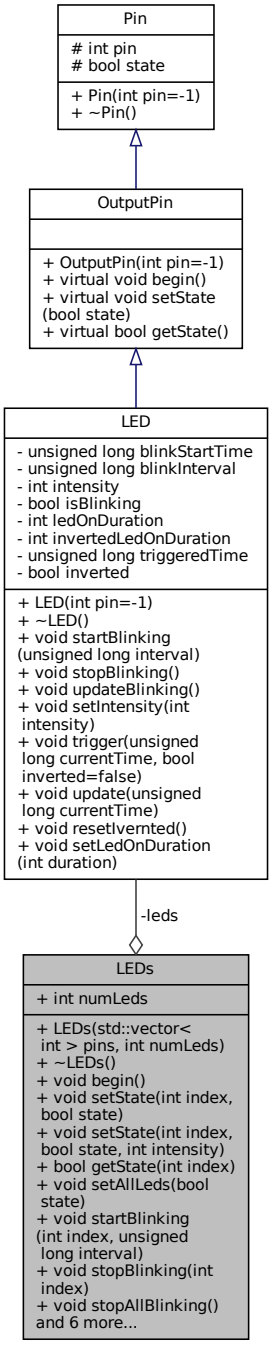
The documentation for this class was generated from the following files:

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

4.12 LEDs Class Reference

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

Collaboration diagram for LEDs:



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
```

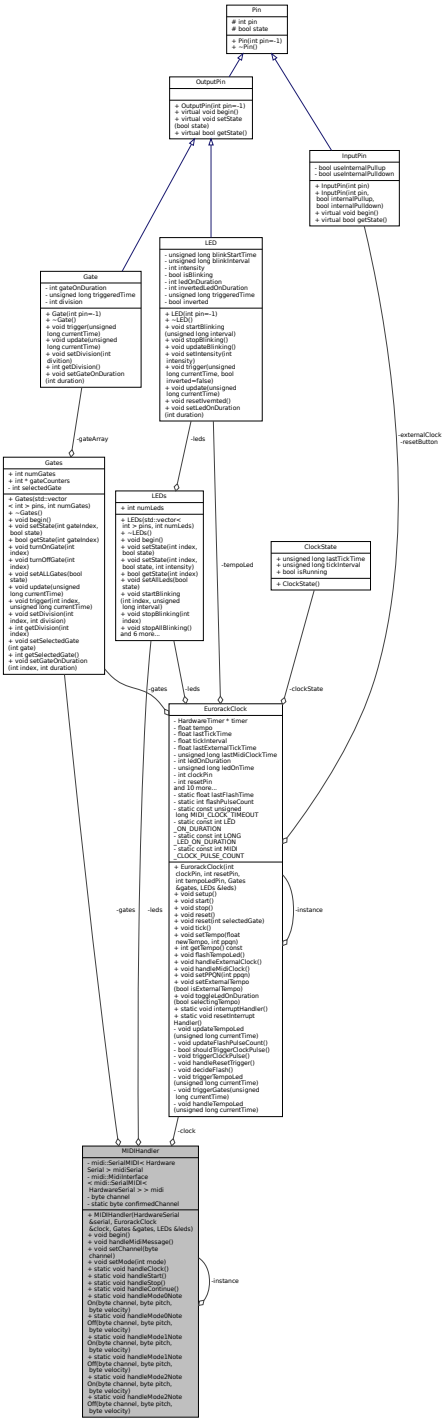
The documentation for this class was generated from the following files:

- 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]
```

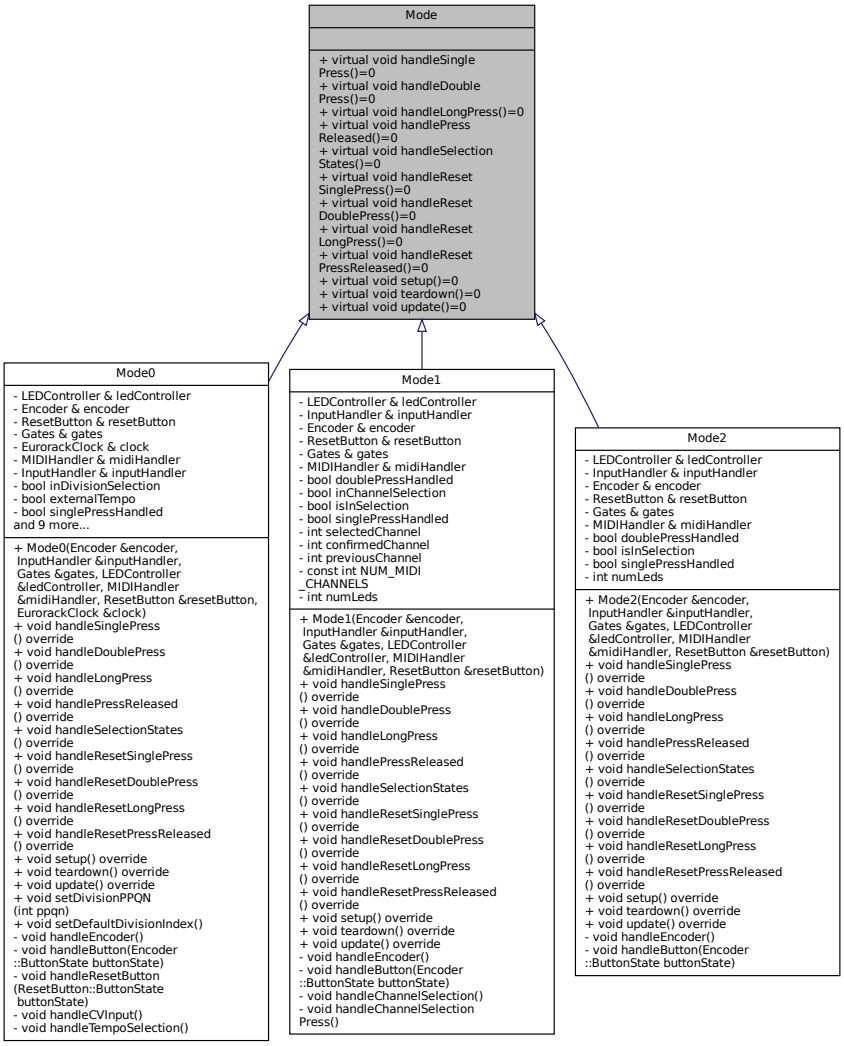
The documentation for this class was generated from the following files:

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

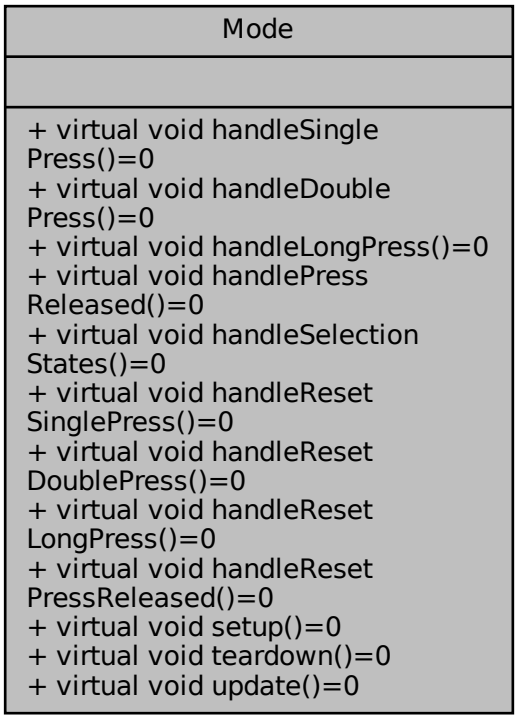
4.14 Mode Class Reference

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

Inheritance diagram for Mode:



Collaboration diagram for Mode:



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.

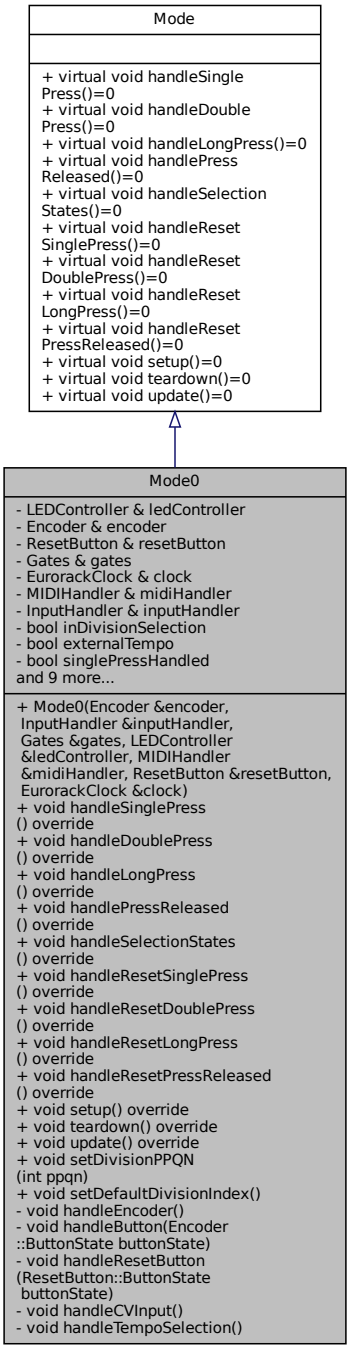
The documentation for this class was generated from the following file:

- include/Mode.h

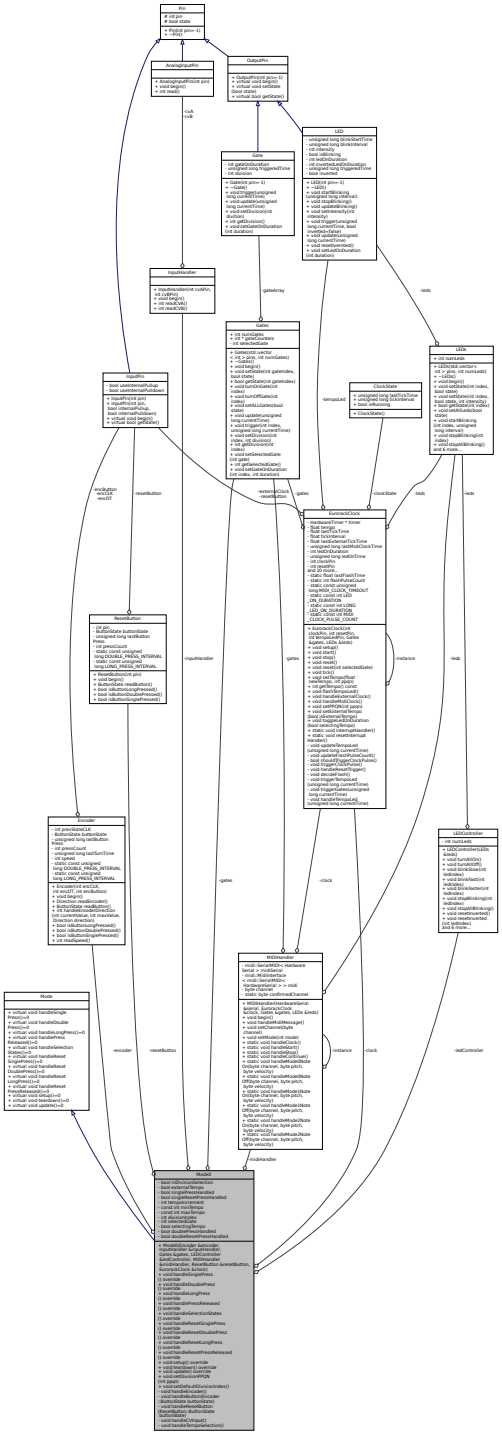
4.15 Mode0 Class Reference

`#include <Mode0.h>`

Inheritance diagram for Mode0:



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]
```

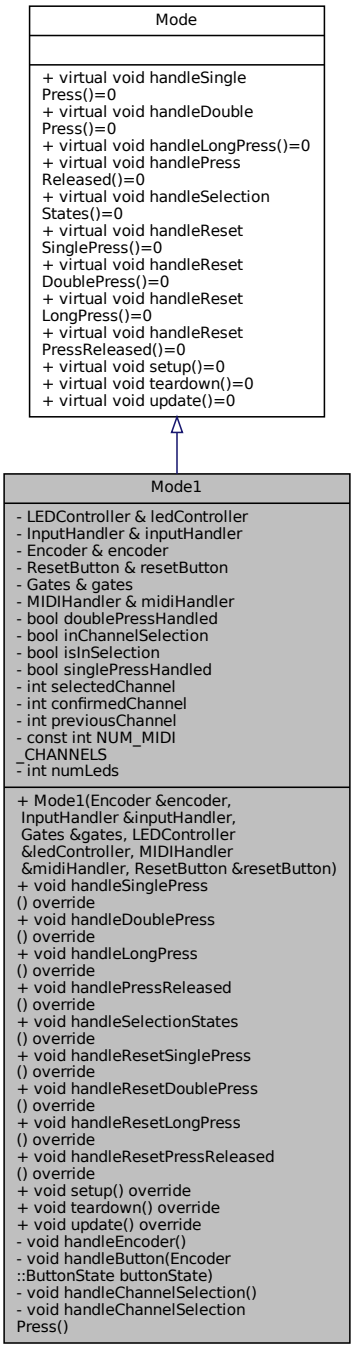
The documentation for this class was generated from the following files:

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

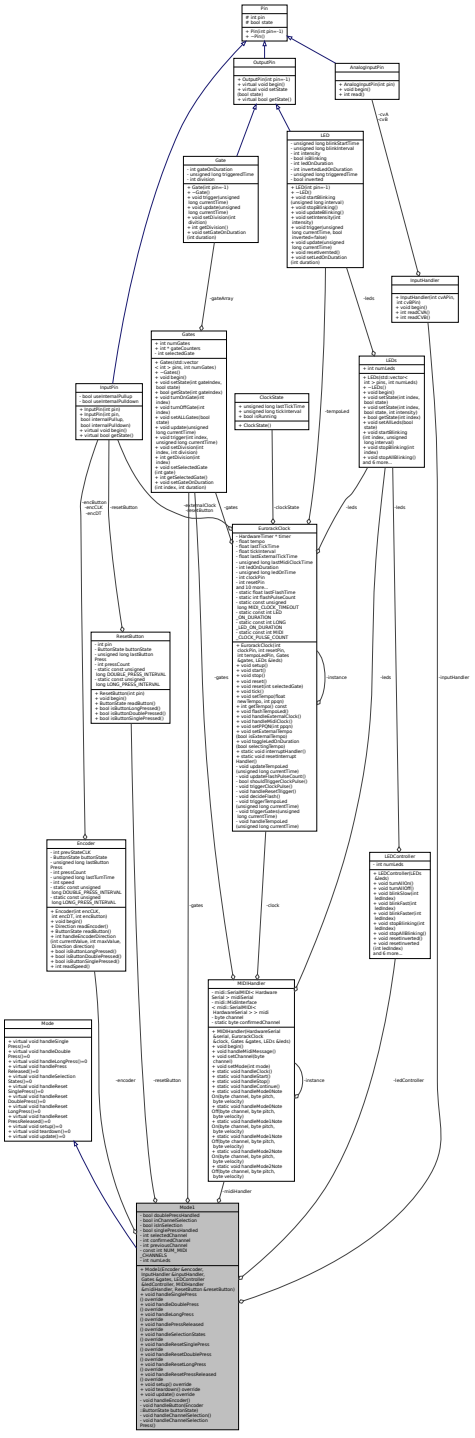
4.16 Mode1 Class Reference

`#include <Mode1.h>`

Inheritance diagram for Mode1:



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]
```

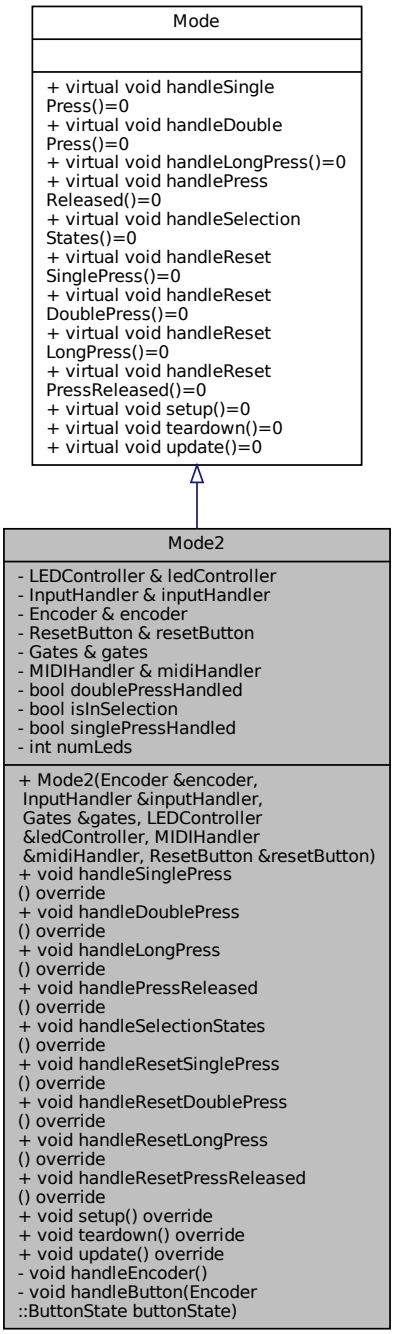
The documentation for this class was generated from the following files:

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

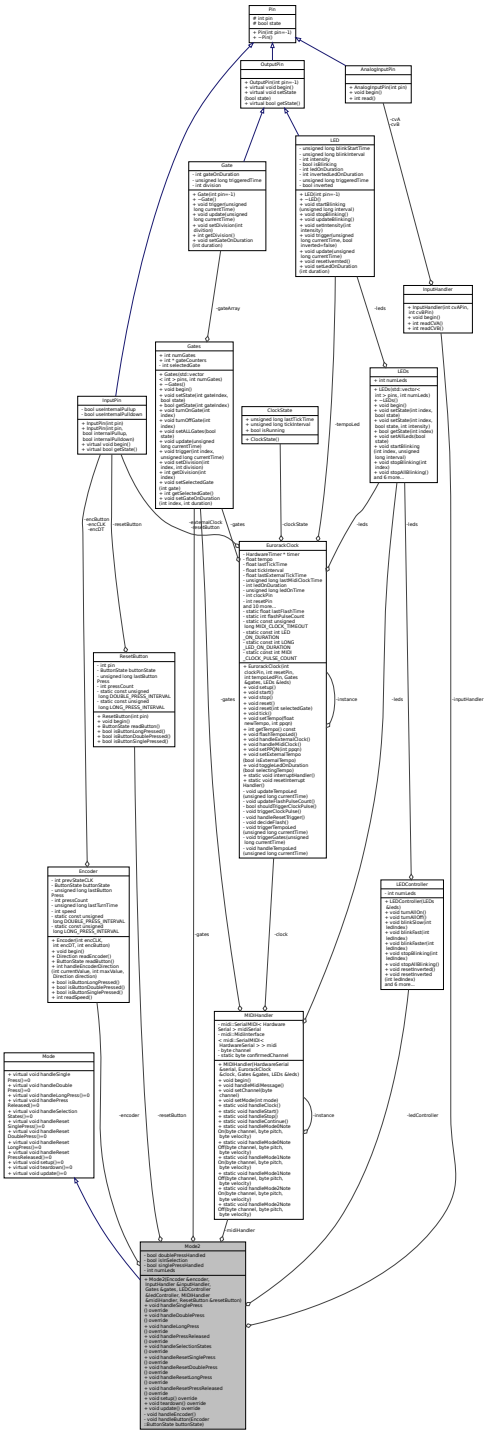
4.17 Mode2 Class Reference

`#include <Mode2.h>`

Inheritance diagram for Mode2:



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]
```

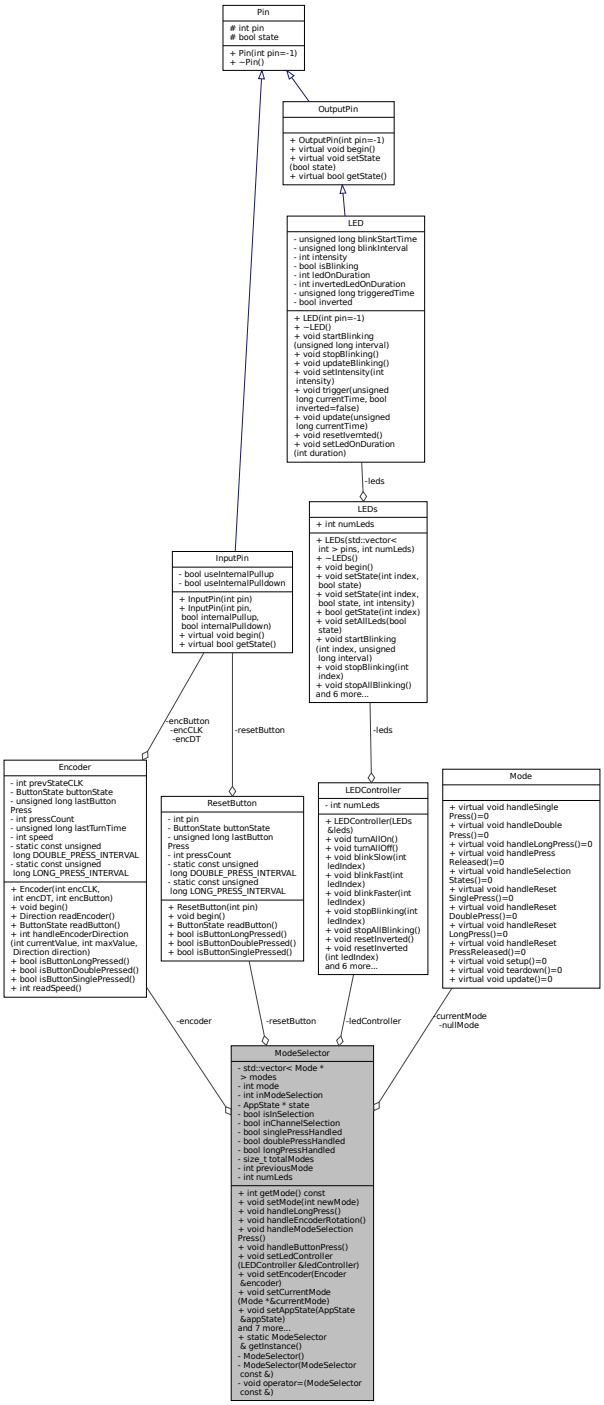
The documentation for this class was generated from the following files:

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

4.18 ModeSelector Class Reference

`#include <ModeSelector.h>`

Collaboration diagram for ModeSelector:



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)
- void setAppState (AppState &appState)
- void saveAppState ()
- void readAppState ()
- void initializeEEPROM ()
- 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
- AppState ∗ state
- 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 initializeEEPROM()

```
void ModeSelector::initializeEEPROM ( )
```

4.18.2.10 isInModeSelection()

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

4.18.2.11 operator=()

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

4.18.2.12 readAppState()

```
void ModeSelector::readAppState ( )
```

4.18.2.13 saveAppState()

```
void ModeSelector::saveAppState ( )
```

4.18.2.14 setAppState()

```
void ModeSelector::setAppState (
            AppState & appState )
```

4.18.2.15 setCurrentMode()

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

4.18.2.16 setEncoder()

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

4.18.2.17 setLedController()

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

4.18.2.18 setMode()

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

4.18.2.19 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 state

```
AppState* ModeSelector::state  [private]
```

4.18.3.17 totalModes

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

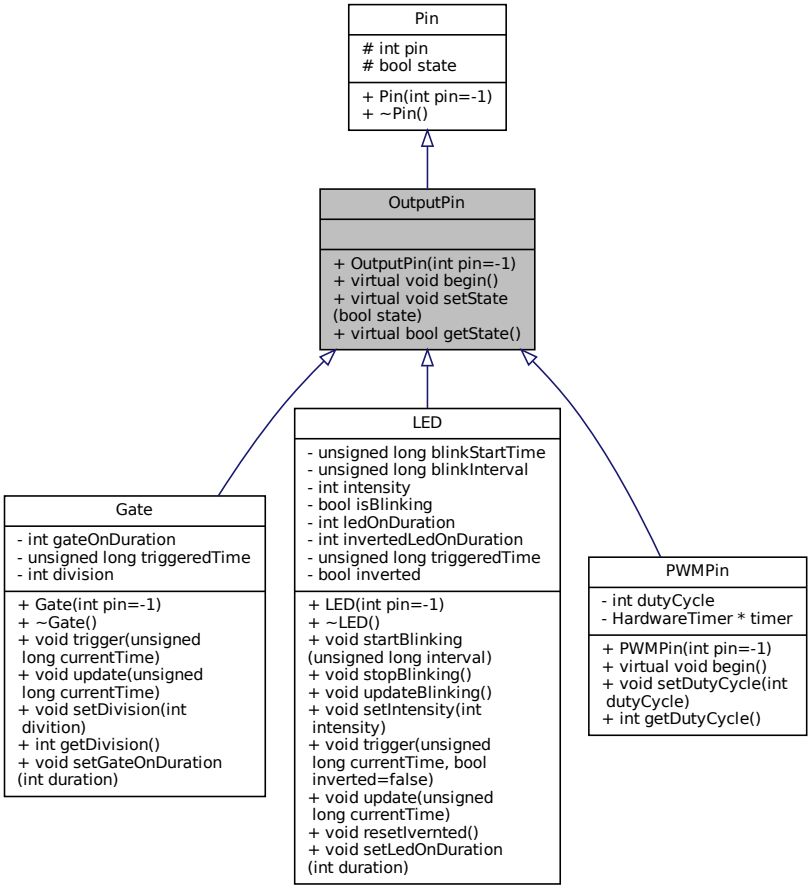
The documentation for this class was generated from the following files:

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

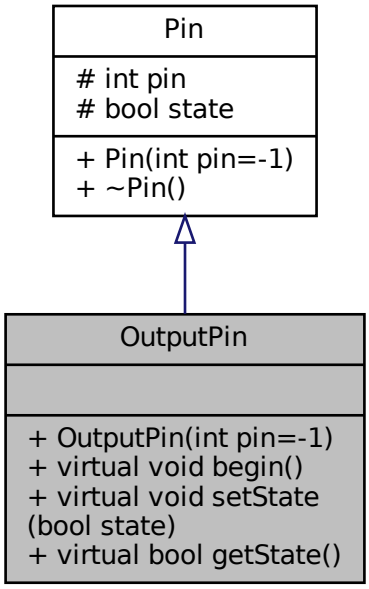
4.19 OutputPin Class Reference

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

Inheritance diagram for OutputPin:



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]
```

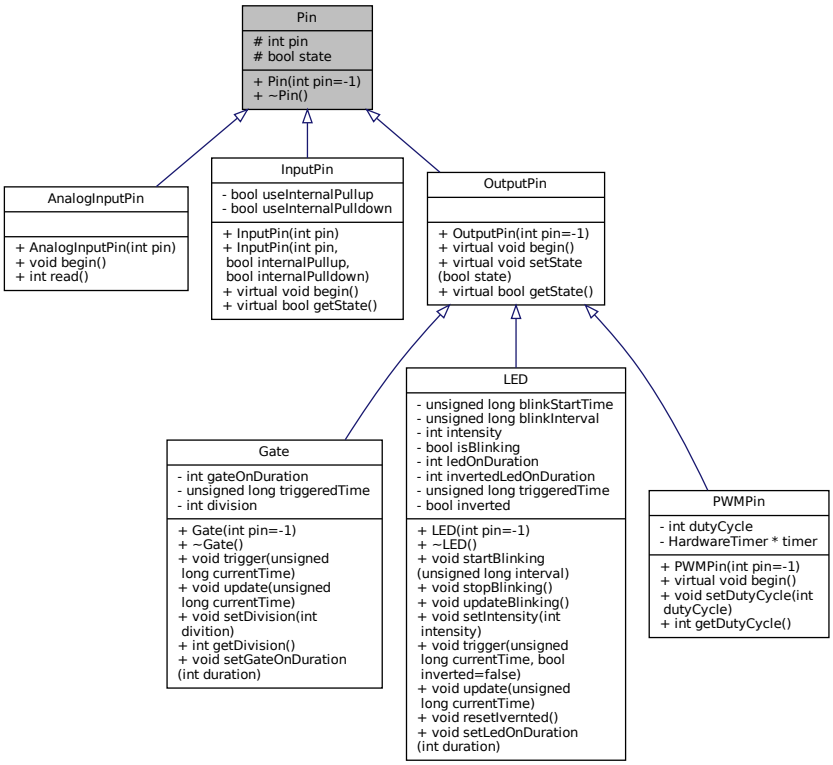
The documentation for this class was generated from the following files:

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

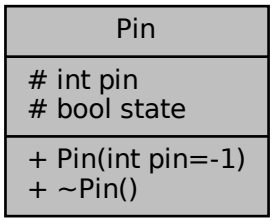
4.20 Pin Class Reference

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

Inheritance diagram for Pin:



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]
```

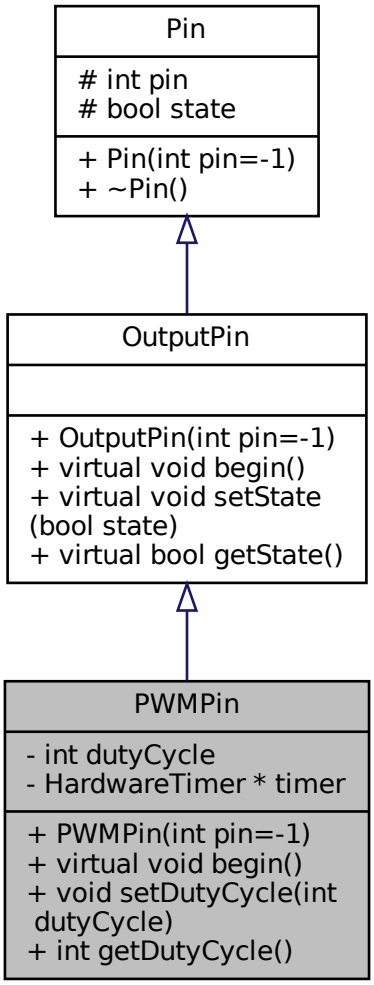
The documentation for this class was generated from the following files:

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

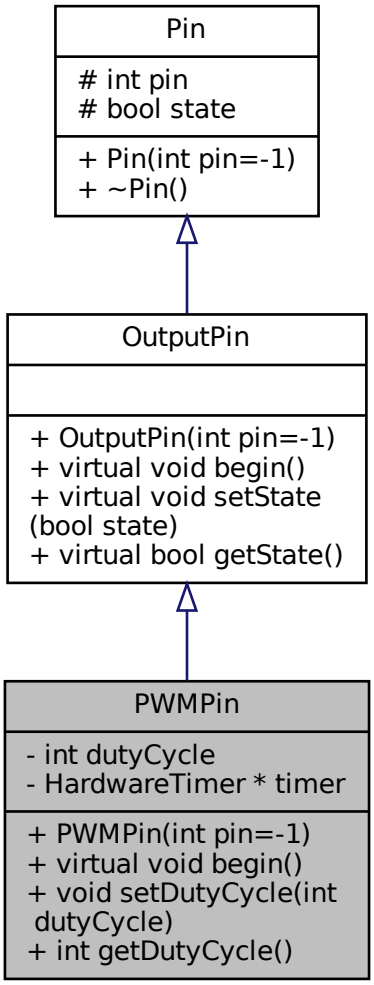
4.21 PWMPin Class Reference

```
#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]
```

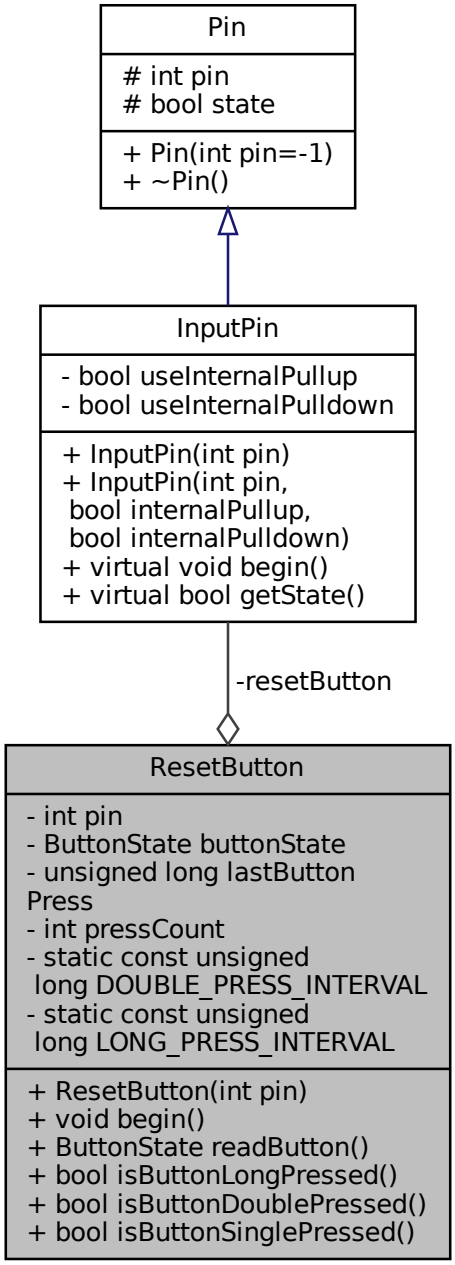
The documentation for this class was generated from the following files:

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

4.22 ResetButton Class Reference

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

Collaboration diagram for ResetButton:



Public Types

- enum ButtonState { OPEN , PRESSED }

Public Member Functions

- ResetButton (int pin)

- void 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()

```
void 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]
```

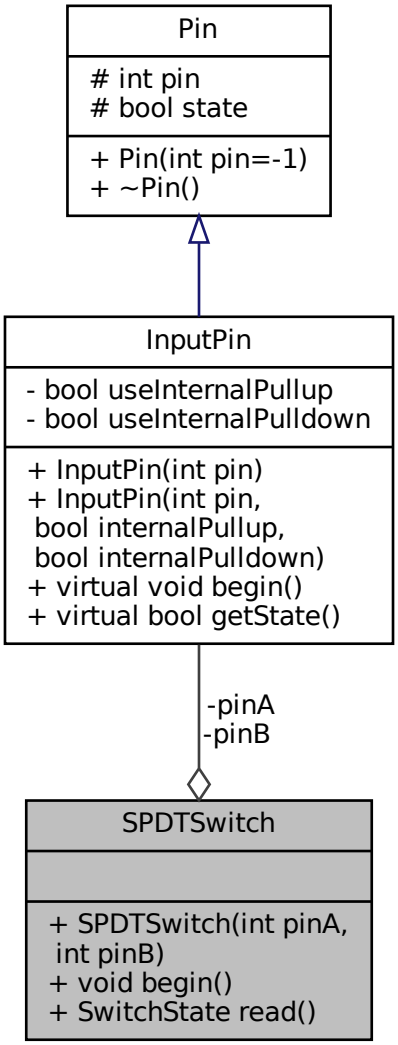
The documentation for this class was generated from the following files:

- 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:

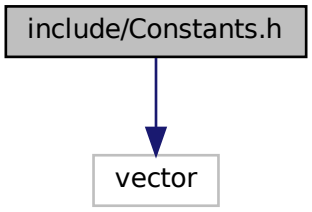
- include/SPDTSwitch.h
- src/SPDTSwitch.cpp

Chapter 5

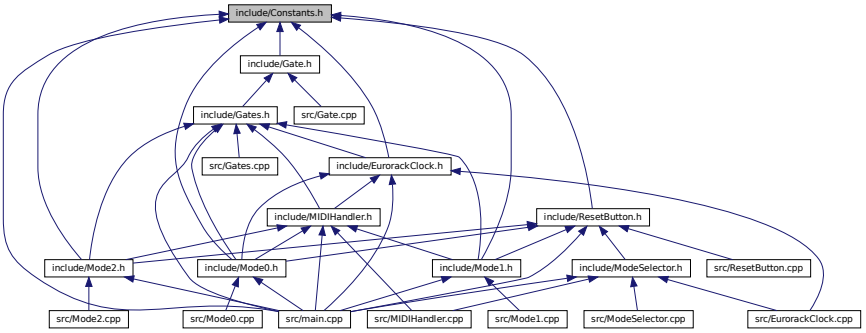
File Documentation

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

  Size of musical intervals array.
- unsigned char internalPPQN

  Pulses per quarter note.

5.1.1 Variable Documentation

5.1.1.1 internalPPQN

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

Pulses per quarter note.

5.1.1.2 musicalIntervals

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

5.1.1.3 musicalIntervalsSize

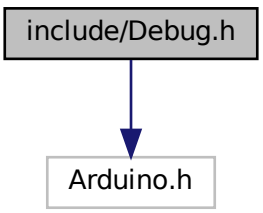
```
const int musicalIntervalsSize  [extern]
```

Size of musical intervals array.

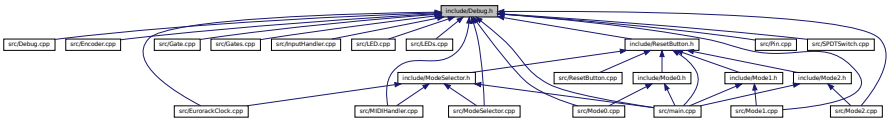
5.2 include/Debug.h File Reference

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

Include dependency graph for Debug.h:



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

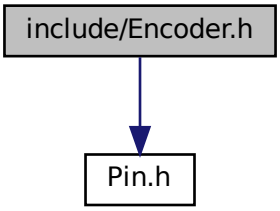


Classes

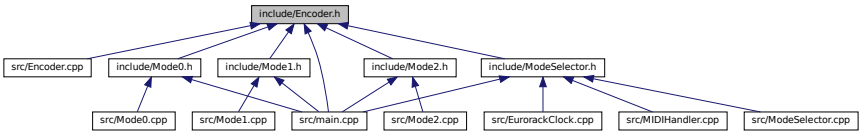
- class Debug

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:



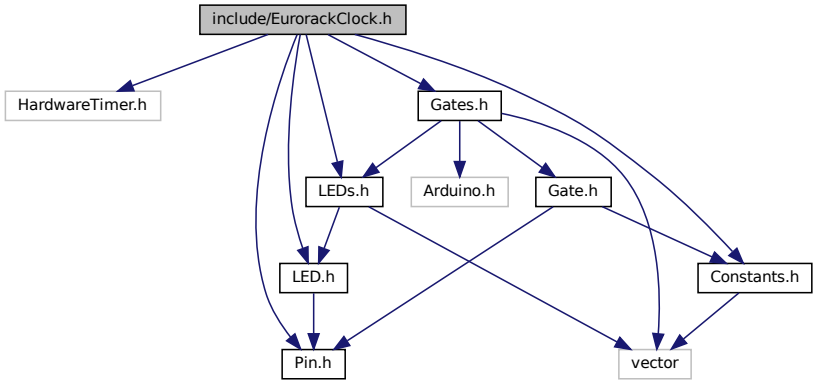
Classes

- class Encoder

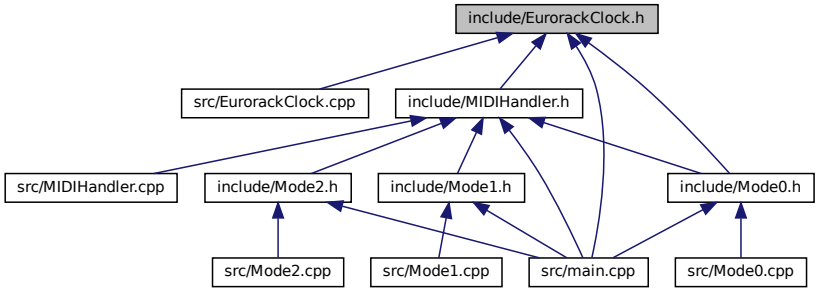
5.4 include/EurorackClock.h File Reference

```
#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:



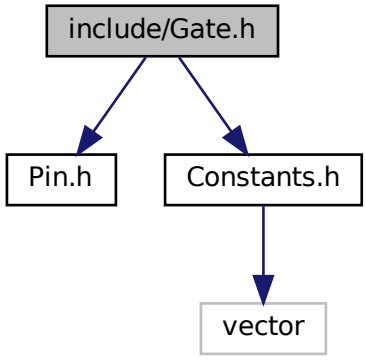
Classes

- struct ClockState
- class EurorackClock

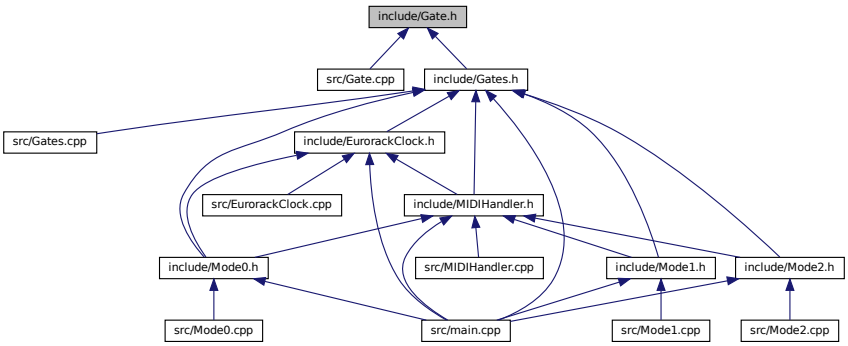
5.5 include/Gate.h File Reference

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

Include dependency graph for Gate.h:



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



Classes

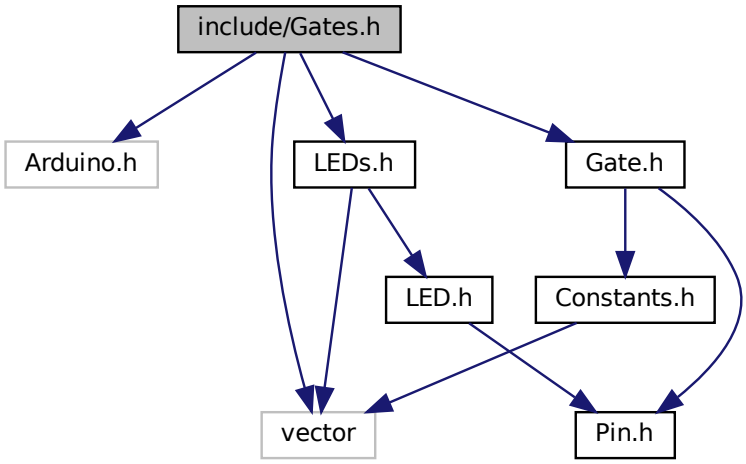
- class Gate

5.6 include/Gates.h File Reference

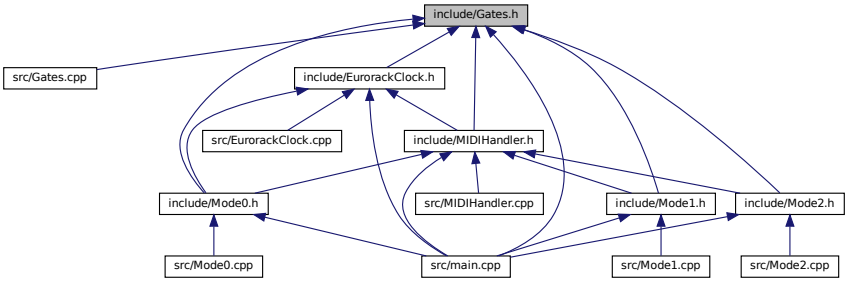
```
#include <Arduino.h>
#include "Gate.h"
#include "LEDs.h"
```

```
#include <vector>
```

Include dependency graph for Gates.h:



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



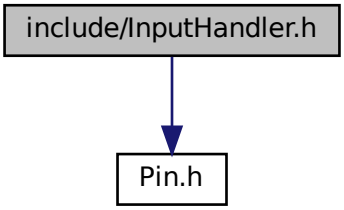
Classes

- class Gates

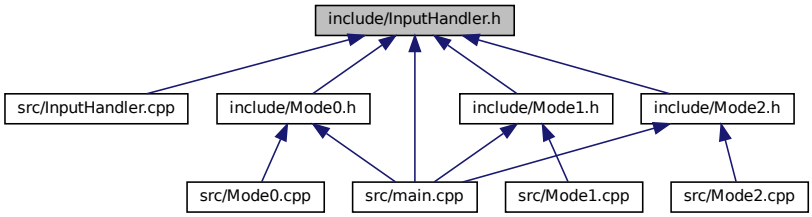
5.7 include/InputHandler.h File Reference

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

Include dependency graph for InputHandler.h:



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



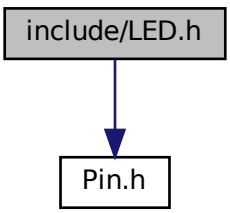
Classes

- class InputHandler

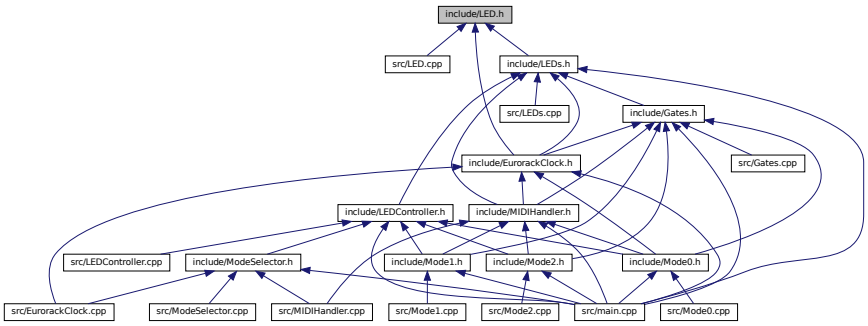
5.8 include/LED.h File Reference

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

Include dependency graph for LED.h:



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



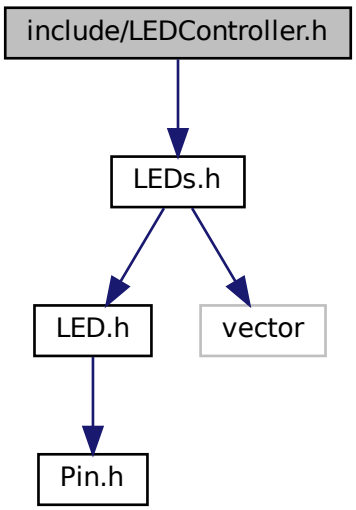
Classes

- class LED

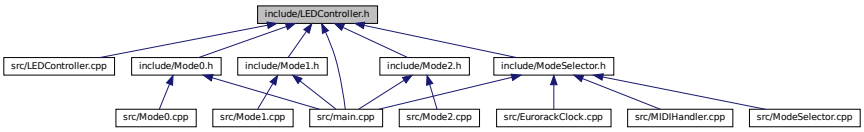
5.9 include/LEDController.h File Reference

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

Include dependency graph for LEDController.h:



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



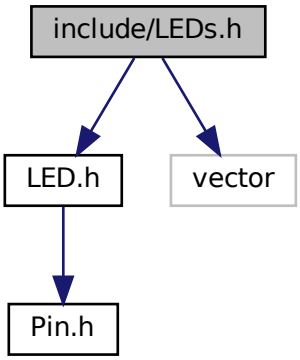
Classes

- class LEDController

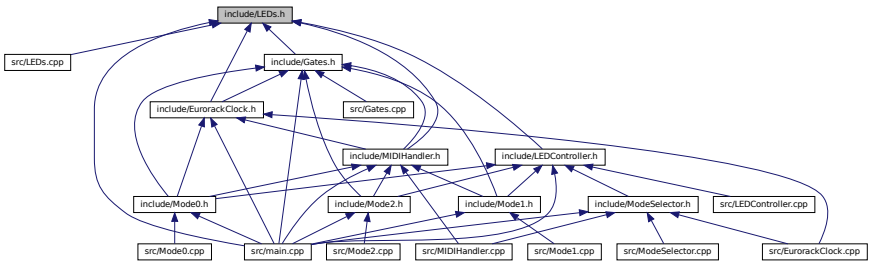
5.10 include/LEDs.h File Reference

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

Include dependency graph for LEDs.h:



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



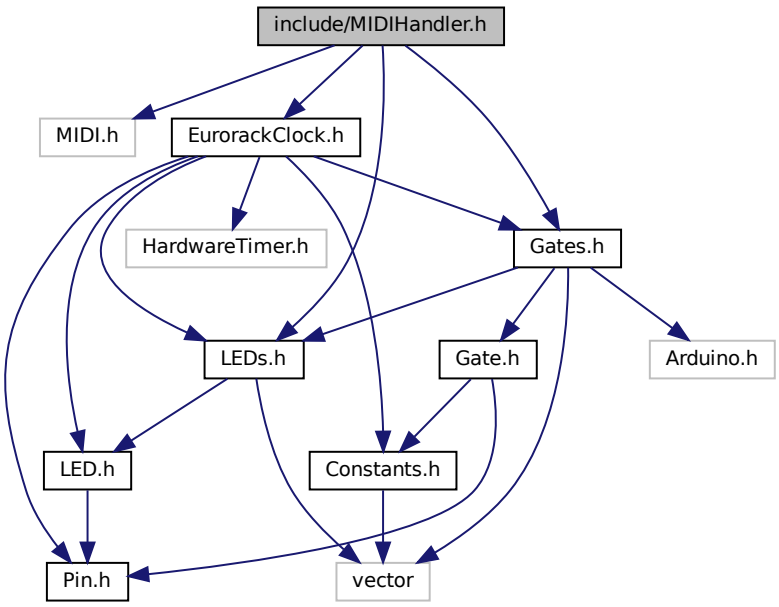
Classes

- class LEDs

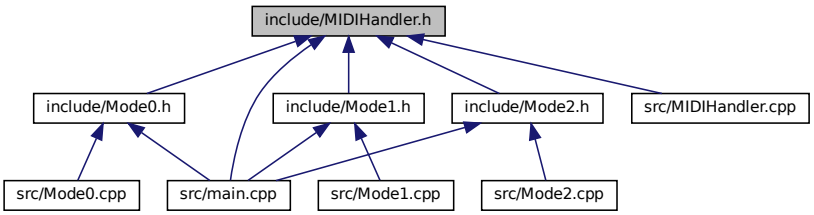
5.11 include/MIDIHandler.h File Reference

```
#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:

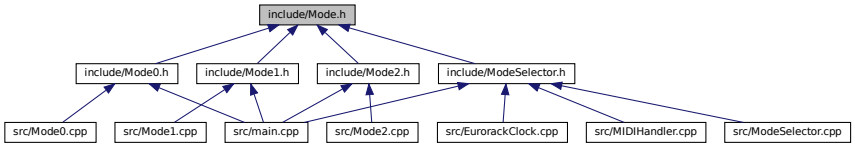


Classes

- class MIDIHandler

5.12 include/Mode.h File Reference

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



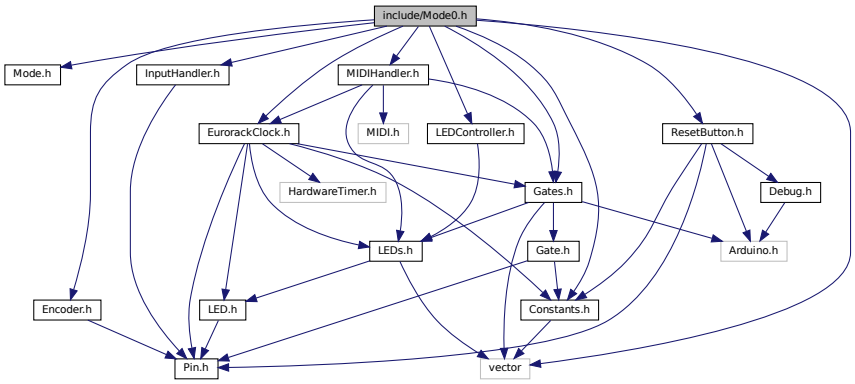
Classes

- class Mode

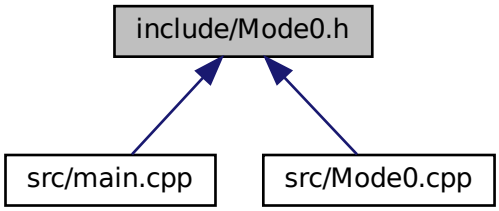
5.13 include/Mode0.h File Reference

```
#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:



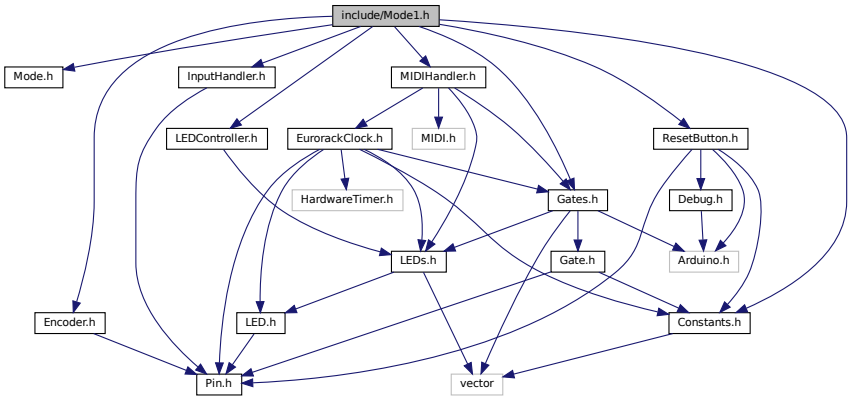
Classes

- class Mode0

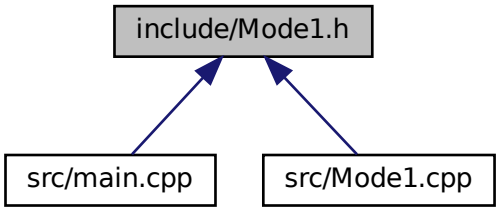
5.14 include/Mode1.h File Reference

```
#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:



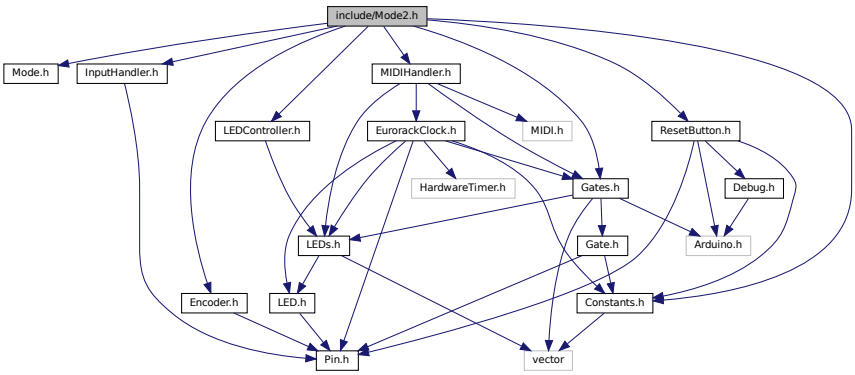
Classes

- class Mode1

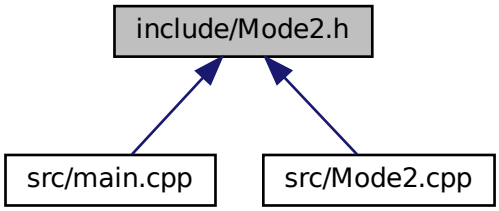
5.15 include/Mode2.h File Reference

```
#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:



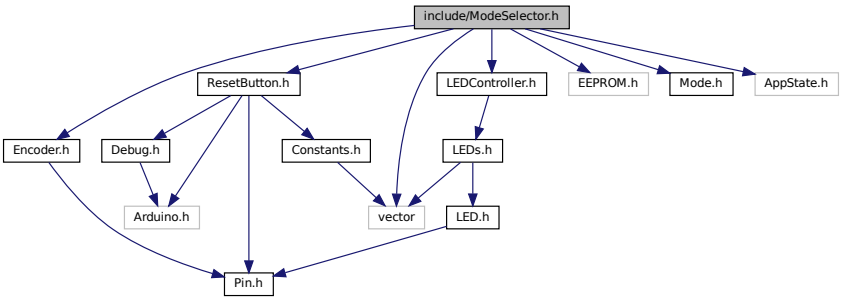
Classes

- class Mode2

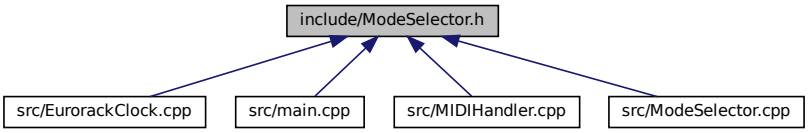
5.16 include/ModeSelector.h File Reference

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

Include dependency graph for ModeSelector.h:



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

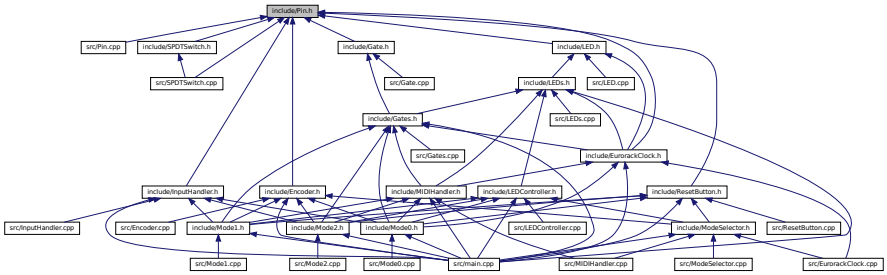


Classes

- class ModeSelector

5.17 include/Pin.h File Reference

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



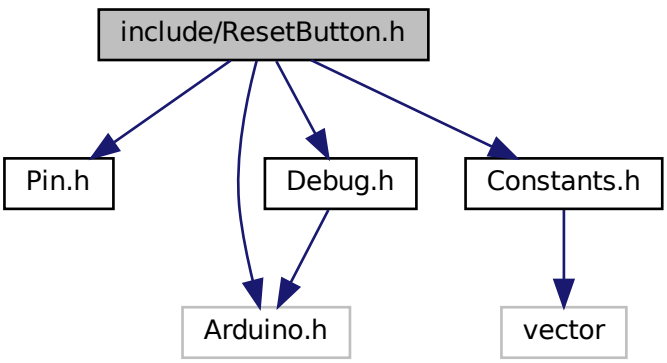
Classes

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

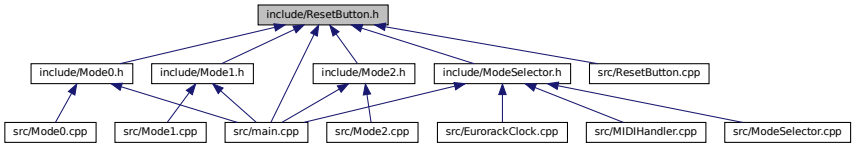
5.18 include/ResetButton.h File Reference

```
#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:



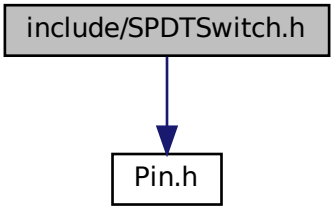
Classes

- class ResetButton

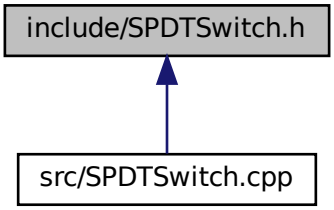
5.19 include/SPDTSwitch.h File Reference

```
#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
```

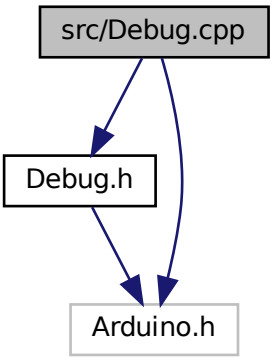
Enumerator

NEUTRAL	
STATE_A	
STATE_B	

5.20 src/Debug.cpp File Reference

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

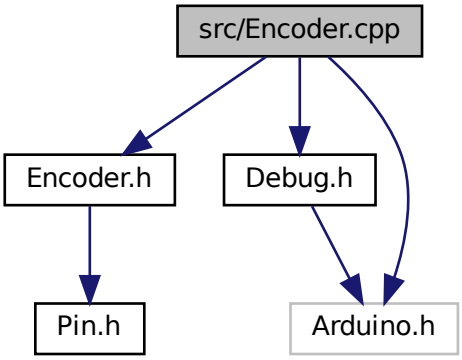
Include dependency graph for Debug.cpp:



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

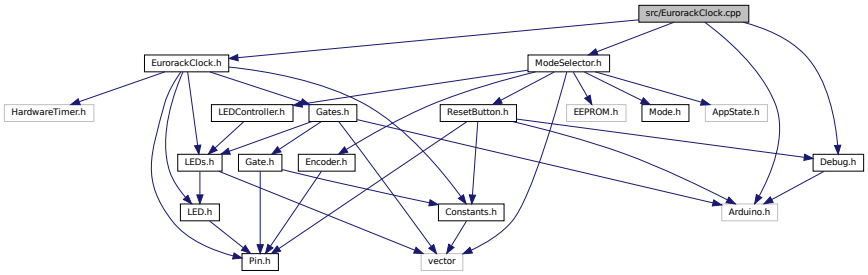
```
#define DEBUG_PRINT(
              message ) Debug::print(__FILE__, __LINE__, __func__, String(message))
```

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

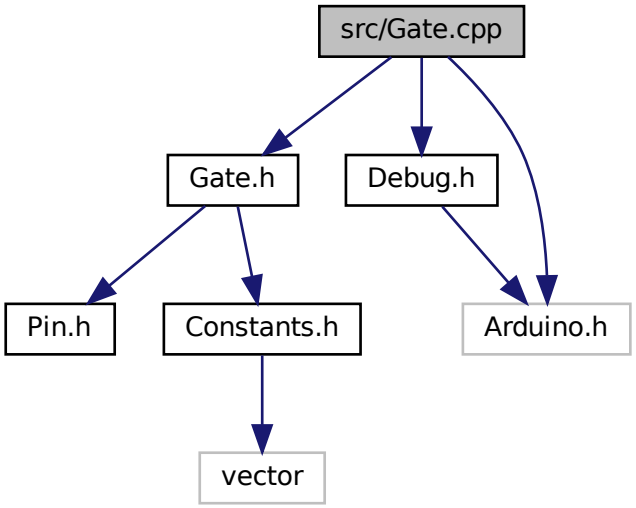
5.22.1.1 DEBUG_PRINT

```
#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

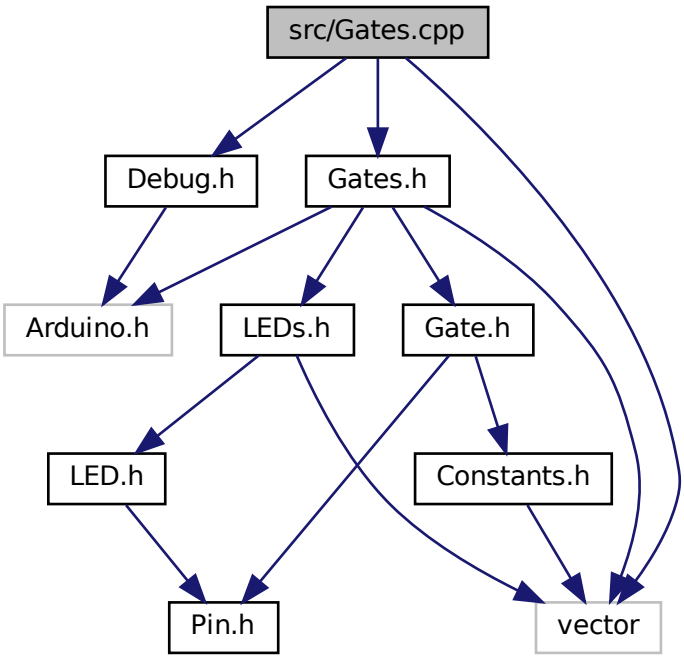
```
#define DEBUG_PRINT(
             message ) Debug::print(__FILE__, __LINE__, __func__, String(message))
```

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

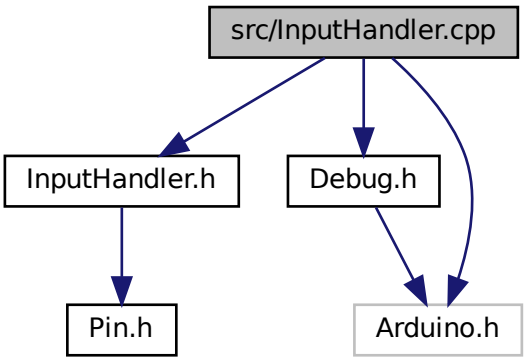
5.24.1.1 DEBUG_PRINT

```
#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

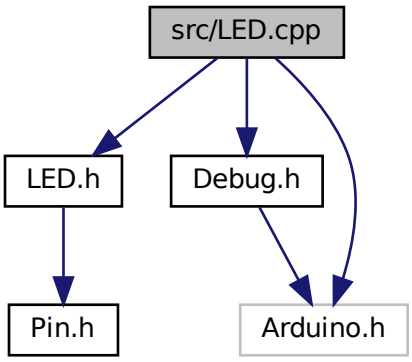
```
#define DEBUG_PRINT(
             message ) Debug::print(__FILE__, __LINE__, __func__, String(message))
```

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

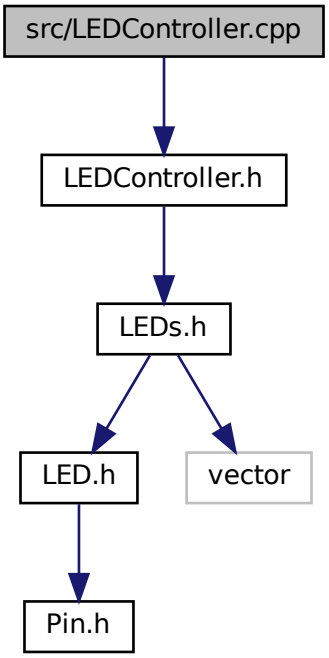
5.26.1.1 DEBUG_PRINT

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

5.27 src/LEDController.cpp File Reference

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

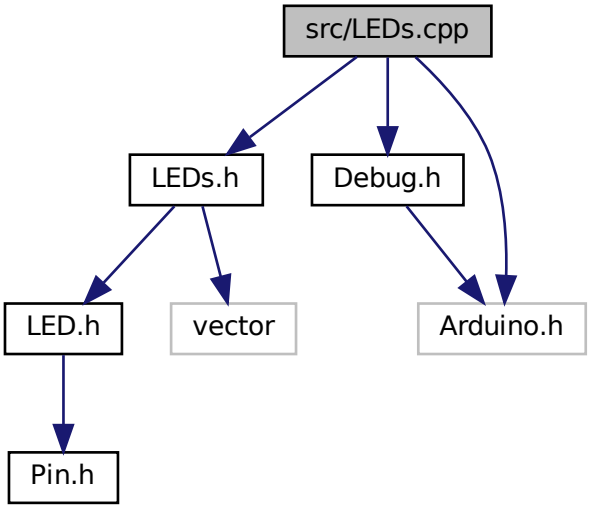
Include dependency graph for LEDController.cpp:



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

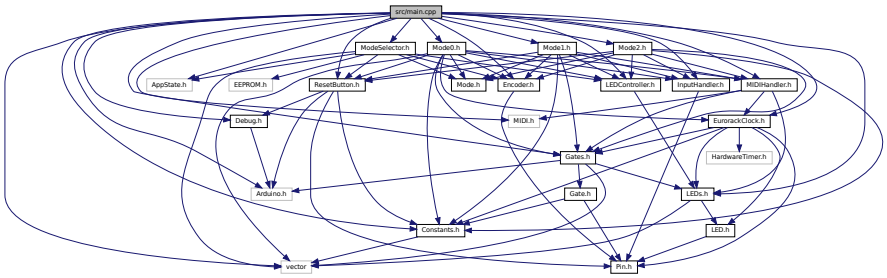
```
#define DEBUG_PRINT(
             message ) Debug::print(__FILE__, __LINE__, __func__, String(message))
```

5.29 src/main.cpp File Reference

```
#include <Arduino.h>
#include <MIDI.h>
#include <vector>
#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 "AppState.h"
```

Include dependency graph for main.cpp:



Macros

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

  Macro for debug print.
- #define RX_PIN PA3

  RX pin for MIDI communication.
- #define TX_PIN PA2

  TX pin for MIDI communication.
- #define ENCODER_PINA PB13

  Encoder pin A.
- #define ENCODER_PINB PB14

  Encoder pin B.
- #define ENCODER_BUTTON PB12

  Encoder button pin.
- #define CLOCK_PIN PB10

  Clock pin.
- #define RESET_PIN PB11

  Reset pin.
- #define RESET_BUTTON PB15

  Reset button pin.
- #define TEMPO_LED PA8

  Tempo LED pin.
- #define CV_A_PIN PA4

  CV A pin.
- #define CV_B_PIN PA5

  CV B pin.

Functions

- void setup ()

  Setup function for the Arduino sketch.
- void loop ()

  Main loop function for the Arduino sketch.

Variables

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

  Example pins for gates.
- const int numPins = pins.size()

  Number of gate pins.
- Gates gates = Gates(pins, numPins)

  Create an instance of Gates.
- std::vector< int > ledPins = {PA12, PA11, PB1, PB0, PA7, PA6, PA1, PA0}

  Placeholder pin numbers for LEDs.
- int numLedPins = ledPins.size()

  Number of LED pins.
- LEDs leds = LEDs(ledPins, numLedPins)

  Create an instance of LEDs.
- int encCLKPin = ENCODER_PINA

  Encoder CLK pin.
- int encDTPin = ENCODER_PINB

  Encoder DT pin.
- int encButtonPin = ENCODER_BUTTON

  Encoder button pin.
- bool inModeSelection = false

  Flag for mode selection.
- int intensity = 255

  Default intensity for LEDs.
- bool isInSelection = false

  Flag to prevent multiple presses from being handled.
- unsigned long lastFlashTime = 0

  Last flash time.
- unsigned char internalPPQN = 24

  Pulses per quarter note.
- 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()

  Size of musical intervals array.
- int total_pages = 16 / leds.numLeds

  Calculate total pages based on number of LEDs.
- int min_intensity = 64

  Set minimum intensity to 25% (64 out of 255)
- int intensity_step = (255 - min_intensity) / (total_pages - 1)

  Calculate intensity step.
- AppState state

  Instance of the AppState struct.
- Encoder encoder = Encoder(encCLKPin, encDTPin, encButtonPin)

Instance of the Encoder class.

- ResetButton resetButton = ResetButton(RESET_BUTTON)

  Instance of the ResetButton class.
- LEDController ledController (leds)

  Instance of the LEDController class.
- EurorackClock clock (CLOCK_PIN, RESET_PIN, TEMPO_LED, gates, leds)

  Instance of the EurorackClock class.
- MIDIHandler midiHandler (Serial2, clock, gates, leds)

  Instance of the MIDIHandler class.
- InputHandler inputHandler = InputHandler(CV_A_PIN, CV_B_PIN)

  Instance of the InputHandler class.
- ModeSelector & modeSelector = ModeSelector::getInstance()

  Instance of the ModeSelector class.
- Mode * currentMode = nullptr

  Pointer to the current mode.
- Mode0 mode0 (encoder, inputHandler, gates, ledController, midiHandler, resetButton, clock)

  Instance of Mode0 class.
- Mode1 mode1 (encoder, inputHandler, gates, ledController, midiHandler, resetButton)

  Instance of Mode1 class.
- Mode2 mode2 (encoder, inputHandler, gates, ledController, midiHandler, resetButton)

  Instance of Mode2 class.

5.29.1 Macro Definition Documentation

5.29.1.1 CLOCK_PIN

```
#define CLOCK_PIN PB10
```

Clock pin.

5.29.1.2 CV_A_PIN

```
#define CV_A_PIN PA4
```

CV A pin.

5.29.1.3 CV_B_PIN

```
#define CV_B_PIN PA5
```

CV B pin.

5.29.1.4 DEBUG_PRINT

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

Macro for debug print.

5.29.1.5 ENCODER_BUTTON

```
#define ENCODER_BUTTON PB12
```

Encoder button pin.

5.29.1.6 ENCODER_PINA

```
#define ENCODER_PINA PB13
```

Encoder pin A.

5.29.1.7 ENCODER_PINB

```
#define ENCODER_PINB PB14
```

Encoder pin B.

5.29.1.8 RESET_BUTTON

```
#define RESET_BUTTON PB15
```

Reset button pin.

5.29.1.9 RESET_PIN

```
#define RESET_PIN PB11
```

Reset pin.

5.29.1.10 RX_PIN

```
#define RX_PIN PA3
```

RX pin for MIDI communication.

5.29.1.11 TEMPO_LED

```
#define TEMPO_LED PA8
```

Tempo LED pin.

5.29.1.12 TX_PIN

```
#define TX_PIN PA2
```

TX pin for MIDI communication.

5.29.2 Function Documentation

5.29.2.1 loop()

```
void loop ( )
```

Main loop function for the Arduino sketch.

This function is called repeatedly as long as the Arduino is powered on. It contains the main logic of the sketch. < Update the ModeSelector

< Update the LEDController's blinking status

< If not in mode selection

< Update the current mode

< If in mode selection

< Teardown the current mode

< Get the new current mode from the ModeSelector

< Setup the new current mode

5.29.2.2 setup()

```
void setup ( )
```

Setup function for the Arduino sketch.

This function is called once when the sketch starts. It is used to initialize variables, input and output pin modes, and start using libraries. < Enable debugging

< Initialize serial communication

< Print debug message

< Set the RESET_BUTTON pin to INPUT_PULLDOWN mode

< Initialize the MIDIHandler

< Set the MIDIHandler to listen to all channels

< Start the clock

< Set the tempo to 120 BPM with internal 4 PPQN

< Add Mode0 to the ModeSelector

< Add Mode1 to the ModeSelector

< Add Mode2 to the ModeSelector

< Set the LEDController for the ModeSelector

< Set the Encoder for the ModeSelector

< Set the AppState for the ModeSelector

< Set the current mode for the ModeSelector

< Get the current mode from the ModeSelector

< Run the setup function for the current mode

< Initialize LED pins

< Initialize gate pins

< Initialize encoder pins

< Print debug message

< Print the current mode

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  )
```

Instance of the EurorackClock class.

5.29.3.2 currentMode

```
Mode* currentMode = nullptr
```

Pointer to the current mode.

5.29.3.3 encButtonPin

```
int encButtonPin = ENCODER_BUTTON
```

Encoder button pin.

5.29.3.4 encCLKPin

```
int encCLKPin = ENCODER_PINA
```

Encoder CLK pin.

5.29.3.5 encDTPin

```
int encDTPin = ENCODER_PINB
```

Encoder DT pin.

5.29.3.6 encoder

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

Instance of the Encoder class.

5.29.3.7 gates

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

Create an instance of Gates.

5.29.3.8 inModeSelection

```
bool inModeSelection = false
```

Flag for mode selection.

5.29.3.9 inputHandler

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

Instance of the InputHandler class.

5.29.3.10 intensity

```
int intensity = 255
```

Default intensity for LEDs.

5.29.3.11 intensity_step

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

Calculate intensity step.

5.29.3.12 internalPPQN

```
unsigned char internalPPQN = 24
```

Pulses per quarter note.

5.29.3.13 isInSelection

```
bool isInSelection = false
```

Flag to prevent multiple presses from being handled.

5.29.3.14 lastFlashTime

```
unsigned long lastFlashTime = 0
```

Last flash time.

5.29.3.15 ledController

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

Instance of the LEDController class.

5.29.3.16 ledPins

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

Placeholder pin numbers for LEDs.

5.29.3.17 leds

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

Create an instance of LEDs.

5.29.3.18 midiHandler

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

Instance of the MIDIHandler class.

5.29.3.19 min_intensity

```
int min_intensity = 64
```

Set minimum intensity to 25% (64 out of 255)

5.29.3.20 mode0

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

Instance of Mode0 class.

5.29.3.21 mode1

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

Instance of Mode1 class.

5.29.3.22 mode2

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

Instance of Mode2 class.

5.29.3.23 modeSelector

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

Instance of the ModeSelector class.

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()
```

Size of musical intervals array.

5.29.3.26 numLedPins

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

Number of LED pins.

5.29.3.27 numPins

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

Number of gate pins.

5.29.3.28 pins

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

Example pins for gates.

5.29.3.29 resetButton

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

Instance of the ResetButton class.

5.29.3.30 state

```
AppState state
```

Instance of the AppState struct.

5.29.3.31 total_pages

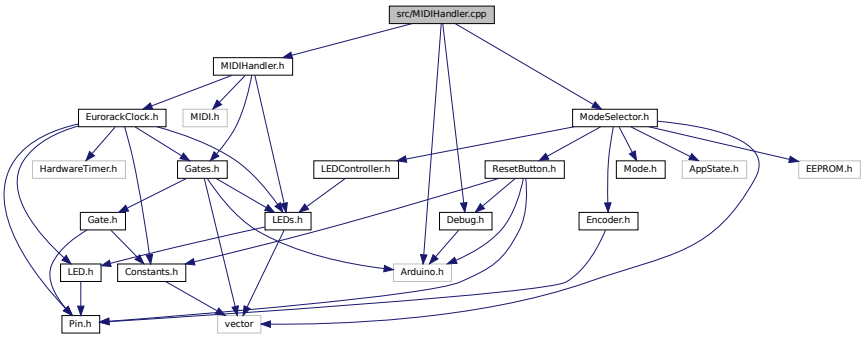
```
int total_pages = 16 / leds.numLeds
```

Calculate total pages based on number of LEDs.

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

  Flag to prevent multiple presses from being handled.

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]
```

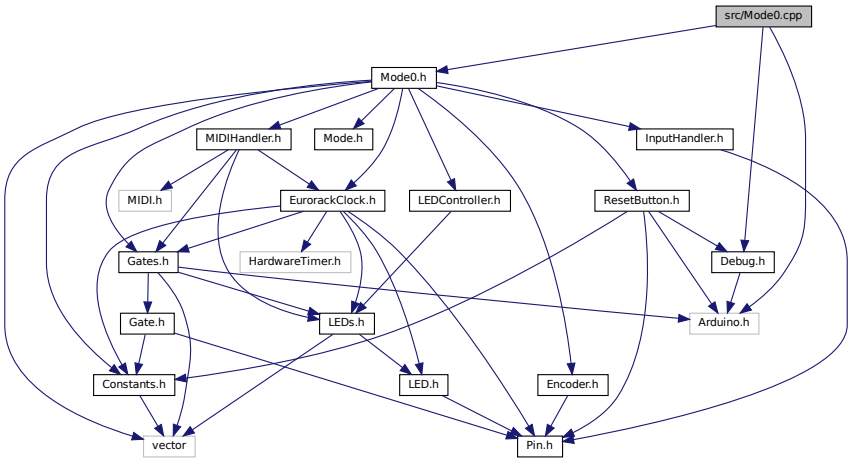
Flag to prevent multiple presses from being handled.

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

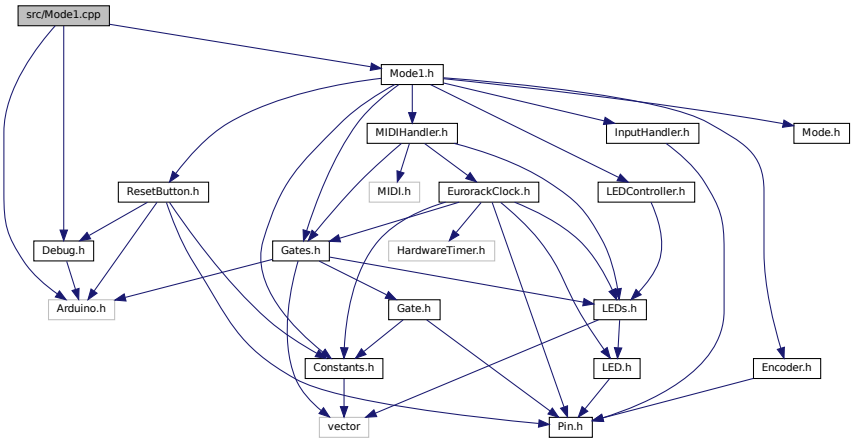
5.32.1.1 DEBUG_PRINT

```
#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

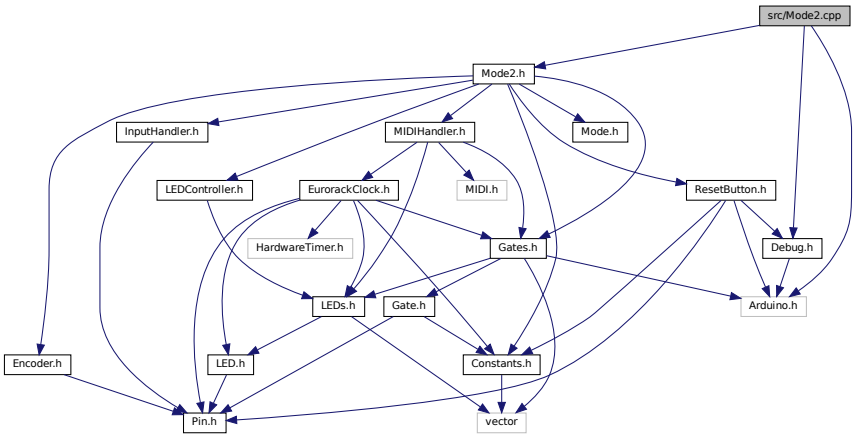
5.33.1.1 DEBUG_PRINT

```
#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 )
```

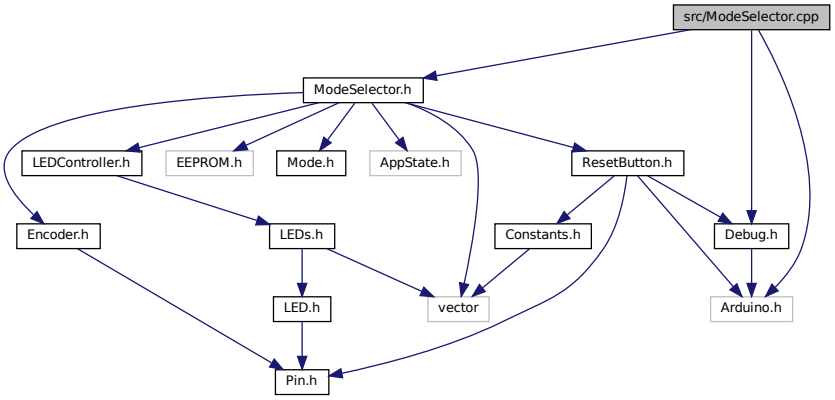
Value:

```
    { \
    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

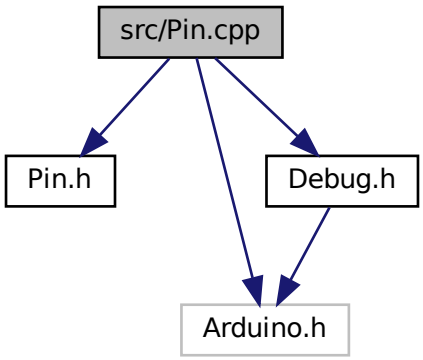
```
#define DEBUG_PRINT(
             message ) Debug::print(__FILE__, __LINE__, __func__, String(message))
```

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

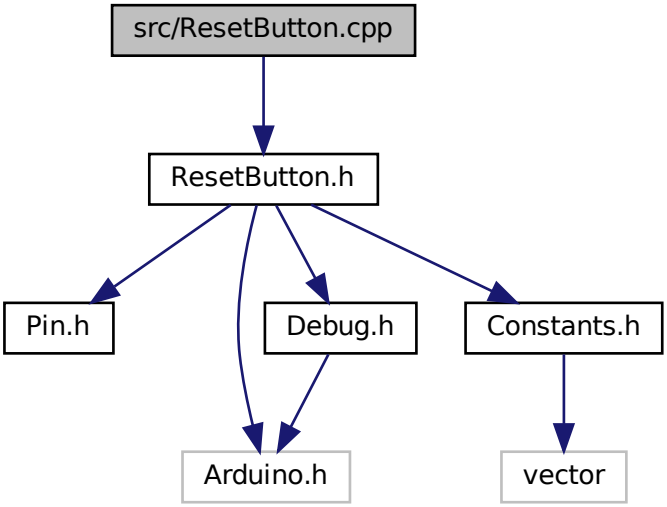
5.36.1.1 DEBUG_PRINT

```
#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: