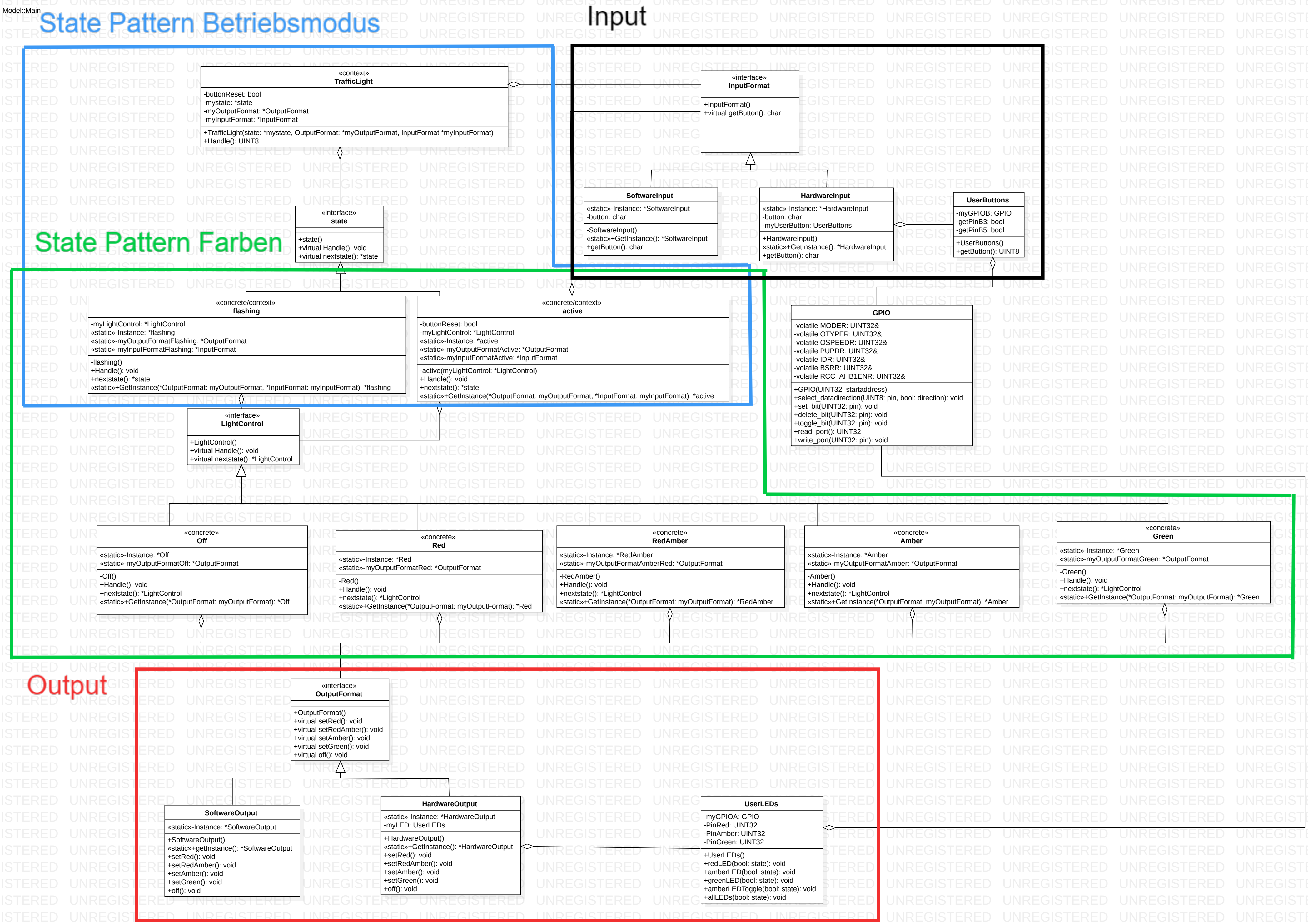
Model::Main
State Pattern Betriebsmodus
Input
State Pattern Farben
Output
«context»
TrafficLight
-buttonReset: bool
-mystate: *state
-myOutputFormat: *OutputFormat
-myInputFormat: *InputFormat
+TrafficLight(state: *mystate, OutputFormat: *myOutputFormat, InputFormat *myInputFormat)
+Handle(): UINT8
«interface»
InputFormat
+InputFormat()
+virtual getButton(): char
SoftwareInput
«static»-Instance: *SoftwareInput
-button: char
-SoftwareInput()
«static»+GetInstance(): *SoftwareInput
+getButton(): char
HardwareInput
«static»-Instance: *HardwareInput
-button: char
-myUserButton: UserButtons
+HardwareInput()
«static»+GetInstance(): *HardwareInput
+getButton(): char
UserButtons
-myGPIOB: GPIO
-getPinB3: bool
-getPinB5: bool
+UserButtons()
+getButton(): UINT8
«interface»
state
+state()
+virtual Handle(): void
+virtual nextstate(): *state
«concrete/context»
flashing
-myLightControl: *LightControl
«static»-Instance: *flashing
«static»-myOutputFormatFlashing: *OutputFormat
«static»-myInputFormatFlashing: *InputFormat
-flashing()
+Handle(): void
+nextstate(): *state
«static»+GetInstance(*OutputFormat: myOutputFormat, *InputFormat: myInputFormat): *flashing
«concrete/context»
active
-buttonReset: bool
-myLightControl: *LightControl
«static»-Instance: *active
«static»-myOutputFormatActive: *OutputFormat
«static»-myInputFormatActive: *InputFormat
-active(myLightControl: *LightControl)
+Handle(): void
+nextstate(): *state
«static»+GetInstance(*OutputFormat: myOutputFormat, *InputFormat: myInputFormat): *active
GPIO
-volatile MODER: UINT32&
-volatile OTYPER: UINT32&
-volatile OSPEEDR: UINT32&
-volatile PUPDR: UINT32&
-volatile IDR: UINT32&
-volatile BSRR: UINT32&
-volatile RCC_AHB1ENR: UINT32&
+GPIO(UINT32: startaddress)
+select_datadirection(UINT8: pin, bool: direction): void
+set_bit(UINT32: pin): void
+delete_bit(UINT32: pin): void
+toggle_bit(UINT32: pin): void
+read_port(): UINT32
+write_port(UINT32: pin): void
«interface»
LightControl
+LightControl()
+virtual Handle(): void
+virtual nextstate(): *LightControl
«concrete»
Off
«static»-Instance: *Off
«static»-myOutputFormatOff: *OutputFormat
-Off()
+Handle(): void
+nextstate(): *LightControl
«static»+GetInstance(*OutputFormat: myOutputFormat): *Off
«concrete»
Red
«static»-Instance: *Red
«static»-myOutputFormatRed: *OutputFormat
-Red()
+Handle(): void
+nextstate(): *LightControl
«static»+GetInstance(*OutputFormat: myOutputFormat): *Red
«concrete»
RedAmber
«static»-Instance: *RedAmber
«static»-myOutputFormatAmberRed: *OutputFormat
-RedAmber()
+Handle(): void
+nextstate(): *LightControl
«static»+GetInstance(*OutputFormat: myOutputFormat): *RedAmber
«concrete»
Amber
«static»-Instance: *Amber
«static»-myOutputFormatAmber: *OutputFormat
-Amber()
+Handle(): void
+nextstate(): *LightControl
«static»+GetInstance(*OutputFormat: myOutputFormat): *Amber
«concrete»
Green
«static»-Instance: *Green
«static»-myOutputFormatGreen: *OutputFormat
-Green()
+Handle(): void
+nextstate(): *LightControl
«static»+GetInstance(*OutputFormat: myOutputFormat): *Green
«interface»
OutputFormat
+OutputFormat()
+virtual setRed(): void
+virtual setRedAmber(): void
+virtual setAmber(): void
+virtual setGreen(): void
+virtual off(): void
SoftwareOutput
«static»-Instance: *SoftwareOutput
+SoftwareOutput()
«static»+getInstance(): *SoftwareOutput
+setRed(): void
+setRedAmber(): void
+setAmber(): void
+setGreen(): void
+off(): void
HardwareOutput
«static»-Instance: *HardwareOutput
-myLED: UserLEDs
+HardwareOutput()
«static»+GetInstance(): *HardwareOutput
+setRed(): void
+setRedAmber(): void
+setAmber(): void
+setGreen(): void
+off(): void
UserLEDs
-myGPIOA: GPIO
-PinRed: UINT32
-PinAmber: UINT32
-PinGreen: UINT32
+UserLEDs()
+redLED(bool: state): void
+amberLED(bool: state): void
+greenLED(bool: state): void
+amberLEDToggle(bool: state): void
+allLEDs(bool: state): void

Model1::ClassDiagram1