Clock::cClock

- mHour: cLimitCounter
- mMinute: cOverflowCounter
- mSecond: cOverflowCounter

+ cClock(Hour: uint32_t, Minute: uint32_t, Second: uint32_t)
+ *~cClock()*
+ setHour(Hour: uint32_t): void
+ setMinute(Minute: uint32_t): void
+ setSecond(Second: uint32_t): void
+ reset(): void
+ getHour(): uint32_t {query}
+ getMinute(): uint32_t {query}
+ getSecond(): uint32_t {query}
+ update(): void
+ show(): void {query}

Counter::cLimitCounter

\# mCountValue: uint32_t
\# mCountLimit: uint32_t {readOnly}

+ cLimitCounter(CountValue: uint32_t, CountLimit: uint32_t)
+ *~cLimitCounter()*
+ setCountValue(CountValue: uint32_t): void
+ getCountValue(): uint32_t {query}
+ *resetCountValue(): void*
+ *count(): void*

-mHour

Counter::cOverflowCounter   {leaf}

- mrefNextCounter: cLimitCounter&

+ cOverflowCounter(CountValue: uint32_t, CountLimit: uint32_t, refNextCounter: cLimitCounter&)
+ ~cOverflowCounter()
+ *resetCountValue(): void*
+ *count(): void*

-mMinute

-mSecond

-mrefNextCounter

Clock_Observer_Application::cClockObserverApplicationBuilder

- mClock: cClock

+ cClockObserverApplicationBuilder()
+ ~cClockObserverApplicationBuilder()
+ execute(): void
- buildRelations(): void

-mClock

Clock_Observer_Application::
cClockObserverApplicationBuilder

- mClock: cClock
- mAnalogClock: cAnalogClock
- mDigitalClock: cDigitalClock
- mAnalogDigitalClock: cAnalogDigitalClock

+ cClockObserverApplicationBuilder()
+ ~cClockObserverApplicationBuilder()
+ execute(): void
- buildRelations(): void

Clock::cClock

- mHour: cLimitCounter
- mMinute: cOverflowCounter
- mSecond: cOverflowCounter

+ cClock(Hour: uint32_t, Minute: uint32_t, Second: uint32_t)
+ *~cClock()*
+ setHour(Hour: uint32_t): void
+ setMinute(Minute: uint32_t): void
+ setSecond(Second: uint32_t): void
+ reset(): void
+ getHour(): uint32_t {query}
+ getMinute(): uint32_t {query}
+ getSecond(): uint32_t {query}
+ *update(): void*
+ *show(): void {query}*

-mClock

-mDigitalClock

Clock::cDigitalClock

+ cDigitalClock(Hour: uint32_t, Minute: uint32_t, Second: uint32_t)
+ *~cDigitalClock()*
+ *show(): void {query}*

Clock::cAnalogClock

+ cAnalogClock(Hour: uint32_t, Minute: uint32_t, Second: uint32_t)
+ *~cAnalogClock()*
+ *show(): void {query}*

-mAnalogClock

-mAnalogDigitalClock

Clock::cAnalogDigitalClock   {leaf}

+ cAnalogDigitalClock(HourAnalog: uint32_t, MinuteAnalog: uint32_t, SecondAnalog: uint32_t, HourDigital: uint32_t, MinuteDigital: uint32_t, SecondDigital: uint32_t)
+ *~cAnalogDigitalClock()*
+ *update(): void*
+ *show(): void {query}*

**Clock_Observer_Application::
cClockObserverApplicationBuilder**

- - mClock: cClock
- - mptrAnalogClock: cAnalogClock*
- - mptrDigitalClock: cDigitalClock*
- - mptrAnalogDigitalClock: cAnalogDigitalClock*

- + cClockObserverApplicationBuilder()
- + ~cClockObserverApplicationBuilder()
- + execute(): void
- - buildRelations(): void

{leaf}

Clock::cClock

- - mHour: cLimitCounter
- - mMinute: cOverflowCounter
- - mSecond: cOverflowCounter

- + cClock(Hour: uint32_t, Minute: uint32_t, Second: uint32_t)
- + ~cClock()
- + setHour(Hour: uint32_t): void
- + setMinute(Minute: uint32_t): void
- + setSecond(Second: uint32_t): void
- + reset(): void
- + getHour(): uint32_t {query}
- + getMinute(): uint32_t {query}
- + getSecond(): uint32_t {query}
- + *update(): void*
- + *show(): void {query}*

Counter::cLimitCounter

- \# mCountValue: uint32_t
- \# mCountLimit: uint32_t {readOnly}

- + cLimitCounter(CountValue: uint32_t, CountLimit: uint32_t)
- + *~cLimitCounter()*
- + setCountValue(CountValue: uint32_t): void
- + getCountValue(): uint32_t {query}
- + *resetCountValue(): void*
- + *count(): void*

-mHour

-mrefNextCounter

Counter::cOverflowCounter

{leaf}

- - mrefNextCounter: cLimitCounter&

- + cOverflowCounter(CountValue: uint32_t, CountLimit: uint32_t, refNextCounter: cLimitCounter&)
- + ~cOverflowCounter()
- + *resetCountValue(): void*
- + *count(): void*

-mMinute

-mSecond

Clock::cDigitalClock

- + cDigitalClock(Hour: uint32_t, Minute: uint32_t, Second: uint32_t)
- + ~cDigitalClock()
- + *show(): void {query}*

-mptrDigitalClock

Clock::cAnalogClock

-mptrAnalogClock

- + cAnalogClock(Hour: uint32_t, Minute: uint32_t, Second: uint32_t)
- + ~cAnalogClock()
- + *show(): void {query}*

Clock::cAnalogDigitalClock

{leaf}

-mptrAnalogDigitalClock

- + cAnalogDigitalClock(HourAnalog: uint32_t, MinuteAnalog: uint32_t, SecondAnalog: uint32_t, HourDigital: uint32_t, MinuteDigital: uint32_t, SecondDigital: uint32_t)
- + ~cAnalogDigitalClock()
- + *update(): void*
- + *show(): void {query}*

---

std::exception

User_Exception::excRangeException

- + excRangeException()
- + *~excRangeException()*
- + *what(): char* {query}*

-mClock

Clock_Observer_Application::cClockObserverApplicationBuilder

- mClock: cClock
- mptrAnalogClock: std::shared_ptr<cAnalogClock>
- mptrDigitalClock: std::shared_ptr<cDigitalClock>
- mptrAnalogDigitalClock: std::shared_ptr<cAnalogDigitalClock>
- mptrClockArray: std::shared_ptr<tcClockArray<10>>

+ cClockObserverApplicationBuilder()
+ ~cClockObserverApplicationBuilder()
+ execute(): void
- buildRelations(): void

Common::icSubject

+ ~icSubject()
+ attach(ptrObserver: std::shared_ptr<icObserver>): void
+ detach(ptrObserver: std::shared_ptr<icObserver>): void
+ notify(): void

«forward_declaration»

Common::icObserver

+ ~icObserver()
+ update(): void
+ show(): void {query}

ArrayLength : uint32_t

Clock_Source::tcClockArray

- mObserverPtrArray: tcArray<std::shared_ptr<icObserver>, ArrayLength>

+ tcClockArray()
+ ~tcClockArray()
+ attach(ptrObserver: std::shared_ptr<icObserver>): void
+ detach(ptrObserver: std::shared_ptr<icObserver>): void
+ notify(): void

-mptrClockArray

-mClock

Clock::cClock

- mHour: cLimitCounter
- mMinute: cOverflowCounter
- mSecond: cOverflowCounter

+ cClock(Hour: uint32_t, Minute: uint32_t, Second: uint32_t)
+ ~cClock()
+ setHour(Hour: uint32_t): void
+ setMinute(Minute: uint32_t): void
+ setSecond(Second: uint32_t): void
+ reset(): void
+ getHour(): uint32_t {query}
+ getMinute(): uint32_t {query}
+ getSecond(): uint32_t {query}
+ update(): void
+ show(): void {query}

< T->std::shared_ptr<icObserver>, ArrayLength->ArrayLength >

-mObserverPtrArray

Element_T : typename
ArrayLength : uint32_t

Common::tcArray

- mInitValue: Element_T
- mArray: Element_T ([ArrayLength])

+ tcArray(InitValue: Element_T)
+ ~tcArray()
+ insert(Element: Element_T): void
+ erase(Element: Element_T): void
+ operator[](Index: uint32_t): Element_T&
+ operator[](Index: uint32_t): Element_T& {query}

«forward_declaration»

Clock_Observer_Application::cClockObserverApplicationBuilder

- mClock: cClock
- mptrAnalogClock: std::shared_ptr<cAnalogClock>
- mptrDigitalClock: std::shared_ptr<cDigitalClock>
- mptrAnalogDigitalClock: std::shared_ptr<cAnalogDigitalClock>
- mptrClockArray: std::shared_ptr<tcClockArray<10>>
- mptrClockContainer: std::shared_ptr<cClockContainer>

+ cClockObserverApplicationBuilder()
+ ~cClockObserverApplicationBuilder()
+ execute(): void
- buildRelations(): void

Common::icSubject

+ *~icSubject()*
+ *attach(ptrObserver: std::shared_ptr<icObserver>): void*
+ *detach(ptrObserver: std::shared_ptr<icObserver>): void*
+ *notify(): void*

Common::icObserver

+ *~icObserver()*
+ *update(): void*
+ *show(): void {query}*

Clock::cClock

-mClock

-mptrClockArray

ArrayLength : uint32_t

Clock_Source::tcClockArray

Clock_Source::cClockContainer

- mObserverPtrList: list<std::shared_ptr<icObserver>>

+ cClockContainer()
+ ~cClockContainer()
+ *attach(ptrObserver: std::shared_ptr<icObserver>): void*
+ *detach(ptrObserver: std::shared_ptr<icObserver>): void*
+ *notify(): void*

< T->std::shared_ptr<icObserver>, ArrayLength->ArrayLength >

-mObserverPtrArray

Element_T : typename
ArrayLength : uint32_t

Common::tcArray

-mptrClockContainer

Range_T : typename
ThrowingObject_T : typename

std::exception

User_Exception::extcRangeException

- mOutOfRangeValue: Range_T = 0
- mptrThrowingObject: ThrowingObject_T* = nullptr

+ extcRangeException()
+ extcRangeException(OutOfRangeValue: Range_T, ptrThrowingObject: ThrowingObject_T* const)
+ *~extcRangeException()*
+ getOutOfRangeValue(): Range_T {query}
+ getThrowingObject(): ThrowingObject_T* {query}
+ *what(): char* {query}*

«forward_declaration»

Clock_Observer_Application::cClockObserverApplicationBuilder

- mClock: cClock
- mptrAnalogClock: std::shared_ptr<cAnalogClock>
- mptrDigitalClock: std::shared_ptr<cDigitalClock>
- mptrAnalogDigitalClock: std::shared_ptr<cAnalogDigitalClock>
- mptrClockArray: std::shared_ptr<tcClockArray<10>>
- mptrClockContainer: std::shared_ptr<cClockContainer>
- mptrTickTimer: std::shared_ptr<cTickTimer>
- mptrTimer: std::shared_ptr<cSTM32F7xxTIMER_Driver>

+ cClockObserverApplicationBuilder()
+ ~cClockObserverApplicationBuilder()
+ execute(): void
- buildRelations(): void

Common::icSubject

+ ~icSubject()
+ attach(std::shared_ptr<icObserver>): void
+ detach(std::shared_ptr<icObserver>): void
+ notify(): void

Common::icObserver

+ ~icObserver()
+ update(): void
+ show(): void{query}

-mptrClockArray

ArrayLength : uint32_t

Clock_Source::tcClockArray

-mptr_icSubject 2

< T->std::shared_ptr<icObserver>, ArrayLength->ArrayLength >

-mObserverPtrArray

Element_T : typename
ArrayLength : uint32_t

Common::tcArray

Clock_Source::cTickTimer   {leaf}

- mArrayLength: uint32_t = 2 {readOnly}
- mptr_icSubject: std::shared_ptr<icSubject> ([mArrayLength])

+ cTickTimer()
+ ~cTickTimer()
+ set_ptr_iSubject(std::shared_ptr<icSubject>, uint32_t): void
+ tick(): void
- notifyInterruptService(): void

-mptrClockContainer

Clock_Source::cClockContainer

-mptrTickTimer

-mClock

Clock::cClock

Device_Driver_Abstraction::icbInterruptDispatcher

+ notifyInterruptService(): void

-mptrCallbackObject 98

-mptrTimer

TIMER::cSTM32F7xxTIMER_Driver

- mptrRegisterBaseAddress: volatile cSTM32F7xxTIMER_Register*

+ cSTM32F7xxTIMER_Driver(vuint32_t)
+ ~cSTM32F7xxTIMER_Driver()
+ notifyInterruptAcknowledge(): void
+ start(): void
+ stop(): void
- clearUIEInterruptFlag(): void

NVIC_Dispatcher::cSTM32F7xxNVIC_Dispatcher

- mptrCallbackObject: icbInterruptDispatcher* ([InterruptVectorNumberAmount])
- mptrAcknowledgeObject: icbInterruptAcknowledge* ([InterruptVectorNumberAmount])

+ init(): void
+ registerInterruptCallback(icbInterruptDispatcher* const, InterruptVectorNumber_t): void
+ unregisterInterruptCallback(InterruptVectorNumber_t): void
+ registerInterruptAcknowledge(icbInterruptAcknowledge* const, InterruptVectorNumber_t): void
+ unregisterInterruptAcknowledge(InterruptVectorNumber_t): void

«interrupt_service»
+ TIM7_IRQHandler(): void