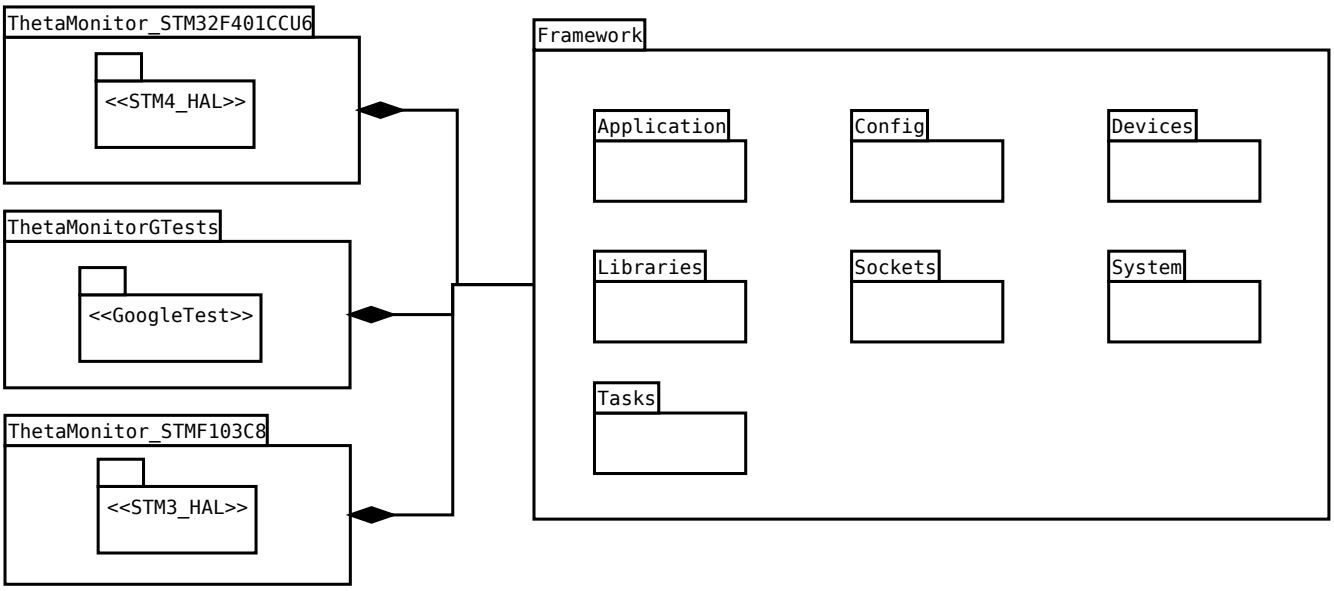
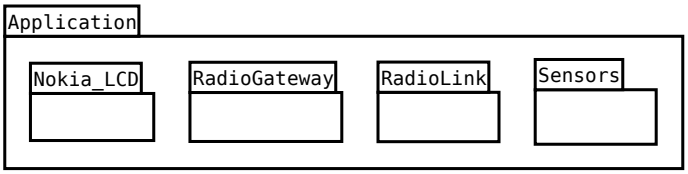
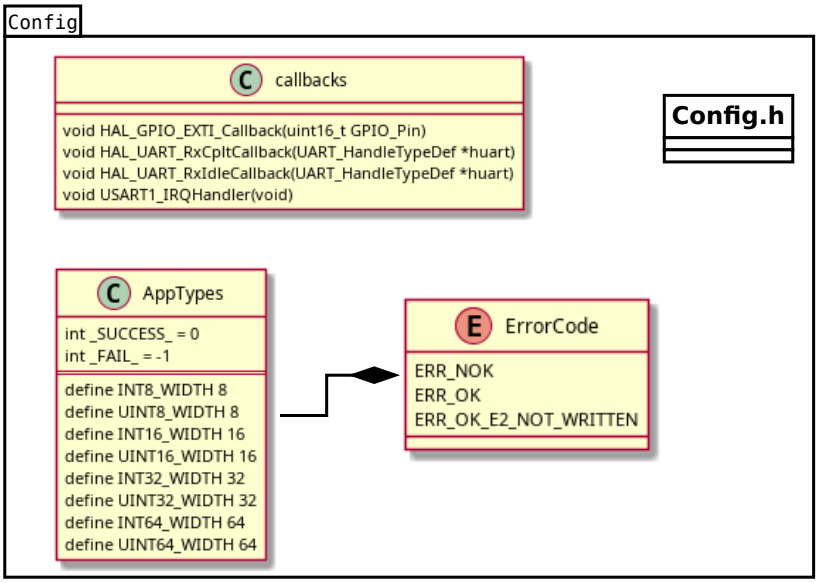
# ThetaMonitorNetwork

## Overview

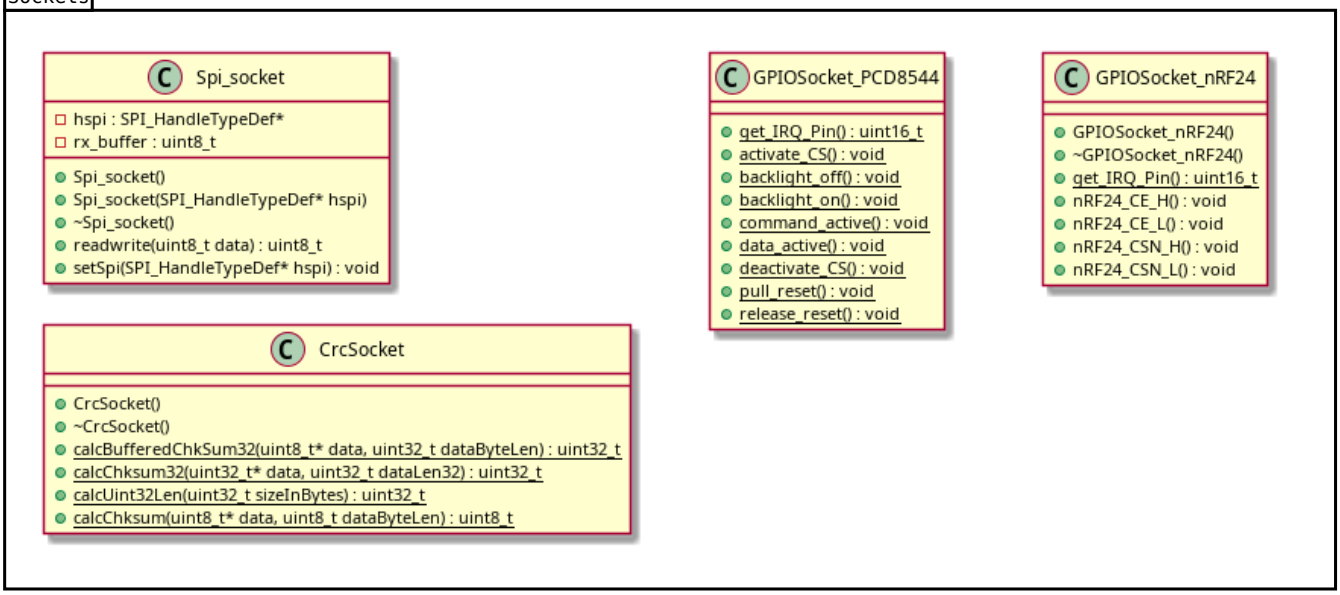



---

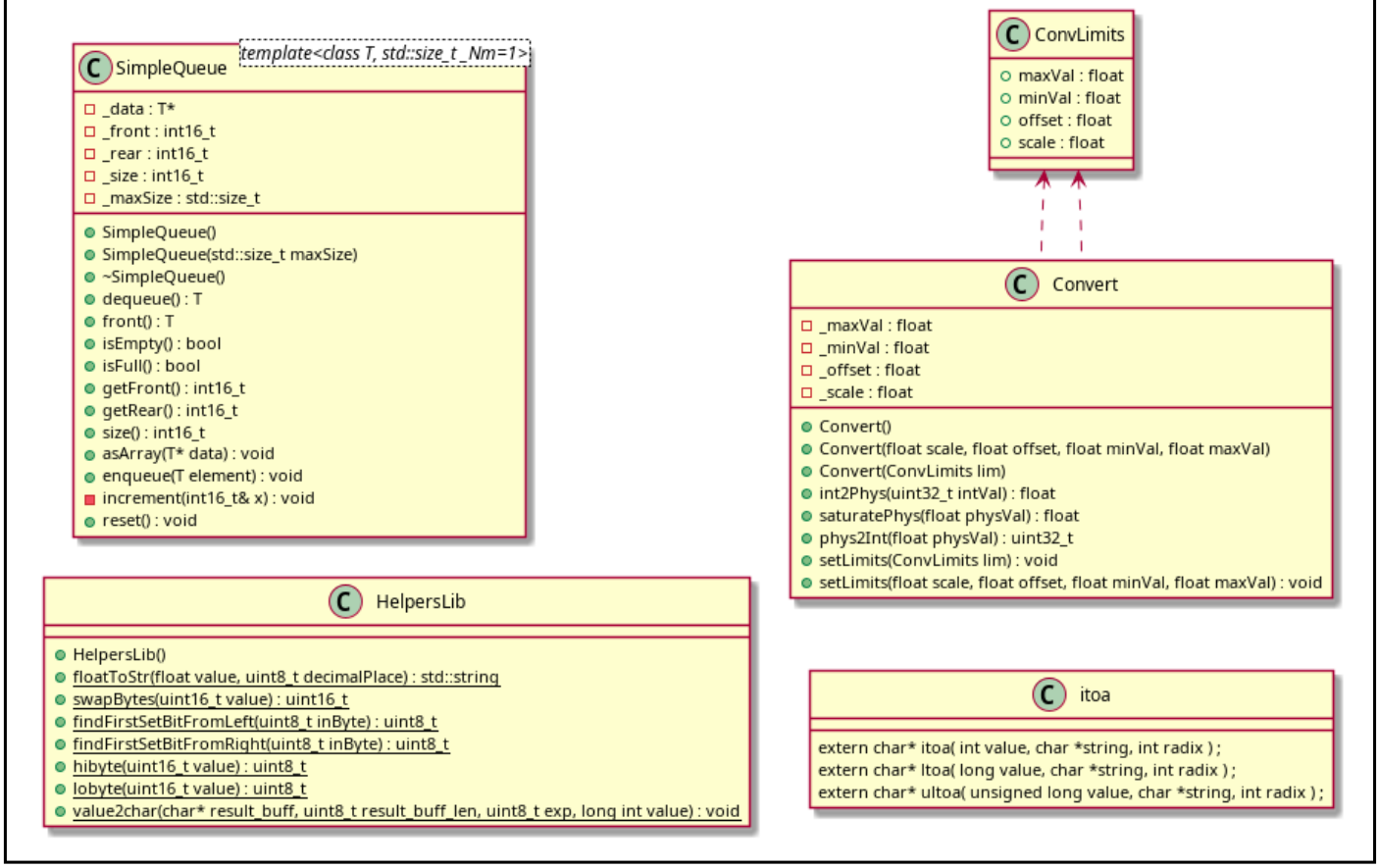


Sockets

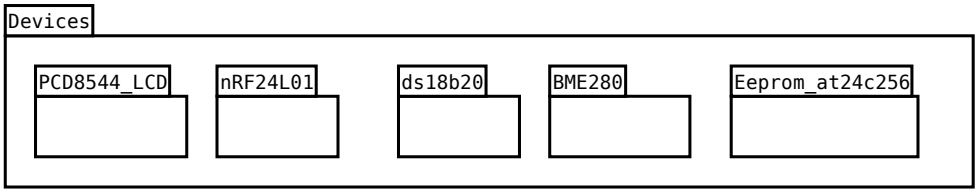



Libraries

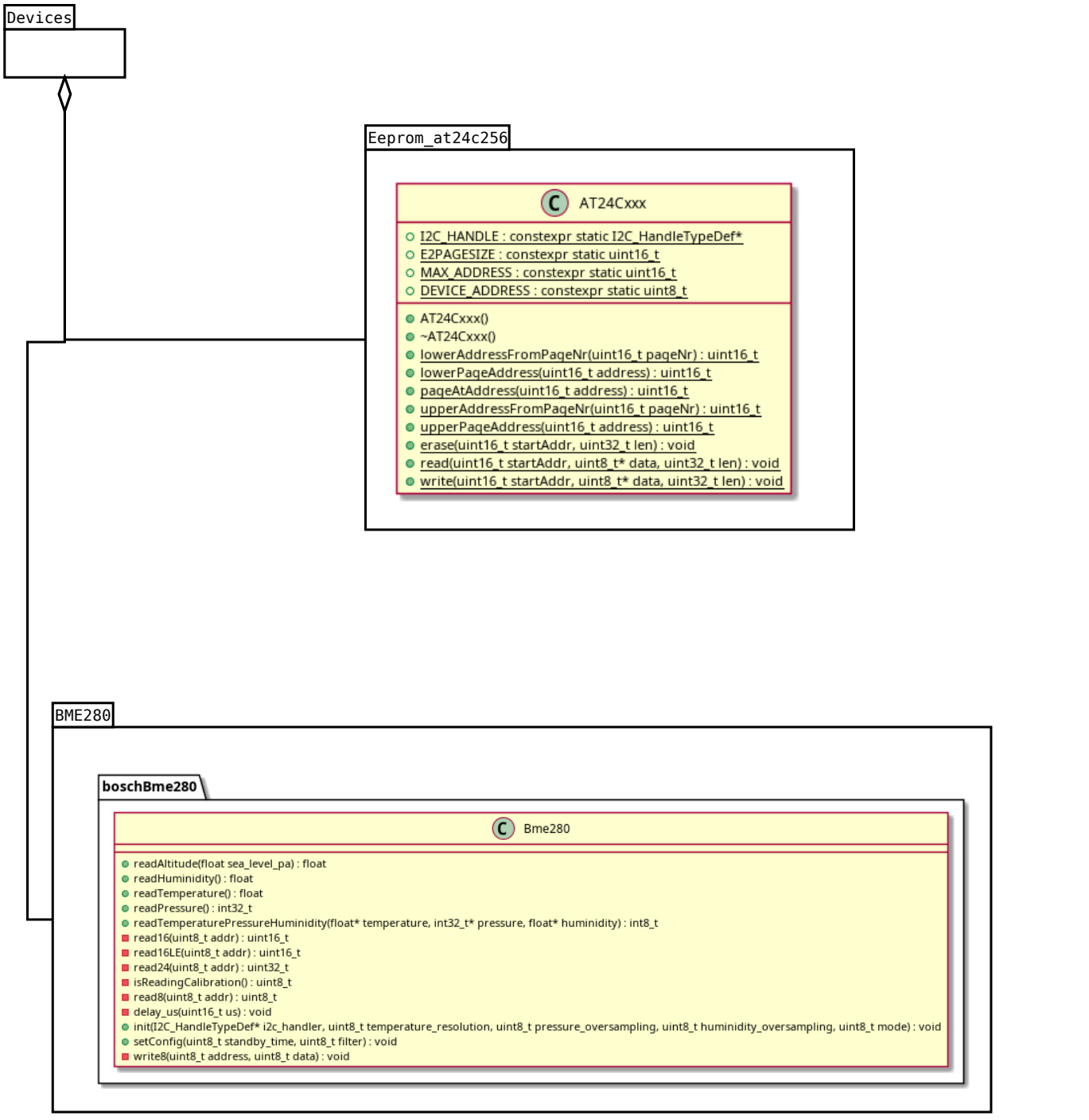


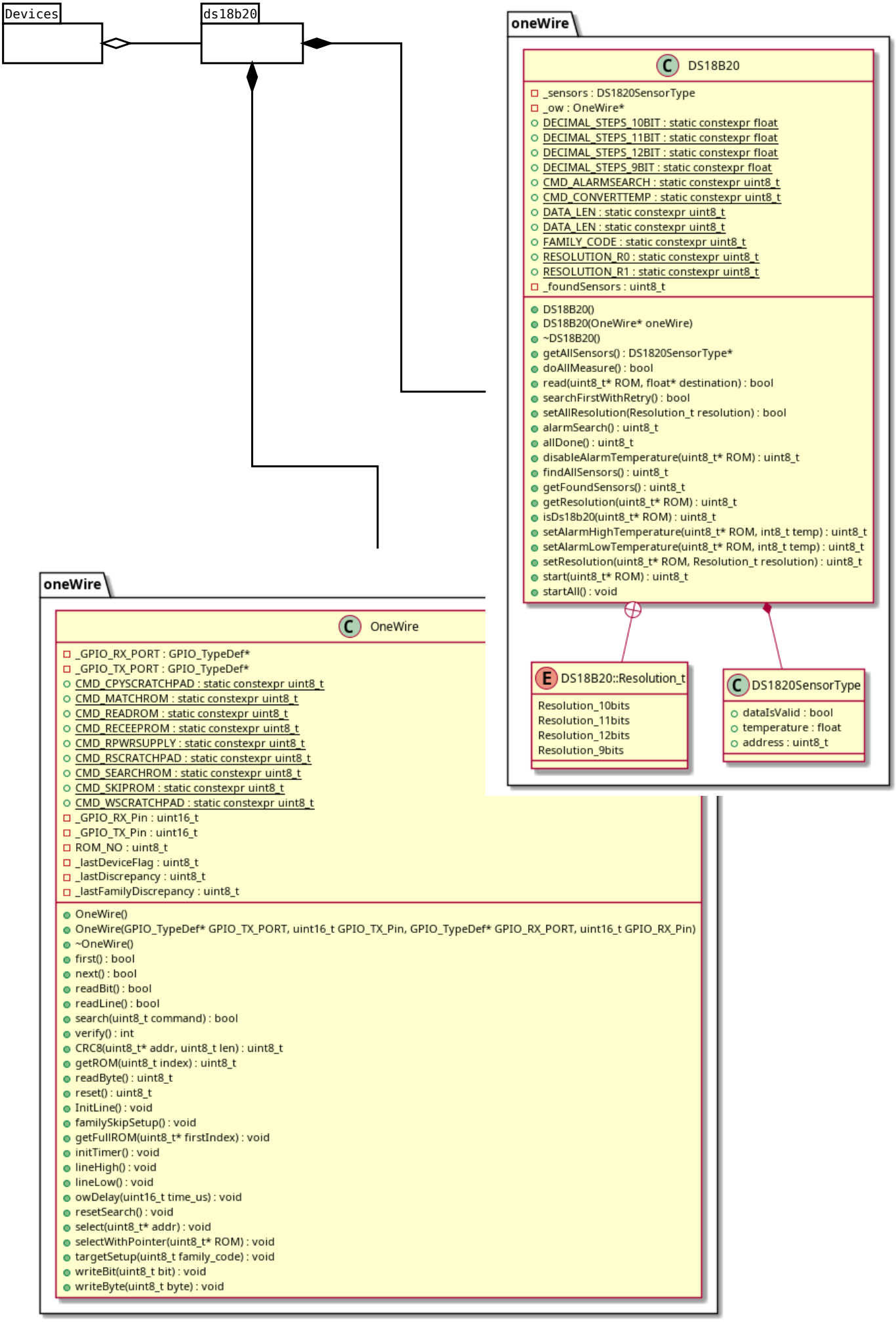
# Devices

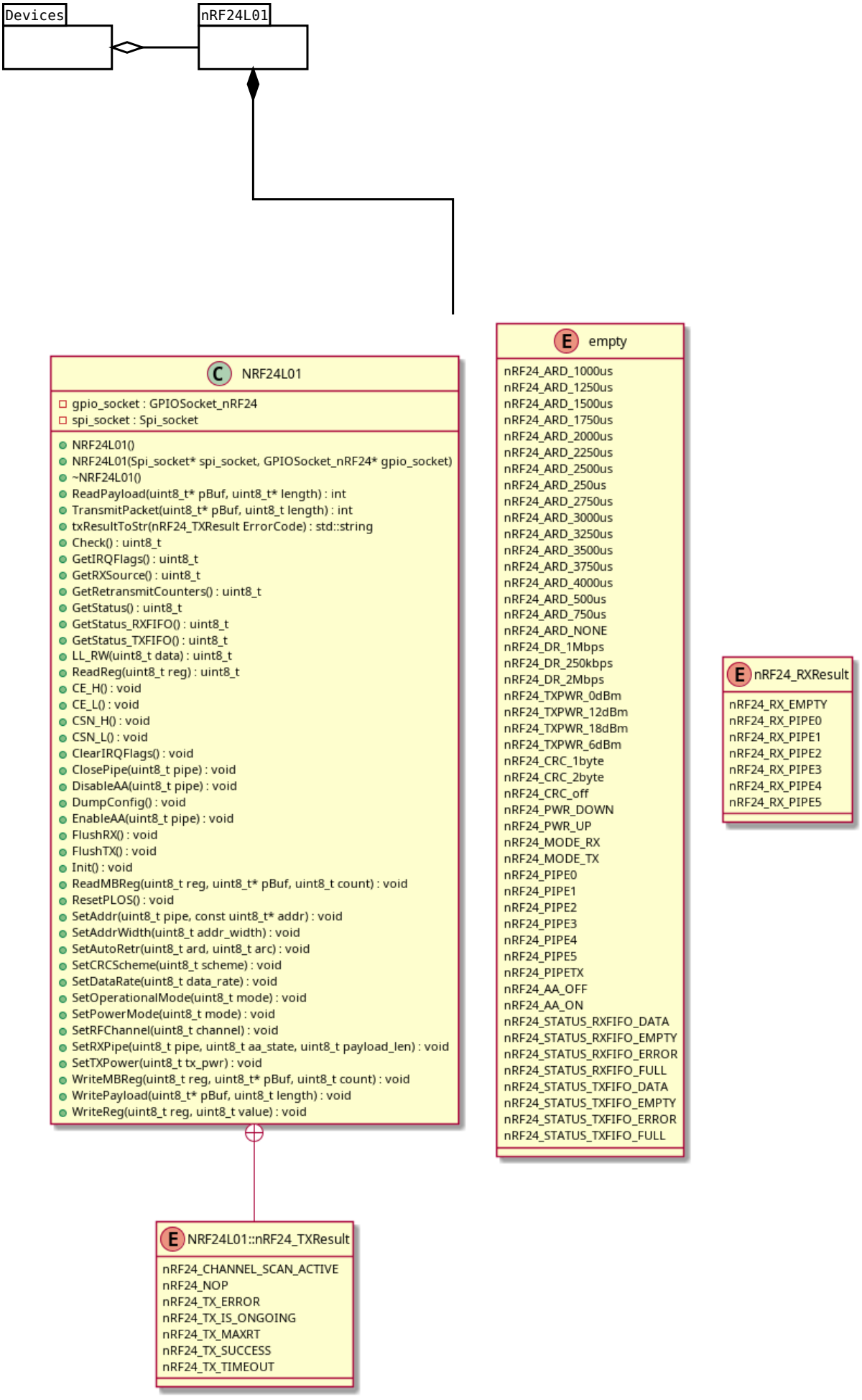
## Overview

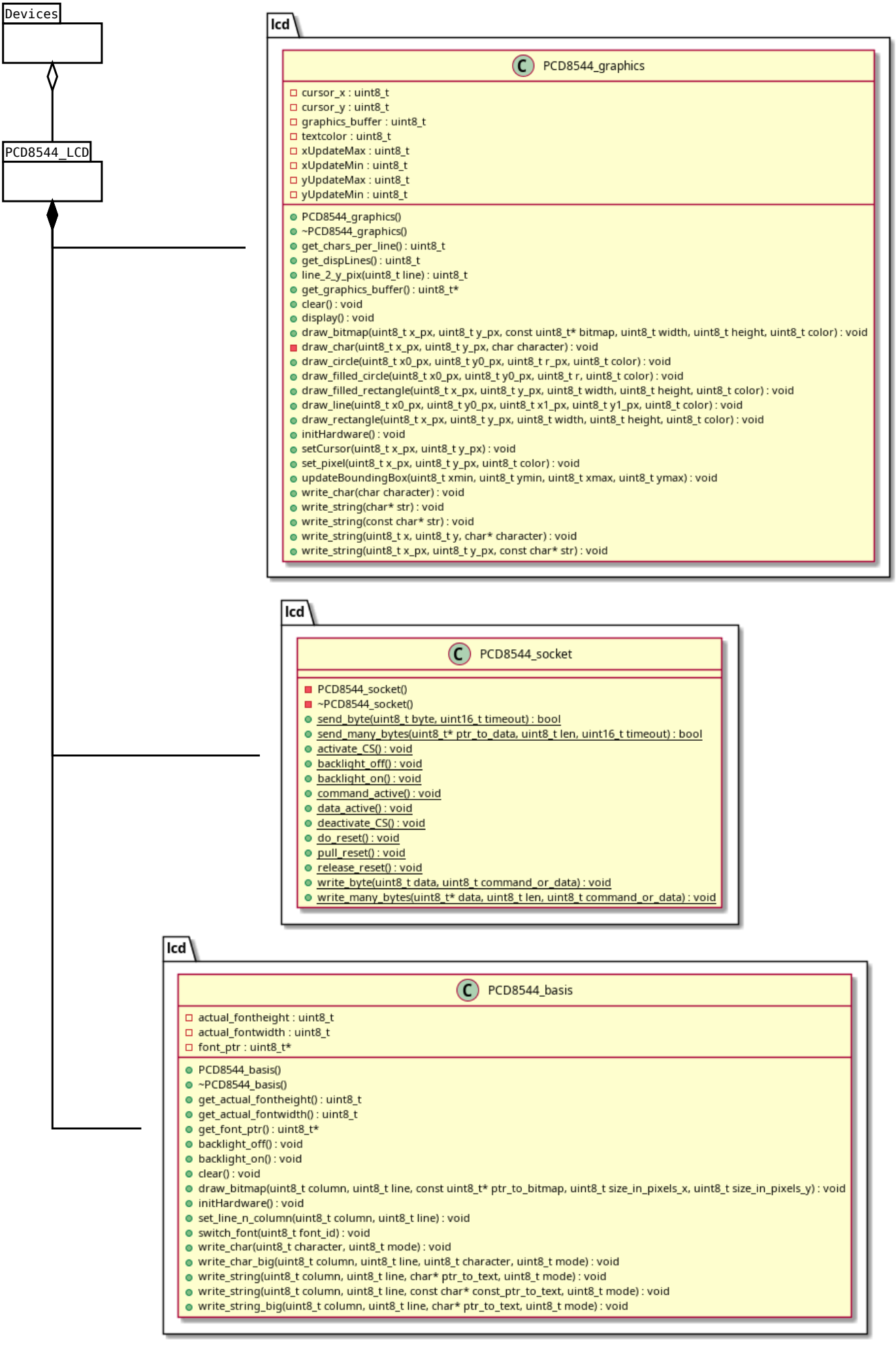



---



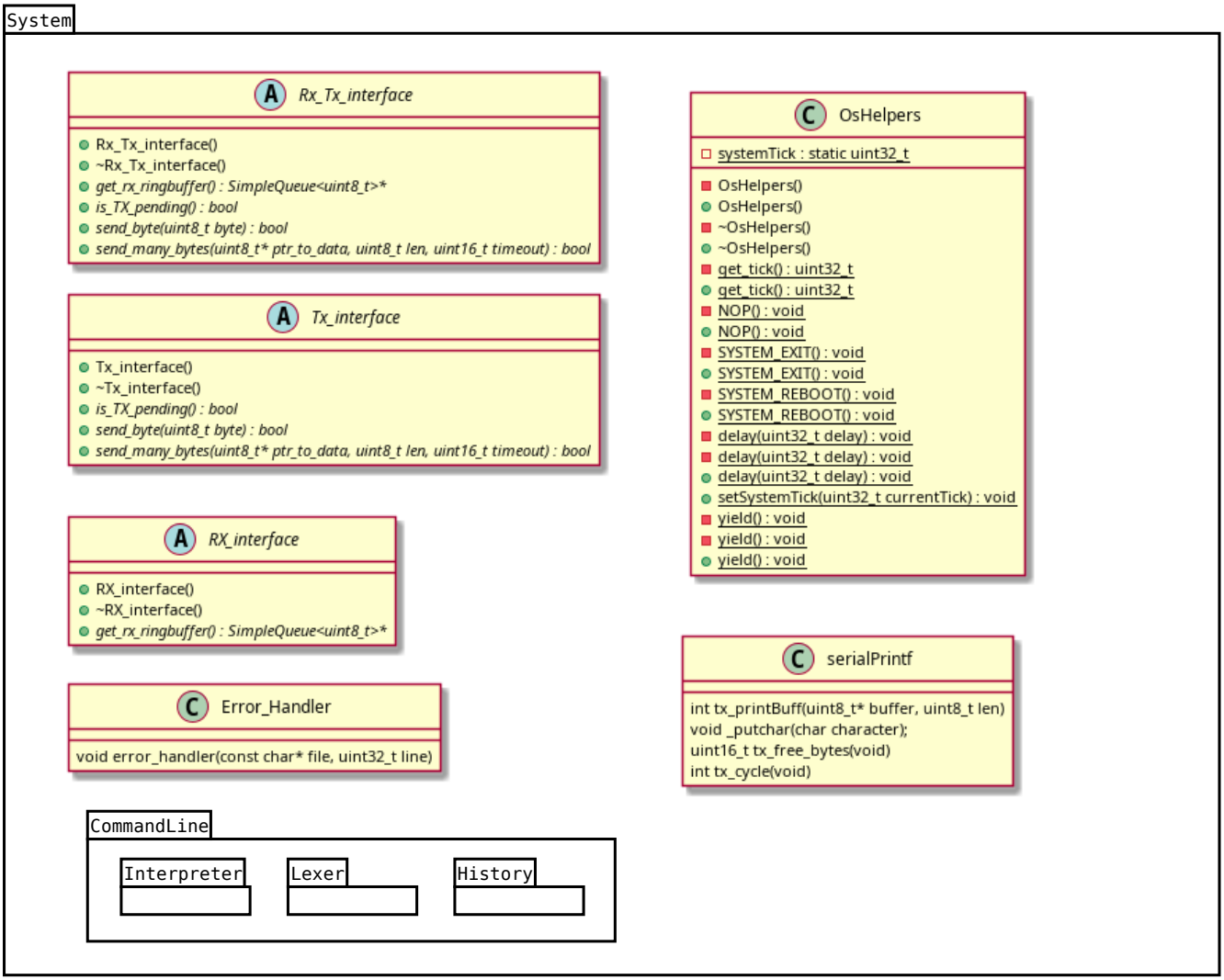

Devices
ds18b20
oneWire
DS18B20
_sensors : DS1820SensorType
_ow : OneWire*
DECIMAL_STEPS_10BIT : static constexpr float
DECIMAL_STEPS_11BIT : static constexpr float
DECIMAL_STEPS_12BIT : static constexpr float
DECIMAL_STEPS_9BIT : static constexpr float
CMD_ALARMSEARCH : static constexpr uint8_t
CMD_CONVERTTEMP : static constexpr uint8_t
DATA_LEN : static constexpr uint8_t
DATA_LEN : static constexpr uint8_t
FAMILY_CODE : static constexpr uint8_t
RESOLUTION_R0 : static constexpr uint8_t
RESOLUTION_R1 : static constexpr uint8_t
_foundSensors : uint8_t
DS18B20()
DS18B20(OneWire* oneWire)
~DS18B20()
getAllSensors() : DS1820SensorType*
doAllMeasure() : bool
read(uint8_t* ROM, float* destination) : bool
searchFirstWithRetry() : bool
setAllResolution(Resolution_t resolution) : bool
alarmSearch() : uint8_t
allDone() : uint8_t
disableAlarmTemperature(uint8_t* ROM) : uint8_t
findAllSensors() : uint8_t
getFoundSensors() : uint8_t
getResolution(uint8_t* ROM) : uint8_t
isDs18b20(uint8_t* ROM) : uint8_t
setAlarmHighTemperature(uint8_t* ROM, int8_t temp) : uint8_t
setAlarmLowTemperature(uint8_t* ROM, int8_t temp) : uint8_t
setResolution(uint8_t* ROM, Resolution_t resolution) : uint8_t
start(uint8_t* ROM) : uint8_t
startAll() : void
DS18B20::Resolution_t
Resolution_10bits
Resolution_11bits
Resolution_12bits
Resolution_9bits
DS1820SensorType
dataIsValid : bool
temperature : float
address : uint8_t
oneWire
OneWire
_GPIO_RX_PORT : GPIO_TypeDef*
_GPIO_TX_PORT : GPIO_TypeDef*
CMD_CPYSCRATCHPAD : static constexpr uint8_t
CMD_MATCHROM : static constexpr uint8_t
CMD_READROM : static constexpr uint8_t
CMD_RECEEPROM : static constexpr uint8_t
CMD_RPWRSUPPLY : static constexpr uint8_t
CMD_RSCRATCHPAD : static constexpr uint8_t
CMD_SEARCHROM : static constexpr uint8_t
CMD_SKIPROM : static constexpr uint8_t
CMD_WSCRATCHPAD : static constexpr uint8_t
_GPIO_RX_Pin : uint16_t
_GPIO_TX_Pin : uint16_t
ROM_NO : uint8_t
_lastDeviceFlag : uint8_t
_lastDiscrepancy : uint8_t
_lastFamilyDiscrepancy : uint8_t
OneWire()
OneWire(GPIO_TypeDef* GPIO_TX_PORT, uint16_t GPIO_TX_Pin, GPIO_TypeDef* GPIO_RX_PORT, uint16_t GPIO_RX_Pin)
~OneWire()
first() : bool
next() : bool
readBit() : bool
readLine() : bool
search(uint8_t command) : bool
verify() : int
CRC8(uint8_t* addr, uint8_t len) : uint8_t
getROM(uint8_t index) : uint8_t
readByte() : uint8_t
reset() : uint8_t
InitLine() : void
familySkipSetup() : void
getFullROM(uint8_t* firstIndex) : void
initTimer() : void
lineHigh() : void
lineLow() : void
owDelay(uint16_t time_us) : void
resetSearch() : void
select(uint8_t* addr) : void
selectWithPointer(uint8_t* ROM) : void
targetSetup(uint8_t family_code) : void
writeBit(uint8_t bit) : void
writeByte(uint8_t byte) : void


Devices
nRF24L01
C NRF24L01
gpio_socket : GPIOSocket_nRF24
spi_socket : Spi_socket
NRF24L01()
NRF24L01(Spi_socket* spi_socket, GPIOSocket_nRF24* gpio_socket)
~NRF24L01()
ReadPayload(uint8_t* pBuf, uint8_t* length) : int
TransmitPacket(uint8_t* pBuf, uint8_t length) : int
txResultToStr(nRF24_TXResult ErrorCode) : std::string
Check() : uint8_t
GetIRQFlags() : uint8_t
GetRXSource() : uint8_t
GetRetransmitCounters() : uint8_t
GetStatus() : uint8_t
GetStatus_RXFIFO() : uint8_t
GetStatus_TXFIFO() : uint8_t
LL_RW(uint8_t data) : uint8_t
ReadReg(uint8_t reg) : uint8_t
CE_H() : void
CE_L() : void
CSN_H() : void
CSN_L() : void
ClearIRQFlags() : void
ClosePipe(uint8_t pipe) : void
DisableAA(uint8_t pipe) : void
DumpConfig() : void
EnableAA(uint8_t pipe) : void
FlushRX() : void
FlushTX() : void
Init() : void
ReadMBReg(uint8_t reg, uint8_t* pBuf, uint8_t count) : void
ResetPLOS() : void
SetAddr(uint8_t pipe, const uint8_t* addr) : void
SetAddrWidth(uint8_t addr_width) : void
SetAutoRetr(uint8_t ard, uint8_t arc) : void
SetCRCScheme(uint8_t scheme) : void
SetDataRate(uint8_t data_rate) : void
SetOperationalMode(uint8_t mode) : void
SetPowerMode(uint8_t mode) : void
SetRFChannel(uint8_t channel) : void
SetRXPipe(uint8_t pipe, uint8_t aa_state, uint8_t payload_len) : void
SetTXPower(uint8_t tx_pwr) : void
WriteMBReg(uint8_t reg, uint8_t* pBuf, uint8_t count) : void
WritePayload(uint8_t* pBuf, uint8_t length) : void
WriteReg(uint8_t reg, uint8_t value) : void
E empty
nRF24_ARD_1000us
nRF24_ARD_1250us
nRF24_ARD_1500us
nRF24_ARD_1750us
nRF24_ARD_2000us
nRF24_ARD_2250us
nRF24_ARD_2500us
nRF24_ARD_250us
nRF24_ARD_2750us
nRF24_ARD_3000us
nRF24_ARD_3250us
nRF24_ARD_3500us
nRF24_ARD_3750us
nRF24_ARD_4000us
nRF24_ARD_500us
nRF24_ARD_750us
nRF24_ARD_NONE
nRF24_DR_1Mbps
nRF24_DR_250kbps
nRF24_DR_2Mbps
nRF24_TXPWR_0dBm
nRF24_TXPWR_12dBm
nRF24_TXPWR_18dBm
nRF24_TXPWR_6dBm
nRF24_CRC_1byte
nRF24_CRC_2byte
nRF24_CRC_off
nRF24_PWR_DOWN
nRF24_PWR_UP
nRF24_MODE_RX
nRF24_MODE_TX
nRF24_PIPE0
nRF24_PIPE1
nRF24_PIPE2
nRF24_PIPE3
nRF24_PIPE4
nRF24_PIPE5
nRF24_PIPETX
nRF24_AA_OFF
nRF24_AA_ON
nRF24_STATUS_RXFIFO_DATA
nRF24_STATUS_RXFIFO_EMPTY
nRF24_STATUS_RXFIFO_ERROR
nRF24_STATUS_RXFIFO_FULL
nRF24_STATUS_TXFIFO_DATA
nRF24_STATUS_TXFIFO_EMPTY
nRF24_STATUS_TXFIFO_ERROR
nRF24_STATUS_TXFIFO_FULL
E nRF24_RXResult
nRF24_RX_EMPTY
nRF24_RX_PIPE0
nRF24_RX_PIPE1
nRF24_RX_PIPE2
nRF24_RX_PIPE3
nRF24_RX_PIPE4
nRF24_RX_PIPE5
E NRF24L01::nRF24_TXResult
nRF24_CHANNEL_SCAN_ACTIVE
nRF24_NOP
nRF24_TX_ERROR
nRF24_TX_IS_ONGOING
nRF24_TX_MAXRT
nRF24_TX_SUCCESS
nRF24_TX_TIMEOUT

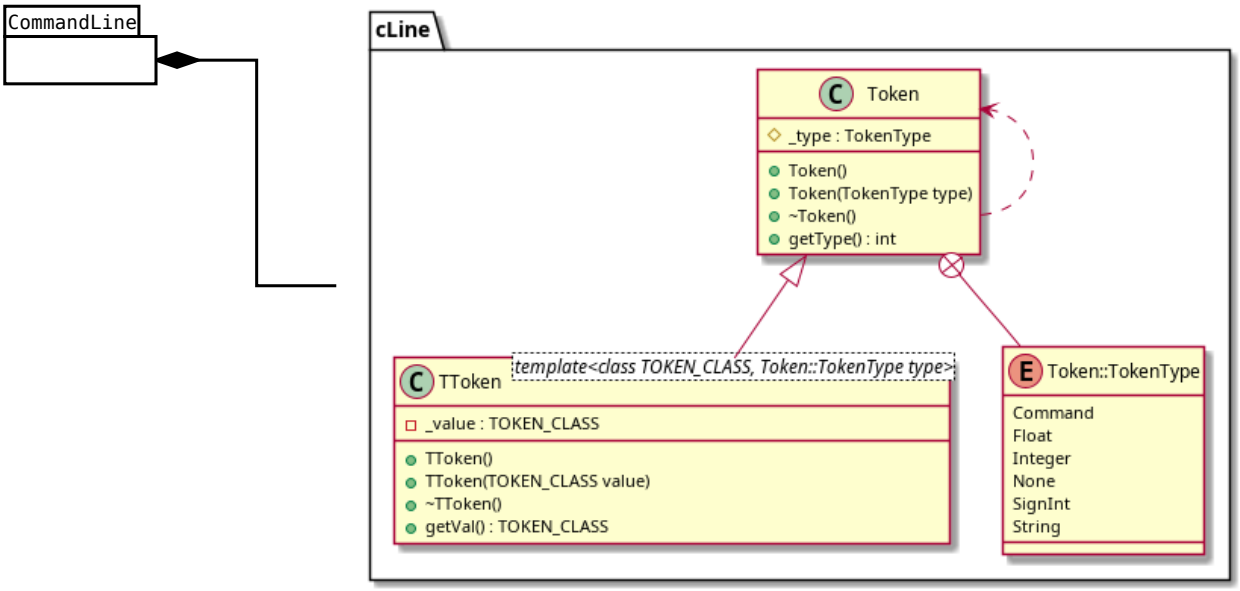

Devices
PCD8544_LCD
lcd
PCD8544_graphics
cursor_x : uint8_t
cursor_y : uint8_t
graphics_buffer : uint8_t
textcolor : uint8_t
xUpdateMax : uint8_t
xUpdateMin : uint8_t
yUpdateMax : uint8_t
yUpdateMin : uint8_t
PCD8544_graphics()
~PCD8544_graphics()
get_chars_per_line() : uint8_t
get_dispLines() : uint8_t
line_2_y_pix(uint8_t line) : uint8_t
get_graphics_buffer() : uint8_t*
clear() : void
display() : void
draw_bitmap(uint8_t x_px, uint8_t y_px, const uint8_t* bitmap, uint8_t width, uint8_t height, uint8_t color) : void
draw_char(uint8_t x_px, uint8_t y_px, char character) : void
draw_circle(uint8_t x0_px, uint8_t y0_px, uint8_t r_px, uint8_t color) : void
draw_filled_circle(uint8_t x0_px, uint8_t y0_px, uint8_t r, uint8_t color) : void
draw_filled_rectangle(uint8_t x_px, uint8_t y_px, uint8_t width, uint8_t height, uint8_t color) : void
draw_line(uint8_t x0_px, uint8_t y0_px, uint8_t x1_px, uint8_t y1_px, uint8_t color) : void
draw_rectangle(uint8_t x_px, uint8_t y_px, uint8_t width, uint8_t height, uint8_t color) : void
initHardware() : void
setCursor(uint8_t x_px, uint8_t y_px) : void
set_pixel(uint8_t x_px, uint8_t y_px, uint8_t color) : void
updateBoundingBox(uint8_t xmin, uint8_t ymin, uint8_t xmax, uint8_t ymax) : void
write_char(char character) : void
write_string(char* str) : void
write_string(const char* str) : void
write_string(uint8_t x, uint8_t y, char* character) : void
write_string(uint8_t x_px, uint8_t y_px, const char* str) : void
lcd
PCD8544_socket
PCD8544_socket()
~PCD8544_socket()
send_byte(uint8_t byte, uint16_t timeout) : bool
send_many_bytes(uint8_t* ptr_to_data, uint8_t len, uint16_t timeout) : bool
activate_CS() : void
backlight_off() : void
backlight_on() : void
command_active() : void
data_active() : void
deactivate_CS() : void
do_reset() : void
pull_reset() : void
release_reset() : void
write_byte(uint8_t data, uint8_t command_or_data) : void
write_many_bytes(uint8_t* data, uint8_t len, uint8_t command_or_data) : void
lcd
PCD8544_basis
actual_fontheight : uint8_t
actual_fontwidth : uint8_t
font_ptr : uint8_t*
PCD8544_basis()
~PCD8544_basis()
get_actual_fontheight() : uint8_t
get_actual_fontwidth() : uint8_t
get_font_ptr() : uint8_t*
backlight_off() : void
backlight_on() : void
clear() : void
draw_bitmap(uint8_t column, uint8_t line, const uint8_t* ptr_to_bitmap, uint8_t size_in_pixels_x, uint8_t size_in_pixels_y) : void
initHardware() : void
set_line_n_column(uint8_t column, uint8_t line) : void
switch_font(uint8_t font_id) : void
write_char(uint8_t character, uint8_t mode) : void
write_char_big(uint8_t column, uint8_t line, uint8_t character, uint8_t mode) : void
write_string(uint8_t column, uint8_t line, char* ptr_to_text, uint8_t mode) : void
write_string(uint8_t column, uint8_t line, const char* const_ptr_to_text, uint8_t mode) : void
write_string_big(uint8_t column, uint8_t line, char* ptr_to_text, uint8_t mode) : void

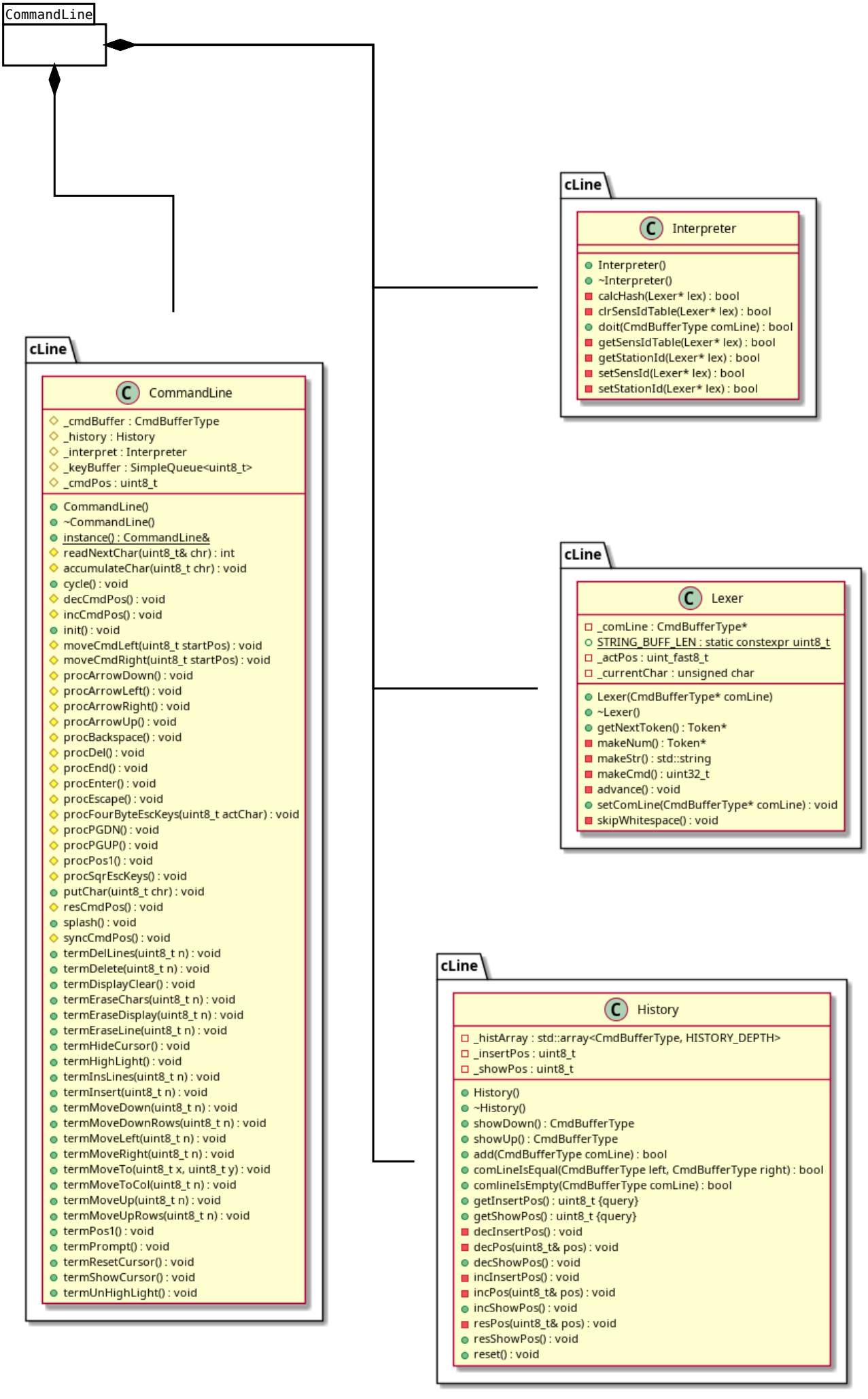
# System

## Overview



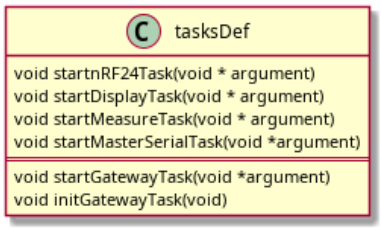

---

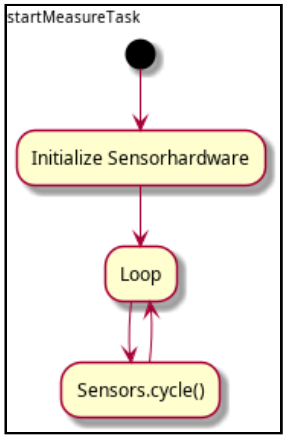


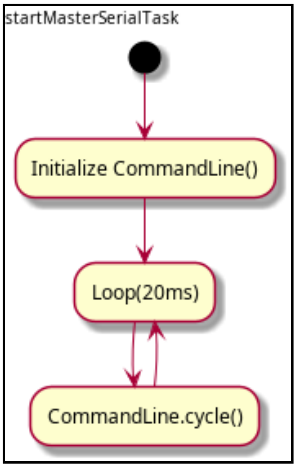

CommandLine
cLine
Interpreter
Interpreter()
~Interpreter()
calcHash(Lexer* lex) : bool
clrSensIdTable(Lexer* lex) : bool
doit(CmdBufferType comLine) : bool
getSensIdTable(Lexer* lex) : bool
getStationId(Lexer* lex) : bool
setSensId(Lexer* lex) : bool
setStationId(Lexer* lex) : bool
cLine
CommandLine
_cmdBuffer : CmdBufferType
_history : History
_interpret : Interpreter
_keyBuffer : SimpleQueue<uint8_t>
_cmdPos : uint8_t
CommandLine()
~CommandLine()
instance() : CommandLine&
readNextChar(uint8_t& chr) : int
accumulateChar(uint8_t chr) : void
cycle() : void
decCmdPos() : void
incCmdPos() : void
init() : void
moveCmdLeft(uint8_t startPos) : void
moveCmdRight(uint8_t startPos) : void
procArrowDown() : void
procArrowLeft() : void
procArrowRight() : void
procArrowUp() : void
procBackspace() : void
procDel() : void
procEnd() : void
procEnter() : void
procEscape() : void
procFourByteEscKeys(uint8_t actChar) : void
procPGDN() : void
procPGUP() : void
procPos1() : void
procSqrEscKeys() : void
putChar(uint8_t chr) : void
resCmdPos() : void
splash() : void
syncCmdPos() : void
termDelLines(uint8_t n) : void
termDelete(uint8_t n) : void
termDisplayClear() : void
termEraseChars(uint8_t n) : void
termEraseDisplay(uint8_t n) : void
termEraseLine(uint8_t n) : void
termHideCursor() : void
termHighLight() : void
termInsLines(uint8_t n) : void
termInsert(uint8_t n) : void
termMoveDown(uint8_t n) : void
termMoveDownRows(uint8_t n) : void
termMoveLeft(uint8_t n) : void
termMoveRight(uint8_t n) : void
termMoveTo(uint8_t x, uint8_t y) : void
termMoveToCol(uint8_t n) : void
termMoveUp(uint8_t n) : void
termMoveUpRows(uint8_t n) : void
termPos1() : void
termPrompt() : void
termResetCursor() : void
termShowCursor() : void
termUnHighLight() : void
cLine
Lexer
_comLine : CmdBufferType*
STRING_BUFF_LEN : static constexpr uint8_t
_actPos : uint_fast8_t
_currentChar : unsigned char
Lexer(CmdBufferType* comLine)
~Lexer()
getNextToken() : Token*
makeNum() : Token*
makeStr() : std::string
makeCmd() : uint32_t
advance() : void
setComLine(CmdBufferType* comLine) : void
skipWhitespace() : void
cLine
History
_histArray : std::array<CmdBufferType, HISTORY_DEPTH>
_insertPos : uint8_t
_showPos : uint8_t
History()
~History()
showDown() : CmdBufferType
showUp() : CmdBufferType
add(CmdBufferType comLine) : bool
comLineIsEqual(CmdBufferType left, CmdBufferType right) : bool
comlineIsEmpty(CmdBufferType comLine) : bool
getInsertPos() : uint8_t {query}
getShowPos() : uint8_t {query}
decInsertPos() : void
decPos(uint8_t& pos) : void
decShowPos() : void
incInsertPos() : void
incPos(uint8_t& pos) : void
incShowPos() : void
resPos(uint8_t& pos) : void
resShowPos() : void
reset() : void

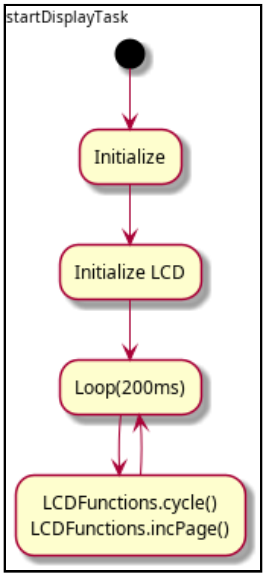
# Tasks

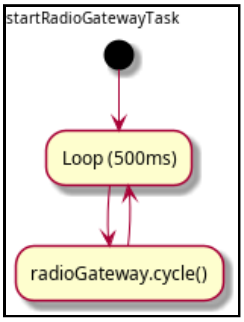
Tasks

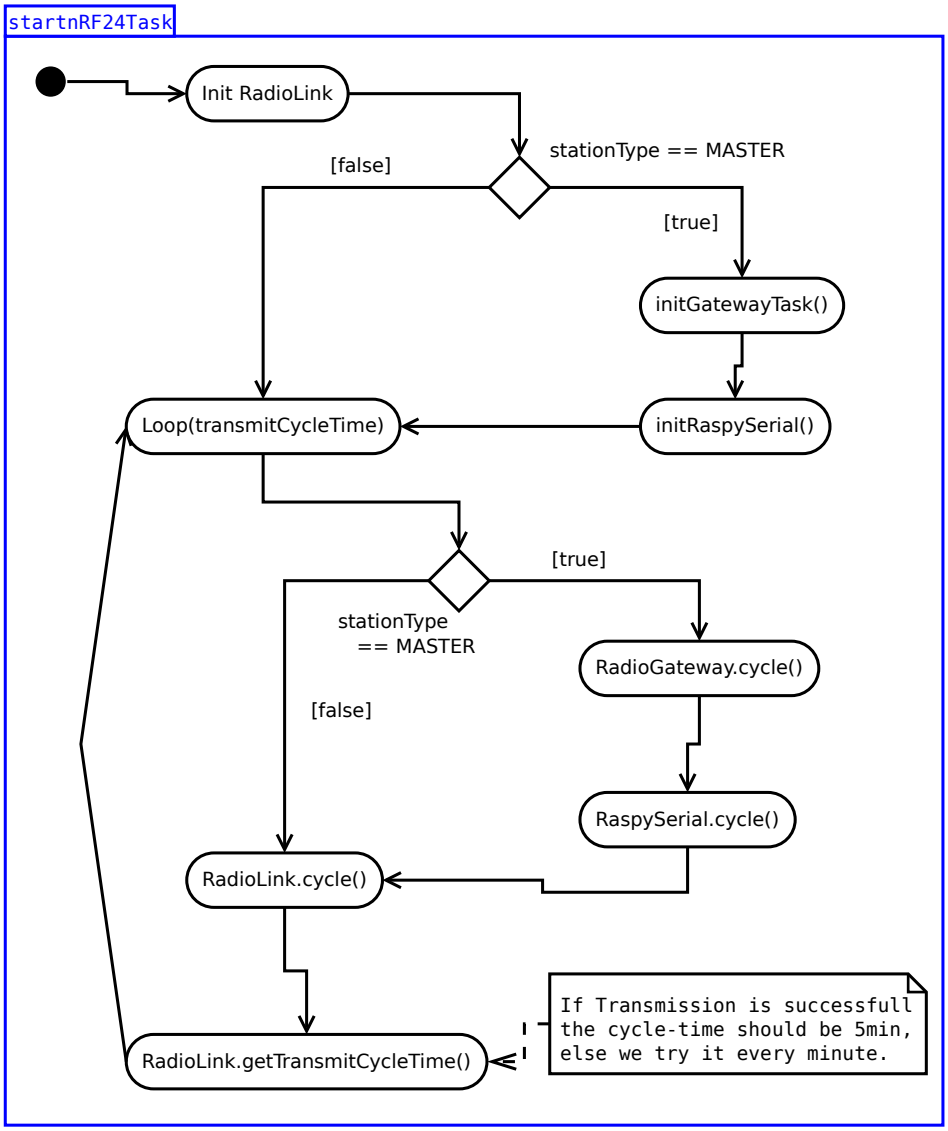

tasksDef
void startnRF24Task(void * argument)
void startDisplayTask(void * argument)
void startMeasureTask(void * argument)
void startMasterSerialTask(void *argument)
void startGatewayTask(void *argument)
void initGatewayTask(void)


startMeasureTask
Initialize Sensorhardware
Loop
Sensors.cycle()


startMasterSerialTask
Initialize CommandLine()
Loop(20ms)
CommandLine.cycle()

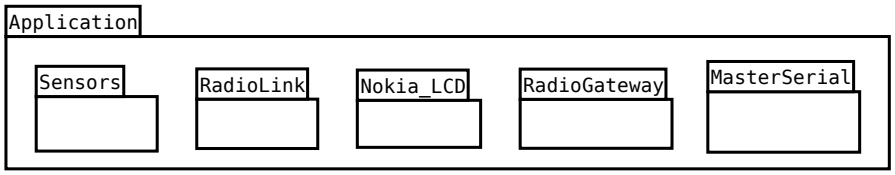

startDisplayTask
Initialize
Initialize LCD
Loop(200ms)
LCDFunctions.cycle()
LCDFunctions.incPage()


startRadioGatewayTask
Loop (500ms)
radioGateway.cycle()


startnRF24Task
Init RadioLink
stationType == MASTER
[false]
[true]
initGatewayTask()
initRaspySerial()
Loop(transmitCycleTime)
[true]
stationType
== MASTER
[false]
RadioGateway.cycle()
RaspySerial.cycle()
RadioLink.cycle()
RadioLink.getTransmitCycleTime()
If Transmission is successfull
the cycle-time should be 5min,
else we try it every minute.

# Application

Overview



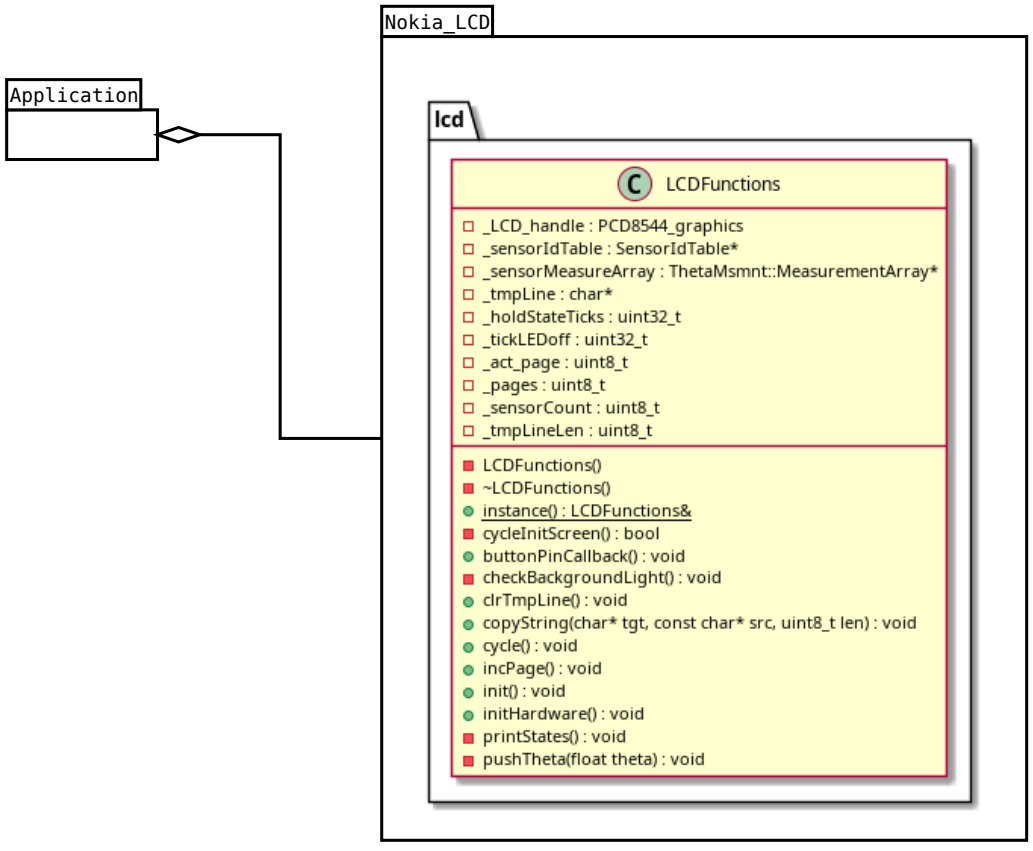

---

# Application / Nokia_LCD

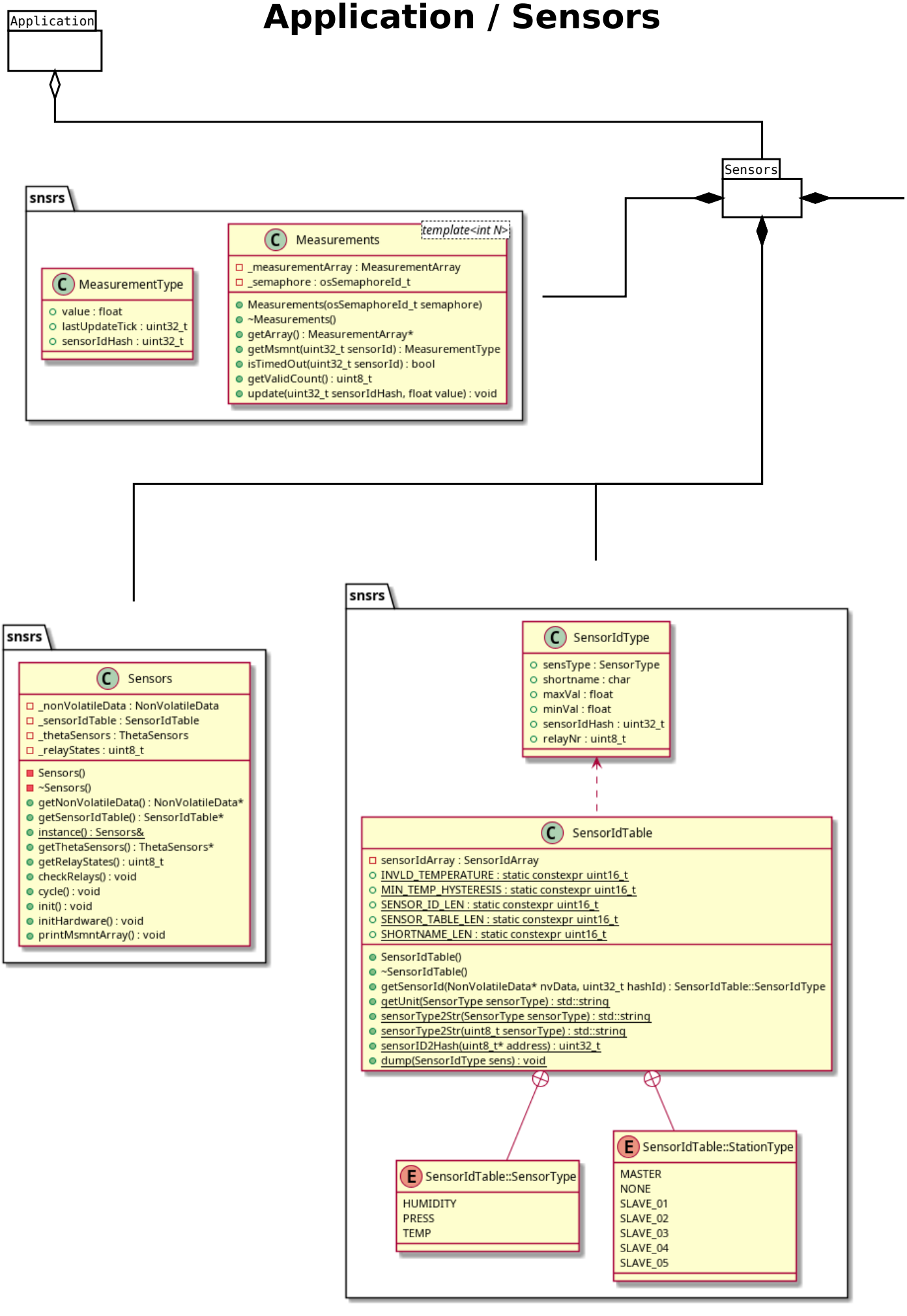
LCD 80x48 (w x l)
FONT_5x8: 16 chars, 6 lines

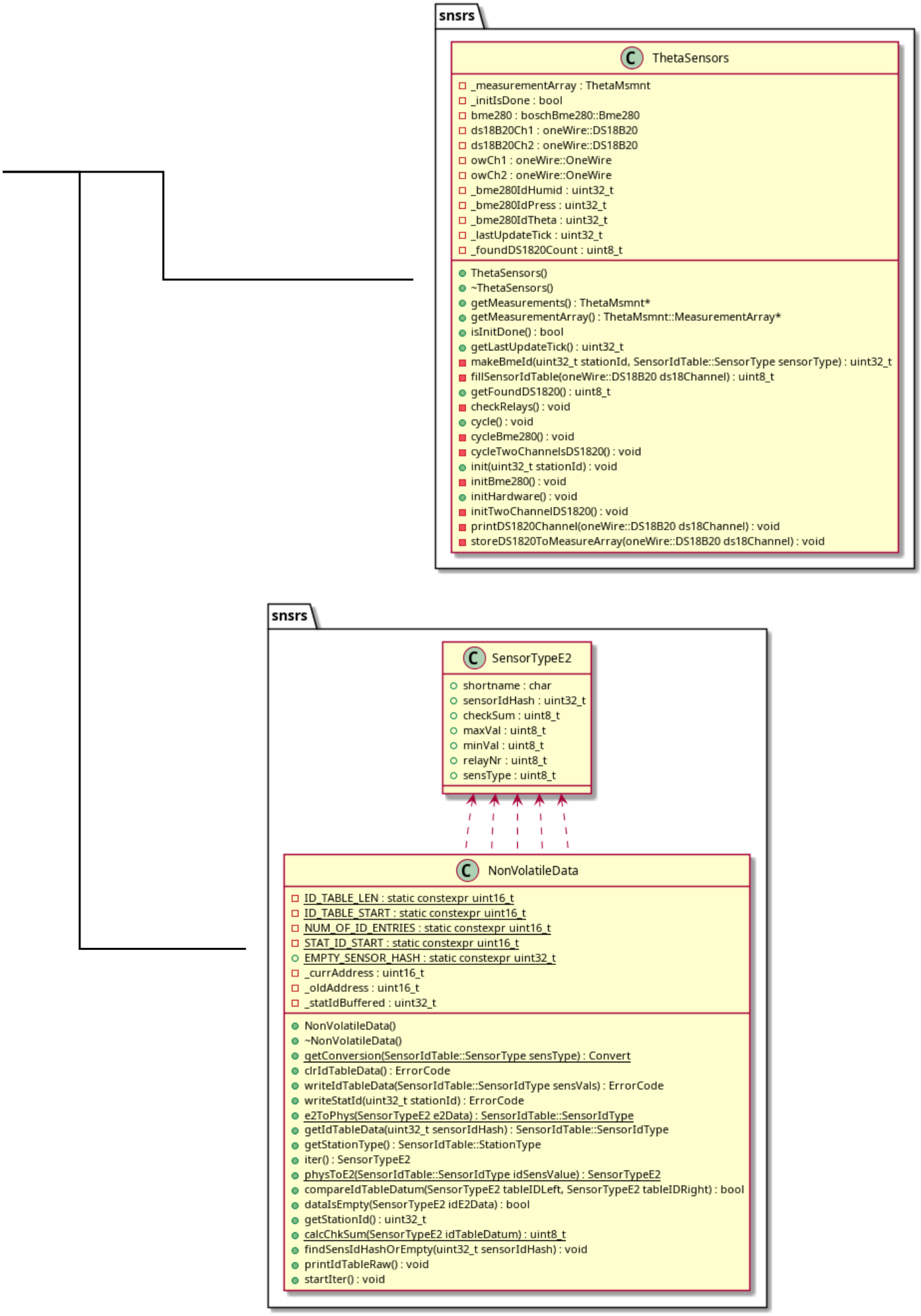
```
..0123456789012345   ..0123456789012345
0 InnenTmp -12.5     0 Station FF1234FF
1 InnenHum  63.5     1 lastUpdt  2123.4
2 InnenPrs  1024     2 lostPkgs       25
3 LagerTmp -12.5     3 rxOverFlows    25
4 LagerHum  63.5     4 valSensors     28
5 LagerPrs  1024     5 relayState      2
```

ButtonPress cycles screen,
longPress switches between statistics and measurement.

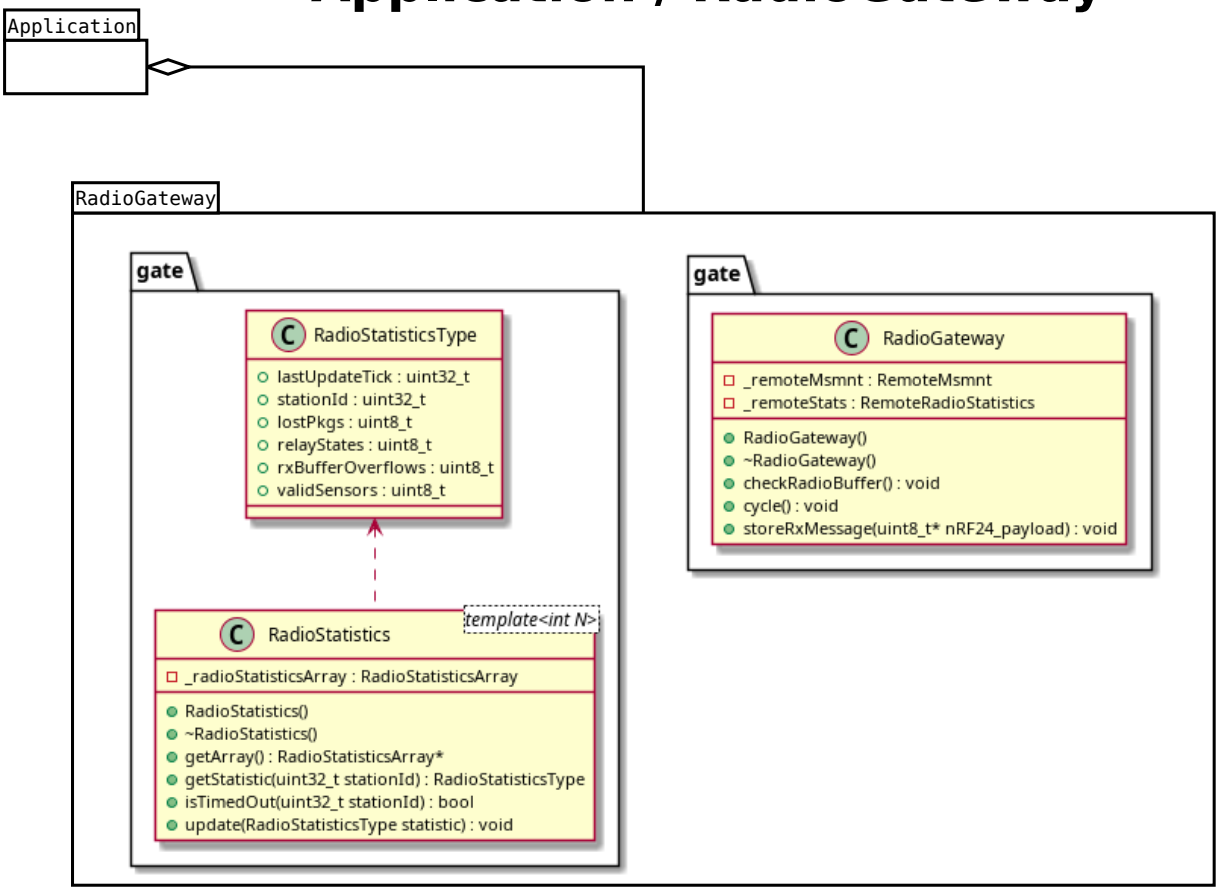


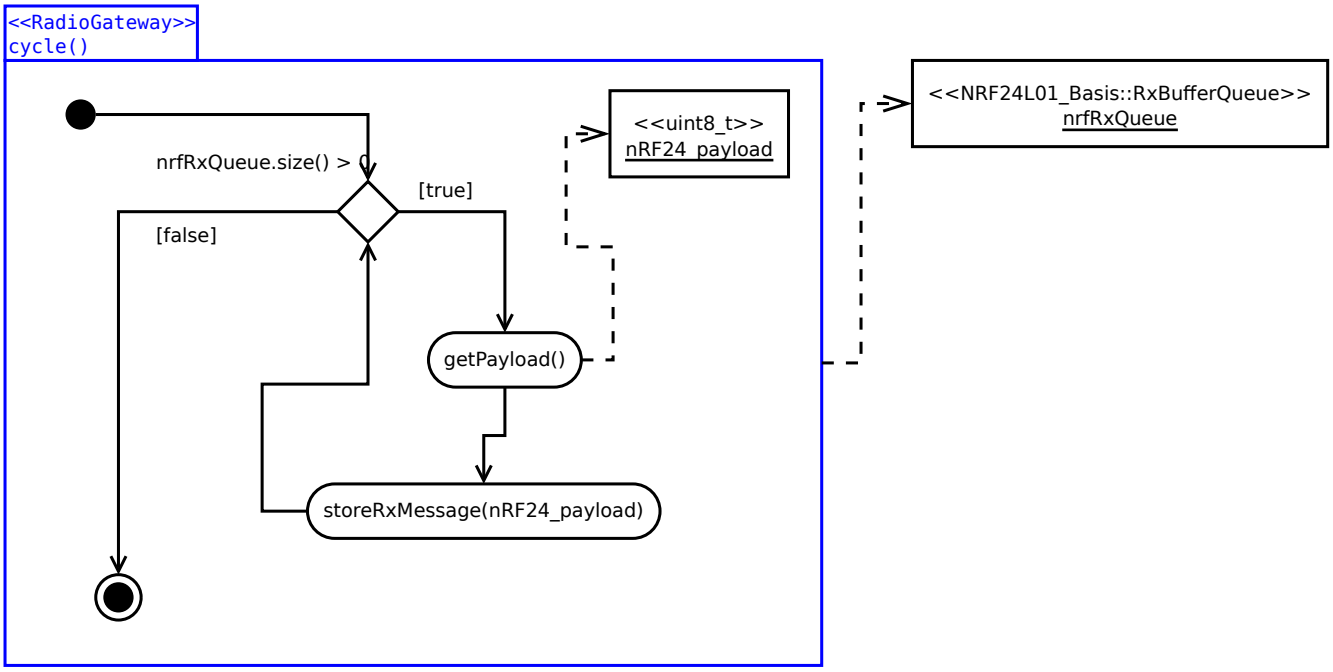
# Application / Sensors

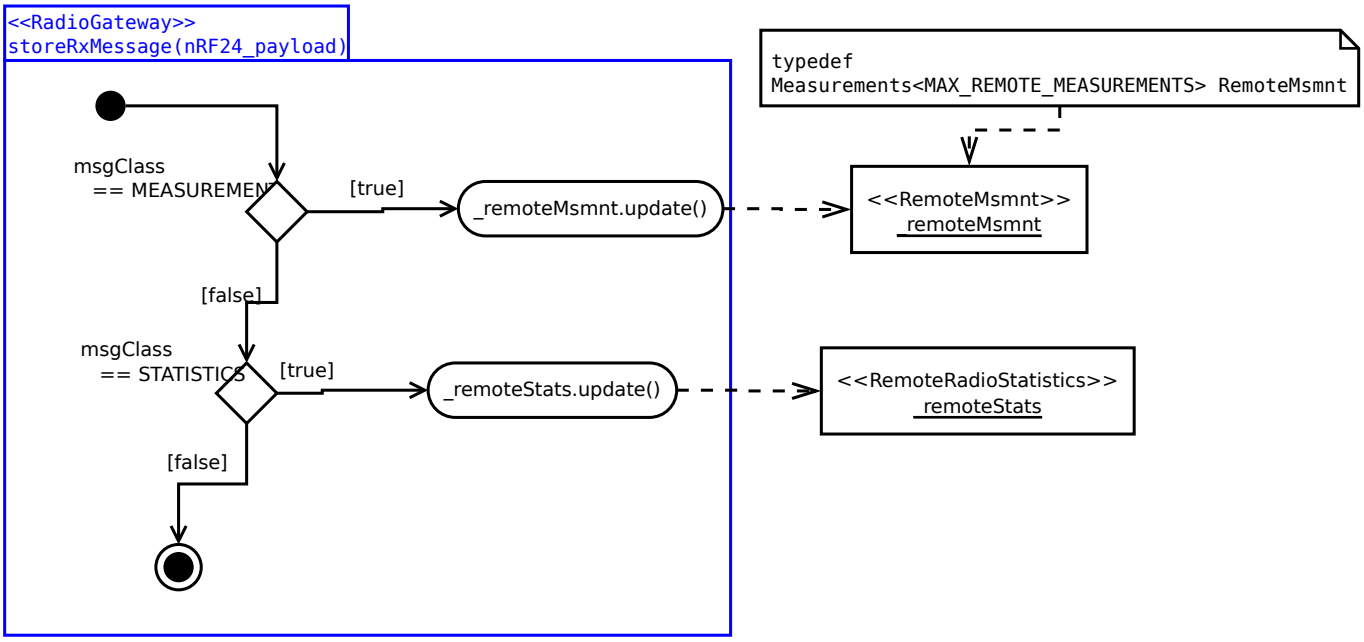



snsrs
ThetaSensors
_measurementArray : ThetaMsmnt
_initIsDone : bool
bme280 : boschBme280::Bme280
ds18B20Ch1 : oneWire::DS18B20
ds18B20Ch2 : oneWire::DS18B20
owCh1 : oneWire::OneWire
owCh2 : oneWire::OneWire
_bme280IdHumid : uint32_t
_bme280IdPress : uint32_t
_bme280IdTheta : uint32_t
_lastUpdateTick : uint32_t
_foundDS1820Count : uint8_t
ThetaSensors()
~ThetaSensors()
getMeasurements() : ThetaMsmnt*
getMeasurementArray() : ThetaMsmnt::MeasurementArray*
isInitDone() : bool
getLastUpdateTick() : uint32_t
makeBmeId(uint32_t stationId, SensorIdTable::SensorType sensorType) : uint32_t
fillSensorIdTable(oneWire::DS18B20 ds18Channel) : uint8_t
getFoundDS1820() : uint8_t
checkRelays() : void
cycle() : void
cycleBme280() : void
cycleTwoChannelsDS1820() : void
init(uint32_t stationId) : void
initBme280() : void
initHardware() : void
initTwoChannelDS1820() : void
printDS1820Channel(oneWire::DS18B20 ds18Channel) : void
storeDS1820ToMeasureArray(oneWire::DS18B20 ds18Channel) : void
snsrs
SensorTypeE2
shortname : char
sensorIdHash : uint32_t
checkSum : uint8_t
maxVal : uint8_t
minVal : uint8_t
relayNr : uint8_t
sensType : uint8_t
NonVolatileData
ID_TABLE_LEN : static constexpr uint16_t
ID_TABLE_START : static constexpr uint16_t
NUM_OF_ID_ENTRIES : static constexpr uint16_t
STAT_ID_START : static constexpr uint16_t
EMPTY_SENSOR_HASH : static constexpr uint32_t
_currAddress : uint16_t
_oldAddress : uint16_t
_statIdBuffered : uint32_t
NonVolatileData()
~NonVolatileData()
getConversion(SensorIdTable::SensorType sensType) : Convert
clrIdTableData() : ErrorCode
writeIdTableData(SensorIdTable::SensorIdType sensVals) : ErrorCode
writeStatId(uint32_t stationId) : ErrorCode
e2ToPhys(SensorTypeE2 e2Data) : SensorIdTable::SensorIdType
getIdTableData(uint32_t sensorIdHash) : SensorIdTable::SensorIdType
getStationType() : SensorIdTable::StationType
iter() : SensorTypeE2
physToE2(SensorIdTable::SensorIdType idSensValue) : SensorTypeE2
compareIdTableDatum(SensorTypeE2 tableIDLeft, SensorTypeE2 tableIDRight) : bool
dataIsEmpty(SensorTypeE2 idE2Data) : bool
getStationId() : uint32_t
calcChkSum(SensorTypeE2 idTableDatum) : uint8_t
findSensIdHashOrEmpty(uint32_t sensorIdHash) : void
printIdTableRaw() : void
startIter() : void

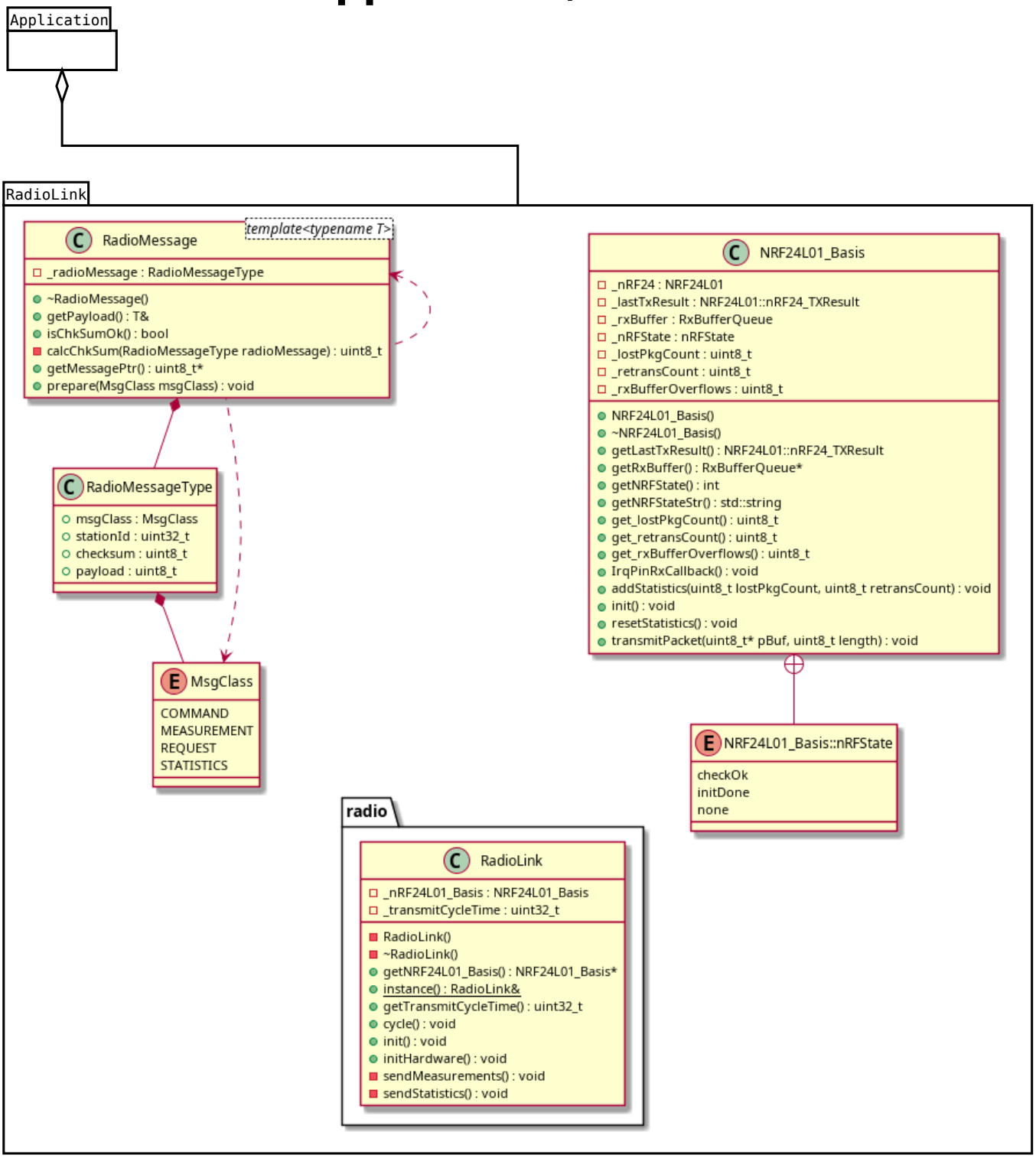
# Application / RadioGateway



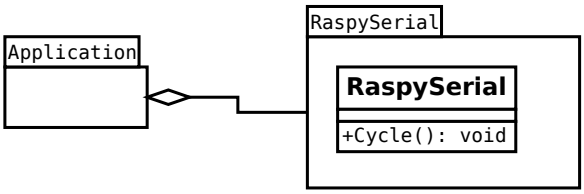



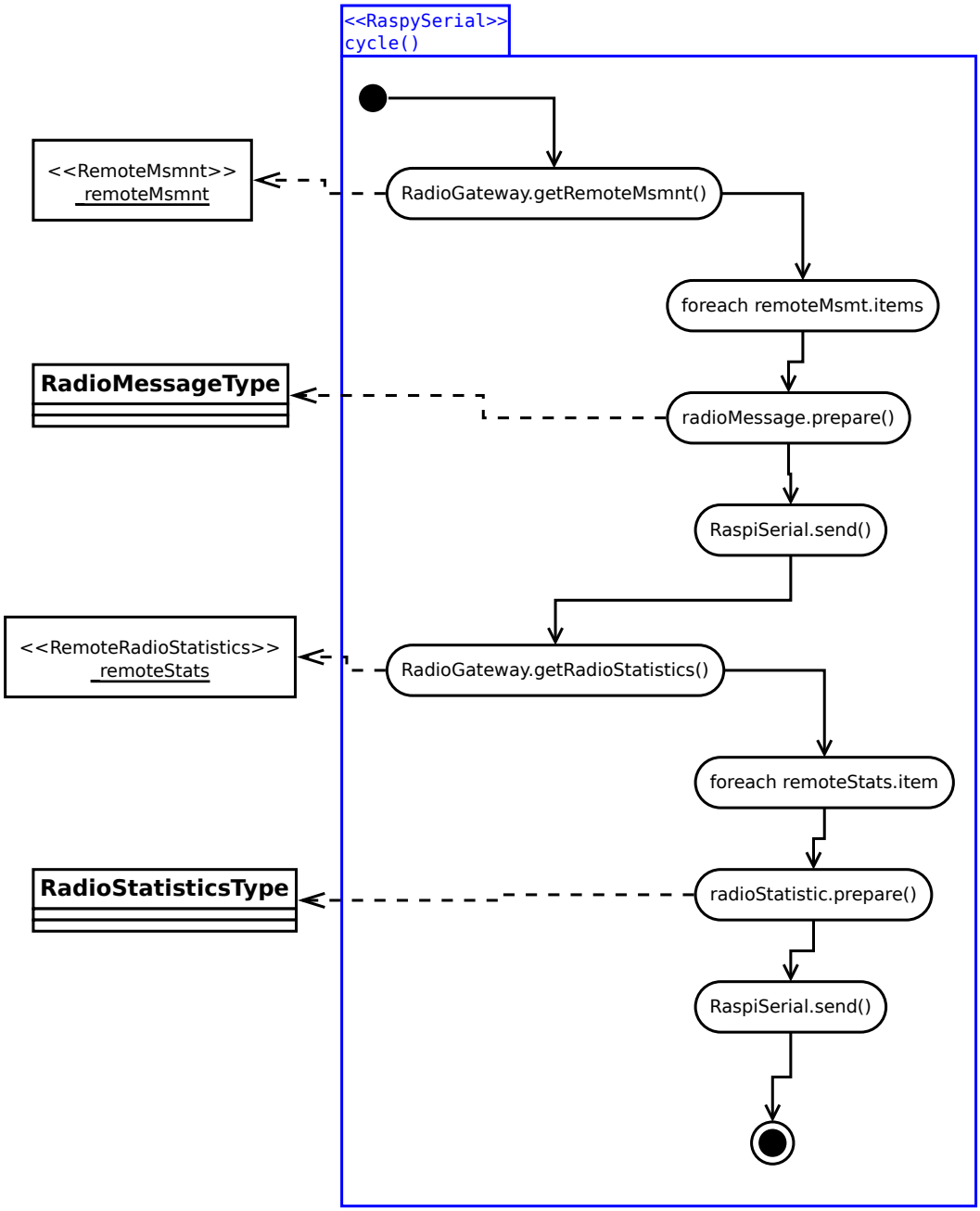

<<RadioGateway>>
storeRxMessage(nRF24_payload)
msgClass
== MEASUREMENT
[true]
_remoteMsmnt.update()
[false]
msgClass
== STATISTICS
[true]
_remoteStats.update()
[false]
typedef
Measurements<MAX_REMOTE_MEASUREMENTS> RemoteMsmnt
<<RemoteMsmnt>>
remoteMsmnt
<<RemoteRadioStatistics>>
remoteStats

# Application / RadioLink



# Application / RaspySerial



# Message-Flow