# Diagram Documentation

**BDD Codesign** - *(SysML Block Definition diagram)*

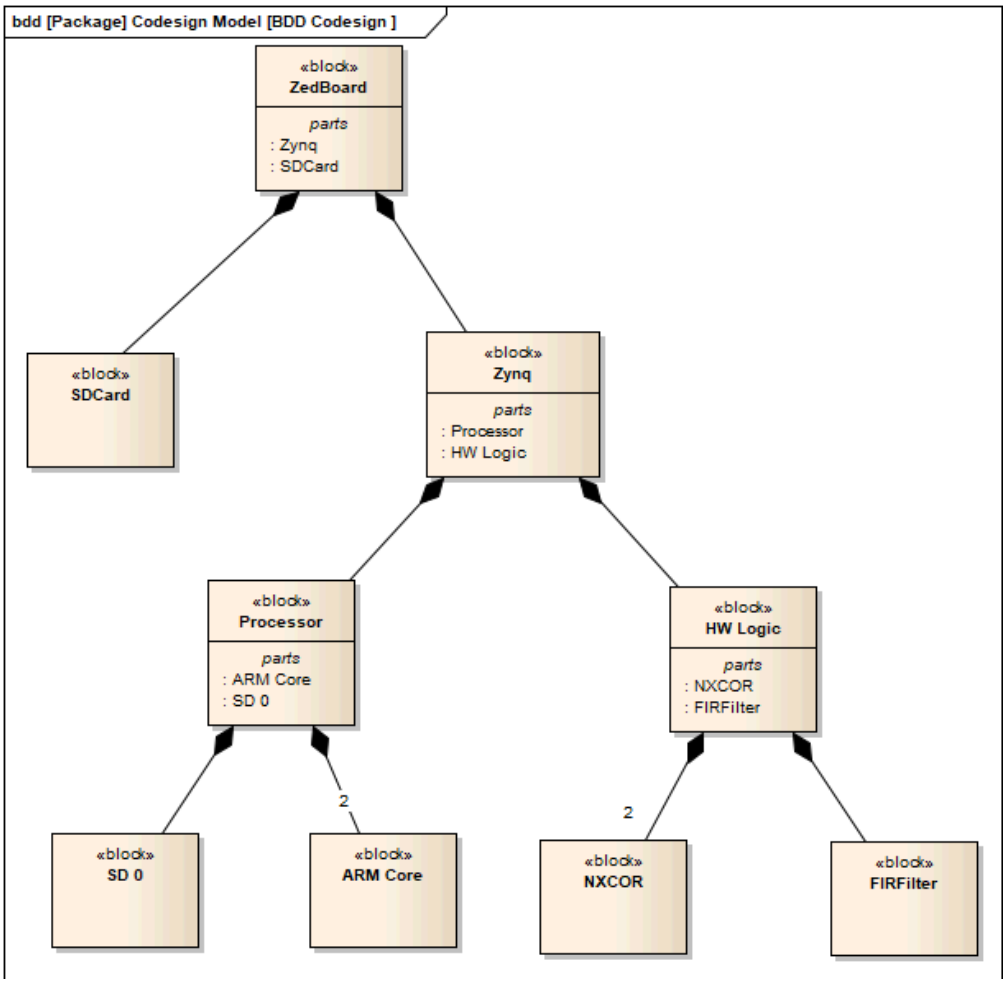


*Figure: 1*

**IBD Codesign** - *(SysML Internal Block diagram)*

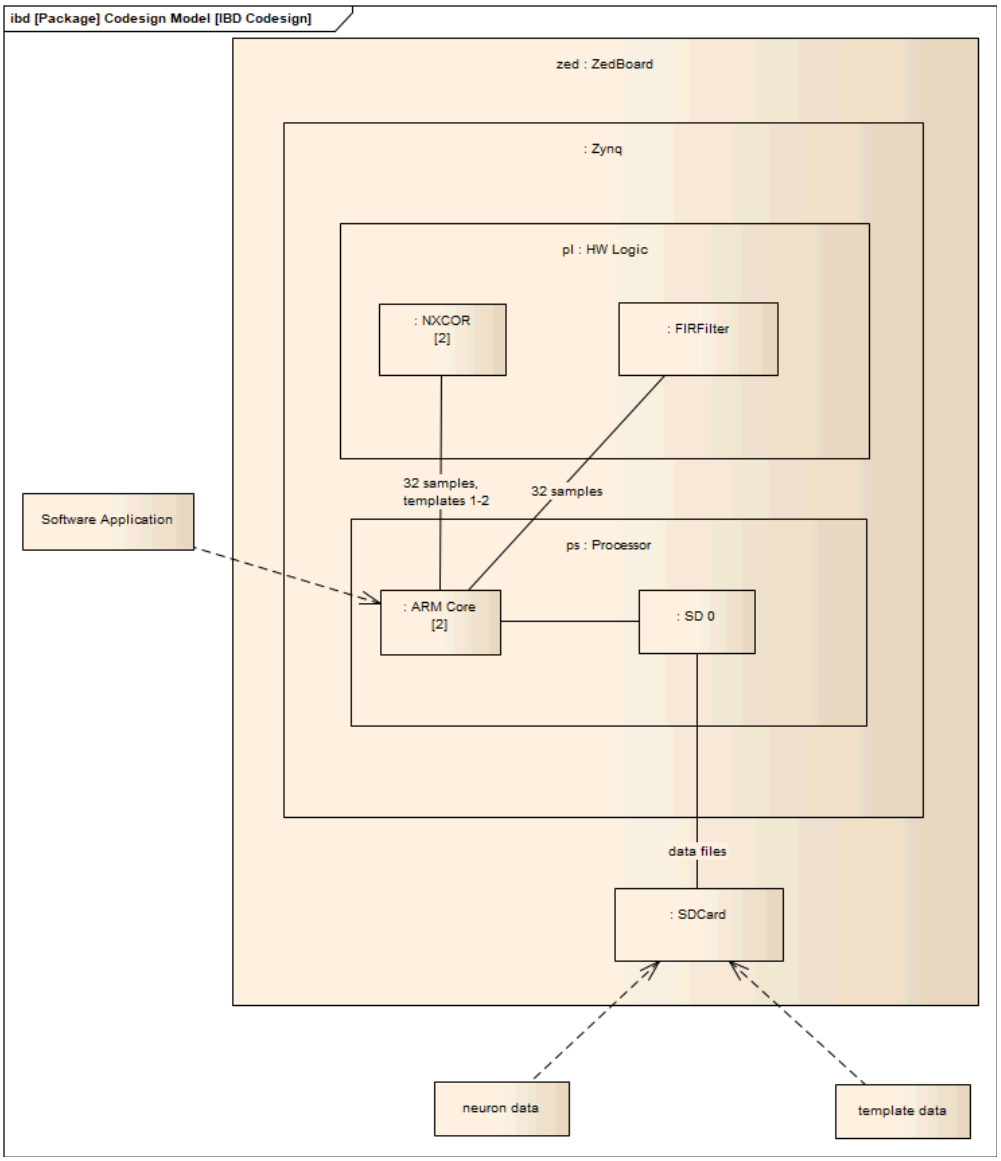


*Figure: 2*

**Seq Template Matching** - *(Sequence diagram)*

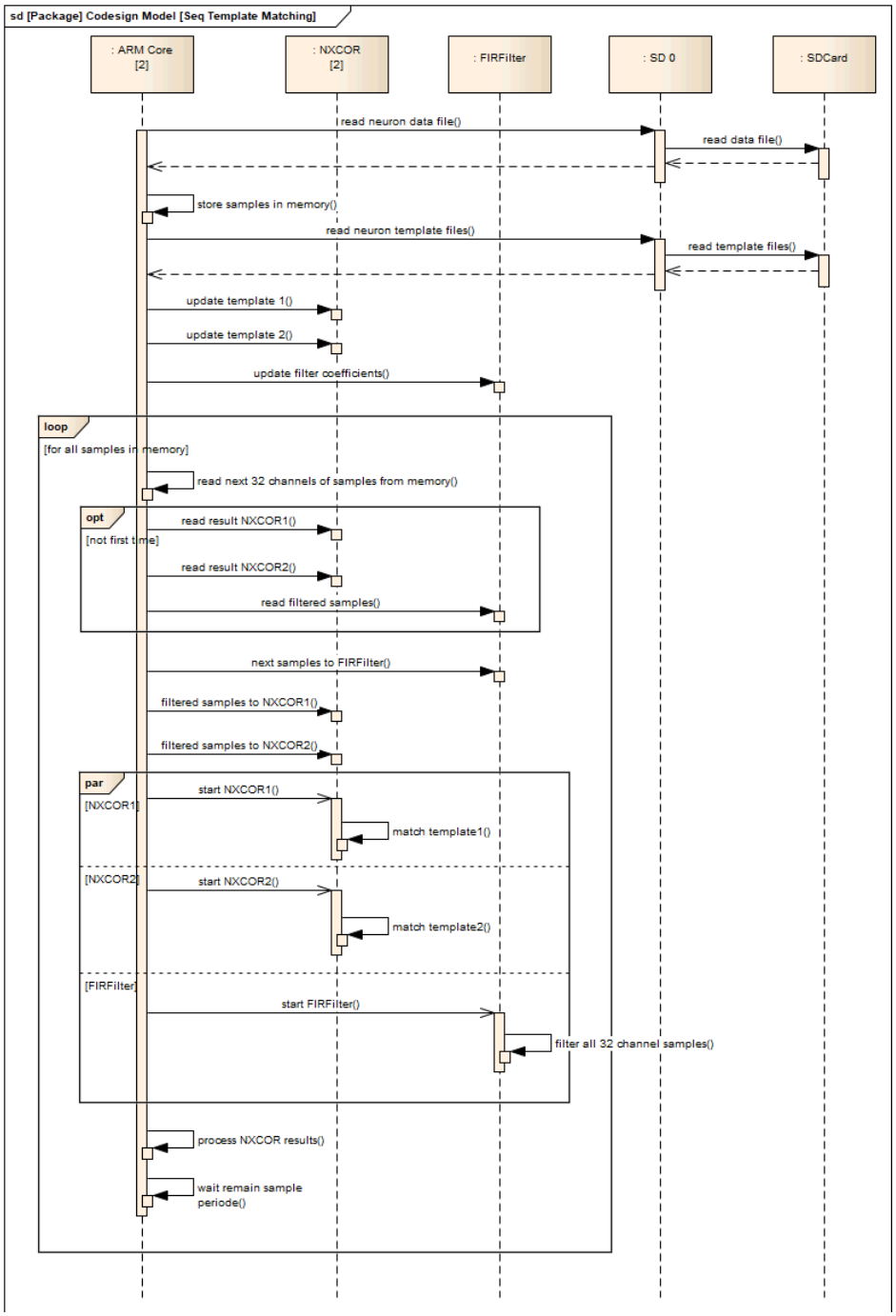
sd [Package] Codesign Model [Seq Template Matching]
: ARM Core [2]
: NXCOR [2]
: FIRFilter
: SD 0
: SDCard
read neuron data file()
read data file()
store samples in memory()
read neuron template files()
read template files()
update template 1()
update template 2()
update filter coefficients()
loop
[for all samples in memory]
read next 32 channels of samples from memory()
opt
[not first time]
read result NXCOR1()
read result NXCOR2()
read filtered samples()
next samples to FIRFilter()
filtered samples to NXCOR1()
filtered samples to NXCOR2()
par
[NXCOR1]
start NXCOR1()
match template1()
[NXCOR2]
start NXCOR2()
match template2()
[FIRFilter]
start FIRFilter()
filter all 32 channel samples()
process NXCOR results()
wait remain sample periode()

*Figure: 3*

**Timing Codesign** - *(Timing diagram)*

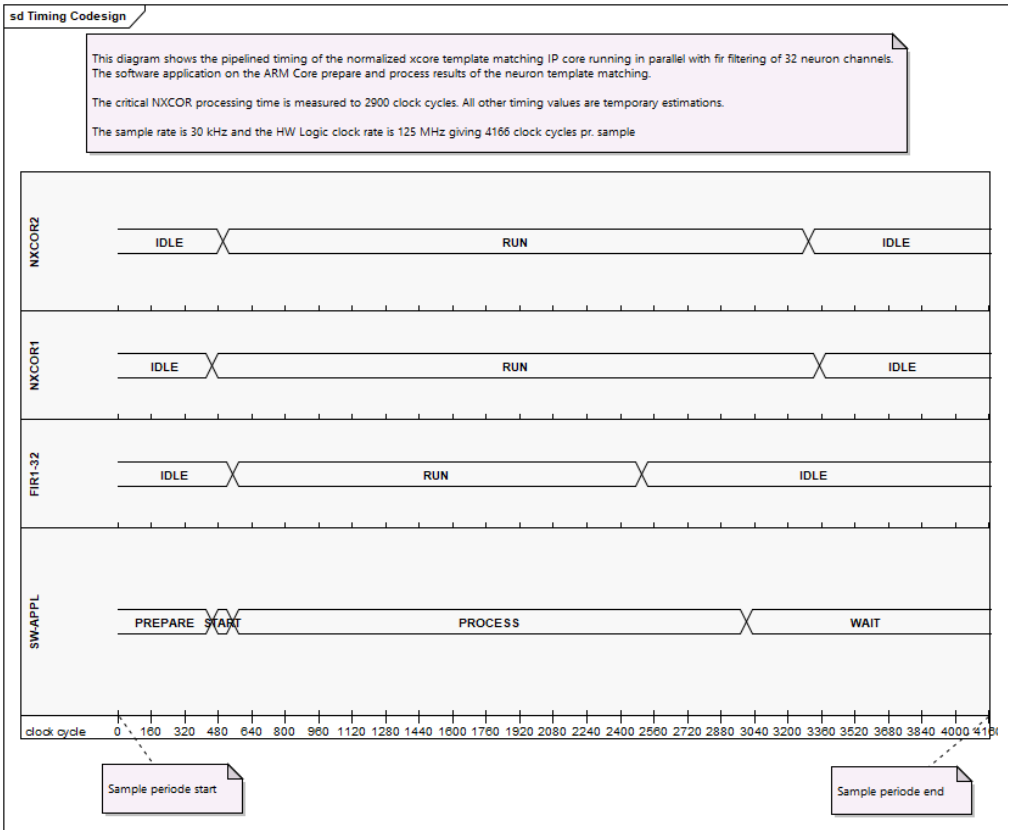


*Figure: 4*

**Class Diagram** - *(Class diagram)*

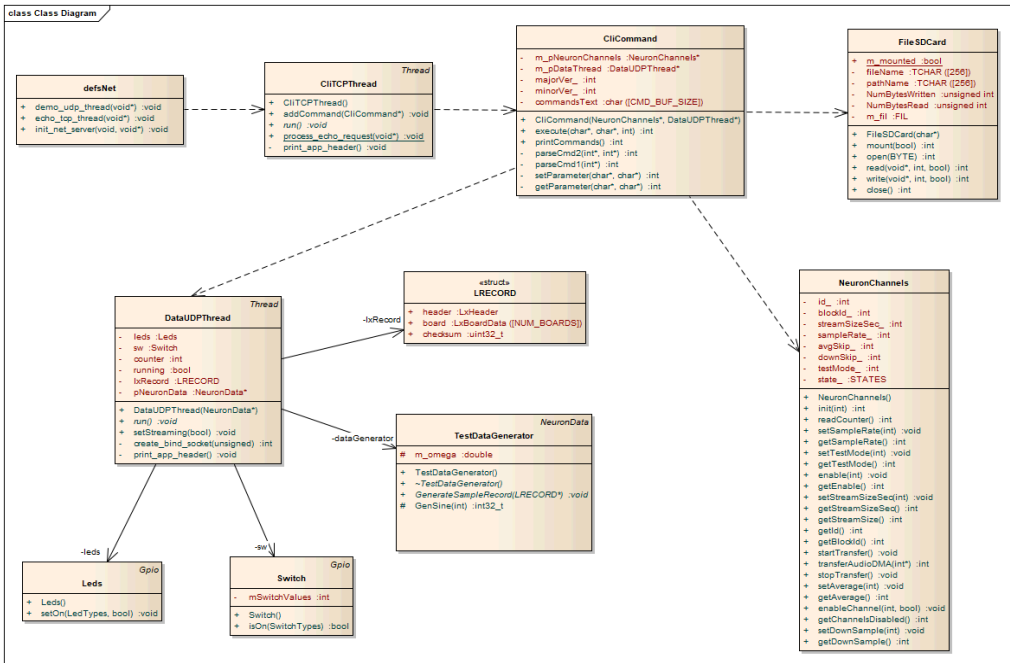


*Figure: 5*

**data** - *(Class diagram)*

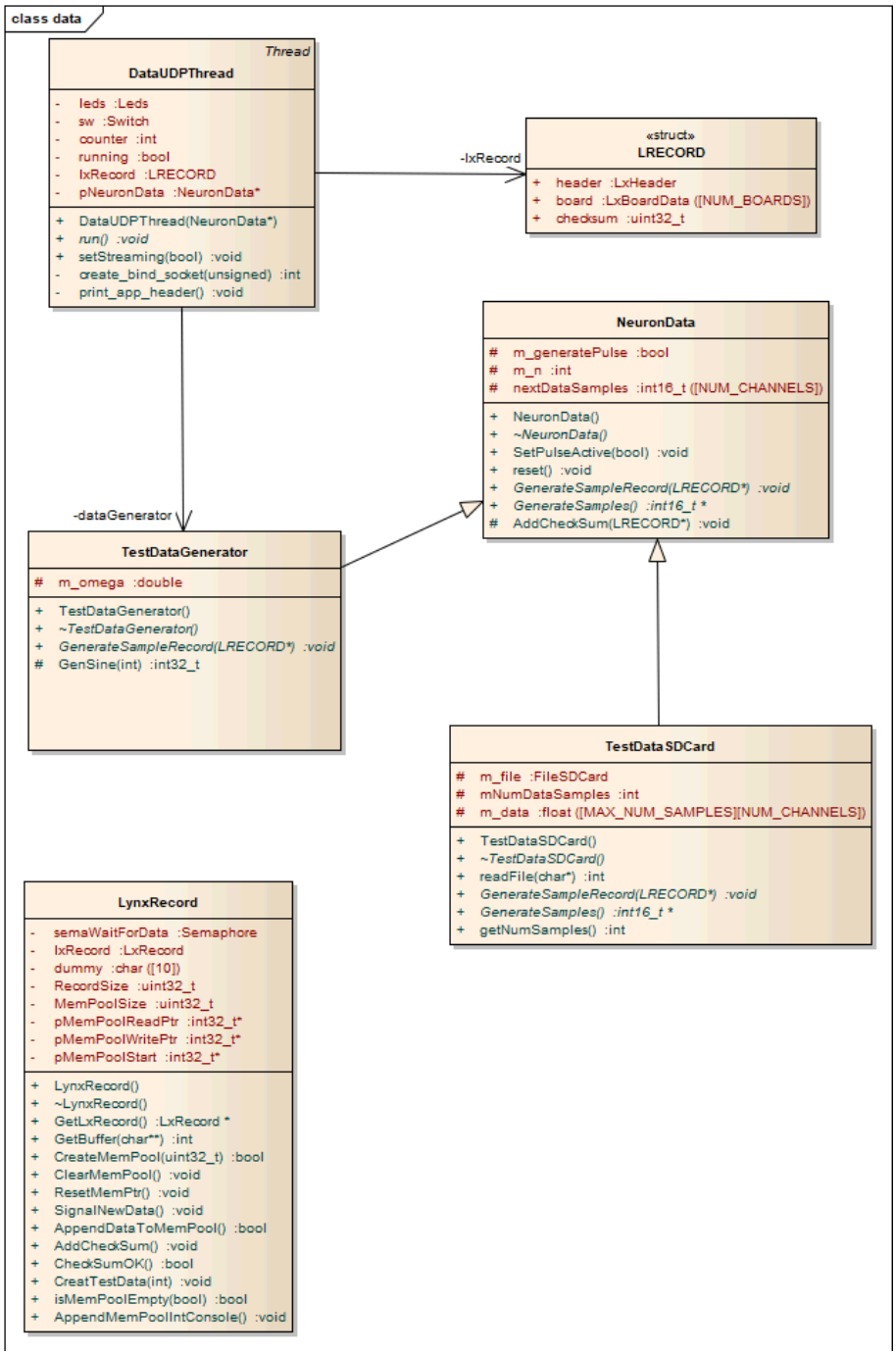
class data
DataUDPThread
Thread
- leds :Leds
- sw :Switch
- counter :int
- running :bool
- lxRecord :LRECORD
- pNeuronData :NeuronData*
+ DataUDPThread(NeuronData*)
+ run() :void
+ setStreaming(bool) :void
- create_bind_socket(unsigned) :int
- print_app_header() :void
-lxRecord
«struct»
LRECORD
+ header :LxHeader
+ board :LxBoardData ([NUM_BOARDS])
+ checksum :uint32_t
NeuronData
# m_generatePulse :bool
# m_n :int
# nextDataSamples :int16_t ([NUM_CHANNELS])
+ NeuronData()
+ ~NeuronData()
+ SetPulseActive(bool) :void
+ reset() :void
+ GenerateSampleRecord(LRECORD*) :void
+ GenerateSamples() :int16_t *
# AddCheckSum(LRECORD*) :void
-dataGenerator
TestDataGenerator
# m_omega :double
+ TestDataGenerator()
+ ~TestDataGenerator()
+ GenerateSampleRecord(LRECORD*) :void
# GenSine(int) :int32_t
TestDataSDCard
# m_file :FileSDCard
# mNumDataSamples :int
# m_data :float ([MAX_NUM_SAMPLES][NUM_CHANNELS])
+ TestDataSDCard()
+ ~TestDataSDCard()
+ readFile(char*) :int
+ GenerateSampleRecord(LRECORD*) :void
+ GenerateSamples() :int16_t *
+ getNumSamples() :int
LynxRecord
- semaWaitForData :Semaphore
- lxRecord :LxRecord
- dummy :char ([10])
- RecordSize :uint32_t
- MemPoolSize :uint32_t
- pMemPoolReadPtr :int32_t*
- pMemPoolWritePtr :int32_t*
- pMemPoolStart :int32_t*
+ LynxRecord()
+ ~LynxRecord()
+ GetLxRecord() :LxRecord *
+ GetBuffer(char**) :int
+ CreateMemPool(uint32_t) :bool
+ ClearMemPool() :void
+ ResetMemPtr() :void
+ SignalNewData() :void
+ AppendDataToMemPool() :bool
+ AddCheckSum() :void
+ CheckSumOK() :bool
+ CreatTestData(int) :void
+ isMemPoolEmpty(bool) :bool
+ AppendMemPoolIntConsole() :void

*Figure: 6*

**algo** - *(Class diagram)*

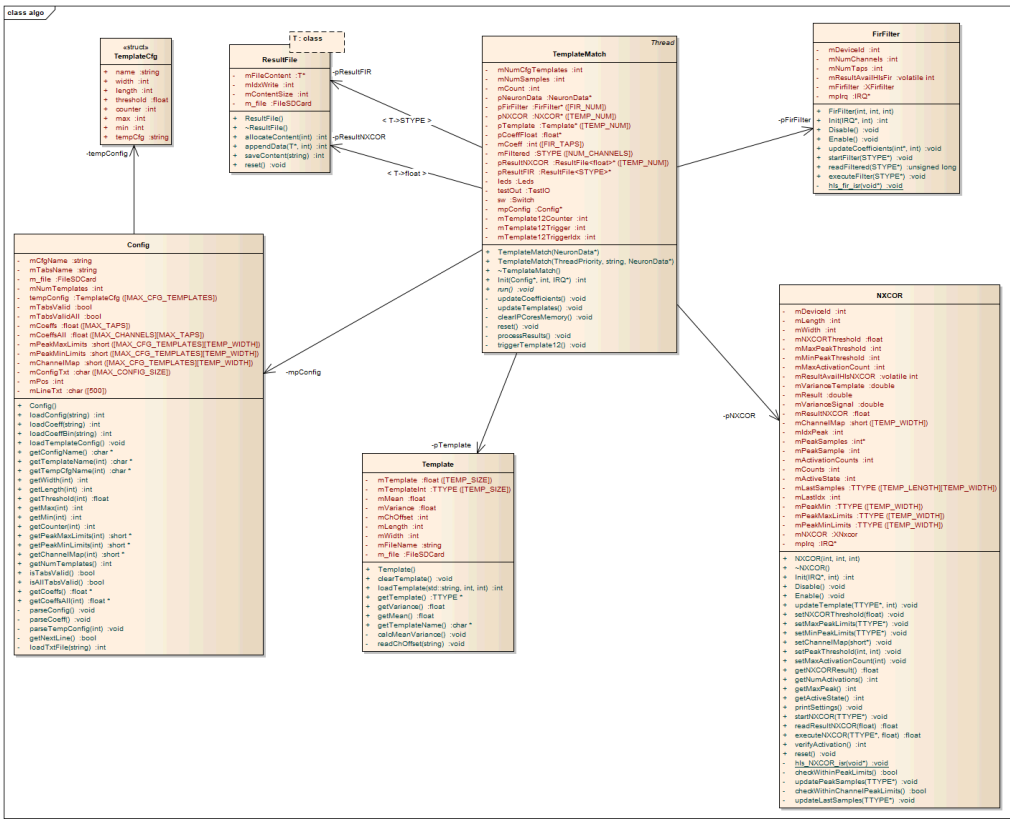


*Figure: 7*

**channels** - *(Class diagram)*

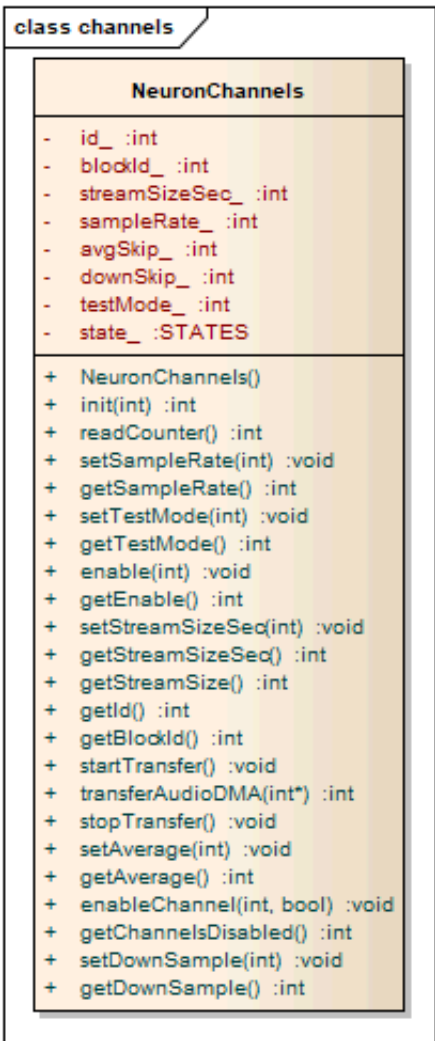


*Figure: 8*

**cli** - *(Class diagram)*

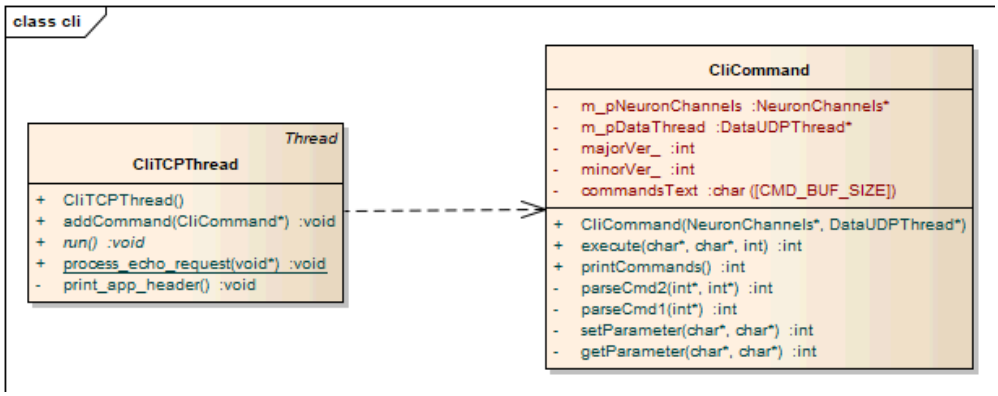


*Figure: 9*

**file** - *(Class diagram)*

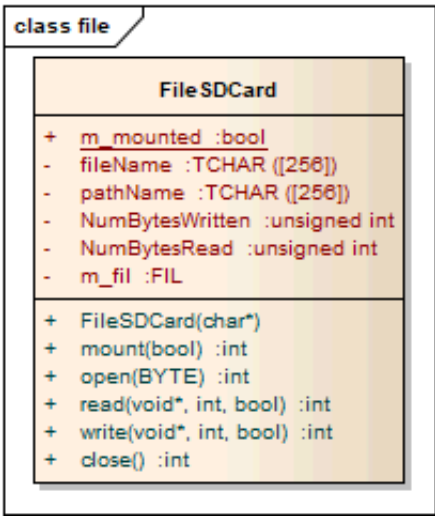


*Figure: 10*

**hal** - *(Class diagram)*

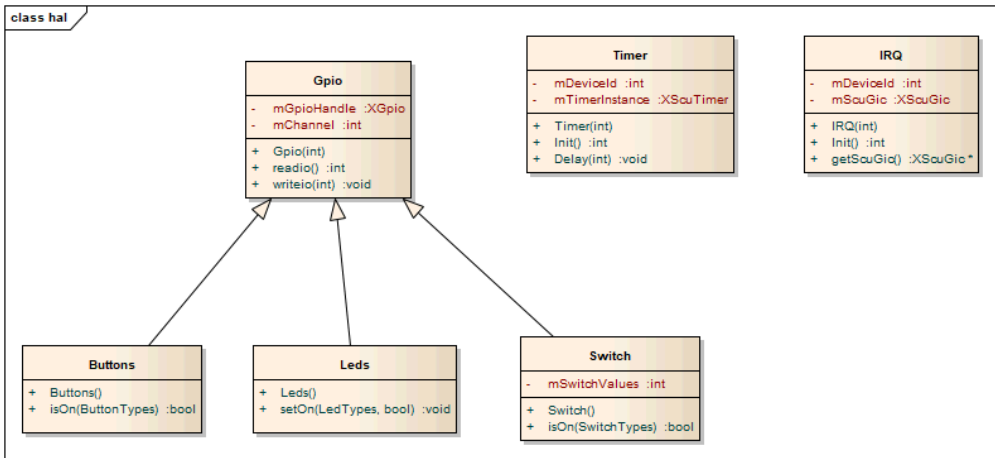


*Figure: 11*

**net** - *(Class diagram)*

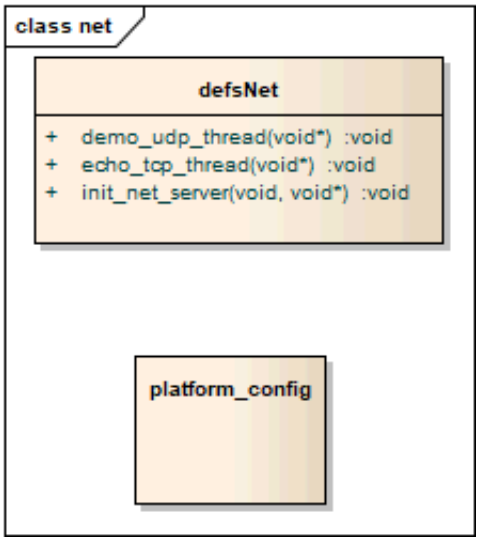


*Figure: 12*

**os** - *(Class diagram)*

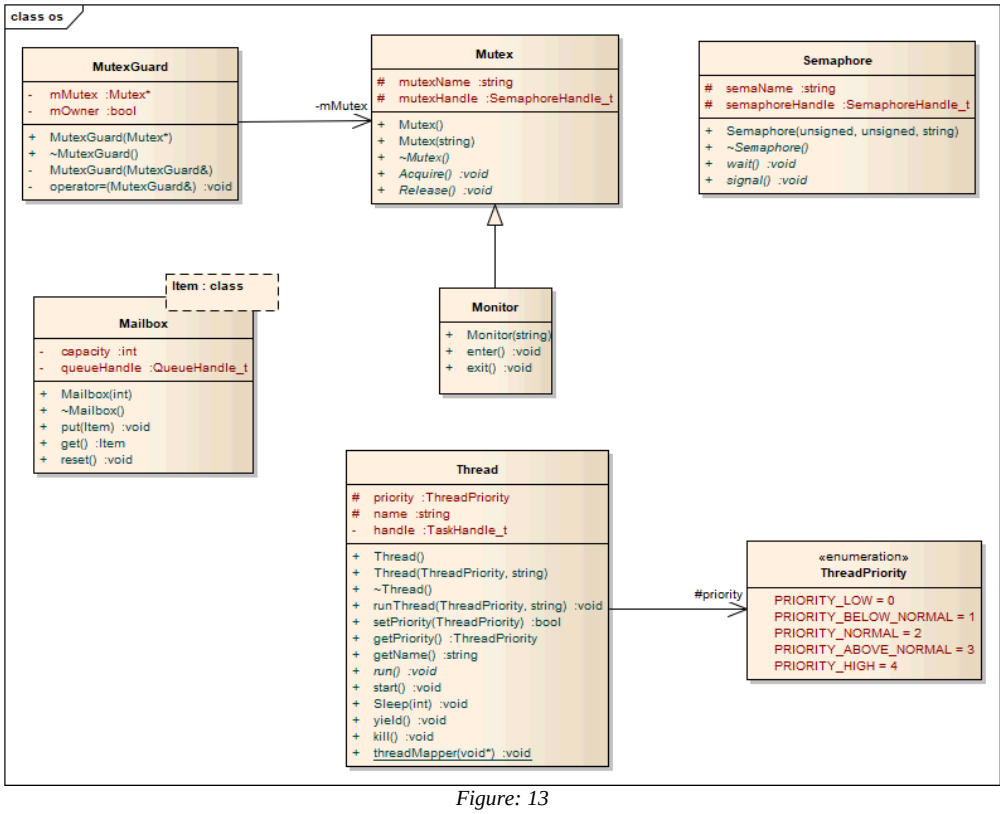
class os
MutexGuard
- mMutex :Mutex*
- mOwner :bool
+ MutexGuard(Mutex*)
+ ~MutexGuard()
- MutexGuard(MutexGuard&)
- operator=(MutexGuard&) :void
-mMutex
Mutex
# mutexName :string
# mutexHandle :SemaphoreHandle_t
+ Mutex()
+ Mutex(string)
+ ~Mutex()
+ Acquire() :void
+ Release() :void
Semaphore
# semaName :string
# semaphoreHandle :SemaphoreHandle_t
+ Semaphore(unsigned, unsigned, string)
+ ~Semaphore()
+ wait() :void
+ signal() :void
Item : class
Mailbox
- capacity :int
- queueHandle :QueueHandle_t
+ Mailbox(int)
+ ~Mailbox()
+ put(Item) :void
+ get() :Item
+ reset() :void
Monitor
+ Monitor(string)
+ enter() :void
+ exit() :void
Thread
# priority :ThreadPriority
# name :string
- handle :TaskHandle_t
+ Thread()
+ Thread(ThreadPriority, string)
+ ~Thread()
+ runThread(ThreadPriority, string) :void
+ setPriority(ThreadPriority) :bool
+ getPriority() :ThreadPriority
+ getName() :string
+ run() :void
+ start() :void
+ Sleep(int) :void
+ yield() :void
+ kill() :void
+ threadMapper(void*) :void
#priority
«enumeration»
ThreadPriority
PRIORITY_LOW = 0
PRIORITY_BELOW_NORMAL = 1
PRIORITY_NORMAL = 2
PRIORITY_ABOVE_NORMAL = 3
PRIORITY_HIGH = 4

*Figure: 13*