1. Transmission DutyCycle.

DutyCycle(DC) выражается в процентах – процент времени, который может занимать передатчик. В структуре ChannelParams\_t это uint16\_t dutyCycle. Для каждого канала DC свой.

dutyCycle=(100/DC)-1.

При запуске передачи запускается таймер RadioConfiguration.timeOnAirTimerId и после передачи вычисляется timeOnAir

Величина Channels[i].channelTimer – для каждого канала количество ms до окончания запрета на передачу или 0, если запрета нет. Вычисляется после каждой передачи (LORAWAN\_TxDone(lorawan\_ru.c) по timeOnAir и величине таймера loRa.dutyCycleTimerId) и после срабатывания DutyCycleCallback(lorawan\_ru.c) по величине таймера loRa.dutyCycleTimerId

Таймер loRa.dutyCycleTimerId запускается, если есть канал с ненулевым channelTimer.

dutyCycle может получать в качестве команды уменьшение DutyCycle кратное 2 в некоторой степени, где степень (от 0 до 15) получается в теле команды DUTY\_CYCLE\_CID.

Перед передачей в функции SelectChannelForTransmission(lorawan\_ru.c) сначала функцией SearchAvailableChannel(lorawan\_ru.c) ищется случайным образом канал с channelTimer==0. Если канал задан , то он просто проверяется на DC. Потом функция ConfigureRadioTx(lorawan\_ru.c) настраивает передатчик на параметры выбранного канала.

1. Таймеры.

Таймеры имеют один HW таймер TMR1, который тактируется от внешнего кварца 32.768 кГц.

Каждый SW таймер имеет структуру SwTimer\_t, в которой uint32\_t ticksRemaining – это тики до срабатывания Callback функции void (\*callback)(uint8\_t). Параметр uint8\_t running показывает тикает таймер или нет.

SwTimerCreate создает таймер, SwTimerSetCallback, SwTimerSetTimeout задают параметры таймера, SwTimerStart стартует таймер.

Принцип действия – задание значения для прерывания от HW таймера таким, которое является минимальным (т.е. произойдет раньше всех) значением оставшихся тиков из всех активных SW таймеров. Необходимо при этом как можно чаще запускать функцию SwTimersExecute для того чтобы не было задержек в выполнении callback-ов.

uint32\_t TMR\_GetDeltaTime - возвращает разницу в тиках после предыдущего вызова этой функции.

uint32\_t SwTimersInterrupt – корректирует значение оставшихся тиков в таймерах и возвращает минимальное значение тиков в активных таймерах. Значение 0 для оставшихся тиков означает запуск callback.

SwTimersExecute – запускает callback функции у таймеров с нулевым значением оставшихся тиков.

uint16\_t TMR\_SwapTimer(uint16\_t timerVal) – устанавливает новое значение HW таймера и возвращает старое.

void TMR\_ISR\_lora(tmr\_lora\_addons.c) – обработка прерывания таймера – вызывает SwTimersInterrupt для корректировки тиков таймеров, полученное ближайшее значение тика устанавливает для HW таймера с помощью TMR\_SwapTimer.

TMR\_OverrideRemaining(uint32\_t ticksRemaining) – перезаписывает тики в HW таймер, если при старте какого-либо SW таймера тики меньше, чем установленные в последнем вызове TMR\_SwapTimer. Возвращаемое значение служит для корректировки значения TMR\_GetDeltaTime.

1. NCO.

Интервал NCO - для выхода из режима SLEEP. При возврате из SLEEP если переменная nco==1, то выходим из SLEEP иначе снова выдаем команду SLEEP.

1. Процесс передачи:

err=LORAWAN\_Send(CNF, 2, &data, sizeof(data));(main.c)

AssemblePacket (confirmed, port, buffer, bufferLength);(lorawan.c)

RADIO\_Transmit (&macBuffer[16], (uint8\_t)loRa.lastPacketLength) (lorawan.c)