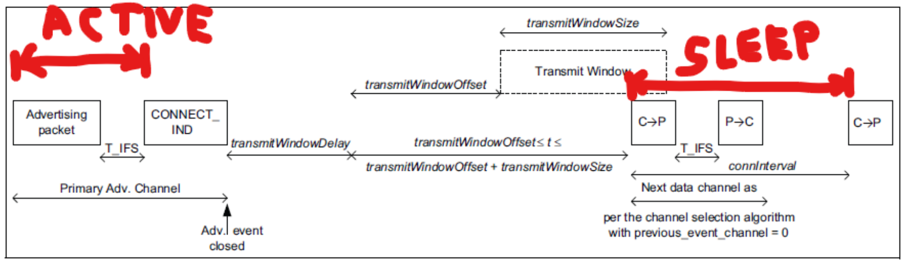
Controller to Host Timestamp Sync

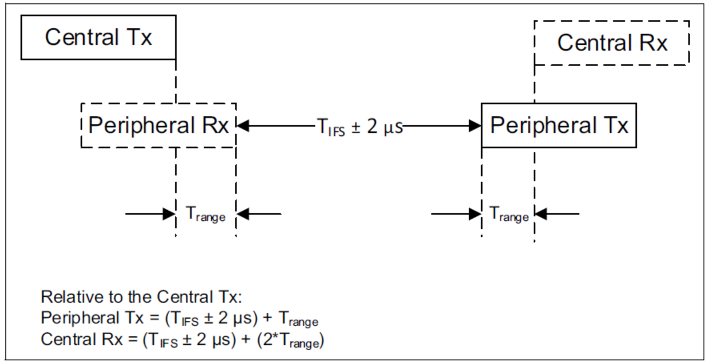
Vinayak Kariappa Chettimada

20th June 2023

Controller Clock

- History
  - Controller uses Sleep Clock and Active Clock
  - Sleep Clock
    - +/- 500 ppm accuracy
    - +/- 16 us deviation from average timing
  - Active Clock
    - +/- 50 ppm accuracy
    - +/- 2 us deviation from average timing
  - Sleep Clock: NRF_RTC0, 32KHz crystal, ~ +/-15 us jitter
  - Active Clock: NRF_TIMER0, 32/16MHz Crystal, +/- 1 us jitter

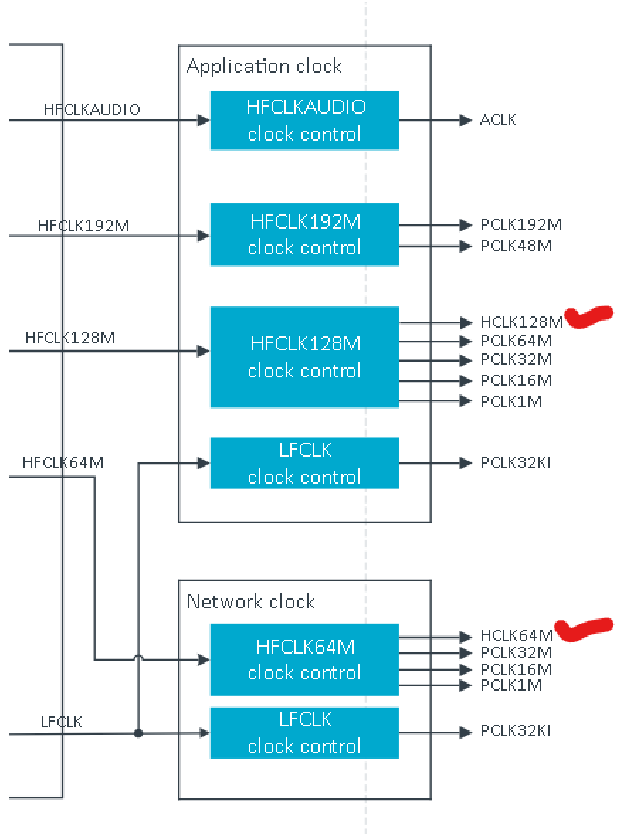




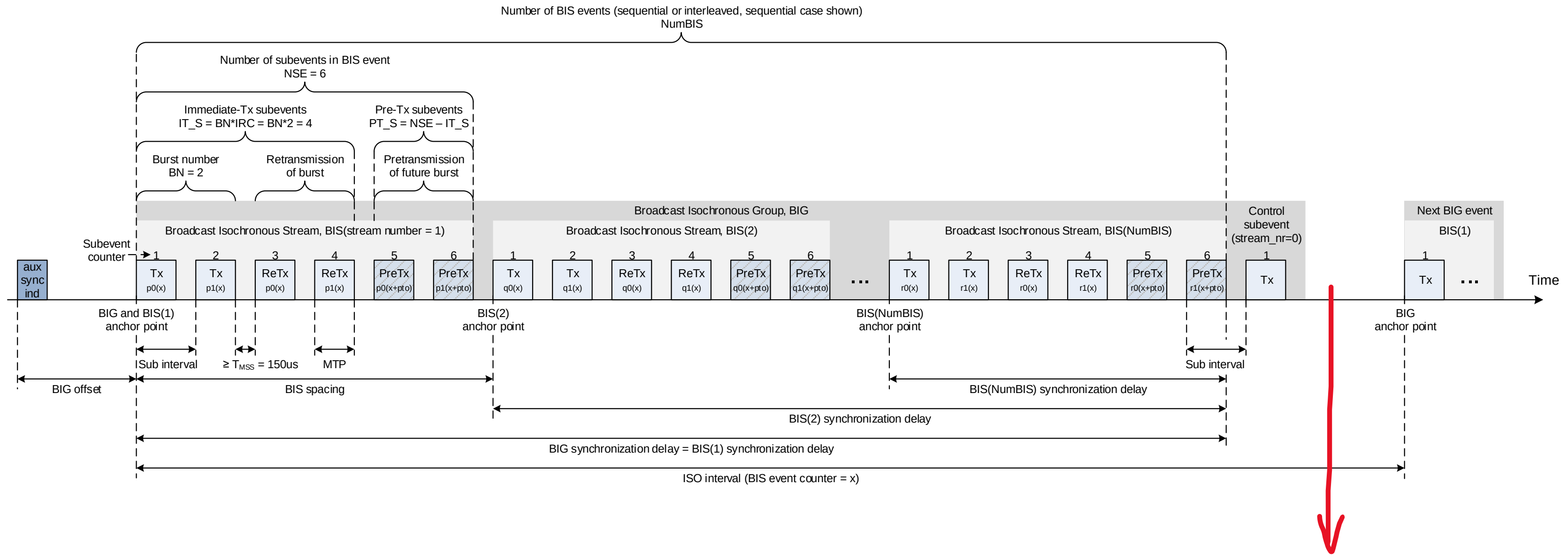
Relative to the Central Tx:
Peripheral Tx = ($T_{IFS} \pm 2\ \mu s$) + T_{range}
Central Rx = ($T_{IFS} \pm 2\ \mu s$) + (2^*T_{range})

Host Clock

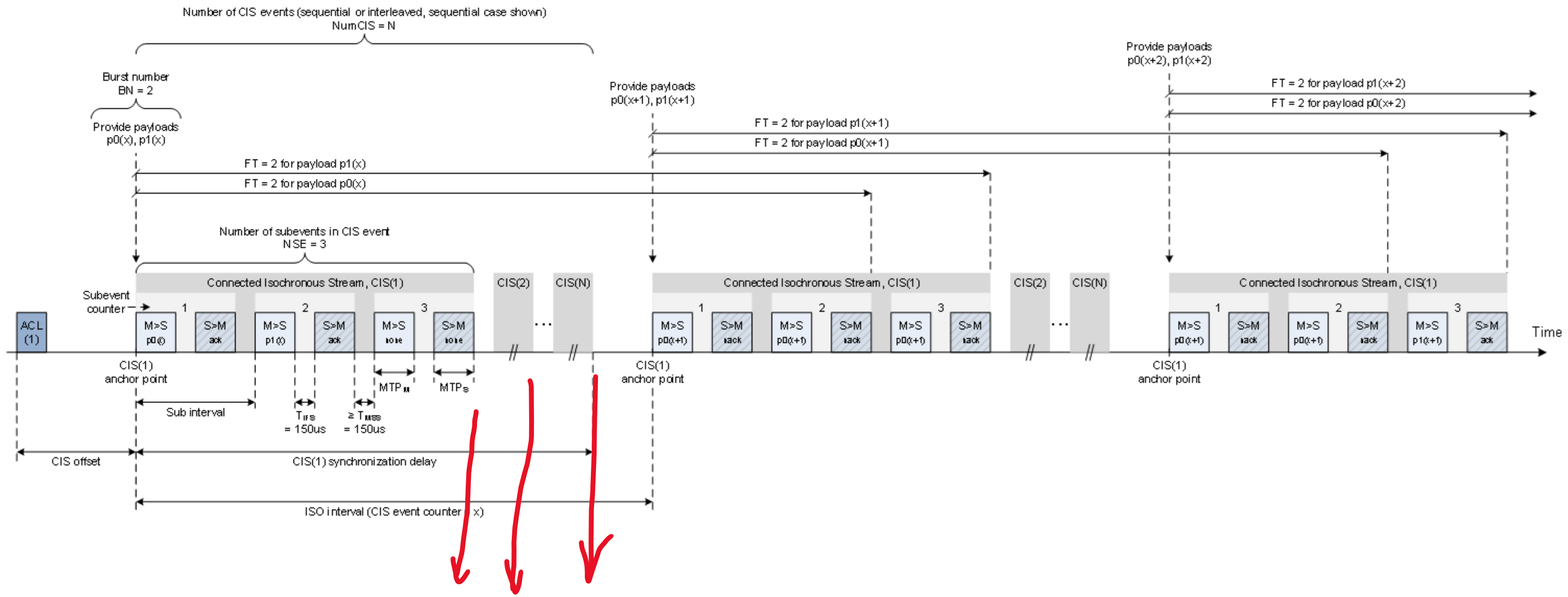
- Zephyr Clock Control
  - HFCLK
  - LFCLK
  - HFCLKAUDIO
- Application Core CPU
  - 128 or 64 MHz
  - HFCLKCTRL, divide by 1 or 2
- System Timer: NRF_RTC1



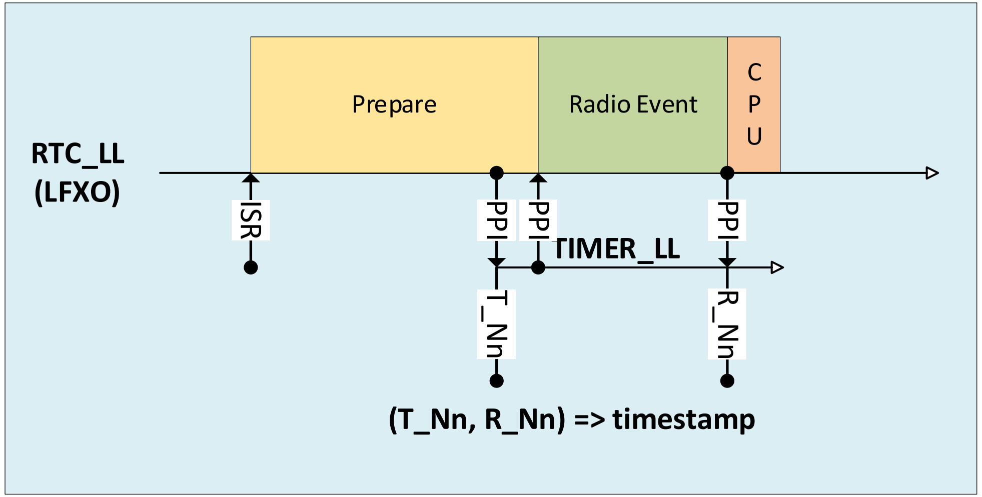
BIG Event


Number of BIS events (sequential or interleaved, sequential case shown)
NumBIS
Number of subevents in BIS event
NSE = 6
Immediate-Tx subevents
IT_S = BN*IRC = BN*2 = 4
Pre-Tx subevents
PT_S = NSE – IT_S
Burst number
BN = 2
Retransmission
of burst
Pretransmission
of future burst
Broadcast Isochronous Group, BIG
Control subevent (stream_nr=0)
Next BIG event
BIS(1)
Broadcast Isochronous Stream, BIS(stream number = 1)
Broadcast Isochronous Stream, BIS(2)
Broadcast Isochronous Stream, BIS(NumBIS)
Subevent counter
aux sync ind
1 2 3 4 5 6
Tx p0(x)
Tx p1(x)
ReTx p0(x)
ReTx p1(x)
PreTx p0(x+pto)
PreTx p1(x+pto)
Tx q0(x)
Tx q1(x)
ReTx q0(x)
ReTx q1(x)
PreTx q0(x+pto)
PreTx q1(x+pto)
...
Tx r0(x)
Tx r1(x)
ReTx r0(x)
ReTx r1(x)
PreTx r0(x+pto)
PreTx r1(x+pto)
Tx
Tx
...
Time
BIG and BIS(1) anchor point
BIS(2) anchor point
BIS(NumBIS) anchor point
BIG anchor point
Sub interval
≥ T_MSS = 150us
MTP
Sub interval
BIG offset
BIS spacing
BIS(NumBIS) synchronization delay
BIS(2) synchronization delay
BIG synchronization delay = BIS(1) synchronization delay
ISO interval (BIS event counter = x)

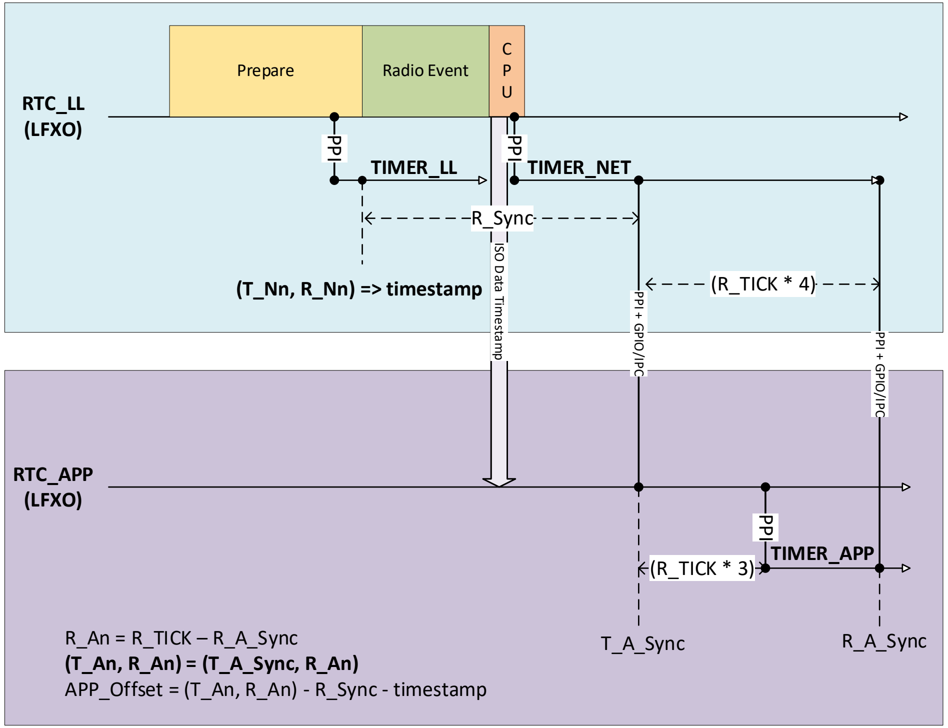
CIG event


Number of CIS events (sequential or interleaved, sequential case shown)
NumCIS = N
Burst number
BN = 2
Provide payloads
p0(x), p1(x)
FT = 2 for payload p1(x)
FT = 2 for payload p0(x)
Number of subevents in CIS event
NSE = 3
Connected Isochronous Stream, CIS(1)
Subevent counter
1
2
3
ACL (1)
M>S p0(x)
S>M ack
M>S p1(x)
S>M ack
M>S none
S>M none
CIS(2)
CIS(N)
CIS(1) anchor point
Sub interval
T_IFS = 150us
≥ T_MSS = 150us
MTP_M
MTP_S
CIS offset
CIS(1) synchronization delay
ISO interval (CIS event counter = x)
Provide payloads
p0(x+1), p1(x+1)
FT = 2 for payload p1(x+1)
FT = 2 for payload p0(x+1)
M>S p0(x+1)
S>M nack
M>S p0(x+1)
S>M nack
M>S p0(x+1)
S>M nack
Provide payloads
p0(x+2), p1(x+2)
FT = 2 for payload p1(x+2)
FT = 2 for payload p0(x+2)
M>S p0(x+1)
S>M nack
M>S p0(x+1)
S>M nack
M>S p1(x+1)
S>M ack
Time

Zephyr Controller Timestamp



Solution Option 1



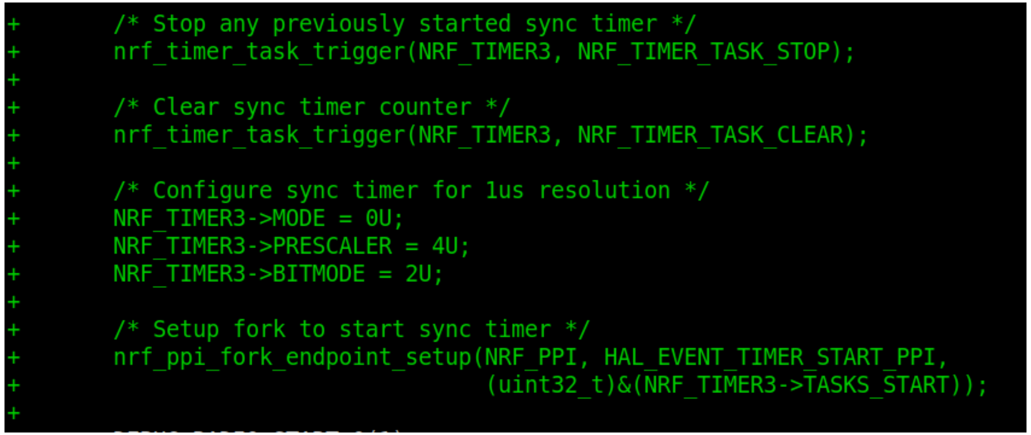
R_An = R_TICK – R_A_Sync

(T_An, R_An) = (T_A_Sync, R_An)

APP_Offset = (T_An, R_An) - R_Sync - timestamp

Experimentation: Sync Timer

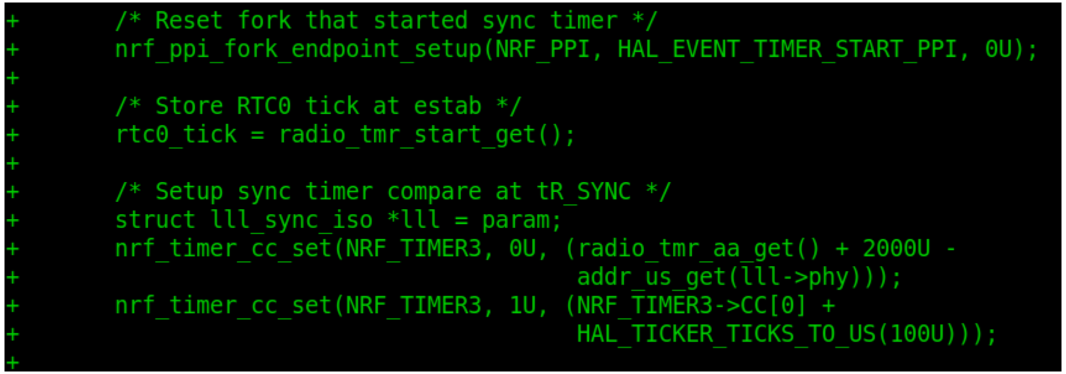
- NRF_TIMER3
  - Used sync timer in controller
  - Start in parallel with NRF_TIMER0



```
+       /* Stop any previously started sync timer */
+       nrf_timer_task_trigger(NRF_TIMER3, NRF_TIMER_TASK_STOP);
+
+       /* Clear sync timer counter */
+       nrf_timer_task_trigger(NRF_TIMER3, NRF_TIMER_TASK_CLEAR);
+
+       /* Configure sync timer for 1us resolution */
+       NRF_TIMER3->MODE = 0U;
+       NRF_TIMER3->PRESCALER = 4U;
+       NRF_TIMER3->BITMODE = 2U;
+
+       /* Setup fork to start sync timer */
+       nrf_ppi_fork_endpoint_setup(NRF_PPI, HAL_EVENT_TIMER_START_PPI,
+                                   (uint32_t)&(NRF_TIMER3->TASKS_START));
+
```

Experimentation: Setup Sync Timer PPI triggers

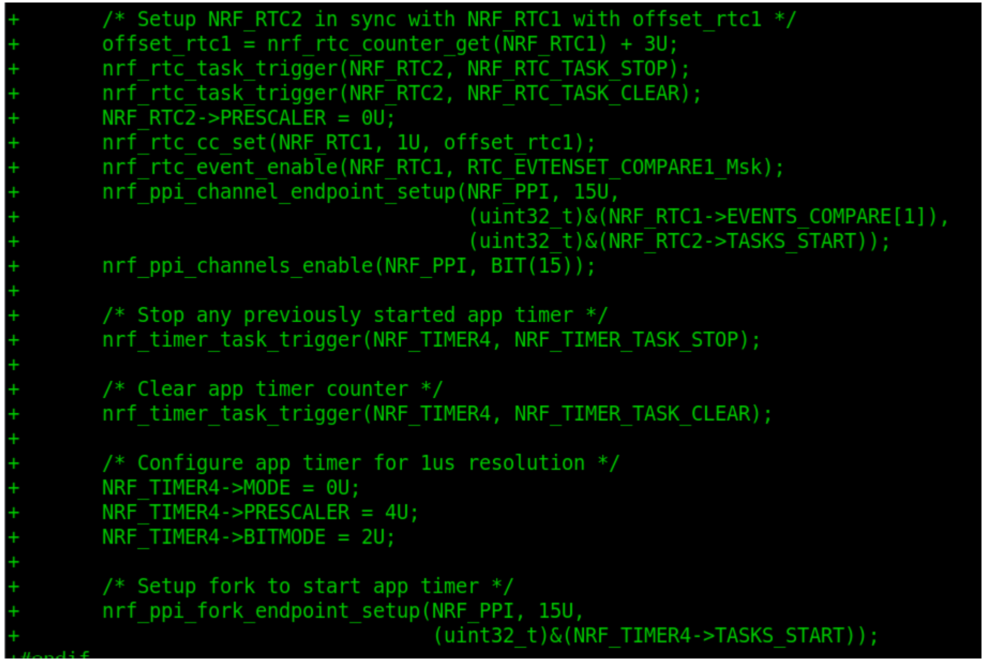
- Setup a NRF_TIMER3 compare to trigger at 2000 us from the BIG Sync Anchor Point
- Setup another compare 100 RTC ticks from the first compare trigger



```
+        /* Reset fork that started sync timer */
+        nrf_ppi_fork_endpoint_setup(NRF_PPI, HAL_EVENT_TIMER_START_PPI, 0U);
+
+        /* Store RTC0 tick at estab */
+        rtc0_tick = radio_tmr_start_get();
+
+        /* Setup sync timer compare at tR_SYNC */
+        struct lll_sync_iso *lll = param;
+        nrf_timer_cc_set(NRF_TIMER3, 0U, (radio_tmr_aa_get() + 2000U -
+                                          addr_us_get(lll->phy)));
+        nrf_timer_cc_set(NRF_TIMER3, 1U, (NRF_TIMER3->CC[0] +
+                                          HAL_TICKER_TICKS_TO_US(100U)));
+
```

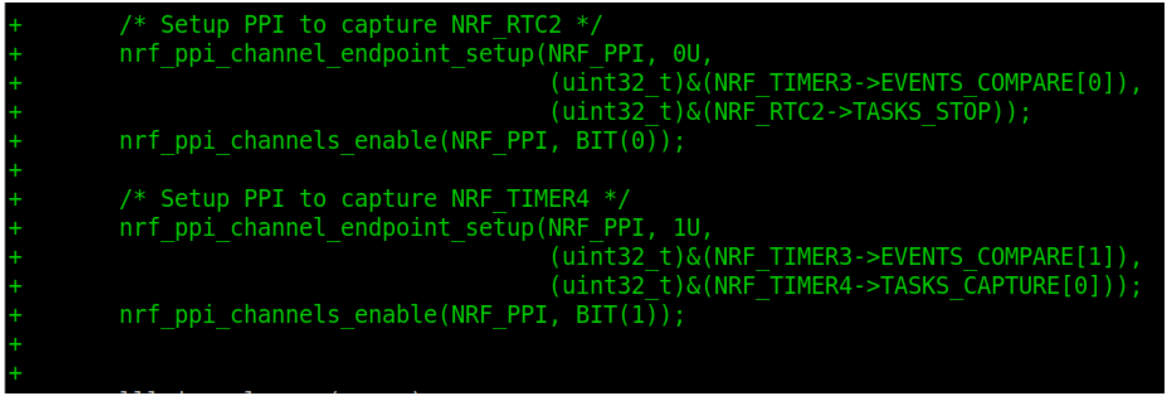
Experimentation: Host Clock Capture

- Start NRF_RTC2 at **`offset_rtc1`**



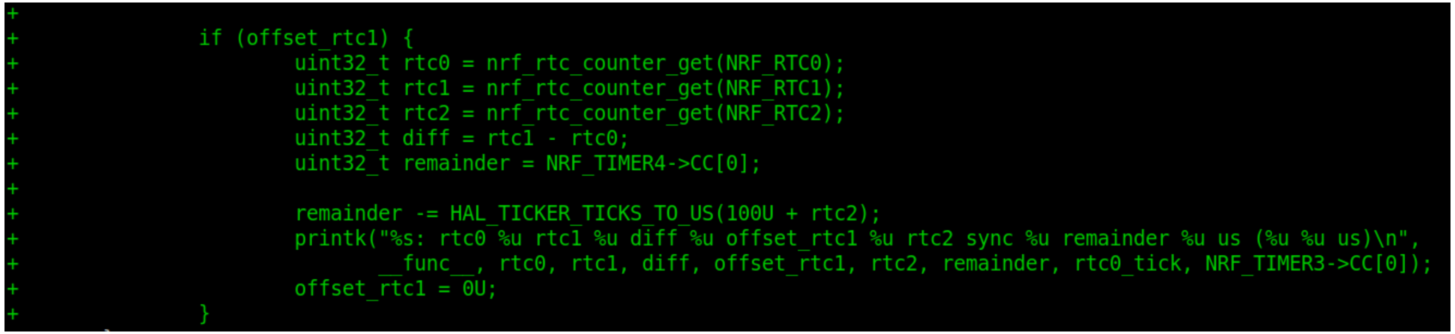
```
+       /* Setup NRF_RTC2 in sync with NRF_RTC1 with offset_rtc1 */
+       offset_rtc1 = nrf_rtc_counter_get(NRF_RTC1) + 3U;
+       nrf_rtc_task_trigger(NRF_RTC2, NRF_RTC_TASK_STOP);
+       nrf_rtc_task_trigger(NRF_RTC2, NRF_RTC_TASK_CLEAR);
+       NRF_RTC2->PRESCALER = 0U;
+       nrf_rtc_cc_set(NRF_RTC1, 1U, offset_rtc1);
+       nrf_rtc_event_enable(NRF_RTC1, RTC_EVTENSET_COMPARE1_Msk);
+       nrf_ppi_channel_endpoint_setup(NRF_PPI, 15U,
+                                      (uint32_t)&(NRF_RTC1->EVENTS_COMPARE[1]),
+                                      (uint32_t)&(NRF_RTC2->TASKS_START));
+       nrf_ppi_channels_enable(NRF_PPI, BIT(15));
+
+       /* Stop any previously started app timer */
+       nrf_timer_task_trigger(NRF_TIMER4, NRF_TIMER_TASK_STOP);
+
+       /* Clear app timer counter */
+       nrf_timer_task_trigger(NRF_TIMER4, NRF_TIMER_TASK_CLEAR);
+
+       /* Configure app timer for 1us resolution */
+       NRF_TIMER4->MODE = 0U;
+       NRF_TIMER4->PRESCALER = 4U;
+       NRF_TIMER4->BITMODE = 2U;
+
+       /* Setup fork to start app timer */
+       nrf_ppi_fork_endpoint_setup(NRF_PPI, 15U,
+                                   (uint32_t)&(NRF_TIMER4->TASKS_START));
+#endif
```

Experimentation: Capture Host Ticks and Remainder



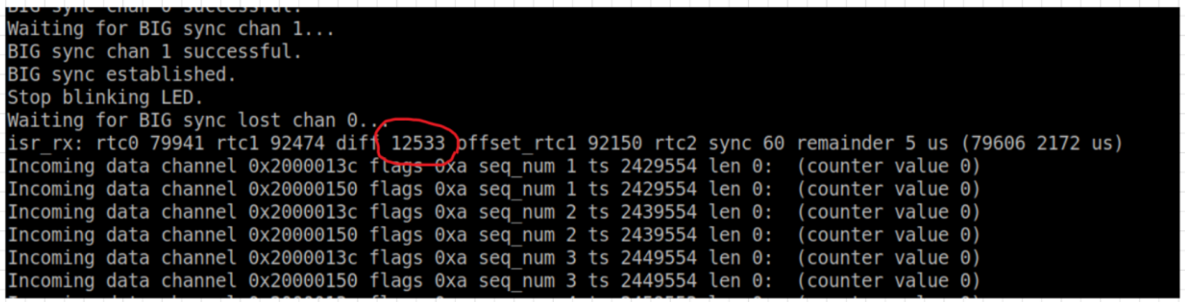
```
+        /* Setup PPI to capture NRF_RTC2 */
+        nrf_ppi_channel_endpoint_setup(NRF_PPI, 0U,
+                                       (uint32_t)&(NRF_TIMER3->EVENTS_COMPARE[0]),
+                                       (uint32_t)&(NRF_RTC2->TASKS_STOP));
+        nrf_ppi_channels_enable(NRF_PPI, BIT(0));
+
+        /* Setup PPI to capture NRF_TIMER4 */
+        nrf_ppi_channel_endpoint_setup(NRF_PPI, 1U,
+                                       (uint32_t)&(NRF_TIMER3->EVENTS_COMPARE[1]),
+                                       (uint32_t)&(NRF_TIMER4->TASKS_CAPTURE[0]));
+        nrf_ppi_channels_enable(NRF_PPI, BIT(1));
+
+
```

Experimentation: output



```
+
+               if (offset_rtc1) {
+                       uint32_t rtc0 = nrf_rtc_counter_get(NRF_RTC0);
+                       uint32_t rtc1 = nrf_rtc_counter_get(NRF_RTC1);
+                       uint32_t rtc2 = nrf_rtc_counter_get(NRF_RTC2);
+                       uint32_t diff = rtc1 - rtc0;
+                       uint32_t remainder = NRF_TIMER4->CC[0];
+
+                       remainder -= HAL_TICKER_TICKS_TO_US(100U + rtc2);
+                       printk("%s: rtc0 %u rtc1 %u diff %u offset_rtc1 %u rtc2 sync %u remainder %u us (%u %u us)\n",
+                              __func__, rtc0, rtc1, diff, offset_rtc1, rtc2, remainder, rtc0_tick, NRF_TIMER3->CC[0]);
+                       offset_rtc1 = 0U;
+               }
```

Experimentation: Analysis



```
BIG sync chan 0 successful.
Waiting for BIG sync chan 1...
BIG sync chan 1 successful.
BIG sync established.
Stop blinking LED.
Waiting for BIG sync lost chan 0...
isr_rx: rtc0 79941 rtc1 92474 diff 12533 offset_rtc1 92150 rtc2 sync 60 remainder 5 us (79606 2172 us)
Incoming data channel 0x2000013c flags 0xa seq_num 1 ts 2429554 len 0:  (counter value 0)
Incoming data channel 0x20000150 flags 0xa seq_num 1 ts 2429554 len 0:  (counter value 0)
Incoming data channel 0x2000013c flags 0xa seq_num 2 ts 2439554 len 0:  (counter value 0)
Incoming data channel 0x20000150 flags 0xa seq_num 2 ts 2439554 len 0:  (counter value 0)
Incoming data channel 0x2000013c flags 0xa seq_num 3 ts 2449554 len 0:  (counter value 0)
Incoming data channel 0x20000150 flags 0xa seq_num 3 ts 2449554 len 0:  (counter value 0)
```

CONTROLLER:
(T_Nn, R_Nn + 2000U) = (79606, 2172) = (79677, 5)

HOST:
T_A_Sync = 92150 + 60 = 92210
(T_A_Sync, R_A_sync) = (92210, 5)
diff = 92210 – 79677 = **12533** ticks

Captured timestamp on NRF_RTC1 + NRF_TIMER4 (say by host, T_A_Sync, R_A_Sync) has the same RTC ticks difference as measured between NRF_RTC0 and NRF_RTC1 on the SoC. NRF_RTC0 + NRF_TIMER3 by controller was used to trigger a PPI.

Q&A