```
// 1 channel RCInput with 5ns accuracy
// 12 channel RCOutput with 1us accuracy

// Timer
#define TICK_PER_US 200
#define TICK_PER_MS 200000

// PWM

// 920 us
#define PWM_PULSE_DEFAULT (920 * TICK_PER_US)

// 50 Hz
#define PWM_FREQ_DEFAULT (20 * TICK_PER_MS)

// Ringbuffer size
#define RCIN_RINGBUFFERSIZE 300

// PRU Constants Table
#define ECAP C3
#define RAM C24
#define IEP C26

// IEP
#define IEP_TMR_GLB_CFG 0x0
#define IEP_TMR_GLB_STS 0x4
#define IEP_TMR_CNT 0xc

#define IEP_CNT_ENABLE 0x0
#define IEP_DEFAULT_INC 0x4

// ECAP
#define ECAP_TSCTR 0x0
#define ECAP_CTRPHS 0x4
#define ECAP_CAP1 0x8
#define ECAP_CAP2 0xc
#define ECAP_CAP3 0x10
#define ECAP_CAP4 0x14
#define ECAP_ECCTL1 0x28
#define ECAP_ECCTL2 0x2a
#define ECAP_ECEINT 0x2c
#define ECAP_ECFLG 0x2e
#define ECAP_ECCLR 0x30
#define ECAP_ECFRC 0x32
#define ECAP_REVID 0x5c

// ECCTL1
#define ECAP_CAP1POL 0
#define ECAP_CTRRST1 1
#define ECAP_CAP2POL 2
#define ECAP_CTRRST2 3
#define ECAP_CAP3POL 4
#define ECAP_CTRRST3 5
#define ECAP_CAP4POL 6
#define ECAP_CTRRST4 7
#define ECAP_CAPLDEN 8
#define ECAP_PRESCALE 9
#define ECAP_FREE_SOFT 14

// ECCTL2
#define ECAP_CONT_ONESHT 0
#define ECAP_STOP_WRAP 1
#define ECAP_RE_ARM 3
```

```
#define ECAP_TSCTRSTOP 4
#define ECAP_SYNCI_EN 5
#define ECAP_SYNCO_SEL 6
#define ECAP_SWSYNC 8
#define ECAP_CAP_APWM 9
#define ECAP_APWMPOL 10

// ECEINT, ECFLG
#define ECAP_INT 0
#define ECAP_CEVT1 1
#define ECAP_CEVT2 2
#define ECAP_CEVT3 3
#define ECAP_CEVT4 4
#define ECAP_CNTOVF 5
#define ECAP_PRDEQ 6
#define ECAP_CMPEQ 7

// RAM
#define CH_ENABLE_RAM_OFFSET (0 * 4)
#define CH_1_PULSE_TIME_RAM_OFFSET (1 * 4)
#define CH_1_T_TIME_RAM_OFFSET  (2 * 4)
#define CH_2_PULSE_TIME_RAM_OFFSET (3 * 4)
#define CH_2_T_TIME_RAM_OFFSET  (4 * 4)
#define CH_3_PULSE_TIME_RAM_OFFSET (5 * 4)
#define CH_3_T_TIME_RAM_OFFSET  (6 * 4)
#define CH_4_PULSE_TIME_RAM_OFFSET (7 * 4)
#define CH_4_T_TIME_RAM_OFFSET  (8 * 4)
#define CH_5_PULSE_TIME_RAM_OFFSET (9 * 4)
#define CH_5_T_TIME_RAM_OFFSET  (10 * 4)
#define CH_6_PULSE_TIME_RAM_OFFSET (11 * 4)
#define CH_6_T_TIME_RAM_OFFSET  (12 * 4)
#define CH_7_PULSE_TIME_RAM_OFFSET (13 * 4)
#define CH_7_T_TIME_RAM_OFFSET  (14 * 4)
#define CH_8_PULSE_TIME_RAM_OFFSET (15 * 4)
#define CH_8_T_TIME_RAM_OFFSET  (16 * 4)
#define CH_9_PULSE_TIME_RAM_OFFSET (17 * 4)
#define CH_9_T_TIME_RAM_OFFSET  (18 * 4)
#define CH_10_PULSE_TIME_RAM_OFFSET (19 * 4)
#define CH_10_T_TIME_RAM_OFFSET  (20 * 4)
#define CH_11_PULSE_TIME_RAM_OFFSET (21 * 4)
#define CH_11_T_TIME_RAM_OFFSET  (22 * 4)
#define CH_12_PULSE_TIME_RAM_OFFSET (23 * 4)
#define CH_12_T_TIME_RAM_OFFSET  (24 * 4)

#define MAX_CYCLE_TIME_OFFSET (25 * 4)

#define RCIN_RING_HEAD_OFFSET 0x1000
#define RCIN_RING_TAIL_OFFSET 0x1002
#define RCIN_RINGBUFFER_RAM_OFFSET 0x1004

// RCOut pins
#define RC_CH_1_PIN r30.t10
#define RC_CH_2_PIN r30.t8
#define RC_CH_3_PIN r30.t11
#define RC_CH_4_PIN r30.t9
#define RC_CH_5_PIN r30.t7
#define RC_CH_6_PIN r30.t6
#define RC_CH_7_PIN r30.t5
#define RC_CH_8_PIN r30.t4
#define RC_CH_9_PIN r30.t3
#define RC_CH_10_PIN r30.t2
#define RC_CH_11_PIN r30.t1
#define RC_CH_12_PIN r30.t0
```

```
// RCOut enable bits
#define RC_CH_1_ENABLE register.ch_enable.t0
#define RC_CH_2_ENABLE register.ch_enable.t1
#define RC_CH_3_ENABLE register.ch_enable.t2
#define RC_CH_4_ENABLE register.ch_enable.t3
#define RC_CH_5_ENABLE register.ch_enable.t4
#define RC_CH_6_ENABLE register.ch_enable.t5
#define RC_CH_7_ENABLE register.ch_enable.t6
#define RC_CH_8_ENABLE register.ch_enable.t7
#define RC_CH_9_ENABLE register.ch_enable.t8
#define RC_CH_10_ENABLE register.ch_enable.t9
#define RC_CH_11_ENABLE register.ch_enable.t10
#define RC_CH_12_ENABLE register.ch_enable.t11

// Register struct
.struct RegisterStruct
   .u32 ch_enable
   .u32 ch_1_next_time
   .u32 ch_2_next_time
   .u32 ch_3_next_time
   .u32 ch_4_next_time
   .u32 ch_5_next_time
   .u32 ch_6_next_time
   .u32 ch_7_next_time
   .u32 ch_8_next_time
   .u32 ch_9_next_time
   .u32 ch_10_next_time
   .u32 ch_11_next_time
   .u32 ch_12_next_time
   .u32 time
   .u32 time_max
   .u32 time_cycle
   .u32 rcin_ram_pointer
   .u32 rcin_ram_pointer_index
   .u32 rcin_ram_pointer_index_max
   .u32 rcin_ram_pointer_head
   .u32 rcin_ram_pointer_tail
   .u32 temp
   .u32 temp1
   .u32 test
 .ends
 .assign RegisterStruct, R4, *, register

.macro RCOUT_PWM
.mparam RC_CH_X_PIN, CH_X_NEXT_TIME, CH_X_ENABLE, CH_X_PULSE_TIME_RAM_OFFSET,
CH_X_T_TIME_RAM_OFFSET
pwm:
   // Handle arithmetic and counter overflow, check if there is something to do
   sub register.temp, CH_X_NEXT_TIME, register.time
   mov register.temp1, 0xF0000000
   qbgt pwmend, register.temp, register.temp1

      mov CH_X_NEXT_TIME, register.time

      // Set pin or clear pin?
      qbbs pwmclear, RC_CH_X_PIN

         // Load pulse duration
         lbco register.temp, RAM, CH_X_PULSE_TIME_RAM_OFFSET, 4

         // Calculate time to next event
         add CH_X_NEXT_TIME, CH_X_NEXT_TIME, register.temp
```

```
        // Check if channel is enabled
        qbbc pwmend, CH_X_ENABLE

          // Set pin
          set RC_CH_X_PIN
        jmp pwmend

      pwmclear:
        // Load pulse time
        lbco register.temp1, RAM, CH_X_PULSE_TIME_RAM_OFFSET, 4

        // Load T time
        lbco register.temp, RAM, CH_X_T_TIME_RAM_OFFSET, 4

        // Calculate time to next event (T - pulse duration)
        sub register.temp, register.temp, register.temp1
        add CH_X_NEXT_TIME, CH_X_NEXT_TIME, register.temp

        // Clear pin
        clr RC_CH_X_PIN
pwmend:
.endm

.macro RCIN_ECAP_INIT
  // Initialize ECAP
  mov register.temp, (1 << ECAP_CTRRST1) | (1 << ECAP_CAP1POL) | (1 << ECAP_CTRRST2) | (1 <<
ECAP_CAPLDEN)
  sbco register.temp, ECAP, ECAP_ECCTL1, 4
  mov register.temp, (1 << ECAP_STOP_WRAP) | (1 << ECAP_TSCTRSTOP) | (2 << ECAP_SYNCO_SEL)
  sbco register.temp, ECAP, ECAP_ECCTL2, 4
.endm

.macro RCIN_ECAP
  // New value?
  lbco register.temp, ECAP, ECAP_ECFLG, 4
  qbbc rcin_ecap_end, register.temp.t2

    // Copy S0 and S1 duration to temp and temp1
    lbco register.temp, ECAP, ECAP_CAP1, 8

    // Copy S0 and S1 duration to RAM
    sbbo register.temp, register.rcin_ram_pointer, 0, 8

    // Clear event flags
    mov register.temp, (1 << ECAP_CEVT1) | (1 << ECAP_CEVT2)
    sbco register.temp, ECAP, ECAP_ECCLR, 4

    // Set new tail value
    RCIN_WRITE_TAIL register.rcin_ram_pointer_index

    // Update pointer
    add register.rcin_ram_pointer_index, register.rcin_ram_pointer_index, 1
    add register.rcin_ram_pointer, register.rcin_ram_pointer, 8

    // Check end of ringbuffer
    qblt rcin_ecap_end, register.rcin_ram_pointer_index_max, register.rcin_ram_pointer_index
      mov register.rcin_ram_pointer, RCIN_RINGBUFFER_RAM_OFFSET
      mov register.rcin_ram_pointer_index, 0
rcin_ecap_end:
.endm

.macro RCIN_WRITE_HEAD
```

```
.mparam RCIN_HEAD
  sbbo RCIN_HEAD, register.rcin_ram_pointer_head, 0, 2
.endm

.macro RCIN_WRITE_TAIL
.mparam RCIN_TAIL
  sbbo RCIN_TAIL, register.rcin_ram_pointer_tail, 0, 2
.endm

.macro MAX_CYCLE_TIME
  lbco register.temp1, IEP, IEP_TMR_CNT, 4
  sub register.temp, register.temp1, register.time_cycle
  mov register.time_cycle, register.temp1
  max register.time_max, register.time_max, register.temp
  sbco register.time_max, RAM, MAX_CYCLE_TIME_OFFSET, 4
.endm

.macro INIT
  // Reset PWM pins
  mov r30, 0x0

  // Clear register
  zero &register, SIZE(register)

  // Initialize max cycle time
  mov register.temp, 0
  sbco register.temp, RAM, MAX_CYCLE_TIME_OFFSET, 4

  // Initialize ringbuffer
  mov register.rcin_ram_pointer, RCIN_RINGBUFFER_RAM_OFFSET
  mov register.rcin_ram_pointer_index_max, RCIN_RINGBUFFERSIZE
  mov register.rcin_ram_pointer_head, RCIN_RING_HEAD_OFFSET
  mov register.rcin_ram_pointer_tail, RCIN_RING_TAIL_OFFSET
  mov register.temp, 0
  RCIN_WRITE_HEAD register.temp
  RCIN_WRITE_TAIL register.temp

  // Load balancing
  mov register.ch_1_next_time, 1000
  mov register.ch_2_next_time, 2000
  mov register.ch_3_next_time, 3000
  mov register.ch_4_next_time, 4000
  mov register.ch_5_next_time, 5000
  mov register.ch_6_next_time, 6000
  mov register.ch_7_next_time, 7000
  mov register.ch_8_next_time, 8000
  mov register.ch_9_next_time, 9000
  mov register.ch_10_next_time, 10000
  mov register.ch_11_next_time, 11000
  mov register.ch_12_next_time, 12000

  // Disable all PWMs
  mov register.ch_enable, 0x0
  sbco register.ch_enable, RAM, CH_ENABLE_RAM_OFFSET, 4

  // Initialize PWM pulse (920us)
  mov register.temp, PWM_PULSE_DEFAULT
  sbco register.temp, RAM, CH_1_PULSE_TIME_RAM_OFFSET, 4
  sbco register.temp, RAM, CH_2_PULSE_TIME_RAM_OFFSET, 4
  sbco register.temp, RAM, CH_3_PULSE_TIME_RAM_OFFSET, 4
  sbco register.temp, RAM, CH_4_PULSE_TIME_RAM_OFFSET, 4
  sbco register.temp, RAM, CH_5_PULSE_TIME_RAM_OFFSET, 4
  sbco register.temp, RAM, CH_6_PULSE_TIME_RAM_OFFSET, 4
```

```
    sbco register.temp, RAM, CH_7_PULSE_TIME_RAM_OFFSET, 4
    sbco register.temp, RAM, CH_8_PULSE_TIME_RAM_OFFSET, 4
    sbco register.temp, RAM, CH_9_PULSE_TIME_RAM_OFFSET, 4
    sbco register.temp, RAM, CH_10_PULSE_TIME_RAM_OFFSET, 4
    sbco register.temp, RAM, CH_11_PULSE_TIME_RAM_OFFSET, 4
    sbco register.temp, RAM, CH_12_PULSE_TIME_RAM_OFFSET, 4

    // Initialize PWM frequency (50Hz)
    mov register.temp, PWM_FREQ_DEFAULT
    sbco register.temp, RAM, CH_1_T_TIME_RAM_OFFSET, 4
    sbco register.temp, RAM, CH_2_T_TIME_RAM_OFFSET, 4
    sbco register.temp, RAM, CH_3_T_TIME_RAM_OFFSET, 4
    sbco register.temp, RAM, CH_4_T_TIME_RAM_OFFSET, 4
    sbco register.temp, RAM, CH_5_T_TIME_RAM_OFFSET, 4
    sbco register.temp, RAM, CH_6_T_TIME_RAM_OFFSET, 4
    sbco register.temp, RAM, CH_7_T_TIME_RAM_OFFSET, 4
    sbco register.temp, RAM, CH_8_T_TIME_RAM_OFFSET, 4
    sbco register.temp, RAM, CH_9_T_TIME_RAM_OFFSET, 4
    sbco register.temp, RAM, CH_10_T_TIME_RAM_OFFSET, 4
    sbco register.temp, RAM, CH_11_T_TIME_RAM_OFFSET, 4
    sbco register.temp, RAM, CH_12_T_TIME_RAM_OFFSET, 4

    // Initialize counter (1 step = 5ns)
    mov register.temp, 1 << IEP_DEFAULT_INC
    sbco register.temp, IEP, IEP_TMR_GLB_CFG, 4

    // Reset counter
    mov register.temp, 0xffffffff
    sbco register.temp, IEP, IEP_TMR_CNT, 4

    // Start counter
    lbco register.temp, IEP, IEP_TMR_GLB_CFG, 4
    or register.temp, register.temp, 1 << IEP_CNT_ENABLE
    sbco register.temp, IEP, IEP_TMR_GLB_CFG, 4
.endm

.origin 0
init:
    INIT
    RCIN_ECAP_INIT
mainloop:
    lbco register.ch_enable, RAM, CH_ENABLE_RAM_OFFSET, 4
    lbco register.time, IEP, IEP_TMR_CNT, 4
    RCOUT_PWM RC_CH_1_PIN, register.ch_1_next_time, RC_CH_1_ENABLE,
CH_1_PULSE_TIME_RAM_OFFSET, CH_1_T_TIME_RAM_OFFSET
    RCOUT_PWM RC_CH_2_PIN, register.ch_2_next_time, RC_CH_2_ENABLE,
CH_2_PULSE_TIME_RAM_OFFSET, CH_2_T_TIME_RAM_OFFSET
    RCOUT_PWM RC_CH_3_PIN, register.ch_3_next_time, RC_CH_3_ENABLE,
CH_3_PULSE_TIME_RAM_OFFSET, CH_3_T_TIME_RAM_OFFSET
    RCOUT_PWM RC_CH_4_PIN, register.ch_4_next_time, RC_CH_4_ENABLE,
CH_4_PULSE_TIME_RAM_OFFSET, CH_4_T_TIME_RAM_OFFSET
    RCOUT_PWM RC_CH_5_PIN, register.ch_5_next_time, RC_CH_5_ENABLE,
CH_5_PULSE_TIME_RAM_OFFSET, CH_5_T_TIME_RAM_OFFSET
    RCOUT_PWM RC_CH_6_PIN, register.ch_6_next_time, RC_CH_6_ENABLE,
CH_6_PULSE_TIME_RAM_OFFSET, CH_6_T_TIME_RAM_OFFSET
    RCOUT_PWM RC_CH_7_PIN, register.ch_7_next_time, RC_CH_7_ENABLE,
CH_7_PULSE_TIME_RAM_OFFSET, CH_7_T_TIME_RAM_OFFSET
    RCOUT_PWM RC_CH_8_PIN, register.ch_8_next_time, RC_CH_8_ENABLE,
CH_8_PULSE_TIME_RAM_OFFSET, CH_8_T_TIME_RAM_OFFSET
    RCOUT_PWM RC_CH_9_PIN, register.ch_9_next_time, RC_CH_9_ENABLE,
CH_9_PULSE_TIME_RAM_OFFSET, CH_9_T_TIME_RAM_OFFSET
    RCOUT_PWM RC_CH_10_PIN, register.ch_10_next_time, RC_CH_10_ENABLE,
CH_10_PULSE_TIME_RAM_OFFSET, CH_10_T_TIME_RAM_OFFSET
```

```
   RCOUT_PWM RC_CH_11_PIN, register.ch_11_next_time, RC_CH_11_ENABLE,
CH_11_PULSE_TIME_RAM_OFFSET, CH_11_T_TIME_RAM_OFFSET
   RCOUT_PWM RC_CH_12_PIN, register.ch_12_next_time, RC_CH_12_ENABLE,
CH_12_PULSE_TIME_RAM_OFFSET, CH_12_T_TIME_RAM_OFFSET
   RCIN_ECAP
//   MAX_CYCLE_TIME
jmp mainloop
```