Table of Contents

PWM — Pulse width modulation

Last Updated Sep 10, 2025 | 44 minute read | Summarize | # nRF54 Series # nRF54L15 # nRF54L10 # nRF54L05 # Datasheet

The pulse width modulation peripheral (PWM) enables the generation of pulse width modulated signals on GPIO. The peripheral implements a counter with up-count mode and up-and-down-count mode, consisting of four PWM channels that can drive assigned GPIO pins.

The main features of PWM are the following:

- Programmable PWM frequency
- Up to four PWM channels with individual polarity and duty cycle values
- Edge or center-aligned pulses across PWM channels
- Multiple duty cycle arrays (sequences) defined in RAM
- Autonomous and glitch-free update of duty cycle values directly from memory through EasyDMA (no CPU involvement)
- Change of polarity, duty cycle, and base frequency on every PWM period
- RAM sequences can be repeated or connected into loops

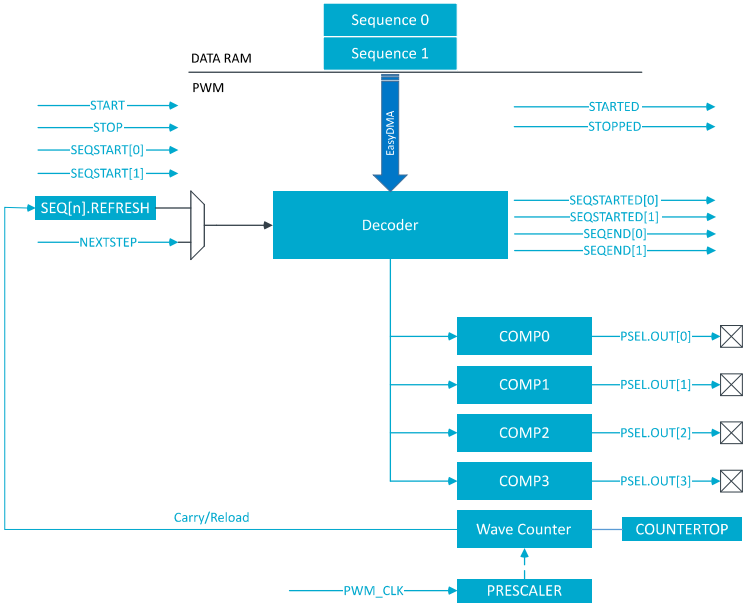

Figure 1. PWM module

Wave counter

The wave counter is responsible for generating the pulses at a duty cycle that depends on the compare values, and at a frequency that depends on COUNTERTOP.

There is one common 15-bit counter with four compare channels. Thus, all four channels will share the same period (PWM frequency), but can have individual duty cycle and polarity. The polarity is set by the most significant bit (MSB) of the value read from RAM (see figure Decoder memory access modes). When the MSB bit is high (FallingEdge polarity), OUT[n] starts high to become low during the given PWM cycle, whereas the inverse occurs for RisingEdge polarity. Whether the counter counts up, or up and down, is controlled by the MODE register.

The timer top value is controlled by the COUNTERTOP register. This register value, in conjunction with the selected PRESCALER of the PWM_CLK, will result in a given PWM period. A COUNTERTOP value smaller than the compare setting will result in a state where no PWM edges are generated. OUT[n] is held high, given that the polarity is set to FallingEdge. All compare registers are internal and can only be configured through decoder presented later. COUNTERTOP can be safely written at any time.

Sampling follows the START task. If DECODER.LOAD=WaveForm, the register value is ignored and taken from RAM instead (see section Decoder with EasyDMA for more details). If DECODER.LOAD is anything else than the WaveForm, it is sampled following a STARTSEQ[n] task and when loading a new value from RAM during a sequence playback.

The following figure shows the counter operating in up mode (MODE=PWM_MODE_Up), with two PWM channels with the same frequency but different duty cycle:

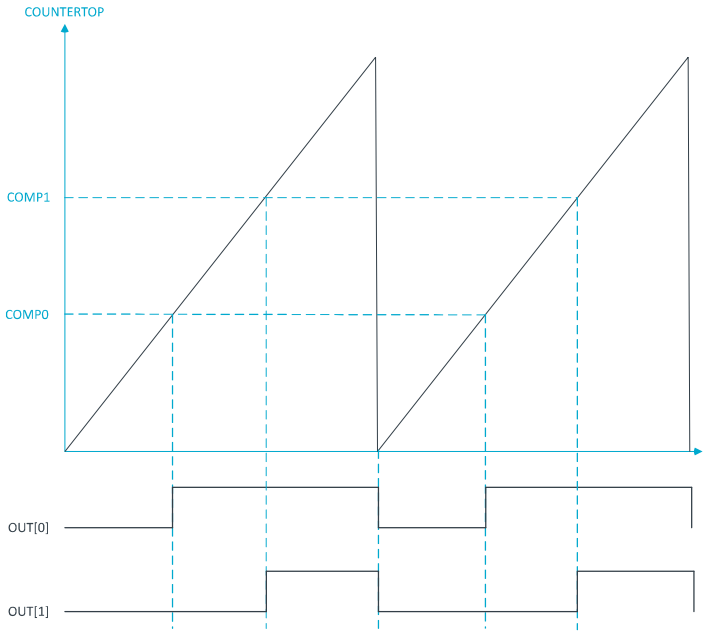

Figure 2. PWM counter in up mode example - RisingEdge polarity

The counter is automatically reset to zero when COUNTERTOP is reached and OUT[n] will invert. OUT[n] is held low if the compare value is 0 and held high if set to COUNTERTOP, given that the polarity is set to FallingEdge. Counter running in up mode results in pulse widths that are edge-aligned. The following is the code for the counter in up mode example:

Explain this code

```
uint16_t pwm_seq[4] = {PWM_CH0_DUTY, PWM_CH1_DUTY, PWM_CH2_DUTY, PWM_CH3_DUTY};
NRF_PWM0->PSEL.OUT[0] = (first_port << PWM_PSEL_OUT_PORT_Pos) |
                        (first_pin << PWM_PSEL_OUT_PIN_Pos) |
                        (PWM_PSEL_OUT_CONNECT_Connected <<
                                            PWM_PSEL_OUT_CONNECT_Pos);
NRF_PWM0->PSEL.OUT[1] = (second_port << PWM_PSEL_OUT_PORT_Pos) |
                        (second_pin << PWM_PSEL_OUT_PIN_Pos) |
                        (PWM_PSEL_OUT_CONNECT_Connected <<
                                            PWM_PSEL_OUT_CONNECT_Pos);
NRF_PWM0->ENABLE      = (PWM_ENABLE_ENABLE_Enabled << PWM_ENABLE_ENABLE_Pos);
NRF_PWM0->MODE        = (PWM_MODE_UPDOWN_Up << PWM_MODE_UPDOWN_Pos);
NRF_PWM0->PRESCALER   = (PWM_PRESCALER_PRESCALER_DIV_1 <<
                                            PWM_PRESCALER_PRESCALER_Pos);
NRF_PWM0->COUNTERTOP  = (16000 << PWM_COUNTERTOP_COUNTERTOP_Pos); //1 msec
NRF_PWM0->LOOP        = (PWM_LOOP_CNT_Disabled << PWM_LOOP_CNT_Pos);
NRF_PWM0->DECODER   = (PWM_DECODER_LOAD_Individual << PWM_DECODER_LOAD_Pos) |
                      (PWM_DECODER_MODE_RefreshCount << PWM_DECODER_MODE_Pos);
NRF_PWM0->DMA.SEQ[0].PTR    = ((uint32_t)(pwm_seq) << PWM_DMA_SEQ_PTR_PTR_Pos);
NRF_PWM0->DMA.SEQ[0].MAXCNT = (sizeof(pwm_seq) << PWM_DMA_SEQ_MAXCNT_MAXCNT_Pos);
NRF_PWM0->SEQ[0].REFRESH  = 0;
NRF_PWM0->SEQ[0].ENDDELAY = 0;
NRF_PWM0->TASKS_DMA.SEQ[0].START = 1;
```

When the counter is running in up mode, the following formula can be used to compute the PWM period and the step size:

PWM period: $T_{PWM(Up)} = T_{PWM\_CLK} * COUNTERTOP$

Step width/Resolution: $T_{steps} = T_{PWM\_CLK}$

The following figure shows the counter operating in up-and-down mode (MODE=PWM_MODE_UpAndDown), with two PWM channels with the same frequency but different duty cycle and output polarity:

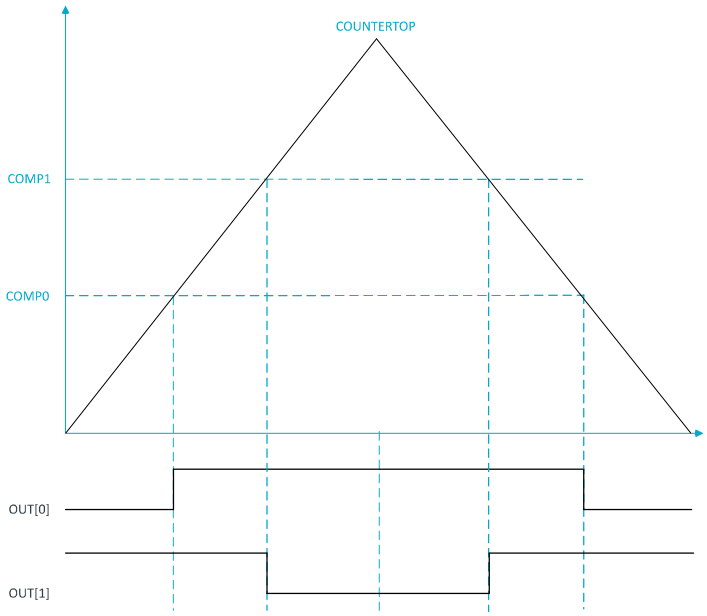

Figure 3. PWM counter in up-and-down mode example

The counter starts decrementing to zero when COUNTERTOP is reached and will invert the OUT[n] when compare value is hit for the second time. This results in a set of pulses that are center-aligned. The following is the code for the counter in up-and-down mode example:

Explain this code

```
uint16_t pwm_seq[4] = {PWM_CH0_DUTY, PWM_CH1_DUTY, PWM_CH2_DUTY, PWM_CH3_DUTY};
NRF_PWM0->PSEL.OUT[0] = (first_port << PWM_PSEL_OUT_PORT_Pos) |
                        (first_pin << PWM_PSEL_OUT_PIN_Pos) |
                        (PWM_PSEL_OUT_CONNECT_Connected <<
                                            PWM_PSEL_OUT_CONNECT_Pos);
NRF_PWM0->PSEL.OUT[1] = (second_pin << PWM_PSEL_OUT_PIN_Pos) |
                        (PWM_PSEL_OUT_CONNECT_Connected <<
                                            PWM_PSEL_OUT_CONNECT_Pos);
NRF_PWM0->ENABLE      = (PWM_ENABLE_ENABLE_Enabled << PWM_ENABLE_ENABLE_Pos);
NRF_PWM0->MODE        = (PWM_MODE_UPDOWN_UpAndDown << PWM_MODE_UPDOWN_Pos);
NRF_PWM0->PRESCALER   = (PWM_PRESCALER_PRESCALER_DIV_1 <<
                                            PWM_PRESCALER_PRESCALER_Pos);
NRF_PWM0->COUNTERTOP  = (16000 << PWM_COUNTERTOP_COUNTERTOP_Pos); //1 msec
NRF_PWM0->LOOP        = (PWM_LOOP_CNT_Disabled << PWM_LOOP_CNT_Pos);
NRF_PWM0->DECODER   = (PWM_DECODER_LOAD_Individual << PWM_DECODER_LOAD_Pos) |
                      (PWM_DECODER_MODE_RefreshCount << PWM_DECODER_MODE_Pos);
NRF_PWM0->DMA.SEQ[0].PTR    = ((uint32_t)(pwm_seq) << PWM_DMA_SEQ_PTR_PTR_Pos);
NRF_PWM0->DMA.SEQ[0].MAXCNT = (sizeof(pwm_seq) << PWM_DMA_SEQ_MAXCNT_MAXCNT_Pos);
NRF_PWM0->SEQ[0].REFRESH  = 0;
NRF_PWM0->SEQ[0].ENDDELAY = 0;
NRF_PWM0->TASKS_DMA.SEQ[0].START = 1;
```

When the counter is running in up-and-down mode, the following formula can be used to compute the PWM period and the step size:

$T_{PWM(Up\ And\ Down)} = T_{PWM\_CLK} * 2 * COUNTERTOP$

Step width/Resolution: $T_{steps} = T_{PWM\_CLK} * 2$

Decoder with EasyDMA

The decoder uses EasyDMA to take PWM parameters stored in RAM and update the internal compare registers of the wave counter, based on the mode of operation.

PWM parameters are organized into a sequence containing at least one half word (16 bit). Its most significant bit[15] denotes the polarity of the OUT[n] while bit[14:0] is the 15-bit compare value.

Bit number	31 30 29 28	27 26 25 24	23 22 21 20	19 18 17 16	15 14 13 12	11 10 9 8	7 6 5 4	3 2 1 0
Id					B	A A A	A A A A	A A A A
Reset 0x00000000	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0

Id	RW	Field	Value Id	Value	Description
A	RW	COMPARE			Duty cycle setting - value loaded to internal compare register
B	RW	POLARITY			Edge polarity of GPIO.
			RisingEdge	0	First edge within the PWM period is rising
			FallingEdge	1	First edge within the PWM period is falling

The DECODER register controls how the RAM content is interpreted and loaded into the internal compare registers. The LOAD field controls if the RAM values are loaded to all compare channels, or to update a group or all channels with individual values. The following figure illustrates how parameters stored in RAM are organized and routed to various compare channels in different modes:

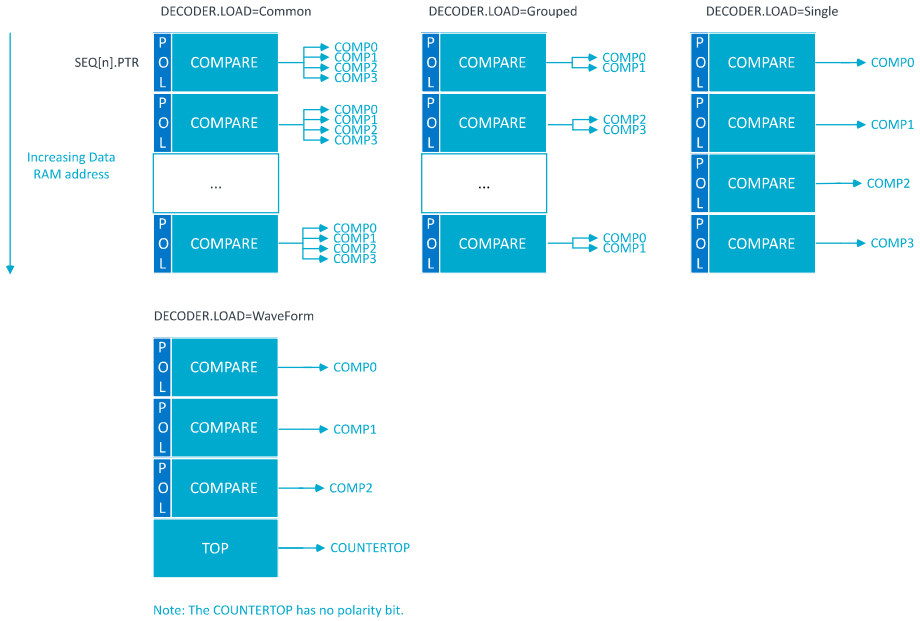

Figure 4. Decoder memory access modes

A special mode of operation is available when DECODER.LOAD is set to WaveForm. In WaveForm mode, up to three PWM channels can be enabled - OUT[0] to OUT[2]. In RAM, four values are loaded at a time: the first, second and third location are used to load the values, and the fourth RAM location is used to load the COUNTERTOP register. This way one can have up to three PWM channels with a frequency base that changes on a per PWM period basis. This mode of operation is useful for arbitrary wave form generation in applications, such as LED lighting.

The register SEQ[n].REFRESH=N (one per sequence n=0 or 1) will instruct a new RAM stored pulse width value on every $(N+1)^{th}$ PWM period. Setting the register to zero will result in a new duty cycle update every PWM period, as long as the minimum PWM period is observed.

Note that registers SEQ[n].REFRESH and SEQ[n].ENDDELAY are ignored when DECODER.MODE=NextStep. The next value is loaded upon every received NEXTSTEP task.

SEQ[n].PTR is the pointer used to fetch COMPARE values from RAM. If the SEQ[n].PTR is not pointing to a RAM region, an EasyDMA transfer may result in a HardFault or RAM corruption. See Memory for more information about the different memory regions. After the SEQ[n].PTR is set to the desired RAM location, the SEQ[n].MAXCNT register must be set to the number of bytes in the sequence. It is important to observe that the Grouped mode requires one half word #concept_wxj_hnw_nr__load_data_fn per group, while the Single mode requires one half word #concept_wxj_hnw_nr__load_data_fn per channel, thus increasing the RAM size occupation. If PWM generation is not running when the DMA.SEQ[n].START task is triggered, the task will load the first value from RAM and then start the PWM generation. A SEQSTARTED[n] event is generated as soon as the EasyDMA has read the first PWM parameter from RAM and the wave counter has started executing it. When LOOP.MAXCNT=0, sequence n=0 or 1 is played back once. After the last value in the sequence has been loaded and started executing, a SEQEND[n] event is generated. The PWM generation will then continue with the output defined in the IDLEOUT register. The following figure illustrates an example of a simple playback.

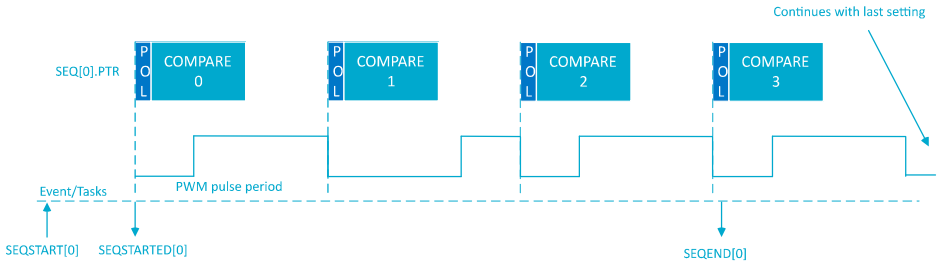

Figure 5. Simple sequence example

The following source code is used for configuration and timing details in a sequence where only sequence 0 is used and only run once with a new PWM duty cycle for each period.

Explain this code

```
NRF_PWM0->PSEL.OUT[0] = (first_port << PWM_PSEL_OUT_PORT_Pos) |
                        (first_pin << PWM_PSEL_OUT_PIN_Pos) |
                        (PWM_PSEL_OUT_CONNECT_Connected <<
                                            PWM_PSEL_OUT_CONNECT_Pos);
NRF_PWM0->ENABLE      = (PWM_ENABLE_ENABLE_Enabled << PWM_ENABLE_ENABLE_Pos);
NRF_PWM0->MODE        = (PWM_MODE_UPDOWN_Up << PWM_MODE_UPDOWN_Pos);
NRF_PWM0->PRESCALER   = (PWM_PRESCALER_PRESCALER_DIV_1 <<
                                            PWM_PRESCALER_PRESCALER_Pos);
NRF_PWM0->COUNTERTOP  = (16000 << PWM_COUNTERTOP_COUNTERTOP_Pos); //1 msec
NRF_PWM0->LOOP        = (PWM_LOOP_CNT_Disabled << PWM_LOOP_CNT_Pos);
NRF_PWM0->DECODER   = (PWM_DECODER_LOAD_Common << PWM_DECODER_LOAD_Pos) |
                      (PWM_DECODER_MODE_RefreshCount << PWM_DECODER_MODE_Pos);
NRF_PWM0->DMA.SEQ[0].PTR  = ((uint32_t)(seq0_ram) << PWM_DMA_SEQ_PTR_PTR_Pos);
NRF_PWM0->DMA.SEQ[0].MAXCNT  = (sizeof(seq0_ram) << PWM_DMA_SEQ_MAXCNT_MAXCNT_Pos);
NRF_PWM0->SEQ[0].REFRESH  = 0;
NRF_PWM0->SEQ[0].ENDDELAY = 0;
NRF_PWM0->TASKS_DMA.SEQ[0].START = 1;
```

To completely stop the PWM generation and force the associated pins to a defined state, a STOP task can be triggered at any time. A STOPPED event is generated when the PWM generation has stopped at the end of the currently running PWM period, and the pins go into their idle state as defined by the IDLEOUT register. PWM generation can then only be restarted through a DMA.SEQ[n].START task. DMA.SEQ[n].START will resume PWM generation after having loaded the first value from the RAM buffer defined in the SEQ[n].PTR register.

The following table indicates when specific registers get sampled by the hardware. Care should be taken when updating these registers to avoid that values are applied earlier than expected.

Register	Taken into account by hardware	Recommended (safe) update
SEQ[n].PTR	When sending the DMA.SEQ[n].START task	After having received the SEQSTARTED[n] event
SEQ[n].MAXCNT	When sending the DMA.SEQ[n].START task	After having received the SEQSTARTED[n] event
SEQ[0].ENDDELAY	When sending the SEQSTART[0] task Every time a new value from sequence [0] has been loaded from RAM and gets applied to the Wave Counter (indicated by the PWMPERIODEND event)	Before starting sequence [0] through a SEQSTART[0] task When no more value from sequence [0] gets loaded from RAM (indicated by the SEQEND[0] event) At any time during sequence [1] (which starts when the SEQSTARTED[1] event is generated)
SEQ[1].ENDDELAY	When sending the SEQSTART[1] task Every time a new value from sequence [1] has been loaded from RAM and gets applied to the Wave Counter (indicated by the PWMPERIODEND event)	Before starting sequence [1] through a SEQSTART[1] task When no more value from sequence [1] gets loaded from RAM (indicated by the SEQEND[1] event) At any time during sequence [0] (which starts when the SEQSTARTED[0] event is generated)
SEQ[0].REFRESH	When sending the SEQSTART[0] task Every time a new value from sequence [0] has been loaded from RAM and gets applied to the Wave Counter (indicated by the PWMPERIODEND event)	Before starting sequence [0] through a SEQSTART[0] task At any time during sequence [1] (which starts when the SEQSTARTED[1] event is generated)
SEQ[1].REFRESH	When sending the SEQSTART[1] task Every time a new value from sequence [1] has been loaded from RAM and gets applied to the Wave Counter (indicated by the PWMPERIODEND event)	Before starting sequence [1] through a SEQSTART[1] task At any time during sequence [0] (which starts when the SEQSTARTED[0] event is generated)
COUNTERTOP	In DECODER.LOAD=WaveForm: this register is ignored. In all other LOAD modes: at the end of current PWM period (indicated by the PWMPERIODEND event)	Before starting PWM generation through a DMA.SEQ[n].START task After a STOP task has been triggered, and the STOPPED event has been received.
MODE	Immediately	Before starting PWM generation through a DMA.SEQ[n].START task After a STOP task has been triggered, and the STOPPED event has been received.
DECODER	Immediately	Before starting PWM generation through a DMA.SEQ[n].START task After a STOP task has been triggered, and the STOPPED event has been received.
PRESCALER	Immediately	Before starting PWM generation through a DMA.SEQ[n].START task After a STOP task has been triggered, and the STOPPED event has been received.
LOOP	Immediately	Before starting PWM generation through a DMA.SEQ[n].START task After a STOP task has been triggered, and the STOPPED event has been received.
PSEL.OUT[n]	Immediately	Before enabling the PWM instance through the ENABLE register

Table 1. When to safely update PWM registers

Note: SEQ[n].REFRESH and SEQ[n].ENDDELAY are ignored at the end of a complex sequence, indicated by a LOOPSDONE event. The reason for this is that the last value loaded from RAM is maintained until further action from software (restarting a new sequence, or stopping PWM generation).

The following figure shows a more complex example using the register LOOP.

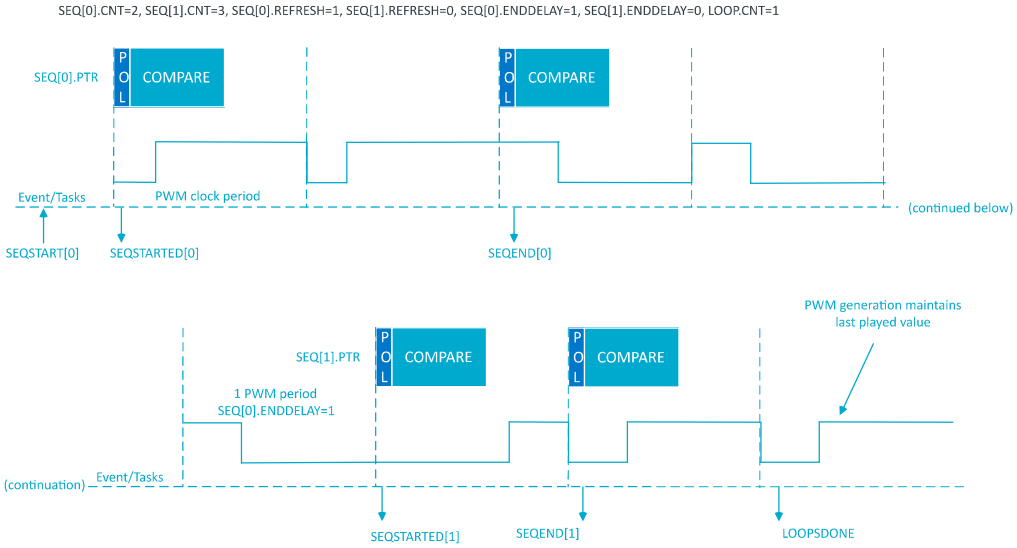

Figure 6. Example using two sequences

In this case, an automated playback takes place, consisting of SEQ[0], delay 0, SEQ[1], delay 1, then again SEQ[0], etc. The user can choose to start a complex playback with SEQ[0] or SEQ[1] through sending the SEQSTART[0] or SEQSTART[1] task. The complex playback always ends with delay 1.

The two sequences 0 and 1 are defined by the addresses of value tables in RAM (pointed to by SEQ[n].PTR) and the buffer size (SEQ[n].MAXCNT). The rate at which a new value is loaded is defined individually for each sequence by SEQ[n].REFRESH. The chaining of sequence 1 following the sequence 0 is implicit, the LOOP.CNT register allows the chaining of sequence 1 to sequence 0 for a determined number of times. In other words, it allows to repeat a complex sequence a number of times in a fully automated way.

In the following code example, sequence 0 is defined with SEQ[0].REFRESH set to 1, meaning that a new PWM duty cycle is pushed every second PWM period. This complex sequence is started with the SEQSTART[0] task, so SEQ[0] is played first. Since SEQ[0].ENDDELAY=1 there will be one PWM period delay between last period on sequence 0 and the first period on sequence 1. Since SEQ[1].ENDDELAY=0 there is no delay 1, so SEQ[0] would be started immediately after the end of SEQ[1]. However, as LOOP.CNT is 1, the playback stops after having played SEQ[1] only once, and both SEQEND[1] and LOOPSDONE are generated (their order is not guaranteed in this case).

Explain this code

```
NRF_PWM0->PSEL.OUT[0] = (first_port << PWM_PSEL_OUT_PORT_Pos) |
                        (first_pin << PWM_PSEL_OUT_PIN_Pos) |
                        (PWM_PSEL_OUT_CONNECT_Connected <<
                                          PWM_PSEL_OUT_CONNECT_Pos);
NRF_PWM0->ENABLE      = (PWM_ENABLE_ENABLE_Enabled << PWM_ENABLE_ENABLE_Pos);
NRF_PWM0->MODE        = (PWM_MODE_UPDOWN_Up << PWM_MODE_UPDOWN_Pos);
NRF_PWM0->PRESCALER   = (PWM_PRESCALER_PRESCALER_DIV_1 <<
                                        PWM_PRESCALER_PRESCALER_Pos);
NRF_PWM0->COUNTERTOP  = (16000 << PWM_COUNTERTOP_COUNTERTOP_Pos); //1 msec
NRF_PWM0->LOOP        = (1 << PWM_LOOP_CNT_Pos);
NRF_PWM0->DECODER   = (PWM_DECODER_LOAD_Common << PWM_DECODER_LOAD_Pos) |
                      (PWM_DECODER_MODE_RefreshCount << PWM_DECODER_MODE_Pos);
NRF_PWM0->DMA.SEQ[0].PTR   = ((uint32_t)(seq0_ram) << PWM_DMA_SEQ_PTR_PTR_Pos);
NRF_PWM0->DMA.SEQ[0].MAXCNT  = (sizeof(seq0_ram) << PWM_DMA_SEQ_MAXCNT_MAXCNT_Pos);
NRF_PWM0->SEQ[0].REFRESH  = 1;
NRF_PWM0->SEQ[0].ENDDELAY = 1;
NRF_PWM0->DMA.SEQ[1].PTR   = ((uint32_t)(seq1_ram) << PWM_DMA_SEQ_PTR_PTR_Pos);
NRF_PWM0->DMA.SEQ[1].MAXCNT  = (sizeof(seq1_ram) << PWM_DMA_SEQ_MAXCNT_MAXCNT_Pos);
NRF_PWM0->SEQ[1].REFRESH  = 0;
NRF_PWM0->SEQ[1].ENDDELAY = 0;
NRF_PWM0->TASKS_DMA.SEQ[0].START = 1;
```

The decoder can also be configured to asynchronously load new PWM duty cycle. If the DECODER.MODE register is set to NextStep, then the NEXTSTEP task will cause an update of internal compare registers on the next PWM period.

The following figures provide an overview of each part of an arbitrary sequence, in various modes (LOOP.CNT=0 and LOOP.CNT>0). In particular, the following are represented:

- Initial and final duty cycle on the PWM output(s)
- Chaining of SEQ[0] and SEQ[1] if LOOP.CNT>0
- Influence of registers on the sequence
- Events generated during a sequence
- DMA activity (loading of next value and applying it to the output(s))

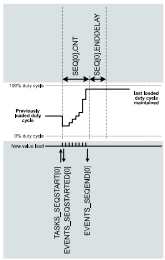

Figure 7. Single shot (LOOP.CNT=0)

Note: The single-shot example also applies to SEQ[1]. Only SEQ[0] is represented for simplicity.

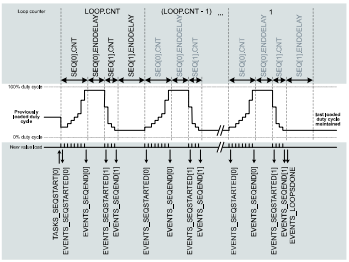

Figure 8. Complex sequence (LOOP.CNT>0) starting with SEQ[0]

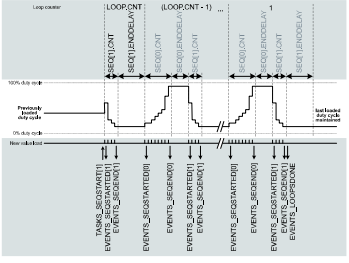

Figure 9. Complex sequence (LOOP.CNT>0) starting with SEQ[1]

Note: If a sequence is in use in a simple or complex sequence, it must have a length of SEQ[n].MAXCNT > 0.

This example shows how the PWM module can be configured to repeat a single sequence until stopped.

Explain this code

```
NRF_PWM0->PSEL.OUT[0] = (first_port << PWM_PSEL_OUT_PORT_Pos) |
                        (first_pin << PWM_PSEL_OUT_PIN_Pos) |
                        (PWM_PSEL_OUT_CONNECT_Connected <<
                                            PWM_PSEL_OUT_CONNECT_Pos);
NRF_PWM0->ENABLE      = (PWM_ENABLE_ENABLE_Enabled << PWM_ENABLE_ENABLE_Pos);
NRF_PWM0->MODE        = (PWM_MODE_UPDOWN_Up << PWM_MODE_UPDOWN_Pos);
NRF_PWM0->PRESCALER   = (PWM_PRESCALER_PRESCALER_DIV_1 <<
                                            PWM_PRESCALER_PRESCALER_Pos);
NRF_PWM0->COUNTERTOP  = (16000 << PWM_COUNTERTOP_COUNTERTOP_Pos); //1 msec
// Enable the shortcut from LOOPSDONE event to DMA.SEQ1.START task for infinite loop
NRF_PWM0->SHORTS      = (PWM_SHORTS_LOOPSDONE_DMA_SEQ1_START_Enabled <<
                                    PWM_SHORTS_LOOPSDONE_DMA_SEQ1_START_Pos);
// LOOP_CNT must be greater than 0 for the LOOPSDONE event to trigger and enable looping
NRF_PWM0->LOOP        = (1 << PWM_LOOP_CNT_Pos);
NRF_PWM0->DECODER     = (PWM_DECODER_LOAD_Common << PWM_DECODER_LOAD_Pos) |
                  (PWM_DECODER_MODE_RefreshCount << PWM_DECODER_MODE_Pos);
// To repeat a single sequence until stopped, it must be configured in SEQ[1]
NRF_PWM0->DMA.SEQ[1].PTR  = ((uint32_t)(seq0_ram)) << PWM_DMA_SEQ_PTR_PTR_Pos;
NRF_PWM0->DMA.SEQ[1].MAXCNT  =(sizeof(seq0_ram) << PWM_DMA_SEQ_MAXCNT_MAXCNT_Pos);
NRF_PWM0->SEQ[1].REFRESH  = 0;
NRF_PWM0->SEQ[1].ENDDELAY = 0;
NRF_PWM0->TASKS_DMA.SEQ[1].START = 1;
```

Limitations

The previous compare value is repeated if the PWM period is shorter than the time it takes for the EasyDMA to retrieve from RAM and update the internal compare registers. This is to ensure a glitch-free operation even for very short PWM periods.

Only SEQ[1] can trigger the LOOPSDONE event upon completion, not SEQ[0]. This requires looping to be enabled (LOOP > 0) and SEQ[1].MAXCNT > 0 when sequence playback starts.

Pin configuration

The OUT[n] (n=0..3) signals associated with each PWM channel are mapped to physical pins according to the configuration of PSEL.OUT[n] registers. If PSEL.OUT[n].CONNECT is set to Disconnected, the associated PWM module signal will not be connected to any physical pins.

Once PWM has been enabled, the PSEL.OUT[n] registers take effect and PWM generation starts from the IDLEOUT register. PWM can then be started and sequences generated.

To ensure correct behavior in the PWM module, the pins that are used must be configured in the GPIO peripheral in the following way before the PWM module is enabled:

PWM signal	PWM pin	Direction	Output value	Comment
OUT[n]	As specified in PSEL.OUT[n] (n=0..3)	Output	0	Idle state defined in GPIO OUT register and the IDLEOUT register

Table 2. Recommended GPIO configuration before starting PWM generation

The idle state of a pin is defined by the OUT register in the GPIO module and the IDLEOUT register, to ensure that the pins used by the PWM module are driven correctly. Both OUT register in the GPIO module and the IDLEOUT register should be set with same value for each PWM channel before enabling the PWM module . When PWM is disabled using the ENABLE register the PWM module stops controlling the GPIO pins, and the corresponding pins are then controlled by the GPIO peripheral.

Only one peripheral can be assigned to drive a particular GPIO pin at a time. Failing to do so may result in unpredictable behavior.

Registers

Instances

Instance	Domain	Base address	TrustZone			Split access	Description
			Map	Att	DMA		
PWM20 : S PWM20 : NS	GLOBAL	0x500D2000 0x400D2000	US	S	SA	No	Pulse width modulation unit PWM20
PWM21 : S PWM21 : NS	GLOBAL	0x500D3000 0x400D3000	US	S	SA	No	Pulse width modulation unit PWM21
PWM22 : S PWM22 : NS	GLOBAL	0x500D4000 0x400D4000	US	S	SA	No	Pulse width modulation unit PWM22

Configuration

Instance	Domain	Configuration
PWM20 : S PWM20 : NS	GLOBAL	Use GPIO port P1 IDLEOUT register is available. EVENTS_COMPAREMATCH events are available. CURRENTAMOUNT register included.
PWM21 : S PWM21 : NS	GLOBAL	Use GPIO port P1 IDLEOUT register is available. EVENTS_COMPAREMATCH events are available. CURRENTAMOUNT register included.
PWM22 : S PWM22 : NS	GLOBAL	Use GPIO port P1 IDLEOUT register is available. EVENTS_COMPAREMATCH events are available. CURRENTAMOUNT register included.

Register overview

Register	Offset	TZ	Description
TASKS_STOP	0x004		Stops PWM pulse generation on all channels at the end of current PWM period, and stops sequence playback
TASKS_NEXTSTEP	0x008		Steps by one value in the current sequence on all enabled channels if DECODER.MODE=NextStep. Does not cause PWM generation to start if not running.

Register	Offset	TZ	Description
TASKS_DMA.SEQ[n].START	0x010		Starts operation using easyDMA to load the values. See peripheral description for operation using easyDMA.
TASKS_DMA.SEQ[n].STOP	0x014		Stops operation using easyDMA. This does not trigger an END event.
SUBSCRIBE_STOP	0x084		Subscribe configuration for task STOP
SUBSCRIBE_NEXTSTEP	0x088		Subscribe configuration for task NEXTSTEP
SUBSCRIBE_DMA.SEQ[n].START	0x090		Subscribe configuration for task START
SUBSCRIBE_DMA.SEQ[n].STOP	0x094		Subscribe configuration for task STOP
EVENTS_STOPPED	0x104		Response to STOP task, emitted when PWM pulses are no longer generated
EVENTS_SEQSTARTED[n]	0x108		First PWM period started on sequence n
EVENTS_SEQEND[n]	0x110		Emitted at end of every sequence n, when last value from RAM has been applied to wave counter
EVENTS_PWMPERIODEND	0x118		Emitted at the end of each PWM period
EVENTS_LOOPSDONE	0x11C		Concatenated sequences have been played the amount of times defined in LOOP.CNT
EVENTS_RAMUNDERFLOW	0x120		Emitted when retrieving from RAM does not complete in time for the PWM module
EVENTS_DMA.SEQ[n].END	0x124		Generated after all MAXCNT bytes have been transferred
EVENTS_DMA.SEQ[n].READY	0x128		Generated when EasyDMA has buffered the .PTR and .MAXCNT registers for the channel, allowing them to be written to prepare for the next sequence.
EVENTS_DMA.SEQ[n].BUSERROR	0x12C		An error occured during the bus transfer.
EVENTS_COMPAREMATCH[n]	0x13C		This event is generated when the compare matches for the compare channel [n].
PUBLISH_STOPPED	0x184		Publish configuration for event STOPPED
PUBLISH_SEQSTARTED[n]	0x188		Publish configuration for event SEQSTARTED[n]
PUBLISH_SEQEND[n]	0x190		Publish configuration for event SEQEND[n]
PUBLISH_PWMPERIODEND	0x198		Publish configuration for event PWMPERIODEND
PUBLISH_LOOPSDONE	0x19C		Publish configuration for event LOOPSDONE
PUBLISH_RAMUNDERFLOW	0x1A0		Publish configuration for event RAMUNDERFLOW
PUBLISH_DMA.SEQ[n].END	0x1A4		Publish configuration for event END
PUBLISH_DMA.SEQ[n].READY	0x1A8		Publish configuration for event READY
PUBLISH_DMA.SEQ[n].BUSERROR	0x1AC		Publish configuration for event BUSERROR
PUBLISH_COMPAREMATCH[n]	0x1BC		Publish configuration for event COMPAREMATCH[n]
SHORTS	0x200		Shortcuts between local events and tasks
INTEN	0x300		Enable or disable interrupt
INTENSET	0x304		Enable interrupt
INTENCLR	0x308		Disable interrupt
INTPEND	0x30C		Pending interrupts
ENABLE	0x500		PWM module enable register
MODE	0x504		Selects operating mode of the wave counter
COUNTERTOP	0x508		Value up to which the pulse generator counter counts
PRESCALER	0x50C		Configuration for PWM_CLK
DECODER	0x510		Configuration of the decoder
LOOP	0x514		Number of playbacks of a loop
IDLEOUT	0x518		Configure the output value on the PWM channel during idle
SEQ[n].REFRESH	0x528		Number of additional PWM periods between samples loaded into compare register
SEQ[n].ENDDELAY	0x52C		Time added after the sequence
PSEL.OUT[n]	0x560		Output pin select for PWM channel n
DMA.SEQ[n].PTR	0x704		RAM buffer start address
DMA.SEQ[n].MAXCNT	0x708		Maximum number of bytes in channel buffer
DMA.SEQ[n].AMOUNT	0x70C		Number of bytes transferred in the last transaction, updated after the END event.
DMA.SEQ[n].CURRENTAMOUNT	0x710		Number of bytes transferred in the current transaction
DMA.SEQ[n].TERMINATEONBUSERROR	0x71C		Terminate the transaction if a BUSERROR event is detected.
DMA.SEQ[n].BUSERRORADDRESS	0x720		Address of transaction that generated the last BUSERROR event.

TASKS_STOP

Address offset: 0x004

Stops PWM pulse generation on all channels at the end of current PWM period, and stops sequence playback

Bit number	31 30 29 28	27 26 25 24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID																										A
Reset 0x00000000	0 0 0 0	0 0 0 0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ID	R/W	Field	Value ID	Value	Description
A	W	TASKS_STOP			Stops PWM pulse generation on all channels at the end of current PWM period, and stops sequence playback
			Trigger	1	Trigger task

TASKS_NEXTSTEP

Address offset: 0x008

Steps by one value in the current sequence on all enabled channels if DECODER.MODE=NextStep. Does not cause PWM generation to start if not running.

Bit number	31 30 29 28	27 26 25 24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID																										A
Reset 0x00000000	0 0 0 0	0 0 0 0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ID	R/W	Field	Value ID	Value	Description
A	W	TASKS_NEXTSTEP			Steps by one value in the current sequence on all enabled channels if DECODER.MODE=NextStep. Does not cause PWM generation to start if not running.
			Trigger	1	Trigger task

TASKS_DMA

Peripheral tasks.

TASKS_DMA.SEQ[n] (n=0..1)

Peripheral tasks.

TASKS_DMA.SEQ[n].START (n=0..1)

Address offset: 0x010 + (n × 0x8)

Starts operation using easyDMA to load the values. See peripheral description for operation using easyDMA.

Bit number	31 30 29 28	27 26 25 24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID																										A
Reset 0x00000000	0 0 0 0	0 0 0 0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ID	R/W	Field	Value ID	Value	Description
A	W	START			Starts operation using easyDMA to load the values. See peripheral description for operation using easyDMA.
			Trigger	1	Trigger task

TASKS_DMA.SEQ[n].STOP (n=0..1)

Address offset: 0x014 + (n × 0x8)

Stops operation using easyDMA. This does not trigger an END event.

Bit number	31 30 29 28	27 26 25 24	23 22 21 20	19 18 17 16	15 14 13 12	11 10 9 8	7 6 5 4	3 2 1 0
ID								A
Reset 0x00000000	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0000	0000

ID	R/W	Field	Value ID	Value	Description
A	W	STOP			Stops operation using easyDMA. This does not trigger an END event.
			Trigger	1	Trigger task

SUBSCRIBE_STOP

Address offset: 0x084

Subscribe configuration for task STOP

Bit number	31 30 29 28	27 26 25 24	23 22 21 20	19 18 17 16	15 14 13 12	11 10 9 8	7 6 5 4	3 2 1 0
ID		B					AAAA	AAAA
Reset 0x00000000	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 00	0000	0000

ID	R/W	Field	Value ID	Value	Description
A	RW	CHIDX		[0..255]	DPPI channel that task STOP will subscribe to
B	RW	EN			
			Disabled	0	Disable subscription
			Enabled	1	Enable subscription

SUBSCRIBE_NEXTSTEP

Address offset: 0x088

Subscribe configuration for task NEXTSTEP

Bit number	31 30 29 28	27 26 25 24	23 22 21 20	19 18 17 16	15 14 13 12	11 10 9 8	7 6 5 4	3 2 1 0
ID		B					AAAA	AAAA
Reset 0x00000000	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 00	0000	0000

ID	R/W	Field	Value ID	Value	Description
A	RW	CHIDX		[0..255]	DPPI channel that task NEXTSTEP will subscribe to
B	RW	EN			
			Disabled	0	Disable subscription
			Enabled	1	Enable subscription

SUBSCRIBE_DMA

Subscribe configuration for tasks

SUBSCRIBE_DMA.SEQ[n] (n=0..1)

Subscribe configuration for tasks

SUBSCRIBE_DMA.SEQ[n].START (n=0..1)

Address offset: 0x090 + (n × 0x8)

Subscribe configuration for task START

Bit number	31 30 29 28	27 26 25 24	23 22 21 20	19 18 17 16	15 14 13 12	11 10 9 8	7 6 5 4	3 2 1 0
ID		B					AAAA	AAAA
Reset 0x00000000	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 00	0000	0000

ID	R/W	Field	Value ID	Value	Description
A	RW	CHIDX		[0..255]	DPPI channel that task START will subscribe to
B	RW	EN			
			Disabled	0	Disable subscription
			Enabled	1	Enable subscription

SUBSCRIBE_DMA.SEQ[n].STOP (n=0..1)

Address offset: 0x094 + (n × 0x8)

Subscribe configuration for task STOP

Bit number	31 30 29 28	27 26 25 24	23 22 21 20	19 18 17 16	15 14 13 12	11 10 9 8	7 6 5 4	3 2 1 0
ID		B					AAAA	AAAA
Reset 0x00000000	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 00	0000	0000

ID	R/W	Field	Value ID	Value	Description
A	RW	CHIDX		[0..255]	DPPI channel that task STOP will subscribe to
B	RW	EN			
			Disabled	0	Disable subscription
			Enabled	1	Enable subscription

EVENTS_STOPPED

Address offset: 0x104

Response to STOP task, emitted when PWM pulses are no longer generated

Bit number				31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
ID				A
Reset 0x00000000				0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

ID	R/W	Field	Value ID	Value	Description
A	RW	EVENTS_STOPPED			Response to STOP task, emitted when PWM pulses are no longer generated
			NotGenerated	0	Event not generated
			Generated	1	Event generated

EVENTS_SEQSTARTED[n] (n=0..1)

Address offset: 0x108 + (n × 0x4)

First PWM period started on sequence n

Bit number				31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
ID				A
Reset 0x00000000				0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

ID	R/W	Field	Value ID	Value	Description
A	RW	EVENTS_SEQSTARTED			First PWM period started on sequence n
			NotGenerated	0	Event not generated
			Generated	1	Event generated

EVENTS_SEQEND[n] (n=0..1)

Address offset: 0x110 + (n × 0x4)

Emitted at end of every sequence n, when last value from RAM has been applied to wave counter

Bit number				31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
ID				A
Reset 0x00000000				0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

ID	R/W	Field	Value ID	Value	Description
A	RW	EVENTS_SEQEND			Emitted at end of every sequence n, when last value from RAM has been applied to wave counter
			NotGenerated	0	Event not generated
			Generated	1	Event generated

EVENTS_PWMPERIODEND

Address offset: 0x118

Emitted at the end of each PWM period

Bit number				31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
ID				A
Reset 0x00000000				0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

ID	R/W	Field	Value ID	Value	Description
A	RW	EVENTS_PWMPERIODEND			Emitted at the end of each PWM period
			NotGenerated	0	Event not generated
			Generated	1	Event generated

EVENTS_LOOPSDONE

Address offset: 0x11C

Concatenated sequences have been played the amount of times defined in LOOP.CNT

This event triggers after the last SEQ[1] completion of the loop, and only if looping was enabled (LOOP > 0) when the sequence playback was started.

Bit number				31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
ID				A
Reset 0x00000000				0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

ID	R/W	Field	Value ID	Value	Description
A	RW	EVENTS_LOOPSDONE			Concatenated sequences have been played the amount of times defined in LOOP.CNT This event triggers after the last SEQ[1] completion of the loop, and only if looping was enabled (LOOP > 0) when the sequence playback was started.
			NotGenerated	0	Event not generated
			Generated	1	Event generated

EVENTS_RAMUNDERFLOW

Address offset: 0x120

Emitted when retrieving from RAM does not complete in time for the PWM module

Bit number				31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
ID				A
Reset 0x00000000				0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

ID	R/W	Field	Value ID	Value	Description
A	RW	EVENTS_RAMUNDERFLOW			Emitted when retrieving from RAM does not complete in time for the PWM module
			NotGenerated	0	Event not generated
			Generated	1	Event generated

EVENTS_DMA

Peripheral events.

EVENTS_DMA.SEQ[n] (n=0..1)

Peripheral events.

EVENTS_DMA.SEQ[n].END (n=0..1)

Address offset: 0x124 + (n × 0xC)

Generated after all MAXCNT bytes have been transferred

Bit number				31 30 29 28	27 26 25 24	23 22 21 20	19 18 17 16	15 14 13 12	11 10 9 8	7 6 5 4	3 2 1 0
ID											A
Reset 0x00000000				0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0000	0000
ID	R/W	Field	Value ID	Value	Description						
A	RW	END			Generated after all MAXCNT bytes have been transferred						
			NotGenerated	0	Event not generated						
			Generated	1	Event generated						

EVENTS_DMA.SEQ[n].READY (n=0..1)

Address offset: 0x128 + (n × 0xC)

Generated when EasyDMA has buffered the .PTR and .MAXCNT registers for the channel, allowing them to be written to prepare for the next sequence.

Bit number				31 30 29 28	27 26 25 24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID																											A
Reset 0x00000000				0 0 0 0	0 0 0 0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ID	R/W	Field	Value ID	Value	Description																								
A	RW	READY			Generated when EasyDMA has buffered the .PTR and .MAXCNT registers for the channel, allowing them to be written to prepare for the next sequence.																								
			NotGenerated	0	Event not generated																								
			Generated	1	Event generated																								

EVENTS_DMA.SEQ[n].BUSERROR (n=0..1)

Address offset: 0x12C + (n × 0xC)

An error occured during the bus transfer.

When this event is generated, the address which caused the error can be read from the BUSERRORADDRESS register.

Bit number				31 30 29 28	27 26 25 24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID																											A
Reset 0x00000000				0 0 0 0	0 0 0 0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ID	R/W	Field	Value ID	Value	Description																								
A	RW	BUSERROR			An error occured during the bus transfer. When this event is generated, the address which caused the error can be read from the BUSERRORADDRESS register.																								
			NotGenerated	0	Event not generated																								
			Generated	1	Event generated																								

EVENTS_COMPAREMATCH[n] (n=0..3)

Address offset: 0x13C + (n × 0x4)

This event is generated when the compare matches for the compare channel [n].

Bit number				31 30 29 28	27 26 25 24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID																											A
Reset 0x00000000				0 0 0 0	0 0 0 0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ID	R/W	Field	Value ID	Value	Description																								
A	RW	EVENTS_COMPAREMATCH			This event is generated when the compare matches for the compare channel [n].																								
			NotGenerated	0	Event not generated																								
			Generated	1	Event generated																								

PUBLISH_STOPPED

Address offset: 0x184

Publish configuration for event STOPPED

Bit number				31 30 29 28	27 26 25 24	23 22 21 20	19 18 17 16	15 14 13 12	11 10 9 8	7 6 5 4	3 2 1 0
ID				B						AAAA	AAAA
Reset 0x00000000				0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0000	0000
ID	R/W	Field	Value ID	Value	Description						
A	RW	CHIDX		[0..255]	DPPI channel that event STOPPED will publish to						
B	RW	EN									
			Disabled	0	Disable publishing						
			Enabled	1	Enable publishing						

PUBLISH_SEQSTARTED[n] (n=0..1)

Address offset: 0x188 + (n × 0x4)

Publish configuration for event SEQSTARTED[n]

Bit number				31 30 29 28	27 26 25 24	23 22 21 20	19 18 17 16	15 14 13 12	11 10 9 8	7 6 5 4	3 2 1 0
ID				B						AAAA	AAAA
Reset 0x00000000				0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0000	0000
ID	R/W	Field	Value ID	Value	Description						
A	RW	CHIDX		[0..255]	DPPI channel that event SEQSTARTED[n] will publish to						
B	RW	EN									
			Disabled	0	Disable publishing						
			Enabled	1	Enable publishing						

PUBLISH_SEQEND[n] (n=0..1)

Address offset: 0x190 + (n × 0x4)

Publish configuration for event SEQEND[n]

Bit number				31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
ID				B AAAA AAAA
Reset 0x00000000				0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

ID	R/W	Field	Value ID	Value	Description
A	RW	CHIDX		[0..255]	DPPI channel that event SEQEND[n] will publish to
B	RW	EN			
			Disabled	0	Disable publishing
			Enabled	1	Enable publishing

PUBLISH_PWMPERIODEND

Address offset: 0x198

Publish configuration for event PWMPERIODEND

Bit number				31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
ID				B AAAA AAAA
Reset 0x00000000				0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

ID	R/W	Field	Value ID	Value	Description
A	RW	CHIDX		[0..255]	DPPI channel that event PWMPERIODEND will publish to
B	RW	EN			
			Disabled	0	Disable publishing
			Enabled	1	Enable publishing

PUBLISH_LOOPSDONE

Address offset: 0x19C

Publish configuration for event LOOPSDONE

This event triggers after the last SEQ[1] completion of the loop, and only if looping was enabled (LOOP > 0) when the sequence playback was started.

Bit number				31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
ID				B AAAA AAAA
Reset 0x00000000				0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

ID	R/W	Field	Value ID	Value	Description
A	RW	CHIDX		[0..255]	DPPI channel that event LOOPSDONE will publish to
B	RW	EN			
			Disabled	0	Disable publishing
			Enabled	1	Enable publishing

PUBLISH_RAMUNDERFLOW

Address offset: 0x1A0

Publish configuration for event RAMUNDERFLOW

Bit number				31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
ID				B AAAA AAAA
Reset 0x00000000				0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

ID	R/W	Field	Value ID	Value	Description
A	RW	CHIDX		[0..255]	DPPI channel that event RAMUNDERFLOW will publish to
B	RW	EN			
			Disabled	0	Disable publishing
			Enabled	1	Enable publishing

PUBLISH_DMA

Publish configuration for events

PUBLISH_DMA.SEQ[n] (n=0..1)

Publish configuration for events

PUBLISH_DMA.SEQ[n].END (n=0..1)

Address offset: 0x1A4 + (n × 0xC)

Publish configuration for event END

Bit number				31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
ID				B AAAA AAAA
Reset 0x00000000				0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

ID	R/W	Field	Value ID	Value	Description
A	RW	CHIDX		[0..255]	DPPI channel that event END will publish to
B	RW	EN			
			Disabled	0	Disable publishing
			Enabled	1	Enable publishing

PUBLISH_DMA.SEQ[n].READY (n=0..1)

Address offset: 0x1A8 + (n × 0xC)

Publish configuration for event READY

Bit number				31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
ID				B AAAA AAAA
Reset 0x00000000				0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

ID	R/W	Field	Value ID	Value	Description
A	RW	CHIDX		[0..255]	DPPI channel that event READY will publish to
B	RW	EN			
			Disabled	0	Disable publishing
			Enabled	1	Enable publishing

PUBLISH_DMA.SEQ[n].BUSERROR (n=0..1)

Address offset: 0x1AC + (n × 0xC)

Publish configuration for event BUSERROR

When this event is generated, the address which caused the error can be read from the BUSERRORADDRESS register.

Bit number					31 30 29 28	27 26 25 24	23 22 21 20	19 18 17 16	15 14 13 12	11 10 9 8	7 6 5 4	3 2 1 0
ID					B						A A A A	A A A A
Reset 0x00000000					0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0
ID	R/W	Field		Value ID	Value		Description					
A	RW	CHIDX			[0..255]		DPPI channel that event BUSERROR will publish to					
B	RW	EN										
			Disabled	0			Disable publishing					
			Enabled	1			Enable publishing					

PUBLISH_COMPAREMATCH[n] (n=0..3)

Address offset: 0x1BC + (n × 0x4)

Publish configuration for event COMPAREMATCH[n]

Bit number					31 30 29 28	27 26 25 24	23 22 21 20	19 18 17 16	15 14 13 12	11 10 9 8	7 6 5 4	3 2 1 0
ID					B						A A A A	A A A A
Reset 0x00000000					0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0
ID	R/W	Field		Value ID	Value		Description					
A	RW	CHIDX			[0..255]		DPPI channel that event COMPAREMATCH[n] will publish to					
B	RW	EN										
			Disabled	0			Disable publishing					
			Enabled	1			Enable publishing					

SHORTS

Address offset: 0x200

Shortcuts between local events and tasks

Bit number					31 30 29 28	27 26 25 24	23 22 21 20	19 18 17 16	15 14 13 12	11 10 9 8	7 6 5 4	3 2 1 0
ID											H G F E	D C B A
Reset 0x00000000					0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0
ID	R/W	Field		Value ID	Value		Description					
A-B	RW	SEQEND[i]_STOP (i=0..1)					Shortcut between event SEQEND[n] and task STOP					
			Disabled	0			Disable shortcut					
			Enabled	1			Enable shortcut					
C-D	RW	LOOPSDONE_DMA_SEQ[i]_START (i=0..1)					Shortcut between event LOOPSDONE and task DMA.SEQ[n].START					
			Disabled	0			Disable shortcut					
			Enabled	1			Enable shortcut					
E	RW	LOOPSDONE_STOP					Shortcut between event LOOPSDONE and task STOP					
			Disabled	0			Disable shortcut					
			Enabled	1			Enable shortcut					
F	RW	RAMUNDERFLOW_STOP					Shortcut between event RAMUNDERFLOW and task STOP					
			Disabled	0			Disable shortcut					
			Enabled	1			Enable shortcut					
G-H	RW	DMA_SEQ[i]_BUSERROR_STOP (i=0..1)					Shortcut between event DMA.SEQ[n].BUSERROR and task STOP					
			Disabled	0			Disable shortcut					
			Enabled	1			Enable shortcut					

INTEN

Address offset: 0x300

Enable or disable interrupt

Bit number					31 30 29 28	27 26 25 24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID												R	Q	P	O	N	M	L	K	J	I	H	G	F	E	D	C	B	A	
Reset 0x00000000					0 0 0 0	0 0 0 0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ID	R/W	Field		Value ID	Value		Description																							
A	RW	STOPPED					Enable or disable interrupt for event STOPPED																							
			Disabled	0			Disable																							
			Enabled	1			Enable																							
B-C	RW	SEQSTARTED[i] (i=0..1)					Enable or disable interrupt for event SEQSTARTED[i]																							
			Disabled	0			Disable																							
			Enabled	1			Enable																							
D-E	RW	SEQEND[i] (i=0..1)					Enable or disable interrupt for event SEQEND[i]																							
			Disabled	0			Disable																							
			Enabled	1			Enable																							
F	RW	PWMPERIODEND					Enable or disable interrupt for event PWMPERIODEND																							
			Disabled	0			Disable																							
			Enabled	1			Enable																							
G	RW	LOOPSDONE					Enable or disable interrupt for event LOOPSDONE																							
							This event triggers after the last SEQ[1] completion of the loop, and only if looping was enabled (LOOP > 0) when the sequence playback was started.																							
			Disabled	0			Disable																							
			Enabled	1			Enable																							
H	RW	RAMUNDERFLOW					Enable or disable interrupt for event RAMUNDERFLOW																							
			Disabled	0			Disable																							
			Enabled	1			Enable																							
I	RW	DMASEQ0END					Enable or disable interrupt for event DMASEQ0END																							
			Disabled	0			Disable																							
			Enabled	1			Enable																							

Bit number				31 30 29 28	27 26 25 24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID											R	Q	P	O	N	M	L	K	J	I	H	G	F	E	D	C	B	A	
Reset 0x00000000				0 0 0 0	0 0 0 0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ID	R/W	Field	Value ID	Value	Description
J	RW	DMASEQ0READY			Enable or disable interrupt for event DMASEQ0READY
			Disabled	0	Disable
			Enabled	1	Enable
K	RW	DMASEQ0BUSERROR			Enable or disable interrupt for event DMASEQ0BUSERROR When this event is generated, the address which caused the error can be read from the BUSERRORADDRESS register.
			Disabled	0	Disable
			Enabled	1	Enable
L	RW	DMASEQ1END			Enable or disable interrupt for event DMASEQ1END
			Disabled	0	Disable
			Enabled	1	Enable
M	RW	DMASEQ1READY			Enable or disable interrupt for event DMASEQ1READY
			Disabled	0	Disable
			Enabled	1	Enable
N	RW	DMASEQ1BUSERROR			Enable or disable interrupt for event DMASEQ1BUSERROR When this event is generated, the address which caused the error can be read from the BUSERRORADDRESS register.
			Disabled	0	Disable
			Enabled	1	Enable
O-R	RW	COMPAREMATCH[i] (i=0..3)			Enable or disable interrupt for event COMPAREMATCH[i]
			Disabled	0	Disable
			Enabled	1	Enable

INTENSET

Address offset: 0x304

Enable interrupt

Bit number				31 30 29 28	27 26 25 24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID											R	Q	P	O	N	M	L	K	J	I	H	G	F	E	D	C	B	A	
Reset 0x00000000				0 0 0 0	0 0 0 0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ID	R/W	Field	Value ID	Value	Description
A	RW W1S	STOPPED			Write '1' to enable interrupt for event STOPPED
			Set	1	Enable
			Disabled	0	Read: Disabled
			Enabled	1	Read: Enabled
B-C	RW W1S	SEQSTARTED[i] (i=0..1)			Write '1' to enable interrupt for event SEQSTARTED[i]
			Set	1	Enable
			Disabled	0	Read: Disabled
			Enabled	1	Read: Enabled
D-E	RW W1S	SEQEND[i] (i=0..1)			Write '1' to enable interrupt for event SEQEND[i]
			Set	1	Enable
			Disabled	0	Read: Disabled
			Enabled	1	Read: Enabled
F	RW W1S	PWMPERIODEND			Write '1' to enable interrupt for event PWMPERIODEND
			Set	1	Enable
			Disabled	0	Read: Disabled
			Enabled	1	Read: Enabled
G	RW W1S	LOOPSDONE			Write '1' to enable interrupt for event LOOPSDONE This event triggers after the last SEQ[1] completion of the loop, and only if looping was enabled (LOOP > 0) when the sequence playback was started.
			Set	1	Enable
			Disabled	0	Read: Disabled
			Enabled	1	Read: Enabled
H	RW W1S	RAMUNDERFLOW			Write '1' to enable interrupt for event RAMUNDERFLOW
			Set	1	Enable
			Disabled	0	Read: Disabled
			Enabled	1	Read: Enabled
I	RW W1S	DMASEQ0END			Write '1' to enable interrupt for event DMASEQ0END
			Set	1	Enable
			Disabled	0	Read: Disabled
			Enabled	1	Read: Enabled
J	RW W1S	DMASEQ0READY			Write '1' to enable interrupt for event DMASEQ0READY
			Set	1	Enable
			Disabled	0	Read: Disabled
			Enabled	1	Read: Enabled
K	RW W1S	DMASEQ0BUSERROR			Write '1' to enable interrupt for event DMASEQ0BUSERROR When this event is generated, the address which caused the error can be read from the BUSERRORADDRESS register.
			Set	1	Enable
			Disabled	0	Read: Disabled
			Enabled	1	Read: Enabled
L	RW W1S	DMASEQ1END			Write '1' to enable interrupt for event DMASEQ1END
			Set	1	Enable
			Disabled	0	Read: Disabled
			Enabled	1	Read: Enabled
M	RW W1S	DMASEQ1READY			Write '1' to enable interrupt for event DMASEQ1READY
			Set	1	Enable
			Disabled	0	Read: Disabled
			Enabled	1	Read: Enabled
N	RW W1S	DMASEQ1BUSERROR			Write '1' to enable interrupt for event DMASEQ1BUSERROR When this event is generated, the address which caused the error can be read from the BUSERRORADDRESS register.
			Set	1	Enable

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID														R	Q	P	O	N	M	L	K	J	I	H	G	F	E	D	C	B	A	
Reset 0x00000000	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ID	R/W	Field	Value ID	Value	Description
			Disabled	0	Read: Disabled
			Enabled	1	Read: Enabled
O-R	RW W1S	COMPAREMATCH[i] (i=0..3)			Write '1' to enable interrupt for event COMPAREMATCH[i]
			Set	1	Enable
			Disabled	0	Read: Disabled
			Enabled	1	Read: Enabled

INTENCLR

Address offset: 0x308

Disable interrupt

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID														R	Q	P	O	N	M	L	K	J	I	H	G	F	E	D	C	B	A	
Reset 0x00000000	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ID	R/W	Field	Value ID	Value	Description
A	RW W1C	STOPPED			Write '1' to disable interrupt for event STOPPED
			Clear	1	Disable
			Disabled	0	Read: Disabled
			Enabled	1	Read: Enabled
B-C	RW W1C	SEQSTARTED[i] (i=0..1)			Write '1' to disable interrupt for event SEQSTARTED[i]
			Clear	1	Disable
			Disabled	0	Read: Disabled
			Enabled	1	Read: Enabled
D-E	RW W1C	SEQEND[i] (i=0..1)			Write '1' to disable interrupt for event SEQEND[i]
			Clear	1	Disable
			Disabled	0	Read: Disabled
			Enabled	1	Read: Enabled
F	RW W1C	PWMPERIODEND			Write '1' to disable interrupt for event PWMPERIODEND
			Clear	1	Disable
			Disabled	0	Read: Disabled
			Enabled	1	Read: Enabled
G	RW W1C	LOOPSDONE			Write '1' to disable interrupt for event LOOPSDONE This event triggers after the last SEQ[1] completion of the loop, and only if looping was enabled (LOOP > 0) when the sequence playback was started.
			Clear	1	Disable
			Disabled	0	Read: Disabled
			Enabled	1	Read: Enabled
H	RW W1C	RAMUNDERFLOW			Write '1' to disable interrupt for event RAMUNDERFLOW
			Clear	1	Disable
			Disabled	0	Read: Disabled
			Enabled	1	Read: Enabled
I	RW W1C	DMASEQ0END			Write '1' to disable interrupt for event DMASEQ0END
			Clear	1	Disable
			Disabled	0	Read: Disabled
			Enabled	1	Read: Enabled
J	RW W1C	DMASEQ0READY			Write '1' to disable interrupt for event DMASEQ0READY
			Clear	1	Disable
			Disabled	0	Read: Disabled
			Enabled	1	Read: Enabled
K	RW W1C	DMASEQ0BUSERROR			Write '1' to disable interrupt for event DMASEQ0BUSERROR When this event is generated, the address which caused the error can be read from the BUSERRORADDRESS register.
			Clear	1	Disable
			Disabled	0	Read: Disabled
			Enabled	1	Read: Enabled
L	RW W1C	DMASEQ1END			Write '1' to disable interrupt for event DMASEQ1END
			Clear	1	Disable
			Disabled	0	Read: Disabled
			Enabled	1	Read: Enabled
M	RW W1C	DMASEQ1READY			Write '1' to disable interrupt for event DMASEQ1READY
			Clear	1	Disable
			Disabled	0	Read: Disabled
			Enabled	1	Read: Enabled
N	RW W1C	DMASEQ1BUSERROR			Write '1' to disable interrupt for event DMASEQ1BUSERROR When this event is generated, the address which caused the error can be read from the BUSERRORADDRESS register.
			Clear	1	Disable
			Disabled	0	Read: Disabled
			Enabled	1	Read: Enabled
O-R	RW W1C	COMPAREMATCH[i] (i=0..3)			Write '1' to disable interrupt for event COMPAREMATCH[i]
			Clear	1	Disable
			Disabled	0	Read: Disabled
			Enabled	1	Read: Enabled

INTPEND

Address offset: 0x30C

Pending interrupts

Bit number				31 30 29 28	27 26 25 24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID											R	Q	P	O	N	M	L	K	J	I	H	G	F	E	D	C	B	A	
Reset 0x00000000				0 0 0 0	0 0 0 0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ID	R/W	Field	Value ID	Value	Description
A	R	STOPPED			Read pending status of interrupt for event STOPPED
			NotPending	0	Read: Not pending
			Pending	1	Read: Pending
B–C	R	SEQSTARTED[i] (i=0..1)			Read pending status of interrupt for event SEQSTARTED[i]
			NotPending	0	Read: Not pending
			Pending	1	Read: Pending
D–E	R	SEQEND[i] (i=0..1)			Read pending status of interrupt for event SEQEND[i]
			NotPending	0	Read: Not pending
			Pending	1	Read: Pending
F	R	PWMPERIODEND			Read pending status of interrupt for event PWMPERIODEND
			NotPending	0	Read: Not pending
			Pending	1	Read: Pending
G	R	LOOPSDONE			Read pending status of interrupt for event LOOPSDONE This event triggers after the last SEQ[1] completion of the loop, and only if looping was enabled (LOOP > 0) when the sequence playback was started.
			NotPending	0	Read: Not pending
			Pending	1	Read: Pending
H	R	RAMUNDERFLOW			Read pending status of interrupt for event RAMUNDERFLOW
			NotPending	0	Read: Not pending
			Pending	1	Read: Pending
I	R	DMASEQ0END			Read pending status of interrupt for event DMASEQ0END
			NotPending	0	Read: Not pending
			Pending	1	Read: Pending
J	R	DMASEQ0READY			Read pending status of interrupt for event DMASEQ0READY
			NotPending	0	Read: Not pending
			Pending	1	Read: Pending
K	R	DMASEQ0BUSERROR			Read pending status of interrupt for event DMASEQ0BUSERROR When this event is generated, the address which caused the error can be read from the BUSERRORADDRESS register.
			NotPending	0	Read: Not pending
			Pending	1	Read: Pending
L	R	DMASEQ1END			Read pending status of interrupt for event DMASEQ1END
			NotPending	0	Read: Not pending
			Pending	1	Read: Pending
M	R	DMASEQ1READY			Read pending status of interrupt for event DMASEQ1READY
			NotPending	0	Read: Not pending
			Pending	1	Read: Pending
N	R	DMASEQ1BUSERROR			Read pending status of interrupt for event DMASEQ1BUSERROR When this event is generated, the address which caused the error can be read from the BUSERRORADDRESS register.
			NotPending	0	Read: Not pending
			Pending	1	Read: Pending
O–R	R	COMPAREMATCH[i] (i=0..3)			Read pending status of interrupt for event COMPAREMATCH[i]
			NotPending	0	Read: Not pending
			Pending	1	Read: Pending

ENABLE

Address offset: 0x500

PWM module enable register

Bit number			31 30 29 28	27 26 25 24	23 22 21 20	19 18 17 16	15 14 13 12	11 10 9 8	7 6 5 4	3 2 1 0
ID										A
Reset 0x00000000			0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0000	0000

ID	R/W	Field	Value ID	Value	Description
A	RW	ENABLE			Enable or disable PWM module
			Disabled	0	Disabled
			Enabled	1	Enable

MODE

Address offset: 0x504

Selects operating mode of the wave counter

Bit number			31 30 29 28	27 26 25 24	23 22 21 20	19 18 17 16	15 14 13 12	11 10 9 8	7 6 5 4	3 2 1 0
ID										A
Reset 0x00000000			0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0000	0000

ID	R/W	Field	Value ID	Value	Description
A	RW	UPDOWN			Selects up mode or up-and-down mode for the counter
			Up	0	Up counter, edge-aligned PWM duty cycle
			UpAndDown	1	Up and down counter, center-aligned PWM duty cycle

COUNTERTOP

Address offset: 0x508

Value up to which the pulse generator counter counts

Bit number			31 30 29 28	27 26 25 24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID														A	A	A	A	A	A	A	A	A	A	A	A	A	A	A
Reset 0x000003FF			0 0 0 0	0 0 0 0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1

ID	R/W	Field	Value ID	Value	Description
A	RW	COUNTERTOP		[3..32767]	Value up to which the pulse generator counter counts. This register is ignored when DECODER.MODE=WaveForm and only values from RAM are used.

PRESCALER

Address offset: 0x50C

Configuration for PWM_CLK

Bit number			31 30 29 28	27 26 25 24	23 22 21 20	19 18 17 16	15 14 13 12	11 10 9 8	7 6 5 4	3 2 1 0
ID										A A A
Reset 0x00000000			0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0000	0000
ID	R/W	Field	Value ID	Value	Description					
A	RW	PRESCALER			Prescaler of PWM_CLK					
			DIV_1	0	Divide by 1 (16 MHz)					
			DIV_2	1	Divide by 2 (8 MHz)					
			DIV_4	2	Divide by 4 (4 MHz)					
			DIV_8	3	Divide by 8 (2 MHz)					
			DIV_16	4	Divide by 16 (1 MHz)					
			DIV_32	5	Divide by 32 (500 kHz)					
			DIV_64	6	Divide by 64 (250 kHz)					
			DIV_128	7	Divide by 128 (125 kHz)					

DECODER

Address offset: 0x510

Configuration of the decoder

Bit number			31 30 29 28	27 26 25 24	23 22 21 20	19 18 17 16	15 14 13 12	11 10 9 8	7 6 5 4	3 2 1 0
ID									B	A A
Reset 0x00000000			0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0000	0000
ID	R/W	Field	Value ID	Value	Description					
A	RW	LOAD			How a sequence is read from RAM and spread to the compare register					
			Common	0	1st half word (16-bit) used in all PWM channels 0..3					
			Grouped	1	1st half word (16-bit) used in channel 0..1; 2nd word in channel 2..3					
			Individual	2	1st half word (16-bit) in ch.0; 2nd in ch.1; ...; 4th in ch.3					
			WaveForm	3	1st half word (16-bit) in ch.0; 2nd in ch.1; ...; 4th in COUNTERTOP					
B	RW	MODE			Selects source for advancing the active sequence					
			RefreshCount	0	SEQ[n].REFRESH is used to determine loading internal compare registers					
			NextStep	1	NEXTSTEP task causes a new value to be loaded to internal compare registers					

LOOP

Address offset: 0x514

Number of playbacks of a loop

Bit number			31 30 29 28	27 26 25 24	23 22 21 20	19 18 17 16	15 14 13 12	11 10 9 8	7 6 5 4	3 2 1 0
ID							A A A A	A A A A	A A A A	A A A A
Reset 0x00000000			0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0000	0000
ID	R/W	Field	Value ID	Value	Description					
A	RW	CNT			Number of playbacks of pattern cycles					
			Disabled	0	Looping disabled (stop at the end of the sequence)					

IDLEOUT

Address offset: 0x518

Configure the output value on the PWM channel during idle

Writes to this register are ignored when the PWM is enabled.

Bit number			31 30 29 28	27 26 25 24	23 22 21 20	19 18 17 16	15 14 13 12	11 10 9 8	7 6 5 4	3 2 1 0
ID										D C B A
Reset 0x00000000			0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0000	0000
ID	R/W	Field	Value ID	Value	Description					
A-D	RW	VAL[i] (i=0..3)			Idle output value for PWM channel [i]					

SEQ[n].REFRESH (n=0..1)

Address offset: 0x528 + (n × 0x20)

Number of additional PWM periods between samples loaded into compare register

Bit number			31 30 29 28	27 26 25 24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID					A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A
Reset 0x00000001			0 0 0 0	0 0 0 0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
ID	R/W	Field	Value ID	Value	Description																							
A	RW	CNT			Number of additional PWM periods between samples loaded into compare register (load every REFRESH.CNT+1 PWM periods)																									
			Continuous	0	Update every PWM period																							

SEQ[n].ENDDELAY (n=0..1)

Address offset: 0x52C + (n × 0x20)

Time added after the sequence

Bit number			31 30 29 28	27 26 25 24	23 22 21 20	19 18 17 16	15 14 13 12	11 10 9 8	7 6 5 4	3 2 1 0
ID					A A A A	A A A A	A A A A	A A A A	A A A A	A A A A
Reset 0x00000000			0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0000	0000
ID	R/W	Field	Value ID	Value	Description					
A	RW	CNT			Time added after the sequence in PWM periods					

PSEL.OUT[n] (n=0..3)

Address offset: 0x560 + (n × 0x4)

Output pin select for PWM channel n

Bit number	31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
ID	C B B B A A A A
Reset 0xFFFFFFFF	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

ID	R/W	Field	Value ID	Value	Description
A	RW	PIN		[0..31]	Pin number
B	RW	PORT		[0..7]	Port number
C	RW	CONNECT			Connection
			Disconnected	1	Disconnect
			Connected	0	Connect

DMA.SEQ[n].PTR (n=0..1)

Address offset: 0x704 + (n × 0x24)

RAM buffer start address

Bit number	31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
ID	A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A
Reset 0x00000000	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

ID	R/W	Field	Value ID	Value	Description
A	RW	PTR			RAM buffer start address for this EasyDMA channel. This address is a word aligned Data RAM address. **Note:** See the memory chapter for details about which memories are available for EasyDMA.

DMA.SEQ[n].MAXCNT (n=0..1)

Address offset: 0x708 + (n × 0x24)

Maximum number of bytes in channel buffer

Bit number	31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
ID	A A A A A A A A A A A A A A A A
Reset 0x00000000	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

ID	R/W	Field	Value ID	Value	Description
A	RW	MAXCNT		[1..0x7fff]	Maximum number of bytes in channel buffer

DMA.SEQ[n].AMOUNT (n=0..1)

Address offset: 0x70C + (n × 0x24)

Number of bytes transferred in the last transaction, updated after the END event.

Bit number	31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
ID	A A A A A A A A A A A A A A A A
Reset 0x00000000	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

ID	R/W	Field	Value ID	Value	Description
A	R	AMOUNT		[1..0x7fff]	Number of bytes transferred in the last transaction. In case of NACK error, includes the NACK'ed byte.

DMA.SEQ[n].CURRENTAMOUNT (n=0..1)

Address offset: 0x710 + (n × 0x24)

Number of bytes transferred in the current transaction

Bit number	31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
ID	A A A A A A A A A A A A A A A A
Reset 0x00000000	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

ID	R/W	Field	Value ID	Value	Description
A	R	AMOUNT		[1..0x7fff]	Number of bytes transferred in the current transaction. Continuously updated.

DMA.SEQ[n].TERMINATEONBUSERROR (n=0..1)

Address offset: 0x71C + (n × 0x24)

Terminate the transaction if a BUSERROR event is detected.

Bit number	31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
ID	A
Reset 0x00000000	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

ID	R/W	Field	Value ID	Value	Description
A	RW	ENABLE			
			Disabled	0	Disable
			Enabled	1	Enable

DMA.SEQ[n].BUSERRORADDRESS (n=0..1)

Address offset: 0x720 + (n × 0x24)

Address of transaction that generated the last BUSERROR event.

Bit number	31 30 29 28	27 26 25 24	23 22 21 20	19 18 17 16	15 14 13 12	11 10 9 8	7 6 5 4	3 2 1 0
ID	A A A A	A A A A	A A A A	A A A A	A A A A	A A AA	AAAA	AAAA
Reset 0x00000000	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 00	0000	0000

ID	R/W	Field	Value ID	Value	Description
A	R	ADDRESS			