DMA and Radio Configuration

By Siri Johnsrud

Keywords

- *DMA*
- *Packet Handling Configuration*
- *Radio*
- *CC1110*
- *CC1111*
- *CC2510*
- *CC2511*

1 Introduction

The purpose of this design note is to describe how the DMA should be configured for the different packet formats supported by the radio.

In the following sections, an `n` in the register name represents the channel number 0, 1, 2, 3, or 4 if nothing else is stated.

Table of Contents

2 Abbreviations

CPU	Central Processing Unit
DMA	Direct Memory Access
ISR	Interrupt Service Routine
RF	Radio Frequency
RX	Receive
TX	Transmit

3 DMA Configuration

The DMA channel parameters have to be configured before a DMA channel can be armed and activated. These parameters are written in a special DMA configuration data structure in memory. The DMA configuration data structure consists of eight bytes and is described below. Please see the CC1110/CC1111 [1] and/or the CC2510/CC2511 [1] data sheet for more details.

- Source Address (**SRCADDR**)

  This is the address where the DMA shall start to read data. In TX mode, this should be the address to a buffer in XDATA memory space, *txBuffer*, holding the data to be transmitted. In RX mode, this should be the address of the RF Data register, `RFD`.

- Destination Address (**DESTADDR**)

  This is the address where the DMA shall start to write the data read from the source address. In TX mode, this should be the address of the RF Data register, `RFD`. In RX mode this should be the address to a buffer in XDATA memory space, *rxBuffer*, where the received data should be stored.

  Note: The size of txBuffer and rxBuffer must be equal to, or greater than, the maximum transfer count (see Table 1).

- Transfer Count (**VLEN** and **LEN**)

  The transfer count gives the number of bytes/words needed to be moved from source to destination. There are two parameters used for configuring the transfer count. These are VLEN and LEN. How these parameters should be set depends on the packet format of the radio packets and will be discussed in details in Section 4.

- Byte or Word Transfer (**WORDSIZE**)

  The radio packet format is byte oriented, hence each DMA transfer should be one byte (*WORDSIZE* = 0_b).

- Trigger Event (**TRIG**)

  When used to move data to and from the `RFD` register, the trigger event should be DMA trigger #19, which is the radio trigger. A trigger event will occur for each new byte the radio writes to the `RFD` register in RX mode and for each byte the radio reads from the `RFD` register in TX mode (*TRIG* = 10011_b).

- Transfer Mode (**TMODE**)

  Since there is a trigger event for every byte transmitted/received, the transfer mode should be set to single mode (*TMODE* = 00_b). On each trigger, a single byte transfer occurs and the DMA channel awaits the next trigger.

- Source Increment (**SRCINC**)

  - TX mode:

    The source address is the address to a buffer in XDATA memory space, *txBuffer*, and the source address should be configured to increment by one after each transfer (*SRCINC* = 01_b).

  - RX mode:

    The source address is the address of the RF Data register, `RFD` and the source address should not change between transfers (*SRCINC* = 00_b).

- Destination Increment (**DESTINC**)
    - TX mode:

      The destination address is the address of the RF Data register, `RFD`, and the destination address should not be changed between transfers (*SRCINC = 00_b*).
    - RX mode:

      The destination address is the address to a buffer in XDATA memory space, *rxBuffer*, and the destination address should be configured to increment by one after each transfer (*SRCINC = 01_b*).

- Interrupt Mask (**IRQMASK**)

  If this bit is set to 1, the CPU interrupt flag `IRCON.DMAIF` will be asserted when the transfer count is reached and an interrupt request will be generated if the corresponding CPU interrupt mask bit, `IEN1.DMAIE`, is 1. Note that the DMA interrupt flag `DMAIRQ.DMAIFn` will be set when transfer count is reached regardless of the IRQMASK bit.
    - TX mode:

      Since the DMA will be done transferring data to the `RFD` register before the radio is done transmitting the data on the air, the general RF interrupt associated with the `IRQ_DONE` flag should be used instead of the DMA interrupt to make sure that the radio is not turned off before the packet is properly transmitted (*IRQMASK = 0_b*).
    - RX mode:

      In receive mode the radio and DMA interrupt will happen almost simultaneously except in the case where *`PKTCTRL1.APPEND_STATUS` = 0_b* and *`PKTCTRL0.CRC_EN` = 1_b*. In this case the DMA interrupt will happen before the RF interrupt. If one uses the RF interrupt one must be aware that this interrupt will also be executed in TX mode when a packet has been transmitted.

> *Note: If the radio implements maximum length filtering (`PKTCTRL0.LENGTH_CONFIG` = 01_b and `PKTLEN` ≠ 0xFF) or address filtering (`PKTCTRL1.ADR_CHK` ≠ 00_b), filtering of packets will cause the `IRQ_DONE` flag to be asserted, but will not give a DMA trigger (`DMAIRQ.DMAIFn` will not be asserted).*

- Mode 8 Setting (**M8**)

  This configuration is only applicable when doing byte transfers (WORDSIZE = 0_b) and the transfer count is of variable length (VLEN ≠ 000_b and VLEN ≠ 111_b). When this is the case, this field determines whether to use seven or eight bits of the first byte in source data to determine the transfer count. To be compliant with the radio packet format (see Figure 1), all 8 bits should be used as the transfer length (*M8 = 0_b*).

- DMA Priority (**PRIORITY**)

  The DMA priority is used to determine the winner in the case of multiple simultaneous internal memory requests, and whether the DMA memory access should have priority or not over a simultaneous CPU memory access. The priority should be set to high (*PRORITY = 10_b*) when the DMA is used to move data to and from the `RFD` register, to avoid having the radio enter TX_UNDERFLOW or RX_OVERFLOW state. See the data sheets ([1] and [2]) for more details on the different radio control states.

4 Packet Format

The packet format supported by the radio is shown in Figure 1.

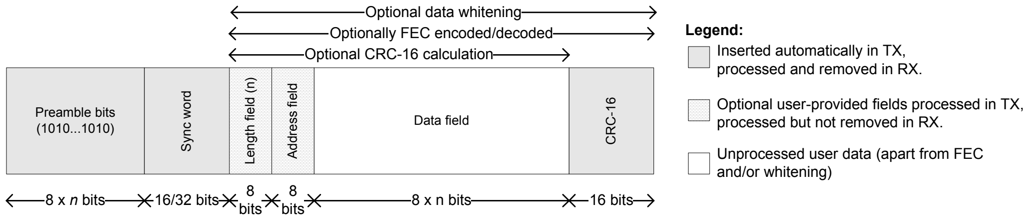

Figure 1. Packet Format

In addition to a preamble and a sync word (2 or 4 bytes long), the packet consist of an optional length byte n, an optional address byte, the payload, and an optional 2 byte CRC. The address byte is part of the payload and is not interpreted by the DMA. If the radio implements address filtering (`PKTCTRL1.ADR_CHK` ≠ 00_b) and a packet is being discarded, RX mode will be restarted (regardless of the `MCSM1.RXOFF_MODE` setting), and the `RFIF.IRQ_DONE` flag will be asserted but the DMA will not be triggered.

4.1 Radio using Variable Packet Length Mode

Variable packet length mode is selected by setting `PKTCTRL0.LENGTH_CONFIG` = 01_b. In this mode, the length byte, n, is following the sync word in the packet ($1 \leq n \leq 255$). The packet length is defined as the payload data, excluding the length byte and the optional CRC bytes. The DMA has 4 different configurations which supports variable length transfer count and these are VLEN = {001_b, 010_b, 011_b, 100_b} (see Figure 2).

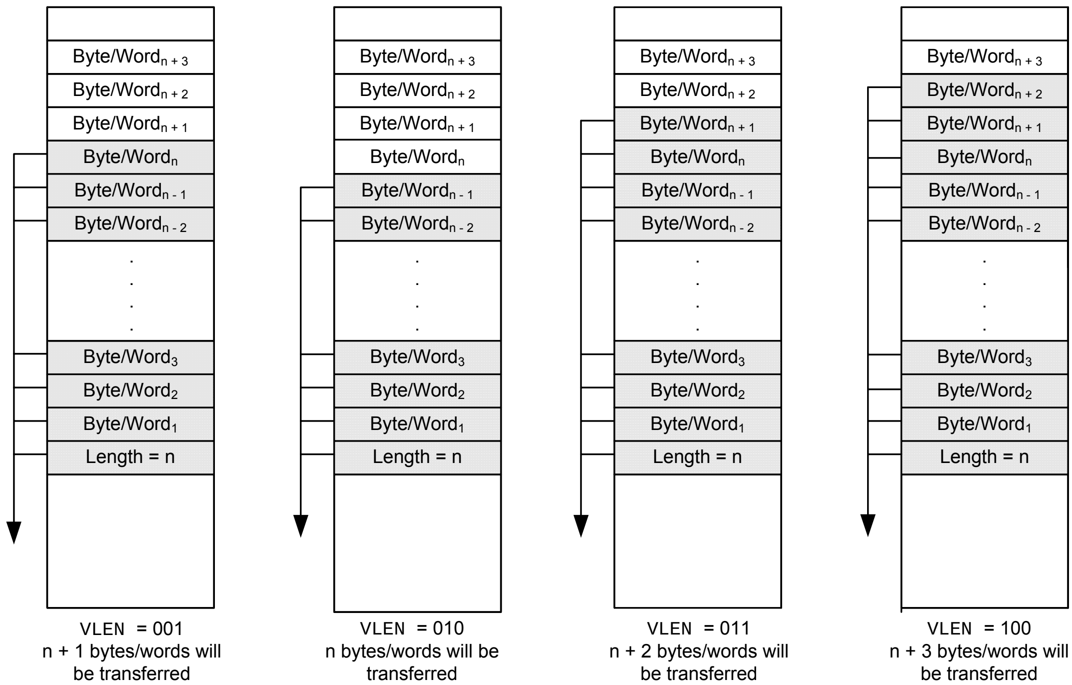

Figure 2. Variable Length Transfer Count Options

However, only two of these configurations are useful when moving data packets to and from the `RFD` register; VLEN = 001_b and VLEN = 100_b. Which one to use depends on the active mode of the radio (RX or TX) and on the `APPEND_STATUS` field in the `PKTCTRL1` register (RX mode only). Assume transmitting and receiving the data packet shown in Figure 3:

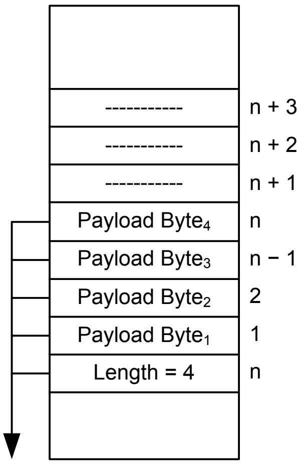

Figure 3. Variable Packet Length Mode (n = 4)

- TX mode:

  A total of 5 (n + 1) bytes should be transmitted (1 length byte + n payload bytes) → *VLEN = 001_b* (see Figure 2).

- RX mode
  - `PKTCTRL1.APPEND_STATUS` = 0_b:
    Nothing is appended to the payload → *VLEN = 001_b* (see Figure 2)
  - `PKTCTRL1.APPEND_STATUS` = 1_b:
    Two bytes are appended to the received payload at position (n + 1) and position (n + 2). This means that a total of n + 3 bytes should be transmitted → *VLEN = 100_b* (see Figure 2).

4.1.1 Maximum Length Filtering in RX Mode

Assume that the packets to be received are of variable length but the max length byte value is less than 255 (n_{max} < 255). In this case, maximum length filtering can be used in the radio to avoid receiving packets intended for other receivers (or noise). To enable maximum length filtering, `PKTLEN.PACKET_LENGTH` should be set to n_{max}. In RX mode, the radio will discard packets with a length byte larger than n_{max} and RX mode will be restarted. The `RFIF.IRQ_DONE` flag will be asserted but the DMA will not be triggered. See the data sheets ([1] and [2]) for more details on the different interrupt flags associated with the radio.

Note: The `PKTLEN` register is not used by the radio in TX mode when configured for variable packet length mode (`PKTCTRL0.LENGTH_CONFIG` = 01_b).

When a DMA channel is configured to operate with variable length transfer counts, VLEN = {001_b, 010_b, 011_b, 100_b}, the transfer count will be limited to LEN bytes/words when n ≥ LEN (see Table 1). Table 2 shows the transfer count for different values of n when LEN = 13.

	Transfer Count			
	VLEN = 001_b	**VLEN = 010_b**	**VLEN = 011_b**	**VLEN = 100_b**
n < LEN	n + 1	n	n + 2	n + 3
n ≥ LEN	LEN	LEN	LEN	LEN
Max Transfer Count	LEN	LEN	LEN + 1	LEN + 2

Table 1. Transfer Count

From Table 1 we see that when VLEN = {001_b, 011_b, 100_b}, LEN should be greater than n to make sure that the complete packet is transferred. When VLEN = 010_b, LEN can be set equal to n. Due to the maximum length filtering implemented in the radio, only packets with length byte ≤ n_{max} will trigger the DMA.

LEN = n_{max} + 1 when VLEN = {001_b, 011_b, 100_b}

LEN = n_{max} when VLEN = 010_b

<u>Example 1:</u>

A transmitter transmits packets of variable lengths (see Figure 4) and the length byte will have a value n, where 1 ≤ n ≤ 255. CRC is appended.

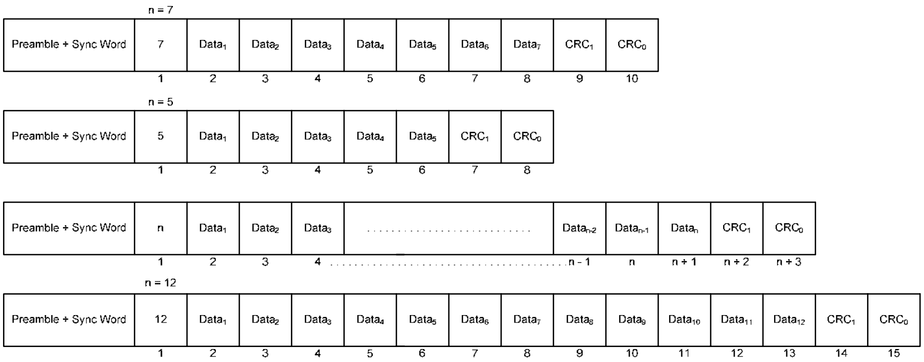

Figure 4. Packets Transmitted

- Transmitter Configuration:

  The CRC bytes at the end of the packets are appended automatically by the radio, hence the DMA controller should be configured to transfer the length byte and the data bytes to the `RFD` register → VLEN = 001_b (transfer count is n + 1).

  LEN = n_{max} + 1 = 255 + 1 = 256 → LEN = 0000100000000_b

  `PKTLEN` = $xxxxxxxx_b$ (don't care)

- Receiver Configuration:

  The receiver has `PKTCTRL1.APPEND_STATUS` = 1_b meaning that 2 status bytes are appended to the payload. The CRC bytes are processed and removed automatically, hence they will never appear in the `RFD` register. By default, `PKTLEN` = 255, and the receiver will accept all packets with a valid sync word (there are 8 different sync word qualifier modes configured through `MDMCFG.SYNC_MODE`) See the data sheets ([1] and [2]) for more details.

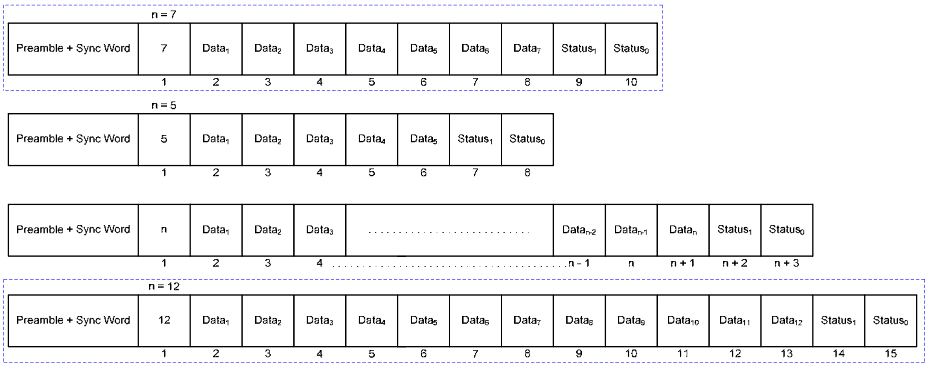

Figure 5. Packets Received

A receiver is e.g. only interested in packets where n = {7, 12} (see Figure 5), hence maximum length filtering is enabled → PKTLEN = n_{max} = 12 → PKTLEN = 00001100_b This means that if the radio receives a length byte greater than 12, the packet will be discarded and nothing will be put in the RFD register → the DMA will not be triggered.

VLEN = 100_b, (transfer count is n + 3)

LEN = n_{max} + 1 = 12 + 1 = 13 → LEN = 0000000001101_b

> *Note: All packets with length byte ≤ 12 will be received by the radio and trigger the DMA. Filtering of packets with length byte not equal to 7 or 12 must be implemented in software.*

4.2 Radio using Fixed Packet Length Mode

Fixed packet length mode is selected by setting PKTCTRL0.LENGTH_CONFIG = 00_b. In this mode the packet does not contain a length byte, and the PKTLEN register determines how many bytes will be transmitted/received (1 ≤ PKTLEN ≤ 255). By setting VLEN = {000_b, 111_b} the DMA will be configured for fixed length transfer count, and the transfer count is given by the LEN setting (see Figure 6).

- TX mode:

  LEN = PKTLEN

- RX mode:
  - PKTCTRL1.APPEND_STATUS = 0_b:
    LEN = PKTLEN
  - PKTCTRL1.APPEND_STATUS = 1_b:
    LEN = PKTLEN + 2

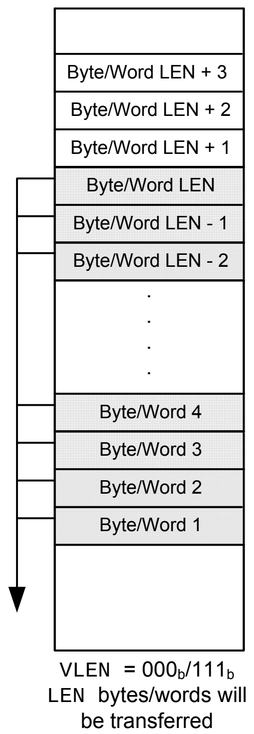

Figure 6. Fixed Length Transfer Count

5 Using the Wrong Configuration

The following two sections (Section 5.1 and Section 5.2) will show what will happen if there is a mismatch between the radio configuration and the DMA configuration with respect to how many bytes should be transferred. In RX mode, the radio will enter RX_OVERFLOW state if the radio tries to write more data to the `RFD` register than what the DMA will read from the same register. In TX mode, the radio will enter TX_UNDERFLOW state if the radio tries to read more data from the `RFD` register than what the DMA writes to the same register. The `RFIF.IRQ_RXOVF` flag will be asserted if RX_OVERFLOW state is entered while the `RFIF.IRQ_TXUVF` flag will be asserted when TX_UNDERFLOW state is entered. In both cases, the `RFIF.IRQ_DONE` flag will be set to 1.

5.1 Variable Packet Length Mode

In this section Example 1, page 7, will be used to show what might happens if the transfer count configuration of the DMA channel is not correct.

- RX Settings:
  - `PKTLEN` = n_{max} = 12
  - LEN = n_{max} + 1 = 12 + 1 = 13
  - VLEN = 100_b

Table 2 shows the transfer count, i.e. how many bytes the DMA will transfer, for different VLEN settings and different length bytes, n, given that LEN = 13. Maximum transfer counts for the different VLEN settings are emphasized in **bold**.

			Transfer Count			
	n	**LEN**	**VLEN = 001_b**	**VLEN = 010_b**	**VLEN = 011_b**	**VLEN = 100_b**
n < LEN	**1**	**13**	2	1	3	4
	2	**13**	3	2	4	5
	3	**13**	4	3	5	6
	4	**13**	5	4	6	7
	5	**13**	6	5	7	8
	6	**13**	7	6	8	9
	7	**13**	8	7	9	10
	8	**13**	9	8	10	11
	9	**13**	10	9	11	12
	10	**13**	11	10	12	13
	11	**13**	12	11	13	14
	12	**13**	**13**	12	**14**	**15**
n ≥ LEN	**13**	**13**	13	**13**	13	13
	14	**13**	13	13	13	13
	.	.	.	.	.	.
	.	.	.	.	.	.
	.	**13**	13	13	13	13
	.	.	.	.	.	.
	254	**13**	13	13	13	13
	255	**13**	13	13	13	13

Table 2. Transfer Count for Different VLEN Settings (LEN = 13 and n = {1, 2, . . , 255})

Using the wrong VLEN setting, VLEN = {001_b, 010_b, 011_b}, will cause the DMA to complete the transfer (transfer count reached) before the radio is done writing data to the `RFD` register. Remember that `PKTCTRL1.APPEND_STATUS` = 1_b, meaning that the radio will try to write n + 3 bytes to this register.

Figure 7 shows the largest packet that will pass the maximum length filtering implemented in the radio when `PKTLEN` = 12.

	n = 12														
Preamble + Sync Word	12	$Data_1$	$Data_2$	$Data_3$	$Data_4$	$Data_5$	$Data_6$	$Data_7$	$Data_8$	$Data_9$	$Data_{10}$	$Data_{11}$	$Data_{12}$	$Status_1$	$Status_0$
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Figure 7. Maximum Length Filtering

- VLEN = 001_b:

  Transfer count is reached after $Data_{12}$ has been transferred from the `RFD` register. Neither `RFIF.IRQ_DONE` nor `RFIF.IRQ_RXOVF` is being asserted, as one should expect. The radio will be stuck in RX state (`MARCSTATE` = 0x0D), but it will not be able to receive any more data (see the errata notes [3] and [4] for more details).

- VLEN = 010_b:

  Transfer count is reached after $Data_{11}$ has been transferred from the `RFD` register. Both `RFIF.IRQ_DONE` and `RFIF.IRQ_RXOVF` is being asserted. The radio will enter RX_OVERFLOW state (`MARCSTATE` = 0x11).

- VLEN = 011_b:

  Transfer count is reached after $Status_1$ has been transferred from the `RFD` register. Neither `RFIF.IRQ_DONE` nor `RFIF.IRQ_RXOVF` is being asserted, as one should expect. The radio will be stuck in RX state (`MARCSTATE` = 0x0D), but it will not be able to receive any more data (see the errata notes [3] and [4] for more details).

If the correct VLEN setting is used, VLEN = 100_b, but maximum length filtering is not used on the radio (`PKTLEN` = 255), all packets received with length byte n > 12 will cause the radio to enter RX_OVERFLOW state.

- TX Settings:
  - PKTLEN = $xxxxxxxx_b$ (don't care)
  - LEN = n_{max} + 1 = 255 + 1 = 256
  - VLEN = 001_b

	n	LEN	Transfer Count VLEN = 001_b	VLEN = 010_b	VLEN = 011_b	VLEN = 100_b
n < LEN	**1**	**256**	2	1	3	4
	2	**256**	3	2	4	5
	3	**256**	4	3	5	6
	4	**256**	5	4	6	7
	5	**256**	6	5	7	8
	6	**256**	7	6	8	9
	7	**256**	8	7	9	10
	8	**256**	9	8	10	11
	9	**256**	10	9	11	12
	10	**256**	11	10	12	13
	11	**256**	12	11	13	14
	12	**256**	13	12	14	15
	13	**256**	14	13	15	16
	14	**256**	15	14	16	17
	.	.	.	.	.	.
	.	.	.	.	.	.
	.	**256**	n + 1	n	n + 2	n + 3
	.	.	.	.	.	.
	254	**256**	255	254	256	257
	255	**256**	**256**	**255**	**257**	**258**

Table 3. Transfer Count for Different VLEN Settings (LEN = 256 and n = {1, 2, . . , 255})

The radio is configured to use variable packet length mode, and will always read n + 1 bytes from the RFD register.

Assume setting VLEN = 010_b. In this case the transfer count will be reached when the radio has one more byte left to transmit, and the radio will enter TX_UNDERFLOW state (RFIF.IRQ_TXUNF = 1_b and MARCSTATE = 0x16). The only way to proceed is by issuing an SIDLE strobe command (RFST = 0x04).

If using VLEN = {001_b, 011_b}, the radio might also enter TX_UNDERFLOW state, but not before transmitting the subsequent packet.

Consider the following pseudo code:

```
//-----------------------------------------------------------------------------
//-----------------------------------------------------------------------------
// Transmit one radio packet every time the button is being pushed.
void main (void)
{
    // Init
     .
     .
    // Init Radio
    // Init DMA
    // Enable RF interrupt (IRQ_DONE)

    while (TRUE)
    {
        // Wait for button to be pushed
        // Arm DMA
        // Strobe TX
    }
}
//-----------------------------------------------------------------------------
//-----------------------------------------------------------------------------
#pragma vector=RF_VECTOR
__interrupt void rf_IRQ(void)
{
    // Clear interrupt flag
    // Increment packet counter
}
//-----------------------------------------------------------------------------
//-----------------------------------------------------------------------------
```

The *txBuffer* containing the data to be transmitted is shown in Figure 8.

txBuffer	5	$Data_1$	$Data_2$	$Data_3$	$Data_4$	$Data_5$	$Data_6$	$Data_7$	$Data_8$	$Data_9$	$Data_{10}$	....	$Data_{254}$	$Data_{255}$
	1	2	3	4	5	6	7	8	9	10	11	12 254	255	256

Figure 8. *txBuffer*

When using the correct VLEN setting (VLEN = 001_b), the transfer count is n + 1 = 6. The radio will transmit 6 bytes, meaning that it will trigger the DMA 6 times. By pushing the button twice, the radio packet shown in Figure 9 will be sent twice.

Preamble + Sync Word	5	$Data_1$	$Data_2$	$Data_3$	$Data_4$	$Data_5$	CRC_1	CRC_0
	1	2	3	4	5	6	7	8

Figure 9. Packet Sent Twice when VLEN = 001_b

When VLEN = 011_b, the transfer count is 7. The radio, however, will only trigger the DMA 6 times. That means that the first time the button is pushed, the radio will transmit the packet shown in Figure 9 and an RF interrupt request will be generated. The DMA, however, has not yet reached its transfer count and awaits its last trigger; i.e. it is still armed (`DMAARN.DMAARMn` = 1_b and `DMAIRQ.DMAIFn` = 0_b). When the button is pushed the second time, the radio will trigger the DMA and the DMA will transfer the $Data_6$ byte (see Figure 8) to the `RFD` register. The DMA has now reached its transfer count and will be disarmed. The radio, however, will enter TX_UNDERFLOW state since it only received one byte from the DMA. The same scenario will occur when using VLEN = 011_b, but the DMA would transfer both $Data_6$ and $Data_7$ before being disarmed and causing the radio to enter TX_UNDERFLOW state.

Setting LEN < 256 when `PKTLEN` = 255 will cause the DMA to complete a transfer before the radio is done transmitting in cases where n ≥ LEN. This will make the radio enter TX_UNDERFLOW state.

5.2 Fixed Packet Length Mode

If the radio is configured for fixed packet length mode, as is the DMA, and `PKTLEN` < LEN (or `PKTLEN` + 2 < LEN in the case where `PKTCTRL1.APPEND_STATUS` = 1_b and the radio is in RX mode), the DMA will expect more triggers than what the radio will provide. For the following examples, assume that *txBuffer* has the content as shown in Figure 8.

- TX mode:

  The radio will be done transmitting a packet before the DMA transfer count is reached.

  - LEN modulo `PKTLEN` = 0:

    LEN / `PKTLEN` = x (integer division) radio packets will be sent. The transfer count is reached after packet number x has been sent.

    > *Note: The packets that are transmitted are not x equal packets, but x packets created from consecutive parts of txBuffer (see Figure 10).*

Figure 10. TX Mode; `PKTLEN` < LEN and (LEN modulo `PKTLEN`) = 0

  - LEN modulo `PKTLEN` = y (y ≠ 0):

    LEN / n = x (integer division) radio packets will be sent before packet number x + 1 is tried transmitted. After y bytes of this packet have been transmitted, the radio will enter TX_UNDERFLOW state (see Example 2).

    Example 2:

    LEN = 20 and `PKTLEN` = 7 →

    LEN / `PKTLEN` = 20 / 7 = 2

    LEN modulo `PKTLEN` = 20 modulo 7 = 6

    Two 7 bytes long packets will be sent. For packet number three, only six bytes will be sent on the air before transfer count is reached and the radio enters TX_UNDERFLOW state.

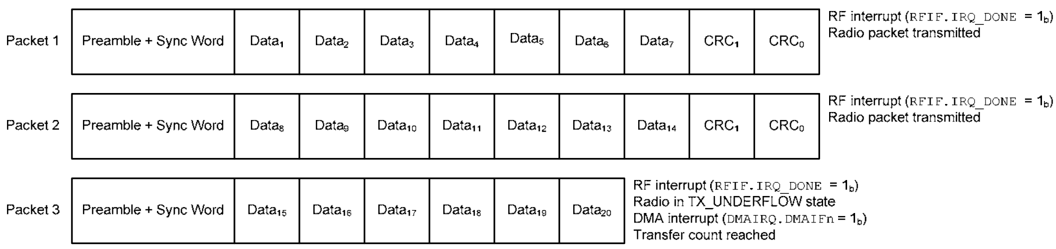

Figure 11. TX Mode; `PKTLEN` < LEN and (LEN modulo `PKTLEN`) ≠ 0

- RX mode:

  The radio will be done receiving a packet before the DMA transfer count is reached. If the radio is configured to enter IDLE state after a packet has been received (`MCSM1.RXOFF_MODE` = 00_b) and the application waits for the DMA interrupt before strobing RX again, the application will hang. If, however, the RF interrupt is used instead, or the radio is configured to stay in RX after a packet has been received (`MCSM1.RXOFF_MODE` = 11_b), the following will occur (assume that the transmitter is configured correctly and that the same packet is transmitted repeatedly):

  - `PKTCTRL1.APPEND_STATUS` = 0_b:
    - LEN modulo `PKTLEN` = 0:

      LEN / `PKTLEN` = x (integer division) radio packets will be received and moved to rxBuffer. The transfer count is reached after packet number x has been received.

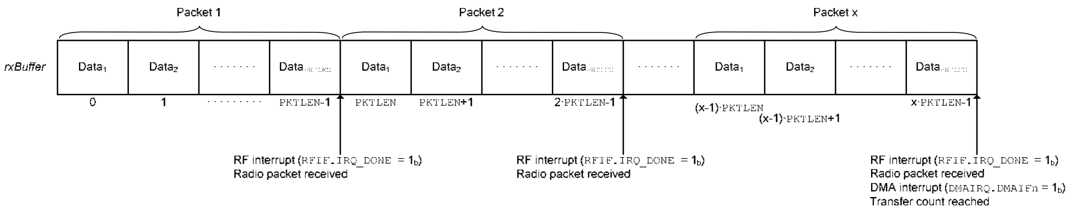

Figure 12. RX Mode; `PKTLEN` < LEN and (LEN modulo `PKTLEN`) = 0

    - LEN modulo `PKTLEN` = y (y ≠ 0):

      LEN / n = x (integer division) radio packets will be received before packet number x + 1 is tried received. After y bytes of this packet have been received, the radio will enter RX_OVERFLOW state since transfer count is reached in the middle of a packet (see Figure 13).

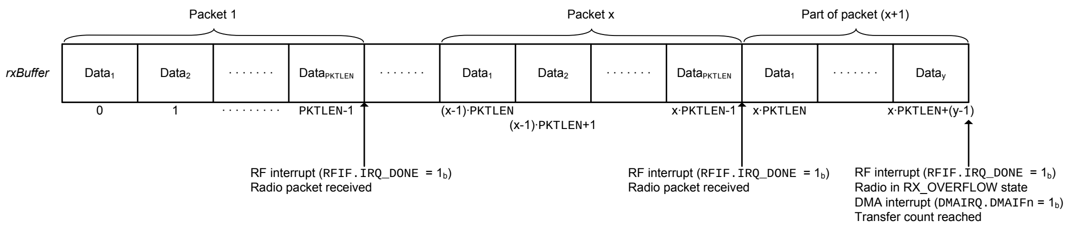

Figure 13. RX Mode; `PKTLEN` < LEN and (LEN modulo `PKTLEN`) ≠ 0

- o `PKTCTRL1.APPEND_STATUS` = 1_b:

Remember that when append status is enabled, LEN should be set to `PKTLEN` + 2.

- LEN = x·(`PKTLEN` + 2), where x is an integer:

  x radio packets will be received and moved to *rxBuffer*. The transfer count is reached after packet number x has been received. See Example 3 and Figure 14.

  <u>Example 3:</u>

  LEN = x·(`PKTLEN` + 2), `PKTLEN` = 3 and x = 3

  LEN = 15

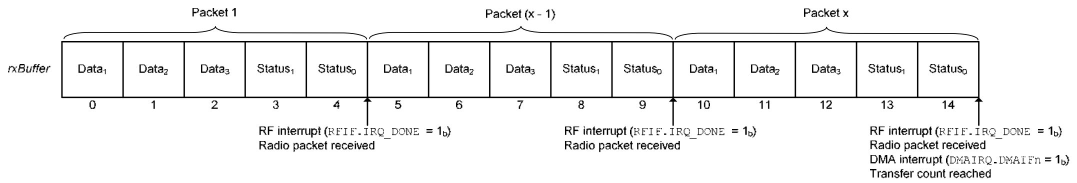

Figure 14. LEN = x·(`PKTLEN` + 2)

- LEN = x·(`PKTLEN` + 2) − 1 or LEN = x·(`PKTLEN` + 2) − 2, where x is an integer: x − 1 radio packets will be received before packet number x is tried received. Transfer count will be reached either right before the first status byte is moved to *rxBuffer* or right after it has been moved. Neither `RFIF.IRQ_DONE` nor `RFIF.IRQ_RX_OVF` is being asserted, and the radio will be stuck in RX state (`MARCSTATE` = 0x0D), but it will not be able to receive any more data (see the errata notes [3] and [4] for more details). See Example 4 and Figure 15.

  <u>Example 4:</u>

  LEN = x·(`PKTLEN` + 2) − 1, `PKTLEN` = 3 and x = 3

  LEN = 14

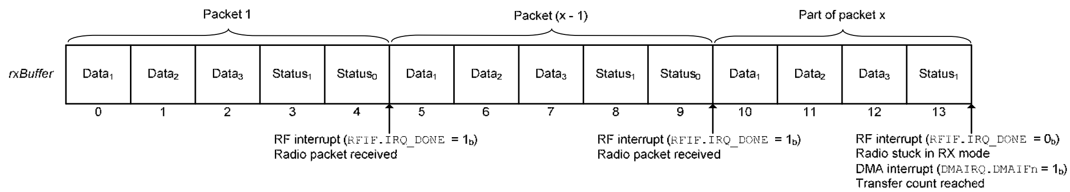

Figure 15. LEN = x·(`PKTLEN` + 2) − 1

- LEN = x·(`PKTLEN` + 2) − y, where 3 ≤ y ≤ (`PKTLEN` + 1) and x is an integer:

  x − 1 radio packets will be received before packet number x is tried received. When transfer count is reached, the radio will enter RX_OVERFLOW state. See Example 5 and Figure 16.

Example 5:

LEN = x·(PKTLEN + 2) − y, PKTLEN = 3, x = 3 and y = 3

LEN = 12

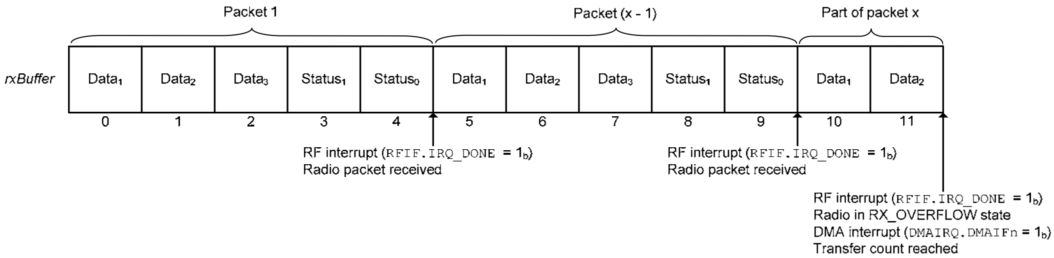

Figure 16. LEN = x·(PKTLEN + 2) − y

If PKTLEN > LEN (or PKTLEN + 2 > LEN in the case where PKTCTRL1.APPEND_STATUS = 1_b and the radio is in RX mode), the radio will enter TX_UNDERFLOW state from TX mode, and RX_OVERFLOW state from RX mode.

Note: In the case where the radio is in RX state and append status is enabled, the radio will get stuck in RX state if LEN = PKTLEN or LEN = PKTLEN + 1 (see the errata notes [3] and [4] for more details).

6 References

[1] CC1110Fx/CC1111Fx Low-Power SoC (System-on-Chip) with MCU, Memory, Sub-1 GHz RF Transceiver, and USB Controller (cc1110f32.pdf)

[2] CC2510Fx/CC2511Fx Low-Power SoC (System-on-Chip) with MCU, Memory, 2.4 GHz RF Transceiver, and USB Controller (cc2510f32.pdf)

[3] Errata Note CC1110Fx/CC1111Fx (swrz022.pdf)

[4] Errata Note CC2510Fx/CC2511Fx (swrz014.pdf)

7 General Information

7.1 Document History

Revision	Date	Description/Changes
SWRA164A	2009.12.16	Additional info about the DMA and radio interrupts added to Section 3.
SWRA164	2007.12.17	Initial release.