FIFO Mode, Direct Mode, and Packet Handler Operation for EZRadioPRO®

1. Introduction

This application note discusses the differences between FIFO mode and direct mode of operation of the EZRadioPRO® family of chips. A detailed discussion of the operation of the automatic Packet Handler (PH) is also provided. Details of operation in both TX mode and RX mode are provided. The purpose of this application note is to expand upon the information available in the data sheets for EZRadioPRO devices. This application note does ***not*** discuss Raw Data mode (as may be required by legacy systems with non-standard packet structures). Please refer to "AN463: Raw Data Mode with EZRadioPRO" for more information on such applications.

Throughout this document reference will be made to the general packet structure as shown in Figure 1.

Figure 1. General Packet Structure

2. FIFO Mode

In FIFO mode, the transmit and receive data is stored in integrated FIFO register memory for later transmission or retrieval via the SPI interface. Direct real-time output of the RXDATA on a physical output pin may *additionally* be obtained by configuring a GPIO pin appropriately, but it is not required. The FIFOs are accessed via SPI Register 7Fh and are most efficiently accessed with burst read/write operation.

2.1. FIFO Mode in TX

In TX FIFO mode, the TXDATA to be transmitted is not applied to a physical input pin, but instead is programmed (byte by byte) into the chip's FIFO registers over the SPI bus for transmission at a later point in time. If the PH is enabled, these stored bytes of data are retrieved from internal FIFO memory and "packaged" together with Preamble bytes and Sync word(s) to create a packet structure. With optional register settings, additional fields such as Header byte(s), a Payload length byte, and CRC checksums may additionally be assembled into the packet structure. If the PH is disabled, ***only*** the bytes stored into FIFO memory are transmitted; this allows for complete customization of the transmit packet. The entire packet is transmitted at the programmed data rate and deviation. A primary advantage of TX FIFO mode is thus that the transmit data may be programmed at ***any*** time (including sleep mode) without the need for synchronous or real-time coordination with the transmit burst. The chip is placed into FIFO mode by setting dtmod[1:0] bits D5-D4 of SPI Register 71h = 2'b10.

Register 71h. Modulation Mode Control 2

Bit	D7	D6	D5	D4	D3	D2	D1	D0
Name	trclk[1:0]		dtmod[1:0]		eninv	fd[8]	modtyp[1:0]	

Figure 2. SPI Register 71h, FIFO Mode

The chip may be configured to generate an interrupt upon completion of the transmit packet. The status of this interrupt may be found as the ipksent status bit D2 in SPI Register 03h. This interrupt status bit will cause an actual interrupt to occur on the nIRQ output pin if its corresponding enable bit (enpksent bit D2 of SPI Register 05h) is also set. This interrupt is available regardless of the status of the PH (i.e., enabled or disabled). However, the ipksent interrupt may be obtained only when the chip is in TX FIFO mode; in TX Direct mode the chip has no prior knowledge of the length of the packet and thus cannot automatically generate this signal.

Register 03h. Interrupt/Status 1

Bit	D7	D6	D5	D4	D3	D2	D1	D0
Name	ifferr	itxffafull	ixtffaem	irxffafull	iext	ipksent	ipkvalid	icrerror

Register 05h. Interrupt Enable 1

Bit	D7	D6	D5	D4	D3	D2	D1	D0
Name	enfferr	entxffafull	entxffaem	enrxffafull	enext	enpksent	enpkvalid	encrcerror

Figure 3. SPI Register 03h & 05h, Packet Sent Interrupt

Typical signals that may be observed during a TX packet in FIFO mode are shown in Figure 4.

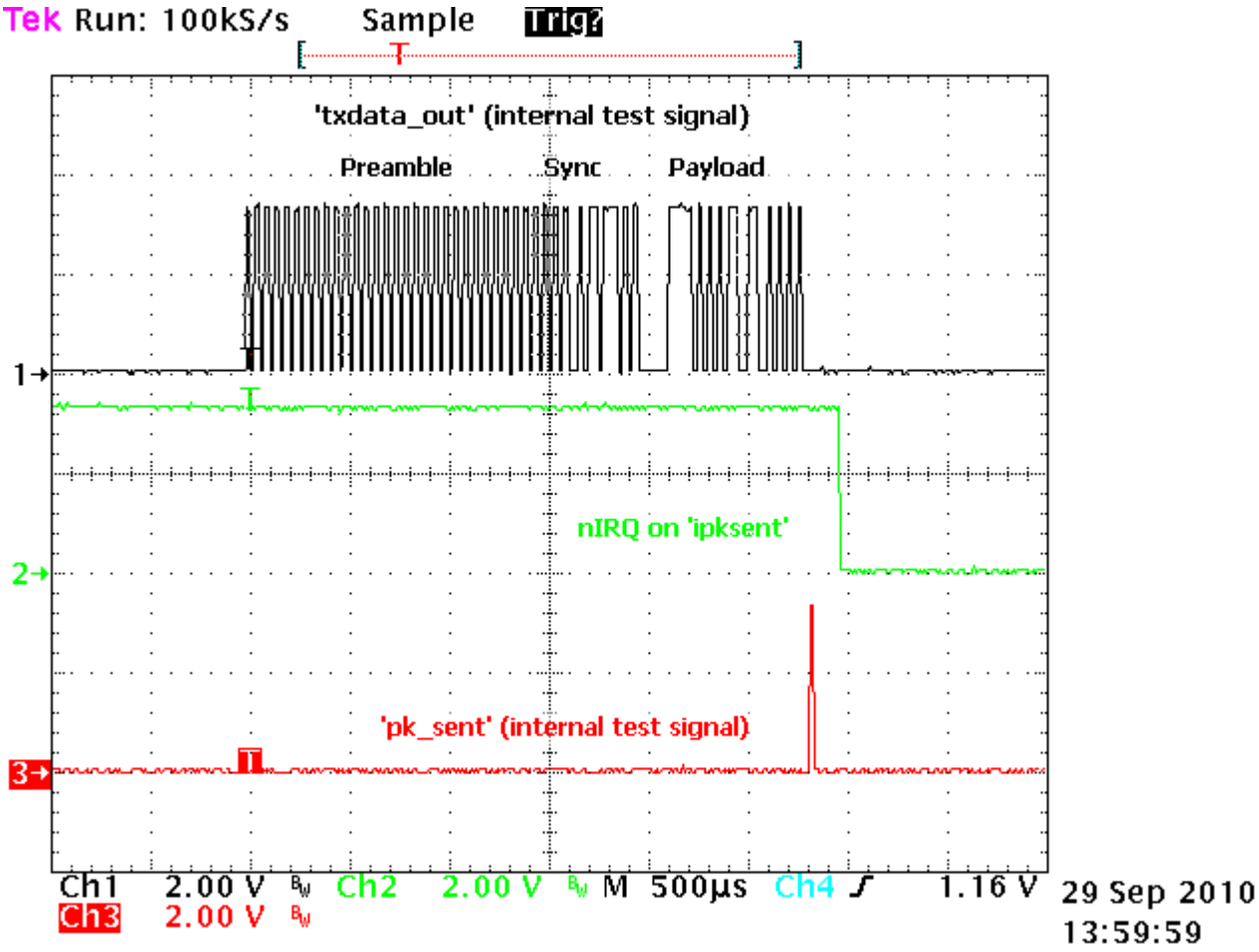


Figure 4. Typical TX FIFO Packet Signals

When in TX FIFO mode, the chip automatically exits the TX State after the ipksent interrupt occurs. The chip will return to the mode state selected by the remaining bits in SPI Register 07h. For example, the chip may be placed into TX mode by setting the txon bit, but with the xton bit additionally set (refer to Figure 5). The chip will transmit all of the contents of the FIFO and the ipksent interrupt will occur. When this interrupt event occurs, the chip will clear the txon bit and return to READY mode, as indicated by the set state of the xton bit. If the pllon bit D1 is set when entering TX mode (i.e., SPI Register 07h = 0Ah), the chip will exit from TX mode after sending the packet and return to TUNE mode.

However, the chip will not automatically return to STANDBY mode upon exit from the TX state, in the event the TX packet is initiated by setting SPI Register 07h = 08h (i.e., setting only txon bit D3). The chip will instead return to READY mode, with the crystal oscillator remaining enabled. This is intentional; the system may be configured such that the host MCU derives its clock from the MCU_CLK output of the RFIC (through GPIO2), and this clock signal must not be shut down without allowing the host MCU time to process any interrupt signals that may have occurred. The host MCU must subsequently perform a WRITE to SPI Register 07h = 00h to enter STANDBY mode and obtain minimum current consumption.

Register 07h. Operating Mode and Function Control 1

Bit	D7	D6	D5	D4	D3	D2	D1	D0
Name	swres	enlbd	enwt	x32ksel	txon	rxon	pllon	xton

Figure 5. SPI Register 07h, TX Mode and READY Mode

Retrieval and transmission of the data bytes from the TX FIFO varies, depending upon whether or not the PH is enabled. A complete discussion of the PH is provided in section “4. Automatic Packet Handler” ; in the following sections, only the effects of the PH upon the TX FIFO buffer are discussed.

2.1.1. TX FIFO Mode with Packet Handler Enabled

If the PH is enabled, the number of bytes retrieved from the TX FIFO and assembled into the Payload data field is ***always*** specified by the pklen[7:0] field in SPI Register 3Eh. Accordingly, the maximum possible Payload length when operating with the PH enabled is 255 bytes.

Register 3Eh. Packet Length

Bit	D7	D6	D5	D4	D3	D2	D1	D0
Name	pklen[7:0]							

Figure 6. SPI Register 3Eh, TX Packet Length

The length of the Payload is not simply determined by the total number of bytes programmed into the TX FIFO; the number of bytes retrieved from the TX FIFO may be less than the total number of bytes currently stored in TX FIFO memory. In such a case, the unused bytes remaining in the TX FIFO will simply be used in subsequent packet transmissions. This is shown in Figure 7. In this case, a total of 9 data bytes have been loaded into the TX FIFO while the pklen field is only set to pklen=3 bytes. These 9 data bytes are thus transmitted over three separate packets.

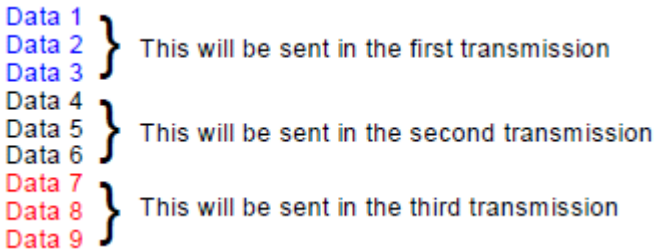


Figure 7. Multiple Packets in TX Packet Handler

The chip is capable of automatically retransmitting the last packet loaded in the TX FIFO. Automatic retransmission is set by entering the TX state (i.e., setting the txon bit) without reloading the TX FIFO. This feature is useful for beacon transmission or when retransmission is required due to the absence of a valid acknowledgement. Only packets that fit completely in the TX FIFO can be automatically retransmitted. For example, if only 3 data bytes have been loaded into the TX FIFO and the pklen field is also set to pklen=3 bytes, then the sequence of transmissions shown in Figure 8 will occur. Retransmission of a packet stored in the TX FIFO does ***not*** generate an ifferr interrupt (FIFO Error, bit D7 of SPI Register 03h); this is not considered an underflow condition of the TX FIFO buffer.

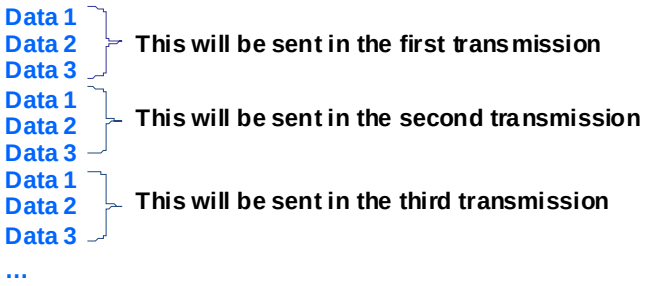


Figure 8. Automatic Retransmission of Packet

Any number of data bytes remaining in the TX FIFO that exactly matches the pklen field may be automatically retransmitted, even if the original number of bytes programmed was greater in length. For example, if 6 data bytes have been loaded into the TX FIFO and the pklen field is set to pkeln=3 bytes, the sequence of transmissions shown in Figure 9 will occur. Again, this is not considered an underflow condition and does not generate an ifferr interrupt.

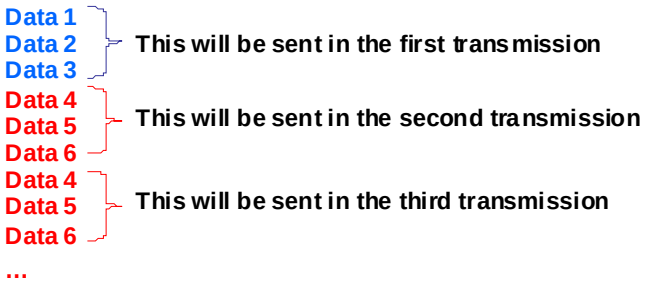


Figure 9. Automatic Retransmission of Remaining Packet

However, it is unwise to attempt to retrieve more bytes from the TX FIFO than have been programmed into the TX FIFO. In such a case, the pointer in the TX FIFO will underflow and corrupted data will occur. This will result in generation of an ifferr interrupt in bit D7 of SPI Register 03h. For example, if 5 data bytes have been loaded into the TX FIFO and the pklen field is set to pklen=3 bytes, the sequence of transmissions shown in Figure 10 will occur.

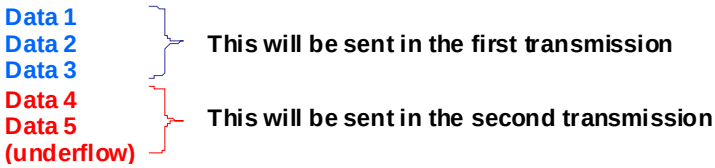


Figure 10. Underflow of TX FIFO Buffer

The first 3 bytes are sent with the first packet in normal fashion; however, attempting to retrieve a sixth byte for transmission on the second packet results in underflow of the buffer.

2.1.2. TX FIFO Mode with Packet Handler Disabled

If the PH is disabled, any value programmed into the pklen[7:0] field is completely ignored. The number of bytes retrieved from the TX FIFO and assembled into the Payload data field is simply equal to the number of bytes programmed into the TX FIFO. That is, bytes continue to be retrieved from TX FIFO memory and transmitted until the TX FIFO buffer is empty. If the MCU continues to re-fill the TX FIFO with more data bytes during transmission, packets of essentially infinite length may be constructed. After transmission of the last byte in TX FIFO memory, the ipksent signal is issued and the packet will terminate. Note that the ipksent status signal remains available even if the PH is disabled, as it is possible to construct and send the entire packet from TX FIFO memory.

As the last byte in the TX FIFO buffer is retrieved and transmitted, an underflow condition is generated and will thus generate an ifferr interrupt in bit D7 of SPI Register 03h. This is considered normal operation.

2.1.3. TX FIFO Thresholds

The TX FIFO is 64 bytes in length. Packets of greater length may be accommodated by making use of the programmable thresholds. The TX FIFO has two such programmable thresholds. An interrupt event occurs when the data in the TX FIFO reaches these thresholds.

Register 7Ch. TX FIFO Control 1

Bit	D7	D6	D5	D4	D3	D2	D1	D0
Name	Reserved		txafthr[5:0]					

Register 7Dh. TX FIFO Control 2

Bit	D7	D6	D5	D4	D3	D2	D1	D0
Name	Reserved		txfaethr[5:0]					

Figure 11. SPI Register 7Ch-7Dh, TX FIFO Thresholds

The first threshold is the TX FIFO Almost Full Threshold txafthr[5:0] in SPI Register 7Ch. The value in this register corresponds to the desired threshold value in number of bytes, minus 1 byte. When the data being filled into the TX FIFO crosses this threshold limit, an interrupt to the microcontroller is generated so the chip can enter TX mode to transmit the contents of the TX FIFO. As an example, if SPI Register 7Ch is programmed with a value of 0x3C=60d, the itxffafull interrupt will not be generated if 60 bytes (or less) are written to the TX FIFO, but will be generated if 61 bytes (or more) are written to the TX FIFO.

The second threshold is the TX FIFO Almost Empty Threshold txaethr[5:0] in SPI Register 7Dh. An interrupt will be generated when the number of data bytes remaining for transmission in the TX FIFO drops below 'txfaethr[5:0] + 1'. As an example, if SPI Register 7Dh is programmed with a value of 0x05, the itxffaem interrupt will be generated when 6 or less bytes remain in the TX FIFO. The microcontroller must switch out of TX mode or fill more data into the TX FIFO.

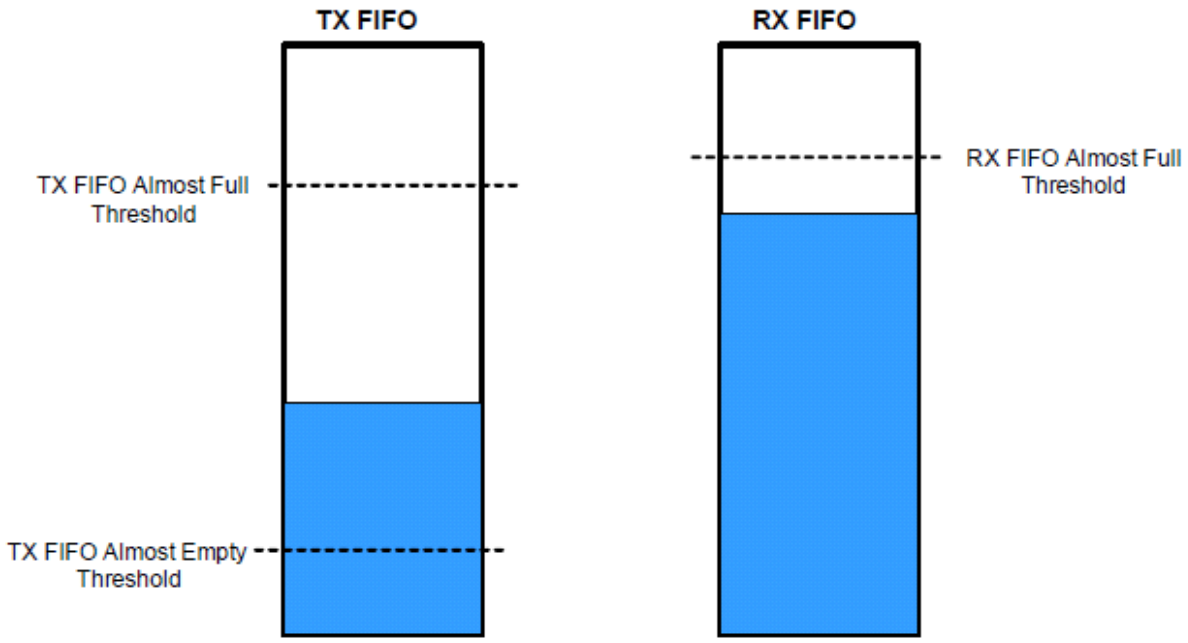


Figure 12. FIFO Thresholds

By using these interrupts to signal the host MCU when it is necessary to fill or empty the FIFO, a Payload length exceeding 64 bytes may be constructed.

As discussed in “2.1. FIFO Mode in TX” , the chip can be configured so that when the TX FIFO is empty it will automatically exit the TX state. This is accomplished by setting or clearing the appropriate mode control bits in SPI Register 07h.

2.1.4. Clearing the TX FIFO

The contents of the TX FIFO may be cleared by first setting, and then clearing, the ffclrtx bit D0 of SPI Register 08h. This bit is not self-clearing, two successive WRITE operations must be made to this register.

Register 08h. Operating Mode and Function Control 2

Bit	D7	D6	D5	D4	D3	D2	D1	D0
Name	antdiv[2:0]			rxmpk	autotx	enldm	ffclrrx	ffclrtx

Figure 13. SPI Register 08h, Clear TX FIFO

2.2. FIFO Mode in RX

In RX FIFO mode, the received RXDATA is not required to be output on a physical output pin, but instead is retrieved (byte by byte) from the chip's RX FIFO registers over the SPI bus. A primary advantage of RX FIFO mode is thus that the receive data may be retrieved at **any** time (including sleep mode) without the need for synchronous or real-time coordination with the receive burst. The chip is placed into FIFO mode without setting dtmod[1:0] bits D5-D4 of SPI Register 71h=2'b10 (see Figure 2). Direct real-time output of the RXDATA on a physical output pin may *additionally* be obtained by configuring a GPIO pin appropriately, but it is not required.

In RX mode, only the bytes of the received packet structure that are considered to be "data bytes" are stored in RX FIFO memory. The PH functionality (if enabled) determines which bytes of the received packet are considered data bytes. If the PH is disabled, all bytes following the Sync word are considered data bytes and are stored in RX FIFO memory. Thus the preamble detection threshold and Sync word must still be programmed so that the RX Modem knows when to start filling data into the FIFO, even if operation with the PH is not desired. If the PH is enabled, certain fields after the Sync word such as the Header and CRC bytes are automatically stripped off and only the data bytes of the Payload are stored in RX FIFO memory. In both cases, neither the bytes of the Preamble or Sync fields are ever stored in FIFO memory; only the bytes considered to be "data" are stored to FIFO memory.

The chip may be configured to generate an interrupt upon reception of a valid packet. The status of this interrupt may be found as the ipkvalid status bit D1 in SPI Register 03h. This interrupt status bit will cause an actual interrupt to occur on the nIRQ output pin if its corresponding enable bit (enpkvalid bit D1 of SPI Register 05h) is also set. This interrupt is available only if the PH is enabled; when operating with the PH disabled, the chip has no knowledge of the length of the packet and thus cannot automatically generate this signal.

Register 03h. Interrupt/Status 1

Bit	D7	D6	D5	D4	D3	D2	D1	D0
Name	ifferr	itxffafull	ixtffaem	irxffafull	iext	ipksent	ipkvalid	icrerror

Register 05h. Interrupt Enable 1

Bit	D7	D6	D5	D4	D3	D2	D1	D0
Name	enfferr	entxffafull	entxffaem	enrxffafull	enext	enpksent	enpkvalid	encrcerror

Figure 14. SPI Register 03h & 05h, Packet Valid Interrupt

Typical signals that may be observed during an RX packet in FIFO mode are shown in Figure 15.

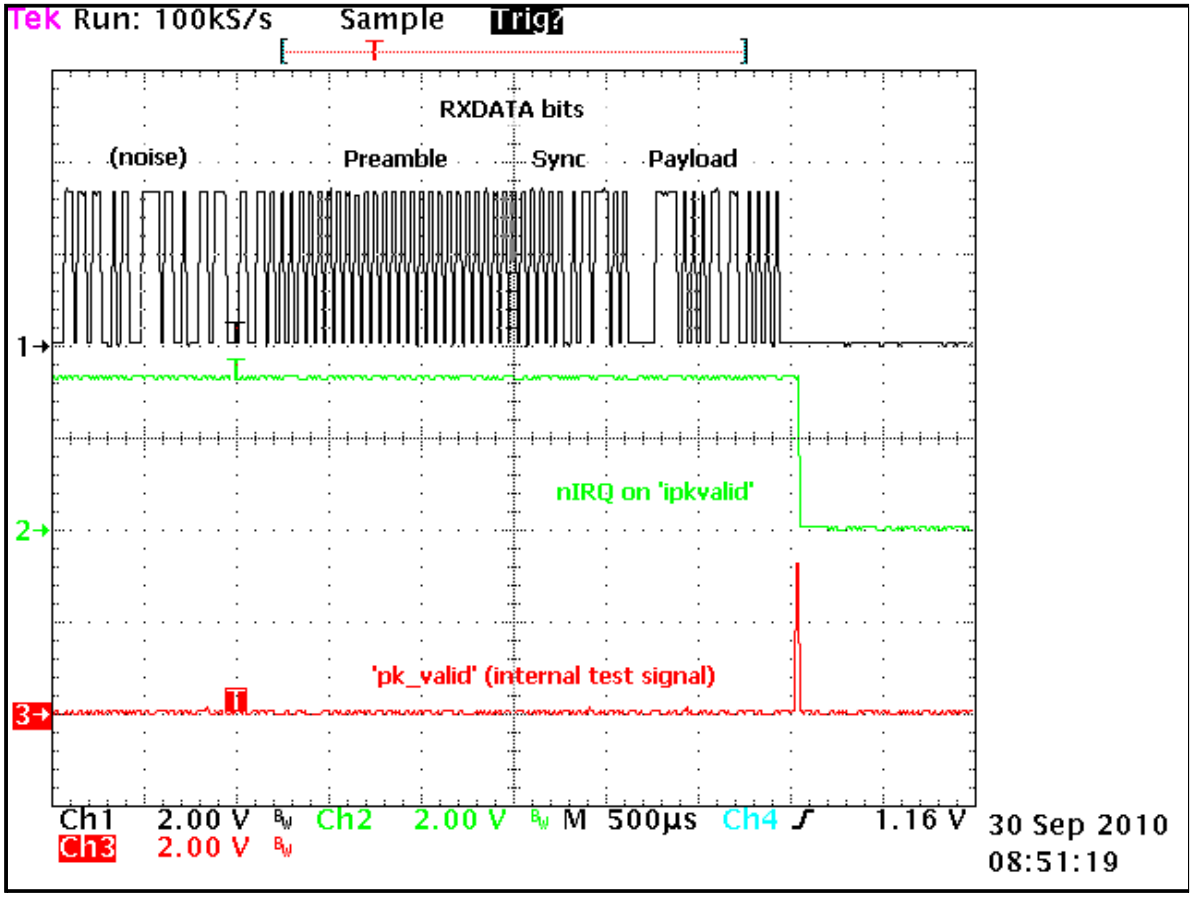


Figure 15. Typical RX FIFO Packet Signals

When in RX FIFO mode with PH enabled, the chip automatically exits the RX State after the ipkvalid interrupt occurs. The chip will return to the mode state selected by the remaining bits in SPI Register 07h. For example, the chip may be placed into RX mode by setting the rxon bit, but with the xton bit additionally set (refer to Figure 5). The chip will receive the packet and the ipkvalid interrupt will occur. When this interrupt event occurs, the chip will clear the rxon bit and return to READY mode, as indicated by the set state of the xton bit. If the pllon bit D1 is set when entering RX mode (i.e., SPI Register 07h = 06h), the chip will exit from RX mode after receiving the packet and return to TUNE mode.

However, the chip will not automatically return to STANDBY mode upon exit from the RX state, in the event the RX packet reception is initiated by setting SPI Register 07h = 04h (i.e., setting only rxon bit D2). The chip will instead return to READY mode, with the crystal oscillator remaining enabled. This is intentional; the system may be configured such that the host MCU derives its clock from the MCU_CLK output of the RFIC (through GPIO2), and this clock signal must not be shut down without allowing the host MCU time to process any interrupt signals that may have occurred. The host MCU must subsequently perform a WRITE to SPI Register 07h = 00h to enter STANDBY mode and obtain minimum current consumption.

In the event that an ipkvalid interrupt does not occur (e.g., if a CRC checksum error occurs), the device will remain in RX mode and does not turn off automatically. The host MCU must process the reason for the invalid packet and subsequently turn of the RFIC by SPI command.

2.2.1. RX FIFO Mode with Packet Handler Enabled

When the PH is enabled, all the fields of the receive packet structure need to be configured. Inclusion of certain fields (e.g., Header, packet length, and CRC) in the packet structure is optional; however, it remains necessary to configure these fields for zero length in the event they are excluded. The PH processes and strips off these additional fields, storing only the data bytes in RX FIFO memory. It can be configured to handle packets of fixed or variable length, with or without a Header and/or CRC. The PH may even be configured for storing multiple packets into the RX FIFO. Further detail regarding the operation of the Packet Handler and the function of its associated register fields may be found in section “4. Automatic Packet Handler” .

Figure 16. Packet Structure with Packet Handler Enabled

If the PH is enabled, the receiver can automatically determine the end of the packet as it has knowledge of the expected packet length. Upon reception of a valid packet, the chip automatically drops of out of receive mode and returns to the residual state (i.e., TUNE mode or READY mode) additionally programmed into SPI Register 07h.

2.2.2. RX FIFO Mode with Packet Handler Disabled

When the PH is disabled, the Preamble and Sync fields in the received packet are still required as the modem needs these fields for proper operation. However, any bits occurring after the Sync word are treated as raw data, and are stored into RX FIFO memory with no further qualification or processing. This mode allows for the creation of a custom packet handler in the host MCU when the automatic qualification parameters in the RFIC are not sufficient. Specifically, such functions as data whitening, CRC, and Header checks are not supported, while Manchester encoding/decoding across the data field remains supported.

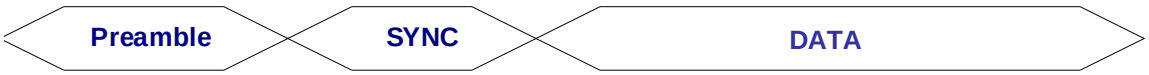


Figure 17. Packet Structure with Packet Handler Disabled

If the PH is disabled, the receiver can no longer automatically determine the end of the packet. Thus the chip remains in receive mode until the host MCU explicitly commands the chip to exit the RX state. Data bits are continually stored into the RX FIFO until the chip exits from RX mode, and therefore it is important for the user to correctly configure the RX FIFO threshold appropriately. The MCU must retrieve data bytes from the RX FIFO in a timely fashion, else an RX FIFO overflow condition may occur.

2.2.3. RX FIFO Threshold

The RX FIFO is also 64 bytes in length (see Figure 12). Packets of greater length may be accommodated by making use of a programmable threshold. The RX FIFO has one such programmable threshold, called the RX FIFO Almost Full Threshold rxafthr[5:0] in SPI Register 7Eh. The value in this register corresponds to the desired threshold value in number of bytes, minus 1 byte. An interrupt will be generated when the number of data bytes received and stored in the RX FIFO exceeds this threshold value. As an example, if SPI Register 7Eh is programmed with a value of 0x3C=60d, the itxffafull interrupt will not be generated if 60 bytes (or less) are received and stored in the RX FIFO, but will be generated when 61 bytes (or more) are received and stored in the RX FIFO. The interrupt signals the microcontroller to read the data bytes from the FIFO via the SPI bus.

Register 7Eh. RX FIFO Control

Bit	D7	D6	D5	D4	D3	D2	D1	D0
Name	Reserved		rxafthr[5:0]					

Figure 18. SPI Register 7Eh, RX FIFO Threshold

There is no RX FIFO Almost Empty threshold parameter, as there is for the TX FIFO. The device already knows the number of receive data bytes, either from receiving the packet length byte in the Header or by fixing the receive packet length and specifying its length with the pklen[7:0] field in SPI Register 3Eh. As a result, there is no need to detect when the last byte has been retrieved from the RX FIFO.

2.2.4. Clearing the RX FIFO

The contents of the RX FIFO may be cleared by first setting, and then clearing, the ffclrrx bit D1 of SPI Register 08h. This bit is not self-clearing; two successive WRITE operations must be made to this register.

Register 08h. Operating Mode and Function Control 2

Bit	D7	D6	D5	D4	D3	D2	D1	D0
Name	antdiv[2:0]			rxmpk	autotx	enldm	ffclrrx	ffclrtx

Figure 19. SPI Register 08h, Clear RX FIFO

3. Direct Mode

The main difference between Direct mode and FIFO mode (in both TX and RX) is that in Direct mode the data bit streams are input to and output from the chip in real-time on a physical I/O pin, as compared to performing READs and WRITEs of the data bytes over the SPI bus in FIFO mode. This is often useful in legacy systems that do not use a conventional packet structure, or when the host MCU must perform customized packet handling or processing.

3.1. Direct Mode in TX

In TX Direct mode, the TX modulation data is applied to an input pin of the chip and processed in “real time” (i.e., not stored in a register for transmission at a later time). A variety of pins (e.g., all three GPIO pins and the SDI pin) may be configured for use as the TX Data input function. The source of the TX data (GPIO pin or SDI pin) is selected by the dtmod[1:0] field of SPI Register 71h.

Furthermore, an additional pin **may** be required for a TX Clock output function. This TX Clock output pin is required only if GFSK modulation is desired; in the case of FSK or OOK modulation it may be used, but it is not required. A variety of pins (e.g., all three GPIO pins, SDI pin, SDO pin, and nIRQ pin) may be configured for use as the TX Clock output function. The TX Clock output pin is selected by the trclk[1:0] field in SPI Register 71h.

Register 71h. Modulation Mode Control 2

Bit	D7	D6	D5	D4	D3	D2	D1	D0
Name	trclk[1:0]		dtmod[1:0]		eninv	fd[8]	modtyp[1:0]	

Figure 20. SPI Register 71h, Modulation Control

3.1.1. Synchronous Direct Mode in TX

In TX Synchronous Direct Mode, the RFIC is configured to provide a TX Clock signal as an ***output*** to the external device that is providing the TX Data stream. This TX Clock signal is a square wave with a frequency equal to the programmed transmit data rate. The external modulation source (e.g., the host MCU) must accept this TX Clock signal as an input and respond by providing one bit of TX Data back to the RFIC, synchronous with one edge of the TX Cock signal. In this fashion, the rate of the TX Data input stream from the external source is controlled by the programmed data rate of the RFIC; no TX Data bits are made available at the input of the RFIC until requested by another cycle of the TX Clock signal. The TX Data bits supplied by the external source are transmitted by the RFIC directly in real-time (i.e., not stored internally for later transmission). Synchronous Direct mode is selected by setting trclk[1:0] to any value other than 2’b00; the exact value of trclk[1:0] simply determines which pin is used for the TX Clock output function.

All modulation types (FSK/GFSK/OOK) are supported in TX Synchronous Direct Mode. Synchronous mode ***must*** be used if GFSK modulation is desired while using Direct mode; GFSK modulation is not possible in Asynchronous Direct mode. In Asynchronous Direct mode, the chip does not have control or knowledge of the transmit data rate, and thus cannot apply the appropriate Gaussian filtering function.

3.1.2. Asynchronous Direct Mode in TX

In TX Asynchronous Direct mode, the RFIC no longer controls the data rate of the TX Data input stream. Instead, the data rate is controlled only by the external TX Data source; the RFIC simply accepts the data applied to its TX Data input pin, at whatever rate it is supplied. This means that there is no longer a need for a TX Clock output signal from the RFIC, as there is no synchronous "handshaking" between the RFIC and the external data source. The TX Data bits supplied by the external source are again transmitted directly in real-time, and are not stored internally for later transmission. Asynchronous Direct mode is selected by setting trclk[1:0] = 2’b00.

It is not necessary to program the transmit data rate parameter when operating in TX Asynchronous Direct mode. The chip still internally samples the incoming TX Data stream to determine when edge transitions occur; however, rather than sampling the data at a pre-programmed data rate, the chip now internally samples the incoming TX

Data stream at its maximum possible oversampling rate. This allows the chip to accurately determine the timing of the bit edge transitions without prior knowledge of the data rate. Of course, it remains necessary to program the desired peak frequency deviation.

As the chip does not have knowledge of the data rate of the TX Data input stream, it cannot apply an appropriate Gaussian filtering function and thus GFSK modulation is not available in TX Asynchronous Direct mode. However, FSK and OOK modulation remains available in this mode.

3.2. Direct Mode in RX

In RX Direct mode, the received RXDATA is output on a physical output pin in real-time. The automatic PH may be either enabled or disabled in RX Direct mode; however, the primary reason for using RX Direct mode is the need for handling non-standard packet structures that cannot be processed by the PH in the RFIC. In such a case, the PH in the RFIC is disabled and this function is performed in the MCU. Because use of RX Direct mode is usually accompanied by disabling the PH, it is often assumed that the PH ***must*** be disabled in RX Direct mode. This is technically not correct. However, because this is the typical use case for RX Direct mode, the examples below assume the PH is disabled.

Unlike TX Direct mode, there are no configuration bits that place the chip into RX Direct mode. Specifically, the dtmod[1:0] bits D5-D4 of SPI Register 71h (see Figure 2) are only concerned with configuring the chip for TX Direct or TX/RX FIFO mode; they do not have similar functionality for RX Direct mode. All that is required to place the chip into RX Direct mode is to configure one of the three GPIO pins as the RX Data output function. This causes all of the RXDATA output stream to appear on the selected output pin, where it may be further processed by the host MCU. (It is common to simultaneously configure an additional GPIO output pin for use as the RX Clock output function. In this fashion, the host MCU may use the rising edge of the RXCLOCK signal to sample the RXDATA signal.)

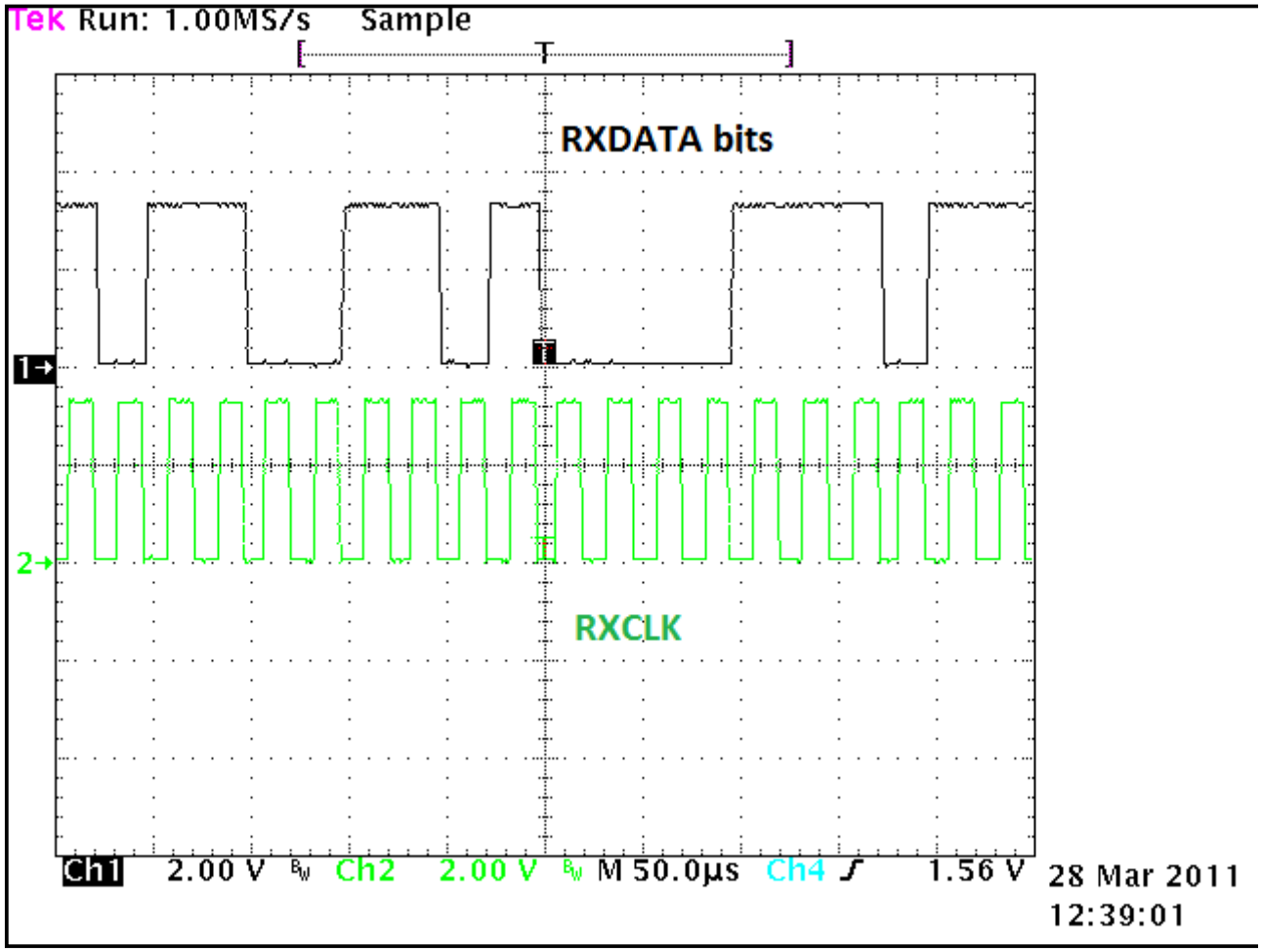


Figure 21. Typical RXDATA and RXCLOCK Output Signals in RX Direct Mode

All of the received data bytes are output onto the selected RX Data output pin, including Preamble and Sync bytes. In RX Direct mode, no attempt is made to strip off non-Payload bytes; the host MCU must separate the Payload bytes from the remainder of the packet structure. The Preamble and Sync fields must still be present in the packet structure, as the RFIC relies upon these fields to acquire bit timing, perform automatic frequency correction (AFC), and to qualify a valid signal.

The automatic PH is typically not used in RX Direct mode, and thus the chip has no knowledge of the packet structure (such as packet length or the presence of any CRC checksum bytes). As a result, the chip has no way of determining if a valid packet has been received. The user should therefore not expect to obtain an interrupt or status bit indicating the validity (or invalidity) of the packet. Specifically, the ipkvalid status bit D1 in SPI Register 03h is not available in this mode. Again, the non-functionality of this interrupt accompanies disabling of the PH and is not due to operation in RX Direct mode; it is simply assumed that the reason for operation in RX Direct mode precludes use of the automatic PH.

As the packet handling functionality is performed by the host MCU, it is also the responsibility of the MCU to determine when the end of the packet occurs and to subsequently turn of the RFIC by SPI command. That is to say, the RFIC cannot automatically drop out of RX mode upon receiving a valid packet, as it has no way to determine when that end of packet occurs.

4. Automatic Packet Handler

The EZRadioPRO family of chips contains circuit functionality known as the automatic Packet Handler (PH). The purpose of the PH is to automatically perform basic packet structure construction (in TX mode) or deconstruction (in RX mode), without the need for MCU control or intervention. The usual fields needed for packet generation (such as Preamble, Sync word, TX Header, Packet Length, and CRC) normally change infrequently and can therefore be stored in registers. Automatically adding these fields to the Payload data greatly reduces the required computational power of the MCU, allowing use of a less-complex (i.e., cheaper) MCU.

The functionality of the PH may be separately enabled or disabled in either TX or RX mode (or both). Enabling of the PH functionality is provided in SPI Register 30h by the enpacrx bit D7 and enpactx bit D3. It is important to note that the PH may be enabled in TX mode but not RX mode, or vice versa, if desired.

Register 30h. Data Access Control

Bit	D7	D6	D5	D4	D3	D2	D1	D0
Name	enpacrx	lsbfrst	crcdonly	skip2ph	enpactx	encrc	crc[1:0]	

Figure 22. SPI Register 30h, Control of Packet Handler

The functionality of the PH includes:

- Detection/validation of Preamble quality in RX mode (PREAMBLE_VALID signal)
- Detection of Sync word in RX mode (SYNC_OK signal)
- Detection of valid packets in RX mode (PKT_VALID signal)
- Detection of CRC errors in RX mode (CRC_ERR signal)
- Storage of Payload Data bytes into FIFO memory in RX mode
- Construction of Preamble field in TX mode
- Construction of Sync field in TX mode
- Construction of TX Header field (if enabled) in TX mode
- Construction of Packet Length field (if enabled) in TX mode
- Construction of Payload Data field from FIFO memory in TX mode
- Construction of CRC field (if enabled) in TX mode
- Data whitening and/or Manchester encoding (if enabled) in TX mode

4.1. Operation of Packet Handler in TX Mode

In TX mode, operation of the PH has no meaning ***unless*** the chip is also commanded to FIFO mode. This is self-evident; if the bits of the transmit data stream are provided by the host MCU in real-time on a physical input pin, the entire structure of the transmit packet is already determined and there is no additional functionality for the PH to provide. Thus the true benefit of the PH in TX mode is only realized when the Payload data bytes are programmed into FIFO memory, and the remainder of the packet structure is automatically constructed using the PH functionality. The functionality described below assumes that the PH has been enabled by setting the enpactx bit D3 of SPI Register 30h (refer to Figure 22).

4.1.1. Preamble Field

The PH automatically constructs a Preamble field with length specified by the prealen[8:0] field in SPI Registers 33h-34h.

Register 33h. Header Control 2

Bit	D7	D6	D5	D4	D3	D2	D1	D0
Name	skipsyn	hdlen[2:0]			fixpklen	synclen[1:0]		prealen[8]

Register 34h. Preamble Length

Bit	D7	D6	D5	D4	D3	D2	D1	D0
Name	prealen[7:0]							

Figure 23. SPI Registers 33h-34h, Preamble Length Field

The value in the prealen[8:0] field corresponds to the number of nibbles (4 bits) in the Preamble. For example, a value of prealen = 0x008 corresponds to a length of 32 bits (8 x 4 bits), or a total of 4 bytes. The maximum length is 511 nibbles, or 255.5 bytes. It is not possible to obtain a Preamble of length zero; setting prealen = 0x000 will have the same result as writing prealen = 0x001, which corresponds to one single nibble (4 bits) of Preamble. Thus if the user desires no Preamble at all, or a Preamble whose length exceeds 255.5 bytes, it is not possible to use the PH. In such a case, it will be necessary to disable the PH and to program the ***entire*** packet structure in the FIFO memory, or to construct the entire packet structure in the host MCU and to provide it to the RFIC in real-time over a physical input pin (i.e., TX Direct mode).

It should be noted that the first transmitted bit of the Preamble field is always a “0”. That is, the bit format of the Preamble field is “010101…” rather than “101010…”.

The prealen[8:0] field is only used in TX mode. There is no purpose to specifying a Preamble length in RX mode, as the chip does not use prior knowledge of the length of the Preamble field in determining when the Sync field begins.

4.1.2. Sync Field

The PH automatically constructs a Sync field of length specified by the synclen[1:0] field in SPI Register 33h.

Register 33h. Header Control 2

Bit	D7	D6	D5	D4	D3	D2	D1	D0
Name	skipsyn	hdlen[2:0]			fixpklen	synclen[1:0]		prealen[8]

Figure 24. SPI Register 33h, Sync Length Field

The value in the synclen[1:0] field corresponds to the number of bytes in the Sync word, minus one byte. The number of bytes in the Sync word can range from 1 to 4 bytes. For example, a value of synclen = 2’b00 corresponds to a length of 1 byte, while a value of synclen = 2’b11 corresponds to a length of 4 bytes. It is not possible to obtain a Sync word of length zero. Thus if the user desires no Sync word at all, or a Sync word whose length exceeds 4 bytes, it is not possible to use the PH. In such a case, it will be necessary to disable the PH and to program the ***entire*** packet structure (including Preamble and Sync word) in the FIFO memory, or to construct the entire packet structure in the host MCU and to provide it to the RFIC in real-time over a physical input pin.

The values of the Sync word byte(s) are specified in SPI Registers 36h-39h.

Register 36h. Synchronization Word 3

Bit	D7	D6	D5	D4	D3	D2	D1	D0
Name	sync[31:24]							
Type	R/W							

Register 37h. Synchronization Word 2

Bit	D7	D6	D5	D4	D3	D2	D1	D0
Name	sync[23:16]							
Type	R/W							

Register 38h. Synchronization Word 1

Bit	D7	D6	D5	D4	D3	D2	D1	D0
Name	sync[15:8]							
Type	R/W							

Register 39h. Synchronization Word 0

Bit	D7	D6	D5	D4	D3	D2	D1	D0
Name	sync[7:0]							
Type	R/W							

Figure 25. SPI Registers 36h-39h, Sync Word Value(s)

The Sync word bytes are transmitted/received in descending order. Thus if synclen = 2'b00 (corresponding to a Sync word of 1 byte length), then only Sync Word 3 (Reg 36h) is transmitted. If synclen = 2'b01 (corresponding to a Sync word of 2 bytes length), then Sync Word 3 (Reg 36h) is transmitted first, followed by Sync Word 2 (Reg 37h), and so on. Values programmed into Sync Word registers (36h-39h) are ignored if the synclen field is not configured for a Sync word length that requires use of that Sync Word register.

4.1.3. TX Header Field

The PH automatically constructs a TX Header field of length specified by the hdlen[2:0] field in SPI Register 33h.

Register 33h. Header Control 2

Bit	D7	D6	D5	D4	D3	D2	D1	D0
Name	skipsyn	hdlen[2:0]			fixpklen	synclen[1:0]		prealen[8]

Figure 26. SPI Register 33h, Header Length Field

The value in the hdlen[2:0] field corresponds to the number of bytes in the Header. The number of bytes in the TX Header can range from 0 to 4 bytes. For example, a value of hdlen = 3'b000 corresponds to no TX/RX Header (length of 0 bytes), a value of hdlen = 3'b001 corresponds to a length of 1 byte, and so on up to hdlen = 3'b100 corresponding to a maximum TX Header length of 4 bytes. Setting the hdlen[2:0] field to a value larger than hdlen = 3'b100 will have the same result as writing hdlen = 3'b100.

The values of the Transmit Header word byte(s) are specified in SPI Registers 3Ah-3Dh.

Register 3Ah. Transmit Header 3

Bit	D7	D6	D5	D4	D3	D2	D1	D0
Name	txhd[31:24]							

Register 3Bh. Transmit Header 2

Bit	D7	D6	D5	D4	D3	D2	D1	D0
Name	txhd[23:16]							

Register 3Ch. Transmit Header 1

Bit	D7	D6	D5	D4	D3	D2	D1	D0
Name	txhd[15:8]							

Register 3Dh. Transmit Header 0

Bit	D7	D6	D5	D4	D3	D2	D1	D0
Name	txhd[7:0]							

Figure 27. SPI Registers 3Ah-3Dh, Transmit Header Word Value(s)

The TX Header bytes are transmitted/received in descending order. Thus if hdlen=3'b001 (corresponding to a TX Header of 1 byte length), then only Transmit Header 3 (Reg 3Ah) is transmitted. If hdlen = 3'b010 (corresponding to a TX Header of 2 bytes length), then Transmit Header 3 (Reg 3Ah) is transmitted first, followed by Transmit Header 2 (Reg 3Bh), and so on. Values programmed into Transmit Header registers (3Ah-3Dh) are ignored if the hdlen field is not configured for a Transmit Header length that requires use of that Transmit Header register.

The hdlen[2:0] field is also used in RX mode to specify the expected number of Header bytes in the received packet.

4.1.4. TX Packet Length

The PH may be configured to automatically include the length of the Payload in the Header. This functionality is controlled by the fixpklen bit in SPI Register 33h. When this bit is set, a byte corresponding to the length of the Payload is **not** included in the Header. In this scenario, it is assumed that the length of the Payload is always fixed and known by the RX end of the link, and therefore there is no need to transmit a Header byte containing this information. If this bit is cleared, a byte containing the length of the Payload may be variable in length and thus it is necessary to send this information to the RX end of the link by including it in the Header.

Register 33h. Header Control 2

Bit	D7	D6	D5	D4	D3	D2	D1	D0
Name	skipsyn	hdlen[2:0]			fixpklen	synclen[1:0]		prealen[8]

Figure 28. SPI Register 33h, TX Packet Length in Header

When using the PH, the number of bytes in the TX Payload must ***always*** be specified by the pklen[7:0] field in SPI Register 3Eh. This is self-evident; the PH cannot construct the entire transmit packet structure unless the number of bytes in the Payload is known. The value contained in this field corresponds directly to the number of bytes in the TX Payload. As this field is 8 bits in length, the maximum length of Payload is 255 bytes. It is possible to set this field to pklen = 0x00 and obtain a Payload of length zero (no bytes in the Payload).

If the user desires a Payload exceeding 255 bytes in length, it is not possible to use the PH. In such a case, it will be necessary to disable the PH and to program the ***entire*** packet structure (including Preamble and Sync word) in the FIFO memory, or to construct the entire packet structure in the host MCU and to provide it to the RFIC in real-time over a physical input pin.

It should be noted that the packet length byte (if present) is ***not*** considered as part of the Transmit Header Word byte(s) discussed in “4.1.3. TX Header Field” . That is to say, the conditional packet length byte is separate from the Transmit Header Word byte(s); the user does not need to specifically program one of the Transmit Header Word bytes in SPI Registers 3Ah-3Dh with the packet length value. The location of the packet length byte (if present) is after the Transmit Header Word byte(s) and before the Payload, as shown in Figure 1.

4.1.5. CRC

The PH may be configured to generate a cyclic redundancy check (CRC) across various fields of the TX packet structure. This functionality is enabled by setting the encrc bit D2 in SPI Register 30h. When CRC is enabled, two bytes representing the CRC checksum are appended to the end of the transmit packet.

Register 30h. Data Access Control

Bit	D7	D6	D5	D4	D3	D2	D1	D0
Name	enpacrx	lsbfrst	crcdonly	skip2ph	enpactx	encrc	crc[1:0]	

Figure 29. SPI Register 30h, CRC Control

The crc[1:0] field in bits D1-D0 of SPI Register 30h allows selection of one of four different CRC polynomials:

- 2’b00 = CCITT
- 2’b01 = CRC-16 (IBM)
- 2’b10 = IEC-16
- 2’b11 = Biacheva

The value programmed into the crc[1:0] field has no effect unless the encrc bit D2 is also enabled. The PH may be configured to generate the CRC across ***only*** the Payload data field, or across the Header, Packet Length, and Payload fields. This functionality is controlled by the crcdonly bit D5 in SPI Register 30h. When this bit is set, the CRC is calculated across only the Payload data field. When this bit is cleared, the CRC is calculated across the Header, Packet Length, and Payload data fields. This is shown in Figure 30.

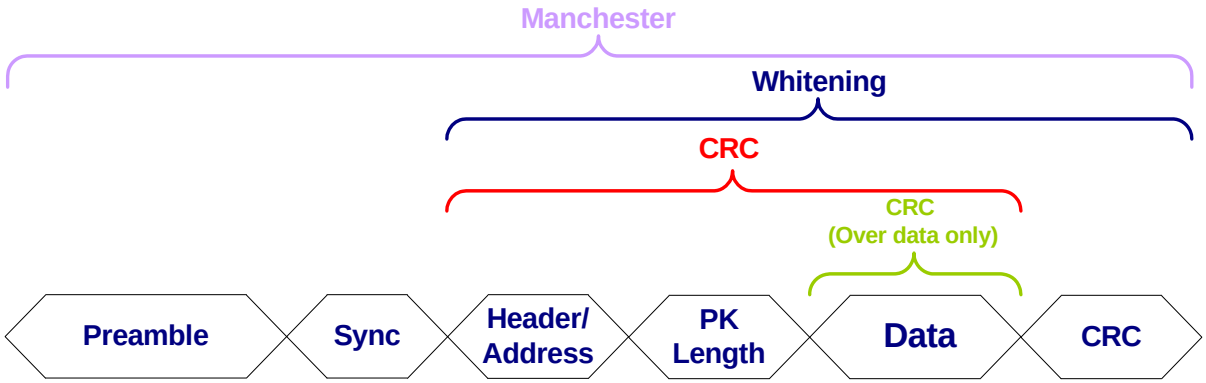


Figure 30. Operation of CRC/Whitening/Manchester Encoding Across Various Fields

It should be noted that the pklen[7:0] field (discussed in section “4.1.4. TX Packet Length”) refers to only the length of the Payload data field, and does ***not*** normally include the two CRC bytes (if present). (There is a way to reconfigure the PH to include the CRC bytes in the packet length byte; this is discussed in further detail in “AN545 EZRadioPRO and 802.15.4d Compliance”.)

4.1.6. Data Whitening

Data whitening can be used to avoid extended sequences of 0s or 1s in the transmitted data stream in order to achieve a more uniform spectrum. This functionality is enabled by setting the enwhite bit D0 in SPI Register 70h. When enabled, all bits **except** the Preamble and Sync Word are XORed with a pseudorandom sequence output from the built-in PN9 generator. The generator is hard-wired as a PN9 sequence, and is not adjustable by the user. The generator is initialized at the beginning of the payload. The receiver recovers the original data by repeating this operation. Data whitening does not change the effective bit rate. The enwhite bit has no effect unless the PH is also enabled by setting the enpactx bit D3 of SPI Register 30h (refer to Figure 22).

Register 70h. Modulation Mode Control 1

Bit	D7	D6	D5	D4	D3	D2	D1	D0
Name	Reserved		txdtrtscale	enphpwdn	manppol	enmaninv	enmanch	enwhite

Figure 31. SPI Register 70h, Data Whitening and Manchester Encoding

4.1.7. Manchester Encoding

Although data whitening provides an effective means of obtaining a uniform spectrum, it does not ensure a dc-free transmission; the whitened packet may have an unequal number of 1's and 0's. Manchester encoding can be used to ensure a dc-free transmission and to provide good synchronization properties. This functionality is enabled by setting the enmanch bit D1 in SPI Register 70h. As shown in Figure 30, Manchester encoding is applied across the entire packet; it is not possible to apply Manchester encoding across only a selected number of fields. Manchester encoding is available regardless of whether the PH is enabled or disabled (i.e., by setting or clearing the enpactx bit D3 in SPI Register 30h, as shown in Figure 22).

Manchester encoding works by replacing each data bit by two data bits. The two replacement data bits are of opposite polarity (i.e., either a 10 or 01 pattern), thus ensuring a balanced number of 1's and 0's and regular edge transitions. The effective data rate is unchanged but the actual data rate (over the air interface) is doubled. When using Manchester encoding, the effective data rate is limited to 128 kbps as the actual data rate of EZRadioPRO chips is limited to 256 kbps.

It is possible to invert the polarity of the Manchester encoding by setting the enmaninv bit D2 in SPI Register 70h. Normal Manchester encoding replaces each 1 bit in the transmit data stream with a 01 sequence, and each 0 bit with a 10 sequence. When the enmaninv bit is set, this coding is inverted: each 1 bit is replaced with a 10 sequence and each 0 bit is replaced with a 01 sequence. This is illustrated in the Manchester encoding example of Figure 32, where the original (un-encoded) packet bits are compared with the packet bits after Manchester encoding. The enmaninv bit has no effect unless the enmanch bit is also set.

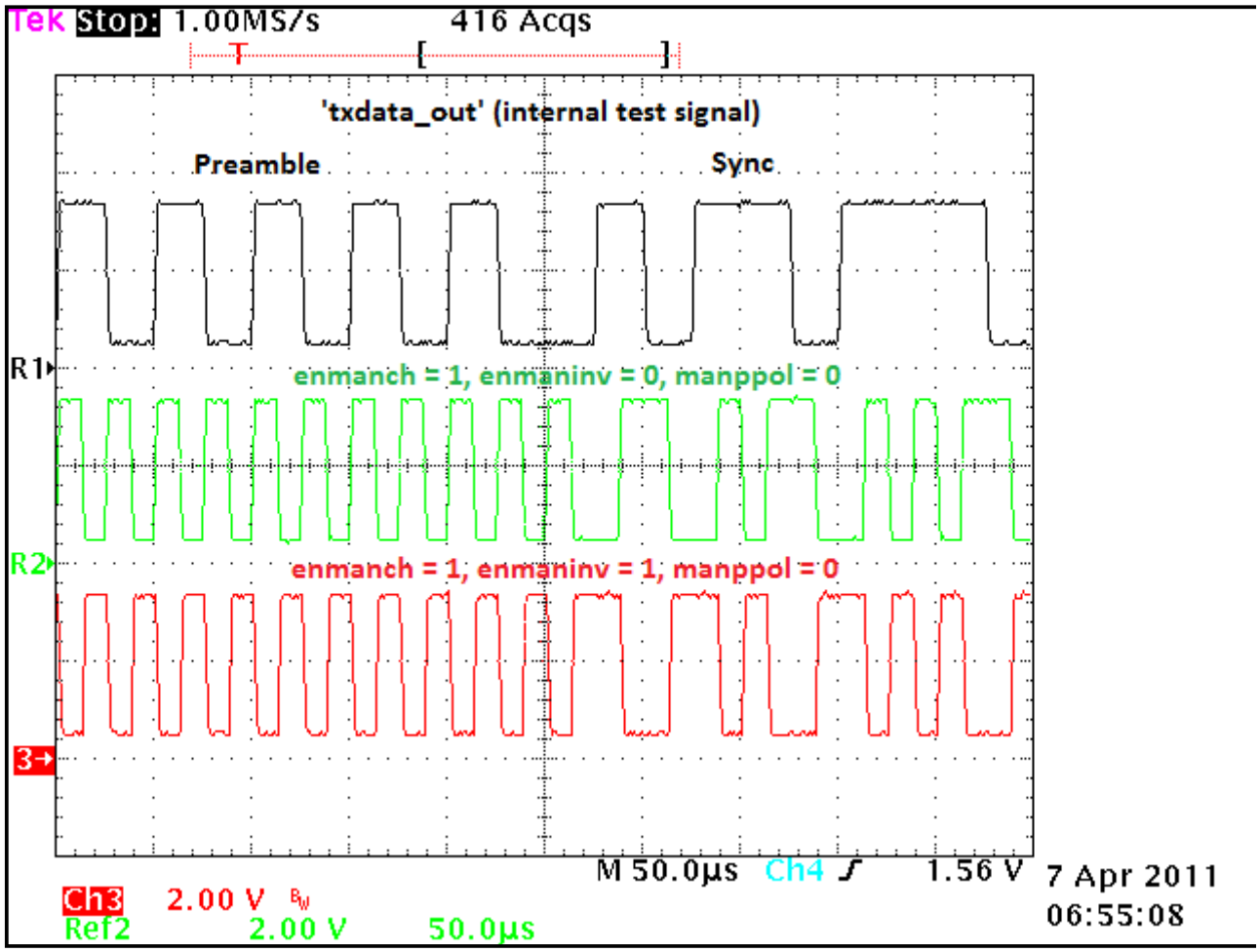


Figure 32. Manchester Coding Example (manppol=0)

Performing Manchester encoding across the Preamble field presents potential difficulties. As a normal Preamble already consists of a '1010…' pattern that contains regular bit transitions, there is little need for further encoding. Furthermore, if normal Manchester encoding is applied to a '1010…' pattern, the result would be a '10011001…' pattern at twice the data rate, and thus not discernible from the original pattern anyways. As a result, the Preamble field is handled in a special fashion when Manchester encoding is enabled. Each bit of the original un-encoded data (regardless of whether a 1 or a 0) is replaced by the same Manchester encoded bit sequence. Thus the preamble-like structure of alternating 1's and 0's is retained, but now occurs at twice the original data rate. The manppol bit (Manchester preamble polarity) may be used to control whether each un-encoded preamble bit is replaced by a '10' bit sequence or by a '01' bit sequence, as may be seen by comparing Figure 32 and Figure 33. The manppol bit has no effect unless the enmanch bit is also set.

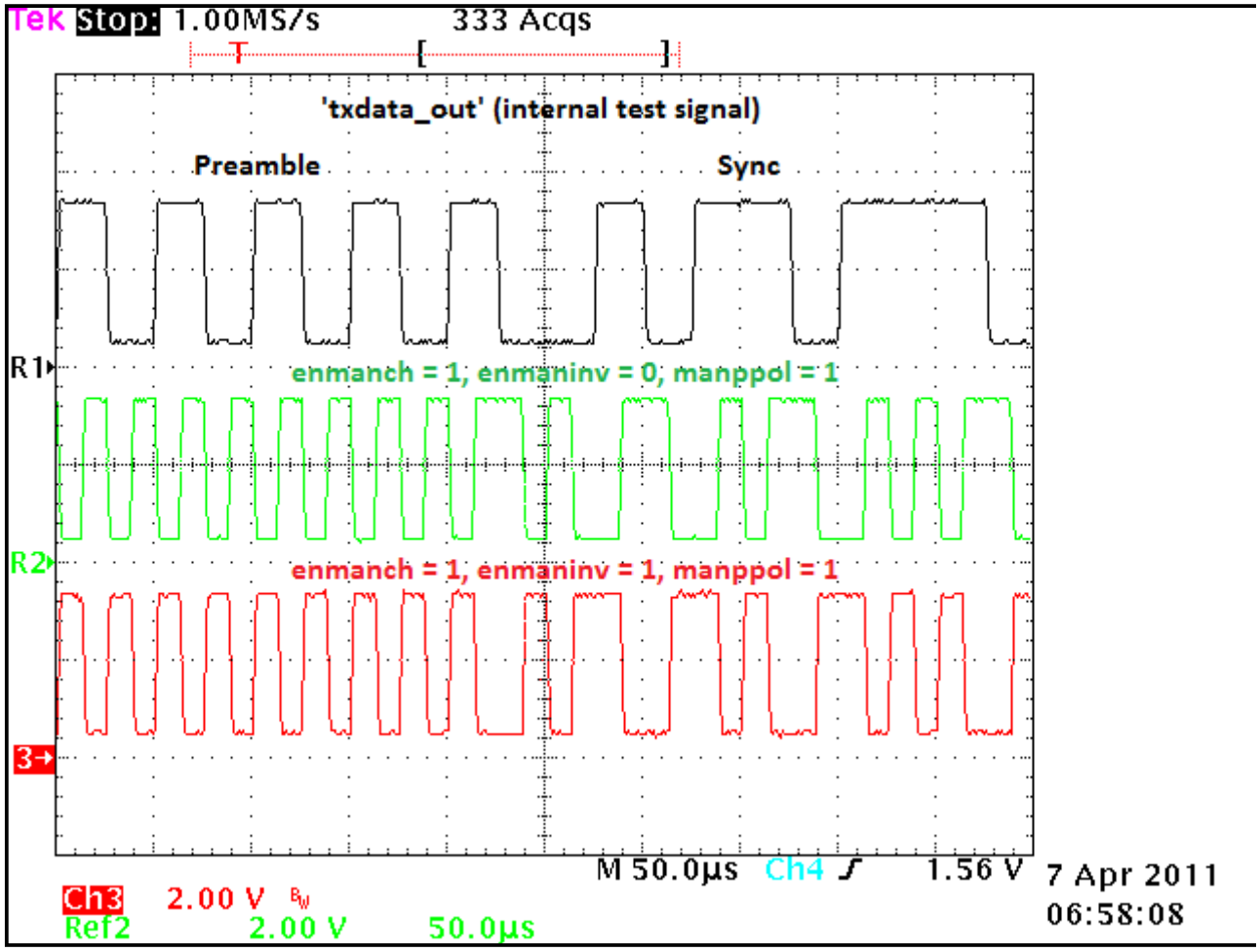


Figure 33. Manchester Coding Example (manppol=1)

4.2. Operation of Packet Handler in RX Mode

In RX mode, operation of the PH has meaning in both FIFO mode or Direct mode. However, this functionality changes depending upon which mode is selected. In RX mode, the PH is enabled by setting the enpacrx bit D7 of SPI Register 30h (refer to Figure 22).

4.2.1. Preamble Detection

The PH provides Preamble Detector functionality to qualify reception of a signal with a valid Preamble. This functionality remains even if the exnpacrx bit D7 of SPI Register 30h is cleared (refer to Figure 22), and is operational in both FIFO and Direct modes.

The Preamble Detector circuitry searches for a minimum number of consecutive “0101...” bits, as specified in the preath[4:0] field in SPI Register 35h. The value in this field corresponds to the number of consecutive nibbles (4 bits) that must be received correctly before a valid Preamble is considered to have been received.

Register 35h. Preamble Detection Control 1

Bit	D7	D6	D5	D4	D3	D2	D1	D0
Name	preath[4:0]					rssi_offset[2:0]		

Figure 34. SPI Register 35h, Preamble Detection Threshold

Upon correctly receiving the number of Preamble bits specified by the preath[4:0] field, the chip issues a PREAMBLE_VALID signal. This signal may be directly observed as an output on a GPIO pin by programming the GPIO Configuration registers accordingly. A typical PREAMBLE_VALID signal is shown in Figure 35. The PREAMBLE_VALID signal remains high until detection of the Sync word, or until the search for the Sync word times out and the chip returns to searching for another Preamble.

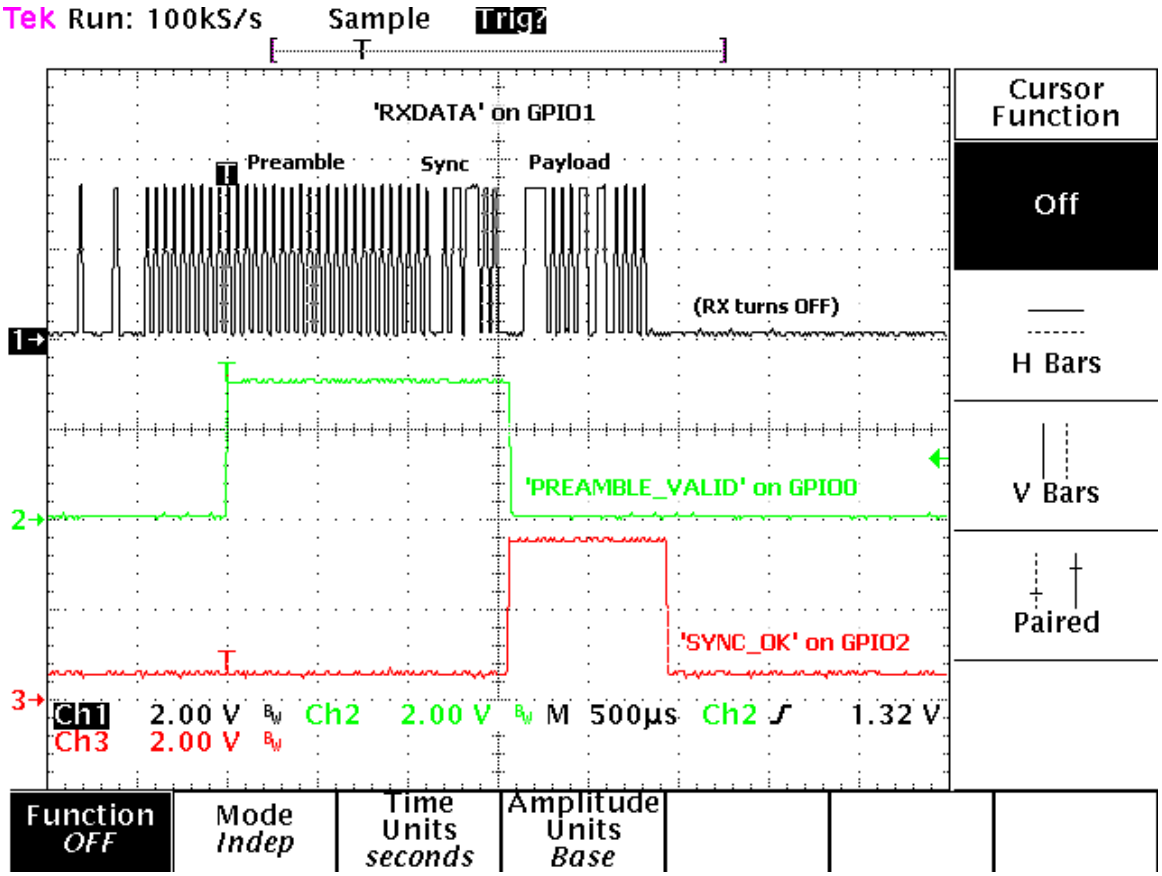


Figure 35. Typical PREAMBLE_VALID and SYNC_OK Signals

The chip may also be configured to generate an interrupt upon detection of the PREAMBLE_VALID signal. The status of this interrupt may be found as the ipreaval status bit in SPI Register 04h, bit D6. This interrupt status bit will cause an actual interrupt to occur on the nIRQ output pin if its corresponding enable bit (enpreaval bit D6 of SPI Register 06h) is also set.

Register 04h. Interrupt/Status 2

Bit	D7	D6	D5	D4	D3	D2	D1	D0
Name	iswdet	ipreaval	ipreainval	irssi	iwut	ilbd	ichiprdy	ipor

Register 06h. Interrupt Enable 2

Bit	D7	D6	D5	D4	D3	D2	D1	D0
Name	enswdet	enpreaval	enpreainval	enrssi	enwut	enlbd	enchiprdy	enpor

Figure 36. SPI Register 04h & 06h, Preamble Valid Interrupt

Detection of the Preamble is required for proper reception of the packet. Prior to issuing the PREAMBLE_VALID signal, the bit clock recovery (BCR) circuitry operates in fast tracking mode. This allows for fast acquisition of bit clock timing, but results in higher clock jitter. If the chip were to remain in this fast tracking mode during the reception of the remainder of the packet, bit errors would likely result due to this excess clock jitter. After detection of PREAMBLE_VALID, the chip switches to a slow tracking mode which greatly reduces the clock jitter and allows for error-free reception of bits.

4.2.2. Invalid Preamble Detection

The PH provides Invalid Preamble Detector functionality to determine when reception of a signal with a valid Preamble has ***not*** occurred within a specified period of time. This functionality remains even if the enpacrx bit D7 of SPI Register 30h is cleared (refer to Figure 22), and is operational in both FIFO and Direct modes. This functionality is useful when rapidly scanning channels for the detection of a valid signal, as in a frequency hopping system.

The Preamble Detector circuitry searches for a minimum number of consecutive '0101…' bits, as specified in the preath[4:0] field in SPI Register 35h. If a valid Preamble is not detected within the timeout period specified in the invalid_preamble_threshold[3:0] field in SPI Register 60h, the chip issues a PREAMBLE_INVALID signal. The value in this field corresponds to the length of the timeout period in nibbles (4 bits). After issuance of a PREAMBLE_INVALID signal, the chip will return to looking for Preamble again.

Register 60h. Channel Filter Coefficient Address

Bit	D7	D6	D5	D4	D3	D2	D1	D0
Name	invalid_preamble_threshold[3:0]				Reserved			

Figure 37. SPI Register 60h, Invalid Preamble Detection Threshold

The PREAMBLE_INVALID signal may be directly observed as an output on a GPIO pin by programming the GPIO Configuration registers accordingly. A typical PREAMBLE_INVALID signal is shown in Figure 38. In this example, the receiver is enabled in advance of arrival of the TX packet and thus the RXDATA signal initially corresponds to reception of random noise. The chip fails to detect a sufficient number of Preamble-like bits within the specified invalid_preamble_threshold period and issues a PREAMBLE_INVALID signal. The PREAMBLE_INVALID signal pulses briefly high upon each occurrence of timeout of the search period. Thus multiple issuances of the PREAMBLE_INVALID signal may be detected before the packet arrives and is successfully detected.

Note that the period between successive PREAMBLE_INVALID signals may not be exact multiples of the programmed timeout period, as the chip automatically extends the allowed search period upon partial detection of a Preamble pattern; this ensures that reception of a valid Preamble is not accidentally "cut off" by early issuance of a PREAMBLE_INVALID signal.

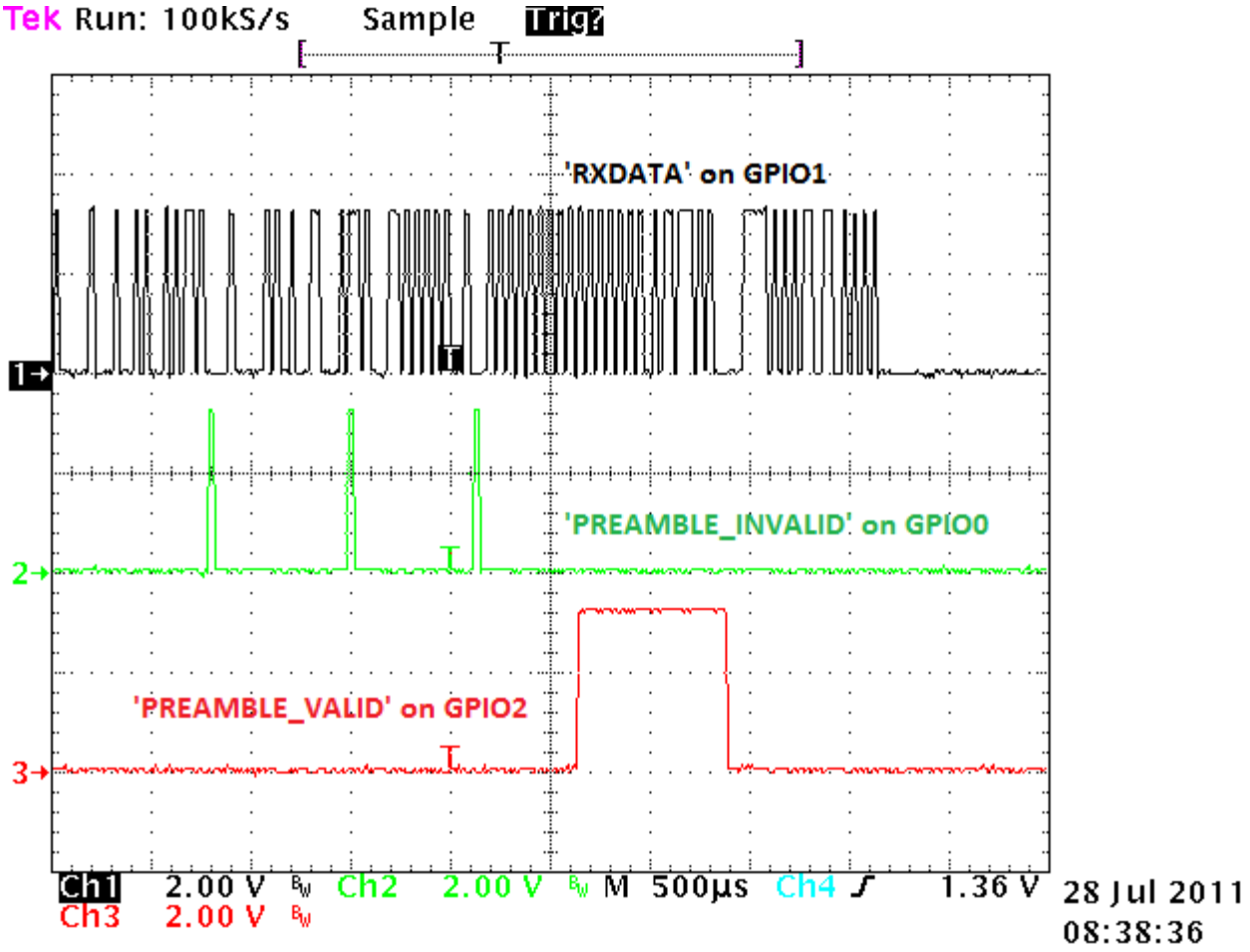


Figure 38. Typical PREAMBLE_INVALID and PREAMBLE_VALID Signals

The PREAMBLE_INVALID signal is also generated immediately upon failure to detect the Sync Word. In the example shown in Figure 39, the chip fails to detect the Sync Word correctly and returns to searching for Preamble. The INVALID_PREAMBLE signal is issued immediately and then continues to repeat as a valid Preamble pattern is not detected in the remainder of the packet.

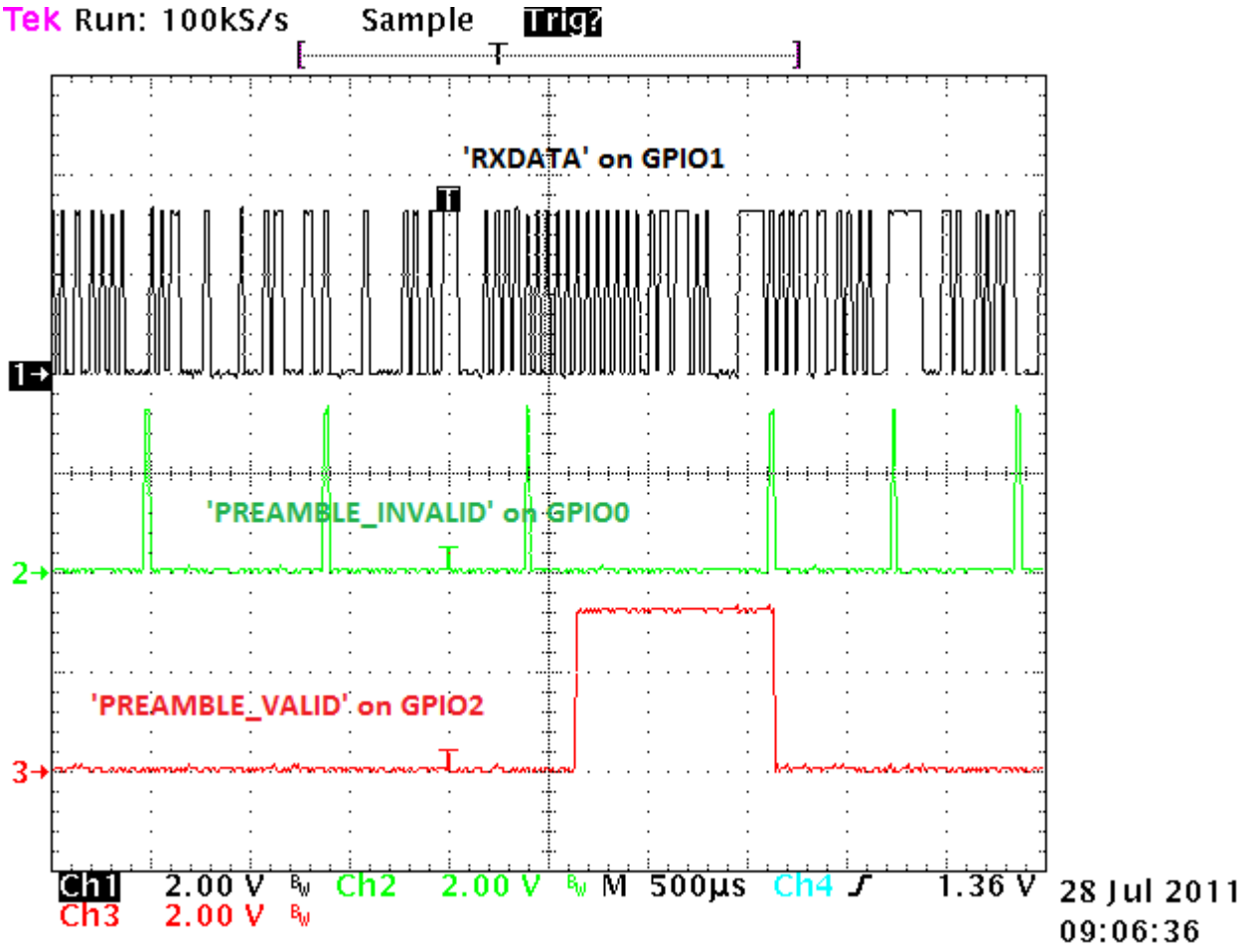


Figure 39. Typical PREAMBLE_INVALID upon Sync Detect Timeout

The chip may also be configured to generate an interrupt upon detection of the PREAMBLE_INVALID signal. The status of this interrupt may be found as the ipreainval status bit in SPI Register 04h, bit D5. This interrupt status bit will cause an actual interrupt to occur on the nIRQ output pin if its corresponding enable bit (enpreainval bit D5 of SPI Register 06h) is also set.

Register 04h. Interrupt/Status 2

Bit	D7	D6	D5	D4	D3	D2	D1	D0
Name	iswdet	ipreaval	ipreainval	irssi	iwut	ilbd	ichiprdy	ipor

Register 06h. Interrupt Enable 2

Bit	D7	D6	D5	D4	D3	D2	D1	D0
Name	enswdet	enpreaval	enpreainval	enrssi	enwut	enlbd	enchiprdy	enpor

Figure 40. SPI Register 04h & 06h, Preamble Invalid Interrupt

4.2.3. Sync Detection

The PH provides Sync Word Detector functionality to verify that the packet is intended for the receiver in use. This functionality remains even if the enpacrx bit D7 of SPI Register 30h is cleared (refer to Figure 22), and is operational in both FIFO and Direct modes.

The Sync Word Detector circuitry searches for the number of Sync Word bytes specified by the synclen[1:0] field in SPI Register 33h (refer to Figure 24). The value(s) of the Sync Word bytes are specified in SPI Registers 36h-39h (refer to Figure 25). The Sync word bytes are received in descending order; that is, the first byte received is compared against Sync Word 3, the second byte received is compared again Sync Word 2, and so on.

Because the length of Preamble may be much greater than the Preamble Detection Threshold, the PREAMBLE_VALID signal may be detected quite early during the Preamble. Thus there may be a significant number of remaining bits of Preamble before the Sync Word actually starts. The Sync Word detection process does not begin until a non-Preamble bit is received; that is, a bit that clearly does not fit in with the usual '0101…' pattern of Preamble bits is required to trigger the search for Sync Word. Thus it is strongly recommended that the user ***not*** specify a Sync Word that is similar to the Preamble (e.g., 0xAAh or 0x55h). The chip will remain searching until either the Sync Word is detected, or until the timeout period is exceeded. The timeout period is fixed at the length of the Sync Word plus 4 bits.

Upon correctly receiving the Sync Word byte(s), the chip issues a SYNC_OK signal. This signal may be directly observed as an output on a GPIO pin by programming the GPIO Configuration registers accordingly. A typical SYNC_OK signal is shown in Figure 35. The SYNC_OK signal remains high until the end of the packet. The PREAMBLE_VALID signal goes low upon detection of the SYNC_OK signal.

The chip may also be configured to generate an interrupt upon detection of the SYNC_OK signal. The status of this interrupt may be found as the iswdet status bit in SPI Register 04h, bit D7. This interrupt status bit will cause an actual interrupt to occur on the nIRQ output pin if its corresponding enable bit (enswdet bit D7 of SPI Register 06h) is also set.

Register 04h. Interrupt/Status 2

Bit	D7	D6	D5	D4	D3	D2	D1	D0
Name	iswdet	ipreaval	ipreainval	irssi	iwut	ilbd	ichiprdy	ipor

Register 06h. Interrupt Enable 2

Bit	D7	D6	D5	D4	D3	D2	D1	D0
Name	enswdet	enpreaval	enpreainval	enrssi	enwut	enlbd	enchiprdy	enpor

Figure 41. SPI Register 04h & 06h, Sync Word Detect Interrupt

Detection of the Sync Word is strongly recommended for proper reception of the packet in all modes, and is required when operating in RX FIFO mode. Detection of the Sync Word is the chip's only means of determining the start of the Payload data field, and thus no bytes can be stored into RX FIFO memory unless the boundary between these fields is correctly determined. Verification of the Sync Word is also the chip's primary method of determining whether the received packet is intended for it, or is coming from an undesired transmitter. Finally, failure to detect the Sync Word normally results in the chip switching back to fast tracking mode and resuming search for another Preamble. Thus in order for the chip to **remain** in slow tracking mode (with resulting low clock jitter), it is important for the chip to detect the Sync Word.

If desired, the user may program the chip to skip the timeout period when searching for the Sync Word. This is done by setting the skipsyn bit D7 in SPI Register 33h. Setting this bit does **not** result in completely skipping the search for Sync Word; proper detection of a Sync Word remains necessary for determining the start of the Payload data field (a requirement when operating in a RX FIFO mode). If the skipsyn bit is set, the chip will simply ignore the timeout period after failure to detect the Sync Word and will remain in slow tracking mode. The chip will not return to searching for Preamble, but will remain in "search for Sync Word" mode for the remainder of the packet.

As it is not possible to program the synclen[1:0] field for a Sync Word length of zero bytes, the only means of ignoring the Sync Word entirely is to set the skipsyn bit (to prevent a return to fast tracking mode with its corresponding increase in clock jitter) and to use RX Direct mode (in which the host MCU is responsible for determining the start of the Payload data field).

Register 33h. Header Control 2

Bit	D7	D6	D5	D4	D3	D2	D1	D0
Name	skipsyn	hdlen[2:0]			fixpklen	synclen[1:0]		prealen[8]

Figure 42. SPI Register 33h, Skip Sync Timeout Bit

4.2.4. RX Header Field

The PH expects to receive a RX Header field of length specified by the hdlen[2:0] field in SPI Register 33h (refer to Figure 26) and may range in value from 0 to 4 bytes. The contents of the actual received Header bytes are placed into the rxhd[31:0] field in SPI Registers 47h-4Ah. Note that these are read-only registers.

If the packet contains only one Header byte, its received value is placed into Received Header 3 SPI Register 47h. If the packet contains two Header bytes, their received values are placed into Received Header 3 and 2, SPI Registers 47h and 48h, and so on.

Register 47h. Received Header 3

Bit	D7	D6	D5	D4	D3	D2	D1	D0
Name	rxhd[31:24]							

Register 48h. Received Header 2

Bit	D7	D6	D5	D4	D3	D2	D1	D0
Name	rxhd[23:16]							

Register 49h. Received Header 1

Bit	D7	D6	D5	D4	D3	D2	D1	D0
Name	rxhd[15:8]							

Register 4Ah. Received Header 0

Bit	D7	D6	D5	D4	D3	D2	D1	D0
Name	rxhd[7:0]							

Figure 43. SPI Registers 47h-4Ah, Receive Header Word Value(s)

Once the Header bytes are received, their values may be checked against pre-programmed expected values, called check values. The number of header bytes to be checked by the PH may be set to less than the number of Header bytes received. The number of Header bytes to be checked is specified by the hdch[3:0] field in SPI Register 32h. For instance, there may be four bytes of Header in the packet structure but only one byte of the Header is set to be checked.

Register 32h. Header Control 1

Bit	D7	D6	D5	D4	D3	D2	D1	D0
Name	bcen[3:0]				hdch[3:0]			

Figure 44. SPI Register 32h, Header Check Word

This field uses one-hot encoding to specify the Header bytes to check. Thus a value of hdch[3:0] = 4'b0000 specifies no receive Header check, a value of hdch[3:0] = 4'b0001 specifies to check Header byte 0, a value of hdch[3:0] = 4'b0101 specifies to check Header bytes 2 and 0, etc.

For the Header bytes that are set to be checked, the expected received value of the Header bytes should be programmed in the chhd[31:0] field in SPI Registers 3Fh–42h. Values programmed into Receive Header value registers (3Fh-42h) are ignored if the hdch[3:0] field is not configured to check the value of that Header byte.

Register 3Fh. Check Header 3

Bit	D7	D6	D5	D4	D3	D2	D1	D0
Name	chhd[31:24]							

Register 40h. Check Header 2

Bit	D7	D6	D5	D4	D3	D2	D1	D0
Name	chhd[23:16]							

Register 41h. Check Header 1

Bit	D7	D6	D5	D4	D3	D2	D1	D0
Name	chhd[15:8]							

Register 42h. Check Header 0

Bit	D7	D6	D5	D4	D3	D2	D1	D0
Name	chhd[7:0]							

Figure 45. SPI Registers 3Fh-42h, Receive Header Word Check Value(s)

It is further possible to enable or disable the checking of individual bits within the selected Header bytes. This is accomplished with the Header Enable field hden[31:0] in SPI Registers 43h–46h. For example, if you want to check all bits in Header byte 3 then set hden[31:24] = 0xFF, but if only the last 4 bits are desired to be checked then set hden[31:24] = 0x0F.

Register 43h. Header Enable 3

Bit	D7	D6	D5	D4	D3	D2	D1	D0
Name	hden[31:24]							

Register 44h. Header Enable 2

Bit	D7	D6	D5	D4	D3	D2	D1	D0
Name	hden[23:16]							

Register 45h. Header Enable 1

Bit	D7	D6	D5	D4	D3	D2	D1	D0
Name	hden[15:8]							

Register 46h. Header Enable 0

Bit	D7	D6	D5	D4	D3	D2	D1	D0
Name	hden[7:0]							

Figure 46. SPI Registers 43h-46h, Header Enable Words

Header checking is useful for qualifying a packet that is intended for only one receive node in a multiple-node system. However, there are other times when a transmitted packet is intended to be broadcast to ***all*** receive nodes in the system. In such a scenario, it is desirable for the receive node to respond to either its unique header address, or to the general broadcast header address.

The PH supports such a Broadcast Header feature. The number of Header bytes to be checked for broadcast address is specified by the bcen[3:0] field in SPI Register 32h. This field also uses one-hot encoding, in a fashion similar to that used by the hdch[3:0] field. The broadcast address is hard-wired as 0xFF; thus the PH will consider the header check to be valid if the value of a particular received Header byte matches 0xFF ***or*** the programmed check value. Note that it is necessary to enable header checking of the desired byte in hdch[3:0] before additionally enabling broadcast address checking in bcen[3:0]; it is not possible to perform broadcast checking of a particular Header byte without also performing header checking.

Register 32h. Header Control 1

Bit	D7	D6	D5	D4	D3	D2	D1	D0
Name	bcen[3:0]				hdch[3:0]			

Figure 47. SPI Register 32h, Broadcast Address Check Enable

The logic equivalent of the checking algorithm described here is shown in Figure 48 (using Header byte 3 as an example). A similiar logic check is performed for Header byte 2, Header byte 1, and Header Byte 0 if enabled.

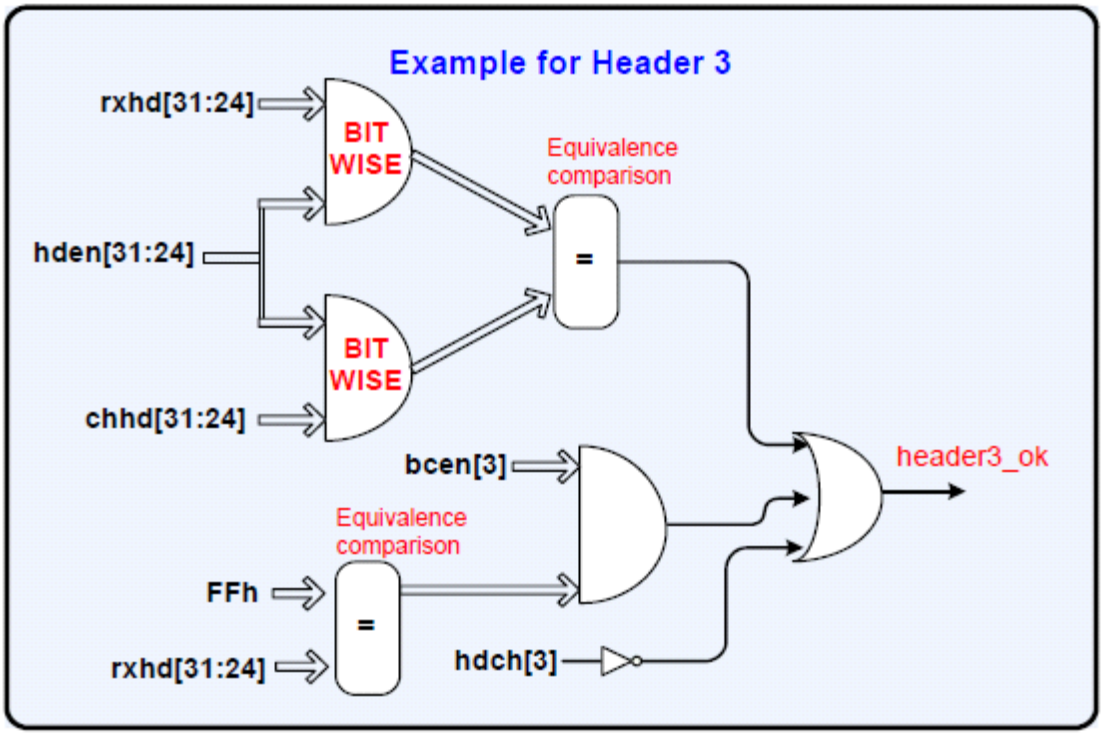


Figure 48. Header Checking Logic

If the Header check fails, the chip does not attempt to process the remainder of the packet but instead remains in RX mode and returns to searching for Preamble.

4.2.5. RX Packet Length

In order for the receive node to determine if a packet has been received correctly, it is necessary for it to know the number of bytes in the transmit packet. There are two different ways in which the packet length information can be provided to the receive node: it can be pre-programmed into a register on the RX side, or it can be transmitted (as an extra packet length byte appended to the Header) and recovered by the receive node. Either method of determination of RX packet length depends upon the RX PH being enabled by setting the enpacrx bit D7 of SPI Register 30h; it is not possible for the receiver to receive a packet of determined length without enabling the PH.

If the length of the packet is fixed, then it is assumed that both sides of the link know this information and there is no need to include the packet length byte in the packet structure. In this scenario, the fixpklen bit D3 of SPI Register 33h is set (refer to Figure 28) and the expected RX packet length should be programmed in the pklen[7:0] field in SPI Register 3Eh (refer to Figure 6). Note that when a fixed packet length is used, SPI Register 3Eh is used to specify the packet length on both sides of the link.

If the packet length is variable, the fixpklen bit must be cleared and the transmitter must include the packet length information within the transmit packet structure. As shown in Figure 1, the Packet Length byte is located directly after the Header bytes (if any). On the RX side of the link, the received value of this packet length byte is stored into the rxplen[7:0] field in SPI Register 4Bh. The host MCU may retrieve this information to determine how many bytes to read from the RX FIFO. Note that this is a read-only byte; specifically, this register is ***not*** used to program the expected receive packet length when a fixed packet length is used.

Register 4Bh. Received Packet Length

Bit	D7	D6	D5	D4	D3	D2	D1	D0
Name	rxplen[7:0]							

Figure 49. SPI Register 4Bh, Received Packet Length Word

4.2.6. CRC

The PH in the RX node may be configured to generate a cyclic redundancy check (CRC) across various fields of the received packet structure. This functionality is enabled by setting the encrc bit D2 in SPI Register 30h (refer to Figure 29). When CRC is enabled, it is assumed that the last two bytes of the received packet represent the CRC checksum (refer to Figure 1). The received checksum is compared against the calculated checksum; a CRC error is generated if the two checksums do not match.

The crc[1:0] field in bits D1-D0 of SPI Register 30h allows selection of one of four different CRC polynomials:

- 2'b00 = CCITT
- 2'b01 = CRC-16 (IBM)
- 2'b10 = IEC-16
- 2'b11 = Biacheva

The value programmed into the crc[1:0] field has no effect unless the encrc bit D2 is also enabled. The PH may be configured to calculate the CRC across ***only*** the Payload data field, or across the Header, Packet Length, and Payload fields (refer to Figure 30). This functionality is controlled by the crcdonly bit D5 in SPI Register 30h. When this bit is set, the CRC is calculated across only the Payload data field. When this bit is cleared, the CRC is calculated across the Header, Packet Length, and Payload data fields.

The chip may be configured to generate an interrupt when a CRC error is encountered. The status of this interrupt may be found as the icrcerror status bit D0 in SPI Register 03h. This interrupt status bit will cause an actual interrupt to occur on the nIRQ output pin if its corresponding enable bit (encrcerror bit D0 of SPI Register 05h) is also set.

Register 03h. Interrupt/Status 1

Bit	D7	D6	D5	D4	D3	D2	D1	D0
Name	ifferr	itxffafull	ixtffaem	irxffafull	iext	ipksent	ipkvalid	icrerror

Register 05h. Interrupt Enable 1

Bit	D7	D6	D5	D4	D3	D2	D1	D0
Name	enfferr	entxffafull	entxffaem	enrxffafull	enext	enpksent	enpkvalid	encrcerror

Figure 50. SPI Register 03h & 05h, CRC Error Interrupt

If the PH receives all of the expected bytes and calculates a CRC checksum that matches the transmitted CRC checksum value, the packet is considered to be a valid packet. The chip may be configured to generate an interrupt when a valid packet is received. The status of this interrupt may be found as the ipkvalid status bit D1 in SPI Register 03h. This interrupt status bit will cause an actual interrupt to occur on the nIRQ output pin if its corresponding enable bit (enpkvalid bit D1 of SPI Register 05h) is also set. This interrupt is available only if the PH is enabled; when operating with the PH disabled, the chip has no knowledge of the length of the packet and thus cannot automatically generate this signal.

Note that even if the PH is configured to calculate the CRC checksum across only the Payload data field, it is still

necessary for the chip to correctly receive the Header and Packet Length fields (if enabled). If the Header check fails, the chip will not attempt to process the remainder of the packet but instead remains in RX mode and returns to searching for Preamble.

It should be noted that the pklen[7:0] field (discussed in “4.2.5. RX Packet Length”) refers to only the length of the Payload data field, and does ***not*** normally include the two CRC bytes (if present). (There is a way to reconfigure the PH to include the CRC bytes in the packet length byte; this is discussed in further detail in “AN545 EZRadioPRO and 802.15.4d Compliance”.)

4.2.7. Data Whitening

If data whitening is enabled on the TX side of the link (as discussed in “4.1.6. Data Whitening”), it must obviously also be enabled on the RX side of the link as well. This functionality is enabled by setting the enwhite bit D0 in SPI Register 70h. When enabled, all received bits ***except*** the Preamble and Sync Word are XORed with a pseudorandom sequence output from the built-in PN9 generator. This essentially “de-whitens” the received data packet, and occurs before the remainder of the PH functionality processing occurs (e.g., Header check, CRC checksum, etc.). The generator is hard-wired as a PN9 sequence, and is not adjustable by the user. The generator is initialized at the beginning of the payload. Data de-whitening does not change the effective bit rate. The enwhite bit has no effect unless the PH is also enabled by setting the enpacrx bit D7 of SPI Register 30h (refer to Figure 22).

4.2.8. Manchester Decoding

If the TX side of the link uses Manchester encoding, it is obvious that the RX side of the link must also use Manchester decoding in order to recover the original data stream. There are two places that Manchester decoding may take place: in the PH of the RFIC, or in the host MCU.

If it is desired to perform Manchester decoding in the host MCU, the RX node must be configured to disable Manchester decoding by clearing the enmanch bit D1 in SPI Register 70h (refer to Figure 31). The RX modem should be configured as if it were receiving a packet at the Manchester-encoded data rate (i.e., twice the original un-encoded data rate). The receive node must correctly receive the entire packet at the 2X effective data rate, and provide all bits to the host MCU for decoding and further processing.

If it is desired to perform Manchester decoding in the RFIC itself, the enmanch bit D1 in SPI Register 70h should be set. Manchester decoding is performed prior to providing the resulting bits to the PH; thus the PH processing operates on the original un-encoded data stream. The data bytes stored into the RX FIFO are the decoded bytes at the original un-encoded data rate.

It should be apparent that the proper configuration of the RX modem is dependent upon whether or not Manchester decoding is enabled. If enabled, the RX modem expects to receive a data stream at twice the un-encoded data rate, and must be configured accordingly (i.e., bit clock oversampling rate, IF channel select filter bandwidth, etc.) Manchester decoding may be performed without enabling the PH (i.e., without setting the enpacrx bit D7 in SPI Register 30h, as shown in Figure 52).

4.2.9. RX Multi-Packet Mode

In normal operation with the PH enabled, the receiver exits RX mode upon reception of a valid packet and returns to an IDLE state (e.g., TUNE mode or READY mode). In this scenario, the RX FIFO contains one packet of information, and it assumed that the host MCU will read this data from the RX FIFO before the chip re-enters RX mode. However, it is possible to configure the chip so that it remains in RX mode after reception of the first valid packet, for the purpose of receiving (and storing to RX FIFO memory) additional valid packets. This is known as RX multi-packet mode, and is enabled by setting the rxmpk bit D4 in SPI Register 08h, RX Multi-Packet Mode.

Register 08h. Operating Mode and Function Control 2

Bit	D7	D6	D5	D4	D3	D2	D1	D0
Name	antdiv[2:0]			rxmpk	autotx	enldm	ffclrrx	ffclrtx

Figure 51. SPI Register 08h, RX Multi-Packet Mode

In normal operation (without the rxmpk bit set), enabling the PH results in storage of only the Payload data bytes in RX FIFO memory, as other fields such as the Header byte(s), Packet Length byte, and CRC bytes are stripped off by the PH. However, in RX Multi-Packet mode it may be necessary to store either the Header bytes or Packet Length byte (or both) into RX FIFO memory, as this information may change from one packet to another; the host MCU cannot assume that the Header and/or Packet Length information received for the first valid packet remains applicable for the remainder of the packets. The storage of various fields into RX FIFO memory is illustrated in Figure 52.

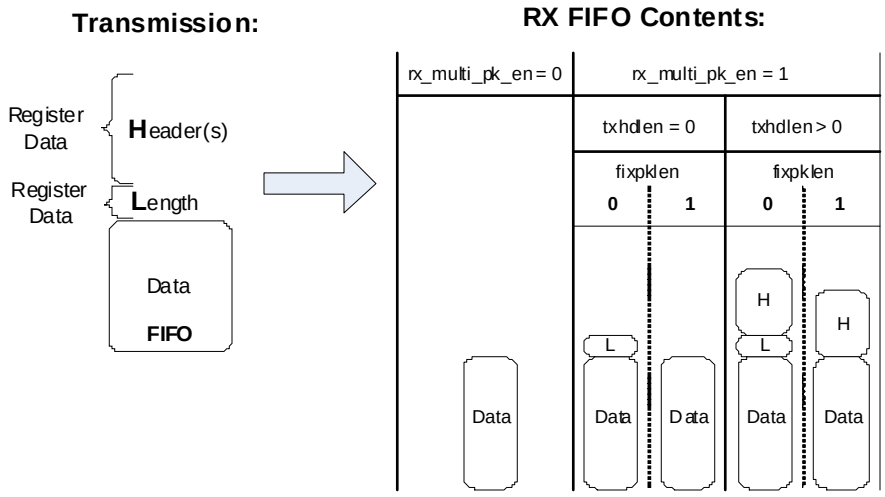


Figure 52. Multiple Packets in RX Packet Handler

The PH continues to valideate the reception of each packet as it arrives, by performing the Header check and CRC check. Invalid packets are not stored into RX FIFO memory. This is illustrated in Figure 49.

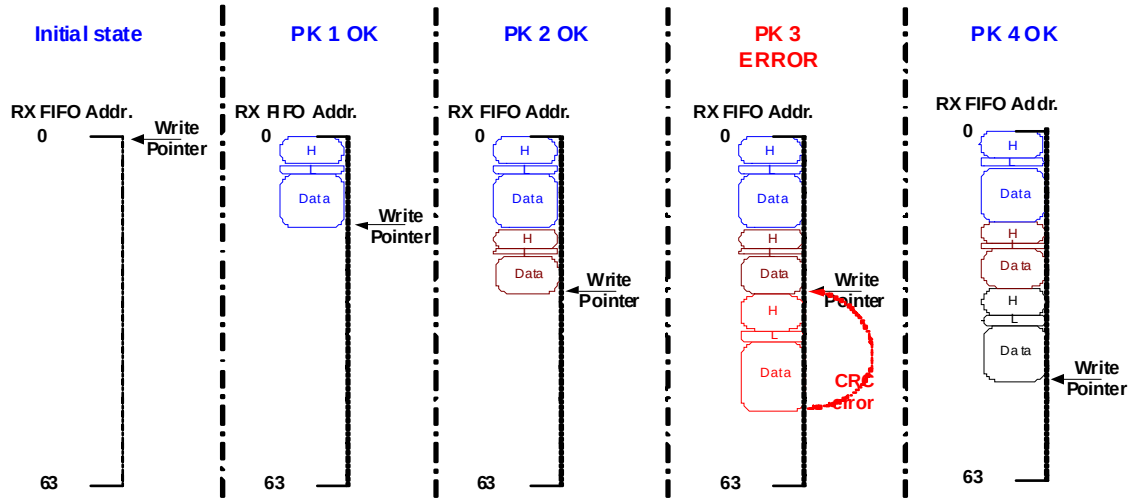


Figure 53. Multiple Packets in RX with CRC or Header Error

In RX Multi-Packet mode, it is necessary for the host MCU to command the RFIC to exit the RX state.

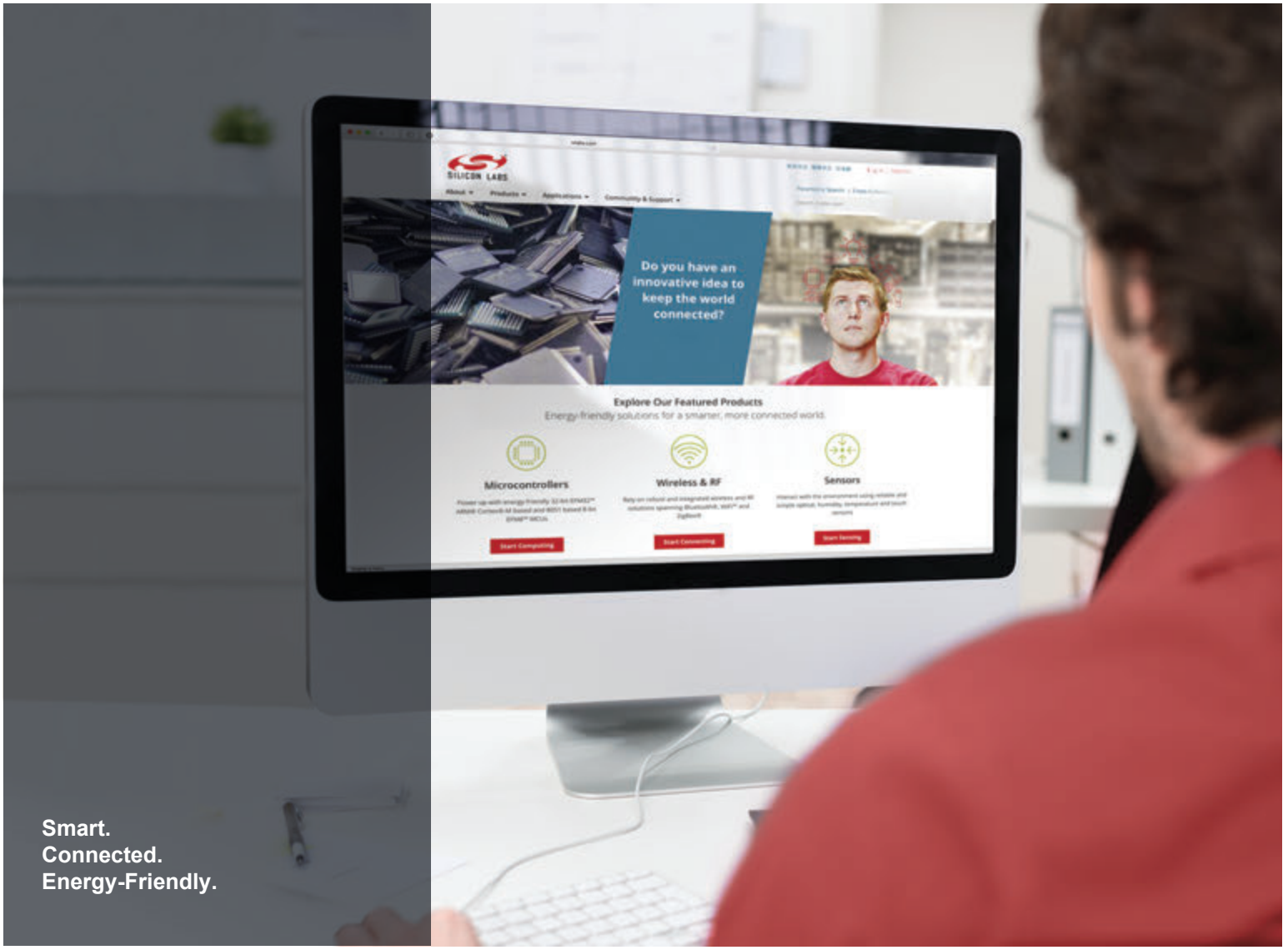
Document Change List

Revision 0.1 to 0.2

- Added description of Invalid Preamble Detection function.
- Corrected description of Skip Sync Timeout function.

Revision 0.2 to 0.3

- Revised the description of the TX and RF FIFO thresholds to clarify the exact byte value(s) at which the interrupts are triggered.



Products
www.silabs.com/products

Quality
www.silabs.com/quality

Support and Community
community.silabs.com

SILICON LABS®

Silicon Laboratories Inc.
400 West Cesar Chavez
Austin, TX 78701
USA

http://www.silabs.com