CC1101 stops receiving after some hours

Resolved

Daniel Berenguer

I have a CC1101-based design that sniffs network traffic coming from some low-power motes. My sniffer device needs to stay in Rx mode forever except for sending some wireless commands. The following are the read/write functions:

```
/**
 * sendData
 *
 * Send data packet via RF
 *
 * 'packet'    Packet to be transmitted
 *
 *  Return:
 *    True if the transmission succeeds
 *    False otherwise
 */
boolean CC1101::sendData(CCPACKET packet)
{
 byte marcState;

 // Enter RX state
 setRxState();

 // Check that the RX state has been entered
 while ((readStatusReg(CC1101_MARCSTATE) & 0x1F) != 0x0D)
  delay(1);
 delayMicroseconds(500);

 // Set data length at the first position of the TX FIFO
 writeReg(CC1101_TXFIFO,  packet.length);
 // Write data into the TX FIFO
 writeBurstReg(CC1101_TXFIFO, packet.data, packet.length);

 // CCA enabled: will enter TX state only if the channel is clear
 //cmdStrobe(CC1101_STX);
 setTxState();

 // Check that TX state is being entered (state = RXTX_SETTLING)
```

```
  marcState = readStatusReg(CC1101_MARCSTATE) & 0x1F;
  if((marcState != 0x13) && (marcState != 0x14) && (marcState != 0x15))
  {
   setIdleState();      // Enter IDLE state
   flushTxFifo();       // Flush Tx FIFO
   setRxState();        // Back to RX state
   return false;
  }

  // Wait for the sync word to be transmitted
  wait_GDO0_high();

  // Wait until the end of the packet transmission
  wait_GDO0_low();

  // Enter back into RX state
  setRxState();

  // Check that the TX FIFO is empty
  if((readStatusReg(CC1101_TXBYTES) & 0x7F) == 0)
   return true;

  return false;
}

/**
 * receiveData
 *
 * Read data packet from RX FIFO
 *
 * 'packet'    Container for the packet received
 *
 * Return:
 *     Amount of bytes received
 */
byte CC1101::receiveData(CCPACKET * packet)
{
 byte val;

 // Rx FIFO overflow?
 if ((readStatusReg(CC1101_MARCSTATE) & 0x1F) == 0x11)
 {
  setIdleState();      // Enter IDLE state
  flushRxFifo();       // Flush Rx FIFO
  packet->length = 0;
 }
 // Any byte waiting to be read?
 else if (readStatusReg(CC1101_RXBYTES) & 0x7F)
 {
  // Read data length
  packet->length = readConfigReg(CC1101_RXFIFO);
  // If packet is too long
  if (packet->length > CC1101_DATA_LEN)
   packet->length = 0;   // Discard packet
  else
  {
   // Read data packet
   readBurstReg(packet->data, CC1101_RXFIFO, packet->length);
```

```
    // Read RSSI
    packet->rssi = readConfigReg(CC1101_RXFIFO);
    // Read LQI and CRC_OK
    val = readConfigReg(CC1101_RXFIFO);
    packet->lqi = val & 0x7F;
    packet->crc_ok = bitRead(val, 7);
   }
  }
  else
   packet->length = 0;

  // Back to RX state
  setRxState();

  return packet->length;
}
```

Everything starts working as expected. My device receives wireless packets and transmits commands without problems. However, after some hours without transmitting, my device stops receiving packets. I've checked multiple times that, when the problem occurs, this device is still in reception mode (MARCSTATE = 0x0D) and RXBYTES > 0 (and not overflowed) so I don't really know what could be happening. I've noted however that transmitting a packet (above senData function) takes the CC1101 IC out from the locked state, so it becomes immediately able to receive packets again.

As a side note, before the locking problem, my device is frequently entering the "Rx FIFO overflow" state, maybe due to local wireless traffic coming from some unknown devices. Nevertheless, "receiveData" always flushes the FIFO when the overflow occurs. Mizanur Chowdhury from TI has suggested me a couple of things (thanks Mizanur!):

1. Check for a possible calibration-related problem. I need to say that MCSM0.FS_AUTOCAL = 1 for this device so calibration should be performed automatically when going from IDLE to RX or TX.

2. Check for the RX_FIFO Overflow issue described in errata note:http://www.ti.com/lit/er/swrz020b/swrz020b.pdf

Well, I've checked both the above possibilites but running periodic calibrations (SCAL strobe) and flushing the RXFIFO after each reception do not solve the problem. Only sending a packet unlocks the CC1101 from its uncertain passive state so if I find myself unable to solve the problem I'm thinking in sending a heartbeat packet periodically as a way to prevent the CC1101 IC from get blocked on the reception side forever. But before doing this workaround I'd like to get some feedback from this community.

Thank you very much for your ideas!

 Daniel Berenguer

---

 Daniel Berenguer

I've got some additional informations that, instead of clarifying things, are making them still more confusing. Well, I'm now printing RXBYTES at the moment of unlocking the CC1101 by transmitting a packet. I was expecting RXBYTES to be 0 since the CC1101 is not interrupting the uC. However, RXBYTES=41 every time. This would mean that the CC1101 would be receiving a packet, then it wouldn't interrupt the uC via GDO0 and finally it would quite the Rx state.

I'm very confused since MARCSTATE=Rx at the moment of unlocking the situation and, if I try to read RXFIFO the MISO line never goes low after selecting the IC. It's really showing a locked state. The only way to unlock it is by transmitting a packet or applying SRES.

TA12012

In reply to Daniel Berenguer:

Daniel,

It almost sounds like you are having some kind of digital communications issue between the MCU the RF device. Would it be possible for you to get some oscilloscope measurements done on your SPI port during some of the transactions?

Regards,
/TA

Please click the Verify Answer button on this post if it answers your question.

Siri

In reply to Daniel Berenguer:

Hi
It is a bit difficult to know exactly what is going on since I do not have your register settings, but I assume that you use variable packet length, CRC enabled, append status enabled, no length filtering, no FEC, and that you use CCA.

First I want to point out a couple of problems with your RX function. I assume that this is a function that is called every time you have an interrupt on GPIOn, where IOCFGn = 0x06.

Assume that you have received a packet where (packet->length > CC1101_DATA_LEN). In this case you do not flush the FIFO or read out the rest of the bytes in the FIFO. That means that there are bytes left in the FIFO the next time you enter RX (unless you do a strobe in the setRxState() function).

What you describe when reading RXBYTES = 0x51 and the radio is still in RX is the bug described in the errata note (see the case where APPEND_STATUS = 1, CRC = 1 and FEC_EN = 0). MARCSTATE will show RX but you will not receive anything, you get no interrupt and there are 0x41 bytes in the FIFO.

The way to get around this but is to use packet filtering in RX and set PKTLEN = 61 and always empty the FIFO after an interrupt has occurred (even if the length is not what you are looking for).

Another comment I have to your RX function is that if your radio is in overflow when this function is called you simply need to strobe SFRX (you do not have to strobe IDLE first).

BR

Siri

Daniel Berenguer

In reply to Siri:

Thanks Siri for your response. You've correctly guessed my settings :-)

I'm now following your suggestions and I hope to get some results soon. Meanwhile, what did you mean with "Rx filtering"? Receiver Channel Filter Bandwidth? BTW, RF studio did set MDMCFG4 for me: CHANBW_E = 2, CHANBW_M = 0

Thanks again!

Siri

In reply to Daniel Berenguer:

Hi Daniel

I was referring to packet length filtering in RC. By setting PKTLEN = 61 the radio will never accept a length byte greater than 61. If the length byte is 61, 64 bytes will be put in the RXFIFO; The length byte, 61 data bytes, and 2 status bytes. This way you will never get an RXFIFO overflow or enter the error state explained in the errata. You must however make sure that the FIFO always is empty when entering RX state. This can be done by always read all the bytes present in the FIFO when the interrupt occurs or by always flush the FIFO before entering RX state. Make sure that the FIFO is only flushed while in IDLE state or while in RXFIFO_OVERFLOW state.

When using filtering there is another thing you must be aware of. If an un-valid length byte is received (length byte > 61) the packet is discarded (not put in the RX FIFO), but since GDOn has already been asserted, the signal will de-assert when this happens but the radio will remain in RX. This means that you can get interrupt when the radio is in RX and the FIFO is empty. The first thing you should do in you interrupt is therefore to read RXBYTES to see if this is zero. If it is, you know that you entered the interrupt due to filtering, and you can simply return from the function (since the radio is still in RX).

I am really hoping that you are able to solve your problems. Good luck with your debugging :-)

BR

Siri

Daniel Berenguer

In reply to Siri:

Doing PKTLEN = 61 seems to have solved the problem. My board has been running for more than four days without a single problem!

Thanks Siri for the help!!

This thread has been locked.
If you have a related question, please click the "Ask a related question" button in the top right corner. The newly created question will be automatically linked to this question.