CB-OBS4XX OPTIMIZATION GUIDE

Document Revision
Document number: 6553651
Release: Feb 17, 2015 17:33
Document version: 3

1 Table of Content

2 Introduction

This document describes what optimization that can be done for the cB-OBS4xx Serial Port Adapters.

2.1 Related Documents

- The **Bluetooth Serial Port Adapter AT Commands** document contains a short introduction to the concepts of the Serial Port Adapter as well as a description of the AT commands supported.
- The **Bluetooth Serial Port Adapter Electrical Mechanical Datasheet** contains important information about the OEM Serial Port Adapter. Read this document if you are using the OEM Serial Port Adapter.

2.2 Overview

The default settings of the Serial Port Adapter is normally adequate for most type of applications. However, there may be special requirements regarding throughput, range, latency, connection/discovery time, power consumption or WLAN co-existence that requires special configuration.

Robustness and throughput of the link is affected by the packet type and length. A DM packet is the only packet type which contains forward error correction which means it will be more robust regarding disturbances and long range. Especially the DM1 packet is good since 10 out of 28 bytes consists of the forward error correction and therefore a large number of errors can be recovered without retransmission of the packet. Of course this will affect the maximum throughput which will be lower or much lower compared with DH or EDR packets.

The highest throughput for a good link is achieved with the EDR packets. Since there is no forward error correction (compare DM) and the SNR is poor (compared to both the DM and DH packets), the performance for a longer range decreases faster than for the DM and DH packets. Hence, for a maximum throughput the range must be short and the link must be good.

The only packet control performed by the Bluetooth radio itself is the following:

- *2-EDR (2 MBits/s modulation) and 3-EDR (3 Mbits/s modulation)*: Depending on the link quality the radio will use 2-EDR or 3-EDR packets. The throughput is higher with 3-EDR packets but the link is much more sensitive to disturbances and range. Note that it will not use DM or DH packets for a poor link if EDR is allowed.
- *Forward Error Correction (FEC) On/Off*: The radio will switch between DM and DH packets depending on the quality of the link.
- *1-, 3-, or 5-slot Packet Size*: The radio will match packet size to amount of data to transmit. For exampel, for 3-EDR packets it will use 3-DH1 packets for small amount of data (< 75 bytes) and 3-DH5 packets for large amount of data (> 543 bytes). The link quality does not affect packet size (1-, 3-, 5-slot).

Since there is only limited packet control depending on link quality, connectBlue has introduced a very simple RSSI based packet control algorithm which forces the radio to use 1-slot packets if RSSI is low. This will improve the robustness of the link and a longer range can be achieved. Note that it does not necessarily mean that it will change from EDR to DM/DH.

By default the master polling time of the slave is 40 slots (25 ms). This means that if a data packet is sent there is a random delay of 0-25 ms before the packet is sent over air. If a short latency is required this may not be good enough. It is possible to decrease the polling time down to a minimum of two slots using either the Quality of Service (QoS) or Sniff functionality. The cost is a higher power consumption since both radios must be more active.

If the module is optimized for one thing there is most times a cost for something else. If the module is configured for maximum throughput (big EDR packets), the link will be more sensitive to disturbances and long range. If the module is configured for a short latency, the power consumption increases.

3 Throughput

The data transmission throughput between two Bluetooth modules depends very much on the following factors.

- Radio Environment
- Distance between modules
- Obstacles or line of sight
- Selected packet types (1, 3 or 5 slot packets and DM, DH or EDR packets.)
- Application protocol

To configure the module for the highest possible throughput, for short range and good link quality, one of the following link policies should be used.

- *AT*AMLP=0,3,1*: EDR packets with the connectBlue RSSI packet control algorithm (low RSSI forces the module to 2-DH1 packet).
- *AT*AMLP=2,0,1*: EDR packets without the connectBlue RSSI packet control algorithm (data size determines size of EDR packet).

This means that the module is allowed to use EDR packets which is best regarding a high maximum throughput. The connectBlue RSSI packet control algorithm will force the radio to use 1-slot EDR packets when RSSI value goes down (e.g. long range) which is otherwise only used for a small data size. The drawback of using only EDR packets compared to DM/DH packets is that the link gets more sensitive to disturbances or long range. Hence, it works best at close range in a good radio environment.

It is recommended to configure the module for DH/DM packets if a throughput of ~500-700 kbits/s is required (depending on which module is used). The DH/DM choice will mean a more robust and reliable link.

- *AT*AMLP=0,4,1*: DM/DH packets with the connectBlue RSSI packet control algorithm (low RSSI forces the module to DM1 packets).
- *AT*AMLP=13,0,1*: DM/DH packets without the connectBlue RSSI packet control algorithm (data size determines DM/DH packet).

With the 3xx modules it was a good idea to configure one side for 5-slot packets and the other for 1-slot packets to get the maximum *asynchronous* bit rate. With the 4xx modules this is normally not necessary. The radio will select 1-slot packets if data fits, which means you will automatically get an *asynchronous* link if only one side transmits large data packets. Hence, in most cases, both sides can be configured with the link policy of 2,0.

If the *application protocol* transmits packets with little data, the throughput will go down. The CPU will have to handle many small packets that will increase the CPU load, and there is a lot of Bluetooth protocol header data compared to application data in packets over air. Hence, to get a high throughput make sure that data packets transmitted on the UART are large.

Also, if the radio link is poor there will be many retransmissions, which will bring down the throughput. Poor radio environment, bad antenna placement, or a long range can cause the link to get poor.

When using a ping-pong protocol over SPP e.g. iAP1 the latency has big impact on the throughput. The latency for the UART(s) can be minimized by shorten the UART poll time. This is done with feature mask 1 bit 7 with AT*AMWFM e.g. AT*AMWFM=1,0x80,1. This might increase the power consumption slightly.

4 Latency

The latency consists of several different parts. First data must go through the serial line, then the CPU, over air, CPU on receiving side and finally the serial line on the receiving side.

For a 10 bytes packet that is transmitted using a baud rate of 57,6 kbits/s this typically means ~2 ms on the serial line on the transmitting side and another ~2 ms on the receiving side (not including CPU time and over air). Also note that there is a minimum receive timeout of ~3 bytes (or minimum ~1 ms) included to trigger the UART.

A very rough estimate on the CPU time is ~2-4 ms on each side. However, in addition to this, by default, the Bluetooth stack polls for data every 40 slots. Since each slot is 0,625 ms this means an additional delay of 0-25 ms. The *Link Policy* AT command can be used to force a shorter poll time. The Quality of Service (QoS) Bluetooth mechanism is used to change the poll time. It is recommended that the link policy for QoS in combination with DM1 packets be used.

- AT*AMLP=10,0,1 on the master side (QoS for shortest poll and DM1 packets)
- AT*AMLP=9,0,1 on the slave side (DM1 packets)

The second parameter can be used to set a different poll time. Since polling often means higher power consumption it may not be a good idea to poll all the time. It is then possible to set a poll time shorter than the default (40) but not the shortest one (which is 2 for the above case).

Please note that the DM1 packets are used to get a more robust and deterministic behaviour (see Long Range Section). However, it can only contain 8 bytes of application data for an SPP link. That means if the application data packet contains 9 bytes, the delay will roughly be doubled since two packets over air is needed.

If application packets are bigger than 8 bytes there is also another link policy to select QoS and DM packets (AT*AMLP=12,0,1). This will then be some kind of compromise between the above choices. The module may still select 5 slot packets but it will always use forward error correction, which makes it more robust and deterministic.

The cost of polling more often, using the QoS link policy, is that the power consumption on both sides increases. Furthermore, it will also mean more activity over air, which might not be desirable.

It is also possible to shorten the latency using a sniff link policy. This may be a better choice if the power consumption is also important.

The UART poll time can also be reduced to minimize latency, see Throughput chapter above.

5 Range

For a poor connection (e.g. long range), the module should use DM1 packets since DM1 packets are short packets with 10-bytes of forward error correction. This means that the chances to receive (or receive and successfully correct) packets increases. Hence, less re-transmissions and a more robust link regarding range and poor radio environment.

To configure the module for long range consider the following link policies.

- *AT*AMLP=0,4,1*: DM/DH packets with the connectBlue RSSI packet control algorithm to force DM1 packets at a long range.

- *AT*AMLP=9,0,1*: DM1 packets only which is always the best regarding range.
- *AT*AMLP=1,0,1*: DM1 and DH1 packets only. The radio will itself change between DM1 and DH1 depending on link quality. This setting allows for better throughput than DM1 packets only and still use DM1 when the link is poor.

The connectBlue RSSI packet control algorithm is quite slow which means that it may use DH5/DM5 some time before using DM1 when a link goes from good to bad. Therefore, it may be better to manually configure the module to use only DM1 packets even if the link is temporarily good. This configuration is done using the *Link Policy* AT command (AT*AMLP=9,0,1). The cost of this configuration is lower maximum throughput of ~90 kbits/s.

Furthermore, it is easier to keep a connection that is active than to set-up a new connection. Hence, at a certain range it may be possible to send data on an active connection. However, if the connection is lost it may be difficult to set it up again. To set-up connections more robustly, the *fast connect* scheme can be used (see Fast Connect & Discovery Section). Tests show that using the *fast connect* configuration (AT*AMWFM=1,2,1 on server) not only makes faster connections but also more robust connections. Hence, it will be easier to set-up connections when the range is long or the radio environment is poor. The cost of enabling fast connect is increased power consumption of the server when the connection is not active since the server listens more actively for incoming connections.

6 Power Consumption

For many applications it is important that the power consumption is kept low. Both when there is no connection and when there is a connection.

Apart from turning the power off there are a few configuration options to lower the power consumption.

For modules not configured as servers, it is normally possible to make the module both *non-discoverable* (AT*AGDM=1,1) and *non-connectable* (AT*AGCM=1,1). This means that other modules will not find it doing inquiries and they cannot set-up connections to that module (which is ok since it is a client only).

If the modules are located close to each other it is also possible to decrease the maximum output power for transmission using the *Max Output Power* AT command (AT*AMMP=128,1). By default the value is set to 255, which always means maximum output power the module can achieve. Normally, the module will tune down the output power once a connection is active.

For many applications it is possible to set the module in stop mode, which is the lowest power consumption mode. This is done using the *Power Mode* AT command (AT*AMPM=3,1). There are some restrictions on the behaviour of the module using this mode. Hence, it may not be feasible for all applications. For example, the DSR pin of the module must be activated some ~10 ms before transmitting data and inactivated to enable stop mode. Also, a remote peer may not be enabled with the “always connected” parameter set (AT*ADWDRP).

Furthermore, it is also possible to turn off the LED (if any) when the module enters stop mode. This configuration is set using the *Feature Mask* AT command (AT*AMWFM=1,1,1).

To lower the power consumption when there is an active connection, the *Link Policy* AT command can be used. Typically one of the *sniff mode* configurations are used. The longer the sniff period is, the lower the power consumption is. Of course this also means a longer reaction time. For example, a sniff period of 200 ms (AT*AMLP=7,0,1) can delay a transmission for up to 200 ms depending on when the transmission is made. The second parameter of the *Link Policy* command for sniff configuration has a specific meaning. The default value of 0 means that sniff is always active. If set to 1 (AT*AMLP=7,1,1), the module exits sniff mode when data needs to be transmitted. Sniff is activated again when the there is no data to transmit for 1 second. *Please read the release notes regarding the use of sniff mode*. It is also possible to combine stop mode and sniff mode for the lowest possible power consumption. When doing this, thorough tests must be performed to guarantee robust behaviour.

7 Connection and Discovery Time

In Bluetooth 2.0 and later there is something called interlaced page and inquiry scan. Basically this means that the module listens on more frequencies to quicker detect if a remote device is trying to set-up a connection to it (page scan) and/or if a remote device is searching for Bluetooth devices (inquiry scan). Hence, there is nothing changed on the side that makes the connection attempt (page) or the inquiry.

The cost is a high increase in power consumption since the receiver listens more actively.

To configure the module for *fast connect* (interlaces page scan) and/or *fast discovery* (interlaces inquiry scan), the *Feature Mask* AT command is used (AT*AMWFM=1,6,1). Please note that typically this is done on the server side and not client side.

The average connection time, with default settings, is ~2.2 seconds and with *fast connect* the average is decreased to ~0.5 seconds.

8 WLAN Co-existence

First of all there is the *Adaptive Frequency Hopping* (AFH) algorithm that is active by default. This means that the Bluetooth module will automatically remove frequencies that are blocked (by e.g. a WLAN network). Typically, this means that the frequencies that are used by the WLAN network will not be used. Normally it works very well. However, there are cases where this is not enough.

Problems with the AFH algorithm

- The AFH algorithm takes a few seconds before getting active. This means that for a few seconds after a Bluetooth connection is setup, the WLAN network may be affected.
- Frequencies previously removed are tested again after 30 seconds. Hence, if there is no traffic on the WLAN network for 30 seconds, the removed frequencies are added again which means that the WLAN network may again be affected (at least for a few seconds).
- The AFH algorithm is only used on an active Bluetooth connection. This means that for inquiries and connection attempts it is not used. Therefore, inquiries and connection attempts may still affect the WLAN network.

For the problem with inquires there are no really good solutions. Best is if you can avoid inquires or at least decrease the maximum output power of the module (AT*AMMP) to not disturb the WLAN network.

It is possible that a longer distance between the Bluetooth module and the WLAN module is enough to remove any interference between the two technologies and the configuration options described below might then not be needed.

There are some configuration options available to handle some of the remaining problems.

There is a *Channel Map* AT command to manually exclude frequencies (AT*AMCM) that are used by the Bluetooth module. Hence, if the frequencies of the WLAN network are known (which is normally the case), these frequencies can be excluded using the channel map command and the Bluetooth module will not use them once the connection is active. However, inquiries and connection attempts still uses all frequencies as for the AFH algorithm.

When a remote peer is defined (to set-up a connection) it is possible to change the *connection timeout* (page timeout, default 5 seconds) to a much shorter time. We have made tests with as short timeout as 80 ms. The client will then only try to connect for 80 ms before giving up. Since a normal Bluetooth connection normally takes ~2 seconds this would fail unless the server is configured for *fast connect* (See Connection & Discovery Time section). During our tests, a client configured with a page timeout of 80 ms will succeed most of the times setting up a connection to a server configured for *fast connects*. Of course it will fail every now and again and this must be handled by the application. If using the "always connected" option for the *remote peer* AT

command (AT*ADWDRP) it is also possible to change the default connection period from 10 seconds to a much lower one. A client that failed to set-up a connection will then try again after a much shorter time. We have run tests with the client page timeout to 80 ms and the "always connected" period to 3 seconds. The affect on the WLAN network is then hardly measurable.