www.qorvo.com
lpw.support@qorvo.com

gpCom Reference Manual

API Description

Version 2.10.2.0
November 15, 2021

Contents

Chapter 1

Introduction

The *gpCom* component is the embedded implementation of the Greenpeaks communication protocol.

1.1 Message Format

Each message has a six-byte header, followed by an optional string of data of variable length, and an optional footer of two bytes, consisting of a 16 bit CRC value . Maximum total packet length is restricted to 6 + 4095 + 2 bytes = 4103 bytes. 1.1 displays the general format of a message sent over the serial interface.

Bytes:	3	1	1	1	N (max 4095)	2
Field:	SynHeader (*SYN*)	Length	FrameControl	ModuleId	Payload	CRC

Figure 1.1: General message format

The *SynHeader* field contains the string *SYN* (or the sequence (0x53, 0x59, 0x4E) in hexadecimal format). The *Length* field contains the 8 bits LSB of the payload length. The payload length excludes the header and footer bytes. The *FrameControl* field contains the 4 MSB bits of the length, while the *ModuleId* identifies the embedded module which data packet came from or is sent to. The *Payload* field contains the bytes you want to send to the module. The LSB of the *Payload* is sent first. The footer contains a two byte *CRC*: first the LSB, followed by the MSB.

1.1.1 Synchronization

As soon as the application is started, it starts sending out messages over its communication channel. Before a listening application is able to interpret data from the target, it needs to synchronise with the data stream generated by the target. The synchronisation can be achieved by following these steps:

1. Read the next byte and increment the read pointer of the received stream until the pattern *SYN* is found. The read pointer points now to the byte after the *SYN* pattern. Go to point 2.
2. Read the length byte. This forms the 8 LSB bits of the 12-bit length.
3. Read the FrameControl byte. The lower 4 bits contain the MSB part of the length. The top 4 bits are reserved.
4. Read ModuleId byte. Do not check this value but store it for later checking. Packets for and from an unknown moduleId should be discarded later, but the synchronisation process should go on. Go to the next step.

bits:	4	4
field:	Reserved	Length MSB

Figure 1.2: FrameControl byte

5. Read the bytes of the serial payload based on the length information. Go to the next step.
6. Read the CRC bytes and check the correctness of this value by comparing it to the CRC calculated over all bytes of the packet you received. If the CRC is correct, then put the read pointer after the read CRC bytes, send the packet to the handling block (based on the moduleId) and go back to step 1. Otherwise do not change the read pointer (it stands already after the *SYN* pattern and go directly back to step 1.

The client should keep on parsing the input stream to ensure synchronisation is kept. Each time the *SYN* pattern is not found at the beginning of a packet, an exception should be raised. When the moduleId is unknown the message can be safely discarded.

1.1.2 CRC calculation

Every message is terminated with a 16 bit CRC. The LSB of the CRC is the first byte of the footer, the MSB of the CRC is the second byte of the footer. The target will generate this CRC at the end of each sent message and likewise the target will expect a CRC at the end of every message coming from the PC.

The PC should calculate the CRC of each incoming message and compare the calculated CRC against the CRC value sent as the last byte of the message. If the calculated CRC does not match the CRC value sent with the message, then the PC should discard the message. The target uses the same strategy in interpreting incoming messages from the PC. The CRC is a 16 bit CRC with the generic polynomial 0xA001. The following shows an example implementation in C:

```
UInt16 crc 16_update(IInt16 crc, UInt8 a)
{
    int i;
    crc ^= a;
    for (i = 0; i <8 ; ++i)
    {
         if (crc &1)
            crc = (crc >> 1) ^ 0xA001;
        else
            crc = (crc >> 1);
    }
    return crc;
}
```

To check the correctness of the CRC of a data packet, the CRC calculated on all bytes of a packet excluding the CRC bytes should match the read CRC value. Or the CRC value can be calculated on all bytes of the data packet including the CRC bytes, and then the result should be 0.

Chapter 2

File Documentation

2.1 gpComLinkEvents.h File Reference

Macros

- #define **GP_COMPONENT_ID_COMLINKEVENTS** GP_COMPONENT_ID_COM

Typedefs

- typedef UInt32 **gpCom_CommunicationId_t**

Functions

- void **gpComLinkEvent_Ping** (void)

2.1.1 Detailed Description

Declarations of the public functions and enumerations of gpComLinkEvents.