Two-Wire Master over Serial Protocol (PR01)

1 Introduction

This document defines the PR01 protocol for controlling a two-wire master node over a framed or message oriented link.

1.1 Revision History

Version	Author	Date	Change
0.01	CJH	13 Sep 2012	Initial release
0.02	CJH	28 Apr 2013	Adjusted ASN.1 description

1.2 Definitions

{1B} byte/octet

[2B] word (big endian)

(4B) long (big endian)

2 Communication Profile

PR01 is designed to fit into low speed USB control transfers; All packets are less than 9 bytes in size. It is also possible to pass PR01 over any framed or message oriented link.

3 Services

The first octet of any message identifies the message type. Unrecognised or invalid messages are silently discarded.

Version number allows new services to be added and existing services expanded. Version is always backwards compatible.

Identifier	Service	Version
0x00	VERSION_req	0
0x01	VERSION_cnf	0
0x02	READ_req	0
0x03	READ_cnf	0
0x04	WRITE_req	0
0x05	WRITE_cnf	0

Table 1: Service Primitives

3.1 Response Codes

Response	Name	Meaning
0	success	Operation complete
1	temporary_failure	Invalid input
2	busy	Primary did not claim the bus
3	collision	Collision detected during write; bus is released
4	timeout	SCL is low for timeout; bus is released
5	no_ack	Secondary does not ACK; write unsuccessful
6	no_bus	Primary does not hold the bus (need to start)

Table 2: PR01 Response Codes

3.2 VERSION

Get the protocol version supported by the bridge.

```
{0x00}
{0x01}{version}
```

3.3 READ

Read a byte from the two-wire bus.

```
{0x02}{option}
{0x03}{0x0}{data}
      {0x1}{response}
```

Option	Meaning
Bit 0	Begin with start
Bit 1	End with stop
Bit 2	End with start

3.4 WRITE

Write a byte to the two-wire bus.

```
{0x04}{option}{data}
{0x05}{response}
```

Option	Meaning
Bit 0	Begin with start
Bit 1	End with stop
Bit 2	End with start

4 ASN.1

```
-- encoding and tagging rules are as indicated in the previous chapters
PR01
DEFINITIONS
AUTOMATIC TAGS
::= BEGIN

PR01_pdu ::= CHOICE
{
    -- VERSION 0 SERVICES

    --get version
    version-rq  [0] Version-RQ,
    version-cnf [1] Version-CNF,

    --read a byte from bus
    read-rq     [2] Read-RQ,
    read-cnf    [3] Read-CNF,

    --write a byte to the bus
    write-rq    [4] Write-RQ,
    write-cnf   [5] Write-CNF,
}

--get the version number
Version-RQ ::= NULL
Version-CNF ::= SEQUENCE
{
    version Uint8
}

--try to read a byte from the bus
--options
--bit 0: begin with start
--bit 1: end with stop
--bit 2: end with start
Read-RQ ::= SEQUENCE
{
    option Uint8
}
Read-CNF ::= SEQUENCE
{
    result  CHOICE {data [0] Uint8, result [1] PR01_result}
}

--try to write a byte to the bus
--options
--bit 0: begin with start
--bit 1: end with stop
--bit 2: end with start
Write-RQ ::= SEQUENCE
{
    option Uint8,
    data Uint8
}
Write-CNF ::= SEQUENCE
{
    result  PR01_result
}

--result codes
PR01_Result ::= ENUMERATED
{
    success             (0),
    temporary-failure   (1),
    busy                (2),
    collision-detected  (3),
    timeout             (4),
    no-ack              (5),
    no-bus              (6)
}
```

```
Uint8 ::= INTEGER(0..255)

END
```