TPI Serial Interface

Development Notes

Agnetha Korevaar

20 Mar 2018

v1.10

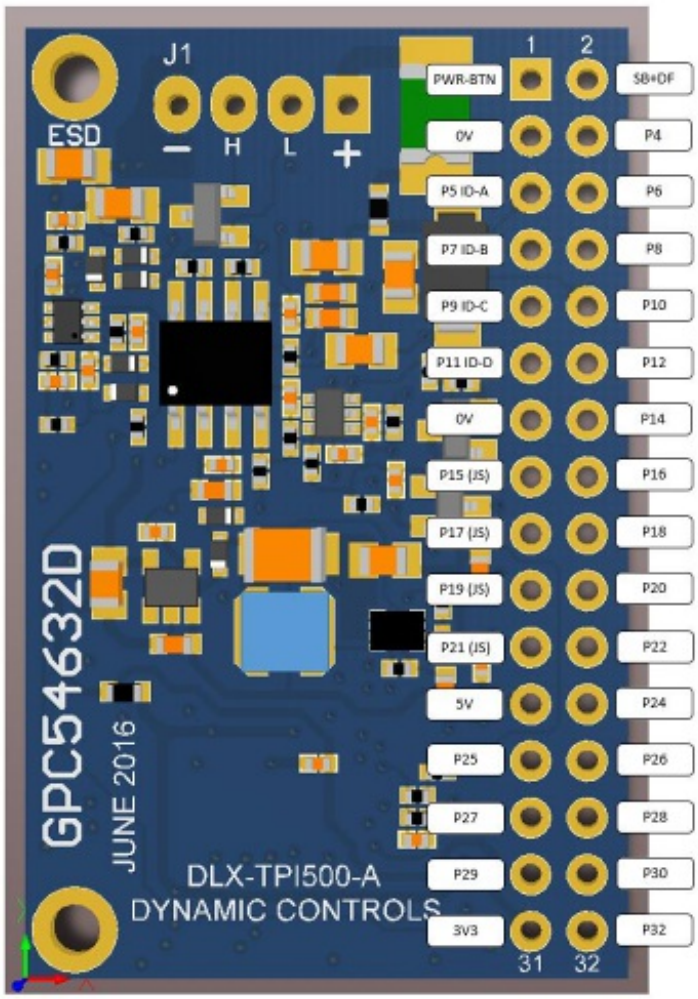
TABLE OF CONTENTS

1 TPI Custom Serial Interface

The TPI can be used to allow read and write access to specific variables.

1.1 TPI Configuration

The TPI has a 32 pin header as shown below. Note: TPI rev D hardware is required.



TPI Pin Description

Common pin out:

- Ground: P3
- 3v3: P31

LEDs:

- On: indicates 3v3 power supply is on
- Status: blinks to indicate CPU is active
- Fault: if this LED is on it indicates that jumpers are incorrectly fitted

Switched inputs can be used for testing drive affecting mods (or extra emergency stops).

The TPI configuration can be selected using the hardware id and configuration pins as described below.

1.2 Serial Configuration

Note USART is at 3.3V logic level, 115200 8n1.

To enable the serial interface, the following pins must be connected to ground:

- P7, P9 (for configuration ID 0x09, on breakout board: jumper B & C in place)
- P4 (for new hardware id 0xE7)

For a read only version of the serial interface (i.e. a data logger) the following pins must be connected to ground:

- P9 (for configuration ID 0x0B, on breakout board: jumper C in place)
- P4 (for new hardware id 0xE7)

Pin out:

- USART TX: P24
- USART RX: P32

2 Serial Protocol Specification

2.1 Protocol Overview

2.1.1 Packet Structure

The structure of a packet is as described below.

Start Delimiter	Type Identifier	Length	Data Field	CRC	End Delimiter
1 Byte: 0xF0	1 Byte	1 Byte: N	N Bytes	1 Byte	1 Byte: 0xF0

Packet Structure Diagram

Note:

- The delimiting bytes shall always have the same value, 0xF0.
- The Type Identifier byte is described in the following sections. Its value may range from 0x00 to 0xEF.
- The Length (N) value is the length of the Data Field. It does not include the delimiters, Length, Type Identifier, or CRC. N is in the range 0 - 255.
- No stuffing or padding is done in either the data field or the CRC calculation.
- The CRC is 8 bits in size, and uses CRC-8/SAE-J1850. Refer to: https://www.autosar.org/fileadmin/files/standards/classic/4-3/software-architecture/safety-and-security/standard/AUTOSAR_SWS_CRCLibrary.pdf
- The CRC is calculated over the Type Identifier, Length and Data Field
- If the CRC is incorrect in a received packet, then the receiving module shall discard the packet. It shall also discard the packet if the next received byte after the CRC is

not the packet delimiter.

2.1.2 Sample Packet

Note: all sample packets are given as the hexadecimal values of the bytes. TPI is used to indicate communications originating from the third party interface board. Device is used to indicate communications originating from a third party device. For example, the packet `f0 01 02 00 01 da f0` is deconstructed as follows:

- `f0` Start delimiter
- `01` Type identifier
- `02` Data length
- `00 01` Data Field
- `da` CRC
- `f0` End delimiter

2.1.3 Type Identifiers

Available type identifiers (in hexadecimal):

- RESPONSE_STATUS : `01`
- REQUEST_CONNECTED_MODULES : `70`
- RESPONSE_CONNECTED_MODULES : `71`
- REQUEST_ENABLE_USER_INPUT : `90`
- RESPONSE_USER_INPUT : `91`
- REQUEST_ENABLE_MOTOR_SPEED : `92`
- RESPONSE_MOTOR_SPEED : `93`
- REQUEST_ENABLE_BUTTON_PRESSES : `94`
- RESPONSE_BUTTON_PRESSES : `95`
- REQUEST_ENABLE_GYRO_TURN_SPEED : `96`
- RESPONSE_GYRO_TURN_SPEED : `97`
- REQUEST_ENABLE_ACTIVE_USER_FUNCTION : `98`
- RESPONSE_ACTIVE_USER_FUNCTION : `99`
- REQUEST_ENABLE_SPEED_SCALING : `9A`
- RESPONSE_SPEED_SCALING : `9B`
- REQUEST_MODIFY_DEMAND : `88`

2.2 Data Packets

2.2.1 Status Response Packet

The TPI sends a status OK message when the serial interface is ready. Otherwise it sends a status packet in response to a request from the third party device. A third party device can also send a status esponse packet and the TPI will respond with a status OK. This can be used to check if the wheelchair (and hence TPI) is powered up.

Type Identifier: RESPONSE_STATUS

2 data bytes:

- Status code
- Type Identifier of the request packet the TPI is responding to

Status Codes (in hexadecimal)

- STATUS_OK : `00`
- UNKOWN_TYPE_IDENTIFIER : `01`
- INVALID_DATA : `02`
- INVALID_CRC : `03`

2.2.2 Status Sample packets:

- Device sends a STATUS_OK: `f0 01 02 00 00 c7 f0`
- TPI replies with STATUS_OK: `f0 01 02 00 01 da f0`

2.2.3 Request Connected Modules Packet

This packet has zero data bytes. Note this packet should be sent a few seconds after system power up otherwise some modules may not be reported.

Type Identifier:

- REQUEST_CONNECTED_MODULES

2.2.4 Response Connected Modules Packet

Data length is the number of modules detected. Module types are encoded as follows:

```
PMDO       : 00
REMDO      : 01
LAK        : 02
PMLE       : 03
REMLE      : 04
PMAL       : 05
REMAL      : 06
GYRO       : 07
ACT        : 08
TPI        : 09
REMRE      : 0A
```

```
TILT      : 0B
DISP      : 0C
ACU       : 0D
INPUT     : 0E
OUTPUT    : 0F
CR        : 10
TPI_ACU   : 11
```

Note: each module type will only be reported once.

2.2.5 Connected Modules Sample Packets:

- Device sends a REQUEST_CONNECTED_MODULES: `f0 70 00 95 f0`
- TPI responds with RESPONSE_CONNECTED_MODULES (TPI, REMRE, PMAL): `f0 71 03 09 0a 05 58 f0`

2.2.6 Data Stream Enable Packets

Type Identifiers:

- REQUEST_ENABLE_USER_INPUT
- REQUEST_ENABLE_MOTOR_SPEED
- REQUEST_ENABLE_BUTTON_PRESSES
- REQUEST_ENABLE_GYRO_TURN_SPEED
- REQUEST_ENABLE_ACTIVE_USER_FUNCTION
- REQUEST_ENABLE_SPEED_SCALING

1 data byte: * Disable: 0x00 or Enable: 0x01

2.2.7 Data Stream Response Packets

Once a particular data stream is enabled, a response message shall be sent every time new data is available.

Type Identifiers:

- RESPONSE_USER_INPUT
- RESPONSE_MOTOR_SPEED
- RESPONSE_BUTTON_PRESSES
- RESPONSE_GYRO_TURN_SPEED
- RESPONSE_ACTIVE_USER_FUNCTION
- RESPONSE_SPEED_SCALING

RESPONSE_USER_INPUT has 3 data bytes:

- Joystick X demand % (as an int8_t, range -100 to 100)
- Joystick Y demand % (as an int8_t, range -100 to 100)

- Speedpot demand % (as a uint8_t, range 0 to 100)

RESPONSE_MOTOR_SPEED has 4 data bytes:

- Left Motor % high byte
- Left Motor % low byte
- Right Motor % high byte
- Right Motor % low byte

Note: for each motor the demand is given as a percentage scaled from -32000 to 32000 (int16_t).

RESPONSE_BUTTON_PRESSES has atleast 3 data bytes:

- Number of button press events
- Button identifier
- Button state. Pressed: 0x01, Released:0x00

Note: for each button press event, there shall be button identifier and button state.

RESPONSE_GYRO_TURN_SPEED has 2 data bytes:

- Current chair rotational speed as an int16_t in units of degrees per second (dps) x 128 eg: `f7 18 = -17.81 dps`

RESPONSE_ACTIVE_USER_FUNCTION has 1 data byte:

- Index of current active user function

RESPONSE_SPEED_SCALING has 4 data bytes:

- Forward (as a uint8_t, range 0 to 100)
- Reverse (as a uint8_t, range 0 to 100)
- Left (as a uint8_t, range 0 to 100)
- Right (as a uint8_t, range 0 to 100)

These represent the scaling applied to the demand in each direction.

2.2.8 Data Stream Sample Packets:

- Device: Enable user input data stream: `f0 90 01 01 78 f0`
- TPI: STATUS_OK, in response to (enable user input): `f0 01 02 00 90 2c f0`
- TPI: User Input Response (joystick x: 12%, joystick y: 92%, speed pot: 26%): `f0 91 03 0c 5c 1a 31 f0`
- TPI: User Input Response (joystick x 0%, joystick y 0%, speed pot 26%): `f0 91 03 00 00 1a 12 f0`
- Device: Disable user input data stream: `f0 90 01 00 65 f0`
- TPI: STATUS_OK, in response to (disable user input): `f0 01 02 00 90 2c f0`

2.3 Drive Modifier Request Packets

2.3.1 Direct X,Y Drive Input Modification

This packet allows direct modification of the drive demand **and so it should be used with care!**

- REQUEST_MODIFY_DEMAND

2 Data bytes:

- x (int8_t, %, range -100 to +100)
- y (int8_t, %, range -100 to +100)

X modifies the turn demand, Y modifies the linear demand. **Note: Modified demand is only applied if the wheelchair joystick is deflected, and the modified demand is less than 45ms old.**

2.3.2 Drive Modification Example

This example shows the user input stream being listened too, and a new drive demand being set to half the user input value. Note: The User Input data is updated approximately every 15ms, and ideally the modified user demand should be sent before the next user demand message is received.

```
TX: REQUEST_ENABLE_USER_INPUT:    [True]
TX: f0 90 01 01 78 f0

RX: RESPONSE_STATUS:    STATUS_OK, in response to
(REQUEST_ENABLE_USER_INPUT)
RX: f0 01 02 00 90 2c f0

RX: RESPONSE_USER_INPUT:    UI: x 0%, y 85%, sp 91%
RX: f0 91 03 00 55 5b 23 f0

TX: REQUEST_MODIFY_DEMAND:    [0, 42]
TX: f0 88 02 00 2a a6 f0

RX: RESPONSE_STATUS:    STATUS_OK, in response to (REQUEST_MODIFY_DEMAND)
RX: f0 01 02 00 88 09 f0
```