CANbus protocol and applications for STAR TOF Control

J Schambach[1], L Bridges[2], W Burton[2], G Eppley[3], K Kajimoto[1] and T Nussbaum[3]

[1]Physics Department, University of Texas at Austin, Austin, Texas 78712, USA
[2]Blue Sky Electronics, 401 Studewood, Suite 203, Houston, Texas, 77007, USA
[3]Bonner Laboratory, Rice University, Houston, Texas 77005, USA

Email: jschamba@physics.utexas.edu

Abstract. A large-area Time-of-Flight (TOF) system based on Multi-gap Resistive Plate Chambers (MRPCs) has recently been installed in the STAR experiment at RHIC. The approximately 23000 detector channels are read out and digitized using custom electronics based on the CERN NINO and HPTDC chips. The data are sent to the experimental data acquisition system (DAQ) using the ALICE fiber optics based Detector Data Link (DDL). The readout system consists of a total of approximately 2100 custom electronics boards mounted directly on 120 TOF trays, as well as four DAQ and trigger interface boards outside the detector that collect data from 30 trays each and send it to DAQ. Control and monitoring of these electronics boards is done using a tiered network of CANbus connections to a control PC. We describe the physical implementation and topology of the CANbus connections and the custom protocol developed for this project. Several command-line tools as well as a Qt4-based graphical tool developed on the host side to facilitate configuration, control, and monitoring of the TOF system are also described.

1. The TOF detector upgrade to STAR at RHIC

STAR is designed to search for signatures of quark-gluon plasma formation and to investigate the behavior of strongly interacting matter at high energy density. A unique strength of the STAR detector is its large, uniform acceptance capable of measuring and identifying a substantial fraction of the particles produced in heavy-ion collisions. Central to this capability is a cylindrical time projection chamber (TPC) four meters in diameter and five meters long. The ionization left in the TPC gas volume allows tracking of the particles as well as particle identification (PID) via dE/dx. A recently completed barrel Time-Of-Flight (TOF) detector upgrade extends the PID capabilities of STAR to higher momentum.

The TOF detector consists of a total of 3840 Multi-gap Resistive Plate Chambers (MRPCs) with 6 readout pads each, distributed over 120 trays surrounding the TPC. There are 23040 MRPC channels in the TOF system that achieve an average timing resolution of 85 ps (rms). The MRPC channels provide a stop time for the TOF measurement, with timing relative to a common clock. The start time, or time of collision, is determined by the Vertex Position Detector (VPD), two barrels of scintillator-PMT based detectors each containing 19 scintillators with their PMTs close to the beam pipe, which are digitized with electronics identical to the stop-time electronics and relative to the same common clock.

2. TOF Electronics

Shown in figure 1 is the top level diagram of the electronics for the start and stop sides of the present system. The electronics chain on the start side is similar to that on the stop side.

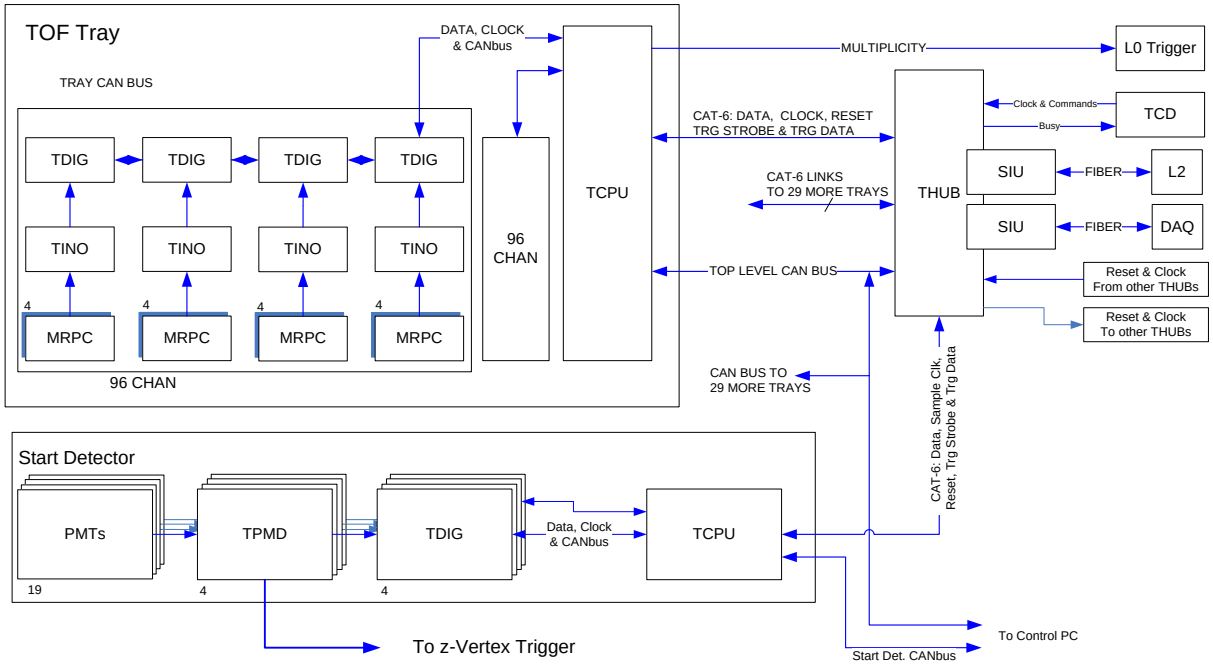

Figure 1: Top-level diagram of the TOF electronics, showing the tray electronics containing the TINO, TDIG and TCPU boards, the start detector electronics containing the TPMD, TDIG, and TCPU boards, as well as the off-detector electronics board THUB, which interfaces to DAQ and Trigger.

The basic unit of the TOF is a tray. The TOF readout electronics design uses on-detector electronics on the MRPC trays, tray-level data buffering, and fiber-optics data communication with STAR's DAQ and Level-2 Trigger. Each tray consists of 32 MRPC modules (192 detector channels) and eight each of three different types of electronic circuit cards: in the order of the data flow TINO, TDIG, and TCPU. Groups of 30 trays send their data to a THUB card that interfaces to the STAR trigger and transmits the data over optical fiber to two fiber receivers, which are part of the STAR DAQ and Level-2 Trigger system, respectively. Two of the THUB cards also receive data from the two barrels of the VPD, and combine these data with the tray data before sending it to DAQ.

The start detector PMTs are connected to the TINO board, which contains the front end electronics to amplify and digitize analog MRPC signals in NINO a custom ASIC (developed by CERN/LAA). The output of TINO is passed to the TDIG card, where timing of the signals relative to a master TOF clock is recorded using the CERN HPTDC ASIC. Parallel data words from the 8 TIDG cards of each tray are read out in turn by the TCPU card, which buffers and formats the digital detector information. These formatted data are then passed over a copper SerDes link to the THUB card. The TCPU card also distributes the 40MHz reference clock to the TDIG cards, and sends multiplicity information from TOF trays to the lowest level logic of the trigger system (L0-trigger).

The start detector VPD consists of two barrels of 19 scintillator detectors each with associated PMTs connected to a card called TPMD, which contains level adapters and discriminators. From there, the data flow is similar to the tray electronics, through TDIG and TCPU to one of the THUB cards on each side of the STAR detector.

The THUB card interfaces 30 or 31 TCPU cards to STAR trigger and DAQ. The data interface consists of a high-speed serial/deserializer (SerDes) device that transmits and receives data over twisted pairs in low-cost CAT-6 cables. Each THUB card has two custom daughter cards developed by the CERN ALICE experiment, called the Source Interface Unit (SIU), that provides a dual-fiber interface to STAR DAQ and the Level-2 trigger system. The receivers on the other side of the fibers in DAQ and trigger are Linux PCs with a PCI-X based optical interface card called the Read-Out Receiver Cards (RORC). The receiver PCs are connected via Ethernet and Myrinet switches to the rest of DAQ and trigger, respectively.

There are a total of 960 TINO, 10 TPMT, 970 TDIG, 122 TCPU, and 4 THUB cards in the TOF system. All of the TDIG, TCPU, and THUB cards have CANbus interfaces, and are connected via CANbus networks to a control PC for configuration, calibration, and control of the electronics.

3. Controller Area Network (CAN)

CAN is a message based protocol designed in 1983 at Robert Bosch GmbH and updated to CAN 2.0 in 1991 [1]. Originally intended for automotive applications it is now also used in many other control and data acquisition areas. CAN is a multi-drop, multi-master broadcast serial bus standard for connecting electronic control units (ECU's) where each node is able to send and receive messages, and messages are sensed by all nodes. Data messages transmitted from any node on a CAN bus do not contain addresses of either the transmitting node, or of any intended receiving node; instead, the message is labelled by an identifier that is unique throughout the network. All other nodes on the network receive the message and each performs an acceptance test (filtering) on the identifier to determine if the message, and its content, is relevant to that particular node. If the message is relevant, it will be processed; otherwise it is ignored (multi-cast). The identifier also determines the priority of the message: the lower the numerical value of the identifier, the higher the priority. This allows implicit arbitration if two (or more) nodes compete for access to the bus at the same time. The higher priority message is guaranteed to gain bus access as the other nodes notice that a higher priority message is being sent and cease transmission. Lower priority messages are automatically retransmitted in a subsequent bus cycle when there are no higher priority messages pending.

A data frame (message) on the CAN bus contains the following elements (in addition to various frame-control, CRC, and delimiter fields):

- **Arbitration Field:** contains the Message Identifier (MID) which is 11 bits for standard format (CAN 2.0A/B) or 29 bits for extended format (CAN 2.0B only) messages, and the Remote Transmission Request (RTR) bit (= 0 for data frames)
- **Control Field (CONTROL):** amongst other things, contains the Data Length Code (DLC), which is the number of payload data bytes to follow (which is allowed to be 0 to 8)
- **Data Field (DATA):** 0 to 8 payload data bytes

CAN uses NRZ encoding (to minimize the number of transitions in a message stream) with bit stuffing (to ensure sufficient numbers of synchronization edges) for data transmission on a differential two-wire bus. The bus structure is multi-drop with termination resistors on each end of the cable. CAN speeds of up to 1 Mbit/s are guaranteed by the standard on bus lengths of up to 40m, or 500 kbit/s on bus lengths up to 100m. Even though the number of nodes on a CAN network is theoretically unlimited, typical transceiver devices limit this number to 32 or 64 nodes per network. A typical CANbus node contains a CAN transceiver for CAN signaling and bus failure management, and a micro-controller containing a CAN module with a protocol controller, several message filters, message buffers, a CPU interface and a CPU module for additional message filter capabilities higher level protocols and the user application.

4. The TOF CANbus Network

The structure of the TOF CANbus network is shown in figure 2. It is implemented in a hierarchical topology to deal with the large number (1096) network nodes. Starting from the control PC, there are four top level networks each connecting the control PC to 1 THUB and 30 tray TCPU's; and two additional top level networks attaching the two start detector TCPU's to the control PC. Each tray contains a separate tray level network connecting the 8 TDIG's within a tray to the corresponding TCPU. The TCPU's on each tray route messages between the top level networks and the tray networks based on the High-Level Protocol described below.

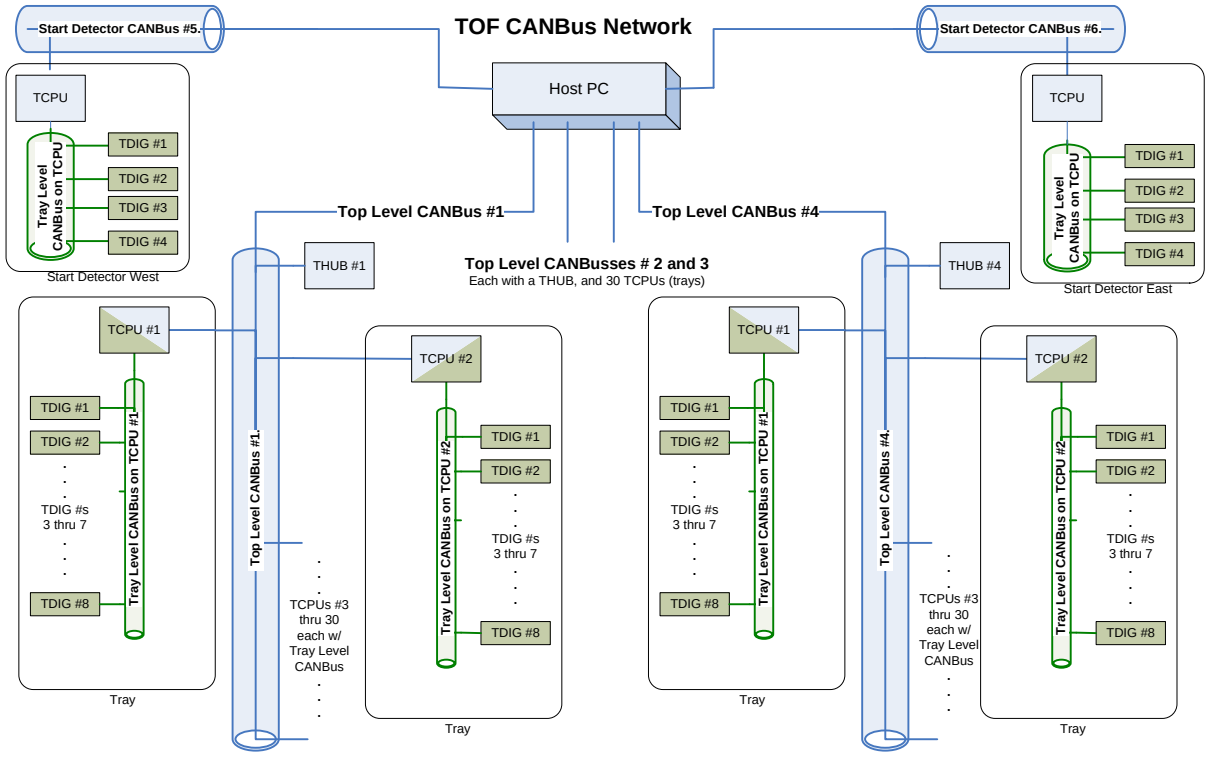

Figure 2: The structure of the TOF CANbus network. A total of 6 top-level networks connect the trays and the start detectors to the controls PC. Tray level networks connect the TDIG's with their corresponding TCPU's.

The CANbus interface to the PC is accomplished with the PCAN-USB module from PEAK Systems [2]. The PCAN-USB supports transfer rates up to 1 Mbit/s, and is compliant with both the CAN 2.0A (11-bit IDs) and CAN 2.0B (29-bit IDs) specifications. Each top level network has 1 PCAN-USB interface to the PC. Driver software and utility applications as well as a software API are provided by PEAK for both Linux and Windows.

The CANbus interface on the individual electronics boards is provided by one of three different PIC microcontroller (MCU) devices from Microchip Technology: PIC24HJ64GP506 (a 16bit MCU on TDIG boards), PIC24HJ256GP610 (a 16bit MCU with 2 CAN interfaces on TCPU's), and PIC18F8680 (an 8bit MCU on THUB). Programming of the PIC MCUs is done either in C or Assembly language. All three PIC MCUs have the following important CAN features: 1) support the full CAN protocol, 2) contain several acceptance filters and masks with dynamic association, 3) contain several programmable buffers to use as Rx or Tx message buffers with double buffering on CAN, and 4) interrupt or polling interface to the rest of the CPU. The microcontrollers interface to

devices on the electronics boards to allow for their configuration, monitoring, and control in response to CANbus messages. They also have access to the produced data, so that calibration and data debugging can also be done over CANbus.

5. The TOF CANbus High-Level Protocol (HLP)

We defined a custom HLP on top of the CAN protocol as follows. Messages are divided into two classes: messages contained within a sub-network and, messages crossing between sub-networks.

All messages contained within a sub-network (top level or tray level) use only standard message ID's (11 bits). This ID is subdivided into two fields: a 7 bit nodeID (msgID[10:4]) and a 4 bit command code (msgID[3:0]). The nodeID uniquely identifies the addressed or responding node on the network. Address 0 is forbidden, and address 0x7f (all 1's) is defined as a broadcast address (all nodes on a sub-network need to respond). This allows for up to 126 nodes to be addressed on each sub network. We further defined nodeID ranges for different kinds of boards: TDIG nodeID[6:0] = [010bbb], TCPU nodeID[6:0] = [01bbbbb], THUB nodeID[6:0] = [100000]. Each node's MCU can filter on the nodeID of the message and ignore all others.

The command code defines the kind of action a node is supposed to take in response to the message. The command codes we defined are:

Command Code	Meaning
1	DATA
2	WRITE command
3	Write Response
4	READ command
5	Read Response
7	STATUS or ALERT

DATA commands are initiated by the electronics boards and contain either one or two 4-byte words of TDC or Trigger data in message payload data bytes 0 – 7

WRITE commands use the first payload data byte to define the address, which can be a real address on the board or a memory mapped action. Data to be written is contained in payload bytes 1 – 7, with the least significant byte in data byte 1. In case of memory mapped actions, these bytes further qualify the command. Write Response messages are issued by addressed boards and repeat data byte 0, and a status code for the action in byte 1. Transmitting large blocks of data (e.g. HPTDC setup information, new FPGA or MCU firmware, or other configuration data) requires more data than can be contained in a single CAN message. Therefore a sequence of WRITE type sub-commands was defined consisting of the sub commands Block-Start, Block-Data, Block-End, and Block Disposition (what to do with the block of data).

READ commands cause data to be sent back from the addressed node to the requesting node. Byte 0 contains the address to be read which can be a real address on the board or a memory mapped action to take to produce the data. Read Response messages repeat byte 0 of the request, while bytes 1 – 7 contain the requested data, with the least significant byte of multi-byte words in data byte 1. Invalid Read messages result in a read response with only 1 data byte (the address) in the payload. Only the requesting node should listen for Read Response messages.

STATUS or ALERT messages are issued asynchronously when boards are first turned on or when they encounter error or alarm conditions. Payload byte 0 defines the nature of the alert, while bytes 1 – 7 provide additional information about the specific alert.

For messages traverseing the network heirarchy (i.e. from top-level network to tray-level network or the other way around) we use extended IDs, which contain 29 bits: 11 bits equivalent to the standard ID bits, and 18 bits of the extended ID bits. The standard ID bits are as described above (containing the nodeID of the originating node or the nodeID of the ultimate destination). The Extended ID bits are used to encode the nodeID of the TCPU (right adjusted in the extended field

which will forward the message from the top-level network to the tray-level network or from the tray-level network to the top-level network. A TCPU receiving a CAN message with an extended ID strips the extended ID bits from the message ID and sends the otherwise unmodified CAN message with the standard message ID on its tray network to the TDIG's.

6. Examples of TOF CANbus Applications

Using the above described HLP and the application development environment provided by PEAK, we wrote a large set of applications, both command line and graphics based, to interact with the TOF electronics boards. Here we describe four examples from these applications.

6.1. PCANView

PCANView is a simple graphical debugging tool provided by PEAK for the Windows OS. Received messages are displayed with ID, DLC and data bytes in a list, along with the time between messages with the same ID. Messages can be created and put into a transmit list to be sent either manually or automatically in fixed time intervals. Messages can be filtered based on their message ID.

6.2. Pcanloop

Pcanloop is a Linux command line toolset written in C++ consisting of two separate tools: 1) a program to send and receive message, display and/or save assembled and received messages, and handle error conditions; 2) a program to interact with the previous tool to assemble Tx messages symbolically and to process received messages. This toolset is used extensively for the debugging of the TOF system.

6.3. HPTDC Configurator

This is a Python based graphical user interface running both under Windows and Linux to visually construct the configuration bits for the TDC's on TDIG and to transmit those bits over CANbus to the TDIG's. The Python GUI packages used in writing this GUI are TkInter and Pmw.

6.4. Anaconda II

This GUI is Qt4 based C++ program to control, configure and monitor the TOF system in the STAR control room during data taking. This application provides the following services:

- Reads the expected electronics configuration from an Sqlite database
- Allows the user to automatically configure the electronics over CANbus after power-on or power cycle (depending on running user mode)
- Reads the current electronics status periodically over CANbus
- Displays firmware versions and chip IDs; displays and alarms problems with the configuration and status registers and board temperatures
- Between runs, resets tray electronics having radiation caused single-event upsets in their programmable devices
- Logs electronics status as well as error conditions and actions taken in response to these events

Figure 3 shows Anaconda II running with several boards in an error condition. Each tray as well as the four THUBs and two start detectors are displayed as rows in the table in the middle of the GUI. The overall condition of each tray is displayed with a color code, where red or yellow indicate problems. Clicking on a row shows more details about each tray, with problem values indicated in red. Balloon help displays the decoding of the register values. Clicking on the left table allows filtering the display for different error conditions, as well as selecting only parts of the TOF system.

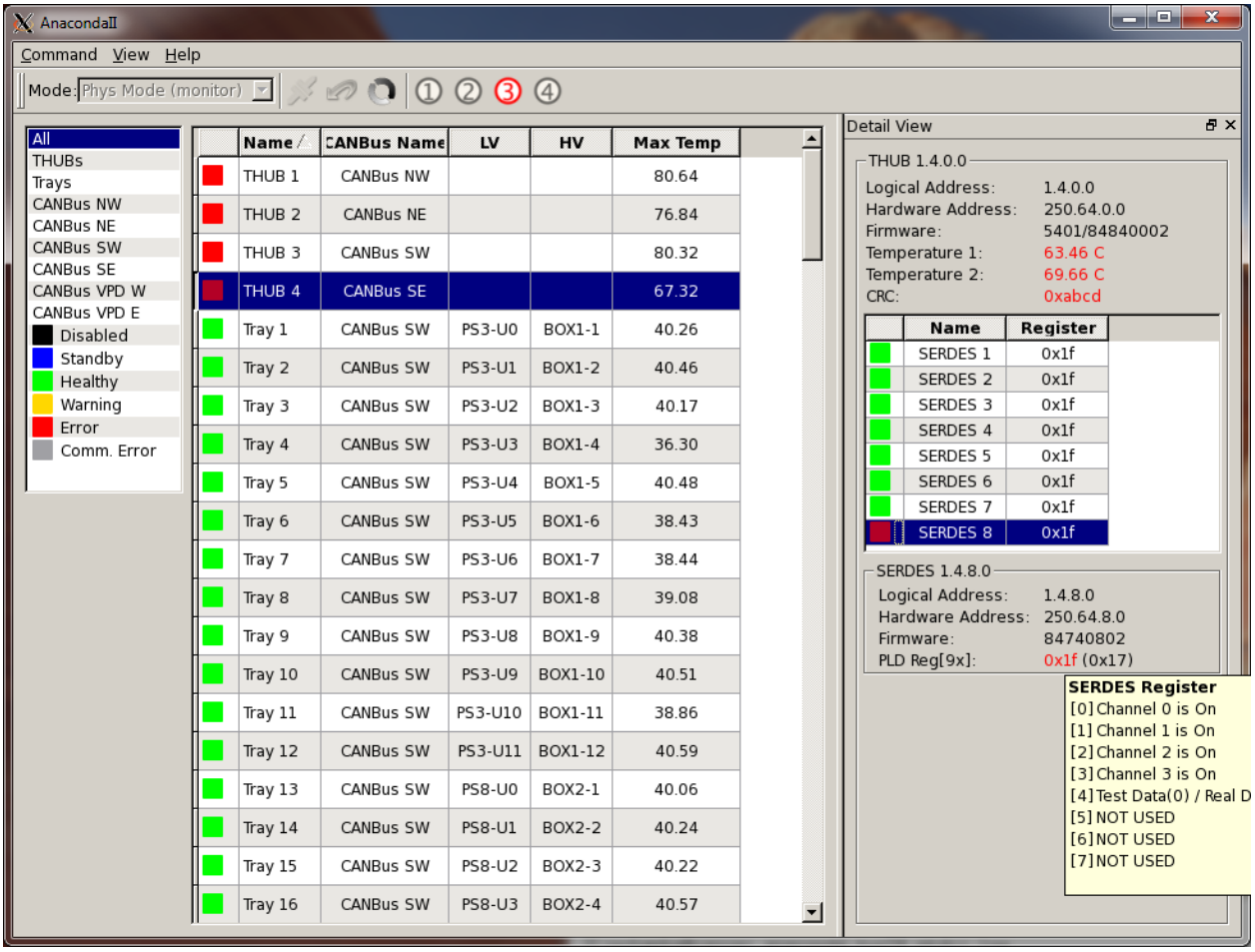

Figure 3: The graphical user interface Anaconda II. Four THUB boards are showing errors. The specific error condition of THUB 4 is shown on the right side of the GUI, with balloon help decoding of one of the registers with errors (shown in brackets is the expected value for this register)

Acknowledgment

This work was supported in part by the U.S. Department of Energy Grant DE-FG02-94ER40845.

References

[1] Robert Bosch GmbH 1991 *CAN Specification Version 2.0* (Stuttgart, Germany)

[2] PEAK-System Technik GmbH *PCAN-USB Adapter PC USB Port to High-Speed CAN User Manual* (Darmstadt, Germany)