Controller Area Network (CAN) — a Field Bus Gives Access to the Bulk of BESSY II Devices

J. Bergl, B. Kuner, R. Lange, I. Müller, R. Müller,
G. Pfeiffer, J. Rahn, H. Rüdiger

Berliner Elektronenspeicherring-Gesellschaft für Synchrotronstrahlung m.b.H.
(BESSY), Lentzeallee 100, D–14195 Berlin, FRG *

Abstract

The superior properties of the CAN fieldbus will be widely utilized for BESSY II. Literally any power converter, the PLC of the radio frequency, vacuum valves and gauges etc. will be interfaced by embedded controllers attached to a CAN segment. Furthermore it is envisaged to set up a feed forward insertion device compensation scheme that is based on a specific CAN installation. Today a unified data link layer (called SCI: Simple CAN Interface) simplifies the development of communication software between CAN chips of different manufacturers on the embedded controllers or the VME master. The usefulness of the specific subset CAN Message Specification (CMS) of the CAN Application Layer (CAL: proposed standard) protocol is under study.

Introduction

The control system at BESSY I has used the CAN field bus for controlling many power supplies for two years. The installation has proven to be robust and reliable in a very noisy environment [2]. This excellent experience has resulted in the extended use of CAN as the field bus in the BESSY II control system. DESY also uses the CAN bus for device control, so it is planned to combine the efforts of the two laboratories in this field. BESSY II is designed to use insertion devices to provide high brilliance synchrotron light. This leads to some special control system demands. One is to implement an insertion device (ID) compensation scheme that guarantees a correction of the machine orbit every 100 ms.

I. A Field-Bus-Based Control System Solution

The BESSY II control system will be distributed over the storage ring, the synchrotron and the transfer line. At the primary IO level, VMEbus computers attached to an ethernet with EPICS as the control system software will be placed at strategic places around the storage ring and the synchrotron.

A first goal is to incorporate intelligent actuators and provide a high flexibility in connecting distributed hardware to the primary IO level. Therefore a secondary IO level based on the CAN field bus will be installed. Only a small amount of equipment will be directly interfaced by VME cards.

A. Embedded Controller Concept

The main power supplies require analog signals with 16-bit resolution and low thermal drift. Therefore the DAC has to be placed as close as possible to the power supply. Each power supply at BESSY II will be equipped with an interface card that consists of the DAC, an ADC and digital IO. The interface card carries an embedded controller board based on the Intel i80386EX processor. This ensemble fits directly into a slot in the power supply. Furthermore the embedded controller provides the CAN interface using the Intel 82527 full CAN chip. This solution leads to power supplies fully controllable via the CAN bus with short distances between the interface and power supply electronics.

For software development the C programming language has been chosen. The embedded controller is fully hardware compatible with an 80x86-PC. Therefore the software can be used on both platforms with minimal modifications. A CAN interface card for 80x86-PCs using the i82527 CAN chip is also available.

B. ID Compensation

*eMail: bergl@bii.bessy.de, kuner@bii.bessy.de, lange@bessy.de, muelleri@bii.bessy.de, mueller@acc.bessy.de, pfeiffer@bii.bessy.de, rahn@acc.bessy.de, ruediger@bii.bessy.de

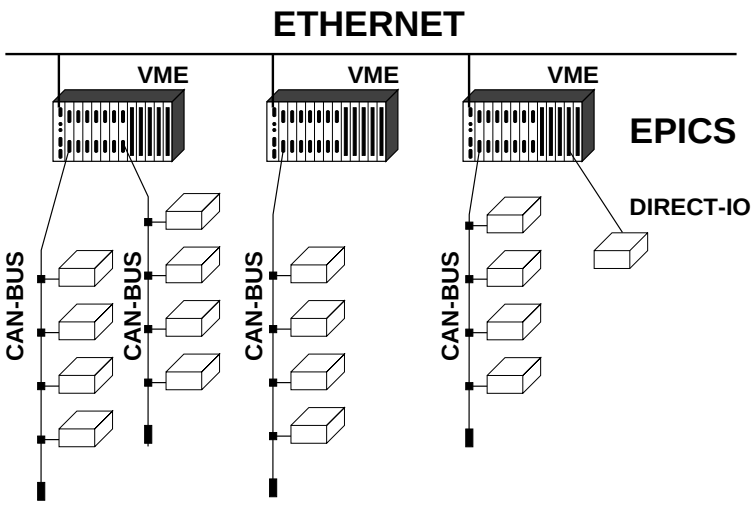

Figure. 1. IO Level Overview

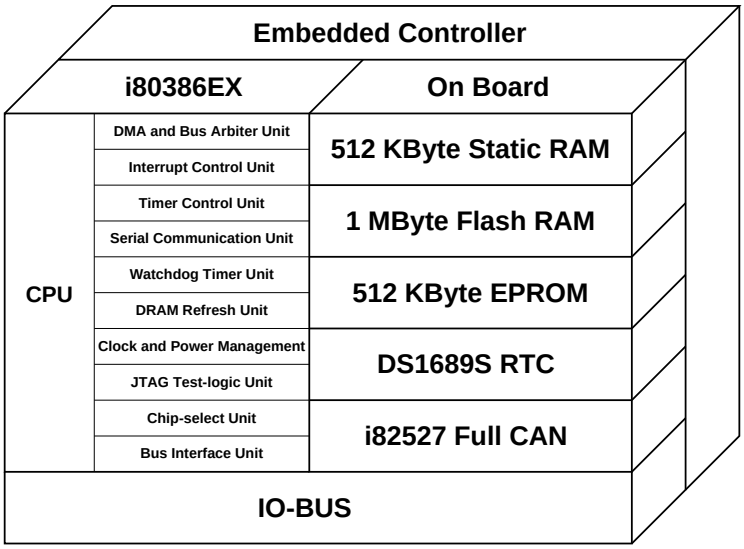

Figure. 2. Embedded Controller Schematic Diagram

To compensate the influence of the IDs on the beam in the storage ring a feed forward correction scheme has to be implemented. Every 100 ms it reads the gap information of all IDs and distributes the information to all power supplies (about 200) involved in the correction scheme. In this approach the ID compensation master takes the gap information of the IDs connected by separate 100 Kbit CAN segments. An additional 100 Kbit CAN segment will be used to distribute the gap information to the IOCs controlling the power supplies involved.

The chosen CAN interfaces at the VME level control two CAN segments and are capable of mapping automatically a specific CAN read object to a CAN write object on the other bus segment. Using that feature the CAN interface forwards objects from the 100 Kbit CAN segments to the 1 Mbit segments without wasting CPU resources on the IOCs. The embedded controller for each power supply holds a experimentally-determined correction table to alter the setpoint with respect to the GAP information. The deterministic message transfer time combined with the multicast capability of the CAN bus allows an implementation where the time needed for distributing the gap information does not depend on the number of power supplies involved.

II. CAN Communication

Within the BESSY II control system the CAN field bus will serve a variety of tasks: In addition to the ‘normal’ device control it will interface underlying intelligent subsystems (like measurement PCs, PLCs), can be used for downloading configuration data and possibly object code to the embedded controllers, carries the ID compensation data and can be

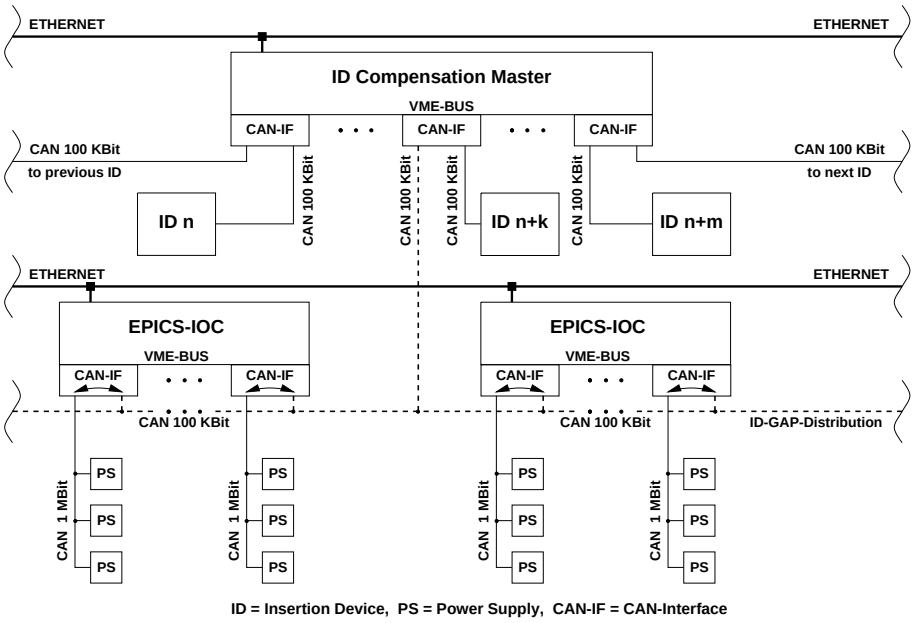

Figure. 3. ID Compensation Hardware

used for synchronization of IOC tasks. These different communication tasks require different protocols to be used in parallel on one CAN segment.

The CAN interface has to be implemented on three platforms: The IOC (Motorola MVME with VxWorks/EPICS and commercial 68000-based CAN cards), the embedded controller (i80386EX without OS with a single on-board CAN chip – see fig. 2) and a PC used for measurement and debugging (80x86 using DOS and an AT bus CAN card).

A. SCI - a Protocol Independent Data Link

SCI, the *Simple CAN Interface*, is the specification of a library that embodies a standard for accessing CAN interface cards independent of the actual hardware. Programs that use this library to access the CAN field bus will be portable between different CAN cards and even different operating systems. SCI is intentionally kept simple and it largely represents the data-link layer of the OSI model.

SCI is prepared to handle several CAN ports which are distinguished by port numbers. The ports may be realized on one or more interface cards, not necessarily from the same vendor. Different interface cards may have to be accessed through different drivers, but these differences are hidden by SCI.

SCI is capable of multithreading. In order to facilitate this function, each thread that opens the library is provided with a unique pointer. This pointer, which is a parameter for all SCI functions, is used to distinguish different threads. The properties of a COB[1]– identifier, data length, timeout and type, where type can be one of read, write, remote-read and remote-write - - are defined when it is initialized. SCI then returns a pointer to an internal structure that is allocated and initialized per object. All other functions use this sci_object pointer as a handle to a certain object. Read and write functions exist in order to transfer data to and from CAN objects.

Reading can be done in three basically different ways: *Immediate read* gets whatever data is stored on the CAN hardware for a certain object. *Read with timeout* gets new data or – if there is no new data – waits for the COB to arrive. *Install a callback function* that is called when new data arrives for an object is the asynchronous mode to retrieve data. On multitasking operating systems this callback is called from within a signal handler.

[1] *Communication Object*, message with a unique tag.

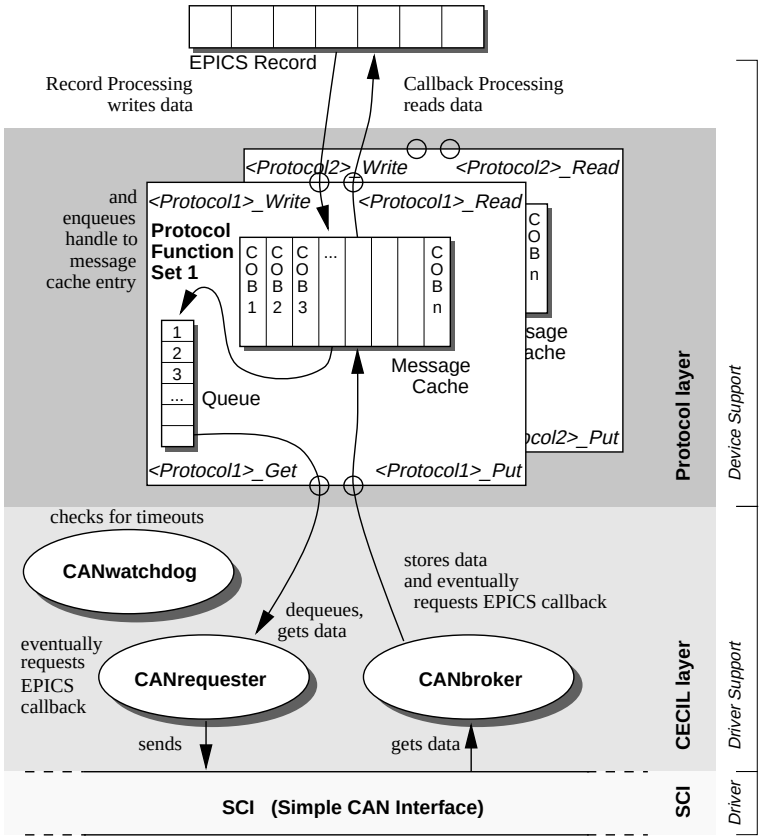

Figure. 4. CAN Interface Structure

B. CAN Integration into EPICS

B.1 Concept

The requirements of portability and reusability are fulfilled best by a layered interface structure. This allows the parallel use of different protocols through well- defined interfaces into the protocol layer and collects the platform dependencies in another layer. Thus the support for a protocol can be implemented fully independent from hardware and OS properties. Fig. 4 shows the task level structure of the CAN interface in front of its layer structure.

Above the low level CAN library SCI is the OS/hardware dependent CECIL (*Common EPICS CAN Interface Layer*). The tasks and functions in this layer set up the data transfer between the CAN segments (through SCI) and a COB oriented cache structure which is an interface to the next layer handling the different protocols. This layer consists of a small set of functions for each protocol to be run over the CAN bus. These functions include initialization, reading and writing data in protocol specific format (the upper interface) as well as reading and writing data in SCI conformal format (the lower interface).

EPICS device support writes and reads data to/from the cache using the upper interface functions of the appropriate Protocol Function Set. For output records the write request is fed into a queue. The CANrequester dequeues it, gets the data through the lower interface of the protocol layer and calls SCI to send the message. For asynchronous record processing an EPICS callback may be requested. On an incoming message SCI calls the CANbroker task. The CANbroker gets the COB, stores it through the appropriate protocol function and eventually requests an EPICS callback, which gets the data out of the protocol layer and moves it to the EPICS record. To avoid any mutual influences between field bus links there is one three task ensemble for each CAN segment connected to the VME.

B.2 Protocols

It is intended first to implement the following three protocols:

• *flatCAN*, a very basic protocol for reading and writing raw messages over the CAN bus, will be used to interface commercial CAN nodes that do not support our message transfer standard.
• *lowCAL* is a subset of the CiA[2] CAL (*CAN Application Layer*) [1]. CAL is a huge protocol suite defining an open CAN environment, including services for message transfer, network management, on-line address distribution as well as remote configuration of certain layer parameters. Our lightweight lowCAL subset defines only the message specification as well as the encoding rules for the variable types used in BESSY II device control. This will be the protocol used for communication with the embedded controllers.
• *CANal* (*CAN arbitrary length*) specifies two protocol variants for sending data of arbitrary length such as tables, configuration data or object code: A multicast transfer which is not confirmed and a point-to-point transfer which is confirmed and allows partial re-transfer of data.

C. Status

SCI has been implemented on an HP9000/743 (HP-RT) and on an MVME162 (VxWorks 5.2). It is being implemented at the moment on the i80386EX Embedded Controller and the 80x86 (DOS).

The CAN–EPICS interface has been designed with the protocols, layer interfaces and data structures being defined already. We are beginning to implement it under VxWorks and in parallel to develop the embedded controller software.

References

[1] CAN in Automation International Users and Manufacturers Group e.V.: *CAN Application Layer for Industrial Applications (CAL)*, CiA e.V., Nürnberg, 1995.
[2] G. v. Egan, T. Stein, J. Rahn: *Object Oriented Device Control Using the CAN Bus*, presented at ICALEPCS '93, Berlin, 1993.

[2]CAN in Automation International Users and Manufacturers Group e.V.