

XA-C3 Supports CAN Higher Layer Protocols

Peter Hank
Systems Laboratory Hamburg

Introduction

CAN bus systems in automotive and industrial applications are increasingly using Higher Layer Protocols (HLP's). Starting with the Physical and Data Link Layer specified by the CAN protocol, additional functions are added for complex communication and system services. Typically, these functions are implemented in software resulting in considerable CPU load. To improve system performance it is necessary to implement important higher layer functions of the ISO/OSI reference model already on chip. One important communication mechanism, specified in any of the main standards of CAN-based HLP's, is the fragmentation of data blocks larger than eight bytes. Long data packages are transferred as multiple CAN data frames. Today's CAN controllers generate high processor load when operating with CAN Transport Layer functions.

The XA-C3 is the first CAN Microcontroller featuring a powerful combination of FullCAN and PeliCAN functionality together with a unique on-chip Transport Layer Co-processor supporting Higher Layer Protocols such as DeviceNet, CANopen and OSEK.

XA-C3 Architecture

The XA-C3 is a member of the Philips XA (eXtended Architecture) family of high performance, 16-bit, single-chip micro-controllers. The XA's speed and memory addressing capabilities are enabling designers to achieve advanced performance compared to other 16-bit microcontrollers. This architecture is tailored for multi-tasking software and real time operating systems (RTOS) including high-level language support.

Functional Block Diagram

The XA-C3 comes in a 44-pin package and is pin-compatible with the XA-G3. As shown in figure 1, this new CAN Microcontroller includes a:

- Powerful CAN2.0B controller (FullCAN)
- Unique 'Transport Layer Co-processor' (TLC), supporting CAN Higher Layer Protocols
- 512 byte on-chip XRAM message buffer (extendable to 8kbyte off-chip)
- Message Handler with DMA engine
- SPI, UART and three 16-bit timers
- 32kbyte of on-chip program memory and 1kbyte of Data RAM

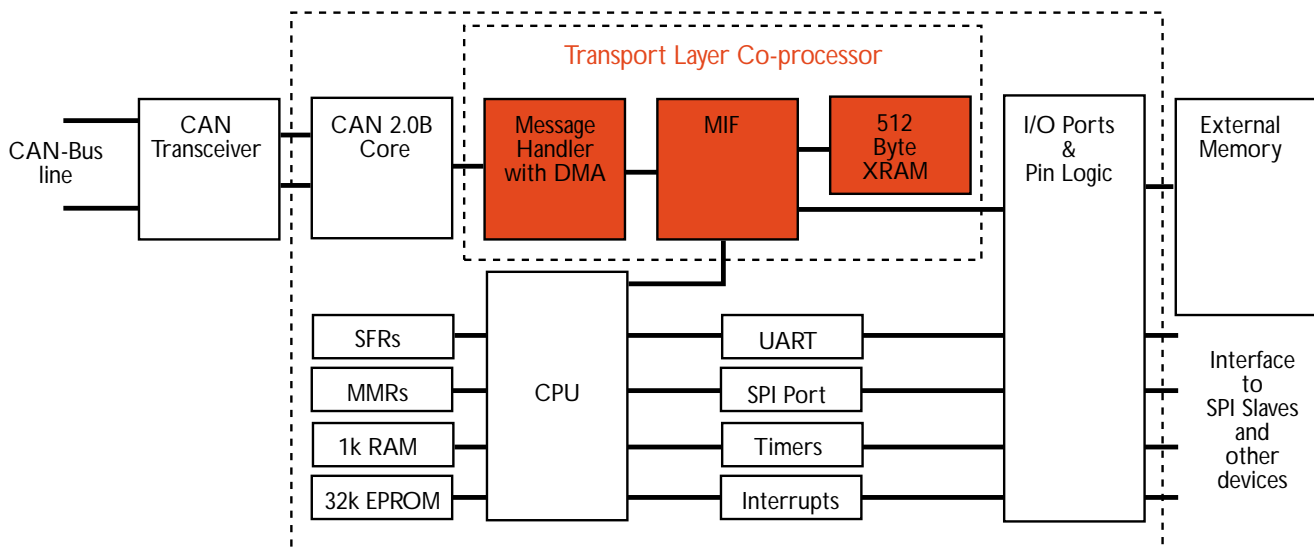


Figure 1. Block Diagram of the XA-C3 CAN Microcontroller

CAN Message Handling

The functionality of the CAN 2.0B compatible controller is a superset of today's FullCAN controllers. Message handling is accomplished by using up to 32 independent Message Objects.

Each object is associated with a set of Memory Mapped Registers (MMR) dedicated to that object. On setup, the user can define transmit or receive objects either for 11-bit or 29-bit Message Identifiers. In addition, identifier mask registers are included to allow reception of groups of messages, see also Figure 2.

Associated with each Message Object is a free configurable Message Buffer, located in the XRAM space, either on-chip or off-chip. This Message Buffer is intended to store all message data bytes belonging to a certain Message ID. The Message Buffer Base Address is defined as a pointer to the first buffer location. A Buffer Size register defines the buffer length, which can vary between 2 and 256 bytes.

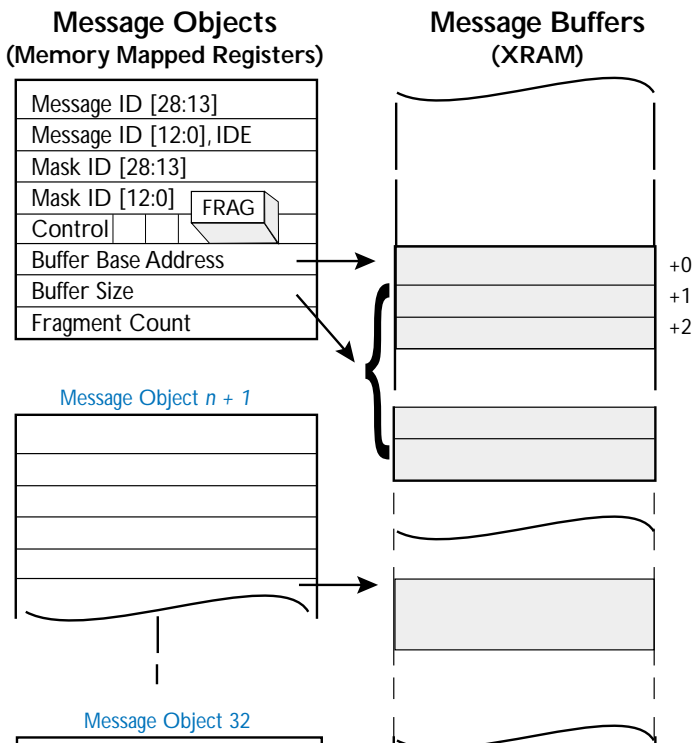


Figure 2. Message Storage Scheme

CAN Reception

When receiving messages, the standard case of operation is message storage in single frame buffers. Whenever a new CAN frame matches one of the programmed message ID's, the contents of the according message buffer is automatically updated. The user can also define multiple frame buffers for reception. In this case, data bytes of several received CAN frames belonging to the same identifier are stored sequentially. This is achieved by setting FRAG=1 and choosing a Buffer Size greater than 8.

CAN Transmission

Whenever certain objects are enabled for transmission, they will participate in the so-called "pre-arbitration" process. This process will determine which transmit message will be sent first. To achieve optimized real-time performance, the user can choose pre-arbitration, based on Object Number or CAN Identifier (priority). If arbitration based on Object Number is selected, transmission is started with the lowest object number first. If arbitration of priority is selected, transmission is started with the lowest CAN Message ID (figure 3).

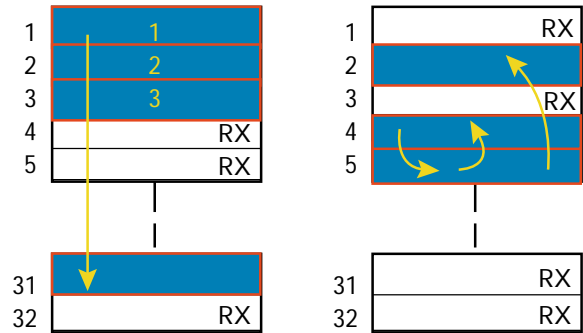


Figure 3. Pre-Arbitration

In addition to this powerful set of functions, several enhanced PeliCAN features, from the SJA1000 Stand-alone CAN Controller, have been included as well.

- * Self Test Mode (supports complete CAN Node check)
- * Listen Only Mode (Autobaud Detection)
- * Readable Error Counters and Error Code Capture functionality (System optimization and diagnosis)

Due to this, the CAN controller on the XA-C3 combines all the benefits of well-known FullCAN and PeliCAN implementations.

Higher Layer Protocol Support

Long data packages are transferred as multiple CAN data frames. The worst case situation for a conventional FullCAN or BasicCAN Controller occurs if all fragments are received consecutively at high busload.

The CPU is interrupted for every subsequent fragment, as long as the “end of message” (last fragment) has not been detected by the software. Servicing the interrupt results in a significant constant CPU load during the whole download process. This is a real bottleneck for the whole system processing resources, and it gets even worse when multiple, interleaved messages are received simultaneously.

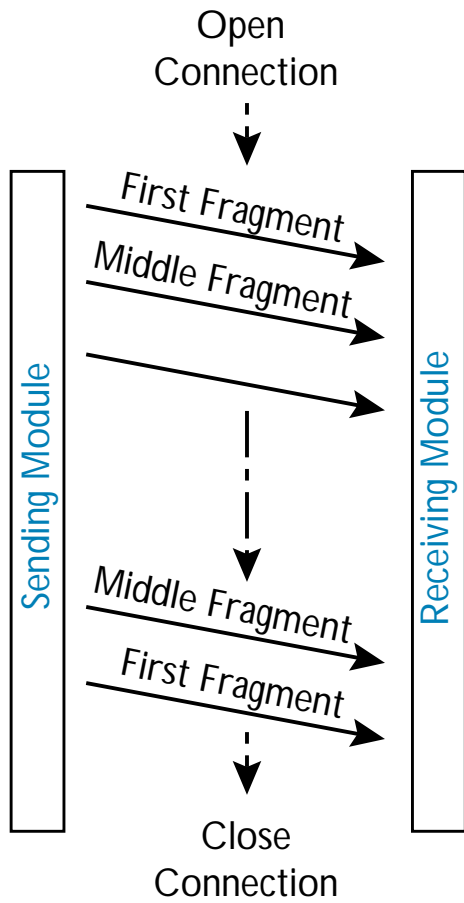


Figure 4. General Download Flow

Transport Layer Co-processor (TLC)

To reduce processor load, caused by servicing the received CAN data frame fragments of long messages, the XA-C3 manages Communication at the Transport Layer of Higher Layer Protocols in hardware (figure 5). Long messages are reassembled automatically and the CPU is interrupted only when a particular message is complete. The whole process of receiving, comparing and storing fragmented messages is performed by the Transport Layer Co-processor, which is a unique DMA driven ‘message handling’ engine. With these CAN Controller properties, the interrupt overhead can be reduced significantly, because the CPU is only interrupted at the end of a certain message. This allows the 16-bit CPU to focus more on main application tasks and less on housekeeping of the Transport Layer Protocol.

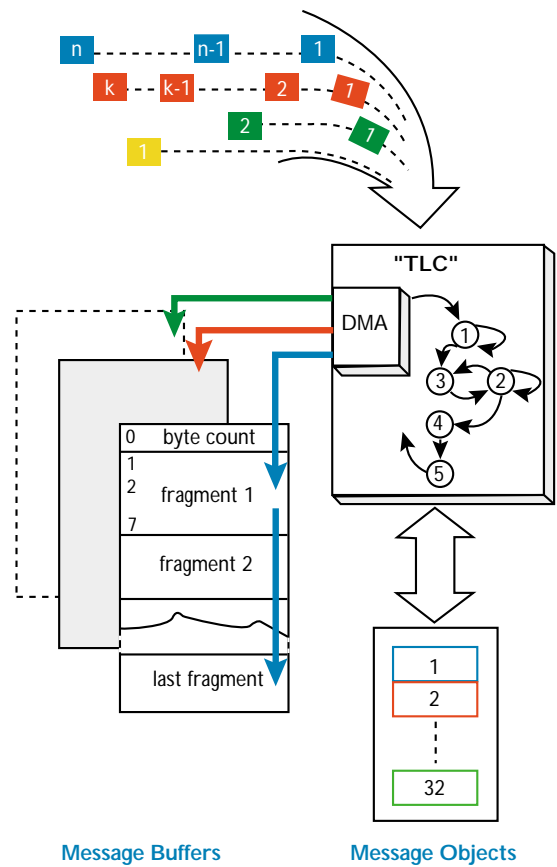


Figure 5. Fragmented Message Reception

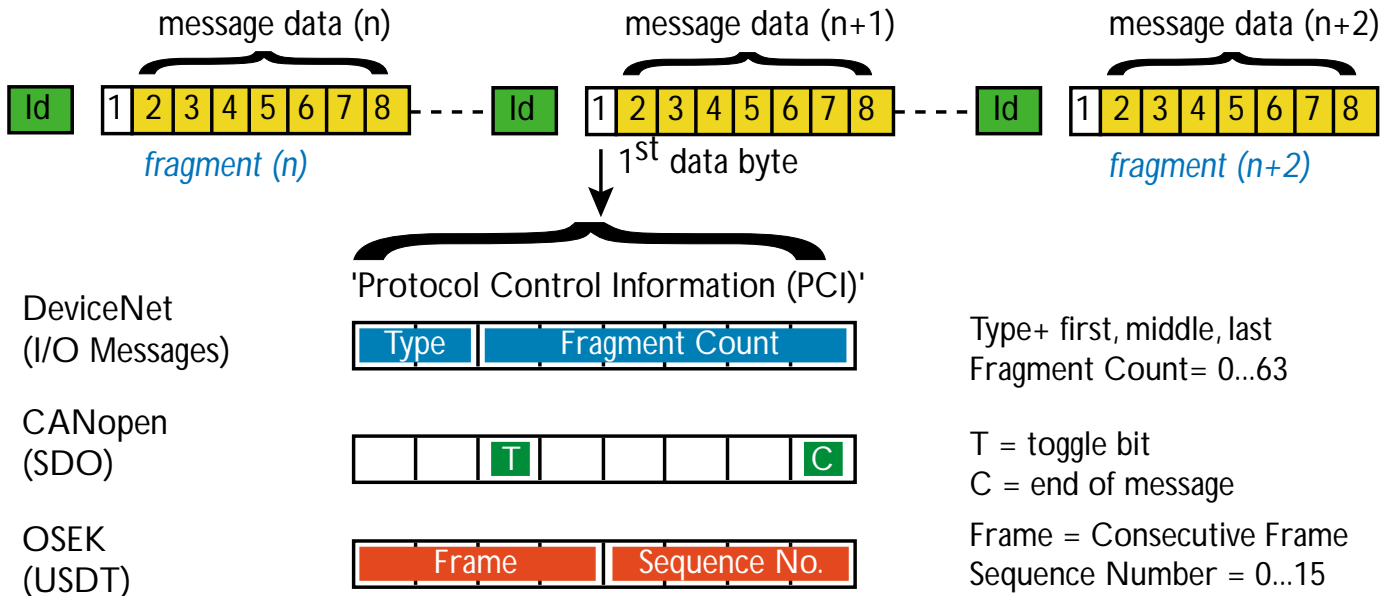


Figure 6. Protocol Control Information Byte

For the XA-C3 one of three Higher Layer Protocols (DeviceNet, CANopen, OSEK) can be selected. Whatever HLP has been chosen, it applies to all receive message objects which are enabled for fragmented message reception.

Fragmented message reception supported by the XA-C3 is based on eight bytes of data per fragment. In this context, the first data byte has an important control function for the transfer (see figure 6). In some HLPs, this byte is called Protocol Control Information Byte (PCI). It specifies start/end of a fragmented transmission and includes a fragment count, which is incremented by one for every subsequent fragment. According to the value of the PCI byte, data will be automatically stored into the corresponding buffer area via DMA. This byte is always stripped off and gets never stored in the message buffer. Data bytes from successive fragments are stored in the message order received. When the end of message has been decoded, and the final frame's data bytes have been received, the byte count is written into address 0 of the buffer (Buffer Base Address). The byte count represents the total number of data bytes, which have been stored in the buffer. The end of a certain message will be set only after the special encoded 'Last Frame' has been received.

DeviceNet

The XA-C3 provides hardware support for DeviceNet I/O Messages. They are transmitted in an unacknowledged fashion. In other words download of fragments can take place without acknowledge frames from the receiving node. As shown in figure 6 the PCI byte contains a fragment type field indicating whether this is the first, middle or last fragment of a message. The fragment count field marks each separate fragment such that the receiver can determine, whether or not a fragment has been missed.

CANopen

CANopen SDO's (Service Data Objects) have a toggle bit that is changing its value with every subsequent fragment. In addition, a single bit indicates the end of a message. Each download segment request is confirmed by the XA-C3 with a download segment response. This is organized in such a way, that a pre-defined message object is taken as a response frame. Upon valid reception of a download segment, the TLC starts transmission of the response frame without any software intervention.

OSEK / ISO 15765 (Diagnostics on CAN)

The USDT (Unacknowledged Segmented Data Transfer) protocol used for OSEK and Diagnostics on CAN does not define an end-of-frame indication as known from DeviceNet and CANopen. However, a message length indicator is transmitted within the first frame. Due to this, the user software can calculate the size and the start address for the message buffer ahead of time. As soon as segmented data are received as consecutive frames, the XA-C3 can fully profit from the TLC functionality. All data bytes are stored in the sequential order received. Whenever a message is longer than the specified message buffer size, an RX Buffer Full interrupt can be enabled. This type of interrupt allows, e.g., to enable a new buffer space for the remaining message bytes by re-defining the object's Buffer Base Address. This feature is important because USDT specifies up to 4095 user data bytes per message.

However, the TLC is not only performing a simple data transfer to the message buffer but also managing a complete data integrity check. Any kind of errors and buffer overflow conditions during fragmented message reception are detected and signaled to the user. In case of, e.g., detecting an 'out of sequence' fragment, the automated data transfer to the message buffer is aborted. In addition, a fragmentation error interrupt can be enabled for the CPU. In this case, a re-initialization of the download process has to be done.

Application Example

The behavior of the TLC will become even more transparent with the following configuration and processing example for DeviceNet (figure 7). Whenever the general initialization of the XA-C3 has been finished, one or more message objects can be enabled for fragmented message reception.

First, the 'Higher Layer Protocol' has to be selected within the Global Control Byte. Next the object's Message ID must be specified in the MID registers. Note that masking of message IDs is not allowed for fragmented messages. The Buffer Location Register (BLR) includes the start address within the message buffer. In this example, a message with 26 data bytes (4 fragments) shall be received. Therefore, a buffer size of 32 has been chosen. Finally, the 'FRAG' bit is set '1' (message data is stored sequentially) and the object is enabled for reception. From now on, the TLC takes care of the whole reception process.

As soon as the last fragment of the message has been stored in the message buffer, the TLC generates a message complete interrupt for the CPU. On receipt of this interrupt the software will reset the message complete status and process the data.

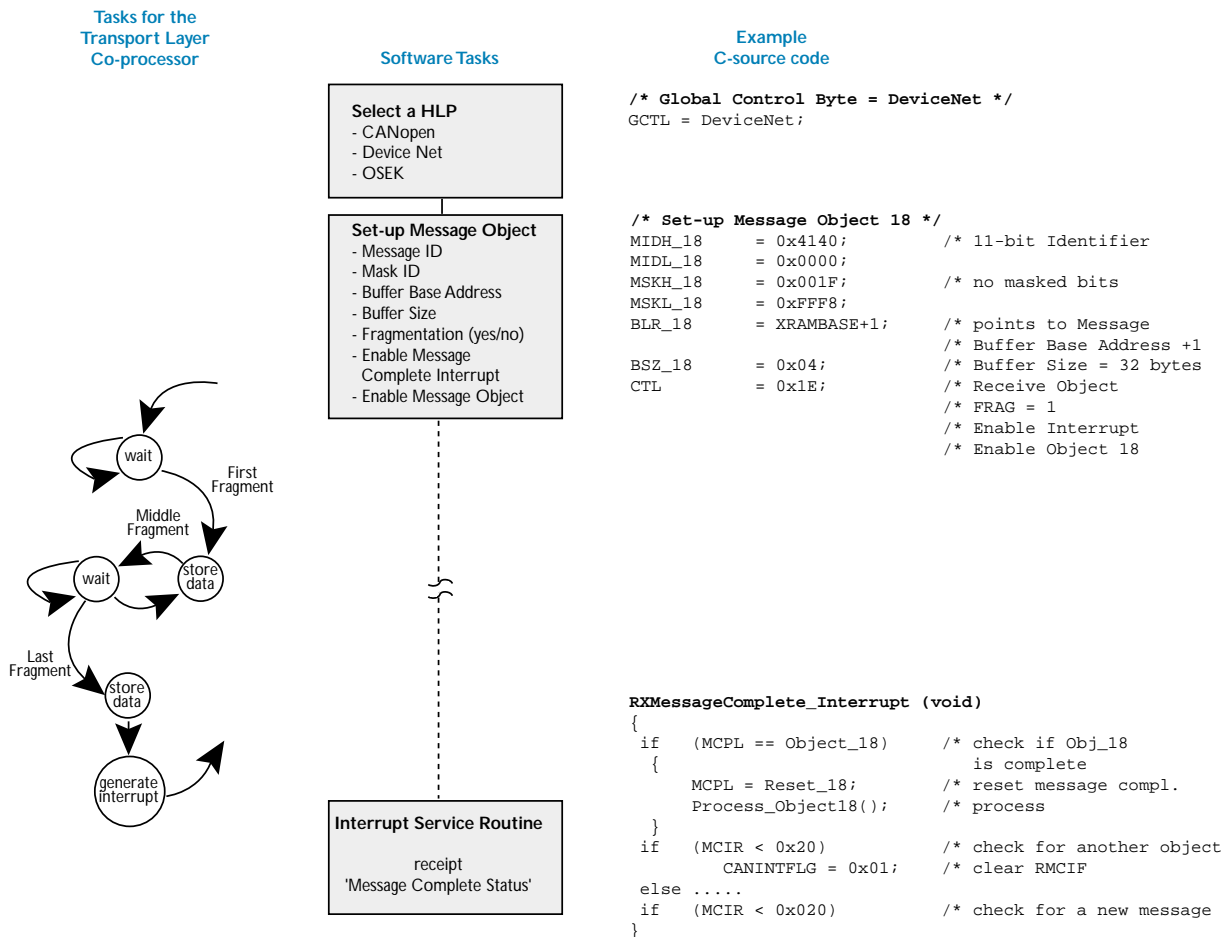


Figure 7. Simplified Processing Flow for Fragmented Message Reception

For more information, contact your Philips Semiconductors distributor or www.PhilipsMCU.com

North America

Tel: 1 800 234-7381
Internet: (in English)
www.PhilipsMCU.com

Europe

Fax: +31 79 3685126

Asia

Fax: 886 2 2134-2941

Japan/Korea

Fax: +81-3-3740-5057
Internet (in Japanese):
www.philips.co.jp/semicon/

© Philips Electronics N.V. 1999

All rights reserved. Reproduction in whole or in part is prohibited without the prior written consent of the copyright owner. The information presented in this document does not form part of any quotation or contract, is believed to be accurate and reliable and may be changed without notice. No liability will be accepted by the publisher for any consequence of its use. Publication thereof does not convey nor imply any license under patent - or industrial or intellectual property rights.

Printed in the USA 601653/5K/FP/6pp/0100

9397-750-06772

Let's make things better.



PHILIPS