# CAN Bus Interface (R3.0)

March 23, 1998 | Product Specification

## SICAN Microelectronics Corp.


400 Oyster Point Blvd., Suite 512
South San Francisco, CA 94080 USA
Phone: +1 650-871-1494
Fax: +1 650-871-1504
E-mail: info@sican-micro.com
URL: www.sican-micro.com

SICAN GmbH
Garbsener Landstrasse 10
Hannover
D-30419, Germany
Phone: +49 511-277-1601
Fax: +49 511-277-2600
E-mail: info@sican.de
URL: www.sican.de


## Features

- Supports CAN 2.0A, 2.0B passive/active protocol functions
- Data transfer rate of up to 1 Mbit per second
- 10 Mhz cycle frequency
- Supports CAN bus line arbitration
- Sampling of receive data
- Supports all required CAN functions
  - Error handling
  - Stuff bit generation
  - CRC generation
  - Multiple sample points
  - Acknowledge generation
- Individual acceptance filtering

## AllianceCORE™ Facts

| Core Specifics[1] | | |
|---|---|---|
| Device Family | XC4000E | |
| CLBs Used:<br>CAN Core Alone<br>CAN + User Module | <br>298<br>~520[1] | |
| IOBs Used:<br>CAN Core Alone<br>Can + User Module | <br>71[2]<br>43[3] | |
| System Clock $f_{max}$ | 10 MHz | |
| Device Features Used | 3-State Buffers<br>Global Clocks | |
| **Supported Devices/Resources Remaining[1]** | | |
| CAN Core Alone | **I/O** | **CLBs** |
| XC4013E | 121 | 278 |
| XC4020E | 153 | 486 |
| **Provided with Core** | | |
| Documentation | CAN Core module documentation<br>XC4000E data sheet | |
| Design File Formats | .xnf netlist | |
| Constraint Files | .cst | |
| Verification Tool | None | |
| Schematic Symbols | None | |
| Evaluation Model | None | |
| Reference designs & application notes | User interface module in Verilog source code format | |
| Additional Items | None | |
| **Design Tool Requirements** | | |
| Xilinx Core Tools | Alliance 1.3 | |
| Entry/Verification Tool | FPGA Compiler | |
| **Support** | | |
| Support provided by SICAN Microelectronics | | |

Notes:
1. CLB count depends upon user configuration. Total CLB count is ~520 for User Module and CAN core together.
2. Assuming all core signals are routed off-chip.
3. I/O count provided is total for both CAN Core and User Module with a 16-bit data interface, assuming all core signals are routed off-chip. See Pinout section.

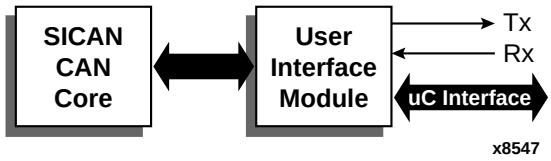


**Figure 1: Use of CAN Core and User Module**

- Interrupt outputs can be generated for the following events:
  - CAN module becomes error passive
  - CAN module status bus off
  - CAN error counters exceeds a preset, programmable level
  - Telegram sent successfully
  - Telegram received successfully
  - Receive memory overflow
- Includes separate, customizable user interface module shipped in Verilog source code format
- Requires external CAN bus transceivers

## Applications

The core can be used in automotive network and home automation systems. Typical applications range from simple sensor/actuator systems to complex CAN controllers with microcontroller cores used for data pre-processing.

## General Description

The SICAN CAN core provides all CAN specification 2.0B protocol functions including Extended CAN (2.0B active) functionality. The core can support acceptance filtering (implemented in external user logic) according to Basic CAN (mask and code registers are similar to the Philips 82C200) or according to Full CAN (similar to the Intel 82527). It is also possible to combine the two processes.

The core incorporates all features required from the CAN specification including error handling capabilities, stuff bit generation, CRC, and multiple sample points. It has a universal interface to connect to the receive and transmit buffers, allowing the module to be optimized for specific applications.

An optional CAN User Module in Verilog source code format is included with the core. The User module provides an example interface between all signals of the CAN Core and the outside world, including external CAN transceiver and microcontroller. It consumes approximately 220 CLBs depending upon how it is configured. Figure 1 shows how the CAN core and User Module are used together.

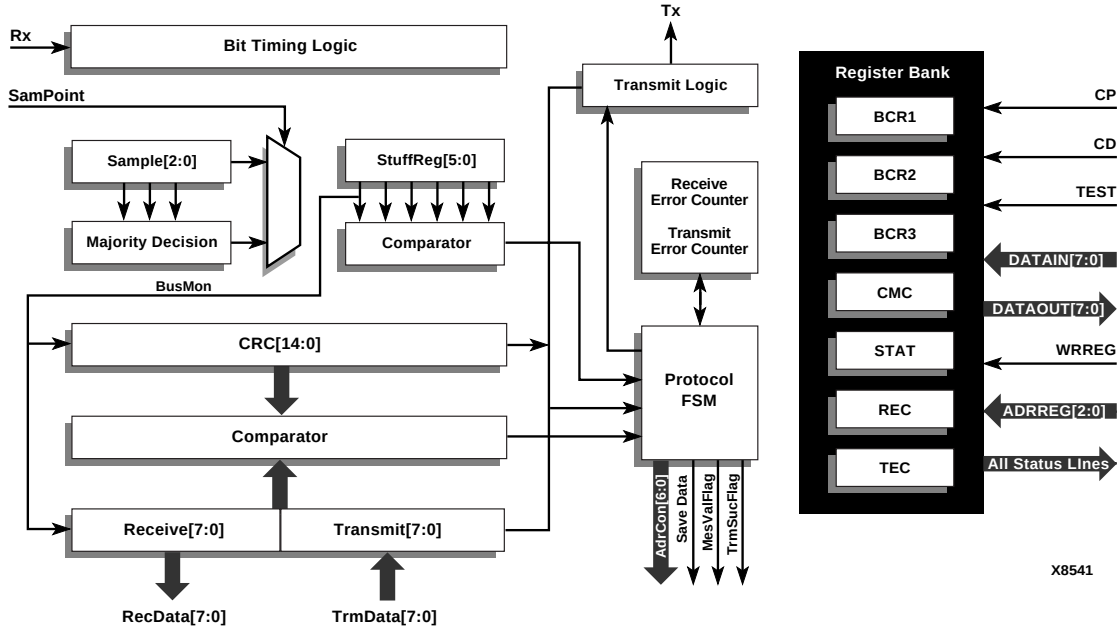


**Figure 2: CAN Core Module**

# Functional Description

The CAN core is divided into modules as shown in the block diagram of Figure 2. The core module has no receive buffer memory. Basic operation is described below.

## Rx and Tx – CAN Bus Interface

The CAN core uses a simple two-wire (Rx and Tx) connection. Both signals operate at TTL levels that comply with ISO/DIS 11898 so it can connect to standard CAN bus transceivers (e.g. Philips PCA 82C250, Bosch CF150 or Siliconix SI 9200) or to a modified RS-485 interface.

## Receive and Transmit Error Counters

The CAN protocol contains mechanisms for automatic fault location and for switching-off defective nodes. This is implemented through two counters - a Receive Error Counter (REC) and a Transmit Error Counter (TEC). These are incremented and decremented according to CAN specification rules.

## CRC and Comparator Blocks

These blocks provide error protection. Each telegram is provided with a 15-bit-long CRC code, generated from fields (start of frame, arbitration field, control field, data field) in the preceding telegram. When receiving a telegram, a new CRC is generated from the received data and compared with the original CRC in the telegram. Any variation generates an error telegram.

## Transmit Logic – Bit Stream Processor/ Internal Interface

User data is input to the core over an 8-bit data bus. Next, it is converted into a serial bit stream. After five bits of the same polarity, a stuff bit of opposite polarity is inserted to force the edges required for resynchronization. Stuff bits are filtered from the received bit stream and the coded data is transmitted to the user interface.

## Bit Timing Logic, Sample and Majority Decision Blocks

All controllers on a CAN bus must have the same data rate and bit length. Bit length is determined by the parameters TSEG1, TSEG2 and BRP. These parameters are used to adjust the data rate when individual controllers have different clock frequencies.

The Bit Timing Logic block modifies the parameters to insure proper timing. It is then possible to perform several samplings of the bus line at the sample point. The level determined by the CAN bus corresponds to the result from a majority decision of three sample values.

## Acceptance Filtering Options

The core supports three options for acceptance filtering. NOTE: Acceptance filtering is not included in the core module. The filtering has to be implemented in an application specific interface module that meets the customer's requirements.

- *Full CAN* - where one or several identifiers can be indicated explicitly. Each identifier has a memory, where the telegram and the corresponding identifier are stored upon successful reception. The number of telegrams to be received depends on the quantity of storage cells.
- *Basic CAN* - where the acceptance filter is described by the contents of two registers. The Acceptance Mask Register (AMR) contains a value of 1 for each bit regarded as a don't-care. The set value of the other bits is indicated in the Acceptance Code Register (ACR). The received identifier is combined with the data stored in ACR by an EXCLUSIVE-NOR (equivalence) and the result of this operation is combined afterwards with the contents of AMR by an OR function. The telegram is accepted only if all bits are set to 1. In the Philips PCA 82C200 CAN controller, only the upper eight bits of the identifier are considered during acceptance filtering.
- *Combination* - A combination of both Full and Basic CAN acceptance filtering.

## Interrupts

The core can support one or several interrupt outputs. NOTE: Interrupt handling is not included in the core module. It has to be implemented in an application specific interface module that meets the customer's requirements. The following events are possible as interrupt-triggering sources:

- CAN module becomes error passive
- CAN module reaches the status bus off
- Telegram sent successfully
- Telegram received successfully
- Receive memory overflow

The interrupts can be masked. The CPU can read the status register to determine which event has triggered the interrupt. The interrupts are reset by the CPU.

# Core Modifications

SICAN will customize the User Module interface to meet your specific application. Examples include integrating the required memory to suit the target application. Telegrams can be stored in a register or in a storage area structured as FIFO or DPRAM. CLB count for the interface module depend on the application. An example 16-bit interface will use approximately 220 CLBs.

**Table 1: CAN Core Signal Pinout**

| Signal Name | Signal Direction | Description |
|---|---|---|
| **Transceiver Interface Signals** | | |
| Tx | Output | CAN transceiver transmit signal |
| Rx | Input | CAN transceiver receive signal |
| **CAN-Data Interface Signals** | | |
| RecData[7:0] | Output | Received data CAN Core (user interface |
| TrmData[7:0] | Input | Transmit data user interface (CAN Core |
| SaveData | Output | Save received data. |
| AdrCan[6:0] | Output | Data select bus. |
| **Control Interface Signals** | | |
| CP | Input | Clock |
| CD | Input | Reset |
| TEST | Input | Test mode (must be connected to GND) |
| DATAIN[7:0] | Input | CPU–data user interface (CAN Core |
| DATAOUT[7:0] | Output | CPU–data CAN Core (user interface |
| WRREG | Input | Write CAN internal register |
| ADRREG[2:0] | Input | Addressbus for CAN internal registers |
| MesValFlag | Output | Received error free CAN message |
| TrmSucFlag | Output | Transmission has been carried out successfully |
| **Status Lines to Control Interface** | | |
| TxClk | Output | Transmit Clock |
| TxRqst | Output | Transmission request |
| ArbLost | Output | Module lost bus arbitration |
| AbAckFlag | Output | Current transmission aborted |
| SOF | Output | Start of frame detected |
| SamPoint | Output | Sample point |
| CanOvlFlag | Output | Bus overload telegram received |
| RecRtrBit | Output | Value of Remote frame bit in telegram |
| RecIdeBit | Output | Received telegram is an Extended frame |
| RecDlc[3:0] | Output | Data length code |
| MesErrFlag | Output | Error detected on CAN bus line |
| CanResFlag | Output | CAN module is in reset mode |
| CanSusFlag | Output | CAN module is in suspend mode |
| Receiver | Output | CAN module is receiver of a message |
| Transmitter | Output | CAN module is transmitter of a message |
| ErrPas | Output | CAN core in *Error Passive* mode |
| ErrBusOff | Output | CAN transmit error counter reaches level 256 (CAN module is in *Bus Off* mode |

**Table 2: Signal Pinout for CAN Core with Optional User Module**

| Signal Name | Signal Direction | Description |
|---|---|---|
| **Transceiver Interface Signals** | | |
| Tx | Output | CAN transceiver transmit pin |
| Rx | Input | CAN transceiver receive pin |
| **Data Interface Signals** | | |
| CP | Input | Clock |
| CD | Input | Reset (active low) |
| RDn | Input | Read Signal |
| WRn | Input | Write Signal |
| CSn | Input | Chip Select |
| ADDR[8:0] | Input | Address Lines |
| DATA[x:0] | Output | Data Lines; tri state in/out; bus width depends on user's application |
| **Control Interface Signals** | | |
| TxClk | Output | Transmit |
| SamPoint | Output | Sample Point (status signal) |
| Receiver | Output | CAN module is receiver of a message (status signal) |
| Transmitter | Output | CAN module is transmitter of a message (status signal) |
| MesValFlag | Output | Received error free CAN message |
| MesErrFlag | Output | Error detected on CAN bus line (status signal) |
| TrmSucFlag | Output | Transmission has been carried out successfully |
| CanResFlag | Output | CAN module is in reset mode |
| CanSusFlag | Output | CAN module is in suspend mode |
| ErrPas | Output | CAN error counters reached level 128 (module is in *Error Passive* mode |
| ErrBusOff | Output | CAN transmit error counter reached level 255 (CAN module is in *Bus Off* mode |

SICAN can also perform modifications to the CAN Core module itself. All modifications performed by SICAN, whether to the User Module or the CAN core are at additional cost.

## Pinout

The pinout is not fixed to any specific device I/O. By itself, the CAN core has 71 I/O. Signal names are provided in the block diagram shown in Figure 1, and described in Table 1.

When using the optional User Module, with a 16-bit wide DATA bus the I/O count for the combined design is 43. All I/Os of the CAN User Module connect to a microcontroller and external CAN transceivers. The User Module is like a shell around the CAN core module. None of the core I/O signals can be accessed from outside. Every access to the core is made via the User Module. Signals for this configuration are described in Table 2.

## Verification Methods

The CAN core module has been used successfully in Xilinx FPGAs. SICAN constructed a Xilinx-based prototyping board during to perform in-house verification.

## Recommended Design Experience

Users should be familiar with the CAN protocol as described in the CAN specification by Bosch.

## Available Support Product

An evaluation board that includes Xilinx XC4025E devices is available for purchase separately from SICAN.

## Ordering Information

The CAN Core module is delivered as a Xilinx netlist. The User Module is provided in Verilog source code format. It is named "can_user.v" and represents an example application for a simple microcontroller interface to the CAN Core module. It implements all data and control interfaces.

The CAN Core module is available for purchase directly from SICAN. Contact them for pricing and additional information.

## Related Information

### CAN Bus Licensing

The CAN Bus specification is owned by Robert Bosch GmbH. In order to use the SICAN CAN Interface Core, you must contact Bosch directly to take care of the appropriate license and/or royalty fees.

Robert Bosch GmbH
PO Box 1342
D-72703 Reutlingen
Phone: +49 7121 35 2294
Fax: +49 7121 35 1740

### CAN Bus Transceivers

The Xilinx FPGA requires an external CAN Bus compliant transceiver that is available from several sources.

Robert Bosch GmbH
(see contact info above)
Device: CF150

Philips Semiconductors
811 East Arques Avenue
P.O. Box 3409
Sunnyvale, CA 94088-3409
URL: www.semiconductors.philips.com
Device: PCA 82C250

Siliconix Incorporated
2201 Laurelwood Rd.
Santa Clara, CA 95056-0951
URL: www.siliconix.com
Device: SI9200

### International CAN Users and Manufacturers Group

The international trade association CAN in Automation (CiA) provides technical, product and marketing information regarding CAN Bus applications. For more information, contact them directly at:

CAN in Automation e.V.
E-Mail: headquarters@can-cia.de
Phone: +49-9131-69086-0
Fax: +49-9131-69086-79

### Xilinx Programmable Logic

For information on Xilinx programmable logic or development system software, contact your local Xilinx sales office, or:

Xilinx, Inc.
2100 Logic Drive
San Jose, CA 95124
Phone: +1 408-559-7778
Fax: +1 408-559-7114
URL: www.xilinx.com

For general Xilinx literature, contact:

Phone: +1 800-231-3386 (inside the US)
+1 408-879-5017 (outside the US)
E-mail: literature@xilinx.com

For AllianceCORE™ specific information, contact:

Phone: +1 408-879-5381
E-mail: alliancecore@xilinx.com
URL: www.xilinx.com/products/logicore/alliance/tblpart.htm