MiniCAN - a low-cost concept to integrate sensors and actuators into CAN networks

Martin Trautmann, Hans Strack
Technical University of Darmstadt
Institute of Solid State Circuits
Schloßgartenstraße 8, D-64289 Darmstadt, Germany
traut@iht.e-technik.th-darmstadt.de

MiniCAN is a concept that combines CAN compatibility with simplified functionality. It is optimised to attach basic components directly to a CAN bus. Those components are e.g. switches, lamps or sensors and actuators that do not require more than 6 bits of data. It is a master-slave system that uses an inframe response within the CAN data field to run a simplified protocol. Special concern was taken for security, low cost and ease of use. It may be used both for low and high speed communications, compliant to CAN specification 2.0, parts A and B.

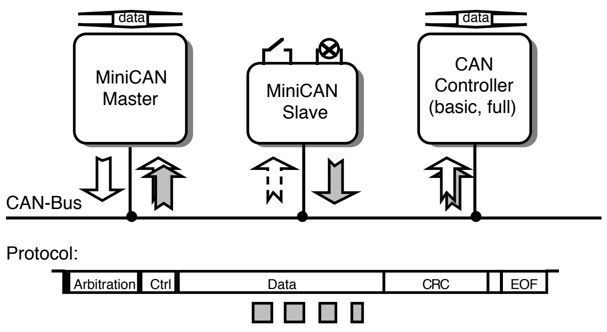

Figure 1: Coexistance of MiniCAN and CAN

Introduction

The Controller Area Network (CAN) protocol has been defined within the mid eighties, originally constructed for automotive applications. Since that time it has become an open international standard. Bus controllers have been developed by multiple manufacturers. Applications were built by numerous companies. It has also become popular for industrial automation and distributed measurement systems.

It is best suited for the fail safe transmission between several controllers. Every controller may handle incoming and outgoing short data packages, up to eight bytes, with a multitude of type identifiers. The usual CAN implementations require a rather complex set-up and programmable micro controller interface. Thus the CAN concept permits to add simple sensors or actuators, but only at comparably high costs. It is a common standard, but the full functionality is not required for binary components. Various attempts have been made to tackle this problem.

MiniCAN (figure 1) is a new concept. It shall handle:

- High Number of Bus Nodes

  While the usual CAN implementation collects data from various sensors within one bus node, this is not suited best to connect single, widely spread sensors and actuators. CAN provides a range of more than 2000 identifiers (11 bits), extended through the Specification 2.0B by yet another 18 possible bits. The major handicap is due to physical limitations of the transceiver components. First controllers were limited to a number of 32 for high speed operation. Newer transceivers guarantee more than 100 nodes [PCA82C250], depending on the physical bus implementation, transfer speeds and transceiver capabilities. Thus the limitation on MiniCAN bus nodes shall not be smaller than 100.

- Short Data Packages

  Binary sensors and actuators, such as a single switch or lamp, which will be connected to the bus require a single bit of information only. Other typical applications have shown that the exchange of max. six bits per sensor or actuator is sufficient. A typical example is a switch with four switching positions, two status displays and a variable background illumination.

- Low and High Speed Operation

  The concept shall be applicable both for low speed operation, specified up to 125 kbit/s, and high speed, up to 1 Mbit/s. Other physical implementations will be possible as well.

- Low Hardware Demands

  The hardware requirements shall be minimised. The internal circuitry may be reduced by less complex behaviour, e.g. by a simplified protocol or a reduced functionality. External circuits may be omitted by a single chip realisation of integrated oscillator, transceiver and interface logic.

- Low Set-up Requirements

  Expense for programming and hardware configuration shall be reduced as far as possible.

- Compliant to CAN specifications
  The solution will support the current specifications, according to 2.0 part A and B.

Protocol

Those requirements could not be fulfilled by the original multi-master concept of CAN. Therefore a master-slave system was invented. All usual processing may be done by the master, while a simplified protocol is used for the communication between master and slave.

The total frame is similar to an ordinary CAN Data Frame. The master works similar to other CAN nodes. He manages bus monitoring, arbitration or error management. When he managed to gain bus access, he may start the master-slave communication. This takes place only within the CAN data field while no other CAN node has bus access. The slave returns its data within slots provided by the master. As basic element the MiniCAN protocol has a group of 3 bit blocks, framed by 1-0 sequences as described later on.The basic protocol of a MiniCAN remote frame is shown in figure 2:

Every slave has to monitor the bus for a valid master identifier. The actual realisation permits 16 different master identifiers (ID) by coding the CAN identifier as:

ID10	ID9	ID8	ID7	ID6	ID5	ID4	ID3	ID2	ID1	ID0
1	0	1	0	m_3	m_2	m_1	1	0	m_0	1

Any differing signal, either within the identifier or compared to other pre-defined bits, will cause a form error and turn the slave passive. For example the control bit of an extended CAN frame will be handled this way.

Since the master-slave communication takes place within the data field, the CAN Remote Transmission Request bit (RTR) is "0", The control bits which include the data field length were defined as "000100" for a four bytes data field or as "000111" for seven bytes respectively.

The MiniCAN protocol within the CAN data field resembles a conventional CAN transmission. First, a 5 bit slave identifier is sent. 32 different slaves may get addressed this way. This type of node based addresses is most appropriate for this type of application, compared to the message based coding that is supported by CAN.

A single MiniCAN Remote Transmission Request bit (RTR) determines, whether the master will request (RTR = "1") or send (RTR = "0") data to the slaves. As specified, the slave may return or receive six bits of data respectively.

In both cases the slave will send a computed checksum to the master. The polynomial used for the Cyclic Redundancy Check (CRC) is $x^5 + x^2 + 1$. It is computed from the Start Of Frame (SOF) bit to the last data bit. It is sent inverted in order to differ the possible combinations from the passive state. Obviously this CRC itself is less secure than its 15-bit CAN equivalent. Concerning that it only has to insure the integrity of six data bits and that it has frequent supplemental fixed form bits, it promises sufficient and high security.

The master will compare the MiniCAN CRC with the own checksum and acknowledge proper reception by a dominant acknowledge bit returned to the single slave. At this stage the transmission between master and slave is complete. The master will terminate the CAN protocol by the transmission of the usual 15 bit CRC, according acknowledgement processing and End Of Frame (EOF).

Other CAN nodes will not observe a difference to an ordinary CAN frame. They may not call the slaves on their own, but they may read and interpret the MiniCAN frame directly.

A full MiniCAN data or remote frame will take at least 76 bits, including 7 bits for an End Of Frame delimiter. Due to the 1-0 form bits the length up to the end of the MiniCAN frame is fixed, while bit stuffing may occur within the CRC part and thus in-

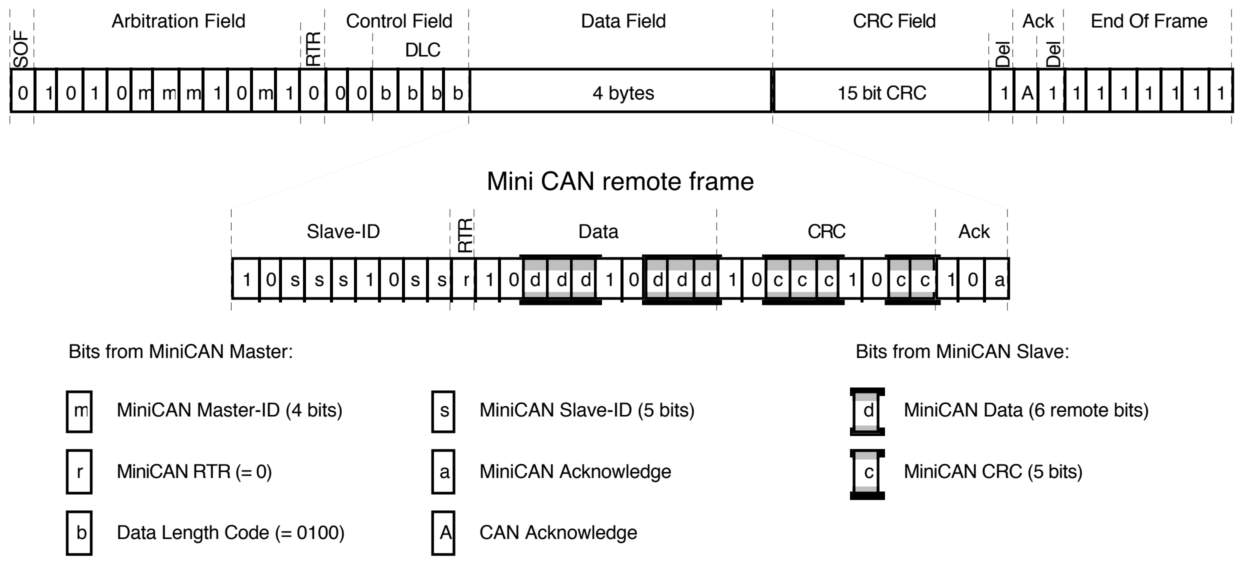

Figure 2: Remote Frame

crease the total length with some more bits. A minimum of three further bits is required as Interframe space. As long as the slaves show a rare need of communication, this will only add a minor overhead to net traffic. But while the master has to perform a cyclic state request of its slaves, this may cause frequent, redundant requests. For better net performance another type of inquiry frame has been invented (figure 3). This frame allows the master to detect by a single transmission whether there had been a change at any of its slaves.

For each slave there is a reserved slot within the seven bytes long data field (Control Field = 000111). Each slave which observed a modification on its input port will overwrite its slot by a dominant bit. The master then may send another remote frame to the slave in order to obtain the value of the changed state. For example one may compute the net load on a 125 kbit/s line. The total inquiry frame, including EOF delimiters, is 100 bits long. A cyclic check for every 50 ms will cause a net load of 1.6 %. Another solution that sends wakeup signals from a slave to the master is described later on.

Bit Timing

The mechanism of the inframe response is nothing specific of MiniCAN, but a standard feature of CAN: The acknowledge bit requires an equal processing, arbitration is comparable. Thus the same limitations may be observed. For the physical transfer of information along the transmission and reception path various asynchronous delays will occur. For high speed transmission they are described e.g. within [ISO11898].

Figure 4 shows the typical delays along the bus line. Those delays are caused by output delay of the transmitter (t_1), delay along the bus line (t_2), input delay of the receiver (t_3), processing time within the electronic control unit (t_4), output delay of the second transmitter (t_5), delay in opposite direction along the bus line (t_6) and input delay of the original receiver (t_7).

[ISO11898] names a specific line delay of nominal 5 ns/m and a max. bus length of 40 m for High Speed operation. For a worst case estimation a signal will take $t_2 = t_6 = 40\ \text{m} * 5\ \text{ns/m} = 200\ \text{ns}$ along the bus line. Assuming a perfect compensation of delays within the second node ($t_3 + t_4 + t_5 = 0$), a dominant signal from a second node will appear after $t_2 + t_6 = 400$ ns delay at the bus connection of the first node. For a bit rate of 1 Mbit/s the total bit time length itself is 1 000 ns.

For those reasons a rather sophisticated synchronisation mechanism is required. CAN treats these delays by programming the specified bit segments. Strict limitations are set on oscillator tolerance within [CAN2.0]. The original specification asked for an oscillator precision of 0.5 %. This was eased for low speed transmissions to 1.5 %. Ceramic resonators may fulfil this specification, while quartz oscillators are requested for higher precision.

Another approach to minimise the hardware demands is not to use an external oscillator at all. Therefore a special integrated phase locked loop (PLL) unit within the slaves will detect and use the valid bus speed. For those reasons the 1-0 sequences are not only used to increase security, detect error conditions, avoid bit stuffing and force a fixed protocol length, but also to cause suited edges for exact hard synchronisation, sample points and bit time length computation.

This PLL uses an integrated oscillator of low precision, but takes both the bus signals and verification of the protocol contents to adjust the bit time length. Signal disturbances may get detected and filtered. The master provides exact edges for resynchronisation since it has to use the usual reference oscillator. Thus a lower precision is required for the slaves that never send more than three consecutive bits until a hard resynchronisation is possible.

Actual tests are processed to verify whether an automatic detection of sample point and synchronisation is suitable. As default a differential high speed bus line is assumed. Single line and low speed mode are possible as well. Open and short bus line failures do not hinder the communication as far as possible.

These mechanisms are not part of the Mini-

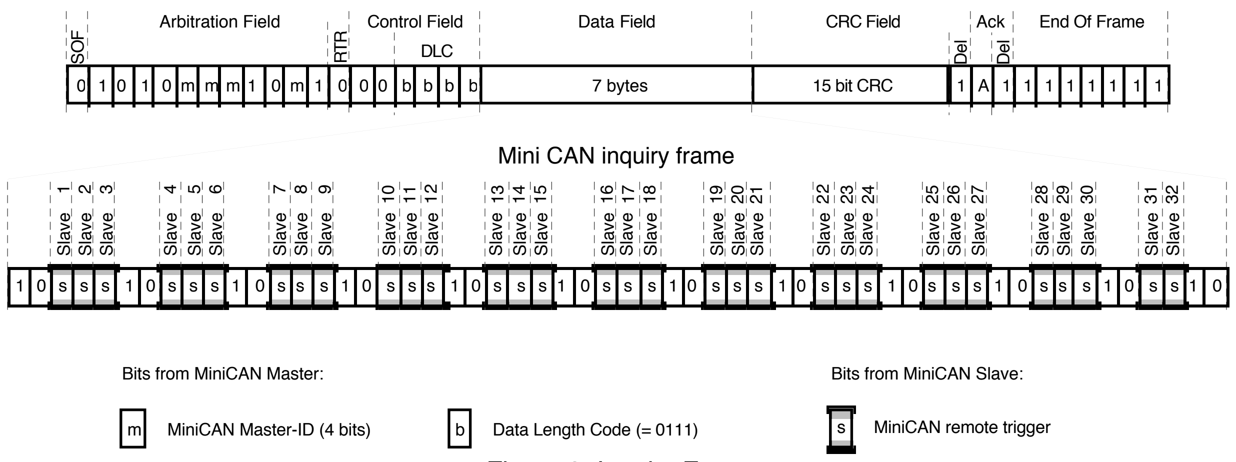
SOF
Arbitration Field
RTR
Control Field
DLC
Data Field
CRC Field
Del
Ack
Del
End Of Frame
0 1 0 1 0 m m m 1 0 m 1 0 0 0 b b b b
7 bytes
15 bit CRC
1 A 1 1 1 1 1 1 1 1
Mini CAN inquiry frame
Slave 1 Slave 2 Slave 3 Slave 4 Slave 5 Slave 6 Slave 7 Slave 8 Slave 9 Slave 10 Slave 11 Slave 12 Slave 13 Slave 14 Slave 15 Slave 16 Slave 17 Slave 18 Slave 19 Slave 20 Slave 21 Slave 22 Slave 23 Slave 24 Slave 25 Slave 26 Slave 27 Slave 28 Slave 29 Slave 30 Slave 31 Slave 32
1 0 s s s 1 0 s s s 1 0 s s s 1 0 s s s 1 0 s s s 1 0 s s s 1 0 s s s 1 0 s s s 1 0 s s s 1 0 s s s 1 0 s s 1 0
Bits from MiniCAN Master:
m MiniCAN Master-ID (4 bits)
b Data Length Code (= 0111)
Bits from MiniCAN Slave:
s MiniCAN remote trigger

Figure 3: Inquiry Frame

CAN concept itself, but depend on a special set-up of the physical connection. The aim is to use a slave that does neither require any programming of bit timing or bus speeds, nor uses an external oscillator, but relies on the built-in mechanisms.

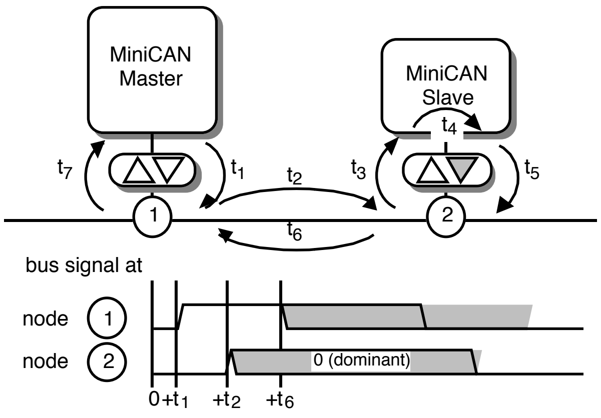

Figure 4: MiniCAN Bus Timing

Realisation

The MiniCAN concept is a definition of the data link layer, according to the Open System Interconnection standard (OSI: ISO 7498). From the user interface as highest layer, a slave accepts the address of master and slave ID. Six bits of data may get requested from the input lines by a remote frame. Six bits of data may get received and given to the output lines. From the physical layer it receives the bit information. It sends the bits for remote data, CRC or inquiry frame. As the PLL has to rely on further protocol analysis, some more information has to be exchanged. As indicated within the previous section, the physical layer may be either programmable or use auto configuration.

Different alternatives are possible for building the full slave system. Figure 5 shows a suggested minimal configuration. The programming of the master ID is done by mask programmable manufacturing. Thus every series of chips owns one of the fixed 16 master IDs. The slave ID may get defined e.g. by special input pins, by floating I/O pins or a programmable address (PROM). For a low-cost interface it is suggested to use an only one time programmable mode, e.g. by fuse technology. The input and output ports may get used separately or unified. The example shows a bi-directional 6 bits I/O port. Other variants may get adopted to customers' demands. The actual protocol is expected as optimum solution by number of slaves, data bits and security.

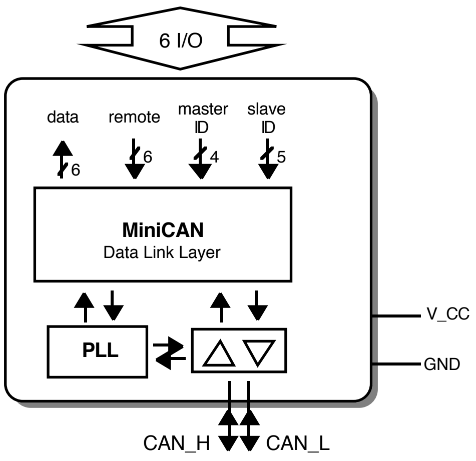

Figure 5: MiniCAN Slave: building blocks

A typical application takes both rising and falling signals from the input port. Without analysis a slave forwards these data within a remote frame to its master. The master may use a free programmable micro controller to interpret and translate this data. According to this translation it releases the actions at its slaves by a data frame, that appears at the slaves' output ports as toggle, switch, analogue digitised value etc.

Some further design variants show considerable improvements for these functions. The input has to get buffered when a signal is set and reset, while there had been no state request by a remote frame in between. A proper state machine was designed. Thus short events get buffered.

Another extension may implement a sleep mode. While in general a slave may not initiate a transmission on its own, there may be the need for a low-power mode. Cyclic requests by the master cause a power consuming standby or repeated wake-up. Instead a slave may initiate a wake-up signal via the bus. One solution is to send out a short dominant signal. Since the slave has neither an arbitration unit nor an exact bit time unit, this signal is of no exact predictable length, but may be read as error frame or cause an error frame of other nodes. This will tell the master to send an inquiry frame.

A preliminary software emulation has proved reliable transmission. Both the master and the slaves were emulated by standard micro controllers of type Philips: PCA 80C552. They were clocked by a 24 MHz oscillator, but use 12 clock cycles for one processor cycle and multiple processor cycles for one interrupt. Thus the maximum possible bus speed is about 60 kbit/s.

A MiniCAN Master may have the same building blocks as the usual CAN controllers, but requires modifications:

- The bus monitoring has to be set passive while slaves transmit data.

Since a slave may overwrite a recessive bit from the master within its slot by a dominant bit, the master has to read this information, but may not cause a bit error. A bit error is detected, when the bit value differs from the bit value sent. According to the actual CAN specification the monitoring is already deactivated "when a recessive bit is sent during arbitration, or a recessive bit is sent during ACK slot" ([ISO11898]). This must be extended for the slots reserved for the MiniCAN protocol.

- The master must compute and compare the 5 bits MiniCAN CRC and send the corresponding MiniCAN acknowledge.

This mechanism is identical to the 15 bits CAN CRC. This CRC is computed during reception. At least one bit time may be used for its computation before an acknowledge has to be sent. Both CAN and MiniCAN require a positive acknowledge on a correct checksum.

- The master must compute the 15 bits CAN CRC on all valid data, including the data received from the slaves.

The actual CAN CRC is computed by the generator-polynomial $x^{15} + x^{14} + x^{10} + x^{8} + x^{7} + x^{4} + x^{3} + 1$ on SOF, Arbitration field, Control field and Data field. Since the Data field may get modified by the slaves, the CRC must be computed not based on data sent from the master but on data sent and slave bits monitored. This computation is similar to the CRC verification by a receiver.

A combined CAN controller and MiniCAN master has been developed. It is based on the programmable fieldbus controller IX1 [DELTAt]. It is built on a FORTH programmed prototyping system. Although this controller provides two dedicated CRC units, the 5 bits MiniCAN CRC had to be computed by serial software routines that required multiple processor cycles. Thus its actual speed is limited to 500 kbit/s, while 1Mbps is aimed at. By an external CRC module a further speed increase is possible. Due to the limited resources of the IX1 a second, programmable micro controller is required to evaluate the slaves' information and to release the proper actions. It offers the full flexibility to configure only one master node for all attached slaves. A customised CAN controller with integrated MiniCAN master ability may perform both. The actual configuration is controlled by a PC interface directly connected to the IX1.

For higher performance the slaves were implemented as gate level netlist and realised within FPGAs. Those Free Programmable Gate Arrays were from the Texas Instruments TPC10 Series. Design, simulation and programming was done on the software systems ViewLogic and TI Action Logic. The gate arrays are based on regular multiplexer structures with antifuse interconnection. The max. clock frequency is specified as 100 MHz, more than actually required. A single TPC1010 chip provides 295 logic modules, equivalent to 1200 gates, and is sufficient to hold the full MiniCAN logic.

The actual design of a PLL shows a comparable complexity of several hundred modules. Actual research aims at test and optimisation of this building block. CAN transceivers require special buffers and thus could not be mapped within the FPGAs, but are attached as external components.

A migration from the FPGA's gate-level netlist to a real chip synthesis may include both. It will have to include the according mechanisms for programming the master and slave ID and the I/O port configuration. As described above, the compensation of buffer delays and the integrated oscillator is to be taken into consideration for the manufacturing process.

Results

A new concept has been developed that can be used within the existing CAN environment. Its inframe response and master-slave concept is not fully conform, but compliant to the actual CAN specifications, since other CAN controllers will not observe any difference to the ordinary CAN communication. This concept offers

- a six bit data port for sensors and actuators of lower complexity
- direct connectability to low and high speed CAN bus lines
- 32 slaves per every single of 16 possible master IDs
- low and high bus speeds up to 1 Mbit/s
- high transfer security,
- low hardware demands
- low configuration effort
- compliancy to the CAN specification 2.0, part A and B.

References

[CAN2.0]	*CAN Specification, Version 2.0.* Robert Bosch GmbH, Stuttgart, 1991.
[DELTAt]	DELTA t: *Programmable Fieldbus Controller IX1*. Hamburg, 1995.
[ISO11898]	*ISO 11898: Road Vehicles – Interchange of digital information – Controller Area Network (CAN) for high-speed communication*. International Organization for Standardization, 1993.
[PCA82C250]	Philips: *Data Sheet PCA82C250. CAN controller Interface.* Eindhoven, September 1994