Revision and Verification of an Enhanced UART

Raffaele Gallo, Martin Delvai, Wilfried Elmenreich, Andreas Steininger
Vienna University of Technology
Institut für Technische Informatik, Vienna, Austria
e9725989@stud3.tuwien.ac.at, delvai@ecs.tuwien.ac.at,
wil@vmars.tuwien.ac.at, steininger@ecs.tuwien.ac.at

Abstract

This paper describes the design of an eUART (enhanced Universal Asynchronous Receiver and Transmitter) that has been designed as an extension module for the SPEAR processor in order to interact with a TTP/A (Time-Triggered Protocol Class A) or LIN (Local Interconnect Network) network. The eUART is able to automatically synchronize its baud rate using a synchronization pattern from the message sender. Furthermore, to enhance the robustness of the communication, the eUART module is equipped with a filter state-machine and provides 16fold oversampling. The interpretation of the samples can be configured in order to optimize towards fault detection or availability. The paper describes the capabilities and limits of this approach and shows the advantages of the eUART when used for real-time communication.

1. Introduction

Technical advantages made by the silicon industry make it economically attractive to build even low-cost systems with a distributed microcontroller-based control system. In order to get the information from the environment, several sensors/actuators are connected over a field bus network. Due to the fact that such field-bus based systems are used for controlling purpose, in many cases real-time features are demanded.

TTP/A[1] and LIN[2] are two field-bus protocols that fulfill the requirements mentioned above. Both protocols aim at the implementation of predictable communication on the one hand and at the reduction of costs by using commercial-off-the-self (COTS) hardware like standard UARTs on the other hand. To reduce size and cost of a network node it is desirable to integrate all components on a single silicon die.

Since quartz crystals cannot yet be integrated on silicon dies, the clock source must either be placed beside the die, increasing size and price, or an imprecise RC-oscillator must be used. The disadvantage of an RC-oscillator is that it shows, depending on its temperature and the supply voltage, a rated frequency up to ±50% with a frequency-drift of ±10% per second. The low clock frequency limits the adjustability of the transfer rate of the UART leading to an arithmetic error. Another problem results from the high frequency drift of the RC-oscillator since this directly affects the baud rate of the UART. In fact, an analysis of the impact of an imprecise oscillator on a TDMA-based network showed that standard components are unable to work under these conditions[3]. According to the results of this analysis a prototype of a UART, called *eUART ("e"* stands for *enhanced*) has been developed [4]. By reducing the arithmetic error in baud rate setting and eliminating the send jitter, the eUART is able to operate even under worst-case assumption.

First experiences have shown that some improvements were required, especially to be more suitable for time-triggered protocols. The synchronization process has been revised so that it recognizes a synchronization pattern from a stream of messages and adjusts the baud rate respectively.

Another weak point of the eUART was the receive mechanism: the first version performed only one sample point per bit cell. To be more robust against interferences, the need of an oversampling mechanism arose. In case of an oversampling error, an additional mechanism allows to conclude if the error is caused by a disturbance on the communication channel or due to the fact that the node has lost synchronization.

The remainder of the paper is organized as follows: Section 2 gives the current state of our work. The synchronization condition and the fault hypothesis, which are the theoretical background of our eUART, are discussed in section 3. The paper ends with a conclusion.

2. First Prototype

Protocols such as TTP/A and LIN that are intended to be supported by our eUART use a time triggered communication schedule to achieve a predictable timing behavior. For these purpose issues of clock drift, baud rate deviations, send jitter, and the problems with imprecise oscillators had to be considered and quantified. In time triggered protocols with a TDMA arbitra-

tion scheme the transmission and reception of messages (frames) has to occur within specified temporal boundaries referred to as time slots. For a correct transmission two conditions have to be fulfilled:

(1) Inter-Slot Condition: The UART frame must not exceed the boundaries of its assigned time slot.

(2) Intra-Slot Condition: The baud rate difference between transmitter and receiver must be low enough, so that transmitted messages can be decoded correctly by the receiver.

The first prototype of the eUART was optimized according to these conditions. Figure 1 shows the resulting devices.

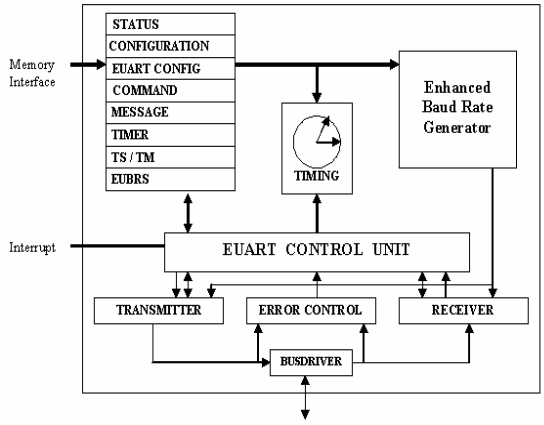

Figure 1. Block Diagram of the eUART extension module

The communication conditions are specified in [4], the eUART is depicted in [3].

3. Improvements

3.1. Synchronization Algorithm

Both protocols, TTP/A and LIN, transmit a synchronization pattern at the begin of a new communication round. This synchronization pattern consists of a sequence of zeros and ones, thus appearing as a regular pattern on the bus that cannot be reproduced by general data traffic due to data encoding constraints. The pattern is used for two purposes: First, a new node uses this pattern to integrate itself in a running network. Second, already synchronized slave nodes resynchronize their time base and respectively their baud rate with the reception of this message. The main difference between these two cases is that a new node does not have knowledge about the baud rate and the instant, when the synchronization pattern will be sent. So the only way to detect the synchronization pattern is to listen at the communication channel and to analyze the time elapsed between transitions. In the case of the synchronization pattern, all transitions are equidistant. Hence, if eight equidistant transitions were detected then a valid synchronization pattern was received[1]. If only one transition fails the equidistance condition, then the algorithm has to be restarted. It is clear that in a real system with imprecise oscillators the measurement of identical bit cells at different points in time can apparently yield to different results. Therefore, the algorithm has to consider a tolerance concerning the duration of a transmitted bit.

Further, "aliasing" has to be taken into account: a sequence of messages which contain only two transitions can look like a synchronization pattern transmitted with much more lower baud rate than the actual baud rate of the network. Due to the fact that the synchronization algorithm has no a priori knowledge concerning baud rate, this would result in a synchronization error.

The tolerance that has to be implemented in the algorithm has to be chosen a follows:

$$\mathrm{Tol}_{\text{bit cell duration}} < \mathrm{Tol}_{\text{algorithm}} < \mathrm{Tol}_{\text{aliasing}}$$

where the bit cell duration tolerance is given by the drift of the receiver and the aliasing tolerance is given by the communication protocol.

The lower boundary is derived as follows. As mentioned in the introduction, the clock signal of the receiver is subject to a drift rate ρ. Let us assume that the reference signal does not have any drift.

Let us further assume that a correct synchronization pattern is transmitted. What happens at the receiver side: the duration between the first two transitions is measured with a clock signal at the time t_0. The duration between the last two transitions is measured with the clock signal at the time t_1. Due to the drift of the receiver's clock, the duration of a clock cycle may change in between introducing a measurement error[2]:

$$\text{deviation} = \rho \cdot \frac{n}{\text{Baudrate}}$$

where n is the number of bit cells of the synchronization pattern. By using conventional parameters like n=8, f_{clk}= 1 MHz, ρ = 10 % and baud rate = 20000, the resulting deviation is less than 0,01 %. This deviation is neglectable, but it can cause a difference of one clock tick between the duration of the last two transi-

[1] Different parity check setting between normal messages and the synchronisation message guarantees that only the synchronisation pattern can produce eight transition in a single time slot.

[2] Strictly speaking, even the drift during the transmission of a single bit cell has to be considered. But the resulting ulterior deviation is so small that it can be neglected.

tions in comparison to the duration between the first two. In this case the resulting deviation is

$$\text{deviation} = \frac{n+1-n}{n} = \frac{1}{n}$$

where n is the number of clock cycles per bit, formally:

$$n = \frac{f_{clk}}{\text{Baudrate}}$$

In order to define the upper boundary of the tolerance we assume a worst-case scenario in which the eUART is erroneously synchronizing on a "malicious" bit pattern over multiple time slots. Since the algorithm parses a synchronization pattern only by means of equidistant transitions, continuous sequences of arriving zeros and ones would lead to the calculation of a wrong baud rate.

Depending on the duration of a time slot we have to consider two conditions in order to prevent a wrong synchronization: the symmetric and the asymmetric time slot conditions.

In a symmetric time slot, where the message length is an even number of bits, the case in which half of the time slot is made up of a sequence of zeroes whereas the other half are ones, has to be eliminated. This can be achieved by choosing the parity check as follows:

$$\frac{\text{MsgLength}}{2} \begin{cases} even \Rightarrow EvenParity \\ odd \Rightarrow OddParity \end{cases}$$

As we can see in Figure 2, it is impossible for the eUART to synchronize on a "malicious" sequence of 12 bit patterns since the small duration of the parity bit leads the synchronization algorithm to break up, starting all over again.

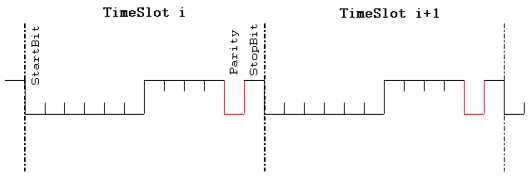

Figure 2. Symmetric time slot condition

In asymmetric time slots, where the message length n=2k+1 is an odd number, we have to prevent the case that sequences of k zeros and k+1 ones (or vice versa) are interpreted as a synchronization pattern by setting a tolerance upper limit as follows:

$$Tol_{aliasing} = \frac{k+1}{k} - 1$$

The first prototype of the eUART implemented a rather simple synchronization algorithm: the synchronization command had to be set just before the synchronization pattern is transmitted by the master node. The new synchronization mechanism applies the synchronization algorithm described in this section. Hence, the synchronization command can be activated any time, and the eUART is able to extract the synchronization pattern from all messages transmitted over the bus. In order to map the synchronization algorithm to hardware we have to fix the tolerance values:

According to the previous calculation the lower boundary was set to 4 %. The upper boundary is defined by the length of a time slot. In case of TTP/A a time slot has a duration of 13 bit cells; therefore, a "critical" message can comprise either 6 ones and 7 zeros or 7 zeros and 6 ones. This results in an upper boundary of (7/6 –1) = 16,6%. We have chosen a tolerance of 6,25% for the synchronization algorithm, because it is easy to implement in hardware:

$$t_{Bit_i} \in \left\lfloor t_{Bit_j} - 2^{-4}; t_{Bit_j} + 2^{-4} \right\rfloor$$

3.2. Robustness

Experiences with real hardware have shown that one sample per bit cell is not accurate enough. A higher reliability in receiving data is demanded. For this purpose an oversampling mechanism that scans each received bit cell 16 times has been inserted in the receiver unit. In this section we briefly describe the assumptions we made concerning the types and number of faults that our eUART is expected to tolerate. This fault hypothesis divides the fault space into two disjoint partitions: the partition of the covered and those of the uncovered faults. Moreover, we regard faults on bit cell level and faults on UART-frame level:

Covered faults:

- Short interferences: Disturbances on the bus that do not last longer than 2 clock ticks are filtered by a digital filter situated in the bus driver unit [4].
- Noise: Disturbances on the bus that cannot be preprocessed by the mentioned filter are submitted to the configurable 16fold oversampling mechanism that evaluates the samples. All interferences of a duration shorter than ½ bit cell can be detected and corrected by the oversampling mechanism. Interferences of a longer duration are not guaranteed to be detected by the oversampling mechanism, but are detected by the parity check, if the disturbance affects an odd number of bits within one UART frame.
- The oversampling mechanism can be adjusted to provide either maximum availability of the communication (thus a sampling is considered correct if there is a majority of *high* or *low* samples) or to provide maximum robustness (thus a sampling must have a configurable significantly higher number of *high* or *low* samples)

Uncovered faults:

- Physical channel breakdown: The eUART is set in the impossibility to work.
- Crash of the master node: As long as the eUART is in synchronization mode it will try to detect the synchronization pattern in order to fulfil its function.
- Burst errors that affect more than half of the bit samples: In this case, a bit is interpreted incorrectly. If the number of incorrectly interpreted bits per frame is an even number, the parity check will not detect this error at frame level either.

The oversampling mechanism provides also the result from the last sampling for diagnostic purposes, thus it is possible to distinguish the case of a timing failure from a noisy channel (see Figure 3).

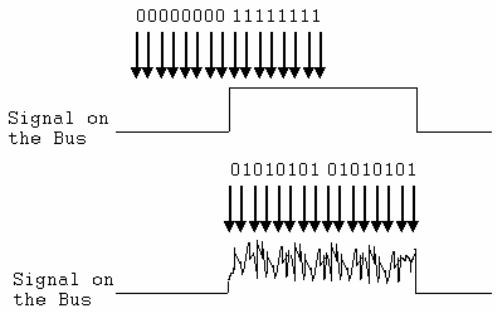

Figure 3. Evaluation of the sampling ticks for one bit cell

4. Conclusion and Outlook

In this paper we have described and analyzed a modified UART to be used in a TTP/A, a LIN or similar low-cost automotive networks. We have presented the improvements made on an existing prototype of the eUART. The introduction of a configurable oversampling mechanism in the design offers the possibility of a user definable robustness factor for the expected quality of the communication channel. Moreover, the user is set in the position to analyze and interpret received sample patterns that lead to an error. Our theoretical approaches have taken shape through the implementation of a more flexible synchronization algorithm thus relieving the user. The next step will be to integrate the eUART with existing software, for example a TTP/A implementation. Furthermore, we are planning to verify our theoretical approaches by measurements using the synthesized design of the eUART.

Acknowledgments

This work was supported by the Hochschuljubiläumsstiftung der Stadt Wien via project MOSAIC (H-1147/2002).

References

[1] H. Kopetz et al. Specification of the TTP/A protocol. Technical report, Technische Universität Wien, Institut für Technische Informatik, March 2000. Available at http://www.vmars.tuwien.ac.at

[2] Audi AG, BMW AG, DaimlerChrysler AG, Motorola Inc., Volcano Communication Technologies AB, Volkswagen AG, and Volvo Car Corporation. LIN specification and LIN press announcement. SAE World Congress Detroit, http://www.lin-subbus.org, 1999.

[3] W. Elmenreich, M. Delvai, Time Triggered Communication with UARTs. In *Proceedings of the 4th IEEE International Workshop on Factory Communication Systems, Aug. 2002*.

[4] M. Delvai, U. Eisenmann, W. Elmenreich, *Intelligent UART Module for Real-Time Applications.* First Workshop on Intelligent Solutions in Embedded Systems (WISES), pages 177-185, Vienna, Austria, June 2002