Time-Triggered Communication with UARTs

Wilfried Elmenreich and Martin Delvai
Institut für Technische Informatik
TU Vienna, Austria
wil@vmars.tuwien.ac.at, delvai@vlsivie.tuwien.ac.at

Abstract

The simple UART (Universal Asynchronous Receiver and Transmitter) encoding is used in two novel fieldbus protocols for real-time systems, the Local Interconnect Network (LIN) and the Time-Triggered Protocol for SAE class A applications (TTP/A). These protocols use a time-triggered communication schedule to achieve a predictable timing behavior. The employment of standard components like a standard UART reduces costs, but the issues of clock drift, send jitter, and the adjustability of the send frequency have to be considered.

This paper examines common timing problems with standard UARTs and imprecise oscillators and presents a calculation of upper bounds for the timeliness of UART driven communications.

Furthermore we discuss methods to solve the timing problems when imprecise on-chip oscillators are used. The synchronization support of time-triggered fieldbus protocols often relies on a configurable UART without send jitter. Standard hardware UARTs usually do not hold this requirement. It is possible to use a software UART implementation but at the cost of node performance. We have developed an enhanced UART architecture that behaves better for time-triggered systems than standard UARTs. Together with proper protocol synchronization support this approach allows the integration of nodes with imprecise clocks in time-triggered real-time systems.

1. Introduction

A UART (Universal Asynchronous Receiver/Transmitter) is a component used for serial communications, containing a transmitter (parallel-to-serial converter) and a receiver (serial-to-parallel converter). The parallel side of a UART is usually connected to the bus of a computer. When the computer writes a byte to the UART's transmit data register, the UART will start to transmit it on the serial line.

The simple UART encoding is used in two novel fieldbus protocols for real-time systems, the Local Interconnect Network (LIN) and the Time-Triggered Protocol for SAE class A applications (TTP/A). Both protocols aim at the implementation of predictable communication for sensors and actuators on commercial-off-the-shelf (COTS) hardware. Since today most microcontrollers already include a unit for serial transmission this new class of protocols opens a way for low-cost implementations of real-time fieldbus networks. LIN and TTP/A both use a time-triggered communication schedule to achieve predictable timing behavior and provide clock synchronization for imprecise cheap on-chip oscillators [9]. To achieve the intended goal of predictable communication, it must be ensured, that the UART communication of a microcontroller with an on-chip oscillator can hold the timing requirements imposed by the communication protocol, although UARTs were not originally designed for this application class.

It is the objective of this paper to investigate the timing properties of a UART communication and discuss consequences on protocol and hardware design.

The remainder of the paper is organized as follows: Section 2 states the boundary conditions of the problem. Section 3 examines the timing properties of a UART communication and derives two conditions that are sufficient for a successful communication. Section 4 analyzes the timing properties of two standard UARTs. Section 5 presents approaches to solve the stated problems, i. e. a software UART implementation and the VLSI design of an enhanced UART architecture. Furthermore this section describes the clock synchronization mechanisms of the LIN and the TTP/A protocol. The paper is concluded in Section 6.

2. UART communication timing

The timing of a UART transmission is influenced by various factors. The baud rate of a UART is usually configured by integer values – the arithmetic rounding error leads to baud rate deviations. The architecture of common UARTs furthermore leads to intrinsic delays at the sending UART. Furthermore one has to regard the qualities of the clocks of the communication partners. Clock drift and offsets influence the baud rate and the instant of communication start. The network itself adds timing variations by the signal runtime. In this section we will quantify these five timing deviations separately.

2.1. Arithmetic error in baud rate setting

The baud rate (BR) depends on the frequency of the UART clocking. The baud rate is set by choosing a value $UBRS$ (**U**ART **B**aud **R**ate **S**etting) as follows:

$$BR = \frac{f_{clk}}{C_1 \cdot (UBRS + C_2)} \tag{1}$$

The integer constants C_1 and C_2 depend on the UART implementation. Typically C_1 represents the number of samples per bit cell (e. g. 16 for Atmel AVR RISC series) and C_2 is typically 1.

Using formula 1 the variable UBRS is set to define the baud rate. To achieve a desired baud rate, UBRS has to be set to:

$$UBRS_{Ideal} = \frac{f_{clk}}{C_1 \cdot BR} - C_2 \tag{2}$$

Equation 2 yields a rational number for UBRS. Common UARTs only accept integer values for UBRS, so UBRS will be rounded to an integer value in approximation of the ideal value (see Equation 3). The error resulting from this rounding will be considered as the arithmetic error.

$$UBRS_{Real} = \left\lfloor \frac{f_{clk}}{C_1 \cdot BR} - C_2 + 0.5 \right\rfloor \tag{3}$$

The rounding error can amount at most $\pm\frac{1}{2}$ in the worst case. Assume that we want to set the UART to a given (ideal) baud rate. For every clock frequency f_{clk} we can determine a UBRS that best approximates the desired baud rate. Thus we can give an upper bound for the ratio between the fastest and the slowest approximated baud rate:

$$\frac{BR_{fast}}{BR_{slow}} < \frac{UBRS + C_2 + \frac{1}{2}}{UBRS + C_2 - \frac{1}{2}} \tag{4}$$

2.2. Send jitter problem

At initialization of the UART the baud rate generator is started. Running at a frequency corresponding to the configured baud rate, the baud rate generator periodically generates ticks which are possible start events of a message bit. When the UART receives a transmit signal, it starts the transmission at the next tick from the baud rate generator. Thus, depending on the internal state of the baud rate generator the transmission of a message may be delayed up to the interval time of two subsequent baud rate generator ticks. Usually the state of the baud rate generator cannot be read, leading to an indeterministic send delay jitter:

$$0 \leq t_{Jitter} < \frac{1}{BR} \tag{5}$$

Subsequent transmissions may not be affected by this send jitter, however if transmissions can start at particular instants, the send jitter jeopardizes the achievable accuracy for this send instant.

2.3. Baud rate drift

Usually components performing a UART communication are set apart from each other and are clocked by different clock sources. Clock synchronization algorithms are used to synchronize these clocks periodically. After synchronization clocks can drift apart from each other, depending on their drift rate. The drift of a physical clock is the ratio between the actual clock frequency f_{clk} and the reference frequency f_{ref}. Good clocks have drifts that are very close to 1, so the *drift rate* ρ is introduced as [8]:

$$\rho = \frac{f_{clk}}{f_{ref}} - 1 \tag{6}$$

A perfect clock will have a drift rate of 0. Real clocks have varying drift rates depending on temperature, voltage variations, or aging effects. Typical drift rates for quartz crystal oscillators are in the range of 10^{-2} to $10^{-7}\frac{s}{s}$. Embedded microcontrollers (e. g. Atmel AVR RISC series [1]) often contain on-chip RC oscillators which show drift rates of up to $0.5\frac{s}{s}$. The drift rate itself can vary over time.

We assume that a clock synchronization algorithm synchronizes all clocks to have a maximum drift rate of $|\rho| < \rho_0$ at time $t = 0$. Furthermore we assume a maximum change per time unit of the drift rate $|\dot{\rho}| < \dot{\rho}_0$. Thus the maximum clock drift accumulates over time and calculates to:

$$|\rho_{max}(t)| = \rho_0 + \dot{\rho}_0 \cdot t \tag{7}$$

The baud rate depends on the frequency of the UART clocking and is therefore influenced by the clock drift. The worst cases for the baud rate with respect to drift can be calculated as:

$$BR(t) = \frac{f_{ref} \cdot (1 \pm \rho_{max})}{C_1 \cdot (UBRS + C_2)} \tag{8}$$

2.4. Clock offset

This problem refers to the offset between the particular clocks of the communicating nodes. The offset of the local clock affects the instant when the communication partners send or expect to receive a message. The maximum clock offset of a free running clock with a maximum drift rate $\rho(t) = \rho_0 + \dot{\rho}_0 \cdot t$ can be calculated as follows:

$$t_{Offset}(t) = \pm \int_0^t \rho_0 + \dot{\rho}_0 \cdot t \; dt \tag{9}$$

The lower bound of the integral defines the instant when the clock was last synchronized to zero offset and calibrated to a drift $|\rho| < \rho_0$.

2.5. Signal runtime

Electric signals in a cable travel at approximately 2/3 of the speed of light. This results in a delay of:

$$t_{Signal} \approx \frac{l}{2 \cdot 10^8 \frac{m}{sec}} \tag{10}$$

l is the length of the cable between two transceivers. Generally this length takes different values for every two nodes in linear and ring bus topologies. By calculating the signal delay for all possible signal ways, the delay splits up into a *minimal delay* and a *variable delay* between 0 and (*maximal delay* - *minimal delay*).

In a star topology the delay between two arbitrary nodes can even be made all the same so that the *variable delay* results to 0. When the send instant is a priori known to the sender and collisions between subsequent UART frames are obviated the signal delay can be compensated by sending the time of *minimal delay* earlier.

3. Timing limits of UART communications

Considering the timing problems presented in the previous Section it is the purpose of this Section to examine the timing limits for UART communications.

We assume a system of multiple communication nodes using serial communication with broadcast characteristics. There are no handshake or control lines. Bus arbitration will be solved by a time-slotting method, thus it is necessary to guarantee that the beginning and the ending instants of a UART frame lie within a given transmission window. While this transmission window is defined with respect to a global time, the transmitting nodes only have access to a local time established by a local clock with a known drift rate. The local time is frequently synchronized to the global time. The smallest unit of transmission is one data word which is transmitted in a UART frame. A UART frame consists of 1 start bit, a number of data bits, an optional parity bit and 1, 1.5, or 2 stop bits. The number of data bits, the parity bit and the number of stop bits must be set a priori in all communication partners. An investigation on the features of selected UARTs showed that the number of data bits ranges is typically 7, 8, or 9. A frame format of 1 start bit, 8 data bits, 1 parity bit, and 1 stop bit is supported by all examined UARTs [4].

Many UARTs perform multiple sample points to detect a bit cell and decide on a majority vote. This method affords a multiple of the sampling frequency for single bit detection but provides immunity to short spike disturbances on the communication line. In our paper we will assume, that there is only one detection point in the middle of a bit cell. UARTs with multiple sample points per bit cell may be even more critical to baud rate differences.

3.1. Maximal baud rate difference

This section investigates the maximum allowed deviation of the baud rates of sender and receiver in order to correctly communicate.

We analyze the single bit cells of a UART frame and calculate the maximum admissible delay until bits are detected incorrectly. Figure 1 depicts a message in transmission. Sender and receiver use baud rates that deviate from the ideal baud rate. The worst case is a slow receiver trying to listen to a sender transmitting too fast. The other extreme with a slow transmitter and a fast receiver is less critical. For correct transmission, the instant of the detection point of the last bit cell of a frame has to be prior the instant of the end of the transmitted UART frame (see Figure 1).

The condition for a successful communication is given in Equation 11:

$$\frac{n - \frac{1}{2}}{BaudRate_{slow}} - \frac{n}{BaudRate_{fast}} \leq 0 \tag{11}$$

n is the UART frame length in bit cells includ-

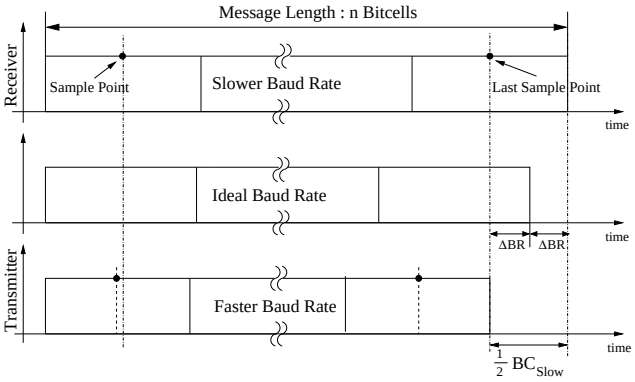

Figure 1. Maximum baud rate deviation

ing start, parity and stop bits. By rewriting (11) we find that the ratio between Baud Rate$_{fast}$ and Baud Rate$_{slow}$ must fulfil the following condition in order to guarantee correct communication.

$$\frac{BR_{fast}}{BR_{slow}} < \frac{n}{n - \frac{1}{2}} \tag{12}$$

3.2. Difference between transmit and receive window

The receiver is triggered by the first falling edge of the transmitted UART frame. Therefore the receiver must start to listing before the sender starts the transmission.

For further analysis we assume that any message will be transmitted within one time slot comprising UART frame length plus 2 times a duration named t_{Tol}. The UART frame consists of n bit cells, where the first is named start bit and the last stop bit (see Figure 2).

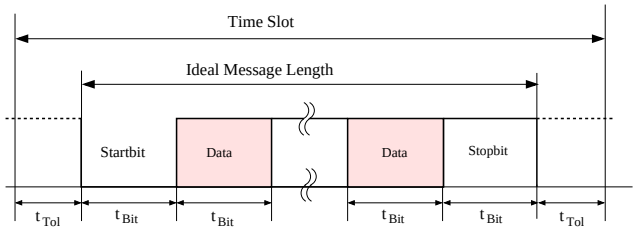

Figure 2. Maximum delay

Since the UART frame has to fit into the time slot, the start of transmission can lie between the beginning of the slot and $2 \cdot t_{Tol}$ later.

The receiver is triggered by the first edge of the start bit of the message. We assume clocks of identical quality for receiver and transmitter. Thus, both communication partners can drift $\pm t_{Tol}$. The receiver must listen for incoming messages t_{Tol} *before* it expects the slot and up to $3 \cdot t_{Tol}$ *after* it expected the beginning of the slot in order to cover the cases when the sender's and receiver's clocks have diametrical offsets of $\pm t_{Tol}$.

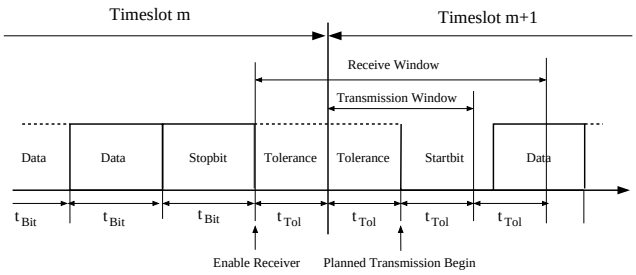

Figure 3. Receiver enable and transmit window

The value of t_{Tol} must be chosen to be less than the duration of a bit cell t_{Bit}. Otherwise the receive window may intersect with a bit of a UART frame in the previous slot and cause a false alarm start bit.

3.3. Synchronization interval

This section determines the maximal duration, that two systems can communicate with each other until a resynchronization of local clock times and a recalibration of baud rate settings is necessary. For correct transmission two conditions have to be fulfilled:

(1) **Inter-Slot Condition**:
The UART frame must not exceed the boundaries 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.

3.4. Inter-slot condition

To guarantee that a UART frame lies within its assigned time slot, the deviation from the planned transmission start and the maximum length of a UART frame must be taken into account. As explained in section 3.1 the frame length may differ by at most half a bitcell, thus, a UART frame may be either $\frac{1}{4}t_{Bit}$ longer or $\frac{1}{4}t_{Bit}$ shorter . Taking the longest UART frame into account, we shift the start of transmission $\frac{1}{8}$ bit cells ahead the planned transmission begin (see Figure 4).

$t_{maxDeviation}$ is the maximal allowed displacement between real and ideal transmission start. This displacement is composed of send jitter and timer offset. By setting the transmission start half a bit cell prior to the ideal transmission instant, only $\frac{1}{2}t_{Jitter}$

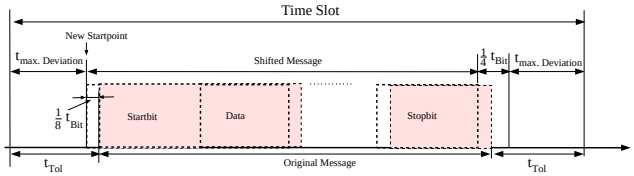

Figure 4. Shifted transmission startpoint

must be taken into consideration.

Thus the Inter-Slot condition can be expressed by the following relation:

$$t_{Offset}(t) + \frac{1}{2}t_{Jitter} + t_{Signal} \leq t_{maxDeviation} \quad (13)$$

t_{Offset} can be expressed by Equation (9) and $t_{max.Deviation}$ by $(t_{Tol} - \frac{1}{8}t_{bit})$:

$$\int_0^{t_{InterSl}} \rho_0 + \dot{\rho}_0 t \; dt + \frac{1}{2}t_{Jitter} + t_{Signal} \leq t_{Tol} - \frac{1}{8}t_{Bit} \quad (14)$$

Solving unequation 14 yields the duration from the last clock synchronization where the Inter-Slot condition can be guaranteed:

$$t_{InterSl} \leq \frac{-\rho_0 + \sqrt{\rho_0^2 + 2\dot{\rho}_0(t_{Tol} - \frac{1}{8}t_{Bit} - \frac{1}{2}t_{Jitter} - t_{Signal})}}{\dot{\rho}_0} \quad (15)$$

In case of a time invariant drift rate condition 15 becomes:

$$t_{InterSl} \leq \frac{t_{Tol} - \frac{1}{8}t_{Bit} - \frac{1}{2}t_{Jitter} - t_{Signal}}{\rho_0} \quad (16)$$

3.5. Intra-slot condition

The deviation between the sender's and the receiver's baud rate has to be low enough to ensure that all bits are detected correctly. Figure 5 illustrates a detection error caused by differing baud rates on bit 6 of a UART frame.

A deviation from the ideal baud rate can be caused by rounding errors in the baud rate setting (see Section 2.1) and baud rate drift (see Section 2.3).

By combining Equations 4, 7, 8, and 12, the Intra-Slot condition for a successful communication is given by :

$$\frac{BR_{fast}}{BR_{slow}} \cdot \frac{1 + (\rho_0 + \dot{\rho}_0 \cdot t)}{1 - (\rho_0 + \dot{\rho}_0 \cdot t)} < \frac{n}{n - \frac{1}{2}} \quad (17)$$

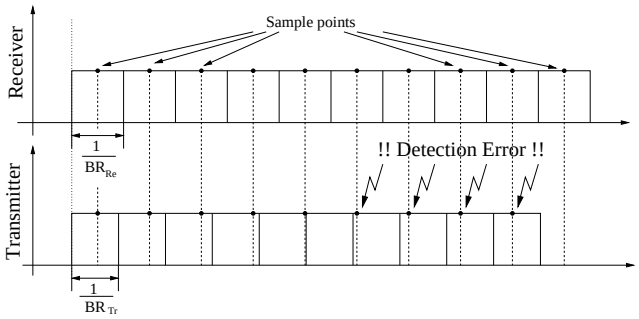

Figure 5. Detection error

The clock drift during the transmission of the message itself will be neglected, because the impact compared to arithmetic and drift error is less than 10^{-3}. The Intra-Slot condition is true if:

$$t_{IntraSl} \leq \frac{n(1-\rho_0)BR_{slow} - (n - \frac{1}{2})(1+\rho_0)BR_{fast}}{n\dot{\rho}_0 BR_{slow} + (n - \frac{1}{2})\dot{\rho}_0 BR_{fast}} \quad (18)$$

The synchronization interval time must comply the Inter-Slot (15) and the Intra-Slot (18) condition to guarantee the communication timing.

4. Evaluation

This section evaluates the timing properties for two common UARTs. For some f_{clk}, ρ_0, and $\dot{\rho}_0$, we have calculated the time intervals $t_{InterSl}$ and $t_{IntraSl}$ for valid Inter-Slot and the Intra-Slot conditions, respectively. Using these two values we derived the maximal number of messages n_{MSG} that can be transmitted until a resynchronization of local clocks and a recalibration of baud rate settings is necessary.

For the calculation the following communication parameters have been used:

- Baud Rate$_{Ideal}$= 19200 Baud
- Maximum cable length = 10 m
- UART frame length =11 Bit (1 start, 8 data, 1 parity and 1 stop bit)
- Each UART frame is transmitted within a 13 bit time slot (t_{Tol} = 1 bit cell)

Table 1 and 2 show an evaluation of the synchronization conditions for Atmel and AMD[1] microcontroller families.

[1] As recommended in the manual, the *clock mode* was set to 16

The Intra-Slot condition is mainly influenced by the arithmetic error. For most frequencies the arithmetic error exceeds a limit of ±4.5% which makes it impossible to establish a communication, even if the drift parameters of the quartz are perfect. However the market supplies quartz crystals with particular frequencies like 1.8432 MHz or a multiple of this value. At this frequencies the arithmetic error results to 0 and the Intra-Slot condition is fulfilled as long as the clock frequency is stable within a given window with respect to drift parameters.

When the Intra-Slot condition can be fulfilled, usually the Inter-Slot condition determines the maximum interval between synchronization events. For example a system with 1.5 MHz oscillator with a drift of $10^{-3}\frac{s}{s}$ and a maximum change of the drift rate by $0.1\frac{s}{s^2}$ must be resynchronized every 17.9 messages. This duration depends mainly on the quality of the clock, especially on the initial drift rate.

5. Possible solutions

Systems with imprecise RC oscillators or with quartz clocks that do not support the standard UART frequencies (most standard frequencies can be set without error if the clocking is a multiple of 1,843200 MHz) will not be able to use standard hardware UART components for communication. In this section we will investigate on solutions for this cases.

5.1. Software UART implementation

For low-speed communication it is feasible to implement a software UART routine, as long as the processor performance is sufficient. We implemented a software UART routine for an ATMEL 2313 and achieved a performance of up to 20 kBit/sec at a clocking of 1 MHz [6]. The baud rate setting for 19200 Bit/sec had an error of at most 0,25%.

Since the software UART routine performs only one sampling per bit cell, the software UART approach is not only less performant but also more vulnerable to short spike interferences on the bus.

5.2. An enhanced UART architecture

The smart sensor technology implies the integration of an analog or digital sensor or actuator element and a local microcontroller that contains the interface circuitry, a processor, memory, and a network controller in a single unit [10]. More and more sensor elements are themselves microelectronic mechanical systems (MEMS) that can be integrated on the same silicon die as the associated microcontroller.

An architecture supporting modular design of integrated smart sensors is the **S**calable **P**rocessor for **E**mbedded **A**pplication in **R**eal-Time Environments (**SPEAR**) [3]. It provides a predesigned, preverified, silicon circuit block. Its core-based system design [5] helps building a larger or more complex application on a semiconductor chip.

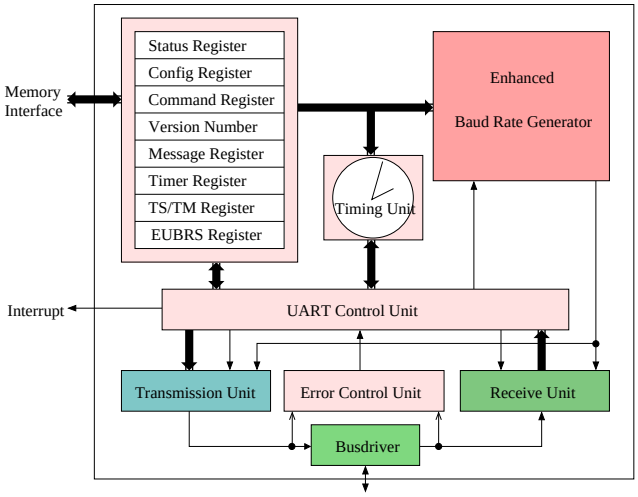

Figure 6. Block diagram of the UART I/O module

We have investigated on implementing a SPEAR module with a dedicated UART that is able to provide various baud rates independently of the clocking source.

The SPEAR core SPEAR features a 16 bit processor that executes instructions via a 3-stage pipeline. All processor parts are synchronized by a Wait Control Unit which resolves control and data hazards. The processor core features 20 general purpose registers and 12 control, pointer, and configuration registers. SPEAR comprises a 4 KB instruction cache and a 4 KB data cache. Usually these values suffice for many embedded applications, but it is also possible to add up to 128 KB external memory for instructions and 127 KB external data memory.

SPEAR supports up to 64 I/O modules which are mapped into the data memory. Examples for I/O modules are internal units such as additional timers, floating point units, and external units such as sensors, actuators, or network interfaces.

The UART is realized as an I/O module. The interface between such modules and processor in the

f_{clk} [MHz]	ρ_0 $[\frac{s}{s}]$	$\dot{\rho}_0$ $[\frac{s}{s^2}]$	Arith. Error [%]	Send Jitter [ms]	$t_{InterSl}$ [ms]	$t_{IntraSl}$ [ms]	n_{MSG} [–]
0.5	10^{-3}	0.1	18.62	0.052	12.1	0	**0**
1.0	10^{-3}	0.1	-8.51	0.052	12.1	0	**0**
1.0	10^{-5}	0.0	-8.51	0.052	1948.1	0	**0**
1.5	10^{-3}	0.1	2.34	0.052	12.1	104.1	**17.9**
1.8432	10^{-3}	0.1	0.00	0.052	12.1	222.6	**17.9**
1.8432	10^{-3}	0.0	0.00	0.052	19.5	∞	**28.8**
1.8432	10^{-5}	0.0	0.00	0.052	1948.1	∞	**2877.2**
2.0	10^{-3}	0.1	6.99	0.052	12.1	0	**0**
4.0	10^{-3}	0.1	-0.16	0.052	12.1	214.6	**17.9**
8.0	10^{-3}	0.1	-0.16	0.052	12.1	214.6	**17.9**

Table 1. UART timing properties of the ATMEL AT90S microcontroller family

f_{clk} [MHz]	ρ_0 $[\frac{s}{s}]$	$\dot{\rho}_0$ $[\frac{s}{s^2}]$	Arith. Error [%]	Send Jitter [ms]	$t_{InterSl}$ [ms]	$t_{IntraSl}$ [ms]	n_{MSG} [–]
1.0	10^{-3}	0.1	37.24	0.052	12.1	0	**0**
1.0	10^{-5}	0.0	37.24	0.052	1948.1	0	**0**
1.5	10^{-3}	0.1	-44.14	0.052	12.1	0	**0**
1.8432	10^{-3}	0.1	0.00	0.052	12.1	222.6	**17.9**
1.8432	10^{-3}	0.0	0.00	0.052	19.5	∞	**28.8**
1.8432	10^{-5}	0.0	0.00	0.052	1948.1	∞	**2877.2**
2.0	10^{-3}	0.1	-12.76	0.052	12.1	0	**0**
4.0	10^{-3}	0.1	8.16	0.052	12.1	0	**0**
8.0	10^{-3}	0.1	-0.17	0.052	12.1	214.6	**17.9**

Table 2. Timing properties of the AMD Z85C30 UART

SPEAR architecture is defined in [7]. The interface contains signals for reset, addressing, data I/O, and an interrupt. Via the memory interface eight 16-bit registers are accessible. The first two registers are the *status* and the *config* register with standard assignments for all I/O modules, the remaining registers are module-specific.

Figure 6 illustrates the block diagram of the UART I/O module. This UART component overcomes the disadvantages of send jitter and arithmetic error in baud rate setting described in Section 2. It starts immediately after receiving the transmit signal with the message transmission. In this way the send jitter is completely eliminated. The UART takes only one sample per bit cell. Instead of oversampling the bus signal we use a digital filter to circumvent spike interference problems. Figure 7 depicts the state diagram of the digital filter. At each clock cycle a transition takes place. Interfering signals shorter than n clock cycles are filtered out. The filter has a dead time of n cycle times, but needs less area on the silicon chip and allows higher transmission speeds than oversampling.

The enhanced UART baud rate setting (EUBRS) is the key element of the UART. The EUBRS calculates the duration of a bit for transmission by using a fixed comma value EUBRS according to the following formula:

$$t_{bit} = \frac{EUBRS}{16} \cdot \frac{1}{f_{clk}} \tag{19}$$

Because EUBRS is a 16 bit register the maximum bit duration calculates to $\approx 2^{12}$ timer ticks. The duration of a bit cell can be defined with a granularity of $\frac{1}{16}$ of a system clock period.

Table 3 shows an evaluation of the synchronization conditions for SPEAR.

5.3. Communication protocol clock synchronization support

Although our custom UART is able to overcome some of the deficiencies of a UART communication lined up in Section 2, the support of clock synchronization by the protocol is still necessary to enable the utilization of low-cost imprecise clock oscillators.

The UART encoding as examined in this paper is used in two novel fieldbus protocols for real-time systems, the Local Interconnect Network (LIN) [2] and the Time-Triggered Protocol for SAE class A applications (TTP/A) [11].

LIN has a communication scheme where each of up to 64 different messages is requested by the master with

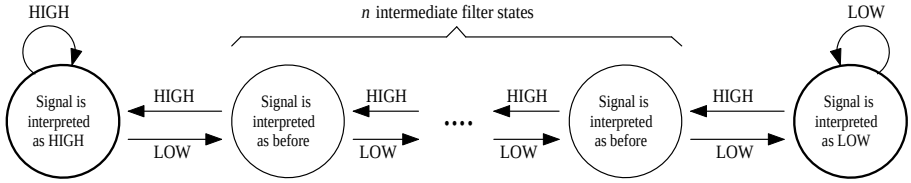

Figure 7. State diagram of the digital filter

f_{clk} [MHz]	ρ_0 $[\frac{s}{s}]$	$\dot{\rho}_0$ $[\frac{s}{s^2}]$	Arith. Error [%]	Send Jitter [ms]	$t_{InterSl}$ [ms]	$t_{IntraSl}$ [ms]	n_{MSG} [–]
0.5	10^{-3}	0.1	−0.16	0.0	21.8	214.6	**32.2**
1.0	10^{-3}	0.1	−0.16	0.0	21.8	214.6	**32.2**
1.0	10^{-5}	0.0	−0.16	0.0	4552.3	∞	**6723.4**
1.5	10^{-3}	0.1	−0.16	0.0	21.8	214.6	**32.2**
1.8432	10^{-3}	0.1	0.00	0.0	21.8	222.6	**32.2**
1.8432	10^{-3}	0.0	0.00	0.0	45.5	∞	**67.2**
1.8432	10^{-5}	0.0	0.00	0.0	4552.1	∞	**6723.4**
2.0	10^{-3}	0.1	−0.16	0.0	21.8	214.6	**32.2**
4.0	10^{-3}	0.1	−0.16	0.0	21.8	214.6	**32.2**
8.0	10^{-3}	0.1	0.08	0.0	21.8	218.6	**32.2**

Table 3. Timing properties of the enhanced SPEAR UART design

a one-byte identifier consisting of a six-bit address field and a two-bit checksum. Each message is uniquely assigned to a single slave or the master in a particular bus configuration. A message has a fixed length encoded in the identifier of the message with two, four, or eight data bytes and an additional checksum. The synchronization in LIN for non-time-aware slave nodes occurs before each master-slave round by sending a "break" byte and a synchronization byte which contains a regular bit pattern. Since start-up synchronization occurs before every master-slave round, no mechanism for the resynchronization of already integrated nodes is required. In LIN each message is preceded by a message from the master, thus, the Inter-Slot condition (Section 3.4) can be neglected. However the Intra-Slot (Section 3.4) condition has to be regarded, thus LIN supports either standard hardware UARTs together with appropriate clock sources ($k \cdot 1.8432MHz$) or imprecise oscillators with software UARTs/enhanced UARTs (Section 5.2).

TTP/A uses a TDMA scheme for bus arbitration. Each node in a cluster is aware of a predefined message schedule, a so-called round. Each round is initiated by the so-called fireworks byte which is issued by the master. After the fireworks byte each node transmits in its assigned slot. One time slot is 13 bit cells long and contains one UART frame of 11 bits length.

The start-up synchronization in TTP/A for non-time-aware slave nodes is performed by a special fireworks byte which contains a regular bit pattern that can be used by integrating nodes.

In the TTP/A protocol both the Inter-Slot condition (Section 3.4) and the Intra-Slot (Section 3.4) condition have to be taken into account. Thus, already integrated slave nodes need periodic resynchronization. It is performed at the instance when a byte from a node with a precise clock arrives, e.g., the fireworks byte from the master. In a multi-partner round those bytes that originate from a node with precise clocks are known to the slaves and are used for adjusting the node's local time. Like LIN, TTP/A also supports either standard hardware UARTs together with appropriate clock sources ($k \cdot 1.8432MHz$) or imprecise oscillators with software UARTs/enhanced UARTs (Section 5.2).

6. Outlook and conclusion

We have examined the applicability of common UARTs in time-triggered systems and developed mathematical means to guarantee that a UART frame stays within its given time slot (Inter-Slot condition) and to guarantee that all sent bits are detected correctly (Intra-Slot condition). Together, the Inter-Slot and the Intra-Slot condition form a sufficient criterium for correct communication timing.

Two standard UARTs have been evaluated using these conditions and have detected that certain factors limit the applicability of common UARTs, espe-

cially when combined with imprecise on-chip oscillators. Commercial UARTs exhibit timing imprecisions caused by an arithmetic error in their baud rate setting and a potenial send jitter. The unstableness of a real clock leads to baud rate drift and clock deviations. Special crystal frequencies (e. g. $1.8432 \cdot 10^6$) must be selected to cancel the arithmetic error.

However some new fieldbus real-time protocols aim at the low-cost commercial-off-the-shelf market. This market segment offers low-cost general purpose microcontrollers with imprecise on-chip oscillators. The protocols provide frequent clock synchronization and adequate slackness in the communication timing to fulfill the Inter-Slot condition. In general the Intra-Slot condition can only be fulfilled by using adequate quartz crystals, software UART implementations, or an enhanced UART architecture.

We have developed an alternative VHDL implementation of a UART unit that resolves the intrinsic UART problems and allows the implementation of more efficient protocols respectively the employment of cheap on-chip oscillators with large drift rates.

The VLSI module was designed as an I/O module for the composable SPEAR design. Utilizing the SPEAR design it will be possible to build an embedded transducer node, incorporating a microelectronic mechanical systems (MEMS) sensor, a microcontroller, an oscillator, and a communication controller on a single die.

Acknowledgments

This work was supported in part by the Austrian Ministry of Science, project TTSB and by the European IST project DSoS under contract No IST-1999-11585.

References

[1] Atmel Corporation, San Jose, CA. *AVR Enhanced Risc Microcontroller Data Book*, May 1997.

[2] Audi AG, B. AG, D. AG, M. Inc. V. C. T. AB, V. AG, and V. C. Corporation. LIN specification and LIN press announcement. SAE World Congress Detroit, http://www.lin-subbus.org, 1999.

[3] M. Delvai, W. Huber, B. Rahbaran, and A. Steininger. SPEAR - Design-Entscheidungen für den "Scalable Processor for Embedded Applications in Real-Time Environments". In *Tagungsband of Austrochip 2001, Oct. 12, Vienna, Austria*, 2001.

[4] C. Ebner. Description and comparison of selected UARTs. Technical Report 22/1994, Technische Universität Wien, Institut für Technische Informatik, 1994.

[5] R. K. Gupta and Y. Zorian. Introducing core-based system design. *IEEE Design & Test of Computers*, 14(4):15–25, Oct.-Dec. 1997.

[6] M. Holzmann and W. Elmenreich. Implementation details on the TTP/A slave protocol. Technical Report 4/1999, Technische Universität Wien, Institut für Technische Informatik, July 1999.

[7] W. Huber. Peripherieanbindung an SPEAR. Technical report, Technische Universität Wien, Institut für Technische Informatik, 2001.

[8] H. Kopetz. *Real-Time Systems, Design Principles for Distributed Embedded Applications*. Kluwer Academic Publishers, Boston, Dordrecht, London, 1997.

[9] H. Kopetz, W. Elmenreich, and C. Mack. A comparison of LIN and TTP/A. *Proceedings of the 3rd IEEE International Workshop on Factory Communication Systems, Porto, Portugal*, pages 99–107, September 2000.

[10] H. Kopetz, M. Holzmann, and W. Elmenreich. A universal smart transducer interface: TTP/A. *International Journal of Computer System Science & Engineering*, 16(2), March 2001.

[11] 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.ttpforum.org.