In Proc. 3rd Conference on Mechatronics and Robotics, Paderborn, Germany, Oct. 1995, Teubner, pp. 485–498

MERKUR: A real–time and fault tolerant communication system for mechatronic applications

Pfefferl Johann
Department of Process Control Computers, Prof. Dr. G. Färber
Technical University of Munich, Germany

Abstract: A mandatory condition for mechatronic systems to be realized in a unified manner is a communication system interconnecting the different, spatially distributed subsystems. In the last few years so called fieldbusses have been proposed to fulfill this task. A new communication concept presented in this paper provides the means to integrate sensors, actuators and computers to a complete mechatronic system. The requirements are analyzed and discussed in this paper. The article also explains, how these requirements are taken into account to realize MERKUR. The last part outlines the main aspects of the hardware implementation.

1 Introduction

Distributed real–time computer systems are replacing conventional centralized control techniques especially in this field of mechatronic systems, e.g. process control systems or hardware–in–the–loop simulations. Typical distributed systems are characterized by a controlled object (the mechanical system) and a control system (the computers). These two elements are connected via sensor based and actuator based interfaces. The control system accepts data from the sensors, processes them and outputs the results to the controlled object via the actuators. The aim of this action is to affect the dynamic mechanical object in such a way, that its behavior is optimized.

The general information flow between the two main components is illustrated in Figure 1. This process of merging mechanical parts with sensors, actuators, computer and software to an integrated system is known as mechatronics.

In practice, Figure 1 describes only a global view of the system. Typically, the mechanical system is spatially distributed. It consists of multiple subsystems and its dynamics model is represented by several degrees of freedom. The situation of the control system is similar. Due to the complexity of today's mechanical systems and due to the increasing iteration rates of the control loops, the computational loads of the algorithms are extremely high.

For that reason, most control units do not "fit" within a single computer module. Instead, many require multiple Central Processing Units (CPUs) to handle the mas-

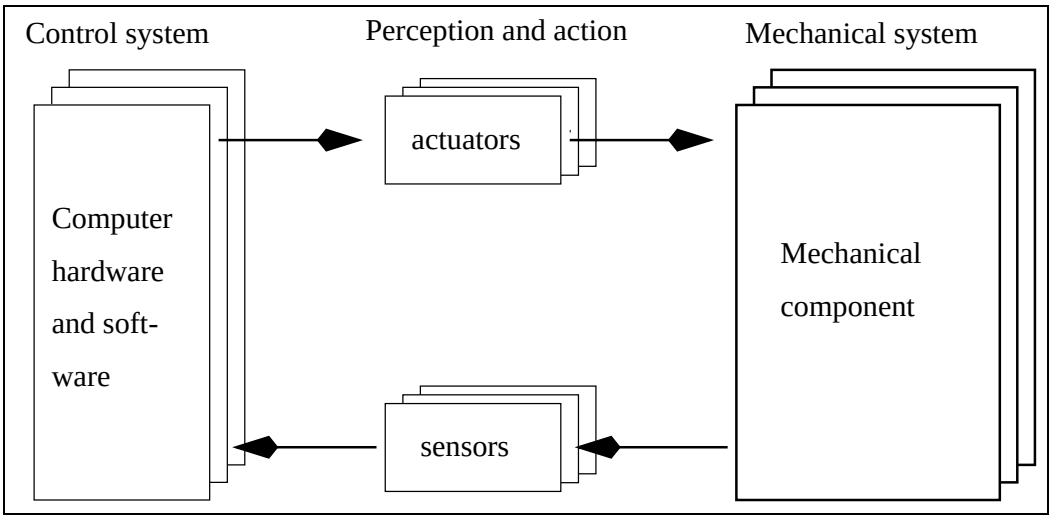

Figure 1: A general mechatronic system

sive load of the algorithms. These distributed approaches of realizing the mechanical and the control system have presented some problems in themselves. Probably the most demanding of these problems has been the question how to interconnect all these distributed components, including the sensors and actuators, to form an unit as displayed in Figure 1.

This paper will describe a ring–structured communication system based on a deterministic channel access protocol. First of all the requirements of the system will be analyzed and compared to existing concepts. The next part of the article will outline the main architectural principles which have been followed in the design of **MERKUR** (**ME**chatronic **R**edundant **K**[C]ommunication **U**nder **R**eal–Time Conditions). Finally, the paper will take a look at the hardware structure of the implemented system.

2 Architectural characteristics of MERKUR

2.1 Requirements and existing concepts

As already mentioned, the fundamental processing work of the control stations consists of the following sequence. First of all, the actual state of the mechanical system is perceived by reading new values of the connected sensors, then the new actions that should take place are calculated and finally the new control values are transmitted via the actuators to the controlled system.

This general processing sequence is essential for mechatronic systems, but not exclusively found there. It requires a suitable communication system for transporting the input and output information of the control loop from respectively to the controlled

object. It is extremely important to avoid inconsistencies between the internal states of the control system and the controlled object. The control system has to respond to an external stimulus from the environment within an interval dictated by the object, called response time. This time must be guaranteed on the one hand by the control, on the other hand by the communication system. In a multitude of mechatronic applications typical sample rates are in the order of 1 to 100 milliseconds. In addition, the communication system must support a kind of fault tolerance, timeliness, maintainability and extensibility.

In detail, the communication system has to offer the following criteria to satisfy the requirements of the mechatronic system:

- The number of participants, consisting of sensors, actuators and computer nodes, is typically ≤ 100.
- The distributed components reside within a radius of about 50 meters.
- The main load on the communication media is caused by periodic messages transporting sensor and actuator values.
- The sample rate of the control loops are in the range of 10 to 1000 Hz.
- A typical resolution of 16 bits for the data types has to be supported.
- Jitter in data transfer can only be tolerated when its order is much smaller than the sample period.
- Under certain circumstances, sporadic communication with predictable small latency should be possible. Such a situation can occur when switching to an emergency mode is necessary.
- Because of the critical environmental conditions, in which the communication system will operate, the system has to be insensitive to electromagnetic disturbance.
- Integration respectively disintegration of communication nodes during operation is a desired feature, but not of utmost importance.
- A simple interface to the application layer should allow an efficient access to the distributed data.

These requirements are the results of a study made by the interdisciplinary research project [14].

MERKUR isn't the first concept for a bus system interconnecting distributed systems. Many other concepts for so called fieldbusses like "PROFIBUS" [2, 5], "Interbus–S" [8, 3], "CAN" [13, 6], "ASI" [12], "SERCOS" [15] and the real–time network "SCRAMNET" [4] exist. All these implementations were examined and compared with regard to the important items mentioned above. It's far beyond the scope of this article to discuss this study in all its details here.

As a short conclusion, we can establish, that none of the existing fieldbus concepts accomplish all the mechatronic requirements listed above which are necessary for coupling multiple computer nodes and sensor/actuator units. Many of them can't guarantee the timing constraints because of too small transmission rates of the communication medium or too much overhead in the communication protocol. An exception represents the "SCRAMNET" implementation, whose design philosophy is to realize a distributed real–time simulation environment consisting of multiple workstations. This design allows a node delay of 1μsecs. The actual concept of MERKUR combines many aspects found in the different realizations and some new ones especially with respect to a fast, deterministic and reliable communication system.

2.2 Topology and transmission medium

The node interconnection topology is perhaps the most common way to describe local network architectures. The MERKUR net topology can be characterized as a ring structure. Each node possesses 4 ports. The two neighbors are connected via 2 ports to this node (Figure 2). This structure has been chosen for the following reasons.

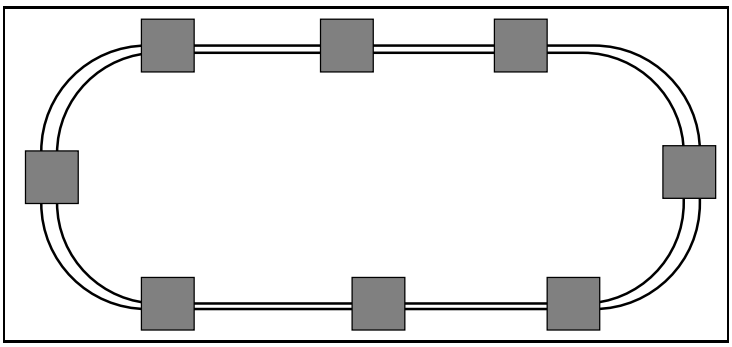

Figure 2: MERKUR net topology

First of all, the choice of the physical medium influences the whole system design. Due to the increasing importance of optical fibers, MERKUR is based on a low cost plastic optical fiber. This fiber is the only supported transmission medium. By employing this fiber, we can profit by different advantages: fibers minimize the electromagnetic disturbance caused by the environment. Also the aspired high bandwidth of 10 to 50 MBit/sec can be realized with few effort over a length of about 20 meters.
One disadvantage of optical fibers is manifested in the fact that only point–to–point connections between two participants can be built with low expense. For that reason, the chosen topology of the communication system is a ring structure as found in many other systems based on optoelectronic transmission techniques.

2.3 Deterministic information distribution

Each real–time system has to provide the specified timely service to its environment (see section 2.1). To meet this requirement, two fundamentally different methods exist: the event–triggered approach and the time–triggered approach [10].

In an event driven system, a significant event in the environment or in the computer triggers the start of a corresponding system action — for example the activation of a special task on a node. In such systems the communication happens only on demand.

A time driven system is characterized by the fact, that the moment when a particular message is passed over the communication system is predefined and therefore known a priori. Because of this property, the system behavior concerning the information transport is totally fixed. Data transmission happens permanently, even when no task needs the data at the moment.

These two main concepts are compared comprehensively by Kopetz [10, 11]. By analyzing the desired requirements of the mechatronic communication architecture as described in section 2.1 and by comparing them with the theses in [10], the only possible realization consists of a time based system architecture. The following considerations will emphasize the decision for a time orientated concept.

To satisfy the demand of timeliness in all imaginable situations like peak load, the system performance must not degrade with variations in the frequency of external stimuli or due to message congestion on the real–time bus [9]. The medium access delay time of the bus must be independent of the communication traffic on it. To realize and especially to prove this behavior by event–triggered concepts is much more complicated than by time–triggered approaches. Therefore in MERKUR a TDMA (Time Division Multiple Access) strategy provides a deterministic, load independent and collision free procedure for medium access like other existing real–time busses designed for special purposes as MARS [9] or SERCOS [15].

2.4 Data communication

Distributing information over the MERKUR ring is both simple and fast. The engineer of the control system software only has to know, that the system–wide common data is represented as a contiguous dataset. The application program can access this dataset by linking a start address to the beginning of the shared–memory block. The operating system can assist this access by a special service routine.

Once this is accomplished, data communication can take place. Each time a new variable value is written to the shared–memory, it is automatically updated in all other nodes on the network ring.

Refer to Figure 3 for an overview of the just mentioned process. The CPU

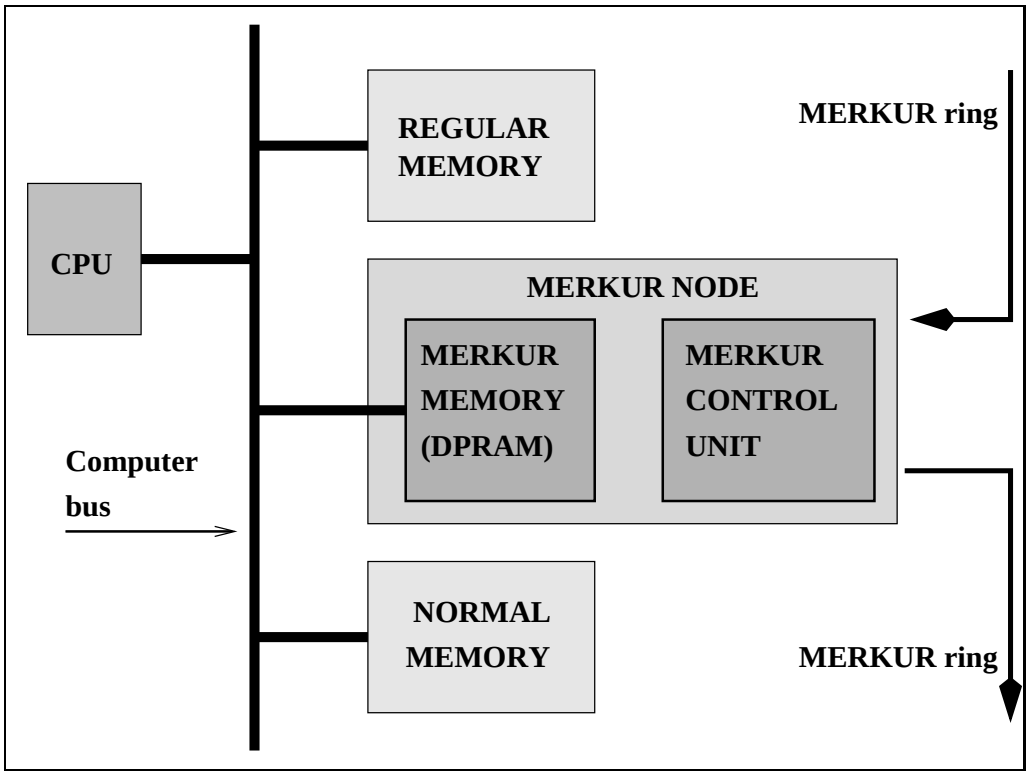

Figure 3: MERKUR data access

"writes" a dataword of 16 bit width to a memory location, which physically resides on the MERKUR card, by simply executing an assignment statement like *speed = actual_speed* or a system call such as *WriteDataValue(Speed, actual_speed)*. This formulation is well known from most high–level programming languages. The MERKUR memory, also named real–time data base, "looks" to the CPU as all other memory existing in the computer system, because it is mapped in the normal address space of the host computer.

All the other work concerning the transmission of the data is managed transparently by the MERKUR electronics. Every other component connected to the ring places the 16-bit message automatically in its own local memory at the same relative physical address as the producer node during the next communication cycle. This cycle is started periodically.

The key features of this technique are the simplicity, the speed of data exchange, the access at any time and the unified and easy structure of the software.

2.5 Aperiodic communication

Beside the normal data traffic over the communication channel, a mechanism that handles reactions on sporadic, external events and exceptions is required. For treat-

ing this type of message, the basic tool of the system engineer is to utilize hardware interrupts. Because of the decentralized system concept, these interrupts must be communicated among nodes efficiently, fast, and "deterministically".

This fact is also taken into consideration by the design of MERKUR. Every node has the possibility to generate two different interrupts whose meanings can be defined freely by the application designer. In addition to this, all nodes have the ability to generate a so called failsafe interrupt, whose meaning can't be modified.

The failsafe interrupt is used, if a serious failure occurs, either in the control system or the controlled object. If such a situation happens, a further continuation of the operation of the object is not possible or does no longer make sense. The system must shut down and stop in a controlled predetermined manner. Similar to normal exceptions, the activation of failsafe can be done by every node integrated in the communication scheme.

Up to now only the possibility of generating aperiodic events has been discussed. The mechanism, how all these abilities are implemented in MERKUR, is described in section 2.7.

2.6 Fault tolerance

The ring topology naturally includes an uncertainty, because a break of one transmission line stops the complete information transfer of the entire system.

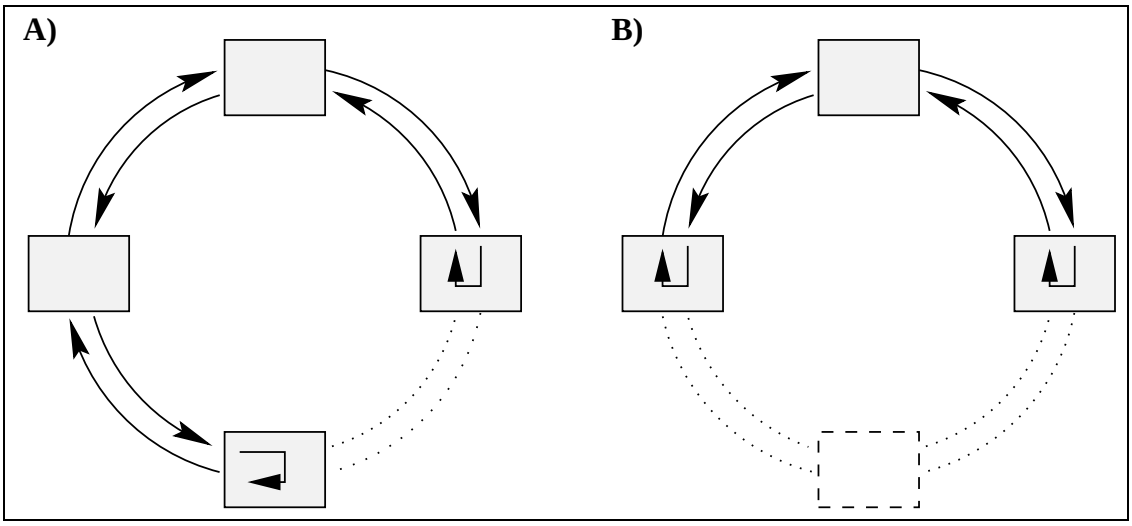

Figure 4: Failure situations: A) channel failure B) node failure

Therefore, the communication topology of MERKUR consists of an antiparallel duplex–ring (see Figure 2). This design feature covers permanent transmission line faults by using the secondary ring segment for bypassing the data flow at the error location as shown in Figure 4. With this method it is also possible to bypass

a defective node. When the ring is already operating in one of the two states illustrated in Figure 4, a repeated occurrence of a similar fault forces the system to enter a failsafe state. This step is absolutely necessary because this additional event would divide the remaining ring into two independent ring systems. Such a constellation does no longer represent a valid configuration.

Another fault situation occurs, when data transport fails because of mainly transient failures. This is the case, when a bit toggles falsely during communication. This kind of error is handled by the protocol mechanism.

2.7 Efficient protocol

In the protocol, driven on the communication system, each participant is represented at least by one 16–bit slot. These slots of the TDMA frame are arranged successively according to their slot number i. This number is also used by the MERKUR control unit as an offset for addressing the real–time data base (see section 2.4). The start of a communication cycle is marked by the header packet "CS" as illustrated in Figure 5.

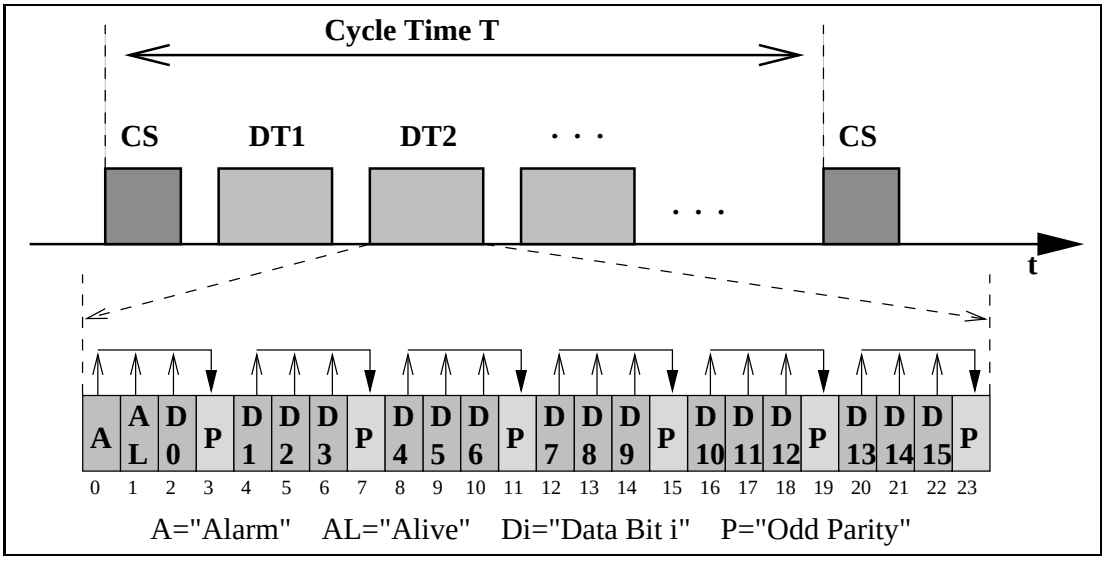

Figure 5: TDMA protocol of MERKUR: CS Communication synchronization slot, DTi Data telegram slot i

This form of protocol allows an implicit addressing of the different nodes inside one communication cycle whereby the efficiency of the protocol increases and the overhead of sending explicit addresses is avoided. When a message is detected faulty — the odd parity bits indicate this — it is rejected and replaced by the message of the following cycle. This is done as fast as possible and avoids overhead of message acknowledge and message repeating mechanisms.

Furthermore, every instant of the control system can be configured in such a way that it checks the actuators for being alive and operating correctly. The method for doing this is the "Alive"–bit in the data telegram. All bad actuators are not able to toggle this bit. For example this is the case, when an actuator is powered down.

Finally, the protocol has to handle reactions on sporadic, external events and exceptions. As mentioned in section 2.5, every node possesses the possibility to generate two universal and one failsafe interrupt. Their transmission is done by replacing the normal data packet. First of all, the "Alarm"–bit is set in the own slot. Then the alarm reason is encoded in the data field (D0 ...D15). Again, the address of the node generating the interrupt is given implicit. This method of overriding implies, that one data message gets lost. To avoid the loss of more than one message, these two different information types are alternated in worst case situations.

3 Node architecture

3.1 Types of participants

A mechatronic system is composed of a passive mechanical system and computer stations. These two parts are linked together by the sensors and actuators. The communication involves only the sensors, actuators and computers. Therefore, MERKUR offers two basic types of nodes. The first type is named "slave" and allows to connect interface components (sensors, actuators). Its functionality is limited to simple IO operation. A slave occupies exactly one slot of the TDMA frame (see Figure 5).

More than one slot can only be assigned to the so called "master" nodes which are part of the control system. Such a node is always realized as an extra board, mounted in a computer as shown in Figure 3. These master nodes are the second type of nodes.

3.2 Slave node

The slave node is arranged in multiple functional modules. One of the most important units is the shift register in the middle of Figure 6. Its function is to read or write values from or to the mechanical objects depending on the chosen operating mode of the node (sensor or actuator). This register reflects also the whole organization of the ring as a large distributed shift register. Every transmitted information passes this register unit. This method is one of several to realize medium access [7]. The slot counter supports the selection of the slot, which corresponds to the node, by comparing the actual counter value with the node address adjusted by DIP switches.

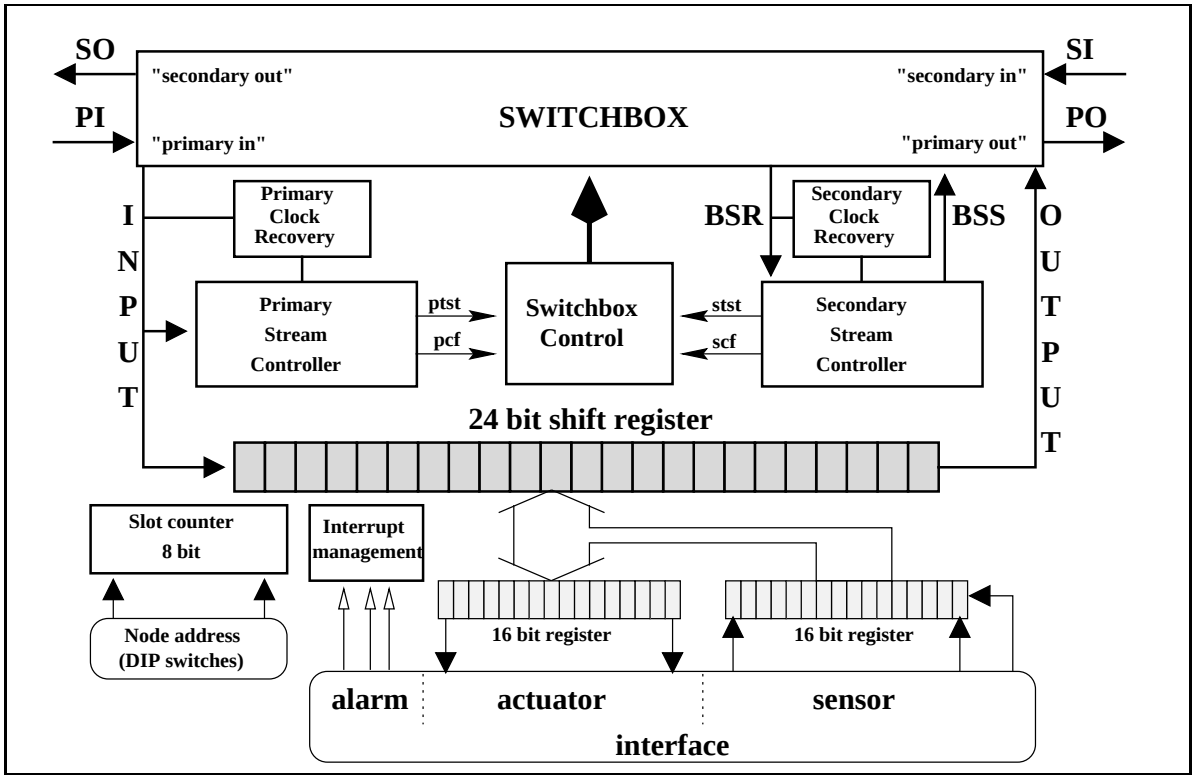

Figure 6: Functional block diagram of a slave node

Another essential part is the switchbox. Its task is to attach the physical inputs PI and SI (respectively outputs PO and SO) to the logical inputs $INPUT$ and BSR (respectively outputs $OUTPUT$ and BSS). The primary stream controller is utilized to supervise the logical input $INPUT$ for a channel failure (signal pcf) or a test signal ($ptst$), which serves for checking an already broken line. The same work is done by the secondary part for the input BSR (signals $stst$ and scf). Refer to Table 1 for an overview of the mapping process for all imaginable channel faults. In normal operating mode, the signals BSR and BSS are used to handle the data streams on the secondary ring segments.

channel failure type	$INPUT$	$OUTPUT$	BSR	BSS
all 4 ports ok	PI	PO	SI	SO
SO or PI defective	SI	PO	PI	SO
SI or PO defective	PI	SO	SI	PO

Table 1: Channel mapping from physical to logical ports of the switchbox: BSR bit stream receiver, BSS bit stream sender

This concept implements a local fault management for the data channels, because no central station is needed to handle such situations. Every node is checking and reacting for itself, but all master nodes are informed of the new ring configuration

by sending an alarm packet after the fault detection. This mechanism allows a fast rearrangement of the ring in the case of a line break. The detailed implementation is described in the masterthesis [1].

3.3 Master node

The master node integrates the general aspects of a slave with the extended functionality necessary for a master node. One of these features is the interconnection between MERKUR and the computer. For that reason, the master possesses an AT bus interface. The actual design is implemented for a personal computer (PC) as master station. The AT bus allows to access two different memory blocks on the MERKUR board. On the one hand, the normal data memory already described in section 2.4 is located on the master board. On the other hand, a special configuration memory is accessible.

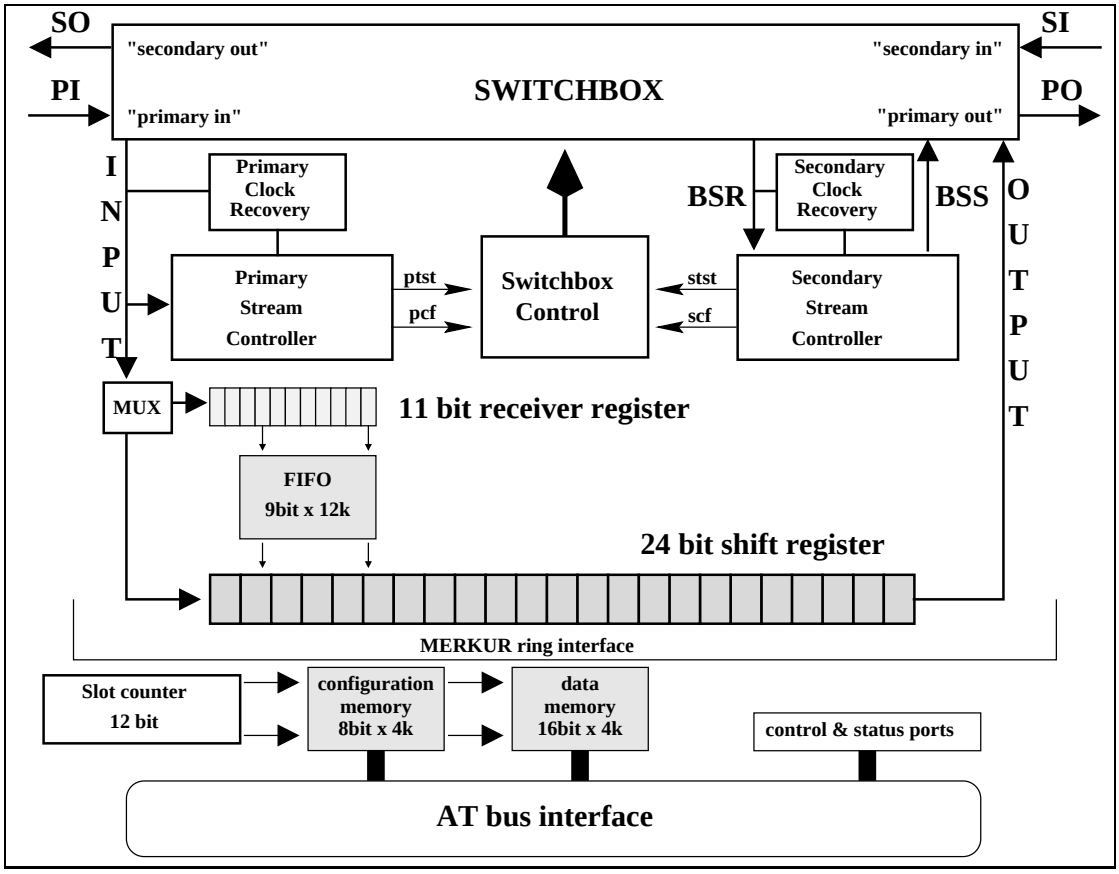

Figure 7: Functional block diagram of a master node

The meaning of that memory block is to describe every TDMA slot (maximum number is 4096) more precisely by one byte, so that the hardware knows how to handle every slot. The 8 bit width dataword encodes the following attributes of

one slot: the first bit marks the last valid slot of the whole frame, another one determines, whether the slot is an actuator, which must be served by the node. The third and fourth bit declare, if the "Alive" bit is enabled or ignored and if a slot belongs to the local or a foreign host.

To evaluate the contents of this SRAM–based configuration memory for each slot during execution, it is necessary to read the value at the beginning of the outgoing message slot. Before the ring is started by setting a bit in the control port, it must be guaranteed, that the configuration memory is initialized correctly. A slot counter is used for addressing both memory types.

As illustrated in the Figures 6 and 7, every node uses a clock recovery unit to extract the clock from the data stream. The ring structure must possess one node, which generates this clock base. Exactly one master must be selected by setting a bit in the control port to do this job. Such a master is named "special master".

As mentioned above, a master can also possess more than one slot. In a system with n nodes the shift registers can only store n datawords. A FIFO buffer is activated exclusively by the special master to hold the rest of the distributed information. This FIFO also compensates minimal timing discrepancies between the incoming and outgoing data stream clocks.

Finally, the special master initiates also the operation of the secondary ring. This work includes the generation of a special test pattern for the output BSS (bit stream sender), the corresponding clock base and the evaluation of the received pattern at port BSR (bit stream receiver). During normal operation (no channel faults exist), all nodes communicate over the primary ring segments and the secondary part is only tested for a fault.

4 Conclusion

The design of distributed mechatronic systems requires a suitable communication system. In this paper a real–time interconnection concept has been presented, that provides all services needed to distribute the necessary information.

For proving the correct operation of the described ideas in practice, the MERKUR system has been implemented as a prototype consisting of two slave modules and one master node. The last mentioned one is realized as a PC based system. The aspired bandwidth of 20 MBit/sec is not reached at the moment. Actually a transmission rate of 4 MBit/sec is working. This lower rate is caused by the applied fiber optic transmission devices and the provisional board layout used to implement the prototypes.

The whole interpretation of the communication protocol is done by hardware without a microcontroller. The slave modules consist of two CPLDs (complex programmable logic devices). The master uses five of these chips. A further improvement in system

behavior is expected, when the message protocol format is modified in such a way, that the encoding of the aperiodic information is separated from the normal data. This can be done by extending the slot format with several additional bits. This extension avoids the loss of one data message in the case of an alarm.

Beside the hardware, a run–time system is needed to allow a unified integration of the application software. This run–time system serves as an interface between the hardware and the user software. The architecture of the run–time software is developed at the moment. Results of this design process will be presented in following papers.

5 Remarks

The work presented is sponsored by the Volkswagen–Stiftung, Hannover, Germany, as part of the interdisciplinary research project "Integration of distributed mechatronic systems with special regard to real–time behavior".

References

[1] Wolfgang Baldauf. Implementierung eines echtzeitfähigen und fehlertoleranten Sensor/Aktor Kommunikationssystems für mechatronische Anwendungen. Diplomarbeit, Lehrstuhl für Prozeßrechner, Technische Universität München, December 1994.

[2] Klaus Bender, editor. *Profibus. Der Feldbus für die Automation.* Hanser, 1990.

[3] R. Bent. Interbus–S: Offene Kommunikation für Sensoren/Aktoren. Technical description, Phoenix Contact, Blomberg, Germany, 1992.

[4] Tom Bohman. Shared–memory computing architectures for real–time simulation — simplicity and elegance. Technical description, SYSTRAN Corporation, Dayton, Ohio, 1993.

[5] J. Ehrenberg, E.-J. Heins, P. Leymann, and W. Schumacher. Automatisierungspraxis: Profibus Anwendungen, Produkte, Trends. *Elektronik plus*, 1, 1993.

[6] Konrad Etschberger, editor. *CAN. Controller–Area–Network. Grundlagen, Protokolle, Bausteine, Anwendungen.* Hanser, 1994.

[7] Georg Färber. Feldbus–Technik heute und morgen. *Automatisierungstechnische Praxis*, 36, 1994.

[8] Bernhard Jünger. Profibus contra Interbus–S. Ein aktueller Vergleich. *Elektronik*, 21, 1994.

[9] H. Kopetz, A. Damm, Ch. Koza, M. Mulazzani, W. Schwabl, Ch. Senft, and R. Zainlinger. Distributed fault–tolerant real–time systems: The MARS approach. Research Report Nr. 4/88, Institut für technische Informatik, Technical University of Vienna, 1988.

[10] Hermann Kopetz. Event–triggered versus time–triggered real–time systems. Research Report Nr. 8/91, Institut für technische Informatik, Technical University of Vienna, 1991.

[11] Hermann Kopetz. Should responsive systems be event–triggered or time–triggered? *IEICE Trans. on Electronics, Inst. of Electronics, Information and Comm. Engineers, Tokyo Japan*, E76–C(11), November 1993.

[12] Werner Kriesel and Otto W. Madelung, editors. *ASI. Das Aktuator–Sensor–Interface für die Automation.* Hanser, 1994.

[13] Wolfhard Lawrenz, editor. *CAN. Controller–Area–Network. Grundlagen und Praxis.* Hüthig, 1994.

[14] J. Richert, A. Rükgauer, U. Petersen, V. Hadwich, T. Raste, K. Gresser, and J. Pfefferl. Integration verteilter Systeme der Mechatronik mit besonderer Berücksichtigung des Echtzeitverhaltens. Interdisciplinary Research Report Az.: I/67975-9, Uni–GH Paderborn, Fachbereich 10 Automatisierungstechnik Prof. Dr.-Ing. J. Lückel, June 1994.

[15] Heribert Winkler. SERCOS Interface auf dem Weg zum Standard. *Elektronik*, 6:116–124, 1991.

Dipl.–Ing. Pfefferl Johann
Lehrstuhl für Prozeßrechner, Prof. Dr.–Ing. G. Färber
Technische Universität München
Arcisstr. 21, 80333 München, Germany
Tel +49–89–2105 3557
Fax +49–89–2105 3555
Email: pfefferl@lpr.e–technik.tu–muenchen.de