Extension Module for Application-Specific Hardware on the Minirobot Khepera

J.-C. Niemann, U. Witkowski, M. Porrmann, U. Rückert

System and Circuit Technology, Heinz Nixdorf Institute,
Paderborn University, Fürstenallee 11, D-33102 Paderborn

E-mail: niemann@hni.uni-paderborn.de

Abstract. In this paper a hardware extension for the minirobot Khepera is presented. We are researching into algorithms for neural data processing and their hardware implementation by increasingly using the minirobot Khepera. This robot is supposed to record its environment continuously and to map obstacles. This is the basis for the implementation of an efficient navigational system. As the algorithms in question are complex and since the microcontroller on the Khepera does not possess the necessary processing capacity, it is requisite to develop a discrete hardware module that extends the Khepera.
An expansion module with 20 MHz clock rate, 1 MB memory and a XILINX-FPGA with 1,600 CLBs will be presented. This module is especially designed for its flexible use to make the dealing with multiple tasks possible. In order to evaluate the algorithms it is an advantage to edit the generated data and to visualize it. Consequently, the specification of an RS-232 interface is carried out in the hardware description language VHDL. This interface makes it possible to send the results that are generated by the Khepera to a personal computer and to forward commands to the robot.
The user interface is a program designed for the Microsoft operating systems Windows 95/98. This facilitates a user-friendly recording of all data, the archiving, documentation and visualisation of the results. A description of how the system works and of the results of those processes concludes the paper.

1 Introduction

The exploration of space, land, or, especially, hazardous environments is a task for which robots are ideally suited. This implies that the machines must be capable of intelligent behaviour. Based on their perceptions, they have to make decisions autonomously in every situation [1]. The computational requirements for navigation and additional, sometimes even more complex tasks are extremely high and, in most cases, demand real time processing. These demands cannot be fulfilled by the basic Khepera module. In order to solve the problem posed by the Khepera's weakness in computational power, it is necessary to implement an optimized application-specific hardware. This module should be flexible enough to map certain classes of different navigational and behavioral strategies. Furthermore, it must have a low power consumption to allow an acceptable lifetime of the battery-driven robot.

In this paper, we demonstrate the implementation of an FPGA module on the minirobot Khepera. First, the basic hardware configuration of the minirobot is described. In section 2 the developed hardware module is characterized and the functionality of the FPGA board is explained. As power consumption is a very important aspect in connection with autonomous robots, the different hardware blocks are analyzed with regard to their current consumption in section 3. The debug features for testing new algorithms on the hardware by using a serial link are also presented. Section 4 gives an overview of the hardware features including visualization software.

2 Information processing on the minirobot Khepera

One application of the minirobot Khepera is its use as a mobile autonomous platform for the exploration of its immediate environment. The Khepera, in its basic configuration, consists of two printed circuit boards each measuring 55 mm in diameter. The bottom PCB contains the rechargeable NiCd batteries and two DC motors with incremental encoders. Eight infrared sensors are implemented for obstacle recognition (six in the front and two at the back). The core component of the second printed circuit board is a Motorola MC68331 microcontroller, running at 16.7 MHz and possessing 256 Kbyte RAM as well as a maximum of 256 Kbyte ROM. The K-Extension bus opens up a wide variety of modular expansion options. Figure 1 shows the Khepera in its basic configuration.

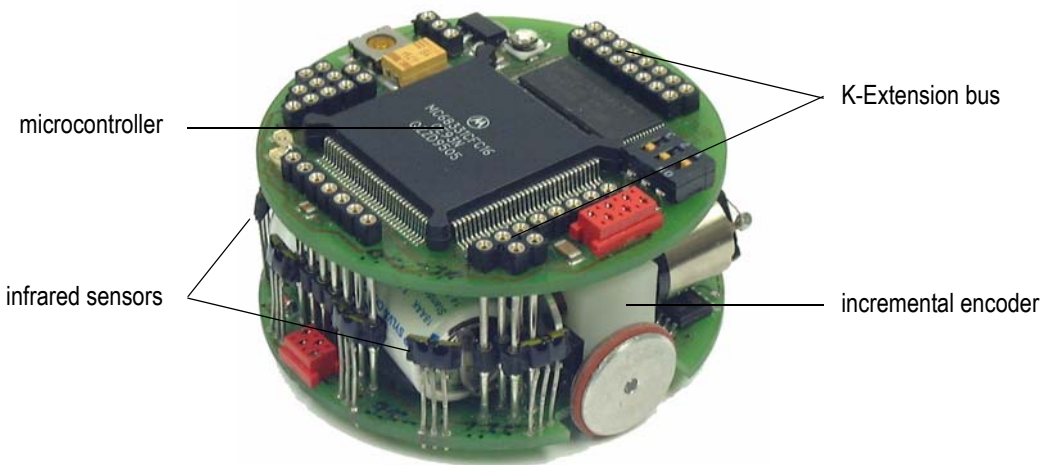

Figure 1. Basic Khepera configuration

This basic configuration allows experiments such as obstacle avoidance, wall following, target searching and collective behaviour. In this study, the robot is intended to solve a general navigation problem. After an analysis of the existing hardware it becomes apparent that the given facilities are not sufficient for a complex navigation task.

The inevitable slipping of the Khepera wheels during an exploration causes an accumulation of errors in its absolute position calculation. This systematic problem cannot be solved because of its undeterminable occurrence and severity. A further aspect is the restricted computational capability of the microcontroller because it does not meet the actual requirements imposed by the implemented algorithms described in chapter 2.2. In order to solve this dilemma the existing hardware is improved by two additional turrets, which are connected to the Khepera via the K-Extension bus (cf. chapter 3).

2.1 Environment exploration by the minirobot

Due to the limited range of the Khepera's infrared sensors it is a little complicated for the robot to globally picture its surroundings. One solution to cope with this problem is the installation of landmarks in the environment that can be clearly indentified by the robot. In [3] a light source is used as a global reference point. By finding this light the robot is able to calibrate its position every time it reaches the light. Every time the robot is out of the range of the light source it navigates on the basis of internal incremental motor encoders. Thus, the actual position and the internal position representation of the robot diverge over time. After some time of driving around in the environment there is a significant position error and the robot may be unable to find the light source again.

One possibility to avoid the robot's disorientation is to use an internal representation of the surroundings by learning a map that incorporates walls, obstacles and points of special interest such as the robot's home base. Consequently, we implemented a strategy for the learning of a map of the environment by using a global positioning scheme that enables the robot to know its position and orientation. As auxiliary devices we use four beacons that emit a pulsed infrared light scheme. This light is detected by the robot so that it will calculate the corresponding position by analyzing the light pattern. Figure 3 depicts the arrangement of the beacons in the environment.

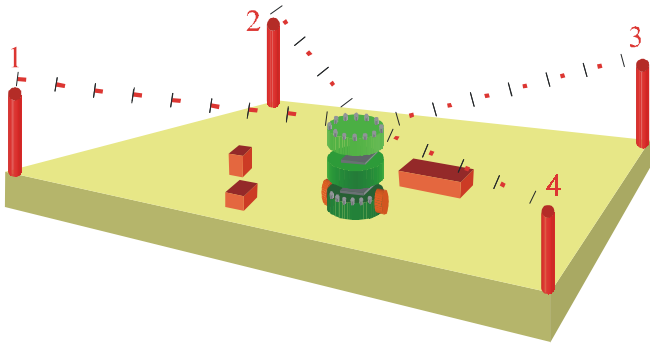

Figure 2. Schematic environment with four beacons sending a pulsed light scheme

For the detection of the beacon light, the Khepera uses an additional module with an infrared array of 32 circularly arranged photo transistors. This equips the robot with a 360° view and enables it to observe all light sources in parallel. The hardware of the infrared array module is controlled by an FPGA module. This module analyzes the light pattern and provides the Khepera basis module with the position information.

2.2 Beacon detection algorithm

Every 91.08 ms each beacon emits a non-overlapping light impulse with a length of 409.6 µs except of beacon number one, which emits a pulse of 812.2 µs to allow a global absolute orientation in the environment. During the sending period we have an emission break of about 87.8 ms. In this break, the near-range infrared sensors of the basic Khepera module are analyzed by the microprocessor without any disturbance from the global infrared transmitters. After the signals of the four beacons have been recorded, the center of maximum luminous intensity is calculated with respect to each beacon. This results in four angles, that are then composed to one four-dimensional vector. For determining the position and the orientation of the robot, two self-organizing feature maps [6] are used: One is used for position calculation and one for orientation detection. For mapping the environment a third self-organzing map of 128×128 neurons is being trained continuously by the position data, the orientation data and data from six proximity sensors placed in the front of the robot. This sensor data is combined with a small data set of six training vectors to train the map by using an on-line learning algorithm. During the robot's activities in its environment an internal representation is being formed by the self-organizing map, and the robot is receiving information about the position of obstacles and walls. While the self-organizing map calculations are carried out, the microcontroller shows a lack of performance. However, we can achieve real-time capability by the hardware implementation of the required algorithms. The learning of the map is marked by the recall in the self-organizing map algorithm. At a clock frequency of 20 MHz, about 820 µs are needed for a best-match search for one vector. In every clock cycle, two vector components are read from the RAM memory. The time for the vector update depends on the neighbourhood width, i.e. the number

of weight vectors being updated in the environment of the best-match element. As we use a small neighbourhood width of 16 or smaller the maximum update time is 130 µs. For learning the proximity sensor data, which forms a training vector set for one position, we need a maximum time of 5.7 ms. This offers enough calculation resources to enlarge the maps for the environment learning in order to increase the resolution of the internal representation.

3 Application-specific hardware

As mentioned in chapter 2, it was essential to improve the Khepera's global navigation capabilities. This is why we designed an IR-sensor turret with 32 infrared photo transistors that are able to receive IR-signals from transmission beacons being far more than 1 m away. This new module can receive signals from all sides without any movement of the robot. With this additional module, the Khepera can determine its global position in a given scenario with the respective beacon environment with a precision that allows an error of less than ± 15 mm and an average error in its angular positioning under $\pm 12°$. The processing of the incoming data from the IR sensors needs a much higher computational power than the microcontroller of the Khepera can provide. Likewise, the memory needs to be enlarged because we want to store the whole environment map on the robot itself. So we decided to design an additional module that fulfils our requirements in regard to computational power and, at the same time, memory and reduces the traffic on the Khepera's data bus. The class of algorithms that are to process the data allows parallel computing. This leads to the conclusion that an FPGA (Field Programmable Gate Array) solution as core component will outrange the capabilities of general purpose CPUs in our field of interest. Figure 3 shows the Khepera with the additional turrets needed for the algorithms for navigation and exploration tasks.

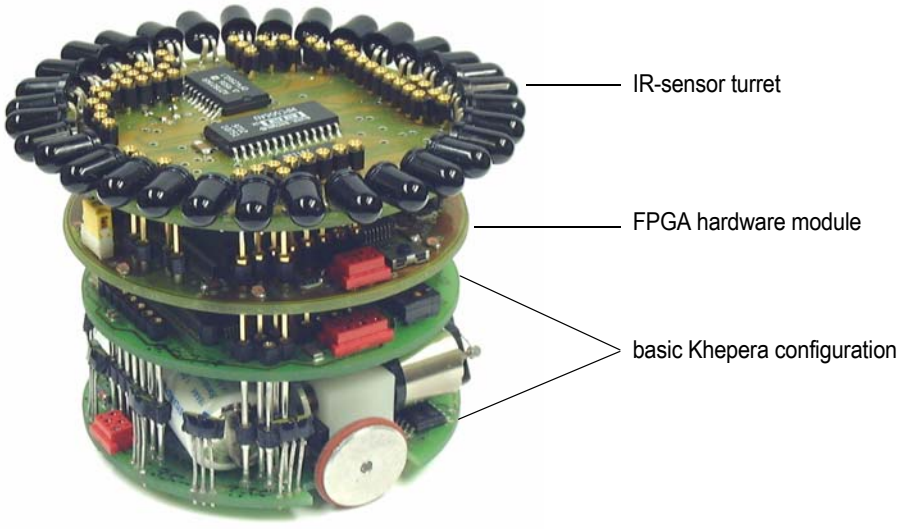

Figure 3. The Khepera with two additional turrets for navigation and exploration tasks

3.1 The FPGA hardware module

We have designed an FPGA hardware module, the *K-Flex-FPGA-Board*. This additional turret is applicable for various tasks and is described below. The intention behind the design of the K-Flex-Board was to provide a multi-functional and easy-to-use hardware extension for the Khepera.

The main features of this board are:

- 20 MHz system clock (easy to modify, if necessary)
- XILINX FPGA (XC4044XLA-9 HQ160) (1600 CLBs)
- 512 KByte SRAM (256K×16-bit)
- 512 KByte flash EPROM (512K×8-bit)
  (128K×8-bit used for FPGA configuration)
- K-Extension-bus-compatible
- 24 additional and freely programmable TTL-compatible I/O-ports
- 1 push button, 1 switch for freely programmable manual interference
- 9 freely programmable sub-miniature LEDs

We have implemented an XC4044XLA Xilinx FPGA as the computational core. This FPGA type is marked by its low power consumption (40% less than a XILINX XL type) and thus ideally serves for a mobile use. Additionally, it possesses a high number of I/O-pins (129), which make a versatile use possible. This FPGA has 1600 CLBs that facilitate large VHDL designs. The SRAM serves as a quickly addressable volatile memory, which the implemented algorithms can use as a data and provisional-result storage medium. As for many calculations an accuracy of eight bits is not sufficient, we use a bandwidth of 16 bits. This increases the rate of data transfer and, in comparison to an 8-bit access, halves the time needed for the data transfer. The integrated flash EPROM is needed to configure the FPGA and to store non-volatile data either before or during the operation of the robot. Based on the fact that this device is on a socket, it can easily be substituted by an equivalent flash EPROM with different data or configuration data. A special feature is that our design allows an on-line reconfiguration of the FPGA, which can become necessary because of the results of the implemented algorithms. This opens up the possibility for on-line learning and, to a small degree, for an assimilation of the hardware in response to its environment. Our board is fully compatible with the K-Extension bus of the Khepera, which allows a direct communication with the microcontroller and makes it possible to access all significant signals of the Khepera. We have implemented 24 additional I/O ports to enhance the possibilities of interaction with other modules or its environment. The push button and the switch are intended to enable the user to manipulate the hardware. Through nine freely programmable LEDs (of which eight are symmetrically arranged round the border) the hardware is able to give detailed status information to the user.

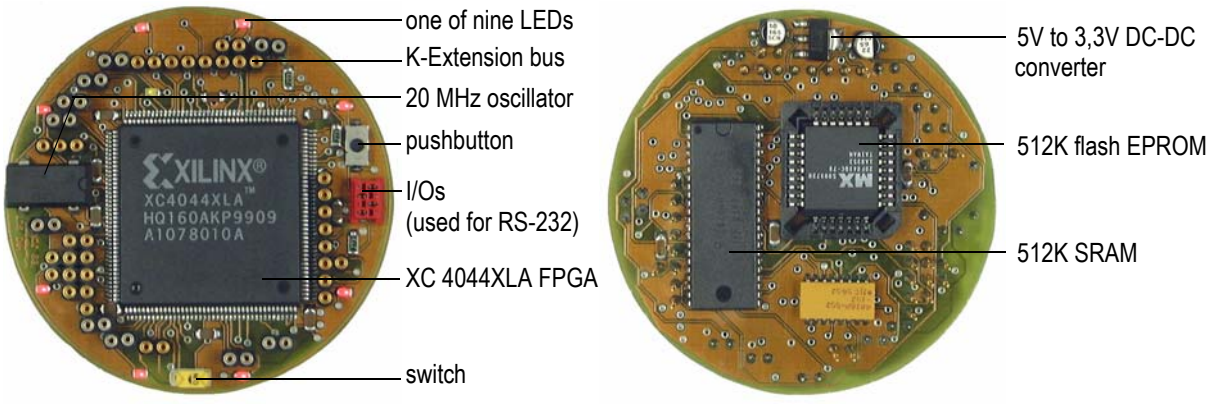

Figure 4. Top and bottom view of the K-Flex-FPGA-Board

In consequence of its high flexibility, this board has proved its capabilities as a functional hardware expansion in several other projects. Figure 4 shows the K-Flex-FPGA-Board in detail.

We have realized this board as a 4-layer PCB with the outer layers being power and ground planes. This helps to reduce electromagnetic interference with other turrets since the inner signal layers of our board are shielded by these planes.

As we mentioned before, the power resources of an autonomous mobile robot are limited. Consequently, we took care to keep the power consumption of our additional hardware as low as possible. Figure 5 highlights the main power dissipators of our board. This diagram is based on the data sheets and the system design and has been verified by diverse measurements. For this presentation, the implemented VHDL design is a stand alone serial link communication module, which will be described in chapter 3.3.

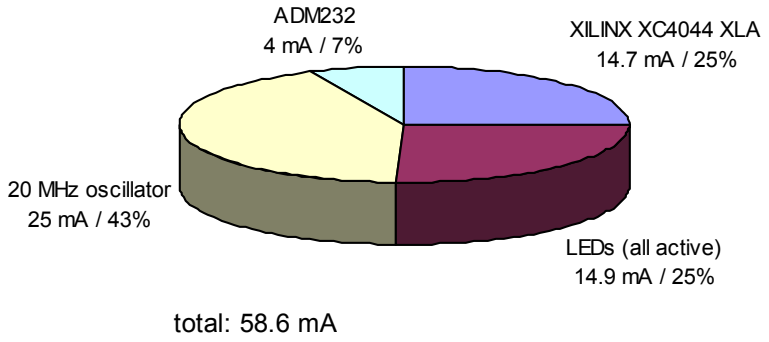

Figure 5. Power Consumption of the K-Flex-FPGA-Board with serial link

As shown above, the total power consumption is 58.6 mA, which is nearly a quarter of the Khepera's power consumption, which approximately amounts to 200 mA during normal operation. As the implemented VHDL design is noticeably small (only 3.7% of the CLBs are used, cf. chapter 3.3), the FPGA consumes less than a quarter of the total power. This changes when the implemented VHDL design becomes more complex, as we will see in chapter 4. The oscillator consumes almost half of the total power even though we used an economic one. This, obviously, hints at an important demand on future designs: the use of even more economic devices. The ADM232 is a RS-232 interface, which establishes the serial link communication with a PC and needs just 7% of the total power. Hence, the a power consumption of 293 mW reduces the operating time but seems to be justifiable.

3.2 System architecture

The minirobot Khepera constitutes the basis for the hardware implementation. Main components of the system architecture are the Khepera basis module and an add-on FPGA module. All additional hardware is coupled to the microcontroller of the Khepera basis module by the K-Bus [5].

Figure 6 gives an overview of the hardware architecture used on the minirobot. The FPGA-module controls the data transfer to the microcontroller and holds all hardware resources that are needed for the preprocessing and the hardware implementation of the self-organizing map algorithm. The external RAM on the FPGA module holds the weights of the continuously learned self-organizing map. The flash memory stores – in a non-volatile way – the configuration of the FPGA as well as the control parameters and the off-line trained self-organizing maps for the position recall. The advantages of the proposed architecture are its flexibility in case of changes of the hardware implementation and the possibility of decoupling

the basic module including the microcontroller and the additional hardware. The hardware implementation of the algorithms works independently, only the bus for communication with the microcontroller is shared. At the K-Bus, a small bandwidth is needed, because there have to be transfered data from the six 8-bit proximity sensors and two motor control commands every 91.08 ms only. On request of the microcontroller, information on the actual position and orientation as well as on some details of the environment is transmitted. Despite the additional functionality of the minirobot, the microcontroller does not operate to its full capacity and is therefore capable of performing other tasks.

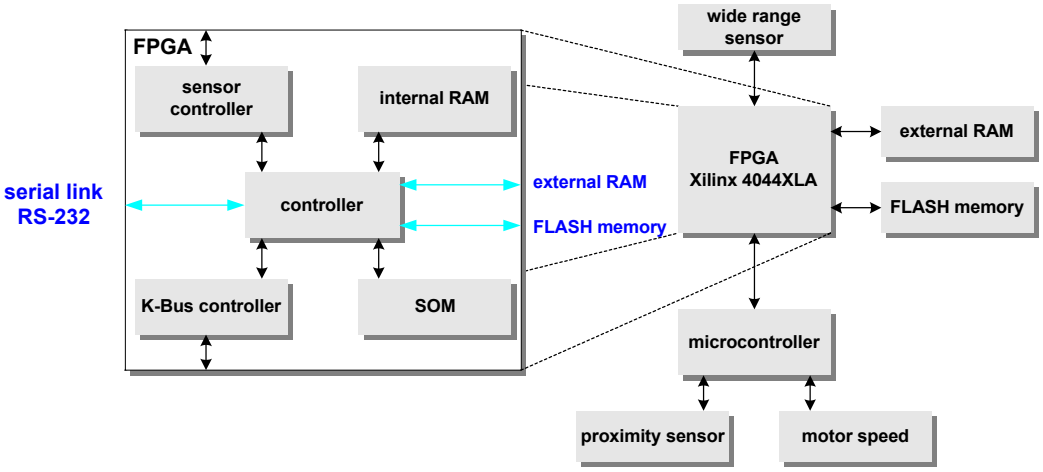

Figure 6. Architecture of the minirobot system with FPGA and infrared array module

3.3 Debugging features

For an easy-to-use debug mode and a robust and fast transmission of data, we considered it necessary to connect the K-Flex-FPGA-Board to a PC. This feature allows the visualization of the stored data and the results of the implemented algorithms. Therefore, we realized a special PC software (cf. chapter 4) and a communication module called K-Flex-Link. In order to avoid interference with the IR beacons, the transmission is carried out via wire. However, for other applications a wireless communication channel is desirable.

As seen in chapter 3.2, a data set comprises 64 KByte and has to be transferred continually to the PC. Because of the immense amount of data that has to be sent to the PC and because of the limited capabilities of the Khepera's serial link bandwidth (38,400 bps), we decided to implement a separate high speed serial communication link based on the RS-232 standard. This makes it possible to connect almost all PCs to the K-Flex-FPGA-Board without interfering with the operation of the Khepera microcontroller. In order to fulfil the demands on data bandwidth, we implemented an optimized and scalable RS-232 controller, which we use at a data rate of 115,200 bps. Other RS-232 conform data rates are easy to implement by a modification of one counter parameter in the K-Flex-Link VHDL design. The implemented protocol uses 1 startbit, eight bits of data, 1 stopbit and no parity. In this mode, we gain an effective data rate of 92,160 bps, which means that a set of data can be transmitted in intervals of six seconds.

We reduced the needed CLBs for the serial link to 59 (less than 3.7% of the FPGA resources). Consequently, additional designs can be larger than 1,500 CLBs. Figure 7 shows the interface of the K-Flex-Link serial communication module.

There are two registers that store the received data (RX_IN_REG) or the data that has to be transmitted (TX_OUT_REG). In order to realize a full duplex communication, we have to

separate internal clocks (tx_clk and rx_clk) with the RX clock being synchronized by the incoming data stream. Several control signals enable the top level design to supervise the serial link entity.

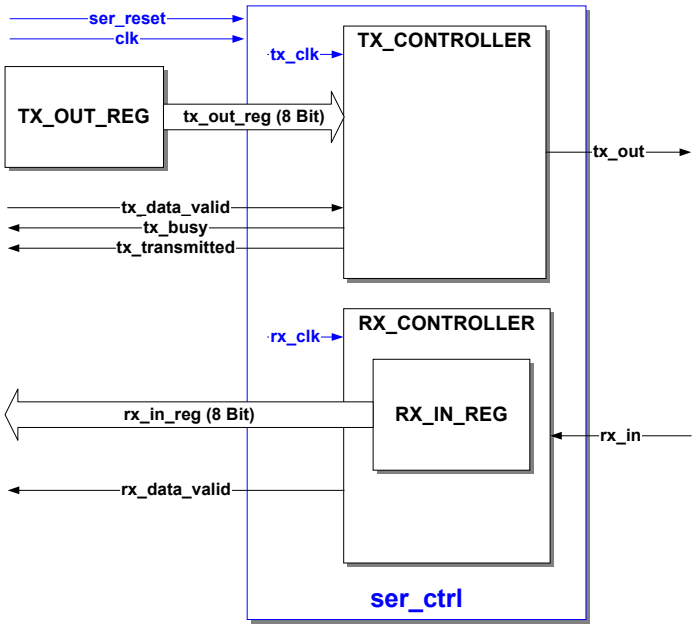

Figure 7. RS-232 serial communication link interface

In order to verify the implemented algorithms and to check the complete system we implemented the K-Flex-Beacon-Evaluation toolkit (cf. Figure 8).

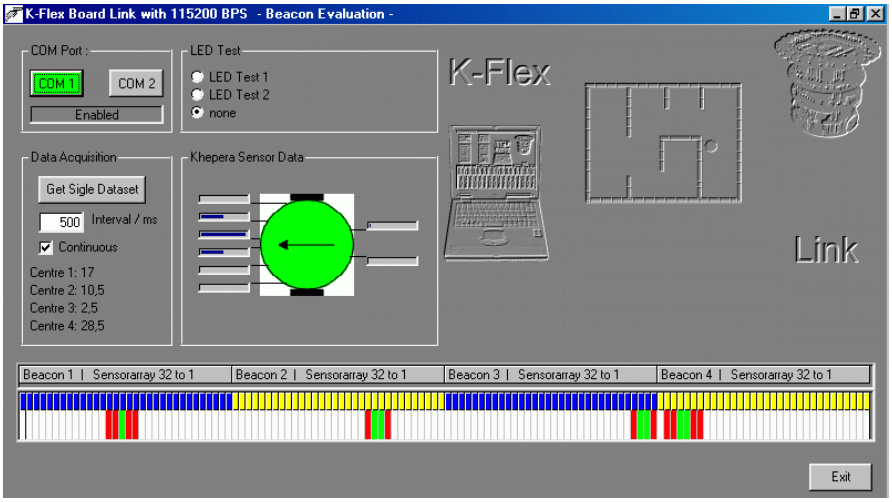

Figure 8. The K-Flex-Beacon-Evaluation toolkit

The Beacon Evaluation tool can send configuration commands to the K-Flex-FPGA-Board and visualizes the computed positions of the four beacons. The values of the eight Khepera IR sensors are displayed, and all received data is saved to a log file for further debugging. This leads us to the final application, the so called *K-Flex-Link*, which is described in chapter 4.

4 Results

The developed FPGA hardware has been used both for the calculation of the robot's absolute position and orientation as well as for learning the environment under real-time conditions. In order to record the learning results and mostly for debugging purposes, the PC-based tool *K-*

Flex-Link has been developed (see Figure 9). This tool allows the setup of the serial link connection and the dump of the map data that is generated during the on-line learning of the environment. The recorded data may be used for testing the functionality of the beacon analysis and for verifying the training algorithm.

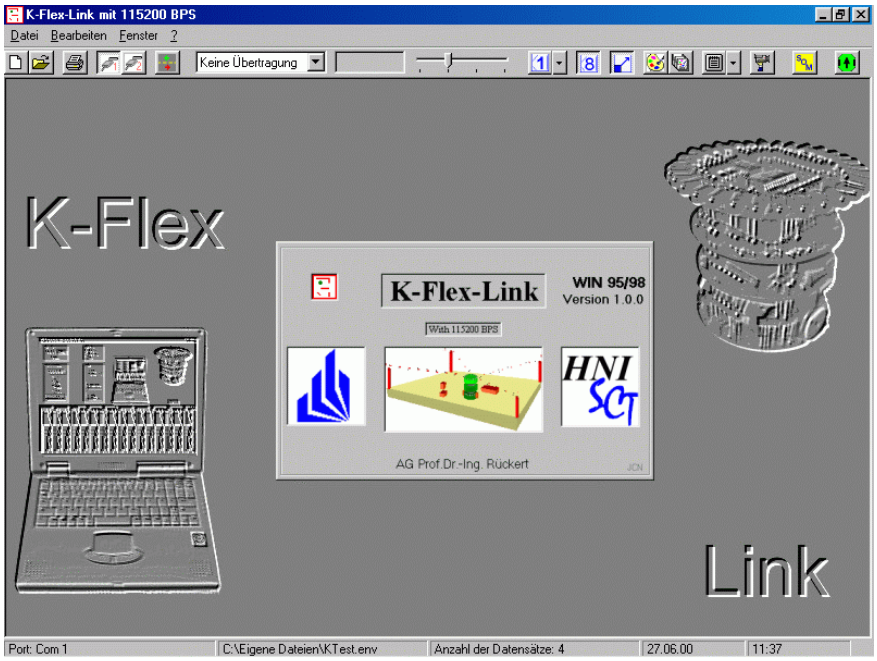

Figure 9. *K-Flex-Link* for recording and debugging the on-line learning of the environment

As the data is transmitted at a speed of 115.200 bps, its error-free handling by a PC is possible. It is, however, impossible to perform an on-line test of the training by transmitting the map data over the on-board serial link of the basic Khepera module. This link has a slower transmission speed and the microcontroller would be busy. Another software feature is the capability of recording map images periodically. This means that the representation of the environment learned by the self-organizing feature maps can be analyzed over time. This feature may be used to improve algorithms for the exploration of unknown environments or to test navigation algorithms.

Figure 10 gives an overview of the global power consumption of the hardware extension in different running modes.

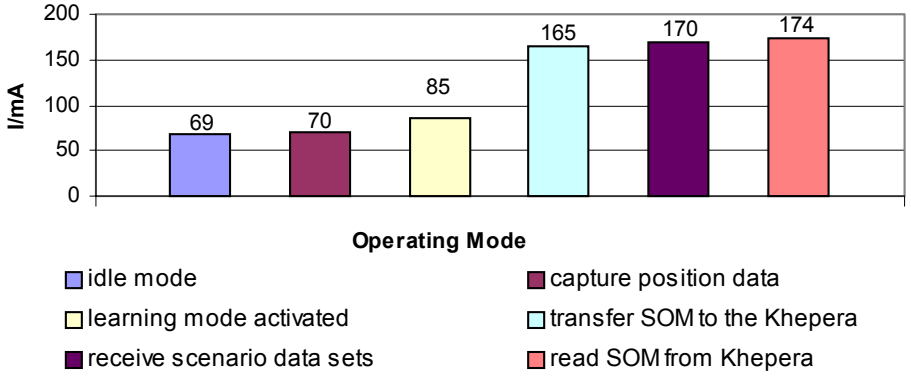

Figure 10. Power consumption of the two additional turrets during operation

In idle mode, no calculation is performed, which causes a low current consumption of 69 mA mainly due to the oscillator. When capturing the light pattern from the sensor array, the current increases slightly to 70 mA. By activating the learning mode on the FPGA, the current rises

moderately to 85 mA. This is the standard operation mode. When the hardware is being debugged and a transmission of data over the serial link is necessary, there is a significant increase of the current consumption to about 170 mA. However, this mode is not used when the robot acts completely autonomously without any cable. Consequently, we have an acceptable total power consumption of the additional FPGA hardware module.

5 Conclusion

In this paper, we have presented an FPGA hardware module for the minirobot Khepera. The module provides most flexibility for the implementation of various algorithms in order to enlarge the capabilities of the minirobot Khepera. Due to its modular structure the hardware can be integrated into the robot by plugging the FPGA module into the basic Khepera module. Together with additional I/O ports from the FPGA the K-Bus is connected to the top of the module. This allows the use of manifold auxiliary modules controlled by the FPGA and expands the number of applications that can be implemented on the minirobot.

One possible application is the global positioning algorithm that uses four beacons in the corners of the environment. A sensor array, which is controlled by the FPGA, detects the emitted light pattern. The recorded patterns are analyzed by a self-organizing map algorithm, which is also performed by the FPGA. By using this hardware, the calculations are fast enough to employ the results for controlling the robot.

Important features of the hardware module are its memory resources, the flexible programmability of the FPGA and, when using the serial link to a PC, the debug capability. The total power consumption of the robot is not increased significantly by adding the hardware. In future realisations of the board, a new FPGA series [7] may provide more computational power at a lower power consumption.

References

[1] **I.J. Cox, G.T. Wilfong**, Eds., Autonomous Robot Vehicles, Springer, 1990.

[2] **J.-C. Niemann**, Entwurf eines FPGA-Moduls mit Softwareumgebung für den Miniroboter Khepera, Universität Paderborn, 2000.

[3] **Löffler, A., Klahold, J., Rückert, U.**: "The Dynamical Nightwatch´s Problem Solved by the Autonomous Mini-Robot Khepera", Artificial Evolution: 3rd European Conference; selected papers, AE´97, Nimes, France, October 22-24, 1997, Springer 1998, Lecture Notes in Computer Science Vol. 1363, S. 303 - 313.

[4] **Witkowski, U., Heittmann, A., Rückert, U.**: "Implementation of Application-Specific Neural Hardware on the Minirobot Khepera", 1st International Khepera Workshop, Paderborn, Germany, December 10-11, 1999, S. 189-196.

[5] **K-Team S.A**., Khepera User Manual ver. 4.06, Lausanne, Switzerland, 1995. www.k-team.com.

[6] **T. Kohonen**, Self-Organizing Maps, Springer, 1995.

[7] **M. Porrmann, H. Kalte, U. Witkowski, J.-C. Niemann and U. Rückert**: "A Dynamically Reconfigurable Hardware Accelerator for Self-Organizing Feature Maps". Proceedings of The 5th World Multi-Conference on Systemics, Cybernetics and Informatics, SCI 2001, Volume 3, pages 242-247, Orlando, Florida, USA, July 2001.