APPLICATION OF CAN IN THE MODULAR CONTROL SYSTEM OF A WALKING ROBOT

Mladen Milushev, *Faculty of Mechanical Engineering, Technical University of Sofia*
Milen Spasov , *FDIBA, Technical University of Sofia*

Abstract - *The aim of this work is to provide for the communication between all microcontrollers within the open hardware structure. The control of the hexapod (the walking robot) is based on the layer principle, with one major-"master" and six minor - "slaves" microcontrollers used. The CAN-Protocol is based on the information exchange between all microcontrollers and is also used with the perspective for further development and equipping of the walking robot with other information processing devices. The paper contains descriptions of the system architecture and organization and the connections between the different processors and processes. It also contains descriptions of the target system, the RENESAS boards and the programming and debugging tools. Another task of importance is to design the algorithms and the main logic in a similar way and thus provide for the improvement of the robot. The paper presents an example for the configuration and implementation of the system aimed at achieving the desired results.*

1. INTRODUCTION

In the field of intelligent robotic systems, the current research has concentrated on the realization of adaptive control algorithms, the availability of training for improvement of the behavioral management, and on the adequate knowledge to present and interpret the sensory data, which in most cases are incomplete and noise-polluted.

The selection of adequate methods for solution of the subtasks in real-time is one of the critical points when developing the control behavior for a real robot.

From the methodology perspective the use of neural networks, fuzzy logic, genetic algorithms and an approach, based on predefined rules looks like applying promising tools to fix elementary behavioral control modes [1],[2],[3].

Other approaches use hierarchically organized control strategies [4] [5]. Since there are no general rules that describe precisely the spheres of application of these methods or their effectiveness compared to others, the choice of instruments was made by a process of trial and error techniques.

The hierarchical modular control architecture of BiMoR **(Bi**ologically **Mo**tivated **R**obot**).** is similar of walking machines [6] and [7]. A PC system at the top level of the hierarchy performs the main robotic tasks e.g. path planning, calculation of foot trajectories and communication with the environment (Man Machine Interface). On second level, RSKM16C29- microcontroller is connected with the PC system and is used for movement realization, sensor data acquisition and pre-computation. For the aim of execution the basic functions of legs like closed-loop joint control (valve control, recording the signals of joint encoders) are installed six EVBR8C22/23-microcontrollers. On base level, each sensor and actuator is connected with the interface board to the microcontroller board. The RSKM16C29 and EVBR8C22/23 (made by RENESAS) microcontrollers are installed in industrial controller boards and contain one Full CAN (ver.2.0B) module, which can transmit and receive messages in both standard (11-bit) ID and extended (29-bit) ID formats. Controller Area Network (CAN) bus is one of the Field bus control system type used in decentralization, intelligence and networking.

The CAN-controller from RENESAS which is integrated in microcontrollers is designed to provide reliable, error-free network communication for applications in which safety and real-time operation cannot be compromised. CAN is based on a "multiple master, multiple slave" topology. Its main attributes can be summarized as follows:

- High reliability and noise resistance
- Fewer connections
- Flexible architecture
- Error handling through peripherals
- Low wiring cost
- Scalability

2. TOPOLOGY OF THE CAN-BUS

CAN uses bus topology. Here all CAN-Nodes (CAN-Station) are connected with relatively short wires with the actual bus. (Figure 1.)

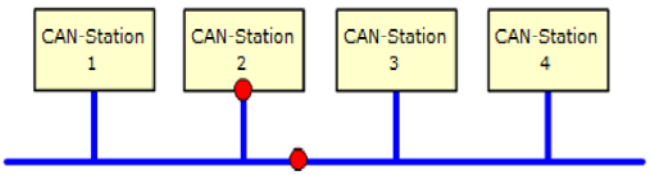

Fig. 1. Line topology at the CAN bus

This topology has the advantage, that if one CAN-Station fails, it has no direct influence on the CAN bus. All other CAN-Stations can still communicate without restriction (Figure 2.).

The main disadvantage of this topology is, that if the CAN bus is interrupted, then all of the CAN-Stations from the rest of the CAN bus will be unavailable. All other CAN-Stations remain functional to the point of interruption,

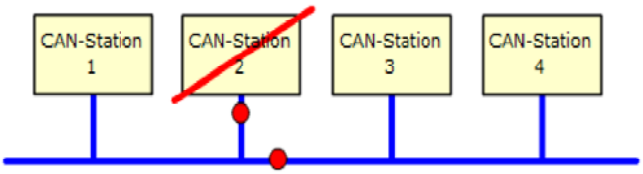

Fig. 2. Failure of the bus station

however, communication with CAN-Stations which are connected after the breakpoint is not possible. Therefore, functions that require data transmission after the breakpoint, are no longer available. (Figure 3)

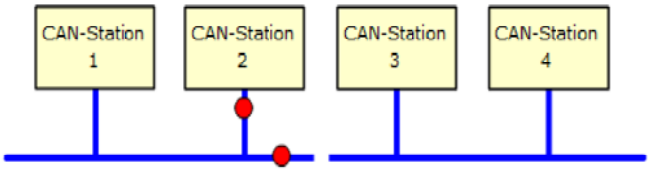

Fig. 3. Cut through the bus

The ISO 11898 standard specification defines a maximum bus length of 40 meters, a maximum branch length of 0.3 meters, and a maximum of 30 nodes. However, the robust design of the CAN bus physical layer allows the use of much longer cable lengths. As bus length increases, a corresponding decrease in maximum data rate will be experienced.

There is an alternative that could potentially increase the integrity of the bus, but most of all it would simplify the system integration and test and that is to use a central hub and where each subsystem is connected to it and not to the other units. The CAN protocol can optionally support this kind of topology.

The main parameters which define the CAN-Bus data transmission speed quality are:

- the length of the bus line,
- the length of the branch lines to the CAN-stations,
- the quality of the bus lines and the plug contacts,
- the execution of the bus cable (twisted, one-or two-wire bus),
- the design of the bus and
- the nature and strength of external interference.

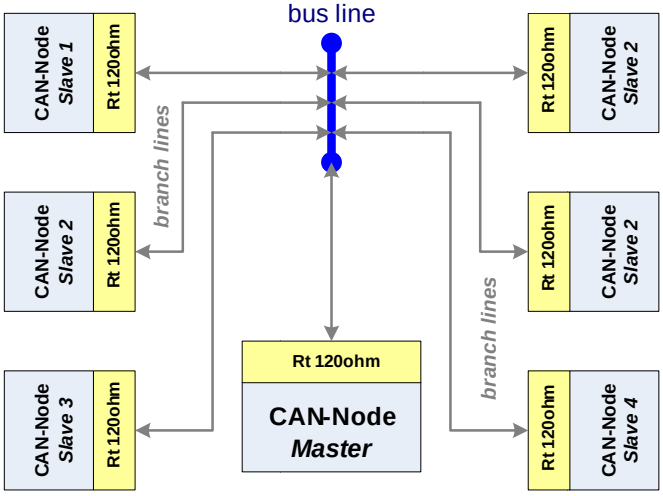

Fig. 4. CAN topology

In order to avoid the main drawback of this topology, namely the break of CAN topology is the following configuration (Figure 4) is suggested which later will be tested:

- ✓ the length of the bus line is 0.1 m
- ✓ the length of the branch lines to the CAN-stations is 0.6 m
- ✓ the type of the branch cable - twisted, three-wire (CANL, CANH and GND)
- ✓ each CAN-Station is defined with $\mathbf{R_t}$=120 ohms.

3. APPROACH TO THE SOFTWARE

The applied software is bound for a mere demonstration of the CAN communication but at the same time is sufficient to prove the functional validity of the test procedure. For further information and exact application look up [8].

The algorithm for the MCU and CAN initialization and implementation is obtainable from the next Flowchart (Figure 5.). Upon the first start of the program the frequency of the MCU is selected. In our case it equals 20 MHz for both *Master* and *Slave*. Further follows the initialization of the MCY registers and Ports (i. e. the one, used to control LED).

Next comes the CAN initialization. First the CAN registers are initialized and then the necessary speed defined. The determination of the transmission speed is related to MCU's tact-frequency. If necessary a mask can be chosen as a guarantee that every CAN-Node would process only messages meant directly at it.

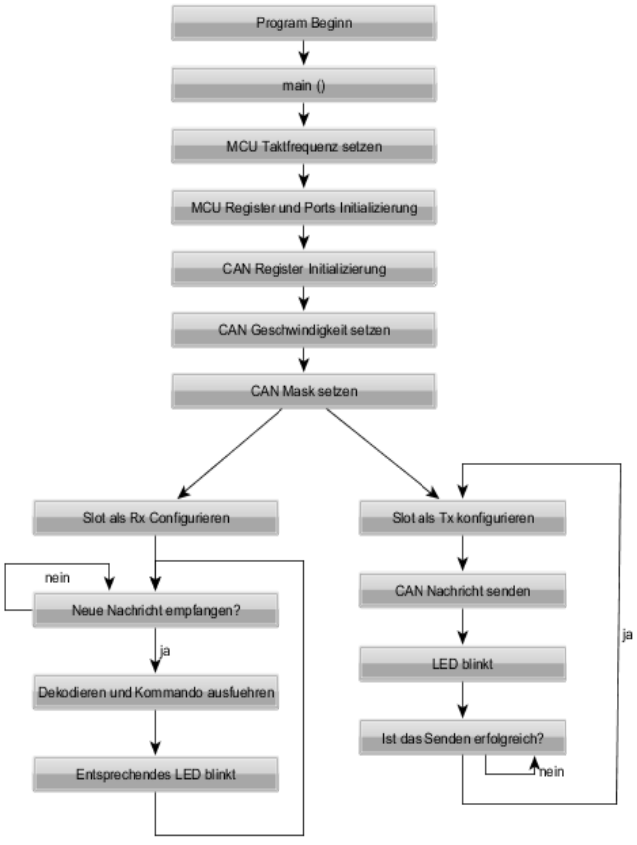

Fig. 5. Algorithms for MCU and CAN Initialization and data transmission

On the left-hand side of the Flowchart, the CAN initialization describes the receivers (Slave) behavior. First

off, the receiving slot is configured - Rx CAN. Then the messages are awaited. If the new message relates to the control system a decision, based on the message contained data, as to what LED should flash is originated. On the next step new messages are awaited.

The right side of the flowchart shows the Master logic. It configures the Tx CAN slot to transmit and sends the message. Then the Master waits for the Tx interrupt (which means that the message is successfully sent), then the LED diode flashes and a new cycle begins. If there is no interruption, the Master does not try to send new messages.

The length of the messages that are sent by the communication between microcontrollers is 8 bytes. Each byte contains different data (Figure 6). The last two bytes in this case do not contain actual data and their value should be 0 to avoid an errors in communication.

Leg ID	Command	Joint ID	Start Position	Middle Position	End Position	0x00	0x00

Fig.6. CAN Message

• Leg ID - the ID of that leg, which is to fulfill this command.
• Command - the command code, which must be met.
• Joint ID - the ID of this joint, which is to fulfill the command.
• Start position - the position from which to start the movement of the joint.
• Middle position - the middle position in the joint movement.
• End position - the position on which to end the movement of the joint.

Program code is written in C programming language using the following software tools:

- HEW - High-performance Embedded Workshop integrated user interface for developing and debugging embedded applications using microcontrollers, RENESAS . HEW is a powerful, yet easy to use development environment that provides a standardized user interface. C / C + + compilers and debuggers, including emulators and evaluation boards can be seamlessly integrated modules. This gives the user a uniform interface for the entire development cycle is given by the hand, enabling him to take advantage of the full functionality of development tools.
- The E8 is acting as an on-chip debugger for the entire M16C family and the H8/Super. In conjunction with the High-performance Embedded Workshop (HEW), it offers a wide range of debug functions.

4. APPROACH TO THE HARDWARE

The first test shows the path to achieving a preset ancillary goal. The idea is to connect just two *Master* (RSKM16C29) and *Slave* (EVBR8C22/23) microcontrollers, and to establish a communication between them over a CAN-Bus. Figure 7 visualizes how the Master - Slave is connected to twisted three-wire cable (CANL, CANH and GND).

The Master periodically tries to send a message. If the sending is successful, it is confirmed by the blinking LEDm. The latter is associated with a CAN TX interruption. In the Slave there is also a LEDs connected to a CAN Rx interruption. If no message is received LEDs does not blink. If after sending the message there is no Acknowledge, none of the LED blinks because the Tx interruption hasn't been generated. With this kind of communication testing the following experiments were carried out.

- Activating the Reset button on the *Master* shows that there is no transmission of any message and an Rx interruption in *Slave* is not generated.
- When interrupting the cable between *Master* and *Slave* the same results are observed: there is, of course, no communication.
- These procedures were tested at different communication speeds- up to 500Kbps on the *CAN System Clock* by way of CCLKR and C0CINR Register.

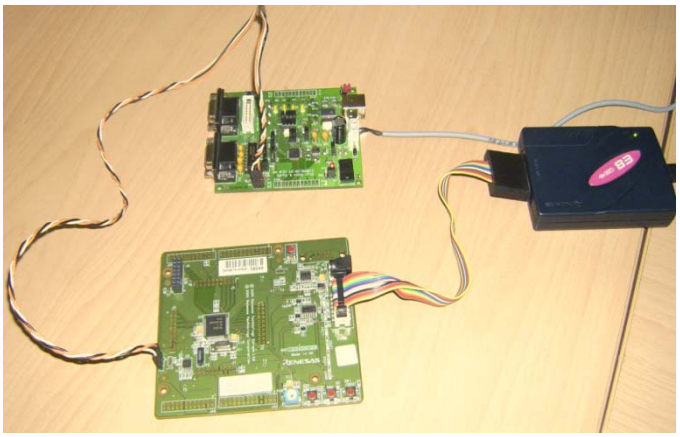

Fig. 7. The Master in connection with aslave

- In the second experiment, the Master and all six Slaves are involved. The function of all Slaves is the same - they have to wait for a message and not send anything. They are connected to the Master and receive the same message from it. This system can be seen in Figure 8.

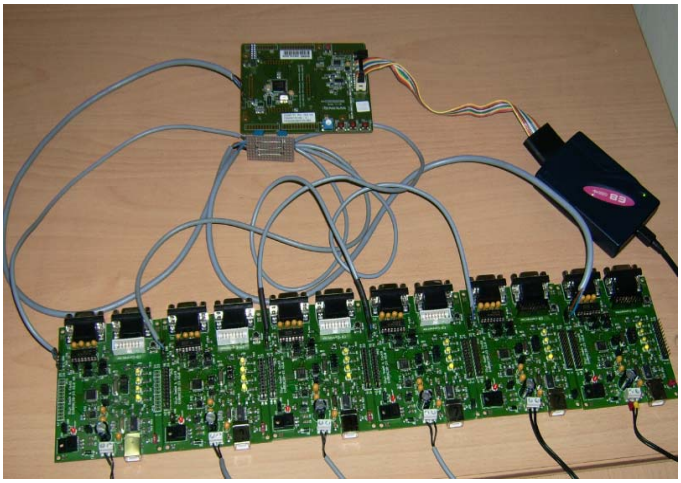

Fig.8. The Master in connection with all six slaves

- Another possibility to test the communication is the sending of new messages to all Slaves. The Master board has three buttons. When you press the second button, the Master sends a new message. In fact, the ID is the same as before, but the data are different. Slaves receive this new message and instead LED0, LED3 is flashing. If you release the button LED0 flashes as before, because once again the old message is being sent.
- The third possibility to test communication is the sending of real messages. This test involves sending of

six different messages separately for each Slave so that each Slave can only receive and process its own message. This test starts after the third button on the Master is pressed. LED2 confirms that the Slave has received its message by flashing.

5. RESULT AND DISCUSSION

The main subject of the paper is the significant increase of stability and scope of use of the CAN-BUS in the control of walking robots.

The efficiency of the chosen approach for local control was confirmed by the use of an experimental setup for validation of the used hardware and implemented software modules.

The real-time measured oscilloscope signals (Figure 9) for the CANL and CANH show stable levels and no significant noise.

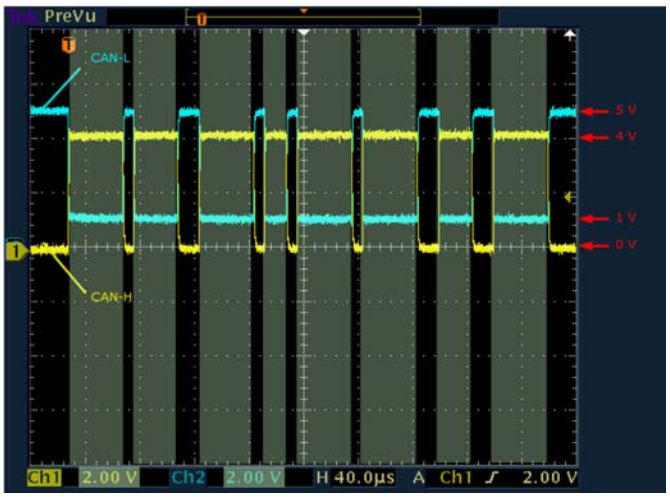

Fig.9. Voltage level of the CAN-bus measured with a two-channel oscilloscope

The results achieved in real time show a considerable performance reserve, allowing the implementation of additional algorithms.

The achieved basis is the fundament for further exploration of the system and communication limits by using additional hardware.

LITERATURA

[1] Berns, K. , Steuerungsans¨atze auf der Basis Neuronaler Netze f¨ur sechsbeinige Laufmaschinen, DISKI, Nr.61, Infix-Verlag, Bonn 1994.

[2]….Brooks, R. A (1989). A Robot that Walks; Emergent Behaviors from a Carefully Evolved Network. Neural Computation 1:2, pp.365-382, 1989

[3] Simmons et al. Autonomous task control for mobile robots. Proceedings of the 1990 IEEE International Conference on Intelligent Control, Philadelphia, PA, 1990

[4] Albus, J. et al. NASA/NBS Standard Referenc Model for Telerobot Control System Architecture (NASREM),National Bureau of Standards, Technical Note 1235, 1987

[5] McGhee, R. et al. A Hierarchically Structured System for Computer Control of a Hexapod Walking Machine.Theory and Practice of Robots and Manipulators. Proceedings of RoManSy 1984. The Fifth CISM-IFToMMSymposium. Edited by A. Morecki, G. Bianchi and K. Kedzior. Kogan Page, London, Hermes Publishing, pp.375-381, 1984

[6] K. Berns, W. Ilg, M. Deck, J. Albiez, and R. Dillmann. Mechanical construction and computer architecture of the four-legged walking machine BISAM. *IEEE Transactions on Mechatronics*, 4(1):1–7, 1999.

[7] K. Berns, V. Kepplin, R. Miller, M. Schmalenbach: Six-Legged Robot Actuated by Fluidic Muscles. In Proc. of the *3th Int. Conference on Climbing and Walking Robots* (CLAWAR), 2000.

[8] Spasov M. Entwicklung und Implementierung von CAN-Netzen in einem dezentralen Steuersystem eines mobilen Roboters. *Bachelorarbeit,* 2010