Arm To Arm Interface Using Embedded C

Manivannan.D[1], Mohanraj.C[2]
School of Computing, SASTRA University, Tirumalaisamudram,
Thanjavur, Tamilnadu, India - 613 401.
[1]dmv@cse.sastra.edu
[2]rjmohan90@gmail.com

***ABSTRACT*- Embedded systems are the most emerging field in these recent years. In this paper a different number of ARM processors (LPC2148 and LPC2378) are interconnected using C for distributed services. N numbers of processors are connected as the network and each processing devices are interlinked with each other, so that the each data that is processed by the devices and it can be used by the other device to activate their entire process. All the processed data's are communicated to other device through Xbee interface card. LPC2148 and LPC2378 ARM processors are used in this prototype and winXtalk is used as a software terminal window. In this paper, the ultimate benefits of multiple processor interactions related to the embedded applications and design issues of processor interconnection are discussed. The features of multiple processor interaction in inter process communication and executions of embedded multitasking are also discussed. In modern embedded computing platform, embedded processor used in various applications like home automation, industrial control, medical system, access control, etc. In this paper, using embedded processor interactions, the several data communication is established.**

Keywords: Advanced RISC Machine (ARM), LPC2148, LPC2378, Zigbee Interface Card.

I. INTRODUCTION

ARM (Advanced RISC Machine) is the industry leading processor which provides high performance with common RISC (Reduced Instruction Set Computing) architecture [2]. The arm processor supports the three states in which they are Arm, Thumb and Jazelle State; each operates 32 bit, 16 bit and 8 bit respectively. This processor is preferred more because of the load-store instructions, that process data on instruction and access memory on separate instruction and can access all 32 registers that are inbuilt. In this work, ARM7TDMI processor is used. The feature of this processor is that it supports 3-stage pipelining and uses Von-Neumann architecture for achieving minimized energy consumption. It has memory and interrupt controllers for interrupt actions to be done. The Thumb code takes about 40% of less space compared to 32-bit Arm codes. These processors are most widely used in various embedded applications like mobile phones, micro-wave oven to countless products. This processor also used in industrial smart control, medical system, access control, point-of-scale, communication gateways, embedded soft modem, smart parking, home automation and in automobiles. The microcontrollers [1] are used to execute the set of the preloaded task that are defined in the controller. If the two microcontrollers are used, the processed result of the task in the time t1 will be used as the input for the next task in time t3 and parallel the output of the task in the time t2 will be used by the task that going to be performed at time t4 and so on. By this, the concurrency in the task execution can be achieved. The interface of the microcontroller can be done both wired and wireless [6].

The common issues that arise in the embedded network systems are power problem in which the system uses a high amount of the power for the simple process. The next issue is that distance coverage of about 50 meters wherein this Xbee/Zigbee interface card [5] covers the distance of about 100 meters and it uses the IEEE 802.15.4 standard. Wireless transceivers are available in low cost and it reduces the overall system cost. The frequency range and the processor speed are comparatively higher than the existing system.

Most of the work in the embedded system is implemented using the wired interface. But in this work wireless transceivers like Xbee/Zigbee interface card [6] are used. With the use of these interface cards, efficiency and accuracy can be increased and also low cost. Coverage for the wireless device will be higher than the wired device.

In this work, LPC2148 [3] and LPC2378 [4] as ARM processor, the experimental setup is made. The LPC2148 Microcontroller [3] is the 16/32-bit ARM7TDMI-S RISC processor which provides the embedded system with high speed flash memory of about 32kB to 512kB and also On-Chip RAM of 8kB to 40kB. Maximum CPU clock available for the controller is 60MHz for settling time 100µs. CPU operating voltage is 3.0v to 3.6v. The oscillating frequency for the LPC2148 is from 1MHZ to 25MHz. The LPC2148 uses AMBA High Performance Bus (AHB) interface for interrupt control. LPC2148 has various serial communication interfaces ranging from USB 2.0 full speed device, multiple UART's, SPI (Serial Peripheral Interface), SSP (System Service Provider) to I^2C's which is suitable for communication gateways and protocol support, soft modems, low end image

processing, providing high processing power and large buffer size. It also supports two 10 bit ADC's, PWM and DAC operations.

The LPC2378 Microcontroller [4] is the 16/32-bit ARM7TDMI-S RISC processor which provides the embedded system with 512 kB high speed flash memory and also On-Chip RAM of 32kB, in which 16 kB is used for Ethernet interface and 8 kB for USB interface. Maximum CPU clock available for the controller is 72MHz for settling time 100µs. CPU operating voltage is 3.3v (3.0v to 3.6v). The LPC2378 oscillating frequency is from 1MHZ to 25MHz. LPC2378 offers multi-purpose serial communication application which incorporates 10/100 Ethernet Media Access Controller (MAC), USB full speed device with 4 kB RAM, four UARTs, two CAN channels, an SPI interface, two Synchronous Serial Ports (SSP), three I2C-bus interfaces and an External Memory Controller (EMC). An 8-bit data/16-bit address parallel bus is available. Microcontroller interactions are needed for enhancing the performance of the processor by performing load sharing. Traffic in the network can be resolved. Switching over to other process in the system can be done easily. Section 2 explains the way in which the processor interaction can be made in the network. Section 3 gives the overall functional block and the hardware unit of the processor interaction and also summarizes the overall execution timing analysis of the system.

II. FUNTIONAL DESCRIPTION

Microcontrollers are interconnected [1] with another controller with the use of the UART (Universal Asynchronous Receiver/Transmitter). The communication between microcontrollers can be done serially where data can be transmitted as frames. These frames have one start and stop bit with the 8-bit data line in between [3]. ARM processor supports two UART where corresponding pin for UART0 is pin 19 & 21 and for UART1, pin 33 & 34 are used. These pins are used to send data serially by using Zigbee through RS-232 cable interconnecting it.

In our system model, LPC2148 and LPC2378 processors are used. Various types of interconnections are made as per design and application using that processors and different connection methodologies [7] are evaluated. Node to node [8], multi-hop, and master-slave controls is the simple methodologies are evaluated.

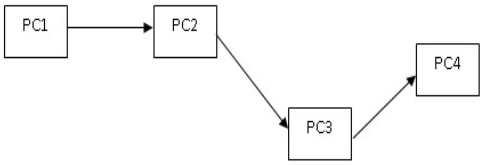

Fig. 1. Sequential Network

Fig. 1 shows that the system where the data processed by the process controller (PC) in the sequential manner i.e., The input is passed to the first processor (PC1) where that process will do some process and the result will be passed to the next processor (PC2, PC3,.. etc) as the input through wireless transceivers. Then the data will be again processed by the predefined task in the processor and that result will be passed to the next processor and so on.

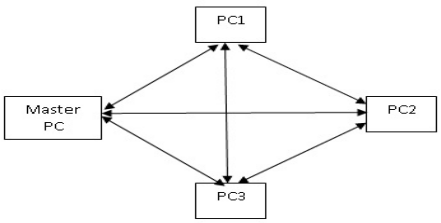

Fig. 2. Distributed Embedded System

In distributed embedded system [9], the master controller will transmit control command to the each processor node. The process controller [11] has to perform the specified task and the results are noted. In this system, n-number of process controller can be connected and n-number of tasks can be executed at the time [6]. Above defined distributed embedded system can be reformulated in which each controller will be acting as the master controller as well as individual process controller, so that the command can be send to the other controller and also processing can be done individually. By means, load sharing [10] can be done in the entire system to increase the overall system performance. The important notification in this system is that the processor used in this network formation [12] can use either same processor for all the process or can use different processor to support this system.

In specific applications like automation industries, designers might design their circuit in which way that can be placed in a different location or might they will use n-number of processors for controlling the process that they working on. In situation, either the processor usage will be idle or utilization may vary. For example, says having one master controller that has to do big process, what the master controller will do is that, it will check whether the process controller's are free. If they are free, then the master controller[13] will allocate some process to the process controller and the end result will again be passed to the master controller. By this mean, load sharing can be done easily with the help of the process controller and also a utility factor for all the controllers will moreover same. When the execution time of the tasks in processor decreases, the efficiency of the system will be increased [14]. Hence the lifetime and energy gets improved.

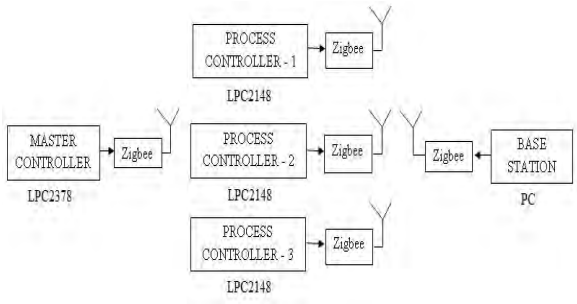

Fig. 3. Functional Blocks of the System

Fig. 3 shows the funtional blocks of the System in which the entire process will be occuring.

III. HARDWARE SETUP

In this experimental setup, three LPC2148 mini development board used as process controller and one LPC2378 project board used as master controller. The ultimate purpose of this work is to enhance the system performance [15] in terms of load sharing, energy consumption and reduce the execution time for each and every process controller in this proposed system. Wireless interface cards are used between the controllers to achieve heterogeneity and data communication [16]. Fig. 4 shows the functional hardware setup of 3 LPC2148, 1 LPC2378 and Zigbee Interface Card XB-24.

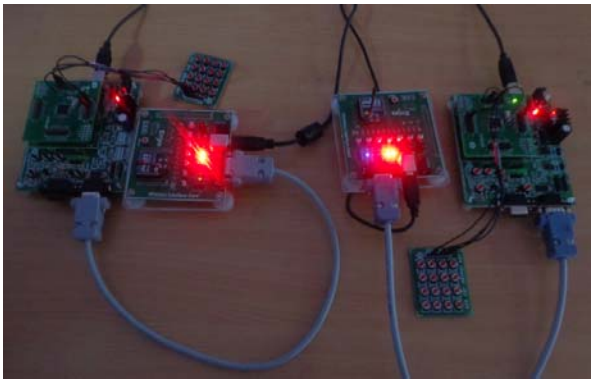

Fig. 4. Hardware Setup of Arm To Arm Interface

In this work using hardware setup (Fig. 4), all controllers[12] will perform their process individually for the instance of time. If the processed result of the controller is needed by the master controller or by the base station, then the master controller will send the command to the process controller by unicast or multicast mode[17]. Process controller on receiving the command will check whether the command is meant for particular process controller. When the commad checking is satisfied, the Process controller will send the processed result to the master controller as well as to the base station for the further execution. By this way of process, each controller in the system can able to communicate with other controller (master or process) in the system by which the load sharing can be achieved[16]. Various performance evaluation factors like life time, energy, communication overhead are measured.

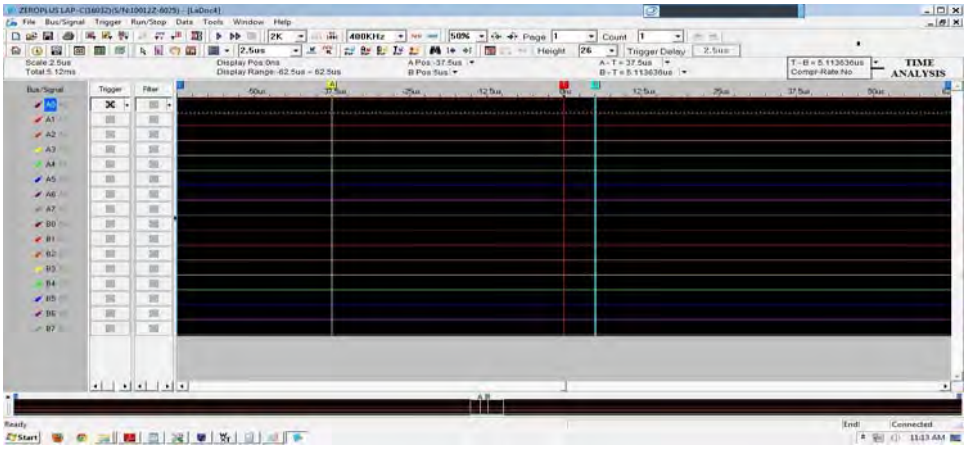

Fig. 5. Execution Time of the System

The above shown figure shows the overall execution time taken by the using ARM to Arm interaction with the help of logic analyzer. In this Fig. 5, the red vertical line indicates the start of the signal flow in the process and the blue vertical line indicates the end of the signal flow of the process. The approximate time taken by the uniprocessor for the executing the task is about 20µs where as the Arm to Arm interaction takes 5µs for execution. Table I shows the number of task executed and the time taken by uniprocessor as well as by Arm to Arm interaction. Table II shows the comparison table of the uniprocessor and Arm to Arm processor relating to some of its assessment factors.

Table I. Execution Time Analysis of the System

NUMBER OF TASK	Using ARM To ARM	UNIPROCESSOR
T1,T2,T3,T4,T5	5µs	20µs

Table II. Uniprocessor Vs Arm To Arm

Assessment Factors	Uniprocessor	ARM To ARM
Communication Overhead[19]	Normal	Good
Time for Transaction	Normal	Better
Life Time[19]	Normal	Good
Processor Management[20]	Normal	Good
Node to Node Communication[20]	Normal	Better
Storage[19]	Normal	Good
Data Management[19]	Normal	Good

In Distributed WSN, the life time, energy conservation issues are resolved.

IV. CONCLUSION

Arm is a leading industry processor which provides better interoperability service for all the intercommunication process than other embedded processor. In industry, n-number of processor is used to perform various functions like monitoring, controlling and commanding. So the interconnection between more than one processor using ARM will lead and enhance the system performance of the embedded systems. By using interconnection between more than two processors, the utilization rate of all the processors in the network is increased. Also the delay between the processor executions is reduced. In order to achieve perfect co-ordination between the processors in the network the total system load are drastically reduced. Hence the processor failure due to practical reasons is drastically reduced. Thus the overall system efficiency is improved.

ACKNOWLEDGEMENT

We thank SASTRA University for providing financial support for this research work under the Research and Modernization fund– R&M/0008/SOC – 001/2009-10.

REFERENCES

[1] Muji, S.Z.M.; Rahim, R.A.; Rahiman, M.H.F, "Two Microcontrollers Interaction Using C", Second International Conference on Computer Research and Development, pp.290-292, May 2010.
[2] Trevor Martin, *The Insider's Guide to the Philips ARM7-Based Microcontrollers*, ISBN: 0-9549988 1, Hitex (UK) Ltd., April 2005.
[3] "LPC214x user manual," rev. 3 - 4, nxp b.v. October 2010.
[4] "LPC23xx user manual," rev. 2, nxp b.v. February 2009.
[5] Johan Lonn, *Zigbee for Wireless Networking*, March 2005.
[6] MANOJ KOLLAM, S.R. Bhagya Shree, "zigbee wireless sensor network for better interactive industrial automation", *IEEE Transactions on Advance Computing*, 978-1-4673-0671-3, 2011.
[7] Liu An-Feng, Zhang Peng-Hui, Chen Zhi-Gang, "Theoretical analysis of the lifetime and energy hole in cluster based wireless sensor networks", *J. Parallel Distrib. Comput.*, vol. 71, pp. 1327–1355, 2011.
[8] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, E. Cayirci, "Wireless sensor networks: A survey", *Computer Networks*, vol. 38 (4), pp. 393–422, 2002.
[9] D.M. Blough, P. Santi, "Investigating upper bounds on network lifetime extension for cell-based energy conservation techniques in stationary ad hoc networks", in MobiCom'02: Proceedings of the 8th annual international conference on Mobile computing and networking. New York, USA, 2002, pp. 183–192.
[10] Z. Cheng, M. Perillo, W. Heinzelman, "General network lifetime and cost models for evaluating sensor network deployment strategies", *IEEE Transactions on Mobile Computing*, vol. 7 (4), pp. 103–115, 2008.
[11] Yifeng He, Ivan Lee, Ling Guan, "Distributed algorithms for network lifetime maximization in wireless visual sensor networks", *Transactions on Circuits and Systems for Video Technology*, vol. 19 (5), pp.704–718, 2009.
[12] J. Hill, R. Szewczyk, A. Woo, S. Hollar, et al., "System architecture directions for networked sensor", *ACM SIGPLAN Notices*, vol. 11 (35), pp. 93–104, 2002.
[13] R. Madan, S. Lall, "Distributed algorithms for maximum lifetime routing in wireless sensor networks", *IEEE Transactions on Wireless Communications*, vol. 5(8), pp.2185–2193, 2006.
[14] M. Noori, M. Ardakani, "A probabilistic lifetime analysis for clustered wireless sensor networks", in: Proc. IEEE Wireless Commun. and Networking Conf. (WCNC), Las Vegas, USA, March 2008.
[15] S. Olariu, I. Stojmenovic, "Design guidelines for maximizing lifetime and avoiding energy holes in sensor networks with uniform distribution and uniform reporting", in: J. Domingo-Pascual (Ed.), Proc. of the IEEE INFOCOM. IEEE Communications Society, New York, 2006, pp. 1–12.
[16] X. Tang, J. Xu, "Optimizing lifetime for continuous data aggregation with precision guarantees in wireless sensor networks", *IEEE/ACM Transactions on Networking*, vol. 16 (4), pp. 904–917, 2008.
[17] Q. Xue, A. Ganz, "On the lifetime of large scale sensor networks", *Computer Communications*, vol. 29 (4), pp. 502–510, 2006.
[18] O. Younis, S. Fahmy, "HEED: a hybrid, energy-efficient, distributed clustering approach for ad hoc sensor networks", *IEEE Transactions on Mobile Computing*, vol. 3 (4), pp. 366–379, 2004.
[19] Ousmane Diallo, JoelJ.P.C.Rodrigues, MbayeSene, "Real-time data management on wireless sensor networks: A survey", *Journal of Network and Computer Applications, vol.* 35, pp.1013–1021, 2012.
[20] Jing Dong, Kurt Ackermann, Cristina Nita-Rotaru, "Secure group communication in wireless mesh networks", *Ad Hoc Networks*, vol. 7, pp.1563–1576, 2009.