Multiprocessor Configuration of 8051 Microcontroller Chip

Vikalp Dwivedi[1], Prof Shripad G. Desai[2]
[1]Student, Department of Electrical Engineering, Department of Electrical Engineering, Bharati Vidyapeeth Deemed (To Be) University, College of Engineering, Pune, India
[2]Assistant Professor, Department of Electrical Engineering, Bharati Vidyapeeth Deemed (To Be) University, College of Engineering, Pune, India

***Abstract* - This paper extends the parallel operation of the 8051-microcontroller chip and presents the use of multiple 8051s that are connected to a common loop in a multiprocessor configuration. The 8051 multiprocessing implies many processors acting in some unified manner and connected so that data can be interchanged between them. There is generally a controlling or "talker" microcontroller that directs the activities of the remainder of the loop microcontrollers, or "listeners". One particular characteristic of a talker-listener loop is the frequent transmission of data between the talker and individual listeners. All data broadcast by the talker is received by all the listeners, although often the data is intended only for one or a few listeners. While sometimes, data is broadcast that is meant to be used by all the listeners. Communication through multiple 8051s use standard UART technology which assign unique addresses to all the listeners using mode 1, in this mode the listeners will waste a lot of processing time rejecting data not addressed to them. Mode 2 and 3 reduces processing time by enabling character reception based upon the state of SM2 in a listener and the state of bit ten in the transmitted character. A single strategy is used to enable a few listeners to receive data while the majority ignores the transmissions. This system is implemented and tested for transmitting data through talker as master and two listeners as slaves.**

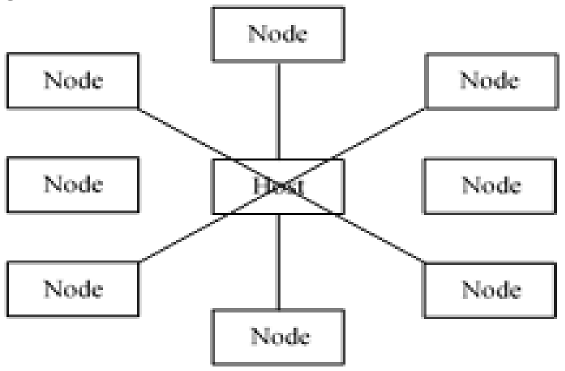
***Index Terms* - 8051 Microcontroller based system, RS232 serial communication port, systolic parallel processing, parallel computers.**

1.INTRODUCTION

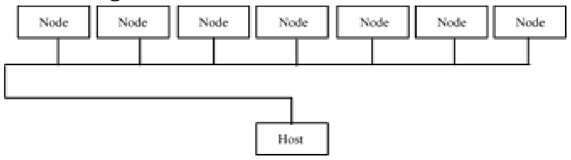
Microcontrollers are highly integrated, low-cost, programmable computers that are used to design embedded systems. A typical microcontroller comprises a central processing unit (CPU), random access memory (RAM), and input/output (I/O) interfaces in a single package. Additional features such as on-chip analog-to-digital converters (ADC), timers, and low power consumption make them adaptable enough to be applied to a wide variety of applications. The benefit of such tight integration is a small footprint as compared to a more general processor that would require several additional modules and greater wiring complexity to perform the same function. The small footprint, low cost, and high flexibility of microcontrollers make them ideal for integration in everything from automobiles to household appliances. These benefits, however, come at a cost. In order to maintain low power consumption, clock speeds are typically on the order of tens of MHz Unlike general-purpose computers, most have firmware to store static programs in place of RAM for dynamic loadable programs. Thus, the microcontroller must be reprogrammed to run new software. Modifiable data, such as variables, must be stored in volatile memory so on-chip RAM may be present. In order to maintain the cost-effectiveness of the chip, RAM is typically only a few kilobytes in size and the data path may be only 8 or 16 bits wide [1].

2. NETWORK CONFIGURATIONS

The first problem faced by the network system designer is how to physically hook the microcontrollers together. The two possible basic configurations are the star and the loop, which are shown in Figure 1 [2]. The star features one line from a central microcontroller to each remote microcontroller, or from "host" to "node." This configuration is often used in time-sharing applications. Each node sees only the data on its line; all communication is private from host to node. The loop uses one communication line to connect all of the microcontrollers together. There may be a single host that controls all actions on the loop, or any microcontroller may be enabled to be the host at any given time.



Star configuration



Loop configuration

Figure 1: Communications configurations

The loop configuration is often used in data-gathering applications where the host periodically interrogates each node to collect the latest information about the monitored process. All nodes see all data; the communication is public between host and nodes. The star is a good choice when the number of nodes is small, or the physical distance from host to node is short. But, as the number of nodes grows, the cost and physical space represented by the cables from host to nodes begins to represent the major cost item in the system budget. The loop configuration becomes attractive as cost constraints begin to outweigh other considerations [3].

Microcontrollers are usually applied in industrial systems in large numbers distributed over long distances. Loop networks are advantageous in these situations. Many hybrid network arrangements have evolved from the star and the loop configurations.

3.MULTIPROCESSORS

The simplest definition of a multiprocessor system is a computer that has more than one CPU. However, this belies the breadth of technologies and configurations that consist of multiple CPUs. At the highest level of integration, multiple CPU cores can be placed on a single silicon die to allow chip multiprocessing (CMP). Identical CPUs may also be packaged separately but share a common memory interface in symmetric multiprocessing (SMP) computers. CPUs need not be identical, as in the case of asymmetric multiprocessing. Memory may be shared equally among all processors, resulting in a uniform memory access (UMA) machine or it may be divided among the processors in non-uniform memory access (NUMA) machines.

At the other end of the spectrum, clusters combine several individual computers with a high-speed network to give the illusion of a single computer with many processors. Grids computers are even more loosely coupled than clusters and may have high latency, low bandwidth interconnects. Clusters and grids have become popular because they can be constructed from low-cost off-the-shelf computers and local area networks, thus making affordable supercomputers possible [4].

3.1.Uniform Memory Access (UMA) Multiprocessors:

The simplest processor interconnection pattern assumes that all the processors work through a central switching mechanism to reach a centralized shared memory as shown in Figure 2. There are a variety of ways to implement this switching mechanism, including a common bus to global memory, a crossbar switch, and a packet-switch network. Systems using a bus are limited in size, since only so many processors can share the bus before it becomes saturated, while systems using crossbar switch, the cost of the switch soon becomes the dominant factor, again limiting the number of processors which may be connected. UMA multiprocessors based on switching networks can conceivably contain a large number of processors [3].

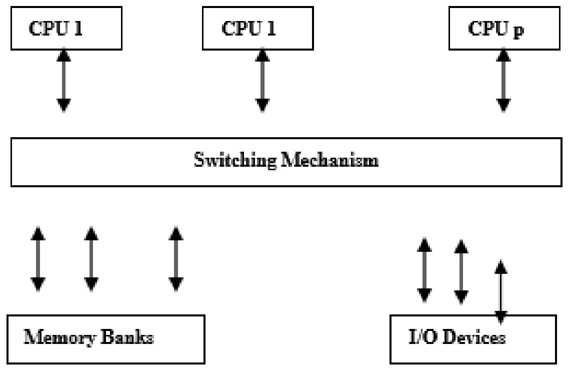


Figure 2: The Uniform Memory Access (UMA) multiprocessor model

3.1. Multi-microcontrollers

In Multimicrocontroller architecture, there has no shared memory. Each microcontroller has its own internal private memory and process interaction occurs through message passing. Advances in multi-microcontrollers systems are the change in how processors communicate. In order to send a message from one processor to a nonadjacent processor, every intermediate processor along the message's path must store the entire message and then forward the message to the next processor. There is another method for communication between multiple- microcontrollers, the message flows in a pipelined fashion from the source node to the destination node, none of the intermediate nodes store the message. A message being passed from one node to a nonadjacent node does not interrupt the CPUs of the intermediate nodes and only the direct connected modules are involved [6].

3.2. Modes 2 and 3: Multiprocessor

Modes 2 and 3 are identical except for the baud rate. Mode 2 uses baud rate of f/32 if SMOD (PCON.7) is cleared or f/64 if SMOD is set. Data transmission using modes 2 and 3 features eleven bits per character, as shown in Figure 4 [7]. A character begins with a start bit, which is a high-to-low transition that lasts one bit period, followed by 8 data bits, LSB first. The tenth bit of this character is a programmable bit that is followed by a stop bit. The stop bit remains in a high state for a minimum of one period bit.

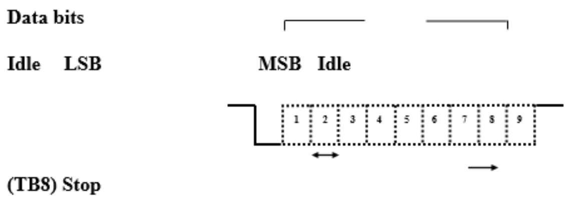


Figure 4: Asynchronous 9-bit character used in modes 2 and 3.

When the 8051 transmits a character in mode 2 and 3, the eight data bits are whatever value is loaded in SBUF. The tenth bit is the value of bit SCON.3, named TB8. This bit can be cleared or set by the program. Interrupt bit T1 (SCON.1) is set after a character has been transmitted and must be reset by program action [6]. Characters received using mode 2 and 3 have the eight data bits placed in SBUF and the tenth bit is in SCON.2, called RB8, if certain conditions are true. Two conditions apply to receive a character. First, interrupt bit RI (SCON.0) must be cleared before the last bit of the character is received, and second, bit SM2 (SCON.5) must be a zero or the tenth bit must a one. If these conditions are met, then the eight data bits are loaded in SBUF, the tenth bit is placed in RB8, and the received interrupt bit RI is set. If these conditions are not met, the character is ignored, and the receiving circuitry awaits the next start bit [2]. The significant condition is the second. If RI is set, then the software has not read the previous data (or forget or reset RI), and it would serve no purpose to overwrite data [7]. Clearing SM2 to zero allows the reception of multiprocessor characters transmitted in mode 2 and 3. Setting SM2 to one prevents the reception of those characters. Put another way, if bit ten is a one, then reception always takes place; SM2 is ignored. If bit ten is a zero then only those receivers with SM2 set to zero are interrupted[8] [9].

4.8051 DATA COMMUNICATION MODES

4.1. Serial Data Communication Scheme

The 8051 has one serial port - pins p3.0 (RXD) and p3.1 (TXD) – that receives and transmits data. All data is transmitted or received in two registers with one name: SBUF [7]. Writing to SBUF results in data transmission; reading SBUF accesses received data. Transmission and reception takes place simultaneously, and the receiver can be in the process of receiving a byte while a previous byte is still in SBUF. The first byte must be read before the reception is complete, or the second byte will be lost. Physically the data is a series of voltage levels that are sampled, in the center of the bit period, at a frequency that is determined by the serial data mode and the program that controls that mode.

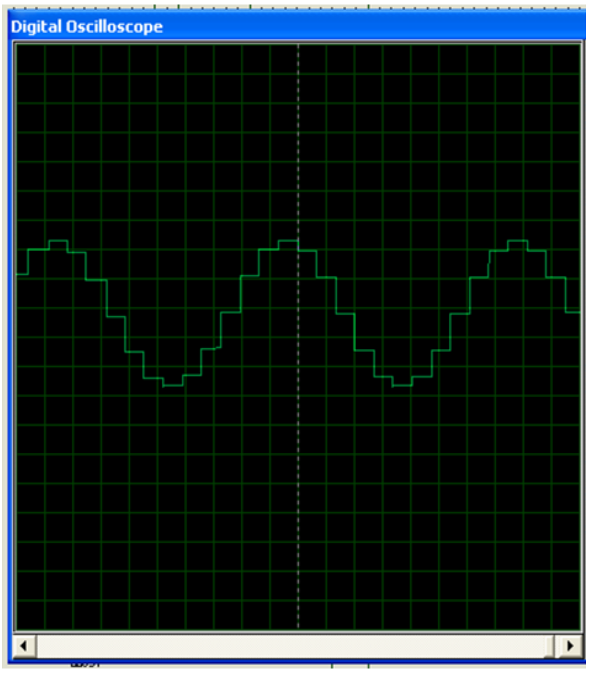
All devices that wish to communicate must use the same voltage levels, mode, character code, and sampling frequency (baud rate). The wires that connect the ports must also have the same polarity so that the idle state, logic high, is seen by all ports. The four communication modes possible with the 8051 present the system designer and programmer with opportunities to conduct very sophisticated data communication networks.

4.2. Parallel Data Communication Scheme

The main benefit to this scheme is that data is transferred in parallel so a significant speed improvement over the serial schemes can be expected.

Flow chart of parallel processing shown in figure (6). This Scheme uses multiple microcontrollers (MCUs) that cooperate to solve one problem. This "parallel microcontrollers" can be used as an educational system that will help the development, understanding and fine-tuning of many parallel algorithms. It can also be used to achieve high performance in many cases such as in Digital Signal Processing. Our Project is easily constructed using many microcontrollers of type 8051.
However, the efficient use of it requires more effort as the development and implementation of parallel algorithms is a rather difficult task. This parallel computer is of type "Single Instruction Multiple Data" (SIMD) which means that all MCUs execute the same program, but each MCU works with different data. All MCUs execute code from their internal flash memory. A key feature for the implementation of the parallel machine is that all MCUs have to execute the program in absolute synchronization with each other. This is achieved by using a common external clock for all the MCUs, and by forcing all MCUs to begin execution at the same time. This is feasible with the AT8051 family and is achieved through controlling the CLK and RESET inputs of the MCUs. A clock enable/disable switch is also implemented to allow the user to "freeze" the code execution at any time as the parallel computer may be used for educational reasons. We use this feature to construct function generator with high frequency. There are many different types of signal generators, with different purposes and applications (and at varying levels of expense). Such devices contain an electronic oscillator, a circuit that is capable of creating a repetitive waveform.
The most common waveforms are a sine wave, saw tooth, step (pulse), square, and triangular waveform generators. The accuracy of the waveform at the generator output is mainly determined by the waveform approximation technique. Besides waveform quality output signal requirements another quality criterion is the frequency range. A signal generator provides a high-fidelity sine wave signal ranging from low frequencies to more than 1MHz

4.2.1 Practical sine wave generation using parallel processing
When the function generator implemented by using one microcontroller, we acquire a low frequency sine wave as shown in Figure 7 [10].



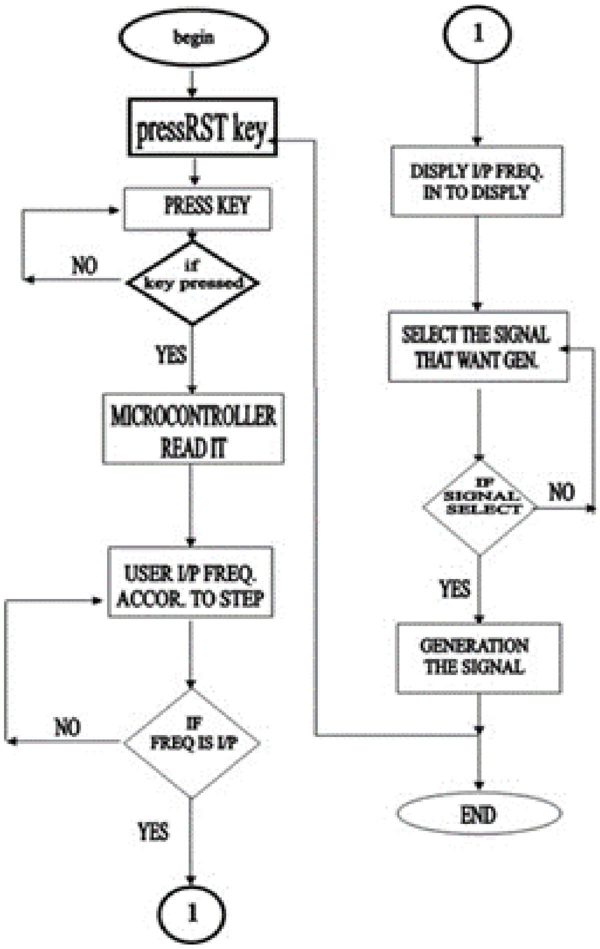


Figure 7 : Signal generator implemented by using one microcontroller

a. : Sine wave Frequency = 125 kHz
b. : Flowchart for signal generation using one microcontroller

But this drawback overcome by using parallel processing that lead to high frequency with high precision as shown in figure 8.

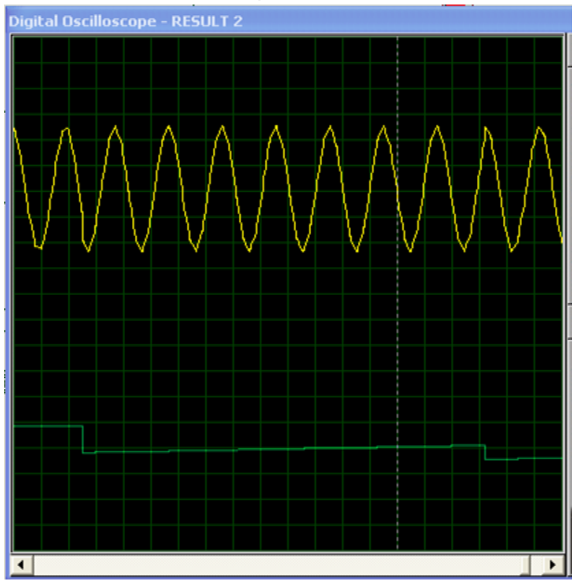
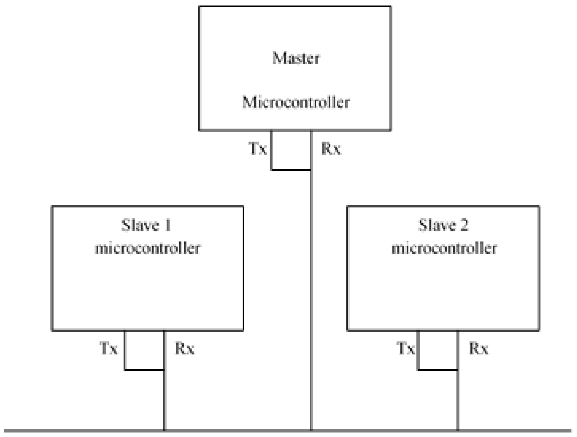


Figure 8: Signal generator implemented by using parallel processing Sine wave Frequency = 1.14 MHz

5.SYSTEM HARDWARE

Modes 2 and 3 have been included in the 8051 specifically to enhance the use of multiple 8051s that are connected to a common loop in a multiprocessor configuration. When the processors are connected in a loop configuration, then there is generally a controlling or "master" processor that directs the activities of the remainder of the loop units, or "slaves" as shown in Figure 5. One particular characteristic of a master-slave loop is the frequent transmission of data between the master and the individual slaves. All data broadcast by the master is received by all the slaves, although often the data is intended only for one or a few slaves. At times, data is broadcast that is meant to be used by all the slaves. Modes 2 and 3 reduces processing time by enabling character reception based upon the state of SM2 in a slave and the state of bit ten in the transmitted character[11]. A single strategy is used to enable a few slaves to receive data while the majority ignores the transmissions.



Single Wire Communication
Figure 9: Multi-microcontroller Configuration: Master-Slave connection

The hardware part presents a single wire bus system where the Master and Slaves are all microcontrollers using their UARTs in mode 3, which is multiprocessor mode. An example having two slaves is represented by Figure 9. The receive (Rx) and transmit (Tx) pins of all the microcontrollers are connected together and data can be shifted out of the master or into it but not at the same time.

Modes 2 or 3 may be used between a master device and multiple slave devices. Mode 2 is the fixed transfer rate and mode 3 is the variable baud rate. In theory there could be 256 slaves (i.e. 8-bit address) but in practice too many slave devices would cause loading effects. If this system uses the same microcontroller type for master and slaves then mode 2 would be preferable to mode 3 since it would not be necessary to program timers for baud rate generators.

6.COMMUNICATION PROTOCOL FOR MASTER/SLAVE COMMUNICATION

All slaves initialize SM2 to one after power-up, and the master configures all address messages using a one in bit ten. So all slaves receive and process any address message to see whether action is required. Addressed slaves transmit data characters to the master with bit ten set to zero. The master has SM2 set to zero so that all communications from slaves acknowledged. Data characters from a slave to the master are ignored by remaining slaves. At the end of the data, the addressed slave resets SM2 to one. The data rate is set by timer one in the auto-reload mode to be 83333 baud. That portion of the master and slave program has to do with setting up the multiprocessor environment will be programmed. The messages that are sent from the master to the slaves can be placed in the memory for later use. A communication protocol exists for data transfer between master and slave devices as shown in figure 10.

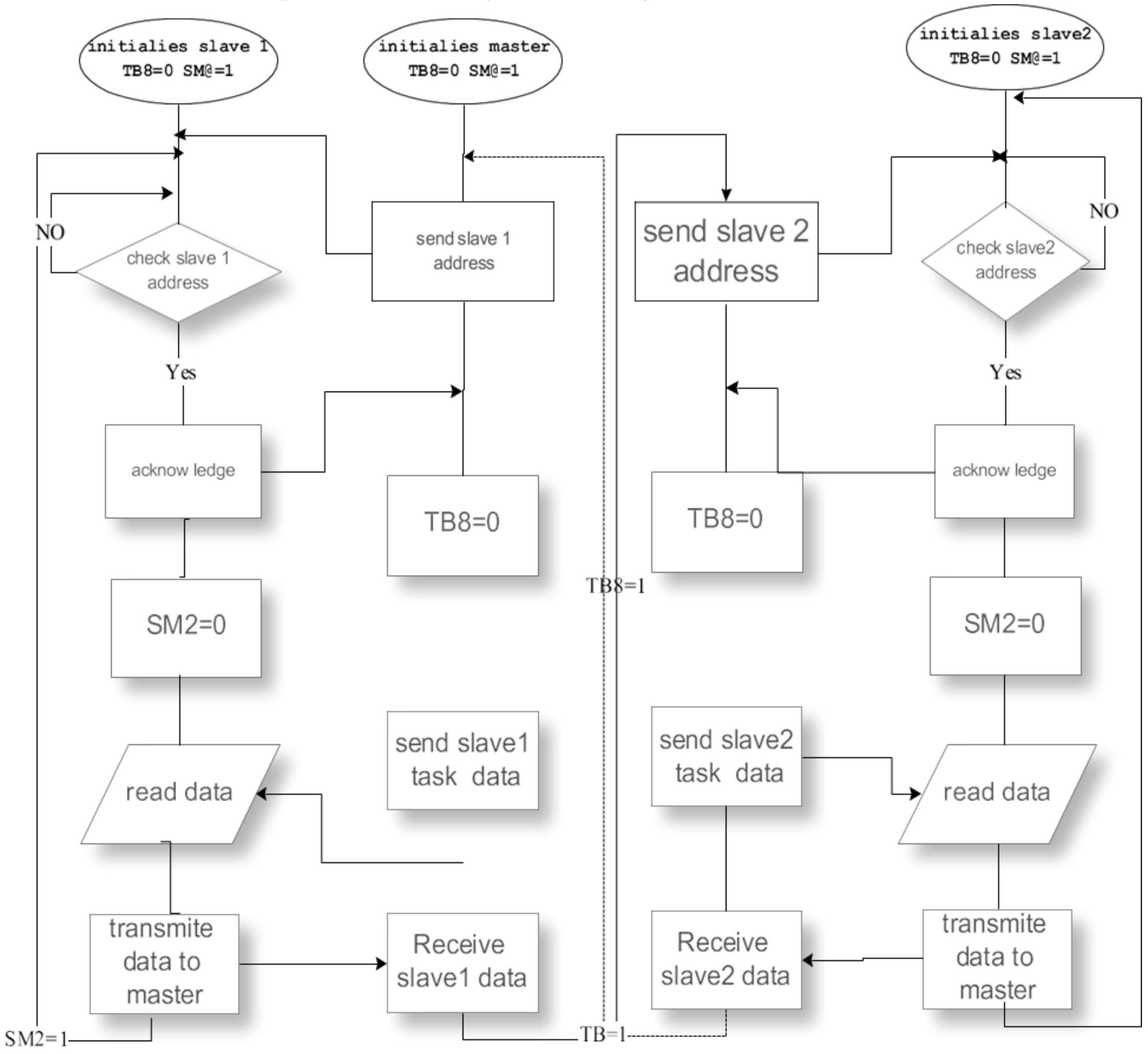


Figure 10: Flow chart for communication protocol for master/slaves communication

7.CONCLUSIONS

In this paper we see the difference between single microcontroller system and the multiprocessing system by improvement the performance. Now it can connect a system of microcontrollers work in parallel to control a group of tasks (each one task or more from the overall job controlled by one slave) implemented sequentially as in serial multi-processing or simultaneously as in parallel processing, or we can use them to implement several works at same time. Also, it can generate arbitrary high frequency waveforms.

REFERENCES

[1] Kalim Moghul, “Design of a message passing interface for multiprocessing with Atmel microcontrollers”, Presented to the Engineering Division of the Graduate School of Cornell University, May 2006.

[2] K. J. Ayala, "The 8051 Microcontroller Architecture, Programming, and Applications" West Publishing Company, 1991.

[3] M. J. Quinn, "Parallel Computing Theory and Practice", MC GRAW-HILL, INC., 1994.

[4] Peterson, Larry L. and Davie, Bruce S., “Computer Networks: A Systems Approach”, 3rd ed. Morgan Kaufmann 2003.

[5] W. Gear, “Computer Organization and Programming”, Addison-Wesley, 1998.

[6] D. Calcutt, F. Cowan, and H. Parchizadeh, “8051 Microcontroller an Applications-Based Introduction” Newnes, 2004.

[7] S. Yeralanand and A. Ahluwalia, “Programming and Interfacing the 8051 Microcontroller”, Addison-Wesley, 1995.

[8] Kashif Altaf and Javaid Iqbal, “Multiprocessor Communication using 8051 Microcontroller and RS-485 Line Driver”, 2003.

[9] Mohsin Murad and Uzair Kamal, "Multiprocessor Communication using 8051 Microcontroller and RS-232 Interface", 2006.

[10] R.A.Abdul Husain, “Design and implement high frequency function generator using parallel microcontrollers”, project No.437, 2010.

[11] Alfredo del Río and Juan José Rodríguez Andina, UVI51, “A simulation tool for teaching/learning the 8051 Microcontroller”, 30th ASEE/IEEE Frontiers in Education Conference, Kansas City, 2000.