International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 10 Issue 01, January-2021

A Robust Protocol for Multiprocessor Architectural Message Passing System

Rashmi Abhay Kamde
Department of Electrical Engineering.
K. J. Somaiya Polytechnic
Mumbai, India

Bhagyashree Shailesh Firake
Department of Electrical Engineering.
K. J. Somaiya Polytechnic
Mumbai, India

Jayshree Suhas Wasnik
Department of Electrical Engineering.
K. J. Somaiya Polytechnic
Mumbai, India

***Abstract*— This is a project to designing system and developing protocol to enable communication within system environment. Embedded systems are the brains of today's most digital and industrial control systems. In systems where more than one processor is incorporated, the need for multiprocessor communication often arises. It fully utilizes microcontroller features & embedded technology concepts to minimize the complications of digital gates, size and cost too.**

Keywords*— Multi processing; multiple peripheral devices; communication Protocol; I2C protocol ; SPI protocol ; packet format; Packet transfer; time slicing;

I. INTRODUCTION

A. What is Multi processing?

Multiprocessing, as generally defined, is the use of two or more central processing units (CPUs) within a single computer system. The term also refers to the ability of a system to support more than one processor and/or the ability to allocate tasks between them. There are many variations on this basic theme, and the definition of multiprocessing can vary with context, mostly as a function of how CPUs are defined. In present project Multi processing refers to use of multiple peripheral devices.

B. What is communication Protocol?

A communications protocol is the set of standard rules for data representation, signaling, authentication and error detection required to send information over a communications channel. An example of a simple communications protocol adapted to voice communication is the case of a radio dispatcher talking to mobile stations. The communication protocols for digital computer network communication have many features intended to ensure reliable interchange of data over an imperfect communication channel. Communication protocol is basically following certain rules so that the system works properly.

II. BACKGROUND OVERVIEW

We are aware of the fact that 40 pin microcontroller provide only 4 ports for external world. But for our complex requirement we need to interface a large number of peripheral devices which is not possible with a single microcontroller. The limitation of using single Controller is

- Limited code memory.
- A very few peripheral devices can be interfaced.
- Complexity in design.
- Time consuming.
- Error probability.
- Software complexity more.
- Debugging is difficult.

A. Existing System

The solution of above defined Problem is to use more than one controllers or multi controllers in a single project. Now communication in different controllers can be done mainly by two ways i.e. by using I2C protocol and SPI protocol.

B. Drawbacks of Existing System

- Synchronization of the clock is required.
- Data is transferred serially hence the system becomes slow.
- Master slave configuration is to be strictly maintained.
- Slaves cannot transfer data directly among each other as data transfer has to take place through master itself.

C. Proposed System

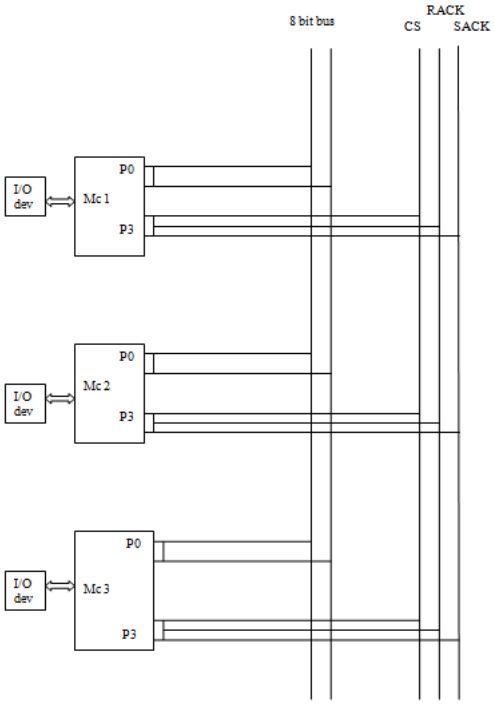
- We will be using 8-bit parallel data lines to transfer the data and 3 bit parallel line for handshaking signal.
- There will be no master Slave configuration.
- Each controller can communicate or transfer data independently.
- Priorities can be set by the controllers IDs.
- We had developed that algorithm by which up to 255 microcontrollers can transfer their data to desire destination microcontroller independently.
- Data transfer here means that any microcontroller can get data from any controller but with its

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 10 Issue 01, January-2021

permission. Similarly any microcontroller can send data from itself to any destination controller.

- The clock frequencies of these controllers may or may not be same but the protocol will work efficiently.
- The same program will burn in all the controllers without any modification.
- Addition and removal of any controller will not affect the protocol

III. PROPOSED SYSTEM



A. *Hardware Interface*

There will be 8 bit parallel Data lines that are connected to all controllers; we will call it the 'Bus'. There will be 3 bit handshaking or control lines which are defined below.

1. First signal line (CS) indicates the status of the 8-bit Data bus
   If CS = 0 bus is busy
   If CS = 1 bus is free
2. Second signal line (SACK) sender's acknowledgment
   SACK = 1 IDLE
   SACK = 0 ACTIVE
3. Third signal line (RACK) received acknowledgment
   SACK = 1 IDLE
   SACK = 0 ACTIVE

Hence there will be only 11 lines, for providing communication between 255 microcontrollers almost.

B. *Developing the Protocol*

Step 1: Define packet format.
Step 2: Defining function format or parameter list.
Step 3: Connection establishment.
Step 4: Packet transfer.
Step 5: Time slicing.

Defining packet format

DAddress	SAddress	PLength	Packet

DAddress [Destination Address]:-
It gives the destination address of the data to be sent. It is one byte long .The maximum number of destination that can be addressed is 255.
SAddress [Source Address]:-
It gives the source addressed of the data to be sent. It is one byte long.
PLength [Packet Length]:-
It gives the length of the data to be sent. The packet length field is one byte long.
Packet [Data]:-
This field gives the data to be sent. The length of the data to be sent is maximum up to 255 bytes.

C. *Function in the Protocol*

Init_Comm(void)

This function initializes all the lines as input line.

BUS=0xff

Initializes the Bus

CS=1

Initializes the CS line

SACK=1

Initializes the sender's acknowledgment

RACK=1

Initializes the receiver's acknowledgment

WaitTILLRACK (unsigned char Type, unsigned long Delay)
This function waits for Receiver's acknowledgment for specified Type for a particular time delay.
Type values can be -1
Type values can be -0
Suppose, if we want RACK for becoming H → L
Then we pass (1)
Suppose, if we want RACK for becoming L → H Then we pass (0)
In all, this function waits for Receiver's ACK until its (RACK's) values become unequal of passed value in the function within specified time .If time elapses before that then the function returns (0) value indicating unable to get response from, Rx MC. Otherwise successful ACK returns (1)indicating ACK for Rx found.
SHS (unsigned long Delay)
Sender's Hand Shaking Settlement function.
This function is called after sending the one byte information to the Rx. This function is used to finalize the H/S between both the Tx & Rx in the following way –

1. Make SACK = 0 after sending the info byte indicating the Rx that I have send the byte.
2. Wait for Rx's ACK until its becomes L.
3. Settle the SACK as 1 indicating Rx that I got the ACK sent by you.
4. Wait for response from the Rx that it got my

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 10 Issue 01, January-2021

settlement msg.

Finally both – RACK & SACK pins again becomes H. If SHS function returns (0) then it indicates that communication is broken else is it healthy.

Function for Send packet:

unsigned character Send Packet(unsigned character D Address, unsigned character S Address, unsigned character *P length, unsigned long CS, unsigned long ACK Delay)

1-data transfer is success

0-data transfer is failure

Function for Read Packet:

unsigned character Read packet (unsigned character y Add, unsigned character *S Add, unsigned character *Packet length, unsigned long CS Delay, unsigned long ACK Delay)

1-data received is success

0-data received is failure

ACK Delay is provided to avoid the receiver from entering into infinite loop.

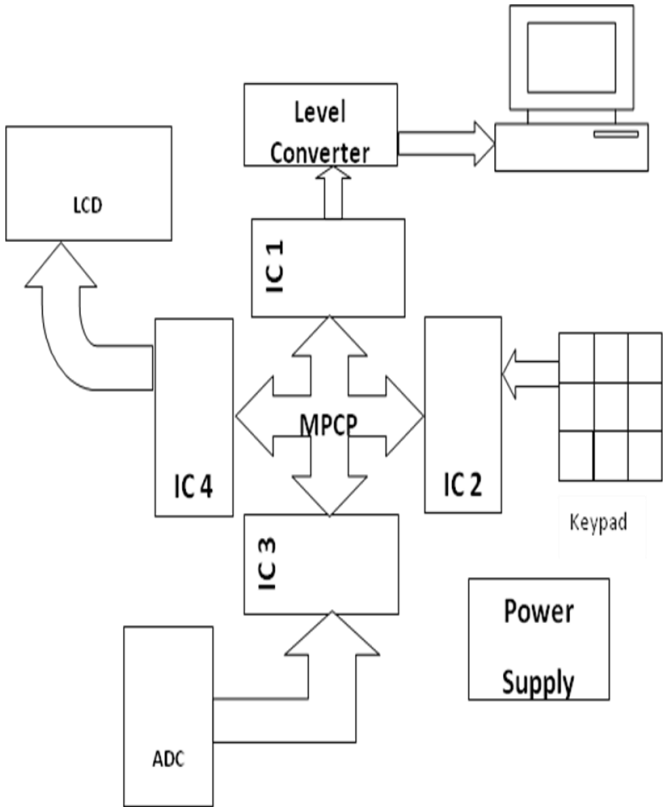
CS Delay is provided for slow receiver to cope up with the sender.

D. Algorithm for Sending Packet

- Initialize the pins
- Check for CS pin which indicates that if communication is already being done by other device
- If CS=0
  bus is busy
  if CS=0 is found then return (0)
- If CS=1
  bus is free, indicating failure of comm...
- Capture the Bus my making CS =0
- Capture Dadd to the bus /rx
- Make a SACK =0 indicating Dadd has been sent on bus
- Wait until Rx gives ACk or time out happens if timeout –then communication establishment failure
- Settle SACK
- Wait for Rx settlement
- Send SAdd on Bus
- Wait for H/S settlement
- Send Plength
- Wait for H/S settlement
- Send all the bytes in the packet & wait for H/S settlement
- Release the bus by making CS=1

E. Algorithm for Receiving Packet

- Initialize the pins
- Check for CS pin which indicates that if he Data is present on the Bus or not
- If CS=0 Data is present
- If CS=1 Bus is free
- Wait till CS=0
- Wait till SACK=0
- Check whether Data on Bus= My address
- Send RACK
- Wait for H/S settlement i.e. communication started
- Receive the sender's address from the Bus
- Wait for H/S settlement
- Receive the packet length from the Bus
- Wait for H/S settlement
- Receive the packets by executing a for loop (from 1 to packet length)
- communication completed



F. Auto Priority Resolving

For auto priority resolving in case if two or more transmitter wants to transmit or capture the bus at the same instant – this is not included in the code. Here is the explanation as to how this can be achieved.

CS line

↑t=0	↑t=0	↑t=0	←the same instant of monitoring CS pin
Tx -1 ID- 1	Tx-2 ID-2	Tx-3 ID-3	

Is CS is found 1 (i.e. free) by all the Tx then go into sleep mode / delay for (ID * constant Value) time.

↑t=0 ↑t=1 Sleep if c=1 ID * C Tx -1	↑t=0↑t=1↑t=2 sleep Tx- 2	↑t=0↑t=1↑t=2↑t=3 sleep Tx-3

Thus sleep time depends upon Tx ID .After coming out of sleep again monitor the CS pin. The lowest ID value gets the

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 10 Issue 01, January-2021

first chance of capturing bus. By the time the Tx are in sleep & when they will wake up they will again monitor the CS pin & will find that it is already captured by someone else.

IV. SCOPE & APPLICATIONS

MULTIPROCESSOR COMMUNICATION PROTOCAL developed by us is more efficient protocol than I2C and SPI protocols. Here data transfer is parallel leading to a very high speed of data transfer. Synchronization is not necessarily required i.e. ICs can have different clock frequencies. Addition and removal of new controllers can be easily done without any modification of software and it has many more advantages than others.

Applications and Scope:

- In Network Communication for microcontrollers,
- In Onboard communication in embedded systems.

Limitations

As generally all systems have some limitation, below are some listed ones from the proposed system.

- This protocol can support maximum 255 nodes only.
- As it supports parallel data transfer and hence it requires more pins or lines for communication.

REFERENCES

A. Embedded Books & Websites

[1] Myke Predko, *Programming and Customizing the 8051 Microcontroller*, Edition 1999, Tata McGraw-Hill, Page:157-167.
[2] Muhammad Ali Mazidi, Janice Gillispie Mazidi, *8051 Microcontroller and Embedded Systems*, Prentice-Hall, Page:183-193, 236, 243.
[3] Dogan Ibrahim, *Microcontroller Projects in C for the 8051*, Newnes, Page:29-161.
[4] Kenneth J. Ayala, *The 8051 Microcontroller ARCHITECTURE, PROGRAMMING and APPLICATIONS*, WEST PUBLISHING COMPANY, Page:131-197.
[5] Michael J. Pont, *Embedded C*, Edition 2002, Addison Wesley, Page: 57-87,217.
[6] www.beyondlogic.org
[7] www.electronicsforu.com

B. Electronics Books & Websites

[1] Ramakant A. Gayakwad, *Op-Amps and Linear Integrated Circuits*, 4th Edition, Prentice-Hall, Page:342, 417, 455.
[2] Robert L. Boylestad, Louis Nashelsky, *Electronic Devices and Circuit Theory*, 10th Edition, Prentice-Hall, Page:342, 417, 455.
[3] R.P.Jain, *Digital Electronics*, Tata McGraw-Hill
[4] www.electronic-circuits-diagrams.com
[5] www.circuitstoday.com
[6] www.circuitlake.com

C. Software Books & Websites

[1] Gary Cornell & Jonathan Marrison, *Programming VB.Net: A Guide for experienced programmers*, Second Edition 2002, ASPToday Publication, ISBN (pbk): 1-893115-99-2, 424 Pages.
[2] Dave Grundgeiger, *Programming Visual Basic.Net*, First Edition 2002, O'Reilly Publication, ISBN: 0-596-00093-6, 464 Pages.
[3] Evangelos Petroutsos, Mark Ridgeway, *Mastering Microsoft Visual Basic 2008*, First Edition, Wiley Publishing, ISBN: 978-0- 4701-8742-5.
[4] Brian W. Kernighan, Dennis M. Ritchie, *The C programming Language*, First Edition 1988, Prentice-Hall, ISBN 0-13-110370-9.

D. Other Books & Websites

[1] www.alldatasheets.com
[2] www.wikipedia.org
[3] www.keil.com
[4] www.hobbyprojects.com