ISSN No:-2456-2165

A Survey on Emulation of Communication Protocols in Microcontrollers

Ashok Rathish S[1], Dr. Paramasivam K[2]
[1]PG Scholar M.E- Embedded Systems Technologies, [2]Professor, EEE
Kumaraguru College of Technology, Coimbatore, India

Abstract:- Basic emulation of communication protocols involves the emulation or replicating the frames of the communication protocols using the port pins. This is useful when there is a particular need for a protocol inside a microcontroller where the required communication protocol is not present. The survey on emulation is suitable for the users to have a brief knowledge about the emulation before proceeding. This survey on emulation of communication protocols gives a brief information regarding the parameters, timing, and also the issues and problems faced during the emulation. A brief comparison was made with some different communication protocol emulation using a simple timer module. This will be helpful in concluding the behavior of each communication protocol on a simple timer module using which the protocol will be emulated.

***Keywords*:-** *emulation, microcontroller, timer, port pins, communication protocol.*

I. INTRODUCTION

The communication protocols play a major role in communicating with other devices, sensors, or even with more than one microcontroller. In order to achieve this, the specific communication device has to be present inside the selected microcontroller. In case, if a user selects a microcontroller which does not have any communication peripherals for a specific and desired purpose and suddenly there comes a requirement for the use of a communication protocol. One way of rectifying this issue is to go for a higher microcontroller with communication peripherals which in turn will increase the cost of the microcontroller. Another way is to emulate the communication protocol on a port pin using the help of the timer module present inside the microcontroller. Thus, in order for the emulation to work, the microcontroller should definitely have one or more timer modules based on the requirement. This helps in reducing the cost required for a communication peripheral. But this emulation using port pins also invites certain limitations and challenges. The proper emulation of the communication protocols will be highly useful in the automobile industries since it can reduce the cost of the microcontroller and also provides a platform to enhance the features of communication protocols such as number of data bytes transferred or received, speed of the frame, etc., The further expansion of this emulation is provided in different sections.

A. General Timer:

For the demonstration of this emulation, timer module is considered from PIC16F87XA as shown in the Fig 1. Here, the timer is capable of producing timed pulses based on the two registers which gets updated based on the user specified time. The timer will provide the timing between the high and low pulses using which duty cycle of the pulse can be determined the as follows.

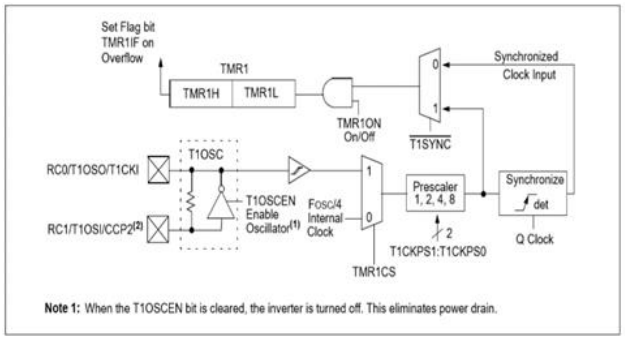

Fig 1:- Timer 1 of PIC16F87XA

Duty Cycle = Ton / (Ton+Toff) * 100

Where,
Ton – High pulse time
Toff – Low pulse time

B. *Port Pins:*

The port pins act as the input or output for the microcontroller. Here, the port pins can be as the output to get the frames of the frames and also to send in the frames to the microcontroller. Initially, the port pins if activated will be in high or low state based on the direction of the port. The port pins are highly important for the input/output of the frames which can be analyzed using another microcontroller or using oscilloscope.

C. *Communication protocols:*

The communication protocol is important in order to communicate between devices or microcontrollers. There are many communication protocols such as UART, SPI, I2C, CAN, Flexray, LIN, etc., Each communication protocols have their own unique frame structure and speed in which the frames are transmitted or received. The table 1 gives an idea about speed with which each protocol communication works on. The emulation of the communication protocol mainly depends on the speed in which the particular protocol is being transmitted. The speed of some of the communication protocols are given below:

Protocol	Speed
UART	Upto 115200 baud rate
SPI	Upto 10Mbps
I2C	Upto 3 Mbits/s
CAN	Upto 1Mbits/s
Flexray	Upto 10Mbits/s
LIN	Upto 19.2kbps

Table 1:- Speed of different communication protocol

The base form of the communication protocol frames always has a start bit, data bits and stop bits. Some protocols will have addition supporting bits to separate them from other protocols uniquely.

II. PARAMETERS

The input and output parameters are required for the emulation of protocols. Timing is the main factor that is required for the emulation.

For example, the maximum speed of UART is 115200 baud rate which is 115.2 KHz. This the time for a bit will be 1/115.2KHz which is equal to 8.68 micro seconds. Thus, each of the communication protocol has its own bit timing based on the selected speed.

The following parameters or the emulation has to be considered:

- The bit timing of the frame based on the speed
- Length of the frame
- Time difference between two frames
- Reception of the data from the incoming frames

It is also to be noted that the switching speed of the timer module is important, since the delay in which the timer is made to change its timing value also affects the length of the bits of the fame.

The following illustrates the frame of the simple UART frame which can be used as an example to specify the timing parameter.

A. Frame timing:

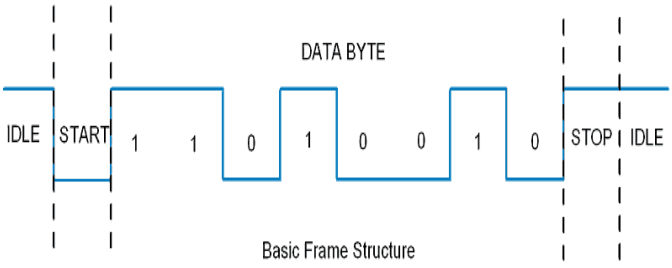

Fig 2:- UART frame format

Where,

Start – Start bit
11010010 – data bits
Stop – Stop bit

Here, the high state is considered to be the idle state, since start bit is always detected at low pulse.

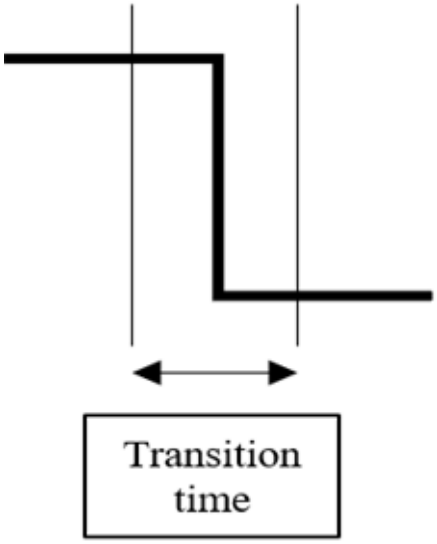

Fig 3:- Pulse transition time

Also, the transition time, which is the time taken to reach high pulse to low pulse or low pulse to high pulse plays a major role in the delay time of the frame timing. The pulse transition time from the fig 3 is also important in order to determine the extact delay time of the pulse.

The next major thing to be considered is the program on which the emulation runs on. The program should be well optimized in order to reduce the programming delays inside the frame.

B. Length of the frame:

Here, from the above bit timing, the emulated frame should match the entire time of the frame length. For most of the frames, there are some acceptable range for the timing of the entire frame and the emulated frame timing should be within that range.

C. *Distance between the frames:*

The distance between the frames should also be taken in to account since this will lead to overlapping of frames. This is most important for frames such as flexray, where the null frame will be transmitted continuously even though there is no communication between the host and the target module.

D. *Reception of the data from the incoming frame:*

It is to be handled that there should be no overwriting of new data on an existing data until the old data is read. Also, the reception is a crucial part in programming which has to be handled along the continuous check for the delay.

All the parameters and conditions as mentioned above has to be taken care during the emulation of communication protocol. For a simple protocol emulation, these parameters will be useful and also the user will be able to provide an architecture for both the transmission and reception. Here, emulation of communication protocol invites various issues

and challenges to be faced during the emulation. The challenges are mostly involved with the delay timing and also with the program which in turn creates a timing delay.

III. PROBLEMS IN EMULATION

The emulation of communication protocol using port in might seem simple but it has its own issues. Resolving each of the issues is very crucial since some of the protocols have to be emulated exactly as it is. The issues create a major problem during emulation since the emulation is time dependent and there are multiple factors that can affect the timing of the frame or the frame bits. Each of these problems and issues listed contributes a certain delay in timing for the frame which make it hard for the other modules or devices to determine the information from the frame. This in turn creates a frame that are not suitable for the specified communication protocol. Even software can create the timing issues that has to be eliminated during the emulation of communication protocol. Eliminating all the issues is not possible, but it is possible to reduce them to a maximum extend in order for the proper emulation using port pins. The following are the problems and issues faced during the emulation:

A. Transition timing delay:

The transition delay between the high and low pulse or the low and high pulse has the ability to alter the timing of the entire frame. Mostly, it affects the bits with the timing of nanoseconds, which in case, will never be produced as an output. The cause of this transition delay happens on the switching time of the gate from high to low or from low to high. Thus, the transition delay can even inhibit the production of a high or low bit on the output since the entire time for the bit will itself be a transition delay. The fig 4 gives an idea about where and how the transition timing delay will occur in a pulse. This transition delay is one of the major issues that has to be taken care of while emulating the frame since this might make the output to produce frame with undesired bits.

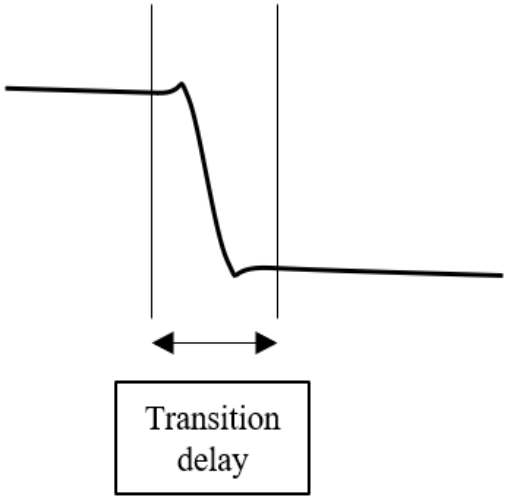

Fig 4:- Pulse transition delay

B. Program delays:

Programming delays mostly affect the bit timing but adding some delay time into the actual bit timing. This has a negative impact on the frame by increasing the length of the frame, if each bit has an added delay time to it. This can be rectified by optimizing the code and also reducing the usage of unnecessary registers as much as possible.

The following fig 5 illustrates the bit timing that occurs in a bit.

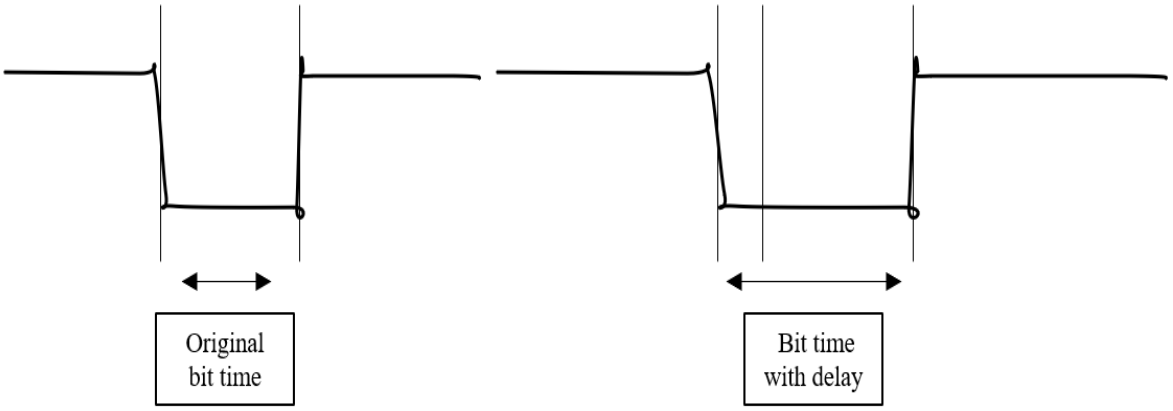

Fig 5: Pulse delay time

Here, as we can see that the second bit shown has an increased bit time due to the programming delay. Even if the bit timing is within the range of the bit timing, we have to make sure that the timing of the frame is also within the range.

The program delay also affects the reception part, where the reception of data bits is also affected. In reception part, loss of bits or reading the same bits twice issue might occur because of this programming issue.

C. Interrupts:

Interrupts are basically handled by the interrupt service routine whenever the interrupt occurs. When an interrupt occurs, the current program is stopped, the interrupt is handled and then the program is continued from where it is left. Interrupt plays a major role when multiple devices are running and the user has to change the value of certain register during the middle of the program.

Running only the emulation is possible but when the emulation is functioning along with several other modules, there comes an issue called interrupts, where there is a high possibility of the emulation being interrupted by the other module interrupts. In case, if the interrupt occurs in between the generation of frame, then the frames will not be complete, i.e., the frame will be produced until the interrupt occurs, then the interrupt is processed and the rest of the frame will be generated where the time for the interrupt handling will be delay time for the frame. Even handing the emulation inside a semaphore or a lock can produce certain issues. The issues occur when there is a continuous transmission of frame. Consider for flexray, which is a continuous protocol.

When the flexray emulation is performed within a lock where none of the interrupt can happen, any of the interrupts of other modules will not be handled, since the only program running will be the emulation. This becomes a major disadvantage when multiple modules are involved. The same issue can arise if the program runs on a polling method instead of interrupt service routine.

One way of handling this issue is by running this emulation on a separate core, while the other modules run on another core. This can make it possible for the emulation to run separately without any interrupts by interrupt service routine. This emulation using port pins will not be suitable when multiple modules are running along with their own interrupts. Thus, the interrupt has to be taken care in order for a proper emulation of the frames.

D. Real time constrains:

Even if the above mentioned time delays are reduced, there is a problem of delay that can happen if the system runs for a long time. For example, as of now we know that the delay of the emulation cannot be eliminated but can be reduced to a minimum. Thus, when a system with the communication protocol emulation runs for a long time, the minimum delays tend to add up and finally they create a big delay either at the beginning or end of the frame which impacts the functionality of the emulation. This issue may not be considered for an application of short lifetime but it has a major importance on the applications with long run time. This impact can be found on industrial applications, where continuous transmission of data is required in certain areas for the normal processing of the application.

There might be much more delays which involves with the time taken for the switching of the gates inside the modules, etc., which will have an impact on the delay time of the frame. Like always, eliminating these time delay is not possible but they can be reduced to minimum. Taking care of all the issues in a port pin emulation is possible but it is very difficult. If we take the program, optimizing along with the changes in the timing values is difficult since the timing values are required for the generation of frames. Also, the timing delay and issues may vary for different types of communication protocol and it is not necessary that all the communication protocols should face the same issue. It is to be noted that different microcontrollers have different properties such as clock frequency, fast switching gate, etc., which can help improve the emulation of communication using port pins. Thus, the selection of microcontroller architecture should also be taken into account.

IV. EVALUATION

Here is the evaluation segment which gives the information regarding the suitable application for different communication protocol to be emulated on.

- UART:
- DATA LENGTH - 8 bits
- BIT TIMING - 8680 micro seconds (max)
- CHARACTERISTICS - Non-continuous
- SUITABLE APPLICATION:
- Normal timer and a single port or double port is suitable
- Has the possibility of extending the data length beyond 8 bits (increases timing of the frame)
- SPI:
- DATA LENGTH - multiple bits (based on the version)
- BIT TIMING - 100 to 150 nano seconds
- CHARACTERISTICS - Non-continuous
- SUITABLE APPLICATION:
- Frame with low resolution or speed has a possibility of being emulated using a simple timer but the highspeed frames are not possible in a simple timer.
- Independent high speed timer is required for this emulation
- Flexray:
- DATA LENGTH - 254 bytes
- BIT TIMING - 100 to 150 nano seconds
- CHARACTERISTICS - Continuous
- SUITABLE APPLICATION:
- Since, flexray is continuous, there should not be any interrupt during the communication.
- Independent highspeed timer is mandatory for this emulation since timing and non- interruption of the frames play an important part here.

Each communication protocol has its own requirements and the user has to taken them into account for a proper emulation of communication protocol.

V. CONCLUSION

The emulation of communication protocol using port pin faces a lot of issues which can be reduced, but the guarantee of the proper working of the emulation will be very less. This brings down the percentage of possibility of emulating the communication protocol using a port pin of the microcontroller. This emulation of communication protocol on a port pin seems suitable for experimental levels and also for the development of some basic modules and applications. This emulation provides way for further developments and options o improve the emulation and also its efficiency. One point of development will be the timer module, where the timer can have its own control unit to handle its own programming. This is useful to reduce the load of the microcontroller since our program will be running on the CPU of the timer module. Also, the interrupts from other modules will not have an effect on the emulation since the emulation is being taken care of the control unit of the timer and not by the main CPU of the microcontroller. Only the initialization part of the timer with control unit will be handled by the main control unit of the microcontroller. The rest of the process will be handled by the control unit of the timer. Also making the control unit of the timer to run on a particular scheduling algorithm can be useful to process the information for the emulation. Here, the port of the timer can itself be used to emulate the behavior of the communication protocol. With a timer of this kind, the emulation can be done with highly minimized delay timing since the pulses are directly produced from the timer itself.

Furthermore, the emulation for high speed protocols can also be made possible using this type of timer with its own control unit to run the programs.

In the near future, further developments will be done on the timer modules to make it work separately and to emulate all types of protocols which will eliminate the use of the modules for the desired communication protocol. Furthermore, this can even replace the communication protocols for the industrial applications such as automobiles if the emulation proves to run for a long time. Further developments will be conducted based on the emulation not only for the communication protocols but also for other peripherals based on the requirements in the future.

REFERENCES

[1]. Umakanta Nanda & Sushant Kumar Pattnaik “Universal Asynchronous Receiver and Transmitter”, 3rd International Conference on Advanced Computing and Communication Systems (ICACCS -2016), Jan. 22 – 23, 2016, Coimbatore, INDIA, 2016

[2]. Maman Abdurohman ,Aji Gautama Putrada S ,Rizka Reza Pahlevi ,“Fast UART and SPI Protocol for Scalable IoT Platform”, 2018 6th International Conference on Information and Communication Technology (ICoICT), 2018

[3]. Shadi Al-Sarawi , Mohammed Anbar , Kamal Alieyan , Mahmood Alzubaidi , “The design of high speed UART”, 8th International Conference on Information Technology (ICIT), 2017

[4]. Anand N, George Joseph, Suwin Sam Oommen, and R Dhanabal, “Design and implementation of a high-speed Serial Peripheral Interface”, IEEE, 2014

[5]. Vivek Kumar Pandey, Sparsh Kumar, Vimal Kumar, Pankaj Goel, “A review paper on I2C communication protocol”, International Journal of Advanced Research, Ideas and Innovations in Technology, volume 4, issue 2, 2018

[6]. Zheng-wei HU, “I2C protocol design for reusability”, IEEE, 2010

[7]. Lukman Adewale Ajao, Mutiu Adesina Adegboye, Eustace M. Dogo1, Salihu O. Aliyu, and Danlami Maliki1, “Development and Implementation of Microcontroller based improved digital tier and alarm system”, International Conference on Information and Communication Technology and Its Applications, 2016

[8]. “Programmable timer for repeated work”, International Research Journal of Engineering and Technology (IRJET), 2016

[9]. “PIC16F87XA datasheet”, www.microchip.com, microcontroller datasheet, page – 53 to 61

[10]. Ilya Galkin, Olegs Tetervenoks, “The study of microcontroller based embedded system for smart lighting applications”, IEEE, 2014