Negligible Time-consuming RC5 Remote Decoding Technique

Conference Paper · January 2014

DOI: 10.1109/ICCCI.2014.6921832

3 authors, including:

Th. Birjit Singha
Indian Institute of Technology Guwahati
6 PUBLICATIONS **7** CITATIONS
SEE PROFILE

Lakshay Jain
Homi Bhabha National Institute
8 PUBLICATIONS **5** CITATIONS
SEE PROFILE

Negligible Time-consuming RC5 Remote Decoding Technique

Thockchom Birjit Singha[1]
[1]Department of ECE
Tezpur University, Tezpur
Assam(784002), India

Lakshay Jain[2]
[2]Department of Applied Physics
Delhi Technological University
Delhi(110042), India

Bikash Kant[3]
[3]Department of ECE
Lovely Professional University
Phagwara, Punjab (144402), India

[1]Corresponding Author E-mail: thbirjitsingha@gmail.com

Abstract - **Remote decoding techniques based solely on polling waste a considerable amount of time in the bit-reading process, which is undesirable. Better techniques involving interrupts have been proposed, but, these still waste some amount of precious execution time. In this paper, we propose a technique which consumes negligible time (few µs) in the bit reading process, thus, utilizing all the available time for execution of the main task.**

Keywords – Interrupts, Interrupt service routine (ISR), Polling, RC5 Protocol, Remote decoding, IEEE 802.3, IEEE 802.4

I. INTRODUCTION

The bit reading process for an RC5 protocol based remote, which follows IEEE 802.4 and IEEE 802.3 Manchester (bi-phase) encoding standard, is generally an auxiliary task performed by some other multi-tasking system. Wastage of execution time in bit reading process thus, hampers the execution speed of the main tasks that the controller is supposed to perform. With higher clock speeds, this causes the wastage of more and more execution cycles which remain unutilized for the intended tasks. This leads to reduction in efficiency of the overall system.

In order to reduce the amount of time wasted by the technique based purely on polling as proposed in [1], an interrupt based technique was proposed in [2] which involved using 2 interrupts - 1 external interrupt and 1 timer interrupt in addition to polling. On an average, the technique could successfully save approximately 49% of the time but due to partial use of polling, some time was still being wasted [2].

To completely eliminate time-wastage, we propose a new technique solely based on interrupts which does away with polling completely. This technique employs the usage of 3 interrupts – 1 external interrupt and 2 timer interrupts. The second timer interrupt serves as the more efficient alternative for the polling phase in the interrupt-based technique [2]. Thus, it is an improvised effort to eliminate the wastage of time due to polling.

II. ALGORITHM AND FLOWCHART

A. Algorithm

- Main Routine

Step 1: Initialize external interrupt in edge triggered mode and timer-0 and timer-1 in 16-bit mode
Step 2: Reset Bit-Count=0, Flag=0, Toggle=0
Step 3: Enable external interrupt, timer-0 interrupt and timer-1 interrupt
Step 4: Perform Main Task
Step 5: If external/timer interrupt occurs, call their respective ISRs
Step 6: If BIT_COUNT = 14, separate address bits and data bits
Step 7: Repeat Steps 4 to 6

- External ISR

Step 1: If Flag=0, initialize timer-1 for 3/4 delay else initialize timer-1 for 1/4 delay
Step 2: Return from the ISR

- Timer-0 ISR

Step 1: Read the received bit and copy it to flag
Step 2: Complement the received bit and save it
Step 3: Increment bit-count
Step 4: Initialize timer-0 with 1 full bit period and start it
Step 5: Return from the ISR

- Timer-1 ISR

Step 1: Disable external interrupt and timer-0 interrupt
Step 2: If Flag=1, discard the read bit, else save the bit and increment Bit-Count
Step 3: Initialize timer-0 with 1/2 bit period and start it
Step 4: Enable external interrupt and timer-0 interrupt
Step 5: Return from the ISR

978-1-4799-2352-6/14/$31.00 ©2014 IEEE

B. Flowchart

Figures 1, 2, 3 and 4 represent the flowchart for the above mentioned algorithm for implementing the proposed technique. Fig. 1 represents the flowchart of the execution of the main task. Fig. 2 represents the flowchart of External ISR. Fig. 3 and 4 represent the flowchart of Timer-0 ISR and Timer-1 ISR respectively.

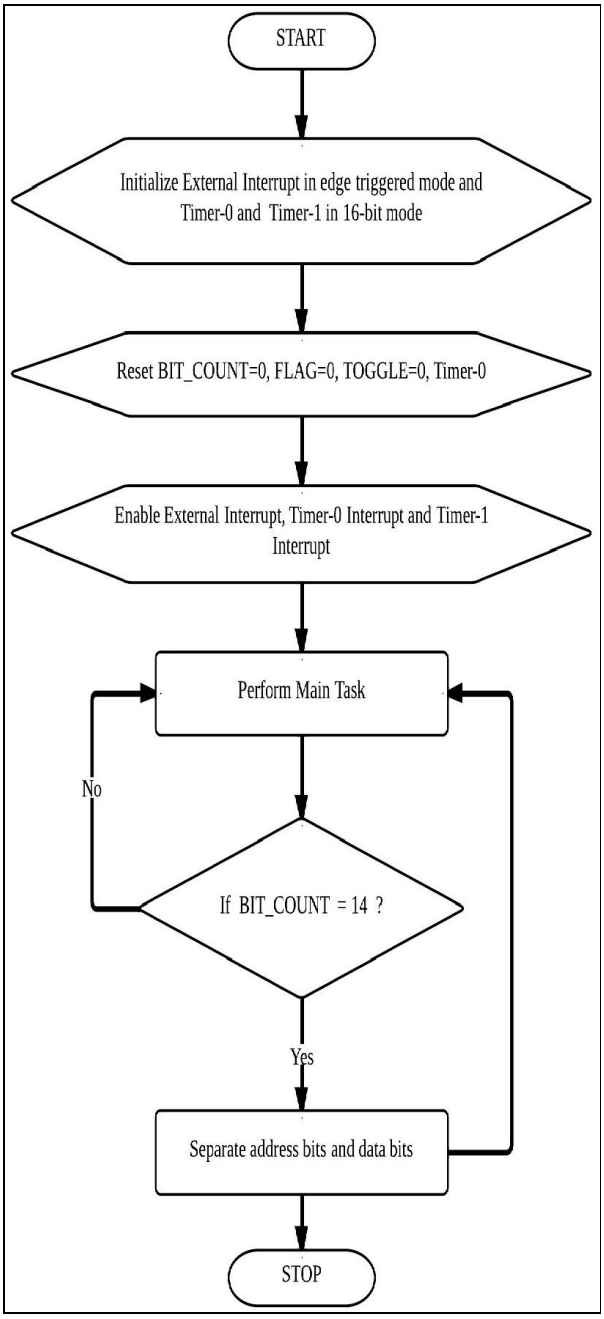

Figure 1. Flowchart of Main Routine

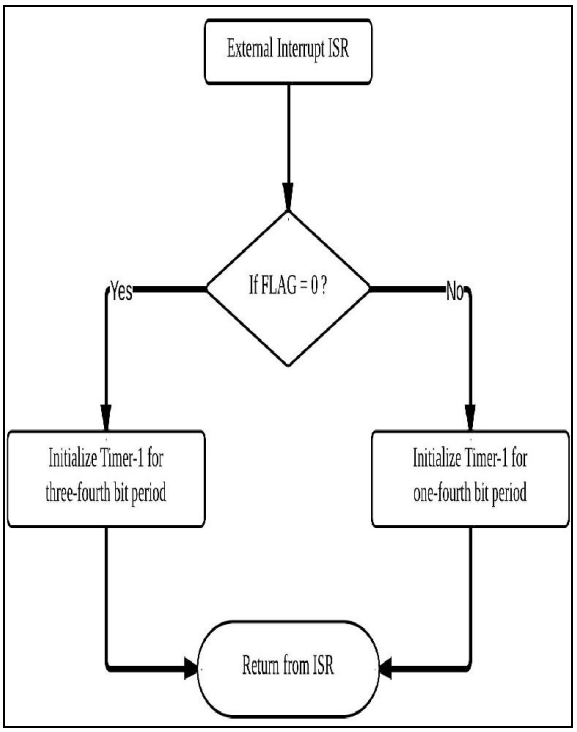

Figure 2. Flowchart of External ISR

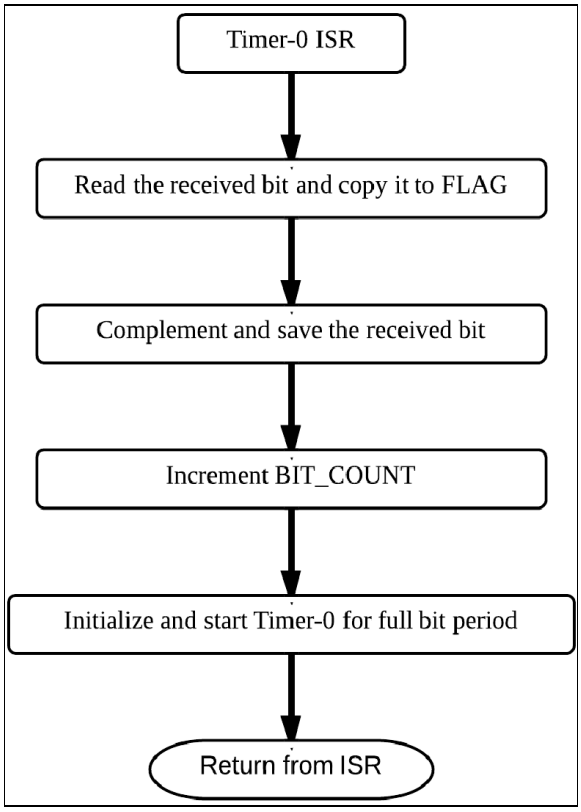

Fig. 3: Flowchart of Timer-0 ISR

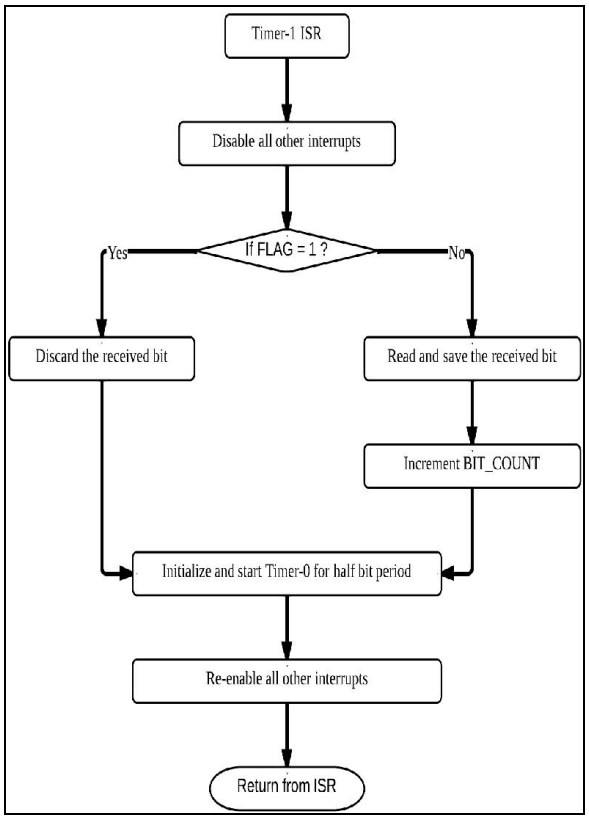

Fig. 4: Flowchart of Timer-1 ISR

III. BIT READING PROCESS

If the bit reading process uses the proposed technique, the decoding process occurs in the following manner:

- A '0' is read in Timer-0 ISR which is followed by External ISR and then, Timer-1 ISR.
- A '1' can be read in the following 2 ways:

1. If '1' is preceded by a '0', then it is read using Timer-0 ISR only.
2. If '1' is preceded by another '1' or if it is the 1st bit of the 14-bit sequence, then it is read using the External ISR followed by Timer-1 ISR.

Thus, for a '0' to be read, all the 3 ISRs are executed. However, a '1' may be read by the execution of External ISR and Timer-1 ISR or Timer-0 ISR only.

Table I gives the number of machine cycles required in the execution of the 3 ISRs.

IV. COMPARISON

The timing analysis of the proposed algorithm is as follows:

Case 1 - Average Case:

An average case would consist of a bit sequence containing 7 '0's and 7 '1's in a random order.

Thus, the total number of machine cycles required in reading the bit sequence
= 7 * 41 + 7*55
= 672 machine cycles
= 729.12 µs
= 0.729ms

Case 2 – Worst Case :

The worst case consists of a bit sequence in which the 2 start bits (1 1) are followed by 12 '0's.

Thus, the total machine cycles consumed in reading the bit-sequence in the worst case
= 2*41+12*55
= 742 machine cycles
= 805.07µs
= 0.805ms

Case 3 - Best Case:

The best case for the proposed algorithm occurs if the 2 start bits (1 1) are followed by 6 pairs of '0 1'.

TABLE I. NUMBER OF MACHINE CYCLES REQUIRED IN THE EXECUTION OF THE ISRS

External ISR	Timer-0 ISR	Timer-1 ISR
- Reset Timer-0 : 2 - If FLAG = 0 : 2 - Initialize and start Timer-1 for 3/4 Bit Period Delay : 9 - If FLAG = 1 : 2 - Initialize and start Timer-1 for 1/4 Bit Period Delay : 7 - Overheads (for jumps) : 8	- Reset Timer-0 : 2 - Read the received bit and copy it to FLAG : 3 - Complement the received bit and save it : 2 - Increment BIT_COUNT : 1 - Initialize and start Timer-0 for 1 Bit Period Delay : 3 - Overheads (for jumps) : 8	- Reset Timer-1 : 2 - Disable all other interrupts : 1 - If FLAG = 0 : 2 - Read and save the received bit : 2 - Increment BIT_COUNT : 1 - If FLAG = 1, discard received bit : 2 - Initialize and start Timer-0 for 1/2 Bit Period Delay : 3 - Re-enable all interrupts : 1 - Overheads (for jumps) : 8

TABLE II (A). TIME CONSUMED IN DIFFERENT CASES

Case	Time consumed by purely polling based technique(ms)	Time consumed by interrupt + polling based technique(ms)	Time consumed by the technique proposed in this work(ms)
Average	24.4475	12.446	0.729
Worst	24.4475	18.669	0.805
Best	24.4475	8.001	0.571

TABLE II (B). TIME SAVED W.R.T. EXISTING TECHNIQUES

Cases	Time saved (w.r.t. polling)	% Time saved (w.r.t. polling)	Time saved (w.r.t. interrupt + polling)	% Time saved (w.r.t. interrupt + polling)
Average	23.7185 ms	97.018	11.717 ms	94.143
Worst	23.6425 ms	96.707	17.864 ms	95.688
Best	23.8765 ms	97.664	7.43 ms	92.863

Thus, the number of machine cycles consumed in reading the bit-sequence for the best case
= 2*41 + 6*(55+19)
= 526 machine cycles
= 570.71 μs
= 0.571ms

Table II (A) and table II (B) summarize the results of the above-mentioned timing analysis and compare the proposed technique with other existing techniques based on timing.

NOTE: All calculations for converting machine cycles to equivalent time are based on the considerations that the μC used is 8051 at a clock frequency of 11.0592 MHz. For higher clock frequencies, the time consumed decreases. Time consumed however increases if the clock frequency is decreased.

V. CONCLUSION

This technique is aimed at reducing the time wasted in the bit-reading process to a bare minimum and, thus utilizing the time saved for performing the intended tasks assigned to the system. This in turn improves the overall efficiency of the embedded system for its intended use.

In this paper, we proposed an improvised technique of decoding RC5 protocol based remotes using 8051 μC (although the same algorithm can be used on other μCs as well) based solely on interrupts. The algorithm successfully implemented on hardware has also been discussed. Thus, by using the proposed technique, time wasted due to polling has been completely eliminated without the need of any additional circuitry by utilizing the additional resources already available in 8051μC. Comparison of the proposed technique with available techniques revealed that, on average, ***97.018 %*** time is saved with respect to the technique based solely on polling and ***94.143 %*** time is saved with respect to the technique proposed in [2]. Since this technique is based solely on interrupts, time is consumed only during execution of the ISRs. This takes only a few microseconds (μs), which is negligible in real – life application based system. Thus, the μC remains almost uninterrupted while performing its intended task.

REFERENCES

[1] Other, A.N., 2001. IR remote control codes (1) formats, protocols and (in) compatibility. Elektor Electronics Magazines Issue 297, March 2001.

[2] L. Jain, T.B. Singha and B. Kant, 2013,Interrupt Based RC5 Remote-decoding Using 8051 μC, Journal of Artificial Intelligence, 6: 187-190.