# A Responder Game Based on STM32F4

## Abstract

Our project implemented a responder game system based on the stm32f4-discovery board. The system consists of a control unit, which is discovery board, a display unit, which is Alphanumeric Display based on ht16k33, and a player unit, which concludes four LEDs and buttons.

## Introduction

The main purpose of our project is to build a responder game based on the stm32f4-discovery board and some additional relevant peripherals. Such a game system can be used in the primary school. Students can learn knowledge about mathematics while enjoy the joy given by the game.

Typically, our system consists of a control unit, which is the stm32f4-discovery board, a Quad Alphanumeric Display, four buttons, and four normal LEDs.

In our designed scenario, the general goal of the game is to see which player can solve a simple math problem first. In our game, we have a host and 4 competitors. The typical process of the game is as following: When the host presses START button, the problem is generated by microcontroller. Each player has a push button and a LED in front of him. Players answer the question by pressing down their buttons. The one who can give the correct answer will win the game. If players do not press the button again within the preset interval of time (i.e. 2 seconds) after their last press, the microcontroller will automatically end the responding process and judge whether the answers are right or not.

## Hardware

**Extra parts:**

1. Adafruit quad alphanumeric display

>>http://www.amazon.com/Adafruit-Quad-Alphanumeric-Display-Backpack/dp/B00L2X4JEW/

2. LED

>>http://www.amazon.com/Adafruit-Quad-Alphanumeric-Display-Backpack/dp/B00L2X4JEW/

3. Button

>>http://www.ebay.com/itm/10-Pcs-Tactile-tact-Push-Button-Switch-6x6x5mm-/221388102893?pt=LH\_DefaultDomain\_0&hash=item338bc25ced

4.Dupont wire

>>http://www.ebay.com/itm/Dupont-Wire-20cm-Cable-Line-Color-40Pin-40P-Test-Lines-Connector-/321642565395?pt=LH\_DefaultDomain\_0&hash=item4ae3640f13

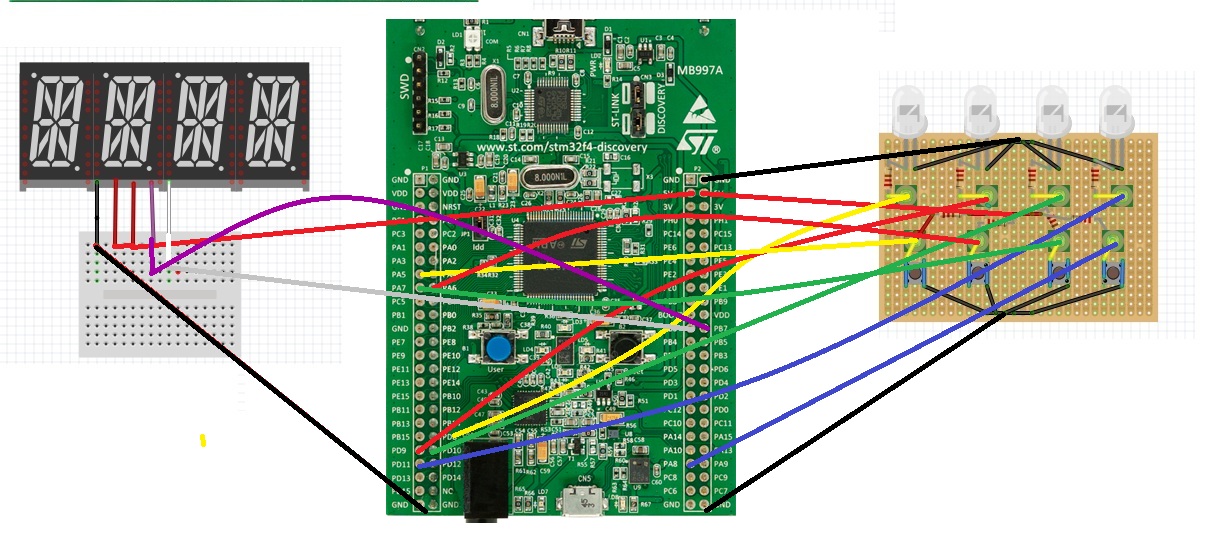
5.Pin header

>>http://www.ebay.com/itm/10-Pcs-40-Pin-2-54-mm-Angle-Single-Row-Pin-Header-/231443601650?pt=LH\_DefaultDomain\_0&hash=item35e31d18f2

6.Universal board

>>http://www.ebay.com/itm/10pcs-DIY-Prototype-Paper-PCB-Universal-Experiment-Matrix-Circuit-Board-5-x-7cm-/231444372553?pt=LH\_DefaultDomain\_0&hash=item35e328dc49

**Connection diagram**



|  |  |  |  |
| --- | --- | --- | --- |
| Pins Table | | | |
| Alphanumeric Display | Universal Board | Discovery Board | Function |
|  | GROUND | GND | Ground |
|  | 5V | 5V | High voltage |
|  | LED YELLOW | PD8 | GPIO output |
|  | LED RED | PD9 | GPIO output |
|  | LED GREEN | PD10 | GPIO output |
|  | LED BLUE | PD11 | GPIO output |
|  | BUTTON YELLOW | PA5 | External interrupt5 input |
|  | BUTTON RED | PA6 | External interrupt6 input |
|  | BUTTON GREEN | PA7 | External interrupt7 input |
|  | BUTTON BLUE | PA8 | External interrupt8 input |
| GROUND |  | GND | Ground |
| 5V |  | 5V | High voltage |
| 5V |  | 5V | High voltage |
| TX |  | PB6 | I2C SCL |
| RX |  | PB7 | I2C SDA |

## Software

**Standard peripheral library:**

"stm32f4xx\_gpio.h"

"stm32f4xx\_rcc.h"

"stm32f4xx\_tim.h"

"stm32f4xx\_exti.h"

"stm32f4xx\_rng.h"

"misc.h"

"stm32f4xx\_syscfg.h"

"stm32f4xx\_i2c.h"

In this project, we include eight standard peripheral libraries from previous lab as given.

**Third-party libraries for extra peripherals:**

"ht16k33.h"

In our design, we use chip ht16k33 to drive the Alphanumeric Display, which needs to be communicated with I2C, so we write a library file "ht16k33.h" for it.

Here is our reference from GitHub:

<https://github.com/devthrash/STM32F4-examples/blob/master/I2C%20Master/main.c>

Because we only need to write data from CPU to ht16k33, so just take some functions we needs, including:

void I2CInit()

void I2C\_start(I2C\_TypeDef\* I2Cx, uint8\_t address, uint8\_t direction)

void I2C\_write(I2C\_TypeDef\* I2Cx, uint8\_t data)

void I2C\_stop(I2C\_TypeDef\* I2Cx)

And add a function void initHt16() to initiate ht16k33 by ourselves.

## User’s Manual

After power on, the Alphanumeric Display will first display three “0”. Then the host presses the user button on the board, and the Alphanumeric Display will display a random arithmetic question. Then four players can start to answer question by pressing the buttons on the universal board. The LED in front of the button will light as soon as the players start pressing the button. And the times that players press the button represent the answer to the question. (Reminder: If time interval between two times that players press the button is longer than 1 second, the system will judge that player has finished answering.) The LED of the first one who answers the question correctly will flash. Then the game is over.

## Discussion

We divide our game system into several subcomponents, state machine design, interrupt for buttons, Alphanumeric Display display. And then, combine them together and debug. At the beginning of the project, we think about several different ways to implement each of the subcomponents. For example, to count the number of pressing buttons for each user, one way is using different ISR to handle different button pressing, one way is reading input register to check whether the button is pressed or not, one way is using multiple sources interrupt to track the button pressing.

**I2C for Alphanumeric Display:**

One difficulty is writing correct code for Alphanumeric Display display. At the beginning, we tried an insufficient way, but LED had no response. We thought that may be using GPIO to drive LED's pin is more straightforward and simple. So we write several programs to drive LED to display different alphabets. But because there are four Alphanumeric Displays sharing same pins to drive LED segment but different pins to ENABLE, so we have to come up with way to synchronize. Plus, if we used this way, we have to utilize at least 32 GPIO pins, which we occupy a lot on board. Realizing these problems, we decide to use the HT16K33 decode chip, which is designed for the multiple segment LEDs. Firstly, we find the user manual for this chip, showing that this chip only can be communicated in I2C mode, and there are several registers on chip could be efficiently configured. Secondly, we found some sample code written in C++ that could instruct us coding. Based on what we found, we wrote our own Alphanumeric Display driving function which made use of I2C library. But the Alphanumeric Display still has no respond. Third, we consult the professor, and search on the Internet and found out, there is strict timing requirement for I2C, that only using I2C library would lead to no response for the peripheral. Searching on Internet we found some useful resource, to make I2C easy use. To communicate with peripheral in I2C mode, we have strictly confirm the timing and the order. Sending peripheral address, register addressing, stop signal, or receiving ACK signal, all of these order should not have a bit change. So after modifying the program, we found the I2C peripheral could send ACK as well as board could receive it. Finally, the Alphanumeric Displays could display what we need.

**Debouncing:**

The big difficulty is debouncing. Although in LAB experiment, we had sample program for debouncing, at the beginning we thought this way is not accurate. we made use of hardware debouncing method, that is using RC circuit, capacity has ability to charging and discharging current, which largely relaxing the button bouncing pulse. In the test for the hardware debouncing, we found that it could somehow give more accurate output than original one, but still had some unaccuracy. No matter how we change the value of capacity and resistor, the mismatching error still exist. So we decided to discard this way and back to software debouncing method. Another way we try for debouncing, is using for-loop debounce. But in the test we found it still didn't work, maybe in the for-loop, the exact delay time was very hard to control and CPU can execute this for-loop in a very fast timing. After that, we modified our delay program based on the system clock, whose delay time could be accurately calculated by us. Apply this new way on our circuit, we found this debouncing method gave us circuit good stability and accuracy.

**Interrupt handler:**

Another big issue is how to handle interrupt, we had several options to do that, writing an ISR for each button, or more straightforward way, reading input register to acquire button information. Finding related part in the course material, we found out that there is kind of interrupt, which could be triggered by multiple sources. So in our project, we chose an interrupt which could be triggered by from Line 5 to Line 9, which is good for our utilization. In this single ISR, we just need to check the status on these Lines to locate exactly which line has an interrupt occurred. And this feature gave us good program flexibility. And finally we could have clear logic about the whole system and make all component add to the system.

**Judgement:**

Another big difficult is how to set the point showing the user finishes answer. We came up with several different methods to deal with that. For example, we could install another button on bread board, when user finishes answer, he must press that specific button once. Considering our circuit connection, we finally choose to use a simple system clock counter to judge the interval for each button press and the next, only if the interval exceeds this threshold will be judged by software that the user finished his answer. And then go to the logic to check the answer value is right or not. This way is quite simple and straightforward, exploiting our button circuit greatly.

**Soldering:**

Although we design our system in software level is quite stable and reasonable, we underestimate the impact hardware connection can make a great on the whole system. After soldering our wires, button, led along with Alphanumeric Display module with the board. We found that sometimes, even a small swing or touch on the wires causes button reading inaccuracy or led light down. But we still believe our system accuracy on software level. Because of time limited, we cannot resolder the whole system, or redistribute the hardware organization. We come up with several ways to optimize our hardware organization, we could be able to solder our button and led on the up-side and all the other wires on the down side. We could fix our system in a small box which make connection between board and module more stable.

## Results

**Power consumption:**

We connect the HT16K33 power pin and board 3V pin with multimeter to measure the current consumption for the peripheral chip. We found out that there is almost 27mA current consumption when the LEDs display a string. And for the breadboard peripheral, we measure the current is about 11mA. So, on average, there is 0.12W power consumption without the board. So it is good to use in the real application, which consume reasonable power.

**Debouncing performance:**

The mechanism we used for debouncing is timing delay, because the timing interval is so small that we cannot directly measure the exactly how long between two button pressed. So we use statistic method, pressing button 30 times, on the condition of reading accurate pressing number, how long the period would take. After several time measurement, the average time slot is 8.73s. Each time it takes 0.291s. Based on that, our design can accommodate our requirement.

In general, the components in our systems cooperate with each other, and work together. The whole system process data smoothly, and the system could have great usage.

## Conclusion

Basically, we implemented a responder game system based on the stm32f4-discovery board with all proposed functions. However the stability of this system can be incremented by reorganizing the distribution of buttons and LEDs on the universal board and optimizing the connections.