Who Wants to Be a Millionaire?

Shaun Fendi Gan, February 28, 2020

***Abstract*—A version of the popular game show "Who Wants to Be a Millionaire" was emulated using the PIC18F87K22 microcontroller. The game featured 8 question and answers, along with 3 different lifelines. A keypad, LCD, LED array, graphics LCD and external memory were implemented to replicate the game show experience.**

I. Introduction

GAME shows have been a staple for TV and radio entertainment for nearly a century. The first TV game show "Spelling Bee" aired on British television on the 31 May 1938 [1].

Who Wants to Be a Millionaire is a game show where contestants answer a series of multiple-choice trivia questions, in order to win the ultimate prize of a million dollars. The format of the show is simple: more correct questions lead to a larger sum of prize money. Lifelines are a special option in the show to aid contestants in tougher questions, these included Phone-a-Friend, Ask-the-Audience and 50-50, where in thte latter two incorrect answers were removed from the four [2].

Originally a British television show, the first winner was an American named John Carpenter on 19th November 1999. Carpenter famously did not use a single lifeline until his final question, where he called his father to let him know that he was about to win a million dollars [3].

Through this project, we aimed to recreate the format of this game show. This would allow us to progressively learn the Assembly language used to program a microprocessor. Additionally due to its modular format, extensions and extra features could be added if required.

II. High Level Design

The layout of the game can be seen in Fig. 1 along with it's usages summarized in Table I. These components combine with the EasyPIC PRO V7 prototype board to form a fully playable game.

The questions are first displayed on the monitor, through the USB-UART port seen in the green box, along with the multiple choice answers on the LCD screen. The Graphics LCD displays a 16 second count down timer and the player then enters their answer onto the keypad. If the user succeeds at a question, the LED array displays their progress of the game. The microprocessor displays a new question on the monitor and new answer choices on the LCD.

III. Software and Hardware Design

In the following paragraphs, the hardware design of the game will be discussed, displaying how all the components are driven and integrated into a cohesive system. Additionally, the software design and memory flow of the game will be covered, conveying subroutines that carry out specific functions, along with how information is carried through the file registers.

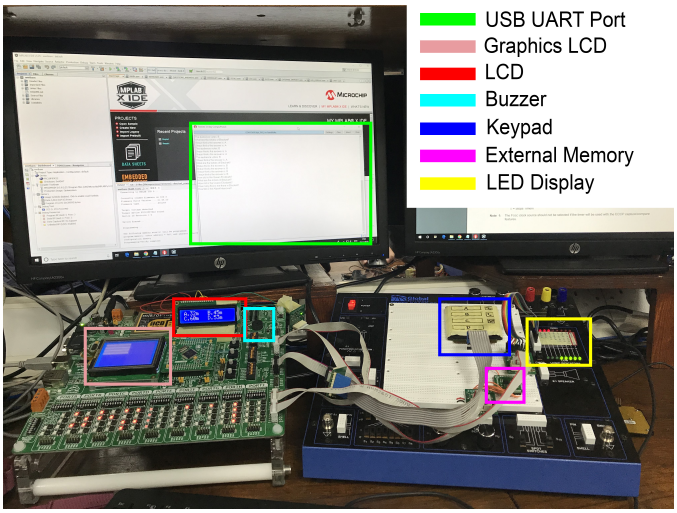

Fig. 1. Physical setup of the game. Component labels can be seen in the top right hand corner.

TABLE I
Hardware Components

Component	Usage
USB UART	Send questions to be displayed on Termite
GLCD	Displays a countdown timer
LCD	Displays answer options
Keypad	Inputs answers from player
External Memory	Stores 8 questions, acts as a question bank
LED Array	Displays prize money progress

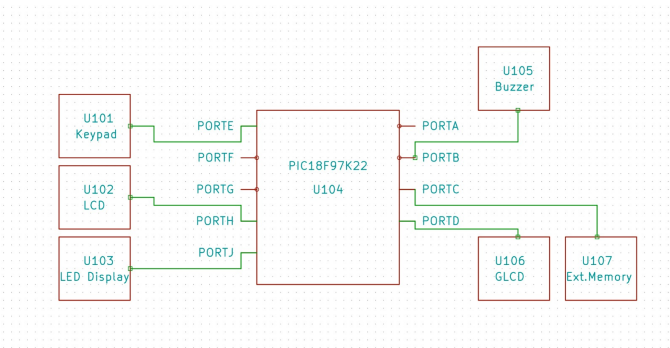

Fig. 2. Simple schematic diagram of the connections made to the PIC18 microprocessor. Each port represents and 8-bit parallel bus that can be used to communicate to the different components.

Advisor: BORG,Johan

A. Hardware Design

All the hardware components listed in Table I are connected to the PIC18 microprocessor ports as shown in Fig. 2. PORT B, D and E are special ports as can be set to a HIGH/LOW state. This feature was necessary for some components such as the keypad in order to process the input. Individual drivers were written for the hardware components, so that modular components could be added by calling a function from a separate file.

Keypad: The 4x4 keypad is split into 16 unique switches so that a button press sets the unique bit to low. From an 8-bit sequence, the first and last 4 bits can be used to determine the key pressed. Using the first four bits, the button's row was determined by noting the bit set to low. Subsequently, the column was found by finding the low bit in the last four bits. A 4x4 keypad was relabelled so that the first 3 columns and rows corresponded to answers choices, and the last column corresponded to lifelines/ special functions. The hex value used for each key can be seen in Fig. 3. Polling was implemented as the most efficient way to wait for a user answer.

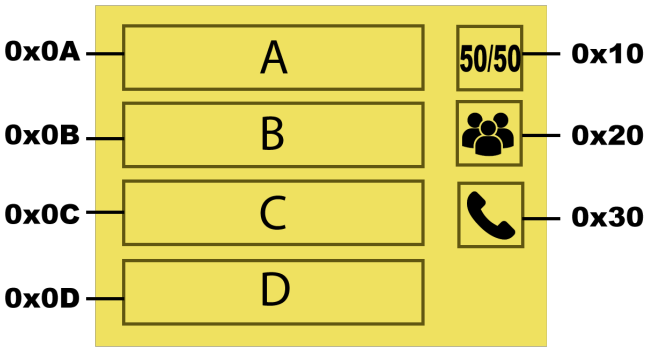

Fig. 3. A 4x4 keypad was used where the first three column and rows all corresponded to multiple choice answers ABCD. Each with the respective hex value of 0x0(character). The last column was used for three lifelines: 50-50, Ask-The-Audience and Phone-A-Friend. Their respective hex value used can be seen labelled above

USB-UART: A UART via USB connection was used as form of serial communication to the computer. UART functions like most typical serial transfers with a start and stop bit, with data wedged in between. This was connected to the Termite serial port on COM4, at 9600 kbps, and was used to display our questions from the external memory bank.

LED Array: An array of eight LEDs was used to display the player progress of the questions across eight questions, corresponding to the 8 bit output of PORT J. Starting from the first correct answer, a 0x01 value was sent to display one LED. If a question was right, one bit was shifted along depending on the question number and added to the value on the LED display. This simplified calculations as it avoided the need to store eight fixed constants that would be added to each correct answer. This process of bit shifting is also known as rotating, the 'RLCF' Assembly command was used for this.

LCD: A 28 character LCD display was used to display the four multiple choice answers. To account for the limited space, each answer was limited to less than five characters, and was padded with spaces if it was four characters or less. This consistent answer length allowed for all the answers to be stored as a complete table in the program memory. Therefore every 20 characters would be a new set of answers to be output.

Initially, the project plan only included the implementation of the hardware described in the previous paragraphs. However due to the efficiency of our progress, the hardware was improved by incorporating an external memory chip, and a graphics LCD (GLCD).

GLCD: The WDG0151-TMI-VN00 GLCD was used as a count-down timer within our hardware, however could be easily extended to display the questions and answers by using a character lookup table. The display consists two pages with 8 rows each, each row had 53 columns of 8-bit characters that could be written to sending a hex character. This structure is more clearly portrayed in Fig. 4.

Although not explicitly stated in the data sheet, the row addresses starts from 0xB8 to 0xBF, and the column addresses ranges from 0x40 to 0x7F. The cursor needs to be pointed to these locations for accurate writing. The GLCD automatically moves one character to the right when written to. Additionally, a high to low pulse is needed on the enable pin sends the command or data from port B to the GLCD.

Furthermore, it was found that the bit values should be written to LAT B, instead of PORT B. This is because writing to PORT B writes it to the pin, and then proceeds to read the output. When setting many pins at once, this could cause pins to be set the wrong values. Using LAT B instead evades this issue.

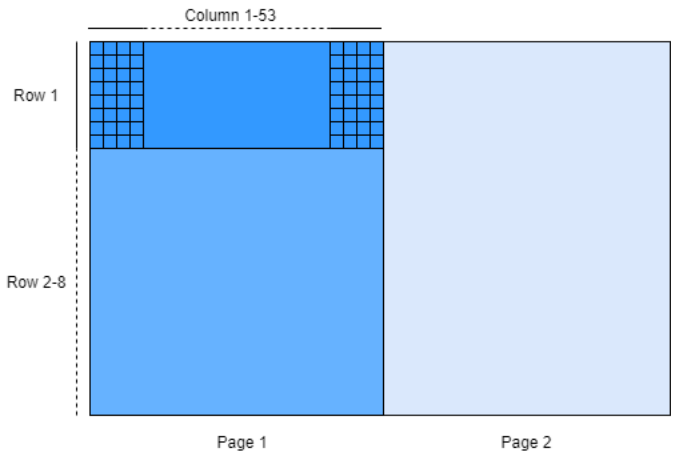

Fig. 4. A graphics LCD was used as a countdown timer for the game. The display is split into two pages with 8 rows, where each row has 53 columns, that are 8-bit long.

External Memory: The CY15B104Q 4-Mbit chip was used to store all questions. There was sufficient memory within the data memory of the PIC, however this appeared as a reasonable extension for the game. The external memory chip contained 0x07FFFF address locations, corresponding to 13,107 characters that could be stored.

The memory locations were segmented into three addresses: a low, mid and high address. For the convenience of our

game, the low addresses stored the question characters to be displayed, while the mid and high addresses represented the addresses where the question is stored. Each unique combination of mid-high address stored a separate question. Thus to increment between questions, only the mid and high addresses would be changed, whilst the question characters in the low addresses would be read in the same way.

A serial port interface driver was written in assembly in order to communicate with the external memory. The PIC18 was externally wired from PORT C as directly mounting it to the board clashed with the chip-select pin on PORT E which is required for the keypad.

This external memory used was non-volatile, allowing you to create multiple question banks that can be easily swapped to change the game experience.

B. Software Design

Moving on to software, the overall flow of our software design can be seen in Fig. 5, where setup and lifeline subroutines are demonstrated in Fig. 6. In the following paragraphs, the overall flow of the software will be described.

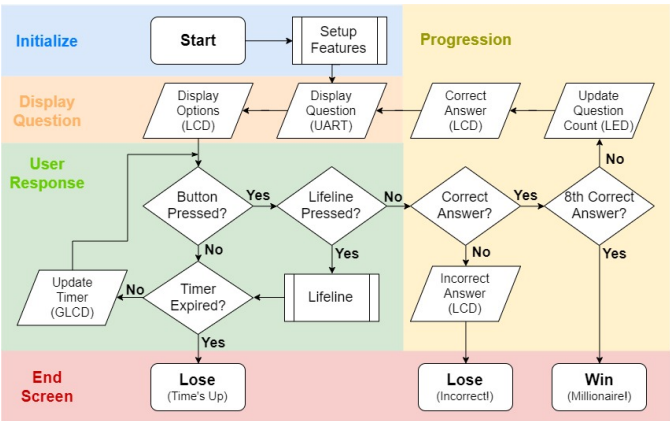

Fig. 5. Above shows the software structure for the Who Wants to Be a Millionaire Assembly game. Parallelograms represent inputs & outputs, double boxes represent a subroutine described in Fig. 6 and diamonds are yes-no progressions.

Software was split into four main sections. Firstly in initialisation, individual drivers for the hardware were run to setup the components, with questions and answers were stored to program memory. The order of the setup can be seen in Fig. 6.

Secondly, the question is then displayed on the screen, and the answer options are displayed on the LCD. Thirdly, the software waits for a button press on the keypad and polls for 16 seconds. If the timer runs out, the game updates to display a "Time's Up" message.

Alternatively, a lifeline option could be called to branch to a new subroutine. 50-50 generates a random numbers from 0-2 and removes two of the wrong options, Ask-A-Friend generates one random number from 0-3 and outputs a random answer, and Ask-The-Audience provides the correct answer. The flow diagram of this can be seen in Fig. 6.

Finally, the game is progressed depending on the user's answer. If it is incorrect, an end-game display showing "Incorrect!" terminated the game. If a correct answer is chosen, the LCD displayed "Correct!" and the UART port is updated with the new question and the LCD with the new answer. Additionally, the LEDs are updated to show progression and the timer is reset to 16 seconds. After 8 correct answers, the game ends to a congratulatory "You are a Millionaire!".

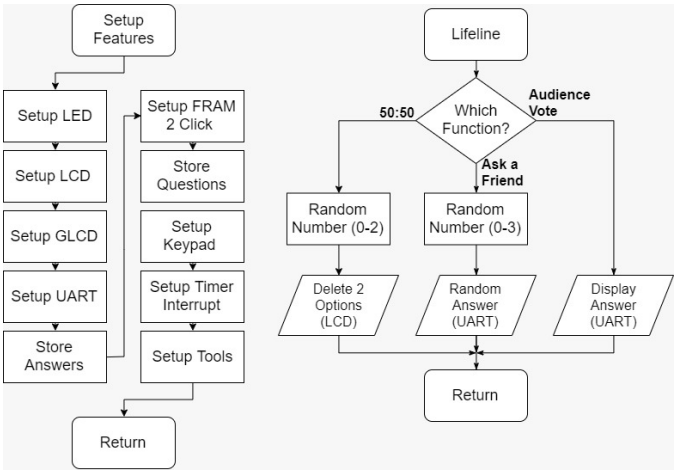

Fig. 6. Two key subroutines are used in this game. The first was used to setup all the components in the hardware components (LHS), the second was used to branch to a special function or lifeline (RHS).

Aside from running routines that performed arithmetic on the W register, two key software methods were implemented in order to incorporate complexity into our game: 1. Timer/Interrupts and, 2. Random number generator.

Timer: An inbuilt timer on the PIC18 was implemented in order to clear the LCD after a fixed time, update the GLCD to show time development, and generate random numbers used in two lifelines. A 16-bit timer of a prescale value of 1:256 was used to cause an interrupt roughly every one second.

By writing interrupt code in the 0x0008 address, updates were able applied to the software independent to the primary string of code. An incrementing counter was used in the interrupt block where it would branch to an end game screen if it reached 16 seconds. An additional subroutine was also added to update the time progress bar on the GLCD every second.

Random number generator: Generation worked by reading the lower 8-bits of a 16-bit timer. These lower 8-bits are put through a modulo function, where a fixed value is subtracted from the value until it reaches a certain threshold. For example, if we need a number between 0-3, the number would be subtracted by 4 repeatedly until it possessed a value between 0-3. This was implemented in the 50-50 lifeline to randomize the deletion of options, and in the Ask-a-Friend option, which gives a random answer for a question.

C. Code

Code for this project can be found at https://github.com/shaunfg/millionaire-lab/tree/dev/ext_rework

IV. Results and Performance

The game performed well and was smooth in its performance. In some walkthroughs, the game would get stuck on the progression to the next question, however would be solved by placing delays in between the subroutines. Furthermore, all lifelines would successfully perform their function and branch back into the main stream of questions and answers.

From five unique people, the average prize money won was £1000 (question 4/8).

V. Updates, Modifications and Improvements

To improve the simulation of the game show, a buzzer was sought to be implemented but wasn't completed in time. This was intended to be used during the Phone-a-Friend lifeline. On its own with only one timer used in the software, it was easily implemented. However, when merging it into all our software, we were unable to apply two timers/ interrupts at once. Potential buzzer connections and hardware can be seen in Fig. 1 and 2.

Additionally, the GLCD could be used to display more complex and interesting images if given more time on the project. It would have also been rewarding to gather more data on player progression in our game. Moreover, additional question banks could be added to our game and the subroutines with our code could have been more standardly named.

In terms of updates, our initial project plan was greatly updated in order to encompass more complexity and interesting components into our game.

VI. Conclusion

In conclusion, the original aim of the project was to create a game show like experience and to learn how to better program microprocessors using Assembly. As described in the report, the primary aims of the course have been achieved through this project. The integration of all the modular hardware components was a success for the game, with additional aspects we would like to add if given the chance. Our primary set-back was if we were able to setup a high-low priority timer system in time for presentation.

Overall, a fully functional game with six different hardware components that are all driven using microchip assembly was achieved from the duration of this project.

VII. Product Specifications

- Display ASCII characters on an LCD display
- Access pre-stored questions banks from using non-volatile external memories
- UART serial communication via USB to ports
- Display graphics on a graphics-LCD
- Random number generation for unique experiences for the games
- Modifiable 4x4 keypad for different inputs, up to 16.
- LED array to show game progression

References

[1] UK GAME SHOWS: Spelling Bee. http://www.ukgameshows.com/ukgs/Spelling_Bee_%281%29 accessed on 21st November 2019

[2] *"Who Wants to Be a Millionaire". Gameshow Hall of Fame. GSN. 21 January 2007.*

[3] Vigoda, Arlene. Million-dollar winner untaxed by celebrity. *USA Today. Gannett Company, 22 November 1999.*