Practice 3

Matrix keyboard, asynchronous serial port and DMA

3.1 Objetivos de la práctica

In this practice, we continue to expand the catalog of devices that we control. First, we will see how to configure and use the matrix keyboard. Next, we will focus on a standard serial communications device, UART. This device will allow to communicate with laboratory board with the PC via an asynchronous serial protocol. Also, we will learn how use the Direct Memory Access (DMA) controller for to copy directly the data received through the port in memory without any intervention of the processor. The main objectives of the practice are:

- Learn to manage the matrix keyboard
- Understand the basic concepts of asynchronous serial communication.
- Know how configure the UART unit S3C44BOX.
- Understand the basic concepts of direct memory access (DMA).
- Know how configure the DMA controller of S3C44BOX.

3.2 Tasks to do in the exercise

All information necessary for configuring the matrix keyboard, UART and DMA is in a separate document submitted with documentation of practice.

In this instriction manual will only describe the expected behavior for student's projects without indications of how to do it. Studeents can start from the codes provided by professors, but studetns are completely free to modify them as they need.

This exercise is comoposed by three parts (the last one is optional) all of them similar in functionality: a simplified version of the game *Mastermind*. In the game *Mastermind* a player puts a key (in our exercise just only 4 digits), the opponent must guess (in our case, we will not limit the number of attempts). In the original game, user receives after each attempt a *feedback* which indicates how much the received attempt approaches to the actual key code. In our exercise, only two answers/feedbacks are possible: right or wrong password.

In the first part, we will use the matrix keyboard to enter the key but ALSO enter the successive attempts. In the second part, attempts will be made from a PC connected via serial port. Finally, the third part will be a modification of the second using DMA.

3.2.1 Part 1: Keyboard management

In this part, we use only the keyboard (and no a serial port) and the 8-segment display and the timer 1. Although the keyboard is not numbered, assume that the key in the upper left corner is the`0`, the key in its right the `1` ... until the key on the lower right corner which is the `F`. Figure 3.1 shows the execution flow that must follow the application. Next, we describe it step by step:

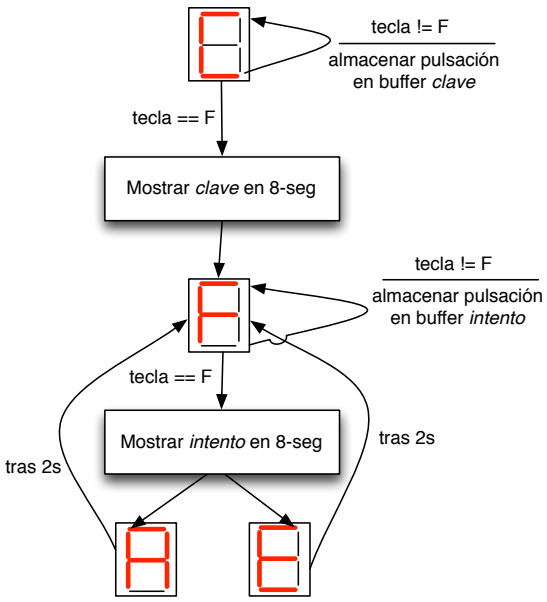

Figure 3.1: Execution schema of the application

At the begining of the execution, the 8-segment display shows a letter `C` indicating that it is waiting for the introduction of the password.

The user begins to insert a password attempt in the matrix keyboard. User can make as many keystrokes as he/she wish, but system only takes into account the last 4 keys pressed before pressing the `F` key. The digits entered are stored in a *buffer*[1] referred as `password`. Upon completion of this step, the 4-digit password is stored in that buffer.

Atter the last step, the character *F* will be show in the 8-segment display the introduced attempt when the user presses the `F` key, (up to this point, the display showed only character `C`). It should show one **digit each second** and for this **timer 1 must be used** (it is forbiddedn to use the `Delay()` function).

After the previous step, the 8-segment display will show a `F` to indicate that user can proceed to try guess the password.

Again, the user press keys (at least 4) and finish by pressing the `F` key. The digits entered are stored in a *buffer* referred as `attempt`, so the 4 digits attempt be, at the end of this step, stored in the *phattempt buffer.*

The attempt introduced by the user will be shown in the 8-segment displays when the user presses the ***`F`*** *key (up to this point, the display showed the character* ***`F`****). It should show one* ***digit per second*** *and for such task must use the* ***timer 1*** *(it is forbidden to use the* ***`Delay()`*** *function).*

After the previous step, the 8-segment will display the character ***`A`*** *if the user guess correctly the password, or show* ***`E`*** *in other case. In either case, the display will remain constant for 2 seconds, after that will show the character* ***`F`*** *to show that system is pending for next attempt.*

Even astudents have absolute freedom when designing and implementationing the practice, ***it is recommended to follow the following steps:***

1. *First step, complete the matrix keyboard management code provided by the preofessors. Try that after each key pressed, be immediately reflected in the 8-segment display (no using any other source of interrupt that the keyboard itself).*

2. *Instead of displaying the result on the display, store it in a* buffer *in memory (can be a simple array, but it is better a data type that works like a fifo which discards the*

[1] Students are free to decide how to implement the *buffer*. It must be an *array* which stores at minimum the four keys, but it is recommended to create a specific data type for this purpose

oldest element when introducing a new one). Confirm, through debugging, that the contents in the buffer *in memory are as expected.*

3. *ENABLE the* timer 1 *interrupts and program it to show the contents of the previous* buffer *in the display, with a frequency of one element every second (approx.)*

4. *Everyone of the elements of the application are implemented:* buffer*, the keyboard manager and the* timer *manager. Now, next step consists on think carefully about the design and integrate these elements.*

3.2.2 Part 2: serial port manager

The functionality of this second part will be very similar to the fincionalilty on the first part, but, in this part, the attempts to guess the password have to be made from the PC instead of using the matrix keyboard. The PC has to be connected via serial port. To communicate with the PC board is not necessary to make any extra connection, since the same USB port used to dump the code will be used as a virtual serial port.

We will use the terminal program `Termite`*, already installed in the laboratory, to establish the connection setting correctly, it must be done accordingly to the initialization done on the board (baud, parity and stop bits ...).*

Figure 3.2 shows the execution flow, as detailed below:

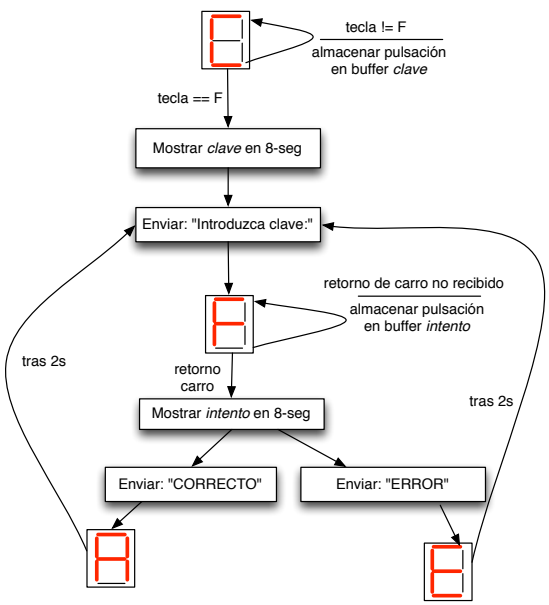

Figure 3.2: Execution flow of the aplication

At the beggining of the execution, the 8-segment display shows the character `C` *indicating that it is waiting for the introduction of the* attemt*.*

The user begins to introduce the password in the matrix keyboard. The user can press as many keys as wish, but only the last 4 before pressing the `F` *key will be taken into account. The digits entered are stored in a* buffer[2] *referred as* `password`*. Upon completion of this step the 4-digit password is stored in that buffer.*

The by the 8-segment display will show the password when the user presses the `F` *key, (up to this point, the display showed only the character* `C`*). It should show one* ***digit each second*** *and for such task must use the* ***timer 1*** *(it is forbidden to use the* `Delay()` *function). In addition, it will be sent through the serial port the* ***string:*** `Enter a key`*, which will appear in the* Termite *terminal in the connected PC. Also, the 8-segment display will show the character* `F` *to indicate that user can proceed to try guess the passsword.*

[2]Students are free to decide how to implement the *buffer*. It must be an *array* which stores at minimum the four keys, but it is recommended to create a specific data type for this purpose

Now, the user must enter four digits through the Termite *terminal and press 'intro' on the PC. At the reception, the board should be storing each digit received thorugh the serial port on a* buffer *referred as* ***attempt*** *(storing the last 4 digits). This step ends when be detected the character of* newline *between the bytes received thorugh the serial port.*

This functionality (read byte by byte from the serial port to find the ***end of line*** *character) is encapsulated in the* **readline()** *function, which will follow the following pseudocode:*

```
int readline(char* buf, int maxsize) {
pos = 0;
repeat {
a = leer_byte_de_puerto_serie();
buf[pos] = a;
pos++;
} while ( a != '\n' && pos < maxsize)
return pos;
}
```

The **readline()** *function will receive a pointer to a memory area where store the bytes received from the serial port (for example, an array) and the maximum number of bytes to read. This function finishes when the* ***end of line*** *character is detected or when be read the maximum of bytes without encountering such character. It returns an integer that indicates how many bytes have been read (including the* ***end of line*** *character).*

After the previous step, the display 8-segment will show the attempt received. It must show one ***digit per second*** *and for such task must use the* ***timer 1*** *(it is forbidden to use the* ***Delay()*** *function).*

It will be sent ***the string:*** ***CORRECT*** *through the serial port when the attempt received is correct (it match with the passord), and the character* ***A*** *will be shown in the display. If not correct, it will be sent* ***the string:*** ***ERROR*** *through the serial port and the character* ***E*** *will be shown in the display. In either case,the display will remain constant for 2 seconds, and after that will display the character* ***F*** *waiting for the next attempt (whether it has been successful or not) and writting* **the string** ***Enter password:*** *through the serial port.*

Remind that students have absolute freedom to design and implement this part, but it is recommended to test each component separately (in this case, the operation of receiving and sending though the serial port) before integrating all the elements.

3.2.3 Part 3: Using DMA (optional)

*This third part is identical to the previous one but the DMA must be used for sending strings ('****CORRECT****', '****ERROR****' and '****Enter password:****') through the serial port. It means, the ARM should only need to provide to the DMA controller, the start address of the string and its length, being the CPU only notified at the end of the transfer process.*