Practical Execution – Writing the System Code in a Portable Way

Introduction:

In this experiment, we aim to develop a system that integrates a hardware accelerator (DMA) for efficient data manipulation and processing. Traditional data processing on structures typically relies on load and store operations, which can be inefficient in terms of time and energy. By harnessing the combined capabilities of a CPU and a DMA within a low power MCU system, this setup aims to minimize energy consumption and improve response times. This approach exemplifies how hardware accelerators can enhance system performance in practical applications.

Hardware Connections Required:

- PB2 - PB0 buttons are connected to controller pins P1.2 – P1.0 respectively.
- LCD: The D7-D4 are connected to the pins P1.7-P1.4 respectively (the LCD operates in four-bit mode). You will need to connect three control lines of the LCD (P2.7, P2.6, P2.5) to the respective pins and update the code for the LCD in the model accordingly.
- Keypad: Connect the keypad to P10 and the interrupt pin to P2.1. See the appendix on the keypad.
- The system's architecture must be based on a simple FSM architecture as described in the preparatory report. Pressing any one of the three buttons should trigger one of four actions given an external interrupt request.
- The code should be portable between the different MSP430 chips by only changing the BSP layer.
- Writing the LCD and the Keypad driver function should be located in the HAL while the function of writing a string based on physical input should be located in the API layer.

FSM System Requirements:

You should define a char- two-dimensional static array in the main layer.

Char **data_matrix_in**[M][N] = {

" An apple a day keeps the doctor away",

"climb on the bandwagon",

"Dot the i's and cross the t's",

"He who pays the piper calls the tune",

"The pen is mightier than the sword",

"The pot calling the kettle black",

"shed crocodile tears",

"Close but no cigar",

"Cut from the same cloth",

"Strike while the iron's hot"

};

- User Input Storage: Define a character type variable idiom_recorder with a capacity of 32 bytes to store user-entered phrases.

FSM states :

State 0: Reset to sleep mode either by pressing the RESET button or after all functions have completed.

State 1: On pressing PB0, allow the user to input a phrase using the LCD and keypad. This phrase, limited to 32 characters, should be saved in idiom_recorder.

Note : The following state will end after clicking only when moving to a different state.

Keypad

1/G	2/H	3/I	C/J
4/K	5/L	6/M	D/N
7/O	8/P	9/Q	E/R
A/S	0/T	B/W/X	F/Y/Z

State 2: Triggered by pressing PB1, activate a swap function to exchange lines (j,i) within the data_matrix_in. This operation should be executed exclusively using DMA, with the result stored in a new matrix called data_matrix_out. Display this matrix on the LCD once the swapping is complete.

Note : This state should be defined after the array strMirror is displayed on the LCD screen

State 3: On pressing PB2, use the block transfer mode of the DMA to animate a 'moving LED' pattern across an array of LEDs, shifting from right to left with a 0.5-second delay, triggered by TimerB (TACCR2_CCIFG).