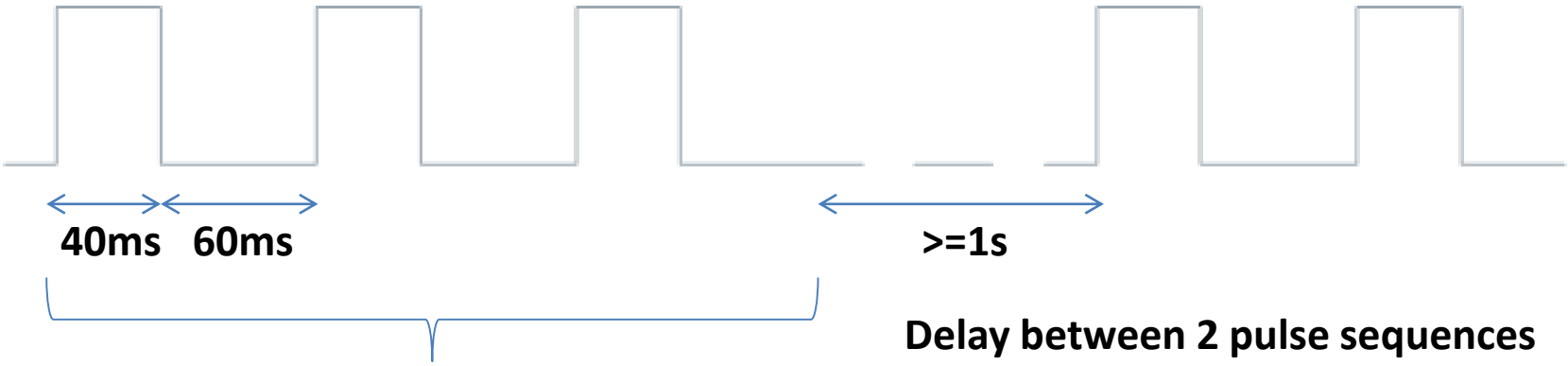
Excercise 5
Cyclic buffer (FIFO), keys driver

1. **Write a driver task for the keys, that stores the pressed buttons into a buffer**
2. **Write a program that produces a pulse train according to the contents of the keys buffer.**

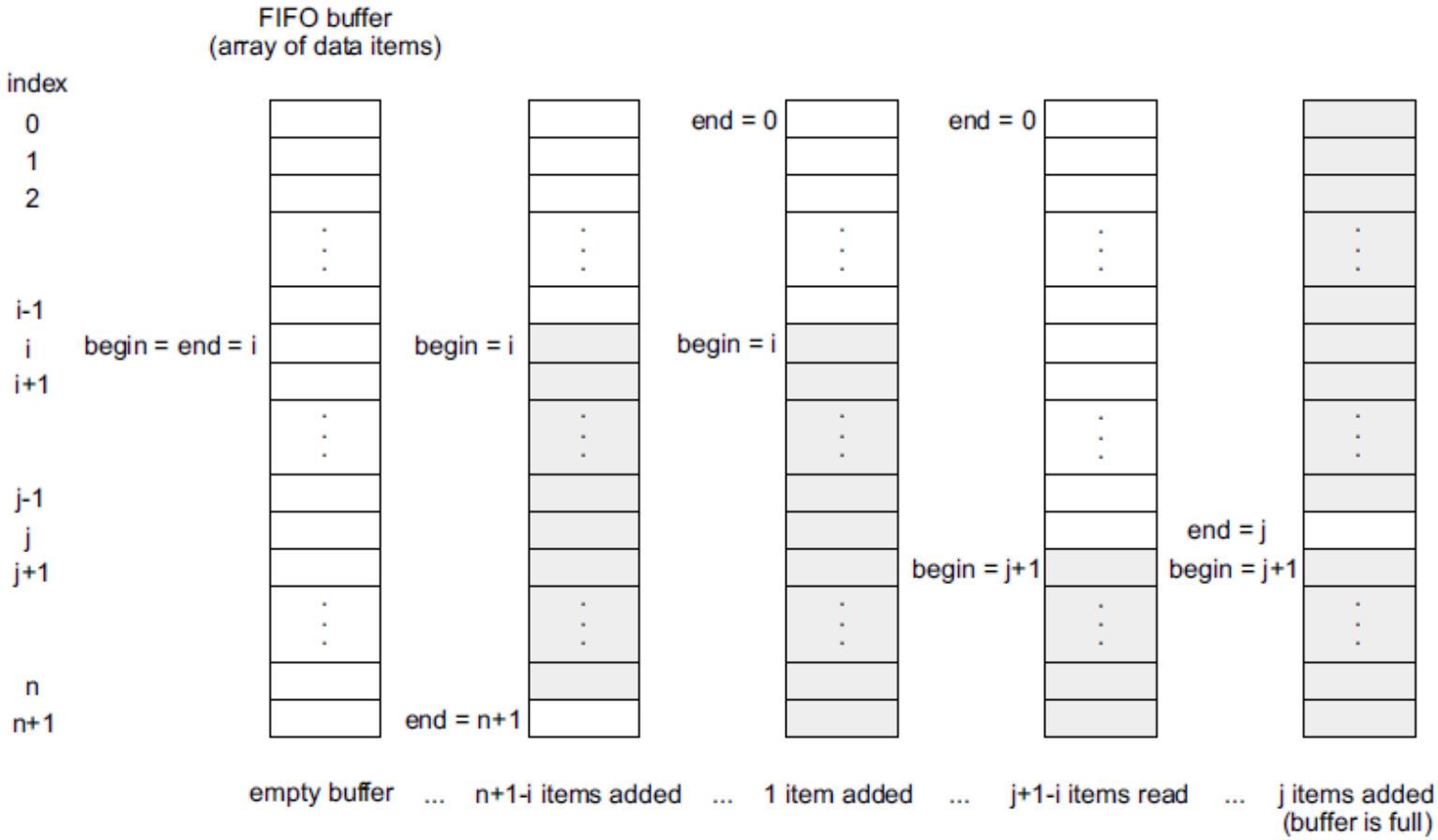
Use LED0 to generate a pulse train signal with number of pulses corresponding to the key pressed by the user. The pressed keys are stored in a buffer untill all previous key presses are processed:

T0 => store in a buffer as '0' => generate 10 pulses
T1 => store in a buffer as '1' => generate 1 pulse
T2 => store in a buffer as '2' => generate 2 pulses
T3 => store in a buffer as '3' => generate 3 pulses



When the oldest data in the buffer is '3'

Cyclic buffer – FIFO (First In First Out):



Cyclic buffer – FIFO (First In First Out):

2 new files in the project (rtos_pipe.h in rtos_pipe.c) for working with the buffers:

Buffer is a structure (described in rtos_pipe.h):

```
struct rtos_pipe{
        unsigned int begin;     // beginning of the buffer, read index (oldest byte index)
        unsigned int end;       // end of the buffer, write index (first free byte index)
        unsigned int size;      // buffer size, number of bytes
        char *buffer;           // pointer to the data buffer (char array)
};
```

Functions for working with buffers:

```
int rtos_pipe_read(struct rtos_pipe *pipe, char *data, unsigned int  num_bytes);
int rtos_pipe_write(struct rtos_pipe *pipe, char *data, unsigned int  num_bytes);
```

Arguments:

pipe: the address of the buffer struct
data: the address of the data we wish to write from (or store read values to)
num_bytes: the number of bytes to read (write)

Returns the number of succesfully read / written bytes

Cyclic buffer – FIFO (First In First Out):

Example:

Creating the FIFO:

```
#define BUFF_LEN 11          // define the number of bytes for the buffer
char test_buff[BUFF_LEN];    // reserve the space for the data
struct rtos_pipe testFifo = {0, 0, BUFF_LEN, test_buff};      // create and initialize the fifo struct
```

Writing to the FIFO:

```
char test[]="Hello";         // test data
int n;                       // will hold the return value
n = rtos_pipe_write( &testFifo, test, 3); // write the first 3 characters from test array into buffer (if possible)
If (n==3) ...                // check if 3 bytes were succesfully writen
```

Reading from the FIFO:

```
char out[5];                 // array, where the read values will be stored
n = rtos_pipe_read( &testFifo, r, 2); // trying to read 2 bytes from the buffer, and store them into out array
```

Pulse sequence generator:

RTOS tasks:

1. **keys_driver() : checks for fresh key presses and writes the corresponding ASCII char into the buffer**
2. **keys2lcd() : prepare the string for the LCD, which must display the characters still waiting in the buffer for processing**
3. **lcd_driver() : writes the contents of the lcd_string to the LCD**
4. **generator() : if idle, it reads 1 character from keys buffer and handles the LED state changes**

Additional info:

- **Set the time slice to 20ms => with 4 tasks the period of tasks is 80ms**
- **Generator task must be run in every time slice (privileged task) to ensure the correct LED timing**
- **Counting the executions of the generator task (privileged task is executed in every time slice) can be used to measure time. LED must be ON for 2 executions of the task (40ms), and OFF for 3 executions (60ms) of the task**
- **The pause between two sequences must be at least 1s => 50 time slices**