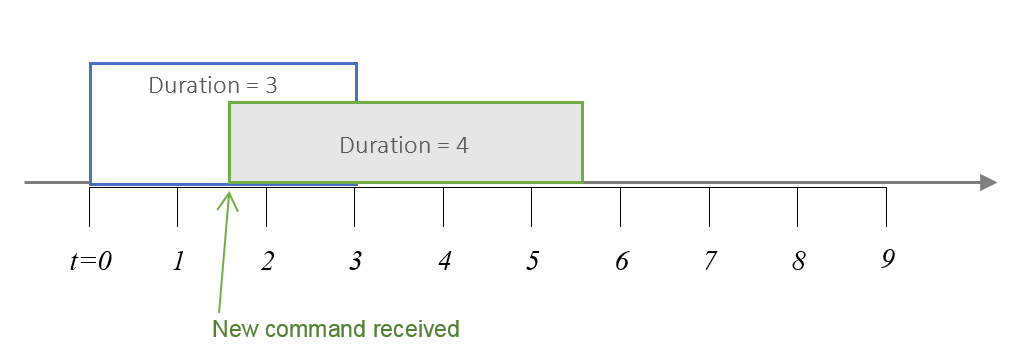
# Objective: Build a basic digital acquisition board accompanied by a PC application.

# Design

This project consists of several core components.

1. Protocol – command for the microcontroller. Response from the microcontroller. Document on how the command protocol works.
2. Encoder – C# code to produce command
3. Parser – C code to extract command content so that the microcontroller can act on the command.
4. Driver
   1. Periodic waveform - 50% duty cycle, user inputs pulse width.
   2. Timed output – user inputs duration (in number of seconds). Timing starts when user enables the output. When the output pin’s active duration equals or exceeds the duration, the output pin reverts to default state (off state).



* 1. ADC – return raw ADC value on input pin. Developer decides and documents which sampling method is being used. Sampling methods can be:
     1. On-demand – sample when command is received.
     2. Background – sample periodically in the background. Returns the most recent ADC value when command is received.

***Instructor’s comment:***

*“A good design should be extensible. A good implementation can also withstand slight changes in design. However, design and implementation are iterative processes. There is no such thing as perfect design. Try to bring your idea to a working state and then refactor your implementation up to a point you feel complete rewrite would be a better-off option.”*

# Implementation

1. Implementation should be done on PIC24FJ256GA702
2. Implementation should use APIs provided by FreeRTOS
   1. Use co-routine for heartbeat with interval of 1 second
   2. Use task for UART data transceiving.
   3. Use software timer, vTaskDelay(), for delay.
3. Document flowchart on encoder and parser.
4. Document components dependencies.
5. Document limitations on user input (such as limiting user input to integer only).

# Acceptance Criteria (for check-off)

### Test Procedures:

TBD

### Timeline:

|  |  |
| --- | --- |
| 2018/11/26, 9:00 pm | Design must be checked off by instructor.   * Protocol design |
| 2018/12/3, 9:00 pm | Implementation must be functional   * Encoder and parser * LED on/off control |
| 2018/12/10, 9:00 pm | * Demonstrate through the C# application, ready for check-off |

# General point deductions on Dropbox submission

[-1 point for each]

1. Missing header comment in main.c file. Header comment shall contain
   1. Summary of what the application does
   2. Author’s name (Your name)
2. Revision log, see example: <https://keepachangelog.com/en/1.0.0/>
   1. *For this simple project, it will likely have 2 – 3 lines of changelog.*
3. Missing development note to OneNote. (You can scan, type, or take picture of your design and implementation journal from your logbook)

[Subject to rejection]

1. Compilation error found on MPLAB XC16, in other words, unable to compile.
2. Excessive amount of dead codes

# Extra Credit

## **Requirement Option 1:** (20% of your score on the final project)

**Objective:**

Automate individual operations

**Description:**

Instead of interfacing GA702 with C# application, this design use KM202 to relay message.



In addition to simply relay the messages, this design shall fulfill the following requirements:

1. KM202 shall automatically execute a group of arbitrary commands sent from the C# application.

Think of this as timer-based macro operations.

Example macros:

|  |
| --- |
| Turn on LED |
| *Wait 2 seconds* |
| Turn on periodic wave (1 second pulse width) |
| *Wait 5 seconds* |
| Turn off periodic wave  Turn off LED |
| Read ADC channel 1 reading |
| *Wait 10 seconds* |
| Repeat this macro for 3 times |

1. KM202 shall support commands that can facilitate this kind of operations.
   1. Developer to propose commands and protocol format
   2. Developer to implement and demonstrate this capability
2. All commands in the examples are sent from the C# application.
   1. Supported operations are specified in the Design section 4a to 4c.

## **Requirement Option 2:** (15% of your score on the final project)

**Objective:**

Cipher / decipher command payload

**Description:**

Use KM202 to obfuscate both commands being sent from the C# application and results being returned from GA702.

Simple cipher and decipher algorithm is sufficient to demonstrate the concept.

Only the command portion needed to be obfuscated.



Example data packet format:

|  |  |  |
| --- | --- | --- |
| Packet header, information  (plain text) | Command payload  (scrambled text) | Other information  (plain text) |

**Requirements:**

1. Protocol shall support variable length command payload
   1. Developer shall explain how that protocol design can support variable length
   2. Developer shall explain how that protocol design can reliably transport the scrambled text.
2. Developer shall propose a protocol design that can accomplish the requirements and objective.
3. Developer shall implement and demonstrate the design.