```
 .include "m2560def.inc"

.def IN_COMMAND = r25

.org 0
rjmp initial_config

.org $0032                      //USART0 Rx Complete
jmp USART0_RXC

.org $0064
jmp TIM5_OVF                    //Timer5 Overflow Handler


.org INT_VECTORS_SIZE

.include "./Configs/motorsConfiguration.asm"
.include "./Configs/usartConfiguration.asm"
.include "./Utility/utility.asm"
.include "./communicationController.asm"
.include "./fireController.asm"
.include "./Utility/commandParser.asm"
.include "./manualMode.asm"
.include "./automaticMode.asm"

initial_config:

        // Configuro la comunicacion serial
        call usart_init

        // Configuro motores
        call motors_init
        call set_topspin

        ldi r16,(1<<SE)         // Habilito el modo sleep, tipo: IDLE
        out SMCR, r16

stand_by:
        /////// SETEO ESTADO STAND BY
        ////////////////////////
        call stop_mixer

        ldi PARAMETER, '0'
        call set_ball_speed

        ldi PARAMETER, '3'      // 90 grados
        call set_fire_angle

        ////////////
        ////////////

await_command:


        call usart_recieve_command
        call parse_execute_command

        jmp await_command
```

```
1  .dseg
2  CURRENT_SPIN_TYPE: .byte 1
3
4
5  .cseg
6
7  ldi Yl, low(CURRENT_SPIN_TYPE)
8  ldi yH, high(CURRENT_SPIN_TYPE)
9
10 .equ DEFAULT_FIRE_SPEED = 0
11
12 .equ DEFAULT_FIRE_ANGLE = 9 // Angulo 0
13 .equ FIRE_ANGLE_REG = OCR1AL
14
15 .equ DEFAULT_FIRE_STATE = 9 // Angulo 0
16 .equ FIRE_ENABLE_REG = OCR1BL
17
18 .equ FIRE_SPEED_REG1 = OCR3CL       // PIN 3
19 .equ FIRE_SPEED_REG2 = OCR3AL       // PIN 5
20
21 .equ DEFAULT_MIXING_SPEED = 100
22 .equ MIXING_SPEED_REG = OCR0A
23
24 motors_init:
25         // EN RESUMEN:
26         // MOTOR DISPARADOR 1: PIN 2, 1 o 0 setea direccion de rotacion
27         // MOTOR DISPARADOR 1: PIN 3, PWM
28
29         // MOTOR DISPARADOR 2: PIN 4, 1 o 0 setea direccion de rotacion
30         // MOTOR DISPARADOR 2: PIN 5, PWM
31
32         // MOTOR MEZCLADOR: PIN 13, PWM, el otro pin ponerlo a GND
33
34         // SERVO DIRECCION: PIN 11, PWM, el otro pin ponerlo a GND
35
36         // SERVO ALIMENTADOR: PIN 12 PWM, el otro pin ponerlo a GND
37
38         ldi r16, TOPSPIN
39         st Y, r16                 // Seteo el tipo de spin inicial
40
41 pin_config:
42         // Pongo los pines de los puertos necesarios como salida
43         // Uno de los pines va a estar en 0 o 1 dependiendo de la direccion y el otro va a tener el PWM
44         sbi DDRE, PE4             // PIN 2, si esta en 0 va en un sentido si esta en 1 va en el otro
45         sbi DDRE, PE5             // PIN 3 PWM motores disparadores
46
47         sbi DDRE, PG5             // PIN 4, si esta en 0 va en un sentido si esta en 1 va en el otro
48         sbi DDRE, PE3             // PIN 5 PWM motores disparadores
49
50         sbi DDRB, PB7             // PIN 13 PWM motor mezclador
51         // Aca no tengo un pin para la direccion ya que no me interesa, es solo un motor para revolver las ↙
       pelotitas
52
53
54         sbi DDRB, PB5             // PIN 11 PWM servo
55
56         sbi DDRB, PB6             // PIN 12 PWM servo habilitador
57
58 fire_motors_config:
59         lds r16, TCCR3B
60         ori r16, 0b00001001       // WGM32 = 1 CS30 = 1: No prescaling
61         sts TCCR3B, r16
62
63         // Configuro PWM para los motores del disparador
64         lds r16, TCCR3A
65         ori r16, 0b10001001       // COM3C1 = 1, COM3A1 = 1
66         sts TCCR3A, r16
67
68         ldi r16, DEFAULT_FIRE_SPEED
```

```
69          sts FIRE_SPEED_REG1, r16
70          sts FIRE_SPEED_REG2, r16
71
72  mixing_motor_config:
73
74
75          IN r16, TCCR0A
76          sbr r16, 0b10000011         // WGM00 = 1, WGM01 = 1, COM0A1 = 1 ----> PWM, TOP=MAX, Actualiza OCR en ↙
        TOP
77          OUT TCCR0A, r16
78
79          IN r16, TCCR0B
80          sbr r16, 0b00000001         // CS01 = 1: No prescaling
81          OUT TCCR0B, r16
82
83          ldi r16, DEFAULT_MIXING_SPEED   // Velocidad inicial
84          OUT MIXING_SPEED_REG, r16       // Comparador
85
86  fire_enable_motor_config:
87          // WGMn3:0 = 15 en fast pwm ----> OCRnA for defining TOP
88          // Configuro PWM para el servo
89          lds r16, TCCR1A
90          ori r16, 0b00100001         // WGM10 = 1, COM1B1 = 1 ----> PWM, TOP=MAX o MAX CUSTOM, Actualiza OCR ↙
        en TOP
91          sts TCCR1A, r16
92
93          //El servo requiere de una frecuencia especifica de 50 Hz para funcionar
94          lds r16, TCCR1B
95          ori r16, 0b00001101         // WGM32 = 1, WGM33 = 1, CS30 = 1: prescaler en 256 para lograr una      ↙
        frecuencia de 60 Hz
96          sts TCCR1B, r16
97
98          ldi r16, DEFAULT_FIRE_STATE     // Angulo inicial
99          sts FIRE_ENABLE_REG, r16
100
101 servo_motor_config:
102         // WGMn3:0 = 15 en fast pwm ----> OCRnA for defining TOP
103         // Configuro PWM para el servo
104         lds r16, TCCR1A
105         ori r16, 0b10000001         // WGM10 = 1, WGM11 = 1, COM1A1 = 1 COM1A0 = 0 ----> PWM, TOP=MAX o MAX ↙
        CUSTOM, Actualiza OCR en TOP
106         sts TCCR1A, r16
107
108
109         //El servo requiere de una frecuencia especifica de 50 Hz para funcionar
110         lds r16, TCCR1B
111         ori r16, 0b00001101         // WGM12 = 1, WGM13 = 1, CS12 = 0, CS11 = 1, CS10 = 1: prescaler en 256 ↙
        para lograr una frecuencia de 60 Hz
112         sts TCCR1B, r16
113
114         ldi r16, DEFAULT_FIRE_ANGLE     // Angulo inicial
115         sts FIRE_ANGLE_REG, r16
116
117     ret
118
119
120
121
```

```
1
2  .equ BAUD_RATE_L = 103
3  .equ BAUD_RATE_H = 0
4
5  usart_init:
6
7          cli                          // deshabilito interrupciones globales mientras inicializo
8
9          ldi r17, BAUD_RATE_H
10         ldi r16, BAUD_RATE_L         // pongo 103 en los registros r17:r16
11
12         sts UBRR0H, r17
13         sts UBRR0L, r16              // setea el baud rate en 9600
14
15         ldi r16, (1<<RXEN0)|(1<<RXCIE0)
16         sts UCSR0B,r16               // activa la recepcion y la interrupcion al terminar una recepcion
17
18         ldi r16, (3<<UCSZ00)         // modo asyncronico
19         sts UCSR0C,r16               // Seteo el format de frame: 8 bits de datos, 1 bit de final
20
21
22         sei                          // vuelvo a activar interrupciones globales
23
24
25     ret
26
27
```

```
1
2  parse_execute_command: // La diferencia con el anterior es que a esta rutina la llamo desde el modo manual
3
4          cpi IN_COMMAND, 's'                 // STAND BY
5          breq stand_by_jump
6
7          cpi IN_COMMAND, 'a'                 // MODO AUTOMATICO
8          breq receive_auto_mode
9
10         cpi IN_COMMAND, 'b'                 // BACKSPIN
11         breq set_backspin_call
12
13         cpi IN_COMMAND, 't'                 // TOPSPIN
14         breq set_topspin_call
15
16         cpi IN_COMMAND, 'v'                 // SET FIRE SPEED
17         breq receive_fire_speed
18
19         cpi IN_COMMAND, 'n'                 // SER FIRE ANGLE
20         breq receive_fire_angle
21
22         cpi IN_COMMAND, 'd'
23         breq shoot_jump
24
25         cpi IN_COMMAND, 'm'
26         breq start_mixer_call
27
28         jmp await_command
29
30 receive_fire_speed:
31         call usart_recieve_command
32         mov PARAMETER, IN_COMMAND
33         call set_ball_speed
34     ret
35
36 receive_fire_angle:
37         call usart_recieve_command
38         mov PARAMETER, IN_COMMAND
39         call set_fire_angle
40     ret
41
42 receive_auto_mode:
43         call usart_recieve_command
44         mov PARAMETER, IN_COMMAND
45
46         // Limpio la interrupcion
47         ldi r16, (1 << RXC0)
48         lds r17, UCSR0A
49
50         or r16, r17
51         sts UCSR0A, r16
52
53         mov PARAMETER, IN_COMMAND
54     jmp automatic_mode
55
56
57 set_topspin_call:
58         call set_topspin
59         ret
60
61 stand_by_jump:
62         jmp stand_by
63
64
65 set_backspin_call:
66         call set_backspin
67         ret
68
69 shoot_jump:
```

```
70          call shoot
71          ret
72
73  start_mixer_call:
74          call start_mixer
75          ret
```

```
1  .equ TOPSPIN = 0b00000000
2  .equ BACKSPIN = 0b00000001
3  .def PARAMETER = r16
4  .def PARAMETER2 = r17
5
6  sixteenth_delay:
7          ldi  r18, 5
8          ldi  r19, 75
9          ldi  r20, 191
10     L5: dec  r20
11         brne L5
12         dec  r19
13         brne L5
14         dec  r18
15         brne L5
16         nop
17     ret
18
19
20 delay_timer:                     // 1 segundo delay
21         ldi r16, low(49910)
22         ldi r17, high(49910)
23
24         sts TCNT5H, r17
25         sts TCNT5L, r16
26
27         ldi r16, (1 << TOIE5)
28         sts TIMSK5, r16
29
30         ldi r16, 0b00000101 //(1<<CS50) | (1<<CS52) // Prescaler 1024
31         sts TCCR5B, r16
32         sei
33         sleep
34         ret
35
36 half_delay_timer:                   // medio segundo delay
37         ldi r16, low(57723)
38         ldi r17, high(57723)
39
40         sts TCNT5H, r17
41         sts TCNT5L, r16
42
43         ldi r16, (1 << TOIE5)
44         sts TIMSK5, r16
45
46         ldi r16, 0b00000101 //(1<<CS50) | (1<<CS52) // Prescaler 1024
47         sts TCCR5B, r16
48         sei
49         sleep
50         ret
51
52 TIM5_OVF:
53         reti
```

```
usart_recieve_command:
        sleep
        //call parse_execute_command
    ret

has_pending_data:               // me pone el carry en 1 si hay datos para leer en el buffer

        lds r17, UCSR0A         // cargo en r17 el registro
        lsl r17                 // el bit 7 de UCSRnA me indica si hay datos para leer
    ret

usart_recieve_data: // Si llegue hasta aca es porque hay informacion para leer, leo y vuelvo
        lds IN_COMMAND, UDR0
    ret

USART0_RXC:

        call usart_recieve_data

    reti

```

```
shoot:
        // Habilito caida
        ldi r20, 13
        sts FIRE_ENABLE_REG, r20
        call sixteenth_delay
        // deshabilito caida
        ldi r20, 9
        sts FIRE_ENABLE_REG, r20
    ret


set_ball_speed:

        ld r18, Y
        cpi r18, BACKSPIN
        brne its_topspin                // Si es backspin tengo que complementar la velocidad

        call map_backspin_firespeed_ToPWM
        sts FIRE_SPEED_REG1, PARAMETER2
        sts FIRE_SPEED_REG2, PARAMETER
    ret

    its_topspin:
        call map_topspin_firespeed_ToPWM        // Mapeo el valor recibido por parametro 0-5 a un valor de PWM 0-255
        sts FIRE_SPEED_REG2, PARAMETER2
        sts FIRE_SPEED_REG1, PARAMETER

    ret

set_fire_angle:
        call map_FireAngle_ToPWM
        sts FIRE_ANGLE_REG, PARAMETER   // asumo que en r16 viene el angulo entre 1 y 5
    ret

start_mixer:
        ldi PARAMETER, DEFAULT_MIXING_SPEED
        OUT MIXING_SPEED_REG, PARAMETER // asumo que el parametro de la velocidad viene en r16
    ret

stop_mixer:
        ldi PARAMETER, 0
        OUT MIXING_SPEED_REG, PARAMETER // asumo que el parametro de la velocidad viene en r16
    ret

set_topspin:
        ld r17, Y                       // Tengo el tipo de spin actual en r17
        cpi r17, TOPSPIN
        breq return                     // si ya tenia topspin no hago nada
        call swich_speeds               // y si es distinto tengo que invertir la velocidad(complementar el PWM)

        ldi r17, TOPSPIN
        st Y, r17                       // guardo el tipo de spin

    ret


set_backspin:
        ld r17, Y                   // Tengo el tipo de spin actual en r17
        cpi r17, BACKSPIN
        breq return                 // si ya tenia backspin no hago nada
        call swich_speeds           // y si es distinto tengo que invertir la velocidad(complementar el PWM)

        ldi r17, BACKSPIN
        st Y, r17                   // guardo el tipo de spin

    ret

```

```
68 return:
69         ret
70
71 swich_speeds:
72
73         lds r1, FIRE_SPEED_REG1
74         lds r2, FIRE_SPEED_REG2
75
76         sts FIRE_SPEED_REG1, r2
77         sts FIRE_SPEED_REG2, r1
78
79     ret
80
81
82 map_topspin_firespeed_ToPWM:
83         cpi PARAMETER, '0'
84         breq set_topspin_fire_to_0
85         cpi PARAMETER, '1'
86         breq set_topspin_fire_to_1
87         cpi PARAMETER, '2'
88         breq set_topspin_fire_to_2
89         cpi PARAMETER, '3'
90         breq set_topspin_fire_to_3
91         cpi PARAMETER, '4'
92         breq set_topspin_fire_to_4
93         cpi PARAMETER, '5'
94         breq set_topspin_fire_to_5
95
96         jmp set_topspin_fire_to_0                    // Si el parametro es mayor a 5 o un caracter invalido
97
98     set_topspin_fire_to_0:
99         ldi PARAMETER, 3
100        ldi PARAMETER2, 3
101        ret
102    set_topspin_fire_to_1:
103        ldi PARAMETER, 105
104        ldi PARAMETER2, 75
105        ret
106    set_topspin_fire_to_2:
107        ldi PARAMETER, 115
108        ldi PARAMETER2, 75
109        ret
110    set_topspin_fire_to_3:
111        ldi PARAMETER, 130
112        ldi PARAMETER2, 75
113        ret
114    set_topspin_fire_to_4:
115        ldi PARAMETER, 140
116        ldi PARAMETER2, 75
117        ret
118    set_topspin_fire_to_5:
119        ldi PARAMETER, 155
120        ldi PARAMETER2, 75
121        ret
122
123 map_backspin_firespeed_ToPWM:
124        cpi PARAMETER, '0'
125        breq set_backspin_fire_to_0
126        cpi PARAMETER, '1'
127        breq set_backspin_fire_to_1
128        cpi PARAMETER, '2'
129        breq set_backspin_fire_to_2
130        cpi PARAMETER, '3'
131        breq set_backspin_fire_to_3
132        cpi PARAMETER, '4'
133        breq set_backspin_fire_to_4
134        cpi PARAMETER, '5'
135        breq set_backspin_fire_to_5
136
```

```
137         jmp set_backspin_fire_to_0                  // Si el parametro es mayor a 5 o un caracter invalido
138
139     set_backspin_fire_to_0:
140         ldi PARAMETER, 3
141         ldi PARAMETER2, 3
142         ret
143     set_backspin_fire_to_1:
144         ldi PARAMETER, 105
145         ldi PARAMETER2, 70
146         ret
147     set_backspin_fire_to_2:
148         ldi PARAMETER, 110
149         ldi PARAMETER2, 70
150         ret
151     set_backspin_fire_to_3:
152         ldi PARAMETER, 115
153         ldi PARAMETER2, 70
154         ret
155     set_backspin_fire_to_4:
156         ldi PARAMETER, 120
157         ldi PARAMETER2, 70
158         ret
159     set_backspin_fire_to_5:
160         ldi PARAMETER, 125
161         ldi PARAMETER2, 70
162         ret
163
164 map_FireAngle_ToPWM:
165
166         cpi PARAMETER, '1'
167         breq set_angle_to_1
168         cpi PARAMETER, '2'
169         breq set_angle_to_2
170         cpi PARAMETER, '3'
171         breq set_angle_to_3
172         cpi PARAMETER, '4'
173         breq set_angle_to_4
174         cpi PARAMETER, '5'
175         breq set_angle_to_5
176
177
178         jmp set_angle_to_1
179
180     set_angle_to_1:             // 0 GRADOS
181         ldi PARAMETER, 14       // Estos numeros son arbitrarios, salen de probar fisicamente el resultado
182         ret
183     set_angle_to_2:             // 45 GRADOS
184         ldi PARAMETER, 18
185         ret
186     set_angle_to_3:             // 90 GRADOS
187         ldi PARAMETER, 23
188         ret
189     set_angle_to_4:             // 135 GRADOS
190         ldi PARAMETER, 27
191         ret
192     set_angle_to_5:             // 180 GRADOS
193         ldi PARAMETER, 31
194         ret
195
```

```
1
2  automatic_mode:
3
4          cpi PARAMETER, '1'
5          breq auto_mode1
6
7          cpi PARAMETER, '2'
8          breq auto_mode2_jump
9
10         cpi PARAMETER, '3'
11         breq auto_mode3_jump
12
13         ldi IN_COMMAND, 0
14
15         jmp stand_by
16
17 auto_mode3_jump:
18         jmp auto_mode3
19
20 auto_mode2_jump:
21         jmp auto_mode2
22
23 auto_mode1: // Aca vario la velocidad efecto y ubicacion de la pelota para generar ejercicios ↙
       preestablecidos
24         call start_mixer
25         call set_topspin
26
27         // BOLA 1
28         ldi PARAMETER, '5'
29         call set_ball_speed
30         ldi PARAMETER, '5'
31         call set_fire_angle
32
33         //  DELAY ANTES DE EMPEZAR
34         call delay_timer
35         call delay_timer
36         call delay_timer
37         call delay_timer
38         ///
39
40         call shoot
41
42         call delay_timer
43         call half_delay_timer
44
45         // BOLA 2
46         ldi PARAMETER, '3'
47         call set_fire_angle
48
49         call delay_timer
50
51         call shoot
52
53         call delay_timer
54         call half_delay_timer
55
56         // BOLA 3
57         ldi PARAMETER, '5'
58         call set_fire_angle
59
60         call delay_timer
61
62         call shoot
63
64         call delay_timer
65         call half_delay_timer
66
67
68         // BOLA 4
```

```
69          ldi PARAMETER, '1'
70          call set_fire_angle
71
72          call delay_timer
73
74          call shoot
75
76          cpi IN_COMMAND, 's'
77          breq go_to_standby
78
79          jmp auto_mode1
80
81
82  auto_mode2:
83
84          call start_mixer
85
86          call set_topspin
87
88          // PARA QUE ARRANQUE MAS RAPIDO
89          ldi PARAMETER, '5'
90          call set_ball_speed
91          ///
92
93          call delay_timer
94          call delay_timer
95
96          // BOLA 1
97          ldi PARAMETER, '3'
98          call set_ball_speed
99          ldi PARAMETER, '5'
100         call set_fire_angle
101
102         //  DELAY ANTES DE EMPEZAR
103
104         call delay_timer
105         call delay_timer
106         call delay_timer
107         ///
108
109         call shoot
110
111         // BOLA 2
112         ldi PARAMETER, '5'
113         call set_ball_speed
114
115         call delay_timer
116         call shoot
117
118         // BOLA 3
119         ldi PARAMETER, '5'
120         call set_ball_speed
121
122         call delay_timer
123         call shoot
124
125         cpi IN_COMMAND, 's'
126         breq go_to_standby
127
128         jmp auto_mode2
129
130 go_to_standby:
131         jmp stand_by
132
133 auto_mode3:
134         call start_mixer
135
136         call set_topspin
137
```

```
138         // PARA QUE ARRANQUE MAS RAPIDO
139         ldi PARAMETER, '5'
140         call set_ball_speed
141         ///
142 
143         call delay_timer
144         call delay_timer
145 
146         // BOLA 1
147         ldi PARAMETER, '3'
148         call set_ball_speed
149         ldi PARAMETER, '1'
150         call set_fire_angle
151 
152         //  DELAY ANTES DE EMPEZAR
153 
154         call delay_timer
155         call delay_timer
156         call delay_timer
157         ///
158 
159         call shoot
160 
161         // BOLA 2
162         ldi PARAMETER, '5'
163         call set_ball_speed
164 
165         call delay_timer
166         call shoot
167 
168         // BOLA 3
169         ldi PARAMETER, '5'
170         call set_ball_speed
171 
172         call delay_timer
173         call shoot
174 
175         cpi IN_COMMAND, 's'
176         breq go_to_standby
177 
178         jmp auto_mode3
179 
180 
```