- can.h

```
/* can.h - Librería con las utilidades del CAN. */
#include <p30f4011.h>
#define MAX_MSG  8

// Transmite message
// DLC = msg's number of bytes
void CANSendBMsg(unsigned int id, unsigned int dlc, unsigned char *msg);
// DLC = msg's number of integer
void CANSendMsg(unsigned int id, unsigned int dlc, unsigned int *msg);
```

- can.c

```
/* can.c - Implementación de las funciones de can.h. */
#include "can.h"

void CANSendBMsg(unsigned int id, unsigned int dlc, unsigned char *msg) {
      // Standard Id
      C1TX0SIDbits.SID5_0 = id;
      id = id >> 6;
      C1TX0SIDbits.SID10_6 = id;

      // RTR
      C1TX0DLCbits.TXRTR = 0;           // Normal message

      // IDE
      C1TX0SIDbits.TXIDE = 0;           // Standard identifier

      // DLC
      C1TX0DLCbits.DLC = dlc;           // Data Length Code

      // Message
      unsigned int aux;
      switch (dlc) {
            case 1:
                  C1TX0B1 = msg[0];
                  break;
            case 2:
                  aux = msg[1];
                  aux = aux << 8;
                  C1TX0B1 = aux | msg[0];
                  break;
            case 3:
                  aux = msg[1];
                  aux = aux << 8;
                  C1TX0B1 = aux | msg[0];
                  C1TX0B2 = msg[2];
                  break;
            case 4:
                  aux = msg[1];
                  aux = aux << 8;
                  C1TX0B1 = aux | msg[0];
                  aux = msg[3];
                  aux = aux << 8;
                  C1TX0B2 = aux | msg[2];
                  break;
            case 5:
                  aux = msg[1];
                  aux = aux << 8;
                  C1TX0B1 = aux | msg[0];
                  aux = msg[3];
                  aux = aux << 8;
                  C1TX0B2 = aux | msg[2];
                  C1TX0B3 = msg[4];
                  break;
            case 6:
                  aux = msg[1];
                  aux = aux << 8;
                  C1TX0B1 = aux | msg[0];
                  aux = msg[3];
                  aux = aux << 8;
                  C1TX0B2 = aux | msg[2];
                  aux = msg[5];
                  aux = aux << 8;
                  C1TX0B3 = aux | msg[4];
                  break;
            case 7:
                  aux = msg[1];
```

```
                aux = aux << 8;
                C1TX0B1 = aux | msg[0];
                aux = msg[3];
                aux = aux << 8;
                C1TX0B2 = aux | msg[2];
                aux = msg[5];
                aux = aux << 8;
                C1TX0B3 = aux | msg[4];
                C1TX0B4 = msg[6];
                break;
          case 8:
                aux = msg[1];
                aux = aux << 8;
                C1TX0B1 = aux | msg[0];
                aux = msg[3];
                aux = aux << 8;
                C1TX0B2 = aux | msg[2];
                aux = msg[5];
                aux = aux << 8;
                C1TX0B3 = aux | msg[4];
                aux = msg[7];
                aux = aux << 8;
                C1TX0B4 = aux | msg[6];
                break;
     }

     C1TX0CONbits.TXREQ = 1;           // Send message
}

void CANSendMsg(unsigned int id, unsigned int dlc, unsigned int *msg) {
     // Standard Id
     C1TX0SIDbits.SID5_0 = id;
     id = id >> 6;
     C1TX0SIDbits.SID10_6 = id;

     // RTR
     C1TX0DLCbits.TXRTR = 0;           // Normal message

     // IDE
     C1TX0SIDbits.TXIDE = 0;           // Standard identifier

     // DLC
     C1TX0DLCbits.DLC = dlc*2;    // Data Length Code

     int aux;
     if (dlc >= 1) {
          aux = msg[0];
          C1TX0B1 = aux;
     }
     if (dlc >= 2) {
          aux = msg[1];
          C1TX0B2 = aux;
     }
     if (dlc >= 3) {
          aux = msg[2];
          C1TX0B3 = aux;
     }
     if (dlc >= 4) {
          aux = msg[3];
          C1TX0B4 = aux;
     }

     C1TX0CONbits.TXREQ = 1;                     // Send message
     while (C1TX0CONbits.TXREQ == 1);        // Wait until successfully transmitted
}
```

- esclavo1c.c

```
#include <p30f4011.h>
#include <uart.h>
#include "can.h"

/*****************************************************************/
/* Configuration words                                           */
/*****************************************************************/
_FOSC(CSW_FSCM_OFF & EC_PLL16);
_FWDT(WDT_OFF);
_FBORPOR(MCLR_EN & PBOR_OFF & PWRT_OFF);
_FGS(CODE_PROT_OFF);

/*****************************************************************/
/* Hardware                                                      */
/*****************************************************************/

#define FXT       7372800         // CPU clock
#define PLL       16              // PLL configuration
#define FCY       (FXT * PLL) / 4 // Clock that feeds the UART

#define BAUD_RATE 115200
#define BRG       (FCY / (16L * BAUD_RATE)) - 1L

/*****************************************************************/
/* Constants                                                     */
/*****************************************************************/
#define    WIDTH      80
#define    LENGTH           24

#define    BALL_L           1
#define    PADDLE_L         5
#define PADDLE_W      2

#define    PAD1_X           2
#define    PAD2_X           76

#define SCORE1_X      31
#define SCORE2_X      44
#define SCORE_Y       1

#define UP            'i'
#define DOWN                'k'
#define SERVICE       'j'

#define FILL                "#"
#define BALL                "O"

#define    M_BALL           00
#define    M_BOUNCE         02
#define    M_POINT          04
#define    S1_PADDLE  10
#define    S1_SERVICE 11
#define    S2_PADDLE  20
#define    S2_SERVICE 21

/*****************************************************************/
/* Global Variable declaration                                   */
/*****************************************************************/
// Ball coordenates (Range: 0-WIDTH, 0-LENGTH)
volatile unsigned int bx, by;
// Paddle 1 and 2 top left coordinates (Range: PAD1_X, 0-(LENGTH-PADDLE_L), PAD2_X, 0-
(LENGTH-PADDLE_L))
volatile unsigned int p1x, p1y, p2x, p2y;
// Scoreboard
volatile unsigned int score[2];
// Previous versions of ball and paddle variables
volatile unsigned int pre_bx, pre_by, pre_p1y, pre_p2y;
// Current screen cursor position (Range: 0-WIDTH, 0-LENGTH)
unsigned int cx, cy;

/*****************************************************************/
/* Interrupts                                                    */
/*****************************************************************/
void _ISR _U1RXInterrupt() {
     unsigned char c = ReadUART1();
```

```
      if (c == UP) if (p1y > 0) {pre_p1y = p1y; p1y -= 1; CANSendMsg(S1_PADDLE, 1, &p1y);}
      if (c == DOWN) if (p1y < LENGTH-PADDLE_L) {pre_p1y = p1y; p1y += 1; CANSendMsg
(S1_PADDLE, 1, &p1y);}
      if (c == SERVICE) CANSendMsg(S1_SERVICE, 0, 0);

      IFS0bits.U1RXIF = 0;
}

void _ISR _C1Interrupt() {
      if (C1INTFbits.RX0IF == 1) {
            int winner;
            unsigned int id = C1RX0SIDbits.SID;
            switch (id) {
                  case M_BALL:
                        pre_bx = bx;
                        bx = C1RX0B1;
                        pre_by = by;
                        by = C1RX0B2;
                        break;
                  case M_BOUNCE:
                        WriteUART1(7);              // Send the buzzer character back to
the UART
                        while (BusyUART1());   // Wait until the character is transmitted
                        break;
                  case M_POINT:
                        winner = C1RX0B1;
                        score[winner-1] = (score[winner-1] + 1) % 10;
                        break;
                  case S2_PADDLE:
                        pre_p2y = p2y;
                        p2y = C1RX0B1;
                        break;
            }

            C1RX0CONbits.RXFUL = 0;      // Clear reception full status flag
            C1INTFbits.RX0IF = 0;
      }
      IFS1bits.C1IF = 0;
}

/*****************************************************************/
/* Procedures                                                    */
/*****************************************************************/
void UARTConfig();
void CAN_config();
void slave1_init();
void clear_screen();
void draw_screen();

int main(void){
      UARTConfig();
      CAN_config();

      slave1_init();

      int j;
      for (j = 0; j < 1600; j++) Delay5ms();
      clear_screen();
      draw_screen();
      while (1) {
            if (pre_bx != bx || pre_by != by || pre_p1y != p1y || pre_p2y != p2y)
                  draw_screen();
      }

      return 0;
}

void UARTConfig(){
      U1MODE = 0;                     // Clear UART config - to avoid problems with
bootloader

      // Config UART
      OpenUART1(UART_EN &             // Enable UART
                  UART_DIS_LOOPBACK &   // Disable loopback mode
                  UART_NO_PAR_8BIT &         // 8bits / No parity
                  UART_1STOPBIT,             // 1 Stop bit
```

```
                  UART_TX_PIN_NORMAL &  // Tx break bit normal
                  UART_TX_ENABLE,       // Enable Transmition

                  BRG);                 // Baudrate
      U1STAbits.URXISEL = 0;

      // Enable UART Rx Interrupts
      IEC0bits.U1RXIE = 1;
      IFS0bits.U1RXIF = 0;
}

void CAN_config() {
      /* Initialize CAN */
      C1CTRLbits.REQOP = 0b100;              // Set configuration mode
      while(C1CTRLbits.OPMODE != 0b100);     // Wait until configuration mode

      C1CTRLbits.CANCKS = 1;                       // FCAN = FCY

      /* Baud rate */

      // BTR config 1
      C1CFG1bits.BRP = 0;                          // Prescaler to 2/Fcan
      C1CFG1bits.SJW = 0;                          // 1 TQ

      // BTR config 2
      C1CFG2bits.PRSEG  = 0b000;             // 1 TQs
      C1CFG2bits.SEG1PH = 0b011;             // 4 TQs
      C1CFG2bits.SEG2PH = 0b001;             // 2 TQs

      /* Interrupts */

      // General CAN interrupt
      IEC1bits.C1IE = 1;               // Enable general CAN interrupt
      IFS1bits.C1IF = 0;               // Clear general CAN interrupt flag

      // Local CAN interrupts
      C1INTEbits.RX0IE = 1;       // Enable CAN interrupt associated to rx buffer 0
      C1INTFbits.RX0IF = 0;       // Clear CAN interrupt flag associated to rx buffer 0

      /* Tx buffer 0 */

      // General transmission configuration
      C1TX0CONbits.TXREQ = 0;          // Clear transmission request flag

      /* Rx buffer 0 */

      // General reception configuration
      C1RX0CONbits.RXFUL = 0;          // Clear reception full status flag
      C1RX0CONbits.DBEN = 0;           // Disable double buffer

      // Configure acceptance mask
      C1RXM0SIDbits.SID = 0b00000000001; // Mask to check if sid is odd or even
      C1RXM0SIDbits.MIDE = 1;                      // Identifier mode as determined by
EXIDE
      C1RX0CONbits.FILHIT0 = 0;              // Link to acceptance filter 0

      // Configure acceptance filters
      C1RXF0SIDbits.EXIDE = 0;               // Enable filter for standard identifier
      C1RXF0SIDbits.SID = 0b000000000;  // Accept messages with even identifier

      C1CTRLbits.REQOP = 0b000;              // Set normal mode
      while(C1CTRLbits.OPMODE != 0b000);// Wait until normal mode
}

void slave1_init() {
      // Initial paddle coordinates
      p1x = PAD1_X;
      p1y = (LENGTH/2) - (PADDLE_L/2);
      p2x = PAD2_X;
      p2y = (LENGTH/2) - (PADDLE_L/2);

      // Initial ball coordinates
      bx = p1x + (PADDLE_W) + 1;
      by = p1y + (PADDLE_L/2);

      // Initial scores
      score[0] = 0;
```

```
	score[1] = 0;

	// Initial previous values at same value
	pre_bx = bx;
	pre_by = by;
	pre_p1y = p1y;
	pre_p2y = p2y;
}

void clear_screen() {
	WriteUART1(12);
	while (BusyUART1());		// Wait until the character is transmitted
	cx = 0; cy = 0;
}

void move_up() {
	WriteUART1(27);
	while (BusyUART1());
	WriteUART1(91);
	while (BusyUART1());
	WriteUART1(65);
	while (BusyUART1());
	cy--;
}

void move_down() {
	WriteUART1(27);
	while (BusyUART1());
	WriteUART1(91);
	while (BusyUART1());
	WriteUART1(66);
	while (BusyUART1());
	cy++;
}

void move_left() {
	WriteUART1(27);
	while (BusyUART1());
	WriteUART1(91);
	while (BusyUART1());
	WriteUART1(68);
	while (BusyUART1());
	cx--;
}

void move_right() {
	WriteUART1(27);
	while (BusyUART1());
	WriteUART1(91);
	while (BusyUART1());
	WriteUART1(67);
	while (BusyUART1());
	cx++;
}

void draw_fill() {
	putsUART1(FILL);
	while (BusyUART1());
	cx++;
}

void draw_ball() {
	putsUART1(BALL);
	while (BusyUART1());
	cx++;
}

void position_cursor(unsigned int x, unsigned int y) {
	while (cx < x) {
		move_right();
	}
	while (cx > x) {
		move_left();
	}
	while (cy < y)  {
		move_down();
	}
```

```
      while (cy > y)  {
            move_up();
      }
}

void reset_cursor() {
      while (cx > 0) {
            move_left();
      }
      while (cy > 0) {
            move_up();
      }
}

void draw_number(unsigned int number);

void draw_screen() {
      int i, j;
      // Clear screen and reset cursor
      clear_screen();

      // Draw player 1 paddle
      position_cursor(p1x, p1y);
      for (i = 0; i < PADDLE_L; i++) {
            for (j = 0; j < PADDLE_W; j++) {
                  draw_fill();
            }
            for (j = 0; j < PADDLE_W; j++) {
                  move_left();
            }
            move_down();
      }

      // Draw player 2 paddle
      position_cursor(p2x, p2y);
      for (i = 0; i < PADDLE_L; i++) {
            for (j = 0; j < PADDLE_W; j++) {
                  draw_fill();
            }
            for (j = 0; j < PADDLE_W; j++) {
                  move_left();
            }
            move_down();
      }

      // Draw player 1 scoreboard
      position_cursor(SCORE1_X, SCORE_Y);
      draw_number(score[0]);

      // Draw player 2 scoreboard
      position_cursor(SCORE2_X, SCORE_Y);
      draw_number(score[1]);

      // Draw ball
      position_cursor(bx, by);
      draw_ball();

      //Restore preview values
      pre_bx = bx;
      pre_by = by;
      pre_p1y = p1y;
      pre_p2y = p2y;
}

void draw_number(unsigned int number) {
      switch (number) {
            // ####
            // #  #
            // #  #
            // #  #
            // ####
            case 0:
                  draw_fill();
                  draw_fill();
                  draw_fill();
                  draw_fill();
                        move_down();
```

```
            move_left();
            move_left();
            move_left();
            move_left();

      draw_fill();
            move_right();
            move_right();
      draw_fill();
            move_down();
            move_left();
            move_left();
            move_left();
            move_left();

      draw_fill();
            move_right();
            move_right();
      draw_fill();
            move_down();
            move_left();
            move_left();
            move_left();
            move_left();

      draw_fill();
            move_right();
            move_right();
      draw_fill();
            move_down();
            move_left();
            move_left();
            move_left();
            move_left();

      draw_fill();
      draw_fill();
      draw_fill();
      draw_fill();
      break;
//  ##
//  ##
//  ##
//  ##
//  ##
case 1:
            move_right();
      draw_fill();
      draw_fill();
            move_down();
            move_left();
            move_left();

      draw_fill();
      draw_fill();
            move_down();
            move_left();
            move_left();

      draw_fill();
      draw_fill();
            move_down();
            move_left();
            move_left();

      draw_fill();
      draw_fill();
            move_down();
            move_left();
            move_left();

      draw_fill();
      draw_fill();
      break;
// ####
//    #
// ####
```

```
// #
// ####
case 2:
      draw_fill();
      draw_fill();
      draw_fill();
      draw_fill();
            move_down();
            move_left();

      draw_fill();
            move_down();
            move_left();
            move_left();
            move_left();
            move_left();

      draw_fill();
      draw_fill();
      draw_fill();
      draw_fill();
            move_down();
            move_left();
            move_left();
            move_left();
            move_left();

      draw_fill();
            move_down();
            move_left();

      draw_fill();
      draw_fill();
      draw_fill();
      draw_fill();
      break;
// ####
//    #
// ####
//    #
// ####
case 3:
      draw_fill();
      draw_fill();
      draw_fill();
      draw_fill();
            move_down();
            move_left();

      draw_fill();
            move_down();
            move_left();
            move_left();
            move_left();
            move_left();

      draw_fill();
      draw_fill();
      draw_fill();
      draw_fill();
            move_down();
            move_left();

      draw_fill();
            move_down();
            move_left();
            move_left();
            move_left();
            move_left();

      draw_fill();
      draw_fill();
      draw_fill();
      draw_fill();
      break;
// #  #
// #  #
```

```
// ####
//    #
//    #
case 4:
      draw_fill();
            move_right();
            move_right();
      draw_fill();
            move_down();
            move_left();
            move_left();
            move_left();
            move_left();

      draw_fill();
            move_right();
            move_right();
      draw_fill();
            move_down();
            move_left();
            move_left();
            move_left();
            move_left();

      draw_fill();
      draw_fill();
      draw_fill();
      draw_fill();
            move_down();
            move_left();

      draw_fill();
            move_down();
            move_left();

      draw_fill();
      break;
// ####
// #
// ####
//    #
// ####
case 5:
      draw_fill();
      draw_fill();
      draw_fill();
      draw_fill();
            move_down();
            move_left();
            move_left();
            move_left();
            move_left();

      draw_fill();
            move_down();
            move_left();

      draw_fill();
      draw_fill();
      draw_fill();
      draw_fill();
            move_down();
            move_left();

      draw_fill();
            move_down();
            move_left();
            move_left();
            move_left();
            move_left();

      draw_fill();
      draw_fill();
      draw_fill();
      draw_fill();
      break;
// #
```

```
// #
// ####
// #  #
// ####
case 6:
      draw_fill();
            move_down();
            move_left();

      draw_fill();
            move_down();
            move_left();

      draw_fill();
      draw_fill();
      draw_fill();
      draw_fill();
            move_down();
            move_left();
            move_left();
            move_left();
            move_left();

      draw_fill();
            move_right();
            move_right();
      draw_fill();
            move_down();
            move_left();
            move_left();
            move_left();
            move_left();

      draw_fill();
      draw_fill();
      draw_fill();
      draw_fill();
      break;
// ####
//    #
//    #
//    #
//    #
case 7:
      draw_fill();
      draw_fill();
      draw_fill();
      draw_fill();
            move_down();
            move_left();

      draw_fill();
            move_down();
            move_left();

      draw_fill();
            move_down();
            move_left();

      draw_fill();
            move_down();
            move_left();

      draw_fill();
      break;
// ####
// #  #
// ####
// #  #
// ####
case 8:
      draw_fill();
      draw_fill();
      draw_fill();
      draw_fill();
            move_down();
            move_left();
```

```
            move_left();
            move_left();
            move_left();

      draw_fill();
            move_right();
            move_right();
      draw_fill();
            move_down();
            move_left();
            move_left();
            move_left();
            move_left();

      draw_fill();
      draw_fill();
      draw_fill();
      draw_fill();
            move_down();
            move_left();
            move_left();
            move_left();
            move_left();

      draw_fill();
            move_right();
            move_right();
      draw_fill();
            move_down();
            move_left();
            move_left();
            move_left();
            move_left();

      draw_fill();
      draw_fill();
      draw_fill();
      draw_fill();
      break;
// ####
// #  #
// ####
//    #
//    #
case 9:
      draw_fill();
      draw_fill();
      draw_fill();
      draw_fill();
            move_down();
            move_left();
            move_left();
            move_left();
            move_left();

      draw_fill();
            move_right();
            move_right();
      draw_fill();
            move_down();
            move_left();
            move_left();
            move_left();
            move_left();

      draw_fill();
      draw_fill();
      draw_fill();
      draw_fill();
            move_down();
            move_left();

      draw_fill();
            move_down();
            move_left();

      draw_fill();
```

```
                    break;
            }
    }
```

- esclavo2c.c

```
#include <p30f4011.h>
#include <uart.h>
#include "can.h"

/*****************************************************************/
/* Configuration words                                           */
/*****************************************************************/
_FOSC(CSW_FSCM_OFF & EC_PLL16);
_FWDT(WDT_OFF);
_FBORPOR(MCLR_EN & PBOR_OFF & PWRT_OFF);
_FGS(CODE_PROT_OFF);

/*****************************************************************/
/* Hardware                                                      */
/*****************************************************************/

#define FXT       7372800         // CPU clock
#define PLL       16              // PLL configuration
#define FCY       (FXT * PLL) / 4 // Clock that feeds the UART

#define BAUD_RATE 115200
#define BRG       (FCY / (16L * BAUD_RATE)) - 1L

/*****************************************************************/
/* Constants                                                     */
/*****************************************************************/
#define    WIDTH      80
#define    LENGTH           24

#define    BALL_L           1
#define    PADDLE_L         5
#define PADDLE_W      2

#define    PAD1_X           2
#define    PAD2_X           76

#define SCORE1_X      31
#define SCORE2_X      44
#define SCORE_Y       1

#define UP            'i'
#define DOWN                'k'
#define SERVICE       'j'

#define FILL                "#"
#define BALL                "O"

#define    M_BALL           00
#define    M_BOUNCE         02
#define    M_POINT          04
#define    S1_PADDLE  10
#define    S1_SERVICE 11
#define    S2_PADDLE  20
#define    S2_SERVICE 21

/*****************************************************************/
/* Global Variable declaration                                   */
/*****************************************************************/
// Ball coordenates (Range: 0-WIDTH, 0-LENGTH)
volatile unsigned int bx, by;
// Paddle 1 and 2 top left coordinates (Range: PAD1_X, 0-(LENGTH-PADDLE_L), PAD2_X, 0-
(LENGTH-PADDLE_L))
volatile unsigned int p1x, p1y, p2x, p2y;
// Scoreboard
volatile unsigned int score[2];
// Previous versions of ball and paddle variables
volatile unsigned int pre_bx, pre_by, pre_p1y, pre_p2y;
// Current screen cursor position (Range: 0-WIDTH, 0-LENGTH)
unsigned int cx, cy;

/*****************************************************************/
/* Interrupts                                                    */
/*****************************************************************/
void _ISR _U1RXInterrupt() {
     unsigned char c = ReadUART1();
```

```
      if (c == UP) if (p2y > 0) {pre_p2y = p2y; p2y -= 1; CANSendMsg(S2_PADDLE, 1, &p2y);}
      if (c == DOWN) if (p2y < LENGTH-PADDLE_L) {pre_p2y = p2y; p2y += 1; CANSendMsg
(S2_PADDLE, 1, &p2y);}
      if (c == SERVICE) CANSendMsg(S2_SERVICE, 0, 0);

      IFS0bits.U1RXIF = 0;
}

void _ISR _C1Interrupt() {
      if (C1INTFbits.RX0IF == 1) {
            int winner;
            unsigned int id = C1RX0SIDbits.SID;
            switch (id) {
                  case M_BALL:
                        pre_bx = bx;
                        bx = C1RX0B1;
                        pre_by = by;
                        by = C1RX0B2;
                        break;
                  case M_BOUNCE:
                        WriteUART1(7);              // Send the buzzer character back to
the UART
                        while (BusyUART1());        // Wait until the character is
transmitted
                        break;
                  case M_POINT:
                        winner = C1RX0B1;
                        score[winner-1] = (score[winner-1] + 1) % 10;
                        break;
                  case S1_PADDLE:
                        pre_p1y = p1y;
                        p1y = C1RX0B1;
                        break;
            }

            C1RX0CONbits.RXFUL = 0;      // Clear reception full status flag
            C1INTFbits.RX0IF = 0;
      }
      IFS1bits.C1IF = 0;
}

/*************************************************************/
/* Procedures                                                */
/*************************************************************/
void UARTConfig();
void CAN_config();
void slave2_init();
void clear_screen();
void draw_screen();

int main(void){
      UARTConfig();
      CAN_config();

      slave2_init();

      int j;
      for (j = 0; j < 800; j++) Delay5ms();
      clear_screen();
      draw_screen();
      while (1) {
            if (pre_bx != bx || pre_by != by || pre_p1y != p1y || pre_p2y != p2y)
                  draw_screen();
      }

      return 0;
}

void UARTConfig(){
      U1MODE = 0;                     // Clear UART config - to avoid problems with
bootloader

      // Config UART
      OpenUART1(UART_EN &             // Enable UART
                  UART_DIS_LOOPBACK &   // Disable loopback mode
                  UART_NO_PAR_8BIT &        // 8bits / No parity
                  UART_1STOPBIT,            // 1 Stop bit
```

```
                  UART_TX_PIN_NORMAL &  // Tx break bit normal
                  UART_TX_ENABLE,       // Enable Transmition

                  BRG);                 // Baudrate
      U1STAbits.URXISEL = 0;

      // Enable UART Rx Interrupts
      IEC0bits.U1RXIE = 1;
      IFS0bits.U1RXIF = 0;
}

void CAN_config() {
      /* Initialize CAN */
      C1CTRLbits.REQOP = 0b100;              // Set configuration mode
      while(C1CTRLbits.OPMODE != 0b100);     // Wait until configuration mode

      C1CTRLbits.CANCKS = 1;                       // FCAN = FCY

      /* Baud rate */

      // BTR config 1
      C1CFG1bits.BRP = 0;                          // Prescaler to 2/Fcan
      C1CFG1bits.SJW = 0;                          // 1 TQ

      // BTR config 2
      C1CFG2bits.PRSEG  = 0b000;             // 1 TQs
      C1CFG2bits.SEG1PH = 0b011;             // 4 TQs
      C1CFG2bits.SEG2PH = 0b001;             // 2 TQs

      /* Interrupts */

      // General CAN interrupt
      IEC1bits.C1IE = 1;               // Enable general CAN interrupt
      IFS1bits.C1IF = 0;               // Clear general CAN interrupt flag

      // Local CAN interrupts
      C1INTEbits.RX0IE = 1;       // Enable CAN interrupt associated to rx buffer 0
      C1INTFbits.RX0IF = 0;       // Clear CAN interrupt flag associated to rx buffer 0

      /* Tx buffer 0 */

      // General transmission configuration
      C1TX0CONbits.TXREQ = 0;     // Clear transmission request flag

      /* Rx buffer 0 */

      // General reception configuration
      C1RX0CONbits.RXFUL = 0;          // Clear reception full status flag
      C1RX0CONbits.DBEN = 0;           // Disable double buffer

      // Configure acceptance mask
      C1RXM0SIDbits.SID = 0b0000000001; // Mask to check if sid is odd or even
      C1RXM0SIDbits.MIDE = 1;                      // Identifier mode as determined by
EXIDE
      C1RX0CONbits.FILHIT0 = 0;              // Link to acceptance filter 0

      // Configure acceptance filters
      C1RXF0SIDbits.EXIDE = 0;               // Enable filter for standard identifier
      C1RXF0SIDbits.SID = 0b000000000;  // Accept messages with even identifier

      C1CTRLbits.REQOP = 0b000;              // Set normal mode
      while(C1CTRLbits.OPMODE != 0b000);// Wait until normal mode
}

void slave2_init() {
      // Initial paddle coordinates
      p1x = PAD1_X;
      p1y = (LENGTH/2) - (PADDLE_L/2);
      p2x = PAD2_X;
      p2y = (LENGTH/2) - (PADDLE_L/2);

      // Initial ball coordinates
      bx = p1x + (PADDLE_W) + 1;
      by = p1y + (PADDLE_L/2);

      // Initial scores
```

```
	score[0] = 0;
	score[1] = 0;

	// Initial previous values at same value
	pre_bx = bx;
	pre_by = by;
	pre_p1y = p1y;
	pre_p2y = p2y;
}

void clear_screen() {
	WriteUART1(12);
	while (BusyUART1());		// Wait until the character is transmitted
	cx = 0; cy = 0;
}

void move_up() {
	WriteUART1(27);
	while (BusyUART1());
	WriteUART1(91);
	while (BusyUART1());
	WriteUART1(65);
	while (BusyUART1());
	cy--;
}

void move_down() {
	WriteUART1(27);
	while (BusyUART1());
	WriteUART1(91);
	while (BusyUART1());
	WriteUART1(66);
	while (BusyUART1());
	cy++;
}

void move_left() {
	WriteUART1(27);
	while (BusyUART1());
	WriteUART1(91);
	while (BusyUART1());
	WriteUART1(68);
	while (BusyUART1());
	cx--;
}

void move_right() {
	WriteUART1(27);
	while (BusyUART1());
	WriteUART1(91);
	while (BusyUART1());
	WriteUART1(67);
	while (BusyUART1());
	cx++;
}

void draw_fill() {
	putsUART1(FILL);
	while (BusyUART1());
	cx++;
}

void draw_ball() {
	putsUART1(BALL);
	while (BusyUART1());
	cx++;
}

void position_cursor(unsigned int x, unsigned int y) {
	while (cx < x) {
		move_right();
	}
	while (cx > x) {
		move_left();
	}
	while (cy < y)  {
		move_down();
```

```
	}
	while (cy > y)  {
		move_up();
	}
}

void reset_cursor() {
	while (cx > 0) {
		move_left();
	}
	while (cy > 0) {
		move_up();
	}
}

void draw_number(unsigned int number);

void draw_screen() {
	int i, j;
	// Clear screen and reset cursor
	clear_screen();

	// Draw player 1 paddle
	position_cursor(p1x, p1y);
	for (i = 0; i < PADDLE_L; i++) {
		for (j = 0; j < PADDLE_W; j++) {
			draw_fill();
		}
		for (j = 0; j < PADDLE_W; j++) {
			move_left();
		}
		move_down();
	}
	//reset_cursor();

	// Draw player 2 paddle
	position_cursor(p2x, p2y);
	for (i = 0; i < PADDLE_L; i++) {
		for (j = 0; j < PADDLE_W; j++) {
			draw_fill();
		}
		for (j = 0; j < PADDLE_W; j++) {
			move_left();
		}
		move_down();
	}
	//reset_cursor();

	// Draw player 1 scoreboard
	position_cursor(SCORE1_X, SCORE_Y);
	draw_number(score[0]);
	reset_cursor();

	// Draw player 2 scoreboard
	position_cursor(SCORE2_X, SCORE_Y);
	draw_number(score[1]);
	//reset_cursor();

	// Draw ball
	position_cursor(bx, by);
	draw_ball();
	//reset_cursor();

	//Restore preview values
	pre_bx = bx;
	pre_by = by;
	pre_p1y = p1y;
	pre_p2y = p2y;
}

void draw_number(unsigned int number) {
	switch (number) {
		// ####
		// #  #
		// #  #
		// #  #
		// ####
```

```
case 0:
      draw_fill();
      draw_fill();
      draw_fill();
      draw_fill();
            move_down();
            move_left();
            move_left();
            move_left();
            move_left();

      draw_fill();
            move_right();
            move_right();
      draw_fill();
            move_down();
            move_left();
            move_left();
            move_left();
            move_left();

      draw_fill();
            move_right();
            move_right();
      draw_fill();
            move_down();
            move_left();
            move_left();
            move_left();
            move_left();

      draw_fill();
            move_right();
            move_right();
      draw_fill();
            move_down();
            move_left();
            move_left();
            move_left();
            move_left();

      draw_fill();
      draw_fill();
      draw_fill();
      draw_fill();
      break;
//  ##
//  ##
//  ##
//  ##
//  ##
case 1:
            move_right();
      draw_fill();
      draw_fill();
            move_down();
            move_left();
            move_left();

      draw_fill();
      draw_fill();
            move_down();
            move_left();
            move_left();

      draw_fill();
      draw_fill();
            move_down();
            move_left();
            move_left();

      draw_fill();
      draw_fill();
            move_down();
            move_left();
            move_left();
```

```
      draw_fill();
      draw_fill();
      break;
// ####
//    #
// ####
// #
// ####
case 2:
      draw_fill();
      draw_fill();
      draw_fill();
      draw_fill();
            move_down();
            move_left();

      draw_fill();
            move_down();
            move_left();
            move_left();
            move_left();
            move_left();

      draw_fill();
      draw_fill();
      draw_fill();
      draw_fill();
            move_down();
            move_left();
            move_left();
            move_left();
            move_left();

      draw_fill();
            move_down();
            move_left();

      draw_fill();
      draw_fill();
      draw_fill();
      draw_fill();
      break;
// ####
//    #
// ####
//    #
// ####
case 3:
      draw_fill();
      draw_fill();
      draw_fill();
      draw_fill();
            move_down();
            move_left();

      draw_fill();
            move_down();
            move_left();
            move_left();
            move_left();
            move_left();

      draw_fill();
      draw_fill();
      draw_fill();
      draw_fill();
            move_down();
            move_left();

      draw_fill();
            move_down();
            move_left();
            move_left();
            move_left();
            move_left();

      draw_fill();
```

```
      draw_fill();
      draw_fill();
      draw_fill();
      break;
// #  #
// #  #
// ####
//    #
//    #
case 4:
      draw_fill();
            move_right();
            move_right();
      draw_fill();
            move_down();
            move_left();
            move_left();
            move_left();
            move_left();

      draw_fill();
            move_right();
            move_right();
      draw_fill();
            move_down();
            move_left();
            move_left();
            move_left();
            move_left();

      draw_fill();
      draw_fill();
      draw_fill();
      draw_fill();
            move_down();
            move_left();

      draw_fill();
            move_down();
            move_left();

      draw_fill();
      break;
// ####
// #
// ####
//    #
// ####
case 5:
      draw_fill();
      draw_fill();
      draw_fill();
      draw_fill();
            move_down();
            move_left();
            move_left();
            move_left();
            move_left();

      draw_fill();
            move_down();
            move_left();

      draw_fill();
      draw_fill();
      draw_fill();
      draw_fill();
            move_down();
            move_left();

      draw_fill();
            move_down();
            move_left();
            move_left();
            move_left();
            move_left();
```

```
      draw_fill();
      draw_fill();
      draw_fill();
      draw_fill();
      break;
// #
// #
// ####
// #  #
// ####
case 6:
      draw_fill();
            move_down();
            move_left();

      draw_fill();
            move_down();
            move_left();

      draw_fill();
      draw_fill();
      draw_fill();
      draw_fill();
            move_down();
            move_left();
            move_left();
            move_left();
            move_left();

      draw_fill();
            move_right();
            move_right();
      draw_fill();
            move_down();
            move_left();
            move_left();
            move_left();
            move_left();

      draw_fill();
      draw_fill();
      draw_fill();
      draw_fill();
      break;
// ####
//    #
//    #
//    #
//    #
case 7:
      draw_fill();
      draw_fill();
      draw_fill();
      draw_fill();
            move_down();
            move_left();

      draw_fill();
            move_down();
            move_left();

      draw_fill();
            move_down();
            move_left();

      draw_fill();
            move_down();
            move_left();

      draw_fill();
            move_down();
            move_left();

      draw_fill();
      break;
// ####
// #  #
// ####
// #  #
// ####
case 8:
```

```
      draw_fill();
      draw_fill();
      draw_fill();
      draw_fill();
            move_down();
            move_left();
            move_left();
            move_left();
            move_left();

      draw_fill();
            move_right();
            move_right();
      draw_fill();
            move_down();
            move_left();
            move_left();
            move_left();
            move_left();

      draw_fill();
      draw_fill();
      draw_fill();
      draw_fill();
            move_down();
            move_left();
            move_left();
            move_left();
            move_left();

      draw_fill();
            move_right();
            move_right();
      draw_fill();
            move_down();
            move_left();
            move_left();
            move_left();
            move_left();

      draw_fill();
      draw_fill();
      draw_fill();
      draw_fill();
      break;
// ####
// #  #
// ####
//    #
//    #
case 9:
      draw_fill();
      draw_fill();
      draw_fill();
      draw_fill();
            move_down();
            move_left();
            move_left();
            move_left();
            move_left();

      draw_fill();
            move_right();
            move_right();
      draw_fill();
            move_down();
            move_left();
            move_left();
            move_left();
            move_left();

      draw_fill();
      draw_fill();
      draw_fill();
      draw_fill();
            move_down();
            move_left();
```

```
                draw_fill();
                      move_down();
                      move_left();

                draw_fill();
                break;
        }
}
```

- maestro.c

```
#include <p30f4011.h>
#include <time.h>
#include <stdlib.h>
#include "can.h"

/*****************************************************************/
/* Configuration words                                           */
/*****************************************************************/
_FOSC(CSW_FSCM_OFF & EC_PLL16);
_FWDT(WDT_OFF);
_FBORPOR(MCLR_EN & PBOR_OFF & PWRT_OFF);
_FGS(CODE_PROT_OFF);

/*****************************************************************/
/* Constants                                                     */
/*****************************************************************/
#define    WIDTH      80
#define    LENGTH           24

#define    BALL_L           1
#define    PADDLE_L         5
#define PADDLE_W       2

#define    PAD1_X           2
#define    PAD2_X           76

#define SERV_NO        0
#define SERV_YES       1

#define    M_BALL           00
#define    M_BOUNCE         02
#define    M_POINT          04
#define    S1_PADDLE   10
#define    S1_SERVICE  11
#define    S2_PADDLE   20
#define    S2_SERVICE  21

/*****************************************************************/
/* Global Variable declaration                                   */
/*****************************************************************/
// Ball coordinates (Range: 0-WIDTH, 0-LENGTH)
unsigned int bx, by;
// Paddle 1 and 2 top left coordinates (Range: PAD1_X, 0-(LENGTH-PADDLE_L), PAD2_X, 0-
(LENGTH-PADDLE_L))
volatile unsigned int p1x, p1y, p2x, p2y;
// Ball movement vector (Values: -1.1, -1.1)
volatile int vector_x, vector_y;
// Ball speed (Range: 0-4)
volatile unsigned int speed;
// Service no/yes (Values: 0.1)
volatile unsigned char service;
// Possession service (Values: 1.2)
unsigned char pos_service;

/*****************************************************************/
/* Interrupts                                                    */
/*****************************************************************/
void _ISR _ADCInterrupt(void) {
      int ADCValue = ADCBUF0;           // get ADC value
      speed = ((float)ADCValue/1023.0)*4;
      IFS0bits.ADIF = 0;                // restore ADIF
}

void _ISR _C1Interrupt() {
      if (C1INTFbits.RX0IF == 1) {
            unsigned int id = C1RX0SIDbits.SID;
            switch (id) {
                  case S1_PADDLE:
                        p1y = C1RX0B1;
                        break;
                  case S1_SERVICE:
                        if (pos_service == 1) {
                              service = SERV_NO;
                              vector_x = 1;
                              vector_y = ((rand() % 2) == 0) ? -1 : 1;
                        }
```

```
                        break;
                  case S2_PADDLE:
                        p2y = C1RX0B1;
                        break;
                  case S2_SERVICE:
                        if (pos_service == 2) {
                              service = SERV_NO;
                              vector_x = -1;
                              vector_y = ((rand() % 2) == 0) ? -1 : 1;
                        }
                        break;
            }

            C1RX0CONbits.RXFUL = 0;      // Clear reception full status flag
            C1INTFbits.RX0IF = 0;
      }
      IFS1bits.C1IF = 0;
}

/*****************************************************************/
/* Procedures                                                    */
/*****************************************************************/
void CAN_config();
void ADC_config();
void master_init();
unsigned char check_paddle_hit();
unsigned char check_wall_hit();
int check_point_made();

int main(void)
{
      CAN_config();
      ADC_config();

      master_init();

      // mode: 0->nothing, 1->bounce, 2->point
      int mode, winner, i;
      unsigned int ball_coordinates[2];
      while (1) {
            mode = 0;
            winner = 0;

            // Update ball coordinates
            if (service) {
                  if (pos_service == 1) {
                        bx = p1x + (PADDLE_W) + 1;
                        by = p1y + (PADDLE_L/2);
                  } else {
                        bx = p2x - 2;
                        by = p2y + (PADDLE_L/2);
                  }
            } else {
                  bx += vector_x;
                  by += vector_y;
            }

            // Check bounces
            unsigned char paddle_bounce = check_paddle_hit();
            if (paddle_bounce) {
                  mode = 1;
                  vector_x = (vector_x == 1) ? -1 : 1;
                  if (bx < (WIDTH/2)) {  // If it bounced with paddle 1
                        if (vector_y == 1 && by < (p1y + (PADDLE_L/2))) vector_y = -1;
                        else if (vector_y == -1 && by > (p1y + (PADDLE_L/2))) vector_y = 1;
                  } else {                     // If it bounced with paddle 2
                        if (vector_y == 1 && by < (p2y + (PADDLE_L/2))) vector_y = -1;
                        else if (vector_y == -1 && by > (p2y + (PADDLE_L/2))) vector_y = 1;
                  }
            }
            unsigned char wall_bounce = check_wall_hit();
            if (wall_bounce) {
                  mode = 1;
                  vector_y = (vector_y == -1) ? 1 : -1;
            }

            // Check if someone has scored
```

```
            winner = check_point_made();
            if (winner) {
                  mode = 2;
                  if (winner == 1) {
                        bx = p2x - 1;
                        by = p2y + (PADDLE_L/2) - BALL_L;
                        pos_service = 2;
                  } else {
                        bx = p1x + (PADDLE_W) + 1;
                        by = p1y + (PADDLE_L/2) - BALL_L;
                        pos_service = 1;
                  }
                  service = SERV_YES;
            }

            // Send messages
            if (mode == 1) CANSendMsg(M_BOUNCE, 0, NULL);
            else if (mode == 2) CANSendMsg(M_POINT, 1, &winner);
            ball_coordinates[0] = bx;
            ball_coordinates[1] = by;
            CANSendMsg(M_BALL, 2, ball_coordinates);

            // Wait until next update
            for (i = 0; i < 100-20*speed; i++) Delay5ms();
      }

    return 0;
}

void CAN_config() {
      /* Initialize CAN */
      C1CTRLbits.REQOP = 0b100;              // Set configuration mode
      while(C1CTRLbits.OPMODE != 0b100);     // Wait until configuration mode

      C1CTRLbits.CANCKS = 1;                       // FCAN = FCY

      /* Baud rate */

      // BTR config 1
      C1CFG1bits.BRP = 0;                          // Prescaler to 2/Fcan
      C1CFG1bits.SJW = 0;                          // 1 TQ

      // BTR config 2
      C1CFG2bits.PRSEG  = 0b000;             // 1 TQs
      C1CFG2bits.SEG1PH = 0b011;             // 4 TQs
      C1CFG2bits.SEG2PH = 0b001;             // 2 TQs

      /* Interrupts */

      // General CAN interrupt
      IEC1bits.C1IE = 1;               // Enable general CAN interrupt
      IFS1bits.C1IF = 0;               // Clear general CAN interrupt flag

      // Local CAN interrupts
      C1INTEbits.RX0IE = 1;       // Enable CAN interrupt associated to rx buffer 0
      C1INTFbits.RX0IF = 0;       // Clear CAN interrupt flag associated to rx buffer 0

      /* Tx buffer 0 */

      // General transmission configuration
      C1TX0CONbits.TXREQ = 0;     // Clear transmission request flag

      /* Rx buffer 0 */

      // General reception configuration
      C1RX0CONbits.RXFUL = 0;     // Clear reception full status flag
      C1RX0CONbits.DBEN = 0;      // Disable double buffer

      // Configure acceptance mask
      C1RXM0SIDbits.SID = 0;           // No bits to compare
      C1RXM0SIDbits.MIDE = 1;          // Identifier mode as determined by EXIDE
      C1RX0CONbits.FILHIT0 = 0;   // Link to acceptance filter 0

      // Configure acceptance filters
      C1RXF0SIDbits.EXIDE = 0;    // Enable filter for standard identifier
      C1RXF0SIDbits.SID = 0;           // Doesn't matter the value as mask is '0'
```

```
      C1CTRLbits.REQOP = 0b000;               // Set normal mode
      while(C1CTRLbits.OPMODE != 0b000);// Wait until normal mode
}

void ADC_config() {
      TRISBbits.TRISB7 = 1;  // Declare AN7 as input
      ADPCFG = 0xFF7F;            // all PORTB = Digital; RB7 = analog
      ADCON1 = 0x00E4;            // SAMP bit is auto-set
                                        // sampling begins immediately after last
conversion
      ADCHS = 0x0007;             // Connect RB7/AN7 as CH0 input ..
      ADCSSL = 0;
      ADCON3 = 0x0000;

      // Set ADC clock 32 times slower than System clock
      ADCON3bits.ADCS = 31;

      // Set time bewteen conversions to 31 ADC clock cycles
      ADCON3bits.SAMC = 31;
      ADCON2 = 0;

      // Set at 16 the number of conversions before throwing an interruption
      ADCON2bits.SMPI = 0b1111;
      ADCON1bits.ADON = 1;   // turn ADC ON
      ADCON1bits.SAMP = 1;        // start sampling

      //Configure interrupts
      IEC0bits.ADIE = 1;          // enable ADC interrupt requests
      IFS0bits.ADIF = 0;          // clear ADIF bit
}

void master_init() {
      srand(time(NULL));
      // Initial service
      service = SERV_YES;
      pos_service = 1;

      // Initial paddle coordinates
      p1x = PAD1_X;
      p1y = (LENGTH/2) - (PADDLE_L/2);
      p2x = PAD2_X;
      p2y = (LENGTH/2) - (PADDLE_L/2);

      // Initial ball coordinates
      bx = p1x + (PADDLE_W) + 1;
      by = p1y + (PADDLE_L/2);

      // Initial ball speed
      speed = 0;

      // Initial ball movement vector
      vector_x = 1;
      vector_y = ((rand() % 2) == 0) ? -1 : 1;
}

/* Checks if the ball hit a paddle
 * return: 0, if it didn't hit
 *             1, otherwise
 */
unsigned char check_paddle_hit() {
      unsigned char hit = 0;

      if (bx <= p1x+PADDLE_W && by >= p1y && by < p1y+PADDLE_L)
            hit = 1;
      if (bx >= p2x && by >= p2y && by < p2y+PADDLE_L)
            hit = 1;

      return hit;
}

/* Checks if the ball hit a wall
 * return: 0, if it didn't hit
 *             1, otherwise
 */
unsigned char check_wall_hit() {
      unsigned char hit = 0;
```

```
      if (by <= 0 || by >= LENGTH)
            hit = 1;

      return hit;
}

/* Checks if a point was made
 * return: 0, if no point made
 *             1-2, the winner
 */
int check_point_made() {
      unsigned char point = 0;

      if (bx <= 0) point = 2;
      else if (bx >= WIDTH) point = 1;

      return point;
}
```

- prueba1.c

```
#include <p30f4011.h>
#include <uart.h>

/*****************************************************************/
/* Configuration words                                           */
/*****************************************************************/
_FOSC(CSW_FSCM_OFF & EC_PLL16);
_FWDT(WDT_OFF);
_FBORPOR(MCLR_EN & PBOR_OFF & PWRT_OFF);
_FGS(CODE_PROT_OFF);

/*****************************************************************/
/* Hardware                                                      */
/*****************************************************************/

#define FXT       7372800         // CPU clock
#define PLL       16              // PLL configuration
#define FCY       (FXT * PLL) / 4 // Clock that feeds the UART

#define BAUD_RATE 115200
#define BRG       (FCY / (16L * BAUD_RATE)) - 1L

/*****************************************************************/
/* Constants                                                        */
/*****************************************************************/
#define    WIDTH      80 //40
#define    LENGTH           24

#define    BALL_L           1
#define    PADDLE_L         5
#define PADDLE_W      2

#define    PAD1_X           2
#define    PAD2_X           76 //36

#define SCORE1_X      31 //11
#define SCORE2_X      44 //24
#define SCORE_Y       1

#define UP            'i'
#define DOWN                'k'
#define BUZZ                'j'

#define FILL                "#"

/*****************************************************************/
/* Global Variable declaration                                   */
/*****************************************************************/
// Ball coordinates (Range: 0-WIDTH, 0-LENGTH)
unsigned int bx, by;
// Paddle 1 and 2 top left coordinates (Range: PAD1_X, 0-(LENGTH-PADDLE_L), PAD2_X, 0-
(LENGTH-PADDLE_L))
unsigned int p1x, p1y, p2x, p2y;
// Scoreboard
unsigned int score[2];
// Previous versions of ball and paddle variables
unsigned int pre_bx, pre_by, pre_p1y, pre_p2y;
// Current screen cursor position (Range: 0-WIDTH, 0-LENGTH)
unsigned int cx, cy;

/*****************************************************************/
/* Interrupts                                                    */
/*****************************************************************/
void _ISR _U1RXInterrupt() {
     unsigned char c = ReadUART1();
     //unsigned char c = U1RXREG & 0xFF;
     //if (U1MODEbits.PDSEL == 3) c = U1RXREG;
     //else c = U1RXREG & 0xFF;
     //unsigned char str[5];
     //sprintf(str, "%u", c);
     //WriteUART1(c);
     //while (BusyUART1());

     if (c == UP) if (p1y > 0) {pre_p1y = p1y; p1y -= 1;}
     if (c == DOWN) if (p1y < LENGTH-PADDLE_L) {pre_p1y = p1y; p1y += 1;}
     if (c == BUZZ) {
```

```
            WriteUART1(7);              // Send the buzzer character back to the UART
            while (BusyUART1());        // Wait until the character is transmitted
      }

      IFS0bits.U1RXIF = 0;
}

/*************************************************************/
/* Procedures                                                */
/*************************************************************/
void UARTConfig();
void slave1_init();
void clear_screen();
void draw_screen();

int main(void){
      UARTConfig();

      slave1_init();

      int j;
      for (j = 0; j < 400; j++) Delay5ms();
      clear_screen();
      draw_screen();
      while (1) {
            if (pre_bx != bx || pre_by != by || pre_p1y != p1y || pre_p2y != p2y)
                  draw_screen();
      }

      return 0;
}

void UARTConfig(){
      U1MODE = 0;                     // Clear UART config - to avoid problems with
bootloader

      // Config UART
      OpenUART1(UART_EN &             // Enable UART
                  UART_DIS_LOOPBACK &   // Disable loopback mode
                  UART_NO_PAR_8BIT &   // 8bits / No parity
                  UART_1STOPBIT,       // 1 Stop bit

                  UART_TX_PIN_NORMAL &  // Tx break bit normal
                  UART_TX_ENABLE,       // Enable Transmition

                  BRG);                 // Baudrate
      U1STAbits.URXISEL = 0;

      // Enable UART Rx Interrupts
      IEC0bits.U1RXIE = 1;
      IFS0bits.U1RXIF = 0;
}

void slave1_init() {
      // Initial paddle coordinates
      p1x = PAD1_X;
      p1y = (LENGTH/2) - (PADDLE_L/2);
      p2x = PAD2_X;
      p2y = (LENGTH/2) - (PADDLE_L/2);

      // Initial ball coordinates
      bx = p1x + (PADDLE_W) + 1;
      by = p1y + (PADDLE_L/2);// + BALL_L;

      // Initial scores
      score[0] = 0;
      score[1] = 0;

      // Initial previous values at same value
      pre_bx = bx;
      pre_by = by;
      pre_p1y = p1y;
      pre_p2y = p2y;

      draw_screen();
}
```

```
void clear_screen() {
	WriteUART1(12);
	while (BusyUART1());		// Wait until the character is transmitted
	cx = 0; cy = 0;
}

void move_up() {
	//if (cy > 0) {
		WriteUART1(27);
		while (BusyUART1());
		WriteUART1(91);
		while (BusyUART1());
		WriteUART1(65);
		while (BusyUART1());
		cy--;
	//}
}

void move_down() {
	//if (cy < WIDTH) {
		WriteUART1(27);
		while (BusyUART1());
		WriteUART1(91);
		while (BusyUART1());
		WriteUART1(66);
		while (BusyUART1());
		cy++;
	//}
}

void move_left() {
	//if (cx > 0) {
		WriteUART1(27);
		while (BusyUART1());
		WriteUART1(91);
		while (BusyUART1());
		WriteUART1(68);
		while (BusyUART1());
		cx--;
	//}
}

void move_right() {
	//if (cx < LENGTH) {
		WriteUART1(27);
		while (BusyUART1());
		WriteUART1(91);
		while (BusyUART1());
		WriteUART1(67);
		while (BusyUART1());
		cx++;
	//}
}

void draw_fill() {
	//if (cx < LENGTH) {
		putsUART1(FILL);
		while (BusyUART1());
		cx++;
	//}
}

void reset_cursor() {
	while (cx > 0) {
		move_left();
	}
	while (cy > 0) {
		move_up();
	}
}

void draw_number(unsigned int number);

void draw_screen() {
	int i, j;
	// Clear screen and reset cursor
	clear_screen();
```

```
	// Draw player 1 paddle
	while (cx < p1x) {
		move_right();
	}
	while (cy < p1y)  {
		move_down();
	}
	for (i = 0; i < PADDLE_L; i++) {
		for (j = 0; j < PADDLE_W; j++) {
			draw_fill();
		}
		for (j = 0; j < PADDLE_W; j++) {
			move_left();
		}
		move_down();
	}
	reset_cursor();

	// Draw player 2 paddle
	while (cx < p2x) {
		move_right();
	}
	while (cy < p2y)  {
		move_down();
	}
	for (i = 0; i < PADDLE_L; i++) {
		for (j = 0; j < PADDLE_W; j++) {
			draw_fill();
		}
		for (j = 0; j < PADDLE_W; j++) {
			move_left();
		}
		move_down();
	}
	reset_cursor();

	// Draw ball
	while (cx < bx) {
		move_right();
	}
	while (cy < by)  {
		move_down();
	}
	draw_fill();
	reset_cursor();

	// Draw player 1 scoreboard
	while (cx < SCORE1_X) {
		move_right();
	}
	while (cy < SCORE_Y)  {
		move_down();
	}
	draw_number(score[0]);
	reset_cursor();

	// Draw player 2 scoreboard
	while (cx < SCORE2_X) {
		move_right();
	}
	while (cy < SCORE_Y)  {
		move_down();
	}
	draw_number(score[1]);
	reset_cursor();

	//Restore preview values
	pre_bx = bx;
	pre_by = by;
	pre_p1y = p1y;
	pre_p2y = p2y;
}

void draw_number(unsigned int number) {
	switch (number) {
		// ####
```

```
// #  #
// #  #
// #  #
// ####
case 0:
      draw_fill();
      draw_fill();
      draw_fill();
      draw_fill();
            move_down();
            move_left();
            move_left();
            move_left();
            move_left();

      draw_fill();
            move_right();
            move_right();
      draw_fill();
            move_down();
            move_left();
            move_left();
            move_left();
            move_left();

      draw_fill();
            move_right();
            move_right();
      draw_fill();
            move_down();
            move_left();
            move_left();
            move_left();
            move_left();

      draw_fill();
            move_right();
            move_right();
      draw_fill();
            move_down();
            move_left();
            move_left();
            move_left();
            move_left();

      draw_fill();
      draw_fill();
      draw_fill();
      draw_fill();
      break;
//  ##
//  ##
//  ##
//  ##
//  ##
case 1:
            move_right();
      draw_fill();
      draw_fill();
            move_down();
            move_left();
            move_left();

      draw_fill();
      draw_fill();
            move_down();
            move_left();
            move_left();

      draw_fill();
      draw_fill();
            move_down();
            move_left();
            move_left();

      draw_fill();
      draw_fill();
```

```
            move_down();
            move_left();
            move_left();

      draw_fill();
      draw_fill();
      break;
// ####
//    #
// ####
// #
// ####
case 2:
      draw_fill();
      draw_fill();
      draw_fill();
      draw_fill();
            move_down();
            move_left();

      draw_fill();
            move_down();
            move_left();
            move_left();
            move_left();
            move_left();

      draw_fill();
      draw_fill();
      draw_fill();
      draw_fill();
            move_down();
            move_left();
            move_left();
            move_left();
            move_left();

      draw_fill();
            move_down();
            move_left();

      draw_fill();
      draw_fill();
      draw_fill();
      draw_fill();
      break;
// ####
//    #
// ####
//    #
// ####
case 3:
      draw_fill();
      draw_fill();
      draw_fill();
      draw_fill();
            move_down();
            move_left();

      draw_fill();
            move_down();
            move_left();
            move_left();
            move_left();
            move_left();

      draw_fill();
      draw_fill();
      draw_fill();
      draw_fill();
            move_down();
            move_left();

      draw_fill();
            move_down();
            move_left();
            move_left();
```

```
            move_left();
            move_left();

      draw_fill();
      draw_fill();
      draw_fill();
      draw_fill();
      break;
// #  #
// #  #
// ####
//    #
//    #
case 4:
      draw_fill();
            move_right();
            move_right();
      draw_fill();
            move_down();
            move_left();
            move_left();
            move_left();
            move_left();

      draw_fill();
            move_right();
            move_right();
      draw_fill();
            move_down();
            move_left();
            move_left();
            move_left();
            move_left();

      draw_fill();
      draw_fill();
      draw_fill();
      draw_fill();
            move_down();
            move_left();

      draw_fill();
            move_down();
            move_left();

      draw_fill();
      break;
// ####
// #
// ####
//    #
// ####
case 5:
      draw_fill();
      draw_fill();
      draw_fill();
      draw_fill();
            move_down();
            move_left();
            move_left();
            move_left();
            move_left();

      draw_fill();
            move_down();
            move_left();

      draw_fill();
      draw_fill();
      draw_fill();
      draw_fill();
            move_down();
            move_left();

      draw_fill();
            move_down();
            move_left();
```

```
            move_left();
            move_left();
            move_left();

      draw_fill();
      draw_fill();
      draw_fill();
      draw_fill();
      break;
// #
// #
// ####
// #  #
// ####
case 6:
      draw_fill();
            move_down();
            move_left();

      draw_fill();
            move_down();
            move_left();

      draw_fill();
      draw_fill();
      draw_fill();
      draw_fill();
            move_down();
            move_left();
            move_left();
            move_left();
            move_left();

      draw_fill();
            move_right();
            move_right();
      draw_fill();
            move_down();
            move_left();
            move_left();
            move_left();
            move_left();

      draw_fill();
      draw_fill();
      draw_fill();
      draw_fill();
      break;
// ####
//    #
//    #
//    #
//    #
case 7:
      draw_fill();
      draw_fill();
      draw_fill();
      draw_fill();
            move_down();
            move_left();

      draw_fill();
            move_down();
            move_left();

      draw_fill();
            move_down();
            move_left();

      draw_fill();
            move_down();
            move_left();

      draw_fill();
      break;
// ####
// #  #
```

```
// ####
// #  #
// ####
case 8:
      draw_fill();
      draw_fill();
      draw_fill();
      draw_fill();
            move_down();
            move_left();
            move_left();
            move_left();
            move_left();

      draw_fill();
            move_right();
            move_right();
      draw_fill();
            move_down();
            move_left();
            move_left();
            move_left();
            move_left();

      draw_fill();
      draw_fill();
      draw_fill();
      draw_fill();
            move_down();
            move_left();
            move_left();
            move_left();
            move_left();

      draw_fill();
            move_right();
            move_right();
      draw_fill();
            move_down();
            move_left();
            move_left();
            move_left();
            move_left();

      draw_fill();
      draw_fill();
      draw_fill();
      draw_fill();
      break;
// ####
// #  #
// ####
//    #
//    #
case 9:
      draw_fill();
      draw_fill();
      draw_fill();
      draw_fill();
            move_down();
            move_left();
            move_left();
            move_left();
            move_left();

      draw_fill();
            move_right();
            move_right();
      draw_fill();
            move_down();
            move_left();
            move_left();
            move_left();
            move_left();

      draw_fill();
      draw_fill();
```

```
                draw_fill();
                draw_fill();
                      move_down();
                      move_left();

                draw_fill();
                      move_down();
                      move_left();

                draw_fill();
                break;
        }
}
```

- prueba3a.c

```
#include <p30f4011.h>
#include <time.h>
#include <stdlib.h>
#include "can.h"

/*****************************************************************/
/* Configuration words                                           */
/*****************************************************************/
_FOSC(CSW_FSCM_OFF & EC_PLL16);
_FWDT(WDT_OFF);
_FBORPOR(MCLR_EN & PBOR_OFF & PWRT_OFF);
_FGS(CODE_PROT_OFF);

/*****************************************************************/
/* Constants                                                     */
/*****************************************************************/
#define    WIDTH      80
#define    LENGTH           24

#define    BALL_L           1
#define    PADDLE_L         5
#define PADDLE_W       2

#define    PAD1_X           2
#define    PAD2_X           76

#define SERV_NO        0
#define SERV_YES       1

#define    M_BALL           00
#define    M_BOUNCE         02
#define    M_POINT          04
#define    S1_PADDLE   10
#define    S1_SERVICE  11
#define    S2_PADDLE   20
#define    S2_SERVICE  21

/*****************************************************************/
/* Global Variable declaration                                   */
/*****************************************************************/
// Ball coordinates (Range: 0-WIDTH, 0-LENGTH)
unsigned int bx, by;
// Paddle 1 and 2 top left coordinates (Range: PAD1_X, 0-(LENGTH-PADDLE_L), PAD2_X, 0-
(LENGTH-PADDLE_L))
volatile unsigned int p1x, p1y, p2x, p2y;
// Ball movement vector (Values: -1.1, -1.1)
volatile int vector_x, vector_y;
// Ball speed (Range: 0-4)
volatile unsigned int speed;
// Service no/yes (Values: 0.1)
volatile unsigned char service;
// Possession service (Values: 1.2)
unsigned char pos_service;

/*****************************************************************/
/* Interrupts                                                    */
/*****************************************************************/
void _ISR _ADCInterrupt(void) {
      int ADCValue = ADCBUF0;           // get ADC value
      speed = ((float)ADCValue/1023.0)*4;
      IFS0bits.ADIF = 0;                // restore ADIF
}

void _ISR _C1Interrupt() {
      if (C1INTFbits.RX0IF == 1) {
            unsigned int id = C1RX0SIDbits.SID;
            switch (id) {
                  case S1_PADDLE:
                        p1y = C1RX0B1;
                        break;
                  case S1_SERVICE:
                        if (pos_service == 1) {
                              service = SERV_NO;
                              vector_x = 1;
                              vector_y = ((rand() % 2) == 0) ? -1 : 1;
                        }
```

```
                        break;
                  case S2_PADDLE:
                        p2y = C1RX0B1;
                        break;
                  case S2_SERVICE:
                        if (pos_service == 2) {
                              service = SERV_NO;
                              vector_x = -1;
                              vector_y = ((rand() % 2) == 0) ? -1 : 1;
                        }
                        break;
            }

            C1RX0CONbits.RXFUL = 0;      // Clear reception full status flag
            C1INTFbits.RX0IF = 0;
      }
      IFS1bits.C1IF = 0;
}

/*****************************************************************/
/* Procedures                                                    */
/*****************************************************************/
void CAN_config();
void ADC_config();
void master_init();
unsigned char check_paddle_hit();
unsigned char check_wall_hit();
int check_point_made();

int main(void)
{
      CAN_config();
      ADC_config();

      master_init();

      // mode: 0->nothing, 1->bounce, 2->point
      int mode, winner, i;
      unsigned int ball_coordinates[2];
      while (1) {
            mode = 0;
            winner = 0;

            // Update ball coordinates
            if (service) {
                  if (pos_service == 1) {
                        bx = p1x + (PADDLE_W) + 1;
                        by = p1y + (PADDLE_L/2);
                  } else {
                        bx = p2x - 2;
                        by = p2y + (PADDLE_L/2);
                  }
            } else {
                  bx += vector_x;
                  by += vector_y;
            }

            // Check bounces
            unsigned char paddle_bounce = check_paddle_hit();
            if (paddle_bounce) {
                  mode = 1;
                  vector_x = (vector_x == 1) ? -1 : 1;
                  if (bx < (WIDTH/2)) {  // If it bounced with paddle 1
                        if (vector_y == 1 && by < (p1y + (PADDLE_L/2))) vector_y = -1;
                        else if (vector_y == -1 && by > (p1y + (PADDLE_L/2))) vector_y = 1;
                  } else {                     // If it bounced with paddle 2
                        if (vector_y == 1 && by < (p2y + (PADDLE_L/2))) vector_y = -1;
                        else if (vector_y == -1 && by > (p2y + (PADDLE_L/2))) vector_y = 1;
                  }
            }
            unsigned char wall_bounce = check_wall_hit();
            if (wall_bounce) {
                  mode = 1;
                  vector_y = (vector_y == -1) ? 1 : -1;
            }

            // Check if someone has scored
```

```
            winner = check_point_made();
            if (winner) {
                  mode = 2;
                  if (winner == 1) {
                        bx = p2x - 1;
                        by = p2y + (PADDLE_L/2) - BALL_L;
                        pos_service = 2;
                  } else {
                        bx = p1x + (PADDLE_W) + 1;
                        by = p1y + (PADDLE_L/2) - BALL_L;
                        pos_service = 1;
                  }
                  service = SERV_YES;
            }

            // Send messages
            if (mode == 1) CANSendMsg(M_BOUNCE, 0, NULL);
            else if (mode == 2) CANSendMsg(M_POINT, 1, &winner);
            ball_coordinates[0] = bx;
            ball_coordinates[1] = by;
            CANSendMsg(M_BALL, 2, ball_coordinates);

            // Wait until next update
            for (i = 0; i < 100-20*speed; i++) Delay5ms();
      }

    return 0;
}

void CAN_config() {
      /* Initialize CAN */
      C1CTRLbits.REQOP = 0b100;               // Set configuration mode
      while(C1CTRLbits.OPMODE != 0b100);      // Wait until configuration mode

      C1CTRLbits.CANCKS = 1;                        // FCAN = FCY

      /* Baud rate */

      // BTR config 1
      C1CFG1bits.BRP = 0;                           // Prescaler to 2/Fcan
      C1CFG1bits.SJW = 0;                           // 1 TQ

      // BTR config 2
      C1CFG2bits.PRSEG  = 0b000;              // 1 TQs
      C1CFG2bits.SEG1PH = 0b011;              // 4 TQs
      C1CFG2bits.SEG2PH = 0b001;              // 2 TQs

      /* Interrupts */

      // General CAN interrupt
      IEC1bits.C1IE = 1;                // Enable general CAN interrupt
      IFS1bits.C1IF = 0;                // Clear general CAN interrupt flag

      // Local CAN interrupts
      C1INTEbits.RX0IE = 1;       // Enable CAN interrupt associated to rx buffer 0
      C1INTFbits.RX0IF = 0;       // Clear CAN interrupt flag associated to rx buffer 0

      /* Tx buffer 0 */

      // General transmission configuration
      C1TX0CONbits.TXREQ = 0;     // Clear transmission request flag

      /* Rx buffer 0 */

      // General reception configuration
      C1RX0CONbits.RXFUL = 0;     // Clear reception full status flag
      C1RX0CONbits.DBEN = 0;            // Disable double buffer

      // Configure acceptance mask
      C1RXM0SIDbits.SID = 0;            // No bits to compare
      C1RXM0SIDbits.MIDE = 1;     // Identifier mode as determined by EXIDE
      C1RX0CONbits.FILHIT0 = 0;   // Link to acceptance filter 0

      // Configure acceptance filters
      C1RXF0SIDbits.EXIDE = 0;    // Enable filter for standard identifier
      C1RXF0SIDbits.SID = 0;            // Doesn't matter the value as mask is '0'
```

```
      C1CTRLbits.REQOP = 0b000;               // Set normal mode
      while(C1CTRLbits.OPMODE != 0b000);// Wait until normal mode
}

void ADC_config() {
      TRISBbits.TRISB7 = 1;  // Declare AN7 as input
      ADPCFG = 0xFF7F;       // all PORTB = Digital; RB7 = analog
      ADCON1 = 0x00E4;       // SAMP bit is auto-set
                                    // sampling begins immediately after last
conversion
      ADCHS = 0x0007;             // Connect RB7/AN7 as CH0 input ..
      ADCSSL = 0;
      ADCON3 = 0x0000;

      // Set ADC clock 32 times slower than System clock
      ADCON3bits.ADCS = 31;

      // Set time bewteen conversions to 31 ADC clock cycles
      ADCON3bits.SAMC = 31;
      ADCON2 = 0;

      // Set at 16 the number of conversions before throwing an interruption
      ADCON2bits.SMPI = 0b1111;
      ADCON1bits.ADON = 1;   // turn ADC ON
      ADCON1bits.SAMP = 1;   // start sampling

      //Configure interrupts
      IEC0bits.ADIE = 1;          // enable ADC interrupt requests
      IFS0bits.ADIF = 0;          // clear ADIF bit
}

void master_init() {
      srand(time(NULL));
      // Initial service
      service = SERV_YES;
      pos_service = 1;

      // Initial paddle coordinates
      p1x = PAD1_X;
      p1y = (LENGTH/2) - (PADDLE_L/2);
      p2x = PAD2_X;
      p2y = (LENGTH/2) - (PADDLE_L/2);

      // Initial ball coordinates
      bx = p1x + (PADDLE_W) + 1;
      by = p1y + (PADDLE_L/2);

      // Initial ball speed
      speed = 0;

      // Initial ball movement vector
      vector_x = 1;
      vector_y = ((rand() % 2) == 0) ? -1 : 1;
}

/* Checks if the ball hit a paddle
 * return: 0, if it didn't hit
 *             1, otherwise
 */
unsigned char check_paddle_hit() {
      unsigned char hit = 0;

      if (bx <= p1x+PADDLE_W && by >= p1y && by < p1y+PADDLE_L)
            hit = 1;
      if (bx >= p2x && by >= p2y && by < p2y+PADDLE_L)
            hit = 1;

      return hit;
}

/* Checks if the ball hit a wall
 * return: 0, if it didn't hit
 *             1, otherwise
 */
unsigned char check_wall_hit() {
      unsigned char hit = 0;
```

```
      if (by <= 0 || by >= LENGTH)
            hit = 1;

      return hit;
}

/* Checks if a point was made
 * return: 0, if no point made
 *             1-2, the winner
 */
int check_point_made() {
      unsigned char point = 0;

      if (bx <= 0) point = 2;
      else if (bx >= WIDTH) point = 1;

      return point;
}
```

- prueba3b.c

```
#include <p30f4011.h>
#include <uart.h>
#include "can.h"

/*****************************************************************/
/* Configuration words                                           */
/*****************************************************************/
_FOSC(CSW_FSCM_OFF & EC_PLL16);
_FWDT(WDT_OFF);
_FBORPOR(MCLR_EN & PBOR_OFF & PWRT_OFF);
_FGS(CODE_PROT_OFF);

/*****************************************************************/
/* Hardware                                                      */
/*****************************************************************/

#define FXT       7372800         // CPU clock
#define PLL       16              // PLL configuration
#define FCY       (FXT * PLL) / 4 // Clock that feeds the UART

#define BAUD_RATE 115200
#define BRG       (FCY / (16L * BAUD_RATE)) - 1L

/*****************************************************************/
/* Constants                                                                  */
/*****************************************************************/
#define    WIDTH      80
#define    LENGTH           24

#define    BALL_L           1
#define    PADDLE_L         5
#define PADDLE_W      2

#define    PAD1_X           2
#define    PAD2_X           76

#define SCORE1_X      31
#define SCORE2_X      44
#define SCORE_Y       1

#define UP1           'w'
#define DOWN1         's'
#define SERVICE1      'a'
#define UP2           'i'
#define DOWN2         'k'
#define SERVICE2      'j'

#define FILL                "#"

#define    M_BALL           00
#define    M_BOUNCE         02
#define    M_POINT          04
#define    S1_PADDLE  10
#define    S1_SERVICE 11
#define    S2_PADDLE  20
#define    S2_SERVICE 21
```

```
/************************************************************/
/* Global Variable declaration                              */
/************************************************************/
// Ball coordinates (Range: 0-WIDTH, 0-LENGTH)
volatile unsigned int bx, by;
// Paddle 1 and 2 top left coordinates (Range: PAD1_X, 0-(LENGTH-PADDLE_L), PAD2_X, 0-
(LENGTH-PADDLE_L))
volatile unsigned int p1x, p1y, p2x, p2y;
// Scoreboard
volatile unsigned int score[2];
// Previous versions of ball and paddle variables
volatile unsigned int pre_bx, pre_by, pre_p1y, pre_p2y;
// Current screen cursor position (Range: 0-WIDTH, 0-LENGTH)
unsigned int cx, cy;

/************************************************************/
/* Interrupts                                               */
/************************************************************/
void _ISR _U1RXInterrupt() {
      unsigned char c = ReadUART1();
      //WriteUART1(c);
      //while (BusyUART1());

      if (c == UP1) if (p1y > 0) {pre_p1y = p1y; p1y -= 1; CANSendMsg(S1_PADDLE, 1,
&p1y);}
      if (c == DOWN1) if (p1y < LENGTH-PADDLE_L) {pre_p1y = p1y; p1y += 1; CANSendMsg
(S1_PADDLE, 1, &p1y);}
      if (c == SERVICE1) CANSendMsg(S1_SERVICE, 0, 0);

      if (c == UP2) if (p2y > 0) {pre_p2y = p2y; p2y -= 1; CANSendMsg(S2_PADDLE, 1,
&p2y);}
      if (c == DOWN2) if (p2y < LENGTH-PADDLE_L) {pre_p2y = p2y; p2y += 1; CANSendMsg
(S2_PADDLE, 1, &p2y);}
      if (c == SERVICE2) CANSendMsg(S2_SERVICE, 0, 0);

      IFS0bits.U1RXIF = 0;
}

void _ISR _C1Interrupt() {
      if (C1INTFbits.RX0IF == 1) {
            int winner;
            unsigned int id = C1RX0SIDbits.SID;
            switch (id) {
                  case M_BALL:
                        pre_bx = bx;
                        bx = C1RX0B1;
                        pre_by = by;
                        by = C1RX0B2;
                        break;
                  case M_BOUNCE:
                        WriteUART1(7);              // Send the buzzer character back to
the UART
                        while (BusyUART1());        // Wait until the character is
transmitted
                        break;
                  case M_POINT:
                        winner = C1RX0B1;
                        score[winner-1] = (score[winner-1] + 1) % 10;
                        break;
            }

            C1RX0CONbits.RXFUL = 0;      // Clear reception full status flag
            C1INTFbits.RX0IF = 0;
      }
      IFS1bits.C1IF = 0;
}

/************************************************************/
/* Procedures                                               */
/************************************************************/
void UARTConfig();
void CAN_config();
void slave_init();
void clear_screen();
void draw_screen();

int main(void){
```

```
	UARTConfig();
	CAN_config();

	slave_init();

	int j;
	for (j = 0; j < 400; j++) Delay5ms();
	clear_screen();
	draw_screen();
	while (1) {
		if (pre_bx != bx || pre_by != by || pre_p1y != p1y || pre_p2y != p2y) {
			draw_screen();
			//CANSendMsg(S1_PADDLE, 1, &p1y);
			//CANSendMsg(S2_PADDLE, 1, &p2y);
		}
	}

	return 0;
}

void UARTConfig(){
	U1MODE = 0;                     // Clear UART config - to avoid problems with
bootloader

	// Config UART
	OpenUART1(UART_EN &             // Enable UART
			UART_DIS_LOOPBACK &   // Disable loopback mode
			UART_NO_PAR_8BIT &   // 8bits / No parity
			UART_1STOPBIT,       // 1 Stop bit

			UART_TX_PIN_NORMAL &  // Tx break bit normal
			UART_TX_ENABLE,       // Enable Transmition

			BRG);                 // Baudrate
	U1STAbits.URXISEL = 0;

	// Enable UART Rx Interrupts
	IEC0bits.U1RXIE = 1;
	IFS0bits.U1RXIF = 0;
}

void CAN_config() {
	/* Initialize CAN */
	C1CTRLbits.REQOP = 0b100;               // Set configuration mode
	while(C1CTRLbits.OPMODE != 0b100);      // Wait until configuration mode

	C1CTRLbits.CANCKS = 1;                        // FCAN = FCY

	/* Baud rate */

	// BTR config 1
	C1CFG1bits.BRP = 0;                           // Prescaler to 2/Fcan
	C1CFG1bits.SJW = 0;                           // 1 TQ

	// BTR config 2
	C1CFG2bits.PRSEG  = 0b000;              // 1 TQs
	C1CFG2bits.SEG1PH = 0b011;              // 4 TQs
	C1CFG2bits.SEG2PH = 0b001;              // 2 TQs

	/* Interrupts */

	// General CAN interrupt
	IEC1bits.C1IE = 1;                // Enable general CAN interrupt
	IFS1bits.C1IF = 0;                // Clear general CAN interrupt flag

	// Local CAN interrupts
	C1INTEbits.RX0IE = 1;        // Enable CAN interrupt associated to rx buffer 0
	C1INTFbits.RX0IF = 0;        // Clear CAN interrupt flag associated to rx buffer 0

	/* Tx buffer 0 */

	// General transmission configuration
	C1TX0CONbits.TXREQ = 0;      // Clear transmission request flag

	/* Rx buffer 0 */

	// General reception configuration
```

```
	C1RX0CONbits.RXFUL = 0;	// Clear reception full status flag
	C1RX0CONbits.DBEN = 0;		// Disable double buffer

	// Configure acceptance mask
	C1RXM0SIDbits.SID = 0b0000000001; // Mask to check if sid is odd or even
	C1RXM0SIDbits.MIDE = 1;		// Identifier mode as determined by EXIDE
	C1RX0CONbits.FILHIT0 = 0;		// Link to acceptance filter 0

	// Configure acceptance filters
	C1RXF0SIDbits.EXIDE = 0;		// Enable filter for standard identifier
	C1RXF0SIDbits.SID = 0b000000000;  // Accept messages with even identifier

	C1CTRLbits.REQOP = 0b000;		// Set normal mode
	while(C1CTRLbits.OPMODE != 0b000);// Wait until normal mode
}

void slave_init() {
	// Initial paddle coordinates
	p1x = PAD1_X;
	p1y = (LENGTH/2) - (PADDLE_L/2);
	p2x = PAD2_X;
	p2y = (LENGTH/2) - (PADDLE_L/2);

	// Initial ball coordinates
	bx = p1x + (PADDLE_W) + 1;
	by = p1y + (PADDLE_L/2);

	// Initial scores
	score[0] = 0;
	score[1] = 0;

	// Initial previous values at same value
	pre_bx = bx;
	pre_by = by;
	pre_p1y = p1y;
	pre_p2y = p2y;

	draw_screen();
}

void clear_screen() {
	WriteUART1(12);
	while (BusyUART1());	// Wait until the character is transmitted
	cx = 0; cy = 0;
}

void move_up() {
	//if (cy > 0) {
		WriteUART1(27);
		while (BusyUART1());
		WriteUART1(91);
		while (BusyUART1());
		WriteUART1(65);
		while (BusyUART1());
		cy--;
	//}
}

void move_down() {
	//if (cy < WIDTH) {
		WriteUART1(27);
		while (BusyUART1());
		WriteUART1(91);
		while (BusyUART1());
		WriteUART1(66);
		while (BusyUART1());
		cy++;
	//}
}

void move_left() {
	//if (cx > 0) {
		WriteUART1(27);
		while (BusyUART1());
		WriteUART1(91);
		while (BusyUART1());
		WriteUART1(68);
```

```
            while (BusyUART1());
            cx--;
      //}
}

void move_right() {
      //if (cx < LENGTH) {
            WriteUART1(27);
            while (BusyUART1());
            WriteUART1(91);
            while (BusyUART1());
            WriteUART1(67);
            while (BusyUART1());
            cx++;
      //}
}

void draw_fill() {
      //if (cx < LENGTH) {
            putsUART1(FILL);
            while (BusyUART1());
            cx++;
      //}
}

void reset_cursor() {
      while (cx > 0) {
            move_left();
      }
      while (cy > 0) {
            move_up();
      }
}

void draw_number(unsigned int number);

void draw_screen() {
      int i, j;
      // Clear screen and reset cursor
      clear_screen();

      // Draw player 1 paddle
      while (cx < p1x) {
            move_right();
      }
      while (cy < p1y)  {
            move_down();
      }
      for (i = 0; i < PADDLE_L; i++) {
            for (j = 0; j < PADDLE_W; j++) {
                  draw_fill();
            }
            for (j = 0; j < PADDLE_W; j++) {
                  move_left();
            }
            move_down();
      }
      reset_cursor();

      // Draw player 2 paddle
      while (cx < p2x) {
            move_right();
      }
      while (cy < p2y)  {
            move_down();
      }
      for (i = 0; i < PADDLE_L; i++) {
            for (j = 0; j < PADDLE_W; j++) {
                  draw_fill();
            }
            for (j = 0; j < PADDLE_W; j++) {
                  move_left();
            }
            move_down();
      }
      reset_cursor();
```

```
	// Draw ball
	while (cx < bx) {
		move_right();
	}
	while (cy < by)  {
		move_down();
	}
	draw_fill();
	reset_cursor();

	// Draw player 1 scoreboard
	while (cx < SCORE1_X) {
		move_right();
	}
	while (cy < SCORE_Y)  {
		move_down();
	}
	draw_number(score[0]);
	reset_cursor();

	// Draw player 2 scoreboard
	while (cx < SCORE2_X) {
		move_right();
	}
	while (cy < SCORE_Y)  {
		move_down();
	}
	draw_number(score[1]);
	reset_cursor();

	//Restore preview values
	pre_bx = bx;
	pre_by = by;
	pre_p1y = p1y;
	pre_p2y = p2y;
}

void draw_number(unsigned int number) {
	switch (number) {
		// ####
		// #  #
		// #  #
		// #  #
		// ####
		case 0:
			draw_fill();
			draw_fill();
			draw_fill();
			draw_fill();
				move_down();
				move_left();
				move_left();
				move_left();
				move_left();

			draw_fill();
				move_right();
				move_right();
			draw_fill();
				move_down();
				move_left();
				move_left();
				move_left();
				move_left();

			draw_fill();
				move_right();
				move_right();
			draw_fill();
				move_down();
				move_left();
				move_left();
				move_left();
				move_left();

			draw_fill();
				move_right();
```

```
            move_right();
      draw_fill();
            move_down();
            move_left();
            move_left();
            move_left();
            move_left();

      draw_fill();
      draw_fill();
      draw_fill();
      draw_fill();
      break;
//  ##
//  ##
//  ##
//  ##
//  ##
case 1:
            move_right();
      draw_fill();
      draw_fill();
            move_down();
            move_left();
            move_left();

      draw_fill();
      draw_fill();
            move_down();
            move_left();
            move_left();

      draw_fill();
      draw_fill();
            move_down();
            move_left();
            move_left();

      draw_fill();
      draw_fill();
            move_down();
            move_left();
            move_left();

      draw_fill();
      draw_fill();
      break;
// ####
//    #
// ####
// #
// ####
case 2:
      draw_fill();
      draw_fill();
      draw_fill();
      draw_fill();
            move_down();
            move_left();

      draw_fill();
            move_down();
            move_left();
            move_left();
            move_left();
            move_left();

      draw_fill();
      draw_fill();
      draw_fill();
      draw_fill();
            move_down();
            move_left();
            move_left();
            move_left();
            move_left();
```

```
      draw_fill();
            move_down();
            move_left();

      draw_fill();
      draw_fill();
      draw_fill();
      draw_fill();
      break;
// ####
//    #
// ####
//    #
// ####
case 3:
      draw_fill();
      draw_fill();
      draw_fill();
      draw_fill();
            move_down();
            move_left();

      draw_fill();
            move_down();
            move_left();
            move_left();
            move_left();
            move_left();

      draw_fill();
      draw_fill();
      draw_fill();
      draw_fill();
            move_down();
            move_left();

      draw_fill();
            move_down();
            move_left();
            move_left();
            move_left();
            move_left();

      draw_fill();
      draw_fill();
      draw_fill();
      draw_fill();
      break;
// #  #
// #  #
// ####
//    #
//    #
case 4:
      draw_fill();
            move_right();
            move_right();
      draw_fill();
            move_down();
            move_left();
            move_left();
            move_left();
            move_left();

      draw_fill();
            move_right();
            move_right();
      draw_fill();
            move_down();
            move_left();
            move_left();
            move_left();
            move_left();

      draw_fill();
      draw_fill();
      draw_fill();
```

```
      draw_fill();
            move_down();
            move_left();

      draw_fill();
            move_down();
            move_left();

      draw_fill();
      break;
// ####
// #
// ####
//    #
// ####
case 5:
      draw_fill();
      draw_fill();
      draw_fill();
      draw_fill();
            move_down();
            move_left();
            move_left();
            move_left();
            move_left();

      draw_fill();
            move_down();
            move_left();

      draw_fill();
      draw_fill();
      draw_fill();
      draw_fill();
            move_down();
            move_left();

      draw_fill();
            move_down();
            move_left();
            move_left();
            move_left();
            move_left();

      draw_fill();
      draw_fill();
      draw_fill();
      draw_fill();
      break;
// #
// #
// ####
// #  #
// ####
case 6:
      draw_fill();
            move_down();
            move_left();

      draw_fill();
            move_down();
            move_left();

      draw_fill();
      draw_fill();
      draw_fill();
      draw_fill();
            move_down();
            move_left();
            move_left();
            move_left();
            move_left();

      draw_fill();
            move_right();
            move_right();
      draw_fill();
```

```
            move_down();
            move_left();
            move_left();
            move_left();
            move_left();

      draw_fill();
      draw_fill();
      draw_fill();
      draw_fill();
      break;
// ####
//    #
//    #
//    #
//    #
case 7:
      draw_fill();
      draw_fill();
      draw_fill();
      draw_fill();
            move_down();
            move_left();

      draw_fill();
            move_down();
            move_left();

      draw_fill();
            move_down();
            move_left();

      draw_fill();
            move_down();
            move_left();

      draw_fill();
      break;
// ####
// #  #
// ####
// #  #
// ####
case 8:
      draw_fill();
      draw_fill();
      draw_fill();
      draw_fill();
            move_down();
            move_left();
            move_left();
            move_left();
            move_left();

      draw_fill();
            move_right();
            move_right();
      draw_fill();
            move_down();
            move_left();
            move_left();
            move_left();
            move_left();

      draw_fill();
      draw_fill();
      draw_fill();
      draw_fill();
            move_down();
            move_left();
            move_left();
            move_left();
            move_left();

      draw_fill();
            move_right();
            move_right();
```

```
                draw_fill();
                        move_down();
                        move_left();
                        move_left();
                        move_left();
                        move_left();

                draw_fill();
                draw_fill();
                draw_fill();
                draw_fill();
                break;
        // ####
        // #  #
        // ####
        //    #
        //    #
        case 9:
                draw_fill();
                draw_fill();
                draw_fill();
                draw_fill();
                        move_down();
                        move_left();
                        move_left();
                        move_left();
                        move_left();

                draw_fill();
                        move_right();
                        move_right();
                draw_fill();
                        move_down();
                        move_left();
                        move_left();
                        move_left();
                        move_left();

                draw_fill();
                draw_fill();
                draw_fill();
                draw_fill();
                        move_down();
                        move_left();

                draw_fill();
                        move_down();
                        move_left();

                draw_fill();
                break;
        }
}
```

- prueba4.c

```
#include <p30f4011.h>
#include <uart.h>

/*************************************************************/
/* Configuration words                                       */
/*************************************************************/
_FOSC(CSW_FSCM_OFF & EC_PLL16);
_FWDT(WDT_OFF);
_FBORPOR(MCLR_EN & PBOR_OFF & PWRT_OFF);
_FGS(CODE_PROT_OFF);

/*************************************************************/
/* Hardware                                                  */
/*************************************************************/

#define FXT       7372800         // CPU clock
#define PLL       16              // PLL configuration
#define FCY       (FXT * PLL) / 4 // Clock that feeds the UART

#define BAUD_RATE 115200
#define BRG       (FCY / (16L * BAUD_RATE)) - 1L

/*************************************************************/
/* Constants                                                                 */
/*************************************************************/
#define    WIDTH      80 //40
#define    LENGTH           24

#define    BALL_L           1
#define    PADDLE_L         5
#define PADDLE_W      2

#define    PAD1_X           2
#define    PAD2_X           76 //36

#define SCORE1_X      31 //11
#define SCORE2_X      44 //24
#define SCORE_Y       1

#define UP            'i'
#define DOWN                'k'
#define BUZZ                'j'

#define FILL                "#"

#define    M_BALL           00
#define    M_BOUNCE         02
#define    M_POINT          04
#define    S1_PADDLE  10
#define    S1_SERVICE 11
#define    S2_PADDLE  20
#define    S2_SERVICE 21

/*************************************************************/
/* Global Variable declaration                               */
/*************************************************************/
// Ball coordenates (Range: 0-WIDTH, 0-LENGTH)
unsigned int bx, by;
// Paddle 1 and 2 top left coordinates (Range: PAD1_X, 0-(LENGTH-PADDLE_L), PAD2_X, 0-
(LENGTH-PADDLE_L))
unsigned int p1x, p1y, p2x, p2y;
// Scoreboard
unsigned int score[2];
// Previous versions of ball and paddle variables
unsigned int pre_bx, pre_by, pre_p1y, pre_p2y;
// Current screen cursor position (Range: 0-WIDTH, 0-LENGTH)
unsigned int cx, cy;

/*************************************************************/
/* Interrupts                                                */
/*************************************************************/
void _ISR _U1RXInterrupt() {
      unsigned char c = ReadUART1();
      //unsigned char c = U1RXREG & 0xFF;
      //if (U1MODEbits.PDSEL == 3) c = U1RXREG;
      //else c = U1RXREG & 0xFF;
```

```
	//unsigned char str[5];
	//sprintf(str, "%u", c);
	//WriteUART1(c);
	//while (BusyUART1());

	if (c == UP) if (p1y > 0) {
		pre_p1y = p1y; p1y -= 1;
		//WriteUART1(105);
		//while (BusyUART1());
		CANSendMsg(S1_PADDLE, 1, &p1y);
		//WriteUART1(105);
		//while (BusyUART1());
	}
	if (c == DOWN) if (p1y < LENGTH-PADDLE_L) {
		pre_p1y = p1y; p1y += 1;
		//WriteUART1(105);
		//while (BusyUART1());
		CANSendMsg(S1_PADDLE, 1, &p1y);
		//WriteUART1(105);
		//while (BusyUART1());
	}
	if (c == BUZZ) {
		WriteUART1(7);              // Send the buzzer character back to the UART
		while (BusyUART1());   // Wait until the character is transmitted
	}

	IFS0bits.U1RXIF = 0;
}

void _ISR _C1Interrupt() {
	if (C1INTFbits.RX0IF == 1) {
		WriteUART1(105);
		while (BusyUART1());
		unsigned int id = C1RX0SIDbits.SID;
		switch (id) {
			case S2_PADDLE:
				pre_p2y = p2y;
				p2y = C1RX0B1;
				break;
		}

		C1RX0CONbits.RXFUL = 0;      // Clear reception full status flag
		C1INTFbits.RX0IF = 0;
	}
	IFS1bits.C1IF = 0;
}

/*************************************************************/
/* Procedures                                                */
/*************************************************************/
void UARTConfig();
void CAN_config();
void slave1_init();
void clear_screen();
void draw_screen();

int main(void){
	UARTConfig();
	CAN_config();

	slave1_init();

	int j;
	for (j = 0; j < 1600; j++) Delay5ms();
	clear_screen();
	draw_screen();
	while (1) {
		if (pre_bx != bx || pre_by != by || pre_p1y != p1y || pre_p2y != p2y)
			draw_screen();
	}

	return 0;
}

void UARTConfig(){
	U1MODE = 0;                     // Clear UART config - to avoid problems with
bootloader
```

```
	// Config UART
	OpenUART1(UART_EN &             // Enable UART
		  UART_DIS_LOOPBACK &   // Disable loopback mode
		  UART_NO_PAR_8BIT &   // 8bits / No parity
		  UART_1STOPBIT,       // 1 Stop bit

		  UART_TX_PIN_NORMAL &  // Tx break bit normal
		  UART_TX_ENABLE,       // Enable Transmition

		  BRG);                 // Baudrate
	U1STAbits.URXISEL = 0;

	// Enable UART Rx Interrupts
	IEC0bits.U1RXIE = 1;
	IFS0bits.U1RXIF = 0;
}

void CAN_config() {
	/* Initialize CAN */
	C1CTRLbits.REQOP = 0b100;               // Set configuration mode
	while(C1CTRLbits.OPMODE != 0b100);      // Wait until configuration mode

	C1CTRLbits.CANCKS = 1;                        // FCAN = FCY

	/* Baud rate */

	// BTR config 1
	C1CFG1bits.BRP = 0;                           // Prescaler to 2/Fcan
	C1CFG1bits.SJW = 0;                           // 1 TQ

	// BTR config 2
	C1CFG2bits.PRSEG  = 0b000;              // 1 TQs
	C1CFG2bits.SEG1PH = 0b011;              // 4 TQs
	C1CFG2bits.SEG2PH = 0b001;              // 2 TQs

	/* Interrupts */

	// General CAN interrupt
	IEC1bits.C1IE = 1;                // Enable general CAN interrupt
	IFS1bits.C1IF = 0;                // Clear general CAN interrupt flag

	// Local CAN interrupts
	C1INTEbits.RX0IE = 1;       // Enable CAN interrupt associated to rx buffer 0
	C1INTFbits.RX0IF = 0;       // Clear CAN interrupt flag associated to rx buffer 0

	/* Tx buffer 0 */

	// General transmission configuration
	C1TX0CONbits.TXREQ = 0;     // Clear transmission request flag

	/* Rx buffer 0 */

	// General reception configuration
	C1RX0CONbits.RXFUL = 0;     // Clear reception full status flag
	C1RX0CONbits.DBEN = 0;            // Disable double buffer

	// Configure acceptance mask
	C1RXM0SIDbits.SID = 0; // Mask to check if sid is odd or even
	C1RXM0SIDbits.MIDE = 1;                 // Identifier mode as determined by EXIDE
	C1RX0CONbits.FILHIT0 = 0;               // Link to acceptance filter 0

	// Configure acceptance filters
	C1RXF0SIDbits.EXIDE = 0;                // Enable filter for standard identifier
	C1RXF0SIDbits.SID = 0;      // Accept messages with even identifier

	C1CTRLbits.REQOP = 0b000;               // Set normal mode
	while(C1CTRLbits.OPMODE != 0b000);// Wait until normal mode
}

void slave1_init() {
	// Initial paddle coordinates
	p1x = PAD1_X;
	p1y = (LENGTH/2) - (PADDLE_L/2);
	p2x = PAD2_X;
	p2y = (LENGTH/2) - (PADDLE_L/2);
```

```
	// Initial ball coordinates
	bx = p1x + (PADDLE_W) + 1;
	by = p1y + (PADDLE_L/2);// + BALL_L;

	// Initial scores
	score[0] = 0;
	score[1] = 0;

	// Initial previous values at same value
	pre_bx = bx;
	pre_by = by;
	pre_p1y = p1y;
	pre_p2y = p2y;

	draw_screen();
}

void clear_screen() {
	WriteUART1(12);
	while (BusyUART1());		// Wait until the character is transmitted
	cx = 0; cy = 0;
}

void move_up() {
	//if (cy > 0) {
		WriteUART1(27);
		while (BusyUART1());
		WriteUART1(91);
		while (BusyUART1());
		WriteUART1(65);
		while (BusyUART1());
		cy--;
	//}
}

void move_down() {
	//if (cy < WIDTH) {
		WriteUART1(27);
		while (BusyUART1());
		WriteUART1(91);
		while (BusyUART1());
		WriteUART1(66);
		while (BusyUART1());
		cy++;
	//}
}

void move_left() {
	//if (cx > 0) {
		WriteUART1(27);
		while (BusyUART1());
		WriteUART1(91);
		while (BusyUART1());
		WriteUART1(68);
		while (BusyUART1());
		cx--;
	//}
}

void move_right() {
	//if (cx < LENGTH) {
		WriteUART1(27);
		while (BusyUART1());
		WriteUART1(91);
		while (BusyUART1());
		WriteUART1(67);
		while (BusyUART1());
		cx++;
	//}
}

void draw_fill() {
	//if (cx < LENGTH) {
		putsUART1(FILL);
		while (BusyUART1());
		cx++;
	//}
```

```
}

void reset_cursor() {
      while (cx > 0) {
            move_left();
      }
      while (cy > 0) {
            move_up();
      }
}

void draw_number(unsigned int number);

void draw_screen() {
      int i, j;
      // Clear screen and reset cursor
      clear_screen();

      // Draw player 1 paddle
      while (cx < p1x) {
            move_right();
      }
      while (cy < p1y)  {
            move_down();
      }
      for (i = 0; i < PADDLE_L; i++) {
            for (j = 0; j < PADDLE_W; j++) {
                  draw_fill();
            }
            for (j = 0; j < PADDLE_W; j++) {
                  move_left();
            }
            move_down();
      }
      reset_cursor();

      // Draw player 2 paddle
      while (cx < p2x) {
            move_right();
      }
      while (cy < p2y)  {
            move_down();
      }
      for (i = 0; i < PADDLE_L; i++) {
            for (j = 0; j < PADDLE_W; j++) {
                  draw_fill();
            }
            for (j = 0; j < PADDLE_W; j++) {
                  move_left();
            }
            move_down();
      }
      reset_cursor();

      // Draw ball
      while (cx < bx) {
            move_right();
      }
      while (cy < by)  {
            move_down();
      }
      draw_fill();
      reset_cursor();

      // Draw player 1 scoreboard
      while (cx < SCORE1_X) {
            move_right();
      }
      while (cy < SCORE_Y)  {
            move_down();
      }
      draw_number(score[0]);
      reset_cursor();

      // Draw player 2 scoreboard
      while (cx < SCORE2_X) {
            move_right();
```

```
	}
	while (cy < SCORE_Y)  {
		move_down();
	}
	draw_number(score[1]);
	reset_cursor();

	//Restore preview values
	pre_bx = bx;
	pre_by = by;
	pre_p1y = p1y;
	pre_p2y = p2y;
}

void draw_number(unsigned int number) {
	switch (number) {
		// ####
		// #  #
		// #  #
		// #  #
		// ####
		case 0:
			draw_fill();
			draw_fill();
			draw_fill();
			draw_fill();
				move_down();
				move_left();
				move_left();
				move_left();
				move_left();

			draw_fill();
				move_right();
				move_right();
			draw_fill();
				move_down();
				move_left();
				move_left();
				move_left();
				move_left();

			draw_fill();
				move_right();
				move_right();
			draw_fill();
				move_down();
				move_left();
				move_left();
				move_left();
				move_left();

			draw_fill();
				move_right();
				move_right();
			draw_fill();
				move_down();
				move_left();
				move_left();
				move_left();
				move_left();

			draw_fill();
			draw_fill();
			draw_fill();
			draw_fill();
			break;
		//  ##
		//  ##
		//  ##
		//  ##
		//  ##
		case 1:
				move_right();
			draw_fill();
			draw_fill();
				move_down();
```

```
            move_left();
            move_left();

      draw_fill();
      draw_fill();
            move_down();
            move_left();
            move_left();

      draw_fill();
      draw_fill();
            move_down();
            move_left();
            move_left();

      draw_fill();
      draw_fill();
            move_down();
            move_left();
            move_left();

      draw_fill();
      draw_fill();
      break;
// ####
//    #
// ####
// #
// ####
case 2:
      draw_fill();
      draw_fill();
      draw_fill();
      draw_fill();
            move_down();
            move_left();

      draw_fill();
            move_down();
            move_left();
            move_left();
            move_left();
            move_left();

      draw_fill();
      draw_fill();
      draw_fill();
      draw_fill();
            move_down();
            move_left();
            move_left();
            move_left();
            move_left();

      draw_fill();
            move_down();
            move_left();

      draw_fill();
      draw_fill();
      draw_fill();
      draw_fill();
      break;
// ####
//    #
// ####
//    #
// ####
case 3:
      draw_fill();
      draw_fill();
      draw_fill();
      draw_fill();
            move_down();
            move_left();

      draw_fill();
```

```
            move_down();
            move_left();
            move_left();
            move_left();
            move_left();

      draw_fill();
      draw_fill();
      draw_fill();
      draw_fill();
            move_down();
            move_left();

      draw_fill();
            move_down();
            move_left();
            move_left();
            move_left();
            move_left();

      draw_fill();
      draw_fill();
      draw_fill();
      draw_fill();
      break;
// #  #
// #  #
// ####
//    #
//    #
case 4:
      draw_fill();
            move_right();
            move_right();
      draw_fill();
            move_down();
            move_left();
            move_left();
            move_left();
            move_left();

      draw_fill();
            move_right();
            move_right();
      draw_fill();
            move_down();
            move_left();
            move_left();
            move_left();
            move_left();

      draw_fill();
      draw_fill();
      draw_fill();
      draw_fill();
            move_down();
            move_left();

      draw_fill();
            move_down();
            move_left();

      draw_fill();
      break;
// ####
// #
// ####
//    #
// ####
case 5:
      draw_fill();
      draw_fill();
      draw_fill();
      draw_fill();
            move_down();
            move_left();
            move_left();
```

```
            move_left();
            move_left();

      draw_fill();
            move_down();
            move_left();

      draw_fill();
      draw_fill();
      draw_fill();
      draw_fill();
            move_down();
            move_left();

      draw_fill();
            move_down();
            move_left();
            move_left();
            move_left();
            move_left();

      draw_fill();
      draw_fill();
      draw_fill();
      draw_fill();
      break;
// #
// #
// ####
// #  #
// ####
case 6:
      draw_fill();
            move_down();
            move_left();

      draw_fill();
            move_down();
            move_left();

      draw_fill();
      draw_fill();
      draw_fill();
      draw_fill();
            move_down();
            move_left();
            move_left();
            move_left();
            move_left();

      draw_fill();
            move_right();
            move_right();
      draw_fill();
            move_down();
            move_left();
            move_left();
            move_left();
            move_left();

      draw_fill();
      draw_fill();
      draw_fill();
      draw_fill();
      break;
// ####
//    #
//    #
//    #
//    #
case 7:
      draw_fill();
      draw_fill();
      draw_fill();
      draw_fill();
            move_down();
            move_left();
```

```
      draw_fill();
            move_down();
            move_left();

      draw_fill();
            move_down();
            move_left();

      draw_fill();
            move_down();
            move_left();

      draw_fill();
      break;
// ####
// #  #
// ####
// #  #
// ####
case 8:
      draw_fill();
      draw_fill();
      draw_fill();
      draw_fill();
            move_down();
            move_left();
            move_left();
            move_left();
            move_left();

      draw_fill();
            move_right();
            move_right();
      draw_fill();
            move_down();
            move_left();
            move_left();
            move_left();
            move_left();

      draw_fill();
      draw_fill();
      draw_fill();
      draw_fill();
            move_down();
            move_left();
            move_left();
            move_left();
            move_left();

      draw_fill();
            move_right();
            move_right();
      draw_fill();
            move_down();
            move_left();
            move_left();
            move_left();
            move_left();

      draw_fill();
      draw_fill();
      draw_fill();
      draw_fill();
      break;
// ####
// #  #
// ####
//    #
//    #
case 9:
      draw_fill();
      draw_fill();
      draw_fill();
      draw_fill();
            move_down();
```

```
                        move_left();
                        move_left();
                        move_left();
                        move_left();

                  draw_fill();
                        move_right();
                        move_right();
                  draw_fill();
                        move_down();
                        move_left();
                        move_left();
                        move_left();
                        move_left();

                  draw_fill();
                  draw_fill();
                  draw_fill();
                  draw_fill();
                        move_down();
                        move_left();

                  draw_fill();
                        move_down();
                        move_left();

                  draw_fill();
                  break;
      }
}
```