WWW.ROBOKITS.CO.IN
EASY TO USE, VERSATILE ROBOTICS KITS

QuickC IDE Library Reference for Rhino Robot Control Board [RKI-1550]

Users Manual

Robokits India
http://www.robokits.co.in
info@robokits.co.in

Rhino Board ATMega16 Resources Available

- **PA0, PA1, PA2, PA3, PA4, PA5, PA6, PA7**
- **PB0, PB1, PB2, PB3, PB4, PB5, PB6, PB7**
- **PC0, PC1, PC2, PC3, PC4, PC5, PC6, PC7**
- **PD0, PD1, PD2, PD3, PD4, PD5, PD6, PD7**
- **TIMER0**
- **TIMER1**
- **TIMER2**
- **UART**
- **I2C (Can be used multiple times)**
- **SPI (Can be used multiple times)**

LIBRARY init.c and init.h

(Included automatically. To make sure all things are initialized properly all programs must start from statement "INIT();" at the start of main)

Resources Used by this library: PC6,PC7,PD6

Functions:

- **INIT(); - Board Initialization (Must be used in the first line in main function)**
- **LED1ON(); - Turns LED1 ON**
- **LED1OFF(); - Turns LED1 OFF**
- **TOGGLELED1(); - Toggles the state of LED1 e.g. If the LED1 is on This statement will turn LED1 off**
- **LED2ON(); - Turns LED2 ON**
- **LED2OFF(); - Turns LED2 OFF**
- **TOGGLELED2(); - Toggles the state of LED2 e.g. If the LED2 is on This statement will turn LED2 off**
- **SWITCH1ON(); - Returns char 1 if the SWITCH1 is pressed and returns char 0 otherwise**
- **SWITCH1OFF(); - Returns char 0 if the SWITCH1 is pressed and returns char 1 otherwise**

LIBRARY delay.c and delay.h

Resources Used by this library: None

Functions:

- **DELAYS(unsigned int delay); - Perform a delay of "delay" seconds. The time duration will change in case there are any interrupt calls in between when this function is getting executed**
- **DELAYMS(unsigned int delay); - Perform a delay of "delay" miliseconds. The time duration will change in case there are any interrupt calls in between when this function is getting executed**
- **DELAYUS(unsigned int delay); - Perform a delay of "delay" microseconds. The time duration will change in case there are any interrupt calls in between when this function is getting executed**

LIBRARY lcd.c and lcd.h

Resources Used by this library: PB0,PB1,PB2,PB4,PB5,PB6,PB7

Functions:

- **LCD_INIT(char attribute); - - This function must be used once before using any other following functions else the functions may not work properly. Parameters can be set directly to LCD_DISP_OFF, LCD_DISP_ON, LCD_DISP_ON_BLINK, LCD_DISP_ON_CURSOR, LCD_DISP_ON_CURSOR_BLINK.**
- **LCD_CLRSCR(); - Clears LCD screen and moves cursor to position 0,0**
- **LCD_HOME_POS(); - Moves cursor to home posisiotn 0,0**
- **LCD_GOTOXY(char x,char y);- Moves cursor to char x on line y**
- **LCD_PUTC(char c); - Writes character c on LCD**
- **LCD_PRINT(char *s); - Writes String to LCD**

LIBRARY motor1.c and motor1.h

Resources Used by this library: PD4,PD5,PC2,PC3,PC4,PC5,PD2,PD3,PD7,PB3,TIMER1

Left Motor - Motor1

Right Motor - Motor2

Functions:

- **MOTORINIT(); - This function must be used once before using any other following functions else the functions may not work properly.**
- **LEFT(unsigned char speed); - Drives Robot in Left direction at given speed. Speed should be between 0 to 100.**
- **RIGHT(unsigned char speed); - Drives Robot in Right direction at given speed. Speed should be between 0 to 100.**
- **FORWARD(unsigned char speed); - Drives Robot in Forward direction at given speed. Speed should be between 0 to 100.**
- **BACKWARD(unsigned char speed); - Drives Robot in Backward direction at given speed. Speed should be between 0 to 100.**
- **LMF(unsigned char speed); - Drives Left Motor in forward direction at given speed. Speed should be between 0 to 100.**
- **LMB(unsigned char speed); - Drives Left Motor in backward direction at given speed. Speed should be between 0 to 100.**
- **RMF(unsigned char speed); - Drives Right Motor in forward direction at given speed. Speed should be between 0 to 100.**
- **RMB(unsigned char speed); - Drives Right Motor in backward direction at given speed. Speed should be between 0 to 100.**
- **STOP(); - All Motors stop (Break)**
- **LMS(); - Left motor stop**
- **RMS(); - Right motor stop**
- **M3F(); - Motor3 Backward at full speed.**
- **M3B(); - Motor3 Backward at full speed.**
- **M3S(); - Motor3 Stop**
- **M4F(); - Motor4 Backward at full speed.**
- **M4B(); - Motor4 Backward at full speed.**
- **M4S(); - Motor4 Stop**

LIBRARY motor2.c and motor2.h

Resources Used by this library: PD4,PD5,PC2,PC3,PC4,PC5,TIMER1

Left Motor - Motor1
Right Motor - Motor2

Functions:

- **MOTORINIT(); - This function must be used once before using any other following functions else the functions may not work properly.**
- **LEFT(unsigned char speed); - Drives Robot in Left direction at given speed. Speed should be between 0 to 100.**
- **RIGHT(unsigned char speed); - Drives Robot in Right direction at given speed. Speed should be between 0 to 100.**
- **FORWARD(unsigned char speed); - Drives Robot in Forward direction at given speed. Speed should be between 0 to 100.**
- **BACKWARD(unsigned char speed); - Drives Robot in Backward direction at given speed. Speed should be between 0 to 100.**
- **LMF(unsigned char speed); - Drives Left Motor in forward direction at given speed. Speed should be between 0 to 100.**
- **LMB(unsigned char speed); - Drives Left Motor in backward direction at given speed. Speed should be between 0 to 100.**
- **RMF(unsigned char speed); - Drives Right Motor in forward direction at given speed. Speed should be between 0 to 100.**
- **RMB(unsigned char speed); - Drives Right Motor in backward direction at given speed. Speed should be between 0 to 100.**
- **STOP(); - All Motors stop (Break)**
- **LMS(); - Left motor stop**
- **RMS(); - Right motor stop**

LIBRARY motor3.c and motor3.h

Resources Used by this library: PD4,PD5,PC2,PC3,PC4,PC5,PD2,PD3,TIMER1

Left Motor - Motor1
Right Motor - Motor2

Functions:

- **MOTORINIT(); - This function must be used once before using any other following functions else the functions may not work properly.**
- **LEFT(unsigned char speed); - Drives Robot in Left direction at given speed. Speed should be between 0 to 100.**
- **RIGHT(unsigned char speed); - Drives Robot in Right direction at given speed. Speed should be between 0 to 100.**
- **FORWARD(unsigned char speed); - Drives Robot in Forward direction at given speed. Speed should be between 0 to 100.**
- **BACKWARD(unsigned char speed); - Drives Robot in Backward direction at given speed. Speed should be between 0 to 100.**
- **LMF(unsigned char speed); - Drives Left Motor in forward direction at given speed. Speed should be between 0 to 100.**
- **LMB(unsigned char speed); - Drives Left Motor in backward direction at given speed. Speed should be between 0 to 100.**
- **RMF(unsigned char speed); - Drives Right Motor in forward direction at given speed. Speed should be between 0 to 100.**
- **RMB(unsigned char speed); - Drives Right Motor in backward direction at given speed. Speed should be between 0 to 100.**
- **STOP(); - All Motors stop (Break)**
- **LMS(); - Left motor stop**
- **RMS(); - Right motor stop**
- **M3F(); - Motor3 Backward at full speed.**
- **M3B(); - Motor3 Backward at full speed.**
- **M3S(); - Motor3 Stop**

LIBRARY motor4.c and motor4.h

Resources Used by this library: PD4,PD5,PC2,PC3,PC4,PC5

Left Motor - Motor1
Right Motor - Motor2

Functions:

- **MOTORINIT(); - This function must be used once before using any other following functions else the functions may not work properly.**
- **LEFT(); - Drives Robot in Left direction at full speed.**
- **RIGHT(); - Drives Robot in Right direction at full speed.**
- **FORWARD(); - Drives Robot in Forward direction at full speed.**
- **BACKWARD(); - Drives Robot in Backward direction at full speed.**
- **LMF(); - Drives Left Motor in forward direction at full speed.**
- **LMB(); - Drives Left Motor in backward direction at full speed.**
- **RMF(); - Drives Right Motor in forward direction at full speed.**
- **RMB(); - Drives Right Motor in backward direction at full speed.**
- **STOP(); - All Motors stop (Break)**
- **LMS(); - Left motor stop**
- **RMS(); - Right motor stop**

LIBRARY uart.c and uart.h

Resources Used by this library: UART,PD0,PD1, 64 bytes of RAM

Functions:

- **UART_INIT(unsigned int baudrate); - This function must be used once before using any other following functions else the functions may not work properly.**
- **UART_GETCHAR(); - Returns Integer data from ring buffer of 32 bytes which is automatically filled with incoming characters**
- **- if (!(c & UART_NO_DATA))**
  Data available in buffer
- **UART_PUTCHAR(unsigned char data); - Puts data into transmit buffer and it is transmitted automatically at initialized baudrate**
- **UART_PRINT(const char *s); - Puts whole string into transmit buffer and it is transmitted automatically at initialized baudrate**

LIBRARY psx.c and psx.h

Resources Used by this library: PB4,PB5,PB6,PB7

Functions:

- **PSXINIT(); - This function must be used once before using any other following functions or variables else the functions may not work properly.**
- **PSXPOLL(); - Gets input from PS2 remote and updates it to following variables. This variables can be used directly as normal variables. This function can return wrong data if used again before 9ms.**

Variables:

- **PSX_L1 - PSX Button L1. 1 when pressed else 0.**
- **PSX_R1 - PSX Button R1. 1 when pressed else 0.**
- **PSX_L2 - PSX Button L2. 1 when pressed else 0.**
- **PSX_R2 - PSX Button R2. 1 when pressed else 0.**
- **PSX_SL - PSX Button Select. 1 when pressed else 0.**
- **PSX_ST - PSX Button Start. 1 when pressed else 0.**
- **PSX_JL - PSX Button Left Jostick press. 1 when pressed else 0.**
- **PSX_JR - PSX Button Right Joystick Press. 1 when pressed else 0.**
- **PSX_UP - PSX Button Up. 1 when pressed else 0.**
- **PSX_DN - PSX Button Down. 1 when pressed else 0.**
- **PSX_LF - PSX Button Left. 1 when pressed else 0.**
- **PSX_RG - PSX Button Right. 1 when pressed else 0.**
- **PSX_A - PSX Button A. 1 when pressed else 0.**
- **PSX_O - PSX Button O. 1 when pressed else 0.**
- **PSX_X - PSX Button X. 1 when pressed else 0.**
- **PSX_D - PSX Button D. 1 when pressed else 0.**
- **LJOYX - Left Joystick X value. Value between 0 to 255 when joystick is untouched it will be 128. Updated only when joystick is in analog mode.**
- **LJOYY - Left Joystick Y Value. Value between 0 to 255 when joystick is untouched it will be 128. Updated only when joystick is in analog mode.**
- **RJOYX - Right Joystick X Value. Value between 0 to 255 when joystick is untouched it will be 128. Updated only when joystick is in analog mode.**
- **RJOYY - Right Joystick Y Value. Value between 0 to 255 when joystick is untouched it will be 128. Updated only when joystick is in analog mode.**

LIBRARY remote.c and remote.h

Resources Used by this library: PA7,TIMER0

Functions:

- **GETRC5(); - Waits 80ms for a RC5 signal and returns RC5 code or 255 if no code is received within 80ms time.**

LIBRARY servo1.c and servo1.h

Resources Used by this library: TIMER0,PA0,PA1,PA2,PA3,PA4,PA5,PA6,PA7

Functions:

- **SERVOINIT(); - This function must be used once before using any other following functions or variables else the functions may not work properly.**
- **SERVOPOSITION(unsigned char servo_num, unsigned char position,unsigned char speed); - Sets servo_num Servo to position at given speed. Where Servo _num from 0 to 7, Position 0 to 255 and speed 0 to 255. If speed=0 speed is disabled and if speed=255 speed is max and speed=1 then speed is minimum.**

Variables:

- **SERVO[servo_num] - Set position of servo servo_num.**
- **SERVOSPEED[servo_num] - Set speed of servo servo_num.**
- **SERVOMIN[servo_num] - Set minimum position of servo servo_num.**
- **SERVOMAX[servo_num] - Set maximum position of servo servo_num.**

LIBRARY servo2.c and servo2.h

Resources Used by this library: TIMER0,PA0,PA1,PA2,PA3,PA4,PA5

Functions:

- **SERVOINIT(); - This function must be used once before using any other following functions or variables else the functions may not work properly.**
- **SERVOPOSITION(unsigned char servo_num, unsigned char position,unsigned char speed); - Sets servo_num Servo to position at given speed. Where Servo _num from 0 to 5, Position 0 to 255 and speed 0 to 255. If speed=0 speed is disabled and if speed=255 speed is max and speed=1 then speed is minimum.**

Variables:

- **SERVO[servo_num] - Set position of servo servo_num.**
- **SERVOSPEED[servo_num] - Set speed of servo servo_num.**
- **SERVOMIN[servo_num] - Set minimum position of servo servo_num.**
- **SERVOMAX[servo_num] - Set maximum position of servo servo_num.**

LIBRARY servo3.c and servo3.h

Resources Used by this library: TIMER0,PA0,PA1,PA2,PA3

Functions:

- **SERVOINIT(); - This function must be used once before using any other following functions or variables else the functions may not work properly.**
- **SERVOPOSITION(unsigned char servo_num, unsigned char position,unsigned char speed); - Sets servo_num Servo to position at given speed. Where Servo _num from 0 to 3, Position 0 to 255 and speed 0 to 255. If speed=0 speed is disabled and if speed=255 speed is max and speed=1 then speed is minimum.**

Variables:

- **SERVO[servo_num] - Set position of servo servo_num.**
- **SERVOSPEED[servo_num] - Set speed of servo servo_num.**
- **SERVOMIN[servo_num] - Set minimum position of servo servo_num.**
- **SERVOMAX[servo_num] - Set maximum position of servo servo_num.**

LIBRARY servo4.c and servo4.h

Resources Used by this library: TIMER0,PA0,PA1

Functions:

- **SERVOINIT(); - This function must be used once before using any other following functions or variables else the functions may not work properly.**
- **SERVOPOSITION(unsigned char servo_num, unsigned char position,unsigned char speed); - Sets servo_num Servo to position at given speed. Where Servo _num from 0 and 1, Position 0 to 255 and speed 0 to 255. If speed=0 speed is disabled and if speed=255 speed is max and speed=1 then speed is minimum.**

Variables:

- **SERVO[servo_num] - Set position of servo servo_num.**
- **SERVOSPEED[servo_num] - Set speed of servo servo_num.**
- **SERVOMIN[servo_num] - Set minimum position of servo servo_num.**
- **SERVOMAX[servo_num] - Set maximum position of servo servo_num.**

LIBRARY stepper1.c and stepper1.h

(Must include delay.c and delay.h files)
Resources Used by this library: PD4,PD5,PC2,PC3,PC4,PC5,PD2,PD3,PD7,PB3

Functions:

- **STEPINIT(unsigned int stp_per_rev); - This function must be used once before using any other following functions or variables else the functions may not work properly.**
- **STEP1PLUS(void); - Basic function to move 1 step in clockwise direction of stepper1**
- **STEP1MINUS(void); - Basic function to move 1 step in anti clockwise direction of stepper1**
- **STEP2PLUS(void); - Basic function to move 1 step in clockwise direction of stepper2**
- **STEP2MINUS(void); - Basic function to move 1 step in anti clockwise direction of stepper2**
- **STEP1POS(unsigned int steps,unsigned int dir,unsigned int rpm); - Move stepper1 at RPM and number of steps provided in given direction (If dir=0 clockwise, If DIR=1 Anti clockwise)**
- **STEP2POS(unsigned int steps,unsigned int dir,unsigned int rpm); - Move stepper2 at RPM and number of steps provided in given direction (If dir=0 clockwise, If DIR=1 Anti clockwise)**

- **STEPLEFTPOS(unsigned int steps,unsigned int rpm); - Move both steppers at RPM and number of steps provided in given direction**
- **STEPRIGHTPOS(unsigned int steps,unsigned int rpm); - Move both steppers at RPM and number of steps provided in given direction**
- **STEPFORWARDPOS(unsigned int steps,unsigned int rpm); - Move both steppers at RPM and number of steps provided in given direction**
- **STEPBACKWARDPOS(unsigned int steps,unsigned int rpm); - Move both steppers at RPM and number of steps provided in given direction (If dir=0 clockwise, If DIR=1 Anti clockwise)**

LIBRARY stepper2.c and stepper2.h

(Must include delay.c and delay.h files)
Resources Used by this library: PD4,PD5,PC2,PC3,PC4,PC5

Functions:

- **STEPINIT(unsigned int stp_per_rev); - This function must be used once before using any other following functions or variables else the functions may not work properly.**
- **STEP1PLUS(void); - Basic function to move 1 step in clockwise direction of stepper1**
- **STEP1MINUS(void); - Basic function to move 1 step in anti clockwise direction of stepper1**
- **STEP1POS(unsigned int steps,unsigned int dir,unsigned int rpm); - Move stepper1 at RPM and number of steps provided in given direction (If dir=0 clockwise, If DIR=1 Anti clockwise)**

LIBRARY io.c and io.h

Resources Used by this library: I/O pins selected for use (Refer IO pin porting for RHINO boards) Below

***Note: If previously used pins are used again the opration for other library functions can be affected.**

Functions:

- **DIGITALWRITE(unsigned char pinno, unsigned char value); - Where pinno refers to pin ported numbers(Refer IO pin porting for RHINO boards). Writes value to the pinno. If value 0 then pin will be in low mode else it will be in high mode.**
- **DIGITALREAD(unsigned char pinno); - Where pinno refers to pin ported numbers(Refer IO pin porting for RHINO boards). Reads value from the pinno and returns 1 if the pin is in high state else returns 0.**
- **PINMODE(unsigned char pinno,unsigned char value); - Where pinno refers to pin ported numbers(Refer IO pin porting for RHINO boards). Makes pinno as input pin if value is 0 else output pin.**

LIBRARY adc.c and adc.h

Resources Used by this library: ADC & I/O pins selected for use

Functions:

- **ADCINIT(void); - This function must be used once before using any other following functions or variables else the functions may not work properly.**
- **GETADC(unsigned char ch); - Where ch is ADC channel. E.g. ch=0 selects ADC0(PA0). Returns 10 bit ADC value propotional to input voltage with refrence to AREF pin voltage.**

LIBRARY sharp.c and sharp.h

Resources Used by this library: ADC & I/O pins selected for use

***NOTE: ADC library must be included and ADCINIT() function must be used atleast once before using any of following function.**

Functions:

- **GETRANGE_GP2D120(unsigned char ch); - Where ch is ADC channel. E.g. ch=0 selects ADC0(PA0). Returns range in cm for sharp range finder GP2D120. Will return 0 if out of range.**
- **GETRANGE_GP2D12(unsigned char ch); - Where ch is ADC channel. E.g. ch=0 selects ADC0(PA0). Returns range in cm for sharp range finder GP2D12 and its replacement part GP2Y0A21YK0F. Will return 0 if out of range.**
- **GETRANGE_GP2Y0A02YK0F(unsigned char ch); - Where ch is ADC channel. E.g. ch=0 selects ADC0(PA0). Returns range in cm for sharp range finder GP2Y0A02YK0F. Will return 0 if out of range.**

LIBRARY ultrasonic.c and ultrasonic.h

Resources Used by this library: TIMER0 & I/O pins selected for use

Functions:

- **GETRANGE(unsigned char TRIG_PIN, unsigned char ECHO_PIN); - Where TRIG_PIN and ECHO_PINS must be selected by user(Refer IO pin porting for RHINO boards). Returns range in cm for Robokits Ultrasonic rangefinder RKI-1540. Will return 0 if out of range.**

LIBRARY i2c.c and i2c.h

Resources Used by this library: I2C, PC0,PC1

Functions:

- **I2C_INIT(void); - This function must be used once before using any other following functions or variables else the functions may not work properly.**
- **I2C_STOP(void); - I2C Stop Condition**
- **unsigned char I2C_START(unsigned char addr); - I2C Start with slave address + (READ or WRITE)**
- **unsigned char I2C_REP_START(unsigned char addr); - I2C Repeated start with slave address + (READ or WRITE)**
- **I2C_START_WAIT(unsigned char addr); - I2C Start with slave address + (READ or WRITE). Waits untill acknoledgement is received**
- **unsigned char I2C_WRITE(unsigned char data); - I2C Write DATA 8bit to slave**
- **unsigned char I2C_READACK(void); - I2C receive data and send acknoledgement.(Used when reading multiple data bytes and there are more bytes to be read from slave)**
- **unsigned char I2C_READNAK(void); - I2C receive data without acknoledgement.(Used when reading last data byte. Generally followed by I2C_STOP() function)**

LIBRARY i2c_slave.c and i2c_slave.h

Resources Used by this library: I2C, PC0,PC1

Functions:

- **I2C_SLAVE_INIT(unsigned char addr); - Initialize slave address with addr. ADDR must be a even number.**
- **unsigned char I2C_TRANSRECEIVER_BUSY(void); - Check whether I2C is busy or not.**
- **I2C_START_TRANSRECEIVER_WITH_DATA(unsigned char *, unsigned char len); - Send string pointer data through I2C of length of len bytes.**
- **I2C_START_TRANSRECEIVER(void); - Enable I2C for transmit and receive.**
- **unsigned char I2C_GET_DATA_FROM_TRANSRECEIVER(unsigned char *, unsigned char len); - Wait for data input and store it to given string of length len bytes.**

Variables:

- **I2C_SLAVE_LAST_TX_OK - Will have data of whether last transmit/receive completed correctly or not.**
- **I2C_SLAVE_RX_DATA_BUF - Will have data of whether Receive data biffer is filled.**

LIBRARY i2c_servo.c and i2c_servo.h

Resources Used by this library: I2C, PC0,PC1

***Note: This library is to be used with "I2C 18 servo control board" from robokits part number RKI-1551. You can use two servo boards in total to run upto 36 servos.**

Functions:

- **I2C_SERVOSET(unsigned char servo_num,unsigned int servo_pos); - Set servo_num servo position to servo_pos value. servo_num value can be between 0 to 35 and servo_pos value should be between 0 to 4000.**
- **I2C_SERVOMIN(unsigned char servo_num,unsigned int servo_pos); - Set servo_num servo minimum value to servo_pos value. servo_num value can be between 0 to 35 and servo_pos value should be between 0 to 4000. Servo max value should not be less then servo min value.**
- **I2C_SERVOMAX(unsigned char servo_num,unsigned int servo_pos); - Set servo_num servo maximum value to servo_pos value. servo_num value can be between 0 to 35 and servo_pos value should be between 0 to 4000. Servo min value should not be more then servo max value.**
- **I2C_SERVONUTRALSET(unsigned char servo_num,unsigned int servo_pos); - Set servo_num servo nutral to servo_pos value. servo_num value can be between 0 to 35 and servo_pos value should be between 0 to 4000.**
- **I2C_SERVOSPEED(unsigned char servo_num,unsigned char value); - Set servo_num servo speed to value. servo_num value can be between 0 to 35 and value should be 0 for speed disable or between 1 to 255 for speed enable.**
- **I2C_SERVOOFFSET(unsigned char servo_num,int value); - Set servo_num servo offset to value. servo_num value can be between 0 to 35 and value should be between 0 to 4000. (Normal center position is 2000. If needed at 3000 the offset value should be 1000 and if needed at 1000 then offset value should be 3000).**
- **I2C_SERVOREVERSE(unsigned char servo_num,unsigned char servo_dir); - Set servo direction. Servo_num between 0 to 35 and servo_dir = 0 normal direction. servo_dir=1 reverse direction.**

- **unsigned char I2C_SERVOEND(void); - Check for all actions complete for 18 servos. Can be only used if servo speed set functions are used.**
- **unsigned char I2C_SERVOEND36(void); - check for all actions comlete for 36 servos. Can be only used if servo speed set functions are used.**
- **unsigned int I2C_SERVOGET(unsigned char servo_num); Get current servo position for servo_num.**

Refrence to IO pin porting for RHINO boards

PA0 - 0
PA1 - 1
PA2 - 2
PA3 - 3
PA4 - 4
PA5 - 5
PA6 - 6
PA7 - 7

PB0 - 8
PB1 - 9
PB2 - 10
PB3 - 11
PB4 - 12
PB5 - 13
PB6 - 14
PB7 - 15

PC0 - 16
PC1 - 17
PC2 - 18
PC3 - 19
PC4 - 20
PC5 - 21
PC6 - 22
PC7 - 23

PD0 - 24
PD1 - 25
PD2 - 26
PD3 - 27
PD4 - 28
PD5 - 29
PD6 - 30
PD7 - 31