Programming the joystick control for Zigbee & Executing

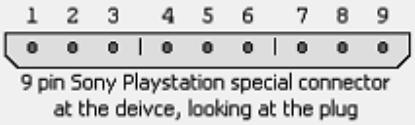
This product is designed to control a variety of robots through the joystick made for Play Station 2. Let's learn the structure of Play station 2, how Play Station2 operates, and moreover, the way robots move according to the signal which is output.

The structure of Play Station 2

Now, Let's learn the joystick (Play Station2) to control the robots. Play Station has been used not only as the entertainment controller, but as the controller for the directions of robots. When it is used for the controller of robots, you can control by putting the CPU to the external port of Play Station2. At this time, you should transmit the serial to the RF wireless communication module in accordance with the controlling signal. Besides, when wirelessly transmitted, it is controlled through zigbee. The pin numbers of the zigbee module are available, Please refer to them. The next picture is about the outward form of Play Station2. You can see the little controlling board made in the bottom and recognize it is expanded. There are the Pin number and its explanation concerning the external expansion port of Play Station2. Please make the best use of them. In the first place, please couple two 1.2 voltage dry cells together, and then, set the direction of the joystick to the side of the two toggle switch(red). [Illustration] Play Station2 & the zigbee controller. [List or Chart] The port number of Play Station2 & the function of Pin

[Fig] Play station2 and Zigbee Module

[Table] Play station2 Port Pin and Description



Pin	Name	Description
1	DATA	Data
2	CMD	Command
3	N/C (9V)	Not connected (newer dual-shock controllers use it as power for the vibration mechanism)
4	GND	Ground
5	VCC	Vcc
6	ATT	ATT select
7	CLK	Clock
8	N/C	Not connected
9	ACK	Acknowledge

Key Code Value

Let's refer to the output code value of the key of Play Station2 (Joystick). When each key is pressed , its code values are output in the serial. The output values are transmitted in a

wireless way through the zigbee module connected to the joystick. The transmitted values go to the reception Zigbee module and appear as the serial. Eventually, the output signals are received to many sorts of CPUs, so that a great number of movements that you want are created.

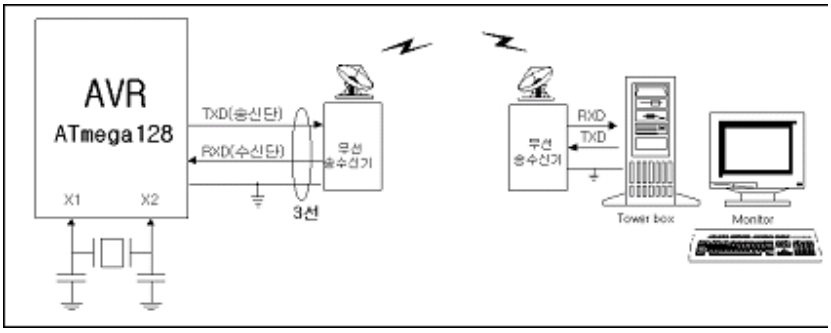
[Table] The table of the joystict value

Key Type	Key Code Value		Key Type	Key Code Value	
▲	0x01		▽	0x05	
▶	0x02		O	0x06	
▼	0x03		X	0x07	
◀	0x04		E	0x08	
L1	0x0b		R1	0x0c	
L2	0x09		R2	0x0a	
◤	0x0d		▽ + E	0x0d	
◣	0x0e		E + X	0x0e	
◢	0x0f		X + O	0x0f	
◥ (	0x10		O + ▽	0x10	
Left Joystick			Right Joystick		
Forward	X		Forward	0x80	
Backward	X		Backward	0x81	
Right	X		Right	0x82	
Left	X		Left	0x83	
Forward/Right	X		Forward/Right	0x84	
Forward/Left	X		Forward/Left	0x85	
Backward/Left	X		Backward/Left	0x86	
Backward/Right	X		Backward/Right	0x87	

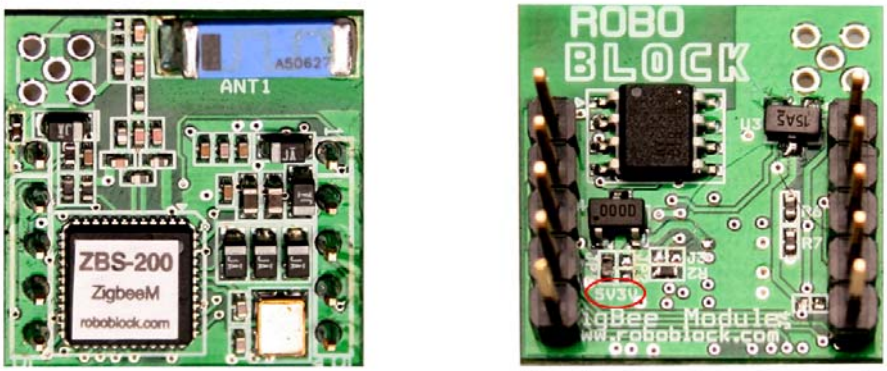
Receive Part

Please connect RXD1&TXD1 with the zigbee module as RXD1&TXD1 in the two serial port of ATmega128 are used.

.



[Fig] A series communication by wireless module.



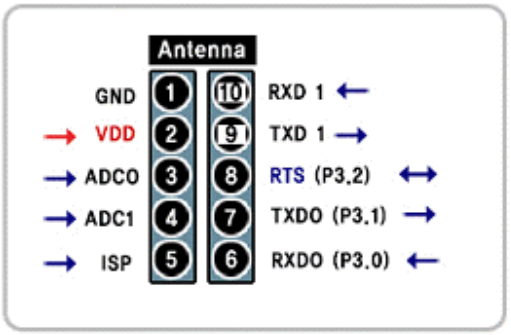
[Fig] the outward form of the zigbee module

The zigbee module comprises ten Pins in total, and the function of each Pin runs as follows. #1 is Gnd, and #2 is the power port that should be powered by 3.3V-5V. #8 TXD& #9 RXD ports should be connected to RXD& TXD ports of ATmega128 each other in a crossed way, and it is okay that the rest are not coupled as well.

Notice： Must use ZBS-100 Module for Play station2

[Table] Zigbee Pin Description

- RS232C 케이블링 대처
- 무선 POS 시스템
- 산업용 무선 기기 제어 및 모니터링
- 공장 자동화 무선 통신 응용
- 교통감시 시스템
- 자동차 진단 시스템
- PLC 프로그래밍
- 무선 물류 시스템



Pin No.	Pin Name	Direction	Description	Signal Level
1	GND		Power Ground	Ground
2	VDD	Input	DC input(3V or 5V)	Power
3	ADC0	Input	Reserved	N.C
4	ADC1	nput	Reserved	N.C
5	ISP	N.C	Not Connected	N.C
6	RXD0	N.C	Reserved	N.C
7	TXD0	Output	Reserved	N.C
8	P3.2(I/O Pin)	Input/Output	Reserved	N.C
9	TXD1(P3.1)	Output	UART1 data output	TTL
10	RXD1(P3.0)	Input	UART1 data output	TTL

[[EX 1]] Let's make the legs of robot move according to the joystick. The values shown in the key table are received to the RXD1 port of ATmega128, and whenever they are received, please put in the relevant and appropriate functions by means of interrupt.

▲ --- upstep(0x01) / ▼ --- downstep(0x03)

☞ Program Example

```
#include <mega128.h>
#include <delay.h>
#define RXB8 1
#define TXB8 0
```

```
#define UPE 2
#define OVR 3
#define FE 4
#define UDRE 5
#define RXC 7

#define FRAMING_ERROR (1<<FE)
#define PARITY_ERROR (1<<UPE)
#define DATA_OVERRUN (1<<OVR)
#define DATA_REGISTER_EMPTY (1<<UDRE)
#define RX_COMPLETE (1<<RXC)


 char a;                  // variable define
// USART1 Receiver buffer
#define RX_BUFFER_SIZE1 8
char rx_buffer1[RX_BUFFER_SIZE1];
unsigned char rx_wr_index1,rx_rd_index1,rx_counter1;
// This flag is set on USART1 Receiver buffer overflow
bit rx_buffer_overflow1;

// USART1 Receiver interrupt service routine
interrupt [USART1_RXC] void usart1_rx_isr(void)
{
char status,data;
status=UCSR1A;
data=UDR1;
a = data;
if ((status & (FRAMING_ERROR | PARITY_ERROR | DATA_OVERRUN))==0)
    {
    rx_buffer1[rx_wr_index1]=data;
    if (++rx_wr_index1 == RX_BUFFER_SIZE1) rx_wr_index1=0;
    if (++rx_counter1 == RX_BUFFER_SIZE1)
        {
        rx_counter1=0;
        rx_buffer_overflow1=1;
        };
    };
}

// Get a character from the USART1 Receiver buffer
#pragma used+
```

```
char getchar1(void)
{
char data;
while (rx_counter1==0);
data=rx_buffer1[rx_rd_index1];
if (++rx_rd_index1 == RX_BUFFER_SIZE1) rx_rd_index1=0;
#asm("cli")
--rx_counter1;
#asm("sei")
return data;
}
#pragma used-
// Write a character to the USART1 Transmitter
#pragma used+
void putchar1(char c)
{
while ((UCSR1A & DATA_REGISTER_EMPTY)==0);
UDR1=c;
}
#pragma used-
void initstep()
{
        int i;
         for(i=0;i<40;i++){
         init
         }

}

void upstep()
{
        int i;
         for(i=0;i<40;i++){


         }

}

void downstep()
{
        int i;
```

```
        for(i=0;i<40;i++){

        }

}


// Declare your global variables here

void main(void)
{
// Declare your local variables here

// Port B initialization
// Func7=Out Func6=Out Func5=Out Func4=Out Func3=Out Func2=Out Func1=Out Func0=Out
// State7=0 State6=0 State5=0 State4=0 State3=0 State2=0 State1=0 State0=0
PORTB=0x00;
DDRB=0xFF;

// Port C initialization
// Func7=Out Func6=Out Func5=Out Func4=Out Func3=Out Func2=Out Func1=Out Func0=Out
// State7=0 State6=0 State5=0 State4=0 State3=0 State2=0 State1=0 State0=0
PORTC=0x00;
DDRC=0xFF;

// Port D initialization
// Func7=Out Func6=Out Func5=Out Func4=Out Func3=In Func2=In Func1=In Func0=In
// State7=0 State6=0 State5=0 State4=0 State3=T State2=T State1=T State0=T
PORTD=0x00;
DDRD=0xF0;

// Port E initialization
// Func7=In Func6=In Func5=In Func4=In Func3=Out Func2=Out Func1=In Func0=In
// State7=T State6=T State5=T State4=T State3=0 State2=0 State1=T State0=T
PORTE=0x00;
DDRE=0x0C;


// USART1 initialization
// Communication Parameters: 8 Data, 1 Stop, No Parity
// USART1 Receiver: On
// USART1 Transmitter: On
```

```
// USART1 Mode: Asynchronous
// USART1 Baud rate: 9600
UCSR1A=0x00;
UCSR1B=0x98;
UCSR1C=0x06;
UBRR1H=0x00;
UBRR1L=0x67;


// Global enable interrupts
#asm("sei")

while (1)
      {
      // Place your code here

       switch(a){

       case 0x01: upstep(); break;     // ▲   key
       case 0x03: downstep(); break;   // ▼ key
       default: initstep(); break;        // init

      };
}
```


Information Roboblock

address: 3,4 floors 8-1, Mullae-Dong 4-ga, Yongdungpo-Gu, Seoul, South Korea, 150-094

Tel: +82-2-2679-8556

Fax: +82-2-2679-8557

Homepage: http://www.roboblock.com

E-mail : robotoz@hotmail.com