ICS converter board

ICS library

for Arduino

Second Edition

Kondo Kagaku Co., Ltd.

The second edition January 19, 2018

table of contents

INTRODUCTION

This time is your purchase the ICS converter board, Thank you very much. Before using this product, so we will carefully read this manual, you

I wish.

The procedure for this manual for controlling the NEC ICS equipment from Arudino system microcomputer board with ICS conversion substrate, libraries, sample programs

I will explain.

The contents may without notice content is changed. , Thank you to decide to get use on your understanding.

Precautions

• Do not wet the product, please do not use in high humidity locations.

This product is for each terminal on a substrate is exposed, there is a risk of short-circuit. Or cause a short-circuit

Please note that there is a risk of heat / fire to an erroneous connection of the terminals. Please do not use in the state in contact with the metal.

- About use outside Japan of this product we can not support.

- cables will be inserted and withdrawn by grasping the plug portion, when inserting, please do not mistake the direction.

- in regard to problems caused by soldering failure or the like by the customer will be covered by the warranty.

Or abnormal Please do as soon as you feel the (heat generation, damage, abnormal odor, etc.) turn off the battery or power supply.

- Immediately stop use if you encounter problems, please consult to our service department.

Disclaimer

In this manual have indicated the corresponding situation in 2018 of January.

With regard to the contents of this manual are subject to change without notice.

Contact Us

Inquiries about this product and accessories, please contact our Support Center.

Yubinbango116-0014 Arakawa-ku, Tokyo Higashinippori 4-17-7

Kondo Kagaku Co., Ltd. Service Department TEL 03-3807-7648 (service Direct)

Excluding weekends and public holidays 9:00 to 12:00 1:00 p.m. to 5:00 p.m.

Announcements and Updater, etc. for the products will be posted on our Web site http://www.kondo-robot.com. Contact with E-mail

With regard to alignment, available at support @ kondo-robot.com, but it might get time to answer.

※ details such as inquiries Arduino how to use and program, so you may not be able to answer, please understand in advance.

Things to prepare

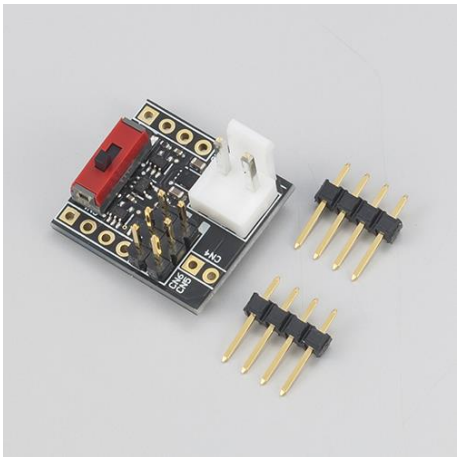
ICS converter board

This converter board for connecting our ICS device to the serial terminal (UART) of commercially available microcomputer board. Tx, to provide a circuit such as communication lines and power supply, such as Rx

That effort is save you, you can easily enables control of the ICS devices by simply connecting.

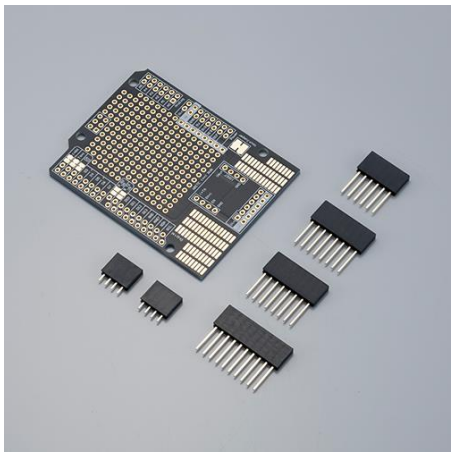
In addition, more and a dedicated shield board (sold separately), it is possible to easily connect eliminates also wiring to the Arduino UNO.

※ This product is a must have soldering work of the header pins.



ICS converter board

No.03121 ¥ 1,800 (excluding tax)



KSB shield 2

No.03149 ¥ 1,200 (excluding tax)

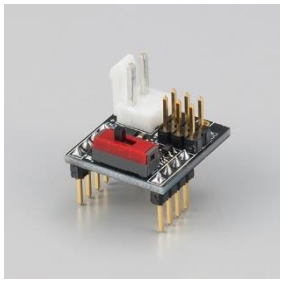
Configuration of the product

ICS And Equipment Arduino Between the microcomputer board, etc. ICS By relaying the converter board, you can make easy communication without having to own the circuit.

ICS devices

KRS servo, such as KRR-5FH

⇔



ICS converter board

⇔

Arduino

About Arduino

Arduino is an open source hardware, microcomputer board which corresponds has been general sale.

This product is using the serial terminal of Arduino (UART, HardwareSerial), to communicate with the ICS equipment. In addition, small, serial terminals, such as Arduino UNO

In gastric board, communication SoftwareSerial using the digital I / O terminals is also possible.

HardwareSerial and SoftwareSerial

It called the HardwareSerial to communicate using a serial terminal (UART), which is implemented in the microcomputer board such as the Arduino. Normally

While it communicates to the stable to use HardwareSerial, or has been used serial terminal to another application by the microcomputer, there is no implementation itself

If there is a. In that case, use as serial terminal of the digital I / O terminal on the software. This method is called the SoftwareSerial.

SoftwareSerial is not a function of the original board have, there is a possibility that interrupted the communication may be unstable. As much as possible HardwareSerial

It is recommended that you use.

Arduino-compatible devices

Compatible device operation check at our company is made will be as follows.

- HardwareSerial

Arduino Uno, Arduino Mega, Arduino M0, Arduino M0 Pro, Arduino mini

- SoftwareSerial

Arduino UNO, Arduino Mega, Arduino mini

※ different digital I / O pins that can be used depending on the type of Arduino. Please use the corresponding terminal of SoftwareSerial

※ HardwareSerial, SoftwareSerial both Arduino Due is not supported

Development environment

Development environment, use the Arduino IDE. Please use to download from the official website at the address given below.

https://www.Arduino.cc/en/Main/Software

Correspondence, etc. of the PC, please check at the official web site.

When creating this manual, version of the Arduino IDE, which has been confirmed by us is 1.8.3. (2018 January)

ICS devices

ICS3.5 and 3.6 Please prepare the devices that support.

Our KRS Series of servo motors, wireless controller (KRC-5FH / KRR-5FH) There can be used.

ICS The

ICS The " Interactive Communication System Stands for ", is Kondo science proprietary data communication standard. conventionally PWM In had sent a directive of the angle to the servo

But for more control over the characteristics of change or expansion, it has changed the standard serial communications.

Compatible devices (2018 Year 1 Month now)

- **HV (9 ~ 12V) Servo motor use**

KRS-6003RHV ICS	KRS-6003R2HV ICS		
KRS-4034HV ICS	KRS-4033HV ICS	KRS-4032HV ICS	KRS-4031HV ICS
KRS-2552RHV ICS	KRS-2572HV ICS	KRS-2542HV ICS	

- **LV (6.0 ~ 7.4V) Servo motor use**

KRS-3304ICS	KRS-3302 ICS	KRS-3301 ICS	KRS-3204 ICS

- **Wireless controller (6.0 ~ 12V)**

KRR-5FH

ICS It is, of the individual to each device ID Since the number has been assigned, daisy chain (multi-drop) connect the device to each other can be connected is. De

If wiring daisy chain connection, so you can connect multiple devices from a single terminal, by controlling the number of devices to minimize the wiring

I can. Also, acquisition of the current value of the setting changes and servo parameters (ICS3.6 Only) various functions such as are included. ICS3.5 and 3.6 To specifications

about, Software manual Please refer to the.

Since the servo can consume an instantaneous large current, KRS Daisy chain If you use a servo 1 For a column 8 It should be connected to the order of number in goal

There. If you want to connect more, please branches at the root separately preparing a hub board connection.

If you want to connect, you must be aware of the servo power supply voltage. LV The servo(6 ~ 7.4V Correspondence) and HV The servo(9 ~ 12V Can not coexist power in correspondence) is

Please note.

Dual USB adapter HS

Is a USB adapter for connecting the ICS device and the PC. Connection RCB-4 a (HV / mini) to PC by switching the mode

It is also possible to. By utilizing this and servo manager ID number and the communication speed of the ICS equipment, set the various parameters

Rukoto you can. In the library, so you specify a target device in a separate ID number that has been set in each of the servo, ICS equipment

Please prepare this USB adapter if you want to use.

"ICS3.5 Manager software R1.0.0.3" download page

Power supply

Power supply, the available ICS Voltage corresponding to the equipment, please prepare the current capacity. Since the servo can consume an instantaneous large current, rechargeable back

Stabilized power source capacitor or using a terry a margin or, AC Please prepare the adapter.

In our company, for the robot Battery Because we sell, there also please consider. In addition, since there is also a charger corresponding to each of the battery, in detail

Please check our web site.

HV (9 ~ 12V) corresponding



Battery (lithium ferrite type): ROBO power cell F3-850 / 1450/2100 (Li-Fe) (3 cell /9.9V)

Charger: BX-20L

AC adapter (12V5A)

※ AC adapter (12V5A) is the amount of a degree that can be walking KHR-3HV equipped with 17 pieces of KRS-2552RHV.

Please refer to us as a measure of the servo to be available.

LV (6V ~ 7.4V) corresponding


Battery (lithium ferrite type): ROBO power cell F2-850 / 1450 (Li-Fe) (2 cell /6.6V)

Charger: BX-31LF / BX-20L **(Both lithium ferrite only)**

AC adapter (6V2A)

※ AC adapter (6V2A) is the amount of the extent to which KRS-3301 KXR-L2, which was equipped with 16 ICS can walk.

Please refer to us as a measure of the servo to be available.

Battery (nickel-metal hydride type): ROBO power cell E type 6N-800mAh (Ni-MH)

Charger: BX-32MH (Nickel-metal hydride only)

【Please be careful】

Battery that is handled by our company, Li-Fe (lithium ferrite (Lihue)) There are two kinds of type and Ni-MH (nickel-metal hydride) type. Li-Fe type

Will strongly to a momentary power fluctuation, but it must be done thoroughly power management. The worst case fuming, because there is a possibility of ignition, be careful in handling

Is required. If you are using a Li-Fe type, please have a look at the following links.

- Support Information "Li-Fe battery of benefits and precautions."

If in the LV servo not so much to use the power, it is recommended Ni-MH type of battery.

※ battery is different from the charger to the corresponding by the type. Please use a combination of always corresponding battery and charger.

ICS Library for Arduino

It is a library and a sample program for moving the ICS equipment from Arduino.

For more information so that we can later, Please refer to there.

Connection Methods

Method of connecting the Arduino

If you want to connect the ICS converter board to Arduino, and connect the terminals as shown in the figure below.

Arduino Uno is, to communicate with the PC using the Serial (HardwareSerial) terminal to send and receive writing and data of the program. Also, the ICS equipment

Use the Serial (HardwareSerial) terminal as. Therefore, when connecting the ICS converter board to communicate with the servo Serial (HardwareSerial) terminal flop

Information between the PC at the time of the writing of the program might become mixed in with information between the ICS equipment. The ICS converter board to solve this, the PC

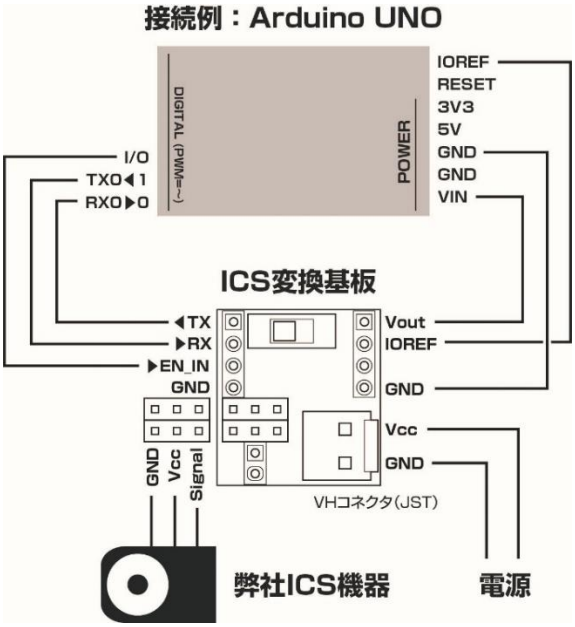
Communication, since the switch for switching the communication with the ICS equipment has been implemented, you can isolate the state.

However, because so you will not be able to communicate the PC and the ICS equipment at the same time, to send information from the ICS equipment in the PC, if you save the trouble of switching, ICS

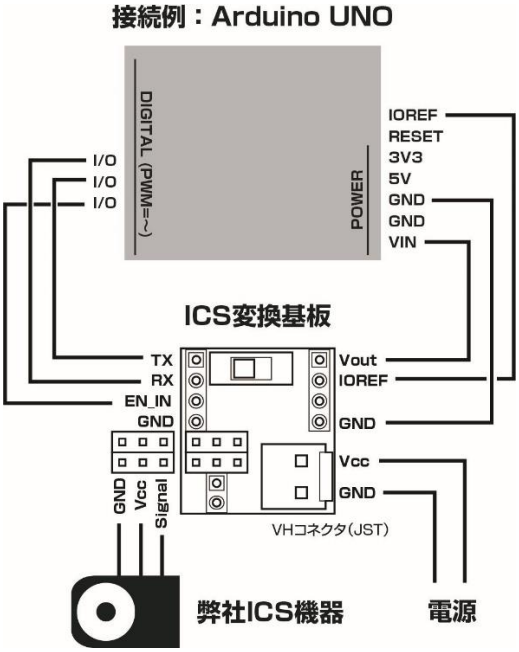
Please use the SoftwareSerial the wiring of the device connected to the Digital terminal.

For the connection, you may find it convenient to use the KSB shield 2. Description of the shield, in the next section " A method of using a shield board **"Please refer to.**

- **Wiring Diagrams**



In the case of HardwareSerial



In the case of SoftwareSerial

- **MCU communication terminal**

  Connect the terminal for sending and receiving information.

[In the case of HardwareSerial]

Arduino		ICS converter board (CN4)
Serial (UART) RX	⇔ TX	
Serial (UART) TX	⇔ RX	
Digital	⇔	EN_IN (transmit and receive switching terminal)

[In the case of SoftwareSerial]

Arduino		ICS converter board (CN4)
Digital	⇔ TX	

Digital	⇔ RX	
Digital	⇔	EN_IN (transmit and receive switching terminal)

Digital terminal that can be used in SoftwareSerial has been determined. Please use to confirm it is different terminals depending on the type of Arduino.

In the sample program, it has already connected the RX of the ICS equipment D8 pin, the TX to D9 pin.

The pin for switching transmission and reception (the EN -) is connected to the D2 pin.

The following power supply, HardwareSerial, wiring of SoftwareSerial are common.

- **Artuino power supply**

Do the wiring for supplying power from the battery to the Arduino.

Arduino		ICS converter board (CN3)
Vin	⇔ Vout	
IOREF	⇔ IOREF	
GND	⇔ GND	

- **ICS Equipment connection terminal (CN5 / CN6 - 2.54mm pitch· 3pin)**

Terminal	type	Description
1pin	Signal	ICS It can make communication with the device.
2pin	Vcc	ICS To supply power to the equipment
3pin	GND	-

ICS Equipment of the connection terminals, power has become to the following specifications. Please use caution because there is a polarity in connection.

Is better that there is a claw of the connector is attached to the connecting cable Signal (1pin)is.

- **Power supply terminal (CN2 - VH connector· 2pin)**

Terminal	type	Description
1pin	Vcc	It supplies the whole of the power supply. Depending on the servo type Power supply Te is different. **- HV The servo: 9 ~ 12V** **- LV The servo: 6 ~ 7.4V**
2pin	GND	

ICS To connect to the converter board, please use the servo connector of the connection cable. It is better that there is a claw of the connector Signal (Signal) is a line.

※ cause of device failure and mistake the direction of the connection. Please pay attention to the direction to which you want to connect.

A method of using a shield board

Although such as Universal board will take time and do the wiring of the above-mentioned view, it can be connected to the ICS conversion board and the Arduino

You can use the KSB shield 2, can be immediately available in only make a header pins. To shield the substrate, Arduino

It has become a mounted directly pin placed in UNO, the following pins are connected.

· TX of Serial (or Serial1), RX

Digital I / O D8, D9 (for SoftwareSerial) (requires a jumper by soldering)

Digital pin 2 to EN_IN

• Power (Vout, IOREF, GND) of the wiring

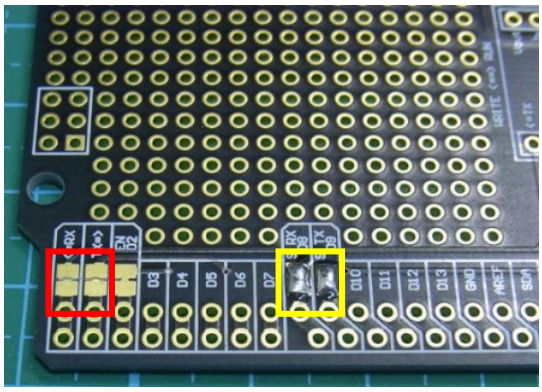
Shield board has front and back, side where the logo is on will be back. Inner places the universal substrate, use may land surface mount components are mounted

Please use if necessary that we have meaning.

HardwareSerial and SoftwareSerial can be used on the same shield, off the connection destination by a jumper by soldering because of different terminal to be used

Ri sort you need. Since the factory has been connected to the HardwareSerial, when used in SoftwareSerial is, off the jumper as follows:

Please be Toggles.



Preparing to use the KSB shield 2 in SoftwareSerial

As in the image, RX of the red frame, the narrow part of the pattern of the TX terminal was cut by a cutter, yellow

Color frame of S_RX (D8), and connect the pattern of S_TX (D9) terminal with solder. EN (D2) is cut

Please note that it is not.

Profile write-out write only way method of g

Arduino, such as the part of the board UNO UART there is only one system, information in the case of connecting the ICS equipment HardwareSerial so can also be used as the USB communication

Distribution There is a possibility that the communication mixed will not go wrong. Therefore, the ICS conversion substrate, the switching circuit at the time of communication between the communication time and the ICS device and PC

We added the switch to kill.

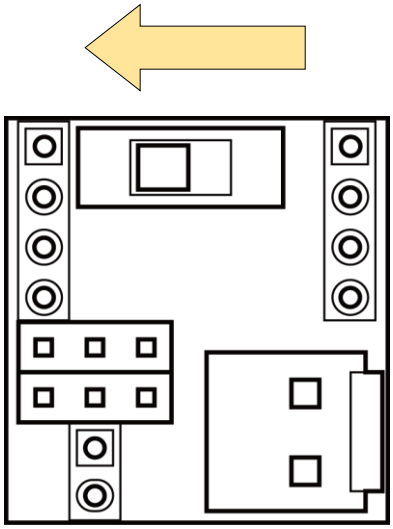
In the case of SoftwareSerial, there is no need to switch the switch. Please use the "execution mode" in the state that led the ICS equipment. In addition, Arduino

There are several UART features such as Mega, even if you do not use to write, switch you can leave the "execution mode".

[Program write mode]

When the program writing (at the time of communication with the PC) is on the side of "write"

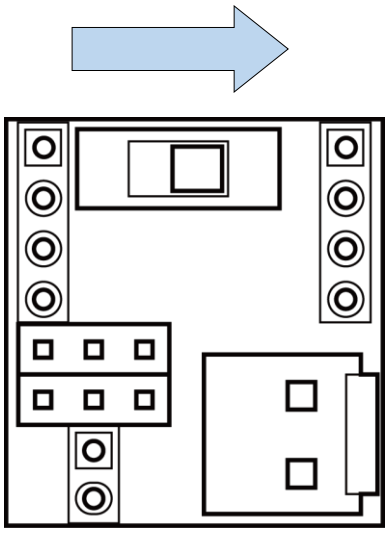
Please switch the switch.



[Program execution mode]

After yelling to write a program, when you run (at the time of communication with the ICS equipment) is

Please switch the switch in the opposite direction "run" side.



When you write a program with the switch to the execution mode, it will result in an error because the program can not write properly. In addition, until the switch of the writing mode

If you run or program, results in an error because it does not come back is communication from the ICS equipment, care must be taken because it becomes an unintended movement.

To have One to ICS L ib ra ryf or Ardu i no

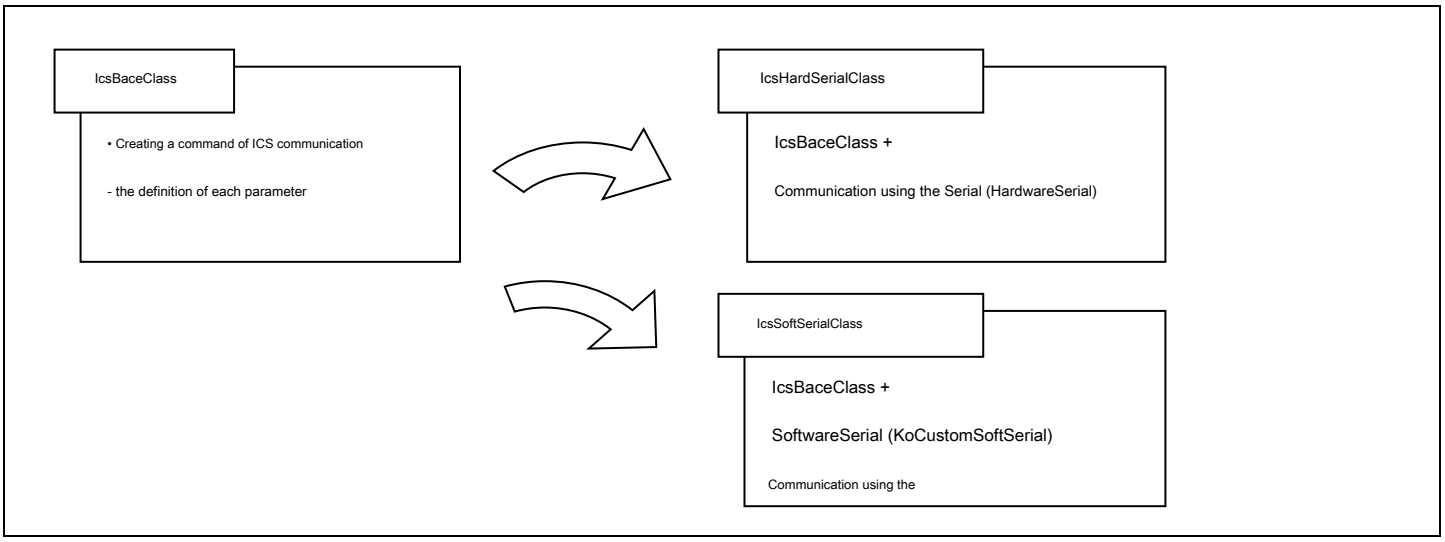
Library Overview of

The ICS equipment we have provided the library so that it can be easily controlled from the Arduino. The KRS servo specified by the ID number, and transmits the position (angle information)

You can easily make the angle control it. In addition, it is also possible to change the servo speed and stretch in the program.

Function for using the ICS equipment, and "IcsHardSerialClass" for using HardwareSerial the "IcsBaseClass" to the base of the class

"IcsSoftSerialClass" are available for using the SoftwareSerial.



IcsSoftSerialClass, which was verification using the SoftwareSerial class of Arduino standard, separately since the communication speed is not able to communicate not enough

We created a "KoCustomSoftSerial". If you want to use the IcsSoftSerialClass, please import also "KoCustomSoftSerial".

Library How to Obtain

It can be downloaded from our web site.

http://kondo-robot.com/faq/ics-library-a2 (All in a compressed file are summarized in one)

The contents of the library compressed file

In the compressed file, there is a library that is compressed by the ZIP to make it easier to import the library in Arduino. Samples for Arduino in the

Programs are also available.

ICS_Library_for_Arduino_V2 /

- ┣ IcsClassV200.zip (IcsBaceClass, IcsHardSerialClass)
- ┣ IcsSoftSerialClassV200.zip (IcsSoftSerialClass)
- ┣ KoCustomSoftSerial.zip (KoCustomSoftSerial)
- ┗ instruction manuals, etc.

KoCustomSoftSerial is used in common with the Rcb4SoftSerialClass. If you already have already imported, you do not need to import again.

IcsBaseClass has entered into the IcsHardSerialClassV200.zip.

Because it does not contain is in the IcsSoftSerialClassV200, if you use the SoftwareSerial, please also import IcsHardSerialClassV200.

Library import of

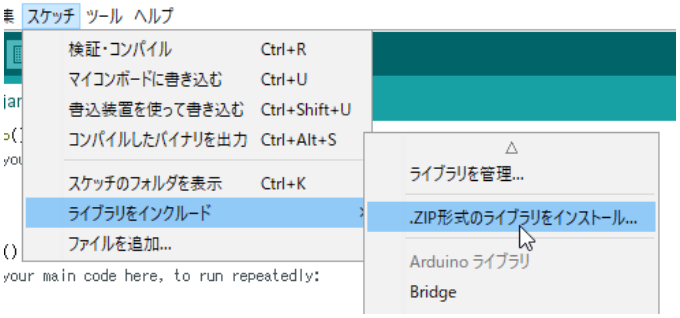
※ import method is an example of the version 1.8.3 of the Arduino IDE.

① Download the library set from our web site, and unzip it. zip file in the generated folder, please do not thaw.

② Start the Arduino IDE

"Sketch" → "Include library" of ③ menu bar

→ Choose "Install the .zip format of the library ..."



④ specify the ZIP file or folder that contains the "library to be installed

And, please ", and since the file specification dialog box will appear, down

Specify the location of the zip file that you load.

In the case of · HardwareSerial: IcsClassVxxx.zip

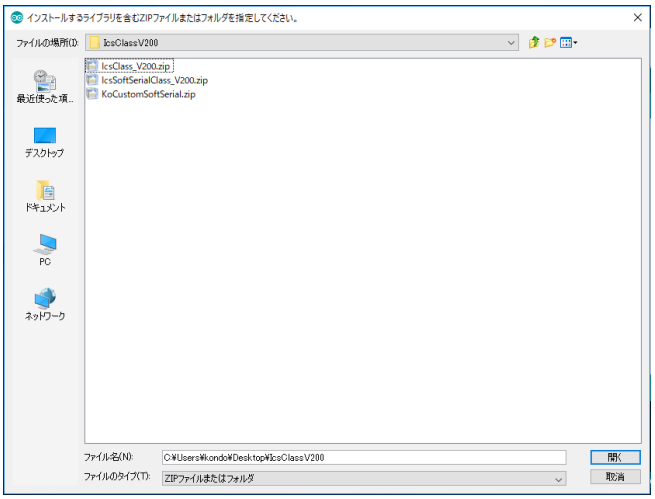
In the case of · SoftwareSerial: IcsSoftSerialClassVxxx.zip

KoCustomSoftSerialVxxx.zip

(Even if used in SoftwareSerial

IcsClassVxxx.zip Please also import)

(Xxx will vary depending on the version)

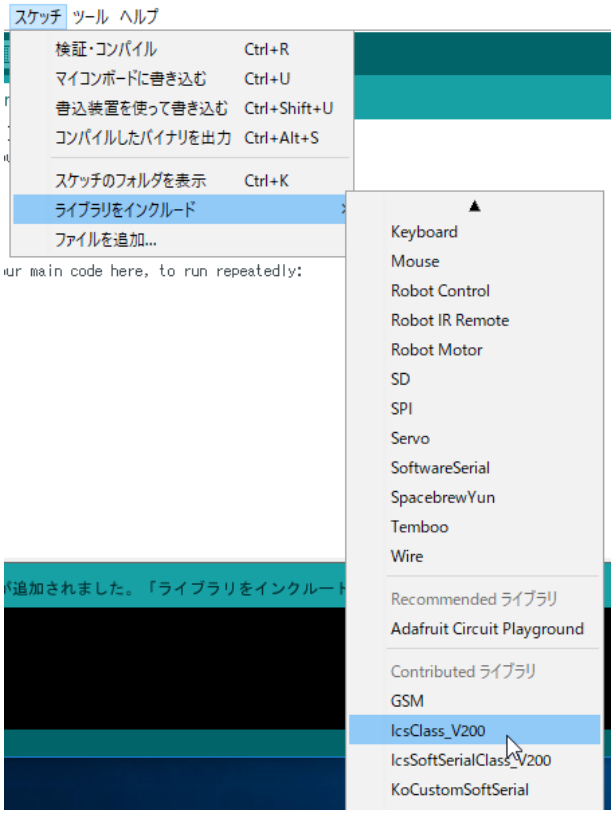


If ⑤ be in error, the library will be imported. Procedure was displayed in ③

"The library include" in the list "IcsClass_Vxxx" or,

It is a success when there is a display of "IcsSoftSerialClassVxxx".

※ The image on the right is when you import a version 200 of IcsClass



Library storage location

Folder of imported library is, by default, the data in the "Documents \ Arduino \ libraries" will be placed. For more information

Please refer to the ArduinoIDE of the manual.

Delete the library

If you want to erase the imported library, please turn off the folder of "Documents \ Arduino \ libraries". It should be noted that, if you turn off the folder

Restart of the Arduino IDE is required.

Stomach One to the difference down pull-profile g

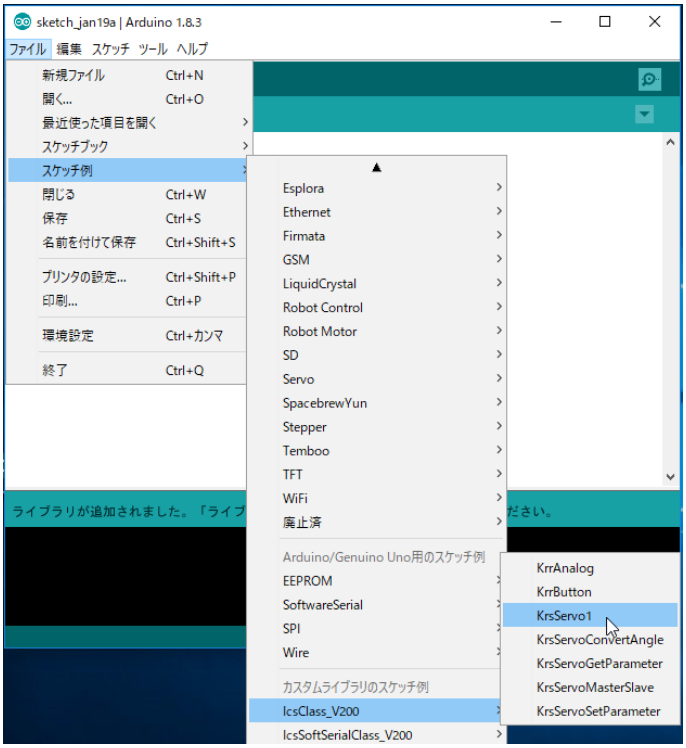
Call of the sample program

Imported at the same time the sample program when you import the library

It will be.

Call method, "file" → "Example of sketch" → "IcsClass_Vxxx"

Or call the sample in the "IcsSoftSerialClassVxxx".



Type of sample program

The sample program is located in the following nine. Function in HardwareSerial and SoftwareSerial but is the same, because there are different places, such as how to declare your Note

Please mind.

Sample program name	ICS Connected devices	Overview
KrsServo1	The servo x1 (ID: 0)	Move the servo at regular intervals.
KrsServoConvertAngle	The servo x1 (ID: 0)	Servo angle (degree) It specifies to move the.
KrsServoMasterSlave	The servo x2 (ID: 0, 1)	ID: 1 Get the servo position data ID: 0 And sends it to.
KrsServoGetParameter	The servo x1 (ID: 0)	Current value, temperature value, position data (ICS3.6 Gets the only).
KrsServoSetParameter	The servo x1 (ID: 0)	Stretch, and the setting of speed.
KrrButton	The servo x1 (ID: 0) KRR-5FH	KRR There move the servo based on the button data received.
KrrAnalog	The servo x1 (ID: 0) KRR-5FH	KRR There move the servo on the basis of the analog data received.

※ Please prepare the equipment that corresponds to it in order to run the sample program.

※ Communication speed of the ICS equipment is set to 115200bps. It can not be set in 1250000bps because there is an upper limit to the communication speed of the serial terminal of Aruduino.

※ Receiving machine KRR-5FH Transmitter to operate the sample program, which are paired KRC-5FH Is required. In addition, the analog data KRR-5FH

If you want to get from, KRC-5FH There is a need to analog output device is connected to.

KrsServo1 details

This section describes the KrsServo1 the underlying as an example. This sample, then move it to the left and right of the servo at regular intervals. Previously HardwareSerial A program that was used

Commentary and, after SoftwareSerial This explains the differences in.

Source code

```
# include <IcsHardSerialClass.h>


const byte EN_PIN = 2;
const long BAUDRATE = 115200;
const int TIMEOUT = 1000;                                  // Deliberately set in the late for sure not be able to communicate
IcsHardSerialClass krs (& Serial, EN_PIN, BAUDRATE, TIMEOUT); // Instance + EN pin( 2 Ban pin) and UART Designated


void setup () {
    krs.begin (); // Servo communication initialization
}


void loop () {
        krs.setPos (0,7500);              // Position command ID: 0 Servo 7500 Central to
        delay (500);                        //0.5 Wait seconds
        krs.setPos (0,9500);              // Position command ID: 0 Servo 9500 Right to
        delay (500);                        //0.5 Wait seconds
        krs.setPos (0,7500);              // Position command ID: 0 Servo 7500 Central to
        delay (500);                        //0.5 Wait seconds
        krs.setPos (0,5500);              // Position command ID: 0 Servo 5500 Left to
        delay (500);
}
```

Source code commentary

Let's take a look at the details of the source code of HardwareSerial. For SoftwareSerial rather than refer to the "If you want to use the SoftwareSerial" below

Please. For more information on Arduino code, please refer to the ArduinoIDE of the manual.

```
# include <IcsHardSerialClass.h>
```

HardwareSerial To communicate in IcsHardSerialClass It allows you to use the.

```
const int EN_PIN = 2;
const long BAUDRATE = 115200;
const int TIMEOUT = 1000;
IcsHardSerialClass krs (& Serial, EN_PIN, BAUDRATE, TIMEOUT); // instance + EN pin (pin 2) and specify the UART
```

IcsClass The class krs Initializes declared named.

- Of serial port settings => Serial

  UNO If you want to use is UART So but there is only one "Serial" Please describe the. Serial It is always required "&" before.

UART But if there is more than one described later.

- Send and receive switching pin (EN_IN) => Digital 2 Ban pin

ICS It is 1 One of the switch the transmission received by the communication line to communicate. To the switching 1 One uses a digital pin.

In the sample program, the digital 2 We are using a turn pin, but you can also use other digital pin.

- communication speed(baudrate) => 115200bps ※ 1250000bps It can not be set to.
- Read timeout (timeout) => 1000ms

ICS Set the waiting time of up to come back the reply data from the device. Beyond the set time to move to the next processing.

The point is, you can also write directly without declaring a variable.

```
IcsHardSerialClass krs (& Serial, 2,115200,1000);
```

```
krs.begin (); // Servo communication initialization
```

Declared krs The Initialize and start the communication.

ID0 In order to move the servo will be described as follows.

```
krs.setPos (0,7500);          // Position command ID: 0 Servo 7500 To (center)
krs.setPos (0,9500);          // Position command ID: 0 Servo 9500 To the (right)
krs.setPos (0,5500);          // Position command ID: 0 Servo 5500 To the (left)
```

krs Servo to send a position that is in is connected to the port that have been set and to operate.

Also, ID: 3 The servo 3500 (-135 If you want to operate in degrees),

```
krs.setPos (3,3500);          // Position command ID: 3 Servo 3500 To the (far left)
```

And described as.

Function that converts the angle to the position degPos () Since it has available, if necessary " KrsServoConvertAngle Please see the sample program ".

If you want to use the SoftwareSerial

```
# include <IcsSoftSerialClass.h>

const byte S_RX_PIN = 8;
const byte S_TX_PIN = 9;

const byte EN_PIN = 2;
const long BAUDRATE = 115200;
const int TIMEOUT = 200;          // softSerial In short there is a possibility that the communication fails
IcsSoftSerialClass krs (S_RX_PIN, S_TX_PIN, EN_PIN, BAUDRATE, TIMEOUT); // IcsSoftSerialClass Definition of, softSerial Edition
```

If you want to use the SoftwareSerial has a different point declare. Each function you can use the same operating such as setPos ().

```
# include <IcsSoftSerialClass.h>
```

It depends on the name of the include files.

```
const byte S_RX_PIN = 8;
```

```
const byte S_TX_PIN = 9;
```

Declare a terminal to be used in the SoftwareSerial. 8 and 9 is the D8 and D9 is a digital I / O pins on the Arduino, respectively. RX pin by Arduino is

Since it is limited, please check the manual of the Arduino.

```
const byte EN_PIN = 2;

const long BAUDRATE = 115200;

const int TIMEOUT = 200;              // softSerial In short there is a possibility that the communication fails
```

The point is, this is the same as the HardwareSerial.

```
IcsSoftSerialClass krs (S_RX_PIN, S_TX_PIN, EN_PIN, BAUDRATE, TIMEOUT); // IcsSoftSerialClass Definition of, softSerial version
```

The IcsSoftSerialClass class and initialize declared in the name of krs. The initialization, we use the value assigned to each variable in front of this sentence.

In the case of using a plurality of UART

In the sample, but it was a declared in the named krs the IcsHardSerialClass class, can be declared in any name, such as krs_1 and krs_2.

```
IcsHardSerialClass krs_1 (& Serial1, EN_PIN, BAUDRATE, TIMEOUT); // Serial1 And digital 2 Using the turn pin

IcsHardSerialClass krs_2 (& Serial2, EN_PIN, BAUDRATE, TIMEOUT); // Serial2 And digital 3 Using the turn pin
```

If the name is the same, you can declare a multiple of IcsHardSerialClass at the same time.

So you can use multiple UART by changing the Serial port declared the IcsHardSerialClass for each UART, increasing the number of ICS devices to be connected

It is also possible. However, please be careful Serial port and EN_PIN of each IcsHardSerialClass do not conflict.

It is also possible to coexist with Rcb4Class by selectively using a plurality of serial terminals.

The contents of the above is true even SoftwareSerial. However, SoftwareSerial becomes unstable is much traffic and operation by devices connected increases situ

Please note that there is a case.

If you declare will need to perform the initial configuration of the communication and the like, respectively.

```
Krs_1.begin (); // krs_1 In declared IcsHardSerialClass Communication initial set of

krs_2.begin (); // krs_2 In declared IcsHardSerialClass Communication initial set of
```

If you move each ICS devices can be described as follows.

```
Krs_1.setPos (3,9500);          // Position command krs_1 In set Serial (Serial1) Using the KRS The servo ID3 Instruction to

Krs_2.setPos (5,6500);          // Position command krs_2 In set Serial (Serial2) Using the KRS The servo ID5 Instruction to
```

For more information on ICS L ib ra ryf or Ardu i no

The contents of the ICS Library for Arduino is composed of the following classes, in the definition.

Class / definition name	Content
IcsBaseClass	Class summarizes the command generation and the definition of the ICS
IcsHardSerialClass	Derived from IcsBaseClass, to communicate with ICS device using Serial (HardwareSerial)
IcsSoftSerialClass	Derived from IcsBaseClass, to communicate with ICS device using SoftwareSerial (KoCustomSoftSerial)
KoCustomSoftSerial	Remodeling the SoftSerial of Arduino to be able to communicate with 115200bps
KRR_BUTTON	It defines the button data of KRR

From the following, we enumerate the functions required to move the ICS equipment. Since there is also a function that is not listed, html version of the decompression folder for more information

Please see the index.html in the _Vxxx folder. (Xxx depends on the version.)

At the example of explanatory text will come out with krs.xxxx, but declares the IcsHardSerialClass / IcsSoftSerialClass named krs.

First set for communicating with ICS equipment

Declare the ICS class (HardwareSerial version) IcsHardSerialClass ()

[Format]

IcsHardSerialClass (HardwareSerial * icsSerial, int enpin)

IcsHardSerialClass (HardwareSerial * icsSerial, int enpin, long baudrate, int timeout)

【Overview】

constructor

Declare a class (instance), function for initializing

【argument】

type	name	Description	Setting range
in	* icsSerial	It is set to ICS UART (HardwareSerial type pointer)	UNO: Serial Mega: Serial1 ~ Serial3
in	enpin	Pin number of transmit and receive switching pin	Any digital pin vacant
in	baudrate	Communication speed of the ICS	115200bps only ※
in	timeout	Receive Timeout (ms)	~ 32767ms

※ Depending on the type of Arduino, you may be able to use 625000,1250000 (1.25M) bps.

【Example of use】

```
IcsHardSerialClass krs (& Serial, 2,115200,5); // 115200bps communication settings using Serial and digital 2 pins, communication latency 5ms
```

【Description】

Use the HardwareSerial communicate with the ICS equipment is a function of declaring a "IcsHardSerialClass". In even the initial setting at the same time the C language to the declaration difference

to come. If you want to start the communication, separately Function for communication start (begin ()) Use.

HardwareSerial type is the class of UART that is used in the Arduino. In the Arduino environment, and Serial the UART1 in HardwareSerial type

It has been declared. Since (in the case of the Mega Serial1 and Serial2) in declared Serial HardwareSerial use directly, a pointer to the initial setting

Pass.

enpin sets the EN_IN pin to switch the transmission and reception of ICS communication. In the shield board is used as a fixed digital pin 2, but the other sky

It is also possible to use Iteiru digital pin.

baudrate, you can set the communication speed to communicate with the ICS equipment. ICS equipment, but it is possible to communicate with up to 1.25Mbps, communication speed of Arduino

Because does not correspond to 1.25Mbps, usually uses the 115200bps.

timeout is the time to wait for a reply data. But if the ICS equipment is connected correctly it will return the data, if you or not connected wait

It will continue so, it's time to abort it.

Declare the ICS class (SoftwareSerial version) IcsSoftSerialClass ()

[Format]

IcsSoftSerialClass (byte rxPin, byte txPin, byte enpin)

IcsSoftSerialClass (byte rxPin, byte txPin, byte enpin, long baudrate, int timeout)

IcsSoftSerialClass (KoCustomSoftSerial * koSoftSerial, byte enpin)

IcsSoftSerialClass (KoCustomSoftSerial * koSoftSerial, byte enpin, long baudrate, int timeout)

【Overview】

constructor

Declare a class (instance), function for initializing

【argument】

type	name	Description	Setting range
in	rxPin	Pin number to receive the data	Note that the pin is different that can be used by the Arduino
in	txPin	Pin number to send the data	Any digital pin vacant
in	enpin	Pin number of transmit and receive switching pin	Any digital pin vacant
in	baudrate	Communication speed of the ICS	115200bps only ※
in	timeout	Receive Timeout (ms)	~ 32767ms
in	* koSoftSerial	Is set to ICS communication Soft (KoCustomSoftSerial type of poi Printer)	UNO: Serial Mega: Serial1 ~ Serial3

※ SoftSerial will be only 115200bps.

【Example of use】

```
IcsSoftSerialClass krs (8,9,2,115200,5);
// Instance + RX pin (8 pin), TX pin (pin 9), EN pin (pin 2) and the communication speed (115200bps), sets a timeout (5 ms)
```

【Description】

Use the SoftSerial communicate with the ICS equipment is a function of declaring a "IcsSoftSerialClass". When the declaration different from the C language can be initialized at the same time.

For the use of Arduino genuine SoftSerial you can not communicate at high speed we have provided the KoCustomSoftSerial a converted SoftSerial.

If you want to start the communication, separately Function for communication start (begin ()) Use.

KoCustomSoftSerial is, you can use in much the same way as SoftSerial. rxPin, but it can be used by specifying the pin in txPin, depending on the Arduino RX

Since there is a pin that can not be used to pin, please make sure SoftSerial of Arduino.

enpin, baudrate, with regard timeout is the same as the IcsHardSerialClass.

To open the ICS class ~ IcsHardSerialClass () / ~ IcsSoftSerialClass ()

[Format] None

【Overview】

Destructor

It frees the class that declares.

Function is the end, if you do not use will be called automatically.

begin to initiate a communication ()

[Format]

bool begin ()

bool begin (long baudrate, int timeout)

bool begin (HardwareSerial * serial, int enpin, long baudrate, int timeout　　long baudrate, int timeout)　　// (IcsHardSerialClass)

bool begin (KoCustomSoftSerial * serial, byte enpin, long baudrate, int timeout)　　// (IcsSoftSerialClass)

【Overview】

The initial setting of the communication, to start the communication.

【argument】

Type name		Description	Setting range
in	* serial	Pointer of the UART be set to ICS (Pointer of HardwareSerial or KoCustomSoftSerial)	

in	enpin	Pin number of transmit and receive switching pin	Any digital pin vacant
in	baudrate	Communication speed of the ICS	115200bps only ※
in	timeout	Receive Timeout (ms)	~ 32767ms

【Return value】

value	Description
true	Communication setting completion
false	Communication settings fail

【Example of use】

```
krs.begin ();
```

【Description】

The initial setting of the communication, to start the communication.

Arduino initiates communication with the ICS equipment from after describing this function. Please be sure to declare before to describe each function.

Although usually use a function that does not have the argument, if you want to re-initial settings, please use the function of there argument.

synchronize to send and receive using the command ()

[Format]

bool synchronize (byte * txBuf, byte txLen, byte * rxBuf, byte rxLen)

【Overview】

Transmission of ICS communication, do the reception.

【argument】

Type name		Description
in, out	* TxBuf	Transmit data storage buffer
in	txLen	The number of transmit data
out	* RxBuf	Receive data storage buffer
in	rxLen	The number of received data

【Return value】

value	Description
true	Communication success
false	Communication failure

【Caution】

The number of transmission data, the number of received data, please note that depends on the command.

【Example of use】

```
byte txCmd [3] = {0x80,0x3A, 0x4C}; // ID0 to the sequence of the direct command to send a 7500
byte rxCmd [3];                     // reception of the data array
. Krs synchronize (txCmd, sizeof (txCmd), rxCmd, sizeof (rxCmd));
```

【Description】

If you want to send commands directly to each ICS devices you use this function. This function, setPos described below (), etc. have been used in each function.

Please use if you want to send and receive commands on your own with reference to the ICS3.5 / 3.6 of the software manual.

Fixed parameter list

【Overview】

It defines the maximum value and the value at the time of communication error of the value to be used in the servo.

Please use at the time of the determination of the servo parameter ranges judgment and communication errors.

【Fixed value】

Name, type	value	Description
static const int MAX_ID	31	The maximum value of the servo ID
static const int MIN_ID	0	The minimum value of the servo ID
static const int MAX_POS	11500	Servo position maximum value
static const int MIN_POS	3500	Servo position minimum value
static const int ICS_FALSE	- 1	Value when that failed in so ICS communication

【Example of use】

```
krs.setPos (0, IcsHardSerialClass :: MAX_POS);          // Position command ID: 0 moves the servo to a maximum value (MAX_POS)
krs.setPos (0, krs.MAX_POS);                            // ↑ the same. Can also be used as the left description and krs is declared
```

Move the servo motor

Move the servo motor setPos ()

[Format]

int setPos (byte id, unsigned int pos)

【Overview】

Move to specify the angle of the servo in the position data.

【argument】

type	name	Description	Setting range
in	id	Servo ID number	0 ~ 31 ※
in	pos	Position data	3500-7500 (Center) - 11500

※ for KRR-5FH is fixed to the ID31, when the receiver is connected servo can not use the ID31. The same is true in the following function.

【Return value】

type	value	Description
In the case of success	3500-115000	Position data is returned
In the case of failure	- 1	Out-of-range, communication failure (**ICS_FALSE)**

【Example of use】

```
krs.setPos (3,3500);                    // position command ID: 3 servo to 3500 (far left)


// If you want to get the current value in the position command
int pos;
pos = krs.setPos (0,7500);                    // position command ID: 0 servo to 7500 (center)
if (pos == HardwareSerial :: ICS_FALSE)                    // check if communication fails
{
    // failure treatment
}
```

【Description】

Servo specified by the ID is the function to operate at any angle. Angle must be specified as a number of position data from 3500 to 11,500. 7500 to specify when the news

Toraru Go to the position (center). If the communication is successful, it will be the current value of the servo is returned as the return value, but if it fails -1 (**ICS_FALSE)**

It is returned. Process in the case where communication using this fails is also possible to program.

setFree to free the servo ()

[Format]

int setFree (byte id)

【Overview】

And the servo-free (weakness) state.

【argument】

Type name		Description	Setting range
In	id	Servo ID number	0 to 31

【Return value】

type	value	Description
In the case of success	3500-115000	Position data
In the case of failure	- 1	Out-of-range, communication failure (**ICS_FALSE)**

【Example of use】

```
krs.setFree (3);                // free instruction ID: 3 to the free state
```

【Description】

It is a function of the order to the servo specified by the ID to free (weakness) state. setPos () and also the communication is successful servo current position, fail when -1 is return value

As it will return.

Please use setPos () to return to again hold (force is entered) state. Get the current position in setFree (), is specified the value in setPos (), robot

You can return each joint of the door to the angle remains the hold state of the status quo.

Writing parameters of the servo motor

Stretch writing setStrc ()

[Format]

int setStrc (byte id, unsigned int strc)

【Overview】

Write the value of the servo stretch (holding force).

【argument】

Type name		Description	Setting range
in	id	Servo ID number	0 to 31
in	strc	Stretch of data	(Soft) 1 ~ 127 (Hard)

【Return value】

type	value	Description
In the case of success	1-127	Data of written stretch
In the case of failure	- 1	**Communication failure (ICS_FALSE)**

【Example of use】

```
krs.setStrc (0,90); // a stretch of ID0 and to 90
```

【Description】

Is a function for writing the value of the specified servo stretch in the ID. Stretch and refers to the holding force of the shaft. Holding force is weakened every time closer to 1,

Strongly will you each time closer to 127. Or it becomes smooth motion of the shaft by lowering moderately stretch, and lower when hunting began to occur effect

It is specific. Also, usually you can use these options as raised to 127 at the timing required hold out in advance is set to about 60.

Speed writing setSpd ()

[Format]

int setSpd (byte id, unsigned int spd)

【Overview】

Change the servo speed (output).

【argument】

Type name		Description	Setting range
in	id	Servo ID number	0 to 31
in	spd	Speed of data	(Slow) 1 ~ 127 (Fast)

【Return value】

type	value	Description
In the case of success	1-127	Data of the written speed
In the case of failure	- 1	**Communication failure (ICS_FALSE)**

【Example of use】

```
krs.setSpd (0,90); // the ID0 speed of you to 90
```

【Description】

Is a function for rewriting the value of the servo speed specified by the ID. Slower each time closer to 1, faster each time closer to 127.

Current value writing setCur ()

[Format]

int setCur (byte id, unsigned int curlim)

【Overview】

Change the current limit (current limit) value of the servo.

【argument】

Type name		Description	Setting range
in	id	Servo ID number	0 to 31
in	curlim	Data of the current value	(Low) 1 ~ 63 (High)

【Return value】

type	value	Description
In the case of success	1 to 63	Servo of the current limit value
In the case of failure	- 1	**Communication failure (ICS_FALSE)**

【Example of use】

```
krs.setCur (0,20); // Set the current limit of ID0 to 20
```

【Description】

Is a function for rewriting the current limit value of the servo specified by the ID. The servo will increase the current value becomes the state in which the lock, but the current detected by the servo

Exceeds the threshold value is set will servo to the weakness state. And then return it falls below the threshold. 0.1A when the 1, when the 20 will be on the threshold of 2.0A. Ship-like

The state is set to 40.

Temperature value writing setTmp ()

[Format]

int setTmp (byte id, unsigned int tmplim)

【Overview】

Servo of the temperature limit to change the (temperature limit) value.

【argument】

Type name		Description	Setting range
in	id	Servo ID number	0 to 31
in	tmplim	Data of temperature values	(High) 1 ~ 127 (Low)

【Return value】

type	value	Description
In the case of success	1-127	Servo of the temperature limit value
In the case of failure	- 1	**Communication failure (ICS_FALSE)**

【Example of use】

```
krs.setTmp (0,20); // Set the temperature limit of ID0 to 20
```

【Description】

Is a function for rewriting the temperature limit of the servo specified by the ID. Temperature rises continue to use servo, the threshold set temperature detected by the servo ultra

Obtain the servo will be the weakness state. And then return it falls below the threshold. Becomes the set value at high temperatures each time closer to 1, low temperature each time closer to 127

It will produce the values specified by. 30 at 100 ℃, is a set of 75 at 70 ℃. By default says is set to 75.

To get the parameters of the servo motor

Stretch read getStrc ()

[Format]

int getStrc (byte id)

【Overview】

It reads the current value of the servo stretch (holding force).

【argument】

Type name		Description	Setting range
in	id	Servo ID number	0 to 31

【Return value】

type	value	Description
In the case of success	1-127	Read stretch of data (Soft) 1 ~ 127 (Hard)
In the case of failure	- 1	**Communication failure (ICS_FALSE)**

【Example of use】

```
int st;

st = krs.getStrc (0); Read the stretch of // ID0
```

【Description】

Is a function for reading the value of the specified servo stretch in the ID.

Speed read getSpd ()

[Format]

int getSpd (byte id)

【Overview】

It reads the current of the servo speed (output).

【argument】

Type name		Description	Setting range
in	id	Servo ID number	0 to 31

【Return value】

type	value	Description
In the case of success	1-127	Read speed of data (Slow) 1 ~ 127 (Fast)
In the case of failure	- 1	Communication failure (**ICS_FALSE)**

【Example of use】

```
int sp;
sp = krs.getSpd (0); to read the speed of the // ID0
```

【Description】

Is a function for reading the value of the servo speed specified by the ID.

Read the current of the current value getCur ()

[Format]

int getCur (byte id)

【Overview】

It reads the current of the current value of the servo.

【argument】

Type name		Description	Setting range
in	id	Servo ID number	0 to 31

【Return value】

type	value	Description
In the case of success	1-127	Present current value of the servo (CW) 1 ~ 63, (CCW) 64 ~ 127
In the case of failure	- 1	Communication failure (**ICS_FALSE)**

【Example of use】

```
int cur;
cur = krs.getCur (0); // Read the current value of ID0
```

【Description】

Is a function for reading the current of the current value of the servo specified by the ID.

It is not in current limit ※.

Read the current temperature value getTmp ()

[Format]

int getTmp (byte id)

【Overview】

It reads the current temperature value of the servo.

【argument】

Type name		Description	Setting range
in	id	Servo ID number	0 to 31

【Return value】

type	value	Description
In the case of success	1-127	Current temperature value of the servo (High) 1 ~ 127 (Low)
In the case of failure	- 1	**Communication failure (ICS_FALSE)**

【Example of use】

```
int tmp;

tmp = krs.getTmp (0); // read the temperature value of ID0
```

【Description】

Is a function for reading the current temperature value of the servo specified by the ID.

※ There is no temperature limit.

Read the current position data getPos ()

[Format]

int getPos (byte id)

【Overview】

It reads the current position data of the servo.

※ is valid from ICS3.6. It does not return a reply if sent to the servo of ICS3.5.

【argument】

Type name		Description	Setting range
in	id	Servo ID number	0 to 31

【Return value】

type	value	Description
In the case of success	3500 ~ 11500	Current position data of the specified ID
In the case of failure	- 1	Communication failure (**ICS_FALSE)**

【Example of use】

```
int pos

pos = krs.getPos (0); // Read the current position data of the ID0

if (pos == IcsHardSerialClass :: ICS_FALSE)                    // check if communication fails

{

    // failure treatment

}
```

【Description】

Is a function for reading the servo of the angle specified by the ID.

Converting the position data and the angle (degree)

Angle (degree) of converting the servo position data degPos ()

[Format]

static int degPos (float deg)

【Overview】

It converts angle (degree) a (float type) to the position data.

【argument】

Type name		Description	Setting range
in	deg	Angle (degree) (float type)	- 135.0 to 135.0

【Return value】

type	value	Description
In the case of success	3500-11500	Position data
In the case of failure	- 1	Communication failure (**ICS_FALSE)**

【Example of use】

```
. Int pos = krs degPos (-90.0);                    // - 90.0deg (degrees) and converts it to position data
krs.setPos (0, pos);                                 // the converted data ID: 0 Sends of the servo
```

【Description】

Is a function for converting the servo angle to the position data. Since the operation angle of the servo is 270 °, convert the center to the 0 ° and the left and right -135.0 ° ~ 135.0 °

I can do it. If you pass the angle as an argument, position data in the return value is returned.

posDeg for converting position data at an angle ()

[Format]

static float posDeg (int pos)

【Overview】

It converts the position data to the angle (degree) (float type).

【argument】

Type name		Description	Setting range
in	pos	Position data	3500-11500

【Return value】

type	value	Description
In the case of success	- 135.0 to 135.0	Angle (degree) (float type)
In the case of failure	9999.9	Positive direction out of range (ANGLE_F_FALS)
	- 9999.9	Negative range (-ANGLE_F_FALS)

【Example of use】

```
int pos;

float deg;

pos = krs.getPos (0); // Read the current position data of the ID0

. Deg = krs posDeg (pos);                // to convert the position data that has been read to the angle
```

【Description】

The degPos () is a function for converting the position data to the angle in reverse. This function only, if it fails 9999.9 (forward), - 9999.9 (reverse) returns

It will be returned as a value.

degPos100 for converting 100 times the angle in the position data ()

[Format]

static int degPos100 (int deg)

【Overview】

Angle (degree) x 100 a (int type) and converts it to position data.

【argument】

Type name		Description	Setting range
in	deg	Angle (degree x 100) (int type)	- 13500-13500

【Return value】

type	value	Description
In the case of success	3500-11500 converted position data	
In the case of failure	- 1	Out of range

【Example of use】

```
. Int pos = krs degPos100 (-9050);                    // - 90.5deg (degrees) and converts it to position data
krs.setPos (0, pos);                                  // the converted data ID: 0 Sends of the servo
```

【Description】

- The 135.0 to 135.0 is a function that converts the 100 multiplied by -13500 ~ 13500 to the position data. Please use if you want to avoid the calculations using the minority.

Converting the position data to 100 times the angle posDeg100 ()

[Format]

static int posDeg100 (int pos)

【Overview】

Converts the position data to the angle (degree) x 100 (int type).

【argument】

Type name		Description	Setting range
in	pos	Position data	3500-11500

【Return value】

type	value	Description
In the case of success	- 13500 ~ 13500	Servo of the temperature limit value
In the case of failure	0x7FFF (32767)	Positive direction out of range (ANGLE_I_FALSE)
	0x8000 (-32768)	Negative range (-ANGLE_I_FALSE)

【Example of use】

```
int pos;
int deg100;
```

```
pos = krs.getPos (0); // Read the current position data of the ID0

. Deg100 = krs posDeg100 (pos);                    // to convert the position data that has been read to the angle
```

【Description】

The angle of the position data to the original is a function for converting to the numerical value to 100-fold (-13500 ~ 13500). This function only, if it fails 0x7FFF (positive

Rolling), will be returned as the return value 0x8000 (reverse).

To obtain the value of KRR (receiver)

enum KRR_BUTTON: unsigned short

[Format]

enum KRR_BUTTON: unsigned short

【Overview】

KRR has been assigned defines the value of a button data in the unsigned short type to receive.

In the case of the press simultaneously takes the logical sum of the respective data.

[Enumeration value]

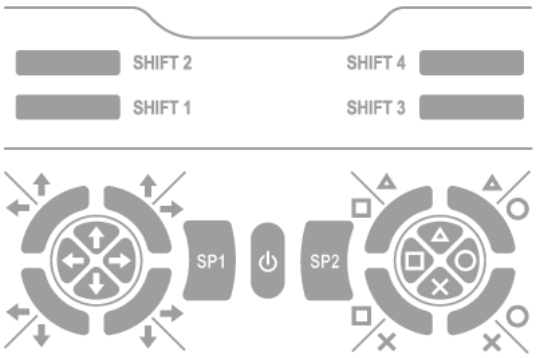
name	Numeric value	Corresponding button
KRR_BUTTON_NONE	0x0000	Nothing has been pressed
KRR_BUTTON_UP	0x0001	↑
KRR_BUTTON_DOWN	0x0002	↓
KRR_BUTTON_RIGHT	0x0004	→
KRR_BUTTON_LEFT	0x0008	←
KRR_BUTTON_TRIANGLE	0x0010	△
KRR_BUTTON_CROSS	0x0020	×
KRR_BUTTON_CIRCLE	0x0040	○
KRR_BUTTON_SQUARE	0x0100	□
KRR_BUTTON_S1	0x0200	Shift 1 left front
KRR_BUTTON_S2	0x0400	Shift 2 left rear
KRR_BUTTON_S3	0x0800	Shift 3 right front
KRR_BUTTON_S4	0x1000	Shift 4 right back
KRR_BUTTON_FALSE	0xFFFF	Error value (reception failure, etc.)

【Example of use】

```
unsigned short S1_Up = KRR_BUTTON_UP | KRR_BUTTON_S1;          // the value of the time you press the same time the S1 and ↑ button
```

[Note]

Button placement of the KRC-5FH will be in the figure below.



【Description】

Numbers have been assigned to each button on the wireless controller KRC-5FH. KRR-5FH is, so you notice the data received by these figures, the table above

Please to determine which button has been pressed in reference to. Button It is also possible to combine. Because the combination will be the logical sum, ↑ (0x0001)

When the → (0x0004) when it is pressed same time, you will be notified 0x0005.

To get the button data getKrrButton ()

[Format]

unsigned short getKrrButton ()

【Overview】

KRR to get the data of the button that is receiving.

【argument】

None

【Return value】

type	value	Description
In the case of success	-	KRR_BUTTON Button data that are defined will come back in. In the case of press simultaneous, it takes the logical sum.
In the case of failure	0xFFFF	Communication failure (**KRR_BUTTON_FALSE)**

【Example of use】

```
unsigned short buttonData;
buttonData = krs.getKrrButton ();
if (buttonData == KRR_BUTTON_UP)                    // ↑ when the only button is pressed
{
    // ↑ processing when the button is pressed
```

```
}
```

【Description】

Received button data is a function for reading.

※ ID does not specify otherwise, but the receiver KRR-5FH has been fixed at ID31.

To get the analog data getKrrAnalog ()

[Format]

int getKrrAnalog (int paCh)

【Overview】

KRR will get the data of the analog port specified has been received (PA).

【argument】

Type name		Description	Setting range
in	paCh	Number of analog ports	1 to 4

【Return value】

type	value	Description
In the case of success	0-127	Data of the specified analog port
In the case of failure	- 1	Communication failure (**ICS_FALSE)**

【Example of use】

```
int adData;
. AdData = krs getKrrAnalog (1);                    // Get the data of PA1
```

【Description】

Is a function to read the value from the analog terminal that is implemented in the KRC-5FH. PA1, which is stamped on the body fall into getKrrAnalog (1).

Since the analog port of the KRC-5FH is PA1 ~ 4, it will be returned with an error if you pass any other value.

To get at once all of the data getKrrAllData ()

[Format]

bool getKrrAllData (unsigned short * button, int adData [4])

【Overview】

KRR to get all of the data of the button data and the analog port is receiving (PA).

【argument】

Type name		Description	Setting range
out	* button	KRR_BUTTON Button data that are defined will come back in. In the case of press simultaneous, it takes the logical sum.	You pass a pointer. The value is assigned to the variable passed if successful You.
out	adData	4ch worth of analog data	The size of the array is 4 fixed

【Return value】

type	value	Description
In the case of success	ture	Communication success
In the case of failure	false	Communication failure

【Example of use】

```
int addata [4];
unsigned short buttonData;
. Krs getKrrAllData (& buttonData, addata);                    // get all of the data (pass a pointer, rewrites the contents)
if (buttonData == KRR_BUTTON_UP)                    // ↑ when the only button is pressed
{
    // ↑ processing when the button is pressed
}
```

【Description】

All of the data that KRR-5FH has received is a function for reading all at once.

Revision history

version	date	More detail
1.0	2017/02/23	Create a first edition.
2. 0	2018/01/19	Create a second edition