

Specifications

Servo Controller Specs

Channels 9

Each channel can be set to 12us to 12.2ms in .2us increments.
12us to 2.2ms without affecting refresh rate

Default refresh rate 20ms

Can be modified by decreasing/increasing the pulse fillin rate or by removing channels.

Controller power requirements (does not include servos)

3.0v—5.5v

3ma

Controller Speed

20Mhz

Protocol specs

Async 9600-1250000 baud (programmable)

115200 default

90 byte IRQ driven buffer

Multiple units can be placed on single bus as slaves.

Protocol 2-9 byte variable byte protocol. With floating transmit (slave)

Slave transmit hold time (programmable)

Slave transmit delay (programmable)

Slave Resync hold time (programmable)

Internal error recovery

Warning

Running a servo past its physical extents can pull excessive power and cause damage to the servo. This can also reset the controller (if providing power for servo)

Never plug a servo into the controller while power is on.

© Copyright 2003

Kronos Robotics

www.kronosrobotics.com

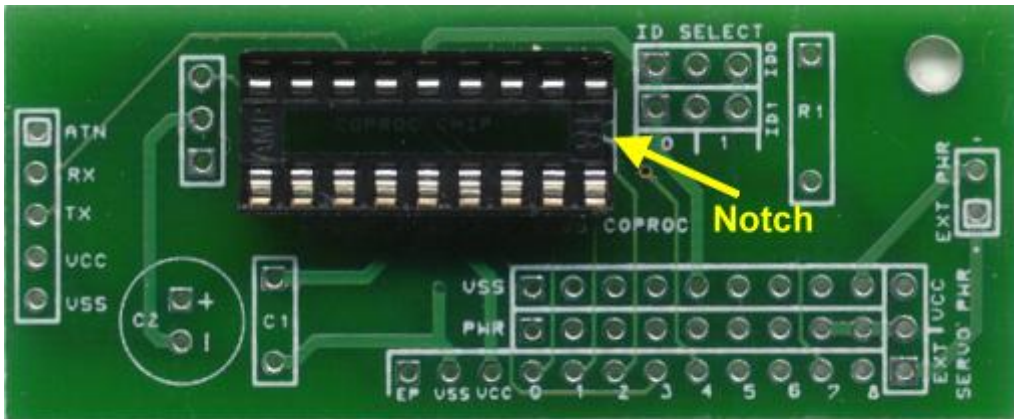
msimpson@kronosrobotics.com

Table of Contents

| | |
|--|----|
| Specifications | 2 |
| Assembly | 4 |
| Hookup | 6 |
| Protocol Definition | 8 |
| Command Definition (protocol Specific) | 9 |
| Command Definition (coproc Specific) | 10 |
| EEPROM Values | 11 |
| Dios Coproc Basic Dios Library | 14 |
| Dios CPServo Dios Library | 15 |
| Links | 16 |

Assembly

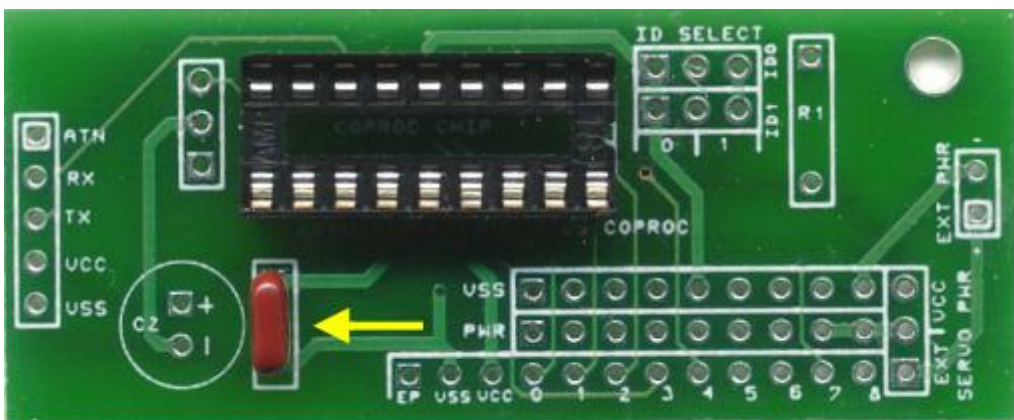
Each kit will take you about 30 minutes to build. You will need a soldering iron, solder, and wire cutters. Please read all the assembly instructions before beginning assembly. You can find a full color version of this document online at: <http://www.kronosrobotics.com/products/pdfs/pdfs.htm>



Step 1

Insert the 18 pin socket into the location marked U1. Carefully flip the board over so that the board rests on the socket. Make sure the notches are facing the right side of the board as shown.

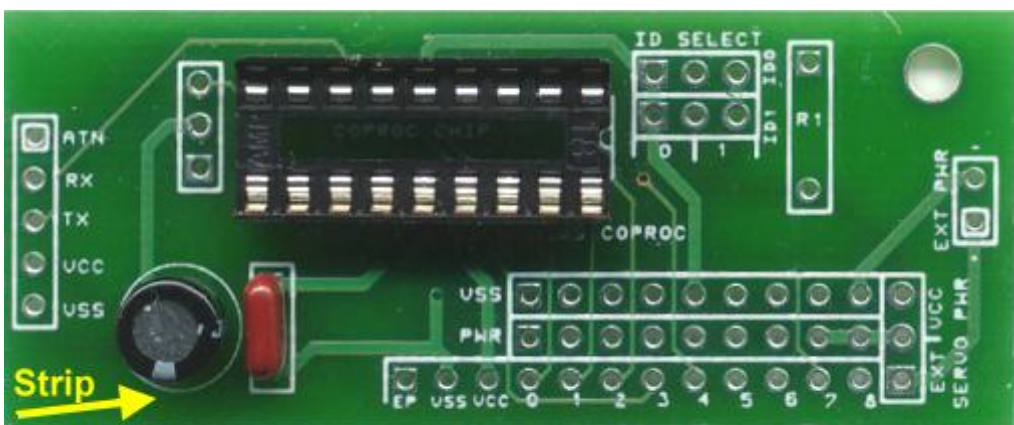
Solder in place.



Step 2

Insert the .1uF capacitor into the position labeled C1. (It's the dark red one)

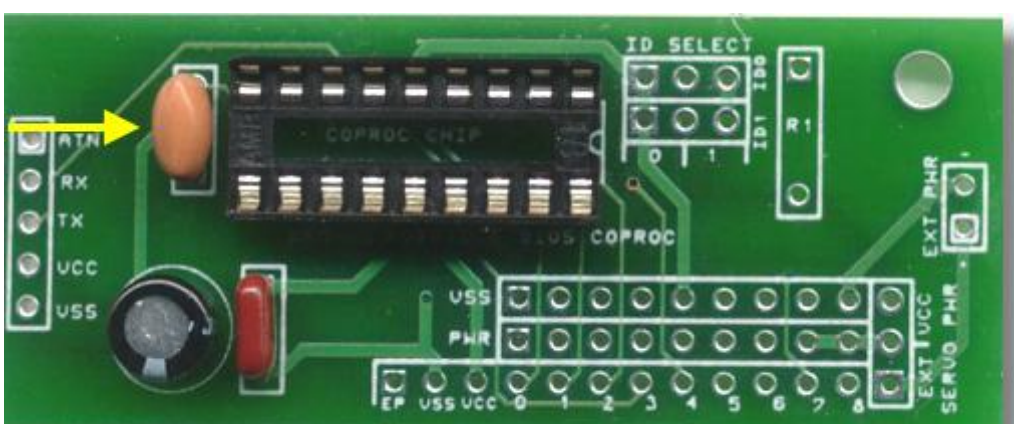
Solder in place and clip the excess leads.



Step 3

Insert the 100uF capacitor into the holes labeled C2. Make sure the negative side (strip) is facing the bottom of the board as shown.

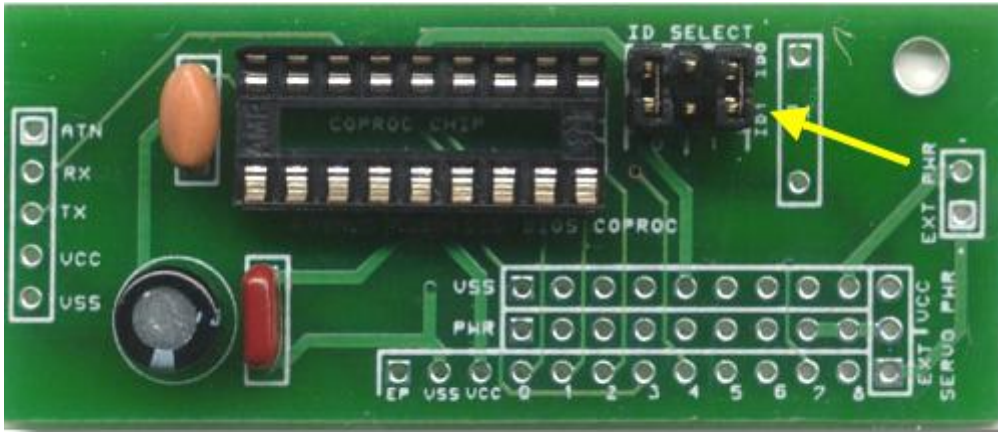
Solder in place and clip excess leads.



Step 4

Insert the 20MHz resonator as shown.

Solder in place.



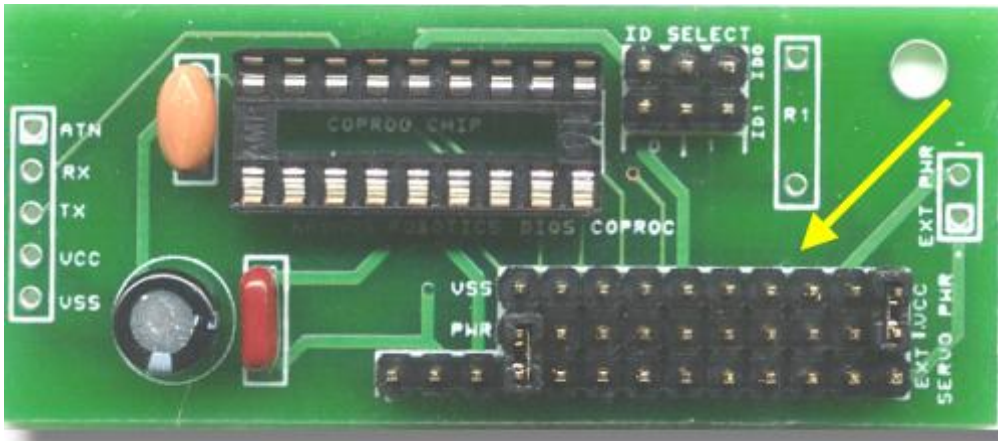
Step 5

Insert the 2, 3 pin headers as shown. Use a couple jumpers to hold in place.

Note the side with the small pins ends is inserted into the board from the top.

Solder in place.

Remove jumpers

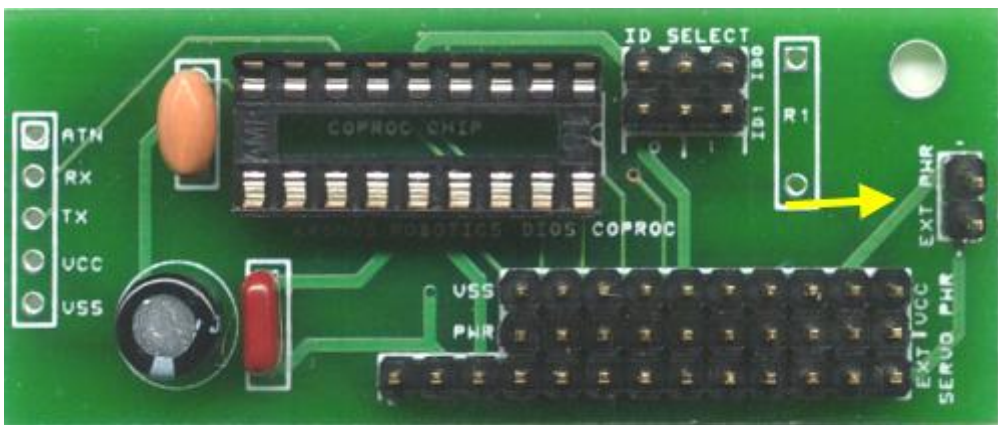


Step 6

I nsert the 2, 10 pin and single 13 pin headers as shown. Use a couple of the jumpers to hold in place.

Solder in place

Remove Jumpers

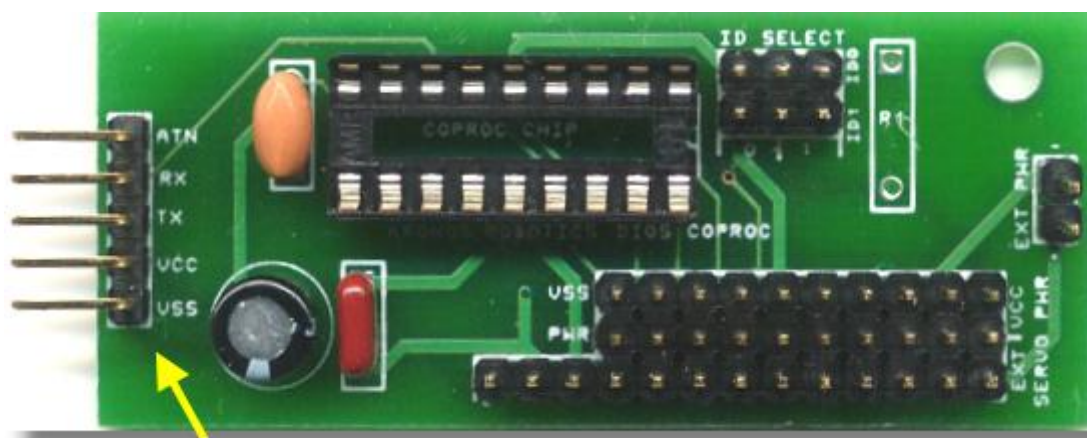


Step 7

Insert the 2 pin header as shown.

Solder in place

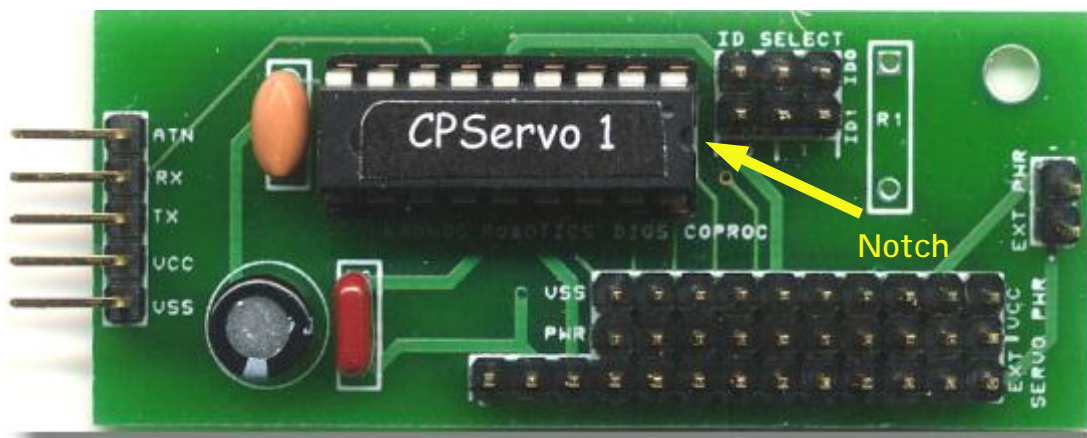
This connector is one that could be used to connect external Servo Power.



Step 8

Insert the 5 pin right angle header as shown.

Solder in place.



Step 9

Insert the CPServo 1 chip as shown. Make sure the notch in the chip is facing right.

Note

R1 is not used. Do not solder the included 10k resistor into this slot.

Hookup

The coprocs are identified by two parameters, Type and ID. The Type on this coproc is 1. The ID can be changed by setting the ID jumpers as shown below.



ID 0



ID 1



ID 2



ID 3



The servos can get there power from the VCC lead on the bus. However there are times when you don't want the logic (vcc) power and servo power to be the same. You use the servo power selector to determine where the power to the server is sourced. If you select the Vcc setting the power comes from the bus. If you select EXT it comes from the two pin Ex Pwr connector.

It is recommended that you use external servo power if you are using more than a couple of servos.

Never connect servos with power on.



The default speed of the Servo Coproc is 115200 baud. This can be changed via software.

You can change the baud speed at any time but the new speed will not take affect until the coproc has been reset. (power off then on)

If at any time you loose track of what baud rate you set the coproc or you have a device that cant communicate at the set speed you may force the coproc to a speed of 9600. You do this by powering down the coproc and inserting a jumper across ports 7 and 8 as shown. When you power the coproc the unit will be forced to 9600 baud. The coproc will stay at 9600 until the jumper is removed and the coproc has been reset. Once reset it resume whatever baud speed you have selected with the eewrite command.

Note that while in the forced baud speed mode servo ports 7 and 8 have will not be active.

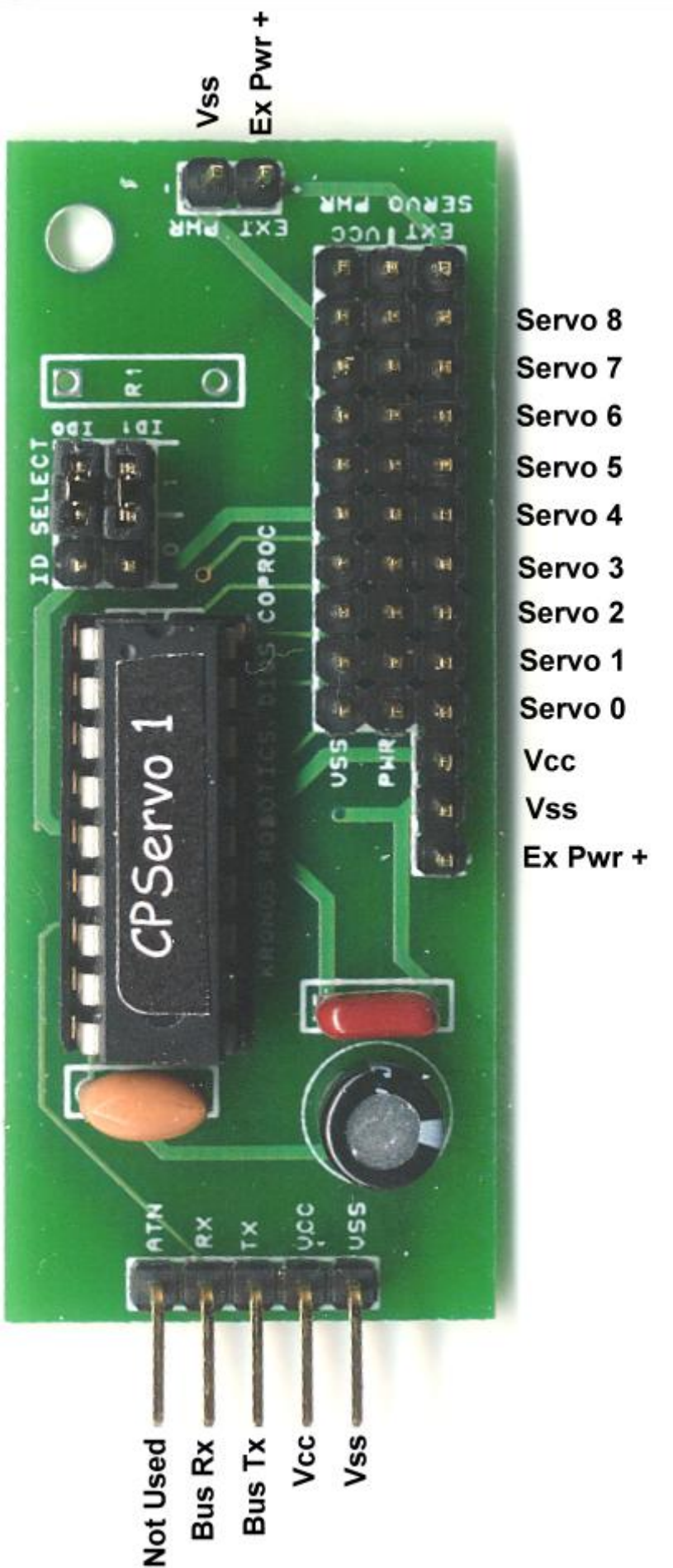
Warning: When connecting servos to the Coproc make sure power is off when they are first connected. The EMP can damage your microcontroller.

Hookup Notes

Coprocs can be connected to a common 5 pin bus or directly to the Dios or other microprocessor. If not connected to the Kronos Robotics minibus the transmit bus lead must be held high with a 10k resistor. This need only be done if you wish to read data from the coprocessor. Note that this is not the resistor marked R1 on the Coproc. This resistor must be connected externally on the connector. IE tie a 10K resistor (included) between the Tx lead and the VCC lead. (This is an external connection only. Don't use R1)

The Kronos Robotics minibus already has holding resistors on board.



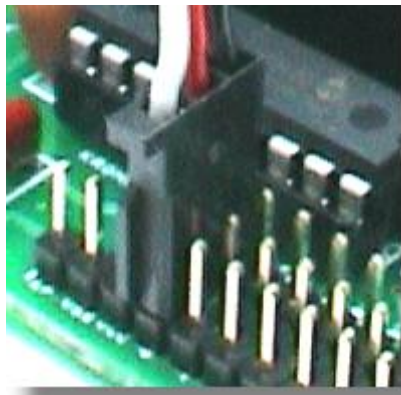


Servo Connection

To connect a servo just plug it into the header as shown. The header is compatible with Hitec, Futaba and other servos.

The Data pin should be on the outside. Servo Power in the middle and Ground on the inside.

Check the manufacture of your servos documentation.



Ex Servo power if enabled can be supplied from the top or side connection. Both are electrically the same.

If connecting as a standalone processor you will have to hold the bus Tx lead high with a 10k resistor.

Just connect the resistor between Vcc and bus Tx on the header. If connecting multiple coprocs the resistor should only be placed on one of them.

If you are using the minibus the resistors are already located on the bus.

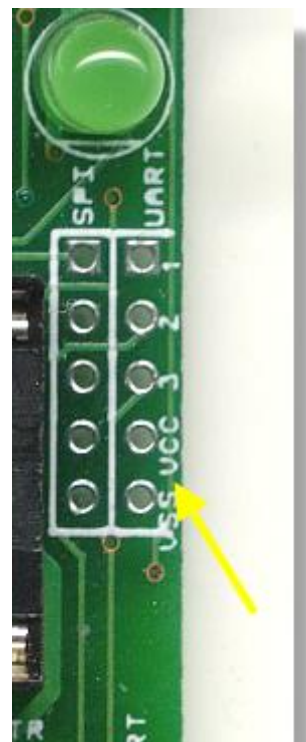


When connecting to the Dinos make the following connections:

- Bus Rx—I O port 8
- Bus Tx—I O port 9
- Vcc—Dinos Vcc
- Vss—Dinos Vss

The Dinos Ultra has a small header pad located on the right side next to the power LED. With the minibus kit a small header can be placed here to allow quick connections to the coproc bus.

You can also place your own male or female header here if you like.



Protocol Definition

The coprocs use a variable length packet. It can be anywhere from 2 to 9 bytes in length. All coprocs on the bus monitor and store the packet. Once all bytes from a packet are received each coproc on the bus will check the sent Type and ID against its own Type and ID. If there is no match the coproc will discard the packet. If the Type and ID do match the coproc will pass the data to its command processor and act on the data sent to it. There is one exception to this rule.

Type 0 : Targets all Types and IDs (every thing on bus will act on data)

Byte Definition

Byte 0

Bits 0-1 ID

Bits 2-7 Type

We refer to the Type and ID as an address pair. You will see it represented as (Type:ID) or (x:x). At times I will refer to this pair as AP (address pair)

Byte 1

Bits 0-4 Cmd

Bits 5-7 Packet Length

These Cmd and packet length are closely related so many times they will be referred to as command pair or just command.

Bytes 2—8

The actual number of bytes needed will depend on the command sent. IE its packet length.

IE

Byte2 Length = 1

Byte3 Length = 2

Byte4 Length = 3

Byte5 Length = 4

Byte6 Length = 5

Byte7 Length = 6

Byte8 Length = 7

Examples

0,0 Tells all coprocs connected to turn off transmitters

5,129,99,200,30,15 Tells Coproc (1:1) to write value 15 to address 30 of its eeprom

5,34,30 Tells coproc (1:1) to return the value of address location 30 of its eeprom

Lets break this one down

Byte 0 (5)

00000101 ID = 1

00000101 Type = 1 or (4 in actual byte. Its shifted 2 bits)

Byte 1 (34)

10000010 Cmd = 2

10000010 Packet Length = 4 (128 in actual byte. Its shifted 5 bits. Add 4 more bytes to packet)

Byte2 (30)

Address to read

Command Definition

Protocol Specific Commands

Cmd 0

AP,0
Turn off Transmitter

Cmd 1 (129)

AP,129,200,99,addr,data
Write data to internal eeprom.

Byte 2 must be 99
Byte 3 must be 200
Byte 4 Eeprom address to write to
Byte 5 Data to write

There are 128 eeprom slots available. Slots 0-9 are reserved for protocol specific parameters such as baud rate and internal timing.

The remaining slots are coprocessor specific.

You must have a small pause after writing to a eeprom address. I E the buss must be quiet for 50ms after a write as all internal interrupts are stopped during the write.

Cmd 2 (34)

AP,34,addr
Requests a the coproc to transmit the contents of eeprom address.

Cmd 3 (3)

AP,3
Tells the coproc to load its default values from eeprom.

Cmd 4 (4)

AP,4
Request version number from coproc.

Coproc Specific commands

Set Servo Value

Cmd 11-20 (75-83)

AP,75,valueH,valueL

Sets the servo timing value of the servo. The values are in .2us units. A value of 1.5ms would be 7500. A value of 1ms would be 5000 and a value of 2ms would be 10000.

You must convert these values to high and low bytes to pass the Data. The Dios CPServo library will do this automatically for you.

Example

4,75,29,76 This tells CPServo ID 0 Servo 0 to set to 1.5ms or a value of 7500.

4,76,29,76 This tells CPServo ID 0 Servo 1 to set to 1.5ms or a value of 7500.

If no other coprocs were on the buss you could use the following to do the same thing

0,75,29,76 This tells CPServo ID 0 Servo 0 to set to 1.5ms or a value of 7500.

0,76,29,76 This tells CPServo ID 0 Servo 1 to set to 1.5ms or a value of 7500.

Remember

75=Servo0

76=Servo1

77=Servo2

78=Servo3

79=Servo4

80=Servo5

81=Servo6

82=Servo7

83=Servo8

Set Servo Count

Cmd 20 (52)

AP,52,count

Sets the number of servos to process. Count must be in the range of 1-9. It is possible to increase the refresh rate by dropping a couple of servos from the coproc. Please note that not all servos can handle a increased refresh rate.

Example

4,52,7 Sets CPServo ID 0 to use only 7 Servos.

Set Servo State

Cmd 21-29 (53-61)

AP,53-61,state

Turns the Servo on or off. When set to a state of 0 (off) pulses are no longer sent to that servo. Some servos react differently to this. Most will hold there last position but will no longer be energized.

State must be 0 or 1

Example

4,53,0 This tells CPServo ID 0 Servo 0 to shut down.

4,54,0 This tells CPServo ID 0 Servo 1 to shut down.

If no other coprocs were on the buss you could use the following to do the same thing

0,53,0 This tells Servo 0 to shut down.

0,54,0 This tells Servo 1 to shut down.

Remember

53=Servo0

54=Servo1

55=Servo2

56=Servo3

57=Servo4

58=Servo5

59=Servo6

60=Servo7

61=Servo8

EEprom Values

(Protocol Specific EEprom Values)

Byte 0

Used for testing should be a value of 255

Byte 1

Version

Byte 2

Type

Byte 3

Reserved

Byte 4

Idlecount.

When no packet data is received an internal packet counter is incremented. When that counter overflows from 255 to 0 the packet byte count is automatically reset to 0. It is an auto packet re-sync. In other words if you stop transmitting packets to the coproc bus all the coprocs will auto re-sync. The Idlecount value is the starting value for this counter. The default value is 254. (very fast). Normally this should never have to be changed as there are quite a few error recovery routines built into the coprocs.

Byte5

Txwaithigh

Byte6

Txwaitlow

These bytes make up a transmit delay for sending requested bytes. When a command is received that requests information a small delay is inserted. This delay is in microsecond and the default is 1us. If you are using a slower processor you may need to change this value

Byte7

Boot baud index

Default is 10 (115200 baud)

Valid Values

129=9600

64=19200

32=38400

21=57600

10=115200

4=250000

1=625000

0=1250000

Byte8

TxIdlecount

When no packet data is received or transmitted an internal packet counter is incremented. When that counter overflows from 255 to 0 the transmitter will be taken off line. The TxIdlecount is the starting value for this counter.

The default is 254 (very fast)

Byte 9

Reserved

(Coproc Specific EEprom Values)

Byte 10

Servo0H

Byte 11

Servo0L

Default Servo Value for Servo 0. Default is value is 7500

Byte 12

Servo1H

Byte 13

Servo1L

Default Servo Value for Servo 1. Default is value is 7500

Byte 14

Servo2H

Byte 15

Servo2L

Default Servo Value for Servo 2. Default is value is 7500

Byte 16

Servo3H

Byte 17

Servo3L

Default Servo Value for Servo 3. Default is value is 7500

Byte 18

Servo4H

Byte 19

Servo4L

Default Servo Value for Servo 4. Default is value is 7500

Byte 20

Servo5H

Byte 21

Servo5L

Default Servo Value for Servo 5. Default is value is 7500

Byte 22

Servo6H

Byte 23

Servo6L

Default Servo Value for Servo 6. Default is value is 7500

Byte 24

Servo7H

Byte 25

Servo7L

Default Servo Value for Servo 7. Default is value is 7500

Byte 26

Servo8H

Byte 27

Servo8L

Default Servo Value for Servo 8. Default is value is 7500

Byte 28

Servo count

Sets the boot up servo count. See Cmd 20

Coproc Specific EEprom values (cont)

Byte 29

Internal Timer Prescale value

The default value of the internal timer prescale is 0. By changing this you can effectively lower the resolution of the servo settings. You can use this to help you match the CPServo to other servo controllers. However this must be considered an advanced option.

0: set resolution to .2us per unit (default)

1: set resolution to .4us per unit

2: set resolution to .8us per unit

3: set resolution to 1.6us per unit

Byte 31

FillinH

Byte 32

FillinL

After a servo pulse has been serviced (on then off) the Fillin time is used to calculate how much time the servo should be placed in the low state. The actual time in this state is based on the pulse time. The formula is $\text{waittime} = \text{Fillin} - \text{pulsetime}$. If the pulse time is greater than the fillin time no wait time is added.

Why do all this. This allows us to maintain a persistent refresh rate regardless of all the 9 servo settings. By increasing the fillin time you decrease the overall refresh rate.

Lets say you wanted a refresh rate of 15 milliseconds. This is not possible when processing all 9 servos. But by setting the number of servos proceed to 6 and the fillinrate to 12488 will create a refresh rate very close to 15ms

Some things to keep in mind if you decide to modify this parameter. You can burn up a servo if you send a refresh rate higher than the servo can handle. This defacto refresh rate for most servos is 20ms.

Don't mess with this parameter if you don't understand how servos work.

Dios Coproc Basic Library

CPinit(*baud*)

This function starts the CP engine

baud This optional command sets the baud rate. If omitted a baud rate of 115200 will be used.

CPversion(*type*,*id*)

This function returns the current version of the given coproc.

type The target coproc type 1-63. If 0 is used all types on buss are the target.

id The target coproc id 0-3. If *type* is 0 then this field is ignored.

CPTxoff(*type*,*id*)

This function turns off the target coprocs transmitter.

type The target coproc type 1-63. If 0 is used all types on buss are the target.

id The target coproc id 0-3. If *type* is 0 then this field is ignored.

CPloaddefaults(*type*,*id*)

The target coproc loads its eeprom defaults.

type The target coproc type 1-63. If 0 is used all types on buss are the target.

id The target coproc id 0-3. If *type* is 0 then this field is ignored.

CPwriteEEProm(*type*,*id*,*address*,*value*)

Write a value to the eeprom of the target coproc.

type The target coproc type 1-63. If 0 is used all types on buss are the target.

id The target coproc id 0-3. If *type* is 0 then this field is ignored.

address The address of the eeprom to write to. 0-128

value The value to write. 0-255

CPreadEEProm(*type*,*id*,*address*)

Reads the value of an address point of the target eeprom.

type The target coproc type 1-63. If 0 is used all types on buss are the target.

id The target coproc id 0-3. If *type* is 0 then this field is ignored.

address The address to read.

CPsendbytes(*type*,*id*,*data*,*data*,.....)

Sends up to 9 bytes to the target coproc.

type The target coproc type 1-63. If 0 is used all types on buss are the target.

id The target coproc id 0-3. If *type* is 0 then this field is ignored.

data These are the data bytes to send. Just include the bytes needed the proper protocol values will be set for

Dios CPServo Library

CPinit(*baud*)

This function starts the CP engine

baud This optional command sets the baud rate. If omitted a baud rate of 115200 will be used.

CPServo(id, servo, value)

This function sets a servos value on the target CPServo coproc.

id The target coproc id 0-3. If type is 0 then this field is ignored.

servo The servo number 0-8 that you want to write to.

value The Position value to send to the servo. 0-65535

CPServoState(id, servo, state)

This function will turn off a servo.

id The target coproc id 0-3. If type is 0 then this field is ignored.

servo The servo number 0-8 that you want to write to.

state The Position value to send to the servo. 0 or 1

Note: If you turn a servo off it will stay off until it is turned back on or the coproc is powered down and back on.

All Coproc libraries are always included with the Dios Software. Dios libraries always include help files.

Links

Be sure to visit the Kronos Robotics web site for more information and updates and example code.

Web Site

<http://www.kronosrobotics.com>

Full color assembly instructions

<http://www.kronosrobotics.com/products/pdfs/pdfs.htm>