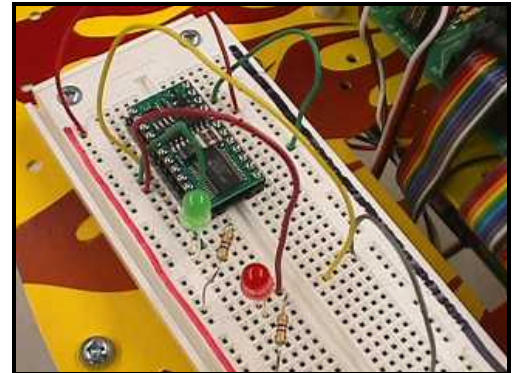


This application note explains how to add a coprocessor to ARobot. A coprocessor is one of the most powerful expansions that can be done because it can remove time consuming tasks from the master processor, leaving it for higher level routines. The coprocessor could be used to add a ring of sonar range finders, read a time consuming compass or GPS, process video images, etc. ARobot's controller board already has one coprocessor that handles DC motor and RC servo motor control. A single pin is used for bidirectional communications between the master and slaves. Any microprocessor, embedded PC, or computer can be used as a coprocessor. We will use the Basic Stamp II as an example coprocessor. It is probably the easiest coprocessor to implement because it doesn't need external circuitry to attach it to the robot. It can also be programmed on the ARobot controller board which reduces the cost of buying a microprocessor programmer. Also, the user will already know how to program the coprocessor using PBasic. The new coprocessor and associated circuitry can be mounted on a breadboard like the picture shows, or soldered to a perf board.



In this example, we'll use a Basic Stamp II as a slave coprocessor which will be used to control 2 LEDs. It is assumed that you have mounted a breadboard to ARobot, that you have made an expansion cable, and that you understand how to use these parts. If you are creating a circuit board (i.e. soldering, wire wrapping, etc) we advise that you place the coprocessor in a socket so that it may be removed for programming later.

Parts List

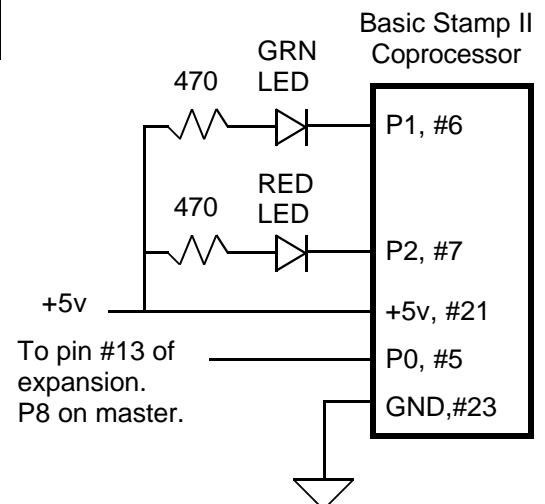
These parts can be found at Radio Shack <http://www.radioshack.com>, Mouser Electronics <http://www.mouser.com> or Digi-Key <http://www.digi-key.com>.

Part	Part #	Price
Basic Stamp II		\$49.00
470 ohm (2)	RS # 271-1317	\$0.49
LEDs (2)	RS # 276-1622	\$2.29

Circuit

A single pin on the master Basic Stamp II which resides on ARobot's controller board is used to communicate with the slave coprocessor Basic Stamp II. This communication signal, ground, and +5v power will be taken from the expansion connector. The coprocessor will control a red and green LED.

NOTE: The LED will not work if it is connected incorrectly. However, it will not damage the LED to hook it up backwards.



Software

There are two programs that need to be created for this example to work. One program goes on the master Basic Stamp II which is on ARobot's controller board, and the other goes on the slave Basic Stamp II coprocessor. This software is available for download on our web site at <http://www.robotics.com/arobot>

```
'co.bs2      Arrick Robotics  www.robotics.com
'
'This routine controls the coprocessor.  It accepts
'commands from the master and controls two LEDs.

charn  var    byte    'input variable.
chart  var    byte    'output variable.
grn    con    1       'led pins.
red    con    2
baud   con    396     'baud rate.
net    con    0       'network I/O pin.
conid  con    "2"     'controller ID.

low    grn     'turn on green led to signify power.

main
  gosub  getchar
  if charn <> "!" then main 'loop if not !.

  gosub  getchar      'get next char.
  if charn <> conid then main  'check conid.

  gosub  getchar
  if charn = "R" then redled 'check for "R".
  if charn = "G" then grnled 'check for "L".

mainb
  pause  100          'wait 100 ms.
  chart="B"
  gosub  putchar
  goto   main         'loop forever.

maina
  chart="A"          'A=acknowledge.
  gosub  putchar
  goto   main         'loop forever.

redled
  gosub  getchar      'get next char.
  if charn = "1" then redled1 '1=on.
  if charn = "0" then redled0 '0=off.
  goto  mainb        'bad char?

redled1
  low red            'turn on red led.
  goto  maina       'done.

redled0
  high red           'turn off red led.
  goto  maina       'done.

grnled
  gosub  getchar      'get next char.
  if charn = "1" then grnled1 '1=on.
  if charn = "0" then grnled0 '0=off.
  goto  mainb        'bad char?

grnled1
  low grn            'turn on red led.
  goto  maina       'done.
```

```
grnled0
  high grn           'turn off red led.
  goto  maina       'done.

getchar
  serin net,baud,[charn] 'input a char.
  return

putchar
  serout net,baud,[chart] 'output a char.
  return
```

Modifications

It may become necessary to add a few more parameters to the serin and serout statements on the master processor and the slave coprocessors. You might want to check for timeout on the master processor. If a response is not received from a slave processor then a timeout will allow you to act upon a non-response from the slave (i.e. re-transmit the command). Pace can also be added to ensure that the coprocessors receive the commands. This is a value that allows the processor to wait a short time before sending the next character. The serin and serout commands are well documented in the BSII manual from Parallax. **THESE MODIFICATIONS ARE NOT NECESSARY TO RUN THE ABOVE SOFTWARE.** If the above software has been tested on working hardware.



```
'cotest.bs2 Arrick Robotics www.robotics.com
'
'This program runs on the master and sends LED
'control commands to the coprocessor.

charn var byte 'input variable.
chart var byte 'output variable.
baud con 396 'baud rate.
net con 8 'network I/O pin.

low 10 'ARobot Red LED off.
low 9 'Speaker off.

main
'Send !2R1 - Red LED on.
pause 500 'short delay.
chart="!"
serout net,baud,[chart] 'output the char.
pause 10
chart="2"
serout net,baud,[chart] 'output the char.
pause 10
chart="R"
serout net,baud,[chart] 'output the char.
pause 10
chart="1"
serout net,baud,[chart] 'output the char.
serin net,baud,[charn] 'input the char.

debug charn 'Display char.

'Send !2R0 - Red LED off.
pause 500 'short delay.
chart="!"
serout net,baud,[chart] 'output the char.
pause 10
chart="2"
serout net,baud,[chart] 'output the char.
pause 10
chart="R"
serout net,baud,[chart] 'output the char.
pause 10
chart="0"
serout net,baud,[chart] 'output the char.
serin net,baud,[charn] 'input the char.

debug charn 'Display char.

goto main
```

The coprocessor can be inserted into the ARobot controller board, programmed, then put on the breadboard. Then the master processor can be inserted into the ARobot board and programmed. The CONID constant can be changed to access multiple coprocessors. A "1" for CONID can not be used because it is used for the coprocessor on ARobot's controller board that runs the servos, DC motor, and counts encoder increments.

The new slave coprocessor will now accept 2 commands. Each command returns "A" to signify that the command has been acknowledged. The two commands are as follows:

```
"!2Rx"
"!2Gx"
```

The "!" gets the attention of every coprocessor. The "2" tells coprocessor 2 that it is the receiver of the command. The "R" or "G" is the command. In this case "R" stands for red LED and "G" stands for green LED. Commands can be more than one character in length (i.e. !2GD could mean Get Data). The "x" is an input variable. In this case "x" can be either a "1" or a "0". "1" means turn on the LED. "0" means turn it off.

NOTE: The quotation marks are used to signify an ascii character. This means that the ascii character "R" has a numerical value(in hex) of 52 hex. The ASCII character "5" has the value 35 hex.

See our web site at:

<http://www.robotics.com/arobot>

for additional application information.



www.robotics.com