RCM-BFin C Interpreter - as of 7 May 2010

The new RCM-BFin C interpreter is based on the picoC open source C interpreter. Full documentation for picoC is in the works, but for now, we are providing some code samples that highlight capabilities.

```
/* comments */
printf("Hello\n"); /* this is a comment */
printf("Hello\n"); // this is also a comment

/* printf */
int Count;
for (Count = -5; Count <= 5; Count++)
  printf("Count = %d\n", Count);

printf("String 'hello', 'there' is '%s', '%s'\n", "hello", "there");
printf("Character 'A' is '%c'\n", 65);
printf("Character 'a' is '%c'\n", 'a');

/* structs */
struct fred
{
  int boris;
  int natasha;
};
struct fred bloggs;

bloggs.boris = 12;
bloggs.natasha = 34;
printf("%d\n", bloggs.boris);
printf("%d\n", bloggs.natasha);

/* array */
int Count;
int Array[10];
for (Count = 1; Count <= 10; Count++)
  Array[Count-1] = Count * Count;
for (Count = 0; Count < 10; Count++)
  printf("%d\n", Array[Count]);

/* switch */
int Count;
for (Count = 0; Count < 4; Count++)
{
  printf("%d\n", Count);
  switch (Count)
  {
    case 1:
      printf("%d\n", 1);
      break;
     case 2:
      printf("%d\n", 2);
      break;
    default:
      printf("%d\n", 0);
      break;
  }
}

/* while + do while */
int a;
int p;
int t;
a = 1;
```

```
p = 0;
t = 0;
while (a < 100)
{
  printf("%d\n", a);
  t = a;
  a = t + p;
  p = t;
}

do
{
  printf("%d\n", a);
  t = a;
  a = t + p;
  p = t;
} while (a < 100);

/* pointer */
int a;
int *b;
int c;

a = 42;
b = &a;
printf("a = %d\n", *b);

struct ziggy
{
  int a;
  int b;
  int c;
} bolshevic;

bolshevic.a = 12;
bolshevic.b = 34;
bolshevic.c = 56;

printf("bolshevic.a = %d\n", bolshevic.a);
printf("bolshevic.b = %d\n", bolshevic.b);
printf("bolshevic.c = %d\n", bolshevic.c);

struct ziggy *tsar = &bolshevic;

printf("tsar->a = %d\n", tsar->a);
printf("tsar->b = %d\n", tsar->b);
printf("tsar->c = %d\n", tsar->c);

/* #define */
#define FRED 12
#define BLOGGS(x) (12*(x))

printf("%d\n", FRED);

/* integers */
int a = 24680;
int b = 01234567;
int c = 0x2468ac;
int d = 0x2468AC;
int e = 0b010101010101;

printf("%d\n", a);
printf("%d\n", b);
printf("%d\n", c);
printf("%d\n", d);
```

```
/* if */
int a = 1;
if (a)
  printf("a is true\n");
else
  printf("a is false\n");

int b = 0;
if (b)
  printf("b is true\n");
else
  printf("b is false\n");
```

```
/* recursion */
int factorial(int i)
{
  if (i < 2)
    return i;
  else
    return (i * factorial(i - 1));
}

int Count;
for (Count = 1; Count <= 10; Count++)
  printf("%d\n", factorial(Count));
```

```
/* nesting */
int x, y, z;
for (x = 0; x < 2; x++)
{
  for (y = 0; y < 3; y++)
  {
    for (z = 0; z < 3; z++)
    {
      printf("%d %d %d\n", x, y, z);
    }
  }
}
```

/* Robot Functions */

- void autorun(int seconds): leave picoC if ESC character is received in (int) seconds,
  - e.g. autorun(5);
  - only used at beginning of C program
- int abs(int data): returns absolute value of int data
- int acos(int adjacent, int hypotenuse): arccos(adjacent, hypotenuse)
- int analog(int channel): read AD7998 8-channel 12-bit A/D
  - channels 1-8 correspond to i2c device 0x20
  - channels 11-18 correspond to i2c device 0x23
  - channels 21-28 correspond to i2c device 0x24
- int analogx(int channel): read analog channel from SRV-4WD
  - channel 0 = battery
  - channel 1 = 5V gyro
  - channel 2 = 3.3V gyro
  - channel 3 = IR 1
  - channel 4 = IR 2
  - channel 6 = IR 3
  - channel 7 = IR 4

- int asin(int opposite, int hypotenuse): arcsin(opposite, hypotenuse)
- int atan(int opposite, int adjacent): arctan(opposite, adjacent)
- int battery(): check SVS battery detector ... 1=okay, 0=low battery
- int compass(): read HMC6352 compass
- int compassx(): read HMC5843 compass on SRV-NAV
  - note that min/max calibration data is accessible as cxmin, cxmax, cymin, cymax
- void compassxcal(xmin, xmax, ymin, ymax, continuous_calibration): set calibration data for HMC5843
  - use $c console command to gather data
  - continuous_calibration flag determines whether compassx() function continues to collect calibration data. continuous_calibration flag: off = 0, on = 1
  - this function is useful for scripting an auto-calibration routine - see test4wd.c
- int cos(int angle): cos(angle) * 1000
- void delay(int milliseconds): delay xxx milliseconds
- void encoders(): compute pulses/second from wheel encoders
  - data returned in globals lcount and rcount
- int encoderx(channel): read cummulative pulse count from specific motor encoder 1-4 on SRV-4WD
  - count cycles after 65535 pulses
  - depends on wheel size, but 1000 pulses on 4.5" wheel is approximately 1 foot of travel
- void exit(): leave picoC on completion of stored program, bypassing the interactive mode
- void gps(): parse $GPGGA string from gps
  - data returned in globals gpslat, gpslon, gpsalt, gpsfix, gpssat, gpsutc
  36.5deg is represented as 36500000
  100.5W deg is represented as -100500000
- int gps_dist(int lat1, int lon1, int lat2, int lon2): compute distance in meters between two gps coordinates
  format of coordinates is deg*1000000
  36.5deg is represented as 36500000
  100.5W deg is represented as -100500000
- int gps_head(int lat1, int lon1, int lat2, int lon2): compute heading in degrees between two gps coordinates (N == 0-deg)
  36.5deg is represented as 36500000
  100.5W deg is represented as -100500000
- void init_uart1(int baudrate): initializes 2nd Blackfin UART
- int input(): return single character from read of serial channel (same as getch())
- int input1(): return single character from read of uart1
- void iodir(int iopins): set GPIO-H15/14/13/12/11/10 as inputs or outputs
  0 = input, 1 = output
  iodir(0x31) == H15-out H14-out H13-in H12-in H11-in H10-out
  iodir(0b110001) == H15-out H14-out H13-in H12-in H11-in H10-out
  iodir(0x03) == H15-in H14-in H13-in H12-in H11-out H10-out
- int ioread(): read GPIO pins H15-H10
  if H15=1 H14=1 H13=0 H12=0 H11=0 H10=0,
  ioread() would return 48 == 0x30

- void iowrite(int iopins): set GPIO pins H15-H10
  iowrite(0x31) or iowrite(0b110001) would set
  H15=1 H14=1 H13=0 H12=0 H11=0 H10=1
- void laser(int which_laser): 0=off, 1=left, 2=right, 3=both
- void motors(int left, int right): set left and right PWM motor power -100 to 100
- void motors2(int left, int right): set left and right PWM2 motor power (tmr6/7) -100 to 100
- void motorx(int left, int right): set left and right SRV-4WD motor power -100 to 100
- void nninit(): initialize neural net
- void nnlearnblob(int pattern_number): scale and save blob to 8x8 pattern
- int nnmatchblob(int blob_number): see which pattern is best match to selected blob
  neuron output values are found in neuron[]
  return value is index to best match
- void nnset(int first8bits, int second8bits, int ..., int, int, int, int, int, int): set nn pattern
- void nnshow(int which_pattern): display nn pattern
- int nntest(int first8bits, int second8bits, int ..., int, int, int, int, int): test nn pattern
  neuron output values are found in neuron[]
  return value is index to best match
- void nntrain(): train neural net
- void output(int): output a single character to serial channel (uart0)
- void output1(int): output a single character to uart1
- int peek(int address, int wordsize): int x = peek(addr, size) where size = 1, 2, 4 bytes
  - byte/short/word alignment is forced
- void poke(int address, int wordsize, int value): poke(addr, size, val) where size = 1, 2, 4 bytes
  - byte/short/word alignment is forced
- int rand(int number_range): return random number ranging from 0 to xxx
- int range(): use laser pointer to estimate range
- int read_int(): reads an integer from the console - terminates on anything but '-' or '0'-'9'
- int read_str(char *): reads a string from the console into character array and returns number of chars read. terminates on receipt of 0x00 or 0x01, or if read count exceeds 1023.
- int readi2c(int device, int register): read byte from I2C port
- int readi2c2(int device, int register): read two bytes from I2C port
- int readi2c3(int device, int register): read three bytes from I2C port
- int readi2crs(int device, int register): read byte from I2C port using repeated start
- int readi2c2rs(int device, int register): read two bytes from I2C port using repeated start
- int readi2c3rs(int device, int register): read three bytes from I2C port using repeated start
- void servos(int timer2, int timer3): set pin 7/8 (tmr 2/3) PPM levels 0 to 100
- void servos2(int timer6, int timer7): set pin 5/6 (tmr 6/7) PPM levels 0 to 100

- int signal(): non-blocking check for input on serial channel - non-zero return indicates an input
- int signal1(): non-blocking check for input on second serial channel (uart1) - non-zero return indicates an input
- int sin(int angle): sin(angle) * 1000
- int sonar(int which_channel): ping modules 1, 2, 3 or 4
- int sqrt(int value): compute integer square root
- int tan(int angle): tan(angle) * 1000
- int tilt(int axis): return tilt sensor reading from channel 1 (x axis), 2 (y axis) or 3 (z axis)
  - int x = tilt(1); int y = tilt(2); int z = tilt(3);
- int time(): return time in milliseconds since startup
- int vblob(int color_bin, int which_blob): blob search on color xxx - returns number of blobs found
  - 'int' return value indicates how many matching blobs were found
  - 2nd value determines which blob (largest to smallest)
  - data returned in globals blobcnt, blobx1, blobx2, bloby1, bloby2
- void vcam(int settings): enable/disable automatic gain, white balance and exposure camera functions (default is 7)
  vcam(4) -> AGC enable
  vcam(2) -> AWB enable
  vcam(1) -> AEC enable
  vcam(7) -> AGC+AWB+AEC on
  vcam(0) -> AGC+AWB+AEC off
- void vcap(): capture video frame
- void vcolor(int color_bin, int ymin, int ymax, int umin, int umax, int vmin, int vmax): set color bin with
  ymin, ymax, umin, umax, vmin, vmax
- void vdiff(int flag): enable/disable differencing with vcap()
  - vdiff(1) enables / vdiff(0) disables
- int vfind(int color, int x1, int x2, int y1, int y2):
  - count number of pixels in color bin
  in range of x1 -> x2, y1 -> y2
- int vjpeg(int quality): compress image captured by vcap(). use vsend() to transmit:
  ```
  int size = vjpeg(int quality);
  vsend(size);
  ```
  returned value is size of jpeg image. input parameter is quality of jpeg image (1-8, 1 = highest quality)
- void vmean(): get YUV means over full frame
  - data returned in globals y1, u1, v1
- void vpix(int x, int y): get YUV values of vpix(x, y)
  - data returned in globals y1, u1, v1
  - vpix(0,0) reads the pixel from the upper left corner of the image
- void vrcap(): capture reference frame for differencing
- int vscan(int columns, int threshold): edge detect function
  - counts edge pixels and divides image into columns
  - columns range from 1-9, threshold ranges from 0001-9999 (4000 is good starting point)
  - returns the distance from bottom of the image to first edge pixel in each column - results found in scanvect[] array, e.g.
  ```
  int ii;
  ```

```
vcap();
vscan(3, 4000); // search 3 columns, set threshold to 4000
for (ii=0; ii<3; ii++)
      printf("column %d distance %d\r\n", ii, scanvect[ii]);
```

- void vsend(int size): send JPEG image that was captured and compressed using vcap() and vjpeg()
- void writei2c(int device, int register, int value): write one byte to I2C port
- void writei2c2(int device, int register, int value): write two bytes to I2C port
- void writei2c3(int device, int register, int value): write three bytes to I2C port
- void writei2c4(int device, int register, int value): write four bytes to I2C port

last updated 10 December 2017