

chipKIT-core 1.4.0

Now with Non-blocking analogRead()

By: Jacob Christ

Blocking vs Non-Blocking Functions

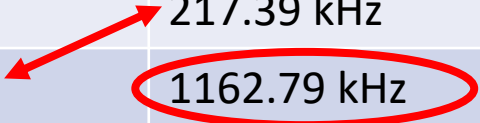
- A blocking function ties up CPU until the requested task is complete.
 - Usually this means lots of wasted CPU cycles
- A non-blocking function, when called, returns a value that lets the calling program know if the requested task has completed.
 - If the task is complete then we can act on the results

Performance (best case)

| chipKIT Board | PIC23 | Clock Freq | loop() Frequency Blocking Analog Read | loop() Frequency Non-Blocking Analog Read | Improvement |
|---------------|-------|------------|---|---|-------------|
| Lenny | MX | 40MHz | 49.14 kHz | 127.39 kHz | 2.59 |
| FubranioSD | MX | 80MHz | 90.91 kHz | 217.39 kHz | 2.39 |
| WiFire | MZEFG | 200MHz | 233.10 kHz | 1162.79 kHz | 4.98 |

Performance (best case)

| chipKIT Board | PIC23 | Clock Freq | loop() Frequency Blocking Analog Read | loop() Frequency Non-Blocking Analog Read | Improvement |
|---------------|-------|------------|---|---|-------------|
| Lenny | MX | 40MHz | 49.14 kHz | 127.39 kHz | 2.59 |
| FubranioSD | MX | 80MHz | 90.91 kHz | 217.39 kHz | 2.39 |
| WiFire | MZEFG | 200MHz | 233.10 kHz | 1162.79 kHz | 4.98 |



- 80MHz PIC32 loop() with non-blocking reads almost as fast as 200MHz PIC32 loop() with blocking reads.
- What can you do with, 1.162MHz loop() speed
 - How about running 20 50kHz PID loops?

How does the code change?

Arduino style blocking AnalogRead

```
#define ADC_TEST_PIN 16
```

```
void setup() {
```

```
}
```

```
void loop() {
```

```
    uint32_t value;
```

```
    value = analogRead(ADC_TEST_PIN);
```

```
}
```

chipKIT-core style non-blocking AnalogRead

```
#define ADC_TEST_PIN 16

void setup() {
  analogReadConversionStart(ADC_TEST_PIN);
}

void loop() {
  uint32_t value;
  if ( analogReadConversionComplete() ) {
    value = analogReadConversion();
    analogReadConversionStart(ADC_TEST_PIN);
  }
}
```

And one more thing...

blocking reads still work in chipKIT-core too.

WARNING!

- Don't mix non-blocking and blocking `analogRead`'s unless you understand how the code works under the hood.
 - Doing so incorrectly could cause your board to hang.

Why not to mix blocking/non-blocking

- analogRead() is really just using the new non-blocking calls
- The non-blocking calls keep track of the currently converting channel in global variable.
- If you attempt to start a conversion on one channel and wait for it to complete on another you may get bad juju.
- The blocking analogRead() now looks like this:

```
int analogRead(uint8_t pin)
{
    analogReadConversionStart(pin);
    while( ! analogReadConversionComplete() );
    return analogReadConversion();
}
```

Find out more:

- [http://chipkit32.github.io/chipKIT-core/api analogread non blocking](http://chipkit32.github.io/chipKIT-core/api_analogread_non_blocking)
- <https://github.com/chipKIT32/chipKIT-core>
- <http://chipKIT.net>
- <mailto:jacob@pontech.com>