Raspberry Pi

Memory and I/O Throughput – A Study

https://github.com/ma16/rpio

2016-10-12 – Version 1.0 (Alpha)

This document was created with LibreOffice (Version 5.1.4.2 on Ubuntu 16.04.1 LTS).

The acompanied program is written in C++ and compiled with GCC by GNU make.

The test case execution and evaluation is performed by Perl scripts.

The 3D-bar images in this document were created in Perl with GD::Graph::bars3d by Jeremy Wadsack.

Motivation

The Raspberry Pi provides, besides others, several I/O pins, paired with a reasonable amount of memory (RAM) and a Linux operating system. Thus, data sampling becomes quite easy even for user-space applications. This projects tries to estimate an upper bound for the I/O throughput.

Table of Contents

1 Motivation

One of the interesting things about the Raspberry Pi[1] is its wide range of peripheral devices – which are often only found on micro-controllers – paired with a reasonable amount of available memory (RAM) and a Linux operating system. This enables the development of applications for data sampling as on any other Linux system.

This project tries to determine the maximum I/O throughput for the Raspberry Pi in order to estimate its fitness for data sampling. The maximum I/O throughput should be considered as upper bound for data sampling, rather than an actually achievable sampling rate. – In order to estimate the sampling rate, a data signal would be required. However, this project simply accesses peripheral registers and ignores the sampled values.

Following three peripheral registers are of interest within the scope of this project:

- The System Timer Counter's (§12) lower 32 bits (CLO) at bus address $7e00:3000_H$.
- The GPIO Pin Level 0 (§6, GPLEV0) at bus address $7e20:0034_H$.
- The GPIO Pin Event Detect Status 0 (§6, GPEDS0) at bus address $7e20:0040_H$.

Please refer to the section §6 and §12 in the “BCM2835 ARM Peripherals” document for details.[2]

Since data sampling involves storing the sampled data into RAM also memory throughput is measured.

1 https://www.raspberrypi.org/
2 https://www.raspberrypi.org/wp-content/uploads/2012/02/BCM2835-ARM-Peripherals.pdf

2 Classification

The tests can be categorized, depending on data source and destination, as follows:

0 : 1	*0* → *[dst]*		Store (constant) value at address *dst*.
0 : N	*0* → *(dst,dst+N]*	*i=0..N-1:*	Store (constant) value at address *dst*+*i*.
1 : 0	*[src]* → *0*		Fetch value at address *src*. Ignore value.
1 : 1	*[src]* → *[dst]*		Fetch value at address *src* and store at address *dst*.
1 : N	*N* × *[src]* → *(dst,dst+N]*	*i=0..N-1:*	Fetch value at address *src* and store at address *dst*+*i*.
N : 0	*(src,src+N]* → *0*	*i=0..N-1:*	Fetch value at address *src*+*i*. Ignore value.
N : 1	*(src,src+N]* → *N* × *[dst]*	*i=0..N-1:*	Fetch value at address *src*+*i* and store at address *dst*.
N : N	*(src,src+N]* → *(dst,dst+N]*	*i=0..N-1:*	Fetch value at address *src*+*i* and store at address *dst*+*i*.

All operations are aligned 32-bit word operations. An address may be the address of a peripheral register or a virtual RAM address in user space. A RAM address-range *(addr,addr+N]* is hereby called a buffer. If a RAM address is involved, this address is page-aligned. If a buffer is involved, the start address of the buffer is page-aligned.

All test cases are performed by iterator (*Iter.*) access (if there are any buffers). That is, a 32-bit iterator is used as index in order to read from and to write to a buffer.

Two test cases are performed by *Libc* calls. To copy a buffer (§6), the library function *memcpy* is called. To fill a buffer (§7), the library function *memset* is called.

All test cases support so-called operation-blocks of size *M*. Such a block of operations is used to minimize iterator overhead (i.e. increment, compare, branch). That is, *M=1..256* (C++) operation (i.e. *0:1,1:0,1:1*) are executed in a sequence before an iterator is checked (manual loop-unrolling).

Test cases involving buffers are performed on varying buffer sizes $N=2^1,2^2,..,2^{24}$. Operation-blocks of size *M* are never greater than buffer-sizes *N*.

3 Setup

The tests have been performed on a Raspberry Pi Zero (Pi-0) and on a Raspberry Pi 2B (Pi-2).

Raspbian serves as Linux operating system, in particular the 2016-10-06 release which is a Raspbian 8.0 (Jessie) on a Linux kernel 4.4.21.

A fresh Raspbian image was installed, for the Pi-0 and the Pi-2 each. None of the default systems settings were changed. However, the package “libboost1.55-dev” needs to be installed before the project can be compiled.

The project's program was compiled with Raspbian's GCC compiler (version 4.9.2-10) with optimization enabled (*-O3*) and debug-checks disabled (*-DNDEBUG*). For additional information refer to the project's *Makefile*.

All tests run as single thread on the ARM core. No DMA (§4[2]) is involved. The process priority was not changed (*nice*, *chrt*). No cpu-set was assigned (*taskset*).

The results presented in this document cover $8000{:}0000_H$ 32-bit word operations for pure memory tests and $800{:}0000_H$ 32-bit word for all others. For example, a test case that uses an $1{:}0000_H$-word buffer in a pure memory test, was repeated 8000_H times; a test case that uses a 2-word buffer, was repeated $4000{:}0000_H$ times. The test cases were clocked (time taken) when all repetitions had finished.

For more information see the source code.

4 Test Cases

The subsequent chapters illustrate and present the results of the test cases as follows:

§5: Peek / Poke / Copy — Word: This is a pure RAM test to estimate the maximum throughput for *1:0*, *0:1* and *1:1* mode. The result may help to judge (by comparison) the other pure RAM tests.

§6: Copy Buffer: This is a pure RAM test to estimate the maximum throughput for *N:N* mode with varying *N*. The result may help to judge (by comparison) the other pure RAM tests. The test includes *Libc's memcpy* as benchmark (i.e. as reference value).

§7: Fill Buffer: This is a pure RAM test to estimate the maximum throughput for *0:N* mode. The result provides an upper bound for sampling peripheral register (including storing their values into RAM). The test includes *Libc's memset* as benchmark (i.e. as reference value).

§8: Peek — Timer / Level / Event: This is a pure peripheral test to estimate the maximum throughput in *1:0* mode for each of the given three peripheral addresses.

§9: Peek / Poke / Reset — Event: This is a pure peripheral test to estimate the maximum throughput for the Event register. Other than the Timer and Level registers, the Event register keeps holding a value until it is explicitly reset. Reset is performed by writing back the last read value (which is actually a copy operation).

§10: Read Timer: This test estimates the maximum throughput when sampling the Timer register to a buffer (*1:N*). The test is performed for a number of varying buffer sizes *N*.

§11: Read Level: This test estimates the maximum throughput when sampling the Level register to a buffer (*1:N*). The test is performed for a number of varying buffer sizes *N*.

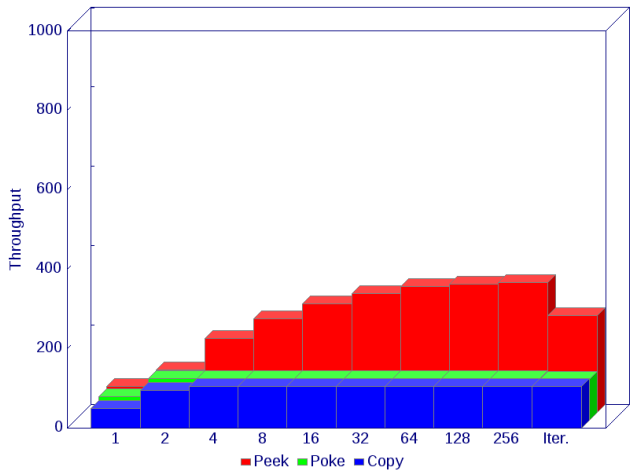
§12: Read Event: This test estimates the maximum throughput when sampling the Event register to a buffer (*1:N*). The test is performed for a number of varying buffer sizes *N*.

The throughput on the following pages is given in millions of 32-bit word operations per second (MS). For example, a value of 12.5 MS means 12.5 million 32-bit word operations per second (which are 50 MB/s).

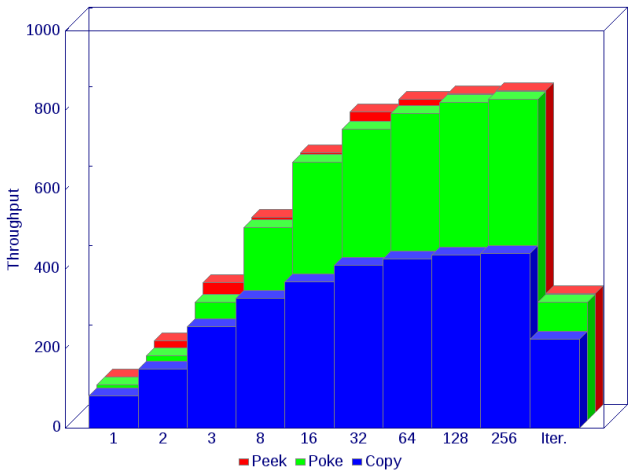
5 Peek / Poke / Copy — Word

The peek (1:0), poke (0:1) and copy (1:1) operations are executed by Iter., and in operation-blocks of size *M*.

Pi-0



Pi-1

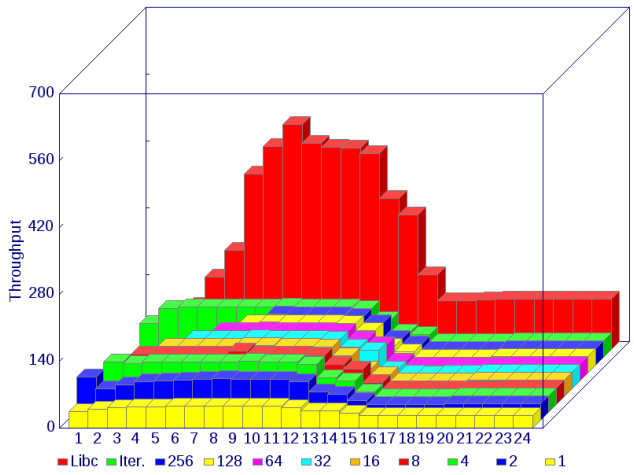


Pi-0	*M*=1	2	3	8	16	32	64	128	256	Iter.
Peek	65.0	108.0	186.0	236.0	274.0	300.0	317.0	324.0	327.0	243.0
Poke	59.8	106.0	106.0	106.0	106.0	106.0	106.0	106.0	106.0	104.0
Copy	50.2	94.9	106.0	106.0	106.0	106.0	106.0	106.0	106.0	106.0
Pi-2	*M*=1	2	3	8	16	32	64	128	256	Iter.
Peek	89.9	180.0	327.0	492.0	654.0	757.0	788.0	803.0	811.0	300.0
Poke	89.3	162.0	298.0	485.0	650.0	733.0	773.0	800.0	808.0	298.0
Copy	81.7	149.0	256.0	327.0	368.0	411.0	427.0	436.0	441.0	223.0

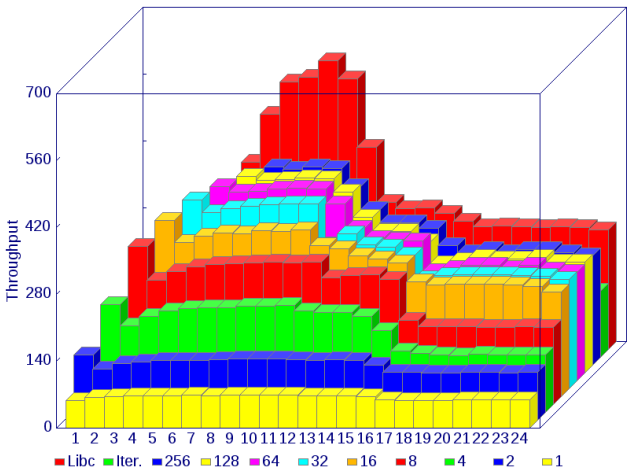
6 Copy Buffer

A buffer of size *N* is copied (*N:N*) by Libc's *memcpy*, by Iter., and in operation-blocks of size *M*.

Pi-0



Pi-2

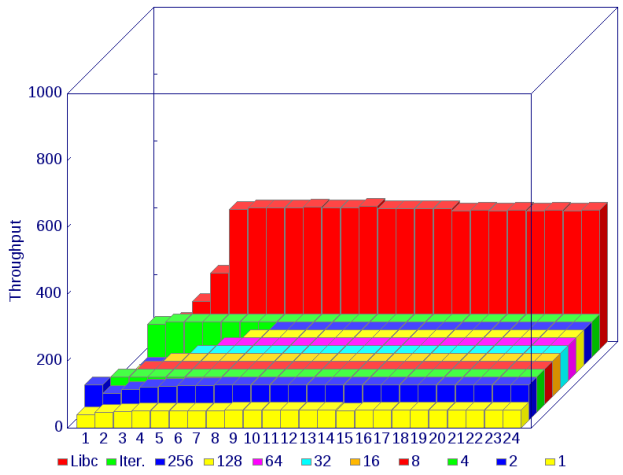


Pi-0	*N*=2^1	2^2	2^3	2^4	2^5	2^6	2^7	2^8	2^9	2^{10}	2^{11}	2^{12}	2^{13}	2^{14}	2^{15}	2^{16}	2^{17}	2^{18}	2^{19}	2^{20}	2^{21}	2^{22}	2^{23}	2^{24}
Libc	28.6	55.6	92.7	151.0	208.0	367.0	425.0	471.0	432.0	423.0	422.0	410.0	316.0	282.0	156.0	102.0	103.0	104.0	105.0	105.0	106.0	106.0	106.0	106.0
Iter.	71.4	103.0	106.0	106.0	106.0	106.0	106.0	107.0	106.0	106.0	106.0	103.0	79.2	52.4	48.0	33.0	33.0	33.2	33.6	33.9	34.0	34.2	34.2	34.2
256	n/a	n/a	n/a	n/a	n/a	n/a	n/a	107.0	106.0	106.0	105.0	103.0	88.3	56.0	44.7	33.7	32.4	33.5	33.7	33.9	34.0	33.9	34.1	34.2
128	n/a	n/a	n/a	n/a	n/a	n/a	106.0	106.0	106.0	106.0	105.0	104.0	88.7	56.5	44.8	32.7	32.5	33.0	33.7	33.9	34.0	34.1	34.2	34.2
64	n/a	n/a	n/a	n/a	n/a	106.0	106.0	107.0	106.0	106.0	105.0	104.0	87.9	76.4	41.1	33.8	32.2	33.2	33.7	33.8	34.1	34.2	34.1	34.2
32	n/a	n/a	n/a	n/a	106.0	106.0	106.0	107.0	106.0	106.0	105.0	104.0	86.8	80.5	41.5	33.7	32.1	33.2	33.5	33.9	34.1	34.2	34.2	34.1
16	n/a	n/a	n/a	106.0	106.0	106.0	106.0	108.0	107.0	106.0	105.0	104.0	85.6	56.9	45.8	33.7	33.4	33.2	33.7	34.0	34.2	34.2	34.2	34.2
8	n/a	n/a	106.0	106.0	106.0	106.0	106.0	112.0	106.0	106.0	106.0	103.0	83.5	74.7	45.6	33.7	32.4	33.1	33.8	33.8	33.7	34.1	34.2	34.1
4	n/a	106.0	102.0	106.0	106.0	106.0	106.0	106.0	108.0	106.0	106.0	101.0	72.5	66.3	43.4	33.5	32.5	33.4	33.6	33.9	34.1	34.1	34.2	34.2
2	90.4	65.7	74.6	80.1	83.1	84.7	85.3	88.2	86.1	86.3	86.2	80.0	59.1	54.7	40.5	32.7	32.9	33.4	33.6	33.9	34.0	34.1	34.1	34.1
M=1	34.6	39.5	42.4	43.9	45.0	45.4	45.5	45.6	45.7	45.7	45.7	43.6	37.1	35.7	30.6	26.2	26.2	26.3	26.4	26.5	26.6	26.6	26.6	26.6
Pi-2	*N*=2^1	2^2	2^3	2^4	2^5	2^6	2^7	2^8	2^9	2^{10}	2^{11}	2^{12}	2^{13}	2^{14}	2^{15}	2^{16}	2^{17}	2^{18}	2^{19}	2^{20}	2^{21}	2^{22}	2^{23}	2^{24}
Libc	24.0	49.0	95.2	166.0	284.0	392.0	493.0	560.0	570.0	605.0	567.0	424.0	309.0	295.0	296.0	285.0	268.0	257.0	259.0	258.0	256.0	258.0	256.0	250.0
Iter.	150.0	179.0	200.0	210.0	218.0	215.0	218.0	222.0	223.0	223.0	223.0	214.0	208.0	207.0	200.0	199.0	173.0	148.0	146.0	142.0	142.0	141.0	140.0	138.0
256	n/a	n/a	n/a	n/a	n/a	n/a	n/a	416.0	411.0	415.0	413.0	375.0	327.0	298.0	298.0	287.0	251.0	236.0	243.0	235.0	242.0	242.0	235.0	230.0
128	n/a	n/a	n/a	n/a	n/a	n/a	412.0	404.0	406.0	409.0	408.0	379.0	327.0	297.0	296.0	275.0	229.0	236.0	243.0	239.0	239.0	242.0	233.0	233.0
64	n/a	n/a	n/a	n/a	n/a	408.0	395.0	398.0	402.0	404.0	402.0	371.0	320.0	298.0	295.0	294.0	230.0	235.0	242.0	243.0	236.0	242.0	233.0	228.0
32	n/a	n/a	n/a	n/a	395.0	369.0	378.0	382.0	386.0	387.0	387.0	302.0	325.0	302.0	296.0	281.0	231.0	238.0	239.0	239.0	239.0	238.0	236.0	230.0
16	n/a	n/a	n/a	369.0	323.0	336.0	343.0	341.0	348.0	347.0	350.0	316.0	310.0	295.0	288.0	281.0	241.0	234.0	236.0	235.0	235.0	234.0	231.0	219.0
8	n/a	n/a	332.0	261.0	279.0	289.0	293.0	295.0	297.0	298.0	297.0	298.0	266.0	270.0	272.0	262.0	177.0	163.0	163.0	164.0	162.0	162.0	163.0	162.0
4	n/a	226.0	182.0	202.0	213.0	217.0	220.0	220.0	223.0	223.0	224.0	213.0	209.0	209.0	201.0	171.0	129.0	123.0	120.0	119.0	122.0	120.0	120.0	120.0
2	137.0	108.0	118.0	122.0	125.0	126.0	126.0	127.0	128.0	128.0	128.0	127.0	125.0	127.0	126.0	116.0	101.0	101.0	99.5	99.7	99.9	101.0	99.5	99.8
M=1	57.1	63.9	65.6	67.4	68.0	68.0	68.6	68.4	69.0	68.7	69.1	68.7	68.4	68.2	67.6	65.3	58.7	58.3	58.5	58.0	58.9	59.0	59.0	58.7

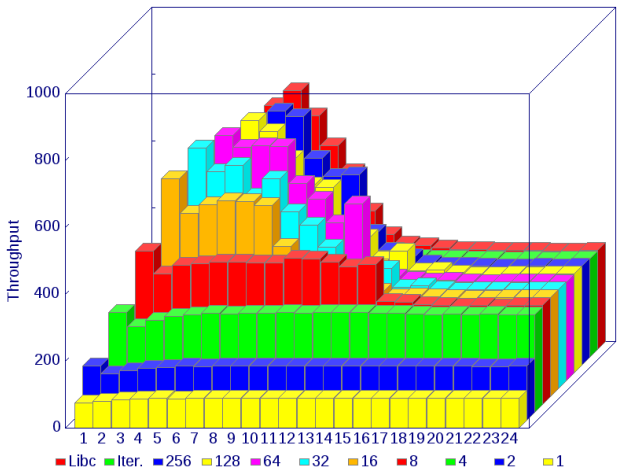
7 Fill Buffer

A buffer of size *N* is filled (*0:N*) by Libc's *memset*, by Iter., and in operation-blocks of size *M*.

Pi-0



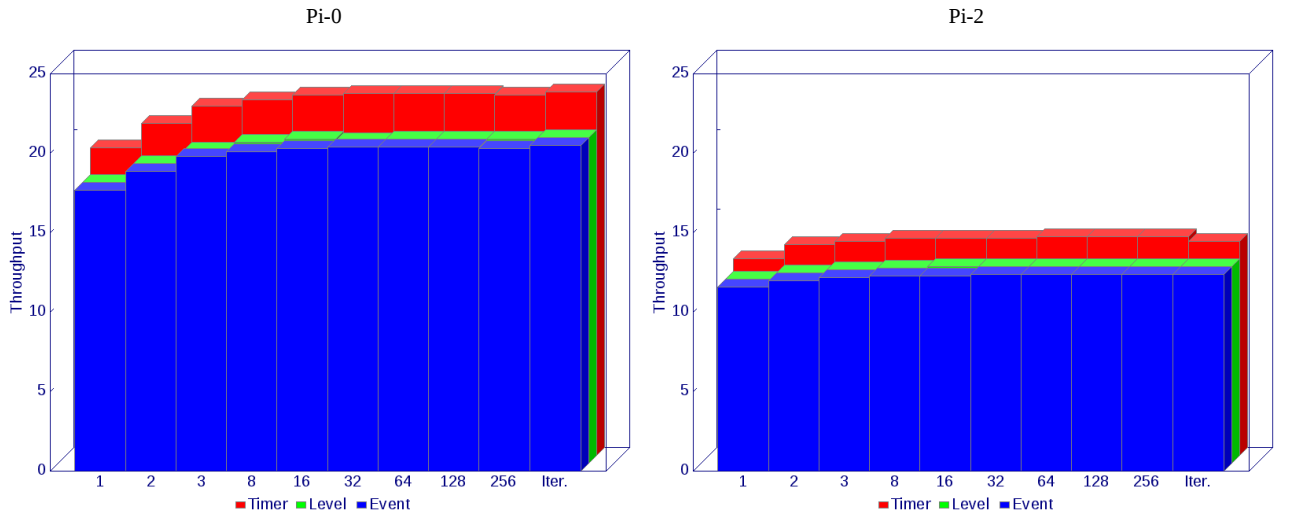
Pi-2



Pi-0	*N*=2^1	2^2	2^3	2^4	2^5	2^6	2^7	2^8	2^9	2^{10}	2^{11}	2^{12}	2^{13}	2^{14}	2^{15}	2^{16}	2^{17}	2^{18}	2^{19}	2^{20}	2^{21}	2^{22}	2^{23}	2^{24}
Libc	43.3	86.7	144.0	228.0	419.0	423.0	424.0	425.0	426.0	425.0	424.0	428.0	422.0	421.0	422.0	422.0	415.0	417.0	414.0	417.0	415.0	416.0	414.0	416.0
Iter.	100.0	106.0	106.0	106.0	106.0	106.0	106.0	106.0	106.0	106.0	106.0	106.0	106.0	106.0	106.0	106.0	106.0	106.0	106.0	106.0	106.0	106.0	106.0	106.0
256	n/a	n/a	n/a	n/a	n/a	n/a	n/a	106.0	106.0	106.0	107.0	106.0	106.0	106.0	106.0	106.0	106.0	106.0	106.0	106.0	106.0	105.0	106.0	106.0
128	n/a	n/a	n/a	n/a	n/a	n/a	106.0	106.0	106.0	106.0	106.0	107.0	106.0	106.0	106.0	106.0	106.0	106.0	106.0	106.0	106.0	106.0	106.0	106.0
64	n/a	n/a	n/a	n/a	n/a	106.0	106.0	106.0	106.0	106.0	107.0	106.0	106.0	106.0	106.0	106.0	105.0	106.0	106.0	106.0	106.0	106.0	106.0	106.0
32	n/a	n/a	n/a	n/a	106.0	106.0	106.0	106.0	106.0	106.0	107.0	106.0	106.0	106.0	106.0	106.0	106.0	106.0	106.0	106.0	106.0	106.0	106.0	106.0
16	n/a	n/a	n/a	106.0	106.0	106.0	106.0	106.0	106.0	106.0	107.0	106.0	106.0	106.0	106.0	106.0	106.0	106.0	106.0	106.0	106.0	106.0	106.0	106.0
8	n/a	n/a	106.0	106.0	106.0	106.0	106.0	106.0	106.0	106.0	106.0	106.0	106.0	106.0	106.0	106.0	106.0	106.0	106.0	106.0	106.0	105.0	106.0	106.0
4	n/a	106.0	106.0	106.0	106.0	106.0	106.0	106.0	106.0	106.0	107.0	106.0	107.0	106.0	106.0	106.0	106.0	106.0	106.0	105.0	106.0	106.0	106.0	106.0
2	106.0	81.0	91.2	98.0	102.0	103.0	104.0	105.0	105.0	105.0	105.0	106.0	105.0	105.0	105.0	105.0	105.0	105.0	105.0	105.0	105.0	105.0	105.0	105.0
M=1	40.9	46.2	49.4	51.2	52.2	52.7	52.7	53.0	53.1	53.2	53.2	53.2	53.2	53.2	53.0	53.2	53.4	53.4	53.4	53.4	53.4	53.3	53.2	53.2
Pi-2	*N*=2^1	2^2	2^3	2^4	2^5	2^6	2^7	2^8	2^9	2^{10}	2^{11}	2^{12}	2^{13}	2^{14}	2^{15}	2^{16}	2^{17}	2^{18}	2^{19}	2^{20}	2^{21}	2^{22}	2^{23}	2^{24}
Libc	56.2	116.0	200.0	326.0	479.0	621.0	731.0	775.0	700.0	611.0	533.0	414.0	343.0	315.0	310.0	303.0	300.0	299.0	296.0	296.0	296.0	294.0	294.0	296.0
Iter.	180.0	223.0	257.0	275.0	288.0	281.0	289.0	295.0	297.0	298.0	299.0	298.0	299.0	298.0	298.0	297.0	296.0	294.0	294.0	291.0	294.0	292.0	294.0	294.0
256	n/a	n/a	n/a	n/a	n/a	n/a	n/a	760.0	745.0	619.0	565.0	570.0	399.0	345.0	326.0	307.0	299.0	300.0	297.0	295.0	295.0	296.0	294.0	296.0
128	n/a	n/a	n/a	n/a	n/a	n/a	756.0	723.0	643.0	563.0	557.0	438.0	408.0	341.0	364.0	306.0	309.0	298.0	295.0	295.0	296.0	294.0	296.0	296.0
64	n/a	n/a	n/a	n/a	n/a	736.0	700.0	705.0	701.0	592.0	545.0	473.0	529.0	334.0	312.0	305.0	298.0	301.0	297.0	295.0	296.0	296.0	296.0	295.0
32	n/a	n/a	n/a	n/a	720.0	649.0	669.0	576.0	628.0	530.0	490.0	424.0	326.0	337.0	361.0	304.0	309.0	298.0	295.0	296.0	296.0	294.0	296.0	296.0
16	n/a	n/a	n/a	652.0	550.0	575.0	587.0	584.0	573.0	451.0	406.0	371.0	360.0	322.0	316.0	300.0	301.0	298.0	297.0	295.0	294.0	296.0	296.0	296.0
8	n/a	n/a	459.0	391.0	418.0	421.0	426.0	427.0	424.0	424.0	438.0	436.0	426.0	413.0	419.0	308.0	306.0	298.0	296.0	295.0	296.0	293.0	296.0	296.0
4	n/a	299.0	257.0	276.0	288.0	293.0	296.0	295.0	297.0	297.0	299.0	297.0	298.0	298.0	298.0	297.0	296.0	295.0	292.0	294.0	291.0	294.0	292.0	292.0
2	163.0	138.0	149.0	155.0	159.0	162.0	161.0	162.0	163.0	163.0	163.0	163.0	162.0	163.0	162.0	162.0	163.0	163.0	163.0	163.0	162.0	161.0	161.0	163.0
M=1	74.6	81.2	85.0	87.7	88.8	87.7	88.9	89.4	89.2	89.7	89.8	89.3	89.3	89.6	89.3	89.8	88.8	89.8	88.5	89.8	88.5	89.6	89.7	89.7

8 Peek — Timer / Level / Event

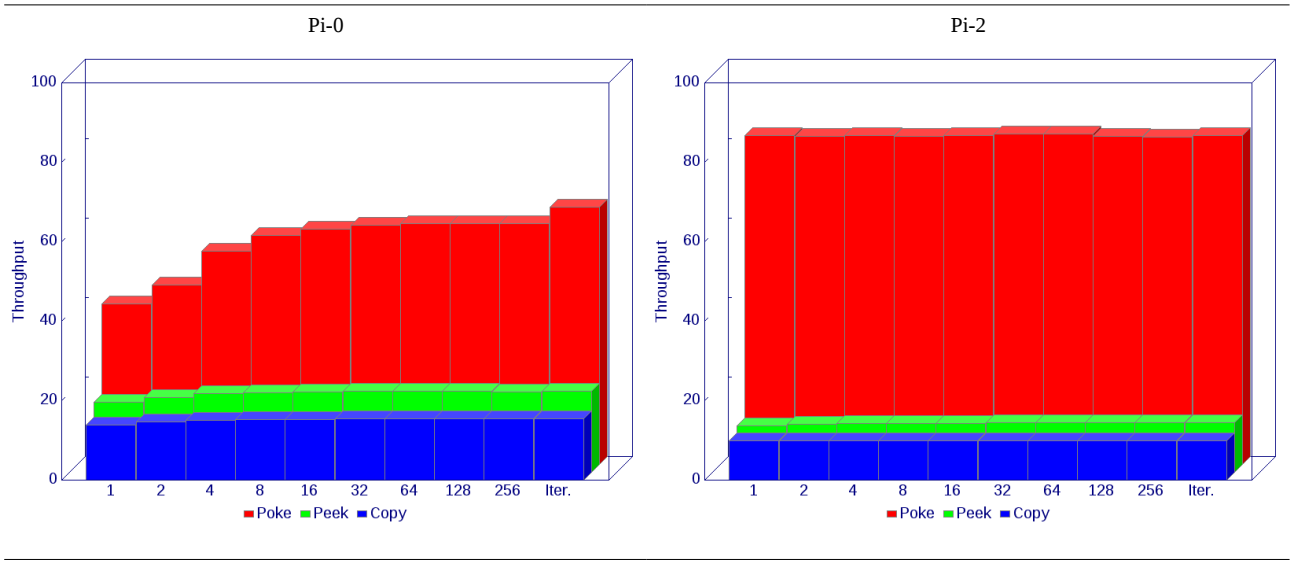
A single words is read (*1:0*) from a peripheral port (Timer, Level, Event) by Iter., and in operation-blocks of size *M*.



Pi-0	*M*=1	2	4	8	16	32	64	128	256	Iter.
Timer	19.4	20.9	22.0	22.4	22.7	22.8	22.8	22.8	22.7	22.9
Level	17.7	18.9	19.7	20.2	20.4	20.3	20.4	20.4	20.4	20.5
Event	17.7	18.9	19.8	20.1	20.3	20.4	20.4	20.4	20.3	20.5
Pi-2	*M*=1	2	4	8	16	32	64	128	256	Iter.
Timer	12.4	13.3	13.5	13.7	13.7	13.7	13.8	13.8	13.8	13.5
Level	11.6	12.0	12.2	12.3	12.4	12.4	12.4	12.4	12.4	12.4
Event	11.6	12.0	12.2	12.3	12.3	12.4	12.4	12.4	12.4	12.4

9 Peek / Poke / Reset — Event

A poke (*0:1*), a peek (*1:0*) and a copy (*1:1 to reset*) operation are executed by Iter., and in operation-blocks of size *M*.

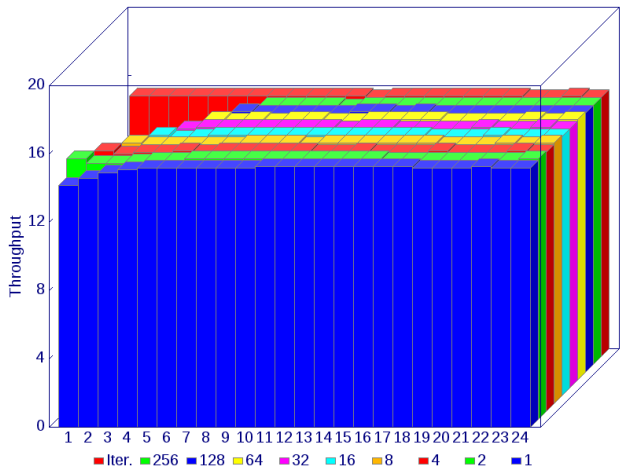


Pi-0	*M*=1	2	4	8	16	32	64	128	256	Iter.
Poke	40.5	45.2	53.7	57.6	59.2	60.3	60.6	60.7	60.7	64.8
Peek	17.7	18.9	19.8	20.1	20.3	20.4	20.4	20.4	20.3	20.5
Copy	13.8	14.6	15.1	15.3	15.3	15.4	15.4	15.4	15.4	15.5
Pi-2	*M*=1	2	4	8	16	32	64	128	256	Iter.
Poke	82.8	82.6	82.8	82.6	82.8	83.2	83.2	82.7	82.5	82.8
Peek	11.6	12.0	12.2	12.3	12.3	12.4	12.4	12.4	12.4	12.4
Copy	9.8	10.0	10.0	10.0	10.0	10.0	9.9	10.0	9.9	10.0

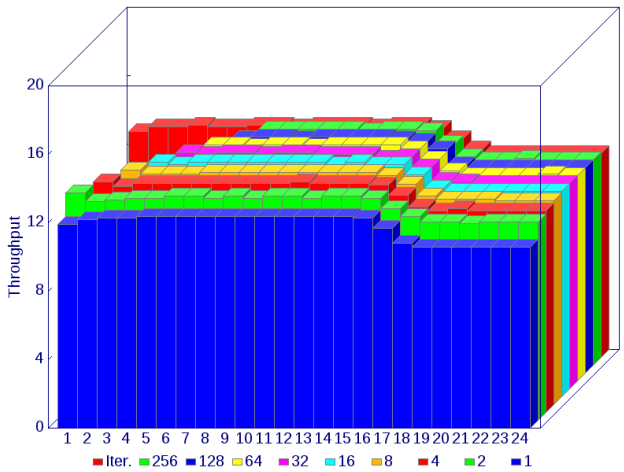
10 Read Timer

A buffer of size *N* is filled (*1:N*) with the sampled *Timer* value by Iter., and in operation-blocks of size *M*.

Pi-0



Pi-2

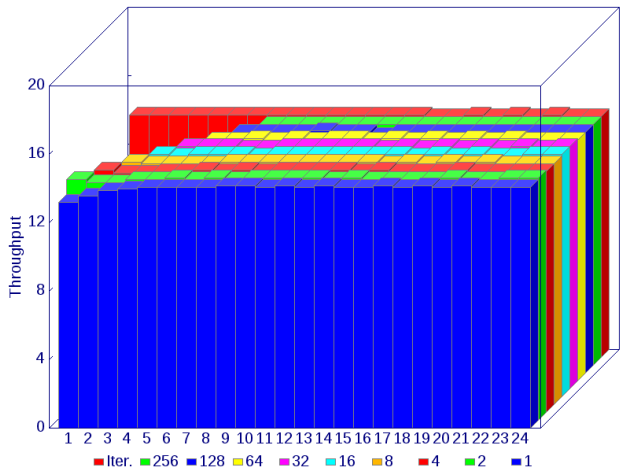


Pi-0	*N*=2^1	2^2	2^3	2^4	2^5	2^6	2^7	2^8	2^9	2^{10}	2^{11}	2^{12}	2^{13}	2^{14}	2^{15}	2^{16}	2^{17}	2^{18}	2^{19}	2^{20}	2^{21}	2^{22}	2^{23}	2^{24}
Iter.	15.3	15.3	15.3	15.3	15.3	15.3	15.3	15.3	15.3	15.3	15.3	15.3	15.2	15.3	15.3	15.3	15.3	15.3	15.3	15.3	15.2	15.2	15.3	15.2
256	n/a	n/a	n/a	n/a	n/a	n/a	n/a	15.2	15.2	15.2	15.2	15.1	15.2	15.2	15.2	15.2	15.2	15.2	15.2	15.2	15.2	15.2	15.2	15.2
128	n/a	n/a	n/a	n/a	n/a	n/a	15.3	15.2	15.2	15.2	15.2	15.2	15.3	15.3	15.2	15.3	15.2	15.2	15.2	15.2	15.2	15.2	15.2	15.2
64	n/a	n/a	n/a	n/a	n/a	15.3	15.2	15.3	15.2	15.3	15.3	15.2	15.2	15.3	15.3	15.3	15.3	15.2	15.2	15.2	15.2	15.3	15.2	15.3
32	n/a	n/a	n/a	n/a	15.2	15.3	15.3	15.3	15.3	15.3	15.3	15.3	15.2	15.3	15.3	15.2	15.2	15.2	15.2	15.3	15.2	15.2	15.3	15.2
16	n/a	n/a	n/a	15.3	15.1	15.2	15.3	15.3	15.3	15.3	15.3	15.3	15.3	15.3	15.3	15.3	15.3	15.2	15.2	15.2	15.3	15.2	15.2	15.2
8	n/a	n/a	15.3	15.2	15.2	15.2	15.2	15.2	15.3	15.3	15.3	15.2	15.3	15.3	15.3	15.3	15.3	15.3	15.2	15.2	15.2	15.3	15.2	15.3
4	n/a	15.3	15.1	15.2	15.2	15.3	15.2	15.3	15.3	15.3	15.3	15.3	15.2	15.2	15.3	15.2	15.2	15.2	15.3	15.2	15.2	15.2	15.3	15.2
2	15.3	15.0	15.0	15.1	15.2	15.2	15.3	15.3	15.3	15.3	15.3	15.3	15.3	15.3	15.3	15.3	15.2	15.2	15.2	15.2	15.2	15.2	15.2	15.3
M=1	14.2	14.6	14.9	15.1	15.2	15.2	15.2	15.2	15.2	15.2	15.3	15.3	15.3	15.3	15.3	15.3	15.3	15.3	15.2	15.2	15.2	15.3	15.2	15.2
Pi-2	*N*=2^1	2^2	2^3	2^4	2^5	2^6	2^7	2^8	2^9	2^{10}	2^{11}	2^{12}	2^{13}	2^{14}	2^{15}	2^{16}	2^{17}	2^{18}	2^{19}	2^{20}	2^{21}	2^{22}	2^{23}	2^{24}
Iter.	13.2	13.5	13.5	13.6	13.5	13.5	13.6	13.6	13.5	13.6	13.6	13.6	13.5	13.6	13.6	13.4	12.8	12.2	11.9	11.9	11.9	11.9	11.9	11.9
256	n/a	n/a	n/a	n/a	n/a	n/a	n/a	13.8	13.8	13.8	13.8	13.8	13.7	13.8	13.8	13.7	13.0	12.2	12.0	12.0	12.1	12.0	12.0	12.0
128	n/a	n/a	n/a	n/a	n/a	n/a	13.7	13.8	13.8	13.7	13.8	13.8	13.8	13.8	13.7	13.6	13.1	12.2	12.0	12.0	12.0	12.0	12.0	12.0
64	n/a	n/a	n/a	n/a	n/a	13.8	13.8	13.7	13.8	13.8	13.7	13.8	13.8	13.8	13.8	13.6	13.0	12.3	12.0	12.0	12.0	12.0	12.1	12.0
32	n/a	n/a	n/a	n/a	13.8	13.7	13.8	13.8	13.7	13.8	13.8	13.7	13.7	13.7	13.5	13.6	13.0	12.2	12.1	12.0	12.0	12.0	12.0	12.0
16	n/a	n/a	n/a	13.7	13.6	13.7	13.7	13.7	13.7	13.7	13.7	13.7	13.6	13.7	13.6	13.6	13.0	12.2	12.0	12.0	12.0	12.0	12.0	12.0
8	n/a	n/a	13.7	13.5	13.5	13.5	13.6	13.6	13.6	13.6	13.6	13.6	13.6	13.6	13.6	13.4	12.9	12.2	12.0	12.0	12.0	11.9	11.9	11.9
4	n/a	13.5	13.2	13.4	13.4	13.4	13.4	13.4	13.4	13.4	13.4	13.5	13.4	13.4	13.4	13.3	12.7	12.0	11.8	11.9	11.8	11.8	11.8	11.8
2	13.3	12.8	12.9	13.0	13.0	13.1	13.1	13.0	13.1	13.0	13.1	13.1	13.0	13.1	13.1	13.0	12.4	11.9	11.6	11.6	11.5	11.6	11.6	11.6
M=1	11.9	12.2	12.3	12.3	12.4	12.4	12.4	12.4	12.4	12.4	12.4	12.4	12.4	12.4	12.4	12.3	11.7	10.8	10.6	10.6	10.6	10.6	10.6	10.6

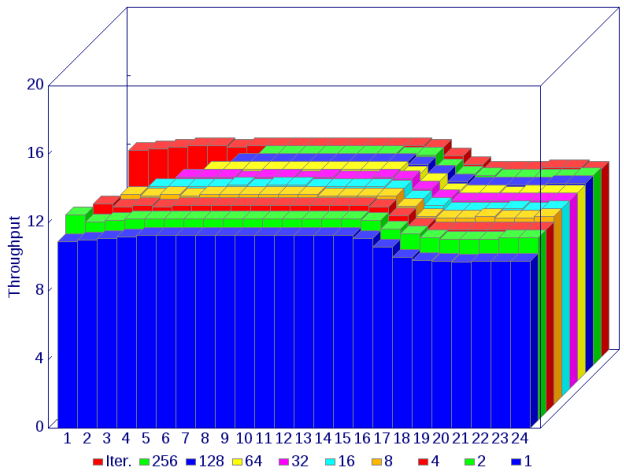
11 Read Level

A buffer of size *N* is filled (*1:N*) with the sampled *Level* value by Iter., and in operation-blocks of size *M*.

Pi-0



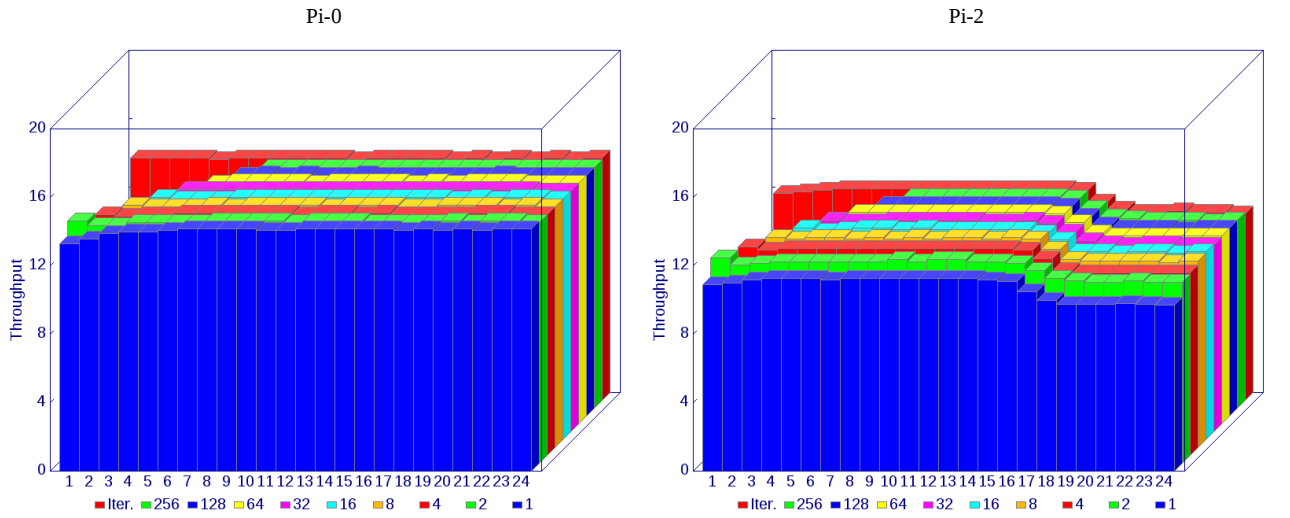
Pi-2



Pi-0	*N*=2¹	2²	2³	2⁴	2⁵	2⁶	2⁷	2⁸	2⁹	2¹⁰	2¹¹	2¹²	2¹³	2¹⁴	2¹⁵	2¹⁶	2¹⁷	2¹⁸	2¹⁹	2²⁰	2²¹	2²²	2²³	2²⁴
Iter.	14.2	14.2	14.2	14.2	14.2	14.2	14.2	14.2	14.2	14.2	14.2	14.2	14.2	14.2	14.2	14.1	14.1	14.2	14.1	14.2	14.1	14.2	14.1	14.1
256	n/a	n/a	n/a	n/a	n/a	n/a	n/a	14.1	14.1	14.1	14.1	14.1	14.1	14.1	14.1	14.1	14.1	14.1	14.1	14.1	14.1	14.1	14.1	14.1
128	n/a	n/a	n/a	n/a	n/a	n/a	14.1	14.1	14.1	14.1	14.2	14.1	14.1	13.9	14.1	14.1	14.1	14.1	14.1	14.1	14.1	14.1	14.1	14.1
64	n/a	n/a	n/a	n/a	n/a	14.1	14.1	14.2	14.1	14.2	14.1	14.2	14.2	14.1	14.2	14.1	14.2	14.1	14.2	14.1	14.2	14.1	14.1	14.1
32	n/a	n/a	n/a	n/a	14.2	14.2	14.2	14.1	14.1	14.2	14.2	14.2	14.2	14.2	14.1	14.2	14.1	14.2	14.1	14.2	14.1	14.2	14.1	14.2
16	n/a	n/a	n/a	14.2	14.1	14.2	14.2	14.2	14.1	14.2	14.2	14.2	14.2	14.2	14.2	14.2	14.1	14.2	14.1	14.2	14.1	14.1	14.2	14.2
8	n/a	n/a	14.2	14.1	14.1	14.1	14.1	14.2	14.2	14.2	14.2	14.2	14.2	14.2	14.2	14.1	14.2	14.1	14.2	14.1	14.2	14.1	14.1	14.1
4	n/a	14.2	14.0	14.0	14.1	14.1	14.2	14.1	14.2	14.1	14.2	14.2	14.2	14.2	14.1	14.2	14.1	14.1	14.1	14.2	14.1	14.1	14.1	14.1
2	14.1	13.9	13.9	14.1	14.1	14.2	14.1	14.2	14.1	14.2	14.2	14.2	14.2	14.1	14.2	14.1	14.1	14.1	14.1	14.2	14.1	14.1	14.2	14.2
M=1	13.2	13.6	13.9	14.0	14.1	14.1	14.1	14.1	14.2	14.2	14.1	14.2	14.1	14.2	14.1	14.1	14.2	14.1	14.2	14.1	14.2	14.1	14.1	14.1
Pi-2	*N*=2¹	2²	2³	2⁴	2⁵	2⁶	2⁷	2⁸	2⁹	2¹⁰	2¹¹	2¹²	2¹³	2¹⁴	2¹⁵	2¹⁶	2¹⁷	2¹⁸	2¹⁹	2²⁰	2²¹	2²²	2²³	2²⁴
Iter.	12.1	12.2	12.3	12.4	12.4	12.3	12.4	12.4	12.4	12.4	12.4	12.4	12.4	12.4	12.4	12.3	11.8	11.3	11.0	11.0	11.0	11.1	11.1	11.0
256	n/a	n/a	n/a	n/a	n/a	n/a	n/a	12.4	12.4	12.4	12.4	12.4	12.4	12.4	12.3	12.3	11.6	11.2	11.1	11.0	11.0	11.0	11.0	11.1
128	n/a	n/a	n/a	n/a	n/a	n/a	12.4	12.4	12.4	12.4	12.4	12.4	12.4	12.4	12.4	12.2	11.7	11.2	11.0	11.0	11.0	11.0	11.1	11.1
64	n/a	n/a	n/a	n/a	n/a	12.4	12.4	12.4	12.4	12.4	12.4	12.4	12.4	12.4	12.4	12.1	11.7	11.1	11.0	11.0	10.9	11.0	11.0	11.0
32	n/a	n/a	n/a	n/a	12.4	12.4	12.4	12.3	12.4	12.4	12.3	12.4	12.3	12.3	12.4	12.3	11.7	11.2	11.0	11.0	11.0	11.0	11.0	11.0
16	n/a	n/a	n/a	12.3	12.3	12.3	12.3	12.4	12.3	12.4	12.4	12.3	12.3	12.3	12.3	12.2	11.6	11.2	10.9	10.9	10.9	11.0	10.9	11.0
8	n/a	n/a	12.3	12.2	12.3	12.2	12.3	12.3	12.3	12.3	12.3	12.3	12.2	12.2	12.2	12.2	11.6	11.0	10.9	10.9	11.0	10.9	10.9	11.0
4	n/a	12.2	12.0	12.1	12.0	12.1	12.1	12.1	12.1	12.1	12.1	12.1	12.1	12.1	12.1	12.0	11.5	10.9	10.7	10.7	10.7	10.7	10.7	10.7
2	12.0	11.6	11.7	11.8	11.8	11.8	11.8	11.8	11.8	11.8	11.8	11.8	11.8	11.8	11.8	11.7	11.2	10.9	10.7	10.6	10.6	10.6	10.7	10.7
M=1	10.9	11.0	11.1	11.2	11.3	11.3	11.3	11.3	11.3	11.3	11.3	11.3	11.3	11.3	11.3	11.1	10.6	10.0	9.8	9.8	9.7	9.8	9.8	9.8

12 Read Event

A buffer of size *N* is filled (*1:N*) with the sampled *Event* value by Iter., and in operation-blocks of size *M*.



Pi-0	*N*=2^{1}	2^{2}	2^{3}	2^{4}	2^{5}	2^{6}	2^{7}	2^{8}	2^{9}	2^{10}	2^{11}	2^{12}	2^{13}	2^{14}	2^{15}	2^{16}	2^{17}	2^{18}	2^{19}	2^{20}	2^{21}	2^{22}	2^{23}	2^{24}
Iter.	14.2	14.2	14.2	14.2	14.1	14.2	14.2	14.2	14.2	14.2	14.2	14.1	14.2	14.2	14.2	14.2	14.1	14.2	14.1	14.2	14.1	14.2	14.1	14.2
256	n/a	n/a	n/a	n/a	n/a	n/a	n/a	14.1	14.1	14.1	14.1	14.1	14.1	14.1	14.1	14.1	14.1	14.1	14.1	14.1	14.1	14.1	14.1	14.1
128	n/a	n/a	n/a	n/a	n/a	n/a	14.1	14.2	14.1	14.2	14.1	14.1	14.2	14.1	14.1	14.1	14.1	14.1	14.1	14.2	14.1	14.1	14.1	14.1
64	n/a	n/a	n/a	n/a	n/a	14.1	14.2	14.2	14.2	14.2	14.1	14.2	14.2	14.2	14.2	14.1	14.2	14.1	14.2	14.2	14.2	14.1	14.1	14.1
32	n/a	n/a	n/a	n/a	14.2	14.2	14.2	14.2	14.2	14.2	14.2	14.2	14.2	14.2	14.2	14.2	14.2	14.2	14.2	14.2	14.2	14.2	14.1	14.1
16	n/a	n/a	n/a	14.2	14.1	14.1	14.1	14.2	14.2	14.2	14.2	14.2	14.2	14.2	14.2	14.2	14.2	14.2	14.1	14.2	14.1	14.2	14.2	14.1
8	n/a	n/a	14.2	14.1	14.1	14.1	14.2	14.1	14.2	14.2	14.2	14.2	14.1	14.2	14.2	14.2	14.1	14.1	14.2	14.2	14.2	14.1	14.2	14.1
4	n/a	14.1	14.0	14.1	14.1	14.2	14.2	14.2	14.2	14.2	14.2	14.1	14.2	14.2	14.1	14.2	14.1	14.1	14.1	14.1	14.2	14.2	14.1	14.1
2	14.2	13.9	13.9	14.1	14.1	14.1	14.1	14.2	14.2	14.2	14.2	14.1	14.2	14.2	14.2	14.1	14.1	14.1	14.1	14.2	14.1	14.2	14.2	14.1
M=1	13.3	13.6	13.9	14.0	14.0	14.1	14.2	14.2	14.2	14.2	14.1	14.1	14.2	14.2	14.2	14.2	14.2	14.1	14.2	14.1	14.2	14.1	14.2	14.2
Pi-2	*N*=2^{1}	2^{2}	2^{3}	2^{4}	2^{5}	2^{6}	2^{7}	2^{8}	2^{9}	2^{10}	2^{11}	2^{12}	2^{13}	2^{14}	2^{15}	2^{16}	2^{17}	2^{18}	2^{19}	2^{20}	2^{21}	2^{22}	2^{23}	2^{24}
Iter.	12.1	12.2	12.3	12.4	12.4	12.4	12.4	12.4	12.4	12.4	12.4	12.4	12.4	12.4	12.4	12.3	11.6	11.2	11.0	11.0	11.1	11.0	11.0	10.9
256	n/a	n/a	n/a	n/a	n/a	n/a	n/a	12.4	12.4	12.4	12.4	12.4	12.4	12.4	12.4	12.3	11.7	11.2	11.1	11.0	11.0	11.0	10.9	11.0
128	n/a	n/a	n/a	n/a	n/a	n/a	12.4	12.4	12.4	12.4	12.4	12.4	12.4	12.4	12.4	12.3	11.7	11.2	11.0	11.0	11.0	10.9	11.0	11.0
64	n/a	n/a	n/a	n/a	n/a	12.4	12.4	12.4	12.4	12.4	12.4	12.4	12.4	12.4	12.4	12.3	11.8	11.2	11.0	11.0	11.0	11.0	11.0	11.0
32	n/a	n/a	n/a	n/a	12.4	12.3	12.3	12.4	12.4	12.4	12.4	12.4	12.3	12.3	12.4	12.2	11.7	11.2	11.0	10.9	11.0	11.0	10.9	10.9
16	n/a	n/a	n/a	12.4	12.2	12.3	12.4	12.4	12.3	12.4	12.3	12.3	12.3	12.3	12.3	12.1	11.7	11.0	11.0	10.9	11.0	11.0	10.9	11.0
8	n/a	n/a	12.3	12.2	12.3	12.3	12.2	12.3	12.2	12.3	12.2	12.3	12.3	12.2	12.3	12.2	11.6	11.0	10.9	10.9	10.9	10.9	10.8	10.9
4	n/a	12.2	12.0	12.1	12.1	12.1	12.1	12.1	12.1	12.1	12.1	12.1	12.1	12.1	12.1	12.0	11.5	10.8	10.7	10.7	10.7	10.7	10.7	10.7
2	12.0	11.6	11.7	11.8	11.8	11.8	11.8	11.8	11.8	11.9	11.8	11.9	11.9	11.8	11.8	11.7	11.3	10.8	10.7	10.6	10.6	10.7	10.6	10.6
M=1	10.9	11.0	11.2	11.3	11.3	11.3	11.2	11.3	11.3	11.3	11.3	11.3	11.3	11.3	11.2	11.1	10.5	10.0	9.8	9.8	9.8	9.8	9.8	9.7