

Doc. No.	AN0117	Ver.	02	Doc. Title	I ² C Throughput Using the ezPyro API
----------	--------	------	----	------------	--



Application Note 0117

I²C Throughput Using the ezPyro™ API

Version Number 2

Issue Date 30-05-18

Doc. No.	AN0117	Ver.	02	Doc. Title	I ² C Throughput Using the ezPyro API
----------	--------	------	----	------------	--



1. SUMMARY

1.1 Overview

This note covers I²C performance testing of the ezPyro device. The report explains the important data throughput related considerations involved in I²C communications design for the ezPyro sensor. The main focus is on the collection of data within the 1 ms frame time (i.e. maximum sampling rate). The question is finding the theoretical limit of the I²C bus on fast mode plus 1 mbit/s and comparing it to the real life implementation. The ezPyro sensor is compatible with this I²C setting.

1.2 Results

1.2.1 Performance

Configuration	Test	Result
I ² C throughput - theoretical calculation	Reading a 5 channel frame of data per millisecond	4.5 frames
I ² C throughput – ezPyro API measurement	Reading a 5 channel frame of data per millisecond	3 frames

Doc. No.	AN0117	Ver.	02	Doc. Title	I ² C Throughput Using the ezPyro API
----------	--------	------	----	------------	--



2. TABLE OF CONTENTS

1.	SUMMARY	2
1.1	Overview	2
1.2	Results	2
1.2.1	Performance	2
2.	TABLE OF CONTENTS	3
3.	I2C PERFORMANCE ANALYSIS	4
3.1	General considerations	4
3.2	I2C calculation	4
3.2.1	I2C Cost	4
3.2.2	I2C throughput.....	5
3.3	I2C on the messaging application.....	5
3.4	Conclusion: Maximum throughput.....	6
4.	TEST SETUP	8
5.	REFERENCES.....	10

Figures

Figure 1	Logic analyser output at maximum throughput captured data	5
Figure 2	Logic capture with no frames lost on the GUI	6
Figure 3	CCP_init() modification	8
Figure 4	main.c modification	9

DOCUMENT HISTORY

Ver.	Date	Change Ref.	Change Details
01	18 Jan 2018	N/A	First Release
02	30-05-18	N/A	Reviewed & published

Doc. No.	AN0117	Ver.	02	Doc. Title	I ² C Throughput Using the ezPyro API
----------	--------	------	----	------------	--



3. I²C PERFORMANCE ANALYSIS

3.1 General Considerations

To establish how long it takes for a command to be sent to a particular ezPyro in order to request a full frame of data we need to consider the cost of an operation on the I²C bus. In general a simple transaction will use a certain number of clock cycles in the following manner (Gosh, 2012):

2 clocks for the start condition

1 clock for the stop condition

9 clocks for the address

9 clocks per byte sent or received.

The beginning of a transaction between the MCU and the sensor will require 2+1+9+9 = 21 clocks.

A typical FIFO_READ_ACTIVE operation (Pyreos, LTD, 2017) will recover from the sensor a packet of sampled data of a maximum of 17 bytes: 3 bytes per enabled channel plus 2 bytes for the frame counter.

The fast mode plus I²C runs at 1 mbit/s or 1000 kHz at top speed. I²C speed varies according to the length of the transaction. A short transaction will not achieve the maximum I²C average throughput; a long transaction will get closer to the nominal.

3.2 I²C Calculation

3.2.1 I²C Cost

For this calculation the number of clock periods required for each operation is taken into account and the number of bytes required for completing the transaction.

Operation	Bytes	Content
setup write	1	ezPyro address
write command	1	FIFO active code
setup read	1	ezPyro address
read response	17	Data
setup write	1	ezPyro address
write command	1	FIFO clear code
setup read	1	ezPyro address
read response	1	result code
Bytes per transaction	24	

Doc. No.	AN0117	Ver.	02	Doc. Title	I ² C Throughput Using the ezPyro API
----------	--------	------	----	------------	--

PYREOS

From the table we see that a complete transaction to read a full packet of data from the sensor requires 24 bytes travelling to and fro, between MCU and an ezPyro sensor with all active channels collecting data at the maximum rate of 1ms.

3.2.2 I²C Throughput

Each byte transmitted or received requires 9 clocks to be delivered, and then we have:

$$9 * 24 = 216 \text{ clocks.}$$

In I²C each operation is marked by a beginning and an end using the bus clock. Two clocks are required to start and one to stop.

FIFO active read : start + stop = 3 clocks

FIFO clear : start + stop = 3 clocks

The total numbers of clock cycles required therefore is: 216 + 6 = 222 clock cycles.

Our theoretical I²C clock runs at 1,000,000 clocks per second, during 1ms we have only 1,000 clocks available, so:

$$1,000 \text{ available} / 222 \text{ required per packet read} = \mathbf{4.50 \text{ frames per millisecond.}}$$

These 4.50 frames contain information of 20 channels spread across 4 ezPyro sensors. **Based on this calculation we could expect to read a maximum of 20 channels per millisecond.**

3.3 I²C on the Messaging Application

Using the messaging application (API) to configure an ezPyro board with four sensors using five channels each, the logic analyser was used to capture the transaction.

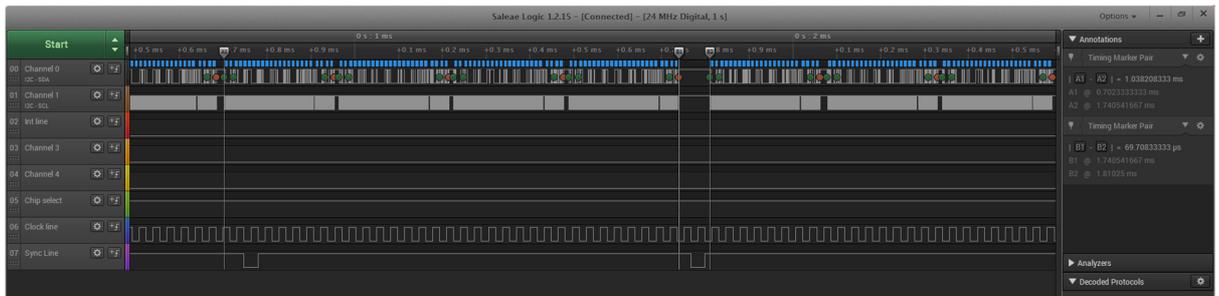


Figure 1 Logic analyser output at maximum throughput captured data

Figure 1 shows a segment of the I²C communication between the ezPyro sensor and the MCU. A pair of markers has been set between the beginning and the end of reading a block of four sensors. The logic analyser shows an elapsed time of 1.03 milliseconds to complete the operation. The gap between operations is 0.07 milliseconds. The average clock speed for the I²C, reported by the analyser is 888.9 KHz.

However the GUI application reports a data rate of 464 Hz. When the data is collected into a CSV file, half of the packets of the sampling are lost. Due to the processing overhead that takes place between the firmware and the GUI and the time the next sample data becomes available.

Doc. No.	AN0117	Ver.	02	Doc. Title	I ² C Throughput Using the ezPyro API
----------	--------	------	----	------------	--

PYREOS

To achieve the best compromise between number of sensors and maximum number of channels sampled, several configurations of number of active channels and number of sensors where tested.

By changing the configuration of the firmware and observing the data rate reported by the ezPyro GUI 15 channels could be sampled with the following distribution:

Sensor	Channels enabled
1	5
2	5
3	5
4	0

The collected data is shown on the logic analyser below.

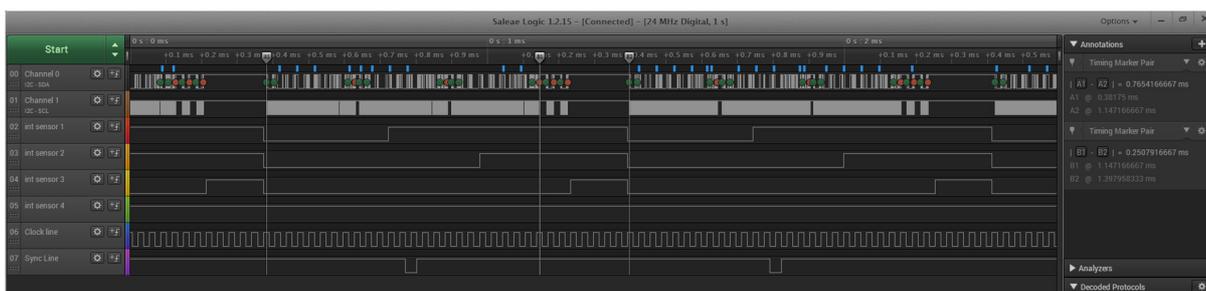


Figure 2 Logic capture with no frames lost on the GUI

The first marker reports 0.76 milliseconds for the 15 channels read. Two false reads take place that the firmware execute as part of the normal execution loop, when there is no data on the sensors. The last marker shows the gap between reads, when no data is available, which amounts to 0.25 milliseconds.

3.4 Conclusion: Maximum throughput

From the perspective of collecting data with no frames lost, and transmitting this data to the existing GUI and the current ezPyro API based firmware can record data from **15 channels across 3 sensors**.

If a mechanism to store the collected data in a separate thread is used then the **theoretical maximum of 20 channels across 4 sensors could be achieved**.

There is an overhead due to normal execution of the firmware that limits the number of channels that can be collected without affecting data collection from the user perspective. How much can this be improved by optimizing the structure of the current firmware could be investigated if required.

Doc. No.	AN0117	Ver.	02	Doc. Title	I ² C Throughput Using the ezPyro API
----------	--------	------	----	------------	--



Doc. No.	AN0117	Ver.	02	Doc. Title	I ² C Throughput Using the ezPyro API
----------	--------	------	----	------------	--

PYREOS

4. TEST SETUP

Data was collected using Saleae logic analyser, with the i2c decoder set to 7 bit addresses (Saleae, 2018).

Special modification to the API and the messaging application.

```

606 **@param this_device pointer to the ezPyro data structure
607 **@retval op_result '1' success / '0' error
608 **/
609 static uint8_t CCP_init(ezPyro_sensor_device_type* this_device)
610 {
611     uint8_t op_result = 0, initialization_failed = 0;
612     uint8_t selected_channel = 0, ch_to_activate = 0;
613     //
614     CCP_INIT_DEFAULT(ccp_defaults);
615     CCP_reg_type CCP_data;
616     // active channels value is checked for 1 or 4 channels, if empty defaults to 1
617     switch((this_device->number_of_active_channels))
618     {
619     case 1:
620         ch_to_activate = 1;
621         selected_channel = 2;
622         break;
623     case 4:
624         ch_to_activate = 4;
625         selected_channel = 1;
626         break;
627     default:
628         // other than 1 or 4 the initialization fails.
629         initialization_failed = 1;
630         break;
631     }
632     if(!initialization_failed)
633     {
634         this_device->number_of_active_channels = ch_to_activate;
635         // only active channels are initialized with default data if empty
636         while(ch_to_activate)
637         {
638             CCP_data = this_device->CCP_register[selected_channel];
639             if(!CCP_register_empty(CCP_data))
640             {
641                 this_device->CCP_register[selected_channel] = ccp_defaults;
642             }
643             selected_channel++;
644             ch_to_activate--;
645         }
646         op_result = CCP_write(this_device);
647     }
648     return op_result;
649 }
650 /**
651 **@brief reads the CCP registers on the ezPyro and loads into this device
652 **@param this_device pointer to the ezPyro data structure
653 **@param CCP_data pointer to the CCP register on the ezPyro
654 **/

```

Figure 3 CCP_init() modification

Doc. No.	AN0117	Ver.	02	Doc. Title	I ² C Throughput Using the ezPyro API
----------	--------	------	----	------------	--

PYREOS

```

165...this_list[1].CS_Pin...=5;
166...this_list[1].INT_Port...=1;
167...this_list[1].INT_Pin...=5;
168...this_list[1].address...=0x69;
169#
170...this_list[2].CS_Port...=2;
171...this_list[2].CS_Pin...=4;
172...this_list[2].INT_Port...=1;
173...this_list[2].INT_Pin...=6;
174...this_list[2].address...=0x68;
175#
176...this_list[3].CS_Port...=2;
177...this_list[3].CS_Pin...=3;
178...this_list[3].INT_Port...=1;
179...this_list[3].INT_Pin...=7;
180...this_list[3].address...=0x67;
181#
182.../*-----COMMIT_CONFIGURATION-----*/
183...this_list[0].number_of_active_channels...=1;
184...this_list[0].AFE_register...=ezPyro_AFE_Master; // MASTER, INT, CLK_ENABLED
185...this_list[0].OCP_register[2]...=ezPyro_Ch_Cfg;
186...this_list[0].WUP_register...=ezPyro_WUP_Cfg;
187#
188...this_list[1].number_of_active_channels...=1;
189...this_list[1].AFE_register...=ezPyro_AFE_Slave; // SLAVE, INT, CLK_DISABLED
190...this_list[1].OCP_register[2]...=ezPyro_Ch_Cfg;
191...this_list[1].WUP_register...=ezPyro_WUP_Cfg;
192#
193...this_list[2].number_of_active_channels...=1;
194...this_list[2].AFE_register...=ezPyro_AFE_Slave; // SLAVE, INT, CLK_DISABLED
195...this_list[2].OCP_register[2]...=ezPyro_Ch_Cfg;
196...this_list[2].WUP_register...=ezPyro_WUP_Cfg;
197#
198...this_list[3].number_of_active_channels...=1;
199...this_list[3].AFE_register...=ezPyro_AFE_Slave; // SLAVE, INT, CLK_DISABLED
200...this_list[3].OCP_register[2]...=ezPyro_Ch_Cfg;
201...this_list[3].WUP_register...=ezPyro_WUP_Cfg;
202#
203/**
204...@brief Detect ezPyro sensors present in the demo board
205...@param this_list ezPyro sensor array
206...@param list_size number of sensors in the array
207...*/
208static void ezPyro_sensor_population(ezPyro_devices this_list, uint8_t* list_size)
209{
210...uint8_t num_of_ezPyro...=*list_size;
211...uint8_t devices_present...=0;
212...memset(uint8_t types[]={0x00, 0x7F, 0x77, 0x67, 0x47});
213...for(uint8_t current=0; current<num_of_ezPyro; ++current)
214...{
215...{
216...}
217...}
218...}
219...}
220...}
221...}
222...}
223...}
224...}
225...}
226...}
227...}
228...}
229...}
230...}
231...}
232...}
233...}
234...}
235...}
236...}
237...}
238...}
239...}
240...}
241...}
242...}
243...}
244...}
245...}
246...}
247...}
248...}
249...}
250...}
251...}
252...}
253...}
254...}
255...}
256...}
257...}
258...}
259...}
260...}
261...}
262...}
263...}
264...}
265...}
266...}
267...}
268...}
269...}
270...}
271...}
272...}
273...}
274...}
275...}
276...}
277...}
278...}
279...}
280...}
281...}
282...}
283...}
284...}
285...}
286...}
287...}
288...}
289...}
290...}
291...}
292...}
293...}
294...}
295...}
296...}
297...}
298...}
299...}
300...}
301...}
302...}
303...}
304...}
305...}
306...}
307...}
308...}
309...}
310...}
311...}
312...}
313...}
314...}
315...}
316...}
317...}
318...}
319...}
320...}
321...}
322...}
323...}
324...}
325...}
326...}
327...}
328...}
329...}
330...}
331...}
332...}
333...}
334...}
335...}
336...}
337...}
338...}
339...}
340...}
341...}
342...}
343...}
344...}
345...}
346...}
347...}
348...}
349...}
350...}
351...}
352...}
353...}
354...}
355...}
356...}
357...}
358...}
359...}
360...}
361...}
362...}
363...}
364...}
365...}
366...}
367...}
368...}
369...}
370...}
371...}
372...}
373...}
374...}
375...}
376...}
377...}
378...}
379...}
380...}
381...}
382...}
383...}
384...}
385...}
386...}
387...}
388...}
389...}
390...}
391...}
392...}
393...}
394...}
395...}
396...}
397...}
398...}
399...}
400...}
401...}
402...}
403...}
404...}
405...}
406...}
407...}
408...}
409...}
410...}
411...}
412...}
413...}
414...}
415...}
416...}
417...}
418...}
419...}
420...}
421...}
422...}
423...}
424...}
425...}
426...}
427...}
428...}
429...}
430...}
431...}
432...}
433...}
434...}
435...}
436...}
437...}
438...}
439...}
440...}
441...}
442...}
443...}
444...}
445...}
446...}
447...}
448...}
449...}
450...}
451...}
452...}
453...}
454...}
455...}
456...}
457...}
458...}
459...}
460...}
461...}
462...}
463...}
464...}
465...}
466...}
467...}
468...}
469...}
470...}
471...}
472...}
473...}
474...}
475...}
476...}
477...}
478...}
479...}
480...}
481...}
482...}
483...}
484...}
485...}
486...}
487...}
488...}
489...}
490...}
491...}
492...}
493...}
494...}
495...}
496...}
497...}
498...}
499...}
500...}
501...}
502...}
503...}
504...}
505...}
506...}
507...}
508...}
509...}
510...}
511...}
512...}
513...}
514...}
515...}
516...}
517...}
518...}
519...}
520...}
521...}
522...}
523...}
524...}
525...}
526...}
527...}
528...}
529...}
530...}
531...}
532...}
533...}
534...}
535...}
536...}
537...}
538...}
539...}
540...}
541...}
542...}
543...}
544...}
545...}
546...}
547...}
548...}
549...}
550...}
551...}
552...}
553...}
554...}
555...}
556...}
557...}
558...}
559...}
560...}
561...}
562...}
563...}
564...}
565...}
566...}
567...}
568...}
569...}
570...}
571...}
572...}
573...}
574...}
575...}
576...}
577...}
578...}
579...}
580...}
581...}
582...}
583...}
584...}
585...}
586...}
587...}
588...}
589...}
590...}
591...}
592...}
593...}
594...}
595...}
596...}
597...}
598...}
599...}
600...}
601...}
602...}
603...}
604...}
605...}
606...}
607...}
608...}
609...}
610...}
611...}
612...}
613...}
614...}
615...}
616...}
617...}
618...}
619...}
620...}
621...}
622...}
623...}
624...}
625...}
626...}
627...}
628...}
629...}
630...}
631...}
632...}
633...}
634...}
635...}
636...}
637...}
638...}
639...}
640...}
641...}
642...}
643...}
644...}
645...}
646...}
647...}
648...}
649...}
650...}
651...}
652...}
653...}
654...}
655...}
656...}
657...}
658...}
659...}
660...}
661...}
662...}
663...}
664...}
665...}
666...}
667...}
668...}
669...}
670...}
671...}
672...}
673...}
674...}
675...}
676...}
677...}
678...}
679...}
680...}
681...}
682...}
683...}
684...}
685...}
686...}
687...}
688...}
689...}
690...}
691...}
692...}
693...}
694...}
695...}
696...}
697...}
698...}
699...}
700...}
701...}
702...}
703...}
704...}
705...}
706...}
707...}
708...}
709...}
710...}
711...}
712...}
713...}
714...}
715...}
716...}
717...}
718...}
719...}
720...}
721...}
722...}
723...}
724...}
725...}
726...}
727...}
728...}
729...}
730...}
731...}
732...}
733...}
734...}
735...}
736...}
737...}
738...}
739...}
740...}
741...}
742...}
743...}
744...}
745...}
746...}
747...}
748...}
749...}
750...}
751...}
752...}
753...}
754...}
755...}
756...}
757...}
758...}
759...}
760...}
761...}
762...}
763...}
764...}
765...}
766...}
767...}
768...}
769...}
770...}
771...}
772...}
773...}
774...}
775...}
776...}
777...}
778...}
779...}
780...}
781...}
782...}
783...}
784...}
785...}
786...}
787...}
788...}
789...}
790...}
791...}
792...}
793...}
794...}
795...}
796...}
797...}
798...}
799...}
800...}
801...}
802...}
803...}
804...}
805...}
806...}
807...}
808...}
809...}
810...}
811...}
812...}
813...}
814...}
815...}
816...}
817...}
818...}
819...}
820...}
821...}
822...}
823...}
824...}
825...}
826...}
827...}
828...}
829...}
830...}
831...}
832...}
833...}
834...}
835...}
836...}
837...}
838...}
839...}
840...}
841...}
842...}
843...}
844...}
845...}
846...}
847...}
848...}
849...}
850...}
851...}
852...}
853...}
854...}
855...}
856...}
857...}
858...}
859...}
860...}
861...}
862...}
863...}
864...}
865...}
866...}
867...}
868...}
869...}
870...}
871...}
872...}
873...}
874...}
875...}
876...}
877...}
878...}
879...}
880...}
881...}
882...}
883...}
884...}
885...}
886...}
887...}
888...}
889...}
890...}
891...}
892...}
893...}
894...}
895...}
896...}
897...}
898...}
899...}
900...}
901...}
902...}
903...}
904...}
905...}
906...}
907...}
908...}
909...}
910...}
911...}
912...}
913...}
914...}
915...}
916...}
917...}
918...}
919...}
920...}
921...}
922...}
923...}
924...}
925...}
926...}
927...}
928...}
929...}
930...}
931...}
932...}
933...}
934...}
935...}
936...}
937...}
938...}
939...}
940...}
941...}
942...}
943...}
944...}
945...}
946...}
947...}
948...}
949...}
950...}
951...}
952...}
953...}
954...}
955...}
956...}
957...}
958...}
959...}
960...}
961...}
962...}
963...}
964...}
965...}
966...}
967...}
968...}
969...}
970...}
971...}
972...}
973...}
974...}
975...}
976...}
977...}
978...}
979...}
980...}
981...}
982...}
983...}
984...}
985...}
986...}
987...}
988...}
989...}
990...}
991...}
992...}
993...}
994...}
995...}
996...}
997...}
998...}
999...}
1000...}

```

Figure 4 main.c modification

Doc. No.	AN0117	Ver.	02	Doc. Title	I ² C Throughput Using the ezPyro API
----------	--------	------	----	------------	--



5. REFERENCES

Gosh, A. (2012, november 28). *I²C communications speed across sensors*.

Retrieved from electrical engineering:

<https://electronics.stackexchange.com/questions/50011/i2c-communication-speed-across-sensors>

Pyreos, Ltd. (2017). *ezPyro sensor reference manual*. Edinburgh..

Saleae. (2018, January 16). *Learn I²C - Inter-Integrated Circuit*. Retrieved from Saleae.com: <https://support.saleae.com/hc/en-us/articles/115005987106-Learn-I2C-Inter-Integrated-Circuit>