

Application Notes

I²C Throughput

The above information is believed to be correct but does not purport to be all inclusive and must be used only as a guide. © KEMET Electronics Corporation • KEMET Tower • One East Broward Boulevard Fort Lauderdale, FL 33301 USA • 954-766-2800 • www.kemet.com



1. TABLE OF CONTENTS

1.	TABLE	E OF CONTENTS	. 2
2.	SUMM	IARY	. 2
	2.1	Overview	. 2
	2.2	Performance Results	. 2
3.	I ² C PE	RFORMANCE ANALYSIS	. 3
	3.1	General Considerations	. 3
	3.2	I ² C Calculation	. 3
	3.2	2.1 I ² C Time Cost	3
	3.2	2.2 I ² C Throughput	4
	3.3	I ² C on the Messaging Application	. 4
	3.4	Conclusion: Maximum Throughput	. 5
4.	TEST	SETUP	. 6

2. SUMMARY

2.1 Overview

This application note covers I²C performance testing of the USEQ*S devices, from the QFS, QGS and QMS series. The report explains the important data throughput related considerations involved in I²C communications design for these sensors. The focus is on the collection of data within the 1ms 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 USEQ*S sensors are compatible with this I²C setting.

2.2 Performance Results

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 – API measurement	Reading a 5-channel frame of data per millisecond	3 frames

DOCUMENT HISTORY

Version	Date	Change Ref.	Change Details
01	11 JUN 2020	N/A	First Release

The above information is believed to be correct but does not purport to be all inclusive and must be used only as a guide.

© KEMET Electronics Corporation • KEMET Tower • One East Broward Boulevard



3. I²C PERFORMANCE ANALYSIS

3.1 General Considerations

To establish how long it takes for a command to be sent to a USEQ*S sensor 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 ¹

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 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 Time Cost

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

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

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 from, between MCU and a USEQ*S sensor with all active channels collecting data at the maximum rate of 1ms.

The above information is believed to be correct but does not purport to be all inclusive and must be used only as a guide.

 $\ensuremath{\textcircled{\sc b}}$ KEMET Electronics Corporation ${\ensuremath{\cdot}}$ KEMET Tower ${\ensuremath{\cdot}}$ One East Broward Boulevard

¹ 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.



3.2.2 I²C Throughput

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

9 * 24 = 216 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 available / 222 required per packet read = 4.50 frames per millisecond.

These 4.50 frames contain information of 20 channels spread across 4 USEQ*S 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 a control board for modules (USEQCSK0000000) with four sensors using five channels each, the logic analyser was used to capture the transaction.

Saleae Logic 1.2.15 - [Connected] - [24 MHz Urgital, 1 s]			Options - C	
Start	1	0::1m; 0::2m;	▼ Annotations -	
00 Channel 0 111 12C-SDA 01 Channel 1	0 15 0 15		Tirming Marker Pair * A1 - A2 = 1.038208333 ms A1 @ 0.7023333333 ms A2 m 1.740541667 ms	
02 Int line				
03 Channel 3	() (+F			
04 Channel 4	0 +F			
06 Clock line	¢ 4			
07 Sync Line			► Analyzers	
			Decoded Protocols	¥)

Figure 1 – Logic Analyser Output at Maximum Throughput Captured Data

Figure 1 shows a segment of the I²C communication between the USEQ*S 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.

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 USEQ*S GUI 15 channels could be sampled with the following distribution:

Sensor	Channels enabled
1	5
2	5
3	5
4	0

The above information is believed to be correct but does not purport to be all inclusive and must be used only as a guide.





The collected data is shown on the logic analyser below.

and the second se	Saleae Logic 12.15 - [Connected] - [24 MHz Digital, 1 s]			Options 👻 🗕 🗗 🗙	
Start	¢	0\$;0ms +0.1ms +0.2ms +0.3mm+0.4ms +0.5ms +0.6ms +0.7ms +0.8ms +0.9ms	:1 ms +0.005 +0.2 ms +0.3 ms 0.4 ms +0.5 ms +0.6 ms +0.7 ms +0.8 ms +0.9 ms	0 s : 2 ms +0.1 ms +0.2 ms +0.3 ms +0.4 ms +0.5 ms	Annotations H Timing Marker Dair
00 Channel 0		n uns ser esta de la compassión de la comp	iumes ara a suu nu nu mesas iu nu un mesas nu		
01 Channel 1 IIII I2C-SCL					
02 int sensor 1					
03 int sensor 2	¢ 19				
04 int sensor 3					
05 int sensor 4	0 (*F				
06 Clock line					
07 Sync Line					► Analyzers
					▼ Decoded Protocols 🏾 🌣

Figure 2 – Logic Capture with no Frames Lost on the GUI

The first maker 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 USEQ*S 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.



4. TEST SETUP

Data was collected using Saleae logic analyser, with the I²C decoder set to 7-bit addresses ². Special modification to the API and the messaging application.



Figure 3 - CCP_init() Modification

Image: Description of the set of the s	2 Winterge - Louise, Working Ease, Revision 1376 - salare, Working Capyl					
<pre>Not We control the submatched bool how () () () () () () () () () () () () ()</pre>						
<pre>media two loss to the provide states and the provide states and</pre>	비 년 년 [10] 여 [11] 오 삼 65 조 [1 4 4 [6 6] 전 일 [2] 이 여					
<pre>value of the set of the set</pre>	man.c.:Working Base, Hevision 13/6 - man.c.:Working Copy					
<pre>1 distance in the list [] (display = bit</pre>	Locationise X main.c : Working Base, Revision 1376	main.c : Working Copy				
<pre></pre>	165 ····this list[1].CS Pin ·····5; =	165 ····this list[1.CS Pin······S)=				
<pre></pre>	100 ····this_list[1].INT_Port = 1/m	100 'this list[1]. INT port = 1) =				
<pre> the function of the second second</pre>	16/ this list[1].iNT_PIN '= 5/m	160 ····Chis_list[1.:NT_Pin ·=·5)#				
<pre>1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1</pre>	lico	100 ····this_list[1].address ····(xcs):				
<pre>10</pre>	109 a static list(2) on perturning	109 m				
<pre>12</pre>	170 ·····this_list[2].cs_Poirt·····4/z	170 ·····this_list[2].cs_port····2/#				
<pre>1 135this_list(2).107_bit = 4/* 136this_list(2).107_bit = 4/* 137this_list(2).107_bit = 4/* 138this_list(2).107_bit = 4/* 139this_list(2).107_bit = 4/*</pre>	171 dils_list[2].tstpin = 1, a	172 this list[2] TAT Dort = 1				
<pre>199this_list(2).address -= 0x80/s 199this_list(2).address -= 0x80/s 199this</pre>	172 dilis_list[2] INT_port = 1, 4	172 chis_list[2].INI_FOID = 1/4				
<pre>195 * """</pre>	174 unitalitatical interest of the second se	174 ····this_list[2] address ·= 0x68:s				
<pre>179</pre>						
<pre>177 ····ths_list(2):CS_Pin'-e_2/s 173 ····ths_list(2):CS_Pin'-e_2/s 173 ····ths_list(3):NT_POrt-+1/s 174 ····ths_list(3):NT_POrt-+1/s 175 ····ths_list(3):NT_POr</pre>	176this list[3].CS Port= 2:=	176 ····this list[3].CS Port ··=·2:=				
<pre>176</pre>	177 this list 31.CS Pin= 3;=	177 this list[3].CS pin = 3;=				
<pre>199</pre>	178 this list[3]. INT Port = 14	178this list 31. INT Port = 1				
<pre>100 ····this_list[3].address ··· 0x87):s 110 ····this_list[3].address ··· 0x87):s 111 ··· 112 ···/* 113 ····this_list[3].address ··· 0x87):s 113 ····this_list[3].address ··· 0x87):s 114 ··· 115 ····this_list[3].address ··· 0x87):s 116 ····this_list[3].address ··· 0x87):s 117 ····this_list[3].address ··· 0x87):s 118 ····this_list[3].address ··· 0x87):s 119 ····this_list[3].address ··· 0x87):s 110 ··</pre>	179 ····this list[3].INT Pin ··=·7; =	179this list [3]. INT Pin= .7; =				
<pre>101 * 101 * 101 * 101 * 102 ************************************</pre>	180 ····this list [3].address ··= ·0x67;=	180 ····this list[3].address ·= ·0x67;=				
<pre>182</pre>	181 =	181 =				
<pre>133 ****this list[0].mubber of active channels=1:# 143 ****this list[0].mubber of active channels=1:# 145 ****this list[0].mubber of active channels=1:# 145 ****this list[1].mubber of active channels=1:# 146 ****this list[1].mubber of active channels=1:# 147 ****this list[1].mubber of active channels=1:# 148 ****this list[1].mubber of active channels=1:# 149 ****this list[1].mubber of active channels=1:# 149 ****this list[1].mubber of active channels=1:# 140 ****this list[1].mubber of active channels=1:# 141 ****this list[1].mubber of active channels=1:# 141 ****this list[1].mubber of active channels=1:# 142 ****this list[1].mubber of active channels=1:# 143 ****this list[1].mubber of active channels=1:# 144 ****this list[1].mubber of active channels=1:# 145 ****this list[1].mubber of active channels=1:# 145 ****this list[1].mubber of active channels=1:# 146 ****this</pre>	182 ····/**/=	182 ····/**/=				
184	183 ····this_list[0].number_of_active_channels = 1;=	183 ····this_list[0].number_of_active_channels = 5;=				
<pre>15 ····ths_list[0].cCC_register[2] * estyro_Ch_Cfg:= 16 ···ths_list[0].cCC_register[2] * estyro_Ch_Cfg:= 16 ···ths_list[1].muber_of_active_channels.ml;s= 17 ··ths_list[1].muber_of_active_channels.ml;s= 18 ···ths_list[1].muber_of_active_channels.ml;s= 18 ···ths_list[1].muber_of_active_channels.ml;s= 18 ···ths_list[1].muber_of_active_channels.ml;s= 19 ···ths_list[1].muber_of_active_channels.ml;s= 10 ···ths_list[1].muber_of</pre>	184 ····this_list[0].AFE_register ····= ·ezPyro_AFE_Master; ·// ·MASTER, ·INT, ·CLK ·ENABLED=	184 ····this_list[0].AFE_register ····= ·ezPyro_AFE_Master; ·// ·MASTER, ·INT, ·CLK ·ENABLED=				
<pre>166 ····this_list[0].WUE_register ··· = ezPyro_WUE_Cfg:s 167 ···this_list[1].WUE_register ··· = ezPyro_WUE_Cfg:s 168 ····this_list[1].WUE_register ··· = ezPyro_WUE_Cfg:s 169 ···this_list[1].WUE_register ··· = ezPyro_WUE_Cfg:s 160 ···this_list[1].WUE</pre>	185 ····this_list[0].CCP_register[2] ·= ·ezPyro_Ch_Cfg;=	185 ····this_list[0].CCP_register[2] ·= ·ezPyro_Ch_Cfg;=				
<pre>107* ***********************************</pre>	186 ····this_list[0].WUP_register ····= 'ezPyro_WUP_Cfg;=	186 ····this_list[0].WUP_register ····= ezPyro_WUP_Cfg;=				
<pre>19 ************************************</pre>	187 =	187 =				
<pre>19 ····this_list[].AF_register ···· = astyro_AFE_slave; //·SLAVE, 'INT, 'CLK DISABLED= 19 ····this_list[].AF_register ···· = astyro_AFE_slave; //·SLAVE, 'INT, 'CLK DISABLED= 19 ····this_list[].AFF_register ···· = astyro_AFE_slave; //·SLAVE, 'INT, 'CLK DISABLED= 10 ····this_list[].AFF_register ···· = astyro_AFE_slave; //·SLAVE, 'INT, 'CLK DISABLED= 20 ····this_list[].</pre>	188 ····this_list[1].number_of_active_channels = 1;=	188 ····this_list[1].number_of_active_channels = 5;=				
<pre>19 ····this_list[].cCp_equister[2] ** estyre_Ch_Ctg;* 19 ····this_list[].ccp_equister[2] ** estyre_Ch_Ctg;* 20 ····this_list[].ccp_equister[2] ** estyr</pre>	189 ····this_list[1].AFE_register ····= 'ezPyro_AFE_Slave; '// 'SLAVE, 'INT, 'CLK'DISABLED"	189this_list[1].AFE_register = 'ezPyro_AFE_Slave; // 'SLAVE, 'INT, 'CLK'DISABLED"				
<pre>14</pre>	190 this list[1].CCP register[2] = ezPyro Ch Cig/m	190 this list[1].ccp register[2] = ezpyro_cn crg;=				
<pre>143 ************************************</pre>	191 ····this_list[1].WUP_register ····= ezPyro_WUP_Crg;=	191 ····this_list[1].WUP_register ····= 'ezPyro_WUP_CTg; =				
<pre>134</pre>	192 a	192 a				
<pre>139</pre>	193 ····this_list(2). https://www.endlist.com///state_inv/state_inv//state_inv//state_inv//state_in	193 ·····(his_list(2), humber_ol_active_chambers - 3)#				
<pre>196</pre>	195this list[2].CD radiater[2]arDuro (b Cfarm	195this_list[2].CCP register[2]:sapure Ch Cfarm				
197 197 197 110 197 110 197 110 197 111 197 111 197 111 197 111 197 111 197 111 197 111 197 111 197 111 197 1111 197	196 ····this list[2] WID register ······ezPyro_UID Cfaus	196this_list(2).WID_register(2) = explore WID_Cfg.s				
<pre>199 ***this_list[3].ruber_of_active_channels=*1:** 199 ***this_list[3].ruber_of_active_channels=*1:** 200 ***this_list[3].ruber_of_active_channels=*1:*** 200 ***this_list[3].ruber_of_active_channels=*1:*** 200 ***this_list[3].ruber_of_active_channels=*1:***********************************</pre>	197	197 m				
<pre>199 ···this_list[3].AFE_register ··· ** esPyro_AFE_diave; /// SLAVE, 'INT, 'CLK 'DISABLEDE 199 ···this_list[3].AFE_register ··· ** esPyro_AFE_diave; /// SLAVE, 'INT, 'CLK 'DISABLEDE 201 ···this_list[3].WDE_register ··· ** esPyro_WDE_Cfg:* 201 ···this_list[3].WDE_register ··· ** esPyro_WDE_Cfg:* 201 ···this_list[3].WDE_register ··· ** esPyro_WDE_Cfg:* 202 :* 203 /·** 204 ·* @brist ibs_list espyro_sensor present in the demo 'boarde 204 ·* @brist ibs_list espyro_sensor present in the demo 'boarde 205 ·** @brist ibs_list espyro_sensor present in the 'demo 'boarde 206 ·** @brist ibs_list espyro_sensor present in the 'demo 'boarde 207 /** 208 statLo vold 'espyro_sensor present in the 'arrays 207 ·* 208 statLo vold 'espyro_sensor population (espyro_devices this_list, 'uint8_t ** 'list_size)* 209 (* 201 ··· uint8_t 'num_of_espyro_sensor population (espyro_devices this_list, 'uint8_t ** 'list_size)* 209 (* 201 ··· uint8_t 'num_of_espyro_sensor population (espyro_devices this_list, 'uint8_t ** 'list_size)* 209 (* 201 ··· uint8_t 'num_of_espyro_sensor population (espyro_devices this_list, 'uint8_t ** 'list_size)* 209 (* 201 ··· uint8_t 'num_of_espyro_sensor population (espyro_devices this_list, 'uint8_t ** 'list_size)* 209 (* 201 ··· uint8_t 'num_of_espyro_sensor copulation (espyro_devices this_list, 'uint8_t ** 'list_size)* 209 (* 201 ··· uint8_t 'num_of_espyro_sensor copulation (espyro_devices this_list, 'uint8_t ** 'list_size)* 209 (* 201 ··· uint8_t 'num_of_espyro_sensor copulation (espyro_devices this_list, 'uint8_t '* 'list_size)* 209 (* 201 ··· uint8_t 'num_of_espyro_sensor copulation (espyro_devices this_list, 'uint8_t '* 'list_size)* 209 (* 201 ··· uint8_t 'num_of_espyro_sensor copulation (espyro_devices this_list, 'uint8_t '* 'list_size)* 209 (* 201 ··· uint8_t 'num_of_espyro_sensor copulation (espyro_devices this_list, 'uint8_t '* 'list_size)* 201 ··· uint8_t 'num_of_espyro_sensor copulation (espyro_devices this_list, 'uint8_t '* 'list_size)* 202 ··· wate 'list_size' 'list_size)* 203 //** 204 //** 204 //**</pre>	198this list[3].number of active channels = 1:#	198this list[3] number of active channels = 5:				
<pre>200 ····this_list[3].com_register[2] == estype_Ch_Cfcy: 201 ····this_list[3].com_</pre>	199 ····this list[3].AFE register ···· = ezPyro AFE Slave; // SLAVE, ·INT, ·CLK ·DISABLED=	199 ····this list[3]. AFE register ···· = ezPvro AFE Slave; // SLAVE, 'INT, 'CLK'DISABLED=				
<pre>201 ····ths_list[3].WUF_register ····= erPyro_WUF_cfg:= 202 /= 203 /*= 203 /*= 203 /*= 204 /*= brief of betect erPyro sensors present in the demo boards 205 ·* @param this_list erPyro sensors present in the demo boards 206 ·* @param this_list erPyro sensors present in the demo boards 206 ·* @param this_list erPyro sensors present in the demo boards 207 /*= 208 static-void erPyro_sensor population (erPyro_devices this_list, uint8_t ** list_size) = 209 (= 201 ····inist_t num_of_erPyro_sensor_population (erPyro_devices this_list, uint8_t ** list_size) = 209 (= 210 ····inist_t num_of_erPyro_sensor_population (erPyro_devices this_list, uint8_t ** list_size) = 210 ····inist_t num_of_erPyro_sensor_population (erPyro_devices this_list, uint8_t ** list_size) = 210 ····inist_t num_of_erPyro_** list_size; = 211 ····inist_t devices present=0) = 212 ····ourst_uint8_t trum_of_erPyro_** list_size; = 212 ····ourst_uint8_t trum_of_erPyro, '** (uint8_t ** uint8_t ** list_size) = 213 ····infe(uint8_t trum_of_erPyro, '** list_size) = 214 ····infe(uint8_t trum_of_erPyro, '** list_size; = 215 ····ourst_uint8_t trum_of_erPyro; '** uirent) = 216 ····infe(uint8_t trum=0) · ourrent < num_of_erPyro; '** current > 217 ····infe(uint8_t trum=0) · ourrent < num_of_erPyro; '** current > 218 ·····infe(uint8_t trum=0) · ourrent < num_of_erPyro; '** current > 219 ·····infe(uint8_t trum=0) · ourrent < num_of_erPyro; '** current > 210 ·····infe(uint8_t trum=0) ·····infe(uint8_t trum=0) ·····infe(uint8_t trum=0) ·····infe(uint8_t trum=0) ······infe(uint8_t trum=0) ·······infe(uint8_t trum=0) ·······infe(uint8_t trum=0) ·······infe(uint8_t trum=0) ····································</pre>	200 ····this list[3].CCP register[2] = ezPvro Ch Cfg/=	200 ····this list[3].CCP register[2] = ezPyro Ch Cfg;=				
202 /= 203 /*= 203 /*= 204 ** @brief Detect 'e2Pyro 'sensors present in the 'demo 'boards 205 ** @braam list_size number of 'sensors in the 'arrays 206 ** @braam list_size number of 'sensors in the 'arrays 207 */s 208 static 'void 'e2Pyro_sensor_population (e2Pyro_devices this_list, 'uint8_t *' list_size) = 208 static 'void 'e2Pyro_sensor_population (e2Pyro_devices this_list, 'uint8_t *' list_size) = 209 (= 201 ··· uint1_* rum of e2Pyro_ *' list_size, : 201 ··· uint1_* rum of e2Pyro_ *' list_size, : 203 /*= 203 /*= 200 (= 201 ··· uint1_* rum of e2Pyro_ *' list_size, : 201 ··· uint1_* rum of e2Pyro_ *' list_size, : 203 ··· for (uint1_* rum of e2Pyro, '+ current) = 201 ··· uint1_* rum of e2Pyro, '+ current) =	201 this list[3].WUP register = ezPyro WUP Cfg;=	201 ··· this list[3].WUP register ··· = ezPyro WUP Cfg;=				
<pre>203 /*** 203 /*** 203 /*** 204 ** @brisf:Detect veZPyro 'sensors present in the 'demo 'boards 205 ** @braam this_list veZPyro 'sensors in the 'arrays 206 ** @braam this_list veZPyro 'sensors in the 'arrays 207 ** @braam this_list veZPyro 'sensors in the 'arrays 208 statle void 'serPyro_sensor_population (seZPyro_sensor_population (seZPyro_sensor_population (seZPyro_sensor_population (seZPyro_sensor_population (seZPyro_sensor_section the 'arrays 209 (s 210 ··· uintd_t - num_of_seZPyro_sensor_population (seZPyro_sensor_population (seZPyro_sensor_senset) = 200 (s 210 ··· uintd_t - num_of_seZPyro_sensor_senset = 201 ··· uintd_t - devices present = 01: 212 ··· soust uintd_t - devices present = 01: 213 ··· soust uintd_t - devices present = 01: 213 ··· soust uintd_t - devices present = 01: 213 ··· soust uintd_t - devices present = 01: 213 ··· soust uintd_t - devices present = 01: 213 ··· soust uintd_t - devices present = 01: 213 ··· soust uintd_t - devices present = 01: 213 ··· soust uintd_t - devices present = 01: 213 ··· soust uintd_t - devices present = 01: 213 ··· soust uintd_t - devices present = 01: 213 ··· soust uintd_t - devices present = 01: 213 ··· soust uintd_t - devices present = 01: 213 ··· soust uintd_t - devices present = 01: 213 ··· soust uintd_t - devices present = 01: 213 ··· soust uintd_t - devices present = 01: 214 ··· devices present = 01: 215 ··· soust uintd_t - devices present = 01</pre>	202) =	202) =				
204 ** @brief 'Detect 'orfyro 'ensors present in the 'demo 'boards 205 ** @param 'list_list erfyro 'ensors 'norsenor' arrays 206 ** @param 'list_list erfyro 'ensors in 'the 'arrays 207 ** @param 'list_list erfyro 'ensors in 'the 'arrays 208 static 'void 'erfyro_sensor_population (erfyro_devices this_list, 'uint8_t ** 'list_size) = 209 static 'void 'erfyro_sensor_population (erfyro_devices this_list, 'uint8_t ** 'list_size) = 201 **	203 /**=	203 /**=				
205 ** 0param:this_list cerzyro:sensor:arrays 205 ** 0param:this_list cerzyro:sensor:arrays 206 ** 0param:this_list cerzyro:sensor:sin:the varrays 205 ** 0param:this_list cerzyro:sensor:sin:the varrays 208 ** 0param:this_list covid verzyro_sensor_population (varrays) 206 ** 0param:this_list cerzyro:sensor:sin:the varrays 209 ** 200 ** 0param:this_list 209 ** 200 ** 0param:this_list 209 ** 200 ** 200 ** 200 ** 201 **: unitd_t:num.of_selzyro:**list_size;= 210 **: unitd_t:num.of_selzyro:**list_size;= 210 **: unitd_t:num.of_selzyro:**list_size;= 210 **: unitd_t:divide greent:*0;= 211 **: unitd_t: divides greent:*0;= 212 **: coust-unitd_t: divides greent:*0;= 213 **: for (unitd_t: unitd_t: unitd_t: unitd_t: divides greent:*0;= 214 **: divides greent:*0;= 215 **: divides greent:*0;= 216 **: divides greent:*0;= 217 **: divides greent:*0;= 218 **: divides greent:*0;= 219 **: divides greent:*0;= 210 **: divides greent:*0;= 211 **: divides greent:*0;= 212 **: coust-unitd_t: divides greent:*0;= 213 **: for (unitd_t: unitd_t: uni	204 '* '@brief 'Detect 'ezPyro 'sensors 'present 'in 'the 'demo 'board"	204 '* '@brief 'Detect 'ezPyro 'sensors 'present 'in 'the 'demo 'board=				
206 ** *@param list_size number of *encors in the *arrays 206 ** *@param list_size number of *encors in the *arrays 207 */s 208 static void *ePyro_sensor_population(e2Pyro_devices this_list, vunt8_t** list_size)s 208 ** @param list_size 206 ** 209 ** 201 ***im of exprosents of sensors in the *arrays 201 ***im of exprosents of sensors in the *arrays 207 */s 202 *** 208 *** 203 *** 208 *** 204 *** 208 *** 205 *** 208 *** 206 *** 208 *** 207 */s 208 *** 208 *** 208 *** 209 *** 200 *** 209 *** 200 **** 201 ************************************	205 ·* ·@param ·this_list ·ezPyro ·sensor ·array=	205 ** @param this_list ezPyro sensor array=				
207 */= 207 */= 207 */= 209 static void erPyro_sensor_population(srPyro_devices this_list, uint8_t ** list_size)= 201 */= 210 ************************************	206 ·* ·@param ·list_size ·number ·of ·sensors ·in ·the ·array=	206 ·* ·@param ·list_size ·number ·of ·sensors ·in ·the ·array=				
200 statle vold espyro_sensor_population (espyro_devices this_list, uint8_t**list_sise)* 200 statle vold espyro_sensor_population (espyro_devices this_list, uint8_t**list_sise)* 200 statle vold espyro_sensor_population (espyro_devices this_list, uint8_t**list_sise)* 200 statle vold espyro_sensor_population (espyro_devices this_list, uint8_t**list_sise)* 210 **** 100 **** 200 statle vold espyro_sensor_population (espyro_devices this_list, uint8_t**list_sise)* 210 **** 100 ***** 200 statle vold espyro_sensor_population (espyro_devices this_list, uint8_t**list_sise)* 210 ***** 100 ****** 200 statle vold espyro_sensor_population (espyro_devices this_list, uint8_t**list_sise)* 210 ***** 100 ***********************************	207 */=	207*/=				
209 (a) 209 (a) 210 (···uintŝ t·num_of_ezPyro·=·*list_size; a) 210 (···uintŝ t·num_of_ezPyro·=·*list_size; a) 211 ····uintŝ t·duïcas present-a0; a) 211 ····uintŝ t·num_of_ezPyro·=·*list_size; a) 212 ····oconst·uintŝ t·rum_of_ezPyro·=·*list_size; a) 211 ····uintŝ t·num_of_ezPyro·=·*list_size; a) 213 ····for(uintŝ t·current=0; ·current ···0; a) 211 ····uintŝ t·duïcas present-a0; a) 213 ····for(uintŝ t·current=0; ·current ···num_of_ezPyro; ++·current) = 213 ····for(uintŝ t·current=0; ·current ···num_of_ezPyro; ++·current) = 1/1 ····intŝ t·current=0; ·current ···num_of_ezPyro; ++·current) = 31 ·····intŝ t·current=0; ·current ····num_of_ezPyro; ++·current) = 1/1 ·····intŝ t·current=0; ·current ····intŝ t·current=0; ·current ·····intŝ t·current=0; ·current ·····intŝ t·current ·····intŝ t·current ····································	208 Static void ezPyro_sensor_population (ezPyro_devices this_list, uint8_t * 'list_size) =	208 static void ezPyro_sensor_population (ezPyro_devices this_list, uint8_t * 'list_size) =				
zw intsnmm_orsavyro						
interpretation interpretation interpretation interpretation interpretation	210 ····unto_t rum of expyro = 'list size's	210 ····uinto_t.hum or ezyro = `·list_size/=				
Image: State	211 ····utilito_tridevices_present0/*	211 ···································				
	213 ····································	213 ··· for (units trourset=0) · (units trourset=0) ··· (usis) ···				
ال المحتود المحت المحتود المحتود المحتو	214(alles carrence, carrence, carrence, namor_erryro, ff carrent) a	214				
Les 22 Coli 1/81 On 1/81 [252 Unix Les 22 Coli 1/81 On 1/81 [252 Unix Les 25 Coli 1/81 [252 Uni						
	1 1252 Unix 1/81 Chi 1/81 Chi 1/81	Juni 22 Coli 1/81 1252 Unix				

Figure 4 - main.c Modification

The above information is believed to be correct but does not purport to be all inclusive and must be used only as a guide.

© KEMET Electronics Corporation • KEMET Tower • One East Broward Boulevard

² Saleae. (2018, January 16). *Learn I²C - Inter-Integrated Circuit*. Retrieved from Saleae.com: https://support.saleae.com/tutorials/learning-portal/learning-resources/learn-i2c#learn-i-2-c-inter-integrated-circuit