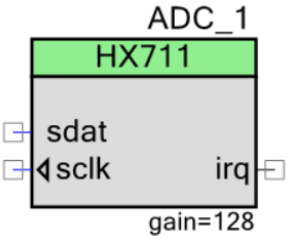
HX711: 24-bit Delta Sigma ADC interface for weight scale

0.0

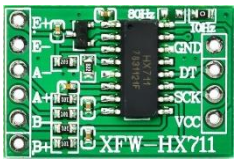
Features

- Implements software interface for HX711 ADC
- Interfaces single HX711 sensor board
- Uses interrupt or polling methods
- Has selectable ADC gain



General description

The HX711[*] component implements software interface to HX711 24-bit analog- to-digital converter by AVIA Semiconductor, designed for weigh scales and industrial control applications [1]. Using this component in conjunction with external HX711 ADC, PSoC can detect small DC signals in the range of ±20 mV, ±40 mV or ±80 mV at 10 Hz or 80 Hz sampling rate. Component consume little to none hardware resources, and spares few clocks (~0.01%). Multiple instances of the component can run asynchronously in the project.



When to use HX711 component

Component was developed for a weight scale project, where multiple load cells must be measured using HX711 boards. It can be useful whenever a weak signal needs to be digitized with high precision, such as bio-potentials, ECG signal etc. Component is useful for a system with limited hardware resources, such as PSoC4. Component was tested using CY8KIT-059 prototyping kit (PSoC5LP) and CY8KIT-042 Pioneer Board (PSoC4200). Several demo projects are provided along with the Application Note.

[*] Hereafter referred to as "ADC"

Input-output connections

sdat – ADC data input

External terminal for connecting to a HX711 annotation component (off-chip). The pin is always visible. The pin doesn't have to be connected; it is merely an external terminal to the annotation component, provided to enhance visibility of the component. Actual assignment of the digital pin is performed in the Pins dialog. See **Implementation** section for details.

sclk – clock output

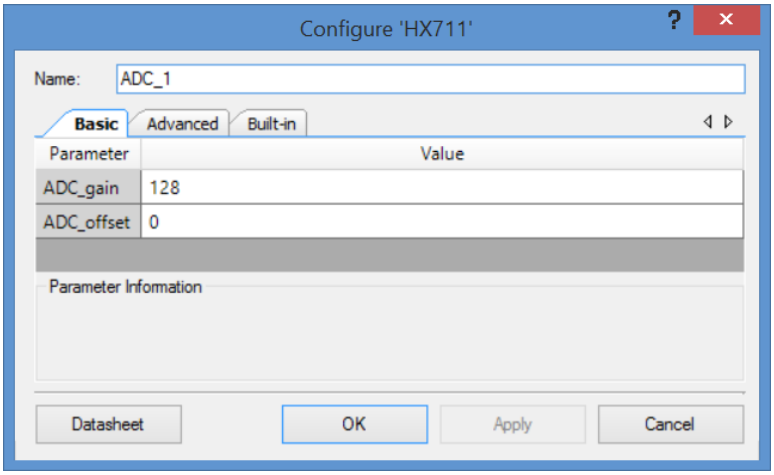
External terminal for connecting to a HX711 annotation component (off-chip). The pin is always visible. The pin doesn't have to be connected; it is merely an external terminal to the annotation component, provided to enhance visibility of the component. Actual assignment of the digital pin is performed in the Pins dialog. See **Implementation** section for details.

irq – interrupt output

Output pin for external interrupt connection. The pin is visible when **interrupt** option is selected in Advanced Dialog. When visible, the pin must be connected to an interrupt.

Parameters and Settings

Basic dialog provides following parameters[*]:



ADC_gain (128 / 64 / 32)

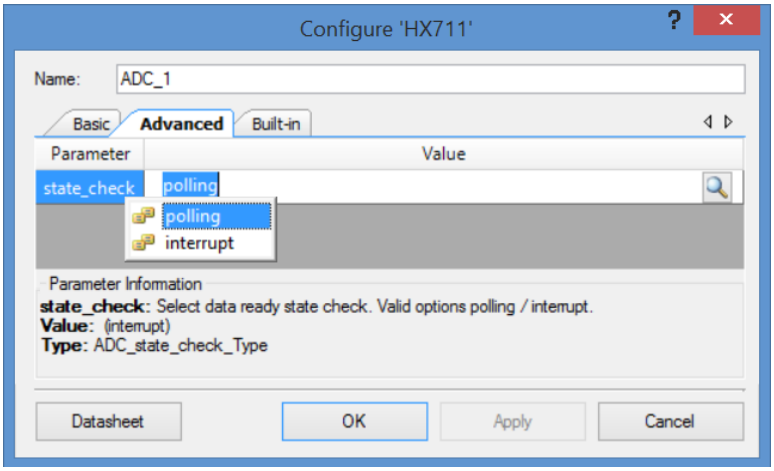
Programmed gain setting of the ADC. Valid values are 128, 64 or 32, which correspond to ADC input scale of ±20 mV, ±40 mV or ±80 mV respectively. Note that setting the ADC gain to 128 or 64 automatically selects ADC input A; gain of 32 automatically selects ADC input B. The value can't be changed during the run-time. Default value is 128. See **Implementation** section for details.

ADC_offset (int32)

Offset value to be subtracted from the ADC raw count reading. This offset can be used to tare weight scale or for zeroing ADC input offset. Default offset value is 0. The value can be changed during the run-time. Its value will be automatically subtracted from all consequent ADC raw count readings. See **Application Programming interface** section for details.

[*] Component was intentionally compiled using Creator 4.0 for compatibility with older versions.

Advanced dialog provides following parameters:



state_check (polling / interrupt)

Selects how the ADC data ready status is detected. Valid options are polling / interrupt. When **interrupt** option is selected, an external interrupt on **irq** pin should be configured by the user. When **polling** option is selected, the **irq** output pin becomes hidden. By default the component is configured for polling. Due to efficient pin polling implementation, taking roughly a single processor clock, there is typically no noticeable performance difference between the interrupt and polling methods. Using the interrupt option may help if another long-running process is blocking ADC polling, or when many HX711 units are running simultaneously in the project. The interrupt technique uses “per-pin” method (not “per-port”), requiring individual interrupt per each instance of the component in the project. This allows for independent configuration of multiple HX711 components using any available pins /ports[*].

[*] Some restrictions apply on PSoC4 chips. See **Pins Assignment** section for details.

Application Programming Interface

Function	Description
`ADC_Start()`	Initialize and start ADC
`ADC_Stop()`	Stop ADC
`ADC_GetResult32()`	Read ADC data
`ADC_Count_to_mV()`	Converts ADC counts to mV
`ADC_SetOffset()`	Set ADC tare offset
Variable	**Description**
`ADC_DataReady`	Flag signaling that data is ready for reading
`ADC_Count`	ADC counts (last read value)
`ADC_Gain`	ADC PGA gain
`ADC_Offset`	ADC tare offset

void ADC_Start()

Description: Initializes component and starts ADC conversion. Note that Delta Sigma ADC conversion needs priming, and first four ADC readings after the start (400ms at 10 Hz) are usually containing errors and must be discarded.

Parameters: none

Return Value: none

void ADC_Stop()

Description: Stops ADC conversion and puts it into low-current (sleep) mode.

Parameters: none

Return Value: none

int32 ADC_GetResult32()

Description: Reads ADC count. This function should be called after ADC signals the data is ready for reading. Reading ADC data clears ADC data buffer; it should be

performed only once after the ADC data is ready. The ADC offset value is subtracted from ADC raw count reading before returning the result.

Parameters: none

Return Value: ADC raw count minus ADC offset.

float ADC_Count_to_mV (int32 value)

Description: Converts ADC count to mV scale according to the selected ADC gain.

Parameters: value – ADC count

Return Value: ADC voltage, mV.

Void ADC_SetOffset (int32 value)

Description: Sets ADC tare offset. This offset value will be automatically subtracted from all consequent ADC raw count readings. This offset can be useful for zeroing ADC (weight scale tare). This value will not automatically adjust upon ADC gain change.

Parameters: value – offset value

Return Value: none

uint8 ADC_DataReady

Description: Flag indicating that ADC data are ready for reading. Read-only.

This flag is available only when polling method is selected. Check this flag in the main() loop to detect when ADC data is ready for reading. The flag is latching – no ADC sampling occurs until the flag is cleared. The flag will reset automatically upon ADC reading.

Return Value: 1 – data is ready, otherwise return value is 0.

int32 ADC_Count

Description: This variable retains ADC count obtained by last call of GetResult32(). Read-only.

Return Value: ADC raw count minus ADC offset.

uint8 ADC_Gain

Description: Returns current ADC gain. Read-only.

Return Value: ADC gain (128, 64 or 32).

int32 ADC_Offset

Description: Returns current ADC offset. Read-only.

Return Value: ADC offset.

Functional Description

The HX711[(*)] is a precision 24-bit analog- to-digital converter (ADC) designed for weigh scales and industrial control applications to interface directly with a bridge sensor [1]. It has ADC, oscillator, bandgap reference and power supply regulator integrated into the single chip. The input multiplexer selects either Channel A or B differential input to the low-noise programmable gain amplifier (PGA). Channel A can be programmed with a gain of 128 or 64, corresponding to a full-scale differential input voltage of ±20mV or ±40mV respectively, when a 5V supply is connected to AVDD analog power supply pin. Channel B has a fixed gain of 32. On-chip power supply regulator eliminates the need for an external supply regulator to provide analog power for the ADC and the sensor. The HX711 is typically available in form of integrated PCB assembly, which includes a transistor for voltage regulation and passive RC network for low-pass filtering the ADC inputs. The weight sensors (load cell) come in various shapes and are can be rated from 0.1 kg to over 1000 kg full scale.

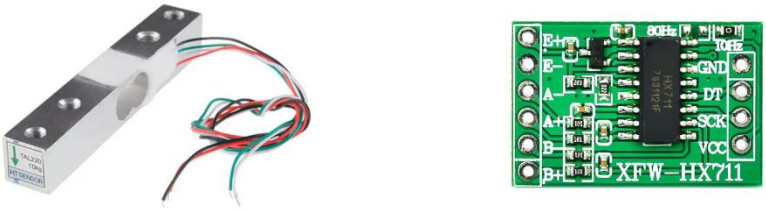


Figure 1. Typical 4-wire load cell (left) and low-cost HX711 PCB assembly (right)

Connection diagram of HX711 is shown on Figure 2. For operation it needs only power (Vcc), and two wires for communicating with MCU: DOUT (data) and SCK (clock).

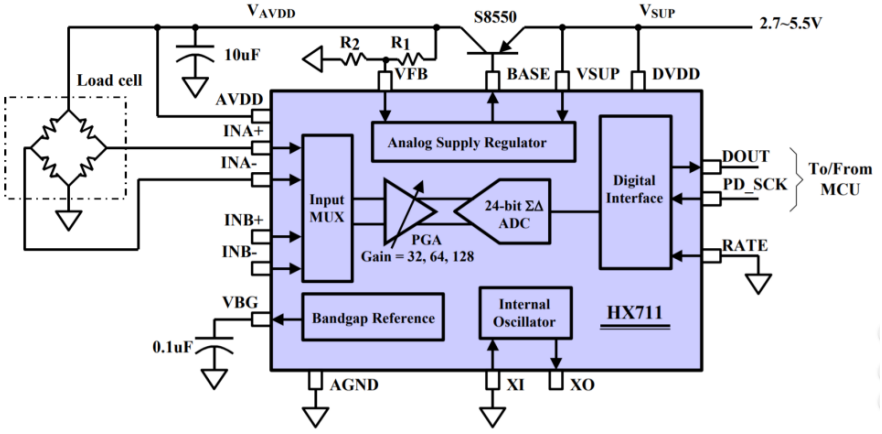


Figure 2. HX711 standard wiring diagram. The LPFs and rate selection resistor are not shown.

[*] Manuf. by AVIA Semiconductor

Implementation

The 2-wire communication algorithm

Component uses 2-wire communication to read ADC data and to set ADC gain [1], implemented entirely in software code. Such approach saves PSoC valuable hardware resources, but is limited to low sampling rates due to the CPU involvement. Fortunately, it is totally sufficient for HX711, operating at low rates (10/80 Hz). The 2-wire communication uses a data pin (SDAT) and a clock pin (SCLK) to clock-in serial data into 24-bit shift-in buffer [1]. It takes 24 clocks to read the data and extra 1 to 3 clocks to set next ADC gain after each data reading[*]:

```
for (i=0; i<24; i++)                        // clock in data
{
    Pin_SCLK_Write(1);                      // toggle the clock
    Pin_SCLK_Write(0);
    result <<= 1;                           // shift-in buffer
    CyDelayCycles(0);                       // add small delay
    if (Pin_SDAT_Read()) result++;          // read next bit
}

for (i=0; i<Sgain; i++)                     // set gain for next measurement
{                                           // =1,2,3 (gain=128,32,64)
    Pin_SCLK_Write(1);                      // toggle the clock
    Pin_SCLK_Write(0);
}

if (CHECK_BIT(result,23))                   // convert final result
   result = result | 0xff000000;            // to signed 32-bit value
```

In the polling mode the DataReady flag is parsed in the main loop. This flag is automatically cleared upon reading the ADC data. For the interrupt mode implementation see demo project provided. Main loop code:

```
for(;;)                                     // main loop
{
  if (ADC_1_DataReady)                      // check for data ready
  {
    int32 result = ADC_1_GetResult32();     // read ADC data (also clears flag)
    . . .
  }
}
```

Timing diagram for ADC single reading cycle is shown on Figure 3. Single ADC conversion period takes processor 9approx. 600 clocks (13 us at 48 MHz BUS_CLK). See **Performance** section for details.

[*] Actual code is optimized for performance and may differ from the one shown.

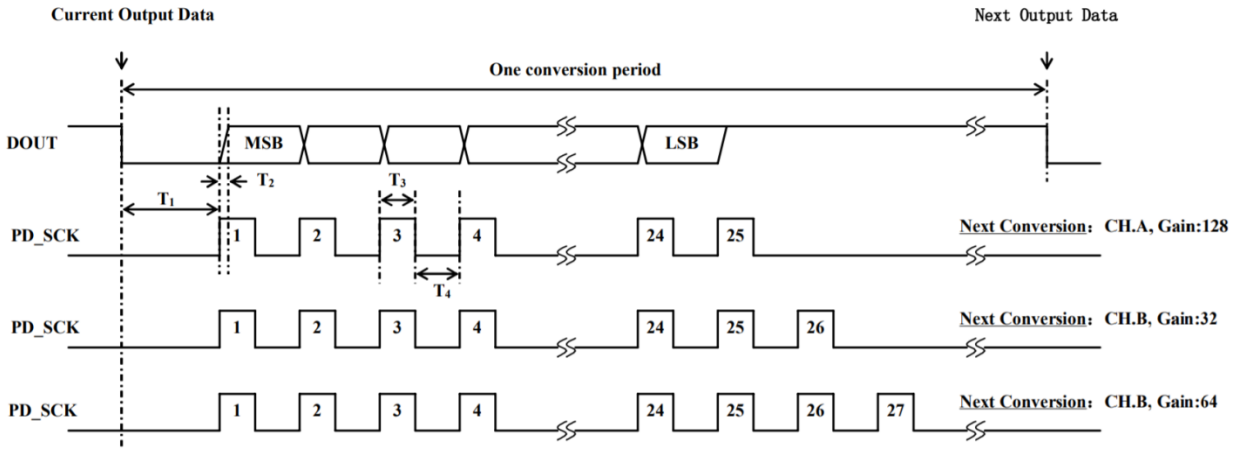


Figure 3. Data reading and gain selection timing [1].

ADC gain and sample rate selection

The HX711 has three available ADC input scales: ±20 mV, ±40 mV and ±80 mV (Table 1). When ADC gain is set to 64 or 128, the ADC samples input A. When the ADC gain is 32, the ADC samples input B.

Table 1. ADC input channel and full scale vs gain.

ADC gain	ADC input	ADC full scale, mV
128	A	±20
64	A	±40
32	B	±80

ADC sampling rate (10/80 Hz) is controlled by the state of the RATE Pin 15. Default rate is 10 Hz. Selecting 80 Hz requires moving a zero-ohm resistor on the PCB assembly (Figure 4).

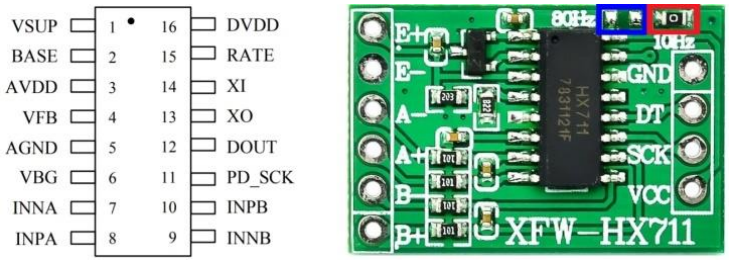


Figure 4. Left: HX711 chip pinout. Right: ADC input channel selection using zero-ohm resistor. Red – selected input is A (default), Blue – selected input is B.

Pins configuration

To communicate with HX711 board, the component utilizes buried pins. By default, the SCLK pin is automatically configured for strong drive, and SDAT pin is configured for high impedance digital; the only job left to user is to assign pins in the Pin Configuration window.

HX711 pins configuration is shown on Figure 5. The pins selection is arbitrary; they can be assigned to any available pins / ports. On PSoC4 certain limitations apply for pins selection in the interrupt mode due to the specific implementation of the component, which involves hardware connection of the SDAT pin to the external interrupt though the logic NOT element (due to reversed ADC logic). In this case the selection of the SDAT pin is limited to the Ports 0 to 3 (other PSoC4 ports do not support hardware connection).

Figure 5. Pins configuration: (a)- component appearance on schematic, (b)- pins assignment. The off-chip annotation component is provided merely for illustration purpose; its presence on the schematic does not affect operation of the component.

Features not implemented

Following features that are not implemented in the current version of the component:

- ADC gain changeable at run-time
- ADC reading timeout
- ADC presence status / disconnect
- Synchronous reading of multiple ADCs

Performance

Component was tested using PSoC5LP (CY8KIT-059) and PSoC4200 (CY8CKIT-042 Pioneer Kit). The component doesn't use UDB, performing all operation entirely by CPU. PSoC5 ADC reading cycle consumes about 600 clocks, making total CPU load very small (~0.01%). Typical results for PSoC5LP and PSoC4 are presented below.

Table 2. Typical CPU clocks consumption by single ADC reading for PSoC5 and PSoC4.

Option	PSoC5	PSoC4
debug(*)	580	1850
release(†)	580	1050

(*) data collected in debug mode with compiler optimization turned off, BUS_CLK=24 MHz

(†) data collected in release mode with compiler optimization set to speed, BUS_CLK=24 MHz

Resources

Component resources consumption is provided below. The component doesn't use UDB 12datapath. Component does not have built-in DMA capabilities.

Table 3. PSoC5 and PSoc4 resources consumption.

Resource	Polling mode	Interrupt mode
interrupts	0	1
PLD macrocells	0	1

Sample Firmware Source Code

Basic component demo is shown in **Appendix 1**. Several other demo projects are provided.

Component Changes

Version	Description of changes	Reason for changes/impact
0.0	Version 0.0 is the first beta release	
0.0.b	Updated datasheet	Rev. A incorrectly described gain and clock setting

References

1. HX711 datasheet, https://cdn.sparkfun.com/datasheets/Sensors/ForceFlex/hx711_english.pdf
2. HX711 MH boards with potential PCB error, https://forum.arduino.cc/index.php?topic=428169.0

Appendix 1

Basic example using the component

Several demo projects provided, showing component use with PSoC4 and PSoC5. Basic example using PSoC4 Pioneer Board (CY8CKIT-042) is shown on Figures 6. The component is configured for polling method, and HX711 is sampling the input A at 10 Hz (default settings). A 1kg-rated load cell was attached to the HX711 board as shown on Figure 7.

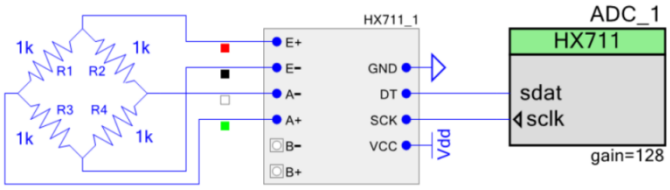


Figure 6. PSoC4 demo schematic. The HX711_1 annotation component and Load Cell resistor bridge are optional off-chip components added for illustration purpose.

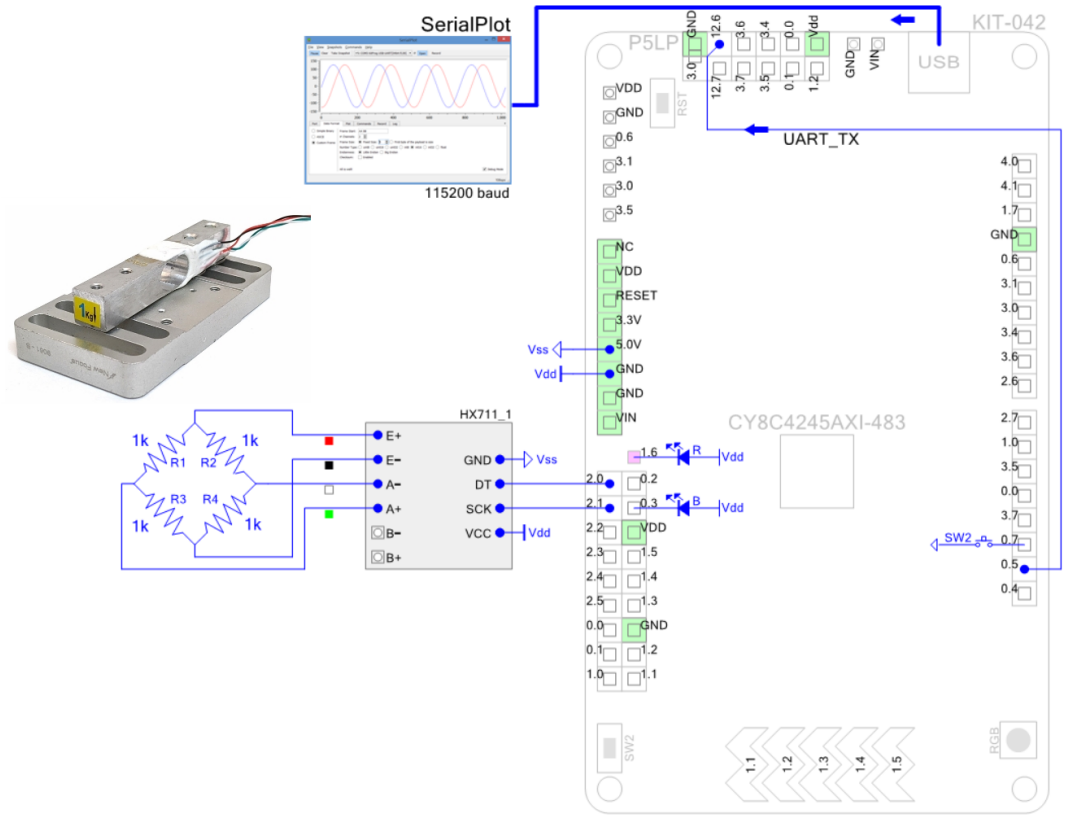


Figure 7. HX711 ADC board connection to the PSoC4 Pioneer Board (CY8CKIT-042) using KIT-042 annotation stub(*).

* KIT-042: Annotation component for CY8CKIT-042 Pioneer Kit, https://community.cypress.com/thread/48742

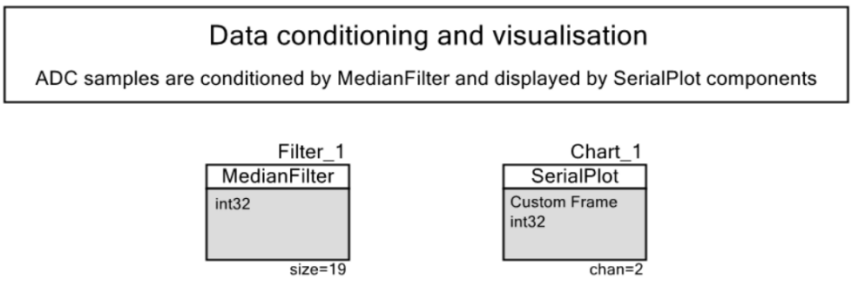


Figure 8. Data conditioning and plotting using MedianFilter and SerialPlot components.

The ADC data are streamed to the SerialPlot charting software(*) or a text terminal using UART communication through USB-UART bridge, built into the KitProg. For that purpose, the PSoC4 UART_Tx output (Pin 0.5) is wired to the PSoC5 USB-UART_Rx (Pin 12.6), which appears on the host computer as a COM port (Figure 7). Custom SerialPlot(†) component handles interface between PSoC and charting software. Prior to output, the raw data from the ADC are conditioned using the MedianFilter component(‡). The length of the filter (19 data points) is causing small (1 sec) delay response, but effectively removes spurious artefacts due to the weight loading procedure, while also reducing statistical noise.

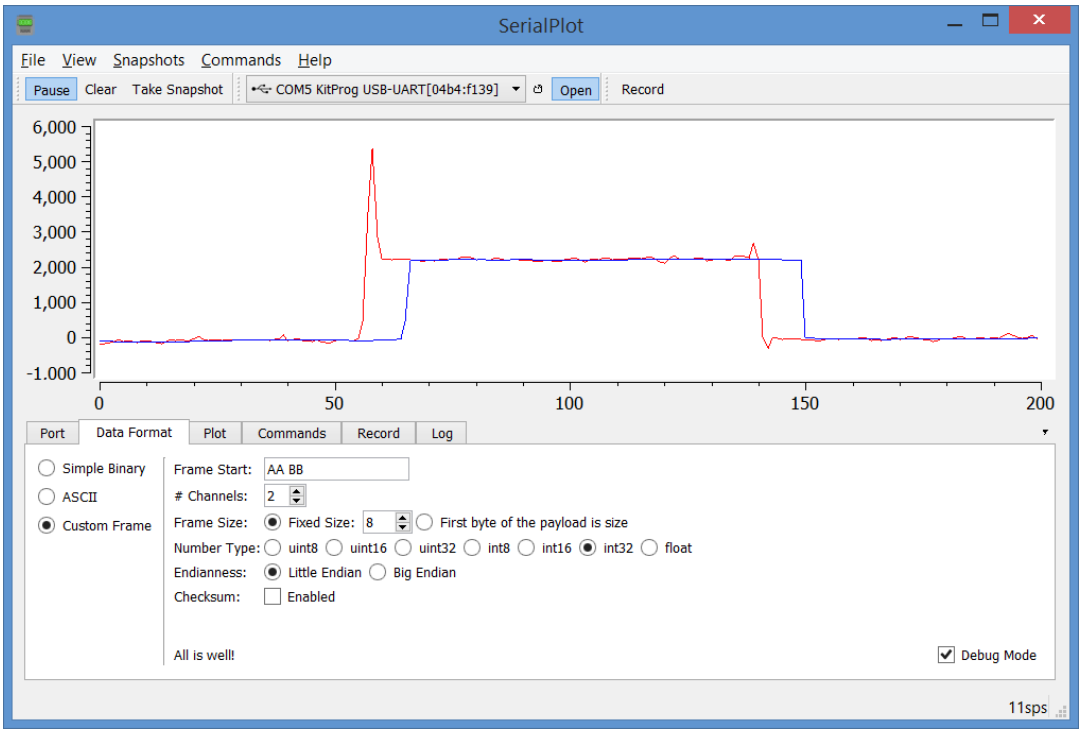


Figure 9. ADC response to 2.5 g weight (1-cent coin). Load cell is rated 1 kg, ADC gain is 128. Red line – ADC raw data, blue line – median filter output.

* SerialPlot – Realtime Plotting Software for UART/Serial Port, https://hasanyavuz.ozderya.net/?p=244
† SerialPlot: interface to real-time data charts, https://community.cypress.com/thread/52310
‡ MedianFilter: sliding window median filter component, https://community.cypress.com/thread/52512

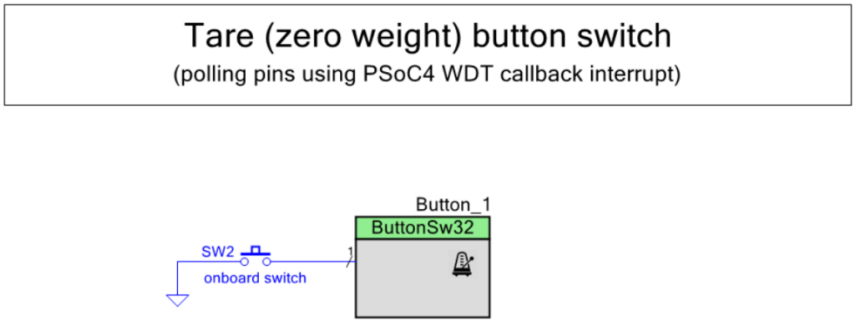


Figure 10. Using ButtonSw32 debouncer component to detect button events.

Project demonstrates a "Tare" option for zeroing the output of the scale. It uses custom debouncing component ButtonSw32[(*)] to detect onboard button press events, and applies compensation offset using component build-in API. Estimated sensitivity of the scale using 1 kg load cell is approx. 920 counts/g, with short-term accuracy of about 27 mg (after the median filter).

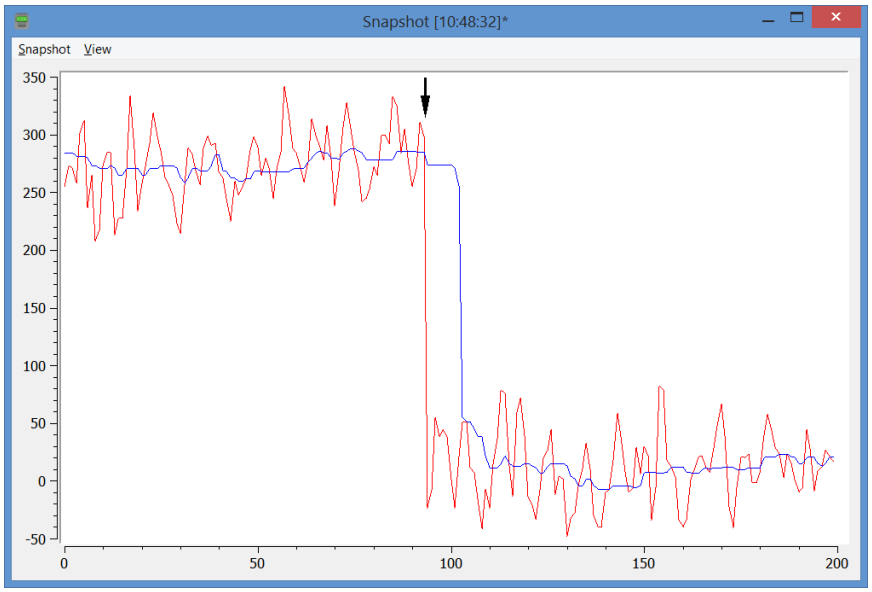


Figure 11. Load scale tare (zero) using a switch button. Button press event is marked by the arrow.

[*] ButtonSw32: button switch debouncer component, https://community.cypress.com/thread/36769

Appendix 2

Off-chip annotation components

The HX711 component is accompanied with off-chip annotation components facilitating schematic drawing and enhancing visibility. They can be used in conjunction with the PSoC Annotation library and KIT-042, KIT-044, and KIT-059 annotation stubs.

Figure 12. HX711 off-chip annotation components: left – full featured; right - compact.

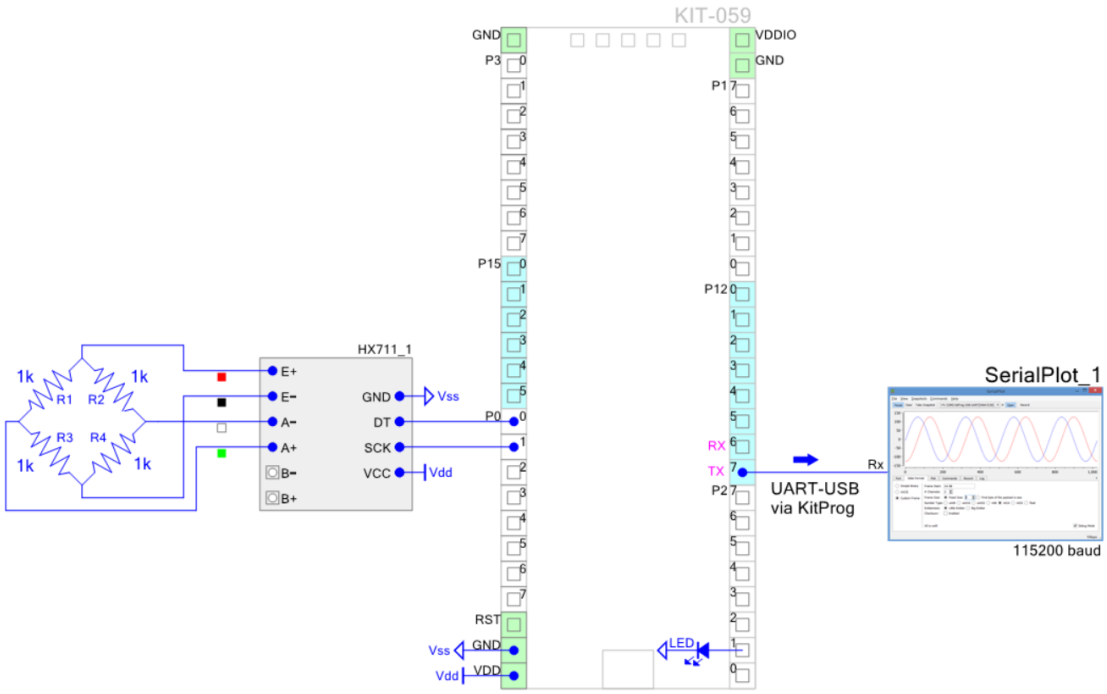


Figure 13. Schematic example for HX711 and CY8CKIT-059 using PSoC Annotation library(*).

* PSoC Annotation Library v1.0, https://community.cypress.com/thread/48049

Appendix 3

Known HX711 PCB design issue

The HX711 PCB assemblies are available from various sources and have some design variations. Many of the low-cost versions share the same defective PCB layout [2], where the E- terminal to the bridge is not physically connected to the ground and floats at about 0.6V. As a result, the bridge excitation pin E+ is not regulated and floats at about 4.9-5V (Vdd). Simple repair solution is to solder a shorting wire between the E- terminal and GND as shown on the Figure 12 (left). Alternative solution, which requires some soldering skills, is to short E- terminal and the bottom ground plane as shown of Figure 12 (right). When the E- terminal is properly grounded, the E+ voltage stabilizes at about 4.3V.

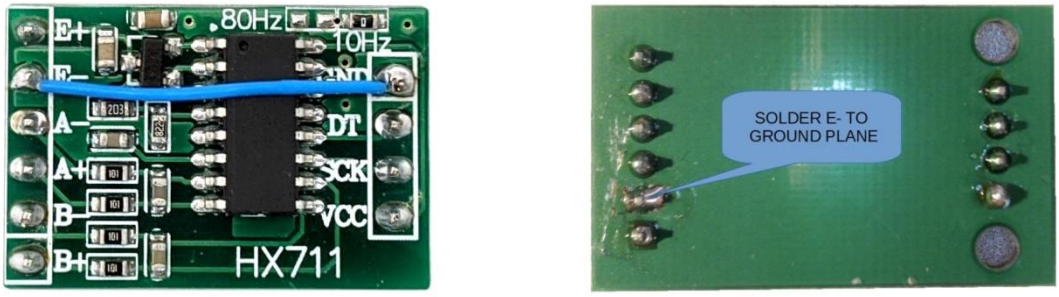


Figure 14. Options for repair HX711 PCB board: left – the E- terminal connected to GNP pin using wire; right – the E- terminal connected directly to the ground plane with a solder blob.