M3 Resistance-to-Digital Conversion (Version 4) Documentation (RDCv4)

Revision 1.0

Yejoong Kim[1], SeokHyeon Jeong[2]

Michigan Integrated Circuits Laboratory, University of Michigan, Ann Arbor

Last Updated: May 31, 2019

[1] yejoong@umich.edu
[2] seojeong@umich.edu

Contents

I. Revision History

A. RDCv3

- Re-designed RDC

B. RDCv4

- Pad Update (rev4)
- Non-Retentive Registers

II. RDC Layer Description

RDC layer is for sensing pressure and it requires external pressure sensor. This layer has been mainly designed to work with C39 (Piezoresistive MEMS device from TDK), but it can also work with other devices as long as it is based on a Wheatstone bridge. Note that using other device than C39 might require change in the register configuration. More details on using different sensor other than C39 can be found in Section IV-F.

- Designed in TSMC180 tsmc18
- Tapedout on May 6th 2019
- Top-Level layout is located at:
  /afs/eecs.umich.edu/vlsida/projects/m3_hdk/virtuoso/TSMC180/RDCv4/layout
- Top-Level LVS CDL is located at:
  /afs/eecs.umich.edu/vlsida/projects/m3_hdk/layer/RDC/RDCv4/cdl/RDCv4.cdl
- Top-Level Finesim CDL is located at:
  /afs/eecs.umich.edu/vlsida/projects/m3_hdk/layer/RDC/RDCv4/ckt/RDCv4.ckt
- MBUS Long Address is 20'h22004

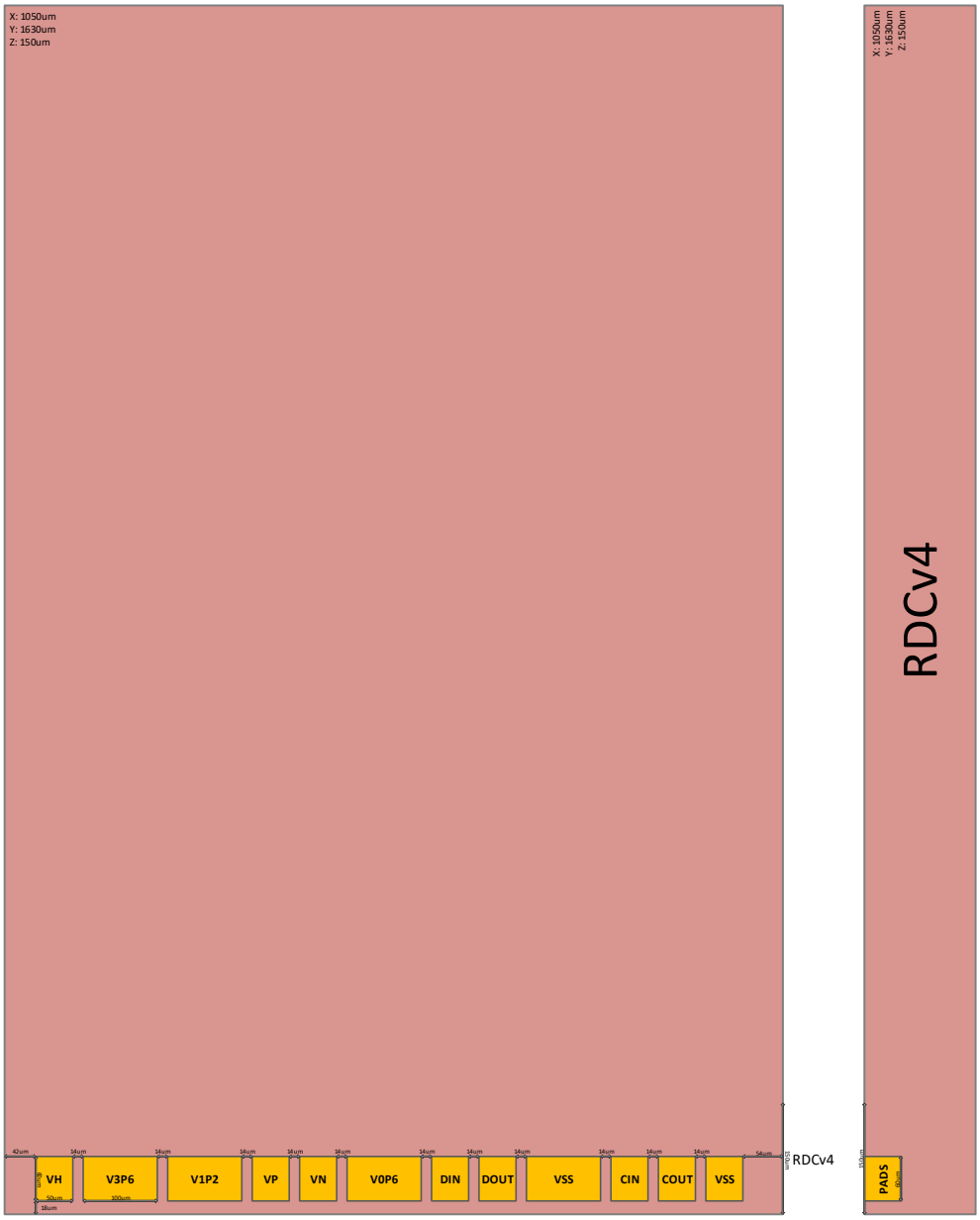
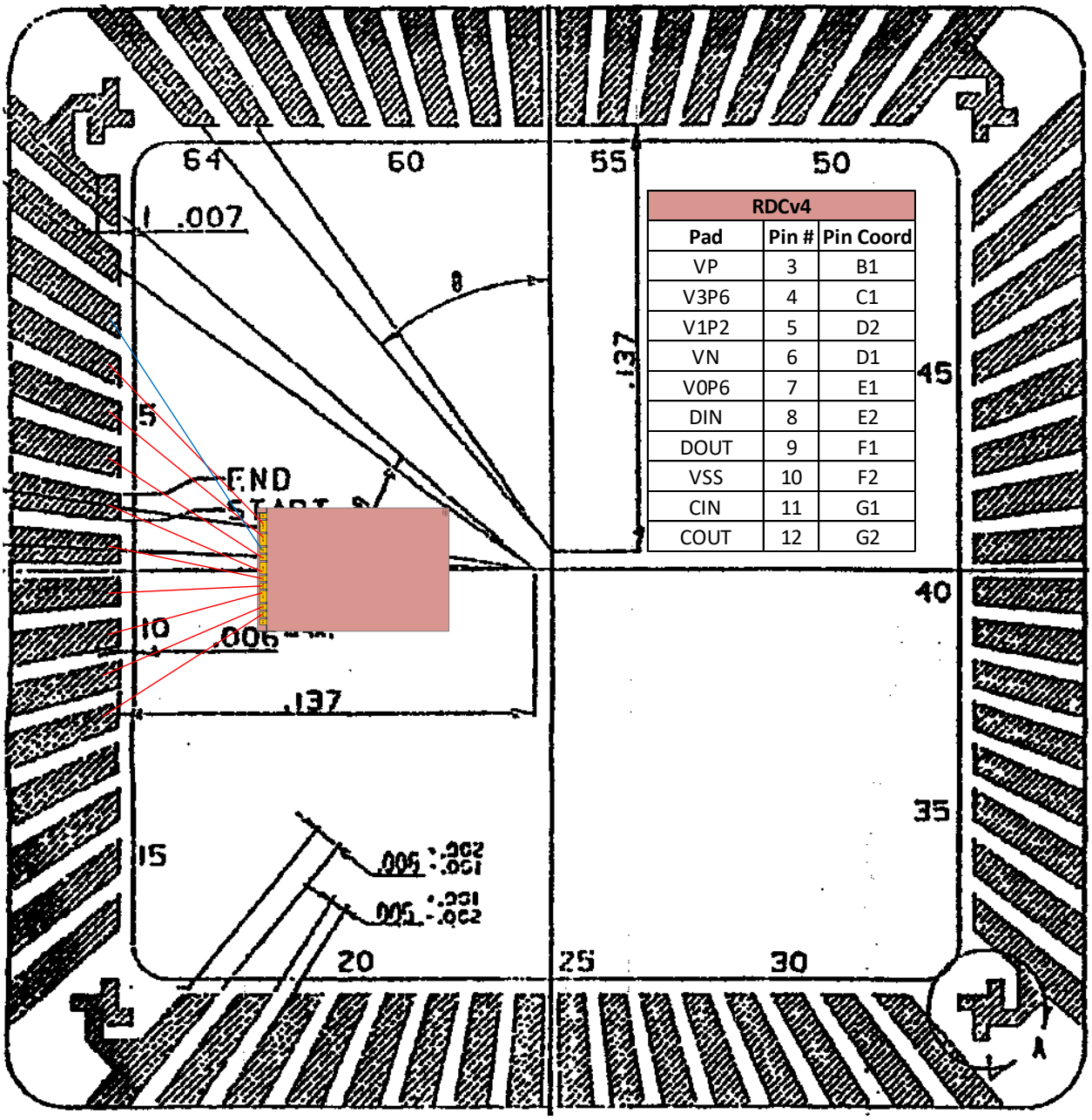


Figure II-i RDC Layer (Version4) (RDCv4) (1050um X 1630um)

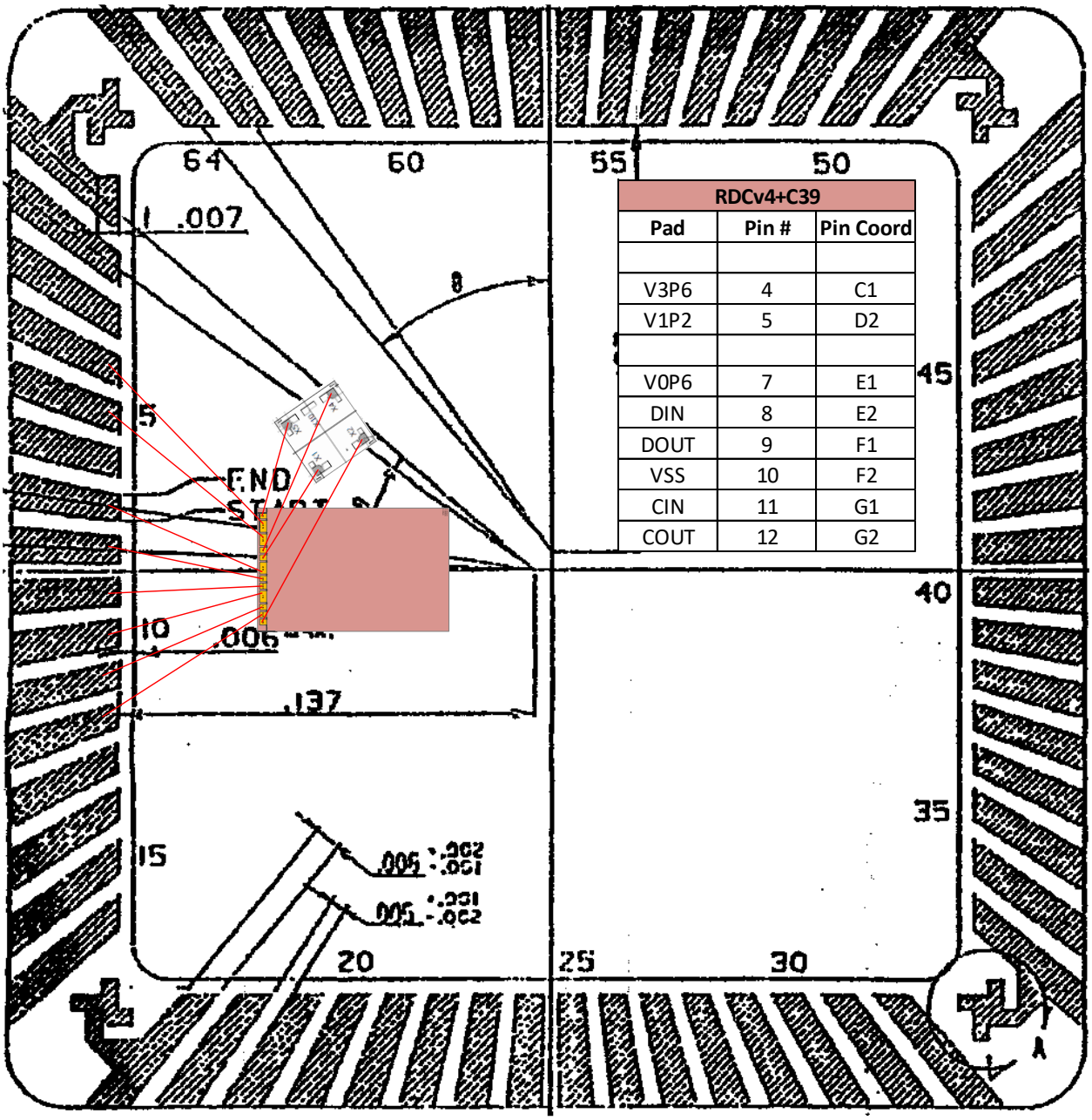
PAD	Type	Description
V3P6	Power	3.6 – 4.2V
V1P2	Power	1.0V - 1.4V
V0P6	Power	0.7V – 0.8V
VSS	Ground	Ground
CIN	MBUS	MBUS clock in
COUT	MBUS	MBUS clock out
DIN	MBUS	MBUS data in
DOUT	MBUS	MBUS data out
VH	Signal	For C39 connection
VP	Signal	For C39 connection
VN	Signal	For C39 connection



RDCv4		
Pad	Pin #	Pin Coord
VP	3	B1
V3P6	4	C1
V1P2	5	D2
VN	6	D1
V0P6	7	E1
DIN	8	E2
DOUT	9	F1
VSS	10	F2
CIN	11	G1
COUT	12	G2

Figure II-ii RDCv4 Wirebonding Diagram

Wirebonding diagram shown in Figure II-iiFigure II-i is for testing RDC layer without C39 where input signals (VP and VN) are wired out. Wirebonded chip can be tested in the 6-socket MBUS board. Use M3_Generic socket. INP is connected to GEN_P3 pin and INN is connected to GEN_P6 pin on the 6-socket MBUS board.



RDCv4+C39		
Pad	**Pin #**	**Pin Coord**
V3P6	4	C1
V1P2	5	D2
V0P6	7	E1
DIN	8	E2
DOUT	9	F1
VSS	10	F2
CIN	11	G1
COUT	12	G2

Figure II-iii RDCv4 + C39 Wirebonding Diagram

Wirebonding diagram shown in **Figure II-iii** is for testing RDC layer with C39. Refer to Section IV-E for details regarding C39 connection. Wirebonded chip can be tested in the 6-socket MBUS board. Use M3_Generic socket.

III. MBUS Register File

Most of the registers are preset and they generally do not require modifications. In the following sub-sections, user-configurable registers are marked with *.

There are different register types which are as follows.

- W : Write. Can be read as well for confirming written values.
- NRI : Write. Non-retentive with isolation. Loses value if the layer goes to sleep.
- R : Read-Only.

A. Register File Mapping

Please see Table III-1.

Address	Register Name	Type	Size & Reset	Bit Field	Module
0x00	WAKEUP_UPON_RDC_	W	1'h0	[6]	
	MBC_WAKEUP_ON_PEND_REQ	W	1'h0	[5]	
	MBC_IGNORE_RX_FAIL	W	1'h1	[4]	
	LC_CLK_DIV	W	2'h2	[3:2]	
	LC_CLK_RING	W	2'h1	[1:0]	
0x01	IRQ_RPLY_SHORT_ADDR	W	8'h10	[15:8]	
	IRQ_RPLY_REG_ADDR	W	8'h07	[7:0]	
0x02	IRQ_RPLY_PYLD_REG_ADDR	W	8'h11	[15:8]	
	IRQ_RPLY_PYLD_LENGTH_1	W	8'h00	[7:0]	
0x10	RDC SIGNATURE	W	24'h022003	[23:0]	
0x11	RDC_DOUT_OS	R	17'h00000	[16:0]	FSM
0x20	RDC_ISOLATE	W	1'h1	[17]	
	RDC_RESETn_FSM	W	1'h0	[15]	
	RDC_CNT_INIT	W	9'h050	[14:6]	
	RDC_CNT_AMP1	W	5'h0C	[5:1]	
	RDC_EN_AZ	W	1'h1	[0]	
0x21	RDC_CNT_AZ_RESETB	W	4'h2	[18:15]	
	RDC_CNT_STORE	W	5'h04	[14:10]	
	RDC_CNT_REDIS	W	5'h08	[9:5]	
	RDC_CNT_AMP2	W	5'h06	[4:0]	
0x22	RDC_CNT_IDLE	W	4'h2	[12:9]	
	RDC_CNT_SKIP	W	2'h1	[8:7]	
	RDC_OSR	W	7'h08	[6:0]	
0x23	RDC_VREF_I_AMP_LC	W	5'h00	[12:8]	V_{CM}_LDO
	RDC_VCM_R_SEL_LC	W	6'h2D	[7:2]	
	RDC_I_VCM_GEN_n_LC	W	2'h2	[1:0]	
0x24	RDC_SEL_DLY	W	4'h5	[17:14]	Bridge
	RDC_SEL_STORE	W	1'h1	[13]	Amplifier
	RDC_SEL_VB2_LC	W	5'h04	[12:8]	
	RDC_SEL_VB3b_LC	W	3'h3	[7:5]	
	RDC_SEL_GAIN_LC	W	5'h16	[4:0]	

0x25	RDC_OFFSET_P_LC	W	5'h16	[12:8]	
	RDC_OFFSET_PB_LC	W	5'h09	[7:3]	
	RDC_I_AMP_BIASGEN	W	3'h3	[2:0]	
0x26	RDC_OFFSET_SELN_B_LC	NRI	3'h7	[7:5]	ADC
	RDC_OFFSET_SELP_B_LC	NRI	3'h7	[4:2]	
	RDC_SEL_ADC_MODE	NRI	1'h0	[1]	
	RDC_ENb_DWA	NRI	1'h0	[0]	
0x27	RDC_I_BUF_VH_n_LC	NRI	3'h5	[2:0]	V_H_LDO
0x28	RDC_RESET_RC_OSC	NRI	1'h1	[18]	Clock_Gen
	RDC_R_REF	NRI	4'h8	[17:14]	
	RDC_I_BUF	NRI	4'h0	[13:10]	
	RDC_I_BUF2	NRI	4'h0	[9:6]	
	RDC_I_CMP	NRI	4'h1	[5:2]	
	RDC_I_MIRROR	NRI	2'h1	[1:0]	
0x29	RDC_PDIFF	NRI	15'h0200	[14:0]	
0x2A	RDC_POLY	NRI	24'h00400	[23:0]	
0x2B	RDC_MIM	NRI	3'h4	[6:4]	
	RDC_MOM	NRI	3'h4	[3:1]	
	RDC_CLK_ISOLATE	NRI	1'h1	[1]	
0x2C	RDC_EN_PG_FSM	W	1'h1	[10]	Power-gate Control
	RDC_EN_PG_AMP_V1P2	W	1'h1	[9]	
	RDC_EN_PG_ADC_V1P2	W	1'h1	[8]	
	RDC_EN_PG_BUF_VH_V1P2	W	1'h1	[7]	
	RDC_EN_PG_RC_OSC	W	1'h1	[6]	
	RDC_ENb_PG_VREF	W	1'h0	[5]	
	RDC_ENb_PG_AMP_VBAT	W	1'h0	[4]	
	RDC_ENb_PG_ADC_VBAT	W	1'h0	[3]	
	RDC_ENb_MIRROR_LDO	W	1'h1	[2]	
	RDC_ENb_PG_BUF_VCM	W	1'h0	[1]	
	RDC_ENb_PG_BUF_VH_VBAT	W	1'h0	[0]	

Table III-1 Register File Mapping

B. Register Descriptions

WAKEUP_UPON_RDC_IRQ

MBC_WAKEUP_ON_PEND_REQ

MBC_IGNORE_RX_FAIL

LC_CLK_DIV

Layer Controller's Clock Divider. See **Table III-2**. Used in conjunction with **LC_CLK_RING**.

Value	Clock Division
2'h0	8

2'h1	4
2'h2	2
2'h3	0

Table III-2 LC_CLK_DIV Register

LC_CLK_RING

Layer Controller's Ring Oscillator's Speed. See **Table III-3** & **Table III-4** for post-pex simulation results. Use in conjunction with **LC_CLK_DIV**.

27°C						
V0P6 (V)	0.45	0.50	0.55	0.60	0.65	0.70
V1P2 (V)	0.90	1.00	1.10	1.20	0.30	1.40
LC_CLK_RING: 2'h3						
Frequency (KHz)	38.6	64.7	95.4	129	166	204
V0P6 Power (nW)	7.64	17.2	33.3	58.0	93.2	141
V1P2 Power (nW)	3.78	6.62	12.0	19.6	29.7	43.0
V0P6 Leakage (pW)	94.0	107	121	135	150	166
V1P2 Leakage (pW)	≈0	≈0	≈0	≈0	≈0	≈0
LC_CLK_RING: 2'h2						
Frequency (KHz)	31.8	53.3	78.6	107	137	169
V0P6 Power (nW)	6.30	14.2	27.5	47.8	76.8	116
V1P2 Power (nW)	2.60	5.60	9.90	16.0	24.4	35.5
V0P6 Leakage (pW)	96.8	110	125	140	154	171
V1P2 Leakage (pW)	≈0	≈0	≈0	≈0	≈0	≈0
LC_CLK_RING: 2'h1						
Frequency (KHz)	27.1	45.4	67.0	90.8	116	144
V0P6 Power (nW)	5.4	12.1	23.5	40.8	65.6	99.2
V1P2 Power (nW)	2.4	4.7	8.6	13.6	20.9	30.3
V0P6 Leakage (pW)	96.8	110	125	140	155	171
V1P2 Leakage (pW)	≈0	≈0	≈0	≈0	≈0	≈0
LC_CLK_RING: 2'h0						
Frequency (KHz)	24.1	40.4	59.7	80.9	104	128
V0P6 Power (nW)	4.9	10.8	20.9	36.3	58.4	88.4
V1P2 Power (nW)	2.0	4.2	7.5	12.1	18.7	27.0
V0P6 Leakage (pW)	99.4	114	129	144	159	177
V1P2 Leakage (pW)	≈0	≈0	≈0	≈0	≈0	≈0

Table III-3 LC_CLK_RING Register Post-Pex Results with LC_CLK_DIV =2'h3 (27°C) (TT)[3]

40°C						
V0P6 (V)	0.45	0.50	0.55	0.60	0.65	0.70
V1P2 (V)	0.90	1.00	1.10	1.20	0.30	1.40
LC_CLK_RING: 2'h3						
Frequency (KHz)	43.4	70.0	100	134	170	207
V0P6 Power (nW)	8.9	19.2	36.2	61.6	97.5	145
V1P2 Power (nW)	3.7	7.3	12.7	20.4	30.7	43.9

[3] Leakage Power does not include power-gating

V0P6 Leakage (pW)	189	216	244	273	303	335
V1P2 Leakage (pW)	≈0	≈0	≈0	≈0	≈0	≈0
LC_CLK_RING: 2'h2						
Frequency (KHz)	35.7	57.6	83.0	111	140	171
V0P6 Power (nW)	7.4	15.9	29.9	50.9	80.4	120
V1P2 Power (nW)	3.0	6.0	10.6	16.8	25.3	36.2
V0P6 Leakage (pW)	196	223	252	282	313	346
V1P2 Leakage (pW)	≈0	≈0	≈0	≈0	≈0	≈0
LC_CLK_RING: 2'h1						
Frequency (KHz)	30.4	49.1	70.8	94.3	119	145
V0P6 Power (nW)	6.4	13.6	25.5	44.5	68.7	103
V1P2 Power (nW)	2.5	5.2	8.9	14.4	21.7	31.1
V0P6 Leakage (pW)	195	220	252	282	313	346
V1P2 Leakage (pW)	≈0	≈0	≈0	≈0	≈0	≈0
LC_CLK_RING: 2'h0						
Frequency (KHz)	27.1	43.7	63.0	83.9	106	129
V0P6 Power (nW)	5.7	12.1	22.8	38.7	61.2	91.3
V1P2 Power (nW)	2.4	4.6	8.1	12.8	19.3	27.7
V0P6 Leakage (pW)	202	230	260	291	323	357
V1P2 Leakage (pW)	≈0	≈0	≈0	≈0	≈0	≈0

Table III-4 LC_CLK_RING Register Post-Pex Results LC_CLK_DIV =2'h3 (40°C) (TT)[4]

IRQ_RPLY_SHORT_ADDR

IRQ_RPLY_REG_ADDR

IRQ_RPLY_PYLD_REG_ADDR

IRQ_RPLY_PYLD_LENGTH_1

RDC SIGNATURE

RDC_DOUT_OS
Oversampled RDC output code

RDC_ISOLATE
Isolates RDC output.

RDC_RESETn_FSM
Reset signal for FSM.

[4] Leakage Power does not include power-gating

RDC_CNT_INIT

Sets initial startup time. RDC uses switched-capacitor based amplifier in the frontend which requires startup time to build common mode voltages at the amplifier output. Startup time is required only for the initial conversion and therefore does not show up afterwards. If this value is set too low, first few conversions will not be accurate. Setting it too high will increase initial conversion time.

RDC_CNT_AMP1

Sets duration of the amplification phase in the first stage amplifier. Setting the value too low will induce error. Increasing this value will reduce error by giving more stabilization time but after certain point improvement will be minimal and will only increase conversion time.

RDC_EN_AZ

Enables auto-zeroing in the VH Buffer.

RDC_CNT_AZ_RESETB

Sets start time of the auto-zeroing in VH_LDO. Auto-zeroing should be done during the first stage amplifier amplification phase (i.e. before beginning of 2^{nd} amplifier amplification phase). As offset shifting operation (Store and Redistribution of charges) which requires VH_LDO output should be ready within the same phase, start time should be set so that there is still enough time left for the offset shifting operation to happen. Also, setting this time too early will result in an error due to finite VH_LDO bandwidth (i.e. requires some stabilization time to follow input (VH) at the output). See IV-D for detailed timing diagram.

RDC_CNT_STORE

Sets when to begin offset storing. Storing should take place during the first stage amplifier amplification phase (i.e. before beginning of 2^{nd} amplifier amplification phase). Note that redistribution (controlled by **RDC_CNT_REDIS**) has to take place after this storing operation within the same amplification phase which sets upper limit. Lower limit is determined by auto-zeroing operation (controlled by **RDC_CNT_AZ_RESETB**) in the VH_LDO. See IV-D for detailed timing diagram.

RDC_CNT_REDIS

Sets when to begin charge redistribution. Redistribution should take place during the first stage amplifier amplification phase (i.e. before beginning of 2^{nd} amplifier amplification phase) but after storing operation (controlled by **RDC_CNT_STORE**). See IV-D for detailed timing diagram.

RDC_CNT_AMP2

Sets duration of the amplification phase in the second stage amplifier. Setting the value too low will induce error. Increasing this value will reduce error by giving more stabilization time but after certain point improvement will be minimal and will only increase conversion time.

RDC_CNT_IDLE

Sets idle time between consecutive measurement. Takes effect when **RDC_OSR** >1

RDC_CNT_SKIP

Sets number of samples to be discarded from the beginning.

RDC_OSR

Sets oversampling ratio. Number of accumulate samples at the **RDC_DOUT_OS** is **RDC_OSR** - **RDC_CNT_SKIP**.

RDC_VREF_I_AMP_LC

Controls current consumed in the body buffers used in the voltage reference.

RDC_VCM_R_SEL_LC

Adjusts common mode voltage.

RDC_I_VCM_GEN_n_LC

Controls current consumed in the VCM buffer.

RDC_SEL_DLY

Controls excitation time of the external transducer (i.e. Wheatstone Bridge). **Table III-5** shows required excitation time of C39 depending on their bridge resistance and resulting energy consumption. Bridge resistance values are from C39 datasheet. Based on the table, required excitation time ranges from 300 to 450nS. **Figure III-i** shows simulated (after PEX) excitation time depending on the **RDC_SEL_DLY**. Note that if **RDC_SEL_DLY** is too low, output code will not have linear relationship with input pressure due to settling issue and this will show up as a measurement error. On the other hand, if the value is too high, voltage drop increases which will degrade resolution. It will also increase energy consumption.

	Min	Typ	Max
Bridge Resistance (kΩ)	4.8	6.0	7.2
RC Time Constant (nS)	36.00	45.00	54.00
Required Excitation Time (nS) Targets 12b Accuracy	299.44	374.30	449.16
Current consumption (µA)	875.00	700.00	583.33
Voltage Drop (mV)	218.34		
Charge (pC)	262.01		
Energy (nJ)	0.94		

Table III-5 Required excitation time and energy consumption vs C39 bridge resistance (Min, Typ, Max)

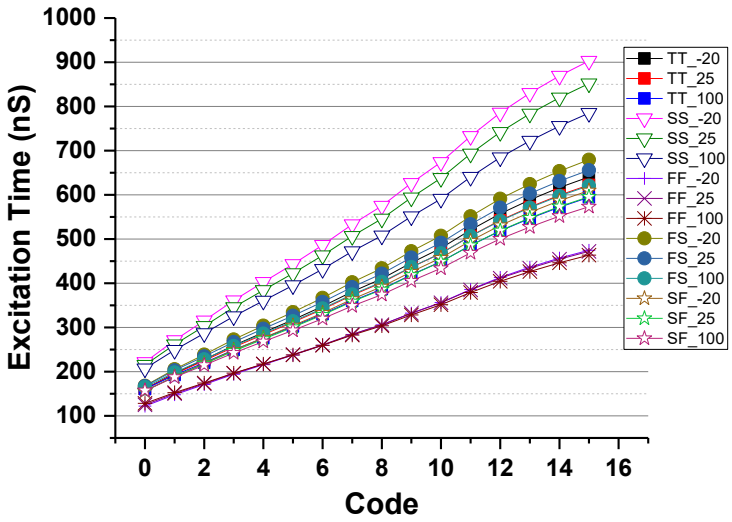


Figure III-i Excitation time vs code

RDC_SEL_STORE
Controls how store operation is triggered (0: From AMP, 1: From FSM).

RDC_SEL_VB2_LC
Adjusts bias voltage for the nMOS cascade in the amplifier bias voltage generator.

RDC_SEL_VB3b_LC
Adjusts bias voltage for the pMOS cascade in the amplifier bias voltage generator.

RDC_SEL_GAIN_LC
Controls gain of the second stage amplifier. Due to variation in the sensitivity of C39, RDC has been designed to have variable gain. Gain value should be set based on the desired resolution and pressure range. Base value is 18 (i.e. second stage gain = 18 + **RDC_SEL_GAIN_LC**).

RDC_OFFSET_P_LC
Controls offset shift amount (negative side).

RDC_OFFSET_PB_LC
Controls offset shift amount (positive side). It should be complementary to **RDC_OFFSET_P_LC** (i.e. **RDC_OFFSET_PB_LC** = 32 -**RDC_OFFSET_P_LC**).

RDC_I_AMP_BIASGEN
Controls current level of the amplifier bias voltage generator

RDC_OFFSET_SELN_B_LC
Comparator offset control in ADC

RDC_OFFSET_SELP_B_LC
Comparator offset control in ADC

RDC_SEL_ADC_MODE
Controls ADC operation modes (0 : RDC 1: Debug).

RDC_ENb_DWA
Enables dynamic weight averaging (DWA). It helps to achieve better linearity with oversampling.

RDC_I_BUF_VH_n_LC
Controls current bias for VH Buffer.

RDC_RESET_RC_OSC
Resets on-chip clock.

RDC_R_REF
Controls resistance of the current source.

RDC_I_BUF
Adjust the buffer bias current (Current REF).

RDC_I_BUF2
Adjusts the buffer bias current (Composite resistor).

RDC_I_CMP
Adjusts the comparator bias current. Increasing bit increases comparator speed and thus improves temperature coefficient (TC) of the reference clock (CLK_{REF}) at the cost of higher power consumption. TC improves until comparator delay becomes negligible to the overall period.

RDC_I_MIRROR
Adjusts main current

RDC_PDIFF
Adjusts P+ diffusion resistor size. Increasing the value (decreasing the resistance) will make period more CTAT.

RDC_POLY
Adjust P+ poly resistor size. Increasing the value (decreasing the resistance) will make period more PTAT.

RDC_MIM
Adjusts MIM cap size. Increasing the value decreases clock frequency.

RDC_MOM
Adjusts MOM cap size. Increasing the value decreases clock frequency.

RDC_CLK_ISOLATE
Isolates clock output.

RDC_EN_PG_FSM
Power gates FSM from V1P2.

RDC_EN_PG_AMP_V1P2
Power gates amplifier control block from V1P2.

RDC_EN_PG_ADC_V1P2
Power gates ADC control block from V1P2.

RDC_EN_PG_BUF_VH_V1P2
Power gate VH buffer control block from V1P2.

RDC_EN_PG_RC_OSC
Power gates on-chip clock gen from V1P2.

RDC_ENb_PG_VREF
Power gates voltage reference from VBAT.

RDC_ENb_PG_AMP_VBAT
Power gates amplifier from VBAT.

RDC_ENb_PG_ADC_VBAT
Power gates ADC from VBAT.

RDC_ENb_MIRROR_LDO
Power gates mirror that supplies current to VH buffer from VBAT.

RDC_ENb_PG_BUF_VCM

Power gates VCM buffer from VBAT

RDC_ENb_PG_BUF_VH_VBAT

Power gate VH buffer from VBAT

IV. RDC

This is a pressure sensor designed by Seokhyeon Jeong. Off-chip transducer along with the on-chip resistance-to-digital converter (RDC) converts pressure into a digital code. RDC has been mainly designed to be used with C39 (TDK, B58600E3914B637) which is a Wheatstone bridge that converts pressure change into a resistance change. Pressure range of C39 is from 0 – 900mmHg. RDC can be also used with different transducer as long as they are based on a Wheatstone bridge. Refer to Section IV-F for using RDC with different transducer.

Figure IV-i show overall block diagram of the sensor. Signal chain consists of C39, 2-stage amplifier and 10b ADC. 2-stage amplifier includes a low noise amplifier (1st stage) followed by a programmable gain amplifier (2nd stage). Between 1st and 2nd stage amplifier, offset shift is performed to enable sub-ranging (see IV-D for details). Other peripherals include V_H_LDO, V_{CM}_LDO, Clock Gen and Digital Logic.

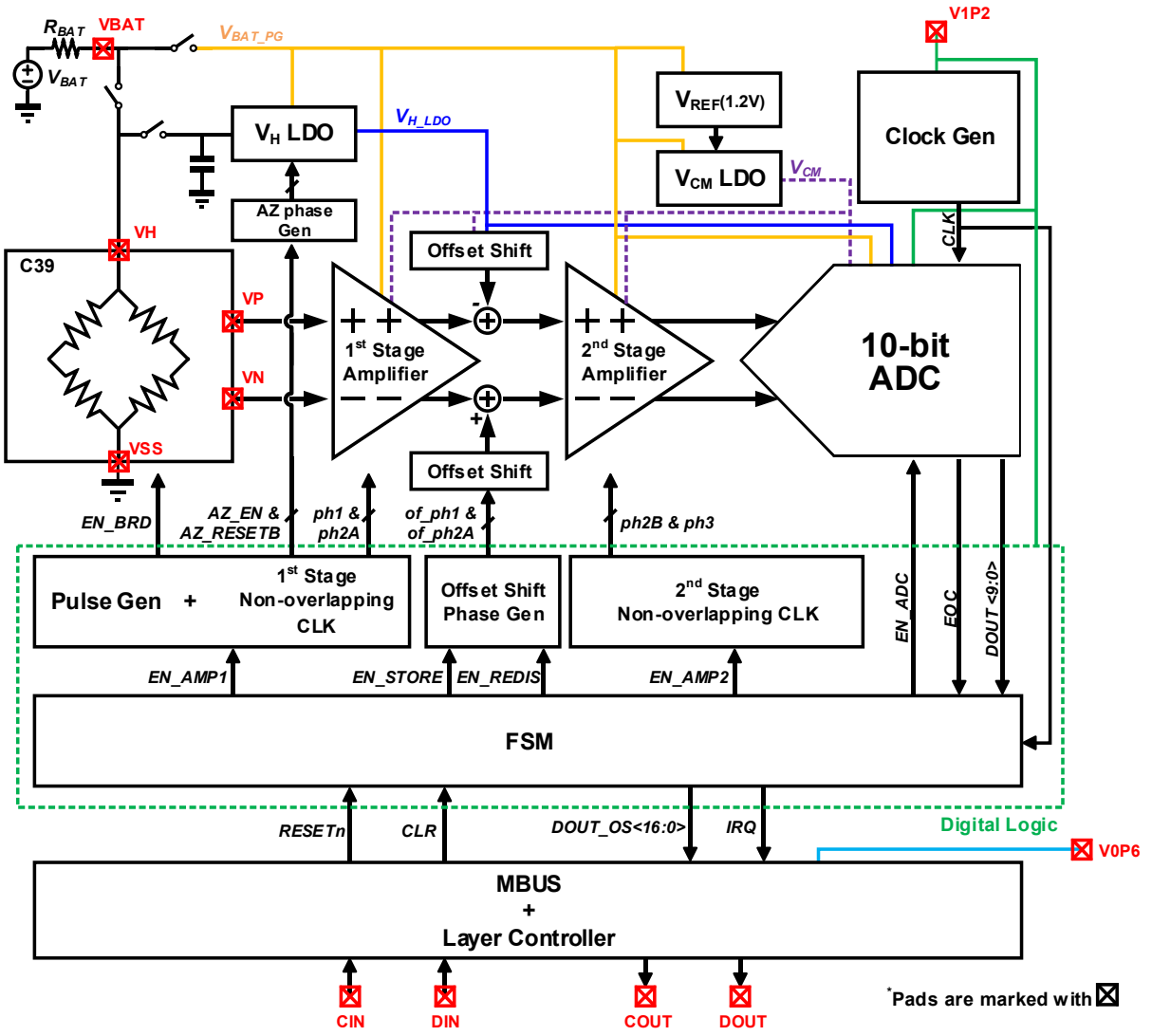


Figure IV-i RDC Block Diagram

A. Power Domains

- VDD_BAT (3.6V - 4.2V)
- VDD_1P2 (1.0V - 1.4V)

B. Interrupt Considerations

Interrupts after IRQ is asserted. Interrupts when request is serviced.

C. Power Draw / Current Draw / Energy Consumption

Mode	V1P2 (1.2V)	VBAT (4.0V)
Active	220nA	2.2µA
Standby	0.36pA	1.47pA

Table IV-1 RDC Current Draw

D. Operation

- Sub-ranging

RDC uses sub-ranging to maximize resolution at the expense of pressure range. This sub-ranging window is designed to be 300mmHg or less in order to achieve resolution of 0.3mmHg with C39 at oversampling ratio of 8. **Figure IV-ii** shows one example of sub-ranging operation with 300mmHg window. It can be seen that different pressure range requires different amount of offset. Upper pressure range needs higher offset in order to reside within the output dynamic range of the amplifier.

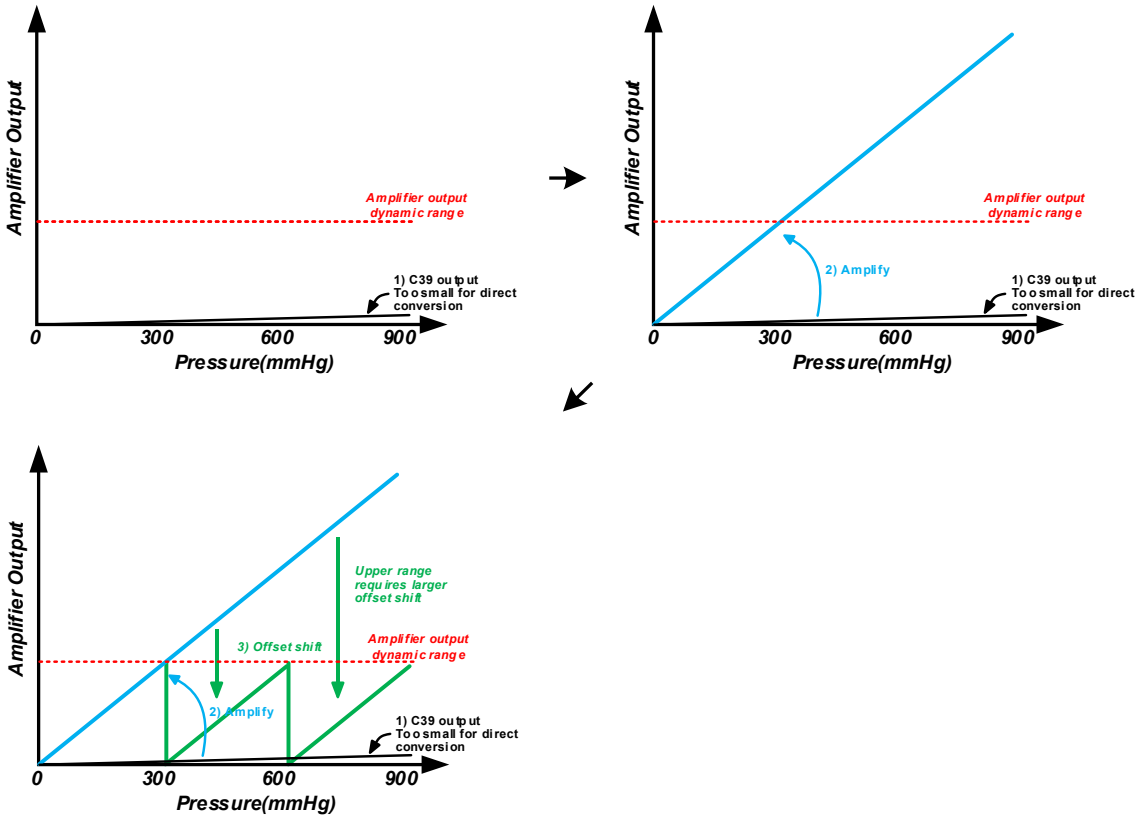


Figure IV-ii Sub-ranging Operation

Note that window size can be increased at the expense of resolution by adjusting the gain of the amplifier (Figure IV-iii). Based on a desired pressure range and resolution, gain (RDC_SEL_GAIN_LC) and offset (RDC_OFFSET_P_LC/RDC_OFFSET_PB_LC) should be set accordingly. As RDC output code increases linearly with pressure, proper configuration can be found using 2-pt calibration.

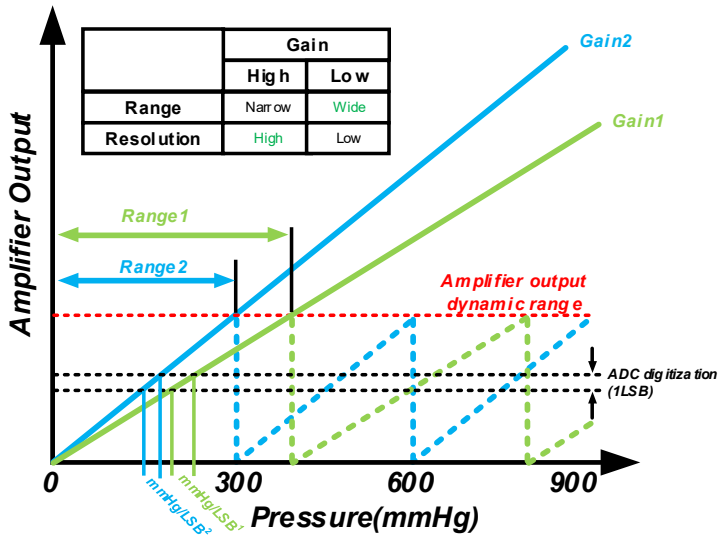


Figure IV-iii Gain vs Pressure range & Resolution

- 2-point calibration

  RDC has linear relationship between its output code and pressure. As a result, proper gain(RDC_SEL_GAIN_LC) and offset (RDC_OFFSET_P_LC/RDC_OFFSET_PB_LC) can be found via 2-point calibration.

  Following calibration procedure is one example which targets 900mmHg pressure range with 300mHg sub-ranging window. 2-point calibration is performed at 300mHg and 600mHg by sweeping gain and offset values at each pressure. This will generate 2-D table of output codes for each pressure. Based on the table, gain and offset should been chosen so that the output varies ~700 codes across 300mmHg where lower code should be ~200 and upper code should be ~900. This will guarantee 0.3mmHg of resolution with OSR of 8 (with C39). After finding proper gain and offset value for this pressure range (300-600mHg), offset values for upper (600-900mmHg) and lower (90-300mmHg) pressure range should be chosen. Upper offset value can be found based on the table generated at 600mmHg which makes output code to be ~200. Similarly, lower offset value can be found based on the table generated at 300mHg which makes output code to be ~900. Following table shows average output code vs offset at different pressure range with fixed gain.

	RDC_OFFSET_P_LC			
Pressure (mmHg)	4	5	6	7
90	476.29	383.14	291.56	197.83
120	549.32	455.81	363.38	270.02
150	621.59	528.87	436.19	342.6
180	694.43	600.86	508.44	415.21

210	767.74	674.49	582.52	488.24
240	839.9	747.03	655.22	561.37
270	913.97	819.73	727.75	634.01
300	986.72	891.59	800.8	708.22

	RDC_OFFSET_P_LC			
Pressure (mmHg)	B	C	D	E
300	334.29	239.6	146.01	53.92
330	407.35	311.98	219.57	126.72
360	480.41	386.56	292.68	200.65
390	554.35	459.72	365.96	273.95
420	627.06	533.35	438.99	347.05
450	700.28	606.17	513.63	421.17
480	774.53	680.36	587.16	494.21
510	847.19	753.15	660.6	568.34
540	922.8	826.42	733.85	642.65
570	995.92	900.9	808.35	716.27
600	1023	975.99	880.4	789.22

	RDC_OFFSET_P_LC			
Pressure (mmHg)	13	14	15	16
600	322.98	228.34	135.13	42.55
630	396.81	302.12	208.89	115.71
660	470.52	375.72	282.99	190.39
690	545.13	450.96	357.05	265.18
720	618.7	525.18	430.83	339.05
750	693.02	598.87	504.56	412.69
780	766.82	672.93	579.46	486.81
810	840.41	746.32	653.46	560.87
840	915.23	819.43	726.94	635.09
870	989.03	893.19	800.96	709.43
900	1023	969.3	874.21	783.46

Table IV-2 RDC average output code vs offset at different pressure range (RDC_SEL_GAIN_LC=7, RDC_OSR=4, RDC_CNT_SKIP=3)

Figure IV-iv(left) shows pressure error across 20 chips after 2-pt calibration. It shows common 2nd order behavior which comes from C39 itself. After batch calibration (**Figure IV-iv**, right), 2nd order error can be removed. Note that for narrow pressure range (≤300mmHg), batch calibration might not be necessary and 1st order calibration (i.e. 2pt calibration) could be enough.

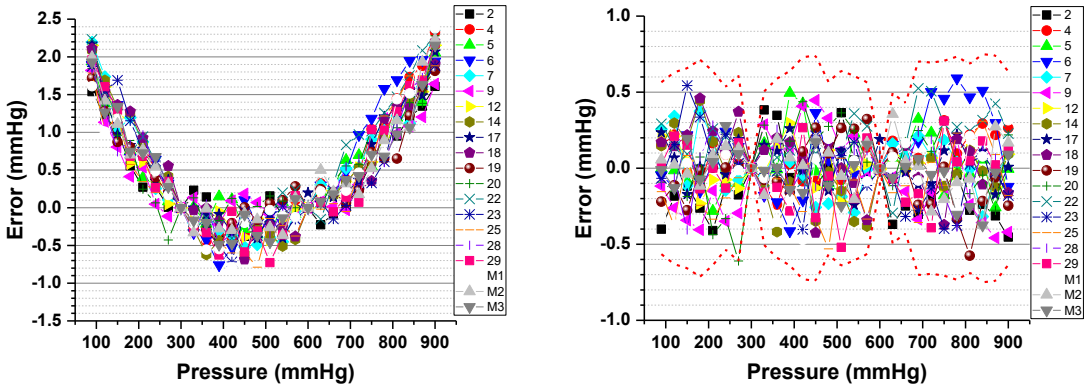


Figure IV-iv Pressure accuracy after 2-pt calibration (left) and after 2pt + batch calibration (right)

- Oversampling (OSR)
  RDC can accumulate its output up to 127 by adjusting RDC_OSR. Increasing OSR will provide better resolution at the expense of longer conversion time.

RDC_OSR	Resolution (mmHg, 1σ)
1	0.97
2	0.70
4	0.53
8	0.33
16	0.24
32	0.18
64	0.12
127	0.09

Table IV-3 RDC OSR vs resolution

- Start-up sequence
  1. Un-powergate voltage reference. Voltage reference consumes only tens of pA so it requires some stabilization time (~30ms). 10 MBUS delay required afterwards.
     RDC_ENb_PG_VREF -> 1'h1

  2. Un-powergate blocks on V1P2. This includes following signals. 1 MBUS delay required afterwards.
     RDC_EN_PG_FSM -> 1'h0
     RDC_EN_PG_AMP_V1P2 -> 1'h0
     RDC_EN_PG_ADC_V1P2 -> 1'h0
     RDC_EN_PG_BUF_VH_V1P2 -> 1'h0
     RDC_EN_PG_RC_OSC -> 1'h0

  3. Un-powergate blocks on VBAT. This includes following signals. 1 MBUS delay required afterwards.
     RDC_ENb_PG_AMP_VBAT -> 1'h1

RDC_ENb_PG_ADC_VBAT -> 1'h1
RDC_ENb_MIRROR_LDO -> 1'h0
RDC_ENb_PG_BUF_VCM -> 1'h1
RDC_ENb_PG_BUF_VH_VBAT -> 1'h1

4. Start clock. 1 MBUS delay required afterwards.
   RDC_RESET_RC_OSC -> 1'h0
   RDC_CLK_ISOLATE -> 1'h0

5. Start RDC
   RDC_RESETn_FSM -> 1'h1
   RDC_ISOLATE -> 1'h0

- Timing diagram / Conversion time
  Following diagram shows RDC timing diagram. Related parameters that effect timing are as follows.
  RDC_CNT_INIT
  RDC_SEL_DLY
  RDC_CNT_AZ_RESETB
  RDC_CNT_STORE / RDC_CNT_REDISRDC_CNT_REDIS
  RDC_CNT_AMP1
  RDC_CNT_AMP2
  RDC_CNT_IDLE

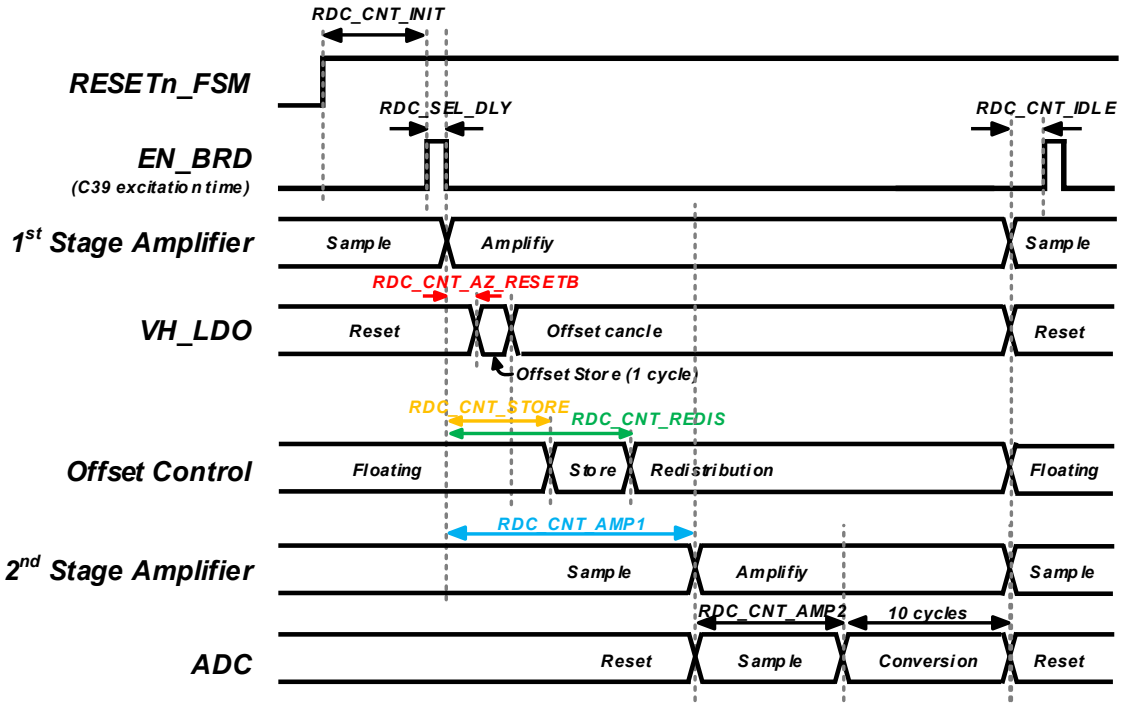


Figure IV-v RDC timing diagram

Following condition should be met for proper operation.

RDC_CNT_AZ_RESETB + 1 < RDC_CNT_STORE < RDC_CNT_REDISRDC_CNT_REDIS < RDC_CNT_AMP1

As it can be seen from the timing diagram, overall conversion time of the RDC is determined as follows (excluding initial start-up time set by **RDC_CNT_INIT**). Default conversion time is ~12.8ms.

Overall conversion time = RDC_OSR × [$T_{Excitation}$* + (RDC_CNT_AMP1 + RDC_CNT_AMP2 + 10 + RDC_CNT_IDLE) × T_{CLK_Period}**]

* Set by **RDC_SEL_DLY**. See **Figure III-i** for details

** 50µs in typical corner

E. Sensor Connection

C39 should be connected to RDC in a specific configuration shown in **Table IV-4**. Refer to **Figure II-iii** as well.

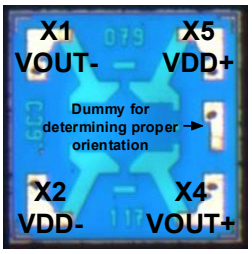


Figure IV-vi C39 pad configuration

C39	RDC PAD Name
X5	VH
X4	VP
X1	VN
X2	VSS

Table IV-4 RDC connection to C39

F. Working with Different Sensors Other than C39

Although RDC layer has been mainly designed for using C39 as a transducer, it can be also used with other sensors as long as they are based on a Wheatstone bridge. When using other sensors, followings should be taken into consideration.

- Sensor connection : Use **Figure II-iii** and **Table IV-4** as a reference.
- Bridge Resistance : Excitation time should be adjusted to have enough settling time depending on the bridge resistance. See RDC_SEL_DLY for details.
- Sensitivity : Gain (**RDC_SEL_GAIN_LC**) and offset (**RDC_OFFSET_P_LC/RDC_OFFSET_PB_LC**) should be adjusted based on the sensor sensitivity. Current default values are based on C39 sensitivity which is shown in **Table IV-5**. Refer to Section IV-D “Operation” for properly setting the gain and offset.

	Min	Typ	Max
Sensitivity (mV/V bar)	12	15	18

Table IV-5 C39 Sensitivity

G. Supply Sensitivity

Theoretically RDC should not have any supply sensitivity as it uses sampled VBAT as a supply voltage via VH_LDO. However, supply sensitivity has been seen in measurements and shows some variation from chip2chip. Following figure shows supply sensitivity of the RDC from 3.6V to 4.2V across 20 chips.

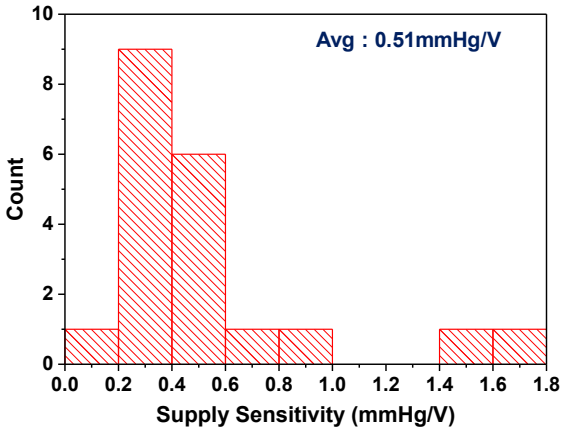


Figure IV-vii RDC Supply Sensitivity (VBAT=3.6-4.2V) across 20 chips

H. Temperature Sensitivity

C39 itself has temperature sensitivity. As this temperature dependence shows 2nd order behavior, at least 3 points are required for calibration. Temperature calibration is performed on first order coefficients (i.e. slope (a) and offset (b)) found at each temperature after 2-pt calibration. **Figure IV-viii** shows pressure error after performing 3-point calibration at 0, 40 and 75°C.

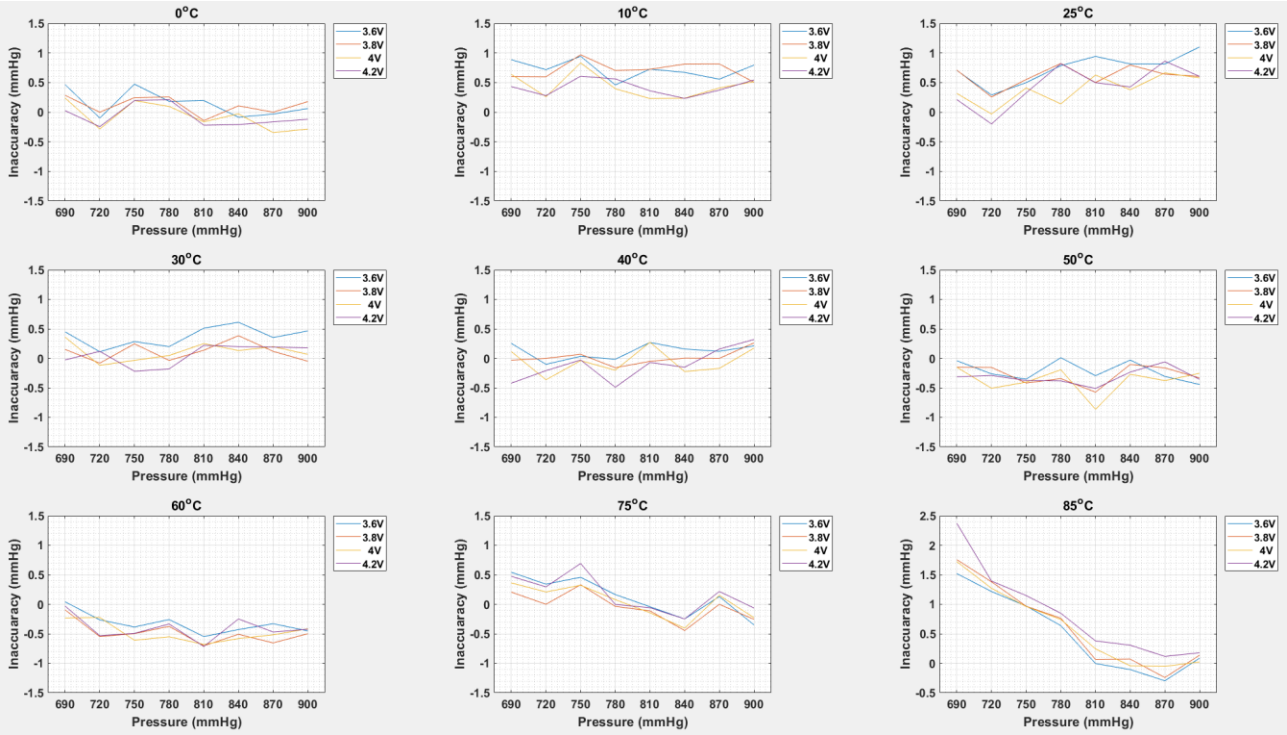


Figure IV-viii RDC Temperature Sensitivity after 3-point calibration

I. Long Term Stability

So far, it has been found that C39 + RDC has good long term stability as long as they are not covered by any epoxy (**Figure IV-ix** lower left and lower right). Both soft and ultrasoft epoxy showed drift. Upper left and right plots in **Figure IV-ix** is with ultrasoft epoxy.

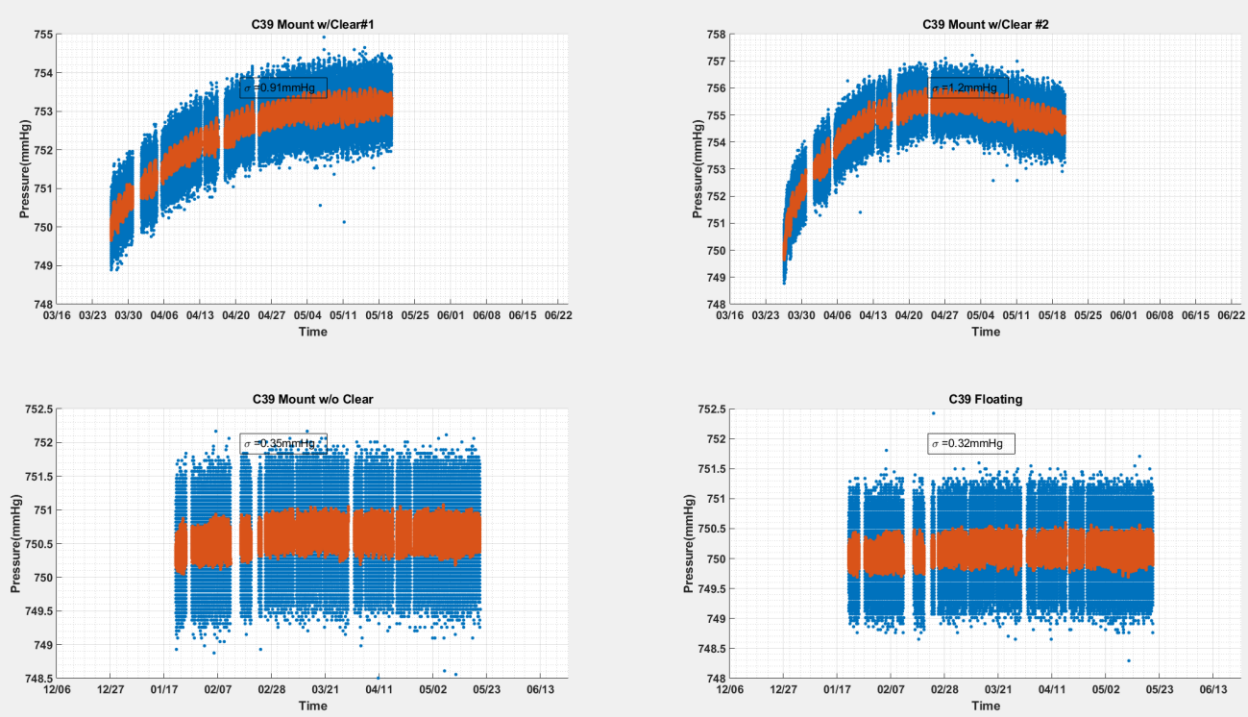


Figure IV-ix RDC long term stability with different configurations. Ignore gaps and big drops as they are due to environmental effects (testing setup issues, power outage, etc)

V. Appendix

A. Sample C Code

Following C code is for testing long term stability of RDC where RDC periodically turns on, take a measurement and goes to sleep. Measured data are grabbed by Labview through MBUS snooper. Note that MBUS loop includes 1 PRC and 4 RDCs.

```
//*****************************************************************************************
//Author:      Seokhyoen Jeong
//Description:  RDC LongTerm Test
//                                    - PRCv17 / RDCv3
//*****************************************************************************************
#include "PRCv17.h"
#include "PRCv17_RF.h"
#include "mbus.h"
#include "RDCv3_RF.h"

// uncomment this for debug mbus message
#define DEBUG_MBUS_MSG
// uncomment this for debug radio message
//#define DEBUG_RADIO_MSG

// uncomment this to only transmit average
//#define TX_AVERAGE

// Stack order  PRC->RDC1->RDC2->RDC3->RDC4
volatile uint32_t RDC_ADDR;
volatile uint32_t RDC_WRITE_ADDR;
//#define PMU_ADDR 0x6

// Parameters
#define   MBUS_DELAY 100 // Amount of delay between successive messages; 100: 6-7ms
#define TIMER32_VAL 0x50000 // 0x20000 about 1 sec with Y5 run default clock (PRCv17)



//*************************************************
// Global variables
//*************************************************
// "static" limits the variables to this file, giving compiler more freedom
// "volatile" should only be used for MMIO --> ensures memory storage
volatile uint32_t enumerated;
volatile uint32_t wakeup_data;
volatile uint32_t wfi_timeout_flag;
volatile uint32_t exec_count;
volatile uint32_t meas_count;
volatile uint32_t exec_count_irq;

volatile uint32_t WAKEUP_PERIOD_CONT_USER;
volatile uint32_t WAKEUP_PERIOD_CONT;
volatile uint32_t WAKEUP_PERIOD_CONT_INIT;
```

```
volatile prcv17_r0B_t prcv17_r0B = PRCv17_R0B_DEFAULT;

volatile uint32_t rdc_output;
volatile uint32_t n_iter;
volatile uint32_t init_fail_cnt;

volatile rdcv3_r00_t rdcv3_r00 = RDCv3_R00_DEFAULT;
volatile rdcv3_r20_t rdcv3_r20 = RDCv3_R20_DEFAULT;
volatile rdcv3_r21_t rdcv3_r21 = RDCv3_R21_DEFAULT;
volatile rdcv3_r22_t rdcv3_r22 = RDCv3_R22_DEFAULT;
volatile rdcv3_r23_t rdcv3_r23 = RDCv3_R23_DEFAULT;
volatile rdcv3_r24_t rdcv3_r24 = RDCv3_R24_DEFAULT;
volatile rdcv3_r25_t rdcv3_r25 = RDCv3_R25_DEFAULT;
volatile rdcv3_r26_t rdcv3_r26 = RDCv3_R26_DEFAULT;
volatile rdcv3_r27_t rdcv3_r27 = RDCv3_R27_DEFAULT;
volatile rdcv3_r28_t rdcv3_r28 = RDCv3_R28_DEFAULT;
volatile rdcv3_r29_t rdcv3_r29 = RDCv3_R29_DEFAULT;

volatile rdcv3_r2A_t rdcv3_r2A = RDCv3_R2A_DEFAULT;
volatile rdcv3_r2B_t rdcv3_r2B = RDCv3_R2B_DEFAULT;
volatile rdcv3_r2C_t rdcv3_r2C = RDCv3_R2C_DEFAULT;


//*****************************************************************
// INTERRUPT HANDLERS (Updated for PRCv17)
//*****************************************************************

void handler_ext_int_wakeup   (void) __attribute__ ((interrupt ("IRQ")));
void handler_ext_int_gocep    (void) __attribute__ ((interrupt ("IRQ")));
void handler_ext_int_timer32  (void) __attribute__ ((interrupt ("IRQ")));
void handler_ext_int_reg0     (void) __attribute__ ((interrupt ("IRQ")));
void handler_ext_int_reg1     (void) __attribute__ ((interrupt ("IRQ")));
void handler_ext_int_reg2     (void) __attribute__ ((interrupt ("IRQ")));
void handler_ext_int_reg3     (void) __attribute__ ((interrupt ("IRQ")));

void handler_ext_int_timer32(void) { // TIMER32
   *NVIC_ICPR = (0x1 << IRQ_TIMER32);
   *REG1 = *TIMER32_CNT;
   *REG2 = *TIMER32_STAT;
   *TIMER32_STAT = 0x0;
          wfi_timeout_flag = 1;
   }
void handler_ext_int_reg0(void) { // REG0
   *NVIC_ICPR = (0x1 << IRQ_REG0);
}
void handler_ext_int_reg1(void) { // REG1
   *NVIC_ICPR = (0x1 << IRQ_REG1);
}
void handler_ext_int_reg2(void) { // REG2
   *NVIC_ICPR = (0x1 << IRQ_REG2);
}
void handler_ext_int_reg3(void) { // REG3
```

```
    *NVIC_ICPR = (0x1 << IRQ_REG3);
}
void handler_ext_int_reg7(void) { // REG7
    *NVIC_ICPR = (0x1 << IRQ_REG7);
}
void handler_ext_int_gocep(void) { // GOCEP
    *NVIC_ICPR = (0x1 << IRQ_GOCEP);
}
void handler_ext_int_wakeup(void) { // WAKE-UP
    *NVIC_ICPR = (0x1 << IRQ_WAKEUP);
    *SREG_WAKEUP_SOURCE = 0;
}

//*************************************************
// RDCv3 Functions
//*************************************************

static void rdc_enable_vref(){
        rdcv3_r2C.RDC_ENb_PG_VREF = 1;
        mbus_remote_register_write(RDC_ADDR,0x2C,rdcv3_r2C.as_int);
}
static void rdc_disable_vref(){
        rdcv3_r2C.RDC_ENb_PG_VREF = 0;
        mbus_remote_register_write(RDC_ADDR,0x2C,rdcv3_r2C.as_int);
}

static void rdc_disable_pg_v1p2(){
        rdcv3_r2C.RDC_EN_PG_FSM = 0;
        rdcv3_r2C.RDC_EN_PG_RC_OSC = 0;
        rdcv3_r2C.RDC_EN_PG_AMP_V1P2 = 0;
        rdcv3_r2C.RDC_EN_PG_ADC_V1P2 = 0;
        rdcv3_r2C.RDC_EN_PG_BUF_VH_V1P2 = 0;
        mbus_remote_register_write(RDC_ADDR,0x2C,rdcv3_r2C.as_int);
}
static void rdc_enable_pg_v1p2(){
        rdcv3_r2C.RDC_EN_PG_FSM = 1;
        rdcv3_r2C.RDC_EN_PG_RC_OSC = 1;
        rdcv3_r2C.RDC_EN_PG_AMP_V1P2 = 1;
        rdcv3_r2C.RDC_EN_PG_ADC_V1P2 = 1;
        rdcv3_r2C.RDC_EN_PG_BUF_VH_V1P2 = 1;
        mbus_remote_register_write(RDC_ADDR,0x2C,rdcv3_r2C.as_int);
}

static void rdc_disable_pg_vbat(){
        rdcv3_r2C.RDC_ENb_PG_AMP_VBAT = 1;
        rdcv3_r2C.RDC_ENb_PG_ADC_VBAT = 1;
        rdcv3_r2C.RDC_ENb_PG_BUF_VCM = 1;
        rdcv3_r2C.RDC_ENb_PG_BUF_VH_VBAT = 1;
        rdcv3_r2C.RDC_ENb_MIRROR_LDO = 0;
        mbus_remote_register_write(RDC_ADDR,0x2C,rdcv3_r2C.as_int);
}
static void rdc_enable_pg_vbat(){
        rdcv3_r2C.RDC_ENb_PG_AMP_VBAT = 0;
```

```
        rdcv3_r2C.RDC_ENb_PG_ADC_VBAT = 0;
        rdcv3_r2C.RDC_ENb_PG_BUF_VCM = 0;
        rdcv3_r2C.RDC_ENb_PG_BUF_VH_VBAT = 0;
        rdcv3_r2C.RDC_ENb_MIRROR_LDO = 1;
        mbus_remote_register_write(RDC_ADDR,0x2C,rdcv3_r2C.as_int);
}

static void rdc_disable_clock(){
        rdcv3_r28.RDC_RESET_RC_OSC = 1;
        rdcv3_r2B.RDC_CLK_ISOLATE = 1;
        mbus_remote_register_write(RDC_ADDR,0x28,rdcv3_r28.as_int);
        mbus_remote_register_write(RDC_ADDR,0x2B,rdcv3_r2B.as_int);
}
static void rdc_enable_clock(){
        rdcv3_r28.RDC_RESET_RC_OSC = 0;
        rdcv3_r2B.RDC_CLK_ISOLATE = 0;
        mbus_remote_register_write(RDC_ADDR,0x28,rdcv3_r28.as_int);
        mbus_remote_register_write(RDC_ADDR,0x2B,rdcv3_r2B.as_int);
}

static void rdc_start_meas(){
        rdcv3_r20.RDC_RESETn_FSM = 1;
        rdcv3_r20.RDC_ISOLATE = 0;
        mbus_remote_register_write(RDC_ADDR,0x20,rdcv3_r20.as_int);
}
static void rdc_reset(){
        rdcv3_r20.RDC_RESETn_FSM = 0;
        rdcv3_r20.RDC_ISOLATE = 1;
        mbus_remote_register_write(RDC_ADDR,0x20,rdcv3_r20.as_int);
}

//************************************************
// End of Program Sleep Operation
//************************************************
static void operation_sleep(void){

        // Reset GOC_DATA_IRQ
        *GOC_DATA_IRQ = 0;

  // Go to Sleep
  mbus_sleep_all();
  while(1);

}

static void operation_sleep_noirqreset(void){

  // Go to Sleep
  mbus_sleep_all();
  while(1);

}
```

```
static void operation_sleep_notimer(void){

        // Disable Timer
        set_wakeup_timer(0, 0, 0);

  // Go to sleep
  operation_sleep();

}


static void operation_init(void){

        // Set CPU & Mbus Clock Speeds
  prcv17_r0B.CLK_GEN_RING = 0x1; // Default 0x1
  prcv17_r0B.CLK_GEN_DIV_MBC = 0x1; // Default 0x1
  prcv17_r0B.CLK_GEN_DIV_CORE = 0x3; // Default 0x3
  prcv17_r0B.GOC_CLK_GEN_SEL_DIV = 0x0; // Default 0x0
  prcv17_r0B.GOC_CLK_GEN_SEL_FREQ = 0x6; // Default 0x6
        *REG_CLKGEN_TUNE = prcv17_r0B.as_int;

  // Disable MBus Watchdog Timer
  //*REG_MBUS_WD = 0;
        *((volatile uint32_t *) 0xA000007C) = 0;


  //Enumerate & Initialize Registers
  enumerated = 0xDEADBEE0;
  exec_count = 0;
  exec_count_irq = 0;

  // Set CPU Halt Option as RX --> Use for register read e.g.
  //set_halt_until_mbus_rx();

  //Enumeration
  RDC_ADDR = 4;
  mbus_enumerate(RDC_ADDR);
        delay(MBUS_DELAY);
  RDC_ADDR = 5;
  mbus_enumerate(RDC_ADDR);
        delay(MBUS_DELAY);
  RDC_ADDR = 6;
  mbus_enumerate(RDC_ADDR);
        delay(MBUS_DELAY);
  RDC_ADDR = 7;
  mbus_enumerate(RDC_ADDR);
        delay(MBUS_DELAY);

  // Set CPU Halt Option as TX --> Use for register write e.g.
        //set_halt_until_mbus_tx();

  ////////////////////////////////////////////////////
  // RDCv3 Settings ------------------------------------
```

```
// Common settings
   // rdcv3_r00
   //rdcv3_r00.WAKEUP_UPON_RDC_IRQ = 0x0;
   //rdcv3_r00.MBC_WAKEUP_ON_PEND_REQ = 0x1;

   // rdcv3_r20
   rdcv3_r20.RDC_CNT_AMP1 = 0xC;

   // rdcv3_r21
   rdcv3_r21.RDC_CNT_AMP2 = 0x8;

   // rdcv3_r22
   rdcv3_r22.RDC_CNT_SKIP = 0x3;
   rdcv3_r22.RDC_OSR = 0xB;

   // rdcv3_r24 & r25
   rdcv3_r24.RDC_SEL_DLY = 0xA;

// Individual settings
   // 1st chip
   RDC_ADDR = 4;
   rdcv3_r24.RDC_SEL_GAIN_LC = 0x8;
   rdcv3_r25.RDC_OFFSET_P_LC = 0x13;
   rdcv3_r25.RDC_OFFSET_PB_LC = 0xC;
   mbus_remote_register_write(RDC_ADDR,0x20,rdcv3_r20.as_int);
   mbus_remote_register_write(RDC_ADDR,0x21,rdcv3_r21.as_int);
   mbus_remote_register_write(RDC_ADDR,0x22,rdcv3_r22.as_int);
   mbus_remote_register_write(RDC_ADDR,0x24,rdcv3_r24.as_int);
   mbus_remote_register_write(RDC_ADDR,0x25,rdcv3_r25.as_int);

   // 2nd chip
   RDC_ADDR = 5;
   rdcv3_r24.RDC_SEL_GAIN_LC = 0x8;
   rdcv3_r25.RDC_OFFSET_P_LC = 0x14;
   rdcv3_r25.RDC_OFFSET_PB_LC = 0xB;
   mbus_remote_register_write(RDC_ADDR,0x20,rdcv3_r20.as_int);
   mbus_remote_register_write(RDC_ADDR,0x21,rdcv3_r21.as_int);
   mbus_remote_register_write(RDC_ADDR,0x22,rdcv3_r22.as_int);
   mbus_remote_register_write(RDC_ADDR,0x24,rdcv3_r24.as_int);
   mbus_remote_register_write(RDC_ADDR,0x25,rdcv3_r25.as_int);

   // 3rd chip
   RDC_ADDR = 6;
   rdcv3_r24.RDC_SEL_GAIN_LC = 0x8;
   rdcv3_r25.RDC_OFFSET_P_LC = 0x13;
   rdcv3_r25.RDC_OFFSET_PB_LC = 0xC;
   mbus_remote_register_write(RDC_ADDR,0x20,rdcv3_r20.as_int);
   mbus_remote_register_write(RDC_ADDR,0x21,rdcv3_r21.as_int);
   mbus_remote_register_write(RDC_ADDR,0x22,rdcv3_r22.as_int);
   mbus_remote_register_write(RDC_ADDR,0x24,rdcv3_r24.as_int);
   mbus_remote_register_write(RDC_ADDR,0x25,rdcv3_r25.as_int);

   // 4th chip
```

```
    RDC_ADDR = 7;
    rdcv3_r24.RDC_SEL_GAIN_LC = 0x7;
    rdcv3_r25.RDC_OFFSET_P_LC = 0x14;
    rdcv3_r25.RDC_OFFSET_PB_LC = 0xB;
    mbus_remote_register_write(RDC_ADDR,0x20,rdcv3_r20.as_int);
    mbus_remote_register_write(RDC_ADDR,0x21,rdcv3_r21.as_int);
    mbus_remote_register_write(RDC_ADDR,0x22,rdcv3_r22.as_int);
    mbus_remote_register_write(RDC_ADDR,0x24,rdcv3_r24.as_int);
    mbus_remote_register_write(RDC_ADDR,0x25,rdcv3_r25.as_int);

  ////////////////////////////////////////////////////
  // Initialize other global variables
  WAKEUP_PERIOD_CONT = 100;   // 10: 2-4 sec with PRCv17
        wakeup_data = 0;

  delay(MBUS_DELAY);

  // Go to sleep
        //set_wakeup_timer(WAKEUP_PERIOD_CONT, 0x1, 0x1);
  //operation_sleep();
}


//*************************************************************************************
// MAIN function starts here
//*************************************************************************************
int main() {

  // Initialize Interrupts
  // Only enable register-related interrupts
        //enable_reg_irq();
        *NVIC_ISER = (1 << IRQ_WAKEUP) | (1 << IRQ_GOCEP) | (1 << IRQ_TIMER32) | (1 << IRQ_REG0)| (1 <<
IRQ_REG1)| (1 << IRQ_REG2)| (1 << IRQ_REG3| (1 << IRQ_REG7));

  // Config watchdog timer to about 10 sec; default: 0x02FFFFFF
  config_timerwd(0xFFFFF); // 0xFFFFF about 13 sec with Y2 run default clock
  //disable_timerwd();

  // Initialization sequence
  if (enumerated != 0xDEADBEEF){
    // Set up PMU/GOC register in PRC layer (every time)
    // Enumeration & RAD/SNS layer register configuration
    operation_init();

  }

        uint32_t ii;

  n_iter = 0;
  init_fail_cnt=0;
  // Proceed to continuous mode
  while(1){
    for(ii=0; ii<4; ii++){
```

```
// Address change
RDC_ADDR = ii+4;

// Initialize
rdc_enable_vref();
delay(MBUS_DELAY*10);

rdc_disable_pg_v1p2();
delay(MBUS_DELAY);

rdc_disable_pg_vbat();
delay(MBUS_DELAY);

rdc_enable_clock();
delay(MBUS_DELAY);

        // Use Timer32 as timeout counter
                wfi_timeout_flag = 0;
        config_timer32(TIMER32_VAL, 1, 0, 0); // 1/10 of MBUS watchdog timer default

// Start measure
rdc_start_meas();

// Wait for output
   WFI();

        // Turn off Timer32
        *TIMER32_GO = 0;

// Grab data
if(wfi_timeout_flag){
   // Reset everything
   rdc_reset();
   rdc_disable_clock();
   rdc_enable_pg_vbat();
   rdc_enable_pg_v1p2();
   rdc_disable_vref();
   init_fail_cnt++;
   if(init_fail_cnt==1){
      mbus_write_message32(0xAA, 0);
      delay(500000);
   }
   else{
   operation_init();
   }
   ii--;
}
else{
   n_iter++;
                        set_halt_until_mbus_rx();
                        mbus_remote_register_read(RDC_ADDR,0x11,0);
   rdc_output = *REG0;
                        set_halt_until_mbus_tx();
```

```
            mbus_write_message32(0xBB, n_iter);
            RDC_WRITE_ADDR = 0xC0 + ii;
               mbus_write_message32(RDC_WRITE_ADDR, rdc_output);
            // Reset everything
            rdc_reset();
            rdc_disable_clock();
            rdc_enable_pg_vbat();
            rdc_enable_pg_v1p2();
            rdc_disable_vref();
          }
        delay(20000);
        }
        delay(50000);
    }

    // Should not reach here
    operation_sleep_notimer();

    while(1);
}
```