



Integrated Predictive and Precision Motion Controller with SPI

FA1210C

DESCRIPTION

FA1210C is a CMOS based mixed signal integrated motion controller with SPI interface. It incorporates both armature and encoder feedbacks to offer precision control in velocity, torque and position modes for low power motors at both high and low speeds while offering features for re-configurability and environmental operation adaptation. It requires both an external amplifier and a supervisory microprocessor.

FA1210C is also open-ended to allow the use of internal ADCs (analog-to-digital converters) and digital tachometer in traditional microprocessor. In this case, the ADCs and tachometer inside FA1210C will be redundant. This could be useful in applications with SNDR (signal to noise + distortion ratio) that do not require the 12 bit ADCs inside FA1210C for enhanced power management. The chip block diagram is shown in Fig.1.

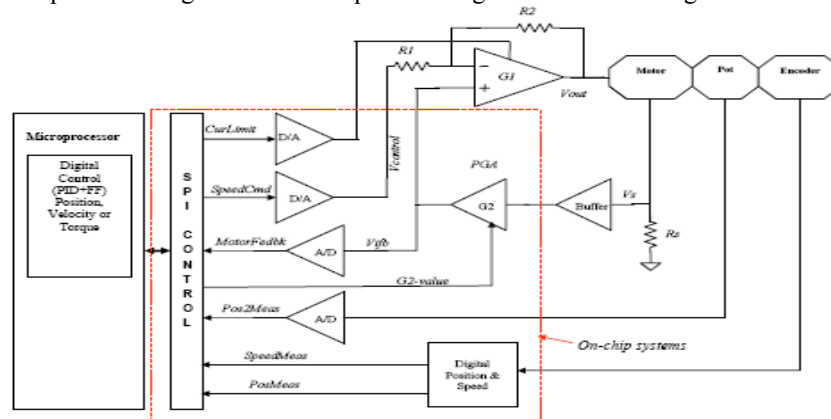


Fig.1 Motor-controller

ABSOLUTE MAXIMUM RATINGS

Parameter	Value	Unit
Supply voltage (VDD)	5	V
Input voltages	0 to 5	V
Output voltages	0 to 5	V
Supply ground (GND)	0	V

PIN CONNECTION (Top view)

The chip is packaged inside an LCC68 package (68 pins) provided by MOSIS.

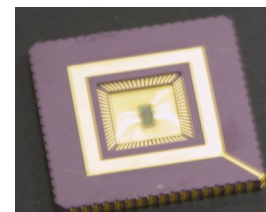
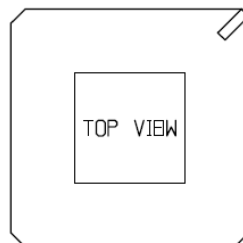
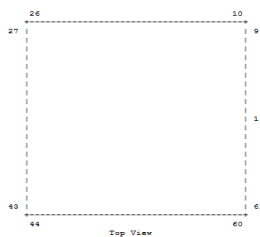


Fig 2. Pin connection

PIN FUNCTIONS

A = Analog; D = Digital; I = Input; O = Output; I/O = Input/output; NC = No connection

For positive references, recommended maximum value is 4.2V; minimum value is 0.8V as the Opamp is not rail-to-rail

All signals are voltages, unless stated otherwise.

DAC = digital to analog converter; PGA = programmable gain amplifier; ADC = analog to digital converter; SPI = serial peripheral interface; PISO = parallel in serial output; G1 = external Opamp; VFC = velocity frequency counting module; VPC = velocity period counting module; POS = period measurement module; DPS = digital position and speed (this comprises VPC, VPC and POS)

Table 1: FA1210C Chip Functions

Pin #	Name	Type	Direction	Function
1	CHA	D	I	One input or channel of the encoder
2	CLK_PER	D	I	Signal for velocity measurement via period counting. This signal is counted in VPC
3	TIMER_CLK	D	I	Signal for velocity measurement via frequency counting. This is the time window signal for VFC
4	DIR	D	O	Motor rotation direction (clockwise or anti-clockwise). If CHB leads CHA, DIR is HIGH.
5	EN_PER	D	O	Interrupt signal for the VPC. This signal (on falling edge) indicates that a counting session has completed and transfer from VPC/DPS to microprocessor is ready. See DPS for more details [for testing purposes]
6	IN_PGA	A	I	When armature current is sensed via sense resistor, the voltage is connected via this pin. This is the input of the PGA.
7	NC			
8	GAIN1_0	D	I	For gain of 1 in the PGA, make this signal LOW. Otherwise, this pin has to be HIGH-the state where other possible gains can be obtained in PGA
9	SO_FREQ	D	O	[For testing purposes] Output (serialized) for frequency counting velocity measurement
10	SO_PER	D	O	[For testing purposes] Output (serialized) for period counting velocity measurement
11	NC			
12	NC			
13	NEG_CMR	A	I	Minimum voltage reference for the PGA. This is the negative common mode voltage (CMR) which must be set to GND on normal chip operation. Making it GND allows the use of the PGA equation (8) given below instead of (6).
14	OUT_DAC2	A	O	Output of the DAC that goes to the inverting input of the external Opamp
15	CS_PGA	D	I	Chip select for the SPI control of PGA. When Low, PGA data is being transferred from a microprocessor via a register to PGA. When it goes High, the transferred data is opaquely latched; the Latch retains the data even when chip select goes low. See PGA for more details
16	CARRYOUT1	D	O	Interrupt signal for the ADC quantizing the output result of the PGA. This signal indicates that a sampling session has completed and transfer from ADC to microprocessor is ready. Detection is rising edge. See ADC for more

				details
17	NC			
18	CLK_POS	D	I	This is the clock that controls the quadrature decoder for the position measurement. The quadrature decoder gives a pulse at every transition (both rising and falling edges) for CHA and CHB; there is a 4X factor change here. This clock has to be at least twice faster than the speed of the encoder signals. See DPS for details
19	OUT_DAC1	A	O	Output of the DAC for the current limit setting of the external Opamp. This is a voltage.
20	EN_FREQ	D	O	Interrupt signal for the VFC. This signal (on falling edge) indicates that a counting session has completed and transfer from VFC/DPS to microprocessor is ready. See DPS for more details [for testing purposes]
21	VREFM_DAC1	A	I	Min reference voltage for DAC for current limit setting
22	VREFP_DAC1	A	I	Max reference voltage for DAC for current limit setting
23	VREFM_DAC2	A	I	Min reference voltage for DAC for inverting input of external Opamp
24	SO_ADC1	D	O	[For testing purposes] Output (serialized) for ADC of PGA output
25	VREFP_DAC2	A	I	Max reference voltage for DAC for inverting input of external Opamp
26	LATCH_SX1	D	I	External latch signal for internal data transfer for the ADC quantizing PGA output. This is activated by interrupt implemented in a supervisory microprocessor. This signal is supplied externally. See ADC for more details
27	RESET_SX1	D	I	External reset signal for internal data transfer for the ADC quantizing PGA output. This is activated by interrupt implemented in a supervisory microprocessor. This signal is supplied externally
28	VDD (SUPPLY)	A	I/O	Power Supply (Max 5V)
29	MOD_OUT	D	O	[For testing purpose only] (output of ADC modulator)
30	MOD_OUT1	D	O	[For testing purpose only] (output of ADC modulator)
31	SPI_CLK	D	I	The SPI clock. This has to be faster than the quantizing rate (ADC_EXT_CLK) of the ADC. This ensures that transfer is completed before next sample is ready. This is used for bi-directional data transfer between chip and microprocessor. See SPI protocol for details
32	ADC_EXT_CLK	D	I	Clock for the two ADCs. This clock must have the frequency giving by 4096 Over sampling ratio (OSR). For instance if the input signal of the ADC is x, the Nyquist frequency is 2x. This clock will be 8196x (4096*2x). Any value below this will degrade the full resolution of the ADC. See ADC for details
33	SI	D	I	This is used for PISO register. Set this Low for normal operation. You could tie it GND on the PCB board
34	VREFP	A	I	Max reference voltage for all ADC
35	OUT_PGA_BUF	A	O	Output of PGA to non inverting input of G1 amplifier
36	VREFM	A	I	Min reference voltage for all ADC
37	NC			
38	VIP_POT	A	I	Connect potentiometer voltage here
39	NC			
40	NEG_CMV_ADC	A	I	Reference voltage for all ADCs. This should be the offset in the board. For instance, in our 5V VDD, tie

				this pin to 2.5V.
41	CS_ADC1	D	I	Chip select for the SPI control of ADC quantizing PGA output. When Low, PGA data is loaded. When High, data is transferred from ADC to microprocessor. See ADC for details
42	GND			Ground supply (0V). Same as pin 62
43	SO_ADC2	D	O	[For testing purposes] Serial output of ADC of potentiometer.
44	CS_ADC2			Chip select for the SPI control of ADC quantizing potentiometer output. When Low, PGA data is loaded. When High, data is transferred from ADC to microprocessor. See ADC for details
45	NC			
46	NC			
47	ACH_GLO_RESET	D	I	To globally reset the chip, set this High. For normal operation, Low. Make this signal to go High momentarily on Power-Up and settle to Low state. This ensures that the chip is ready for normal use.
48	RESET_SX	D	I	External reset signal for internal data transfer for the ADC of potentiometer. This is activated by interrupt implemented in a supervisory microprocessor. Used only when SX1_ONCHIP0 (pin 50) is set High for external interrupt; otherwise, NC. See ADC for details
49	LATCH_SX	D	I	External latch signal for internal data transfer for the ADC of potentiometer. This is activated by interrupt implemented in a supervisory microprocessor. Used only when SX1_ONCHIP0 (pin 50) is set High for external interrupt; otherwise, NC. See ADC for details
50	SX1_ONCHIP0	D	I	For external interrupt mode via the supervisory microprocessor, make this High. To allow an internal on-chip interrupt sequence, make Low. Interrupt is used to transfer ADC data to external microprocessor after quantization has completed. We recommend High for this signal as the chip complexity makes the internal interrupt non-synchronized. In short, set this to High for normal operation. It works with both PINS 57 and 16.
51	NC			
52	NC			
53	CS_DAC2	D	I	Chip select for the SPI control of DAC for the inverting input of external Opamp. When Low, DAC data is being transferred from a microprocessor via a register to DAC. When it goes High, the transferred data is opaquely latched; the Latch retains the data even when chip select goes low. See DAC for more details
54	MOSI	D	I	Master Out Slave Input pin for the SPI. Connect this to the microprocessor for the provision of bits to be transferred to DAC1, DAC2 and PGA based on the sequences controlled by the chip selects.
55	SX1_ONCHIP0_VEL	D	I	For external interrupt mode via the supervisory microprocessor, make this High. This is similar to pin 50 but used in the velocity module for VPC and VFC (POS I is not affected by this signal). To allow the on-chip interrupt sequence, make Low. Interrupt is used to latch the data after a measurement cycle. We recommend High for this operation. Works with pins 5 and 20

56	RESET_FREQ_SX	D	I	External reset signal for internal data transfer for the VFC. This is activated by interrupt implemented in a supervisory microprocessor. Used only when SX1_ONCHIP0_VEL (pin 55) is set High for external interrupt; otherwise, NC. See DPS for details
57	CARRYOUT	D	O	Interrupt signal for the ADC quantizing the output result of the potentiometer. This signal indicates that a sampling session has completed and transfer from ADC to microprocessor is ready. Detection is rising edge. See ADC for more details
58	SO_POS	D	O	[For testing purposes] Output (serialized) of position measurement
59	LATCH_FREQ_SX	D	I	External latch signal for internal data transfer for the VFC. This is activated by interrupt implemented in a supervisory microprocessor. Used only when SX1_ONCHIP0_VEL (pin 55) is set High for external interrupt; otherwise, NC. See DPS for details
60	MISO	D	O	Master In Slave Output pin for the SPI. Connect this to the microprocessor for bits to be transferred to the microprocessor from DPS module and the two ADCs. Based on the chip selects, uses PINS 58, 43, 24, 9, 10. These are the SO_FREQ, SO_PER, SO_POS, SO_ADC1 and SO_AD2 which are selectively transferred from the chip thru MISO to the microprocessor
61	LATCH_PER_SX	D	I	External latch signal for internal data transfer for the VPC. This is activated by interrupt implemented in a supervisory microprocessor. Used only when SX1_ONCHIP0_VEL (pin 55) is set High for external interrupt; otherwise, NC. See DPS for details
62	GND	A	I/O	Ground signal (0V). Same as pin 42
63	RESET_PER_SX	D	I	External reset signal for internal data transfer for the VPC. This is activated by interrupt implemented in a supervisory microprocessor. Used only when SX1_ONCHIP0_VEL (pin 55) is set High for external interrupt; otherwise, NC. See DPS for details
64	CS_POS	D	I	Chip select for the SPI control of POS. When Low, POS data is loaded. When High, data is transferred from POS to microprocessor. See DPS for details
65	CS_DAC1	D	I	Chip select for the SPI control of DAC for current limit setting. When Low, DAC data is being transferred from a microprocessor via a register to DAC. When it goes High, the transferred data is opaquely latched; the Latch retains the data even when chip select goes low. See DAC for more details
66	CS_PER	D	I	Chip select for the SPI control of VPC. When Low, VPC data is loaded. When High, data is transferred from VPC to microprocessor. See DPS for details
67	CHB	D	I	One input of the incremental encoder
68	CS_FREQ	D	I	Chip select for the SPI control of VFC. When Low, VFC data is loaded. When High, data is transferred from VFC to microprocessor. See DPS for details

NB: Except the negative (or minimum) reference of the PGA (pin 13), which has to be tied to GND, other negative references can be tied together in a PCB. VREFM = VREFM_DAC2 = VREFM_DAC1

All positive (maximum) references could be tied to the same value. Hence, $V_{REFP} = V_{REFP_DAC2} = V_{REFP_DAC1}$

FLEXIBILITY WITH PIC MICROPROCESSOR

To conserve power in this chip, it could be used along with ADCs available in PIC microprocessor. For the velocity module, the in-built encoder based tachometer could be used. This will make the ADCs and the DPS module in FA1210C redundant. A typical example is when a low resolution ADC (e.g. 10 bits in PIC) is needed over the 12 bits in FA1210C.

ARCHITECTURE DESCRIPTION OF FA1210C

The architecture combines both on-chip and off-chip circuitries with a supervisory microprocessor control via a digital interface. G_1 is the gain of the power op amp that is used to drive the motor. Because the chip will be limited to 5V signals, G_1 will scale the chip output to the maximum motor voltage. $V_{control}$ is the motor speed control voltage which is connected to the inverting input of the power op amp with gain, $G_1 = -R_2/R_1$. The block, *Digital Position & Speed* (DPS), estimates the motor speed and position by using the output of the incremental encoder. In addition, the *CurLimit* signal specifies the motor current limit for protection while the *Pos2Meas* provides a position reference from a potentiometer. ADC and DAC are analog-to-digital and digital-to-analog converters respectively.

To implement the motor speed control, the basic idea is that when V_{out} is constant, the back emf of the motor, V_e should remain constant for all motor currents I_m .

$$\frac{dV_e}{dI_m} = 0 \quad (1)$$

Using a simple motor model and applying Kirchoff's voltage law on the circuit of Fig. 1 results in the following equations:

$$V_{out} = I_m R_m + V_e + I_m R_s \quad (2)$$

$$\frac{V_{out} - G_2 I_m R_s}{R_2} + \frac{V_{control} - G_2 I_m R_s}{R_1} = 0 \quad (3)$$

where R_m and I_m are motor resistance and current respectively. The resistor values R_1 and R_2 are determined from the desired power amplifier gain, which depends on the range of $V_{control}$ and the rated motor voltage. The value of the sense resistor, R_s , should be selected as a compromise between minimizing the voltage drop and producing a signal with a good signal-to-noise ratio (about 10% of R_m should be appropriate). An off-chip constant gain amplifier is used to buffer V_s providing an option to make R_s smaller.

Equations 2 and 3 are differentiated with respect to I_m , and combining the results gives:

$$\frac{dV_e}{dI_m} = G_2 R_s + \frac{R_2 G_2 R_s}{R_1} - R_m - R_s = 0 \quad (4)$$

Solving for G_2 yields (5) which must be satisfied by the gain control loop (**for single amplifier mode configuration**).

$$G_2 = \frac{R_m / R_s + 1}{R_2 / R_1 + 1} \quad (5a)$$

Similarly for **bridge mode configuration** (the required gain is given by (5b):

$$G_2 = \frac{R_m + R_s}{2R_s G_1} = \frac{R_m / R_s + 1}{2G_1} \quad (5b)$$

This is done through G_2 , a programmable gain amplifier (PGA), to the non inverting input of the external power op amp with gain G_1 . This is the feedback voltage that ensures that V_{out} is maintained at a level for constant speed of the motor. The design of the PGA is made up of a network of resistors that would be tapped to set G_2 . These resistors are selected based on the required gain using the microprocessor.

SPI Protocol in this Chip

SPI is a standard protocol for serial transfer of data between master and slave(s). The master is the supervisory microprocessor while the slave is the chip. However, within the chip there are 8 slaves-two DACs, two ADCs, one PGA, velocity-frequency counting (FREQ), velocity-period counting (PER) and position counting (POS). **NB: The chip select is Active High (unlike the standard SPI which is Active Low).**

Table 2: SPI protocol

Symbol	Name	Function
MOSI	Master-out-serial-in	Bits to be transfereed to DAC and PGA (from microprocessor) passes through this pin
MISO	Master-in-serial-output	Bits to be transfereed to microprocessor (from FA1210C) passes through this pin. ADCs, VFC, VPC and POS are affected by this.
CS_DAC1	Chip select of DAC	When High, DAC latches data on MOSI and that value becomes the digital code for the current limit DAC
CS_DAC2	Chip select of DAC	When High, DAC latches data on MOSI and that value becomes the digital code for the DAC to inverting input of G1
CS_PGA	Chip select of PGA	When High, PGA latches data on MOSI and that value becomes the digital code for PGA
CS_FREQ	Chip select for VFC	When High, FREQ data is transferred to microprocessor via MISO
CS_POS	Chip select for POS	When High, POS data is transferred to microprocessor via MISO
CS_PER	Chip select for VPC	When High, PER data is transferred to microprocessor via MISO
CS_ADC1	Chip select for ADC	When High, ADC data is transferred to microprocessor via MISO
CS_ADC2	Chip select for ADC	When High, ADC data is transferred to microprocessor via MISO
SPI_CLK	SPI clock	Must be at least twice fatsre than ADC_EXT_CLK

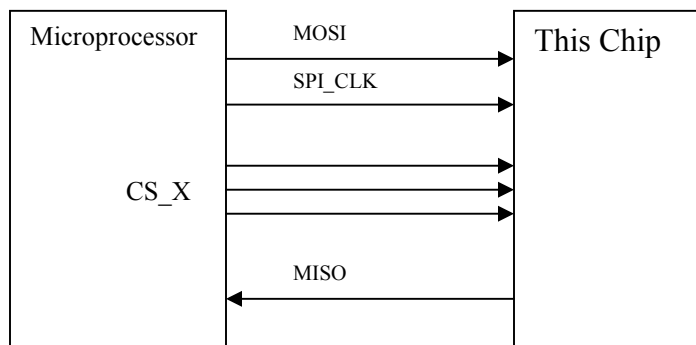


Fig.3 SPI protocol

Programmable Gain Amplifier (PGA)

This main function of the PGA (4 bits for 16 voltage levels) is to give the gain, G_2 which can be digitally programmed to satisfy different operating conditions. You program PGA to satisfy G_2 .

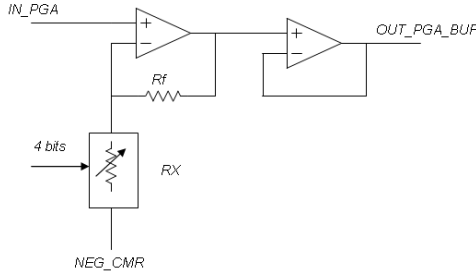


Fig.4 PGA schematic

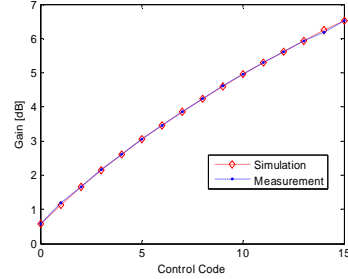


Fig.5 Measured data

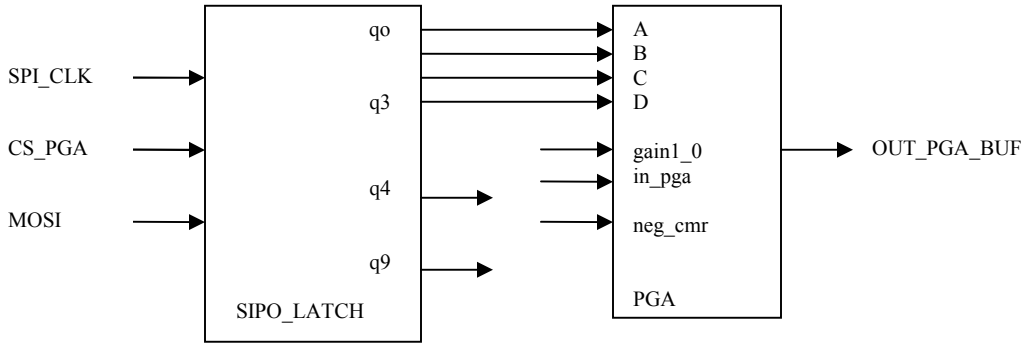


Fig.6 PGA with its SPI interface

The serial input enters the PGA via MOSI (generated by the microprocessor) into a Serial-In-Parallel-Out (SIPO) register. In Fig.6, q9 is shifted first while q0 comes last; that implies that only the **last 4 bits are used (conversely, the first 6 bits are wasted)**. After the shifting which basically converts the serial data from the microprocessor into parallel data courtesy of the SIPO, CS_PGA is raised HIGH at the 10th cycle to latch (opaquely) the data. This becomes the digital input of the PGA. NB: The LSB is D while the MSB is A. [The DAC has 10 cycles and leaving PGA in 10 cycles helps in programming and control]

The PGA has 4 bits and accordingly can be programmed to 16 voltage levels. The gain equation that represents the circuit (see Fig. 4) of the PGA is given by

$$Out_pga_buf = \left(1 + \frac{R_f}{R_x}\right)(in_pga) - \frac{R_f}{R_x}(neg_cmr) \quad (6)$$

If you make $neg_cmr = 0V$, the equation above reduces to

$$Out_pga_buf = \left(1 + \frac{R_f}{R_x}\right)(in_pga) \quad (7)$$

Based on this equation, it is easy to obtain the gain that will satisfy (5). This gain equation is given by

$$\frac{Out_pga_buf}{in_pga} = \left(1 + \frac{R_f}{R_x}\right) = G_2 \quad (8)$$

where R_x is the resistance arising from the execution of different digital codes on a resistor network. Table below gives the selected R_x for different combination of digital codes (A, B, C and D). To obtain a gain of 1, i.e., make $G2=1$, all that is required is to make the signal $GAIN1_0$ to go LOW. For other states in Table 1; $GAIN1_0$ must be HIGH.

Table 3: Digital Codes and Resistances for PGA

ABCD	Rf [k]	Rx [k]
0000	20	285.71
0001	20	142.86
0010	20	95.24
0011	20	71.43
0100	20	57.14
0101	20	47.62
0110	20	40.82
0111	20	35.71
1000	20	31.75
1001	20	28.57
1010	20	25.97
1011	20	23.81
1100	20	21.98
1101	20	20.41
1110	20	19.05
1111	20	17.86
GAIN1_0	20	infinity

Digital to Analog Converter (DAC)

A 10-bit segmented R-2R ladder DAC with thermometer coding (for the 4 MSB) for better matching and higher precision was implemented.

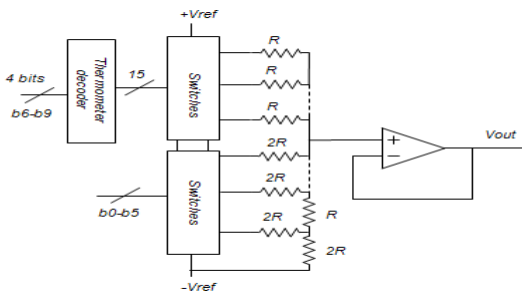


Fig. 7. Segmented R-2R DAC

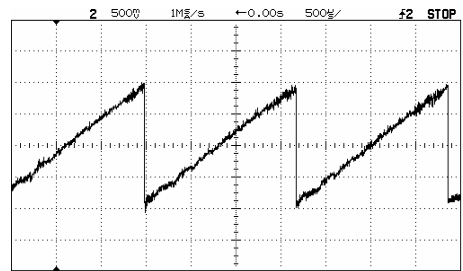


Fig.8 Measured Linearity (@ Vrefm=1.5V; Vrefp=3.5V)

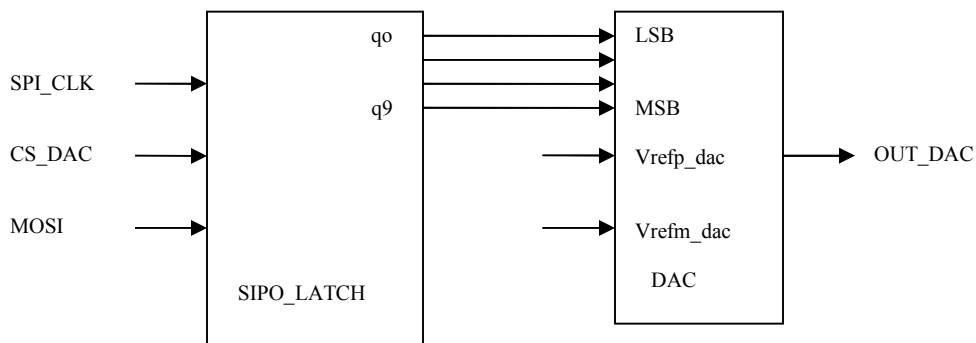


Fig.9 DAC with its SPI interface

The serial input enters the DAC via MOSI (generated by the microprocessor) as explained for PGA. In Fig.9, q9 is shifted first while q0 comes last. **[q9 is the MSB while q0 is the LSB; so MSB goes in first]**. By raising CS_DAC High, the data is latched after the 10th clock cycle. A SIPO (serial-in-parallel-output) register transfers the data on each clock cycle before CS_DAC issues the latch state. VREFP_DAC and VREFM_DAC are respectively the positive and negative references of the DAC (i.e., the highest and the lowest references).

In this DAC, there are 1024 states for the 10 bits; this means that we have 1024 states between the positive and negative references. To convert a digital code to analog code, use this relationship.

$$OUT_DAC = D_k \frac{(Vrefp - Vrefm)}{2^{10}} + Vrefm \quad (9)$$

where D_k is the digital word that is to be converted to analog. See some examples for this:

Table 3: DAC conversion

Vrefp	Vrefm	Reference Difference	Levels	Digital Word	Out
4	1	3	1024	0	1
4	1	3	1024	1	1.00293
4	1	3	1024	510	2.494141
4	1	3	1024	511	2.49707
4	1	3	1024	512	2.5
4	1	3	1024	1022	3.994141
4	1	3	1024	1023	3.99707

From this table, 1023 digital word (in binary, 111111111) gives an analog value of 3.99707V. We recommend making the Vrefm = 0.8 and Vrefp = 4.2 as the Opamp is not rail to rail.

Analog to Digital Converter (ADC)

The ADC is a 12-bit delta sigma ADC designed with a novel decimation filtering. The block diagram is shown in Fig.3. Analog signal is converted into a pulse density modulated signal using a modulator. The output of the modulator is fed into a counter operating at 4096 OSR (over-sampling ratio) and supervised by a similar counter that generates a latch signal after every counting cycle. The data is latched and sent out to the microprocessor.

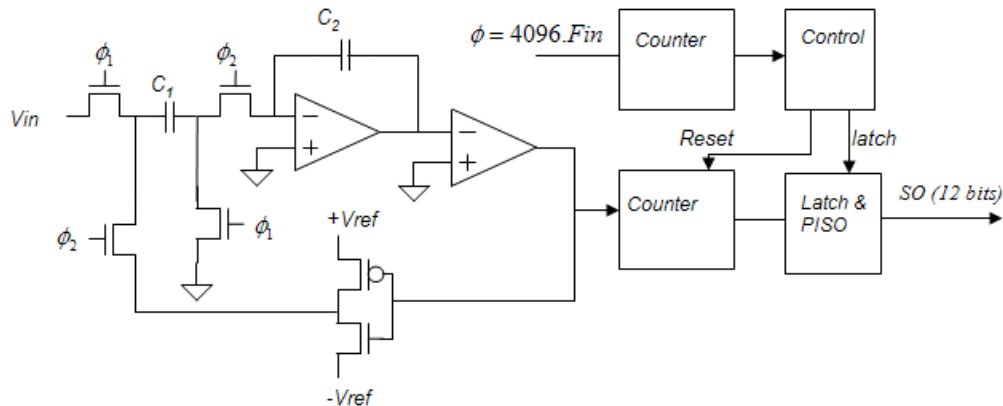


Fig.10. ADC architecture

The two ADCs are similar except for the pin changes; description of one applies to the other. The ADC has positive and negative voltage references (VREFP and VREFM). This would limit the maximum and minimum input voltages. To determine the clock frequency of the ADC (i.e.. ADC_EXT_CLK), we use this relationship.

Resolution	12
OSR	4096
Let input signal frequency	10Hz
Nyquist freq	20Hz
ADC_EXT_CLK freq	20*4096 = 81.920 KHz

So, the clock frequency must be 4096 times the Nyquist of the input signal. This is important for 12 bit resolution; otherwise the resolution could be lower. The way this ADC works is that a sample is completed and transferred when a supervisory counter completes a cycle. The number of pulses of the PDM (pulse density modulated) signal from the modulator counted is latched and then sent outside the chip via PISO (parallel-in-serial-output) register. The counters are reset and the cycle starts all over again.

The NEG_CM_R_ADC should be the same as VREFPM. It is the negative CM_R (common mode voltage of the modulator). Tie NEG_CM_R_ADC to the motor offset value; e.g. 2.5V for a 5V chip like ours.

For the transfer of data from the ADC to the external microprocessor, we have the SPI interface. There are two modes of operation depending if the signal SX1_ONCHIP0 is High or Low. When this signal is LOW, data transfer is executed by signals generated internally. When it is HIGH, we need the use of **Interrupt** along with the external microprocessor to get the data out. **We recommend operation using SX1_ONCHIP0 at HIGH.** This enables us to manage through the supervisory microprocessor all the signals (chip select, MISO, MOSI, etc) better.

To activate an interrupt in the external microprocessor, we use the CARRYOUT signal which confirms that a sampling cycle had been completed. When CARRYOUT goes High, Interrupt is executed and two signals should be sent to the chip to execute a Latch and then reset. The interrupt must be set to rising edge detection on the CARRYOUT pin. The signals are LATCH_SX (should go High to latch the data from counter to a Latch) and RESET_SX (should also go High to reset the two counters, preparing for a new sampling cycle). The LATCH_SX must come first before the RESET_SX.

The Latch in this chip is opaque; it retains the state even when the control signal, LATCH_SX has gone low. This makes it possible that transfer from Latch to PISO and then external microprocessor can follow RESET_SX during programming of the FA1210C. As soon as you latch, you can execute reset; that does not affect the result sent to external world since the reset does not change the value on the Latch. [A code is provided later].

To move the data from LATCH to PISO (this is a parallel to serial transfer), you use the signal CS_ADC (or CS_ADC2, depending on the ADC). This signal is basically a LOAD at Low and SHIFT at High; this means, it loads the parallel data from the Latch when the signal is LOW and then transfer them after each clock cycle when the signal goes High. To ensure data is loaded, you must make sure that when CS_ADC is LOW for loading, two SPI_CLK cycles occur before you raise CS_ADC High. Then execute 12 SPI_CLK cycles to complete the transfer. The data is sent out via the MISO pin. The chip selects among others insure that only one module (ADC, velocity modules) is engaging this MISO.

Code 1: Example below is an actual code (portion) used in testing this ADC using SX_48 chip from Parallax in assembler code.

Interrupt ; this shows where an interrupt begins

```
MODE $9 ; allow checking wake-up pending register;
MOV !RB, #00000000 ; read pending and clear interrupts
```

```
; Latch when CARRYOUT is High; the interrupt mode is rising edge detection on CARRYOUT
```

```

SETB LATCH_SX; make the latch signal High
SETB LATCH_SX
CLR B LATCH_SX; bring it low; this completes a latch

; reset here
SETB RESET_SX; make the reset signal low
SETB SD0
CLR B SD0; take it low and completes reset
;=====
; this must happen fast before the next cycle of CARRYOUT
CLR B CS_ADC ; load the data from latch into PISO
CLOCK_SPI; give this two SPI_CLK clock cycles to ensure load was done
CLOCK_SPI;

SETB CS_ADC ; this initiates shifting the loaded data for 12 bits.
CLOCK_SPI;0
CLOCK_SPI;1
CLOCK_SPI;2
CLOCK_SPI;3
CLOCK_SPI;4
CLOCK_SPI;5
CLOCK_SPI;6
CLOCK_SPI;7
CLOCK_SPI;8
CLOCK_SPI;9
CLOCK_SPI;10
CLOCK_SPI;11

CLR B CS_ADC; completes the 12 cycles and take it low for next one.
Reti ; return and start watching CARRYOUT for the flag.
=====

```

Digital Position and Speed (DPS) module

The function of the DPS module is to provide position and speed estimates to the supervisory microprocessor. Within DPS, we have VFC (velocity frequency counting), VPC (velocity period counting) and position measurement (POS). Two major techniques of obtaining speed estimates from an incremental encoder are period counting and frequency counting:

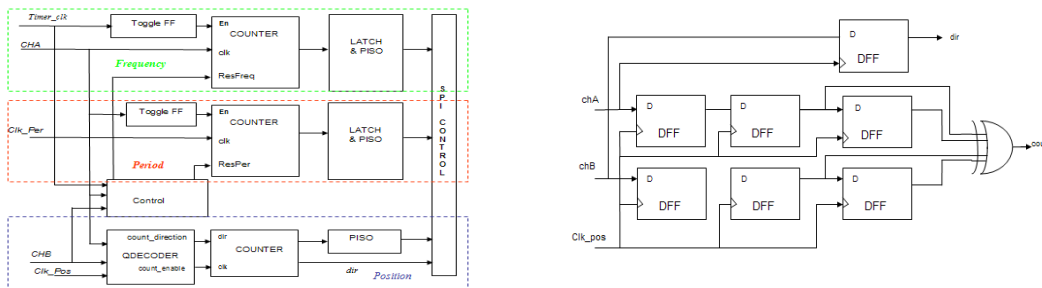


Fig.11. (a) Digital Position & Speed Module (b) QDECODER

a). Frequency counting: This involves counting the number of pulses from the encoder in a known time, T_f . High count value means high speed while small count number indicates low speed. If N_f is the final counter value, then the speed of the encoder is given by (10) where m is the number of encoder slots. Typically, incremental encoders

are made up of circular glass disc imprinted with m slots, which are equally distributed. The relative error of the system is given in (11).

$$\omega_f = \frac{2\pi N_f}{m T_f} [\text{rad / sec}] = \frac{60 N_f}{m T_f} [\text{rpm}] \quad (10)$$

$$\left| \frac{\Delta\omega_f}{\omega} \right| = \frac{60 \Delta N_f}{m \omega T_f} [\text{rpm}] \quad (11)$$

The green section of Fig.11 shows the frequency counting speed implementation. Here, a fixed time pulse (*Timer_clk*) is applied to a toggle flip flop which is connected to an enable input (*En*) of a counter (16-bits). This ensures that counting is only possible when the *En* is active. Channel A, (*CHA or pin 1*), of the encoder is applied to the clock input. The counter counts the number of pulses of the *CHA* within the window when *En* is active. The final value of the counter is latched and through PISO (parallel-in-serial-output) register is transferred to the supervisory microprocessor.

There are two modes of operation for operating the VFC (velocity frequency counting module). When *SX1_onchip0_vel* is set HIGH, external signals (*LATCH_FREQ_SX* and *RESET_FREQ_SX*) are used; when set LOW, these signals are not needed as the chip handles all aspects of the latch and reset internally. Nevertheless, depending on application, the use of external latch/reset could be better in managing the complexity of the different modules of the chip. We recommend that this chip be operated with *SX1_onchip0_vel* set HIGH; this offers better chip control.

NB: The signal, *CHA*, counted in this module is not 4x decoded. The number of pulses of *CHA* is counted within *Timer_clk* window. At the end of a counting sequence- when *EN_FREQ* is about going LOW (falling edge detection interrupt), data from the counter is transferred from the counter to a Latch as soon as a latch signal goes High (the user could decide to use the internal latch signal by setting *SX1_onchip0_vel* LOW or by applying an external latch (*LATCH_FREQ_SX*) by setting *SX1_onchip0_vel* HIGH). This is followed by a reset signal (again, the user could decide to use the internal reset signal by setting *SX1_onchip0_vel* LOW or by applying an external reset (*RESET_FREQ_SX*) by setting *SX1_onchip0_vel* HIGH). This reset clears the counter and prepares it for another cycle. The Latch used here is similar to the one in ADC- an opaque latch.

To move the data from LATCH to PISO (this is a parallel to serial transfer), you use the signal *CS_FREQ*. This signal is basically a LOAD at Low and SHIFT at High; this means, it loads the parallel data from the Latch when the signal is LOW and then transfer them after each clock cycle when the signal goes HIGH. To ensure data is loaded, you must make sure that when *CS_FREQ* is LOW for loading, two *SPI_CLK* cycles occur before you raise *CS_FREQ* High. Then execute 16 *SPI_CLK* cycles to complete the transfer. The data is sent out via the *MISO* pin. The chip selects among others insure that only one module (ADC, velocity modules) is engaging this *MISO*. See Code 1 for ideas.

NB: In internal generated latch and reset scenarios, since latch is executed first and then reset follows (and the reset does not reset the Latch, rather the counter). So, we are fine to have the transfer to PISO after the *EN_FREQ* starts the *interrupt at the falling edge*.

Code 2: Example code for data transfer when an internally generated latch and reset are used; here SX1_onchip0_vel is LOW. Compare with Code 1.

```

Interrupt ; this is not a command
    MODE $9 ; allow checking wake-up pending register
    ;break ; break to update screen during DEBUG (use polling for best)
    MOV !RB, #00000000 ; read pending and clear interrupts
;=====
;    There is need for external latch and reset as they are generated in on-chip

```

```

;=====
CLRBS CS_FREQ
CLOCK_SPI;0
CLOCK_SPI;1
; these 2 cycles ensure that you have indeed loaded

SETBS CS_FREQ

CLOCK_SPI;0
CLOCK_SPI;1
CLOCK_SPI;2
CLOCK_SPI;3
CLOCK_SPI;4
CLOCK_SPI;5
CLOCK_SPI;6
CLOCK_SPI;7
CLOCK_SPI;8
CLOCK_SPI;9
CLOCK_SPI;10
CLOCK_SPI;11
CLOCK_SPI;12
CLOCK_SPI;13
CLOCK_SPI;14
CLOCK_SPI;15

```

CLRBS PGMA ; CS_FREQ

b). Period counting: This involves counting the number of pulses from a clock between successive pulses of the encoder signal. If the clock frequency and counter final values are respectively f_p and N_p , then the speed is given by (12):

$$\omega_p = \frac{2\pi f_p}{m N_p} [\text{rad / sec}] = \frac{60 f_p}{m N_p} [\text{rpm}] \quad (12)$$

The above equation shows that at low speed (large N_p), the technique has a high accuracy while at high speed (small N_p); it suffers accuracy problems as the quantization error of N_p becomes more significant. The relative error arising from the quantization error of the system is given by [13]:

$$\left| \frac{\Delta\omega_p}{\omega} \right| = \frac{m\omega}{2\pi f_p} \Delta N_p \quad (13)$$

The implementation for the period counting speed measurement technique is similar to the frequency method except that the input to the toggle flip flop is channel A (CHA) and a pulse (Clk_Per) is applied to the clock input of a 16-bits counter (red section in Fig.11(a)). The counter counts the number of Clk_Per pulses that occur between successive encoder pulses. Everything that applies to VFC applies here except that LATCH_PER_SX, RESET_PER_SX, CS_PER are used in this case. MISO is still the output of the PISO.

c). *Position Measurement*: Position measurement involves counting the encoder pulses to estimate the position of the encoder (blue section of Fig.11 (a)). The encoder signals are Quadrature signals- two signals generated with a 90 degrees phase difference. The QDECODER circuit is sometimes called a "4x decoder" because it counts all the transitions of the quadrature inputs. A 16-bit UP counter is used to count the 4x decoded output. QDECODER also provides the direction of motor rotation (pin 4). NB: the counter is an UP counter and 4x decoded signal is actually

counted in this implementation. **This does not need any interrupt for operation.** So the output via MISO has to be acquired as much as needed. This is done via CS_POS. This module also gives the motor direction. See code 3 for code ideas.

Code 3: Example code for the acquisition of position data; in this case, no interrupt is needed. The data is acquired as many times as needed via the CS_POS signal

PosData

CLRB CS_POS ; this loads the counter data to PISO (no latch here)

CLOCK_SPI;0

CLOCK_SPI;1

; these 2 cycles ensure that you have indeed loaded

SETB CS_POS ; this starts the data transfer via MISO for position

; transfer 16 cycles

CLOCK_SPI;0

CLOCK_SPI;1

CLOCK_SPI;2

CLOCK_SPI;3

CLOCK_SPI;4

CLOCK_SPI;5

CLOCK_SPI;6

CLOCK_SPI;7

CLOCK_SPI;8

CLOCK_SPI;9

CLOCK_SPI;10

CLOCK_SPI;11

CLOCK_SPI;12

CLOCK_SPI;13

CLOCK_SPI;14

CLOCK_SPI;15

CLRB CS_POS; end transfer

jmp PosData ; go back and acquire another data



First Atlantic Semiconductors & Microelectronics Ltd

(RC908703)

Design, Advisory, Research, Training, Contracting
124a Okigwe Rd, Opp Water Board, Owerri, Imo State, Nigeria

Phone: +234 (0) 83-823195

Email: info@fasmicro.com Web: <http://fasmicro.com>
