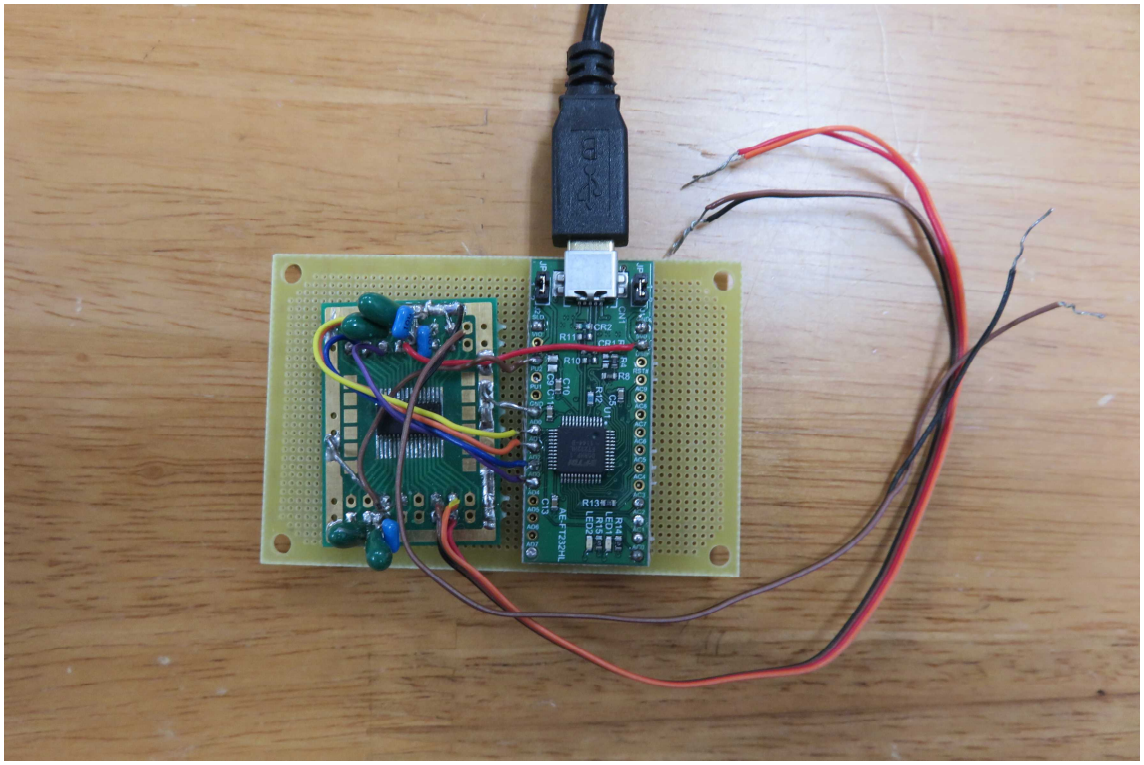
AD コンバーターの製作

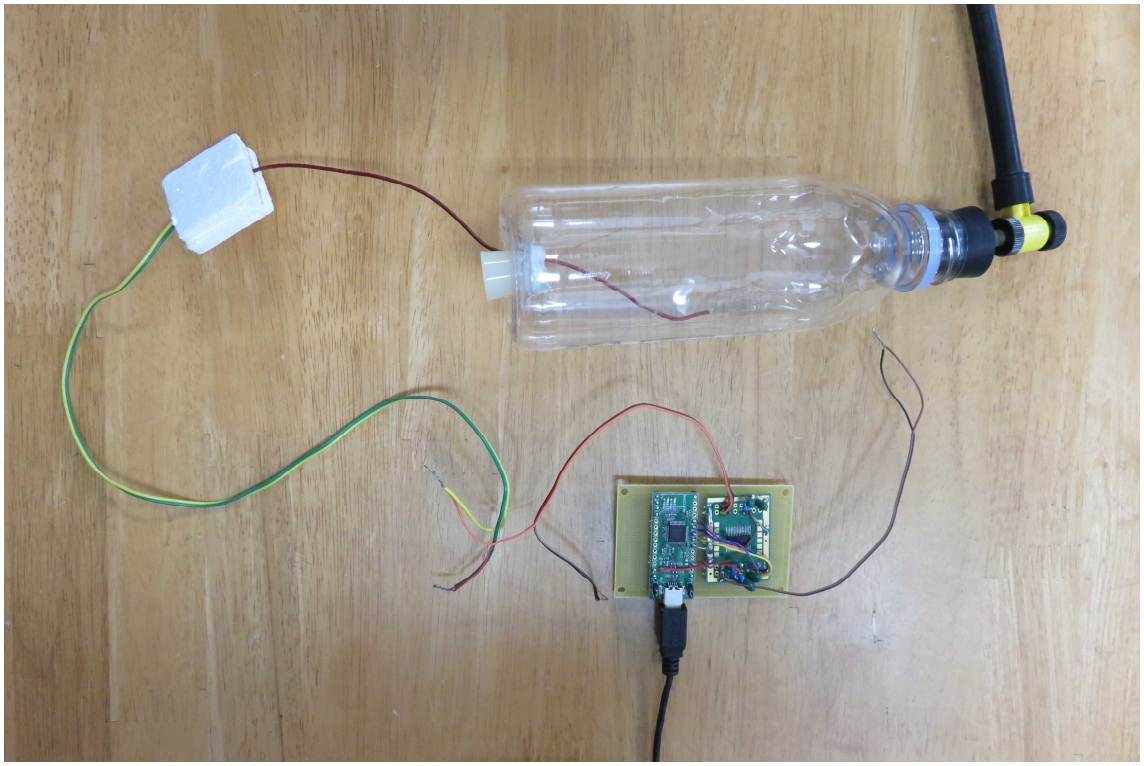
FT232HL（USB シリアル変換モジュール）と ADS1262（AD コンバーター）を使用します。ADS1262 の出力は３２ビットで、３２倍の増幅器を内蔵していますので外部に増幅回路を設けることなく信号を直接 AD コンバーターにつなげて使用しても精度のよい実験ができます。

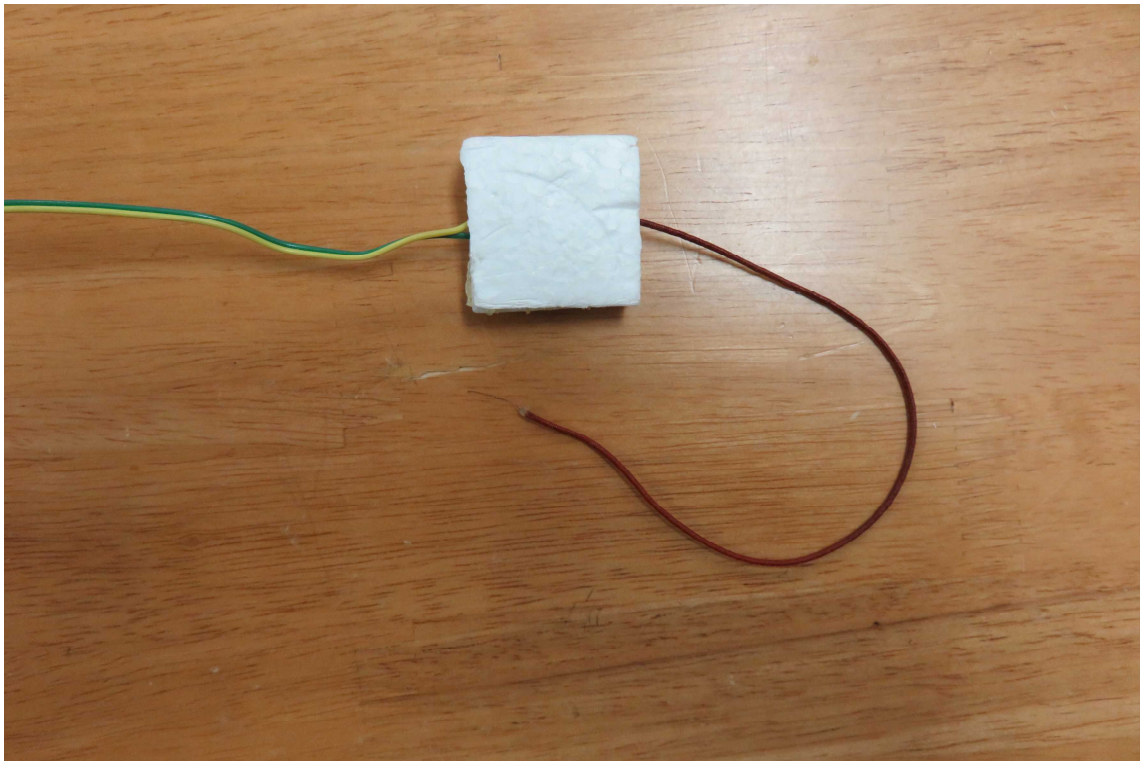
配線図とサンプルプログラムをご覧ください。サンプルプログラムはクロメル アルメル 熱電対(0.07 φ)で空気の断熱圧縮、膨張に伴う温度変化を測定してみたものです。

http://www.cfs.chiba-u.jp/koudai-renkei/information/files/adc_64.xlsm

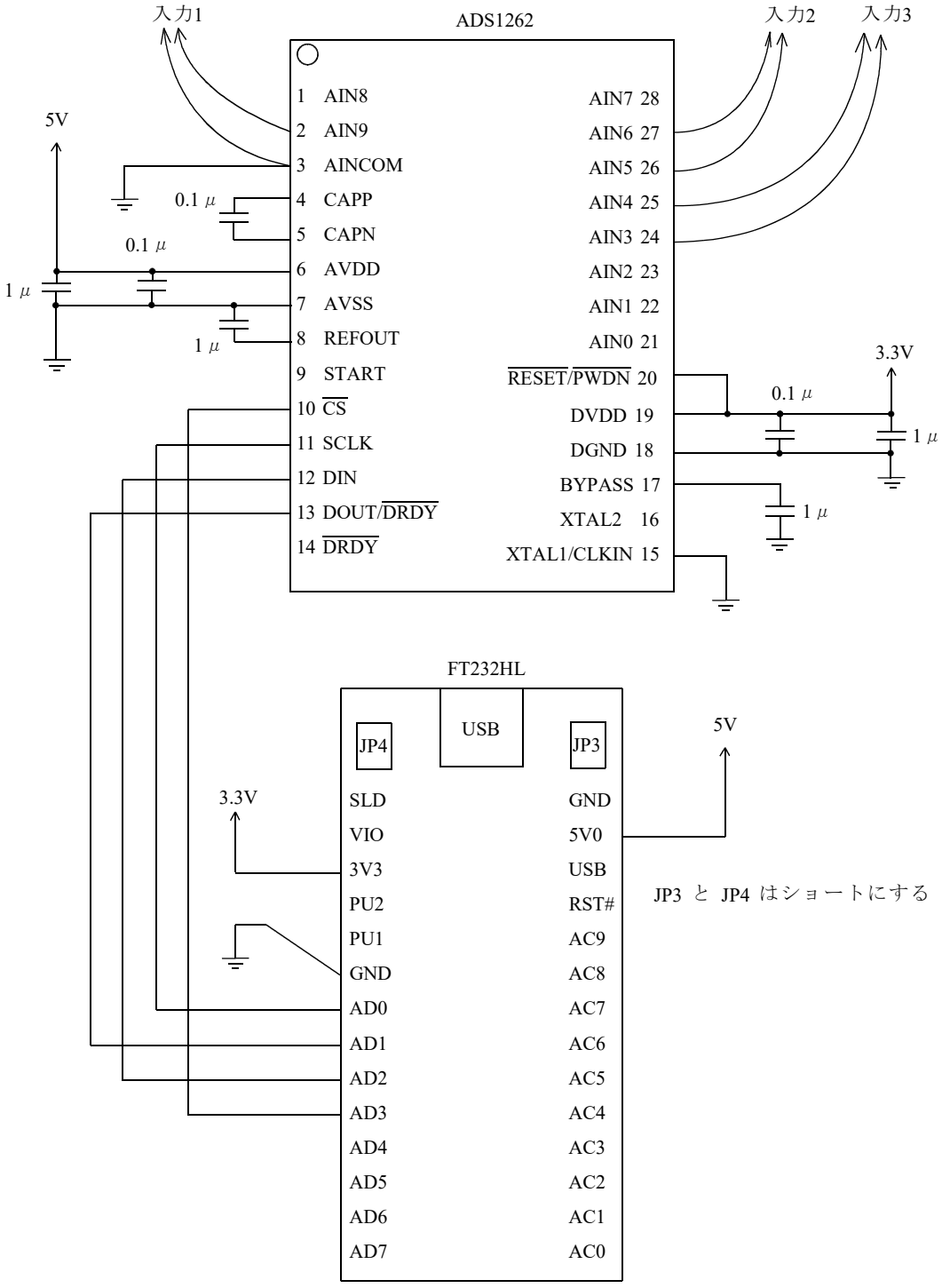
http://www.cfs.chiba-u.jp/koudai-renkei/information/files/adc.xlsm








E-UEGRH
0.07
CLASS
0.4
20M
TOKYO
WIRE
WORKS
LTD.
E-UEGNH
0.07
CLASS
0.4
20M
TOKYO
WIRE


入力1
ADS1262
入力2
入力3
1 AIN8
2 AIN9
3 AINCOM
4 CAPP
5 CAPN
6 AVDD
7 AVSS
8 REFOUT
9 START
10 CS
11 SCLK
12 DIN
13 DOUT/DRDY
14 DRDY
AIN7 28
AIN6 27
AIN5 26
AIN4 25
AIN3 24
AIN2 23
AIN1 22
AIN0 21
RESET/PWDN 20
DVDD 19
DGND 18
BYPASS 17
XTAL2 16
XTAL1/CLKIN 15
5V
0.1 μ
0.1 μ
1 μ
1 μ
3.3V
0.1 μ
1 μ
1 μ
FT232HL
JP4
USB
JP3
3.3V
5V
SLD
VIO
3V3
PU2
PU1
GND
AD0
AD1
AD2
AD3
AD4
AD5
AD6
AD7
GND
5V0
USB
RST#
AC9
AC8
AC7
AC6
AC5
AC4
AC3
AC2
AC1
AC0
JP3 と JP4 はショートにする

以下、箇条書きで参考となる情報を書いてみました。

・FT232HL 及び ADS1262 は検索して情報収集してください

・ADS1262 はサンハヤトの SSP-61 というピッチ変換基板を使用して足幅を広げて使用します。

http://www.sengoku.co.jp/mod/sgk_cart/detail.php?code=55LW-64KY）

・基板にピンをはんだ付（J1,J2,J3（Short）,J4（Short））、電源接続：通常 USB ５ V、IC3.3V 使用

FT232HL で I2C/SPI 通信使用の手順

・FT232HL のドライバーは FTDI 社（秋月電子の商品説明に記載）から入手できます。（http://akizukidenshi.com/catalog/g/gK-06503/）

・デバイスドライバーをインストール：http://www.ftdichip.com/Drivers/D2XX.htm

ここで、ンストールガイドを使用：

http://www.ftdichip.com/Support/Documents/InstallGuides.htm

・FTDI 社が提供している FTD2XX ライブラリの関数（Software Application Development D2XX Programmer's Guide）（AN255 参照）を使用。

From the PC side, the device can communicate in two ways:

・ Virtual COM Port（VCP）接続、I2C 動作モードを使用可能（今回使用しない）

・ D2XX Interface 参照（今回選択） D2XX（DLL）関数を使用する

・USB 接続すると：ユニバーサルシリアルバスコントローラーに

「USB Serial Converter」とポート（COM/LPT）に「USB Serial Port（COM25）」が表示される。

・D2XX_Module.bas に FTD2XX.DLL 中にある関数を使用宣言する：

Public Declare Function FT_Open Lib "FTD2XX.DLL"（ByVal intDeviceNumber As Integer, ByRef lngHandle As Long） As Long などをそのまま貼り付ける。使用する命令：

FT_OpenEx（"FTXQIKRW", 1, ftHandle）

FT_ResetDevice（ftHandle）

FT_SetBaudRate（ftHandle, 115200）

FT_SetTimeouts（ftHandle, 1000, 1000）

FT_SetBitMode（ftHandle, &H0, &H0）

FT_Write（ftHandle, mData（3）, 3, dwNumBytesSent）

FT_GetBitMode（ftHandle, BytesReceived）

FT_Close（ftHandle）

・FT232HL とのパスを開く：FT_OpenEx 命令を使用（デバイス毎の個別認識可能）する

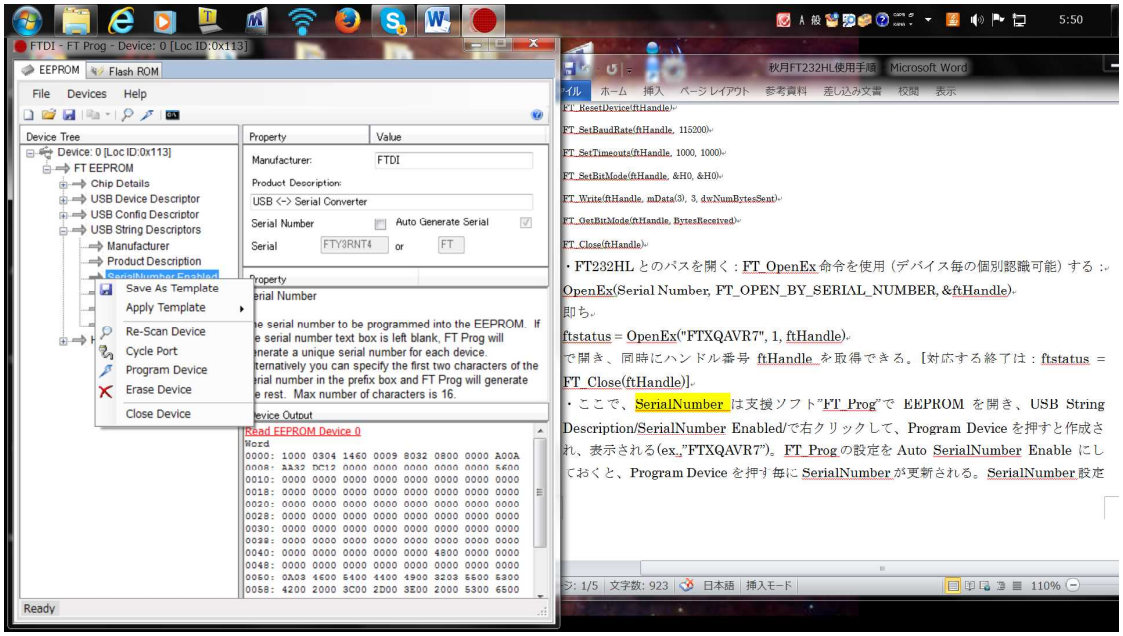
OpenEx（Serial Number, FT_OPEN_BY_SERIAL_NUMBER, &ftHandle）

即ち

ftstatus = OpenEx（"FTXQAVR7", 1, ftHandle）

で開き、同時にハンドル番号 ftHandle を取得できる。［対応する終了は：ftstatus = FT_Close（ftHandle）］

・ここで、SerialNumber は支援ソフト”FT_Prog”（下図参照）



で EEPROM を開き、USB String Description/SerialNumber Enabled/で右クリックして、Program Device を押すと作成され、表示される（ex.,”FTXQAVR7”）。FT_Prog の設定を Auto SerialNumber Enable にしておくと、Program Device を押す毎に SerialNumber が更新されるので、固定する場合は Auto のチェックを外す。

SerialNumber 設定後は PC 本体の、

PC/コントロールパネル/デバイスマネージャー/ユニバーサルシリアルバスコントローラー/USB Serial Converter[USB Serial Port（COMxx）も表示される]/詳細/デバイスインスタンスパス/の値：USB¥VID_0403&PID_6014¥FTXQAVR7 の下¥以下の 8 ケタで表示される（FTDIBUS¥VID_0403+PID_6014+FTY3RNT4A¥0000）。

・SerialNumber を用いると命令は：FT_OpenEx（"FTY3RNT4", 1, ftHandle） となる。

If FT_OpenEx（"FTY3RNT4", 1, ftHandle）<> FT_OK Then …、

又は

ftStatus=FT_ OpenEx（"FTY3RNT4", 1, ftHandle）

を実行すると ftHandle 値を取得できる。

・D2XX 命令で、デバイスリセット、ボーレート設定、通信待ち期限時間を設定する。

・I2C の通信には MPSSE（JTAG, SPI, I2C）を用いる方法と Bit-Bang を用いる方法とがある。現在 MPSSE（I2C）に特に簡単な命令があるわけではないので、ここでは通常の I/O にも使用できる Bit-Bang を使用する。

・AD（0-7）のうち AD0（SCL）、AD1（SDA） を使用する。

AD ポートの入出力設定と非同期 Bit Bang モードの設定は：

ftstatus = FT_SetBitMode（ftHandle, &H3, &H1） 'Set:AD0&1_Out, Other_In, Asynchronous Bit Bang Mode（D2XX 参照）

ftstatus = FT_SetBitMode（ftHandle, &H1, &H1） 'Set:AD0_Out, Other_In, Asynchronous Bit Bang Mode

・AD 出力ポートへの出力：
出力データを mData(0)に書込む：mData(0)=&H0（or 1, 2, 3）
ftstatus = FT_Write(ftHandle, mData(0), 1, dwNumBytesSent)
mData(0)に書込まれた 1 バイトのデータが出力される。
・AD 入力ポートからの読取：
ftstatus = FT_GetBitMode(ftHandle, BytesReceived)
BytesReceived に 1 バイトの数値が入る：SDA(AD1)は下から 2 番目の bit であり BytesReceived&$H2 によって。1 bit の SDA データがえられる。
・FT232HL とのパスを開くには FT_OpenEx 命令を使用（デバイス毎の個別認識可能）する　OpenEx(Serial Number, FT_OPEN_BY_SERIAL_NUMBER, &ftHandle)
命令、即ち　ftstatus = OpenEx("FTXQAVR7", 1, ftHandle)
で開き、同時にハンドル番号 ftHandle を取得できる。[対応する終了は：ftstatus = FT_Close(ftHandle)] プログラム終了時に　FT_Close 命令が必須です。
オープンしたままだと次のオープン命令でエラーとなり実行されません。
・以上の設定が正常であれば、
ftstatus = FT_SetBitMode(ftHandle, &HFD, &H1)
'Set:SCL(AD0/2/3)OUT,SDA(AD1)IN, 00001101
実行後に
ftstatus = FT_WriteByte(ftHandle, &HD, 1, dwNumBytesSent)　'AD3,H; AD2,H; AD1,L; AD0,H(1101)
だけを実行させ、FT232HL 基板上のピンが AD3,H; AD2,H; AD0,H になっている事を確かめて下さい。同様に
ftstatus = FT_WriteByte(ftHandle, &H0, 1, dwNumBytesSent)
を実行させて、AD3,L; AD2,L; AD0,L になっている事を確かめて下さい。
・ftstatus = FT_WriteByte(ftHandle, &H88, 1, dwNumBytesSent) 等としています。ここでは &H88(10001000)が AD7 ～ 0 に出力されます。入力/出力は予め設定されていて、入力設定のピンに H/L を出力しようとしても無視されます。

以下の部分：
ftstatus = FT_WriteByte(ftHandle, &H88, 1, dwNumBytesSent)　'SPI_Close
ftstatus = FT_WriteByte(ftHandle, &H80, 1, dwNumBytesSent)　'1000 SPI_Start
Wdata = &H46: Call ADS_SPIWriteData
Wdata = &H0: Call ADS_SPIWriteData
Wdata = &H78: Call ADS_SPIWriteData
ftstatus = FT_WriteByte(ftHandle, &H88, 1, dwNumBytesSent)　'SPI_Close
では入力（＋）を In7、（－）を In8 に設定しています。
また、
ftstatus = FT_WriteByte(ftHandle, &H88, 1, dwNumBytesSent)　'SPI_Close
ftstatus = FT_WriteByte(ftHandle, &H80, 1, dwNumBytesSent)　'1000 SPI_Start
Wdata = &H45: Call ADS_SPIWriteData

Wdata = &H0: Call ADS_SPIWriteData
Wdata = &H50: Call ADS_SPIWriteData '&H50（2.5sps）/&H52（10sps）
ftstatus = FT_WriteByte（ftHandle, &H88, 1, dwNumBytesSent）　'SPI_Close
部分で、モード 2 設定で PGA（× 32）、2.5sps を設定しています。
これは入力を 2.5V を中心にして 32 倍しているので約 2.495 ～ 2.505V の範囲に入っていない場合は振り切れてしまいます。&H50 を&H80 に置き換えるとプログラムゲインアンプ不使用で 1 倍になります。

ADS1262 駆動プログラム
①リセット Command：
・Write（06h）
②測定条件設定 WREG：対象 Register は；02h（42h）, 03h（43h）, 05h（45h）, 06h（46h）.
・Write（42h, 00h, 00h）：インターフェース設定
・Write（43h, 00h, 40h）：モード 0 設定、1 回測定
・Write（44h, 00h, 80h）：モード 1 設定、Digital Filter:Sinc1（00）/2（20）/3（40）/4（60）/FIR（80）
・［Write（45h, 00h, 50h）：モード 2 設定、TM 用;PGA（× 32）、2.5sps］
［Write（45h, 00h, 51h）：モード 2 設定、TM 用;PGA（× 32）、5sps］
［Write（45h, 00h, 52h）：モード 2 設定、TM 用;PGA（× 32）、10sps］
［Write（45h, 00h, 80h）：モード 2 設定、TS 用;PGA Bypassed、2.5sps］
［Write（45h, 00h, 81h）：モード 2 設定、TS 用;PGA Bypassed、5sps］
・［Write（46h, 00h, 78h）：TM 用入力設定、In7_P、In8_N 使用］
［・Write（46h, 00h, 9Ch）：TS 用入力設定、In9_P、Analog GND_N 使用］
［Write（46h, 00h, 9Ah）：TS 用入力設定、In9_P、InCOM_N 使用］
③ 1 回測定開始 Command：
・Write（08h）
④データ読取 Command + Read：（測定開始後、変換に要する時間経過後に読み取る）
・Write（12h）+Read（4 bytes;32bit）
⑤　測定は③、④の繰り返し
⑥クロック（SCLK）のタイミング
・Write: SCLK ↑時に Dout（PC 側から見て; AD から Din）に Data を設定/SCLK ↓時に書込
・Read: SCLK ↑時に Din（PC 側から見て;AD から Dout）へ Data 設定要求/SCLK ↓時に Din の Data 読取
・上記命令を CS ↓（SPI スタート）と CS ↑（SPI 終了）で囲む：1 命令送信・受信

Write:

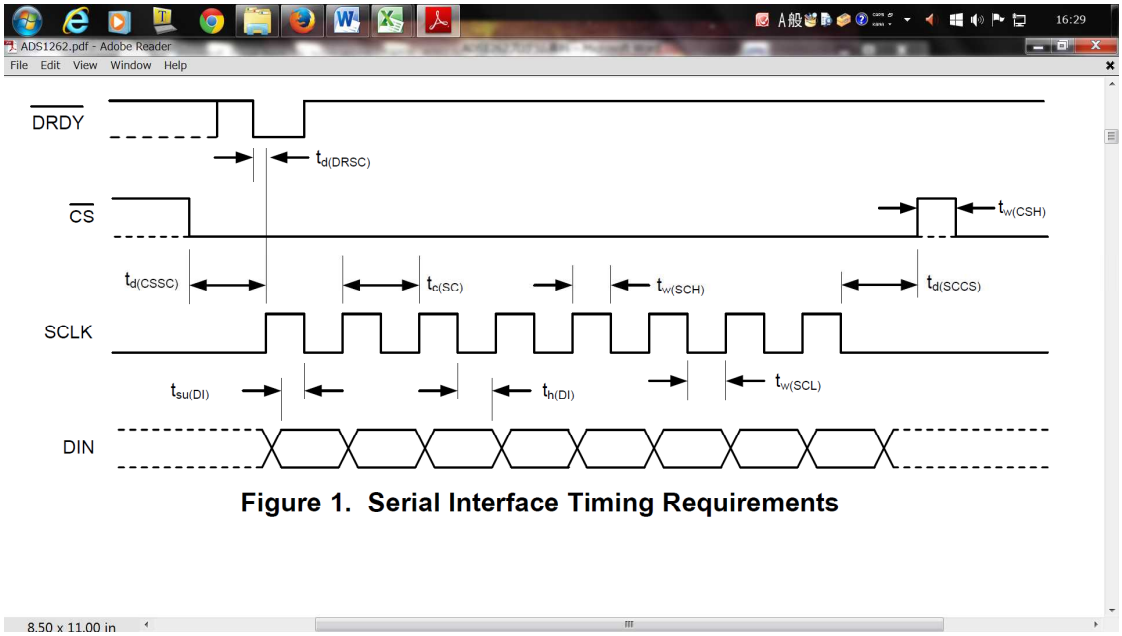
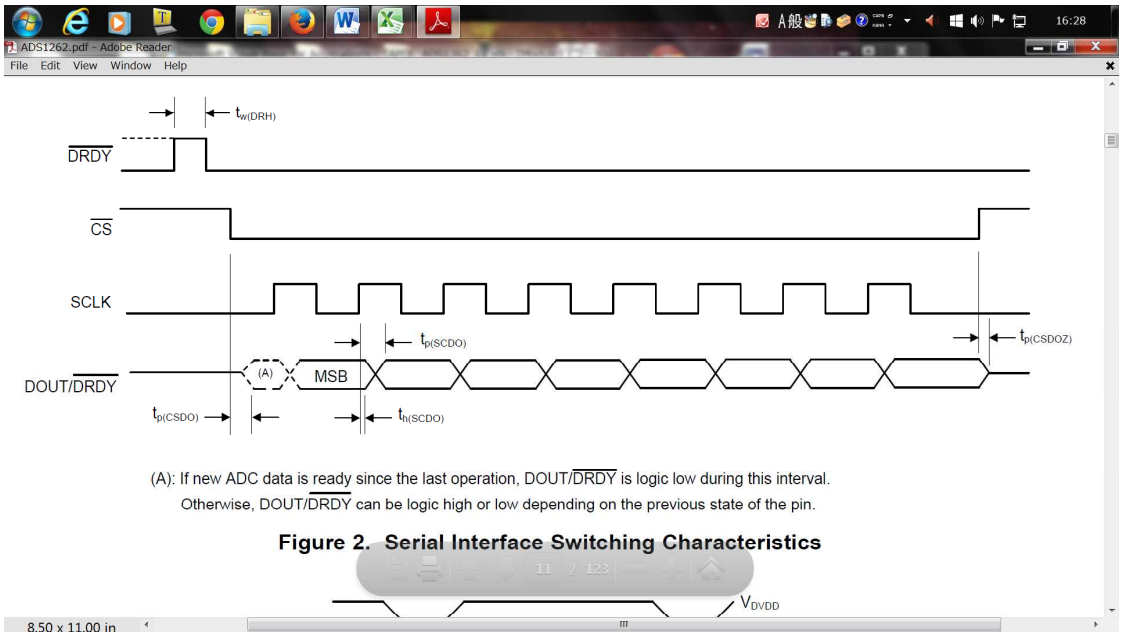


Figure 1. Serial Interface Timing Requirements

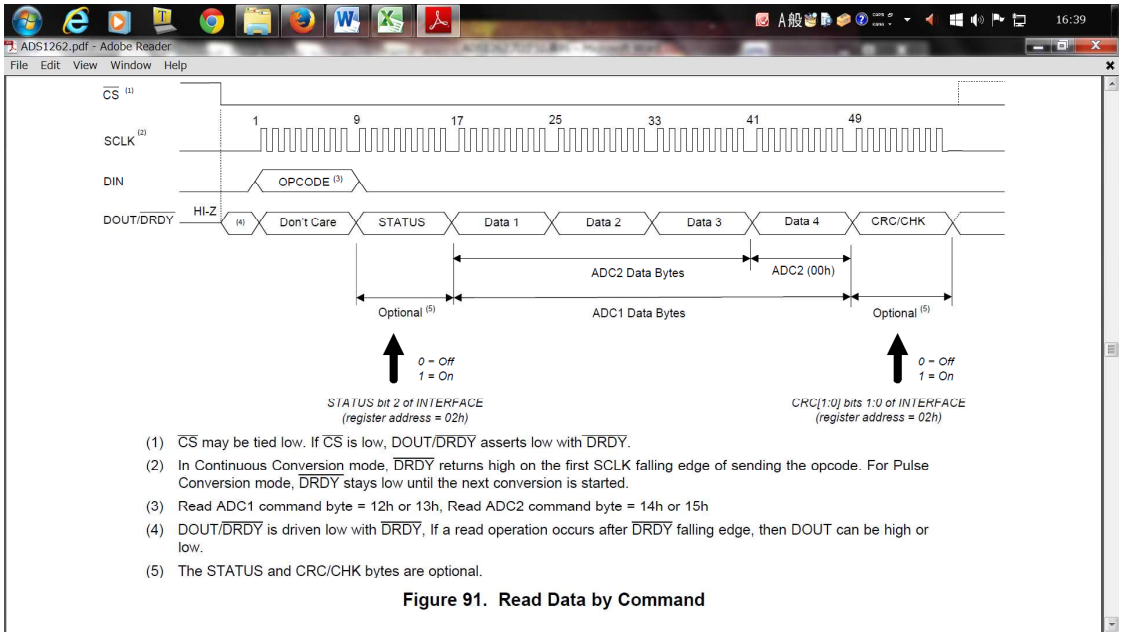
Read:



(A): If new ADC data is ready since the last operation, DOUT/$\overline{\text{DRDY}}$ is logic low during this interval. Otherwise, DOUT/$\overline{\text{DRDY}}$ can be logic high or low depending on the previous state of the pin.

Figure 2. Serial Interface Switching Characteristics

Read Data by Command



(1) $\overline{CS}$ may be tied low. If $\overline{CS}$ is low, DOUT/$\overline{DRDY}$ asserts low with $\overline{DRDY}$.

(2) In Continuous Conversion mode, $\overline{DRDY}$ returns high on the first SCLK falling edge of sending the opcode. For Pulse Conversion mode, $\overline{DRDY}$ stays low until the next conversion is started.

(3) Read ADC1 command byte = 12h or 13h, Read ADC2 command byte = 14h or 15h

(4) DOUT/$\overline{DRDY}$ is driven low with $\overline{DRDY}$, If a read operation occurs after $\overline{DRDY}$ falling edge, then DOUT can be high or low.

(5) The STATUS and CRC/CHK bytes are optional.

Figure 91. Read Data by Command

OPCODE

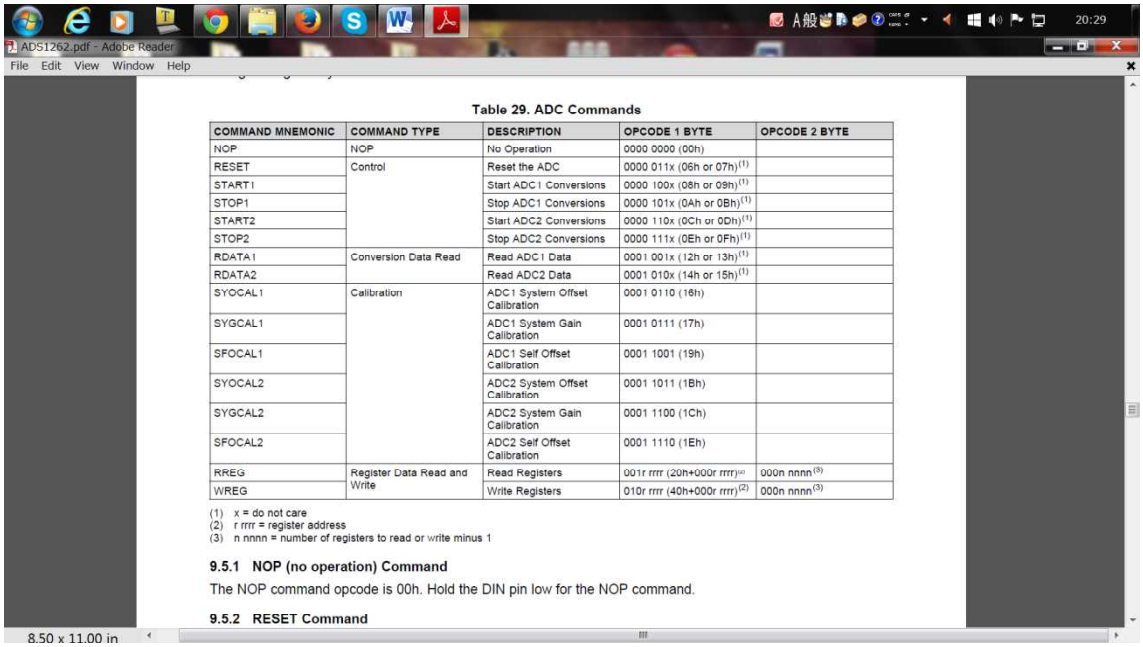


Table 29. ADC Commands

COMMAND MNEMONIC	COMMAND TYPE	DESCRIPTION	OPCODE 1 BYTE	OPCODE 2 BYTE
NOP	NOP	No Operation	0000 0000 (00h)	
RESET	Control	Reset the ADC	0000 011x (06h or 07h)(1)	
START1		Start ADC1 Conversions	0000 100x (08h or 09h)(1)	
STOP1		Stop ADC1 Conversions	0000 101x (0Ah or 0Bh)(1)	
START2		Start ADC2 Conversions	0000 110x (0Ch or 0Dh)(1)	
STOP2		Stop ADC2 Conversions	0000 111x (0Eh or 0Fh)(1)	
RDATA1	Conversion Data Read	Read ADC1 Data	0001 001x (12h or 13h)(1)	
RDATA2		Read ADC2 Data	0001 010x (14h or 15h)(1)	
SYOCAL1	Calibration	ADC1 System Offset Calibration	0001 0110 (16h)	
SYGCAL1		ADC1 System Gain Calibration	0001 0111 (17h)	
SFOCAL1		ADC1 Self Offset Calibration	0001 1001 (19h)	
SYOCAL2		ADC2 System Offset Calibration	0001 1011 (1Bh)	
SYGCAL2		ADC2 System Gain Calibration	0001 1100 (1Ch)	
SFOCAL2		ADC2 Self Offset Calibration	0001 1110 (1Eh)	
RREG	Register Data Read and Write	Read Registers	001r rrrr (20h+000r rrrr)(2)	000n nnnn(3)
WREG		Write Registers	010r rrrr (40h+000r rrrr)(2)	000n nnnn(3)

(1) x = do not care
(2) r rrrr = register address
(3) n nnnn = number of registers to read or write minus 1

9.5.1 NOP (no operation) Command

The NOP command opcode is 00h. Hold the DIN pin low for the NOP command.

9.5.2 RESET Command

9.5.6 RREG Command

The RREG command is used to read the device configuration register data. The register data may be read one register at a time or read as a block of register data. The starting register address may be any register in the map. The RREG opcode consists of two bytes. The first byte specifies the starting register address: 001r rrrr: where r rrrr is the starting register address. The second opcode byte is the number of registers to read (minus 1): 000n nnnn: where n nnnn is the number of registers to read minus 1. After the read command is sent, the ADC responds with one or more register data bytes, most significant bit first. If the byte count exceeds the last register address, the ADC begins to output zero data (the address pointer does not wrap). During the register read operation, if ADC1 data is ready, the conversion data is not loaded to the output shift register to avoid data contention. However, the conversion data can be later retrieved by the RDATA1 command. Once the register read command has been started, further commands are disabled until one of the following conditions:

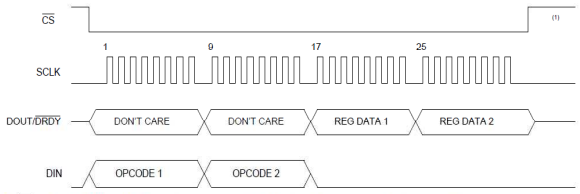
1) The read operation is completed

2) The read operation is terminated by taking $\overline{CS}$ high

3) The read operation is terminated by a serial interface auto-timeout

4) The ADC is reset by toggling the $\overline{RESET}/PWDN$ pin

Figure 103 depicts a two register read operation. For example, the required opcodes to read data from two registers starting at register MODE2 (address = 05h) are: OPCODE1 = 25h and OPCODE2 = 01h. Keep the DIN input low after the two opcode bytes are sent.



(1) $\overline{CS}$ can be set high or kept low between commands. If kept low, the command must be completed.

Figure 103. Read Register Sequence

9.5.7 WREG Command

The WREG command is used to write the device configuration register data. The register data may be written one register at a time or as a block of register data. The starting register address may be any register in the map. The WREG opcode consists of two bytes. The first byte specifies the starting register address: 010r rrrr: where r rrrr is the starting register address The second opcode byte is the number of registers to write (minus 1): 000n nnnn: where n nnnn is the number of registers to write minus 1. The following byte(s) is the register data, most significant bit first. If the byte count exceeds the last register address, the ADC ignores the data (the address pointer does not wrap). During the register write operation, if ADC1 data is ready, the ADC blocks the loading of data to the output shift register. However, the conversion data can be retrieved later by the RDATA1 command. After the register write command has been started further commands are disabled until one of these conditions occur:

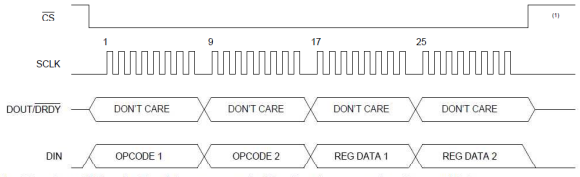
1) The write operation is completed

2) The write operation is terminated by taking $\overline{CS}$ high

3) The write operation is terminated by a serial interface auto-timeout

4) The ADC is reset by toggling the $\overline{RESET}/PWDN$ pin

Figure 104 depicts a two register write operation. For example, the required opcodes to write data to two registers starting at register MODE2 (address = 05h) are: OPCODE1 = 45h and OPCODE2 = 01h.



(1) CS can be set high or kept low between commands. If kept low, the command must be completed.

Figure 104. Write Register Sequence

In the above example, the MODE2 and INPMUX registers are modified. Typically, register changes take effect immediately after the data is written. However, if the registers are part of a group, then the data is written only

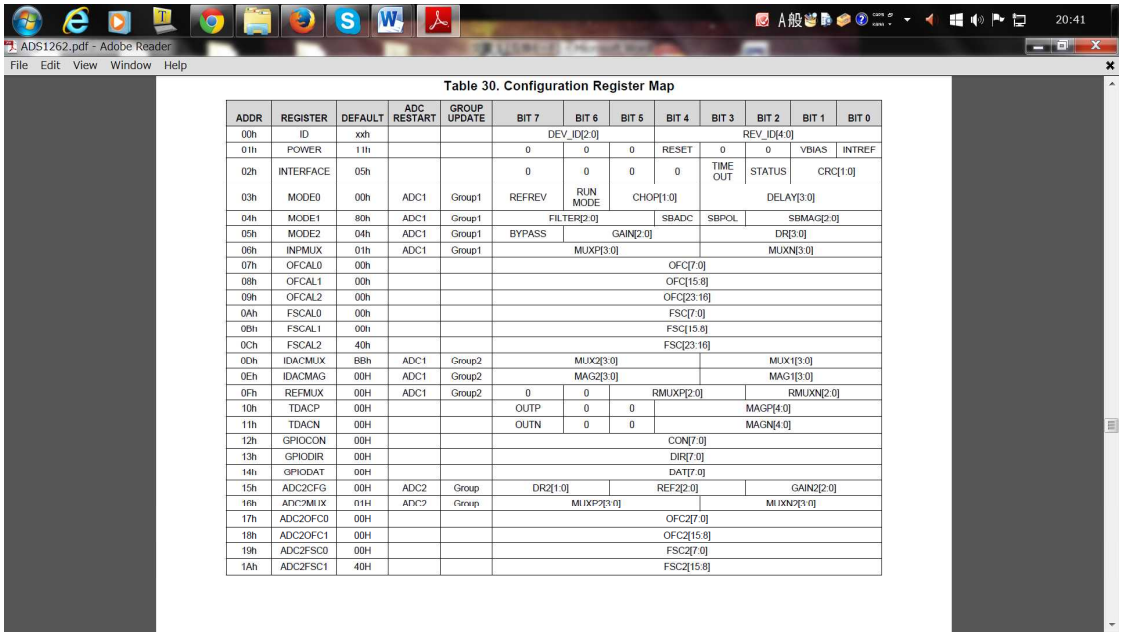
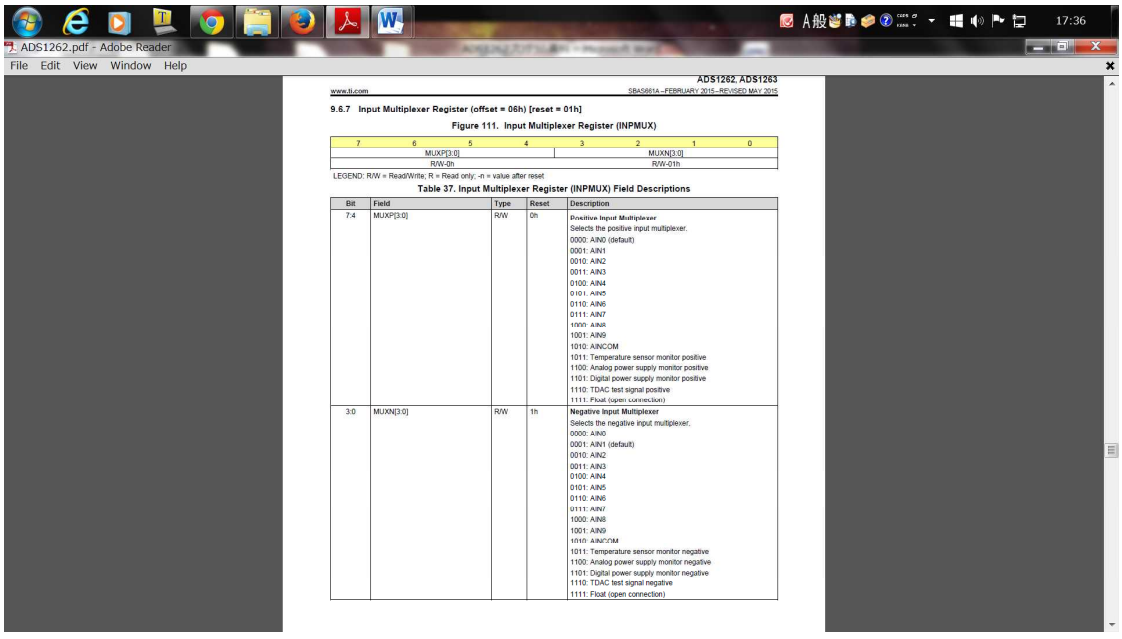


Table 30. Configuration Register Map

ADDR	REGISTER	DEFAULT	ADC RESTART	GROUP UPDATE	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
00h	ID	xxh			DEV_ID[2:0]			REV_ID[4:0]				
01h	POWER	11h			0	0	0	RESET	0	0	VBIAS	INTREF
02h	INTERFACE	05h			0	0	0	0	TIME OUT	STATUS	CRC[1:0]	
03h	MODE0	00h	ADC1	Group1	REFREV	RUN MODE	CHOP[1:0]		DELAY[3:0]			
04h	MODE1	80h	ADC1	Group1	FILTER[2:0]			SBADC	SBPOL	SBMAG[2:0]		
05h	MODE2	04h	ADC1	Group1	BYPASS	GAIN[2:0]		DR[3:0]				
06h	INPMUX	01h	ADC1	Group1	MUXP[3:0]			MUXN[3:0]				
07h	OFCAL0	00h			OFC[7:0]							
08h	OFCAL1	00h			OFC[15:8]							
09h	OFCAL2	00h			OFC[23:16]							
0Ah	FSCAL0	00h			FSC[7:0]							
0Bh	FSCAL1	00h			FSC[15:8]							
0Ch	FSCAL2	40h			FSC[23:16]							
0Dh	IDACMUX	BBh	ADC1	Group2	MUX2[3:0]			MUX1[3:0]				
0Eh	IDACMAG	00H	ADC1	Group2	MAG2[3:0]			MAG1[3:0]				
0Fh	REFMUX	00H	ADC1	Group2	0	0	RMUXP[2:0]		RMUXN[2:0]			
10h	TDACP	00H			OUTP	0	0	MAGP[4:0]				
11h	TDACN	00H			OUTN	0	0	MAGN[4:0]				
12h	GPIOCON	00H			CON[7:0]							
13h	GPIODIR	00H			DIR[7:0]							
14h	GPIODAT	00H			DAT[7:0]							
15h	ADC2CFG	00H	ADC2	Group	DR2[1:0]		REF2[2:0]			GAIN2[2:0]		
16h	ADC2MUX	01H	ADC2	Group	MUXP2[3:0]			MUXN2[3:0]				
17h	ADC2OFC0	00H			OFC2[7:0]							
18h	ADC2OFC1	00H			OFC2[15:8]							
19h	ADC2FSC0	00H			FSC2[7:0]							
1Ah	ADC2FSC1	40H			FSC2[15:8]							

入力設定



9.6.7 Input Multiplexer Register (offset = 06h) [reset = 01h]

Figure 111. Input Multiplexer Register (INPMUX)

7	6	5	4	3	2	1	0
MUXP[3:0]				MUXN[3:0]			
R/W-0h				R/W-01h			

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

Table 37. Input Multiplexer Register (INPMUX) Field Descriptions

Bit	Field	Type	Reset	Description
7:4	MUXP[3:0]	R/W	0h	**Positive Input Multiplexer** Selects the positive input multiplexer. 0000: AIN0 (default) 0001: AIN1 0010: AIN2 0011: AIN3 0100: AIN4 0101: AIN5 0110: AIN6 0111: AIN7 1000: AIN8 1001: AIN9 1010: AINCOM 1011: Temperature sensor monitor positive 1100: Analog power supply monitor positive 1101: Digital power supply monitor positive 1110: TDAC test signal positive 1111: Float (open connection)
3:0	MUXN[3:0]	R/W	1h	**Negative Input Multiplexer** Selects the negative input multiplexer. 0000: AIN0 0001: AIN1 (default) 0010: AIN2 0011: AIN3 0100: AIN4 0101: AIN5 0110: AIN6 0111: AIN7 1000: AIN8 1001: AIN9 1010: AINCOM 1011: Temperature sensor monitor negative 1100: Analog power supply monitor negative 1101: Digital power supply monitor negative 1110: TDAC test signal negative 1111: Float (open connection)

```
'RREG:レジスタ内容（測定データではない）の読取方法；状態確認に使用する
ftstatus = FT_WriteByte（ftHandle, &H88, 1, dwNumBytesSent）  '1000 SPI_Start
ftstatus = FT_WriteByte（ftHandle, &H80, 1, dwNumBytesSent）  '1000 SPI_Start
Wdata = &H23
Call ADS_SPIWriteData
Wdata = &H0
Call ADS_SPIWriteData
ftstatus = FT_WriteByte（ftHandle, &H80, 1, dwNumBytesSent）  '1000 SPI_Start
BytesReceived = 0
For n = 1 To 8
   ftstatus = FT_WriteByte（ftHandle, &H81, 1, dwNumBytesSent）  '0000
   ftstatus = FT_WriteByte（ftHandle, &H80, 1, dwNumBytesSent）  '0001'
   ftstatus = FT_GetBitMode（ftHandle, mData（n））  '1bit Data
   BytesReceived = BytesReceived * 2 +（mData（n） And &H2）/ 2 '受信データを作る
Next n
ftstatus = FT_WriteByte（ftHandle, &H88, 1, dwNumBytesSent）  'SPI_Close
BytesReceived1 = BytesReceived  ‘レジスタ中の 8bit データ
```