FSC - facts about the board

D. Neise

07.02.2011

Contents

1 Basic Facts and Purpose

This document presents electronic facts and figures of FACTs FSC board. FSC stands for **F**act **S**low **C**ontrol. The name is a bit misleading, since the boards sole purpose is to monitor slowly changing parameters, such as:

- up to 64 temperatures (RTD sensors, e.g. Pt1000 or Pt100),
- up to 4 humidities (designed for Honeywell HIH-40xx family),
- the DC levels of all 36 FACT low voltage supply channels,
- the current consumption on each LV-channel (actually a dc-voltage[1]).

FSC uses an 8MHz ATmega32L micro controller(MCU) to readout the muxed 24bit sigma delta ADC (AD7719) which is connected to the RTDs. In addition the ATmegas internal 10bit ADC is used to monitor the humidity, LV-voltages and LV-currents. The muxers on FSC belong to analog devices ADG77xx family, and are controlled by the MCU. The User has access to FSC via Ethernet Interface, provided by WIZNETs W5100, the younger brother of W5300, which is used on FACTs FAD boards.

AD7719 and W5100 communicate with the MCU via an SPI bus, which results in the fact, that a lot of user interaction might slow down temperature measurement or vice versa. Additionally the user should be aware, that sigma delta ADCs have a prolonged settling time, when muxed. Redout of all 64 temperature channels takes about 13s, while one single channel might be readout with at a rate of 10Hz. In addition FSC runs a 32bit timer, counting miliseconds since timer init. The timer is initialized to zero, when FSC is powered. User may synchronize FSCs timer to Unix time.

1.1 Measuring

The user may send a command telling FSC, what to measure. FSC will measure it and return the result, right after the measurement was finished. It is possible to tell FSC to measure several sensors at a time. Users may issue an 'activate' command (see table 4), to specify which channel should be measured during the next measurement phase. In case one is not sure which channel was activated beforehand users may

[1]see FLV specs for further details

issue a 'status' command, to gather information about FSCs current status. When ever the user completed the channel activation process the 'measure' commands may be issued. Since the high resolution ADC AD7719 used for resistance measurement and the 10bit ADC used for voltage measurement run independently on different time scales, different 'measure' commands are defined. 'measure resistances' will start a measurement possibly taking some seconds, while 'measure voltages' will be ready almost instantaneosly. Since a measurement might take some time, FSC generates an answer, once the 'measure' command received, informing the user about the command reception. The results are then submitted in a single message to the user, which is called a telegram. During a 'measure resistance' FSC is not checking for incoming commands. So incoming commands will pile up in W5100 input FIFO. Which means, sending a lot of on demand commands will delay FSCs reaction to the following commannds consecutivly.

1.1.1 RTD switching

FSCs high resolution ADC AD7719 only bears one single input channel. So the RTDs are beeing multiplexed during the resistance measurement. This multplexing includes the RTDs current supply of about $400\,\mu A$. In case the RTDs are near to FACTs G-APD signal lines, low frequency crosstalk might be induced due to these switching processes. In case of FACT there might be two families of RTDs. About 30 pieces will be mounted next to the G-APDs, while a smaller number of sensors will be mounted in the electronics compartement, where this kind of noise will not contribute much.

1.2 The Telegram

As previously outlined the results of a measurement will be submitted in a single message called telegram. The format of this telegram is given in table (1). Despite the fact that the measurement of several RTDs might take several seconds, the storage and transmission of the time of each temperature measurement is regarded as too much overhead, hence the time of the last measurement taken for a telegram is submitted within. Subsequent all activated sensor measurement results are beeing transmitted. A result always consists of a sensor ID and the actual data. Detailed information about the composition of the sensor ID can be found in table (2). The data might be either 3 byte in case of an RTD or 2 byte in case of all other sensors. For detailed information of sensor data see section 1.3.

address	mnemonic	description
byte 0	0x00	special telegram header
byte 1..2	length	length of telegram in byte
byte 3...6	time	time, when measurement was finished.
byte 7	sensor ID 0	
byte 8..x	data 0	data field is eigther 2 or 3 byte wide.
		depending of sensor type.
byte ...	...	sensor IDs and data until end.

Table 1: composition of FSC telegram

ID bits	meaning
00pp.psss	temperatur sensor no. sss on port ppp
01vv.vvvv	voltage sensor no. vvvvvv (between 0..35)
10cc.cccc	current sensor no. cccccc (between 0..35)
1100.0hhh	humidity sensor no. hhh
...	to be completed

Table 2: composition of the sensor ID - ad exemplum

1.3 Sensor Data

FSC supports many different sensors. The first type is any resistive sensor, such as RTDs. The second type is any sensor outputting a voltage between 0VDC and 4.096VDC. In order to keep the firmware independent from the sensors and to keep it as simple as possible, the measured data is treated as less as possible before beeing output.

Since the resistance measurement is perfomed in a ratiometric manner, the resistance is measured as a 24bit fraction of an onboard fix reference resistor of $R_{ref} \approx 6.25k\Omega$ (see table 7). The value of R_{ref} is not stored as a constant value, so the user should measure its value once and submit it to the FSC. In case no value is submitted, $R_{ref} = 6.25\,k\Omega$ is assumed. FSC does not multiply the measured fraction with the reference resistance. Hence the resistance is unknown on FSC and cannot be transformed into a temperature. Similarly the current of the low voltage supply channels, which is converted into a voltage level and measured by FSC is not beeing treated in any way by the MCU but directly transmitted to the user via ethernet. The format of each sensor data is given in table 3.

sensor name	actual physical value	resolution	width	format	unit
temperature	resistance	24 or 16 bit	24	unsigned int	fraction of R_{ref}
humidity	voltage	10 bit	16	unsigned int	V
voltage	voltage	10 bit	16	unsigned int	V
current	voltage	10 bit	16	unsigned int	V

Table 3: sensor data format

2 Ethernet Interface

FSC runs as a TCP/IP server, this means after W5100 was initialized, FSC will listen on a Port, but will not attempt to connect to any server. Since W5100 does not support DHCP, FSC has a fix IP[2]. The Port is defined as 5000. For simplicity of firmware coding, the UI is non human readable. Table (4) shows which commands are defined (so far). Some commands need parameters others don't. Table (5) shows how a message containing a command is defined. Since TCP/IP packages on the ethernet might be delayed up to minutes timescale it is handy to identify command packages by a (nearly-)unique ID instead of its time only. So when the user receives an answer of the FSC, it is possible to relate it to a specific command. Generally FSC copies the package ID of a given command package into its answer.

The following commands are defined:

command	function	description
0x00	-reserved-	-reserved-
0x01	status	returns entire FSC registers. see table (7)
0x02	write reg(U8 addr, U8 data)	write data to register address
0x03	read reg(U8 addr)	return only specified register
0x04	measure active R-channels	see see 1.1
0x05	measure active V-channels	see ...
0x06	measure all active channels	see ...
0x07	set timer(U32 data)	set internal timer
0x08	start timer	start timer after setting. see 2.1.2
0x09	stop timer	stop timer before setting.
...	...	to be completed
0xFF	reset	reset all internal registers and peripherals

Table 4: FSC commands

address	mnemonic	description
byte 0	command	see table4
byte 1	package ID	(nearly-)unique package identifier
byte 2	length	length of data section - if apropriate
byte 3..1k	data	parameters for command - if apropriate

Table 5: composition of FSC commands

FSC answers with a short acknowledgement to any command. An acknowledgement contains the command in the first byte and the (nearly-)unique package ID of the command package in the second

[2]see FACT Elogbook / doc

byte.

address	mnemonic	description
byte 0	command	command, which caused this acknowledgement
byte 1	package ID	package ID of the command package

Table 6: composition of FSCs acknowledgement

If the user issued a measurement command, a telegram is beeing send whenever the measurement is done. In case of free running mode even several telegrams will be submitted. In order to distinguish a telegram from a command acknowledgement, the first byte is always 0x00, see table (1).

2.1 FSC Registers

Table 7 shows an overview of the FSC registers. Most registers may be read and written. Only the first two registers are readonly. Detailed descriptions of the register contents (will) follow. All registers are 8bit wide.

2.1.1 Status Registers

Apart from the possibility of crosstalk due to RTD channel switching, it is of course possible to let FSC monitor all of its active channels an in case one channel exceeds a its threshold, a message is generated. In this case these thresholds need to be stored and this modus needs to be switched on or off. Some additional information might be stored here, such as:

- Timer enabled/disabled
- SPI interface adjusted for: AD7719 / W5100
- ...

2.1.2 Time Registers

Here the current time is stored. Four registers are used to store the current time in seconds, so the current unix time might be transfered to FSC after power up. Additionaly two registers store the current time in milisenconds. This register cannot be set by the user. When the user wants to synchronize FSC to the current time, first the timer should be stopped by issuing the 'stop timer' command. Then the Time Registers should be written and finally the 'start timer' command should be sent, causing the milisecond registers to be reset as well.

2.1.3 The Enable Registers

Most probably not every input channel will be connected to a sensor. In order not to measure unconnected channels, the user may define, which sensor is active, by writing to the registers, TempEnx, HumiEnx, AcceEnx, VoltEnx and CurrEnx the apropriate bitpattern.

2.1.4 Ideas for more registers

address	**name**	**description**
0x00	status3	status register. TBR
0x01	status2	see 2.1.1
0x02	status1	
0x03	status0	
0x02	time_s3	current time in secondsMSB
0x03	time_s2	see 2.1.2
0x04	time_s1	
0x05	time_s0	current time in seconds LSB
0x08	time_ms1	current time, fraction of miliseconds MSB
0x09	time_ms0	-the same- LSB
0x0A	FRperiod1	time in seconds between two free running measurements MSB
0x0B	FRperiod0	-the same- LSB
0x0C	RREF1	reference resistor value in ohms (MSB)
0x0D	RREF0	reference resistor value in ohms (LSB)
0x10	TempEn7	
0x11	TempEn6	
0x12	TempEn5	
0x13	TempEn4	
0x14	TempEn3	
0x15	TempEn2	
0x16	TempEn1	
0x17	TempEn0	bitmap defining which channel
0x18	HumiEn0	is activated
0x19	VoltEn4	
0x1A	VoltEn3	
0x1B	VoltEn2	
0x1C	VoltEn1	
0x1D	VoltEn0	
0x1E	CurrEn4	
0x1F	CurrEn3	
0x20	CurrEn2	
0x21	CurrEn1	
0x22	CurrEn0	
0x30	TempDone7	
0x31	TempDone6	
0x32	TempDone5	
0x33	TempDone4	
0x34	TempDone3	
0x35	TempDone2	
0x36	TempDone1	
0x37	TempDone0	bitmap defining which channel
0x38	HumiDone0	is already measured
0x39	VoltDone4	
0x3A	VoltDone3	
0x3B	VoltDone2	
0x3C	VoltDone1	
0x3D	VoltDone0	
0x3E	CurrDone4	
0x3F	CurrDone3	
0x40	CurrDone2	
0x41	CurrDone1	
0x42	CurrDone0	
...	...	to be completed

Table 7: FSC registers