Reading and writing TMC2300 registers

Firmware has implemented functionality to read and write individual Trinamic TMC2300 registers over internal UART single wire interface. It shares transmit and receive line like an RS485 based interface. Data transmission is secured using a cyclic redundancy check and each TMC2300 driver is addressed individually as follows:

Axis	Address
X	0
Y	1
Z	2
A	3
Broadcast to all axes	**9**

Official TMC2300 documentation

All numbers in g-code arguments are decimal base numbers

Write registers

Write register command forwards g-command arguments to TMC2300 configuration interface. While it is not recommended because of stability issues, write command can be sent even if motors are turning (G0, G1, ... commands)

7'th bit of TMC2300 register has to be set to 1, thus for example **0x10** IHOLD_IRUN becomes **0x90**

G100	**P**<xx>	**L**<xx>	**N**<xx>	**S**<xx>	**F**<xx>	**R**<xx>

G-command	TMC2300 address	TMC2300 register(mask with0x80)	DATA3	DATA2	DATA1	DATA0

UART WRITE ACCESS DATAGRAM STRUCTURE																			
each byte is LSB...MSB, highest byte transmitted first																			
0 ... 63																			
sync + reserved								**8 bit slave address**			**RW + 7 bit register addr.**			**32 bit data**			**CRC**		
0...7								8...15			16...23			24...55			56...63		
1	0	1	0	Reserved (don't cares but included in CRC)				*SLAVEADDR*=(MS2, MS1)			register address		1	data bytes 3, 2, 1, 0 (high to low byte)			CRC		
0	1	2	3	4	5	6	7	8	⋮	15	16	⋮	23	24	⋮	55	56	⋮	63

Required checksum is calculated in firmware

Read registers

Read registers command has two arguments TMC2300 address and register address. It responds to terminal immediately.

G101	**P**<xx>	**R**<xx>
G-command	TMC2300 address	TMC2300 register

Response is decimal coded 12 bytes. Sample reply looks like this:

```
[ TMC: 5, 0, 16, 112, 5, 255, 16, 0, 0, 0, 0, 157]
```

Reply byte	Meaning
0	Echo transmit byte0
1	Echo transmit byte0
2	Echo transmit byte0
3	Echo transmit byte0
4	Reply from TMC2300 sync (0x05)
5	Address
6	Register
7	Data byte3

8	Data byte2
9	Data byte1
10	Data byte0
11	Checksum

UART READ ACCESS REQUEST DATAGRAM STRUCTURE																							
each byte is LSB...MSB, highest byte transmitted first																							
sync + reserved								**8 bit slave address**				**RW + 7 bit register address**				**CRC**							
0...7								8...15				16...23				24...31							
1	0	1	0	Reserved (don't cares but included in CRC)				*SLAVEADDR*=(MS2,MS1)				register address			0	CRC							
0	1	2	3	4	5	6	7	8	...		15	16	...		23	24	...		31				

UART READ ACCESS REPLY DATAGRAM STRUCTURE																							
each byte is LSB...MSB, highest byte transmitted first																							
0 63																							
sync + reserved								**8 bit master address**				**RW + 7 bit register addr.**				**32 bit data**				**CRC**			
0...7								8...15				16...23				24...55				56...63			
1	0	1	0	reserved (0)				0xFF				register address			0	data bytes 3, 2, 1, 0 (high to low byte)				CRC			
0	1	2	3	4	5	6	7	8	...		15	16	...		23	24	...		55	56	...		63

Examples

Set motor X current

Each motor driver can drive different motor. Power and other settings can be set individually. For example IHOLD_IRUN (0x10) register controls motor idle, run and timing parameters.

```
G100 P0 L144 N0 S0 F10 R5
```

G100	Write to TMC2300 g-command
P0	Driver address = 0 (X axis)
L144	TMC2300 register IHOLD_IRUN (0x10) **0x10** ->(set 7'th bit to 1)-> **0x90** = **144** (dec)
N0	Data3 = 0 (not used)
S0	Data2 = 0 (see **IHOLDDELAY** in table below)
F10	Data1 = 10 (see **IRUN** in table below)
R5	Data0 = 5 (see **IHOLD** in table below)

VELOCITY DEPENDENT DRIVER FEATURE CONTROL REGISTER SET (0x10...0x1F)					
R/W	**Addr**	**n**	***Register***	**Description */ bit names***	
W	0x10	5 + 5 + 4	*IHOLD_IRUN*	**Bit**	***IHOLD_IRUN* – Driver current control**
				4..0	*IHOLD* (*Reset default=8*) Standstill current (0=1/32 ... 31=32/32) Setting *IHOLD*=0 allows to choose freewheeling or coil short circuit (passive braking) for motor stand still.
				12..8	*IRUN* (*Reset default=31*) Motor run current (8=9/32 ... 31=32/32) Working with values below 8 is not recommended. *Hint:* Choose sense resistors in a way, that normal *IRUN* is 16 to 31 for best performance.
				19..16	*IHOLDDELAY* (*Reset default=1*) Controls the number of clock cycles for motor power down after standstill is detected (*stst*=1) and *TPOWERDOWN* has expired. The smooth transition avoids a motor jerk upon power down. 0: instant power down 1..15: Delay per current reduction step in multiple of 2^18 clocks

Read input register

Each TMC2300 driver has input status registers GSTAT (0x01) indicating over temperature, short circuit, and more.

```
G101 P0 R1
```

Response looks like:

```
[ TMC: 5, 0, 1, 193, 5, 255, 1, 0, 0, 0, 1, 154]
```

Response contains only 3 bits of useful information. See explanation in table below.

R+ WC	0x01	3	*GSTAT*	**Bit**	***GSTAT* – Global status flags** (Re-Write with '1' bit to clear respective flags)
				0	*reset* 1: Indicates that the IC has been reset since the last read access to *GSTAT*. All registers have been cleared to reset values.
				1	*drv_err* 1: Indicates, that the driver has been shut down due to overtemperature or short circuit detection since the last read access. Read DRV_STATUS for details. The flag can only be cleared when all error conditions are cleared.
				2	*u3v5* 1: Actual state of the supply voltage comparator. A high value means that the voltage sinks below 3.5V. This flag is not latched and thus does not need to be cleared.

Revision #10
Created 25 September 2020 04:48:35 by Saulius
Updated 3 January 2021 13:45:53 by Saulius