

1.GC3355 Register Configuration Format

GC3355 register configuration format include command header and data.

Command header is marked by 55 AA which should be sent firstly, then host send CLUS_ADDR (4 bits), CHIP_ADDR (4 bits), REG_ADDR (8 bits) sequentially.

After command header, host is able to write $N \times 32$ bits data consecutively ($N \geq 1$) to REG_ADDR, REG_ADDR+1, REG_ADDR+2, ... REG_ADDR+N-1.

Whatever command header and data, the order of 32 bits is:

Byte0/Bit [7:0], Byte1/Bit [15:8], Byte2/Bit [23:16], Byte3/Bit [31:24].

Because UART will transmit each byte from bit [0] to bit [7], 32bits data is transmitted from bit [0] to bit [31] on UART.

	Byte3	Byte2	Byte1	Byte0	Transmit Order	Data@Register
Command Header	REG_ADDR	{CLUS_ADDR, CHIP_ADDR}	AA	55	55 AA {CLUS_ADDR, CHIP_ADDR} REG_ADDR	
DATA0	B3	B2	B1	B0	B0 B1 B2 B3	0xB3B2B1B0 @ REG_ADDR
DATA1	B7	B6	B5	B4	B4 B5 B6 B7	0xB7B6B5B4 @ REG_ADDR+1
DATA2	BB	BA	B9	B8	B8 B9 BA BB	0xBBBAB9B8 @ REG_ADDR+1

There are 3 CHIP_ID pins indicate the chip address. GC3355 only accept register access when CHIP_ADDR of command header match CHIP_ID or CHIP_ADDR is 0xf (It means this is a broadcast command)

GC3355 supports burst mode for register configuration, so the below 2 sequences have the same function. GC3355 request time interval between 2 commands must longer than UART timeout value.

Sequence 1: Command Header (REG_ADDR), DATA0, DATA1, DATA2, DATA3

Sequence2: Command Header (REG_ADDR), DATA0 -> Command Header (REG_ADDR), DATA1, DATA2, DATA3

2.CPM Register

CLUS_ADDR=0xE

REG ADDR (8 bits)	NAME	DEFAUT VALUE	Description
0x0	[31]pll_BP	1	PLL Bypass mode select, Fout = Fin
	[30]pll_BS	0	PLL Band Select 1: High band, 500MHz<=Fvco<=1GHz 0: Low band, 300MHz<=Fvco<=600MHz
	[29:28]pll_OD	2'b01	PLL Output divider, valid when pll_BP is 0 00: Fout = Fvco 01: Fout = Fvco /2 10: Fout = Fvco /4 11: Fout = Fvco/8
	[27:21]pll_F	0x27	PLL Feedback divider $Fvco = Fref * (pll_F + 1)$
	[20:16]pll_R	0	PLL Input divider $Fref = Fin / (pll_R + 1)$
	[6]core_clk_out1_diven	0	1: enable core output clock divider 0: disable core output clock divider
	[5]core_clk_sel1	1	0: core clock = core clock out0

			1: core clock = external clock
	[4]core_clk_sel0	0	0: core clock out0 = PLL output clock 1: core clock out0 = PLL output clock/2
	[3]pll_clk_gate	0	PLL output clock gate
	[2]pll_recfg	0	Re-config PLL
	[0]cfg_cpm	0	Config CPM, Write 1, Clear by HW All pll parameters are updated until SW set this bit.
0x1	pad_output_mode [1:0]	2'b00	//pad_output_mode: This register affect RPT_P/PRT_N/BTC_BUSY/LTC_BUSYN //00: open-drain mode //01: output low //10: open-drain mode, same to 00 //11: normal mode Please refer to GC3355 datasheet and GC3355 application note for detail.
0x2	btc_clk_en[31:0]	0xffffffff	BTC core clock N enable
0x3	btc_clk_en[63:32]	0xffffffff	N:0-159
0x4	btc_clk_en[95:64]	0xffffffff	
0x5	btc_clk_en[127:96]	0xffffffff	
0x6	btc_clk_en[159:128]	0xffffffff	
0x10	[27:0]pwm_clkl	0x10000	PWM low period, unit is external clock cycle
0x11	[31]pwm_en	0	PWM enable
	[27:0]pwm_clkh	0x10000	PWM high period, unit is external clock cycle

0x20	[28]uart_rev_fast	1	<p>Uart receiver fast mode</p> <p>0: Uart receiver need wait 1 bit time on UART after stop bit</p> <p>1: Uart receiver don't wait</p>
	[27:26]uart_mode	2'b01	<p>Baud rate divider mode</p> <p>00: 4 times oversample</p> <p>01: 8 times oversample</p> <p>10: 16 times oversample</p>
	[25:16]uart_divider0	0x82	<p>Baud rate divider fraction</p> <p>$X.Y = \text{External Clock Frequency} / (\text{Target Baudrate} * \text{uart_mode})$</p> <p>$\text{uart_divider0} = \text{INT}(0.Y * 1024)$</p>
	[15:0]uart_divider1	0x1b	<p>Baud rate divider integer</p> <p>$X.Y = \text{External Clock Frequency} / (\text{Target Baudrate} * \text{uart_brdiv_mode})$</p> <p>$\text{uart_divider1} = X$</p>
0x21	[31]timeout_en	1	<p>Uart timeout enable</p> <p>Enable UART clear internal state when timeout happened.</p>
	[15:0]timeout_value	0x1f	<p>Uart timeout threshold</p> <p>Time interval between 2 commands.</p> <p>The unit is 1 bit time on UART.</p>
0x30	[6]lrc_sel[1]	0	<p>1: rpt_p is used as lrc_rpt</p> <p>0: rpt_p is not used as lrc_rpt</p>

	[5]ltc_sel[0]	0	1: rpt_n is used as ltc_rpt 0: rpt_n is not used as ltc_rpt
	[3]ltc_clk_disable	0	1: disable LTC clock 0: enable LTC clock
	[2:0]ltc_div	3'b001	LTC core clock divider LTC core clock = BTC core clock/(ltc_div+1)

LTC register is only writable when LTC_EN=1. ltc_sel is effective right way when writing register. Others need write cfg_cpm to be effective

3.BTC Register

CLUS_ADDR=0x0

REG ADDR (8 bits)	NAME	DEFAUT VALUE	Description
0x0	NONCE		BTC Initial NONCE
0x1	A0		BTC MIDSTATE
0x2	B0		
0x3	C0		
0x4	D0		
0x5	E0		
0x6	F0		
0x7	G0		
0x8	H0		
0x9	W0		BTC DATA2
0xa	W1		
0xb	W2		

0x1e	[31]bist_en		Bist mode enable
	[27:16]bist_core_cnt		Bist mode core counter
	[15:0]bist_dly_cnt		Bist mode delay counter
0xff	[31]force_start	1	1: start BTC after write W2 0: start BTC after nonce overflow
	[30]uart_en	0	SW UART enable
	[29]dbg_uart	0	Enable BTC UART mode BTC UART transmits data which is directly from BTC UART receiver.
	[28]rpt_swap	1	BTC report Byte Order 1: Byte0,Byte1,Byte2,Byte3 0: Byte3,Byte2,Byte1,Byte0
	[15:0] rpt_cycle	0x10f4	BTC report Baud rate BTC core clock/rpt_cycle is BTC UART Report Baud Rate For example, 500MHz/0x10f4 ~= 115200 bps

4.LTC Register

CLUS_ADDR=0x1

REG ADDR	NAME	DEFAUT	Description
(8 bits)		VALUE	

0x0-0x7	D0-D7		LTC Target
0x8-0xf	D8-D15		LTC MIDSTATE
0x10-0x22	D16-D34		LTC DATA_IN
0x23	NONCE_MIN		The minimum nonce value
0x24	NONCE_MAX		The maximum nonce value
0x28	[4:3]ltc_cmp_mode	2'b00	LTC compare mode 2'b00: LTC match if scrypt result == Target 2'b10: LTC match if scrypt result <= Target
	[2]use_ltc_uart	0	Use LTC UART receiver
	[1]sw_rpt_rst	1	SW reset LTC report unit, active low This reset also need SW clear
	[0]sw_unit_rst	1	SW reset LTC calculation unit, active low This reset also need SW clear