For LGT8F684A project

Design for Test

Version 2015/10/27

List of Contents:

Overview

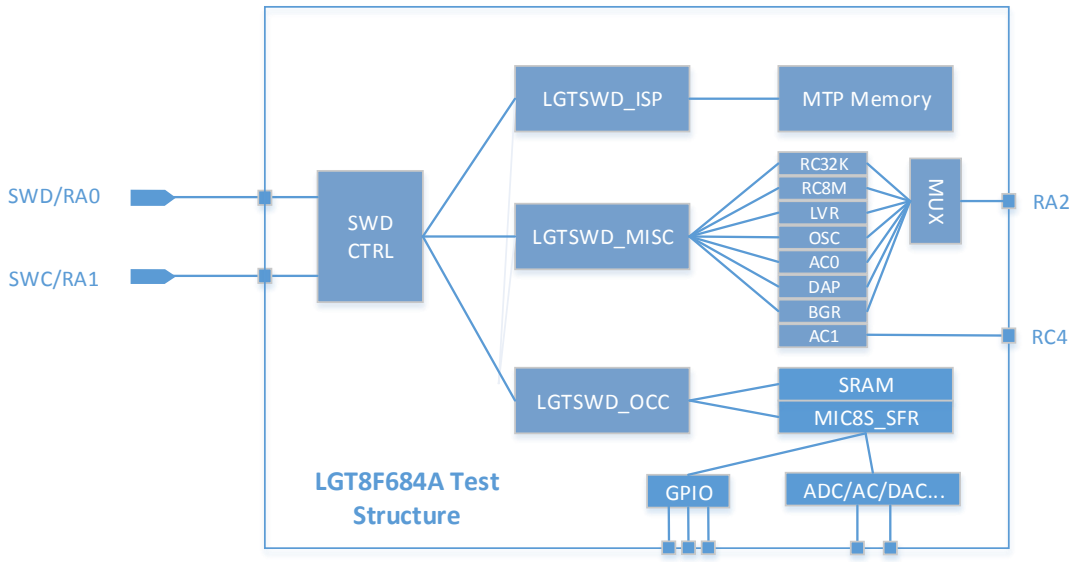
LGT8F684A includes a standard LGTSWD interface for MTP programming and final test.

The final test is designed for mass production, it can be used to test almost all the digital and analog functions with the help of integrated test logic:

LGTSWD_ISP: for MTP memory programming and test (REF1 read test, MTP test after mass-production)

LGTSWD_MISC: for internal analog IP test, including internal 32-KHz RC, LVR and trimming for analog modules

LGTSWD_OCC: mic8s remote control interface, which can be used to implement ram test, digital peripherals test (e.g., GPIO, Timers) and assistant for analog trimming. Test flow is controlled by OCC interface. All of the internal registers and SRAM can be controlled remotely from LGTSWD interface.



As depicted above, a standard LGTSWD controller allocated at the top of all the test logic. All the control and data are serial in/out via two I/O ports. Under SWD controller, there are mainly three type of modules for different test functions.

For MISC functions test, we use an I/O port (RA2) for analog or digital output, which will used to output internal clocks, comparator result and other analog outputs.

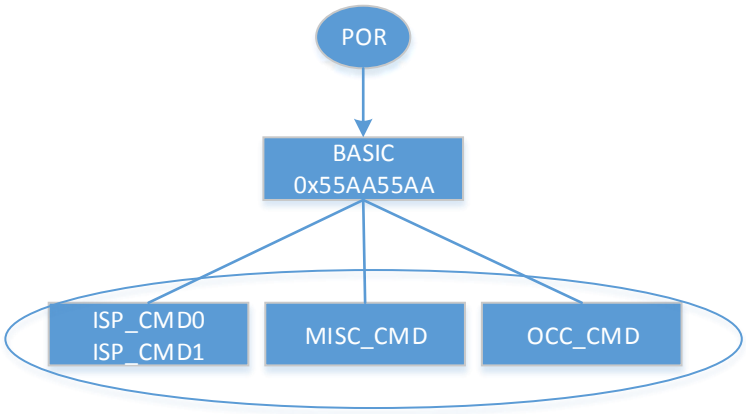
For OCC test cases, all of internal logic can be controlled from SWD by external test master. For example, internal 128x8bit SRAM can be read/write directly. Other digital peripherals can be controller indirectly by SFR registers, e.g. GPIO, Timers and analog modules.

LGTSWD controller use a standard two wire interface which is compatible with other LGT 8bits series microcontroller. But this type implementation has a bit different control timing which we will detailed later.

LGTSWD initialize timing

After system power on, SWD interface is enabled to an input-only model. A [**BASIC]** command sequence must be sent to enable internal test logic. After BASIC command, SWD enters a full function model.

[**BASIC]** command must be the first command before any operations, even before read SWDID. This is different to other LGTSWD implementations.



Just as standard LGTSWD interface timing, data package are byte oriented, send by low byte firstly. For example, to send BASIC command sequence, data bytes should be sent according to following pattern: **AA -> 55 -> AA -> 55**.

After valid [**BAISC]** sequence, we can now feel free to run all other operations.

MTP programming and verify interface

Frame definitions

LGTSWD_ISP is designed for MTP programming and verification only. There is no mass erase or page erase operations, only read/write and test interface are implemented.

LGTSWD_ISP frame uses two type of commands to separate address frame and data frame. Each frame type has 16bits (two bytes) long.

The entire MTP is divided into two blocks: 2Kx14bit main data block and 16x16bit information block. We use a bit named [PIF] in address frame to identify them. Generally, there are no difference between main and information block. Except for data verify process.

Address frame (A-Frame) definitions:

A-Frame Command: **0xB2**

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

[15]	VPPEN	1: Enable VPP for MTP programming 0: For data read or verify
[14]	VERIFY	1: For data verify (only for main data region)
[13]	CLEN	VPPEN = 0 : set CLEN to enable CLEN test mode VPPEN = 1 : CLEN is used to select clock source for programming {VPPEN, CLEN} = 10 : select internal RC8M/128 for MTP's CLKIN 11 : select SWC for MTP's CLKIN

[12]	PWE	MTP write strobe, high active
[11]	PIF	Information block select: 1 = information block 0 = main data block
[10:0]	PADDR	Target Address for MTP. Shared for both block types.

Data frame (D-Frame) definitions:

D-Frame Command: **0xB3**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
[15:0]		DATA		Data frame send to LGTISP controller. The data has very different definition which determined by its corresponding address frame type: 1. 16bit data for MTP programming 2. 16bit data for main block data verification 3. 16bit dummy data for information data read											

Read data/status out of LGTSWD_ISP controller

We need read data from LGTSWD_ISP for three cases:

- Read programming status while writing
- Read result while main data verification
- Read 16bit information data

Reading frame is started by sending a command package, after some SWD idle cycle, issue a two bytes SWD shift-out sequence to receive data. After SWD shift sequence, some SWD cycles (at least 4 cycles, 8 cycles recommended) are also needed for protocol integrity.

Frame definition for LGTISP read operation:

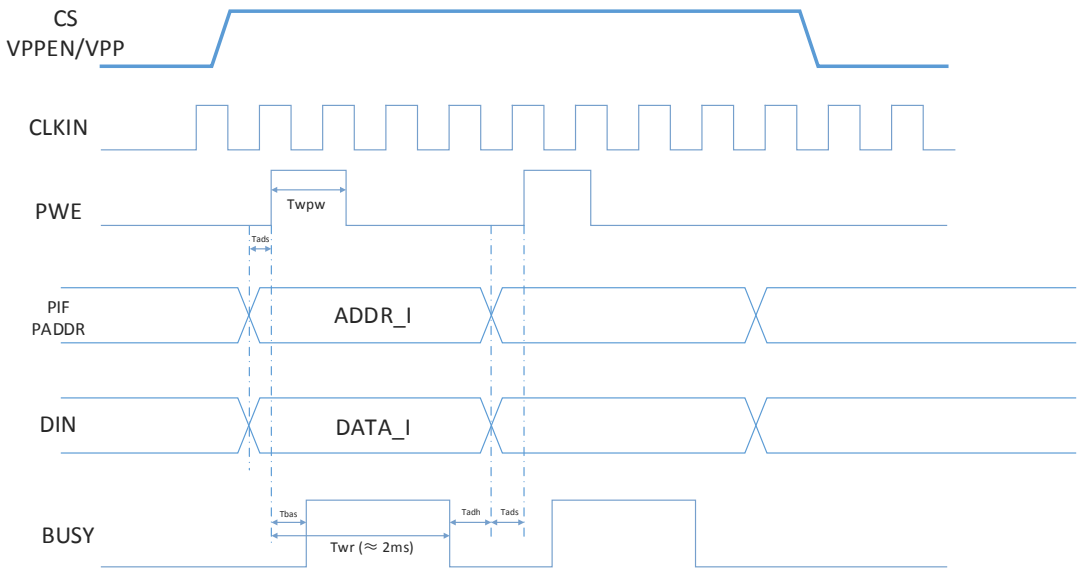
Command: **0xAB**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
[15:0]		DATA		Data read from LGTISP controller. Bits definition is based on the operation type: 1. For programming, DATA[1] = BUSY/IDLE, 1 means busy 2. For main data verify, DATA[0] = PASS/FAIL, 1 mean pass 3. For information reading, DATA[15:0] = raw 16bit information data											

MTP Programming Timing

MTP programming is performed by combine A-frame and D-frame sequence to generate valid program timing of MTP memory.

Word Program Cycle of MTP memory:



Before we issue actually programming timing, there are some preparatory work needed. Firstly, generate a stable clock signal for MTP programming. This is a very slow clock about 10us~15us period, typically, a 12us clock is recommended. In LGT8F684A, there are two options for generating this clock: internal RC8M/128 or SWC/2. Internal clock source is recommended, by using this source, we do not need to care of this clock while generating MTP timing.

We can select this internal clock source by set [CLEN] bit in A-Frame of LGTISP programing sequences. When internal clock is selected, RCM is switch to 8MHz configuration automatically. But by default, this 8MHz RCM is not calibrated. Infect we also do not need 8MHz to generator CLKIN signal, for generating a 10us~15us period, we need about 8.5~12.8MHz divided by 128 ratio. So the first thing is to calibrate RCM to about 8.5~12.8MHz range. In most case, write OSCTUNE to **0x80** is just works for that purpose. OSCTUNE register is allocated at address of 0x90, we need an OCC write command to complete this task (for details of LGTSWD_OCC, please refer to next chapter of this manual).

After select and set clock for programming, it's mostly get ready. But consider of that there is a probability that the SWD interface might be disabled by user code or SWD/SWC port might be used for analog purpose. In those case, we need to hold MCU's core in reset status to enable SWD interface. To do this, we need to following steps:

1. Drive external reset port to low to keep in reset status
2. Send SWD reset command (CID: 0x20)
3. Release external reset port

If external reset port is configured for I/O and SWD port is also disabled or used for analog function, the chip is LOCKED!!!

Demo for programming MTP via SWD interface:

OPERATION	CMD	FRAME
Step1: Reset MCU using MCRE		(optional, only for robust design)
Step2: Send SWD MCU reset	0x20	(optional, only for robust design)
Step3: Release MCRE		(optional, only for robust design)
Step4: Send **BASIC**	0xD0	0x55AA55AA
Step5: RCM Calibration	-------	LGTSWD_OCC write 0x80 to address 0x90 See OCC chapter for details
Step6: set VPPEN, PIF and CLEN	0xB2	Send A-Frame with VPPEN = 1 Set PIF according to target address/region. Set CLEN = 1 for internal programming clock source.
Step7: Drive HV on VPP port	Enable HV(9.6V) on VPP port for MTP programming And check VPP feedback for confirmation.	
Step8: Prepare data	0xB3	Issue D-Frame with data
Step9: set address and PIF	0xB2	Issue A-Frame with address and keep other control bits
Step10: set PWE high	0xB2	Issue A-Frame with PWE set high and keep others
Step11: set PWE low	0xB2	Issue A-Frame with PWE set low and keep others
Delay about 2ms for programming		
Step12: Read BUSY status	0xAB	Issue read command until BUSY is low It's recommended to include a timeout counter here
Delay about 10us for recovery		
Go to Step 8 for next address	Its recommended do programming by sector, a sector of 256 words (512 bytes) is proper, because we do data verification based on sector of 256 words. For information block, program entire 16x16 block is recommended.	
Go to next sector until all done	We will do data verification after all programming tasks.	
Step 13: Remove HV on VPP port	Check and verify HV is removed	
Step 14: Clean up control bits	Reset VPPEN and other control bits Hold [PIF] bit for following data verification	
Programming is done!!		

MTP Main Data Verification

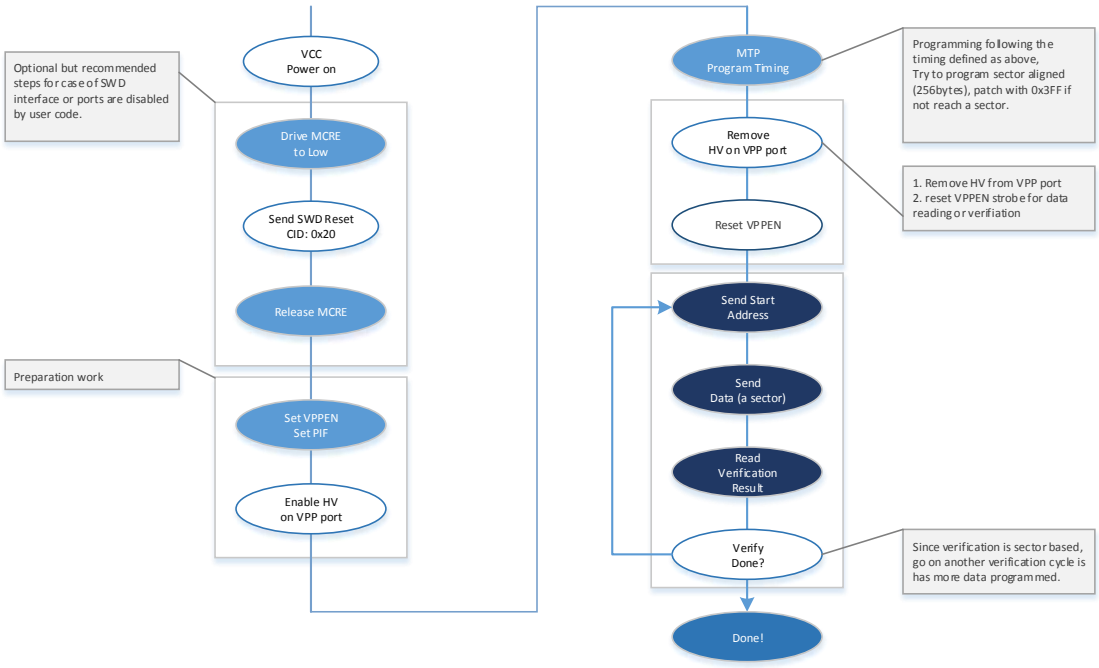
After programming, a data verify sequence is necessary to ensure that data is written without any problem. For data security consideration, there is no channel to read raw data out of MTP's main data memory. To process data verification, you have to resend the right data to compare with internal memory.

Data verification is sector based, with 256 words per sector. To start data verification, an address frame (A-frame) must be send firstly to initialize verification status and setup start address for data confirmation. Then followed by 256 words data (no A-frame needed) which should be the data your just program to that sector. After that, try to get verification result by issue a read operation from SWD interface.

Demo for main data verification:

OPERATION	CMD	FRAME
Step0: send BASIC	0xD0	0x55AA55AA It's optional if we coming from program process.
Step1: Verification setup	0xB2	Issue A-Frame with [VERIFY] = 1 Make sure [PIF] point to right region Setup start address (sector based)
Step2: Send sector data	0xB3	Send data of the entire sector, word by word.
Step3: Read verify result	0xAB	Read a byte to check the result Byte[0] = 1, data verification pass
Step5: Done		

Finally, here is a complete flow for MTP programming and verification in real application:



Configuration Words definition

LGT8F684A has totally 16x16bits information block. The first 6x16bit are reserved for system configuration words, but only first 2x16bit can be written by end users. The lefts are protected by a super string.

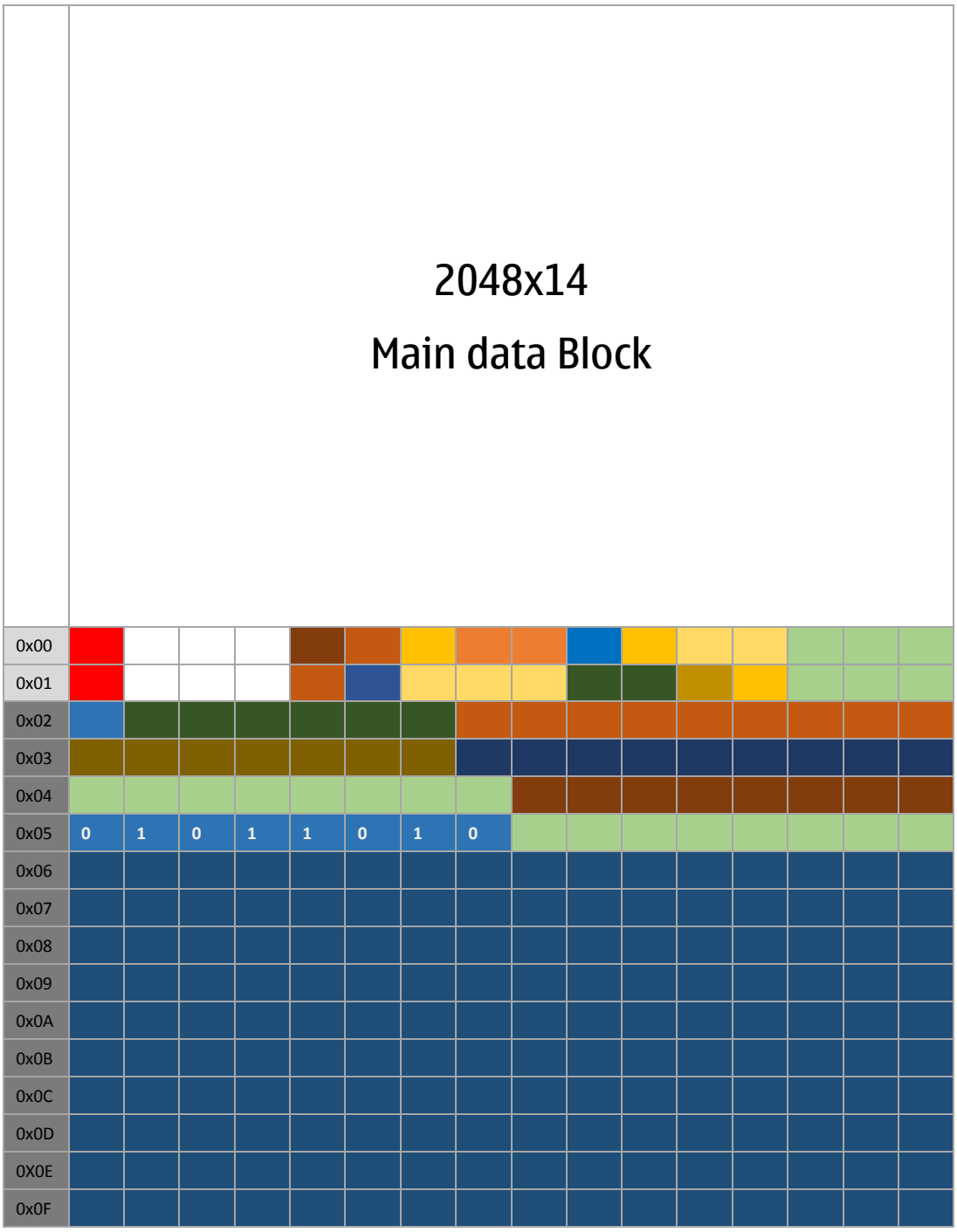
There are two type of configuration words: the first 2x16bits including all the system configuration bits, they can be modified by any users. The followed 4x16bits are trimming bits which should be programmed while do factory test and calibrate. Those 4x16bits with all the left 10x16bits are protected with super mode. Users try to write those address will roll back to first two words.

The programming timing for information block is entirely equal to the timing of main data block. But the data verification process is difference. The raw data of information block can be read out directly by LGTSWD_ISP interface. So data verification is easy and straight.

Note that, end user can only read raw data of first 2x16bit information block. The data of other 14x16bits are protected by super mode.

(For security reason, the super mode will not be descripted in any documentation)

MTP main data and information block definitions:



CF0: Configuration Word 0		
Bit[2:0]	FOSC	Oscillator working mode select 000/111: RCM 110: N/C 101: HFOSC 100: N/C 011: LFOSC 010: CLKIN Others: N/C
Bit[4:3]	RCM	RCM center frequency selection 00: 1MHz 01: 8MHz 1X: 16MHz
Bit[5]	OSCO	FSYS output on RA4 1: FSYS output on RA4 0: RA4 as I/O
Bit[6]	TSSM	Two-speed startup mode enable 1: Enable 0: Disable
Bit[8:7]	SUT[1:0]	Startup timer selection 00: 63ms 01: 254ms 10: 150us 11: 2ms
Bit[9]	WDTE	Watch dog enable 1: Enable 0: Disable
Bit[10]	MCRE	External reset enable 1: Enable 0: Disable
Bit[14:11]	-	N/C
Bit[15]	CF0EN	0 : CF0 is active 1 : CF0 is inactive

CF1: Configuration Word 1		
Bit[2:0]	LVDT	LVD threshold settings 111: Disable LVD 000: 1.8V 001: 2.0V 010: 2.2V 011: 2.4V 100: 2.6V 101: 3.6V 110: 4.0V
Bit[3]	LVDPM	LVD power mode 1: keep LVD in sleep mode 0: disable LVD in sleep mode
Bit[4]	PINRM	PORT Read mode (only for BCF/BSF) 1: read from port register 0: read from PIN status
Bit[6:5]	DPSM	Sleep mode control 00: Normal mode 01: Power save 1X: deep sleep mode
Bit[8:7]	TCYC	MIC8S core cycle settings 00: 1T 01: 2T 1X: 4T
Bit[9]	OSCFSEN	Crystal fail-safe enable 1: Enable 0: Disable
Bit[10]	PPLP	MTP low power mode 1: Normal mode 0: Low power mode
Bit[14:11]	-	N/C
Bit[15]	CF1EN	0: CF1 is active 1: CF1 is inactive

Configuration Word 2		
Bit[8:0]	C0TR	AC0 trimming C0TR[8]: AC0 trimming enable/disable C0TR[7]: AC0 trimming direction C0TR[6:0]: AC0 trimming data
Bit[14:9]	VRTR	BGR trimming
Bit[15]	DPTR[7]	Bit[7] of DAP trimming

Configuration Word 3		
Bit[8:0]	C1TR	AC1 trimming C1TR[8]: AC1 trimming enable/disable C1TR[7]: AC1 trimming direction C1TR[6:0]: AC1 trimming data
Bit[15:9]	DPTR	Bit[6:0] of DAP trimming DPTR[7]: DAP trimming enable/disable DPTR[6]: DAP trimming direction DPTR[5:0]: DAP trimming data

Configuration Word 4		
Bit[7:0]	RCCAL1	RCM:8M trimming
Bit[15:8]	RCCAL2	RCM:16M trimming

Configuration Word 5		
Bit[7:0]	RCCAL0	RCM:1M trimming
Bit[15:9]	Magic	Should be always 0x5A to enable configuration words

Configuration word programming and verification

The programming timing for MTP's information block is equal to its main block area. The difference is that you should set [**PIF] of A-Frame** to address inside information block. The minimize access unit is word based. But it's recommended to write all information data inside one VPP cycle.

The raw data of information block can be read out directly. So the data verification is easy and straightly.

Remember to send **super command sequences** before accessing factory settings region or reserved information block (the high 14x16bit).

Here is a demo for reading information block:

OPERATION	CMD	FRAME
Step1: Send BASIC	0xD0	0x55AA55AA
Step2: Send Super Command	-----	For get full authority for information block
Step3: Enable verify	0xB2	A-Frame with [VERIFY]=1 and [PIF] = 1 Set address to the location inside information block
Step4: Send dummy data frame	0xB3	D-Frame with any data (dummy)
Step5: Read data	0xAB	Read two bytes out of LGTSWD_ISP interface Low byte coming first. Combined with high byte to return 16bit raw information data
Step6: Done		

LGTSWD_MISC test Interface

Test interface include two sub device for testing different parts of final production.

LGTSWD_MISC is for kinds of individual analog IP test. IP settings are set by LGTSWD_MISC interface, test result can be read from LGTSWD_MISC or measured from dedicated pins.

LGTSWD_OCC is design for another type test. Mainly for digital peripherals and memory. You can control the whole digital core via this OCC interface, including all GPIO, timers, Analog comparator and so on.

Frame structure definition

This instance of LGTSWD_MISC implementation is only a write interface, means you cannot read out of from LGTSWD_MISC. It's designed for configurations only. The desired output can be monitor by dedicated test pins.

Control data frame definition for LGTSWD_MISC:

Frame Command: **0xB7**															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

[15]	EN	MISC test enable
[14]	PKGS	Package select: Keep to 1 always.
[13:11]	TUS	Test unit select 000 : RC32K 001 : RC8M 010: LVR 011: crystal I/O test 100: AC0 trimming update 101: AC1 trimming update 110: BGR trimming update

		111: DAP trimming update
[10:9]	RCM	Internal 8MHz RC mode 00: 1MHz 01: 8MHz 1X: 16MHz
[8:0]	CTLD	Control data for selected test unit 1. Bit[7:0] for Trimming data for rc1/8/16m test 2. Bit[2:0] for LVR threshold for LVR test 3. Bit[0] for Crystal mode select for OSC test 4. Bit[8:0] for AC0/1 trimming control 5. Bit[5:0] for BGR trimming data 6. Bit[7:0] for DAP trimming control

Test operations

For most test case, RA2 is used as a dedicated pin for test output. RA2 works as analog type by default. When certain test case valid, RA2 will automatically set to digital type for digital signal output. Those case includes RC32K/RCM/LVR and external crystal I/O test. But for others tests, you might have to set function and direction of this I/O by control the corresponding registers. You can access internal I/O registers over LGTSWD_OCC interface, which descripted in next chapter.

Test output	RC32K	RCM	LVR	OSC	AC0	AC1	DAP	BGR
RA2	✓	✓	✓	✓	✓		✓	✓
RC3						✓		

For cases of RC32K/RCM/LVR/OSC, the test output RA2 works as digital automatically. For DAP/BGR, the test output RA2 works as analog IO which is the default status. But you should do some additional work to redirect analog output to this IO. For example, in order to bring internal DAP's output to RA2, you will need to configure several registers by using LGTSWD_OCC interface.

Case 1: RC32K/RCM/LVR/OSC

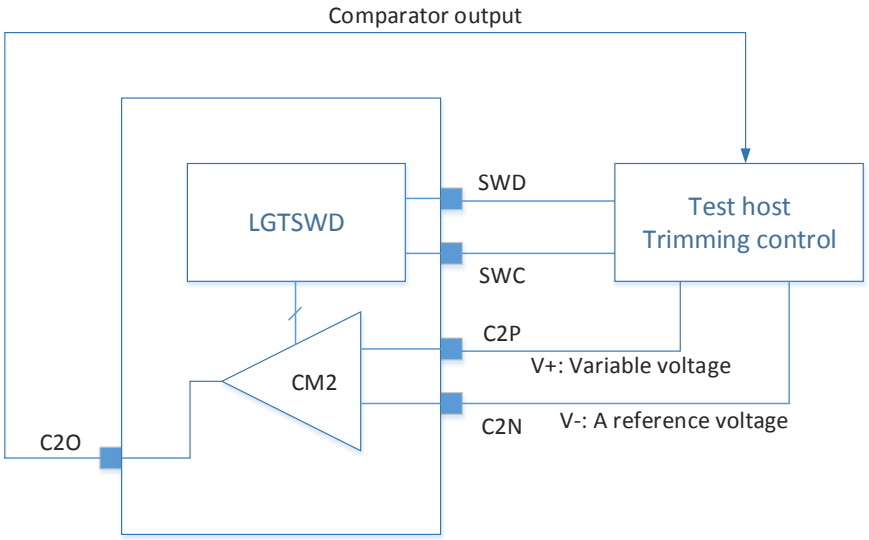
The control of those cases are very simple. Send LGTSWD_MISC control frame with TUS set to desired test unit. The test output will output out of RA2 immediately. Sub-function selection and corresponding control data should be given in low bytes of MISC control frame.

Case 2: AC0/1 trimming

Test case for AC0/1 is used for offset-error correction. LGTSWD_MISC frame for AC0/1 is just used to control the error correction (trimming). The AC's working mode and inputs/output control should be configured using LGTSWD_OCC channel by accessing AC's control registers.

The trimming settings for AC0/1 is 9 bits long, combined with 3 components, including trimming enable/disable control, trimming direction and trimming data.

AC0/1 trimming circuit:



AC0's external input are multiplex with SWD interface, so we have to do trimming control by using AC1 (CM2). The test hardware for AC trimming should be like above structure. The test host control test flow via SWD interface. AC's inputs are given by test host with one channel used as a fixed reference voltage and another channel used as variable source. the AC's output C2O can be used to monitor the compare results.

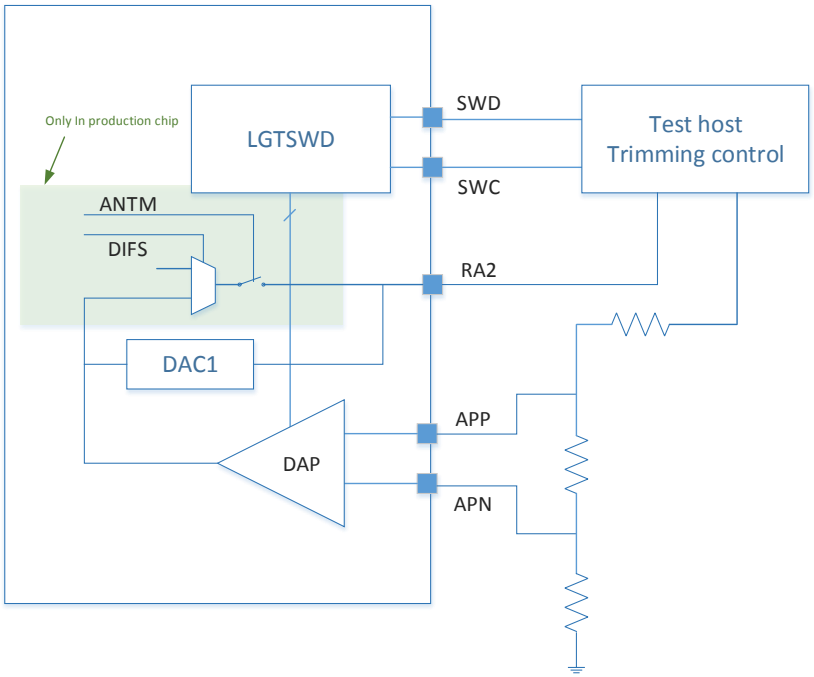
We should set desired resolution firstly for offset-error trimming. Let's say "±10mV".
The trimming procedure should be like this:

1. Configure CM2, sets its input channels and enable output to C2O
2. Given a fixed reference voltage on C2N
3. Increase/Decrease C2P towards C2N and monitor the status of C2O
4. If the C2O changed out of the range of ±10mV, update trimming data through LGTSWD_MISC interface and go to step 3 again.

Case 2: DAP trimming

DAP is test by measure the amplify output from external I/O. But in engineering test chip, DAP's output is routed to AVREF port after multiplexed with ADC's input channels. AVREF is multiplexed with SWC before go to outside world. So we have to find another path to test DAP's output. Here is a temporary way to measure DAP's output from outside of chip:

1. Route the DAP's output to AVREF net by configure DIFS and ANTM bits
2. RA1/AVREF is a digital pin by default, so keep as its initial status.
3. Internal AVREF net is route to internal DAC's input after a voltage follower.
4. We can configure DAC to select AVREF for reference input. Set ADC's level to full-phase level (VR = 0x3F) which is equal to output AVREF directly
5. Configure DAC's output to I/O port. Now we can get DAP from the output port of DAC.
6. For transfer through a voltage follower, there will be a little drop between the final output and the original output of DAP, about 2~5mV range. You can compensate the output by adding this drop for high resolution.



[DAP test structure]

The test circuit for DAP trimming is mostly as above. There are two type of application of DAP: high side current measurement and low-side current measurement. We can try to trim DAP to any side application or try to find a center point to balance those two conditions.

The different of the two application is determined by the common-voltage of inputs. In high side application, the input voltage of APP/APN are very close to power supply which means DAP works around a higher common input level. In low-side measurement, the input voltage of APP/APN is close to ground level, which means smaller common level.

So how to determine the trimming point for DAP's offset error, is dependent on its application. For most case, we can try to find a point which can archive acceptable results for both applications.

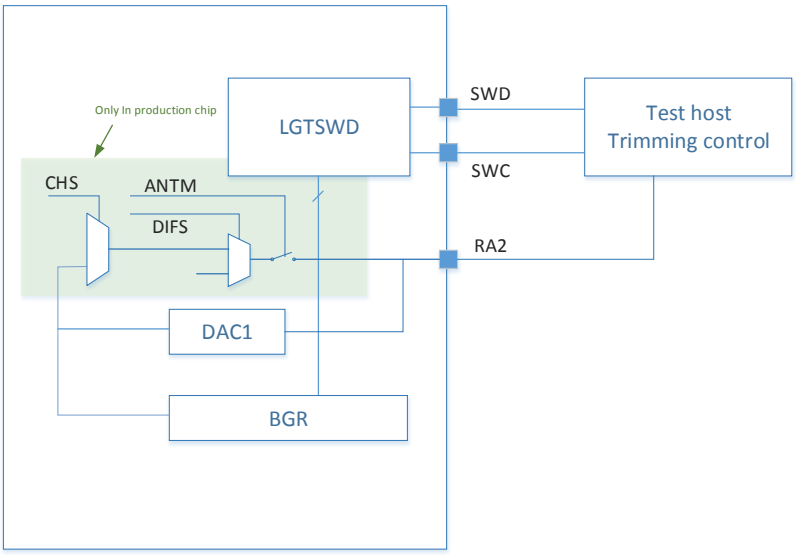
In our final production chip, DAP's output can also be routed from ADC's internal channel and then redirect to outside world from RA2 by ANTM mode. This path has no voltage follower, so the output is more accurate compare to above structure.

Case 2: BGR trimming

BGR trimming is used to calibrate internal 1.2V band-gap reference. The trimming procedure is implemented by measure the output of BGR, compare with 1.2V, then increase/decrease the trimming data according to the current output level.

The test circuit structure is mostly like the structure for DAP calibration, but more simple to control because there is no need of external components.

Just like DAP's situation, we can redirect BGR's output from internal DAC by selecting BGR as its reference source, then configure DAC for full-phase level output. Finally we can test the BGR from outside world by measure RA2/DAC1 port.



[BGR trimming test structure]

LGTSWD_OCC test interface

The function of LGTSWD_OCC is very simple, design for internal SFR register and SRAM accessing. By control those resource, we get control of all the peripherals indirectly.

When LGTSWD_OCC is selected, MIC8S core will be break and hold here. So it's very safe for remote control without interrupt by instruction execution of core.

After all settings done, deselect LGTSWD_OCC will bring core active and run as normal.

LGTSWD_OCD control/data frame definition:

Frame Command: **0xB4 for write, 0xAC for read**		
15 14	13	12 7 6 5 4 3 2 1 0
[15:14]	FS	Frame Type select 10 : address/control frame (A-frame) 11: data frame (D-frame)
[13]	WEN	Write enable for A-frame, N/C for D-frame
[12]	REN	Read enable for A-frame, N/C for D-frame
[11:8]	-	Reserved
[7:0]	AD	Address for A-frame Data for D-frame

For write data through LGTSWD_OCC, firstly a D-frame should be sent for preparation data, then issue a A-frame to write the data to internal SFR or SRAM.

For read data out of inside SFR or SRAM, an A-frame should be sent firstly to set target address, then a read command is need to shift the data out.

Demo for a write procedure:

OPERATION	CMD	FRAME
Step1: Prepare data	0xB4	D-Frame with FS=10 and AD to target data
Step2: Set control and address	0xB4	A-Frame with WEN=1 and AD to target address

Demo for read procedure:

OPERATION	CMD	FRAME
Step1: Set control and address	0xB4	A-Frame with REN=1 and AD to target address
Step2: Read sequence	0xAC	Read a byte from SWD

Update History

V20151027	Birthday