`

faUnit / Module Description:	Installation and first step
Unit / Module Number:	SMT903
Document Issue Number:	2.0
Issue Date:	22/10/10
Original Author:	F.S

Application Note for SMT903

Sundance Multiprocessor Technology Ltd, Chiltern House, Waterside, Chesham, Bucks. HP5 1PS.

Revision History

Issue	Changes Made	Date	Initials
1.0	First release	09/12/09	FS
2.0	Added CCS package	22/10/10	FS

Table of Contents

Table of Figures

1 Introduction

This document describes the SMT903 demo package that comes with the WiMAX board.

The purpose of the demo is to check that the board is operating correctly and to give the user a kick-start for building his own application using the SMT903.

This application note decribes the SMT903 control for a system using one SMT351T, for any systems the demo are based on this application.

2 SMT903 systems

The demo is available for many systems, the following diagrams highlight some ways to control the SMT903:

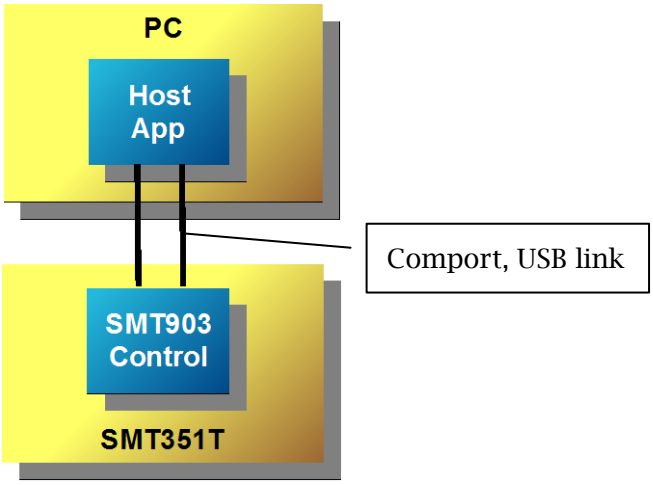

Figure 1 : PC - USB – FPGA TIM - SMT903

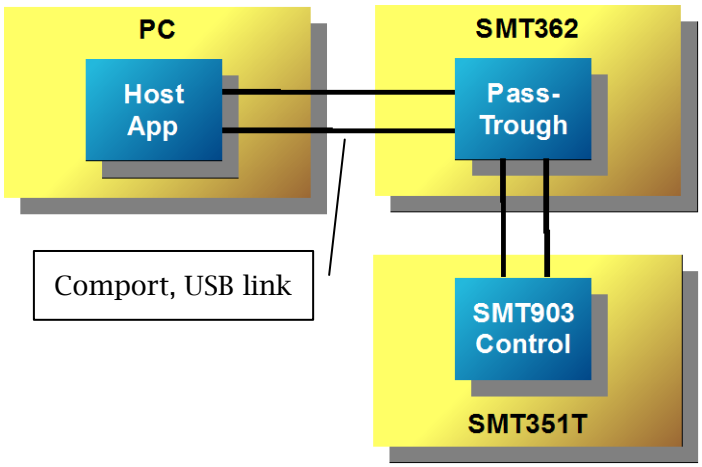

Figure 2 : PC - USB – DSP TIM – FPGA TIM - SMT903

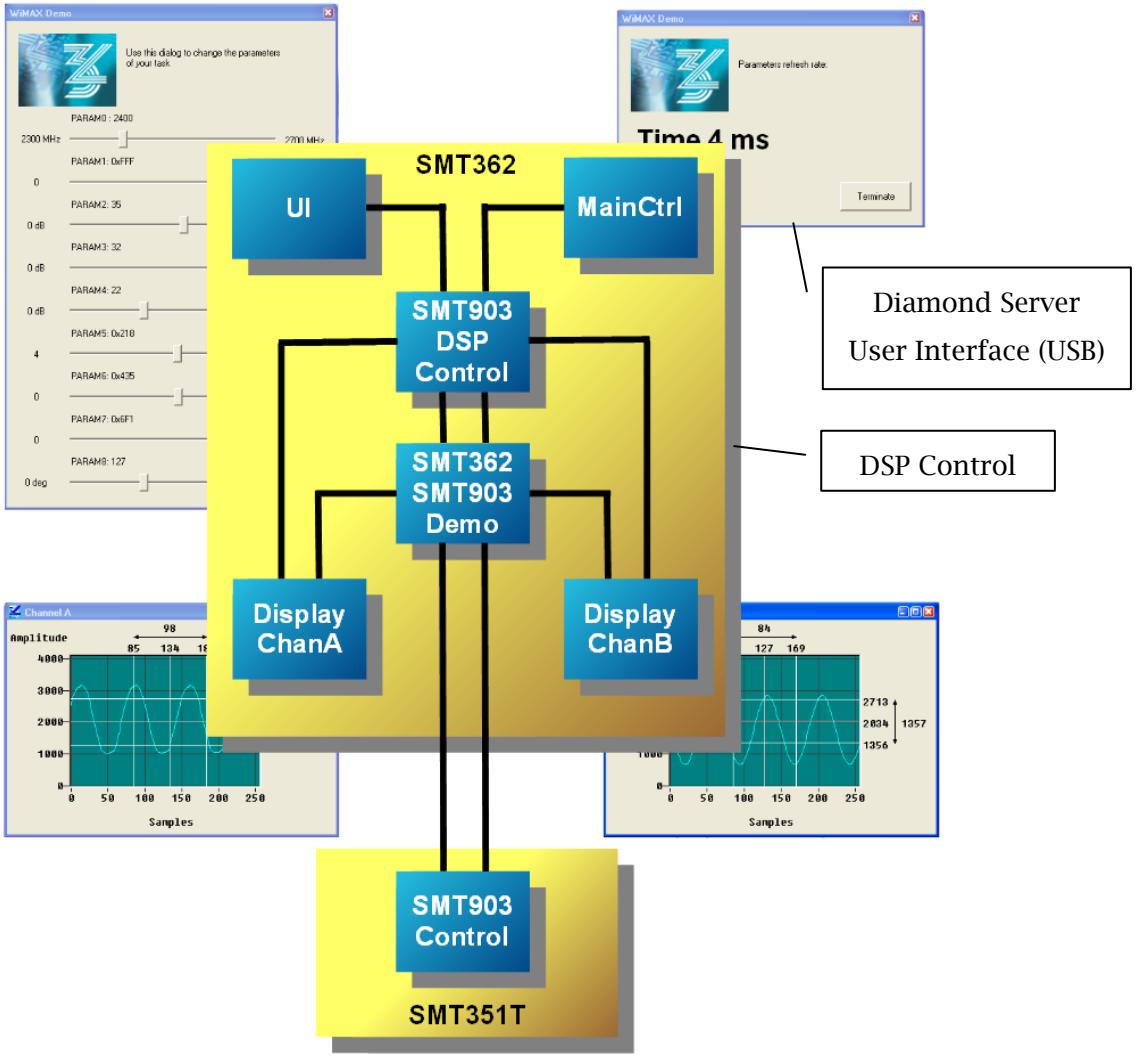

Figure 3 : PC (Diamond server) - USB - DSP TIM – FPGA TIM - SMT903

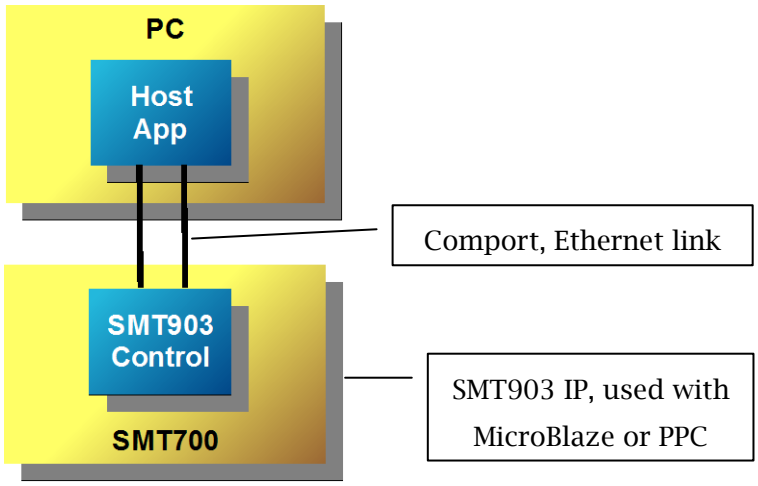

Figure 4 : PC - Ethernet - FPGA - SMT903

2.1 Run time environment

For this application note, the demo is running on the following system configuration:

(1) Standard PC running Windows XP 32-bit OS with USB connection to

(2) SMT111 populated with

(3) SMT351T-SX50 attached to

(4) SMT903 Sundance Local Bus mezzanine module.

The block diagram us shown in Figure 1.

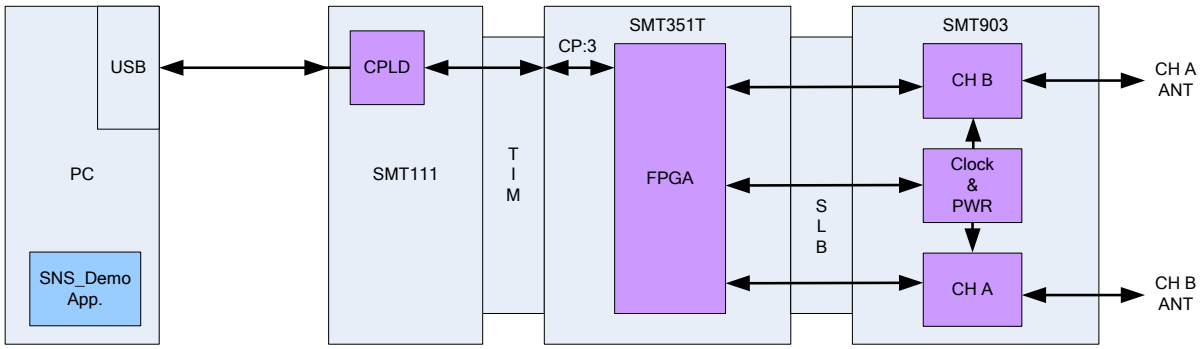

Figure 5 : Demo block diagram

To run the demo the PC needs to execute the ***SMT903_demo.exe*** on a Windows operating system. The SMT6048 package should be installed first. The FPGA bitstream ***SMT903_demo.app*** should be located in the same directory as the ***SMT903_demo.exe***. In order to see the sampled signals from the ADC ***show_res.m*** MATLAB script file is provided. The script should be located in the same directory too.

2.2 Development environment

The demo comes with all source codes, so you can start your own application from the working demo application. There are two developing environments to create the application:

(1) **Software development:** Visual studio C++ 6.0 was used, but any later versions can be used. The software contains two sources: ***SMT903_demo.cpp*** and ***SMT903.h*** .

(2) ***Firmware development:*** Diamond IDE 1.0 and Diamond 3.1.10 were used. Firmware was build using Xilinx ISE 10.1. The Diamond project contains one task ***SMT903_demo*** comprising 7 different VHDL files. The task is connected with dedicated signals via the SLB to the SMT903 from one side and to the PC via comport 3. The full description of the comport registers is given in chapter XX. Four data ports, two inputs and two outputs, for signal samples are provided for BB_TX and BB_RX of both channel A and channel B. The following is the list of all the task ports:

- Ch0_in => control
- Ch0_out => status

- Ch1_in => RF A BB complex input (IQ): CH_A_BB_TX
- Ch1_out => RF A BB complex output (IQ) : CH_A_BB_RX

- Ch2_in => RF B BB complex input (IQ): CH_B_BB_TX
- Ch2_out => RF B BB complex output (IQ): CH_B_BB_RX

3 Firmware Specification

The firmware communicates with the host using COMPORT_3 on 32-bit control words with the following format: 4 bit command; 12 bit address (4K space) and 16 bit data.

The supported commands are:

- 0 – loopback
- 1 – Write
- 2 – Read

3.1 SMT903 demo Registers Address map

The firmware supports the following registers

0x00-0x0F: General SMT903 programming

- **0x00:** Common
- **0x01:** Clock_Select
- **0x02:** CH A
- **0x03:** CH B
- **0x04:** VCXO_DAC
- **0x05-0xD:** PLL
- **0x0E:** Base Module
- **0x0F:** Firmware ID

0x10-0x17: Data path registers

0x17-0x6F: Reserved

0x70-0x7F: Reserved for debug

0x80-0x9F: TRX A registers (32 x 10 bit registers)

0xA0-0xBF: TRX B Registers (32 x 10 bit registers)

0xC0-0xDF: MxFE A Registers (32 x 16 bit register space mapped to 23 x 8 bit space in the IC)

0xE0-0xFF: MxFE B Registers (32 x 16 bit registers space mapped to 23 x 8 bit space in the IC)

0x100-0xFFF: Reserved

Registers description

0x00: Common Register

	Common Register – 0x00							
	Bit 7	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**
Byte 0	GPIO1_MODE		GPIO1_MODE		GPIO2	GPIO1	LED D2	LED D1
Default	'00'		'00'		'0'	'0'	'0'	'0'
Byte 1	RESETn	Reserved			SHUTDOWNn	CLOCK_OUT_OE	ADC_LO_PWR	TRX_RXHP
Default	'0'	'000'			'0'	'0'	'0'	'0'

LEDs bits are connected to the SMT903 LEDs.

GPIO bits functioning depend on the GPIO mode as follows:

- '00' – GPIO disabled
- '01' – GPIO is input pin
- '10' – GPIO is output pin
- '11' – GPIO in debug mode (reserved).

In input mode reading the GPIO bit reflect the state of the pin and in output mode what is written in the bit is driven into the pin.

All other bits drive control signals in the SMT903 (see SMT903 specification document for details)

0x01: Clock Select Register

	Clock Select Register – 0x01							
	Bit 7	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**
Byte 0	SEL_CLKOUT_2	SEL_CLKOUT_1	SEL_SYNT_2	SEL_SYNT_1	SEL_CLKIN_2	SEL_CLKIN_1	SEL_FPGA_2	SEL_FPGA_1
Default	'0'	'0'	'0'	'0'	'1'	'1'	'0'	'0'
Byte 1	Reserved					SEL_DAC	SEL_ADC	SEL_TRX
Default	'00000'					'0'	'0'	'0'

Bits that drive the clocking control signals in the SMT903. (see SMT903 specification document for details)

0x02: Channel A Register

	Channel A Register – 0x02							
	Bit 7	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**
Byte 0	Reserved			ANT_SW_A	TRX_TX_EN_A	TRX_RX_EN_A	TxPwrDwn_A	RxPwrDwn_A
Default	'000'			'0'	'0'	'0'	'0'	'0'
Byte 1	Reserved							
Default	'00000000'							

Bits that drive channel A control signals in the SMT903. (see SMT903 specification document for details)

0x03: Channel B Register

	Channel B Register – 0x03							
	Bit 7	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**
Byte 0	Reserved			ANT_SW_B	TRX_TX_EN_B	TRX_RX_EN_B	TxPwrDwn_B	RxPwrDwn_B
Default	'000'			'0'	'0'	'0'	'0'	'0'
Byte 1	Reserved							
Default	'00000000'							

Bits that drive channel A control signals in the SMT903. (see SMT903 specification document for details)

0x04: VCXO_DAC Register

	VCXO_DAC Register – 0x04							
	Bit 7	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**
Byte 0	DAC Value [7-0]							
Default	'00000000'							
Byte 1	Reserved			Power Down Mode		DAC Value [11-8]		
Default	'00'			'00'		'0110'		

This register, when written, activate a serial programming of the VCXO DAC (AD5621). The DAC is programmed using 16 bit SPI programming cycle with: 2 mode bit + 12 data bits + 2 don't care bits. The data is stored in an internal register and during read operation that value is read.

0x05-0x0D: PLL Registers

	PLL Register – 0x05 to 0x0C							
	Bit 7	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**
Byte 0	Data[0-7]/Data[16-23]/…/Data[112-119]							
Default	See Data Below							
Byte 1	Data[8-15]/Data[24-31]/…/Data[120-127]							
Default	See Data Below							

This register bank store the data to be transmitted to the clocking PLL (ICS307).

The PLL is programmed via SPI cycle with 132 bits. The default setup during reset is:

0x05 - 0x5FF2

0x06 - 0xFF05

0x07 - 0xFFFF

0x08 - 0xFFFF

0x09 - 0xFFFF

0x0A - 0x3BFF

0x0B - 0xFFEE

0x0C - 0x1FFD

0x0D - 0x8003

Writing to the last Register 0x0D bit 15, PLL_prog, initiate a full 132 bits SPI programming cycle. For bits content see ICS307 data sheet. This bit is cleared automatically.

	PLL Register – 0x0D							
	Bit 7	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**
Byte 0	Reserved				Data[131]	Data[130]	Data[129]	Data[128]
Default	‘0’				‘0’	‘0’	‘1’	‘1’
Byte 1	PLL_Prog	Reserved						
Default	‘0’	‘0000000’						

0x0E: Base Module (SMT351T)

	Base Module – 0x0E							
	Bit 7	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**
Byte 0	GPIO				LEDs			
Default	'0000'				'0000'			
Byte 1	Reserved							
Default	'00000000'							

This register is for the base module GPIO and LEDs.

0x0F: Firmware ID

	Firmware ID – 0x0F							
	Bit 7	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**
Byte 0	Version				Revsion			
Default	'The version'				'The revision'			
Byte 1	Board ID							
Default	'0x93'							

This register is read-only. Write data are ignored. The Board ID indicates the FPGA firmware is for SMT903. The version and revision are checked by the software.

0x10: Data Path Control Register

	Data Path – 0x10							
	Bit 7	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**
Byte 0			DP_SINK_RX_A	MEM_SINK_RX_A			SRC_TX_A	
Default			'0'	'0'			'00'	
Byte 1			DP_SINK_RX_B	MEM_SINK_RX_B			SRC_TX_B	
Default			'0'	'0'			'00'	

This register controls the data path. Low byte connrols A channel and High byte control B channel. The controls are identical. The two LSB bits SRC_TX_% control the flow of data to the DAC. The following options are given:

00 – Internal samples (Reserved)

01 – Samples from Memory. (See memory registers below)

10 – Samples from Data Port.

11 – Loopback from the other channel. Note: the other channel should be in RX mode.

The RX samples can be directed to several sources in parallel, currently to sinks are supported:

8K samples memory and data ports:

- MEM_SINK_RX_A /B – set to one this bit trigger a collection of one block of memory. in read always return zero
- DP_SINK_RX_A /B – when high samples are directed to the output data port.

0x11: CP SIDE ADDRESS Register

	CP Side Address – 0x11 (Address)							
	Bit 7	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**
Byte 0	ADDR low							
Default	'0x00'							
Byte 1	Reserved			ADDR HIGH				
Default	'000'			'00000'				

This register index the address in internal memory. It is used both to read from and to write to for both TX and RX samples as well as for both the I and the Q samples for both channels.

0x12: CH A TX ADDRESS COUNT Register

	CH A Address Count – 0x12 (Address)							
	Bit 7	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**
Byte 0	ADDR low							
Default	'0x00'							
Byte 1	Reserved			ADDR HIGH				
Default	'000'			'00000'				

0x13: CH B TX ADDRESS COUNT Register

	CH B Address Count – 0x13 (Address)							
	Bit 7	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**
Byte 0	ADDR low							
Default	'0x00'							
Byte 1	Reserved			ADDR HIGH				
Default	'000'			'00000'				

Those registers determine the number of samples that will be cyclically transferred from the memory to the DAC when TX mode is enabled and memory source is selected.

0x14-17: **MEM_DATA** Registers

	Memory Entry – One sample (0x14-0x17)							
	Bit 7	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**
Byte 0	Ch. A : I low							
Byte 1	Reserved				CH A I High			
Byte 2	Ch. A : Q low							
Byte 3	Reserved				CH A Q High			
Byte 4	Ch. B : I low							
Byte 5	Reserved				CH B I High			
Byte 6	Ch. B : Q low							
Byte 7	Reserved				CH B Q High			

This 4 x 12-bit registers are one sample store contains 4 x 12-bit for both channels A and B, each channel with samples I and Q. Writing to a register writes the data to the memory element that register 0x11 is pointing to. Reading from a register read from memory element that register 0x11 is pointing to. Read (RX) and write (TX) memories are two different independent memories. During write cycle, each write to register initiate a write cycle to the memory address that register 0x11 is pointing to. To fill up the memory for all two channels one must step Register 0x11 from zero to 8K and in each step write the four samples value to each register. In order to play this to the DAC, one must also set registers 0x12 and 0x13 to 8K, select the data source to be memory and set the channels to TX mode.

0x80-0x9F : TRX A Registers

	TRX_A Registers – 0x40 – 0x5F							
	Bit 7	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**
Byte 0	Data[7-0]							
Default	MAX8237 Default							
Byte 1	Reserved						Data[9-8]	
Default	'000000'						MAX8237 Default	

This address space is a shadow of the registers in channel A MAX8237. A write to this space will trigger an SPI write cycle to the MAX8237. A read will initiate a read cycle and the value that read back is send back to the host via the Comport.

0xA0-0xBF : TRX B Registers

	TRX_B Registers – 0x60 – 0x7F							
	Bit 7	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**
Byte 0	Data[7-0]							
Default	MAX8237 Default							
Byte 1	Reserved						Data[9-8]	
Default	'000000'						MAX8237 Default	

This address space is a shadow of the registers in channel B MAX8237. A write to this space will trigger an SPI write cycle to the MAX8237. A read will initiate a read cycle and the value that read back is send back to the host via the Comport.

0xC0-0xDF : 64 Regs (MxFE A Registers)

	MxFE_A Registers – 0x80 – 0xBF							
	Bit 7	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**
Byte 0	Data[7-0]							
Default	AD9863 Default							
Byte 1	Reserved							
Default	'000000'							

This address space is a shadow of the registers in channel A AD9863. A write to this space will trigger a write to the AD9863. A read will initiate a read cycle and the value that read back is send back to the host via the Comport. The serial sequence includes one Read/Write bit determined by the type of instruction plus one Byte/Word (2 Bytes) bit that is always set to Word (2 Bytes) plus the 6 address bits follows by 16 data bits. The AD9863 registers are byte oriented but each two bytes are fetched together in firmware.

0xE0-0xFF : 64 Regs (MxFE B Registers)

	MxFE_B Registers – 0xC0 – 0xFF							
	Bit 7	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**
Byte 0	Data[7-0]							
Default	AD9863 Default							
Byte 1	Reserved							
Default	'000000'							

This address space is a shadow of the registers in channel B AD9863. A write to this space will trigger a write to the AD9863. A read will initiate a read cycle and the value that read back is send back to the host via the Comport. The serial sequence includes one Read/Write bit determined by the type of instruction plus one Byte/Word (2 Bytes) bit that is always set to Word (2 Bytes) plus the 6 address bits follows by 16 data bits. The AD9863 registers are byte oriented but each two bytes are fetched together in firmware.

4 The software

The software uses a GUI were you can select tests and actions by typing your input command using the keyboard, and to see the status on the terminal.

The software initializes the registers of the firmware and performs one of the following tests:

1. comport loopback test – to check the USB communication
2. LEDs test – to check the LEDs on the board
3. Firmware version test – to verify that you use the correct version of firmware
4. Functional setting – functional setting enables you to check that the SMT903 is functionally working with real signals.

4.1 Functional setting

The functional setting enables you to set the channels to transmit and receive modes and to change major parameters of the channels. The MxFE set to work with 40Msps clocked from the internal reference.

When entering functional setting you are requested to set the SMT903 module state. Each channel can be set to transmit (TX) or receive (RX) mode. You can also set the channel to Idle. Since operated channel (especially TX) consume significant power and spread heat, it is recommended not to activate channels you do not use. In any case, take care for proper ventilation condition.

Note: The demo setup is basic, note that the MAX2837 is a complex RFIC with 320 setup bits. To get optimal performance you should use the calibration modes and set all tuning bits. In the demo the RFIC setup is basic hence in some cases it is less then optimal. In order to get optimal performance, set all tuning registers on both the RFIC and MxFE. For more info consult MAXIM and Analog Device datasheets and application notes and the company's technical support.

4.2 Running the software (suggested test procedure)

Before you start check list:

- Make sure you have SMT903_demo.exe and SMT903_demo.app in the same directory.
- Make sure SMT6048 installed in your computer.
- Make sure the USB cable is connected to the SMT111
- Make sure the power is connected to the SMT111
- Make sure proper ventilation is applied.

1. Start SMT903 demo
2. Run test 1,2 and 3 and look that they pass
3. select test 4 - Functional setting

Checking the transmitters

4. Connect a spectrum analyzer or scope to the ANT connector of the channel you would like to test.
5. Select 'ti' for CH A (or 'it' for channel B)

Analyze the generated signal. By default you should see a 10dBm sine wave at 2500 MHz. By default maximum signal strength is generated and the RFIC and PA are saturated. You can change signal strength and come out of saturation by changing RFIC TX VGA (parameter #2) and BB DAC output voltage (parameter #1). You can change the frequency by changing parameter #0.

6. Type 'h' to see the command options and 'I' for general info..

Control Parameters Handling

Currently 9 parameters are controlled by the demo:

- Frequency – The RFIC frequency
- TX BB – The DAC value. Same for I and Q
- TX VGA – The RFIC VGA on the transmitter (0 dB = Maximum gain)
- RX LNA – The RFIC LNA on the receiver (0 dB = = Maximum gain)
- RX VGA – The RFIC LNA on the receiver (0 dB = = Maximum gain)
- TX BB modulator: The demo generate a simple modulated signal by loading the internal memory with two sine waves (I and Q) four parameters are controllable:
    - TX BB SINE MOD FREQ
    - TX BB SINE MOD AMP
    - TX BB SINE MOD OFFSET
    - TX BB SINE MOD PHASE

Each parameter can be easily changed by the following keys:

- Fine increment: '+', '↑'
- Fine decrement: '-', '↓'
- Rough increment: 'PgUp', '←'
- Rough decrement: 'PgDn', '→'
- Default setting: 'd', "Del', 'Ins'
- Minimum setting: 'Home'
- Maximum setting: 'End'
- Explicit setting: s XXXX where XXXX is the value

You can change the current parameter by selecting numbers 0-8

7. Select '2' then multiple presses on '↑' to get out of saturation. See when the signals strength start to decrease then continue to reduce 5 dB.
8. Select '5' to check the IQ modulator. By default the modulator generate AM modulation with over 200% modulation index. Play with the parameters (5-8) and see the changes. If you set the phase to 90 degree you will have SSB modulation or a shifted tune on the spectrum you will still see the image and the LO by tuning the phase the IQ imbalance and the DC offset you can reduce those spectral line to minimum.
9. You now finished basic check of the channel A transmitter repeat those steps to check the second transmitter.

Checking the receivers

10. Connect a 2.3 - 2.7 GHz signal generator to the ANT connector of the channel you like to test. Set the generator to 2501MHz and the power to -20dBm.
11. Select 'ri' for CH A (or 'ir' for channel B)
12. Select 't' to trigger recording of samples from the ADC. A file result.txt is created on the current directory and you can analyze the data with MATLAB.
13. Open MATLAB and set the current directory to the directory of your software.
14. Run on MATLAB show_res.m
15. A figure is opened with two axes one for channel A and the other for channel B look at the relevant channel the red line is the I and the green is the Q signal. By default maximum gain is set on the RFIC so both RFIC and ADC can be saturated. The ADC is AC coupled so you will see either constant DC signal around middle reading of the DAC or switching between underflow and overflow values.
16. Change RX chain gain by controlling RFIC RX VGA (parameter 4) and RFIC RX LNA (parameter 3). Reduce the chain gain until a 1 MHz sine wave is within the dynamic range of the ADC. As a rule of thumb, input power + LNA attenuation + VGA attenuation in should be around 80. It is preferred to avoid using high RX VGA gain (gain values around zero), prefer having more gain on the RF section.
17. Remember each time you change parameter and would like to see the results you need to select 't' to recorded the samples and transfer the to the disk then run the script on the MATLAB to read the data from disk and redraw the figure.
18. You can no change some parameters on the RF generator like power and frequency and analyze the results.
19. Repeat the test on the other channel.
20. Congratulation you have no fully tested the SMT903 board.

5 Code Composer Studio package

You can use CCS instead of 3L diamond application. In that case you don't have an application to load your FPGA bitstreams and DSPs applications. The CCS examples are exactly like the diamonds one (based on one SMT362 and one SMT351T+SMT903), but you have to load the FPGA configuration in the TIMs flash memories and use the JTAG to load the DSP applications.

5.1 FPAG configuration

The FPGA configuration has to be done the first time, when all the bitstreams are store in flash you don't need to do the following steps again to run you application, but if you have done some firmware modification, you have to.

You have two solutions to load the SMT351T bitstream.
First you can directly put your SMT351T on the TIM site one of your carrier board to get the host to SMT351T CP3 link and use the SMT6002 to store your bitstream in flash.

Otherwise you can let your system with the SMT362 TIM site 1 and SMT351T+SMT903 TIM site 2; you will need to load a pass-through application in the SMT362 to provide the host link to the SMT351T for the SMT6002.
To use the SMT6001 you first have to configure your CCS setup for the SMT362 (SMT362 User Manuel, section 6.1> Code Composer Studio). In CCStudio: Parallel Debug Manger, select the option "connect system on startup".
Open the SMT6001 and add the SMT362_CP3_CP0_passthrough.app in the User blocks, select the option bootable, finally select "commit" and "ok" to exit the application when the Flash is programmed.

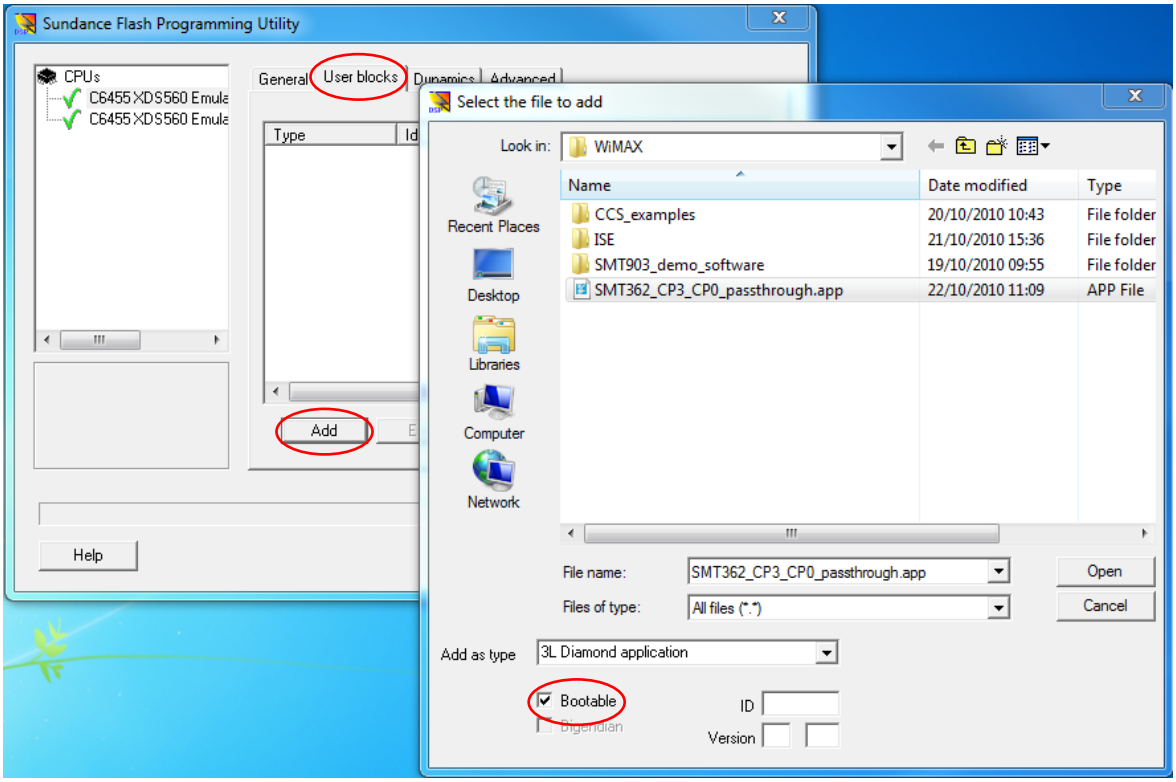

Figure 6 : SMT6001 user blocks

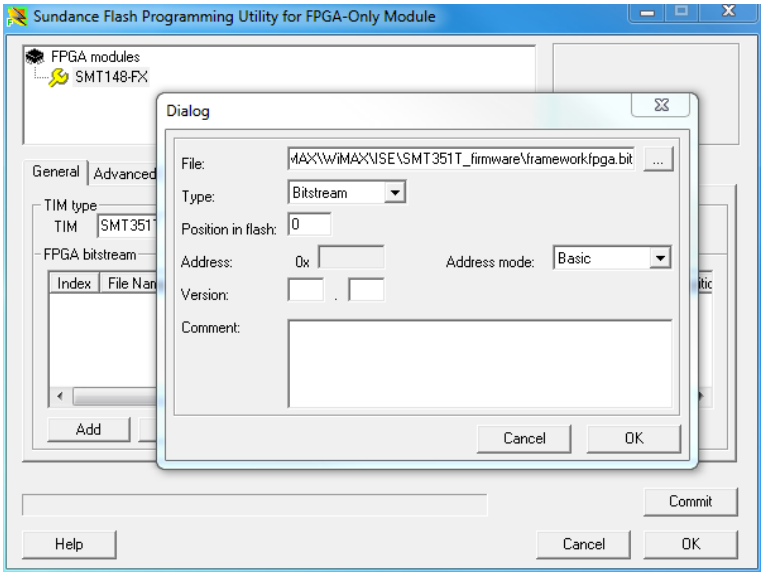

Figure 7 : SMT6002

The DIP switches section in the SMT351T user guide explains the different TIM configurations. To use the SMT6002 set SW1[4:OFF 3:ON 2:OFF 1:OFF] reset your board and open the SMT6002. Add the SMT351T bitstream in flash position 0 as in the above picture.

For SMT351T-SX50 :

..\SMT903_CCS_package\ISE\SMT351T_firmware\SX50\ frameworkfpga.bit

For SMT351T-SX95 :

..\SMT903_CCS_package\ISE\SMT351T_firmware\SX95\ frameworkfpga.bit

When the flash is programmed change the switch position to SW1[4:OFF 3:OFF 2:OFF 1:ON].

If the SMT351 was TIM site 1 of your carrier board move it to TIM site 2, plug the SMT903 on it, and add the SMT362 TIM site 1.

If you used the SMT362 pass-through you need to remove the application from the SMT6001 user blocks.

The SMT351+SMT903 are now ready and will be configured at power up or after a reset.

Before running the SMT903 applications check that you have the standard SMT362 firmware with the SMT6001 (select “standard” button for both bootloader and FPGA bitsream) and that you have nothing in the user blocks section.

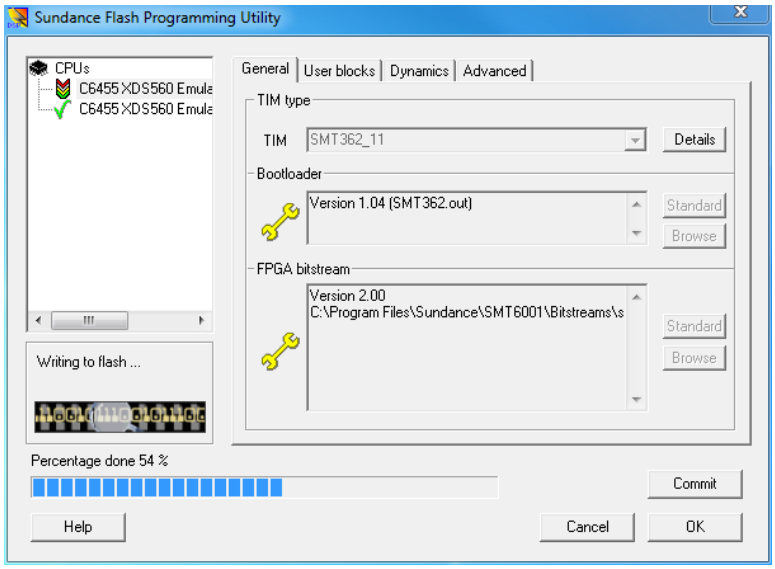

Figure 8 : SMT6001

If you modified the SMT362 firmware you can use the SMT6001 and browse the FPGA bitstream to your custom one, like:

..\SMT903_CCS_package\ISE\SMT362_firmware\frameworkfpga.bit

5.2 Applications

The following example are based on comport connections; use the SMT6400 DSP Module Package for the registers information.

5.2.1 Host application

As in the section 4.2 you can control the registers of the SMT351T+SMT903 firmware from the host the only difference is that the SMT903_host.exe won't load a diamond application and that you have to run the SMT362 DSPs application with CCS.

Open CCS, load and run the SMT903_host.out program for both DSPs, this will send/receive the data from/to:

Host⇔SMT362_DSPA⇔SMT362_DSPB⇔SMT351T_FPGA

Launch the SMT903_demo.exe application from the CCS_package and you can now control the SMT903, you can do the steps describe in section 4.2.

5.2.2 DSP application

For a control of the SMT351T+SMT903 from the SMT362 you have the SMT903_dsp project example. Open this one and load the SMT903_dsp.out program for the DSPB of the SMT362.

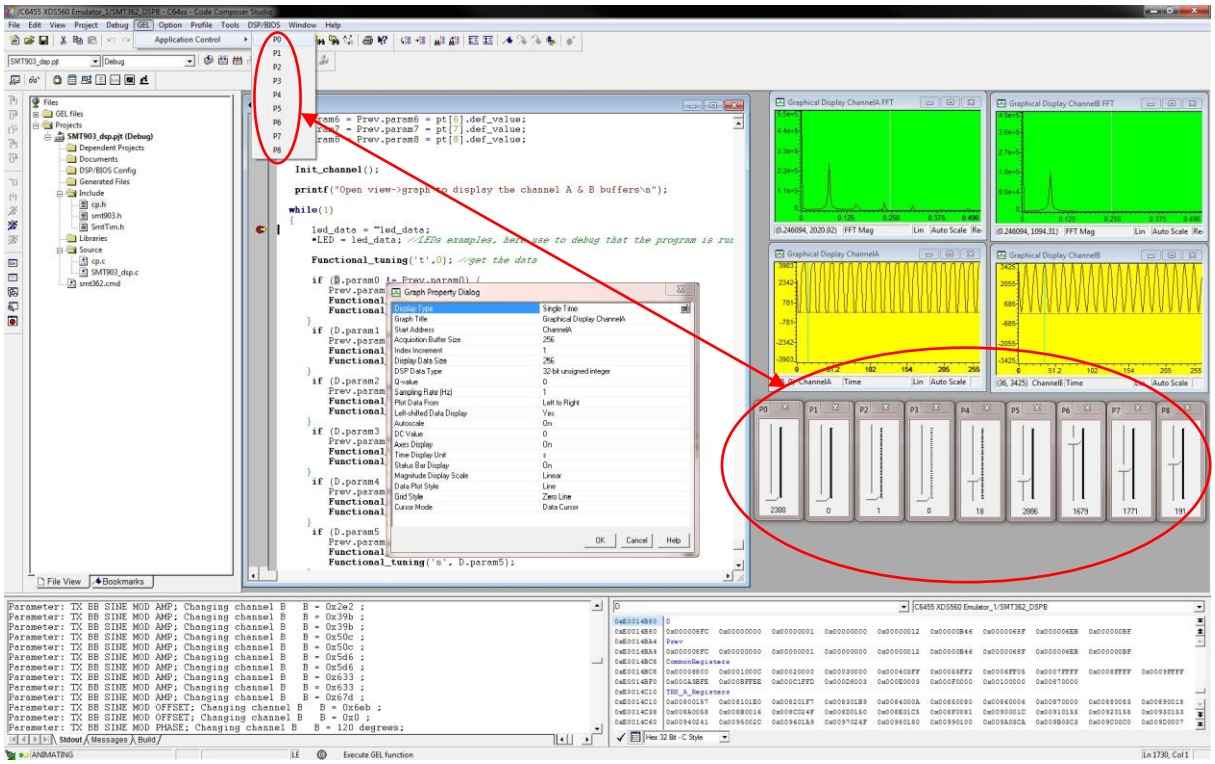

Figure 9 : SMT903_dsp CCS project

This example is exactly like the host application but from the SMT362 DSPB, it just set the channel A as a receiver and channel B as a transmitter.

The parameters can be changed with the sliders as display in the above picture. You need to load the parameters.gel file to get the "Application Control" and "P0 to P1" menu.

For a quick debug of your signals you can open some Graphical display from:

View->Graph->Time/Frequency.

You can change the display type to FFT if you want. You have to configure the start address to the buffer “ChannelA” or “ChannelB” and set the acquisition buffer size and display data size to 256.

Add a breakpoint after the while(1), and select Debug->animate.

The channels buffers are refreshed automatically by the code and the breakpoint provides the Graphical display refreshment.

As example you can now change the slider parameters to get the registers value that you might need for your project. You should see the receiver and transmitter sin wave modified in real time.