HISTORY OF REVISION

Rev.	Date	Author	Reason for change
A	5/29/2002		Initial document
B	7/10/2002		Fast Camp and general purposes frames
C	7/25/2002		Chapter 4.3.7 serial link activity Chapter 4.3.9.1 test speaker's path Chapter 4.3.19.CINT Chapter 4.3.5.LCD Interfaces Chapter 4.3.10.Melody Chip Interface Chapter 4.3.20.SPI Bus
D	10/30/2002		Chapter 3 Way to proceed test and tuning during manufacturing stage => general information on embedded software added Chapter 4.3.5 LCD interfaces => contrast tuning frame added Chapter 4.3.7 OUI key => STMA frame described Chapter 4.3.7 Audio => STMA frame's parameters described, frames to test speaker functionally added Chapter 4.3.19 Generic IT Test => parameters for CINT & OUI key added Added : Chapter 4.3.23.5 Check connection to network Added : Chapter 4.3.24 VBAT : to read VBAT and auto-adjust it Added : Chapter 4.3.25 Reset module Added : Chapter 4.4.3 Reading module code Added : Chapter 4.4.4 Client software version Added : Chapter 4.4.5 Reading CIE code Chapter 4.4.9 Hardware configuration => STMA frame's parameters for LCD & contrast described, warning added Chapter 5 Test sequence traceability => 32 bytes instead of 10 Added : Chapter 6 Way to use the DllTel.dll file for multidownloading

TABLE OF CONTENTS

1. OVERVIEW

1.1 OBJECT OF THIS DOCUMENT

This document represents the mechanical and functional test specification for all products built with the MO130 GSM/GPRS module provided by WONU.

It aims at describing:

- all functionalities embedded on the software used to proceed the test in manufacturing plants.
- Hardware requirements for manufacture test tool
- It also gives some examples to help anyone to implement and create the software tests tools.

1.2 REFERENCE DOCUMENTS

SCT TMO MASV2 SPEC 14 – MO130 Module for voice centric phone – Preliminary Specification

SCT TMO MASV2 SPEC 57 – MO130 Module - Application Note

SCT TMO LOG SPEC 0414– Audio parameters setup

SCT TMO MOD SPEC 0053 A - MO130 Setting Tool User Manual

1.3 TERMINOLOGY

ADC	Analog to Digital Converter
AT command	Attention command
CTS	Clear To Send
DAC	Digital to Analog Converter
DAI	Digital Audio Interface
DCS	Digital Communications System
DSR	Data Set Ready
DTR	Data Terminal Ready
GSM	Global System for Mobile communication
GPRS	General Packet Radio Services

I/O	Input / Output
IRDA	Infra Red Device Application
JTAG	Joint Test Action Group
LCD	Liquid Crystal Display
LED	Light Emitting Diode
MMI	Man Machine interface
MO130	WONU's GSM/GPRS module with full mobile functionalities
PCS	Personal Communication System
RAM	Random Access Memory
RF	Radio Frequency
RI	Ring Indication
RTS	Ready To Send
RX	Reception
SIM	Subscriber Identification Module
SRAM	Static Random Access Memory
STMA	Système de Test Matériel Asap
TBC	To Be Confirmed
TBD	To Be Defined
TX	Transmission
UART	Universal Asynchronous Receiver and Transmitter

2. DESIGN REQUIREMENT

This part describes the design requirements related to the test tool which must be developed and installed on customer manufacture facilities in order to perform all tests on final integrated mobile product. All software in charge of managing the necessary test sequence is embedded on the Mo130 GSM/GPRS module

2.1 HARDWARE INTERFACE

As specified in the application note of MO130 GSM/GPRS module (ref SCT U37 MASV2 SPEC 57), build the final product requires a MMI board to perform all the functionalities (Audio, Sim card, Accessories, Display,…). **As it is, the design of this MMI board must provide an access to following signals when the product will be completely integrated :**

Serial link

TXD1	Output	UART transmit 1
RXD1	Input	UART receive 1

Power supply

VBAT	Input	+3.6V battery power supply
GND	Ground	

Charger

CHARGEUR	Input	Charge

Audio

BFTXP	Input	Differential input from microphone
BFTXN	Input	Differential input from microphone
BFRXP	Output	Differential output to earphone
BFRXN	Output	Differential output to earphone

DAI

DAIRST	Input	DAI reset
DAIOUT	Output	DAI output data
DAIIN	Input	DAI input data
DAICLK	Input	DAI clock

To allow an access to these signals from the outside, you could use the connector or reserve some electrical test pins inside the MMI board. In any case to control these signal by PC, it will also be necessary to build the mechanic and electronic test tool.

TXD1and RXD1 come from the V24 interface of the MO130 GSM/GPRS module; this link offers an access to communicate with the final product. In case the design prescribes these only two signals for communicating, it must also specify a short-circuit between RTS and CTS signals (provided too by the V24 interface).For more information on, refer to “V24 Interface” in paragraph 4.3.12.

Warning, to use any signal coming from the V24 interface, some adaptation are necessary to convert the +3V signals of the MO130 GSM/GPRS module to a level compatible with the serial interface of the PC. Refer to SCT U37 MASV2 SPEC 57“Application Note” to get more information.

The DAI interface is assigned to perform qualification and all audio approval test. During the approval stage, the product will be in final design and it must give access facilities to the DAI interface. That is the reason why it is strongly recommended to design the additional electronic board with some electrical test pins for this interface.

To be complete, if the final product doesn’t use any accessory such as headset combined with a microphone or if it doesn’t require any audio tuning, it won’t be mandatory to have an access to the audio signals (BFTXP, BFTXN, BFRXP, BFRXN).

3. WAY TO PROCEED TEST AND TUNING DURING MANUFACTURE STAGE

All MO130 GSM/GPRS modules provided by WONU runs a specific software for the manufacture stage; this document refers to this software as “embedded software”.

The embedded software was created to perform the mobile tests:

- It allows to have only one software by one kind of test material interface (sometimes one s/w for several kind of interface), and to shift the customer s/w downloading at the end of line production.

- Some STMA frames and some specific functions can be developed (independent of customer s/w) for production needs.

- It has a small size: only material aspects are tested in production, so all s/w processes concerning data functionnalities, customers menu, etc., can be deleted. It allows to reduce downloading time and also the global test.

- It doesn’t keep information concerning tuning fabrication in operators fields (ex: bis file, communication time,…).

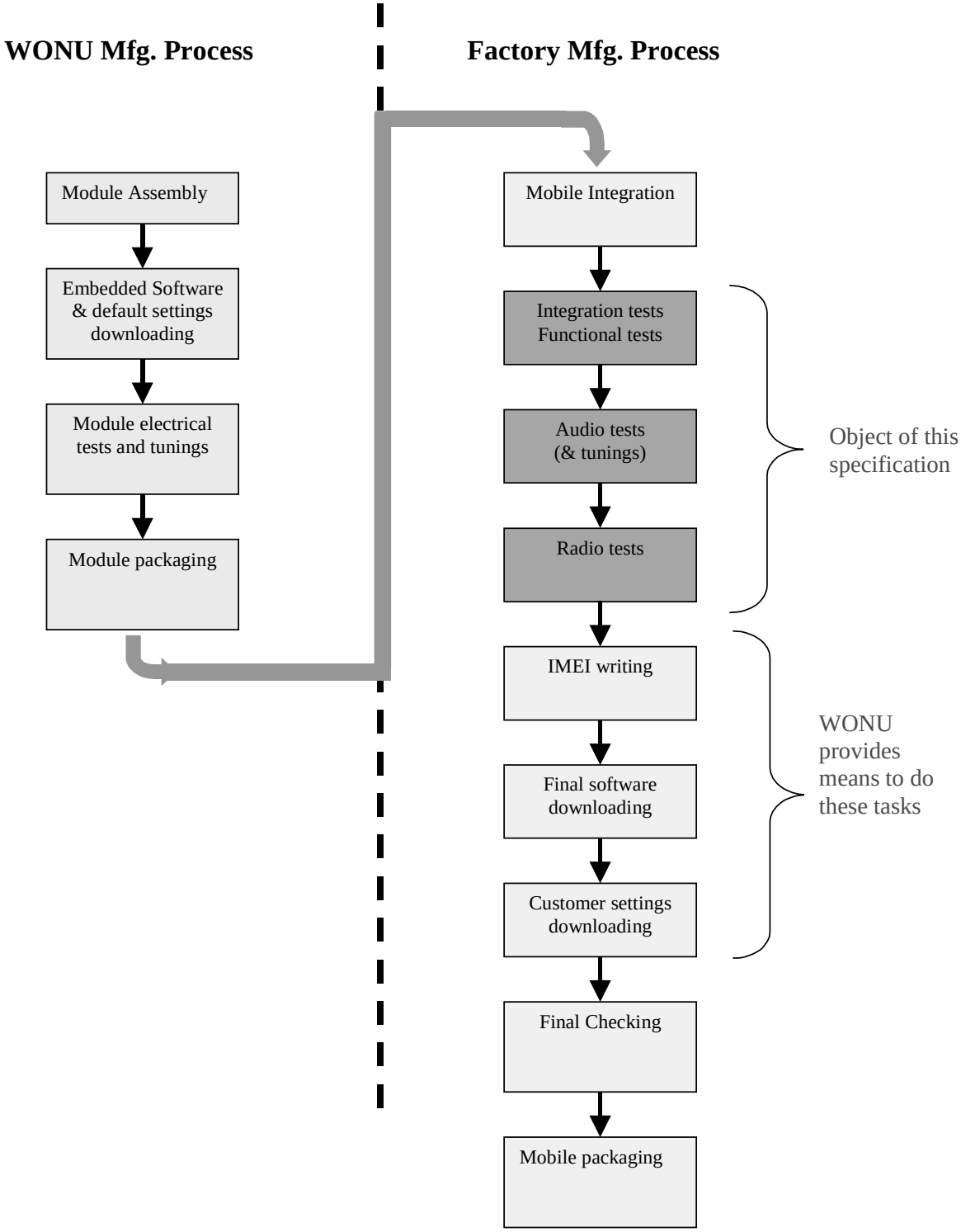
Before delivery the MO130 GSM/GPRS module, WONU performs some tests and fix default setting values on each product:

- Electrical tests; to check the electrical functionalities of the MO130 GSM/GPRS module;
- RF settings and RF tests;
- Default settings for software management.

To build a final product based on MO130 GSM/GPRS module, it will be necessary to update an integration process for running some tests on customer’s manufacture facilities, adjusting some parameters and downloading a final customers’ software with its settings.

3.1 MANUFACTURE PROCESS

Below this schematic represents on the left side, the test proceeding built by WONU which is used to test all MO130 GSM/GPRS module before delivery. On the right part, it gives some ideas on how could be the process for the integration test.



3.2 STMA FRAMES

3.2.1.1 STMA frames principle

As specified above, the part or all software in charge of managing the mobile phone is embedded in the MO130 module. This software will include some services to proceed the tests as STMA frame to control all complex functionality designed by WONU. STMA frame is a communication principle which has been designed and integrated by WONU. This principle allows a total communication control between the GSM device and any command terminal such as a PC.

This document describes the set of STMA frames assigned to the required tests. These STMA frames must be used successively by a global test software created by the customer according to its own test sequence.

3.2.1.2 Overview about STMA frames

MO130 GSM/GPRS module is driven thanks to STMA frames sent over a serial link for manufacture test purpose.

Regular STMA frame is depicted below :

STX	LENGTH	WONU FLAG	STMA FLAG	OPERATION	USEFUL CONTENT	CHECKSUM	ETX

Here is the meaning of each field :

STX : Flag indicates the beginning of a STMA frame. Its value is 0x02

LENGTH : This value indicates the number of bytes of the STMA frame except STX, CHECKSUM, ETX fields (length field is not included).

WONU FLAG : This value shall be set to 0xd2

STMA FLAG : This value shall be set to **0x05**

OPERATION : It indicates whether the STMA frame is a command or a response. In case it is a command its value is **0x00** otherwise its value is **0x01**.

USEFUL CONTENT : It includes both type of command and parameters relative to this command

CHECKSUM : It is responsible for checking STMA frame is valid. It is calculated as described below.

CHECKSUM = STX ⊕ LENGTH ⊕ WONU FLAG ⊕ STMA FLAG ⊕ OPERATION CODE ⊕ each byte of useful content (⊕ = bitwise XOR)

ETX : Flag which indicates the end of a STMA frame. Its value is **0x03**

There are two kinds of STMA frames :

Command STMA frame

Command STMA frames are conform to the below description and are necessary to drive MO130 GSM/GPRS module.

Response STMA frame

If the command is performed successfully a regular response is returned by MO130 GSM/GPRS module, otherwise an error frame is received. ERROR FRAME is depicted below :

0x02	0x05	0xd2	0x05	0x01	0x45 | Code	CHECKSUM	0x03

The field Code is OPERATION CODE of the erroneous frame. It is NULL if type of frame is unknown.

This frame is returned in case the operation is not supported / forbidden or the format of the frame is wrong

4. DESCRIPTION OF TESTS AND TUNING

This section gives a brief description of the functionalities provided by the MO130 GSM/GPRS module and the best way to proceed the overall industrial test. It presents and describes also the software resources developed for the test and a short description of how to use them correctly.

All tests described below, must be proceed on the final product (MO130 GSM/GPRS module embedded in the final customer device).

4.1 MO130 INTERFACE

Signal name	IO type	Function	Description
DAIRST	Input	DAI interface	DAI reset
DAIOUT	Output		DAI output data
DAIIN	Input		DAI input data
DAICLK	Input		DAI clock
SWANTANT	Output	Antenna switch command	
ITFLAP	Input	Flap interrupt	Flap detection interruption
BACKLCD	Output	LED control	LCD backlight control
BACKKEY	Output		Keyboard LED control
LEDR_BUZ	Output		Alarm LED control
LEDG	Output		Network LED control
RSLCD	Output	Serial interface to connect LCD or chip melody (2 chip selects)	LCD driver register select
RESETLCD*	Output		LCD driver reset
ALIMLCD	Output		LCD driver power supply
DIMIW	Input		LCD driver input data of the serial link
DOMIW	Output		LCD driver output data of the serial link
CKMIW	Output		LCD driver clock of the serial link
CSMIW1	Output		LCD1 driver chip select of the serial link
CSMIW2	Output		LCD2 driver chip select of the serial link
R1	Input	Keyboard interface	Keyboard row 1
R2	Input		Keyboard row 2
R3	Input		Keyboard row 3
R4	Input		Keyboard row 4
R5	Input		Keyboard row 5
C5	Output		Keyboard column 5
C4	Output		Keyboard column 4
C3	Output		Keyboard column 3
C2	Output		Keyboard column 2
C1	Output		Keyboard column 1
OUI*	Input	ON key	Keyboard ON key

SIMVCC	Output	SIM interface	SIM power supply
SIMRST	Output		SIM reset
SIMCLK	Output		SIM clock
SIMIO	In/output		SIM data
SIMCD	Input		SIM insertion detection
BFTXP	Input	To external microphone (accessory)	Differential input from microphone
BFTXN	Input		Differential input from microphone
BFRXP	Output	To external HP (accessory)	Differential output to earphone
BFRXN	Output		Differential output to earphone
MICP	Input	Microphone interface	Differential input to handset microphone
MICN	Input		Differential input to handset microphone
HPP	Output	Earphone interface	Differential output to 32ohms or 8ohms earphone
HPN	Output		Differential output to 8ohms earphone
LPHP	Output		Differential output to 32ohms earphone
HPIN	Input	Melody chip interface	HP in for melody chip interface
CMDVIB	Output	Vibrator interface	Vibrator command
RI	Output	V24 interface with flow control	Ring Indicator
DSR	Input		Data Send Ready
DCD	Output		Data Carrier Detect
DTR	Output		Data Terminal Ready
CTS	Output		Clear To Send
RTS	Input		Request To Send
TXD1	Output		UART transmit 1
RXD1	Input		UART receive 1
TXD2	Output	UART interface	UART transmit 2
RXD2	Input		UART receive 2
TXIR	Output	IRDA interface	IRDA transmit
RXIR	Input		IRDA receive
CMDIRDA	Output		IRDA command
INTI2C	Input	I2C interface	I2C interrupt
SCLI2C	Output		I2C clock
SDAI2C	In/out		I2C data
CHARGEUR	Input	Load interface	Charge
ON*	Output		Accessories control
NRESPWRON	Input	Reset	Reset signal
ITDATA	Input	Accessories detection	Interrupt signal
CMDSW1	In/output	Spare IO	I/O
CMDSW2	In/output		I/O
CINT	Input	Interrupt	Interruption

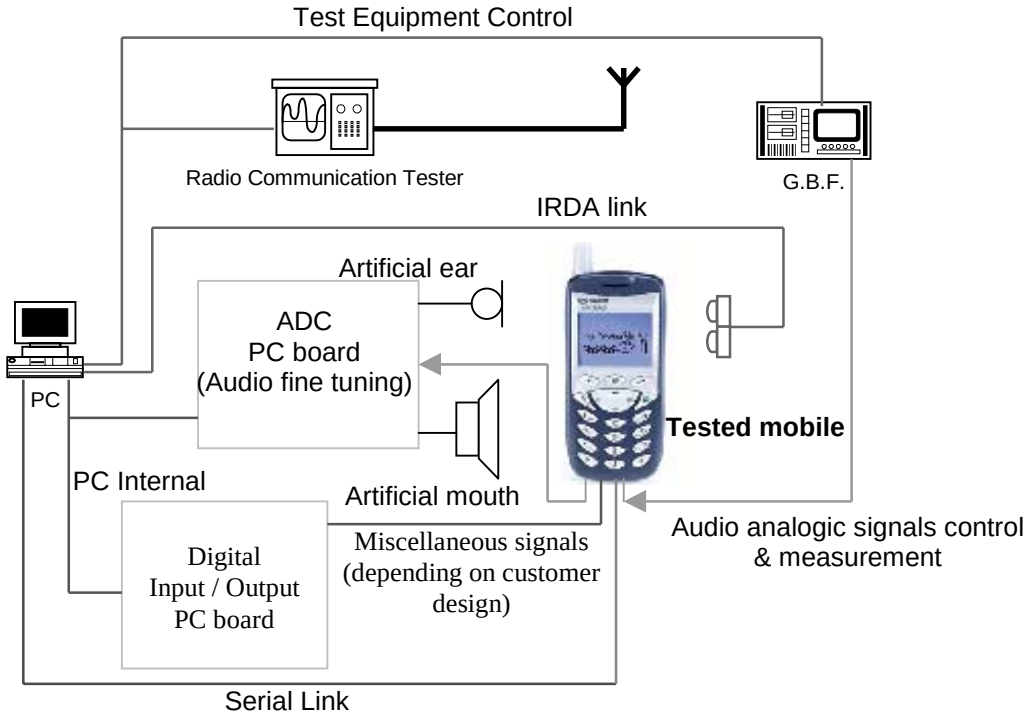
D<0>	In/output	Parallel interface	Data bus
D<1>	In/output		Data bus
D<2>	In/output		Data bus
D<3>	In/output		Data bus
D<4>	In/output		Data bus
D<5>	In/output		Data bus
D<6>	In/output		Data bus
D<7>	In/output		Data bus
D<8>	In/output		Data bus
D<9>	In/output		Data bus
D<10>	In/output		Data bus
D<11>	In/output		Data bus
D<12>	In/output		Data bus
D<13>	In/output		Data bus
D<14>	In/output		Data bus
D<15>	In/output		Data bus
A<0>	In/output		Address bus
A<1>	In/output		Address bus
A<2>	In/output		Address bus
A<3>	In/output		Address bus
A<4>	In/output		Address bus
A<5>	In/output		Address bus
A<6>	In/output		Address bus
OE*	Output		Output Enable
RW*	Output		Read / Write
CS2	Output		Chip select 2
CS3	Output		Chip select 3
CLK13M	Output	Clocks	13MHz clock digital output
CLK32K	Output		32KHz clock digital output
ADC1	Input	ADC/DAC	Analog to digital converter 1
DAC	Output		Digital to analog converter
DISPI	Input	SPI bus	Data input
DOSPI	Output		Data output
CLKSPI	Output		Clock SPI
ENSPI1	Output		SPI enable
VBAT	Input	Power supply	+3.6V battery power supply
V45	Output		+4.5V output power supply
VRDBB	Output		+1.8V output power supply
VRDBBDC	Input		+1.8V DCDC output
VRIO	Output		+2.8V output power supply
VBACKUP	Input		Backup input
GND	Ground	Ground	
OUT_ANT	Output	Antenna output	

4.2 EXAMPLE OF TEST PROCESS EQUIPMENT

The following synoptic gives an example of what could be the equipment configuration for a test done on a mobile with embedded MO130 GSM/GPRS in a customer manufacturing site:

- PC to :
  - control all test equipment
  - send and receive data to/from the tested device
  - run all different programs (test, tuning, setting and final software downloading, manufacturing traceability…),
- Radio Communication Tester, controlled by the PC, to launch RF test sequences,
- artificial ear and mouth with ADC/DAC PC board and Low Frequency Generator (GBF) to test audio quality and optionally, to fine tune it, depending on customer design,
- additional Digital Input/Output PC Board to test others specific signals on the mobile interface (depending on customer design interface),
- Irda link (to test the irda function of the mobile).

This configuration is for reference only and shall be adapted by each customer, following their own equipment and the way they usually use it in their manufacturing site to test their devices.



Example of test process equipment

4.3 BASIC TESTS AND SEQUENCES LIST

4.3.1 DAI

A DAI interface is accessible on the MO130 GSM/GPRS module to perform all audio approval measurements

This interface could not be assigned to other functionalities; so it will be used only in the qualification or approval stages, and doesn't need any test during the test procedure in the manufacturing plant.

4.3.2 SWANTANT

This resource is an analogic output signal; it allows to switch to a RF peripheral for antenna adaptation.

Software resources provided

TBD

Application note for software test tools

TBD

4.3.3 ITFLAP

This signal is a digital input pin on the module; it provides the opening and closing detection on the clam mobile. An interrupt is generated every time the flap is opening or closing.

Software resources provided

The embedded software used for the manufacturing tests provides the following frame for ITFLAP testing.

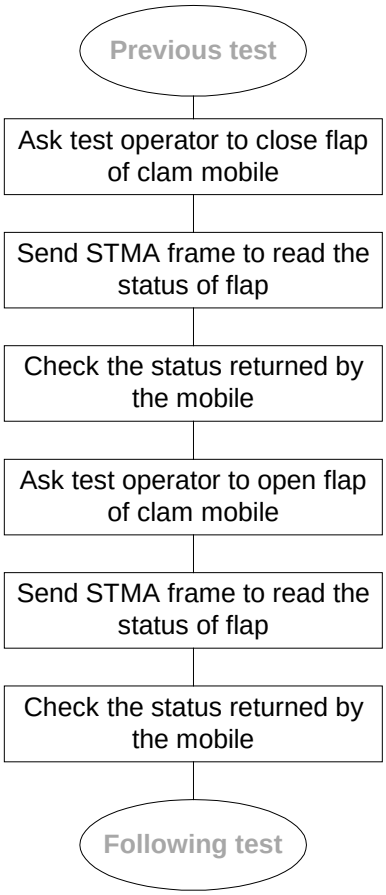
Command Frames		Answer Frames	
0x85	**R_FLAP**	**0x7F**	**V_FLAP**
		0xXX	state of the flap

Frame parameters for answer:

0x00	Flap is closed
0x01	Flap is opened

Application note for software test tools

The following algorithm shows how these frames could be used by a software tool to perform ITFLAP testing.



4.3.4 LED control

The MO130 GSM/GPRS module allows the management of led by using four analogic output signals :

- **BACKLCD** : this pin manages the backlight of the LCD.
- **BACKKEY** : this pin manages the backlight of the keyboard
- **LEDR_BUZ** : this pin manages one part of network led (red part)
- **LEDG** : this pin manages one part of the network led (green part).

If you need more information about the characteristics of these signals, please refer to SCT TMO MASV2 SPEC 57 – MO130 Module - Application Note

Software resources provided

The STMA frames provided by the embedded software are described below :

Command Frames		Answer Frames	
0x54	**W_START_LCD**	**0x4E**	**A_START_LCD**
0x??	type	0x??	type

Frame parameters for command:

type of LCD test	**0x0E *BACKLCD & BACKKEY***
	0x05 *LEDG*
	0x06 *LEDR_BUZ*

Frame parameters for answer (identical to command):

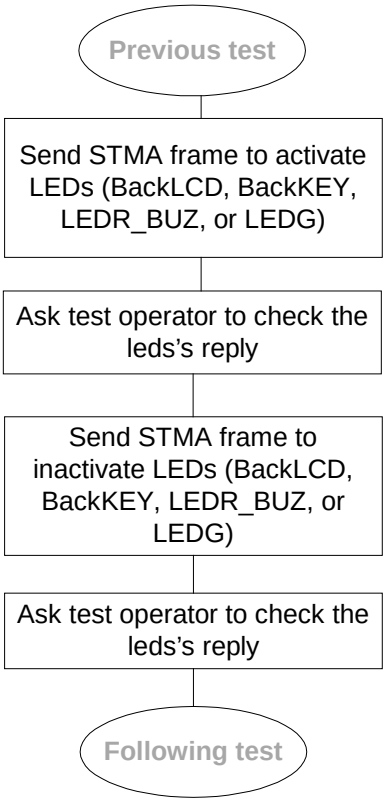
type of LCD test	**0x0E *BACKLCD & BACKKEY***
	0x05 *LEDG*
	0x06 *LEDR_BUZ*

Frame to stop the test (return to no light state, if necessary)

Command Frames		Answer Frames	
0x55	**W_STOP_LCD**	**0x4F**	**A_STOP_LCD**

Application note for software test tools

The following algorithm shows how these frames could be used by a software tool to perform LED testing.



4.3.5 LCD interfaces

Two types of LCD ports are available on the module MO-130. The first one is the serial port, used for traditional monochrome LCD and the second one is the parallel port more suitable for color LCD and dual LCD.

Whatever the interface used to manage the LCD, the embedded software provides some tests' functionalities such as displaying a checkerboard, displaying a black screen, …

Software resources provided

The STMA frames provided by the embedded software are describing below :

Used to manage the LCD

Frame to start the LCD test

Command Frames		Answer Frames	
0x54	**W_START_LCD**	**0x4E**	**A_START_LCD**
0x??	type	0x??	type

Frame parameters for command:

type of LCD test	0x01 To Test Bitmap.
	0x02 To Test Inverted Bitmap
	0x03 To Test black screen
	0x04 To switch off the LCD backlight and keyboard LED
	0x05 To Test green LED Network
	0x06 To Test red LED Network
	0x07 No effect
	0x08 To test color screen (white, then green, then blue, then red, in cycle, each test duration is ¼ s)
	0x09 To Test white screen
	0x0A To Test red screen
	0x0B To test green screen
	0x0C To Test blue color screen
	0x0D To Test backlight LCD OFF
	0x0E To Test backlight LCD ON
	0x0F To Test alignment
	0x10 To Test the contrast

Frame parameters for answer are identical to command.

Frame to stop the LCD test

Command Frames		Answer Frames	
0x55	**W_STOP_LCD**	**0x4F**	**A_STOP_LCD**

Frame to tune contrast

Command Frames		Answer Frames	
0xEE	**W_CONTRAST_TUNING**	**0xEE**	**A_CONTRAST_TUNING**
0x??	LCD	0x??	LCD
0x??	command	0x??	command
		0x??	error code

Frame parameters for command:

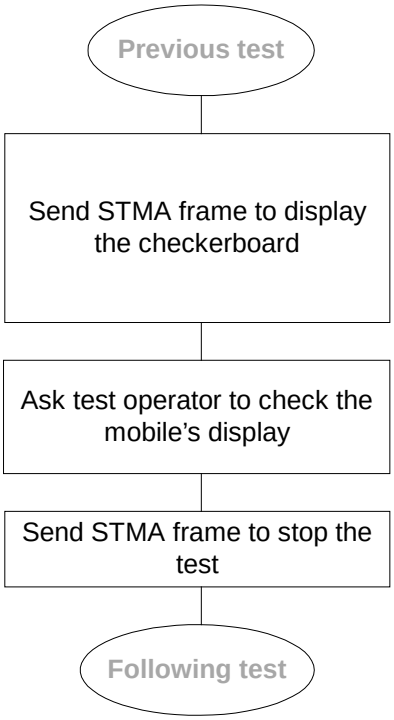
LCD	0x00 to tune main LCD
	0x01 to tune sub LCD
Command	0x00 : display contrast tuning screen
	0x01 : tune down contrast
	0x02 : tune up contrast
	0x03 : return to default value
	0x04 to 0xFE : set contrast value
	0xFF : end of tuning, and return to screensaver

Frame parameters for answer:

LCD	0x00 to tune main LCD
	0x01 to tune sub LCD
Command	0x00 : display contrast tuning screen
	0x01 : tune down contrast
	0x02 : tune up contrast
	0x03 : return to default value
	0x04 to 0xFE : set contrast value
	0xFF : end of tuning, and return to screensaver
Error code	0x00 : no error
	0x01 : contrast requested inferior to minimum authorized
	0x02 : contrast requested superior to maximum authorized
	0x03 : contrast requested too small, doesn't allow full range user values
	0x04 : contrast requested too high, doesn't allow full range user values

Application note for software test tools

The following algorithm shows how these frames could be used by a software test tool to perform LCD testing.



4.3.6 Keyboard interface

This interface is assigned to manage a keypad for MMI. The embedded software provides a test functionality to check if the keypad is working correctly.

This interface could not be assigned to other functionalities.

Software resources provided

Frame to start the Keyboard test

Command Frames		Answer Frames	
0x4F	**W_START_KBD**	**0x49**	**A_START_KBD**

Frame to query the keys hit

Command Frames		Answer Frames	
0x51	**R_PARAM_KBD**	**0x4B**	**V_PARAM_KBD**
		0xNN	NN bytes
		0x??	byte 0
		…	
		0x??	byte NN

Frame parameters for answer:

NN bytes	Number of bytes following
byte 0 to byte NN	Result of keys hit (one key per bit)

Format of the answer

1 byte per key, each key is numbered from 1 to XX
As long as the requested list has not being required, a pressed key, even several times, materialize by 1 byte at state 1 remains at 1. The combination of several keys being possible(Touches 1 1 2 SEND => hexadecimal value : 1084000000 (always coded with 5 bytes). Byte 0=10, byte 1=84, byte 2=00, byte3=00, byte 4=00, byte=00).

Location of the concerned bit inside the concerned byte

Num. byte = (Num.key -1) / 8 (from 0 to Nbytes).

Num. bit = (Num. key -1) % 8 (from 0 to bit 7).

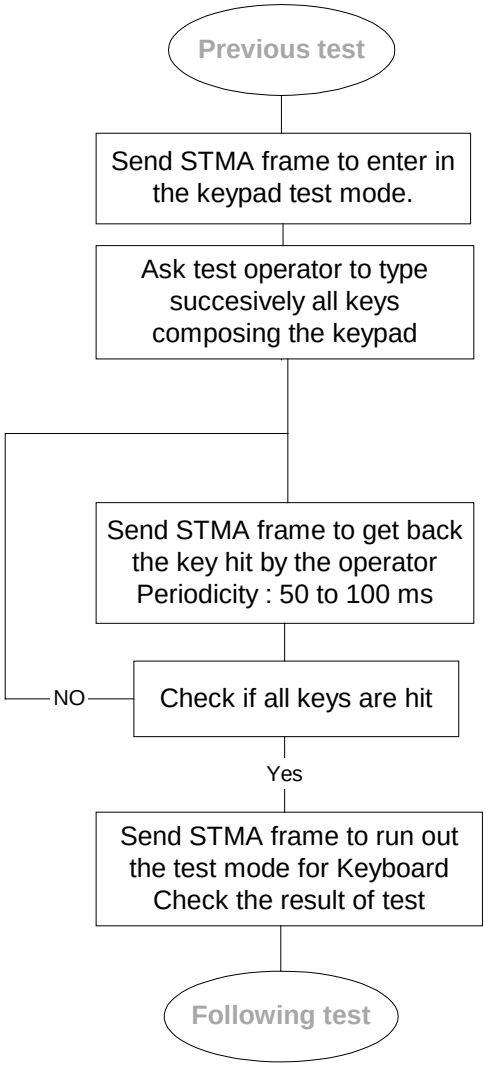
Pressed keys	Num. byte [0, NN]	Num. bit in byte [0, 7]	Associated hexadecimal value
3	0	0	01
6	0	1	02
9	0	2	04
#	0	3	08
SEND	0	4	10
MUTE	0	5	20
DOWN	0	6	40
C	0	7	80
MENU	1	0	01
UP	1	1	02
1	1	2	04
4	1	3	08
7	1	4	10
*	1	5	20
V	1	6	40
2	1	7	80
5	2	0	01
8	2	1	02
0	2	2	04
PROG	2	3	08
END	3	6	40

Frame to stop the Keyboard test

Command Frames	Answer Frames
0x50 W_STOP_KBD	**0x4A A_STOP_KBD**

Application note for software test tools

The following algorithm shows how these frames could be used by a software tool to perform Keypad testing.



4.3.7 OUI key

The “OUI” key is an input signal dedicated to perform the mobile’s ON / OFF function. The embedded software has some others mechanisms to turn the mobile ON during the tests integration (plug charger in,..).So, a dedicated test to this signal is nevertheless needed.

Software resources provided

The STMA frame provided by the embedded software is described below :

Command Frames		Answer Frames	
0xE4	**R_IT_TEST**	**0xE4**	**V_IT_TEST**
0x01		0x0?	result

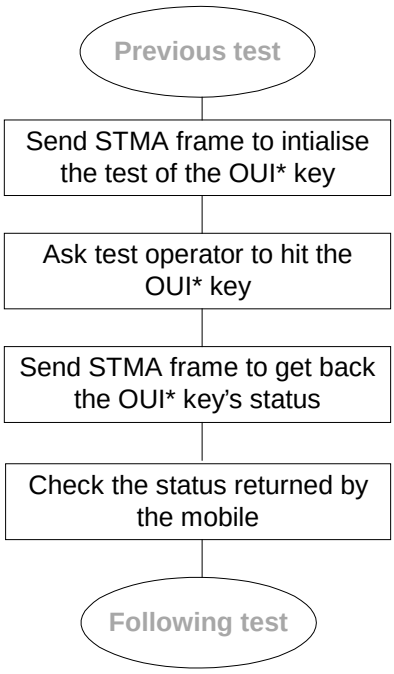
Frame parameters for answer:

0x00	OUI KEY has not occurred
0x01	OUI KEY has occurred

Please note that there is a timeout of 3 seconds before responding negatively if the OUI KEY has not occurred.

Application note for software test tools

The following algorithm shows how this frame could be used by a software tool to perform ON/OFF testing



4.3.8 SIM interface

This interface is assigned to manage a sim card. The purpose of this test is to check all signals of the sim interface. To do that, the embedded software provides some mechanisms such as those described below.

When it starts, the embedded software sets the sim card power to ON, actives the clock and the sim card reset. More, an identification sequence of sim card is launched (cf. 4.3.16)

for the corresponding software resource to know if the sim card is detected).

Software resources provided

The STMA frames provided by the embedded software are described below :

Frame to command the sim signals

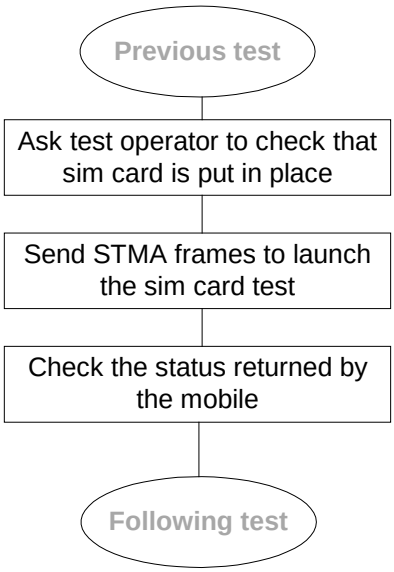
Command Frames		Answer Frames	
0xC4	**W_SIM_TEST**	**0xC4**	**A_SIM_TEST**
0x0X	command		

Frame parameters for command:

0x00	To apply a null voltage on all signals : SIM VCC, IO, CLK, RST
0x01	To apply a 2.9 V voltage on all signals : SIM VCC, IO, CLK, RST
0x02	To apply a 1.8 V voltage on all signals : SIM VCC, IO, CLK, RST
0x03	To apply a 2.9 V voltage on signal SIM VCC
0x04	To apply a 2.9 V voltage on signals SIM VCC, IO
0x05	To apply a 2.9 V voltage on signals SIM VCC, CLK
0x06	To apply a 2.9 V voltage on signals SIM VCC, RST

Application note for software test tools

The following algorithm shows how these frames could be used by a software tool to perform SIM card interface testing.



4.3.9 Audio

The functional audio test must check all components used to manage voice and sounds. This test permits to test all audio paths, the transducers such as microphone and speaker.

MO130 GSM/GPRS module includes a set of programmable memory field assigned for audio tuning management. These settings are used to fix the filters' parameters and the level of the amplifiers. Adjust these settings allow to fine tune and set up the mobile's acoustic (take dispersion of the transducers and mobile's mechanical design constraints into account)

Default settings are written during WONU's MO130 GSM/GPRS module manufacture process

Two ways to proceed the audio testing and settings are available:

- The first one is based on a basic functional test: it runs some electrical tests only on the audio path and a functional test on the transducers. In this case, default settings must be used.

- The second one allows to adjust dynamically settings on the mobile phone. This test is more complete than the previous one : it aims to make final fine tuning in order to be fully compliant with the audio GSM gauges.

Software resources provided

The embedded software used for the manufacturing tests provides the following frame for audio testing.

Frame to set default Audio parameters

Command Frames		Answer Frames	
0xDA	**W_AUDIO_DEF_PARAM**	**0xDA**	**A_AUDIO_DEF_PARAM**
0x01	write	0x01	write
0x01	eeprom	0x01	eeprom
0x??	lgin	0x??	lgout
0x05	num	0x05	num
...	datain (lgin-1 bytes)	...	dataout (lgout-1 bytes)

Frame parameters for command:

lgin	Length of audio field +1
datain	Audio field(lgin –1 parameters)

Frame parameters for answer (identical to command) :

lgout	Length of audio field +1
dataout	Audio field(lgout –1 parameters)

The parameters of this frame must be extracted from the ".sgv" file created with "WONU Audio parameters setup" tool.

The ".sgv" file contains several items, of variable length. The first byte in each item represents the total "size" of the item, expressed in words(2x1Byte)

The "datain" field must contain the parameters of the 6th item of the ".sgv" file (including the byte representing the "size"). Since this item represents the parameters of audio tests, its "size" in words should be (1 + 10 * number_of_tests) ; for a typical 3-tests item, it will be 31 words (62 bytes).

The "lgin" field is the length in bytes of the "datain" field, plus one byte. Therefore, the "lgin" field should be ("size" * 2 + 1).

Frame to proceed Audio test

Command Frames		Answer Frames	
0xCC	**W_TEST_AUDIO_PATH**	**0xBF**	**A_TEST_AUDIO_PATH**
0xaa	id test	0xaa	id test
0xbb	micro gain	0xbb	micro gain
0xcc	speaker gain	0xcc	speaker gain

Frame parameters for command:

id test	choice of audio path
micro gain (uplink)	between 0 (-12 dB) to 24 (+12 dB)
speaker gain (downlink)	between 0 (-6 dB) to 12 (+6 dB)

Frame parameters for answer (identical to command):

id test	choice of audio path
micro gain (uplink)	between 0 (-12 dB) to 24 (+12 dB)
speaker gain (downlink)	between 0 (-6 dB) to 12 (+6 dB)

The "id test" selects one of the tests described in the ".sgv" file. Since there can be up to 9 different tests, this value should be between "100" and "108" (cf. SCT TMO LOG SPEC 0414 - Audio parameters setup).

The two parameters "micro gain" and "speaker gain" set the values of VULPG and VDLPG

Frame to set filters

Command Frames		Answer Frames	
0xB3	**W_FILTER**	**0xAC**	**A_FILTER**
0x??	MSB filter 1 word 0	0x??	MSB filter 1 word 0
0x??	LSB filter 1 word 0	0x??	LSB filter 1 word 0
…		…	
0x??	MSB filter 1 word 4	0x??	MSB filter 1 word 4
0x??	LSB filter 1 word 4	0x??	LSB filter 1 word 4
0x??	MSB filter 2 word 0	0x??	MSB filter 2 word 0
…		…	
0x??	LSB filter 4 word 4	0x??	LSB filter 4 word 4

Frame parameters for command:

filter 1	programming filter UL1 with 10 parameters bytes
filter 2	programming filter DL1 with 10 parameters bytes
filter 3	programming filter UL2 with 10 parameters bytes
filter 4	programming filter DL2 with 10 parameters bytes

Frame parameters for answer (identical to command):

filter 1	programming filter UL1 with 10 parameters bytes
filter 2	programming filter DL1 with 10 parameters bytes
filter 3	programming filter UL2 with 10 parameters bytes
filter 4	programming filter DL2 with 10 parameters bytes

The 10 parameters bytes are the 5 16bit-coefficients as they appear in the setup tool, written in big endian (0x1234 will be written "12 34"). As a result, a flat filter (with 5 coefficients 0x0000, 0x0000, 0x0000, 0x0000, 0x7FFF) is written "00 00 00 00 00 00 00 00 7F FF".

Frame to write audio adjustments

Command Frames			Answer Frames		
0xDB	**W_AUDIOPARAM**		**0xDB**	**A_AUDIOPARAM**	
0x??	rw	0x??	0x??	rw	0x??
0x??	num	0x??	0x??	num	0x??
0x??	val0	0x??	0x??	val0	0x??
0x??	val1	0x??	0x??	val1	0x??
0x??	val2	0x??	0x??	val2	0x??
0x??	val3	0x??	0x??	val3	0x??
0x??	val4	0x??	0x??	val4	0x??
0x??	val5	0x??	0x??	val5	0x??
0x??	val6	0x??	0x??	val6	0x??
0x??	val7	0x??	0x??	val7	0x??
0x??	val8	0x??	0x??	val8	0x??
0x??	val9	0x??	0x??	val9	0x??

Frame parameters for command:

rw	0 : read, 1 : write
data id	0 -> gains 1 -> traceability 2 to 9 -> filters 10 -> filters activation
val	value (10 bytes)

Frame parameters for answer (identical to command):

rw	0 : read, 1 : write
data id	0 -> gains 1 -> traceability 2 to 9 -> filters 10 -> filters activation
val	value (10 bytes)

For more information, refer to SCT TMO LOG SPEC 0414 - Audio parameters setup documentation.

Data id	Size (bytes)	Usage	Comment
0	4	VULPG1, VULPG2, VDLPG1, VDLPG2	VULPG is between -12dB and +12dB, coded between 0 and 24. VDLPG is between -6dB and +6dB, coded between 0 and 12.
1	10	Test data and result	Format and values are free
2 to 9	10	Filters UL1.1,DL1.1,UL1.2, DL1.2,UL2.1,DL2.1, UL2.2 or DL2.2	The 10 bytes represent the 5 16bit-coefficients as they appear in the setup tool, but written in little endian ("0x1234" will be written "34 12"). As a result, a flat filter (with 5 coefficients 0x0000, 0x0000, 0x0000, 0x0000, 0x7FFF) is written "00 00 00 00 00 00 00 00 FF 7F".
10	8	Filters activation	Each byte is set to 1 to activate a filter (i.e. override default one), and 0 to deactivate it The 8 filters order is the same : UL1.1, DL1.1, UL1.2, DL1.2, UL2.1, DL2.1, UL2.2 or DL2.2

Frame to test speaker functionally

Command Frames		Answer Frames	
0x56	**W_START_RING**	**0x50**	**A_ START_RING**
0x??	melody	0x??	melody
0x??	volume	0x??	volume

Frame parameters for command:

Melody	Melody to play (1 to 14, 5 = continuous tone)
Volume	1 = feeble, 4 = strong

Frame parameters for answer (identical to command):

Melody	Melody to play (1 to 14, 5 = continuous tone)
Volume	1 = feeble, 4 = strong

Frame to stop speaker test

Command Frames		Answer Frames	
0x57	**W_STOP_RING**	**0x51**	**A_ STOP_RING**

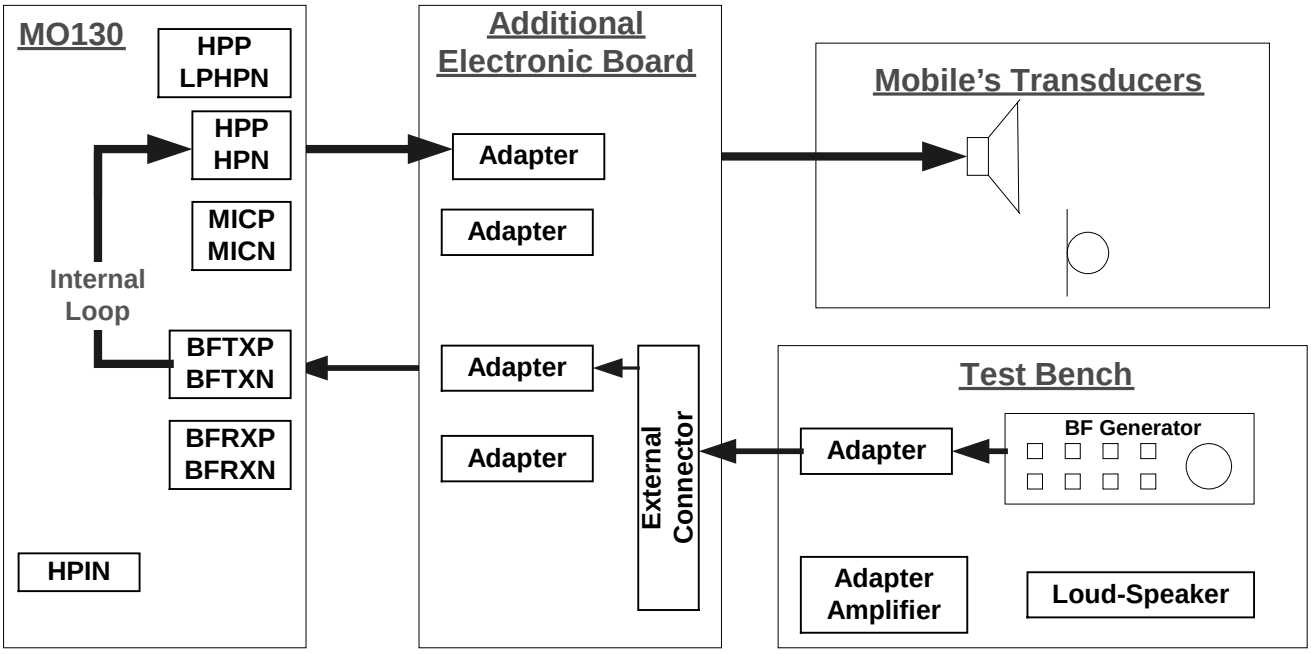
Application note for software test tools

This application note gives some examples for audio' paths testing.

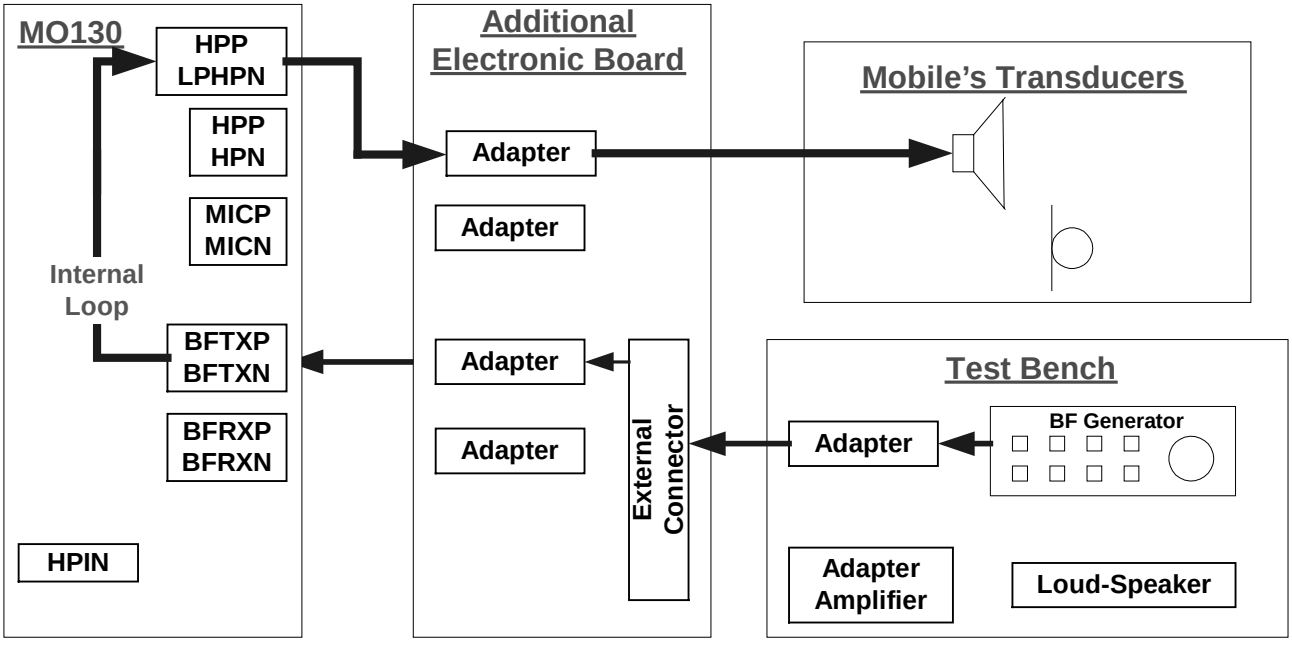
4.3.9.1 Test speaker's path

To do the test of the speaker's path, we can proceed as it is shown below. Build the test bench with

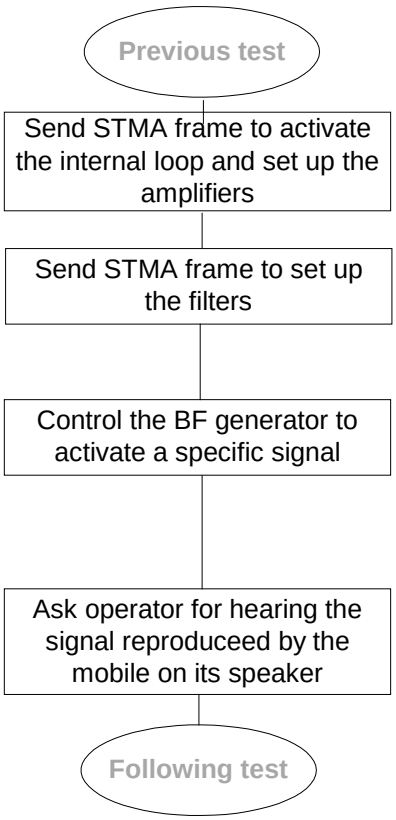
- BF generator;
- Loud-speaker;
- Electronic and audio adapter;
- Amplifier.
- Connector to link the test bench to a mobile



If the final product uses a second speaker for incoming call ringing, the functional test could be as following schematic :

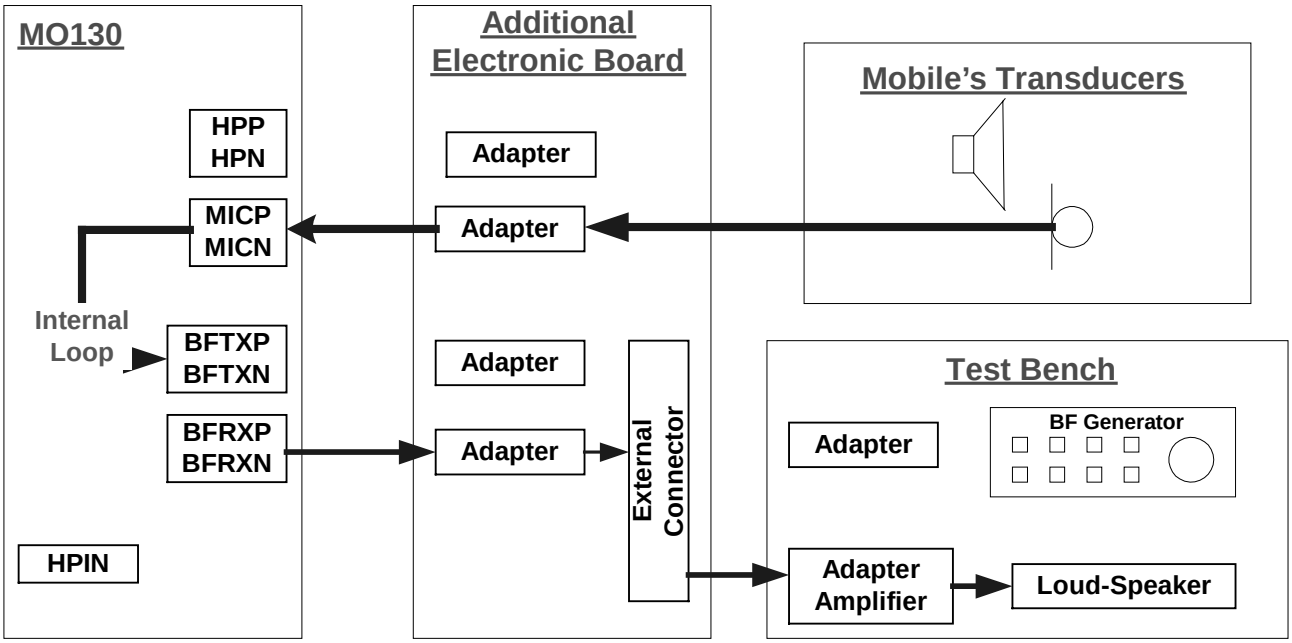


The following algorithm shows how the software tool could include this test.

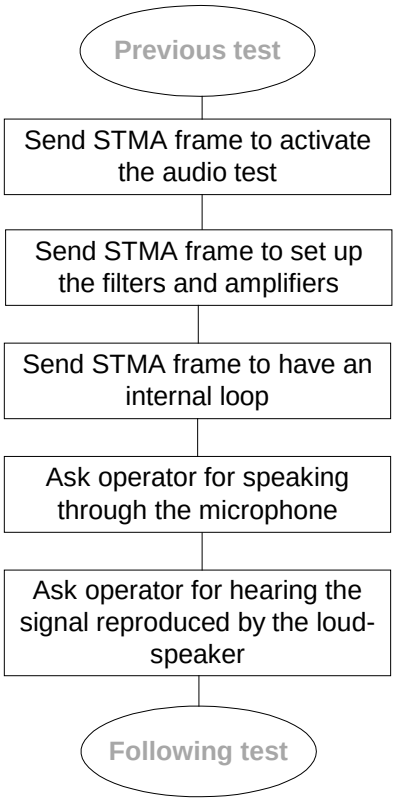


Test microphone's path

The test of microphone's path is depicted below.



The following algorithm shows how the software tool could include this test.



This section will be completed by WONU on next specification release

4.3.10 Melody chip interface

A melody chip can be add on the electronic board by customer. This component is generally managed by the serial bus used traditionally for monochrome LCD (using another chip select).

The audio input signal HPIN is assigned to collect all analog audio signal provided by this additional melody chip.

Depending on customer needs, the software could embed some functionalities to do the tests (such as playing a specific melody on the earphone). These specifications which are using the melody chip have to be designed by the customer.

4.3.11 Vibrator interface

This signal is an analog output signal; it allows to manage the vibrator used through the additional electronic board.

Software resources provided

The STMA frames provided by the embedded software are described below :

Frame to start the vibrator test

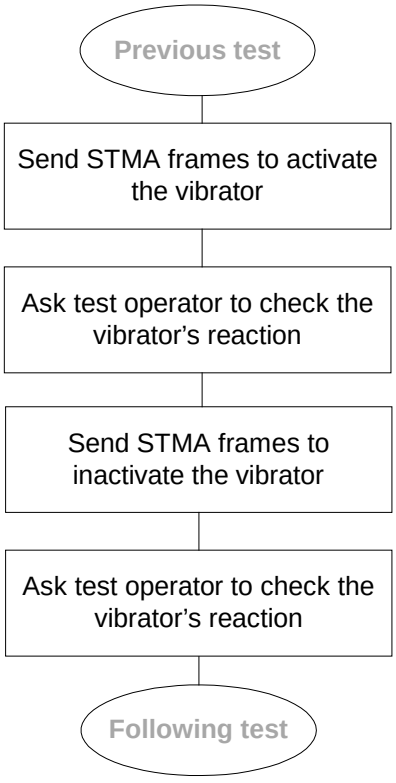
Command Frames		Answer Frames	
0x69	**W_START_VIBRATOR**	**0x63**	**A_START_VIBRATOR**

Frame to stop the vibrator test

Command Frames		Answer Frames	
0x6A	**W_STOP_VIBRATOR**	**0x64**	**A_STOP_VIBRATOR**

Application note for software test tools

The following algorithm shows how these frames could be used by a software tool to perform vibrator interface testing



4.3.12 V24 interface

This interface represents the main way to allow the communication with the MO130 GSM/GPRS module. It is composed with the following signals:

RI	Output	Ring Indicator
DSR	Input	Data Send Ready
DCD	Output	Data Carrier Detect
DTR	Output	Data Terminal Ready
CTS	Output	Clear To Send
RTS	Input	Request To Send
TXD1	Output	UART transmit 1
RXD1	Input	UART receive 1

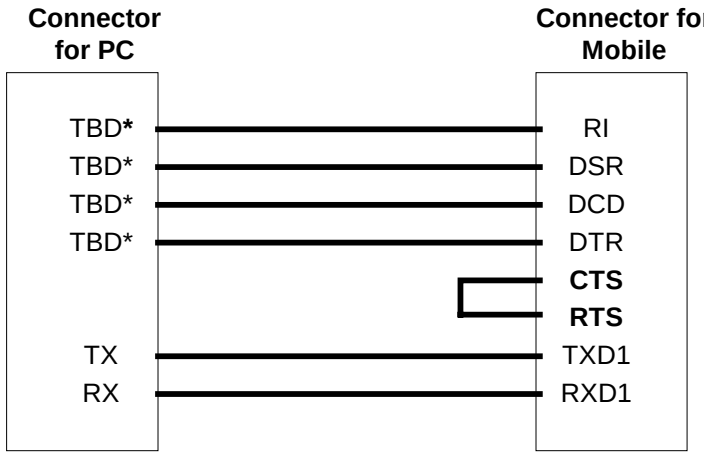
As it was specified on the chapter DESIGN REQUIREMENT, it is mandatory to have an access to one part of these signals from outside (minimal requirement concerns TX and RX signals).

Utility of this test will depend on the final product design. There are two ways:

- The design of the final product requires a full V24 interface;
- The design of the final product needs only a basic serial link (Tx and Rx).

4.3.12.1 Full V24 interface

In case a full V24 interface is required on the final product, it is recommended to perform all tests on each signal to be sure this interface is compliant. To proceed this test, it is necessary to use a specific serial link designed as below :



*** : To be defined**

On mobile's side, **TXD1** and **RXD1** signals come directly from the UART1 of the MO130 GSM/GPRS module. To perform the test, it must be connected to a PC UART (such as COM1 or COM2). As the Voltage levels of the signals delivered by the MO130 GSM/GPRS module are incompatible with standard PC serial link interface, it is recommended to design an adapter to shift the voltage level. If you need more information about the electrical specifications of these signals, please refer to the document SCT TMO MASV2 SPEC 57 - MO130 Application Note.

CTS and **RTS** are two signals provided also by the UART1 on MO130 GSM/GPRS module. To ensure a communication between the PC and the mobile by the serial link, it is necessary to choose :

1. CTS must be connect to the Ground level
2. CTS must, as it was specified above, be connected to RTS.

The others signals (**RI, DSR, DCD, DTR**) could be considered as IOs. On PC side, these signals must also be managed as IOs (for example you can use a DIO board to control them).

Software resources provided

The STMA frame provided by the embedded software is described below :

Frame to test V24

Command Frames		Answer Frames	
0xE1	**W_V24**	**0xE1**	**A_V24**
0x??	Output desired	0x??	Output desired
		0x??	Input states

Frame parameters for command:

Output desired :

	request to change an output state				new value for each output			
Bit	7	6	5	4	3	2	1	0
	W_CTS	W_DCD	W_DSR	W_RI	CTSV	DCDV	DSRV	RIV

W_CTS, W_DCD, W_DSR, W_RI : 1 : request to change
0 : no request to change

CTSV, DCDV, DSRV, RIV : 1 : logical 1
0 : logical 0

Frame parameters for answer:

Input states :

Bit	7	6	5	4	3	2	1	0
	X	X	X	X	X	X	DTRV	RTSV

DTRV, RTSV : 1 : logical 1
0 : logical 0
X : not used

Note : Voltage read can be 0V for logical level 1. Please refer to MO130 module for voice centric phone preliminary specification (SCT TMO MASV2 SPEC 14).

4.3.12.2 Basic serial link

In this case, only the signals TXD1 and RXD1 are input/output on the final product to ensure the communication. The test of this interface will consist to send and receive data. The embedded software provides some functionalities to perform that.

Software resources provided

The STMA frames provided by the embedded software are described below :

Frame to test V24

Command Frames		Answer Frames	
0xE1	**W_V24**	**0xE1**	**A_V24**
0x??	Output desired	0x??	Output desired
		0x??	Input states

Frame parameters for command:

Output desired :

	request to change an output state				new value for each output			
Bit	7	6	5	4	3	2	1	0
	W_CTS	W_DCD	W_DSR	W_RI	CTSV	DCDV	DSRV	RIV

W_CTS, W_DCD, W_DSR, W_RI : 1 : request to change
0 : no request to change

CTSV, DCDV, DSRV, RIV : 1 : logical 1
0 : logical 0

Frame parameters for answer:

Input states :

Bit	7	6	5	4	3	2	1	0
	X	X	X	X	X	X	DTRV	RTSV

DTRV, RTSV : 1 : logical 1
0 : logical 0
X : not used

Note : Voltage read can be 0V for logical level 1. Please refer to MO130 module for voice centric phone preliminary specification (SCT U37 MASV2 SPEC 14C).

Application note for software test tools

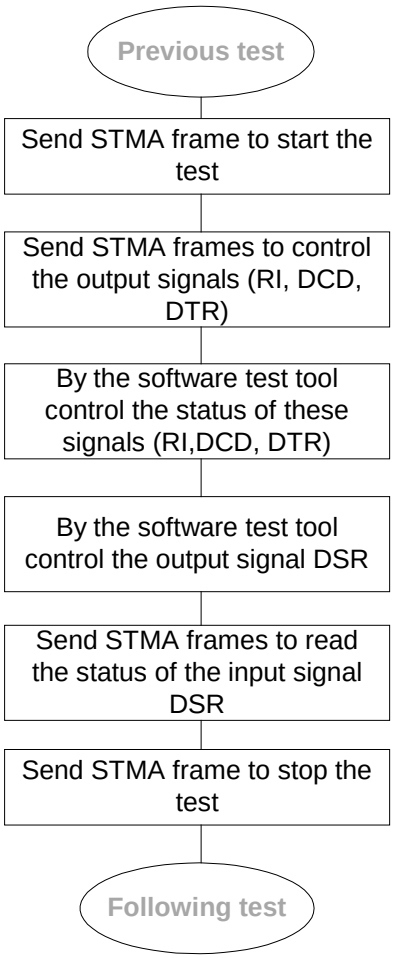
The following algorithm shows how these frames could be used by a software tool in case the full V24 is output.

1. Be sure the serial link is set up correctly between PC and mobile to be tested (RTS connects to CTS);

2. Be sure the serial communication is established by sending and receiving data during a previous test.

If the serial communication is done, that means the signals TxD1, RxD1, RTS and CTS are available and tested.

Use this algorithm to test the others signals:



4.3.13 UART interface2

This interface is a second serial link provided by the MO130 GSM/GPRS module. Like UART1 it is necessary to design an adapter to convert the MO130 GSM/GPRS signals level to be compatible with a PC signals level. If you need more information about the level of these signals, please refer to SCT TMO MASV2 SPEC 57- MO130 Application Note.

It is strongly recommended to let this interface externally accessible for the software download or debug; these access could be provided by two test pins or through the connector on the final product. In this case the software download saves time.

If this UART is used to have an internal dialog with another chipset, the test process will be also specified by the customer; depending on its own implementation.

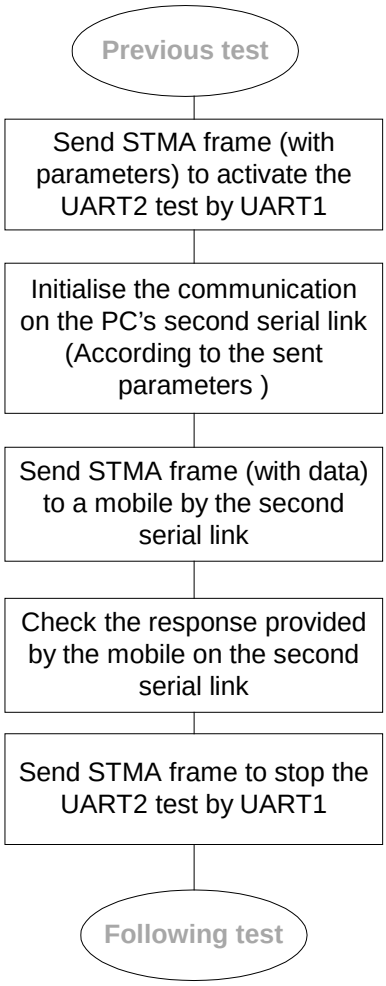
Software resources provided

The embedded software used for the manufacturing tests provides the following frames for UART2 testing.

TBD

Application note for software test tools

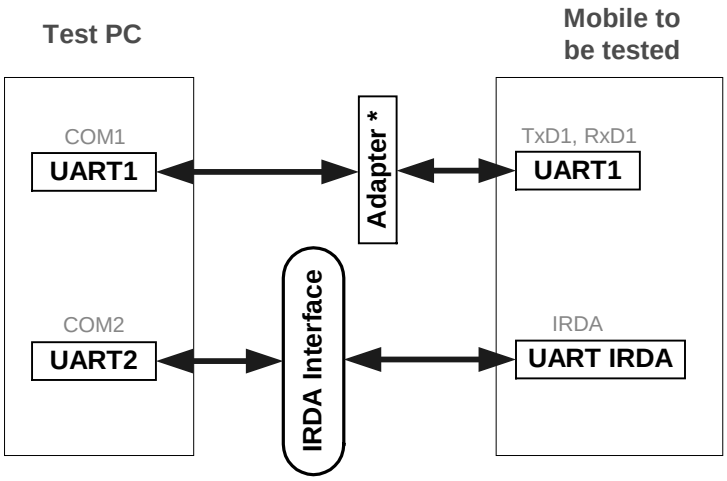
To proceed this test, it will be necessary to connect this second UART to another PC serial link. After that, the software tool used to manage the test could proceed the UART test as below.



4.3.14 IRDA interface

This interface is composed by three signals (TXIR, RXIR, CMDIRDA). The test of this interface consists in creating a real IRDA communication between the test's PC and the tested mobile.

To proceed this test, it will be necessary to have an IRDA interface plugged to the test's PC. The configuration required is shown below. The IRDA interface used for testing purpose shall be compliant with standard IRDA interface.



* : used to shift and ajust the level of the signals

Software resources provided

One STMA frame is provided to perform the test of this interface. Two IRDA frames are designed to check the exchanges between PC and mobile

STMA frame:

Frame to proceed the IRDA test

Command Frames		Answer Frames	
0xC6	**R_IRDA_EXCHANGE**	**0xC6**	**V_IRDA_EXCHANGE**
		0xXX	See the table below

Frame parameters for answer:

0x01	Everything is OK
0x02	PC did not send frame
0x04	PC sent a wrong frame or an incomplete frame

IRDA frame:

This first frame will be received by the PC during the IRDA communication. Composed by 39 bytes, this frame must be processed by the software tool which is in charge of managing the test before deciding to send a second IRDA frame in response. This frame is called "**IRDA FRAME 1**".

The second IRDA frame represents the response to the first IRDA frame receipt by the PC. Composed by 28 bytes, it will be sent by the PC IRDA interface to the mobile. This frame is called "**IRDA FRAME 2**".

IRDA FRAME 1 :

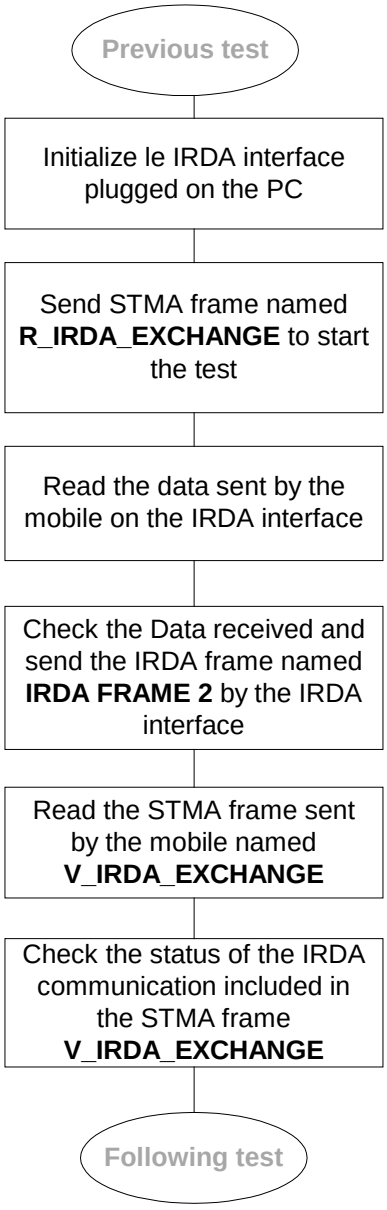
FF FF FF FF FF FF FF FF FF FF C0 FE BF 01 C9 74 6F 1D 96 26 74 1D 02 00 00 90 05 00 53 41 47 45 4D 20 35 32 D6 94 C1

IRDA FRAME 2 :

FF FF FF FF FF FF FF FF FF FF C0 FF 3F 01 00 00 00 00 FF FF FF FF 01 00 00 82 C6 C1

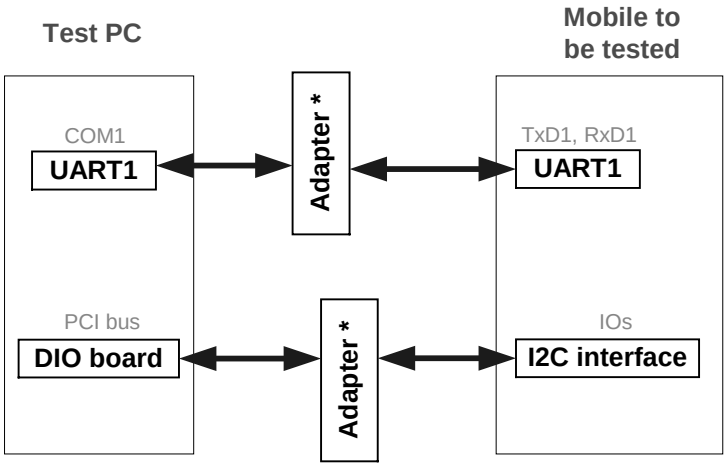
Application note for software test tools

Please refer to this Following algorithm to develop the test functionalities for the IRDA interface.



4.3.15 I2C interface

This interface is composed of three IO signals (INTI2C, SCTI2C, SDAI2C). The purpose of this test is to check the electrical function of each signal which composes this interface. To do that, it recommends to add an interface (such as DIO board) on PC which allows to control these IOs; the test bench is depicted below.



* : used to shift and ajust the level of the signal s

Software resources provided

The embedded software used for the manufacturing tests provides the following frame for INTI2C testing.

Command Frames		Answer Frames	
0xE4	**R_IT_TEST**	**0xE4**	**V_IT_TEST**
0x0D		0x0?	result

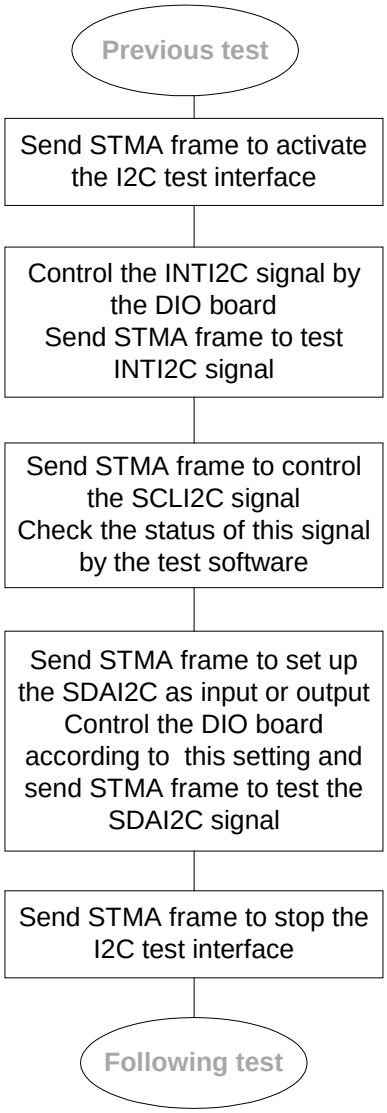
Frame parameters for answer:

0x00	INTI2C has not occurred
0x01	INTI2C has occurred

Please note that there is a timeout of 3 seconds before responding negatively if the INTI2C has not occurred.

Application note for software test tools

Please refer to this following algorithm to develop the test functionalities for the I2C interface.



4.3.16 Charger

The mobile is not charging the battery during the test, because the embedded test s/w doesn't manage this operation. The mobile works with the charger, even without the battery.

Composed of one signal, it is assigned to perform the charge of the battery used by the mobile

Software resources provided

The embedded software used for the manufacturing tests provides the following frame for CHARGER testing.

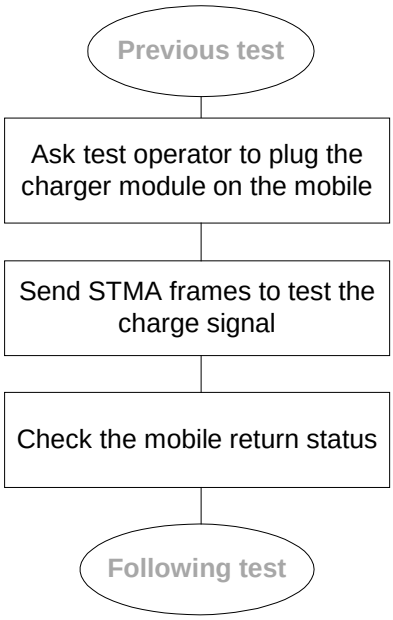
Command Frames		Answer Frames	
0x68	**R_SIMCOMCCCH**	**0x62**	**V_SIMCOMCCCH**
		0x??	charger status & SIM status
		0x??	com status

Frame parameters for answer:

0x00	no charger	no SIM
0x01	no charger	SIM inserted
0x10	charger connected	no SIM
0x11	charger connected	SIM inserted

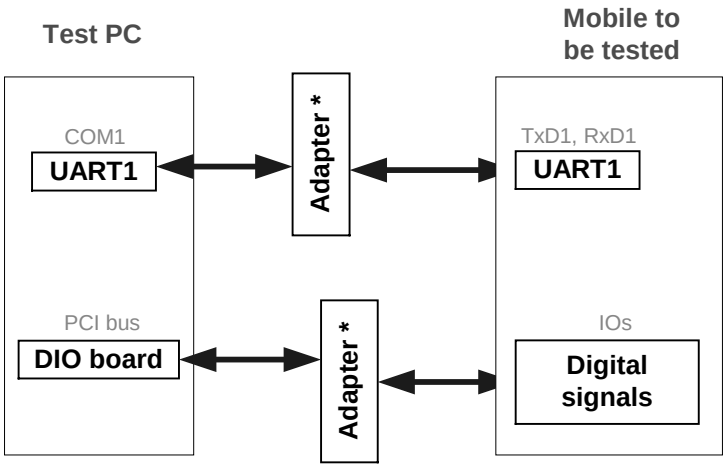
Application note for software test tools

To proceed the test on this signal, follow this algorithm.



4.3.17 ON*

This signal is a digital output and it could be assigned to control an accessory. To perform some tests, it is recommended to have an additional interface (such as DIO board) on the PC used to manage the test. The following draw presents an example of test bench implementation:



* : used to shift and adjust the level of the signals

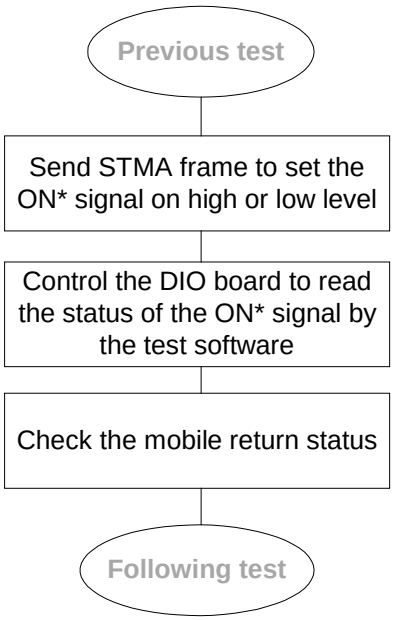
Software resources provided

The STMA frame provided by the embedded software is described below :

TBD

Application note for software test tools

The following algorithm shows how these frames could be used by a software tool to perform ON* testing



4.3.18 ITDATA

This signal is a digital input and is generally assigned to detect an accessory. To perform some tests, it's recommended to have an additional interface (such as DIO board) on the PC used to manage the test. An example of test bench, is shown on section 4.3.17 (ON*).

Software resources provided

The STMA frame provided by the embedded software is described below :

Frame to detect IT_DATA

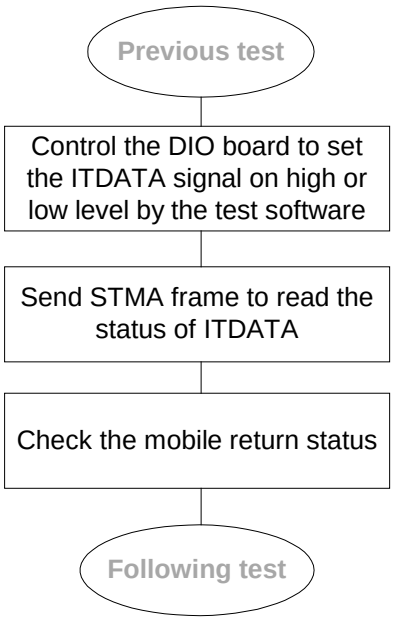
Command Frames		Answer Frames	
0xB4	**R_IT_DATA**	**0xAD**	**V_IT_DATA**
		0x??	result

Frame parameters for answer:

0x00	no IT
0x01	IT_DATA detected

Application note for software test tools

The following algorithm shows how these frames could be used by a software tool to perform ITDATA testing



4.3.19 GENERIC IT TEST

That test can verify if an IT occurs. The frame must be sent before activating the IT. There is a timeout of 3 seconds before responding negatively if the tested IT has not occured. To perform some tests, it's recommended to have an additional interface (such as DIO board) on the PC used to manage the test. An example of the test bench is shown in the section 4.3.17.

Software resources provided

The STMA frame provided by the embedded software is described below :

Frame to verify if an IT occurs

Command Frames		Answer Frames	
0xE4	**R_IT_TEST**	**0xAD**	**V_IT_TEST**
0x??	**IT number**	0x0?	result

Frame parameters for command

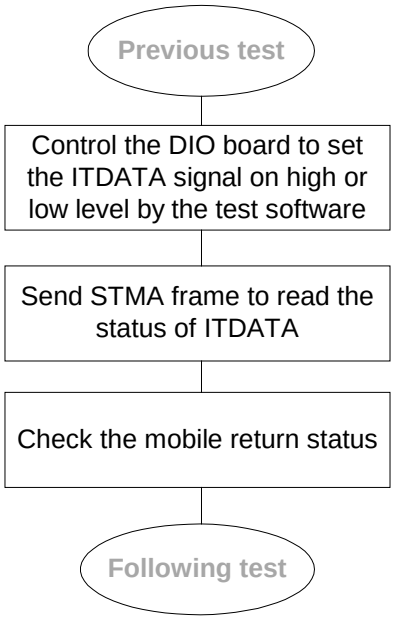
0x13	IT_DATA
0x0C	IT_I2C
0x08	IT_CHARGER
0x02	CINT
0x01	IT OUI KEY

Frame parameters for answer:

0x00	IT has not occured
0x01	IT has occured

Application note for software test tools

The following algorithm shows how these frames could be used by a software tool to perform GENERIC IT testing



4.3.20 Spare IO

These signals are available to manage additional interfaces or external devices. They represent three digital signals. Refer to the SCT TMO MASV2 SPEC- MO130 Application Note for more information.

As it's specified in the "Application Note", these signals could be used to control any accessory on condition that electrical specifications are compliant. The embedded software provided to support the manufacturing test includes some basic functionalities to allow this interface test.

In case these IOs are specified as outputs on the final product, refer to the schematic described on section 4.3.17(ON*) to have an example of the test bench.

Software resources provided

The STMA frame provided by the embedded software is described below :

Frame to test spare IO

Command Frames		Answer Frames	
0xE5	**W_CMDSW**	**0xE5**	**A_CMDSW**
0x??	command	0x??	command

Frame parameters for command byte:

Bit		CMDSW1 & 2 to be used as output	CMDSW1 & 2 to be used as input
7	R/W	1 -> set CDMSW1 & 2 as output	0 -> set CDMSW1 & 2 as input
6	Not used		
5	Not used		
4	Not used		
3	Not used		
2	Not used		
1	CMDSW2	Value of CMDSW2 to write	
0	CMDSW1	Value of CMDSW1 to write	

Frame parameters for answer byte:

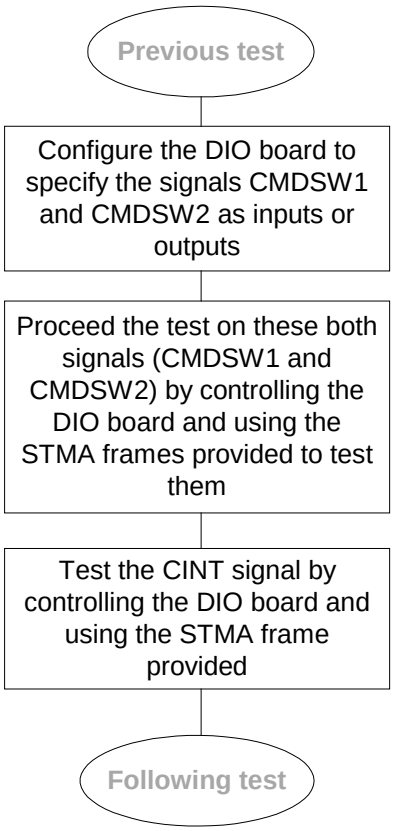
Bit		CMDSW1 & 2 to be used as output	CMDSW1 & 2 to be used as input
7	R/W	Identical to command	Identical to command
6	Not used		
5	Not used		
4	Not used		
3	Not used		
2	Not used		
1	CMDSW2	Identical to command	Value of CMDSW2 read
0	CMDSW1	Identical to command	Value of CMDSW1 read

These IO are initialized as output.

Please note that when you send in the command frame the order to switch a signal from output to input, you will receive the answer frame one second later. This delay will appear only on the switch frame. Subsequent reading frames of the same entry will be as usual.

Application note for software test tools

To proceed the test on this signal, follow this algorithm.



4.3.21 SPI bus

TBD

4.3.22 ADC / DAC

4.3.22.1 ADC

Software resources provided

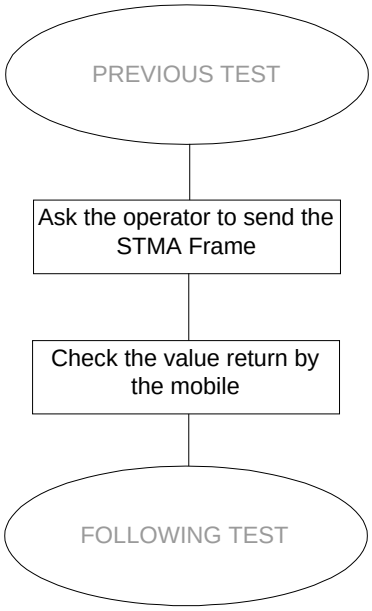
The embedded software used for the manufacturing tests provides the following frame for ADC testing.

Frame to test ADC

Command Frames	Answer Frames
0xDC W_ADC	**0xDC A_ADC**
	0x?? MSB measure
	0x?? LSB measure

Frame parameters for answer:
MSB measure and LSB measure are the resistance measured on ADC1 entry, in multiple of 10 ohms. For example: 0x01 0x40 = 3.2 Kohms.

Application Note for software test tools



4.3.22.2 DAC

Software resources provided

The embedded software used for the manufacturing tests provides the following frame for DAC testing.

Frame to test DAC

Command Frames		Answer Frames	
0xE0	**W_DAC**	**0xE0**	**A_DAC**
0x??	MSB order	0x??	MSB order
0x??	LSB order	0x??	LSB order

Frame parameters for command:

MSB order :

	control						value to convert	
Bit	7	6	5	4	3	2	1	0
	ONDAC	OFFDAC	WRITEDAC	X	X	X	DAC9	DAC8

ONDAC : 1 => power on DAC
OFFDAC: 1 => power down DAC
WRITE DAC : 1 => convert given value in STMA frame

DAC0 to DAC9 : Value to convert

LSB order :

	value to convert							
Bit	7	6	5	4	3	2	1	0
	DAC7	DAC6	DAC5	DAC4	DAC3	DAC2	DAC1	DAC0

Frame parameters for answer:

MSB order :

	control			DAC state			value to convert	
Bit	7	6	5	4	3	2	1	0
	ONDAC	OFFDAC	WRITEDAC	DACISON	X	X	DAC9	DAC8

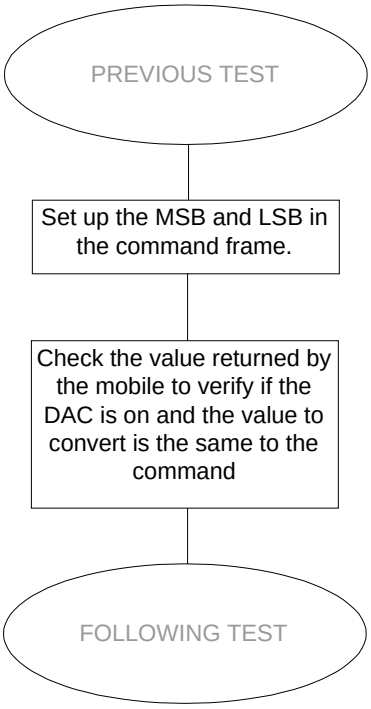
ONDAC : 1 => power on DAC
OFFDAC: 1 => power down DAC
WRITE DAC : 1 => convert given value in STMA frame
DACISON : DAC state. Only present in answer, no use in command.
1 : DAC is on. Conversion is possible.
0 : DAC is off. No conversion possible.

DAC0 to DAC9 : Value to convert

LSB order :

	value to convert							
Bit	7	6	5	4	3	2	1	0
	DAC7	DAC6	DAC5	DAC4	DAC3	DAC2	DAC1	DAC0

Application Note for software test tools



4.3.23 RF

There is no automatic camp, so it 's necessary to launch a fast camp frame.

Two kinds of frame are needed to make RF test:

- a frame to launch fast camp sequence (actually two frames, one for GSM the other for DCS).
- a frame to launch a phone call

After that, all states can be set and measurements can be done using a Radio Communication Tester driven by running dedicated software in the PC.

4.3.23.1 Fast Camp GSM

Software resources provided

The embedded software used for the manufacturing tests provides the following frame for radio testing.

Frame to proceed fast camp in GSM mode

Command Frames		Answer Frames	
0x4D	**W_PARAM_DRV**	**0x47**	**A_PARAM_DRV**
0x??	type	0x??	type
0x??	rf	0x??	rf
0x??	rxlev	0x??	rxlev
0x??	pl	0x??	pl

Frame parameters for command:

type	0x15
rf	channel
rxlev	Rx level
pl	Power level

Frame parameters for answer (identical to command):

type	0x15
rf	channel
rxlev	Rx level
pl	Power level

4.3.23.2 Fast Camp DCS

Software resources provided

The embedded software used for the manufacturing tests provides the following frame for radio testing.

Frame to proceed fast camp in DCS mode

Command Frames		Answer Frames	
0x62	**W_PARAM_DRV2**	**0x5C**	**A_PARAM_DRV2**
0x??	type	0x??	type
0x??	rfHB	0x??	rfHB
0x??	rfLB	0x??	rfLB
0x??	rxlev	0x??	rxlev
0x??	pl	0x??	pl

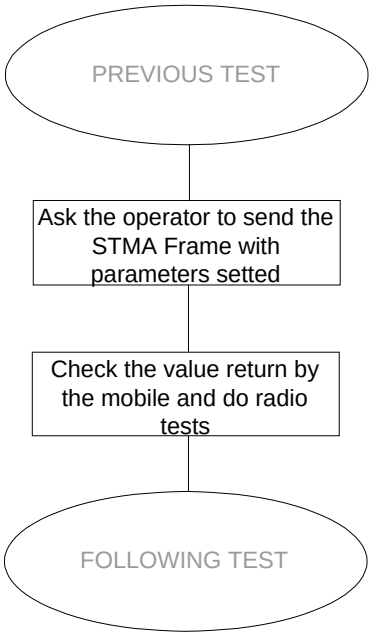
Frame parameters for command:

type	0x15
rfHB	Channel (most significant byte)
rfLB	Channel (less significant byte)
rxlev	Rx level
pl	Power level

Frame parameters for answer (identical to command):

type	0x15
rfHB	Channel (most significant byte)
rfLB	Channel (less significant byte)
rxlev	Rx level
pl	Power level

Application Note for software test tools



4.3.23.3 Phone call

Software resources provided

The embedded software used for the manufacturing tests provides the following frame for radio testing.

All parameters of this frame will be completed by WONU on next specification release. The values hereafter are given for reference only.

Frame to proceed Radio test

Command Frames		Answer Frames	
0x53	**W_OUTCOM**	**0x4D**	**A_OUTCOM**
0x??	type	0x??	answer code
0x??	lg_name		
0x??	name[0]		
0x??	name[1]		
0x??	name[…]		
0x??	name[lg_name-1]		
0x??	lg_num		
0x??	num[0]		
0x??	num[1]		
0x??	num[…]		
0x??	num[lg_num-1]		
0x??	ton		
0x??	npi		

Frame parameters for command:

type	Type of call: 0x00=voice, 0x01=data
lg_name	Length of name field. If lg_name = 0x00,no name[.] fields
name[.]	Only if lg_name > 0
lg_num	Length of num field (always >0)
num[.]	Call number field
ton	Type of phone number (0x10= international, 0x20 = national, TBD)
npi	Type of network (0X01 = ISDN/tel, TBD)

Frame parameters for answer:

answer code	Phone call result: 0x00=accepted, 0x01=refused

Exemple : URGENCY CALL : « EMERGENCY » 112

COMMAND	0215d20500530009454D455247454E4359033131322001C303	
	53	: W_OUTCOM
	00	: Voice call
	09	: Length name
	45	: «E»
	4D	: «M»
	45	: «E»
	52	: «R»
	47	: «G»
	45	: «E»
	4E	: «N»
	43	: «C»
	59	: «Y»
	03	: Length number
	31	: 1
	31	: 1
	32	: 2
	20	: (TON) National number
	01	: (NPI) Type of network ISDN/Tel

Application note for software test tools

TBD

4.3.23.4 Call release

Software resources provided

The embedded software can receive a simulation of a key press with an STMA frame. Use the hang key to release the call.

Frame to release call

Command Frames		Answer Frames	
0xD9	**W_KEY_SIMULATION**	**0xD9**	**A_KEY_SIMULATION**
0x03	press and release key	0x03	press and release key
0x04	hang on key	0x04	hang on key

4.3.23.5 Check connection to network

Software resources provided

Frame to check connection to network

Command Frames		Answer Frames	
0x68	**R_SIMCOMCCCH**	**0x62**	**V_SIMCOMCCCH**
		0x??	charger status & SIM status
		0x??	com status

Frame parameters for answer (charger and SIM):

0x00	no charger	no SIM

0x01	no charger	SIM inserted
0x10	charger connected	no SIM
0x11	charger connected	SIM inserted

Frame parameters for answer (Com Status):

0x00	No campment
0x01	Campment
0x02	Campment in emergency
0x03	In communication

4.3.24 VBAT

Correct Vbat reading is necessary to optimize battery charge. An initial adjustment is made when module is fabricated. It can be necessary to adjust it a bit more when module is integrated in its final environment, to take into account additional resistances.

Software resources provided

Frame to read Vbat

Command Frames		Answer Frames	
0xE3	**R_VBAT**	**0xE3**	**V_VBAT**
0x??	command	0x??	vbat read MSB
		0x??	vbat read LSB

Frame parameters for command:

Command	0 : measure raw Vbat, without adjustment 1 : measure adjusted Vbat

Frame parameters for answer:

Vbat read	Vbat read in mV 00 00 => error while measuring

Frame to auto-adjust Vbat

Command Frames		Answer Frames	
0xEC	**W_VBAT**	**0xEC**	**A_VBAT**
0x??	command	0x??	error code
0x??	VBAT applied MSB		
0x??	VBAT applied LSB		

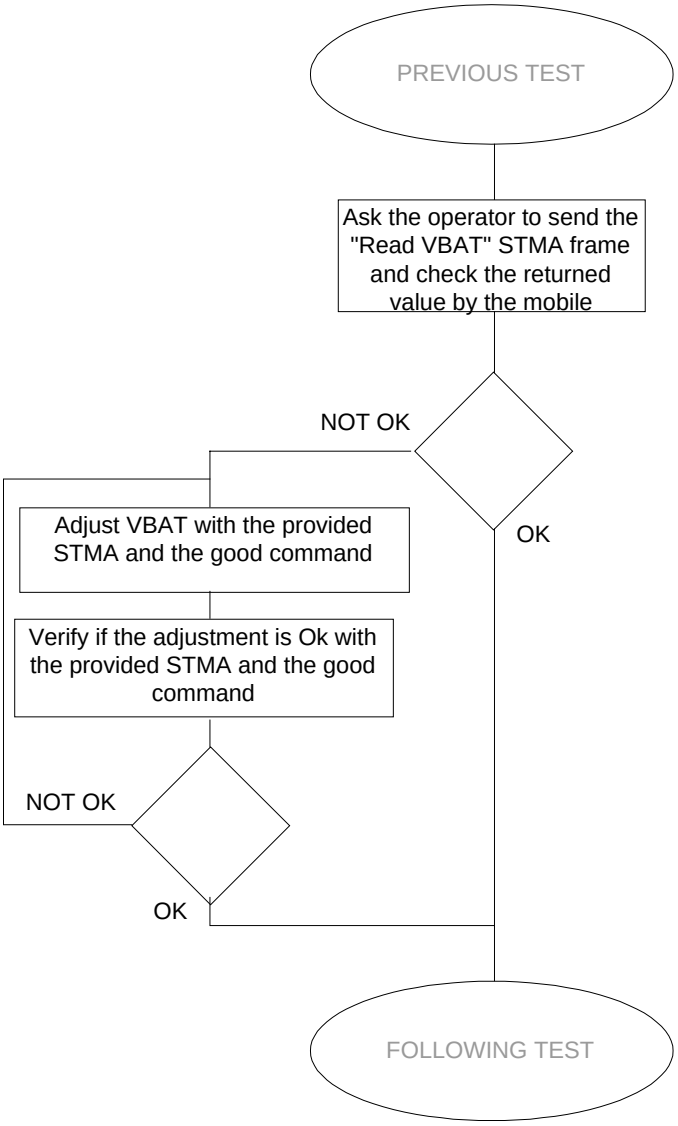
Frame parameters for command:

VBAT applied	VBAT applied by tester on mobile
Command	0x00 for auto-adjustment 0x01 to verify adjustment

Frame parameters for answer:

Error code For command = 0	0x01 ok 0x02 incoherent adjustment error 0x03 adjustment writing error
Error code For command = 1	0x80 command unknown 0x81 adjustment reading error 0x82 if difference between vbat applied and vbat read >120 or <-120

Application note for software test tools



4.3.25 Reset module

Software resources provided

This frame is used to reset module. It doesn't have an answer.

Frame to reset module

Command Frames	Answer Frames
0x89 W_RESET	

4.4 GENERAL PURPOSE FRAMES

4.4.1 Reading of Software release

Software resources provided

This frame is used to read the software version.

Frame to read Software release

Command Frames	Answer Frames	
0x23 R_VER	**0x35**	**V_VER**
	0x??	1ST byte of first string
	…	
	0x??	last byte of first string
	0x00	end of string character
	0x??	1ST byte of second string
	…	
	0x??	last byte of second string
	0x00	end of string character

Frame parameters for answer:

First string	Control string (for WONU internal used)
Second string	Software release reference

4.4.2 Reading Module serial number

Software resources provided

This frame is used to read the serial number of the module.

Frame to read module serial number

Command Frames		Answer Frames	
0xD8	**R_SERIAL_NUMBER**	**0xD8**	**V_SERIAL_NUMBER**
		0x??	1ST byte of serial number
		…	
		0x??	8th byte of serial number

Frame parameters for answer:

Serial Number	Serial number in decimal values (8 bytes)

4.4.3 Client software version

Software resources provided

This frame is used to read the client software version. This parameter is set by the customer thanks to the document SCT TMO MOD SPEC 0053 A - MO130 Setting Tool User Manual in order to know what kind of parameters there are in the client version.

This frame is not available yet because this parameter is not implemented in the "MO130 Setting Tool" soft, for the moment.

Frame to read or write client software version

Command Frames		Answer Frames	
0xF0	**W_CLIENT_VERSION**	**0xF0**	**A_CLIENT_VERSION**
0xAA	order to read or write	0xAA	order to read or write
0x??	1ST byte of client software version	0x??	1ST byte of client software version
…		…	
0x??	16th byte of client software version	0x??	16th byte of client software version

Frame parameters for command:

0xAA	0x00 to read 0x01 to write
Client software version	Client software version (16 bytes)

Frame parameters for answer (identical to command):

0xAA	0x00 to read 0x01 to write
Client software version	Client software version (16 bytes)

4.4.4 Reading CIE+Combined code

Software resources provided

This frame is used to read the CIE Code and combined of the module (it 's a WONU reference for traceability of the module)

Frame to read CIE+combined code

Command Frames		Answer Frames	
0xE8	**R_CIE_CODE**	**0xE8**	**V_CIE_CODE**
		0x??	1ST byte of CIE code
		…	
		0x??	6th byte of CIE code
		0x??	1st byte of combined code (7th of 28 bytes)
		…	
		0x??	6th byte of combined code (12th of 28 bytes)

Frame parameters for answer:

CIE+Combined code	CIE code+Combined (28 bytes)

4.4.5 Writing IMEI

Software resources provided

This frame is used to write IMEI code.

Frame to write IMEI code

Command Frames		Answer Frames	
0xDE	**W_IMEI**	**0xDE**	**A_IMEI**
0x??	1ST byte of IMEI code	0x??	1ST byte of IMEI code
…		…	
0x??	8th byte of IMEI code	0x??	8th byte of IMEI code

Frame parameters for command:

IMEI code	IMEI code in decimal value (8 bytes)

Frame parameters for answer (identical to command):

IMEI code	IMEI code written in decimal value (8 bytes)

Warning:
The last byte has only unit digit (always **0** for ten digit). For example, with IMEI 950825350568396, the frame to write IMEI code is 0x95, 0x08, 0x25, 0x35, 0x05, 0x68, 0x39, 0x**0**6.

4.4.6 Reading IMEI

Software resources provided

This frame is used to read the IMEI code.

Frame to read IMEI code

Command Frames		Answer Frames	
0xD7	**R_IMEI**	**0xD7**	**V_IMEI**
		0x??	1ST byte of IMEI code
		…	
		0x??	8th byte of IMEI code

Frame parameters for answer:

IMEI code	IMEI code in decimal values (8 bytes)

Warning:
The last byte has only unit digit (always **0** for ten digit). For example, with IMEI 950825350568396, the frame answer for IMEI code is 0x95, 0x08, 0x25, 0x35, 0x05, 0x68, 0x39, 0x**0**6.

4.4.7 Hardware configuration

Software resources provided

Frame to configure hardware

Command Frames		Answer Frames	
0xE7	**W_HARD_CONFIG**	**0xE7**	**A_HARD_CONFIG**
0xAA	order to read or write	0xAA	order to read or write
0x??	1st byte	0x??	1st byte
0x??	2nd byte	0x??	2nd byte
…		…	
0x??	32th byte	0x??	32th byte

Frame parameters for command:

0xAA	0x00 to read 0x01 to write
1st byte Main LCD	0x00 : no LCD 0x01 : B&W SAMSUNG S6B0723, S6B0724 or S6B0725 0x02 : CSTN 256 colors Sharp (101 * 80) 0x03 : TFT 65k colors Hitachi HD66770 0x04 : TFT 65k colors Hitachi HD66773 0x05 : CSTN 256 colors Sharp (84*40)
2nd byte Sub LCD	0x00 : no LCD 0x01 : B&W SAMSUNG S6B0723, S6B0724 or S6B0725 0x02 : CSTN 256 colors Sharp (101 * 80) 0x03 : TFT 65k colors Hitachi HD66770 0x04 : TFT 65k colors Hitachi HD66773 0x05 : CSTN 256 colors Sharp (84*40)
3rd byte Backlight	Bit 3 : 1 = digital BACKEY 0 = analog BACKEY Bit 2 : 1 = BACKEY active at high state Bit 1 : 1 = CMDSW1 active at high state Bit 0 : 1 = BACKLCD active at high state Bit 4 to 7 : not used
4th to 32th byte	Not used

Frame parameters for answer (identical to command):

0xAA	0x00 to read 0x01 to write
1st byte Main LCD	0x00 : no LCD 0x01 : B&W SAMSUNG S6B0723, S6B0724 or S6B0725 0x02 : CSTN 256 colors Sharp (101 * 80) 0x03 : TFT 65k colors Hitachi HD66770 0x04 : TFT 65k colors Hitachi HD66773 0x05 : CSTN 256 colors Sharp (84*40)
2nd byte Sub LCD	0x00 : no LCD 0x01 : B&W SAMSUNG S6B0723, S6B0724 or S6B0725 0x02 : CSTN 256 colors Sharp (101 * 80) 0x03 : TFT 65k colors Hitachi HD66770 0x04 : TFT 65k colors Hitachi HD66773 0x05 : CSTN 256 colors Sharp (84*40)
3rd byte Backlight	Bit 3 : 1 = digital BACKEY 0 = analog BACKEY Bit 2 : 1 = BACKEY active at high state Bit 1 : 1 = CMDSW1 active at high state Bit 0 : 1 = BACKLCD active at high state Bit 4 to 7 : not used
4th to 32th byte	Not used

Warning: A reset frame (cf 4.3.25 Reset module), or an off/on on Vbat must be used after setting the hardware configuration for the values to be taken into account.

4.4.8 Simulation of a key pressed

Software resources provided

This frame is used to simulate a key pressed.

Frame to configure hardware

Command Frames		Answer Frames	
0xD9	**W_KEY_SIMULATION**	**0xD9**	**A_ KEY_SIMULATION**
0xaa	type of key pressed	0xaa	type of key pressed
0xbb	code of key	0xbb	code of key

Frame parameters for command:

Type of key pressed	**STMA code of the key pressed**	
Press	0	The key is pressed and maintained. This command can be repeated several times in following (like a cyclic key pressed)
Cyclic	1	The key is pressed and maintained to display all characters related to the key pressed (example: for the key 3 : character 3,D,E,F,d,e,f). Scrolling of a character is obtained after reception of the frame with cyclic type.
Release	2	The key is released. End of scrolling in progress. Pressed on another key is then possible
Press and Release	3	Key is pressed (type = pressed) then immediately released (type = released)

Key of the keyboard	STMA code Simulation of a key pressed
3	0
6	1
9	2
#	3
Lift up (green key)	4
Prog 3 (up right)	5
Down (arrow down)	6
Clear (arrow left)	7
Prog 2 (middle up or not present)	8
UP (arrow up)	9
1	10
4	11
7	12
*	13
OK (arrow right)	14
2	15
5	16
8	17
0	18
Prog 1 (up left)	19
hang up (red key)	20

5. TEST SEQUENCE TRACEABILITY

One field (ten bytes) is reserved for customer to manage traceability of its manufacturing process (for example, value of counter increased by process sequence at each stage of test passed)

Software resources provided

The STMA frame provided by the embedded software to manage writing and reading of that area is described below :

Command Frames		Answer Frames	
0xDF	**W_FAB_STATUS**	**0xDF**	**A_FAB_STATUS**
0xAA	order to read or write	0xAA	order to read or write
0x??	1ST byte	0x??	1ST byte
…		…	
0x??	32nd byte	0x??	32nd byte

Frame parameters for command:

0xAA	0x00 to read
	0x01 to write
32 bytes	32 traceability bytes to write (presence needed to read too, but content is not important in command frame)

Frame parameters for answer (identical to command):

0xAA	0x00 to read
	0x01 to write
32 bytes	32 traceability bytes written or read

6. MULTI DOWNLOADING PROCESS

WONU provides to its customers the possibilitie to build their own multi-downloading application in order to speed up the client software downloading into the mobile during manufacturing process.To this purpose, WONU strongly recommands to use a specific hardware configuration and provides an object library (.DLL).This section describes these 2 parts.

6.1 HARWARE CONFIGURATION

WONU strongly recommands to use the following configuration to build the multi-downloading application:

PC: Industrial PC (P III 600 Mhz 256Mb) , Win NT 4.0
Two ACKSYS PCI cards (8 RS232 output for each card)

In any case this recommandation is mandatory for the customer,customer has to built its own solution.

6.2 SOFTWARE LIBRARY

WONU provides a specific .DLL file in order to help customer to build their own multi-downloading application. The following section describes how to use this library and specifically how to use each function of this DLL.

6.2.1 How calling the function of the DllTel.dll for the multi-downloading Software.

In order to download a protected software (***.bcs) with an application as Telech, a call of two functions is needed. Those functions are described below:

- int **AppelTelechITNET** (char *namefile, int speed, int PortCom, int PortCom2, unsigned char *ExError, HWND hWnd, unsigned char Gui, int SpeedMob, int SpeedRestartMob, int Language)

- int **MsgErreur** (unsigned char *ExError, char *Msg)

6.2.1.1 Description of AppelTelechITNET function

Namefile: complete pathname directory of the soft to download (ex:C:\software\filetoload_3.bcs).

Speed: maximal download speed (ex: 115200 for system base card, 400000 for ACKSYS card)

PortCom: (ex:1,2,3,…): number of the communication port 1.

PortCom2: number of the communication port 2. This value is always null because there is no double port downloading.

ExError: error structure returned by this function (coded on 10 bytes).

HWnd: handle of your windows application. The dll file uses it to show its own messages about the downloading operation (**it should not be null**).

Gui: select the output window .It should initialised to 0 because you don't use the internal window from the downloading dll.

SpeedMob: speed communication with the mobile (before calling this function you should make a loop to send several times in a particular speed the STMA frame which asks the version number of software. If there is no answer then change the speed in order to find the good one).

SpeedRestartMob: speed communication with the mobile after the boot of mobile (same as SpeedMob)

Language: language selection (0 for english, 1 for french).

6.2.1.2 Description of MsgErreur function

ExError: error structure to translate (coded on 10 bytes).

Msg: explicit error message after translation.

6.2.1.3 Error Windows Messages sent by DLLTel.dll file

***Message WM_OPERATION**

Value: WM_USER+3

Param: (int) positive number from 0 to 100. It displays the evolution of downloading.

*WPARAM : it is an integer parameter, indicating which port number is used.

***Message WM_RETRO_INFO**

Value: WM_USER+4

Param: string indicating the hardware configuration about the mobile.

***Message WM_RETROFITE**

Value: WM_USER+2

Param: string indicating the actual action.

6.2.1.4 Downloading Sequence

- Call the AppelTelechITNET function.
- Call the MsgErreur function.

Do no test the returned value from AppelTelechITNET function.

Always call the MsgErreur function and use ExError value form the last function. Test the returned value by MsgErreur. If it is different to 0, that means there is an error. You have to display Msg returned by MsgErreur to have the error explaination.

Note: an example of the call to those functions is in the TelechReduit.zip software, provided by WONU.

It is strongly recommended to have an application, which launches the DllTel.dll file, for one serial port.

6.3 PERFORMANCES OF MULTI-DOWNLOADING PROCESS

- Hardware configuration

PC: Industrial PC (P III 600 Mhz 256Mb)

Mobiles: X62-V5303

Serial card: - One PCI card with 16 RS232 output.

- Two ACKSYS PCI cards (8 RS232 output for each card)

- Software configuration

PC: Win NT 4.0.

Mobiles: V111 securised bootloader (version allows zipped bcs files)

- Test

Downloading a securised software on 16 mobiles with Telech application launched 16 times. The size of this software was about 5 Mb (4Mb in flash1 and 1 Mb in flash 2).

- Results:

With PCI card 16 RS232 output

- Speed :1*115200 Bds. Soft format: not zipped. Average of CPU work: 75%. Downloading time: 9'25''. Event: No Error.

- Speed :1*115200 Bds. Soft format: zipped. Average of CPU work: 100%. Downloading time: 7'24''. Event: Error with some mobiles.

With ACKSYS card

- Speed: 1*400000 Bds. Soft format: not zipped. Average of CPU work: 60%. Downloading time: 3'45''. Event: No error.

- Speed: 1*400000 Bds. Soft format: zipped. Average of CPU work: 100%. Downloading time: 3'04''. Event: Error with some mobiles.

- Speed: 1*200000 Bds. Soft format: not zipped. Average of CPU work: 25%. Downloading time: 5'46''. Event: No error.

- Speed: 1*200000 Bds. Soft format: zipped. Average of CPU work: 60%. Downloading time: 4'50''. Event: Error with some mobiles.

- Speed: 1*115200 Bds. Soft format: not zipped. Average of CPU work: 18%. Downloading time: 8'53''. Event: No error.

- Speed: 1*115200 Bds. Soft format: zipped. Average of CPU work: 100%. Downloading time: 6'48''. Event: No error.

Note: The retrofit dll memorises the software in the RAM memory before downloading in the mobile. In this case, it has used 16*5 Mb in RAM.

Conclusion:

-The PCI card with 16 RS232 output only works with the configuration: 1*115200 Bds, not zipped format, downloading time: 9'25''.

- The best configuration with the ACKSYS card is: 1*400000 Bds, not zipped format, downloading time: 3'45''.

Those tests show that downloading zipped files discreases downloading time but increases errors during the downloading operation. A 2Ghz computer (or more) would be a better solution. An ACKSYS (RS422) card will be better too because it allows to have a downloading speed of 1*800000 Bds.

1. Battery Level Adjustment

To clarify my answer on VBAT auto-adjustment procedure, you can try to reproduce the example given hereafter:

1)Auto adjustment phase
- external power supply set to 4169mV
- W_BAT command frame for auto-adjustment : 02 07 D2 05 00 EC 00 10 49 67 03
(00 = auto adjustment / 10 49 = 4169mV in hexadecimal format)
- W_BAT answer frame : 02 05 D2 05 01 EC 01 3C 03(01 : ok)

2)Verification phase
- external power supply set to 3612mV
- W_BAT command frame for verification : 02 07 D2 05 00 EC 01 0E 1C 2D 03
(01 = verification / 0E 1C = 3612mV in hexadecimal format)
- W_BAT answer frame : 02 05 D2 05 01 EC FC C1 03
(FC => - (FF- FC) mv = -3 mv: there is a gap of - 3mv between the value
sent by the frame and the value measured by the module, so the auto-adjustment is OK)
** Remark:
Rule to calculate the voltage value in mv with the hexa value returns by
W_BAT answer frame:
if MSB = 0, value in mv = + VALUE hexa (from 0 to 7F: 0 to +127)
if MSB = 0, value in mv = -(FF -VALUE hexa) (from 80 to FF: -127 to 0)

2. IMEI Writing

Concerning the problem of writing IMEI, please check that the STMA Frame has the following format: For the example of following IMEI number: 950825350568396 , you have to
send the following STMA Frame:
020CD20500DE5F081923054427060A03
The reading IMEI STMA frame must return the following format:
020CD20501D75F081923054427060203

The red string is the IMEI number (in Hexadecimal format).

3. RF CAMP ON Test

the beginning of the current week were holidays in France: I have only read your e-mail this morning.
The frame "W_PARAM_DRV" (0x4D) is only usable in GSM mode because the channel is coded on one byte only.
The frame "W_PARAM_DRV2" (0x62) is usable in both E-GSM and DCS mode because the channels are coded on two bytes. So, use that one in your case.
For the coding of the RF channel, you are right (RfHB = 0x03 ; RfLB=0xcf)
For the Rx Level, it is an information that we send to the module to inform it what power level (from the Network simulator) is sent to it. So when you apply to the module -102dBm, you

have to set Rx Level at 0x08 (GSM rule:110+Level Tester=Rx Level). In your example we have: Rx Level= 110+(-102)= 8. To conclude, this value allows the module to camp quickly without any measurements of the level applied.
The power level is the value of Pl (normalised value in GSM representing the power class - 5 to 19 for GSM and 0 to 15 for DCS) with the following convention:

For GSM:	For DCS:
Pl = 5 ==> power level = 33dBm	Pl = 0 ==> power level = 30dBm
Pl = 6 ==> power level = 31dBm	Pl = 1 ==> power level = 28dBm
Pl = 7 ==> power level = 29dBm	Pl = 2 ==> power level = 26dBm
Pl = 8 ==> power level = 27dBm	Pl = 3 ==> power level = 24dBm
Pl = 9 ==> power level = 25dBm	Pl = 4 ==> power level = 22dBm
Pl = 10 ==> power level = 23dBm	Pl = 5 ==> power level = 20dBm
Pl = 11 ==> power level = 21dBm	Pl = 6 ==> power level = 18dBm
Pl = 12 ==> power level = 19dBm	Pl = 7 ==> power level = 16dBm
Pl = 13 ==> power level = 17dBm	Pl = 8 ==> power level = 14dBm
Pl = 14 ==> power level = 15dBm	Pl = 9 ==> power level = 12dBm
Pl = 15 ==> power level = 13dBm	Pl = 10 ==> power level = 10dBm
Pl = 16 ==> power level = 11dBm	Pl = 11 ==> power level = 8dBm
Pl = 17 ==> power level = 9dBm	Pl = 12 ==> power level = 6dBm
Pl = 18 ==> power level = 7dBm	Pl = 13 ==> power level = 4dBm
Pl = 19 ==> power level = 5dBm	Pl = 14 ==> power level = 2dBm
	Pl = 15 ==> power level = 0dBm

So, to sum up, in your case, the right frame is given below:

0x62 W_PARAM_DRV2
0x15 type
0x03 RfHB (channel 975)
0xcf RfLB
0x08 Rx Level (-102 dBm)
0x06 Pl (31dBm instead of 30dBm in GSM mode)