FreeCalypso Tango Module Integration Guide

Version 1.2, last edited 2020/09/10

1. TR-800 hardware notes

This chapter provides some important notes about TR-800 physical hardware; these notes remain equally applicable whether you use TR-800 modules in their unmodified form from iWOW or if you use FreeCalypso Tango rebranded version.

1.1. Principal components inside the module

The TR-800 module internally consists of the following 6 principal components:

Component	Function	Notes
TI D751749ZHH	Calypso DBB	Calypso Lite, 256 KiB of IRAM
TI TWL3025BGGM	Iota ABB	
TI TRF6151CJ	GSM RF transceiver	
RFMD RF3133	GSM RF PA	
Epcos D1016	Quadband RF FEM	Called M034 in TI Leonardo docs
Spansion S71PL064JA0	Memory IC	8 MiB of flash, 2 MiB of XRAM

The Calypso chip's nIBOOT strapping pin is tied to GND, thus the Calypso boot ROM is always enabled, meaning that flashing access is always allowed and there is no possibility of bricking the module.

1.2. Main interface connector

1.2.1. Physical connector part

The connector part on the TR-800 module itself is Harwin M402F2-8005; the official mating part to be used on customer application boards is Harwin M402M1-8005, but that part is no longer readily available. However, an equivalent substitute connector part has been found: Panasonic AXK6F80347YG. The new part is equivalent in form, fit and function and is used on FreeCalypso Caramel2 boards.

1.2.2. Interface signals

Aside from the two power rails VBAT and V-IO and the common ground GND, every interface signal brought out on the 80-pin connector internally makes a direct connection to some Calypso or Iota ball pad, without going through any intermediate circuits added by iWOW. (VBAT is connected to 3 power consumers Iota, Rita and the PA; V-IO is a regulated power rail produced by Iota and used by Calypso and Rita.)

Some of the interface signal descriptions given in iWOW's TR-800 Product Technical Specifications document are incorrect or misleading, or could benefit from better clarification; this section provides the necessary factual corrections.

1.2.2.1. Backup power supply

Iota VBACKUP provision which TR-800 brings out on interface pin 6 is described quite well in TI's chipset documents, but iWOW's description is slightly misleading. This backup power supply is needed in order to maintain RTC date and time only when the main battery (VBAT) is completely removed; if VBAT remains present, then RTC maintains the time of day even when the chipset is in the OFF state (as defined in the chipset documents) *without* needing a backup battery.

1.2.2.2. PWON control

Interface pin 18 is wired to Iota PWON. iWOW's documentation describes it as ''ON/OFF Control'', but the physical hardware reality is that only the Switch-ON function is performed by the hardware, whereas all Switch-OFF functions (including the possibility of using the same PWON signal to request a Switch-OFF) are entirely up to firmware. (This fact is well-known to veteran Calypso chipset users, but may not be obvious to those reading iWOW's TR-800 documentation without prior Calypso chipset knowledge.)

PWON can only be driven with OC/OD drivers or with short-to-ground pushbutton switches, not with push-pull drivers! It is pulled up internally to VBAT.

1.2.2.3. RESET control

Interface pin 21 is wired to Iota nTESTRESET, also called TESTRSTz. Because these alternate names for the internal chipset signal are rather long, simply calling this signal RESET is perfectly acceptable — as long as its true nature is always remembered. The reset signal brought out on TR-800 is ''raw'' nTESTRESET — the TR-800 differs in this aspect from the competing GTM900 modules which bring out a transistor-translated version which we call XDS_RESET instead.

The workings of this reset signal are extensively covered in TI's chipset docs and in our Calypso-test-reset article in freecalypso-docs, but one point worth mentioning is that this special reset includes a built-in Switch-ON function, thus PWON can be considered optional when this RESET control is used.

Just like PWON, this RESET control signal can only be driven with OC/OD drivers, not the push-pull kind, and it is pulled up to Iota UPR (which in turn comes from VBAT or VBACKUP) inside the TR-800 module, following TI's Leonardo reference design. Unlike PWON, this RESET control signal is *not* internally debounced — thus it is best driven by other processors rather than pushbutton switches.

1.2.2.4. Analog audio interfaces

TR-800 brings out almost all of the Iota chip's analog audio interface pins: EAR and AUX outputs, primary MIC and AUX inputs. The only Iota analog audio interface which is not brought out is the third one intended for headset accessories, Iota signals beginning with HS.

TR-800 analog audio bringout consists solely of direct signal connections between Iota ball pads and interface connector pins — there are NO extra circuits inserted at the module level. (TR-800 differs in this aspect from the competing GTM900 modules which insert their own extra circuits into both primary and secondary audio input paths, extra circuits that may be undesirable in some applications.) The analog audio signal names given in iWOW's documentation correspond directly to actual Iota signal names, thus no extra explanation is deemed necessary.

1.2.2.5. Calypso GPIO and multifunction pins

Interface pins 28 through 35 are wired to Calypso GPIO and multifunction pins as follows:

Interface connector pin	Wired to Calypso signal	
	Main function (power-up default)	Alternate function
28	MCSI_CLK	GPIO11
29	GPIO1	—
30	GPIO2	—
31	GPIO3	—
32	MCSI_RXD	GPIO10
33	MCSI_FSYNCH	GPIO12
34	MCSI_TXD	GPIO9
35	nRESET_OUT	GPIO7

iWOW's documentation gives their ''logical'' GPIO numbers that are perfectly consecutive from 1 to 8; these are ''logical'' GPIO numbers defined by their firmware, but as one can see from the table above, they do

not correspond to physical Calypso GPIO numbers beyond the first 3. The correspondence between iWOW's logical GPIO numbers and Calypso physical ones for the upper 5 is as follows:

iWOW's logical GPIO number	Calypso physical GPIO number	Alternate peripheral function on the same pin
4	7	nRESET_OUT
5	9	MCSI_TXD
6	10	MCSI_RXD
7	11	MCSI_CLK
8	12	MCSI_FSYNCH

There are NO pull-up or pull-down resistors on any of these nets inside the TR-800 module, thus it is the system integrator's responsibility to ensure that each of these GPIO and multifunction pins is either driven externally or configured as a Calypso output, in order to avoid floating CMOS inputs.

1.2.2.6. UART interfaces

Calypso has two UARTs, called Modem and IrDA in chip docs, and both are brought out on TR-800. UART pinout listings are always somewhat confusing because signal directions are opposite between the Calypso chip's perspective and the perspective of the connected external host; the following table shows both perspectives:

Interface connector pin	Calypso UART signal	Host perspective (TR-800 as DCE)
49	RX_IRDA	TxD2
50	TX_IRDA	RxD2
51	RX_MODEM	TxD
52	TX_MODEM	RxD
53	RTS_MODEM	CTS
54	CTS_MODEM	RTS

There are no pull-up or pull-down resistors on any of the 3 signals that are connected to Calypso UART input pins, thus if any of them are not driven, it is the user's responsibility to pull them up or down externally, or tie them off if they are completely unused.

1.2.2.7. LED control output

There is a LED control output signal brought out on interface pin 55, and iWOW's documentation calls it LPG. However, this aspect of iWOW's documentation is factually incorrect: the actual Calypso signal wired to this interface pin is LT/PWL (ball L7) and not DSR_MODEM/LPG (ball D9). The actual signal that is brought out on TR-800 can be driven by Calypso using either the chip's LT output function or the chip's PWL output function; the latter is generally preferred. Thus the LED control output signal under consideration is really PWL and not LPG.

1.3. Unused Calypso signals

The following Calypso GPIO and multifunction pins are unused and left entirely unconnected (not brought out) inside the TR-800 module:

- GPIO0
- TSPDI/GPIO4
- BCLKX/GPIO6
- MCUEN1/GPIO8
- MCUEN2/GPIO13
- DSR_MODEM/LPG

Anyone who is going to write their own firmware to run on the Calypso inside the TR-800 needs to configure these GPIO and multifunction pins as dummy outputs in order to prevent floating CMOS inputs.

1.4. Antenna interface

The microcoaxial antenna connector on the TR-800 module is believed to be equivalent to Hirose U.FL. We currently use Sunridge microcoaxial cable assemblies terminated with their MCB2G plugs — this is the arrangement used on our Caramel2 development board. The other two antenna connection options provided by iWOW (contact pads on the side facing the application board and contact or soldering pads on the side facing outward) have not been explored by our team yet.

2. Integration guide for FC Tango users

This chapter is intended for application designers and system integrators designing higher-level systems that will incorporate a FreeCalypso Tango module as a component; this chapter is officially valid only for those users who buy FreeCalypso-branded Tango modules from Falconia Partners LLC, flashed with FC Tango firmware rather than iWOW's.

2.1. Two fundamental modes of application

Our FC Tango module product based on iWOW's TR-800 hardware piece supports two different classes of applications:

- The most common class of application can be described as a slave modem: in these applications the GSM/GPRS modem module is fully subservient to an external host, controlled via AT commands or some other conceptually similar interface. In these applications the modem module acts as a peripheral to some larger host system, providing GSM/GPRS functionality to the host, and it only does what the host commands it to do — it doesn't do anything on its own. The vast majority of cellular modem modules on the market including Calypso-based competitor GTM900 can function only in this ''slave'' mode, but for FC Tango this mode of application is just one of two possibilities.
- Using the rich set of additional Calypso and Iota interface signals brought out on TR-800 hardware, our FC Tango module product can also support a different class of applications where the Calypso chip inside the module (running custom firmware) can act a system master, controlling other peripherals via the available interfaces of parallel microprocessor bus, I2C, UART and GPIO. Hybrid applications are also possible where Tango acts as a UART-controlled peripheral to an external host in some aspects, yet at the same time also uses its additional Calypso and Iota interfaces to control or monitor other peripherals.

2.2. Common interfaces for all applications

2.2.1. Power and ground

Recommendations given in iWOW's TR-800 Product Technical Specifications document section 2.2 are equally applicable to our FC Tango derivative product, hence we advise our users to follow those guidelines. It is particularly important to note that all 4 legs that protrude from the module's metal shield MUST be soldered to plated through holes or plated slots in the carrier board on which the module is mounted — these ground legs provide the return path for the power supply current. The main 80-pin system interface connector provides 5 pins for VBAT but only one GND pin, and that one GND pin is not sufficient to carry power supply return current during GSM Tx bursts.

If you are going to use a fixed voltage supply to power your Tango module, as opposed to a battery that is subject to charge and discharge cycles, it is important to note that the chipset inside the module uses only LDO regulators and no switching converters, hence higher input voltages will increase internal heat dissipation and potentially reduce hardware life expectancy. We recommend setting your supply voltage to somewhere between 3.5 and 3.8 V.

2.2.2. Backup power supply

In the vast majority of applications no backup battery supply is needed, and interface pin 6 (VBACKUP) should be left unconnected. A backup battery becomes useful **only** if your application meets both of the following conditions:

1. You use Calypso RTC to maintain time of day, and
2. You wish this RTC time of day to be maintained when the main battery is removed.

2.2.3. PWON and RESET controls

If your application is such that the Calypso processor inside Tango will be the top level system master, not controlled by any other processor (the arrangement found in classic cellular phone handsets), then PWON will need to be wired to a human-operated pushbutton switch whose other side is GND. RESET will typically not be used in such applications, although it should be brought out to whatever your internal development interface (firmware loading and debugging etc) happens to be.

On the other hand, if the Calypso processor and its firmware inside Tango are functionally subservient to some higher-level processor in your system, then it will be extremely helpful if you provide some way for your system host processor to control the RESET input to Tango. Remember that it needs to be driven with a dedicated OC or OD driver, **not** a direct connection to some other processor's GPIO output! Having Tango RESET input under your host processor's control means that you will always be able to regain control of the Calypso from any ''runaway'' code without having to remove and reapply VBAT power — obviously a highly useful feature.

Because Iota nTESTRESET (which is what Tango RESET input really is) includes a built-in Switch-ON function, if your system host processor is controlling this RESET line, then strictly speaking there is no need for PWON, and it would be acceptable to leave the PWON control pin unconnected. But if you have two GPIO outputs and two OC/OD drivers available (for example, using a 74LVC2G07 dual OD buffer IC from Nexperia), then it is best to put both PWON and RESET inputs under your host processor's control.

2.2.4. Calypso UARTs

Even if your application is one where the Calypso processor inside Tango is the top level system master and does not need any UART interfaces in normal operation, you will need to bring out at least one UART to your development interface so you can load and debug your firmware. OTOH, if your Tango module acts as a GSM/GPRS peripheral to some larger system, then you will need at least one UART to serve as the control interface.

If you are going to run our standard Tango modem firmware, the Modem UART provides a standard AT command interface complete with GSM 07.10 MUX and data capabilities, whereas the IrDA UART carries our firmware's debug trace interface called RVTMUX, supported by our FreeCalypso host tools. If your host system for which Tango will act as a GSM/GPRS modem peripheral has only one UART available for communicating with the GSM modem peripheral, we recommend that you connect the Modem UART (carrying standard AT commands) to your internal host, and connect the IrDA UART (carrying our debug interface) to some debug interface header or whatever is appropriate for your system, allowing external debug and development tools to be connected.

If whatever entity you end up connecting to the Modem UART has no RTS/CTS flow control capability, then Tango system interface pin 54 (Calypso CTS_MODEM, host RTS) should be tied to GND (do not leave it floating!), and pin 53 (Calypso RTS_MODEM, host CTS) can be left unconnected. On the other hand, if either of the two UART RxD (host TxD) lines may be left unconnected under some circumstances, then it needs to be pulled up (100 kΩ resistor recommended) to the Calypso chipset's V-IO rail which is brought out on system interface pin 9.

2.2.5. Calypso GPIOs 1 through 3

Our standard Tango modem firmware does not impose any restrictions on how these GPIOs may be used — instead we allow the desired I/O configuration to be programmed in a non-volatile configuration file in our flash file system (FFS). In the initial shipping state with no pin configuration file programmed, all 3 GPIOs are

left in their power-up default state of being inputs, so there will be no driver conflict regardless of how they are wired on any given customer's application board. If a GPIO needs to remain as input in your application, just leave the default configuration unchanged, or if a GPIO needs to be an output in your application, then you can program it to be so in the pin configuration file.

There exists one traditional set of functions for these 3 GPIO pins, although it is entirely up to you whether you wish to use these traditional GPIO functions or not:

GPIO	Function
1	RI output
2	DCD output
3	DTR input

If you like the traditional GPIO functions listed above and include such GPIO wiring in your application board design, you can program the pin configuration file with a standard configuration telling our firmware to use these classic functions. In this case GPIO3 acting as DTR input MUST NOT be left floating! Alternatively if you have no DTR input signal in your application, you can leave GPIO3 unconnected and program the pin configuration file to make it into a dummy output. The other two GPIOs are outputs in the traditional configuration, hence they can be left unconnected if they aren't needed.

2.2.6. MCSI/GPIO multifunction pins

Calypso MCSI is a 4-wire interface carrying a digital voice channel — this digital voice interface capability is a premium feature which sets FC Tango apart from the competition. The 4 pins which carry MCSI are functionally multiplexed between MCSI and GPIO, thus MCSI is available for those applications that need it, but is not imposed on other users.

2.2.6.1. Digital voice applications

If your application will use MCSI for digital voice, the function and direction of the 4 signals are fixed by the combination of Calypso hardware and standard Calypso DSP code. MCSI_TXD is a Calypso output from power-up, MCSI_RXD is always an input to Calypso, but the other two signals (MCSI_CLK and MCSI_FSYNCH) require special attention: they become outputs during active voice calls, putting out a 520 kHz clock and an 8 kHz frame sync, but during idle periods between calls the interface is turned off and the two signals in question go Hi-Z.

The implication for users is that you have to treat MCSI as a PCM master (your connected logic must act as a PCM slave, accepting the clock and frame sync from Calypso), but unless your connected logic has built-in pull-up or pull-down resistors, you will also need to put board-level pull resistors (we recommend 100 kΩ pull-down to GND) on MCSI_CLK and MCSI_FSYNCH nets to keep them from floating during idle periods.

2.2.6.2. Applications without digital voice

If your application won't use digital voice (if you use the analog voice interface or if your application does not use voice at all), then the pin configuration file in FFS should be programmed to switch these 4 multifunction pins from MCSI to GPIO. You can then use these GPIOs for your own custom functions, or you can leave all 4 MCSI/GPIO pins unconnected and program the pin configuration file to drive them as dummy outputs to prevent floating CMOS inputs.

2.2.7. Analog audio interfaces

If your application will use any of the provided analog audio interfaces, follow Iota chip documentation from TI. iWOW's audio design recommendations are also good.

If your application won't use analog audio (if you use the digital voice interface via MCSI instead, or if your application does not use voice at all), it is perfectly acceptable to leave all analog audio interface pins unconnected.

2.2.8. SIM socket interface

The 4 wires that connect between Tango system interface connector and the SIM socket are deemed to be self-explanatory. Calypso signal SIM_CD (card presence detection) is not brought out on TR-800 hardware — instead it is tied to V-IO inside the module like on Leonardo, FCDEV3B and Openmoko boards.

2.3. Basic modem applications

If you are going to run our standard Tango modem firmware for GSM and GPRS, or if you are going to be developing your own custom firmware, but your custom fw still acts as a ‘‘slave’’ modem fully subservient to control from an external host, not acting as a system master to any other peripherals, then all other Tango interfaces besides those detailed above should be left unconnected: they have no function in such applications.

All of these unused interfaces are module outputs, or inputs with built-in pull-ups, or analog signals that are designed to tolerate the no-connect state — thus no digital inputs will be left floating.

2.4. Custom firmware applications

If you are going to connect anything to the parallel microprocessor bus interface or to the I2C interface, you will need to write your own custom firmware code to control those peripherals; our standard Tango modem fw does not expect anything other than built-in flash and RAM on the microprocessor bus, and it does not configure I2C at all. You will also need to customize the firmware in order to collect any useful input from the keypad or ADC interfaces; we do have a driver for Calypso-driven battery charging, but it is meant to be customized, not a one size fits all.

2.4.1. Alternate uses for the keypad interface

If your custom application does not include an actual keypad, Calypso keypad interface can be repurposed for other uses with custom firmware: the 5 KBC pins can be used as general-purpose outputs, and the 5 KBR pins can be used as active-low interrupt inputs.