FreeCalypso Caramel2 Status Report and User's Guide

Version 1.1, last edited 2021/07/21

1. Caramel2 project initiative

In the early 2000s a Singapore company named iWOW Connections Pte Ltd produced a packaged Calypso modem module which they named TR-800, and they also produced a development and evaluation board for this module. Our FreeCalypso team discovered the existence of these modules and historical development boards at the end of 2019, and in the present time the availability situation is as follows: ''bare'' TR-800 modules are still available as a very large NOS surplus, but there is no comparable surplus of original iWOW development boards. Our team was able to obtain just two of those original development boards (iWOW DSK), and these two boards which we got may very well be the only ones left in the world.

However, the TR-800 module itself is very good: for many years prior to our discovery of its extremely obscure existence, we (FreeCalypso) had a strong desire to produce our own packaged Calypso module that would have been very similar to this TR-800 — but no one ever funded those ideas, hence they never came to fruition. But now that we have discovered the existence of iWOW-made TR-800 modules and given the availability of a large NOS surplus of them, the right course of action became clear: we are turning this TR-800 module into a bona fide FreeCalypso product by way of rebranding, and its new name is FC Tango.

In order to properly support our new FC Tango product based on iWOW TR-800, we need a development and evaluation board for it, similar to iWOW's DSK — but unlike ''bare'' TR-800 modules, those original DSK boards are unobtainium beyond the two which our core team scored. The safest course of action (as in the best chance of positive outcome) would be to produce a verbatim clone of iWOW's DSK board, at least as the first version — but we never found any design files for any version of that original board, and having only two of those boards, we cannot afford to sacrifice any to destructive reverse engineering. Therefore, we took the alternate route of producing our own functional semi-clone of iWOW's board with our own entirely new PCB layout, guided only by iWOW's schematics and general plan of component placement. The result is our FreeCalypso Caramel2 development board.

As part of transforming iWOW's DSK into FC Caramel2 at the schematic level, we also made some simplifications:

- We removed iWOW's RS-232 level shifters and DE9F connector interfaces, instead we bring out the two Calypso UARTs in their native LVCMOS form on a 10-pin header in the same pinout as implemented on FCDEV3B.
- We removed iWOW's power supply circuit (5V input and LDO regulator producing lower VBAT), instead our power input is ready-made VBAT via the same orange Weidmuller connector as used on previous TI and FreeCalypso development boards C-Sample, D-Sample, Leonardo and FCDEV3B.

1.1. Antenna interface

Most GSM MS development boards feature a female SMA connector for the antenna interface; iWOW's DSK follows this tradition, and so does Caramel2. However, given the way in which the antenna interface comes out of the TR-800 module on a microcoaxial connector, the design of this RF interface at the development board level becomes somewhat tricky, and the ultimately chosen approach is different between iWOW's original and C2:

- On iWOW's original DSK a piece of microcoaxial cable carries the RF interface from the TR-800 module to a presumably identical microcoaxial connector on the DSK motherboard, and then a short PCB trace (unknown if it was controlled for 50 Ω impedance or not) carries it to a vertical SMA connector.

- To avoid the extra cost of controlled impedance on our Caramel2 PCB, we took a different approach. We use a custom microcoaxial cable assembly made for us by Sunridge; on one end of this assembly a Sunridge MCB2G head mates with the microcoaxial connector on the TR-800 module, and the other end is our external SMA interface. The SMA connector piece has mounting legs that secure it to our motherboard PCB, but the RF signal path does *not* go through our PCB — instead the microcoaxial cable goes to the back of this SMA connector.

The result of this arrangement is that our Caramel2 board features an SMA connector for the antenna interface that hangs off the edge of the board, exactly the same as all previous TI and FreeCalypso development boards.

1.2. Analog audio interfaces

The choice of interface (or multiple interfaces) for analog audio on a GSM MS development board is always a tricky question, and there is no single universally correct answer that would ideally satisfy every possible use case. The approach implemented on our Caramel2 board consists of simply copying what iWOW did on theirs: the TR-800 module brings out two Iota audio channels (main and auxiliary), iWOW's DSK brings out each of these two audio channels to a 2.5 mm headset jack, and we do likewise.

The particular headset jack type chosen by iWOW is 4-wire TRRS, as opposed to the more common 3-wire TRS, and this headset interface which we've copied from iWOW exhibits the following key properties:

- Both main and auxiliary Iota audio outputs are differential, and our FreeCalypso headset interface (formerly iWOW's) brings this differential signal out natively, without giving up either leg of Iota differential output. Two pins on the TRRS jack are used for the differential output: Tip and Ring2. The headset earpiece (which needs to be a 32 Ω speaker) needs to be connected between these two pins as a bridge-tied load.
- iWOW's headset features an electret condenser microphone connected between Ring1 and Sleeve pins, with Ring1 being positive and Sleeve being ground. On the development board (both iWOW DSK and FC Caramel2) Sleeve is the common circuit ground, whereas Ring1 is connected to a microphone input circuit. The latter circuit has been copied unchanged from iWOW's DSK and can be studied from schematic drawings (both iWOW's and ours); it uses Iota MICBIAS output as the power source for the microphone and feeds AC-coupled audio input to Iota MICIN & MICIP for the main audio channel, or to Iota AUXI for the auxiliary audio channel.

So far the only physical headset that has been plugged into Caramel2 headset jacks is the original one from iWOW, the one that came with iWOW's DSK. The Mother's plan is to order a batch of new FreeCalypso headsets to be made for our community as a custom manufacture run, with a 32 Ω earpiece and an electret microphone connected as we require, but as of this writing, this plan has not been realized yet.

1.3. Status indicator LED

Our previous development board was FCDEV3B, and it includes one very useful feature which is an original FreeCalypso invention: a LED that shows the state of the chipset-internal ON_nOFF signal. The Calypso+Iota chipset has switched-on and switched-off states, and when a developer or an enthusiast tinkerer user works or plays with a Calypso board, it is highly useful to be able to see which state it is in. The most reliable way to observe this state is to monitor the internal ON_nOFF signal, which is what we do on FCDEV3B, but this internal signal is not brought out of the TR-800 module, nor is it brought out on any other historical commercial packaged module of this type that we know of.

When working with iWOW's DSK board prior to development of Caramel2, I (Mother Mychaela) really missed not having any way to tell reliably if the chipset is switched on or off. Replicating the ON_nOFF status LED from FCDEV3B on Caramel2 was not possible because that internal signal is inaccessible, but our C2 board features another trick aiming at a similar goal: the yellow ‘‘STATUS’’ LED lights when the Calypso chipset's 13 MHz clock (which is brought out on TR-800) is running, and is intended to be off when the chipset is either switched off or in deep sleep. See §2.2 for how this idea worked out in practice.

2. Status report on the outcome

FC Caramel2 boards were physically produced and made publicly available in the waning months of 2020. The result of various tests done since then is that most functions work as designed, but there are a few blemishes which need to be documented.

2.1. Noise from GSM Tx in the analog audio path

The simplest and most ‘‘natural’’ way to connect a GSM antenna to a Caramel2 board is to screw that antenna directly onto the female SMA tail hanging off the edge of the board. When the antenna is connected in this straightforward manner, everything works great until you plug a headset (currently iWOW’s original, for lack of our own) into the EAR/MIC jack and try to make a voice call. When you do make that test call, you will immediately hear very severe ‘‘buzz’’ getting mixed into the audio path; more careful tests reveal that this buzz gets mixed into both earpiece and microphone signal paths.

Further tests reveal that the interference being injected into the analog audio path comes from the radiating element of the antenna and not from any GSM RF path elements on the Caramel2 board itself. The workaround currently used by the Mother is to insert a long coaxial cable (the one I got is 1.8 m long, was sold as American 6 ft) between the antenna connector on the Caramel2 board and the actual antenna; with the radiating element of the antenna moved far away, voice call audio in the headset becomes perfectly clean.

This noise susceptibility blemish of Caramel2 analog audio is very disappointing, and it came very unexpected. Looking at iWOW’s schematics for their DSK, one can see a lot of extra filtering components in all of the analog audio paths, and all of these circuits have been reproduced verbatim on Caramel2. iWOW’s original DSK does **not** exhibit the same GSM Tx noise problem: I can screw the same antenna directly onto the vertical SMA connector on that board, plug the exact same headset into that board’s ‘‘PHONE AUD’’ jack, and the voice call audio is perfectly clean there. Therefore, the noisy audio problem on C2 defies my understanding.

2.2. LEDs lighting erratically from UART lines

Out of the 20 C2 boards produced in the first batch, some exhibit this erratic behaviour while others don’t — but the actual underlying problem is inherent to the Calypso chipset itself, or more precisely, potential for trouble arises whenever Calypso UART lines are connected to devices in a different power domain.

Just like FCDEV3B, Caramel2 brings out the two Calypso UARTs in their native LVCMOS form without any on-board level shifting or buffering, and the canonical mode of usage is that this LVCMOS DUART interface is connected to some USB to dual UART adapter: at FreeCalypso HQ we use our own DUART28, but the present problem remains exactly the same with any other adapter, such as any COTS FT2232x breakout board or Sysmocom mv-uart. The problem is this: what happens when the Calypso+Iota chipset is in its switched-off state, but the connected USB to DUART adapter has power (USB host connected) and puts out logic high levels on its TxD, RTS and DTR output lines?

What happens in this scenario is that power from the USB domain feeds into the Calypso+Iota chipset’s V-IO rail (which would otherwise be at 0 V with its supplying regulator turned off) through the clamping diodes inside the Calypso chip (these external UART outputs connect to Calypso inputs), and also through 100 kΩ resistors that were put in to pull up Calypso UART inputs to V-IO when they are left unconnected. The current through Calypso input clamping diodes is much greater, 1.27 mA per pin with our current DUART28 setup, compared to 28 μA through each 100 kΩ resistor, but the smaller current bypassing the diodes may play a role in some second-order effects.

Our DUART28 adapter features 2.2 kΩ series resistors on all of its outputs; these resistors were added for the specific purpose of limiting the partial power-down current through Calypso input clamping diodes to 1.27 mA per pin — contrast with 8 mA per pin put out by our competitor’s mv-uart! Unfortunately though, as we found out when we built our batch of C2 boards, this reduced current is still enough to cause erratic LED behaviour on some boards.

Why do Calypso-controlled LEDs sometimes light up erratically as a result of foreign-sourced current feeding into V-IO? We have no way of knowing exactly what happens inside the Calypso chip, but it is being subjected to abnormal operating conditions which it was never designed to handle (it was clearly designed with the assumption that its V-IO rail will never be powered by anything other than its ABB companion chip), and apparently under some conditions (with unit to unit differences being one of the variables) some of the outputs which are supposed to be at 0 V when the chipset is switched off start putting out enough voltage and/or current to turn on connected external BJTs or FETs controlling various peripherals.

Caramel2 features a red PWL LED controlled by Calypso LT/PWL output via a BJT with bias resistors, and a yellow STATUS LED controlled by Calypso CLK13M output via a MOSFET — see our published board schematics. Both LEDs are supposed to be off when the chipset is switched off — but we got 4 boards on which the red LED lights up erratically as a result of UART feeding, and one board on which the yellow LED behaves likewise.

The same problem has also been observed on some Motorola C139 phones, albeit very rarely. These phones have the same arrangement in that their headset jack serial port is connected to Calypso UART pins, and when an externally-powered UART is connected to the headset jack while the phone is off, a similar scenario occurs. On at least one phone I noticed a strange behavior in that a little bit of audible noise can be heard when the phone is off and a powered-on USB-serial cable is inserted into the headset jack; I surmise that the faint noise is coming from the phone's magnetic buzzer, which is slightly turned on by erratic output from Calypso on the BU/PWT line.

The Mother's current reasoning on the basis of our experiences with C2 is that whenever Calypso UART inputs may be coming from a different power domain, or even from a device in the same VBAT domain that may be on when the Calypso+Iota chipset is switched off, the correct solution is to insert LVC buffers in front of those Calypso inputs, with the LVC buffer IC's power supply being Calypso+Iota V-IO. The special quality of LVC buffers that matters in this case is that they are specifically designed for partial power-down applications with a very low Ioff spec. If we ever produce another successor board to FCDEV3B and Caramel2, we are going to implement this LVC buffer approach, and we recommend likewise to anyone else building their own development board around TR-800 or FC Tango modules.

2.3. PPD scenario 2

Now consider the opposite partial power-down scenario: UART lines are connected between Caramel2 and DUART28, Caramel2 is powered and switched on (firmware running), but there is no USB host connected. (This scenario only makes sense in the Luna configuration with LCD and keypad add-ons connected to Caramel2 expansion interface, running firmware built in the handset configuration.) If you are going to run your Caramel2 board in a configuration where this reverse PPD scenario is a possibility, your USB to DUART adapter needs to be FreeCalypso DUART28, not unbuffered FT2232x or CP2105: with either of the latter adapters there will be very high current flowing from Calypso UART outputs into powered-down I/O cells of the USB-serial IC. However, even with DUART28 there is a certain blemish in this PPD scenario; the details are described in the DUART28-PPD-surprise article in *freecalypso-docs* Hg repository, but the end effect is that Calypso UART inputs sense continuous low (break condition) instead of continuous high, and Calypso sleep modes are precluded as the result.

Under present conditions of ultra-low demand coupled with overabundant supply, the most sensible workaround is to issue two DUART28 adapter boards (as opposed to just one) to every FreeCalypso community member who may be operating a Caramel2+Luna configuration and who may thus encounter PPD scenario 2. One of these two DUART28 adapter boards is to remain unmodified, and the other is to be modified as explained in the DUART28-PPD-surprise article in *freecalypso-docs*.

3. User's guide for simple setups

This chapter describes a simple setup consisting of a Caramel2 board, a power supply, a DUART28 adapter and a Linux host for programming and control. The functionality achieved in this configuration is an AT-command-controlled modem similar to FCDEV3B, except that the new Tango-based board is quadband instead of triband. The more advanced Luna setup involving LCD and keypad add-ons is not covered in this guide: at the present time only FC core team members have the necessary LCD add-on piece, we don't have any more LCD modules of the original type, and the new lunalcd2 board featuring a different LCD module is still in development.

3.1. Hardware connections

Every Caramel2 board is supplied with a power adapter that goes from universal AC to 3.6 VDC; in most development scenarios such mains-powered operation is much more convenient than using an actual battery. We ship our power adapters already outfitted with orange Weidmuller connectors that plug directly into our Calypso boards; this connector can only go in one way, preventing reversed polarity.

If you received a DUART28 adapter board with your Caramel2 kit, you should have also received a 10-wire ribbon cable terminated with IDC socket connectors on both ends — this cable connects the UART lines (both Calypso UARTs) between the two boards. The shrouded header on the DUART28 board is keyed, preventing connector reversal, but the orientation of the overall cable is still up to the user. Our official recommendation is to orient the cable in such a way that on the DUART28 end the body of the cable will lie away from the adapter.

The corresponding DUART connection header on Caramel2 is **not** shrouded, hence the user is responsible for correct orientation. Please ensure that the polarizing tab on the cable-mounted IDC socket connector lines up with the corresponding mark on the PCB silk screen.

3.2. Indicator LEDs

Our Caramel2 board features 3 indicator LEDs: POWER (green), STATUS (yellow) and PWL (red). The green POWER LED lights whenever battery-emulating power is present at the orange power input connector, regardless of whether the Calypso+Iota chipset is switched on or off. The yellow STATUS LED behaves as follows:

- When the Calypso+Iota chipset is switched off, the yellow LED is supposed to be off — but see §2.2.
- When the Calypso is fully running and not in deep sleep, the yellow LED will be lit solidly.
- When Calypso firmware goes into and out of deep sleep, the yellow LED will flash: it will light when Calypso wakes up and go out when Calypso goes to sleep.

If you are running standard *tangomdm* firmware, as opposed to Luna, the red PWL LED should never light up on its own — instead you can control it with **AT@PWL** debug command. The key hardware feature is that the light intensity of this PWL LED is subject to smooth control via PWM: see Calypso-PWM-light article in *freecalypso-docs* Hg repository.

3.3. Pushbutton controls and standard firmware operation

Our Caramel2 board features two pushbutton switches: PWON and RESET. The operation of RESET is 100% hardware, whereas the operation of PWON is split between hardware and firmware: transition from OFF to ON state upon any press of the PWON button is a hardware function, whereas the reverse transition from ON to OFF upon a long press of the same button is a firmware function.

Starting from an OFF state, pressing either PWON or RESET will boot the Calypso chipset and cause our flashed firmware to run. However, if you are pressing these buttons with fingers, as opposed to driving boot control signals from another processor with OC/OD drivers (see §4.3), PWON is the preferred switch-on method: PWON is debounced in the Iota chip, RESET is not.

Once our standard firmware runs on boot, it provides a standard ASCII AT command interface on the primary UART and a highly vendor-specific binary packet debug and development interface (RVTMUX, formerly proprietary to TI, now owned and maintained by FreeCalypso) on the secondary UART. The particular variant of AT command interface that is implemented on TI and FreeCalypso modems is described in *FCDEV3B User's Manual*; if you wish to work with RVTMUX binary packet interface, you will need to download and install our FreeCalypso host tools package on your host computer.

If you accidentally booted the board and FC modem firmware by pressing PWON or RESET without a host computer connection, you can perform an orderly switch-off by pressing PWON, holding it down for 1 s, and then releasing it. However, if you do have a working host computer connection, this switch-off method is not recommended — you should implement a software-based shutdown sequence (**AT+CFUN=0**, then **AT@POFF**) instead.

3.4. Running fc-loadtool and related tools

FreeCalypso users are expected to be able to reflash their firmware as needed, and the tool for doing so is *fc-loadtool*. We also have a family of related tools: *fc-iram* and *fc-xram* for running various RAM-based standalone target utilities, *fc-simint* for operating on SIM cards sitting in the on-board SIM socket, and the set continues to grow. All of these tools operate via the Calypso chip's boot ROM, and they can operate on the Calypso target via either of the two UARTs. The choice of which UART to go through is generally arbitrary, but in FreeCalypso we lean toward a convention of using the secondary UART (the one used for RVTMUX) for such operations.

The canonical workflow for all of these tools is as follows:

1) Start with the Calypso+Iota chipset in its switched-off state — the yellow STATUS LED should be off.
2) Run *fc-loadtool* or other tool as appropriate, pointed to the chosen UART for the operation.
3) As the host tool keeps sending interrupt-boot beacons to the UART and waits for a Calypso boot ROM response, press PWON or RESET on the board.

See §4.3 for an alternative way that does not require having a human operator pressing buttons, but instead requires a custom patch to Linux kernel ftdi_sio driver.

When you end your *fc-loadtool* or *fc-simint* session, these tools clean up with a soft power-off, putting the Calypso+Iota chipset back in its switched-off state. Other workflows may require manual cleanup or RESET.

4. Advanced topics

4.1. Pin multiplexing

The Tango module that forms the core of Caramel2 brings out 8 Calypso GPIO and multifunction pins that are subject to functional multiplexing concerns; the subject is covered in detail in the FC Tango Module Integration Guide (§1.2.2.5) and in the Tango-pinmux article in *freecalypso-docs* Hg repository. All 8 of these signals are brought out to the expansion interface on Caramel2 (allowing arbitrary custom hardware interfacing), and two of them (GPIO2 and GPIO3) are also wired to the DUART interface header. The latter wiring assigns GPIO2 and GPIO3 to be DCD output and DTR input, respectively. GPIO3, MCSI_RXD, MCSI_CLK and MCSI_FSYNCH are outfitted with 100 kΩ pull-down resistors to GND; the other signals have no pull resistors in either direction.

The default */etc/tango-pinmux* programming with which Caramel2 board are shipped is **83 88 00 00**; this programming assigns GPIOs 1 through 3 to be RI output, DCD output and DTR input, respectively, whereas the 4 MCSI/GPIO pins are left in the power-up default MCSI configuration. The combination of on-board pull-down resistors and firmware configuration of GPIO1 and GPIO2 as outputs ensures that none of our GPIO and multifunction pins remain floating in the default setup.

4.2. Expansion interface

Our TR-800-based FC Tango module differs significantly from its contemporary competitors such as Huawei GTM900 in that it brings out many more Calypso and Iota signals than are needed for basic modem operation, and all of these extra signals are brought out on the 56-pin expansion interface header on Caramel2, matching iWOW's original DSK board. Most of these signals become useful only if you are going to write your own custom firmware or engage in low-level experimentation with the Calypso+Iota chipset; if you run standard *tangomdm* firmware, only a few of these signals perform useful functions, as covered in the following subsections.

4.2.1. RI output on GPIO1

Our default */etc/tango-pinmux* programming assigns Calypso GPIO1 to be RI modem control output: it is normally high, but goes low on incoming calls. However, this extra modem control signal is not included in the set carried between Caramel2 and DUART28 by the 10-wire ribbon cable: this interface goes back to FCDEV3B, and while we added DCD and DTR to it, using two pins which were unused and unconnected on FCDEV3B, there was no room to squeeze in RI. (It should also be noted that DCD and DTR go back to TI's C-Sample and D-Sample, but the addition of RI on GPIO1 was iWOW's invention.)

If you require or desire a working RI modem control signal, you will need to connect it with an extra jumper wire between GPIO1 pin on the expansion interface header and RI pin on the extra header on DUART28.

4.2.2. MCSI digital voice interface

Calypso supports a PCM digital voice interface on its MCSI pins, and the 4 pins that comprise this interface are brought out to the expansion interface header on Caramel2. This interface is documented in the Calypso-digital-voice article in *freecalypso-docs* Hg repository, but please note that there is no existing off-the-shelf solution for connecting such interfaces to a general-purpose computer — instead this PCM voice interface is an avenue for custom development, likely using an FPGA.

4.3. Boot control interface

The classic workflow for firmware boot-up, shutdown, reloading and recovery from hung states assumes that a human operator is physically present to press PWON and RESET buttons on the board as needed. As one can naturally imagine, this requirement can be unacceptable in some use cases — there are many valid use cases, such as remote development or unattended server operation, where it becomes highly desirable or even imperative to control Calypso board boot and reset functions from the connected host computer, without anyone being present to press buttons.

To support such operation, our Caramel2 board provides a 3-pin header on which PWON and RESET signal nets are brought out — they are the same signal nets that are shorted to GND with PWON and RESET push-button switches. These boot control signals can be driven by an external host — however, they **must** be driven with OC/OD drivers, **not** with conventional logic outputs!

For a complete solution, our DUART28 adapter board also provides two open drain outputs called CTL1 and CTL2, meant to be connected to PWON and RESET, respectively. However, if you make these wire connections between CTL1 and PWON and between CTL2 and RESET, you **must** also apply a custom patch to your Linux kernel ftdi_sio driver and reprogram the DUART28 adapter's EEPROM to the 'C' configuration — if you make these wire connections without doing the other steps, your Caramel2 board will become inoperable, held down in reset whenever the secondary UART is opened for communication. If you are not interested in applying a local patch to your Linux kernel ftdi_sio driver, then you must leave CTL1 and CTL2 unconnected — all other functionality will still work just fine with the standard unpatched kernel driver.

If you do go through all of the necessary steps for boot control connection from DUART28, then you gain the ability to use **-Pdtr** and **-Prts** options with *fc-loadtool* and related tools, and also with *rvinterf*: **-Prts** is equivalent to pressing PWON, and **-Pdtr** is equivalent to pressing RESET. Standalone *fc-pulse-dtr* and *fc-pulse-rts* utilities are also provided, doing the obvious.