FreeCalypso DUART28 User's Manual

Version 1.6, last edited 2023/12/27

1. Hardware product overview

FreeCalypso DUART28 is a compact (58x38 mm) adapter from USB to two LVCMOS UART channels, specifically designed for connecting to UARTs on Calypso GSM devices. The key hardware design features of DUART28 are:

- The USB connector type is mini-B, matching the historico-cultural period we wish to be a part of.
- The USB to dual UART IC is FT2232D, exactly the same as on PLDkit adapters we used previously.
- The set of supported UART signals is complete on UART channel 0 and data leads only on UART channel 1 — the set of available UART signals on various Calypso GSM devices matches this design choice.
- All UART output signals from DUART28 drive at 2.8 V logic levels, as opposed to more common 3.3 V; this slightly lower I/O voltage is native to Calypso and other mobile phone chipsets from that era.
- Each UART output signal is equipped with a 2.2 kΩ series resistor built into DUART28; the resulting arrangement limits the current that will flow into powered-down Calypso inputs to 1.27 mA per connected UART signal.
- All UART inputs to DUART28 pass through LVC buffers before hitting FT2232D I/O pins; this arrangement prevents high current flow from a powered-on and switched-on Calypso device into a powered-down USB-UART adapter — the case where the target Calypso device is on, the DUART28 adapter is connected, but there is no USB host providing USB power.

2. UART signal connections

2.1. Main set of signals

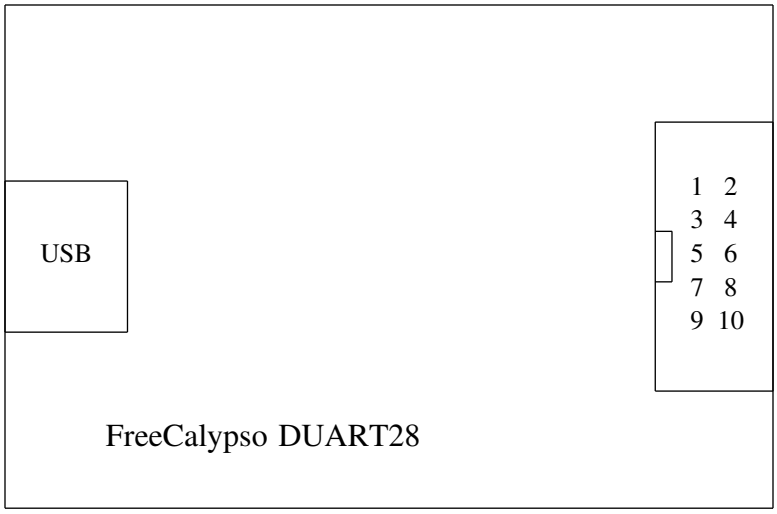
Most signals from both UART channels (all except channel 0 DSR and RI) are gathered on one 10-pin connector (shrouded header) in a pinout that was originally established with Calypso target boards FCDEV3B and MMTB1:

GND	1	2	GND
RxD_B	3	4	RxD
TxD_B	5	6	TxD
DCD	7	8	CTS
DTR	9	10	RTS

All signal names in the table above are given from the perspective of the DUART28 adapter itself, acting as DTE. The _B suffix on signals that appear on pins 3 and 5 signifies that these two signals belong to UART channel 1 (also called Channel B in FT2232D chip documentation); all other listed signals belong to UART channel 0.

2.1.1. Discrete wire connection

If you need to connect individual jumper wires to UART signals listed in the table above, use this drawing to locate them on the board:



2.1.2. FreeCalypso GSM board connection

If the Calypso GSM device you are working with is FreeCalypso FCDEV3B, Caramel2 or MMTB1, the 10-pin DUART interface coming out of the shrouded header on DUART28 is meant to connect directly (in one-to-one pinout) to 10-pin DUART interface headers on these listed boards. A straight-through connection of this 10-wire interface establishes the following canonical signal connection between the two host UART channels and Calypso UART signals on the target:

DUART28 signal	Calypso signal
TxD	RX_MODEM
RxD	TX_MODEM
RTS	CTS_MODEM
CTS	RTS_MODEM
DCD	GPIO2*
DTR	GPIO3*
TxD_B	RX_IRDA
RxD_B	TX_IRDA

* These two GPIO-as-UART connections are missing on FCDEV3B; pins 7 and 9 on the 10-pin DUART interface are unconnected on that board. The idea of assigning these two previously unused pins to GPIO-as-UART functions from TI's D-Sample board came later.

2.2. Extra DSR and RI signals

The complete set of signals is provided for UART channel 0 per design intent, but the 10-wire DUART interface fixed by the existing design of FCDEV3B and MMTB1 leaves no room for DSR and RI. Therefore, these two extra signals are brought out on a separate 3-pin header, along with a ground pin. Silk screen markings on the PCB identify each pin on this extra header, hence connection is unambiguous.

If you are working with a Caramel2 board, you can make RI signal functional by connecting an extra single wire (in addition to the main 10-wire DUART connection) between RI on DUART28 and GPIO1 on the 56-pin expansion header on Caramel2.

2.3. Partial connection of UART signals

There is no requirement to connect every UART signal supported by DUART28 adapter. In the case of outputs from the adapter, it is self-evident that unused signals can be simply left unconnected (there is no loading or termination requirement); in the case of inputs to the adapter, support for partial connection (ability to leave

signals unconnected without adverse effects) is provided by way of pull-up resistors.

Each input signal to DUART28 adapter features a 100 kΩ pull-up resistor to the adapter's local +2.8 V rail, generated on the DUART28 board from USB +5 V input power with an LDO regulator. A logic high on an LVCMOS UART interface corresponds to the inactive state of a conceptually-corresponding RS-232 signal (mark on data lines, negated state on control signals), hence the implemented pull-up resistors assure this generally-desired inactive state on signals that are either unconnected or not actively driven by the connected target device at certain times.

3. Internal electrical design

The internal electrical design of DUART28 adapter features two LDO regulators (one producing +3.3 V and the other producing +2.8 V) and two 74LVC541A buffer ICs. Both LDO regulators are fed from +5 V USB power input, and are used as follows:

- The LDO that produces +3.3 V powers the I/O pins of FT2232D and U6, the LVC buffer that receives LVCMOS UART signals from the target. It also powers the boot control output buffer described in section 6.1.
- The other LDO that produces +2.8 V powers U5 (the LVC buffer that drives UART outputs from the adapter) and the input pull-up resistors described in section 2.3.

3.1. Operating condition of U6 input buffers

LVC buffer U6 is powered (V_{CC}) with 3.3 V (as opposed to 2.8 V) so that it produces true 3.3 V logic levels on its outputs going to FT2232D inputs. However, inputs to this U6 buffer are expected to swing between 0 and 2.8 V, not going up to 3.3 V: the on-board pull-up resistors are to +2.8 V, and Calypso target outputs are expected to have 2.8 V logic levels. This mode of operation is fully within specification per IC datasheet: the V_{IH} spec for this V_{CC} range is 2.0 V. However, because the input voltage level to the buffer is significantly shifted from the V_{CC} rail in the high state, there is increased current flow through the buffer's CMOS input structure. According to the IC datasheet, this current can be as high as 500 µA per affected input, although the typical value is listed as only 5 µA. In any case, even the maximum value of this current is small enough to be negligible for a USB-powered device: 3 mA maximum when all 6 inputs are affected.

4. Operation in partial power-down scenarios

4.1. PPD scenario 1: USB power present, Calypso target is switched-off

This PPD scenario occurs all the time in normal operation: when you have connected your host computer to the Calypso target through DUART28, but haven't pressed the PWON or RESET button on the target yet, or when you have executed an orderly ''power-off'' command on the target, but haven't taken down the setup yet.

In this scenario there is current flowing from USB-powered DUART28 into powered-down Calypso inputs (this current flow cannot be eliminated without a design change on Calypso target side, inserting LVC buffers powered from the chipset's V-IO rail), but series resistors on DUART28 outputs limit this current to 1.27 mA per connected UART signal. In contrast, this current has been measured as 1.77 mA with an unbuffered FT2232D adapter or a whopping 8 mA with Sysmocom mv-uart!

In the other direction, DUART28 inputs will sense logic high when Calypso outputs are powered down, thanks to the pull-up resistors of section 2.3. It should also be noted that even if DUART28 output signals are disconnected from the Calypso target and only Calypso-to-DUART28 signals are connected, a small amount of current can still flow from the USB power domain of DUART28 into the Calypso+Iota chipset's V-IO rail: through the pull-up resistors inside DUART28, and then through the clamping diodes (inside the chip) on Calypso I/O pads that are connected to them. The high value of DUART28 pull-up resistors (100 kΩ) limits this current to a theoretical maximum of 28 µA per connected signal line, but it has been observed that even this tiny amount of current can sometimes feed enough power into the V-IO rail to cause erratic behaviour on Calypso-controlled LEDs!

4.2. PPD scenario 2: USB power absent, Calypso target is switched-on

This PPD scenario is not as common as the other one, but it can very easily arise by mistake (you power on your Calypso device with DUART28 connected, but there is no host computer connected on the USB side), and it can arise more legitimately with Calypso devices that are untethered handsets, rather than modems that absolutely require control from a connected host computer.

In this scenario DUART28 avoids presenting an excessive load to driving Calypso UART outputs: if these outputs were connected directly to I/O pins of a USB-serial chip, the result would be the USB-serial adapter presenting itself to Calypso outputs as a short to ground when there is no USB power present! DUART28 avoids this problem: UART input signals to the adapter go to LVC buffer U6, and in PPD scenario 2 the buffer's I_{OFF} spec applies, listed in the IC datasheet as 0.1 µA typical or 10 µA maximum.

However, the pull-up resistors of section 2.3 create a different surprising effect: despite each 100 kΩ resistor limiting the current that can flow through it (from a 2.8 V source) to a theoretical maximum of 28 µA, enough current flows from the powered-on Calypso device into the DUART28 adapter's internal P_2V8 rail to turn on output buffer U5! Because this buffer senses a logic low on its inputs (coming from powered-down FT2232D), the resulting effect is that the non-USB-powered DUART28 adapter sends logic low on all adapter-to-Calypso outputs, instead of the expected High-Z state. The resulting effects on Calypso are not dangerous (nothing can get damaged by this condition), but odd effects appear at higher levels of functionality, such as Calypso firmware ''inexplicably'' disabling all of its sleep modes in this condition.

Fortunately the only DUART28 internal power rail that gets wayward power fed into it in this scenario is P_2V8. No significant power flows into the other internal rail P_3V3 (the measured voltage on this rail was 8 mV in the condition under consideration), hence no U6 turn-on happens (the buffer does not exit I_{OFF} operation), and no boot control outputs turn on either.

5. DUART28 adapter EEPROM programming

DUART28 is a standard FT2232D device from USB perspective, and it features a 93C46 EEPROM for FT2232D configuration. This EEPROM is freely reprogrammable over USB, and FreeCalypso host software package *fc-usbser-tools* constitutes the official tool suite for FTDI EEPROM programming on DUART28 and other FreeCalypso hardware products.

There are two officially valid EEPROM configurations for DUART28, called the **C** configuration and the **S** configuration. They differ only in the USB ID code presented by the adapter to the host computer, and this scheme with two different ID codes being officially sanctioned as permissible configurations has been conceived as a workaround for a design flaw in Linux kernel serial port handling. The difference between these two EEPROM configs only matters when the optional boot control feature of the following chapter is used, otherwise it doesn't matter.

To check your current DUART28 EEPROM configuration and to switch between **C** and **S** configurations, please use *fc-duart28-conf* utility in *fc-usbser-tools*.

6. Optional target boot control feature

J5 is a 3-pin header on the DUART28 board, providing two extra output signals CTL1 and CTL2, along with a ground pin. Both signals are open drain outputs, controlled by otherwise unused RTS and DTR outputs from FT2232D Channel B: CTL1 drives low when Channel B RTS is asserted and is High-Z otherwise; CTL2 drives low when Channel B DTR is asserted and is High-Z otherwise. If these OD outputs are connected to PWON and RESET boot control signals on a Calypso board such as FCDEV3B or Caramel2, the result is programmatic control of the target's PWON and RESET levers, without requiring an operator to be physically present to press buttons on the board.

6.1. Electrical circuit details

The open drain driver IC for both CTL1 and CTL2 is Nexperia 74LVC2G07, a pair of simple OD buffers in one tiny package. This buffer's V_{CC} supply is powered from the adapter's +3.3 V rail (see chapter 3), and the inputs come from FT2232D BDBUS2 and BDBUS4 outputs. The internal signals that run from FT2232D outputs to OD buffer inputs also feature pull-up resistors to the same +3.3 V rail, thus each OD driver only turns on when FT2232D actively drives its RTS or DTR output low.

The initial power-up state of both RTS and DTR outputs prior to host software action is always negated, hence assertion of either boot control signal can happen only as a result of either an intentional pulse generation by a userspace program (good) or a bad OS design.

6.2. Using DUART28 target boot control with Linux

Unfortunately, usability of this feature under Linux is severely hampered by obstinence of kernel maintainers: making this feature work with Linux requires applying a custom patch to the kernel's `ftdi_sio` driver, adding a special quirk just for this one USB device, and the power-wielding maintainers are refusing to mainline this patch on ideological grounds. Therefore, if you wish to use this hardware feature with a Linux host, you must download a ''rogue'' (in *Homesteading the Noosphere* terms) patch from *fc-linux-patch* repository and apply it locally to your `ftdi_sio` driver source (in Linux kernel source tree) as an act of direct defiance against the stance of official maintainers.

If you are not willing to apply a rebellious, against-maintainers patch to your Linux kernel, then you must not connect anything to CTL1 or CTL2 outputs on your board — connecting those outputs with an unpatched Linux kernel will produce an inoperable system. But as long as those extra outputs remain unconnected, all other DUART28 functionality will work fine without needing any kernel patches.

6.3. Using DUART28 target boot control with FreeBSD

Unlike Linux, FreeBSD (since release 13.0) provides a straightforward way to disable automatic assertion of DTR and RTS on serial port open, allowing use of custom hardware such as DUART28 with boot control signals. However, as of this writing (end of 2023) FreeCalypso community does not have any FreeBSD users, only Linux — hence no work has been done yet to port any of our host tool packages (*fc-host-tools*, *fc-usbser-tools*) to FreeBSD.

If someone genuinely and sincerely wishes to use FreeCalypso hardware with FreeBSD instead of Linux as their host OS, a project can be started to port *fc-host-tools* and *fc-usbser-tools* to FreeBSD — however, at least one real user of FreeCalypso plus FreeBSD combination must step forward first.