GTA04b7 Infrared Subsystem

Jörg Reisenweber[*], Werner Almesberger[†]

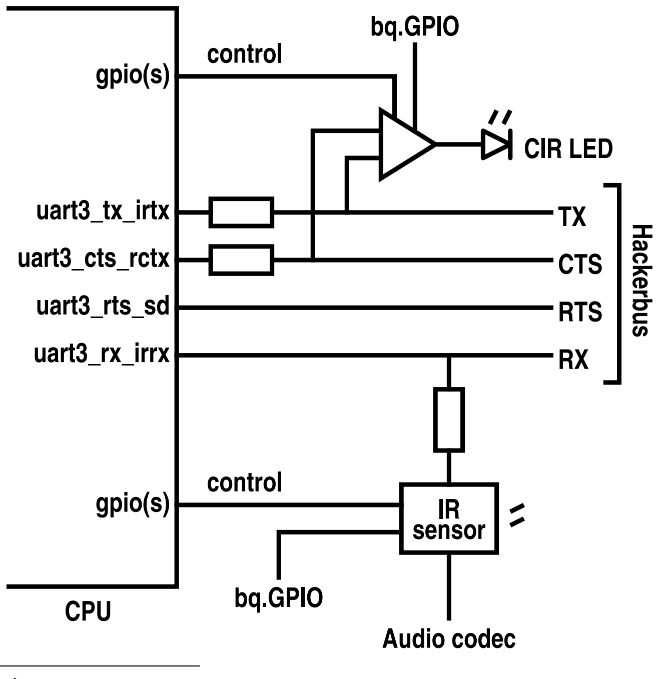
August 26, 2014

This document describes the infrared (IR) system of GTA04b7. It is intended to be used both as tentative specification and as documentation of actual functionality, and may be revised as focus shifts over time from requirements to implementation.

1 System overview

The IR subsystem consists of a high-powered transmitter and a receiver. It connects to UART3 of the DM3730 CPU, where hardware-assisted IrDA[1] (SIR) and CIR[2] ("Consumer CIR") functions are available.

The IR subsystem also connects to the "Hackerbus" which can use the signals to communicate with the CPU, be it as UART or as GPIOs, or to access the IR system. For advanced signal processing, the analog output of the IR sensor connects to an unused input of the audio codec. The following drawing shows the various components:



[*]Concept and design requirements.

[†]Specification details and illustrations.

[1]`http://en.wikipedia.org/wiki/Infrared_Data_Association`

[2]`http://en.wikipedia.org/wiki/Consumer_IR`

The signal named bq.GPIO determines the configuration after CPU reset but may also be used in other system states. It is described in section 3.2.

1.1 IR modes

The IR subsystem supports both CIR and IrDA operation. In CIR mode, it can receive incoming IR signals and thus act as a "learning" remote control. Furthermore, the IR transceiver can be configured to send and receive unmodulated data from the UART. We call this "IR-UART" mode

These characteristics mean that some design conflicts are unavoidable, e.g., because CIR and IrDA operate at different wavelengths. Where a conflict arises, highest priority is given to CIR transmission, followed by CIR reception, and finally IrDA.

Likewise, since the IR data path is shared between CPU and Hackerbus, conflicting configurations are possible. Their resolution is described in sections 6 and 7.

Note that we don't consider IR-UART reception and IrDA to be very important features. If their implementation should prove to be overly troublesome, they could therefore be omitted.

2 Physical layer

The following sections describe the main characteristics of the physical layers of CIR, IrDA, and IR-UART, and then define a set of hardware characteristics that achieve compatibility with all these modes of operation.

2.1 CIR

While there is no common standard for CIR, the characteristics of most IR remote controls typically lie in the following ranges[3], with amplitude-shift keying (ASK) being the most common form of modulation, although other types of modulation are possible:

Parameter	Value	Unit	
Wavelength	870, 930–950	nm	
Data rate	4–120	bps	
Carrier	33-60	kHz	
Beam angle	≥ 50	$^\circ$	(estimate for IR remote)

2.2 IrDA

IrDA (SIR) uses a pulse modulation with UART-like framing and has the following characteristics:[4]

[3]Based on the following Wikipedia article: `http://en.wikipedia.org/w/index.php?title=Consumer_IR&oldid=615137350#Technical_information`

[4]Based on the following Wikipedia article: `http://en.wikipedia.org/w/index.php?title=Infrared_Data_Association&oldid=615436526#IrPHY`

Parameter	Value	Unit	
Wavelength	850–900	nm	
Data rate	9.6-115.2	kbps	
Duty cycle	3/16		“0“ bit
	0/16		“1“ bit
Pulse width	1.6–19.5	μs	
Beam angle	≥ 15	°	

2.3 IR-UART

Last but not least, IR-UART mode uses on-off keying (OOK), and we can define the following characteristics:

Parameter	Value	Unit	
Data rate	9.6–115.2	kbps (typical)	
Duty cycle	9.1–91	%	[5]

2.4 Combined characteristics

Combining the specifications from the preceding sections, we obtain the following common characteristics for transmitter and receiver:

Parameter	Value	Unit	
Wavelength	940	nm	transmit
	850–950	nm	receive
Frequency	0.9–312	kHz	
Beam angle	≥ 50	°	

2.5 Transmit power

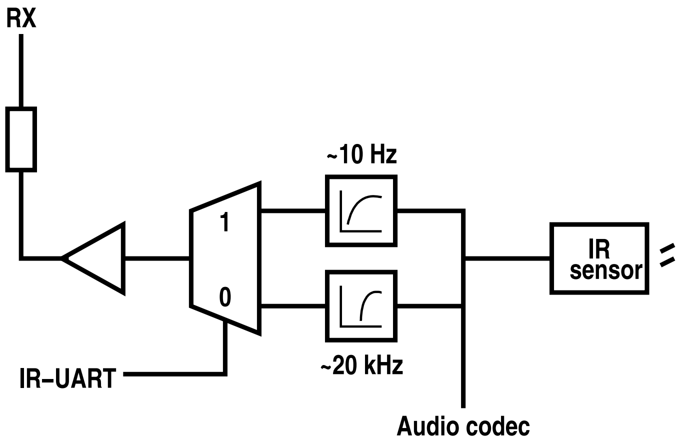
While we do not know the radiant flux density required at the IR receivers of consumer devices and therefore cannot specify the minimum radiant intensity of the transmitter, we expect performance of the IR transmitter to be on par with the average TV remote control. For comparison, the N900 has a range of only approximately 2 m, which we would consider too short.

In order to improve its range, the transmitter should be driven with as high a current as reasonably possible. In CIR and IrDA mode, this current would be well above the permissible continuous forward current. Note that incorrect use could cause the IR LED to overheat and the energy of a pulse therefore has to be limited. We discuss this in section 8.

[5] The duty cycle range of 9.1–91% (1/11 to 10/11) is for 8 bits with even parity, as used by the DM3730 ROM boot loader. The duty cycle range would be 10–90% for the more common 8 bits without parity format.

2.6 Received signal processing

On the receiving side, the signal should in CIR or IrDA mode go through a high-pass filter with a cutoff frequency of about 20 kHz and then be shaped into a digital signal for the RX line. The unfiltered analog signal should also be sent to an unused input of the audio codec. We have no further specification for the analog signal from the IR sensor at this time.



Given that IR-UART can have very long high or low pulses, a high-pass filter with a much lower cut-off frequency has to be used when IR-UART mode is selected. A cut-off frequency of about 1–100 Hz should be sufficient for DC blocking. The figure above illustrates the concept.

3 Configurations

The IR subsystem can be configured for several different modes of operation. The following sections detail the various configurations.

In each case the configuration of both the transmitter and the receiver is shown. In many real-life situations, only one direction will be used at a time while the other is deactivated. A deactivated transmitter or receiver should behave as described in section 3.3.

3.1 CPU powered down

If the CPU is powered down, the IR subsystem shall be deactivated as described in section 3.3.

3.2 Reset

When the control signals from the CPU are in their reset state, the configuration of the IR system is determined by the state of the GPIO pin of the bq27200 battery fuel gauge. We call this pin "bq.GPIO" in the rest of the document. The fuel gauge chip is directly connected to the battery and retains its state during normal system reset.

Note that the bq27200 does not preserve the entire state of bq.GPIO across power-on reset: while the direction of the pin is determined by EEPROM content and is therefore retained across power-cycling, the open-collector output is always set to "1", i.e., it is high-Z.

If bq.GPIO is high-Z and the CPU's control signals are in their reset state, both transmitter and receiver shall be active and operate in IR-UART mode. The purpose of this mode is to allow the CPU's ROM boot loader to try to boot from UART3 via infrared.

If bq.GPIO is "0" and the CPU's control signals are in their reset state, the IR subsystem shall be deactivated as described in section 3.3.

When the CPU has left its reset state and can actively control the IR subsystem, it is also able to change the state of bq.GPIO. bq.GPIO can therefore be used (or ignored) as the implementation sees fit.

The following table summarizes the IR configuration in power-down and after CPU reset:

CPU.controls	bq.GPIO	IR configuration
CPU off	any	off (minimum power)
reset state	"1" (High-Z)	IR-UART TX/RX
	"0"	off (minimum power)
active	to be defined	defined by controls

Use of the data signals in IR-UART mode is further explained in section 3.6. Note that Hackerbus can override the receiver and the CPU can therefore boot from (wired) UART even if the IR-UART configuration is selected.

3.3 IR deactivated

The CPU can deactivate the transmitter and receiver, either both together or one at a time, through the control signals to the IR subsystem.

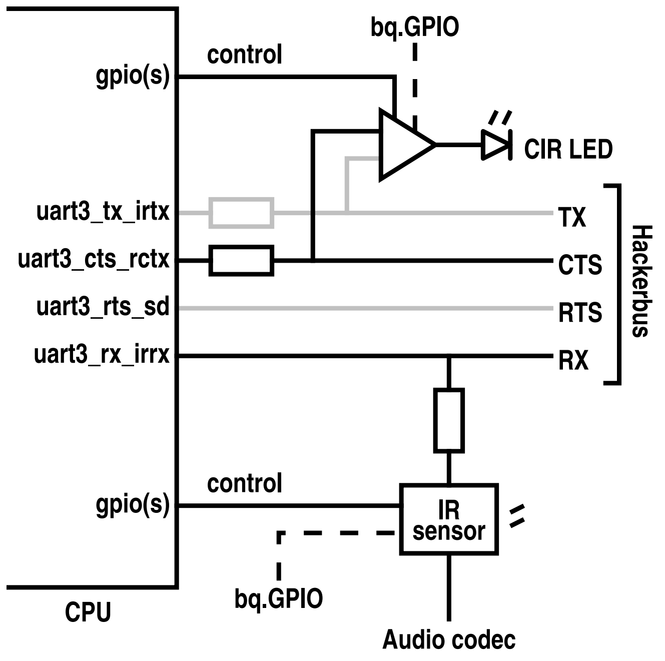
When deactivated, the IR subsystem must draw as little current as possible. The transmitter must ignore its data inputs, e.g., TX and CTS, and under no circumstance activate the IR LED. The receiver must not drive the RX line and use as little power as possible for any filters and/or amplifiers.

3.4 CIR mode

In CIR mode the CPU uses CTS as an active-high output[6] for transmit data. The CIR module in the CPU could also use RTS for configuration purposes, but we seem to have no use for this and the IR circuit must ignore its state. The use of RX in CIR mode is not clearly specified and it seems safe to use it similar to IrDA mode. TX is not used and can be assigned to other tasks. The IR circuit must therefore ignore the state of TX when in CIR mode.

CIR mode configuration is illustrated in the following drawing:

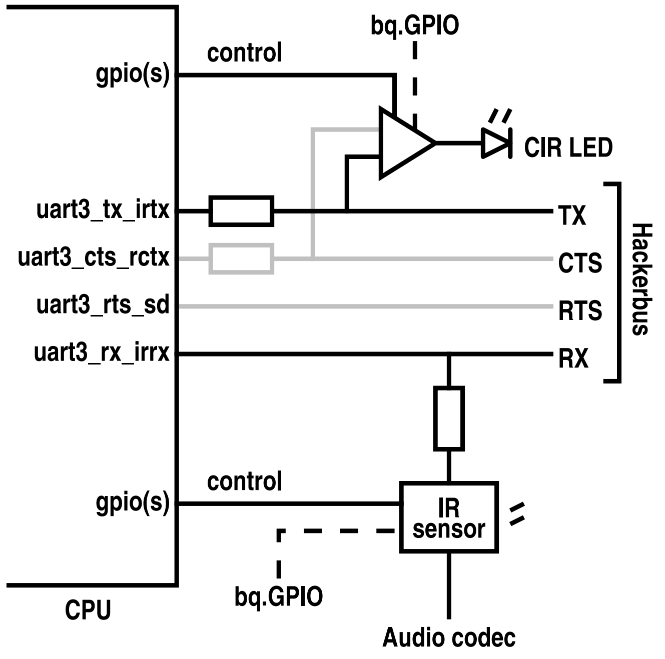
[6]Note that CTS is an input at the CPU when UART3 is used as regular UART.



3.5 IrDA mode

In IrDA (SIR) mode, the CPU uses TX as active-high output and RX as either[7] active-low or active-high input. RTS could again be used for configuration but is not relevant in our application. CTS is not used in IrDA mode. The IR circuit must therefore ignore the state of RTS and CTS when in IrDA mode.

IrDA mode configuration is illustrated in the following drawing:



[7]Depending on the setting of MDR2_REG.RRXINVERT.

3.6 IR-UART mode

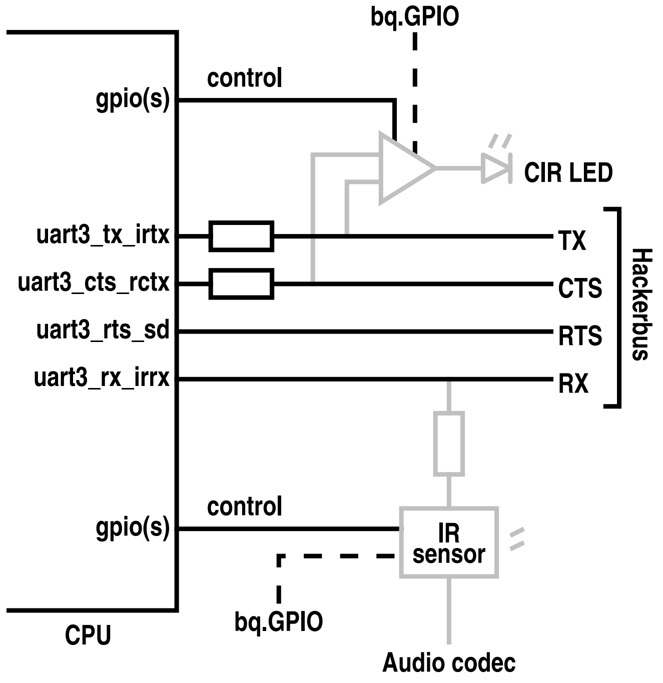
In IR-UART mode, TX and RX both are active-low. RTS and CTS are not used.

The IR LED driver may be configured to operate at a current that does not exceed the LED's maximum continuous forward current. In any case, power dissipation must be controlled as described in section 8.

3.7 UART mode

The IR subsystem is inactive in UART mode and must behave as described in section 3.3. While we call it "UART mode", this configuration applies to any setup where the UART signals are used for some form of communication with the Hackerbus that does not involve IR.

UART mode configuration is illustrated in the following drawing:



3.8 Mode summary

The following table summarizes the configuration of the I/O pins at the CPU in the various IR modes:

Signal	IR subsystem mode				
	Off	CIR	IrDA	IR-UART	UART
TX	—	—	$\overset{\sqcap}{\longrightarrow}$	$\overset{\sqcup}{\longrightarrow}$	$\overset{\sqcup}{\longrightarrow}$
CTS	—	$\overset{\sqcap}{\longrightarrow}$	—	—	$\overset{\sqcup}{\longleftarrow}$
RTS	—	—	—	—	$\overset{\sqcup}{\longrightarrow}$
RX	—	$\overset{\sqcup/\sqcap}{\longleftarrow}$	$\overset{\sqcup/\sqcap}{\longleftarrow}$	$\overset{\sqcup}{\longleftarrow}$	$\overset{\sqcup}{\longleftarrow}$

$\longrightarrow$ indicates an output from the CPU, $\longleftarrow$ an input. An active-high signal is marked with $\sqcap$ while an active-low signal is marked with $\sqcup$. A signal where the CPU can be configured for either polarity it marked with $\sqcup/\sqcap$. An unused signal is marked with —.

4 Possible component selection

Taking into account the requirements specified in section 1.1, available space (see section 5), and general component availability, we consider the Vishay VSMB2948SL[8] IR LED and the Vishay VEMD2023SLX01[9] photodiode suitable for our purpose.

Parameter	Requirement	Unit	Transmitter VSMB2948SL	Receiver VEMD2023SLX01
Wavelength	850–950	nm	940	750–1050
Beam angle	≥ 50	°	50	70
Peak frequency	≥ 0.3	MHz	23	≈ 2

Note that the peak frequency of 2 MHz for the receiver is an estimate based on the specified rise and fall times of 100 ns each, which are specified for a reverse voltage of 10 V whereas we would expect to operate at a lower voltage.

The characteristics of the daylight filter integrated in the IR window of the N900 case are unknown, our design will add multiple internal light sources that may affect the photodiode, and there is also the possibility of outside light reaching the diode through the spacer frame. We therefore consider it desirable for the photodiode to contain its own filter and not have to rely on external filtering and shielding.

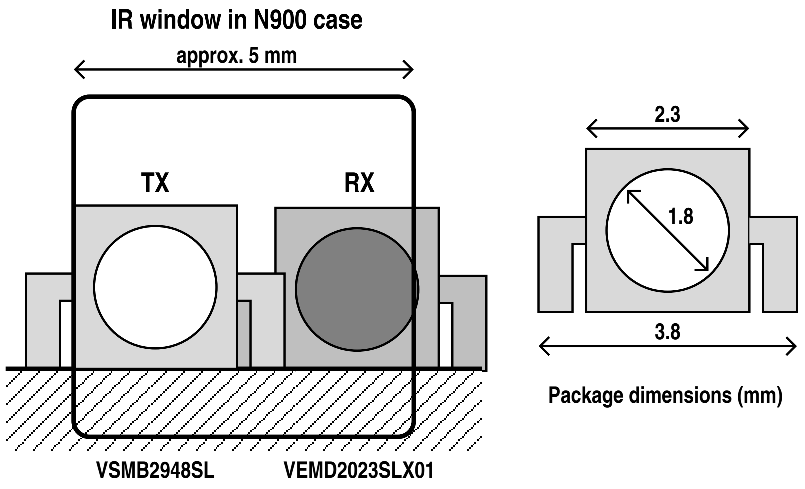
We considered the use of a phototransistor, but found that the readily available SMT parts all had rise and fall times in the order of 15 μs, which would limit the maximum signal frequency to only about 30 kHz. They do not allow improving response time by biasing the transistor.

5 Component placement

The N900 case has a window with an integrated IR filter of unknown characteristics. The window is approximately 5 mm $\times$ 5 mm.

The window is designed for a single IR LED but we will have a LED and a photodiode. Since IR components are generally large, it won't be possible to place both "nicely" side by side. However, as the following drawing illustrates, both components can "see" the window if placed behind each other:

[8] `http://www.vishay.com/docs/83498/vsmb2948sl.pdf`
[9] `http://www.vishay.com/docs/83493/vemd2023slx01.pdf`



The transmitter should be closer to the window since its performance is more important than that of the receiver.

6 Conflict resolution

When a line is driven as output from multiple sources, the following conflict resolution strategy shall be applied:

Line	Dominant	Subordinate	Applicable IR system states
TX	Hackerbus	CPU	IrDA, IR-UART
CTS	Hackerbus	CPU	CIR
RX	Hackerbus or CPU	IR sensor	CIR, IrDA, IR-UART

Conflicts between CPU and Hackerbus on RTS or RX result in undefined behaviour and should be avoided. Conflict increases overall power consumption and should be avoided during normal system operation.

Precedence is established by series resistors next to the CPU for TX and CTS, and a series resistor on the digital output of the IR receiver, before entering the RX line.

7 Undriven signals

The IR circuit must be tolerant to a UART signal being driven by neither CPU, Hackerbus, or the IR receiver. Whether this tolerance is achieved by pulling the signal to a safe state or by applying a more general protection against incorrect inputs is left as an implementation choice.

Control signals that float after CPU reset and that are not ignored by the IR subsystem must be pulled to a safe state.

8 Overload protection

To improve its range, the IR LED can be driven with a peak current significantly above its permitted continuous forward current. If the diode is driven at such a high current for too long, it could overheat and suffer permanent damage.

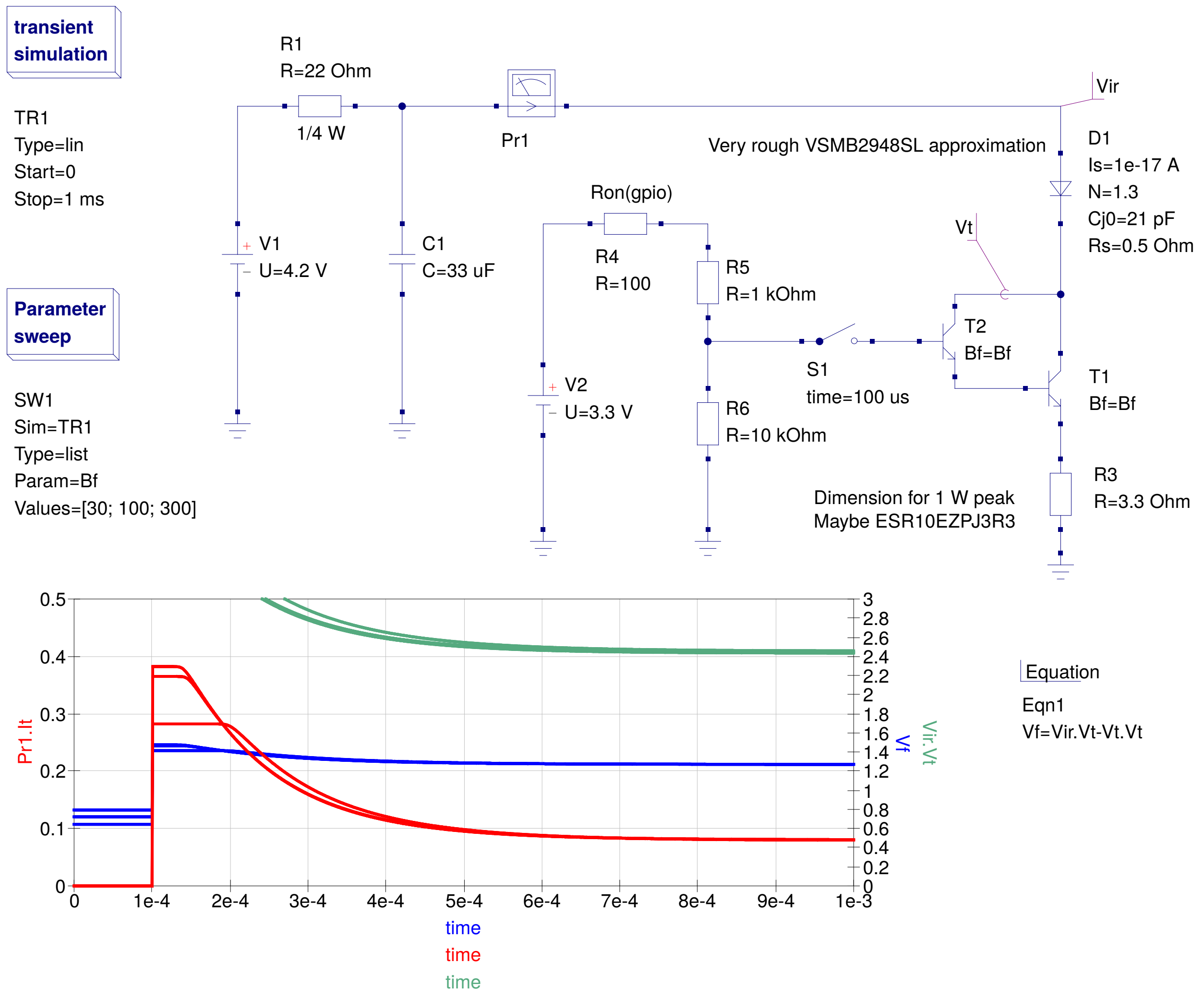
While some of the IR protocols operate with low duty cycles that allow the transmitter to cool down between bursts or pulses, the circuit should not rely on software always setting up things correctly. Also, the IR-UART protocol can have duty cycles of up to 91% and would require special handling.[10]

We considered simply limiting pulse length to the peak duration of 100 μs commonly specified for IR LEDs, but this would still leave many dangerous patterns and would conflict with correct use of IR-UART where the maximum nominal pulse at 19.2 kbps is already 469 μs.

We therefore suggest a simple analog circuit that limits long-term average current to the diode's continuous forward current but that stores energy for short bursts in a capacitor. Use with low duty cycle would give the capacitor time to recharge between pulses while high duty cycle would empty the capacitor in a brief initial flash and then proceed at the "safe" continuous current.

This reflects the thermal processes inside the diode, where the continuous current corresponds to heat dissipation and the peak energy transferred from the capacitor corresponds to the diode's thermal energy.

[10]One possible use of IR-UART is to output kernel diagnostics. This sort of text would have a duty cycle of about 50–60% if sent as 8 bit characters with a start and a stop bit each and the LED lit during every "0" bit.



Example current-limiting circuit: continuous current through the IR LED D1 is limited by R1 and R3. Short pulses in the 100 μs range can draw extra current from C1. The peak current is then limited by the constant current transistor circuit. This example uses a Darlington configuration to obtain a current gain $\beta \gg 100$.

The simulation illustrates the operation of the circuit with an infinite pulse starting at 100 μs. Note that the model is only a rough approximation and should be confirmed by lab measurements. The source of the Qucs simulation can be found here:
`http://neo900.org/git/?p=misc;a=blob;f=ir/ilim-cc.sch;hb=HEAD`