R2Z5000-5001

ARDUINO/RASPBERRY THERMAL PRINTER SHIELDS

Datasheet

1. Device overview

The thermal printer module is available in Arduino shield format (R2Z5000) and in Raspberry hat format (R2Z5001). Both variants have UART interface connected to his specific pinout pins. USB can be optionally connected with a Micro USB cable.
The shield has an independent power connector that powers both printer and Arduino/Raspberry board. Please note that Arduino/Raspberry power source is insufficient and shall not be used with the printer connected.

1.1. Main features

5V at 2A power supply
Serial communication over USB and UART interface
7 dots hot printing head
105 dots per line
Text mode printing with 7 x 5 dots characters
Graphical printing with 105 x 7 dots contiguous areas
Paper width of 37 mm

1.2. Power

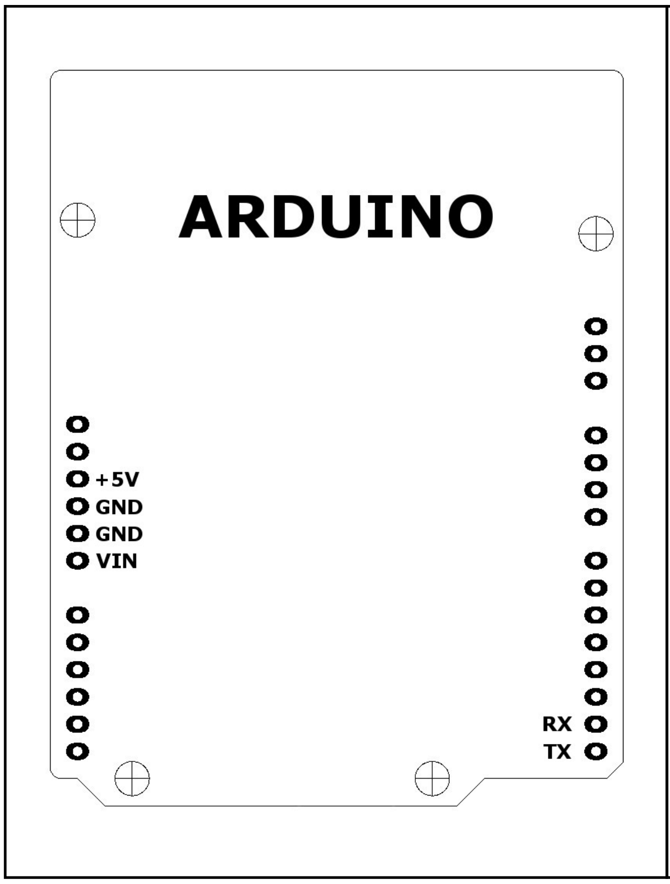
1.2.1. Arduino (R2Z5000)

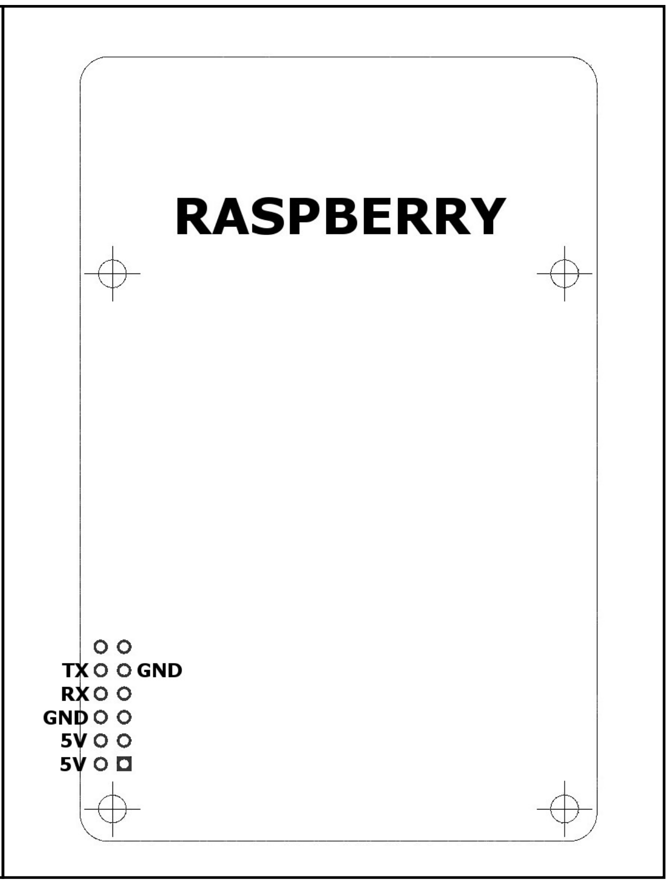
The printer can be powered directly through his power connector within a minimum voltage of 12V/1A and a maximum of 18V. The integrated regulator forwards a proper stabilised 5V to the Arduino/Raspberry main board that does not need to be powered separately.
Alternatively, it's possible to power the module through the VIN line (which is normally connected to Arduino's black power connector), however that limits the available current because of a protection diode usually placed betwheen VIN pin and power connector. The "PWR" jumper allows to insulate printer and Arduino power lines.

1.2.2. Raspberry (R2Z5001)

The printer shall be powered directly through his power connector within a minimum voltage of 12V/1A and a maximum of 18V. The integrated regulator will forward a proper stabilised 5V to the Arduino/Raspberry main board that does not need to be powered separately.
Do NOT use Raspberry's micro-USB power connector because current and voltage are totally insufficient for both devices.

1.3. Pinout





Arduino equivalent pin	Raspberry equivalent pin	Name	Description	Power	UART
5V	2,4	+5V Power	+5V	x	
Vin		Arduino Power Input	Vin	x	
GND	6	Ground	GND	x	
TX	8	UART TX	UART TX on Arduino/Raspberry (printer RX)		x
RX	10	UART RX	UART RX on Arduino/Raspberry (printer TX)		x

2. UART/USB interfacing

2.1. UART Mode

Serial UART link uses TX and RX pins at the speed of 115200 baud, 8bit, no parity, 1 stop bit with 5V TTL logic levels. If the shield is connected via USB to a device, UART link is automatically disabled.

2.2. USB Mode

When connected to USB, the shield is seen as a generic CDC serial port and is automatically detected without need of drivers on most OS including Windows and Linux. In this behaviour, parameters like baudrate, parity, etc are totally ignored. When USB is correctly connected and initialzed, TX/RX UART pins are disabled.

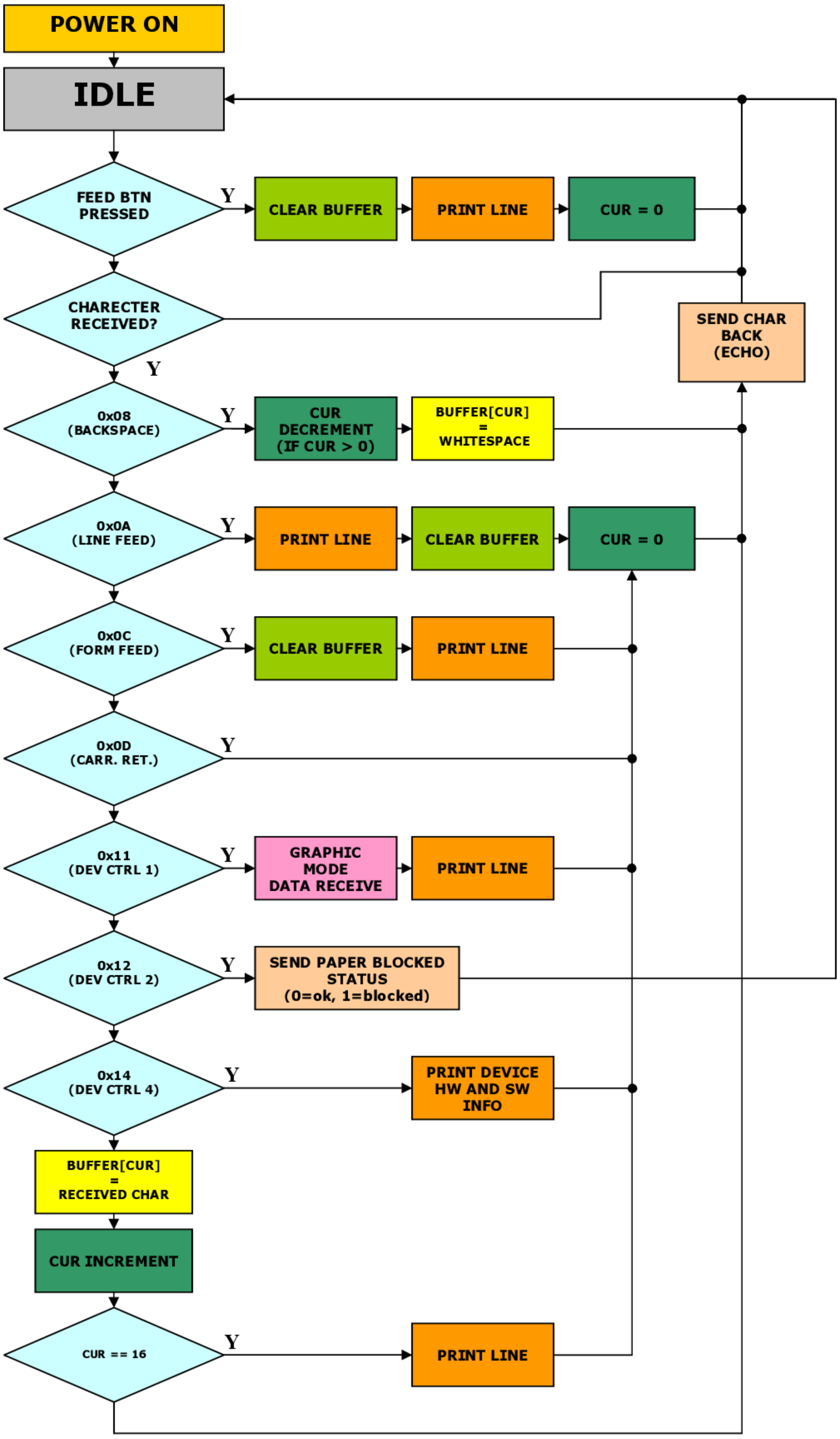
2.3. Flow control

Flow control can be implemented using printer echo as reference. For each byte received will always be sent one back at the end of elaboration. Using USB interface no data will ever be lost because buffering and integrity are guaranted, but in UART mode any byte sent before receiving a response will be lost.
When the printer receives a byte that for any reason triggers a line print operation, it waits for full completion to send the response.

3. Internal device operation

3.1. Internal state machine

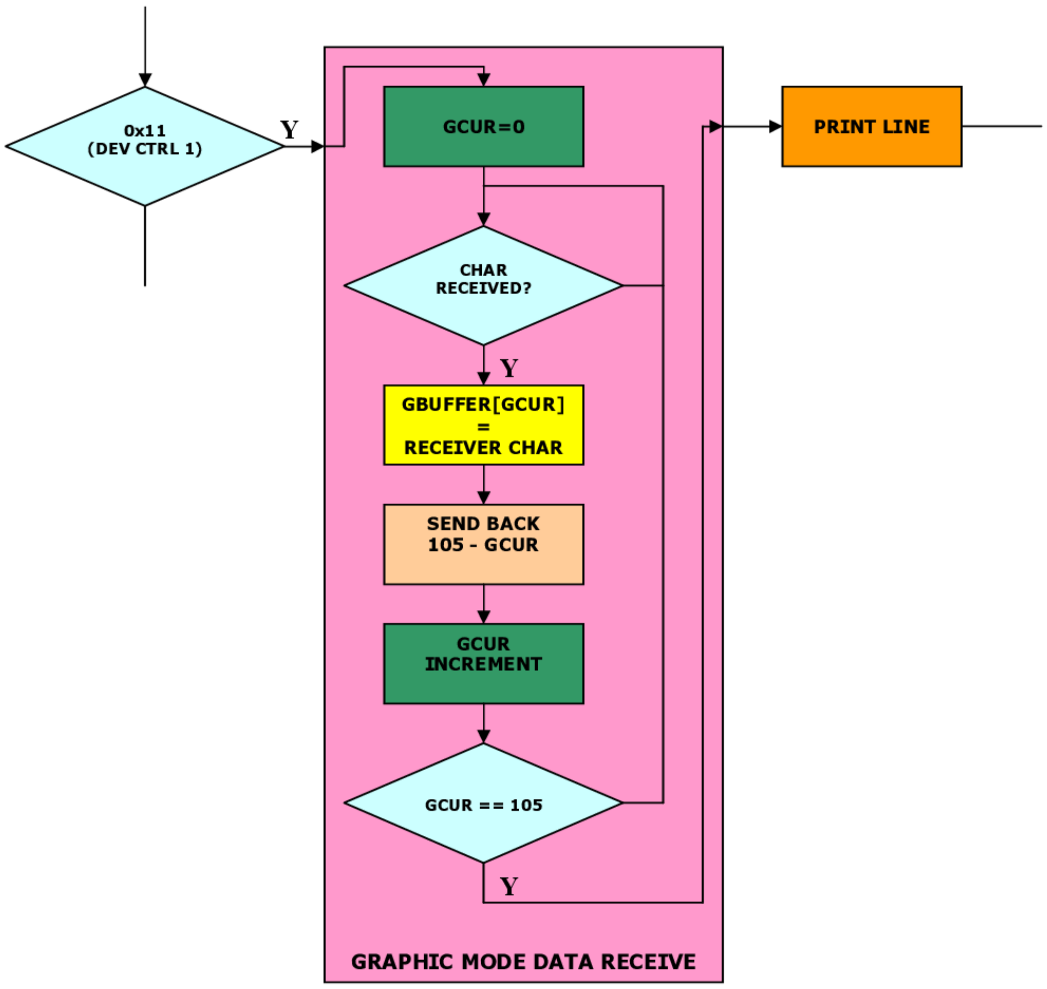
The following diagram shows the internal workflow of the printer. BUFFER represents a byte array of 15 bytes used to store one line of characters and CUR represents an integer variable expressing the current position. In the following chapters some parts of the diagram are described more in detail.



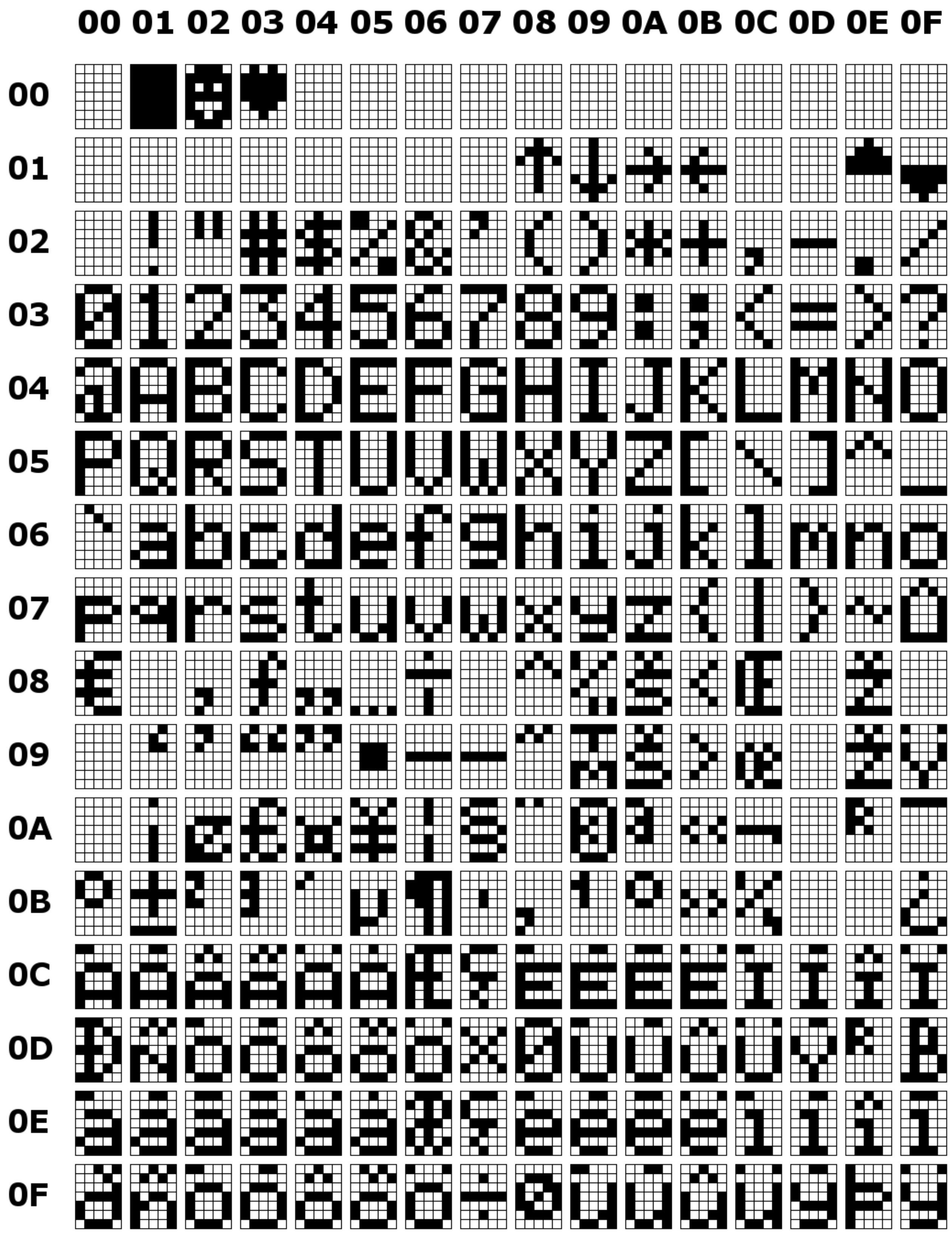
Briefly, a line print occours at the reception of a Line Feed (0x0A) or at the reception of the 15th byte after: latest printed line, latest Carriage Return (0x0D), latest Form Feed (0x0C), latest graphic mode print or Feed button press.

3.2. Graphic mode

Graphic mode allows to freely drive every single pixel of the printing line. This mode is activated after the reception of the characted 0x11. After that, the printer expects to receive 105 bytes for starting the line print operation and returning back in Idle after responding with a 0x11.



4. Character map


00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
00
01
02
03
04
05
06
07
08
09
0A
0B
0C
0D
0E
0F