Quince micro lite (v1.0)

CyrusDreams

January 2026

Introduction

The 10-cent-microcontroller, it was everywhere in the news. With a 32bit RISC-V core and useful peripherals, the CH32V003 from WCH is suited to many projects. But such a small microcontroller needs to be programmed via a dedicated adapter. In a forum at mikrocontroller.net, user F.P. analyzed the WCH LinkE debugging adapter and came to the conclusion that the binary code from WCH can be run on the USB enabled CH32V305 microcontroller. And because the CH32V305 can do so much more than its smaller sibling, the idea of this development board was born.

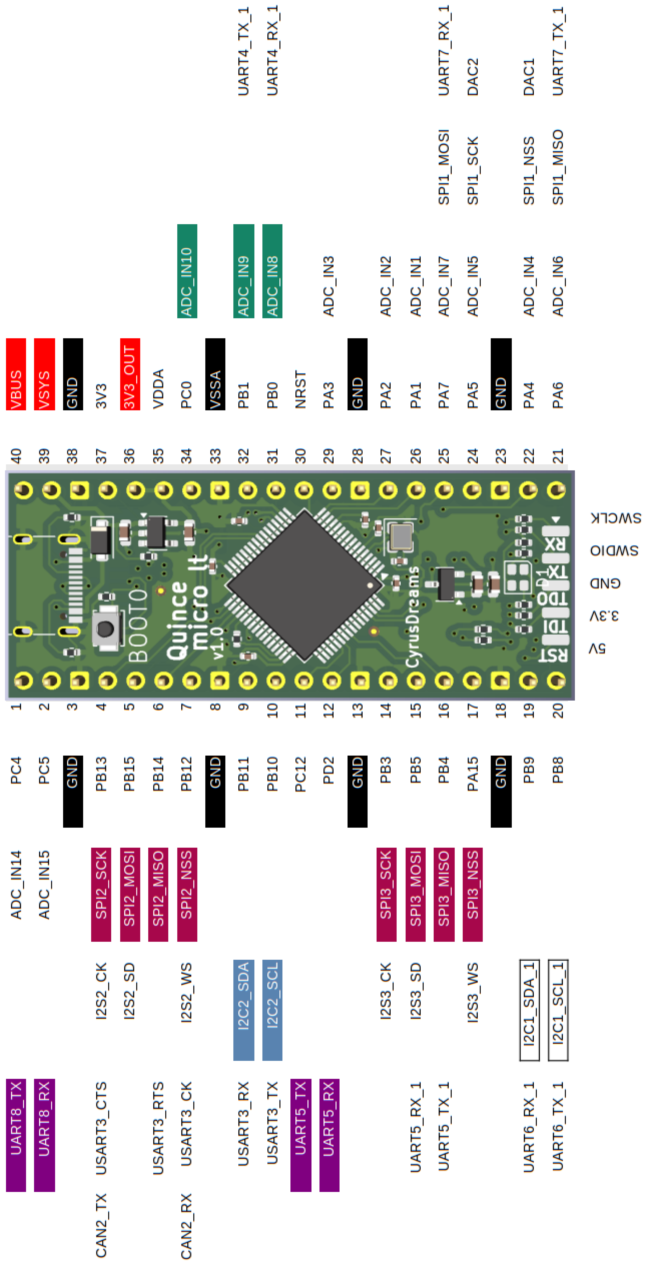
This description is valid for the ‘lite’ variant of said development board and its second production release version ‘1.0’.

Chapter 1

Features of the Quince micro lite

- board dimensions of 21 mm * 51 mm are the same as a *RaspberryPi Pico*, but it lacks the mounting holes, due to the use of an USB-C connector
- the pinout of the board is designed to match the peripherals on the *ClockworkPi Pico Calc* mainboard
- the boards has mounting pads for a microSD card slot and a connector for an OLED screen, if those are mounted, the board has the same features as a *Longan Nano*
- the *LinkE* debug connector is found on the PCB edge, the “lite” version omits the switchable target supply voltages
- on the bottom side of the PCB, a CAN bus transceiver can be mounted, this makes the Quince micro a small CAN bus node
- a RGB LED can be controlled from the microcontroller, two of the LEDs share the port with the Rx/Tx lines of the debug interface

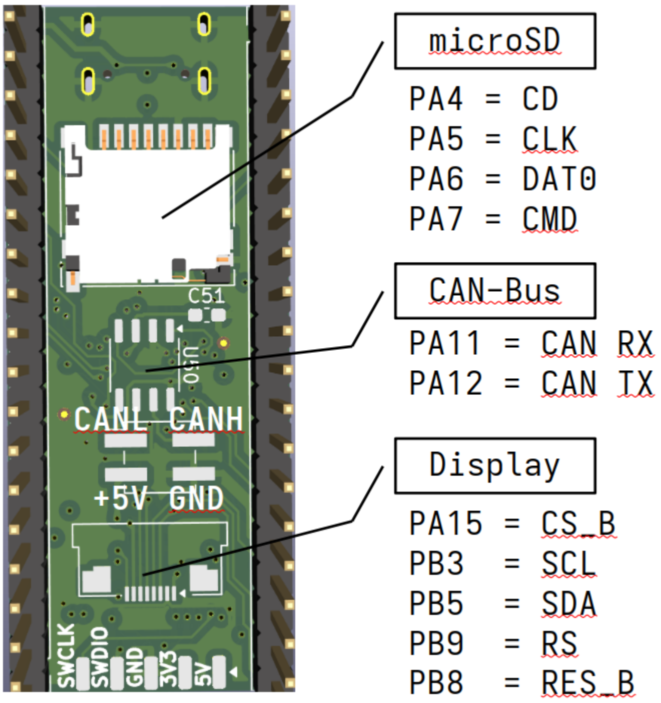
1.1 Pinout of the Quince micro


40 VBUS
39 VSYS
38 GND
37 3V3
36 3V3_OUT
35 VDDA
34 PC0 ADC_IN10
33 VSSA
32 PB1 ADC_IN9 UART4_TX_1
31 PB0 ADC_IN8 UART4_RX_1
30 NRST
29 PA3 ADC_IN3
28 GND
27 PA2 ADC_IN2
26 PA1 ADC_IN1
25 PA7 ADC_IN7 SPI1_MOSI UART7_RX_1
24 PA5 ADC_IN5 SPI1_SCK DAC2
23 GND
22 PA4 ADC_IN4 SPI1_NSS DAC1
21 PA6 ADC_IN6 SPI1_MISO UART7_TX_1
SWCLK
SWDIO
GND
3.3V
5V
BOOT0
Quince
micro
lt
v1.0
CyrusDreams
RX
TX
TDO
TDI
RST
1 PC4 ADC_IN14 UART8_TX
2 PC5 ADC_IN15 UART8_RX
3 GND
4 PB13 SPI2_SCK I2S2_CK USART3_CTS CAN2_TX
5 PB15 SPI2_MOSI I2S2_SD
6 PB14 SPI2_MISO USART3_RTS
7 PB12 SPI2_NSS I2S2_WS USART3_CK CAN2_RX
8 GND
9 PB11 I2C2_SDA USART3_RX
10 PB10 I2C2_SCL USART3_TX
11 PC12 UART5_TX
12 PD2 UART5_RX
13 GND
14 PB3 SPI3_SCK I2S3_CK
15 PB5 SPI3_MOSI I2S3_SD UART5_RX_1
16 PB4 SPI3_MISO UART5_TX_1
17 PA15 SPI3_NSS I2S3_WS
18 GND
19 PB9 I2C1_SDA_1 UART6_RX_1
20 PB8 I2C1_SCL_1 UART6_TX_1

The coloured signals match the ones on a *Raspberry Pi Pico*.

Remark: ports PB8 and PB9 are exchanged in the v0.9 revision of the *Quince micro lite*.

1.2 Quince micro compared to the Longan nano



microSD

PA4 = CD
PA5 = CLK
PA6 = DAT0
PA7 = CMD

CAN-Bus

PA11 = CAN RX
PA12 = CAN TX

Display

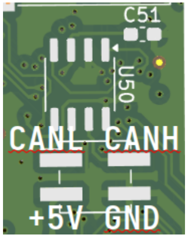
PA15 = CS_B
PB3 = SCL
PB5 = SDA
PB9 = RS
PB8 = RES_B

On the bottom side of the *Quince micro*, additional parts can be assembled to have a similar functionality as a *Longan nano* development board:

- microSD slot, the port pins are also available on pins 21, 22, 24 and 25 of the *Quince micro*
- eight-pin connector for OLED displays

1.3 Quince micro as a CAN bus node

On the bottom side of the PCB, a CAN bus transceiver can be mounted (position U50) with its decoupling capacitor (C51, 100nF) and a four pole SMD pin header. With the available signals CAN-high, CAN-low, 5V-supply and Ground, the Quince micro can be supplied with power and act as a CAN bus node.



Recommended are CAN-SIC transceivers (Signal Improvement Capability), if the additional cost of approximately 0.15€ is bearable for the intended project. These newer transceivers enable higher data rates and/or longer wiring distances than older CAN-FD transceivers. For *Quince micro*, CAN transceivers with ‘standby’ input and separate IO voltage are suitable. Examples of compatible transceivers are TJA1462 , TCAN1462, TLE9371VSJ and others.

Chapter 2

Quince micro is a Debugger

2.1 Features of the Quince micro as a WCH LinkE Debugger

On the short side of the PCB, opposite of the USB-S connector, there are solder pads suitable for two sets of five pin headers. The pinout is close to the *WCH LinkE R0 1v3*; the power supply pins are not switchable. Also not available is the "ModeSelect" button. Its functionality can be controlled via software, as stated in the "WCH-LinkUserManual" (version V2.4).

2.2 Setup of the Quince micro as a LinkE Debugger

In the forum of mikrocontroller.net, user Benutzer F.P. (fail) posted the following Instructions that are similar for the *Quince micro*:

1. Install the software from WCH, commands for Linux:

   (a) `git clone https://github.com/ch32-rs/wchisp`

   (b) `git clone https://github.com/ch32-rs/wlink`

2. Build the software (see respective documentation)

3. Put the microcontroller into the USB programming mode

   (a) unpower the Quince micro
   (b) push and hold BOOT0 switch
   (c) connect USB to a PC
   (d) release BOOT0 switch

4. Flash the firmware from MounRiverStudio (see mikrocontroller.net)

   (a) `wchisp flash WCH-LinkE-APP-IAP.bin`

5. restart the Quince micro: unplug USB, reconnect USB

Now the Quince micro is ready to function as a WCH-LinkE adapter. After connecting it to another CH32 target (for example another Quince micro), the second microcontroller can be programmed with, for example:

```
1. wlink flash blink.bin
```

The connection to a second Quince micro is done by connecting the pins SWDIO and SWCLK of both boards.

Chapter 3

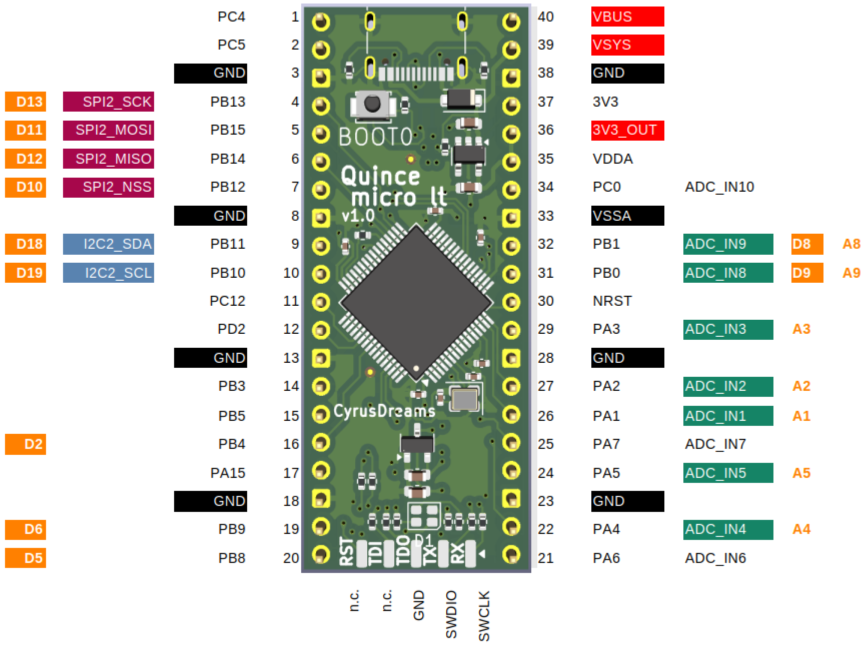
Developing Software for the Quince micro

3.1 MounRiver Studio

MounRiver Studio is the IDE preferred by WCH.

3.2 Arduino

WCH has released their own Arduino core. With it, the CH32V305-Microcontroller of the Qince micro can be programmed with the Arduino IDE. The board itself is not available in the IDE, but the peripherals can be addressed as on the development board 'CH32V305RBT6'. The following pins (D* and A*) are available:



3.3 PlatformIO

There is a GitHub repositry for CH32V in the PlatformIO IDE.

3.4 cnlohr / ch32fun

Developer and YouTuber CNLohr (Charles) released his own development environment for CH32V with the name ch32fun on GitHub.

3.5 Python

There is an experimental port of MicroPython for the CH32V307, this should also be useable for the CH32V305.

Annex

Revision History

Revision v0.9

First production data.

Revision v1.0

- Changed the connection of PB8 and PB9 to match the RaspberryPi pinout, enabling an additional I^{2}C bus.
- Connected 5V and 3.3V to the debug connector (non-switchable supply)

Copyright