

# BladeCore-M54E User Manual

Hardware Revision: 1.0.0 Date: 2026-01-31 Author: DMT

---

## Table of Contents

1. Introduction
  2. General Specifications
  3. Electrical Characteristics
  4. Microcontroller
  5. Memory Subsystem
  6. Ethernet Interface
  7. USB Interface
  8. Power Management
  9. M.2 Connector Pinout
  10. Onboard Peripherals
  11. Power-Up Sequencing
  12. Getting Started – Software
  13. Programming Guide
  14. Code Examples
  15. Design Considerations for Carrier Boards
  16. Mechanical Specifications
  17. Absolute Maximum Ratings
  18. Revision History
- 

## 1. Introduction

BladeCore-M54E is a compact, modular microcontroller card built around the **RP2354B** (ARM Cortex-M33, dual-core) combined with a **W5500 Ethernet controller** with integrated PHY. The module uses an **M.2 2980 M-Key** form factor (29 mm wide, 80 mm long) with a standard M-Key edge connector at a non-standard board width, making it suitable for embedding into carrier boards or systems that provide power and I/O connectivity through the M.2 edge connector.

### Key Features

- RP2354B dual-core ARM Cortex-M33 microcontroller
- 2 MB integrated flash + 16 MB external QSPI flash
- 256 kbit external I2C EEPROM with 200+ year data retention
- 100BASE-TX Ethernet via W5500 (SPI-attached, hardware TCP/IP stack)
- Dual-path USB 2.0 with automatic source selection (M.2 + USB-C)
- 30 GPIO pins + 6 ADC channels routed to M.2 edge connector
- Onboard 3.3 V LDO with power-good signaling
- Defined power-up sequencing
- Onboard heartbeat LED (PWM-capable)
- SWD debug interface

### BladeCore Series

BladeCore-M54E belongs to the BladeCore modular controller series:

Module	MCU	Primary Interface
BladeCore-M40	RP2040	–
BladeCore-M40E	RP2040	Ethernet
BladeCore-M40C	RP2040	CAN
BladeCore-M54	RP2354B	–
BladeCore-M54E	RP2354B	Ethernet
BladeCore-M54C	RP2354B	CAN

**Naming convention:** M40 = RP2040-based, M54 = RP2354B-based. Suffix E = Ethernet, C = CAN, no suffix = base module.

## 2. General Specifications

Parameter	Value
Microcontroller	RP2354B (ARM Cortex-M33, dual core)
Package	QFN-80-EP (10 x 10 mm)
System Clock	150 MHz, overclock possible (Tested max is 200MHz)
External Crystal	12 MHz, 10 pF load
Integrated Flash	2 MB (internal to MCU)
External Flash (QSPI)	16 MB (W25Q128JVPIQ, 133 MHz)
External EEPROM (I2C)	256 kbit (AT24C256C, 400 kHz)
Ethernet	100BASE-TX, W5500 (SPI, hardware TCP/IP)
USB	USB 2.0, dual path (M.2 + USB-C)
GPIO (M.2 connector)	28 digital I/O pins
ADC Channels (M.2)	6 channels (12-bit, GPIO40–GPIO45)
Onboard ADC Channels	2 (VBUS sense, 3.0 V reference)
Operating Voltage (I/O)	3.3 V
Form Factor	M.2 2980 M-Key
PCB Layers	6

## 3. Electrical Characteristics

### 3.1 Recommended Operating Conditions

Parameter	Symbol	Min	Typical	Max	Unit
Supply Voltage (5 V in)	VIN	4.75	5.0	5.5	V
Supply Voltage (3.3 V)	VCC	3.0	3.3	3.6	V
USB VBUS Voltage	VBUS	4.75	5.0	5.25	V
Operating Temperature	TA	-20	25	70	degC
I/O Logic Level (High)	VIH	2.0	–	3.3	V
I/O Logic Level (Low)	VIL	0	–	0.8	V

### 3.2 RP2354B Core Parameters

Parameter	Value
Core Voltage	1.1 V (internal)
GPIO Output Drive	2 / 4 / 8 / 12 mA
GPIO Input Hysteresis	Schmitt trigger
Internal Pull-Up/Down Resistance	50–80 kOhm
ADC Resolution	12 bits
ADC Reference Voltage	3.3 V (AVDD)

### 3.3 Power Consumption (Typical)

Condition	Current (3.3 V)	Notes
MCU idle, Ethernet off	~15 mA	RP2354B in normal idle
MCU active, Ethernet off	~40 mA	Both cores running

Condition	Current (3.3 V)	Notes
MCU active, Ethernet active	~180 mA	W5500 in full operation
MCU dormant (sleep)	<1 mA	Deep sleep, all peripherals off

**Note:** The above values are approximate. Actual consumption depends on clock frequency, active peripherals, GPIO loading, and Ethernet traffic. Consult the RP2354B and W5500 datasheets for detailed characterization.

### 3.4 LDO Regulator (TPS74801)

Parameter	Symbol	Min	Typical	Max	Unit
Input Voltage	VIN	1.1	5.0	5.5	V
Output Voltage	VOUT	–	3.3	–	V
Max Output Current	IOUT	–	–	1.5	A
Dropout Voltage	VDO	–	110	300	mV
Output Voltage Accuracy	–	–	+/-1	–	%
Thermal Shutdown	–	–	160	–	degC

### 3.5 Voltage Reference (LM4040B30FTA)

Parameter	Value
Reference Voltage	3.000 V
Tolerance	+/-0.1%
Package	SOT-23
Connected To	ADC7 (GPIO47) via 10K-10K divider

## 4. Microcontroller

### 4.1 RP2354B Overview

The RP2354B is a dual-core ARM Cortex-M33 microcontroller from Raspberry Pi Ltd. It features hardware floating-point, DSP extensions, and the RP2-series PIO (Programmable I/O) state machines.

Feature	Specification
Architecture	ARM Cortex-M33 (ARMv8-M)
Cores	2
Max Clock Speed	150 MHz
FPU	Single-precision hardware FPU
DSP	Yes (ARMv8-M DSP extension)
PIO State Machines	3 PIO blocks (PIO0, PIO1, PIO2)
DMA Channels	16
Timers	2 hardware timers
PWM Channels	24 (12 slices x 2 channels)
UART	2
SPI	2
I2C	2
USB	1 (USB 1.1 Host/Device)
ADC	12-bit SAR, 8 channels (+ temp)
Internal Flash	2 MB
SRAM	520 KB
Debug	SWD (Serial Wire Debug)
Package	QFN-80-EP (10 x 10 mm)

### 4.2 Clock Configuration

- **External Crystal:** 12 MHz (X3225 footprint), 10 pF load capacitance
- **System PLL:** Configurable up to 150 MHz system clock
- **USB PLL:** 48 MHz (required for USB operation)

### 4.3 GPIO Allocation on BladeCore-M54E

GPIO Range	Count	Assignment
GPIO0–27	28	Routed to M.2 edge connector (user I/O)
GPIO28–31	4	W5500 Ethernet SPI1 (MISO, CS, SCK, MOSI)
GPIO32–33	2	I2C0 – EEPROM (SDA, SCL)
GPIO34	1	W5500 hardware reset
GPIO35	1	W5500 interrupt (active low)
GPIO36	1	Onboard heartbeat LED (blue, PWM)
GPIO37–39	3	<b>Not connected</b> (do not use)
GPIO40–45	6	Routed to M.2 as ADC0–ADC5
GPIO46	1	Onboard ADC6 – USB VBUS sense
GPIO47	1	Onboard ADC7 – 3.0 V reference monitor

**Total GPIO available on M.2 connector: 34** (28 digital + 6 ADC-capable)

**Warning:** GPIO37, GPIO38, and GPIO39 are NOT connected on BladeCore-M54E. Attempting to use these pins will have no effect. The firmware header CONFIG.h includes compile-time guards to catch accidental use.

## 5. Memory Subsystem

### 5.1 Integrated Flash

- Built into the RP2354B
- Capacity: 2 MB
- Used for program storage and execution (XIP – execute in place)

### 5.2 External QSPI Flash (W25Q128JVPIQ)

Parameter	Value
Part Number	W25Q128JVPIQ (Winbond)
Capacity	16 MB (128 Mbit)
Interface	Quad SPI (QSPI)
Max Clock	133 MHz
Package	8-WSON (6 x 5 mm)
Erase Granularity	4 KB sector / 32 KB block / 64 KB block
Write Granularity	256 bytes (page program)
Endurance	100,000 erase cycles per sector

The external flash is connected to the RP2354B’s dedicated QSPI interface and can be used for additional program storage, file systems (e.g., LittleFS), or data logging.

### 5.3 External EEPROM (AT24C256C)

Parameter	Value
Part Number	AT24C256C-XHL-T (Microchip)
Capacity	256 kbit (32,768 x 8 bytes)
Interface	I2C (shared with I2C0)
I2C Address	0x50 (A0 = A1 = GND)
Max Clock	400 kHz (Fast mode)
Write Page Size	64 bytes
Write Cycle Time	5 ms (max)
Endurance	1,000,000 write cycles
Data Retention	200+ years
Package	8-TSSOP

Parameter	Value
Pull-Up Resistors	4.7 kOhm on SDA and SCL

The EEPROM is connected to I2C0 on GPIO32/33, which are internal to the module and not routed to the M.2 connector. When I2C0 is used on a different pin pair via the M.2 edge connector, it operates on a physically separate bus - address 0x50 is fully available on the carrier board side. The EEPROM is ideal for storing configuration data, calibration values, serial numbers, or small persistent datasets.

## 6. Ethernet Interface

### 6.1 W5500 Overview

Parameter	Value
Part Number	W5500 (WIZnet)
Standard	IEEE 802.3 100BASE-TX
PHY	Integrated
Hardware TCP/IP	Yes (8 simultaneous sockets)
Internal Buffer	32 KB (TX + RX, configurable)
Interface to MCU	SPI (up to 80 MHz)
Package	LQFP-48 (7 x 7 mm)
Operating Temp	-40 degC to +85 degC

The W5500 offloads the complete TCP/IP stack from the MCU. Supported protocols include TCP, UDP, IPv4, ICMP, ARP, IGMP, and PPPoE at the hardware level.

### 6.2 SPI Connection to MCU

Signal	GPIO	M.2 Pin	Direction (MCU view)
MISO	GPIO28	46 *	Input
CS	GPIO29	41 *	Output
SCK	GPIO30	40 *	Output
MOSI	GPIO31	39 *	Output

*These GPIO are used internally and are NOT available on the M.2 connector for user applications.*

Control	GPIO	Function
RST	GPIO34	Hardware reset (active low)
INT	GPIO35	Interrupt output (active low)

**SPI Configuration:** - SPI Instance: SPI1 - Baudrate: 10 MHz (default in CONFIG.h) - Mode: SPI Mode 0 (CPOL=0, CPHA=0)

### 6.3 Ethernet Signals on M.2 Connector

M.2 Pin	Signal	Description
18	LINKLED	Ethernet link indicator
20	ACTLED	Ethernet activity indicator
24	RCT	Center tap, receive transformer
28	ETH_RD+	Receive differential pair (+)
30	ETH_RD-	Receive differential pair (-)
34	ETH_TD+	Transmit differential pair (+)
36	ETH_TD-	Transmit differential pair (-)
40	TCT	Center tap, transmit transformer

The carrier board must provide Ethernet magnetics (transformers) and an RJ45 connector. The LINKLED and ACTLED pins can drive indicator LEDs on the carrier board (active low).

## 6.4 Reference Crystal

- Frequency: 25 MHz
- Load Capacitance: 12 pF
- Package: X3225

## 7. USB Interface

### 7.1 Dual-Path Architecture

BladeCore-M54E provides two physical USB connection paths:

1. **M.2 edge connector USB** – for connection through the carrier board
2. **Onboard USB-C connector** – for direct access during development or provisioning

A **FSUSB42MUX** (OnSemi) USB 2.0 multiplexer automatically selects the active path:

Condition	Active USB Path
No USB-C cable connected	M.2 edge connector
USB-C cable connected	Onboard USB-C

This selection is automatic – no firmware intervention or manual jumper configuration is required.

### 7.2 M.2 USB Signals

M.2 Pin	Signal	Description
69	USB_D+	USB Data +
71	USB_D-	USB Data -
72, 74	VBUS	USB VBUS (5V)
75	VBUS	USB VBUS (5V)

### 7.3 USB-C Connector

- Type: USB-TYPE-C-018 receptacle
- Supports plugging/unplugging during operation (hot-swap)
- CC resistors onboard for proper USB-C enumeration
- ESD protection: SP0503BAHTG on data lines

### 7.4 USB VBUS Sensing

The USB VBUS voltage is monitored via GPIO46 (ADC6) through a resistive voltage divider:

- Divider: 5.1 kOhm / 5.1 kOhm
- ADC input voltage at 5.0 V VBUS: ~2.5 V
- Formula:  $VBUS = ADC\_reading * (3.3 / 4096) * 2$

This allows firmware to detect whether USB power is present and measure the actual VBUS voltage.

## 8. Power Management

### 8.1 Power Input Options

BladeCore-M54E can be powered through four paths:

Source	Input Pins	Path
M.2 5 V	Pins 14, 16 (+5V INPUT)	5 V -> TPS74801 LDO -> 3.3 V rail
M.2 VBUS	Pins 72, 74, 75 (VBUS)	VBUS -> TPS74801 LDO -> 3.3 V rail
Onboard USB-C	USB-C connector	VBUS -> TPS74801 LDO -> 3.3 V rail
External 3.3 V	Pins 10, 12 (+3.3V OUTPUT)	Direct to 3.3 V rail (LDO bypassed)

### 8.2 3.3 V Output to Carrier Board

When the module is powered from 5 V (pins 14/16, VBUS, or USB-C), the regulated 3.3 V rail is provided back to the carrier board on M.2 pins 10 and 12 (+3.3V OUTPUT). The LDO can supply **1.5 A** total. Each M.2 contact is rated for 0.5 A, so with two pins the connector limits the carrier board output to **1 A**. The remaining 0.5 A is the module's own budget, of which ~180 mA is typically consumed (with Ethernet active).

When the module is powered externally at 3.3 V on these same pins, the carrier board supplies both the module and itself – the LDO is not active in this mode.

### 8.3 Voltage Regulator – TPS74801DRCT

Parameter	Value
Type	Low-Dropout (LDO)
Output Voltage	3.3 V
Max Output Current	1.5 A
Dropout Voltage	110 mV typical
Package	VSON-10-1EP (3x3 mm)
Enable Pin	VEN (active high)
Power-Good Output	PGOOD (open drain)

### 8.4 Power-Good Indication

The LDO provides a PGOOD signal that goes high once the output voltage reaches regulation. This signal is buffered through an **SN74LVC1G97QDCKRQ1** logic gate before driving an onboard green LED (visual indicator).

### 8.5 Onboard Voltage Reference

A precision **LM4040B30FTA** (3.000 V, +/-0.1%) voltage reference is connected to GPIO47 (ADC7) through a 10K / 10K voltage divider. This enables: - ADC calibration at runtime (known reference voltage) - Detection of supply rail drift - High-accuracy analog measurements

#### To calibrate the ADC:

Expected ADC reading for 3.0V reference through 10K/10K divider:

$$V_{ADC} = 3.0 / 2 = 1.5 \text{ V}$$

$$\text{Expected ADC code} = 1.5 / 3.3 * 4096 = \sim 1862$$

$$\text{Calibration factor} = 1862 / \text{actual\_adc\_reading}$$

Apply this factor to all subsequent ADC readings.

## 9. M.2 Connector Pinout

### 9.1 Left Side (Odd Pins)

Pin	Signal	GPIO	Alternate Functions
1	GND	–	Ground
3	GPIO24	GPIO24	SPI0 TX, UART1 RTS, I2C1 SCL, PWM3 B, PIO0/1/2
5	GPIO23	GPIO23	SPI0 TX, UART1 RTS, I2C1 SCL, PWM3 B, PIO0/1/2
7	GPIO22	GPIO22	SPI0 SCK, UART1 CTS, I2C1 SDA, PWM3 A, PIO0/1/2
9	GPIO21	GPIO21	SPI0 CSn, UART1 RX, I2C0 SCL, PWM2 B, PIO0/1/2
11	GND	–	Ground
13	GPIO20	GPIO20	SPI0 RX, UART1 TX, I2C0 SDA, PWM2 A, PIO0/1/2
15	GPIO19	GPIO19	SPI0 TX, UART0 RTS, I2C1 SCL, PWM1 B, QMI CS1n

Pin	Signal	GPIO	Alternate Functions
17	GPIO18	GPIO18	SPI0 SCK, UART0 CTS, I2C1 SDA, PWM1 A, PIO0/1/2
19	GPIO17	GPIO17	SPI0 CSn, UART0 RX, I2C0 SCL, PWM0 B, PIO0/1/2
21	GPIO16	GPIO16	SPI0 RX, UART0 TX, I2C0 SDA, PWM0 A, PIO0/1/2
23	GND	–	Ground
25	GPIO15	GPIO15	SPI1 TX, UART0 RTS, I2C1 SCL, PWM7 B, PIO0/1/2
27	GPIO14	GPIO14	SPI1 SCK, UART0 CTS, I2C1 SDA, PWM7 A, PIO0/1/2
29	GPIO13	GPIO13	SPI1 CSn, UART0 RX, I2C0 SCL, PWM6 B, PIO0/1/2
31	GPIO12	GPIO12	SPI1 RX, UART0 TX, I2C0 SDA, PWM6 A, PIO0/1/2
33	GPIO11	GPIO11	SPI1 TX, UART1 RTS, I2C1 SCL, PWM5 B, PIO0/1/2
35	GPIO10	GPIO10	SPI1 SCK, UART1 CTS, I2C1 SDA, PWM5 A, PIO0/1/2
37	GPIO9	GPIO9	SPI1 CSn, UART1 RX, I2C0 SCL, PWM4 B, PIO0/1/2
39	GPIO8	GPIO8	SPI1 RX, UART1 TX, I2C0 SDA, PWM4 A, QMI CS1n
41	GND	–	Ground
43	GPIO7	GPIO7	SPI0 TX, UART1 RTS, I2C1 SCL, PWM3 B, PIO0/1/2
45	GPIO6	GPIO6	SPI0 SCK, UART1 CTS, I2C1 SDA, PWM3 A, PIO0/1/2
47	GPIO5	GPIO5	SPI0 CSn, UART1 RX, I2C0 SCL, PWM2 B, PIO0/1/2
49	GPIO4	GPIO4	SPI0 RX, UART1 TX, I2C0 SDA, PWM2 A, PIO0/1/2
51	GPIO3	GPIO3	SPI0 TX, UART0 RTS, I2C1 SCL, PWM1 B, PIO0/1/2
53	GPIO2	GPIO2	SPI0 SCK, UART0 CTS, I2C1 SDA, PWM1 A, PIO0/1/2
55	GPIO1	GPIO1	SPI0 CSn, UART0 RX, I2C0 SCL, PWM0 B, TRACECLK
57	GPIO0	GPIO0	SPI0 RX, UART0 TX, I2C0 SDA, PWM0 A, QMI CS1n
67	GND	–	Ground
69	USB_D+	–	USB Data Plus
71	USB_D-	–	USB Data Minus
73	GND	–	Ground
75	VBUS	–	USB VBUS (5 V)

## 9.2 Right Side (Even Pins)

Pin	Signal	GPIO	Notes
2	GND	–	Ground
4	GPIO27	GPIO27	SPI1 TX, UART1 RTS, I2C1 SCL, PWM5 B
6	GPIO26	GPIO26	SPI1 SCK, UART1 CTS, I2C1 SDA, PWM5 A
8	GPIO25	GPIO25	SPI1 CSn, UART1 RX, I2C0 SCL, PWM4 B
10	+3.3V OUTPUT	–	3.3 V LDO output / external 3.3 V input
12	+3.3V OUTPUT	–	3.3 V LDO output / external 3.3 V input
14	+5V INPUT	–	5 V power input to module

Pin	Signal	GPIO	Notes
16	+5V INPUT	–	5 V power input to module
18	LINKLED	–	Ethernet link indicator (active low)
20	ACTLED	–	Ethernet activity indicator (active low)
22	GND	–	Ground
24	RCT	–	Receive center tap (Ethernet magnetics)
26	GND	–	Ground
28	ETH_RD+	–	Ethernet receive differential (+)
30	ETH_RD-	–	Ethernet receive differential (-)
32	GND	–	Ground
34	ETH_TD+	–	Ethernet transmit differential (+)
36	ETH_TD-	–	Ethernet transmit differential (-)
38	GND	–	Ground
40	TCT	–	Transmit center tap (Ethernet magnetics)
42	GND	–	Ground
44	GND	–	Ground
46	ADC0/GPIO40	GPIO40	Analog/digital, PWM8 A, SPI1, UART1, I2C0
48	ADC1/GPIO41	GPIO41	Analog/digital, PWM8 B, SPI1, UART1, I2C0
50	ADC2/GPIO42	GPIO42	Analog/digital, PWM9 A, SPI1, UART1, I2C1
52	ADC3/GPIO43	GPIO43	Analog/digital, PWM9 B, SPI1, UART1, I2C1
54	ADC4/GPIO44	GPIO44	Analog/digital, PWM10 A, SPI1, UART0, I2C0
56	ADC5/GPIO45	GPIO45	Analog/digital, PWM10 B, SPI1, UART0, I2C0
58	GND	–	Ground
68	GND	–	Ground
70	GND	–	Ground
72	VBUS	–	USB VBUS (5 V)
74	VBUS	–	USB VBUS (5 V)
SH1	SHIELD	–	Shield / Ground

### 9.3 Important Notes

- **SPI1** (GPIO8–15, GPIO25–31, GPIO40–45) overlaps with both user-accessible pins and the onboard W5500. GPIO28–31 are dedicated to the W5500 and are **not** available on the M.2 connector. User code must avoid configuring GPIO28–31 for other purposes.
- **I2C0** can be used on any M.2-accessible GPIO with an I2C0 alternate function. The onboard EEPROM uses GPIO32–33, which are not wired to the M.2 connector, so the carrier board's I2C0 bus is physically independent.
- **GPIO37, GPIO38, GPIO39** are not connected. Do not assign any function to these pins in firmware.

## 10. Onboard Peripherals

### 10.1 Heartbeat LED

Parameter	Value
GPIO	GPIO36
Color	Blue
Package	0402 (1005 Metric)
Series Resistor	1000 Ohm
PWM Slice	Slice 2, Channel A

Parameter	Value
PWM Frequency	1000 Hz (configurable)

The heartbeat LED is intended as a visual “alive” indicator. It can be driven with simple digital on/off or with hardware PWM for breathing/fading effects.

## 10.2 Reset Buttons

Two tactile switches (B3U-1000P, Omron) are provided on the board for manual reset and boot-mode selection.

## 10.3 Status LEDs

LED	Color	Designator	Function
+5V	Green	D10	Connected to REGIN; active when VBUS or +5V is present
PGOOD	Green	D6	Driven via SN74LVC1G97 buffer; active when 3.3V is OK
VBUS	Green	D11	SN74LVC1G97 as 2-input OR gate; monitors onboard + M.2 VBUS
Heartbeat	Blue	D3	Connected to MCU GPIO36; PWM breathing indicator

## 10.4 ESD Protection

Device	Protection
SP0503BAHTG	USB data line ESD clamping
RCLAMP0582BQTCT	General I/O ESD clamping

## 11. Power-Up Sequencing

The board follows a defined power-up sequence with the following timeline:

Step	Event	Time (cumulative)
1	VIN applied (5.0 V)	0 ms
2	VEN asserted (TPS74801 enables)	0 ms
3	VOOUT (3.3 V) reaches regulation	~4 ms (soft-start)
4	PGOOD asserts (power-good valid)	~4 ms
5	RP2354B 3.3 V rail stable	~4 ms
6	RP2354B 1.1 V core valid (POR)	~4.01 ms
7	RP2354B ready for execution	~4.09 ms
8	W5500 fully initialized	~54 ms

**Note:** The W5500 requires approximately 50 ms after power stabilization for its internal PLL to lock and the PHY to initialize. Firmware should wait at least 50 ms after asserting the W5500 reset pin before attempting SPI communication.

## 12. Getting Started – Software

### 12.1 Prerequisites

- **Raspberry Pi Pico SDK** version 2.2.0 or later
- **ARM GCC Toolchain** version 14.2 or later
- **CMake** version 3.13 or later
- **Picotool** version 2.2.0 or later (for UF2 flashing)
- A USB cable (USB-C or via carrier board M.2 connection)

## 12.2 Environment Setup (Windows)

1. Install the Raspberry Pi Pico VS Code Extension which automatically installs and manages the SDK, toolchain, and picotool.
2. Alternatively, install manually:

```
# Install Pico SDK
git clone https://github.com/raspberrypi/pico-sdk.git
cd pico-sdk
git checkout 2.2.0
git submodule update --init

# Set environment variable
set PICO_SDK_PATH=C:\path\to\pico-sdk
```

## 12.3 Building the Heartbeat Example

```
cd src/Software/Heartbeat
```

```
mkdir build
cd build
```

```
cmake -G "Ninja" ..
ninja
```

This produces `Heartbeat.uf2` in the build directory.

## 12.4 Flashing Firmware

1. Hold the BOOTSEL button on the RP2354B while connecting USB (or pressing reset).
2. The board enumerates as a USB mass storage device (RP2350 drive).
3. Copy the `.uf2` file to the drive:  

```
copy Heartbeat.uf2 E:\
```

(Replace `E:\` with the actual drive letter.)
4. The board automatically reboots and begins executing.

Alternatively, use picotool:

```
picotool load Heartbeat.uf2
picotool reboot
```

## 12.5 Serial Monitor (USB CDC)

The Heartbeat example enables USB stdio output. Connect a terminal at any baud rate (USB CDC is rate-independent):

```
# Using PuTTY, Tera Term, or any serial monitor
# Select the COM port assigned to the Pico
```

---

# 13. Programming Guide

## 13.1 Pin Configuration Header

All pin assignments are centralized in `CONFIG.h`. Always use these definitions instead of hardcoding GPIO numbers:

```
#include "CONFIG.h"
```

```
// Ethernet SPI
PIN_ETH_MISO // GPIO28 - SPI1 RX
PIN_ETH_CS   // GPIO29 - SPI1 CSn
PIN_ETH_SCK  // GPIO30 - SPI1 SCK
```

```
PIN_ETH_MOSI    // GPIO31 - SPI1 TX

// I2C EEPROM
PIN_I2C0_SDA    // GPIO32
PIN_I2C0_SCL    // GPIO33

// Ethernet control
PIN_ETH_RST     // GPIO34 - W5500 reset (active Low)
PIN_ETH_INT     // GPIO35 - W5500 interrupt (active Low)

// Heartbeat LED
PIN_HEARTBEAT   // GPIO36

// ADC
PIN_ADC_VUSB    // GPIO46 / ADC6 - USB VBUS sense
PIN_ADC_VREF    // GPIO47 / ADC7 - Voltage reference
```

## 13.2 SPI1 – Ethernet (W5500)

```
#include "hardware/spi.h"
#include "CONFIG.h"

void w5500_spi_init(void) {
    spi_init(ETH_SPI_INSTANCE, ETH_SPI_BAUDRATE);

    gpio_set_function(PIN_ETH_MISO, GPIO_FUNC_SPI);
    gpio_set_function(PIN_ETH_MOSI, GPIO_FUNC_SPI);
    gpio_set_function(PIN_ETH_SCK, GPIO_FUNC_SPI);

    // CS is managed manually (active Low)
    gpio_init(PIN_ETH_CS);
    gpio_set_dir(PIN_ETH_CS, GPIO_OUT);
    gpio_put(PIN_ETH_CS, 1); // Deselect

    // Reset pin
    gpio_init(PIN_ETH_RST);
    gpio_set_dir(PIN_ETH_RST, GPIO_OUT);
    gpio_put(PIN_ETH_RST, 1); // Not in reset

    // Interrupt pin
    gpio_init(PIN_ETH_INT);
    gpio_set_dir(PIN_ETH_INT, GPIO_IN);
    gpio_pull_up(PIN_ETH_INT);
}

void w5500_reset(void) {
    gpio_put(PIN_ETH_RST, 0);
    sleep_ms(1); // Hold reset for at least 500 us
    gpio_put(PIN_ETH_RST, 1);
    sleep_ms(50); // Wait for W5500 PLL Lock + PHY init
}

// Basic SPI read/write for W5500 register access
void w5500_write(uint16_t addr, uint8_t block, uint8_t *data, uint16_t len) {
    uint8_t header[3];
    header[0] = (uint8_t)(addr >> 8);
    header[1] = (uint8_t)(addr & 0xFF);
    header[2] = (block << 3) | (1 << 2); // Write mode
```

```

    gpio_put(PIN_ETH_CS, 0);
    spi_write_blocking(ETH_SPI_INSTANCE, header, 3);
    spi_write_blocking(ETH_SPI_INSTANCE, data, len);
    gpio_put(PIN_ETH_CS, 1);
}

void w5500_read(uint16_t addr, uint8_t block, uint8_t *data, uint16_t len) {
    uint8_t header[3];
    header[0] = (uint8_t)(addr >> 8);
    header[1] = (uint8_t)(addr & 0xFF);
    header[2] = (block << 3) | (0 << 2); // Read mode

    gpio_put(PIN_ETH_CS, 0);
    spi_write_blocking(ETH_SPI_INSTANCE, header, 3);
    spi_read_blocking(ETH_SPI_INSTANCE, 0x00, data, len);
    gpio_put(PIN_ETH_CS, 1);
}

```

### 13.3 I2C0 – EEPROM (AT24C256)

```

#include "hardware/i2c.h"
#include "CONFIG.h"

void eeprom_init(void) {
    i2c_init(EEPROM_I2C_INSTANCE, EEPROM_I2C_BAUDRATE);
    gpio_set_function(PIN_I2C0_SDA, GPIO_FUNC_I2C);
    gpio_set_function(PIN_I2C0_SCL, GPIO_FUNC_I2C);
    // External 4.7K pull-ups are on the board -- no internal pull-ups needed
}

// Write up to 64 bytes (one EEPROM page) at a time
int eeprom_write(uint16_t mem_addr, const uint8_t *data, size_t len) {
    uint8_t buf[66]; // 2 address bytes + 64 data bytes max
    buf[0] = (uint8_t)(mem_addr >> 8);
    buf[1] = (uint8_t)(mem_addr & 0xFF);

    if (len > 64) len = 64;
    for (size_t i = 0; i < len; i++) {
        buf[2 + i] = data[i];
    }

    int ret = i2c_write_blocking(EEPROM_I2C_INSTANCE, EEPROM_I2C_ADDR,
                                buf, 2 + len, false);
    sleep_ms(5); // Wait for write cycle to complete
    return ret;
}

// Read arbitrary number of bytes
int eeprom_read(uint16_t mem_addr, uint8_t *data, size_t len) {
    uint8_t addr_buf[2];
    addr_buf[0] = (uint8_t)(mem_addr >> 8);
    addr_buf[1] = (uint8_t)(mem_addr & 0xFF);

    int ret = i2c_write_blocking(EEPROM_I2C_INSTANCE, EEPROM_I2C_ADDR,
                                addr_buf, 2, true); // No stop

    if (ret < 0) return ret;

    return i2c_read_blocking(EEPROM_I2C_INSTANCE, EEPROM_I2C_ADDR,
                             data, len, false);
}

```

```
}
```

### 13.4 ADC – Voltage Monitoring

```
#include "hardware/adc.h"
#include "CONFIG.h"

void adc_monitoring_init(void) {
    adc_init();
    adc_gpio_init(PIN_ADC_VUSB); // GPIO46 / ADC6
    adc_gpio_init(PIN_ADC_VREF); // GPIO47 / ADC7
}

// Read USB VBUS voltage (5V nominal)
float read_vbus_voltage(void) {
    adc_select_input(6); // ADC6 = GPIO46
    adc_read(); // Throwaway sample after channel switch
    uint16_t raw = adc_read();
    // 5.1K / 5.1K divider, factor = 2
    return (float)raw * 3.3f / 4096.0f * 2.0f;
}

// Read precision voltage reference (3.0V nominal)
float read_vref(void) {
    adc_select_input(7); // ADC7 = GPIO47
    adc_read(); // Throwaway sample after channel switch
    uint16_t raw = adc_read();
    // 10K / 10K divider, factor = 2
    return (float)raw * 3.3f / 4096.0f * 2.0f;
}

// Calculate ADC calibration factor using known reference
float get_adc_calibration_factor(void) {
    adc_select_input(7);
    adc_read(); // Throwaway sample after channel switch
    uint16_t raw = adc_read();
    float measured = (float)raw * 3.3f / 4096.0f * 2.0f;
    return 3.0f / measured; // Ratio of known vs measured
}
}
```

### 13.5 PWM – Heartbeat LED

```
#include "hardware/pwm.h"
#include "hardware/clocks.h"
#include "CONFIG.h"

#define PWM_WRAP 65535

void heartbeat_init(void) {
    gpio_set_function(PIN_HEARTBEAT, GPIO_FUNC_PWM);

    uint slice = pwm_gpio_to_slice_num(PIN_HEARTBEAT);
    pwm_config cfg = pwm_get_default_config();

    float divider = (float)clock_get_hz(clk_sys) /
        ((float)HEARTBEAT_PWM_FREQ_HZ * (PWM_WRAP + 1));
    if (divider < 1.0f) divider = 1.0f;

    pwm_config_set_clkdiv(&cfg, divider);
    pwm_config_set_wrap(&cfg, PWM_WRAP);
}
```

```
pwm_init(slice, &cfg, true);
pwm_set_chan_level(slice, pwm_gpio_to_channel(PIN_HEARTBEAT), 0);
}

// Set LED brightness: 0 = off, 65535 = full
void heartbeat_set(uint16_t brightness) {
    uint slice = pwm_gpio_to_slice_num(PIN_HEARTBEAT);
    uint channel = pwm_gpio_to_channel(PIN_HEARTBEAT);
    pwm_set_chan_level(slice, channel, brightness);
}
```

### 13.6 UART (via M.2 Connector)

```
#include "hardware/uart.h"

// Example: UART0 on GPIO0 (TX) and GPIO1 (RX) -- M.2 pins 57 and 55
void uart_example_init(void) {
    uart_init(uart0, 115200);
    gpio_set_function(0, GPIO_FUNC_UART); // UART0 TX
    gpio_set_function(1, GPIO_FUNC_UART); // UART0 RX
}
```

### 13.7 CMakeLists.txt Template

```
cmake_minimum_required(VERSION 3.13)

set(CMAKE_C_STANDARD 11)
set(CMAKE_CXX_STANDARD 17)
set(CMAKE_EXPORT_COMPILE_COMMANDS ON)

# Pico SDK integration (VS Code extension auto-config)
if(WIN32)
    set(USERHOME $ENV{USERPROFILE})
else()
    set(USERHOME $ENV{HOME})
endif()
set(sdkVersion 2.2.0)
set(toolchainVersion 14_2_Rel1)
set(picotoolVersion 2.2.0-a4)
set(picoVscode ${USERHOME}/.pico-sdk/cmake/pico-vscode.cmake)
if (EXISTS ${picoVscode})
    include(${picoVscode})
endif()

set(PICO_BOARD pico2 CACHE STRING "Board type")

include(pico_sdk_import.cmake)

project(MyProject C CXX ASM)
pico_sdk_init()

add_executable(MyProject main.c)

pico_set_program_name(MyProject "MyProject")
pico_set_program_version(MyProject "0.1")

# Enable USB output, disable UART output
pico_enable_stdio_uart(MyProject 0)
pico_enable_stdio_usb(MyProject 1)
```

```
target_link_libraries(MyProject
    pico_stdlib
    hardware_spi      # For W5500 Ethernet
    hardware_i2c      # For EEPROM
    hardware_adc       # For ADC monitoring
    hardware_pwm       # For heartbeat LED
)

target_include_directories(MyProject PRIVATE
    ${CMAKE_CURRENT_LIST_DIR}
)

pico_add_extra_outputs(MyProject)
```

---

## 14. Code Examples

### 14.1 Heartbeat LED (Breathing Effect)

The included `Heartbeat.c` example demonstrates a smooth PWM breathing pattern on the onboard blue LED. See `src/Software/Heartbeat/` for the complete project.

**Behavior:** The LED linearly fades from off to full brightness and back over approximately 2 seconds (128 steps x 2 directions x 8 ms per step).

### 14.2 Minimal W5500 Initialization

```
#include <stdio.h>
#include "pico/stdlib.h"
#include "hardware/spi.h"
#include "CONFIG.h"

int main(void) {
    stdio_init_all();

    // Initialize SPI1 for W5500
    spi_init(ETH_SPI_INSTANCE, ETH_SPI_BAUDRATE);
    gpio_set_function(PIN_ETH_MISO, GPIO_FUNC_SPI);
    gpio_set_function(PIN_ETH_MOSI, GPIO_FUNC_SPI);
    gpio_set_function(PIN_ETH_SCK, GPIO_FUNC_SPI);

    gpio_init(PIN_ETH_CS);
    gpio_set_dir(PIN_ETH_CS, GPIO_OUT);
    gpio_put(PIN_ETH_CS, 1);

    // Hardware reset the W5500
    gpio_init(PIN_ETH_RST);
    gpio_set_dir(PIN_ETH_RST, GPIO_OUT);
    gpio_put(PIN_ETH_RST, 0);
    sleep_ms(1);
    gpio_put(PIN_ETH_RST, 1);
    sleep_ms(50);

    // Read W5500 version register (address 0x0039, common register block 0)
    uint8_t header[3] = {0x00, 0x39, 0x00}; // Read, common block
    uint8_t version = 0;

    gpio_put(PIN_ETH_CS, 0);
    spi_write_blocking(ETH_SPI_INSTANCE, header, 3);
    spi_read_blocking(ETH_SPI_INSTANCE, 0x00, &version, 1);
```

```
gpio_put(PIN_ETH_CS, 1);

printf("W5500 Version: 0x%02X (expected 0x04)\n", version);

while (true) {
    tight_loop_contents();
}
}
```

### 14.3 EEPROM Read/Write Test

```
#include <stdio.h>
#include <string.h>
#include "pico/stdlib.h"
#include "hardware/i2c.h"
#include "CONFIG.h"

int main(void) {
    stdio_init_all();
    sleep_ms(2000); // Wait for USB enumeration

    // Initialize I2C0
    i2c_init(EEPROM_I2C_INSTANCE, EEPROM_I2C_BAUDRATE);
    gpio_set_function(PIN_I2C0_SDA, GPIO_FUNC_I2C);
    gpio_set_function(PIN_I2C0_SCL, GPIO_FUNC_I2C);

    // Write test data to EEPROM address 0x0000
    uint8_t write_buf[] = {0x00, 0x00, 'H', 'e', 'l', 'l', 'o'};
    int ret = i2c_write_blocking(EEPROM_I2C_INSTANCE, EEPROM_I2C_ADDR,
                                write_buf, sizeof(write_buf), false);
    printf("Write returned: %d\n", ret);
    sleep_ms(5); // EEPROM write cycle

    // Read back
    uint8_t addr_buf[] = {0x00, 0x00};
    i2c_write_blocking(EEPROM_I2C_INSTANCE, EEPROM_I2C_ADDR,
                       addr_buf, 2, true);

    uint8_t read_buf[5] = {0};
    i2c_read_blocking(EEPROM_I2C_INSTANCE, EEPROM_I2C_ADDR,
                     read_buf, 5, false);

    printf("Read back: %c%c%c%c%c\n",
           read_buf[0], read_buf[1], read_buf[2],
           read_buf[3], read_buf[4]);

    while (true) {
        tight_loop_contents();
    }
}
```

### 14.4 ADC Calibration and VBUS Monitoring

```
#include <stdio.h>
#include "pico/stdlib.h"
#include "hardware/adc.h"
#include "CONFIG.h"

int main(void) {
    stdio_init_all();
```

```
sleep_ms(2000);

adc_init();
adc_gpio_init(PIN_ADC_VUSB);
adc_gpio_init(PIN_ADC_VREF);

// Calibrate using onboard 3.0V reference
adc_select_input(7); // ADC7 = GPIO47 (VREF)
adc_read(); // Throwaway sample after channel switch
uint16_t ref_raw = adc_read();
float ref_measured = (float)ref_raw * 3.3f / 4096.0f * 2.0f;
float cal_factor = 3.0f / ref_measured;

printf("Reference ADC raw: %u\n", ref_raw);
printf("Measured: %.3f V (expected 3.000 V)\n", ref_measured);
printf("Calibration factor: %.4f\n", cal_factor);

while (true) {
    // Read VBUS
    adc_select_input(6); // ADC6 = GPIO46 (VUSB)
    adc_read(); // Throwaway sample after channel switch
    uint16_t vbus_raw = adc_read();
    float vbus = (float)vbus_raw * 3.3f / 4096.0f * 2.0f * cal_factor;

    printf("VBUS: %.2f V (raw: %u)\n", vbus, vbus_raw);
    sleep_ms(1000);
}
}
```

---

## 15. Design Considerations for Carrier Boards

### 15.1 Ethernet Magnetics

The carrier board MUST provide Ethernet magnetics (1:1 isolation transformers) and an RJ45 connector. The differential pairs on the M.2 connector are:

- **TX:** Pins 34 (+) and 36 (-)
- **RX:** Pins 28 (+) and 30 (-)
- **Center taps:** Pins 24 (RCT) and 40 (TCT)

Route differential pairs with controlled impedance (100 Ohm differential) and keep trace lengths matched within 50 mils.

### 15.2 Power Supply Design

- **5 V mode:** Provide stable 5 V on M.2 pins 14 and 16 (+5V INPUT). Minimum current capacity: 500 mA (to support MCU + W5500 + peripherals). Recommended: 1 A or more if the carrier board also draws from the 3.3 V output. M.2 pins 10 and 12 provide **3.3 V output** from the onboard LDO (up to 1 A available for carrier board electronics).
- **External 3.3 V mode:** Supply regulated 3.3 V on M.2 pins 10 and 12. The carrier board supplies all current (module draws ~180 mA typical with Ethernet active).
- Add bulk capacitance (47-100 uF) near the M.2 connector on the carrier board.

### 15.3 USB Routing

- Route USB\_D+ (pin 69) and USB\_D- (pin 71) as a 90 Ohm differential pair.
- Keep trace lengths matched.
- Provide VBUS on pins 72, 74, 75 if USB-powered operation is desired.

## 15.4 I2C Bus Extension

- I2C0 can be used on any M.2-accessible GPIO with the I2C0 alternate function. The onboard EEPROM (GPIO32–33) is not wired to the M.2 connector, so the carrier board operates on a physically separate I2C bus – all addresses including 0x50 are available.
- Total bus capacitance should not exceed 400 pF for reliable 400 kHz operation.

## 15.5 SPI Bus Sharing

- SPI1 is occupied by the W5500 (GPIO28–31, directly on the module).
- SPI0 is fully available on the M.2 connector (multiple GPIO options).
- If using SPI1 on M.2 GPIO pins (GPIO8–15, GPIO25–27, GPIO40–45), make sure the W5500 CS (GPIO29) is deasserted (high) to avoid bus contention.

## 15.6 Antenna and EMI

- The Ethernet differential pairs carry high-frequency signals. Route them on inner layers if possible, with solid ground planes above and below.
  - Keep the Ethernet trace area away from high-speed digital signals (SPI clock, USB data lines).
- 

## 16. Mechanical Specifications

Parameter	Value
Form Factor	M.2 2980 M-Key
Card Dimensions	29 mm x 80 mm
Thickness	~1.6 mm (6-layer PCB)
Edge Connector	M-Key (75 pins)
Mating Connector	MDT420E03001 (or equiv)
Mounting	M2.5 threaded standoff
Weight	< 10 g

---

## 17. Absolute Maximum Ratings

Exceeding absolute maximum ratings may cause permanent damage to the module.

Parameter	Rating
VIN (5 V input)	-0.3 to 6.0 V
VCC (3.3 V rail)	-0.3 to 3.6 V
GPIO voltage (any pin)	-0.3 to 3.63 V
Maximum GPIO source current	12 mA per pin
Maximum total GPIO current	50 mA
Storage Temperature	-40 to +85 degC
Operating Temperature	-20 to +70 degC
ESD (HBM)	2 kV

**Caution:** The RP2354B GPIOs are **NOT** 5 V tolerant. All signal levels connected to the M.2 connector must be 3.3 V logic. Use level shifters on the carrier board if interfacing with 5 V systems.

---

## 18. Revision History

Revision	Date	Notes
1.0.0	2026-01-31	Initial creation

---

*This document is derived from the BladeCore-M54E schematics (v1.0.0) and associated design files. Component-level limits and behavior should be verified against the respective datasheets. For the latest information, refer to the project repository. Hardware licensed under CC BY-NC-SA 4.0. Contact: [info@dm-t-hw.com](mailto:info@dm-t-hw.com) | GitHub: [DvidMakesThings](#)*