MODM7AE70LC Quickstart Guide

NetBurner, Inc. 18-Jun-2026

1 Introduction

1.1 Software Licensing

The Software included in your NetBurner Network Development Kit is licensed to run only on NetBurner provided hardware. Please read the license.txt file located (by default) in your \Nburn\docs directory.

If your application involves manufacturing your own hardware, please contact sales@netburner.com for details on a royalty-free software license.

1.2 MODM7AE70 Module

Figure 1 MODM7AE70

NetBurner Ethernet Core Module

300 MHz Microchip SAM E70 (ATSAME70Q21B) | 2 MB Flash | 8 MB SDRAM | 384 kB SRAM | 10/100 Ethernet | -40 to +85 C

2 Kit Contents

Development Kit Contents:

- MODM7AE70 System on a Module
- DEV-MOD-105 Carrier Board
- Mini USB Cable for power and/or serial communication
- Ethernet Cable
- Red card with NNDK software and tools registration number and download link

3 Development Board

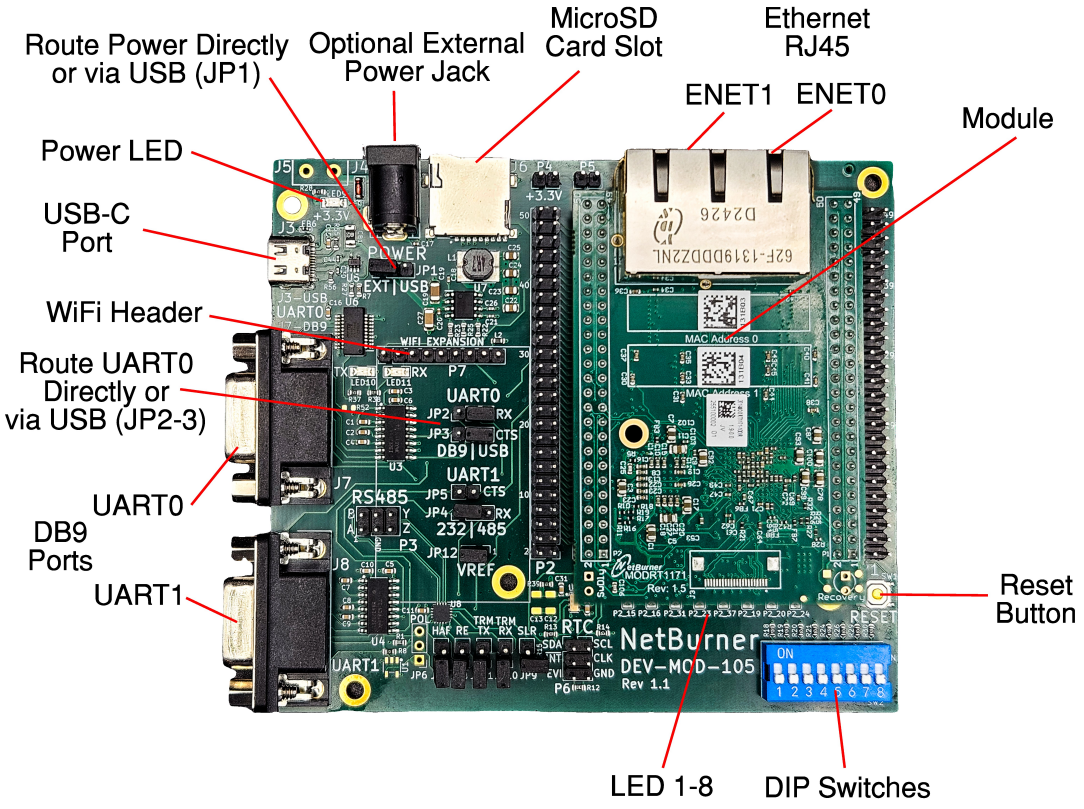

Figure 2 DEV-MOD-105 Development Board

The DEV-MOD-105 is a development carrier for the MODM7AE70 system-on-module. Plug a MODM7AE70 into the dev board's two 50-pin sockets and you have an immediately useful prototyping platform: power input, 10/100 Ethernet (on-module RJ-45 on the 100 version, or routed through the dev board on the 200 version), two configurable UARTs (RS-232, RS-485, or USB-Serial), a microSD socket, an SPI expansion header for a WiFi co-processor, a battery-backed RTC, eight status LEDs, an eight-position DIP switch, a reset push-button, and **every** module pin brought out to breakout headers for wiring access.

This guide walks through each feature, shows which module pins it uses, and notes the small number of co-existence rules you should keep in mind. Use it alongside the Hardware Design Guide and the Software Developer Guide.

3.1 What You Get Out of the Box

Feature	What It Provides
Power input	DC barrel jack + on-board 3.3 V buck regulator
Reset push-button (SW1)	Drives module NRST input through a debounce capacitor
UART0 (USART0)	RS-232 (DB-9 J7) and USB-Serial (USB Type-C J3) – jumper-selectable
UART1 (USART1)	RS-232 (DB-9 J8) and RS-485 (6-pin header P3) – jumper-selectable
microSD socket (J6)	SD card storage in SPI mode via the SAM E70's SPI0 controller
WiFi expansion header (P7)	7 signals + 3.3 V + GND for an SPI WiFi co-processor (or any SPI device)
Real-Time Clock (U2)	RV-3032 with battery backup (BT1), connected to TWI0 (native I2C)
8 status LEDs (LED1-LED8)	General-purpose visual indicators driven by P2 GPIOs
8-position DIP switch (SW2)	User input or configuration switches
Breakout headers (J1, J2)	Every module pin is brought out for prototyping wiring
12 configuration jumpers (JP1-JP12)	Power source, UART routing, RS-485 termination, ADC reference, etc.
Mounting holes	Four M3 holes for enclosure mounting

3.1.1 How the Boards Mate

The MODM7AE70 plugs into the dev board's **P1** (left, 50-pin) and **P2** (right, 50-pin) sockets. Every signal on each module connector is duplicated onto a parallel breakout header:

Module side	Dev-board socket	Dev-board breakout
P1 (50-pin)	P1	J1
P2 (50-pin)	P2	J2

J1 and J2 are simple straight passthroughs of P1 and P2 – whatever the module presents on a P1 pin is available at the corresponding J1 pin (and likewise for P2/J2). You can probe, wire, or extend any module signal regardless of whether the dev board also taps it for an on-board peripheral.

Note

The MODM7AE70 commits most of P1 to the External Bus Interface (EBI) bus that connects the on-module SDRAM and exposes a buffered 16-bit external memory bus. P1 has only 11 freely-usable GPIO pins; the rest are EBI signals that only become active when your application calls `EnableExtBusBuff(true)`.

3.2 Powering the Board

The DEV-MOD-105 has a single 3.3 V buck regulator (U7 = AP64352) that supplies both the module and on-board peripherals.

Input options (selected by jumper JP1):

JP1 position	Source
1-2	VCC_EXT – external 3.3 V via test pad
2-3 (default)	VIN – DC barrel jack J4 (5-24 V)
3-...	VBUS_USB – USB Type-C J3 (5 V)

A green power LED (LED9, 330 ohm) lights when the 3.3 V rail is alive.

Module power pins on the connectors:

- P1-3, P1-48, P2-2, P2-50 = +3.3 V (all four are driven from the dev-board rail)
- P1-1, P1-2, P1-49, P1-50, P2-1, P2-49 = GND (and additional grounds on P2-14, P2-46)

The MODM7AE70 has no separate battery-backed supply pin (unlike some other NetBurner modules); the SAM E70's small backup-RAM domain is powered from VCC3V and loses state on full power-down.

3.3 Reset Button

The push-button **SW1** (TS-1187) drives the module's reset input (P1-28, NRST):

- Normally held high through a 10 kohm pull-up to +3.3 V (on the module)
- Pulled to ground while SW1 is pressed; a 1 nF debounce cap (C10) suppresses contact bounce
- The same line is available at **J1 pin 28** if you want to add an external supervisor IC or watchdog through the breakout header

Press SW1 any time to perform a hardware reset of the module.

3.4 Serial Communication

The DEV-MOD-105 wires both module UART0 and UART1 to multiple physical interfaces, with jumpers selecting which interface drives the receive direction. Transmit goes to all attached transceivers, so you can monitor on one while talking on another.

The MODM7AE70 uses its native USART0 and USART1 hardware blocks for these two channels, so full hardware RTS/CTS flow control is available out of the box.

3.4.1 UART0: RS-232 and USB-Serial (USART0)

UART0 corresponds to the SAM E70's **USART0** peripheral (NetBurner serial port 0). The dev board wires it to:

- **DB-9 RS-232 connector J7** through MAX3232 transceiver U3
- **USB Type-C connector J3** through FTDI FT231X USB-UART bridge U6

USART0 signal	Module pin	SAM E70 pin / alt	Dev-board destination
TXD0	MOD_P2_04	PB1 alt C	MAX3232 T1IN + FTDI RXD (both at once)
RXD0	MOD_P2_03	PB0 alt C	Selected by JP2 (RS-232 or USB-Serial)
RTS0	MOD_P2_38	PB3 alt C	MAX3232 T2IN + FTDI CTS# (both at once)
CTS0	MOD_P2_29	PB2 alt C	Selected by JP3 (RS-232 or USB-Serial)

Selecting the receive source:

Jumper	1-2 (RS-232)	2-3 (USB-Serial)
JP2	RX from MAX3232 U3	RX from FTDI U6
JP3	CTS from MAX3232 U3	CTS from FTDI U6

Move both jumpers to the same side for a fully-routed connection. Because transmit drives both transceivers simultaneously, you can use the unselected interface as a passive monitor (read-only) of outgoing traffic.

Tx/Rx LEDs: LED10 and LED11 on the FTDI side blink with USB-Serial activity.

3.4.2 UART1: RS-232 and RS-485 (USART1)

UART1 corresponds to the SAM E70's **USART1** peripheral (NetBurner serial port 1). The dev board wires it to:

- **DB-9 RS-232 connector J8** through MAX3232 transceiver U4
- **RS-485 6-pin header P3** through THVD1424 transceiver U8 (full or half-duplex, with optional bus termination)

USART1 signal	Module pin	SAM E70 pin / alt	Dev-board destination
TXD1	MOD_P2_22	PB4 alt D	MAX3232 T1IN + THVD1424 D (driver input)
RXD1	MOD_P2_21	PA21 alt A	Selected by JP4 (MAX3232 R1OUT or THVD1424 R)
RTS1	MOD_P2_32	PA24 alt A	MAX3232 T2IN + RS-485 DE/RE direction control
CTS1	MOD_P2_33	PA25 alt A	Selected by JP5 (RS-232 CTS path)

All four USART1 hardware flow-control pins are routed to their native alt-function locations, so you can use peripheral-driven RTS/CTS without software intervention.

Selecting the receive source:

Jumper	1-2 (RS-232)	2-3 (RS-485)
JP4	RX from MAX3232 U4	RX from THVD1424

RS-485 configuration jumpers:

Jumper	Function	Install when
JP7	485_HF (half/full-duplex select on THVD1424)	Half-duplex 2-wire RS-485
JP8	Ties 485_RE to 485_DE (single-line direction control)	Always, for normal DE/RE driving
JP9	485_SLR (slew-rate limit)	Long lines / low baud (reduces emissions)
JP10	485_TERM_RX (120 ohm receive termination)	This node is at the bus end (receiver)
JP11	485_TERM_TX (120 ohm transmit termination)	This node is at the bus end (transmitter)

The RS-485 bus connector P3 exposes both A/B (data pair) and Z/Y (second pair, for full-duplex) terminals plus two GND pins.

3.5 SD Card Storage

The microSD socket (J6 = TF-01A) is wired to the SAM E70's **SPI0** controller, shared with the WiFi expansion header through separate chip-select lines.

Function	Module pin	SAM E70 pin / alt	Notes
SCK	MOD_P2_25	PD22 alt B (SPI0_SPCK)	Shared SPI bus
MISO	MOD_P2_27	PD20 alt B (SPI0_MISO)	Shared SPI bus
MOSI	MOD_P2_28	PD21 alt B (SPI0_MOSI)	Shared SPI bus
CS	MOD_P2_35	PA10 (GPIO)	Dedicated to the SD card
CD	MOD_P2_47	PA12 (GPIO)	Low when a card is inserted (pull-up)

Each MOSI/MISO/CS line has a 10 kohm pull-up to 3.3 V for SD-SPI idle behavior (R1, R2, R3, R6) and the CD line has a 10 kohm pull-up (R4).

The dev board uses the SD card in 1-bit SPI mode rather than 4-bit native SDIO; this is fine for the SD-card protocol but you do not get the higher throughput that SDIO 4-bit mode would offer. If you need full SDIO performance, you would need to repurpose the SAM E70's HSMCI controller (whose native pins map to other P2 locations like P2-16, P2-17, P2-18, P2-33, P2-36, P2-40) on a custom carrier.

Sharing the SPI bus with the WiFi header: see the next section. Because the two devices have separate chip-select lines (MOD_P2_35 for SD, MOD_P2_30 for WiFi), the NetBurner SDK arbitrates between them automatically – you simply use the appropriate device handle and the driver asserts the right CS.

3.6 WiFi and SPI Expansion Header (P7)

The 9-pin **NetBurner Expansion Header P7** is sized to accept a WiFi co-processor module (or any SPI device that fits the pinout). It exposes a clean SPI bus plus enable, reset, and IRQ lines, alongside 3.3 V (through filter inductor L2 = 10 uH for noise isolation) and GND.

P7 Pin	Signal	Module Pin	SAM E70 pin / alt	Notes
1	W_EN	MOD_P2_34	PA9 (GPIO)	WiFi enable / power-down control
2	W_CS	MOD_P2_30	PD12 alt C (SPI0_NPCS2)	Chip select for the WiFi co-processor
3	W_CLK	MOD_P2_25	PD22 alt B (SPI0_SPCK)	Shared SPI clock with SD card
4	W_MISO	MOD_P2_27	PD20 alt B (SPI0_MISO)	Shared SPI MISO
5	W_MOSI	MOD_P2_28	PD21 alt B (SPI0_MOSI)	Shared SPI MOSI
6	W_RST	MOD_P2_42	PA4 alt A (TWCK0)	WiFi reset – **also TWI0 SCL for the RTC**
7	W_IRQ	MOD_P2_26	PD27 (GPIO)	Interrupt input
8	VCC3VWiFi	–	–	+3.3 V through L2
9	GND	–	–	Ground

If no WiFi module is fitted, the entire header is free for any SPI peripheral that matches this pinout (or you can wire individual signals out through J2).

Coexistence with the SD card: the WiFi co-processor and SD card sit on the same SPI bus with separate chip-selects. Normal SPI arbitration in software lets both run, just not simultaneously full-duplex.

Note

MOD_P2_42 is also wired to the RTC's SCL line (TWI0). If your application both resets a WiFi module on P7 and reads the RTC, perform the WiFi reset before the first RTC access (or re-initialize the I2C bus after toggling W_RST). If you are not using the WiFi header, the RTC has the line entirely to itself. See Co-Existence Notes below.

3.7 Real-Time Clock

The **Microcrystal RV-3032-C7** is a low-power I2C real-time clock with a built-in 32.768 kHz crystal, a programmable CLKOUT pin, an interrupt output, and battery-backed time retention.

The dev board wires the RTC to the SAM E70's **TWI0** native I2C controller, so no bit-banging is required:

Function	Module pin	SAM E70 pin / alt	Notes
SCL	MOD_P2_42	PA4 alt A (TWCK0)	4.7 kohm pull-up to +3.3 V; **shared with W_RST**
SDA	MOD_P2_39	PA3 alt A (TWD0)	4.7 kohm pull-up to +3.3 V
INT	(P6 header)	–	Available at P6 pin 5
CLKOUT	(P6 header)	–	Available at P6 pin 4
VBACKUP	BAT1	–	Coin-cell input via 330 ohm R39

To talk to the RTC, mux P2-42 to TWCK0 and P2-39 to TWD0, then use the SDK's `I2C` API.

The dev board has a coin-cell holder (BAT1) so the RTC keeps time when main power is off. Plug in a CR-series cell to enable time retention.

Header P6 breaks out SDA, INT, CLKOUT, and GND for connecting external I2C devices on the same bus, or for using the RTC's interrupt/clock output in your application.

3.8 Status LEDs

Eight red LEDs (LED1-LED8) are mounted along one side of the board. Each is wired with its anode through a 330 ohm resistor to the module pin, cathode to GND – **drive the pin high to light the LED**.

The MODM7AE70 routes the dev-board LED nets to plain GPIO pins, so every LED is software-controllable as a status indicator (no USB-pin restrictions apply here).

LED	Module Pin	SAM E70 pin (per dev-board schematic)	Notes
LED1	MOD_P2_15	PD24	Plain GPIO
LED2	MOD_P2_16	PA28	Plain GPIO
LED3	MOD_P2_31	PA23	Plain GPIO
LED4	MOD_P2_23	PD28	Plain GPIO
LED5	MOD_P2_37	PD11	Plain GPIO
LED6	MOD_P2_19	PA1	Plain GPIO (alt C is EBI A18 – not used here)
LED7	MOD_P2_20	PA29	Plain GPIO
LED8	MOD_P2_24	PD31	Plain GPIO

The SDK's `bsp_devboard.h` exposes the LEDs as `LED1` through `LED8` macros for convenient use:

```
#include <bsp_devboard.h>
LED1 = 1;  // Turn LED1 on
LED1 = 0;  // Turn LED1 off
LEDs[3] = 1;  // Or use the index form
```

Note

the SDK `bsp_devboard.h` shipped with the toolchain references LED3 at P2[18] and LED5 at P2[17], whereas the production DEV-MOD-105 schematic routes those LEDs to P2[31] and P2[37]. If the macro names do not match your physical LEDs, drive the pin directly: `P2[31] = 1` for the LED labeled "LED3" on the silkscreen, `P2[37] = 1` for "LED5".

3.9 User DIP Switches

A single 8-position DIP switch package (**SW2** = SMQS-08B-TP) provides eight user inputs. Each switch connects its module pin to GND when closed; each line has a 4.7 kohm pull-up to +3.3 V (R18-R26), so the pin reads **high when open, low when closed**.

Switch	Module Pin	SAM E70 pin	Notable alt functions
SW2-1	MOD_P2_13	PA8	AFE0 ADTRG (alt B)
SW2-2	MOD_P2_12	PA5 or PB5	UART1 RX (PA5 alt C); TWI1 SCL (PB5 alt A)
SW2-3	MOD_P2_11	PB13	USART0 SCK (alt C), DAC0 (alt B)
SW2-4	MOD_P2_09	PA2	DAC trigger (alt C)
SW2-5	MOD_P2_10	PD18	UART4 RX (alt C)
SW2-6	MOD_P2_07	PD30	UART3 TX (alt A)
SW2-7	MOD_P2_06	PC12	CAN1 RX (alt C)
SW2-8	MOD_P2_08	PA17	QSPI IO2 (alt A)

Warning

module pin P2-9 (PA2) carries the DAC trigger alt function. With SW2-4 closed, the line is pulled to GND through the switch – which conflicts with using the DAC trigger. Leave SW2-4 open when you need DAC trigger functionality.

Tip: because every switch line is on a regular SAM E70 GPIO with no analog reservation, you can also use these P2 pins for digital input from external signals on J2 (with SW2 closed = pin held low; SW2 open = pin available for external driving).

3.10 Analog Outputs and Inputs

3.10.1 DAC

The SAM E70's DAC0 output is on PB13 (P2 pin 11), which the dev board ties to **DIP switch SW2-3**. To use the DAC, keep SW2-3 open (which is the default switch position) – closing it would short the DAC output to ground through the switch's 4.7 kohm pull-up resistor.

For the DAC trigger input (P2 pin 9 = PA2 alt C = DATRG), keep **SW2-4 open** so the trigger line is not held low.

3.10.2 ADC (AFE)

The MODM7AE70 has two 12-bit AFE (Analog Front End) blocks with two 6-to-1 input multiplexers. Many AFE inputs are accessible via the headers:

AFE channel	Module pin	DEV-MOD-105 tap
AFE0 AD0	P2-7	DIP switch SW2-6 (open switch = free)
AFE0 AD1	P2-21	UART1 RX (open JP4 = free)
AFE0 AD3	P2-23	LED4 (drive low to free; the LED current is small enough that you can still get a useful reading with care)
AFE0 AD5	P1-47	Free
AFE0 AD6	P2-8	Free
AFE0 AD8	P1-44	Free

AFE0 AD10	P2-3	UART0 RX (open JP2 = free)
AFE1 AD0	P1-13	Free (when EBI buffers disabled)
AFE1 AD1	P1-6	Free (when EBI buffers disabled)
AFE1 AD3	P1-5	Free (when EBI buffers disabled)
AFE1 AD6	P2-6	DIP switch SW2-7 (open switch = free)

The cleanest analog input is from a P1 pin that you have not committed to EBI – for example P1-13, P1-44, or P1-47 when external bus buffers are disabled.

3.10.3 VREFP Jumper

JP12 is a 2-pin header that, when shorted, ties **MOD_P2_05 directly to +3.3 V**. On the MODM7AE70, P2-5 is the **AD_VREFP** input – the external positive ADC reference voltage for the AFE blocks.

JP12 state	What it does
Open	AD_VREFP is HiZ; the SAM E70 uses its internal reference
Installed	AD_VREFP = +3.3 V; the AFE uses 3.3 V as the positive reference

For better noise immunity, leave JP12 open and supply a clean external reference voltage to J2 pin 5 if you need high-precision ADC readings.

3.11 Breakout Headers J1 and J2

Two 50-pin (2x25) headers expose every module pin for prototyping. **J1 mirrors P1; J2 mirrors P2.** Whatever signal the module presents at a P1 or P2 pin is available at the corresponding J1/J2 pin – including pins that the dev board also uses for an on-board peripheral.

This gives you:

- **All 50 pins of P1** available – though many P1 pins carry EBI bus signals that are HiZ until `EnableExtBusBuff(true)` is called. The 11 freely-usable P1 GPIO pins (PC8, PA22, PC14, PD19, PC11, PD15, PC13, PA6, PC19, PC30, PA19) are accessible at J1 with no contention.
- **All 50 pins of P2** available, including the dozen pins the dev board does not tap (listed in Free Module Pins below).
- Free probing access to any signal the dev board uses, for debugging or for tapping into the on-board peripheral nets externally.

Probing on a tapped pin: if you wire to a J2 pin that is also used by an on-board peripheral (e.g. an LED, a switch, the RTC), remember that your wire shares the net with whatever the dev board has on that line – you may need to remove the dev-board's load (lift the corresponding 0 ohm resistor or LED) for clean external use.

Jumper	Default∗	Function

3.12 Jumper Reference

Jumper	Default∗	Function
JP1	2-3 (VIN)	3.3 V regulator input source: VCC_EXT / VIN (J4 barrel jack) / VBUS_USB (USB-C)
JP2	1-2 (RS-232)	UART0 RX source: RS-232 (1-2) or USB-Serial (2-3)
JP3	1-2 (RS-232)	UART0 CTS source: RS-232 (1-2) or USB-Serial (2-3)
JP4	1-2 (RS-232)	UART1 RX source: RS-232 (1-2) or RS-485 (2-3)
JP5	installed	UART1 CTS from RS-232 transceiver
JP6	depopulated	(Reserved; not fitted on rev 1.2)
JP7	open	RS-485 half-duplex enable on THVD1424 (install for 2-wire half-duplex operation)
JP8	installed	Ties 485_RE to 485_DE for single-signal direction control
JP9	open	RS-485 slew-rate limit (install to enable, recommended for long bus or low baud)
JP10	as needed	RS-485 RX termination (install at the bus-end node only)
JP11	as needed	RS-485 TX termination (install at the bus-end node only)
JP12	open	VREFP: when installed, ties MOD_P2_05 (AD_VREFP) to +3.3 V

Defaults reflect typical out-of-box configuration; verify against the printing on the silkscreen of your particular board.

3.13 Co-Existence Notes and Cautions

The DEV-MOD-105 reuses module pins to deliver many features in a compact form. Almost all combinations work straight away. The handful of cases where you should think before mixing features:

#	Situation	What to Do
1	Using both the RTC and a WiFi module on P7 simultaneously	They share MOD_P2_42 (RTC SCL = TWI0_SCL and W_RST). Reset the WiFi module before initialising the RTC, or re-init the I2C bus after any W_RST pulse. If you are not populating P7, ignore this.
2	SD card + WiFi on P7 active in the same program	They share the SPI0 bus (SCK/MISO/MOSI). The SDK arbitrates via separate CS lines (NPCS3 for SD, NPCS2 for WiFi) – just don't expect concurrent transfers.
3	Need to drive the DAC output	Keep DIP switch SW2-3 open (default position). DAC0 is on PB13 = P2-11 which also routes to SW2-3.
4	Need to use the DAC external trigger (DATRG)	Keep DIP switch SW2-4 open. DATRG is on PA2 = P2-9 which routes to SW2-4.
5	Sending UART0 traffic but only wanting it on one of RS-232 / USB-Serial	TX drives both transceivers in parallel. Either ignore the unused side or pull the cable on the side you do not want. RX is selected by JP2 / JP3.
6	Sending UART1 traffic but only wanting it on one of RS-232 / RS-485	TX drives both. Keep the RS-485 DE line low when running RS-232, or remove JP8 to silence the RS-485 driver.
7	Using MOD_P2_05 as a normal GPIO or ADC reading	Leave JP12 open. JP12 closed ties P2-5 (AD_VREFP) directly to +3.3 V (external analog reference).

#	Situation	What to Do
8	Using P1 pins as GPIO	Only the 11 un-buffered P1 pins are GPIO-usable. The rest are EBI bus signals that are HiZ when `EnableExtBusBuff(false)` – you cannot drive them out to the carrier when EBI is disabled, and you cannot use them for non-EBI purposes when it is enabled.
9	Using the SD card in 4-bit SDIO mode for higher throughput	Not available on this dev board. The wiring is 1-bit SPI mode only. For 4-bit SDIO performance, build a custom carrier that exposes the SAM E70's HSMCI native pins.

None of these prevent the dev board from being a complete working system – they are just the small set of "pick one of the two" or "configure before you toggle" notes to keep in mind.

3.14 Quick Reference: Module Pin to Dev-Board Feature

The table below lists every P2 pin and what the dev board does with it. (All P1 pins pass straight to J1 with no on-board peripheral tap, except P1-28 which is also driven by reset switch SW1.)

P2 Pin	SAM E70 Pin	Dev-Board Feature	Notes
1	–	GND	–
2	–	3.3 V supply	–
3	PB0	U0_RX (via JP2)	USART0 RXD (alt C)
4	PB1	U0_TX -> MAX3232 + FTDI	USART0 TXD (alt C)
5	–	AD_VREFP via JP12	External AFE reference; free unless JP12 installed
6	PC12	Switch7 (DIP SW2-7)	4.7k pull-up; CAN1 RX alt available
7	PD30	Switch6 (DIP SW2-6)	4.7k pull-up
8	PA17	Switch8 (DIP SW2-8)	4.7k pull-up; QSPI IO2 alt available
9	PA2	Switch4 (DIP SW2-4)	DAC trigger pin; keep SW2-4 open for DAC use
10	PD18	Switch5 (DIP SW2-5)	4.7k pull-up
11	PB13	Switch3 (DIP SW2-3)	DAC0 output pin; keep SW2-3 open for DAC use
12	PA5 or PB5	Switch2 (DIP SW2-2)	Multiplexed – see `P2_12_USE_B5`
13	PA8	Switch1 (DIP SW2-1)	4.7k pull-up
14	–	GND	–
15	PD24	LED1	–
16	PA28	LED2	–
17	PA26	(free at J2)	–
18	PA27	(free at J2)	–
19	PA1	LED6	EBI A18 alt available
20	PA29	LED7	–
21	PA21	U1_RX (via JP4)	USART1 RX (alt A); AFE0 AD1 alt available
22	PB4	U1_TX -> MAX3232 + THVD1424	USART1 TX (alt D); TWI1 SDA alt available
23	PD28	LED4	TWI2 SCL / CAN1 RX alts available
24	PD31	LED8	QSPI IO3 alt available
25	PD22	SPI SCK (SD card + WiFi P7)	SPI0 SPCK (alt B)
26	PD27	W_IRQ (WiFi interrupt)	SPI0 NPCS3 alt available
27	PD20	SPI MISO (SD card + WiFi P7)	SPI0 MISO (alt B)

P2 Pin	SAM E70 Pin	Dev-Board Feature	Notes
28	PD21	SPI MOSI (SD card + WiFi P7)	SPI0 MOSI (alt B)
29	PB2	U0_CTS (via JP3)	USART0 CTS (alt C); CAN0 TX alt available
30	PD12	W_CS (WiFi chip-select)	SPI0 NPCS2 (alt C); GMAC RX3 alt available
31	PA23	LED3	USART1 SCK / EBI A19 alts available
32	PA24	U1_RTS	USART1 RTS (alt A)
33	PA25	U1_CTS (via JP5)	USART1 CTS (alt A)
34	PA9	W_EN (WiFi enable)	UART0 RX alt available
35	PA10	SD card chip-select	UART0 TX alt available
36	PA30	(free at J2)	HSMCI DA0 alt available
37	PD11	LED5	–
38	PB3	U0_RTS -> MAX3232 + FTDI	USART0 RTS (alt C); CAN0 RX alt available
39	PA3	RTC SDA	TWI0 SDA (alt A)
40	PA31	(free at J2)	SPI0 NPCS1 alt available
41	PD25	(free at J2)	UART2 RX alt available
42	PA4	W_RST + RTC SCL	TWI0 SCL (alt A) – same module pin used for two purposes
43	PA13	(free at J2)	QSPI IO0 alt available
44	PD26	(free at J2)	UART2 TX alt available
45	PA14	(free at J2)	QSPI SCK alt available
46	–	GND	–
47	PA12	SD card-detect (CD)	Low when card inserted; QSPI IO1 alt
48	PA11	(free at J2)	QSPI CS alt available
49	–	GND	–
50	–	3.3 V supply	–

3.15 Free Module Pins on the DEV-MOD-105

Pins that are not tapped by any on-board peripheral and are available cleanly at the breakout headers J1 / J2.

P1 GPIO pins (un-buffered, not part of the EBI bus, freely usable):

P1 Pin	SAM E70 pin	Useful peripherals available
P1-4	PC8	EBI NWE (alt A) – only useful with EBI enabled
P1-5	PA22	USART1 RK / PWM0 ext trigger / EBI NCS2 / AFE1 AD3
P1-6	PC14	EBI NCS0 / Timer 8 clock / CAN1 TX / AFE1 AD1
P1-7	PD19	EBI NCS3 / USART2 CTS / UART4 TX
P1-8	PC11	EBI NRD / Timer 8 line A
P1-9	–	TIP (EBI buffer signal – not a GPIO)
P1-10	PD15	Plain GPIO
P1-13	PC13	EBI NWAIT / AFE1 AD0
P1-31	PA6	PCK0 / UART1 TX (Serial 3 TX)
P1-33	PC19	EBI A1 / PWM0 H2
P1-44	PC30	EBI A12 / Timer 5 line B / AFE0 AD8
P1-47	PA19	PWM0 L0 / EBI A15 / I2SC1 MCK / AFE0 AD5

P2 pins that pass straight through to J2 (no on-board peripheral tap):

P2 Pin	SAM E70 pin	Note
P2-5	–	AD_VREFP – analog reference (free unless JP12 installed)
P2-17	PA26	USART1 DCD / Timer 2 / HSMCI DA2
P2-18	PA27	USART1 DTR / Timer 2 / HSMCI DA3
P2-36	PA30	PWM0 L2 / HSMCI DA0 / I2SC0 DO
P2-40	PA31	SPI0 NPCS1 / PCK2 / HSMCI DA1 / PWM1 H2
P2-41	PD25	PWM0 L1 / SPI0 NPCS1 / UART2 RX
P2-43	PA13	QSPI IO0 / PWM0 H2 / PWM1 L1
P2-44	PD26	PWM0 L2 / SSC TD / UART2 TX / UART1 TX (alt D)
P2-45	PA14	QSPI SCK / PWM0 H3 / PWM1 H1
P2-48	PA11	QSPI CS / PWM0 H0 / PWM1 L0

If you need any of the alternate functions on these pins, the the full alt-function tables are above.

3.16 Module-Internal Pin Allocation

A number of SAM E70 pins are committed to functions internal to the MODM7AE70 module and **are not exposed at P1 or P2**. They never appear on the DEV-MOD-105 and cannot conflict with anything you do on the carrier. For completeness:

- **SDRAM (IS42S16400J, 8 MB):** SAM E70 EBI bus – 16 data lines, 15 address lines, plus CAS/RAS/SDCK/SDCKE/SDWE/SDA1
- **External bus buffers (SN74LVTH16245A):** the buffered EBI extensions on P1 are gated by these chips; on-module logic drives the OE inputs
- **Ethernet PHY (Microchip KSZ8081):** full RMII bus (REF_CLK, TXEN, TXD0/1, CRS_DV, RXD0/1, RXER) plus MDC, MDIO, IRQ
- **12 MHz CPU crystal:** internal SAM E70 XOUT/XIN (PB8/PB9)
- **25 MHz Ethernet crystal:** Y2 on the KSZ8081
- **Programming bridge (ATSAMD20E aux MCU):** SWD lines (PB6, PB7), ERASE (PB12), RESET
- **USB analog (HSDP, HSDM, VDDUTMI, VDDUTMIC):** routed internally to the P3 header through the magnetics

See the MODM7AE70 Hardware Design Guide for the full module-internal allocation, plus the power-sequencing, EBI buffer-enable, and routing guidance that applies if you go on to build your own carrier board.

4 Quick Reference

4.1 Minimum Connections for Operation

The following signals **must** be connected for the module to boot and run:

REQUIRED

- Power (3.3V)
  - P1-3 Vcc 3.3V -—— 3.3VDC supply (250mA max)

  - P1-48 Vcc 3.3V -—— 3.3VDC supply
  - P2-2 Vcc 3.3V -—— 3.3VDC supply
  - P2-50 Vcc 3.3V -—— 3.3VDC supply
- Ground (connect ALL)
  - P1-1, P1-2, P1-49, P1-50
  - P2-1, P2-14, P2-46, P2-49
- Reset (P1-28 RESET) – typically tied to 3.3V through a 10k pull-up
- Ethernet (100-version: on-module RJ-45 is ready; 200-version: bring TX+/TX-/RX+/RX-/CT taps to external RJ-45)

Signal	Pins	Why Required
Vcc 3.3V	P1-3, P1-48, P2-2, P2-50	Main supply (250 mA max)
GND	P1-1, P1-2, P1-49, P1-50	Return path; connect all for low impedance
GND	P2-1, P2-14, P2-46, P2-49	Return path
RESET	P1-28	Active-low reset input (10 k pull-up to 3.3V)

Optional but recommended:

Signal	Pins	When Needed
AD_VREFP	P2-5	External ADC reference (else internal reference is used)
USB (P3)	P3-1..4	USB device port (VBUSEN, D-, ID, D+)
Debug UART	P2-34, P2-35	UART0 RX/TX for serial console (NetBurner serial port 2)
Ethernet TIP	P1-9 (TIP)	EBI buffer "Transfer In Progress" – only relevant when EBI is used

Note

Simplest possible carrier board: Connect 3.3V power, ground, route Ethernet, and tie P1-28 to 3.3V through a 10 k resistor. The module will boot, bring up the network stack, and be accessible.

4.2 Connector Pinout

Two 50-pin 0.1-inch 2.54 mm dual-in-line headers. Pin 1 is at the top-left of each connector. A large fraction of P1 is committed to the External Bus Interface (EBI), which carries the on-module SDRAM bus through bus-buffer chips out to the header for optional external memory-mapped expansion. EBI signals are marked `EBI_BUF.*` and are HiZ on the header until `EnableExtBusBuff(true)` is called.

Header P1 (Left):

ODD			EVEN	Note
GND	1	2	GND	
Vcc3.3	3	4	PC8 (NWE)	EBI Write Enable / GPIO / Timer 7
PA22	5	6	PC14 (NCS0)	USART1 RK / GPIO / Bus CS / CAN1 TX
PD19	7	8	PC11 (NRD)	Serial Port 6 / GPIO / Bus Read / Timer 8
TIP	9	10	PD15	EBI Buffer enable / GPIO
NWR1	11	12	(D0)	EBI byte select / EBI buffered data 0
(BA0)	13	14	(D2)	EBI bank address / EBI buffered data 2
(D4)	15	16	(D1)	EBI buffered data 4 / EBI buffered data 1

ODD			EVEN	Note
PC13	17	18	(D3)	GPIO / EBI buffered data 3
(D6)	19	20	(D5)	EBI buffered data 6 / 5
(D8)	21	22	(D7)	EBI buffered data 8 / 7
(D10)	23	24	(D9)	EBI buffered data 10 / 9
(D12)	25	26	(D11)	EBI buffered data 12 / 11
(D14)	27	28	(D13)	EBI buffered data 14 / 13
NRST	29	30	(D15)	Reset input / EBI buffered data 15
PA6	31	32	(A0/NSB0)	GPIO/UART1 TX / EBI address 0 (buffered)
(A2)	33	34	PC19 (A1)	EBI A2 (buffered) / EBI A1 + GPIO + PWM
(A4)	35	36	(A3)	EBI A4 / A3 (buffered)
(A6)	37	38	(A5)	EBI A6 / A5 (buffered)
(A8)	39	40	(A7)	EBI A8 / A7 (buffered)
(A10)	41	42	(A9)	EBI A10 / A9 (buffered)
PC30	43	44	(A11)	GPIO / EBI A11 (buffered)
PA19	45	46	(A13)	GPIO / EBI A13 (buffered)
GND	47	48	Vcc 3.3V	
GND	49	50	GND	

P1 GPIO is limited to 11 pins (PC8, PA22, PC14, PD19, PC11, PD15, PC13, PA6, PC19, PC30, PA19); the rest are EBI bus signals that are only useful as GPIO if you disable EBI buffering and accept that all external bus access goes away.

Header P2 (Right)

ODD			EVEN	Note
GND	1	2	Vcc 3.3V	
PB0	3	4	PB1	USART0 RX / USART0 TX
PA17	5	6	PC12	QSPI / GPIO / CAN1 RX / Timer 8
VREFP	7	8	PD30	ADC reference / UART3 TX
PA2	9	10	PD18	DAC trigger / UART4 RX
PB13	11	12	PA5 (or PB5)	GPIO/SCK0 / UART1 RX (multiplexed)
PA8	13	14	GND	AFE0 ADTRG / Ground
PD24	15	16	PA28	GPIO / GPIO
PA26	17	18	PA27	GPIO / GPIO
PA1	19	20	PA29	EBI A18 / GPIO
PA21	21	22	PB4	USART1 RX / USART1 TX (TWI1 SDA alt)
PD28	23	24	PD31	Serial 5 RX / Serial 5 TX
PD22	25	26	PD27	SPI0 SPCK / SPI0 NPCS3
PD20	27	28	PD21	SPI0 MISO / SPI0 MOSI
PB2	29	30	PD12	CAN0 TX / GMAC GRX3 (RMII alt)
PA23	31	32	PA24	USART1 SCK / USART1 RTS
PA25	33	34	PA9	USART1 CTS / UART0 RX (Serial 2)
PA10	35	36	PA30	UART0 TX (Serial 2) / GPIO
PD11	37	38	PB3	GMAC GRX2 / USART0 RTS
PA3	39	40	PA31	TWI0 SDA / SPI0 NPCS1
PD25	41	42	PA4	UART2 RX / TWI0 SCL / UART1 TX
PA13	43	44	PD26	QSPI MOSI / UART2 TX
PA14	45	46	GND	QSPI SCK / Ground
PA12	47	48	PA11	QSPI MISO / QSPI CS
GND	49	50	(NC)	

4.2.1 USB Connector (P3)

P3 Pin	Signal	Description
1	VCCUSB	USB VBUS enable (PC16, GPIO; voltage-divided for 5 V tolerance)
2	USB.D_N	USB data negative
3	USB.ID	USB ID (PA7)
4	USB.D_P	USB data positive

The USB connector is intended for USB Device operation; signals route directly to the SAM E70's high-speed USB controller via internal magnetics and ESD protection on the module.

4.2.2 Ethernet Connector (200-Version 10-Pin Header)

The 200 version omits the on-module RJ-45 and exposes the magnetics-side signals on a 10-pin header so the user can place the jack remotely or add PoE. The 100 version provides a built-in RJ-45 and these signals are not exposed.

Pin	Signal	Description
1	TX-	Transmit -
2	TX+	Transmit +
3	TXCT	Transmit center tap (driven by module)
4	RX+	Receive +
5	RX-	Receive -
6	RXCT	Receive center tap (driven by module)
7	GND	Ground
8	N/C	Not connected
9	LED	Link/activity LED sink
10	LED	Speed LED sink

4.3 Power

Parameter	Value
Supply voltage	3.3 V DC
Typical current	100 mA
Maximum current	250 mA
Supply pins	P1-3, P1-48, P2-2, P2-50
Ground pins	P1-1, P1-2, P1-49, P1-50; P2-1, P2-14, P2-46, P2-49
Internal regs	On-module 1.2 V (VDDCORE), USB analog rails
AD_VREFP	P2-5 (external optional reference)

The module has no battery-backed domain on the headers – the SAM E70's general-purpose backup registers are inside the MCU and depend on whatever supply you provide. There is no separate "Vstby" pin.

4.4 Peripheral Availability Summary

4.4.1 Serial Ports

NetBurner Port #	Hardware Module	Header Pins (default route)	Notes
0	USART0	P2-3 (RX), P2-4 (TX)	RTS = P2-38, CTS = P2-29; SCK = P2-11
1	USART1	P2-21 (RX), P2-22 (TX)	RTS = P2-32, CTS = P2-33; SCK = P2-31
2	UART0	P2-34 (RX), P2-35 (TX)	–
3	UART1	P2-12 (RX) or P1-31 (TX) / P2-42 (TX) / P2-44 (TX)	Multiple TX options
4	UART2	P2-41 (RX), P2-44 (TX)	–
5	UART3	P2-7 (TX), P2-23 (RX) / P2-24 (TX)	–
6	UART4	P2-10 (RX), P1-7 (TX)	–

USART0 and USART1 additionally support ISO7816, IrDA, RS-485, Manchester, and SPI modes. USART1 supports LON mode.

4.4.2 Communications

Bus	Where
SPI0	SCK=P2-25, MISO=P2-27, MOSI=P2-28, NPCS0=P2-29, NPCS1=P2-40/P2-41, NPCS2=P2-30, NPCS3=P2-26
QSPI	QSCK=P2-45, QCS=P2-48, QIO0=P2-43, QIO1=P2-47, QIO2=P2-8, QIO3=P2-24 (also usable as single-bit SPI)
TWI0	SDA=P2-39, SCL=P2-42
TWI1	SDA=P2-22, SCL=P2-12 (when P2_12_USE_B5 selected)
TWI2	SDA=P2-26, SCL=P2-23
CAN0	TX=P2-29, RX=P2-38
CAN1	TX=P1-6 / P2-30, RX=P2-6 / P2-23
USB	P3 connector (USB device on SAM E70 HS USB)

4.4.3 Analog

Function	Channels available at headers
AFE0 (12-bit ADC)	AD0 (P2-21), AD1 (P2-7), AD3 (P2-9), AD5 (P2-29 alt / P2-34 alt), AD6 (P2-10 alt), AD8 (P1-44 alt), AD10 (P2-3 alt)
AFE1 (12-bit ADC)	AD0 (P1-13), AD1 (P2-22), AD3 (P2-23), AD5 (P1-44 alt), AD6 (P2-10 alt)
DAC (12-bit)	DAC0 = P2-15 (alt), DAC1 = (internal mux)
Analog comparator	One ACC channel selectable across AFE inputs
Reference	AD_VREFP on P2-5 (external optional), AD_VREFN on-module ground

For the complete AFE channel-to-pin map see the Hardware Design Guide.

4.4.4 Storage

- On-module **2 MB** parallel flash (memory-mapped via EBI; transparent to the application)
- On-module **8 MB** SDRAM (IS42S16400J, accessed transparently via EBI)
- On-module **384 kB** embedded multi-port SRAM (TCM/system RAM, inside the SAM E70)
- External SD/MMC supported via the HSMCI controller on PA9/PA10/PA26/PA27/PA28/PA30 – on the dev board these pins land on P2-34, P2-35, P2-17, P2-18, P2-16, P2-36
- External QSPI flash supported via the QSPI controller (P2 pins 8, 24, 43, 45, 47, 48)

4.4.5 Ethernet

- One 10/100 Mbps port via on-module **KSZ8081 RMII PHY**
- IEEE 1588 PTP support (GTSUCOMP signal on P2-4 alt or P2-37 alt)
- IEEE 802.3az Energy-Efficient Ethernet supported
- IEEE 802.1AS time-stamping and 802.1Qav AVB credit-based traffic shaping in hardware
- 100-version: on-module RJ-45 (J2)
- 200-version: 10-pin breakout header for external jack

4.5 Serial Port to Hardware-Module Mapping

The NetBurner software API numbers serial ports from 0 to 6. The SAM E70 has two USART blocks and five UART blocks; mapping is per Table 5 of the datasheet:

NetBurner Port	Hardware Module	Software open API
0	USART0	`OpenSerial(0, baud, ...)`
1	USART1	`OpenSerial(1, baud, ...)`
2	UART0	`OpenSerial(2, baud, ...)`
3	UART1	`OpenSerial(3, baud, ...)`
4	UART2	`OpenSerial(4, baud, ...)`
5	UART3	`OpenSerial(5, baud, ...)`
6	UART4	`OpenSerial(6, baud, ...)`

USART0 and USART1 (NetBurner ports 0 and 1) can additionally be configured as SPI masters via the USART SPI mode. USART1 (port 1) supports LON. The standard NetBurner serial driver covers asynchronous mode; for the special modes consult the SAM E70 documentation.

4.6 External Bus Interface (EBI)

The SAM E70's EBI controller manages the on-module SDRAM and is also brought out to P1 through external 16-bit buffer chips (Texas Instruments SN74LVTH16245A) so a carrier board can add memory-mapped external peripherals or extra memory.

Signal group	P1 pins (data + address + control)
Data (buffered)	EBI_BUF.D0 through EBI_BUF.D15 – P1 pins 12, 14, 13, 15, 16, 18, 17, 19, 20, 21, 22, 23, 24, 25, 26, 27
Address (buffered)	EBI_BUF.A0/NSB0 through EBI_BUF.A14 – P1 pins 30, 32, 33, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46
Byte-select (buf)	EBI_BUF.NWR1/NSB1 – P1 pin 11
Bus control	NWE (P1-4, NWrite Enable), NRD (P1-8, NRead), NCS0..NCS3 (P1-6, P1-5, –, P1-7)
Buffer "in progress"	TIP – P1 pin 9 (must be wired correctly when EBI is used)
Reset	NRST – P1 pin 28

By default the buffer chips are disabled and the buffered P1 pins float at HiZ. Call `EnableExtBusBuff(true)` in your application to drive them, and `EnableExtBusBuff(false)` to release them when not in use (reduces EMI and lets `NRD` be repurposed as GPIO).

Important: when EBI buffers are enabled, every buffered P1 pin is committed to its EBI function. Plan your carrier-board signal usage accordingly.

4.7 Common Code Patterns

4.7.1 Blink an LED on P2 Pin 15

```
#include <pins.h>
#include <ucos.h>

void UserMain(void *pd)
{
    init();
    while (1)
    {
        P2[15] = 1;             // drive high
        OSTimeDly(TICKS_PER_SECOND / 2);
        P2[15] = 0;             // drive low
        OSTimeDly(TICKS_PER_SECOND / 2);
    }
}
```

If the application is targeting the DEV-MOD-105 dev board, `bsp_devboard.h` provides `LED1` through `LED8` macros so you can write `LED1 = 1` / `LED1 = 0` directly.

4.7.2 Read a GPIO Input on P2 Pin 13

```
#include <pins.h>

bool ReadInput()
{
    P2[13].hiz();               // configure as input
    return (bool)P2[13];        // read the line
}
```

4.7.3 Open a Serial Port (USART0 on P2-3 and P2-4)

```
#include <serial.h>
#include <pins.h>
#include <pinconstant.h>

void UserMain(void *pd)
{
    init();

    // Mux the pins onto USART0
    P2[3].function(PINP2_3_RXD0);
    P2[4].function(PINP2_4_TXD0);

    int fd = OpenSerial(0, 115200, 1, 8, eParityNone);
    writestring(fd, "Hello from MODM7AE70\r\n");

    while (1) { OSTimeDly(TICKS_PER_SECOND); }
}
```

The pin-function constants come from `pinconstant.h`; each P-pin define lists its available Peripheral A/B/C/D options.

4.7.4 Read an ADC Channel (AFE0 input on P2 Pin 13)

```
#include <hal.h>
#include <pins.h>

void UserMain(void *pd)
{
    init();

    // P2[13] = PA8 is AFE0 external trigger; use a dedicated AFE-input pin instead
    // For example, P2[34] = PA9 (URXD0) or P2[10] = PD18 carry AFE alt functions on
    // the MODM7AE70. Consult the data sheet for AFE channel-to-pin mapping.
    // The example below uses NetBurner's high-level ADC API:

    StartADCConversion(0);
    while (!ADCDoneFlag) {;}
    uint32_t raw = ReadADCResult(0);
    iprintf("AFE0 ch0 = %lu\r\n", raw);
}
```

For full AFE configuration including sample rate, gain, and differential mode see the SAM E70 reference manual.

4.8 Physical Specifications

Parameter	Value
Module dimensions	2.60" x 2.00" (66.0 mm x 50.8 mm)
Weight	1 oz (28 g)
Mounting holes	2 x 0.125" (3.18 mm) dia
Operating temperature	-40 C to +85 C
Headers	Two 25x2 SIP, 0.1" (2.54 mm) pitch, S2011E-25-ND
Ethernet (100-ver.)	On-module RJ-45
Ethernet (200-ver.)	10-pin 0.1" header for external RJ-45
USB connector	4-pin header (P3) -- mini-USB position depopulated
Part numbers	MODM7AE70-100IR (RJ-45 version), MODM7AE70-200IR (header version)

4.9 Pin Index (Sorted by Function)

4.9.1 P1 Connector

Power / Ground

Pin	Signal
3	Vcc 3.3 V
48	Vcc 3.3 V
1, 2, 47, 49, 50	GND

Reset

Pin	Signal	Note
28	NRST	Active-low reset, 10 k pull-up to 3.3 V

General-Purpose I/O Pins (limited – the rest of P1 is committed to EBI)

Pin	Port	Notes
4	PC8	EBI NWE (alt A) / Timer 7 (alt B)
5	PA22	USART1 RK (alt A) / PWM ext-trigger (alt B) / EBI NCS2 (alt C)
6	PC14	EBI NCS0 (alt A) / Timer 8 clock (alt B) / CAN1 TX (alt C)
7	PD19	EBI NCS3 (alt A) / USART2 CTS (alt B) / Serial 6 TX (alt C)
8	PC11	EBI NRD (alt A) / Timer 8 line A (alt B)
9	TIP	EBI buffer transfer-in-progress signal – output from the buffer logic
10	PD15	GPIO; available as a free pin
13	PC13	EBI NWAIT (alt A); pin is tied to SDRAM signal, see warning
31	PA6	PCK0 output (alt B) / UART1 TX (alt C) / Serial 3 TX (alt C)
33	PC19	EBI A1 (alt A) / PWM0 H2 (alt B)
44	PC30	EBI A12 (alt A) / Timer 5 line B (alt B)
47	PA19	PWM0 L0 (alt B) / EBI A15 (alt C) / I2SC1 MCK (alt D)

External Bus Interface (EBI) Buffered Signals

Pin	Signal	Pin	Signal
11	EBI_BUF.NWR1/NSB1	30	EBI_BUF.A0/NSB0
12	EBI_BUF.D0	32	EBI_BUF.A2
13	EBI_BUF.D1	33	EBI_BUF.A3
14	EBI_BUF.D2	35	EBI_BUF.A4
15	EBI_BUF.D3	36	EBI_BUF.A5
16	EBI_BUF.D4	37	EBI_BUF.A6
17	EBI_BUF.D5	38	EBI_BUF.A7
18	EBI_BUF.D6	39	EBI_BUF.A8
19	EBI_BUF.D7	40	EBI_BUF.A9
20	EBI_BUF.D8	41	EBI_BUF.A10
21	EBI_BUF.D9	42	EBI_BUF.A11
22	EBI_BUF.D10	43	EBI_BUF.A12
23	EBI_BUF.D11	45	EBI_BUF.A13
24	EBI_BUF.D12	46	EBI_BUF.A14
25	EBI_BUF.D13		
26	EBI_BUF.D14		
27	EBI_BUF.D15		

4.9.2 P2 Connector

Power / Ground

Pin	Signal
2	Vcc 3.3 V
50	Vcc 3.3 V
1, 14, 46, 49	GND

Analog Reference

Pin	Signal
5	AD_VREFP

GPIO Pins and Primary Alt Functions

Pin	Port	Primary alt (peripheral A / B / C / D) – selected highlights
3	PB0	PWM0_H0 / – / USART0 RX (Serial 0) / SSC TF
4	PB1	PWM0_H1 / GTSUCOMP / USART0 TX (Serial 0) / SSC TK
6	PC12	– / Timer 8 line B / CAN1 RX / –
7	PD30	UART3 TX (Serial 5) / – / – / ISI D10
8	PA17	QSPI IO2 / PCK1 / PWM0_H3 / –
9	PA2	PWM0_H1 / – / DAC trigger / –
10	PD18	– / – / UART4 RX (Serial 6) / –
11	PB13	PWM0_L2 / PCK0 / USART0 SCK / –
12	PA5 (or PB5)	PWM1_L3 (PA5) / ISI D4 / UART1 RX (Serial 3); TWI1 SCL (alt A on PB5)
13	PA8	PWM1_H3 / AFE0 ADTRG / – / –
15	PD24	PWM0_L0 / SSC RF / Timer 11 clock / ISI HSYNC
16	PA28	USART1 DSR / Timer 1 clock / HSMCI CDA / PWM1_FI2
17	PA26	USART1 DCD / Timer 2 line A / HSMCI DA2 / PWM1_FI1

Pin	Port	Primary alt (peripheral A / B / C / D) – selected highlights
18	PA27	USART1 DTR / Timer 2 line B / HSMCI DA3 / ISI D7
19	PA1	PWM0_L0 / Timer 0 line B / EBI A18 / I2SC0 SCK
20	PA29	USART1 RI / Timer 2 clock / – / –
21	PA21	USART1 RX (Serial 1) / PCK1 / PWM1_FI0 / –
22	PB4	TWI1 SDA / PWM0_H2 / – / USART1 TX (Serial 1)
23	PD28	UART3 RX (Serial 5) / CAN1 RX / TWI2 SCL / ISI D9
24	PD31	QSPI IO3 / UART3 TX (Serial 5 alt) / PCK2 / ISI D11
25	PD22	PWM0_H2 / SPI0 SPCK / Timer 11 line B / ISI D0
26	PD27	PWM0_L3 / SPI0 NPCS3 / TWI2 SDA / ISI D8
27	PD20	PWM0_H0 / SPI0 MISO / GTSUCOMP / –
28	PD21	PWM0_H1 / SPI0 MOSI / Timer 11 line A / ISI D1
29	PB2	CAN0 TX / – / USART0 CTS / SPI0 NPCS0
30	PD12	GMAC GRX3 / CAN1 TX / SPI0 NPCS2 / ISI D6
31	PA23	USART1 SCK / PWM0_H0 / EBI A19 / PWM1_L2
32	PA24	USART1 RTS / PWM0_H1 / EBI A20 / ISI PCK
33	PA25	USART1 CTS / PWM0_H2 / EBI A23 / HSMCI MCCK
34	PA9	UART0 RX (Serial 2) / ISI D3 / PWM0_FI0 / –
35	PA10	UART0 TX (Serial 2) / PWM0 EXTRG0 / SSC RD / –
36	PA30	PWM0_L2 / PWM1 EXTRG0 / HSMCI DA0 / I2SC0 DO
37	PD11	GMAC GRX2 / PWM0_H0 / GTSUCOMP / ISI D5
38	PB3	CAN0 RX / PCK2 / USART0 RTS (Serial 0) / ISI D2
39	PA3	TWI0 SDA / LONCOL1 / PCK2 / –
40	PA31	SPI0 NPCS1 / PCK2 / HSMCI DA1 / PWM1_H2
41	PD25	PWM0_L1 / SPI0 NPCS1 / UART2 RX (Serial 4) / ISI VSYNC
42	PA4	TWI0 SCL / Timer 0 clock / UART1 TX (Serial 3) / –
43	PA13	QSPI IO0 / PWM0_H2 / PWM1_L1 / –
44	PD26	PWM0_L2 / SSC TD / UART2 TX (Serial 4) / UART1 TX (Serial 3 alt)
45	PA14	QSPI SCK / PWM0_H3 / PWM1_H1 / –
47	PA12	QSPI IO1 / PWM0_H1 / PWM1_H0 / –
48	PA11	QSPI CS / PWM0_H0 / PWM1_L0 / –

For the full set of alt functions on every pin, see the Hardware Design Guide Appendix A.

5 Software Installation

Overview

This section will show you how to get started with the NNDK tools, version 3.x. For full documentation, please see our **online documentation**, or reference the `\nburn\docs\NetBurner` directory of your tools installation.

Platform Compatibility: NNDK 3.x runs on the ColdFire MCF5441X platforms and all ARM platforms.

The NetBurner Network Development Kit includes all the software and documentation you need to develop an application for NetBurner hardware. For existing customers using the 2.x tools, you can install the 3.x tools in a

different directory name and use both tool sets simultaneously. For simplicity, we do recommend having only one active 3.x tool set active at one time.

Throughout this document the installation directory name is referenced as the default: `\nburn`. For example, documentation is located in `\nburn\docs`. If you selected a different directory during installation, replace `\nburn` with that name.

Download the NNDK

All software should be downloaded from the NetBurner website at: **Download Development Tools**.

Note

Older versions of NBEclipse (3.3.2 and lower) require the 64-bit Java Runtime Environment (JRE) revision 1.8 or higher. NBEclipse tools version prior to 3.0 require the 32-bit JRE. Direct links to recommended versions are provided on the NNDK Download page. Download either the Java SE (Standard Edition) Development Kit (JDK) or Runtime Environment and follow the installation prompts. You may need to restart your computer after installation is complete.

Run Installer

Once the download is complete, run run the setup application and follow the onscreen directions. During the installation you will be prompted to enter the installation keycode that came with development kit. Enter the keycode for each NetBurner platform you wish to use. For example, if you have a MOD5441X and a MODM7AE70, you should enter both keycodes.

Note

The keycode for your installation can be found on the red card that is included with your NNDK kit.

Once the installation is complete you can start NBEclipse by executing the NBEclipse shortcut.

Utilities

The following software utilities are installed on your computer once the NNDK setup process is complete:

Tool	Function
NBEclipse	IDE used to develop, download and debug applications
Local Discovery	Locate device addresses on a LAN without Internet access
MTTTY	Windows serial terminal application
NBUpdate	Download a new application to your device from the command line
Application Wizard	Auto-generate application template
TaskScan	Displays list of RTPS tasks running on the device
UDPTerminal	PC utility to send and receive UDP packets
WinAddr2Line	Decipher trap messages to get source location of program faults
IPSetup*	Identify NetBurner modules on local Network
AutoUpdate*	Update your 2.x device to 3.0

Note

IPSetup is now only used for discovering your devices, and cannot be used for device configuration.

AutoUpdate is now only used to migrate compatible devices running applications built with NNDK 2.x to applications to NNDK 3.x. For more information, please see the section on updating your device to 3.x in the NetBurner Developers Guide.

Documentation

The following provide specific information on using the NetBurner tools, the various systems available, and the API. They can be accessed by opening `\nburn\docs\NetBurner\NetBurner-Documentation.html`. Additionally, the most up-to-date version can always be accessed at the online **Developers Guide**.

Programming Guides and References

We recommend reviewing the following portions of the Developers Guide in order:

NetBurner Guides	Purpose
This Quick Start Guide	
Platform References	Platform-specific information on the hardware you're using.
NBEclipse Getting Started Guide	How to start using NBEclipse. This is required reading before using the integrated development environment (IDE).
NetBurner Programmers Guide	Provides background information on the various systems available to developers as well as their general functionality.
NetBurner NBRTOS Library API Docs	Reference guides which list the function libraries available in the NNDK, as well as for the real-time operating system.
PC Tools and Utilities	Reference manual for Windows NetBurner GUI tools
Command Line Tools	Reference manual for NetBurner CLI tools
Production & Deployment Guide	Releasing finished code and devices to end users
All remaining sections of the Developers Guide	

Processor Manuals and Specifications

The documents listed below provide detailed information on the processors and compilers used with our modules and tools. They can be found in the `\nburn\docs` folder of your NNDK install.

General Documentation	Purpose
ARM Documentation	Reference manuals ARM processors
Microchip Documentation	Reference manuals for Microchip processors
NXP Manual	Reference manuals NXP processors
GNU Manuals	Manuals for GNU C/C++ libraries, compiler and linker, including the C/C++ language API functions
Platform Reference and Hardware Schematics	NetBurner hardware manuals that include memory map and design guides

6 Network Configuration

Overview

There are several ways to configure your device to connect to a network. The default setting for a NetBurner device is DHCP. If a DHCP server is available on the local network the device IP settings will be configured automatically. If DHCP is not available, you have the option of assigning static IP settings, using the AutoIP address, or using the IPv6 Link Local address. Each of these options are described in the following sections.

Dynamic IP Address (DHCP)

When the factory application boots it will attempt to obtain a DHCP address. If you are connected to a network with a DHCP server, the device IP address, mask, gateway and DNS sever should be configured automatically. If your PC is on the same DHCP network you will be able to communicate with the device.

Static IP Address

If the module is plugged in to a network that without a DHCP server, you have the option of configuring the IP settings statically. At a minimum you need an IP address and mask. If you need to route off the local area network you will also need a gateway and DNS. Static addresses should be assigned by your network administrator in order to avoid conflicts. The IP settings of the PC you use to connect to the device must be on the same subnet.

Auto IP Address (APIPA)

The default factory application is capable of AutoIP negotiation. AutoIP enables a device to automatically configure its IP address in the absence of a central DHCP server and without the need for a static IP address. By default AutoIP addressing starts in the 169.XXX.XXX.XXX address range. The network interface of the PC must also be set to the AutoIP range. Rather than assigning a value, if you set the address mode of the network interface to DHCP, the PC should default to it's own AutoIP addresss when it detects a DHCP server is not available.

Finding Your Device Address

Your NetBurner device will have at least three Addresses:

- IPv4
- IPv4 AutoIP
- IPv6 Local Link

Additional IPv6 addresses are possible if your network has an IPv6 router or DHCPv6 Server that assigns additional IPv6 addresses.

The following methods can be used to determine the IP address of your device:

- Discover Server: Easiest method, requires your PC and the NetBurner device have Internet access.
- Local Discover Utility: Run locally from your PC, useful when the device does not have Internet access, such as a direct Ethernet connection to a laptop.
- NBEclipse: The "find window" in the IDE automatically identifies devices.
- IPSetup: This is a NetBurner 2.x utility that can be run on Windows to identify the device's IP address, but since configuration is now done with the configuration server it cannot modify IP settings.

Please refer to the sections below for details on each method.

Discover Server

The recommended option is to locate your device using our online discovery service. You can do this by opening a browser and navigating to https://discover.netburner.com. This webpage should load a table that shows devices on your local area network which are running applications built with NNDK 3.x. From this table, you can access the web interface served from the device by clicking on the link in the "Web Page" column. To access a device's configuration record, click on the link in the "Config" column.

Local Discover Utility

If you don't have internet access from your location, you can run our utility, Local Discovery. This program is available from our tools in <NNDK Install>\pcbin\localdiscover.exe, and sends out a request to all NetBurner devices on the local network. It opens a browser page on the first device to respond that lists all of the discovered devices, or a page that shows that no devices were found.

Local Discover will attempt to check each network interface on the PC. For each interface it will first check the IPv4 subnet, followed by AutoIP and finally IPv6. The factory default setting for IPv4 addressing is DHCP.

Important: If you are using a direct direct Ethernet connection between your PC and NetBurner device, a DHCP server will not be available (unless you happen to be running one on your PC) and the device's primary IPv4 address will be 0.0.0.0. However, the AutoIP and IPv6 addresses will still be active. If your PC's Ethernet interface is set to DHCP, both the PC and device will use AutoIP addresses. If your PC's Ethernet interface is set to a static IP address (e.g. 192.168.1.10), Local Discover will use the device's IPv6 Local Link address.

You can see which address is being used by inspecting the URL field of the web browser. For example, if IPv6 was used the URL would look like: [fe80::203:f4ff:fe0b:xxxx]:20034/DISCOVER.html. If AutoIP: 169.254.xxx.xxx:20034/DISCOVER.html. If your PC has a static Ethernet IP setting and you see an AutoIP address in the URL of the web browser, there may be caching issues. Close the entire web browser application, reset your NetBurner device, wait 10 seconds, then execute localdiscover.exe again.

The NBEclipse IDE

Another option to find the IP address of your device is to use the NBEclipse IDE. As before, ensure that your device is powered on and plugged into the same network as your PC. Open NBEclipse, and in the bottom left of the workspace, NBFind can be found running. NBFind lists the IP, application, and MAC address of all NetBurner devices on your local network.

Python Find Utility

Device Find utilities written in Python are located in your installation directory: "\nburn\pctools\find\python".

IPSetup Utility

A final option is to use the included utility IPSetup. If the device is powered on and plugged into the same network as your PC, run IPSetup. This application shows all NetBurner devices on your network. Be aware that as of NNDK 3.x, IPSetup can not be used to configure your device. Our configuration system now handles this, as described later in this guide.

You can find the IPSetup tool on our website, here.

Note: If these options are failing, there may be a firewall issue blocking the applications from sending the UDP broadcast that is used to locate NetBurner devices. Always grant NetBurner applications the ability to get through your OS firewall and ensure that UDP port 20034 is open for use.

7 Factory Application

Overview

Your NetBurner device is pre-programmed with an application that demonstrates some of the basic features of the platform and peripherals. If you can discover the address of your device as described in the Network Configuration section, the next step is to verify network connectivity by accessing the device's web content. An application with a web server and content has two components:

1. The primary application web site.
2. The configuration web site on port 20034

Primary Application Web Site

The easiest way to view the main site is to use discover.netburner.com or the localdiscover utility and click on the Web Page Device link. Alternaively, you an type the device's IP address in the URL field of your web browser. If your device was at address 192.168.1.2: `http://192.168.1.2`.

Configuration Page

NetBurner 3.x devices are configured through the use of a configuration web page. As with the primary web site, you can use discover.netburner.com or localdiscover and click on the ConfigPage link, or you can enter the address manually into your web brower. Web browsers assume port 80 (or 443 for HTTPS), but for the config page you will need to specify the configuration port number 20034. (Port 20034 is the default port number for the NetBurner device config system because the ASCII initials "NB" are 0x4e42 in hex, which is 20034 in decimal.) So for example if your device is at address 192.168.1.2, then you'll browse to `http://192.168.1.2:20034`.

The configuration page allows you to view and modify all of the device's boot and network settings. Additionally, you can use this page to download a new application to the device. For more on the configuration system, please refer to the "Config Server Programming Guide" section of the "Programmers Guide".

Develop your own Application

Now that the NetBurner tools are installed, the hardware is set up and you have verified this by accessing the device's web server. You have now graduated beyond this quick start guide.

To begin writing an application, we recommend reading through the NBEclipse Getting Started guide and the NetBurner Programmers Guide. In the NBEclipse guide, you will find information on writing NetBurner applications, sending new applications to the module and debugging with the graphical debugger. The Programmers guide will teach you more about the NetBurner libraries, operating system, and writing of applications in a multitask environment. Both of these documents can be found in the "\nburn\\docs\NetBurner" folder of your NNDK software installation by opening the file "NetBurner-Documentation.html"

Examples

The NNDK includes a multitude of examples that demonstrate the various hardware and software applications of the kit. These examples can be found in your NNDK installation directory, under the examples folder. The default location is "\nburn\examples"

8 Disclaimers

8.1 Life Support Disclaimer

NetBurner's products both hardware and software (including tools) are not authorized for use as critical components in life supports or systems, without the express written approval of NetBurner, Inc. prior to use. As used herein: (1) Life support devices or systems are devices or systems that (a) are intended for surgical implant into the body or (b) support or sustain life, and whose failure to perform, when properly used in accordance with instructions for use provided in the labeling, can be reasonably expected to result in a significant injury to the user. (2) A critical component is any component of a life support or system whose failure to perform can be reasonably expected to cause the failure of the life support or system, or to affect its safety or effectiveness.

8.2 Anti-Piracy Policy

NetBurner, Inc. vigorously protects its copyrights, trademarks, patents and other intellectual property rights.

In the United States and many other countries, copyright law provides for severe civil and criminal penalties for the unauthorized reproduction or distribution of copyrighted material. Copyrighted material includes, but is not limited to: computer programs and accompanying sounds, images and text.

Under U.S. law, infringement may result in civil damages of up to $150,000, and/or criminal penalties of up to five years imprisonment, and/or a $250,000 fine. In addition, NetBurner, Inc. may seek to recover its attorneys' fees.

Index