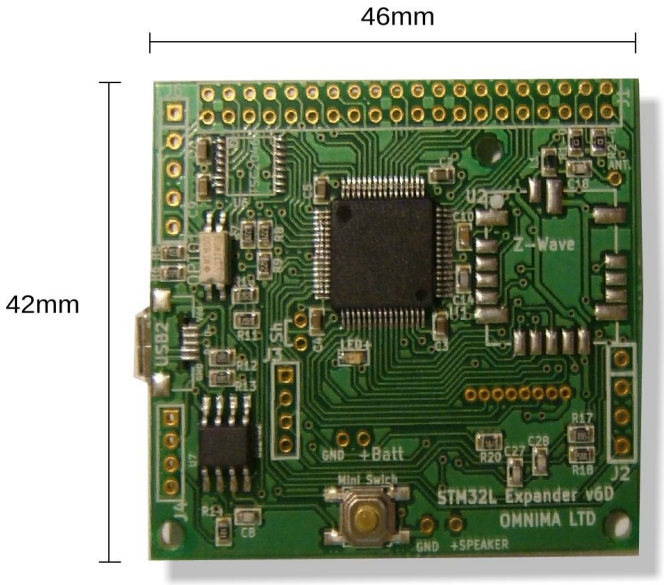
Omnima STM32Expander

SKU16189



Technical overview

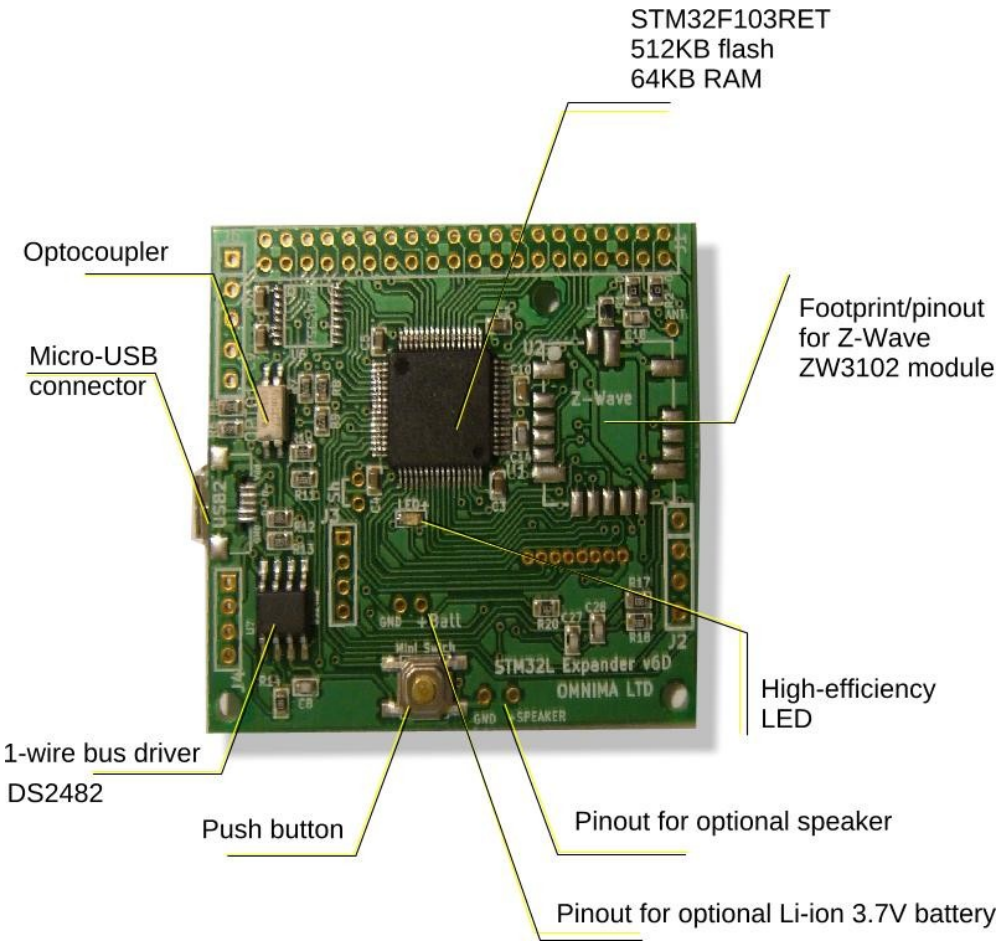
Omnima Limited, 176 Kennington Road, Oxford OX1 5PG
Company reg. 06038874, Tel. 08458692601, www.omnima.co.uk

Table of Contents

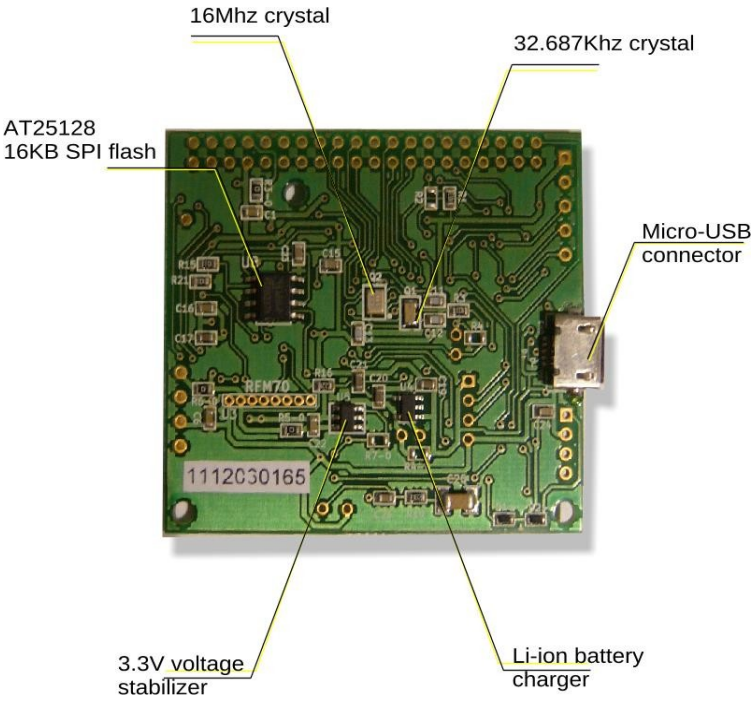
1. Overview

- STM32F105, STM32L132 or STM32F103 MCU 72Mhz ARM Cortex
- Up to 512Kb flash, 64Kb RAM
- USB or battery powered by a Li-ion 3.7V 100mAh battery (includes battery charger)
- USB2.0 device running as USB mass storage, USB serial or USB HID device
- Z-Wave ZM3102 module – controller or device mode (optional)
- Touch-screen controller TSC2046 (optional)
- RFM70 2.4GHz low-cost transreceiver module, up to 2mbps bandwidth
- 12bit multi-channel A/D
- 1 optoisolated input line
- 1-wire controller
- Pinouts for: CANbus, i2s, USB2.0, UART serial, SSD1928 LCD controller, i2c, SPI
  Real-time clock with battery backup

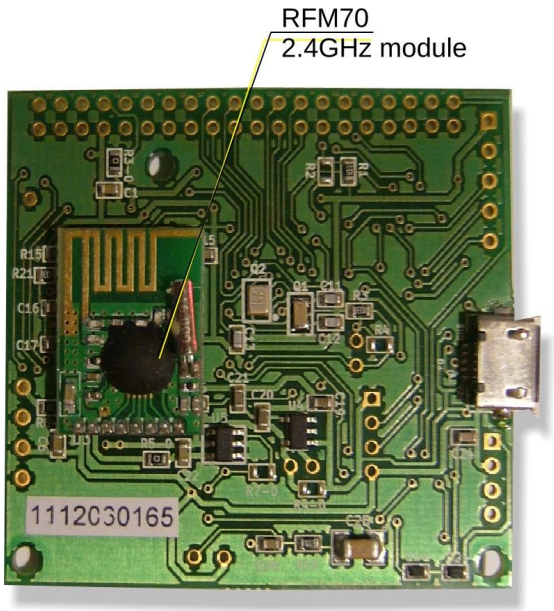
1.1 Components - top layer



1.2 Components – bottom layer



1.3 Components – bottom layer including RFM70



2. Features

- Add-on for the Omnima MiniEMBWiFi Linux WiFi board
- Connects to Omnima 3.5” LCD or other SSD1926/1963 based LCD displays
- Z-Wave controller or device bridge
- 2.4GHz 2mbps low-cost radio link
- 1-wire controller bridge
- Plenty of GPIOs for serial and parallel interfacing to other systems
- Data acquisition using 12bit A/D

3. Hardware specification

PCB	High quality SMT PCB, low profile, gold plated contacts, RoHS compliant
MCU	STM32F103RET 512Kb flash, 64KB RAM
USB	Built-in MCU USB2.0 device support
1-wire	DS2482 bus controller
Power supply	USB powered or 3.7V Li-ion battery source or 3.3V regulated supply
Battery charger	MAX1555 3.7V Li-ion battery charger
Voltage stabiliser	MAX8881 3.3V output
CPU XTAL	16MHz using MCU PLL multiplied to generate 72MHz CPU clock
RTC XTAL	32.687KHz crystal for real-time clock
Audio output	PWM/DAC generated output, pinouts for speaker provided
Flash IC	AT25128 16KB SPI flash IC available for storage of Z-Wave and user data
LED	High-effeciency Knightbright LED
2.4GHz radio	Hope Electronics RFM70 module
Z-Wave	Sigma Designs ZW3102 module
UARTs	3 serial ports 5V tolerant (require external level-shifters for connection to RS232)
Optio isolated input	TCMT1600 opto-coupler isolated input, 3750 Vrms
A/Ds	12bit A/D convertors
Push button	Micro switch connected to the STM32 PA8 GPIO

4. Pinouts

J1 – 8080 16bit bus, SSD1926/8 LCD bus, GPIO expander pinheader

Pin	Signal	Function
J1.1	STM32.PA12	USBDP # USB D+
J1.2	STM32.PA9	TX # UART RX
J1.3	STM32.PA11	USBDM # USB D-
J1.4	STM32.PA10	RX # UART TX
J1.5	n/c	
J1.6	+5V	
J1.7	+3.3V	
J1.8	GND	
J1.9	STM32.PA0	SSD1926.DB0
J1.10	STM32.PA1	SSD1926.DB1
J1.11	STM32.PA2	SSD1926.DB2
J1.12	STM32.PA3	SSD1926.DB3
J1.13	STM32.PA4	SSD1926.DB4
J1.14	STM32.PA5	SSD1926.DB5
J1.15	STM32.PA6	SSD1926.DB6
J1.16	STM32.PA7	SSD1926.DB7
J1.17	STM32.PC0	SSD1926.DB8
J1.18	STM32.PC1	SSD1926.DB9
J1.19	STM32.PC2	SSD1926.DB10
J1.20	STM32.PC3	SSD1926.DB11
J1.21	STM32.PC4	SSD1926.DB12
J1.22	STM32.PC5	SSD1926.DB13
J1.23	STM32.PC6	SSD1926.DB14
J1.24	STM32.PC7	SSD1926.DB15
J1.25	STM32.PC8	SSD1926.nRD
J1.26	STM32.PC9	SSD1926.nWR
J1.27	STM32.PB0	SSD1926.nCS
J1.28	STM32.PB1	SSD1926.nDC
J1.29	STM32.PB2	SSD1926.nRST
J1.30	STM32.PB10	SCL

J1.31	STM32.PB11	SDA
J1.32	STM32.PB12	
J1.33	STM32.PB13	
J1.34	STM32.PB14	
J1.35	XP	
J1.36	XM	
J1.37	YP	
J1.38	YM	
J1.39	1-wire bus	
J1.40	STM32.PB12	Opto-isolated input line (via opto-coupler)

J2 Pinout - Main UART serial port

J2.1	+3.3V	+3.3V supply voltage to the STM32Expander board (if 3.3V is not supplied from the USB VBus)
J2.2	STM32.PA9	TX # UARTRX
J2.3	GND	
J2.3	STM32.PA10	RX # UARTTX

J3 Pinout - STM32 USB port

J3.1	VBUS
J3.2	D
J3.3	D+
J3.4	GND

J4 Pinout - 1-wire controller bus

J4.1	n/c
J4.2	GND
J4.3	1-wire bus
J4.4	n/c

J5 Pinout - RFM70, STM32 I/Os, STM32 SWD port

J5.1	GND	
J5.2	+3.3V	
J5.3	STM32.PA13	RFM70.nCE # STM32.SWDIO
J5.4	STM32.PA14	RFM70.MOSI # STM32.SWCLK
J5.5	STM32.PA15	RFM70.MISO
J5.6	STM32.PC12	RFM70.CSN
J5.7	STM32.PB6	RFM70.SCK # STM32.USART1_TX
J5.8	STM32.PB7	RFM70.nIRQ # STM32.USART1.RX

J6 Pinout - A/D, PWM, opto input line and GPIOs

J6.1	GND	
J6.2	STM32.PB12	Opto-isolated input line (via opto-coupler)
J6.3	STM32.PC7	12bit A/D or PWM output
J6.4	STM32.PC8	12bit A/D or PWM output
J6.5	STM32.PC6	12bit A/D or PWM output

5. Development tools

Required tools

1. GCC ARM EABI compiler
2. stm32flash - flash upload utility using the STM32 system bootloader, the USART1 port pins PA10 and PA9 (on STM32F line)
3. st-link GDB tool using STM32-VL-Discovery board and the SWD debugging interface
4. Any IDE (integrated development environment tool) available or simply the command-line interface

5.1 Using and running the GCC ARM EABI compiler

GCC is used in many commercial projects worlwide and is available and runs on all popular desktop platforms including Linux, Mac OS X and Windows.

5.1.1 Download and uncompress the GCC archive

Linux

Download the following archive:

http://www.codesourcery.com/sgpp/lite/arm/portal/package7813/public/arm-none-eabi/arm-2010.09-51-arm-none-eabi-i686-pc-linux-gnu.tar.bz2

Alternatively download a more recent version of the compiler published by CodeSourcery - select the EABI Lite edition.

Create a directory in a convenient location on the local disk and expand the archive by running: tar xvf {archive.tar.bz2}

Mac

The GCC ARM wasn't available pre-packaged for Mac OS X for elsewhere but needed to be compiled for the platform. Several guides on how to do this are available on the Internet. Omnima have precompiled GCC for Mac OS X and the archive is now available for download from here:

http://omnima.co.uk/docs/arm-none-eabi-osx.tar.gz

You can double-click on the archive to expand it.

Windows

Download the CodeSourcery ARM GCC from CodeSourcery:

http://www.codesourcery.com/sgpp/lite/arm/portal/package7814/public/arm-none-eabi/arm-2010.09-51-arm-none-eabi-i686-mingw32.tar.bz2

You can also check if a more recent version is available from CodeSourcery - select the EABI Lite edition.

Expand the archive using 7zip (download from www.7zip.org) or use another archiving program if already available.

5.1.2 Add the compiler to your environment variable (optional)

On Linux and MAC OS X run: PATH=/opt/buildroot-gcc342/bin:PATH$

On Windows you can edit the environment variables in Control Panel/System/Advanced/Environment Variables and add the path to either you user profile or the global System variables.

Make sure that the bin path points to the directory in the GCC directory tree where arm-none-eabi-gcc binary is located.

This completes the compiler installation. You can now download GCC example projects and compile them by running: *make* or (*cs-make* on Windows)

5.2. Uploading the compiled binary to the STM32 MCU

5.2.1 Configuring the STM32 MCU to use the system bootloader

To upload the binary you can either use a JTAG or SWD debug connection or make use of the STM32 system bootloader.

To enable the STM32 MCU to start in the system bootloader mode make sure that the Boot0 (Pin 60 on STM32F103RB) and Boot1 (Pin 28 on STM32F103RB) pins are correctly configured:
Boot0 - pull high to 3.3V using a 10K resistor Boot1 - pull low to GND using a 10K resistor

5.2.2 STM32VL-Discovery board configuration

If you are using the STM32VL-Discovery board make sure that the SB2 jumper is present and SB16 jumper is removed. Using the Omnima USB-to-UART PL2303 cable connect the white wire to PA10, blue wire to PB9 and red wire to GND.

5.2.3 STM32L-Discovery board configuration

If using the STM32L-Discovery board make sure that SB19 is mounted on the PCB and SB3 is removed. Using the Omnima USB-to-UART PL2303 cable connect the white wire to PA10, blue wire to PA9 and red wire to GND.

5.2.4 Omnima STM32 Expander board configuration

To upload your new firmware over the USART a tool that implements the STM32 system bootloader protocol is required. There are several open source tools that are available.

The tool that we use at Omnima is stm32flash. It's written in plain C and can be customised if required. To obtain a copy of the stm32flash source (extended version to support STM32L MCU) download:

http://code.google.com/p/omnima-stm32l-expander/source/browse/#svn%2Ftrunk%2Ftools%2Fstm32flash

The Original stm32flash source written by geoff@spacevs.com is available here:

http://code.google.com/p/stm32flash/

Run **make** on your host PC to compile the program.

To upload your firmware run:
stm32flash -v -w {firmware.hex} {COMx}
To start running the code on the STM32 after a reset run:
stm32flash -g 0 {COMx}
To upload and start executing the code in one single call run:
stm32flash -v -w {firmware.hex} {COMx} -g 0
To generate hex firmware files configure your toolchain objcopy to output hex file format. {COMx} is /dev/ttyUSBx port on Linux and Mac OS X and one of COM1-COM15 on Windows.

5.3 Debugging your STM32 target using st-link and SWD

5.3.1 Use STM32VL-Discovery SWD port

ST-Link is a adapter found on the STM32VL-Discovery board and provides a low-cost solution to debugging your STM32 projects at a cost of £9 (in the UK). Make sure that the Discovery board is running ST-Link firmware version V1.J11.S0. You can check this by running the ST-Link Upgrade tool available from STM.

5.3.2 Download and buld st-link for Linux and Mac OS X

The software tool that allows ST-Link to work on Linux and Mac OS X is:
https://github.com/whitequark/stlink
Follow the README instructions on the bottom of the page to compile and run st-link and GDB. This will allow you to upload and debug ARM GCC binaries from Linux and Mac OS X.

5.3.3 SWD port connection to target STM32 MCU

- Remove both jumpers from CM3 on the STM32VL-Discovery board - Connect the SWD port on the Discovery to your target MCU.

- Start st-link and GDB as per the instructions from the README files
The advantages of using SWD over JTAG are quite obvious. It's a simple 3 wire connection and no expensive JTAG or cumbersome parallel port adapters are required.

6. Integrated development environment (IDE)

Recommended IDEs are the native platform IDE suites: - On Windows Visual C++ express 2010 (create or import and external make project to run cs- make) or Eclipse - On Mac OS X use X Code and make use of the external make - On Linux use KDevelop - use external make to build (press F8 to build and F11 to upload the firmware using stm32flash or DGB support to debug programs). Alternatively you can also use Eclipse.
Alternatively, all the tools can be used from the command-line and as such there is no direct requirement to use an IDE.

7. References

ST-Link and GDB on Linux
https://github.com/whitequark/stlink

STM32VL-Discovery boot mode
http://gostm32.blogspot.com/2010/09/conver...-boot-mode.html

STM32-VL-Discovery schematic CD00267113

STM32 system memory boot mode AN2606

STM32F system bootloader USART protocol AN3155

STM32L in-application programming using the USART AN3310

Omnima USB-to-UART cable
http://www.omnima.co.uk/store/catalog/USB-to-UART-adapter-p-16145.html

Omnima STM32 Expander PCB forum
http://www.omnima.co.uk/forums/index.php?showtopic=192

Omnima store STM32 Expander PCB store page
http://www.omnima.co.uk/store/product.php?productid=16189

Omnima MiniEMBWiFi Linux box
http://www.omnima.co.uk/store/product.php?productid=16180