STM32-SDR
A board perspective

Revision 0.5
09-24-2013
Dave Miller
VE7PKE
stm32sdr@gmail.com

1. Introduction

The STM32-SDR project is an exciting amateur radio project that was started by Charley Hill (W5BAA), John Fisher (K5FHF), Milt Cram (W8NUE) and Kees Talen (K5BCQ) of the AQRP group. Based in Austin, Texas they have been leaders in innovative and cost effective kits and projects pushing the state of the art on Amateur Radio QRP. They developed and created a very successful project called SDR2GO. In early 2013 Dave Miller (VE7PKE), in conversation with John on the AQRP Yahoo group, found out they had preliminary plans for a newer more powerful project based around the STM32F4 Discovery board.

After exchanging several emails with Charley, Dave Miller was welcomed into the group and this board project to support the software initiative was developed. This board is provided to support the software initiative and is supplied without software or any software support.

2. How the road lead me to here

Using digital techniques to simplify and expand the capabilities of radios has been a long and never ending road. Things like phase locked loops and switched cap filters were things I (Dave Miller, VE7PKE) read about with wonder and amazement in the 1980s when I got interested in Ham Radio. I would read and try to understand these things in the ARRL handbook and in QST magazines.

It prompted me to get interested in electronics when I was in college studying the mechanical aspects of technology. I was lucky to get involved in electronic pursuits after graduation and continued to read about developments on QST and QEX, DDS SDR, and more. Then life got in the way raising children, etc.

In 2012 I happened to attend a Field Day event at a local club (DARS) and that short visit got the fire going again. The spark had always been there. I have always been involved in ham radio in one form or another but mostly on VHF using a local repeater

SDR was not a new term to me but its glossy ads in QST were well beyond what I was prepared to spend. The Flex products were interesting but reading the reviews, it seemed the PC software was a challenge. I then heard about the Soft Rock series of

products from several friends who had built them but could never get them to work to their satisfaction, especially on transmit.

I had to learn more. So many choices out there. UHFSDR, Ensemble RXTX, Peaberry. After chatting with several close friends they told me they had Ensemble RXTX's. After closer examination I ordered a Peaberry kit, got it built and was amazed at the performance on RX. TX was a total disappointment. I could not get what I considered to be acceptable performance or really any performance. I was hooked but not satisfied.

I had simple requirements on TX. First I wanted to get CW TX to work. Then I could look at the output and make sure it was clean enough to put on the air. After several emails to various people it came to me that a PC was not the way to go when you want real time things like CW keying to work.

After much searching I found the AQRP group on Yahoo and things started to happen. Thanks John for getting me involved.

3. Disclaimer

- This is an experimental effort in Software Defined Radio and should be treated as such.
- This is not a prepackaged solution, ready to run out of the box, but a PCB that will allow you to create a standalone, without a PC, graphical interface to use and control many popular I/Q based SDR radio receivers and transceivers.
- There are no promises or guarantees on future software development and support by me, Dave Miller (VE7PKE).
- The board design has been done using my many years of experience and has not been subjected to any testing or certification by FCC CE, etc. Nor is it ROHS certified. Use at your own risk.
- It is offered as is for your experimental pleasure.
- You must base your decision to buy this board based on what you see here and examples I may post. I can only show you what I have done not what you can do or what the future holds.
- All commercial rights are reserved and patents and copyright may apply.
- For amateur and experimental use only.

4. Overview

If you are still reading after that required section on using at your own risk, I will try to describe what the SDR32-SDR is and what it is not at a high level, then drop down into the details.

5. What is SDR?

A Software Defined Radio is a hardware / software combination that allows reception and transmission of signals to and from the ether. Many of the traditional analog techniques have been replaced by software running in a Digital Signal Processor (DSP) or a really fast micro controller.

The best method is direct digitization at the antenna, processing totally in the digital domain then analog reconstruction at the speaker. This is possible and there are many examples in both amateur and commercial fields, your smartphone being a prime example. Today this is not mainstream.

Another popular and more cost effective way of doing this is to do the digital processing, not at the antenna, but at a frequency less than the original frequency. To process this information the signal is broken into an I and a Q signal from 90 degree shifted local oscillators and mixers, then processed and recombined. This technique is not new and is well explained in both the analog and digital domain in the ARRL handbook and many other places. Analog I/Q is is also called the SSB phasing method. For our purposes there will be an I/Q signal that contains all the required information. This I/Q signal is in the audio range.This is what a board like an Ensemble RXTX or a UHFSDR has available at the Line in and Line out jacks. Unless you want to listen to a single frequency you also need a way to tune to the different frequencies.

You also may want to switch in different filters or key an amplifier.

In most PC based SDR applications the RF portion is connected to a PC with a pair of audio cables carrying two stereo signals - I/Q in and I/Q out. These are connected to a Sound card of suitable quality and capability. In particular, stereo in and out is needed which can cause some problems in laptops and cheap USB sound dongles which only have a mono input. The I/Q in and out is shown in Figure 1. In our implementation, the STM32-SDR replaces the PC.

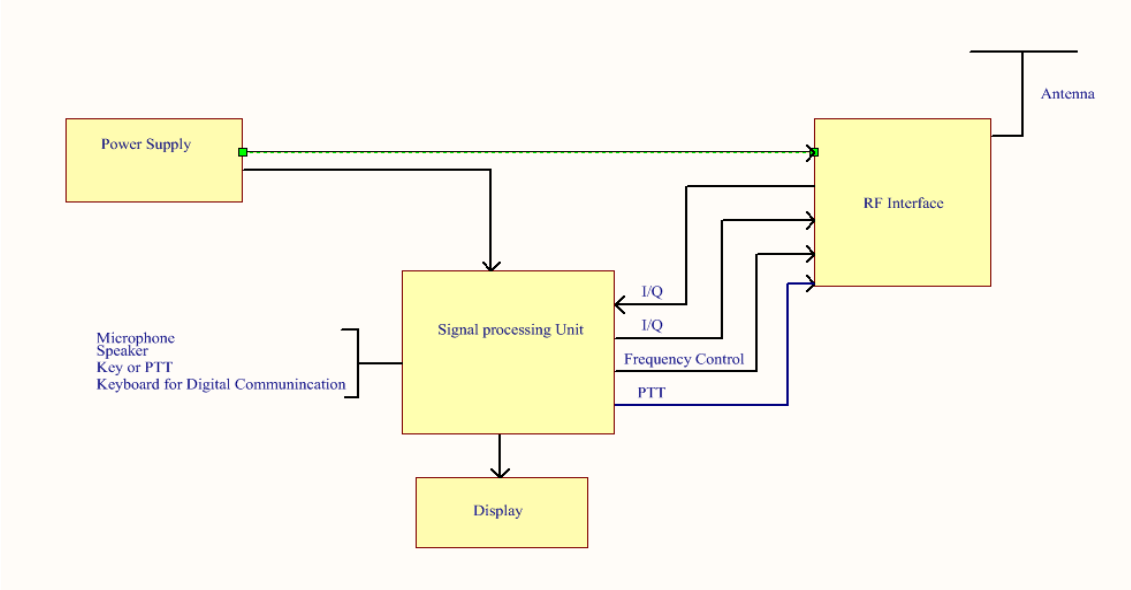

Fig 1 Typical block diagram of SDR with I/Q RF deck

The frequency source in popular radios such as the Ensemble RXTX, UHFSDR and many others is a Silabs Si570 device while others may use some Direct Digital Method (DDS). Frequency control is usually done with an I2C or SPI connection. In the case of the Ensemble RXTX it is done by USB. See Figure 1 Frequency control

PTT is done with a line going between 2 voltages. Either 5V or 12V or open or 0V or short to ground. See Figure 1 PTT

You also need a way to allow the operator to hear and speak, see what frequency the radio is tuned to, which mode, etc. On a PC based solution, these functions are done on a PC screen with a mouse and keyboard. For the STM32-SDR, these functions are done via the touch screen.

6. Why not use a PC?

- PC's use an operating system that is event driven and not real time. This means that things happen only when the OS lets it. Things that require prompt and immediate action such as CW keying happens not as fast as you want. The primary issue here is that the interface to the hardware has many levels of software and this slows things down. Also consider the path of a signal in a PC. For instance my favorite, CW key. This is path on a unit like a RXTX or Peaberry using USB. RF deck sees key closure, sends message to PC via USB, USB software driver then sends message to application that then decides what to do with it then sends a message back to the USB subsystem that then sends it back to the RF deck as an audio signal and a PTT signal which is then processed and RF comes on. Contrast this to an STM32-SDR which senses key closure, sets PTT, then starts sending audio . A clear winner in speed and simplicity. STM32-SDR uses direct control, not USB. More details on this later.
- PC's are big and bulky and hard to package.
- Cost

7. Advantages of STM32-SDR over PC

- Bare metal programming which is responsive and directly connected to hardware. No OS.
- Small 3.5" x 5.25 x 1.5" runs off 5V. You provide the RF deck and power.
- Inexpensive

- Software source is available for future development. Programmed in C using free open source tools.
- Has a colour touchscreen so the keyboard is only required for digital mode QSO's.
- You can build it yourself using boards with SMT parts pre-installed. Add your own case and RF deck.

8. How it all works

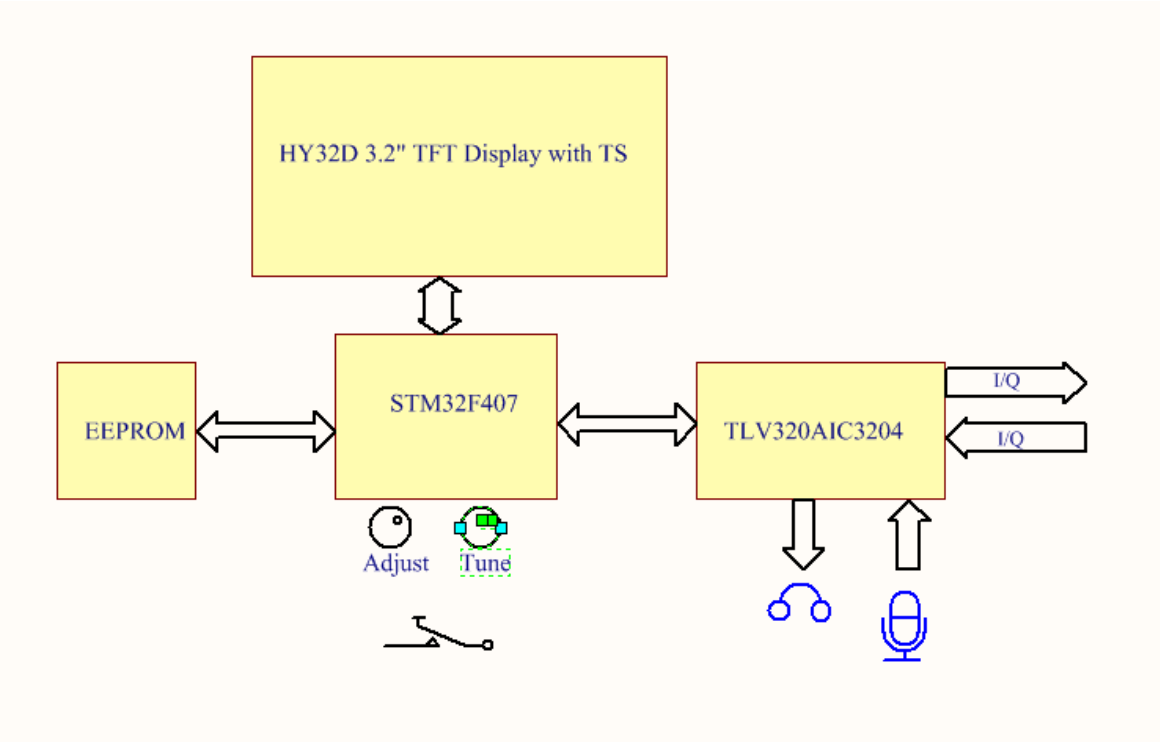

Figure 2 Block diagram STM32-SDR

The STM32-SDR is the Signal Processing unit in Figure 1. See assembly manual on STM32-SDR for more details. You make up suitable cables to interconnect the RF interface of your choice. See example application notes for more detail.
You provide 5V power to the STM32 Board.
You locate and load STM32-SDR firmware.
Read documentation on Software or users manual.
Hook up antenna and enjoy.

9. Hardware Details

Power
- 5V +- .25V
- 218 Ma measured on RX PSK No Si570 connected.

LCD display
- The supported display is a HY32D 3.2” Colour TFT with touchscreen
- Source http://www.hotmcu.com/32-touch-screen-tft-lcd-with-16-bit-parallel-interface-p-36.html?cPath=6_16&zenid=cu5s5r7mnm50nch7pkqatuslc0
- interface 34 pins (2x17) 0.100” header

Rotary encoders
- Two encoders, for user input, 24 pulses per revolution, with pushbutton
- Used in current software, right for frequency control, left for menu
- Selection function (Volume control parameter adjustment).

Microphone in
- 3.5mm Stereo jack, supports microphone bias

Headphone out
- 3.5mm Stereo jack, drives earbuds

I/Q out
- 1x3 0.100” header
- I, Ground, Q

I/Q in
- 1x3 0.1” header
- I, Ground, Q

PTT out
- FET with 5V Pull-up. Pull-up could be removed
- Normal operation 5V out TX
- Normal operation 0V out RX
- Software controlled
- Ground

Key Input
- PTT1 paddle 1 (Straight Key)
- PTT2 (for future Iambic Keyer)
- Ground

I2C connect to Si570 control
- 1x4 pin 0.100” connector
- Power (3.3V or 5V jumper selectable)
- Ground
- I2C Data
- I2C Clock

3.3V UART
- 1x6 pin 0.100” connector set up for FTDI cable pinout
- RX
- TX

- CTS
- RTS
- Power (3.3V or 5V jumper selectable)
- For future use and debugging

LMX9838 Bluetooth module

- Not fully tested
- Can be jumpered to 3.3V UART connector for BT keyboard
- Can be jumpered to SPI for audio interface

SPI connector

- For future use not fully tested
- 5 pin 0.100” connector
- SPI - CS, CLK, MISO, MOSI, Ground

BPF Bits

- For future use, not fully tested or supported in any software yet
- 4 pin 0.100” connector
- Bit1, Bit2, Bit 3, Ground

Extra Bits

- You always need more bits
- For future use, not fully tested or supported in any software yet
- 2 X 4 pin 0.100” connector
- Bit1, Bit2, Bit 3, Ground

USB

- USB OTG Micro AB connector
- Current support is Keyboard only