Line 6 ToneCore DSP Developer Kit (TCDDK)

Powered by Freescale

Hardware Guide

IMPORTANT NOTES:

- The contents of this document are the property of Line 6, and cannot be copied or duplicated without written permission of Line 6.
- The information enclosed in this document is confidential and cannot be used without first agreeing to the terms of the Agreement contained in the following four pages.
- The information is provided only to users of the ToneCore DSP Developer Kit.

Line 6, Inc.
ToneCore® DSP Developer Kit

This ToneCore® DSP Developer Kit Agreement (the "**Agreement**") is between You ("**You**") and Line 6, Inc., located at 26580 Agoura Road, Calabasas, CA 91302 ("**Line 6**") regarding Your use of the ToneCore® DSP Developer Kit and any associated documentation, software code or other materials made available by Line 6.

THIS AGREEMENT GOVERNS INSTALLATION AND USE OF THE SDK. YOU AGREE THAT THIS AGREEMENT IS LIKE ANY WRITTEN NEGOTIATED AGREEMENT SIGNED BY YOU. **By downloading, installing, accessing or otherwise using any aspect of the SDK, You accept all of the terms and conditions of this Agreement. If You do not agree to the terms and conditions of this Agreement, do not download, install, access or use the SDK**.

ACCEPTANCE:

1. You can accept this Agreement only by
    a. clicking to accept or agree to this Agreement, where this option is made available to You; or
    b. actually using the SDK. In this case, You agree that use of the SDK constitutes acceptance of the Licensing Agreement from that point onwards.

2. This Agreement is enforceable against You and any legal entity that obtained the SDK and on whose behalf they are used. If You are agreeing to be bound by this Agreement on behalf of Your employer or other entity, You represent and warrant that You have full legal authority to bind Your employer or such entity to this Agreement. If You do not have the requisite authority, You may not accept this Agreement or use the SDK on behalf of Your employer or other entity.

DEFINITIONS:

"**Developer Applications**" means all software applications created by You using the SDK.

"**Developer ToneCore Products**" means ToneCore Dev-Modules that You programmed with the Developer Applications.

"**Freescale**" means Freescale Semiconductor, Inc.

"**Intellectual Property Rights**" means any and all rights under patent law, trademark law, copyright law, trade secret law, and any and all other proprietary rights.

"SDK" collectively means the ToneCore DSP Developer Kit, including, without limitation, the ToneCore Dev-Platform, the SDK Software, and all associated documentation, applications, code, hardware, or other materials made available by Line 6 as part of the ToneCore DSP Developer Kit.

"**SDK Software**" means the example source code, including, without limitation, source code for (a) a microcontroller to process user input and communicate with the digital signal processor, and (b) a digital signal processor to process, shape and adjust the audio signal.

"**ToneCore**" means any of the range of commercially available Line 6 signal processing devices used for modifying the sound of an electronic guitar or other audio signal.

"**ToneCore Dock**" means the base portion of ToneCore devices that contains audio and DSP circuitry, and has a receptacle to accommodate a ToneCore Module.

"**ToneCore Module**" means the removable user interface portion of ToneCore devices that contains signal processing software, and can be inserted into a ToneCore Module.

"**ToneCore Dev-Dock**" means a ToneCore Dock featuring a USB connector.

"**ToneCore Dev-Module**" means a programmable ToneCore Module that stores digital signal processing code in the Flash memory of the on-board Freescale microcontroller and once programmed can be used in the ToneCore Dev-Dock and standard ToneCore Dock products.

"**ToneCore Dev-Platform**" collectively refers to the ToneCore Dev-Module and the ToneCore Dev-Dock.

LICENSE; INSTALLATION; AND USE:

3. Subject to the terms and conditions of this Agreement, Line 6 grants You the limited, worldwide, royalty-free, non-exclusive right and license to:

a. install, use, reuse, copy, modify, extract, reverse engineer, distribute, sell, lease and create derivative works of the SDK Software in connection with the ToneCore Dev-Platform solely to develop and create Developer Applications; and

b. use and reuse the ToneCore Dev-Platform solely in connection with Your creation and development of Your Developer Applications; and

c. program each ToneCore Dev-Platform You purchase from Line 6 with Your Developer Applications (i.e. the end result being Developer ToneCore Products) and market, license, sell, lease, and otherwise distribute the Developer ToneCore Products in Your sole discretion; and

d. use the "Line 6" and "ToneCore" registered trademarks solely to identify that the Developer ToneCore Products are compatible with Line 6's ToneCore products in substantially the following manner: "[the Developer ToneCore Products] are designed for use with all Line 6 ToneCore® pedals and docks." Further, on the same page as such statement of compatibility you shall provide the following credit in the same font and no less than two-thirds the font size as the surrounding text: "Line 6 and ToneCore are registered trademarks of Line 6, Inc."

LIMITATIONS AND RESERVATIONS:

4. Availability. Line 6 makes no representation or warranty with regard to the availability of the ToneCore Dev-Platform. Further, Line 6 may discontinue offering ToneCore products, ToneCore Dev-Docks and/or ToneCore Dev-Modules at anytime in its sole discretion without notice to You. Subject to availability, You may purchase an unlimited number of ToneCore Dev-Docs and ToneCore Dev-Modules during such time as they are offered for sale by Line 6; provided You are in compliance with the terms and condition of this Agreement.

5. Trademarks. All trade names, trademarks (including, without limitation, "Line 6" and "ToneCore"), service marks, logos, domain names, and other distinctive brand features used by Line 6 (collectively the "Line 6 Trademarks") are the exclusive property of Line 6 and all goodwill associated thereto inures

solely to the benefit of Line 6. Except as expressly provided in Paragraph 3.d. above, no license is granted to You to use any of the Line 6 Trademarks..

6. Proprietary Rights Notices. You agree that You will not remove, obscure, or alter any proprietary rights notices (including copyright and trademark notices) that may be affixed to or contained within the SDK.

7. Reservation of Rights. You acknowledge and agree that Line 6 owns all legal right, title and interest in ToneCore, the SDK Software, and the ToneCore Dev-Platform and all Intellectual Property Rights that subsist therein.

8. ToneCore Dev-Platform. You acknowledge and agree that the ToneCore Dev-Platform contains patent-pending technology owned exclusively by Line 6. You further warrant and represent that You will ***not*** recreate, duplicate, copy, reverse engineer, modify, copy decompile, or disassemble the ToneCore Dev-Platform or any other ToneCore product.

9. Developer Applications. Line 6 acknowledges and agrees that it obtains no right, title or interest from You under this Agreement in or to any Developer Application that You develop using the SDK, including any Intellectual Property rights which subsist therein.

10. Separate Agreements.
    a. You may have a separate written agreement signed by You and Line 6 that supplements or supersedes all or portions of this Agreement.

    b. Use of some third party materials included in the SDK may be subject to other terms and conditions typically found in a separate license agreement. Your installation and use of these third party materials is expressly conditioned upon Your accepting such separate license agreement.

WARRANT DISCLAIMER:
11. YOU EXPRESSLY UNDERSTAND AND AGREE THAT YOUR USE OF THE SDK IS AT YOUR SOLE RISK AND THAT THE SDK IS PROVIDED "AS IS" AND "AS AVAILABLE" WITHOUT WARRANTY OF ANY KIND FROM LINE 6.

12. YOUR USE OF THE SDK AND ANY MATERIAL DOWNLOADED OR OTHERWISE OBTAINED THROUGH THE USE OF THE SDK IS AT YOUR OWN DISCRETION AND RISK AND YOU ARE SOLELY RESPONSIBLE FOR ANY DAMAGE TO YOUR COMPUTER SYSTEM OR OTHER DEVICE OR LOSS OF DATA THAT RESULTS FROM SUCH USE.

13. LINE 6 FURTHER EXPRESSLY DISCLAIMS ALL WARRANTIES AND CONDITIONS OF ANY KIND, WHETHER EXPRESS, IMPLIED, OR STATUTORY, INCLUDING, WITHOUT LIMITATION, ANY WARRANTY WITH REGARD TO PERFORMANCE, MERCHANTABILITY, QUALIFTY, FITNESS FOR A PARTICULAR PURPOSE AND NON-INFRINGEMENT. The foregoing exclusions and limitations will apply to the maximum extent permitted by applicable law, even if any remedy fails its essential purpose.

LIMITATION OF LIABILITY:
14. YOU EXPRESSLY UNDERSTAND AND AGREE THAT IN NO EVENT WILL LINE 6 OR ITS SUPPLIERS BE LIABLE TO YOU FOR ANY DAMAGES, CLAIMS OR COSTS WHATSOEVER

ARISING FROM THE SDK OR YOUR USE OF THE SDK, INCLUDING WITHOUT LIMITATION ANY CONSEQUENTIAL, INDIRECT, INCIDENTAL DAMAGES, OR ANY LOST PROFITS OR LOST SAVINGS, EVEN IF AN LINE 6 REPRESENTATIVE HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH LOSS, DAMAGES, CLAIMS OR COSTS OR FOR ANY CLAIM BY ANY THIRD PARTY. THE FOREGOING LIMITATIONS AND EXCLUSIONS APPLY TO THE EXTENT PERMITTED BY APPLICABLE LAW IN YOUR JURISDICTION. IN ANY EVENT, LINE 6'S AGGREGATE LIABILITY AND THAT OF ITS SUPPLIERS UNDER OR IN CONNECTION WITH THE SDK WILL BE LIMITED TO TEN U.S. DOLLARS.

INDEMNIFICATION:
15. To the maximum extent permitted by law, You agree to defend, indemnify and hold harmless Line 6, its affiliates, subsidiaries and their respective directors, officers, employees and agents from and against any and all claims, actions, suits or proceedings, as well as any and all losses, liabilities, damages, costs and expenses (including reasonable attorneys fees) arising out of or accruing from (a) Your use of the SDK, (b) any Developer Application that infringes any copyright, trademark, trade secret, trade dress, patent or other intellectual property right of any person or defames any person or violates their rights of publicity or privacy, and (c) any non-compliance by You with this Agreement.

GENERAL LEGAL TERMS:
16. Entire Agreement. This Agreement constitutes the whole legal agreement between You and Line 6 and govern Your use of the SDK (excluding any services which Line 6 may provide to You under a separate written agreement), and completely replaces any prior agreements between You and Line 6 in relation to the SDK.

17. Waiver. You agree that if Line 6 does not exercise or enforce any legal right or remedy which is contained in this Agreement (or which Line 6 has the benefit of under any applicable law), this will not be taken to be a formal waiver of Line 6's rights and that those rights or remedies will still be available to Line 6.

18. Severability. If any court of law, having the jurisdiction to decide on this matter, rules that any provision of this Agreement is invalid, then that provision will be removed from this Agreement without affecting the rest of this Agreement. The remaining provisions of this Agreement will continue to be valid and enforceable.

19. Export Restrictions. THE SDK IS SUBJECT TO UNITED STATES EXPORT LAWS AND REGULATIONS. YOU MUST COMPLY WITH ALL DOMESTIC AND INTERNATIONAL EXPORT LAWS AND REGULATIONS THAT APPLY TO THE SDK. THESE LAWS INCLUDE RESTRICTIONS ON DESTINATIONS, END USERS AND END USE.

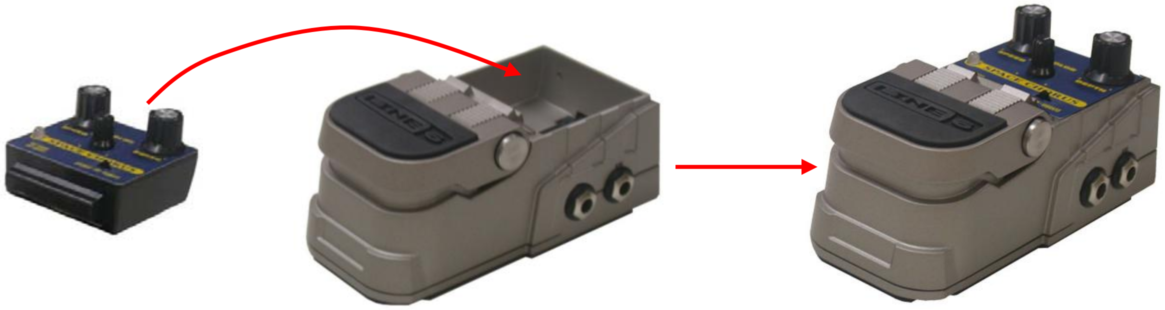
20. Transferability. The rights, licenses, responsibilities and obligations granted to and undertaken by You in this Agreement may not be assigned or transferred by You without the prior written approval of Line 6.

21. Governing Law and Jurisdiction. This Agreement, and Your relationship with Line 6 under this Agreement, shall be governed by the laws of the State of California without regard to its conflict of laws provisions. You and Line 6 agree to submit to the exclusive jurisdiction of the courts located within the county of Los Angeles, California to resolve any legal matter arising from this Agreement. Notwithstanding the foregoing, You agree that Line 6 shall still be allowed to apply for injunctive remedies (or an equivalent type of urgent legal relief) in any jurisdiction.

The ToneCore System

System Overview

All ToneCore Pedals are composed of two parts: the ToneCore Dock and the ToneCore Module.



ToneCore Module + ToneCore Dock (stereo version shown) = ToneCore Pedal

The ToneCore Modules and ToneCore Docks do not do anything individually, but when combined form a fully functional ToneCore Pedal.

The ToneCore Dock contains the DSP, the ADC/DAC (codec), external memory for the DSP, audio input/output jacks, dual-action footswitch, power supply and battery compartment.

The ToneCore Module contains the knobs, the switches, a bi-color LED, and a microcontroller. This MCU (Microcontroller Unit) has an internal flash memory that can be re-programmed.

Since the DSP has no flash memory, the code (both the DSP and the MCU code) is stored in the MCU's flash. The DSP has ROM memory but it can not be used, since it does not contain any preprogrammed code.

When connected together, the dock provides power to the module. The MCU sends the DSP code algorithm (stored in the MCU's flash) to the DSP so it can start executing and processing audio. This process is referred as Bootloading (the program is copied into the DSP's RAM memory). After this process, the DSP starts the execution of the code from RAM.

The DSP code implements audio effects, while the MCU code bootloads the DSP, reads the knobs, switches and controls the LED. When the system is running, the MCU is constantly reading the knobs and switches and sending these values to the DSP so it can change the parameters of audio algorithms. In addition, the DSP sends the status of the LED to the MCU (to turn on/off the LED accordingly).

Two different ToneCore Docks are available: a Mono and a Stereo version. These two are identical, other than the Mono version has one ¼" audio input and one ¼" audio output, and the Stereo version has two ¼" audio inputs and two ¼" audio outputs.

Mono ToneCore Dock

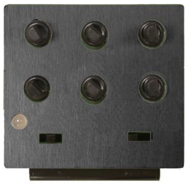
Stereo ToneCore Dock

Line 6 currently makes 11 different ToneCore Modules. Each provides a different set of effects when inserted into a ToneCore Dock. The firmware in each is also designed to identify if the ToneCore Module is inserted into a Mono or Stereo ToneCore Dock, and adapt its operation accordingly (if appropriate). *Note: Although similar in form and function, Line 6's 11 effect modules are not reprogrammable or alterable with the ToneCore DSP Developer Dock. However, they will function properly when used in the ToneCore DSP Developer Dock.*

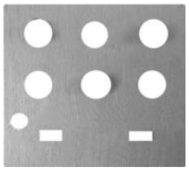
ToneCore DSP Developer Kit Hardware

The ToneCore DSP Developer Kit (TCDDK) consists of a modified Stereo ToneCore Dock called the ToneCore Programmer Dock. This Dock functions identically to the Stereo Dock, but adds a mini-USB connector for use with a PC for programming custom effects into a ToneCore Programmable Module.

The ToneCore Programmable Module can be programmed with custom written effects algorithms. The physical module can be customized as well. Each Programmable Module is shipped with an unlabeled and unattached brushed aluminum overlay. This overlay can be customized by hand, silk screen, or transfer lettering to label the Module controls as desired, and then adhered permanently to the Module. Alternatively, the template file included with the TCDDK download can be used to create custom overlays with other materials. For example, adhesive photo paper can be printed on and cut to size, and then used as a custom overlay.



ToneCore Programmable Module



Brushed aluminum overlay

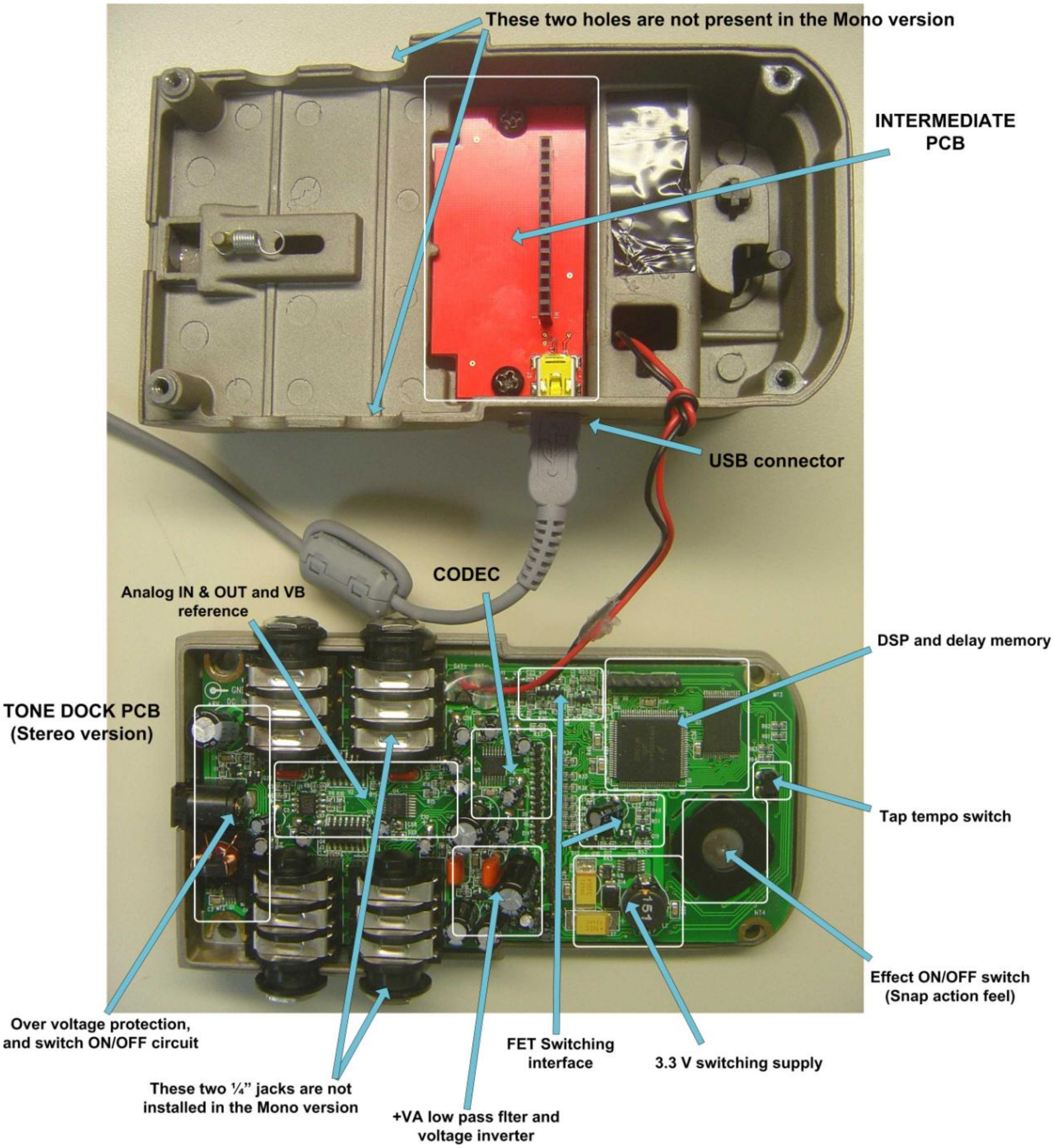
For adventurous types, Modules can be customized to have fewer potentiometers and/or switches. The plastic enclosure of the Module consists of two pieces that snap together. With the knobs removed, the two pieces can be pried apart to reveal the internal circuit board. Any unwanted pots or switches can be (carefully) unsoldered, and the remaining unused holes in the plastic enclosure can be covered up with a custom overlay.

Once programmed, the Programmable Modules can be used in any ToneCore Dock. However, they can only be programmed in the ToneCore Programmer Dock.

Internal Hardware

ToneCore Programmer Dock

There are two circuit boards within the ToneCore Programmer Dock: The Main board, and the Intermediate board. The Main board is identical in every respect to the Main board in the standard Stereo ToneCore Dock. The Intermediate board is different than a standard Stereo ToneCore Dock due to the USB interface. A description of the elements on the boards is shown below, along with the differences that would be found on a Mono ToneCore Dock.


These two holes are not present in the Mono version
INTERMEDIATE PCB
USB connector
CODEC
Analog IN & OUT and VB reference
DSP and delay memory
TONE DOCK PCB (Stereo version)
Tap tempo switch
Effect ON/OFF switch (Snap action feel)
Over voltage protection, and switch ON/OFF circuit
FET Switching interface
3.3 V switching supply
These two ¼" jacks are not installed in the Mono version
+VA low pass flter and voltage inverter

The ToneCore Dock includes the following components:

DSP

The DSP that performs all the audio processing is a DSP56364 from Freescale Semiconductor. It utilizes the single-instruction-per-clock-cycle DSP56300 core while retaining DSP56000 code compatibility. Some of the features are:

- **Digital Signal Processing Core**
  - 100 Million Instructions Per Second (MIPS) with a 100 MHz clock at a nominal 3.3 V
  - Object code compatible with the DSP56000 core with highly parallel instruction set
  - Data Arithmetic Logic Unit (Data ALU)
  - Program Control Unit (PCU)
  - Direct Memory Access (DMA)
  - Software programmable PLL-based frequency synthesizer for the core clock
  - Hardware debugging support: On-Chip Emulation (OnCE) Module, Joint Test Action Group (JTAG) Test Access Port (TAP), and Address Trace mode
- **On-Chip Memory**
  - Modified Harvard architecture allows simultaneous access to program and data memory
  - Program ROMs that may be factory programmed with data/program provided by the application developer
  - 8K x 24-Bit Program ROM (this can not be used since must be factory programmed)
  - 192 x 24-Bit bootstrap ROM
  - 0.5K/1.25K x 24 Bit Program RAM
  - 1.5K/.75K x 24 Bit Y-data RAM
  - 1K x 24 Bit X-Data RAM
  - Various memory switches available
- **Memory Expansion**
  - Memory expansion up to 2-256K x 8-bit word memory for P, X, and Y memory when using SRAM (there is an external SRAM memory in the dock)
  - Memory expansion up to 2-16M x 8-bit word memory for P, X, and Y memory when using DRAM (supported but not used)
  - Chip Select Logic for glueless interface to SRAMs
- **Peripheral and Support Circuits**
  - Enhanced Serial Audio Interface (ESAI) includes:
    - 6 serial data lines, 4 selectable as receive or transmit and 2 transmit only.
    - Master or slave capability
    - I2S, Sony, AC97, and other audio protocol implementations
    - Asynchronous and synchronous operation
  - Serial Host Interface (SHI) features:
    - SPI and I2C protocols
    - Ten-word receive FIFO
    - Support for 8-, 16-, and 24-bit words.

  - On-chip peripheral registers memory mapped in data memory space
- **Reduced Power Dissipation**
  - Very low power (3.3 V) CMOS design
  - Wait and Stop low-power standby modes
  - Fully-static logic, operation frequency down to 0 Hz (DC)
  - Optimized power management circuitry Package
- **Package**
  - 100-pin plastic LQFP package

External Memory

The external memory is a BS62LV4008, a high performance, very low power CMOS Static Random Access Memory organized as 524,288 words by 8 bits and operates from a range of 2.4V to 3.6V supply voltage.

Advanced CMOS technology and circuit techniques provide both high speed and low power features with a typical CMOS standby current of 0.45uA at 3.0V/25°C and maximum access time of 55ns at 3.0V/85°C.

Easy memory expansion is provided by an active LOW chip enable (CE), and active LOW output enable (OE) and three-state output drivers. The BS62LV4008 has an automatic power down feature, reducing the power consumption significantly when chip is deselected.

Codec

The codec (ADC/DAC) in the ToneCore pedals is the AK4552 from AKM. It is a low voltage 24bit 96kHz A/D & D/A converter for digital audio system. Analog signal inputs/outputs of the AK4552 are single-ended, therefore, external filters are not required. Some features are:

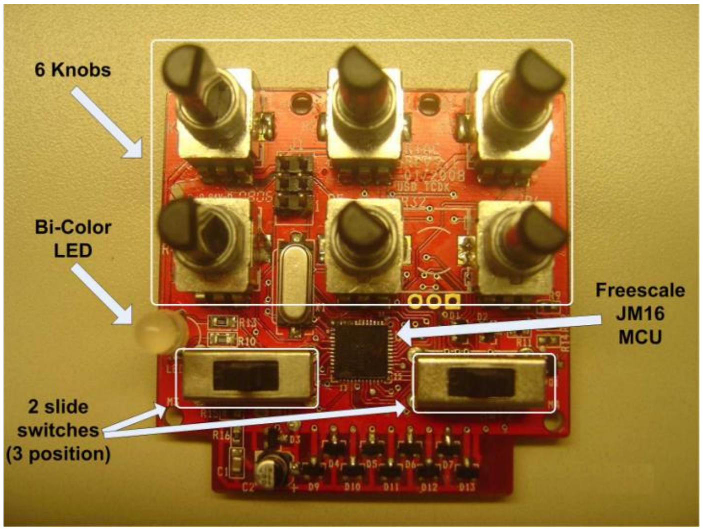
- HPF for DC-offset cancel (fc=3.4Hz@fs=44.1kHz)
- Single-ended ADC
  - S/(N+D):89dB@VA=3.0V
  - Dynamic Range, S/N: 97dB@VA=3.0V
- Single-ended DAC
  - Digital de-emphasis for 32kHz, 44.1 kHz and 48kHz sampling
  - S/(N+D): 88dB@VA=3.0V
  - Dynamic Range, S/N: 100dB@VA=3.0V
- Audio I/F format: MSB First, 2's Compliment
  - ADC: 24bit MSB justified
  - DAC: 24bit LSB justified
- Input/Output Voltage:
  - ADC = 1.85Vpp@VA=3.0V
  - DAC = 1.75Vpp@VA=3.0V

- Sampling Rate:
  - 8kHz to 50kHz (Normal Speed)
  - 50kHz to 100kHz (Double Speed, Double Speed Monitor)
  - 100kHz to 200kHz (Quad Speed Monitor)
- Master Clock:
  - 256fs, 384fs, 512fs or 768fs@Normal Speed
  - 256fs or 384fs@Double Speed
  - 128fs or 192fs@Double Speed Monitor
  - 64fs, 96fs, 128fs or 192fs@Quad Speed Monitor
- Power Supply: 2.4 to 4.0 V
- Supply current: 14 mA
- Package: 16pin TSSOP

For more detailed information, please refer to datasheet of each component.

ToneCore Programmable Module

The ToneCore Programmable Module contains a single PCB, as shown below:



Beside the knobs, switches and LED, each ToneCore Module contains a microcontroller, and the circuitry required to interface with any ToneCore Dock.

Microcontroller

The MCU (Microcontroller Unit) used in the module is the MC9S08JM16 from Freescale Semiconductor. It is an 8-bit MCU featuring on-chip USB 2.0 full-speed device support. Some of the features are:

8-bit HCS08 Central Processing Unit (CPU)

- Up to 24 MHz internal bus (48 MHz HCS08 core) frequency offering 2.7 to 5.5V across temperature range of -40°C to +85°C

- Support for up to 32 peripheral interrupt/request resources

On-Chip Memory

- Up to 16K Flash read/program/erase over full operating voltage and temperature
- Up to 1K RAM
- 256 Byte USB RAM

Power-Saving Modes

- Wait plus two stop modes
- Multi-purpose clock generator (MCG)

Peripherals:

- Full-speed USB 2.0 (12 Mbps) module
- Analog comparators (ACMP) with option to compare to internal reference
- Analog-to-digital converter (ADC): Eight-channel, 12-bit resolution
- Two serial communications interface (SCI) modules offering asynchronous communications
- I2C (inter-integrated circuit) module with up to 100 kbps with maximum bus loading; multi-master operation; programmable slave address; interrupt driven byte-by-byte data transfer; supports broadcast mode and 10-bit addressing
- SPI: Two serial peripheral interfaces with full-duplex or single-wire bidirectional; double-buffered transmit and receive; master or slave mode; MSB-first or LSB-first shifting
- Timer pulse width modulation (TPM): Up to six channels

Input/Output

- Up to seven Keyboard Interrupt (KBI) pins with selectable polarity
- 37 general purpose input/output (GPIO)s

System Protection

- Watchdog computer operating properly (COP) reset with option to run from dedicated 1kHz internal clock source or bus clock
- Low-voltage detection with reset or interrupt; selectable trip points
- Illegal op code detection with reset
- Flash block protection

Hardware Development Support

- Single-wire background debug interface
- Breakpoint capability
- On-chip in-circuit emulator (ICE) debug module.

General System Operation

All of the analog and digital audio processing circuitry is contained in the ToneCore Dock's Main board. In order to give this board's analog processing a lot of flexibility in being able to implement a variety of pedal types (i.e. distortion, delay, compressor, etc); the electronics include a number of digitally controlled analog switches which can configure the audio signal path into the most appropriate way for the desired effect.

All of the information for setting up this circuitry, as well as the DSP code, is stored in the ToneCore Programmable Module's microcontroller's FLASH memory. **There is no default set up or DSP program residing on the ToneCore Dock PCB alone.** By changing the ToneCore Module the mode of operation and function of the pedal can therefore be completely altered.

The microcontroller in the module reads its potentiometers, switches, battery voltage, ToneCore Dock effect on/off switch and Tap tempo switch and also controls the bi-color LED. The status of these potentiometers and switches is sent to the ToneCore Dock DSP where the DSP code adjusts the audio algorithm accordingly.

The DSP code is permanently stored in the ToneCore Programmable Module microcontroller FLASH. Upon powering up the system, the DSP code is downloaded from the ToneCore Module into the ToneCore Dock DSP internal program memory, and the relevant audio path analog switches are set up as determined by the software. A number of sensor switches (input/output jack sensing, etc.) are read on I/O pins of the DSP as well, and can be utilized by the software.

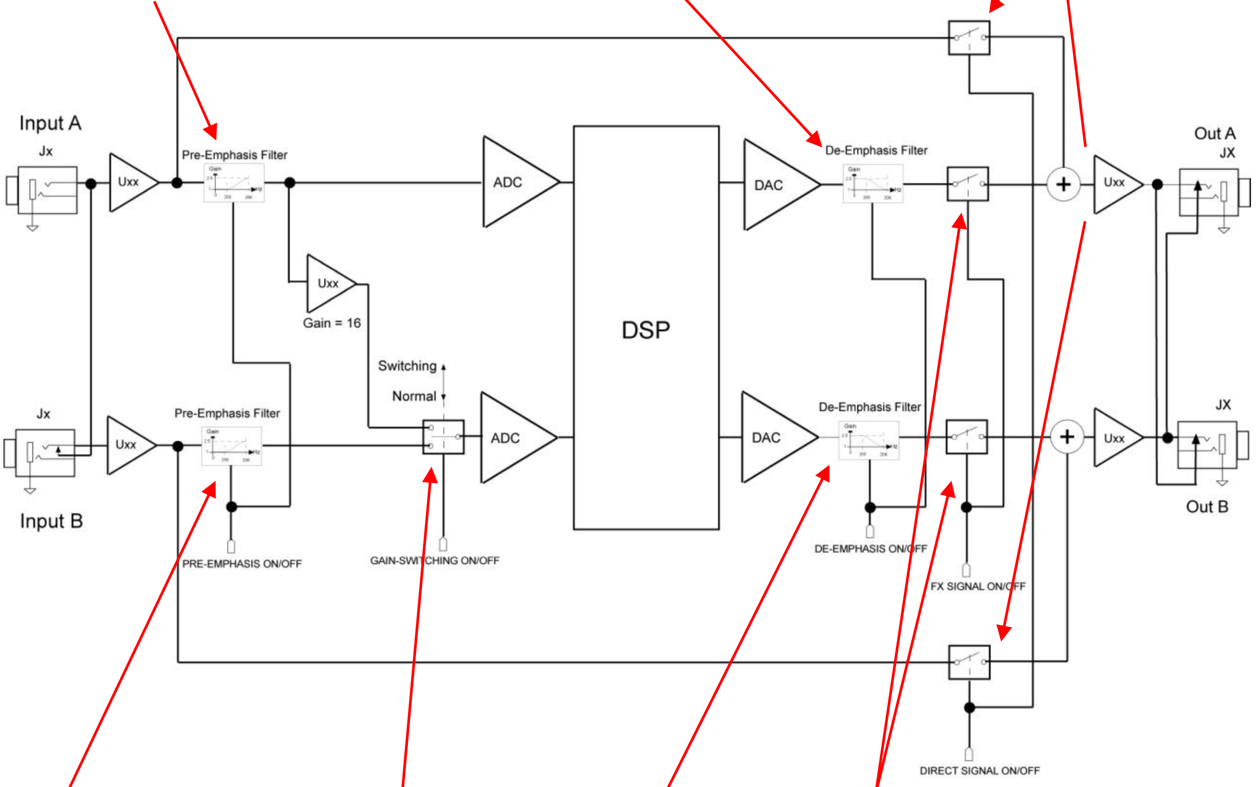
Audio System

The complete schematic of the ToneCore Dock's audio section can be seen on page 2 of the 3 ToneCore schematic pages. As previously mentioned, the analog section has a lot of versatility to accommodate different approaches to processing effects. According to the code of the ToneCore Module installed in the pedal, the audio path is configured to best match the requirements of the effect.
Below is a block diagram and description of the audio path and the audio analog switches:

The Input A pre-emphasis filter is made of U2-C/ R8/C7/R92 and it is switched On or Off with the analog switch U4-A

The Output A de-emphasis filter is made of R71/R74/C11 and it is switched On or Off with the FET Q6

The direct path is switched On and Off with FETs Q4 and Q5



The Input B pre-emphasis filter is made of U2-D/ R11/C9/R77 and it is switched On or Off with the analog switch U4-C

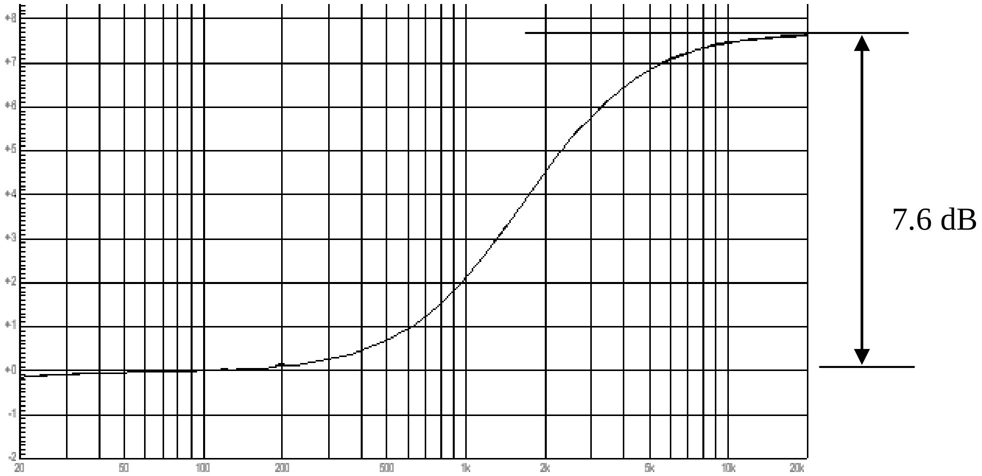
The Output B de-emphasis filter is made of R67/R69/C18 and it is switched On or Off with the FET Q8

The effect path is switched On and Off with FETs Q7 and Q10

The Switching/Normal mode switch is implemented by U4-B

Pre-Emphasis Circuit

The pre-emphasis circuit boosts high frequencies before the ADC with the following curve:



This circuit should be used in conjunction with either the analog de-emphasis circuit, or a digital de-emphasis DSP algorithm (see below). When the pre-emphasis and either of the two de-emphasis functions are used simultaneously, the resulting IN to OUT frequency response is flat. The benefit of using the pre and post emphasis circuits is a Signal Noise Ratio improvement of about 4dB.

The drawback of the pre-emphasis circuit is that it will reduce the available headroom for higher frequencies signals. This can be a problem if the input signal has large amplitude components in the frequencies above 2 kHz. This is usually not the case for a clean guitar signal, but it may become problematic for other sound sources, and for processed guitar signals, which have high frequency content (i.e., after distortion).

De-Emphasis Circuit

The de-emphasis circuit attenuates the high frequencies after the DAC along a curve complementary to the pre-emphasis curve.
Some DSP algorithms (i.e. distortion) intentionally add high harmonics to the input signal. In this case, using the analog de-emphasis circuit after the DAC would reduce the high harmonic content. For this situation, the de-emphasis can be implemented in the DSP code directly after the ADC and before the harmonic creating algorithm using a standard filter algorithm. The noise reduction benefit in this configuration is limited to the ADC alone.

Gain Switching Circuit

This circuit allows you to use the two ADC channels in combination to create a single channel with a greater signal noise ratio (SNR). When this mode is used, only the A input jack can be routed through the DSP, and the B input can only be routed through the analog direct path. When this circuit is active, the A input feeds both ADC channels. The analog gain on the upper ADC branch is set so that this ADC input is fully modulated when the A input signal is at 5Vpp (@ below 200Hz when pre-emphasis is used). The gain on the lower ADC branch is 16 times higher (24dB), and will therefore clip the lower ADC input whenever the A input signal is more than 5/16 Vpp (= 300 mVpp). When this configuration is used, a DSP algorithm is required to monitor both ADC signals and automatically select the most appropriate one in order to maximize the signal/noise performance. As long as the signal on the lower ADC is below its maximum input level, this ADC would be routed into the DSP algorithm (attenuated by 24dB), otherwise the upper ADC will be used. Since the noise performance of the ToneCore design is already of high quality (≈100dB S/N), it is not expected that this circuit would normally be used. If an effect is created that has a significant amount of gain in the algorithm, the gain switching option can reduce the awareness of the input noise floor.

Direct (bypass) and Effect Path Gain

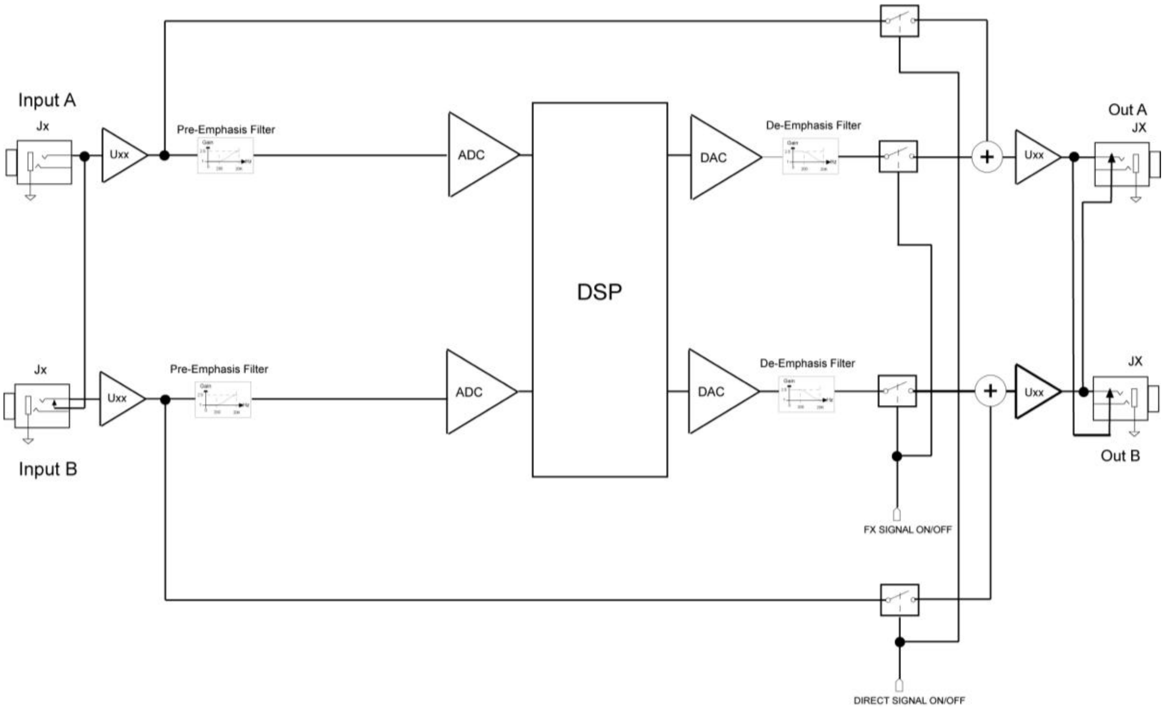
The A and B audio input signals can be routed directly to the ToneCore Dock outputs by switching on the Direct Path analog switch. The gain from the input to output jack is unity. The DAC outputs can be routed to the ToneCore Dock outputs by switching on the Effect Path analog switch. The gain from the DAC to the output jack is 1.5.

Examples of Audio Path Setups

The following pages give three examples of some of the various possibilities of audio setups with the analog controls available in the ToneCore Dock.

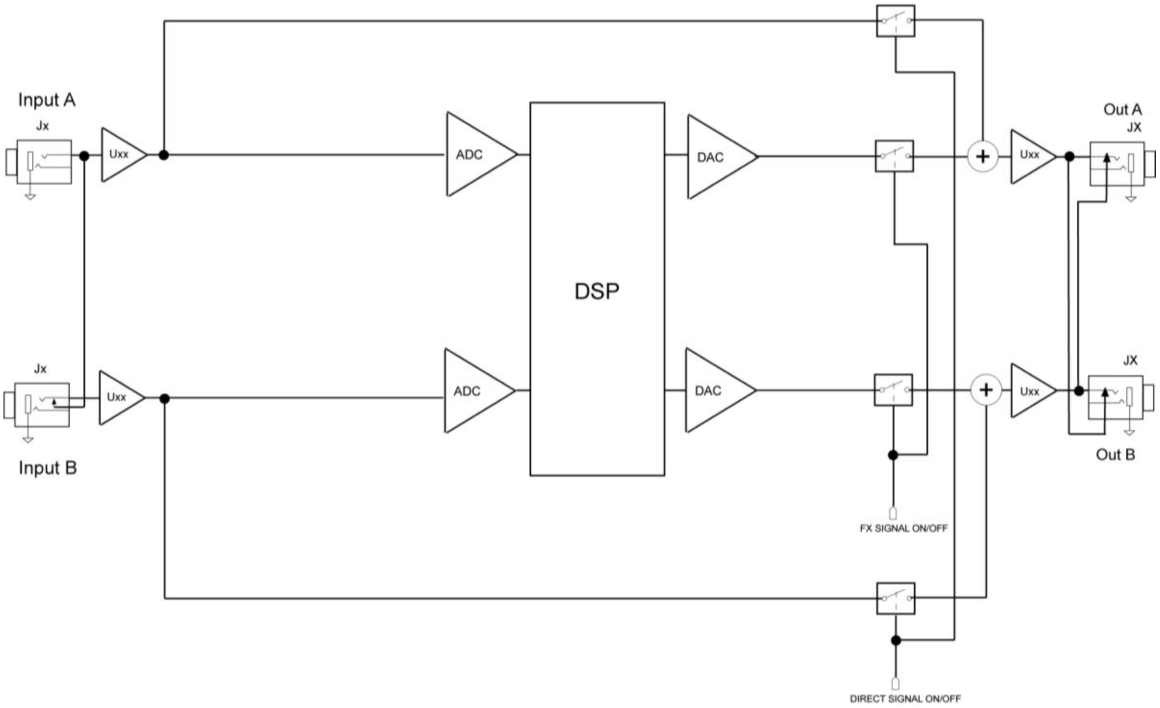
Normal gain stereo with emphasis

For the case of an effect requiring a stereo audio path with pre and post emphasis (i.e. a stereo tremolo pedal), the analog switches are set up to achieve the following audio path:
In the example below, the pre-emphasis and de-emphasis circuits are active.



Low gain stereo without emphasis

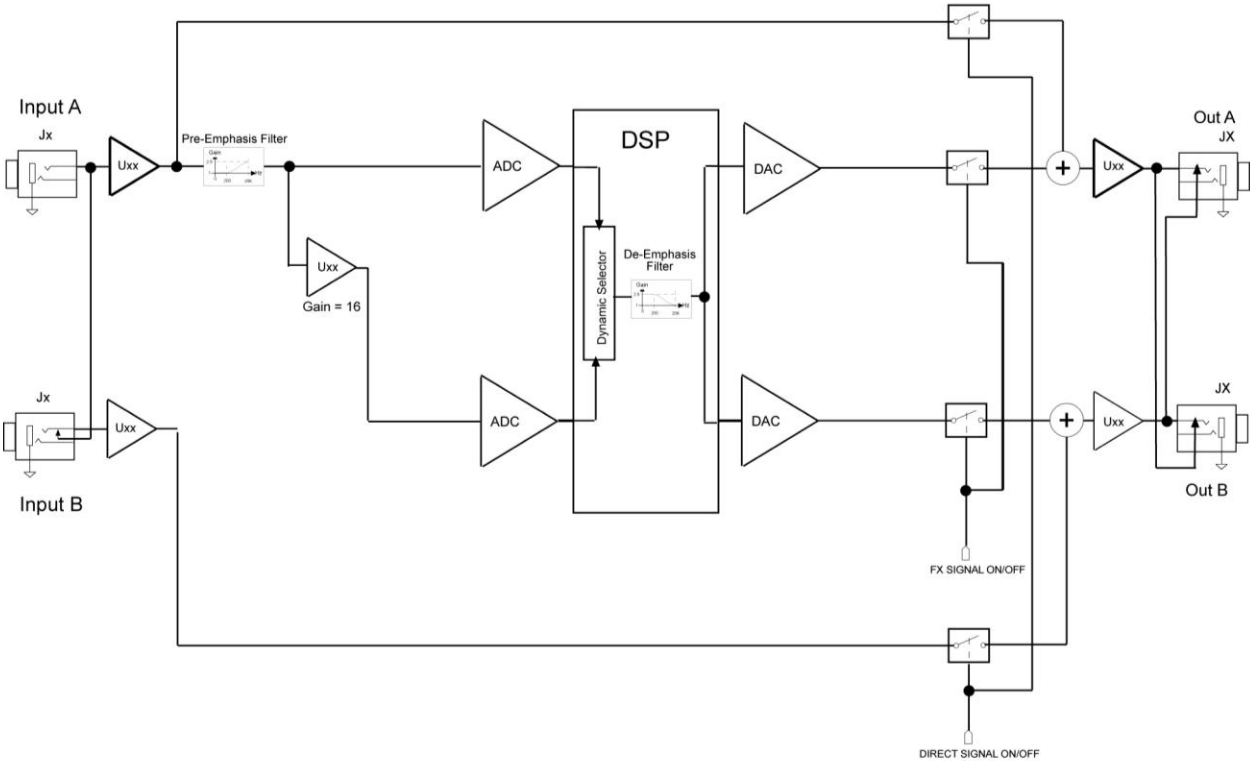
For the case of ToneCore Module requiring a low gain stereo audio path without pre or post emphasis, the analog switches are set up to achieve the following audio path:
In the example below, the pre-emphasis, gain switching and de-emphasis circuits are inactive.



High gain mono

For the case of an effect requiring a high gain mono audio path with pre emphasis and no post de-emphasis (i.e. a distortion pedal), the analog switches are set up to achieve the following audio path:

In the example below, the pre-emphasis and gain switching circuits are active. The de-emphasis and ADC recombination are implemented in the DSP code.



Audio Inputs

The nominal input impedance of each input jack is 1M Ohm when both inputs are used, and 500K Ohm when only the A input is used.
The maximum audio input level depends on the DC voltage available from the battery. With a fully loaded 9V battery, or the external DC1 supply, the maximum input voltage is 5 Vpp. If the pre-emphasis circuit is used this maximum input voltage will decrease for frequencies above 200Hz.

	DC in = 9.6V	DC = 6.0V
- Input max Level before clipping without pre-emphasis:	5.2 Vpp	2.3 Vpp
- Input max Level before clipping with pre-emphasis:		
• From DC to 200 Hz	5.2 Vpp	2.3 Vpp
• @ 1KHz	4.0 Vpp	1.9 Vpp
• @ 2KHz	3.0 Vpp	1.5 Vpp
• @ 10KHz	2.2 Vpp	1.1 Vpp

Audio Outputs

The output impedance is 680 Ohms when both outputs are used and 340 Ohms when only the A or B output is used (since the outputs are summed in this case).
The maximum audio output level depends on the DC voltage available from the battery. With a fully loaded 9V battery, or the external DC1 supply, the maximum output voltage is 7.3 Vpp.

	DC in = 9.6V	DC = 6.0V
- Output maximum level:	7.3 Vpp	3.7 Vpp

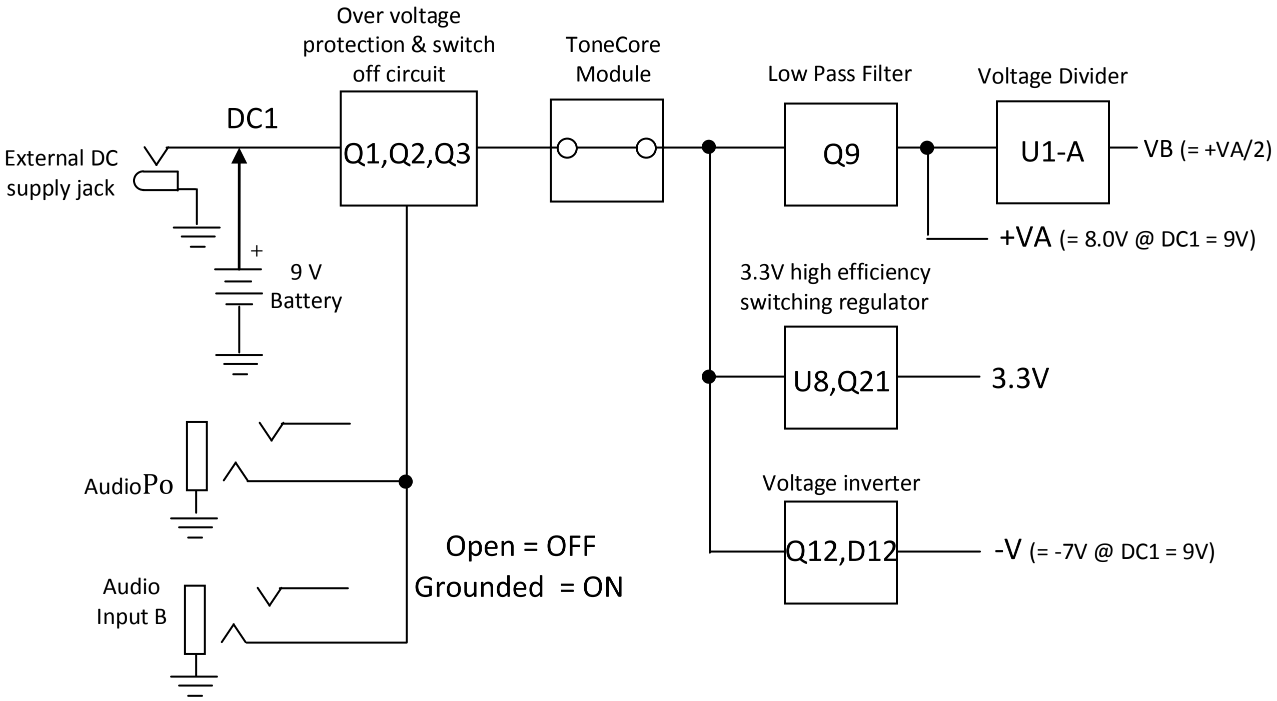
Power Supply

The power supply and analog reference voltage circuitry is contained on page 1 of the ToneCore Dock schematic.

For the ToneCore to power up, three conditions must be simultaneously satisfied:

- Either a battery with at least 6V of voltage or an external supply must be present
- A MONO ¼" jack must be plugged into either the A or B audio inputs.
- A ToneCore Module must be inserted in the ToneCore Dock.

Power Supply Flow Chart



With a DC1 supply connected at the external DC supply jack, the supply voltage readings will be:

Supply name	Value	Function
DC1	9.5V +/- 10%	External AC adaptor supply
+VA	8.3V +/- 10%	Analog supply
-V	-7.7V +/- 10%	Analog supply for audio switches
VB	VA/2 = 4.15V+/- 10%	Analog mid supply bias point
3.3V	3.3V +/- 10%	DSP, ToneCore Module, Codec supply

Power Consumption

The ToneCore power consumption is significantly affected by the activity of the DSP. The more DSP instructions are executed per unit of time, the higher the consumption will be (this assumes that the DSP is set in sleep mode when it is idling). The type of DSP code executed will therefore affect the battery life, but will not have any impact when using a DC1.

% of DSP use	25	50	75	100
Current at the Battery terminals (mA)	45	55	72	90
Expected battery life (hours)*	9	7	5	4

* Starting with a fully loaded alkaline battery, running continuously with a 5Vpp input signal.

Minimum Battery Voltage

The minimum battery voltage necessary for useful operations is 6V.

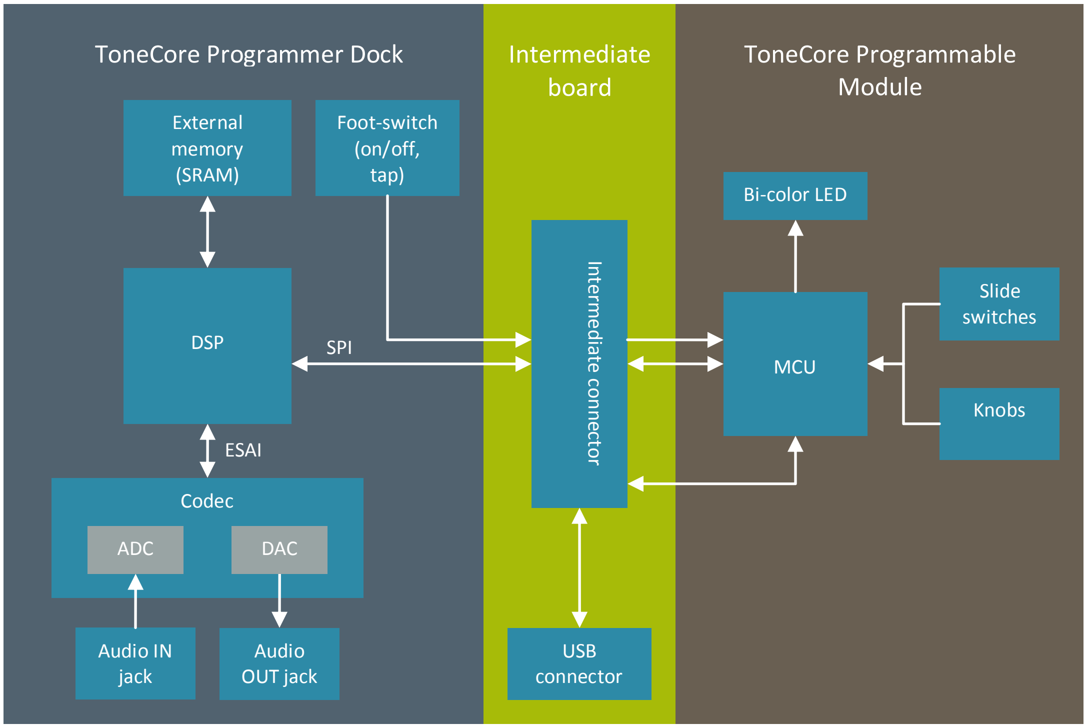
Digital System

The digital system includes the DSP in the dock and the MCU in the module. It is mostly contained on page 1 and 3 of the schematics.

Note: In this document, the * sign next to a digital signal name indicates an active low signal.

Digital Block Diagram

The block diagram below presents the digital system of the ToneCore DSP Developer Kit:



The dock and the module are connected together by an intermediate board. This intermediate board has the mini-USB connector. The DSP and the MCU communicate with each other using the SPI protocol. The SPI (Serial Peripheral Interface) is a synchronous serial data link standard that can operate in full duplex mode. Devices communicate in a master/slave mode. The master device initiates the transfers and it can communicate with multiple slaves by using slave select (chip select) lines.

In the TCDDK, the communication between the MCU and the DSP uses the following lines:

- **SPI_MOSI**: The SPI Master Out Slave In serial data line.
- **SPI_MISO**: The SPI Master In Slave Out serial data line.
- **SPI_SCK**: The SPI serial bit clock. There is one clock cycle on this line for every serial data bit transferred between devices.
- **SPI_SS_DOCK***: The MCU uses this line as a master to select the slave DSP.
- **SPI_SS_MODULE***: The DSP uses this line as a master to select the slave MCU.

- **SPI_HREQ***: This line is used by the slave to throttle the data transfer coming in from the master.

The MCU uses an external crystal of 12 MHz. The DSP has no external crystal; instead, it uses a clock signal provided by an output pin of the MCU. The MCU clock (12 MHz) is divided by four and sent out to the DSP as 3 MHz nominal. This is done using a pin of the MCU configured as a PWM output. The DSP uses its internal PLL to multiply this up to 100 MHz.

The DSP in the Dock sends and retrieves serial digital audio data to and from the CODEC (U5) using an I2S serial digital audio communication scheme. This is performed using the DSP's Enhanced Serial AUDIO Interface (ESAI). The I2S lines are defined below:

- **256FS_CLK**: 10 MHz used as a master clock for the CODEC. This clock is sourced by the DSP from its master clock.
- **64FS_CLK**: 2.5 MHz is used as a bit clock for the serial digital audio transfer. There is one clock cycle on this clock for every bit of serial digital audio transferred.
- **FS_CLK**: 39.0625 kHz. This clock is cycled once per digital audio sample period. A complete set of 24 bit stereo words is sent during this clock cycle. **This is the audio sampling frequency**.
- **DAC_SDATA**: This is the serial digital audio output stream from the DSP to the CODEC's DAC.
- **ADC_SDATA**: This is the serial digital audio input stream from the CODEC's ADC to the DSP.

I/O Control

The following GPIO pins on the DSP are used to control or read the states of the audio path analog switches:

Port bit	I/O	Content	Format	Description	DSP Pin & Schematic Name
Port B bit 0	Output	Pre-emphasis switch	0=OFF, 1 = ON	Sets the input pre-emphasis when high (3.3V).	91 - IN_EMPH
Port B bit 1	Output	De-emphasis switch	0=OFF, 1 = ON	Sets the output de-emphasis when high (3.3V).	94 - OUT_EMPH
Port B bit 2	Output	Direct path switch	0=OFF, 1 = ON	Sets the audio analog direct path on when high (3.3V).	95 - DIRECT_ON
Port B bit 3	Output	Effect path switch	0=OFF, 1 = ON	Sets the audio effect path on (DAC output) when high (3.3V).	96 - FX_ON
Port C bit 2	Output	Gain Switching (16x on ADC input 2)	0=OFF, 1 = ON	Sets the gain switching mode when high (3.3V).	13 - GAIN_SWITCHING
Port C bit 7	Input	ToneCore Dock input B audio jack status	0=Out, 1=In	Senses if a jack is plugged into Audio Input B. The signal is low if nothing is plugged in, 3.3V if it is plugged in.	19 - B_IN
Port C bit 8	Input	ToneCore Dock input A audio jack status	0=Out, 1=In	Senses if a jack is plugged into Audio Input A. The signal is low if nothing is plugged in, 3.3V if it is plugged in.	18 - A_IN
Port C bit 9	Input	ToneCore Dock output audio jack status	0=Mono, 1=Stereo	Senses if both audio input jacks are plugged in. Low if not, 3.3V if true.	17 - STEREO/MONO

Boot Up Sequence

The ToneCore Pedal boot up sequence is outlined below:
Upon power on, the boot up sequence begins in the ToneCore Module. As the MCU (U1) clock and power get stable, it sends the MASTER_RESET* signal out to the ToneCore Dock to reset the DSP. When the signal MASTER_RESET* goes high, the reset period is over.

After the reset period is over, the DSP goes into power up boot mode and awaits its firmware. The MCU sends the DSP firmware (which is in the MCU flash) to the DSP over its SPI port serially. After the DSP receives all of its firmware, it begins to execute its runtime code from RAM and the system is now fully running.

Mono or Stereo ToneCore Dock

The PCB jumpers installed in a Mono ToneCore Dock make it behave exactly like a Stereo ToneCore dock with nothing connected to the B input and B output jacks. Consequently, as long as a ToneCore Module's code is designed to handle this situation in a Stereo ToneCore Dock, it will work the same on a Mono ToneCore Dock.

DSP Code Storage

The MCU has 16 kB of flash memory and the default memory map can be found in Appendix C of the TCDDK Users Guide. You can use the default MCU memory mapping or change it according to your needs (**warning**: advanced embedded programming skills needed). By default, the section of flash dedicated to store the DSP code is 4 kB. This means that your DSP code must be smaller than 4095 Bytes (or 1365 x 24 bit words).

The DSP has the following memory spaces:

- 1.5k × 24 Bit Y-Data RAM.
- 1k × 24 Bit X-Data RAM.
- 8k × 24 Bit Program ROM (cannot be used)
- 0.5k × 24 Bit Program RAM and 192 × 24 Bit Bootstrap ROM.
- 0.75k × 24 Bit from Y Data RAM can be switched to Program RAM resulting in up to 1.25k × 24 Bit of Program RAM.

Remember that the DSP executes the code stored in the program RAM. Note that the DSP can be configured to use 1.25 k x 24 Bit (1280 words) of program RAM. By default, the MCU code configures the DSP at start-up to work in this extended memory mode where the memory space is:

- 0.75k of 24 bit Y-Data RAM
- 1k of 24 bit X-Data RAM.
- 1.25 k of 24 bit Program RAM

Alternatively, the MCU code can be modified to reduce the program RAM to 0.5k words and expand the Y-Data RAM space to 1.5K words. Refer to the DSP56364 Users Manual for more information.

External Memory

For effects, like reverb and delay, that require more memory than what the DSP has available internally, an external memory is connected to the DSP. The type of external memory is an SRAM, and its memory is organized as 512k by 8 bits. For a sample rate of 40 kHz, this SRAM provides 4.37 seconds of delay time.

(4.37 seconds = ((512 * 1024 words) / 3 bytes per sample) / 40000 samples/second)

While the DSP56364 has only an 8-bit data bus, the DMA may be used to automatically pack and unpack 24-bit data into the 8-bit wide external memory, during DMA data transfers. Refer to TCDDK Users Guide Appendix A for an example on how to use the external memory.

Graphical User Interface (GUI)

The TCDDK comes with a GUI that allows the user to download new code to the MCU (and therefore to the DSP). The communication to the GUI is achieved through USB by the MCU.

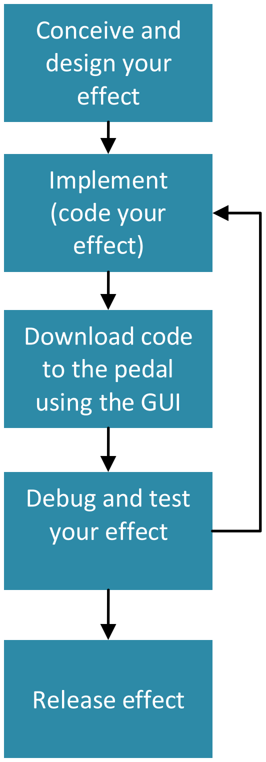
The GUI allows you to perform the following actions:

- **Download code for the DSP**. You can download your compiled DSP code to the module to run your audio effect. Even though the DSP code is stored in the MCU's flash, the behavior of the MCU is not affected by this process.
- **Monitor debug variables, knobs, switches, and pedal status**. The GUI allows you to watch the status of the pedal inputs. You can also receive/send debug variables from/to the DSP.
- **Re-flash MCU**. If you have modified the MCU code you can download it to the module and change the behavior of the whole system (not only the DSP).

Please refer to the TCDDK Users Guide for more information on how to install and use the GUI.

Effect development

The typical development cycle can be represented with the following diagram:


Conceive and design your effect
Implement (code your effect)
Download code to the pedal using the GUI
Debug and test your effect
Release effect

Before you begin to write any code, you need to understand how the effect will work, how it will modify the audio input and how to implement it. You may have to perform some research before you get started.

After knowing what you want to do, you need to translate it to code. You can use the template provided with the TCDDK as a starting point. You can use Symphony Studio (free) or other 3rd party (non-free compiler/IDE) to edit the DSP code. For advanced effects you can also modify the MCU code using CodeWarrior.

Download the code to the pedal using the GUI. Refer to the TCDDK Users Guide for more information.

You can debug your application using the Symphony Studio Simulator. You can also use the debug variables to send information from the DSP to the GUI and vice versa. This stage may take some time and will probably require code iterations to correct problems. Remember to test your effect under all possible usage scenarios.

Now your effect is working as expected and is ready for release. You can perform some extra actions such as disabling the USB connectivity in the MCU (to reduce power consumption).

ToneCore Module Coding

Depending on the complexity and requirements of your effect, you can take two different approaches in coding: the “simple” way and the “advanced” way.

The “Simple” way

In this first approach you will modify only the DSP code, not the MCU code. Using this method, you can take advantage of some basic code already pre-programmed in the ToneCore Programmable Module micro processor’s Flash (a blank ToneCore Module is therefore not really “blank” after all).

The MCU is already programmed to perform the following actions:

1. **Initialize MCU**. Upon power up, the MCU will configure the clock module, USB module, GPIOs, timers, ADC, etc.

2. **Bootload DSP**. The MCU will serially bootload the DSP with the code stored in a special area of flash. By default (factory setting), the module comes with a 2-band stereo equalizer code. Refer to the Users Guide for more information.

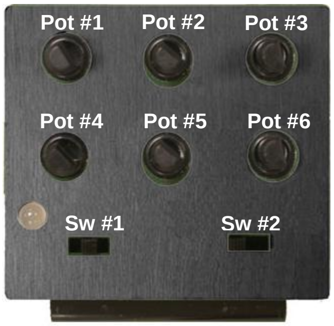
3. **Continuously scan inputs**. After bootloading the DSP, the MCU will periodically read the module potentiometers and slide switches; read the dock foot switches (effect On/Off and Tap tempo). It will also read the battery voltage from the dock.

4. **Continuously send/receive information to/from the DSP**. The MCU will periodically communicate with the DSP to send and receive the status of the following variables:

DSP Address:		Content:		Format
				MSB LSB
0x0000	Input	Potentiometer #1 (24 bit value)	0 @ Full CCW (See note 1)	aaaaaaaaaaaaaaaaaaaaaaaa
0x0001	Input	Potentiometer #2 (24 bit value)	0 @ Full CCW	aaaaaaaaaaaaaaaaaaaaaaaa
0x0002	Input	Potentiometer #3 (24 bit value)	0 @ Full CCW	aaaaaaaaaaaaaaaaaaaaaaaa
0x0003	Input	Potentiometer #4 (24 bit value)	0 @ Full CCW	aaaaaaaaaaaaaaaaaaaaaaaa
0x0004	Input	Potentiometer #5 (24 bit value)	0 @ Full CCW	aaaaaaaaaaaaaaaaaaaaaaaa
0x0005	Input	Potentiometer #6 (24 bit value)	0 @ Full CCW	aaaaaaaaaaaaaaaaaaaaaaaa
0x0006	Input	ToneCore Module switch #1 status	0=Left, 1=Center, 2=Right	XXXXXXXXXXXXXXXXXXXXXXaa
0x0007	Input	ToneCore Module switch #2 status	0=Left, 1=Center, 2=Right	XXXXXXXXXXXXXXXXXXXXXXaa
0x0008	Input	Tap tempo switch status	0=Open, 1=Closed	XXXXXXXXXXXXXXXXXXXXXXXa
0x0009	Input	Effect ON/OFF switch status	0=Open, 1=Closed	XXXXXXXXXXXXXXXXXXXXXXXa
0x000A	Output	RED LED status	0=OFF, 1=ON	XXXXXXXXXXXXXXXXXXXXXXXa
0x000B	Output	GREEN LED status	0=OFF, 1=ON	XXXXXXXXXXXXXXXXXXXXXXXa
0x0012	Input	Debug Variable To DSP 1 (24 bit value)		aaaaaaaaaaaaaaaaaaaaaaaa
0x0013	Input	Debug Variable To DSP 2 (24 bit value)		aaaaaaaaaaaaaaaaaaaaaaaa
0x0014	Input	Debug Variable To DSP 3 (24 bit value)		aaaaaaaaaaaaaaaaaaaaaaaa
0x0015	Input	Debug Variable To DSP 4 (24 bit value)		aaaaaaaaaaaaaaaaaaaaaaaa
0x0016	Output	Debug Variable From DSP 1 (24 bit value)		aaaaaaaaaaaaaaaaaaaaaaaa
0x0017	Output	Debug Variable From DSP 2 (24 bit value)		aaaaaaaaaaaaaaaaaaaaaaaa
0x0018	Output	Debug Variable From DSP 3 (24 bit value)		aaaaaaaaaaaaaaaaaaaaaaaa
0x0019	Output	Debug Variable From DSP 4 (24 bit value)		aaaaaaaaaaaaaaaaaaaaaaaa

Note 1: All potentiometer values are linearly proportional to the potentiometer rotation angle. In other words, all potentiometers are linear taper.
Note 2: These variables are sent/received every 1 ms by default. You can change this value (see Advanced way)

The following picture shows the ToneCore Module Controls mapping:



5. **Communicate with the GUI**. The MCU will periodically send (through USB) the above data to the GUI and receive back 4 debug variables which are then sent to the DSP along with the rest of the variables.

To implement a custom DSP algorithm you need to follow these steps (refer to the Users Guide for a detailed description):

1. Modify the code using Symphony Studio.
2. Compile the code to generate a CLD file (absolute object file).

3. Open the GUI and load the CLD file. The GUI will convert the CLD file into an array of bytes that will be stored in the MCU's flash.
4. Download the code to the MCU. This will store the new code in the flash. The MCU will then restart the DSP and bootload it with the new code.

This "Simple way" of programming can have some limitations. Since you only modify the DSP code you must follow the rules established to communicate with the MCU. You don't modify what the MCU is doing, but you have the most straightforward way to develop an effect algorithm without needing to write any MCU code.

The "Advanced" Way

In this method you modify both the DSP code and the MCU code. With this you have full control of what the system is doing. The MCU can assist the DSP in some functions, or you can distribute the processing tasks (keeping in mind that the MCU has an 8-bit architecture).

Since this results in full control of both processing devices, there are no rules about how the interface between the ToneCore Module and ToneCore Dock is handled (except of course for the hardware constraints).

To change the MCU code you must download and install CodeWarrior for Microcontrollers version 6.2 or higher (there is a free version). With this IDE you can modify, compile and simulate code for the MCU.

Keep in mind that modifying the MCU code can completely alter the behavior of the pedal. If you modify the MCU code and for some reason you lose USB communication with the PC through the GUI, you can always roll back to the factory default condition using a special method (refer to the Users Manual section on "How To Start The TCDDK In Bootloader Mode").

This restore-firmware capability is possible because there is a small protected area in the MCU flash that you cannot modify. This area contains a small MCU bootloader that will run when you power up the pedal with the foot switch pressed down. This MCU bootloader establishes communication with the GUI and allows you to re-flash the rest of the memory with a code that works correctly (for example: the code provided with the TCDDK installer).

This protected MCU bootloader should not be confused with the bootloading process of the DSP. The MCU bootloader, also referred to as the "TCDDK bootloader mode", is the small protected application in the MCU that allows restoring the system when things go wrong. On the other hand (when the system is working ok) after powering on, the MCU serially sends the code that the DSP will run. This process is also referred as "bootloading the DSP" or "DSP bootload".

DSP and MCU sample code

The TCDDK installer includes two different code examples, one for the DSP and the other for the MCU.

The DSP example implements a 2-band stereo equalizer and by default is installed in the following path:

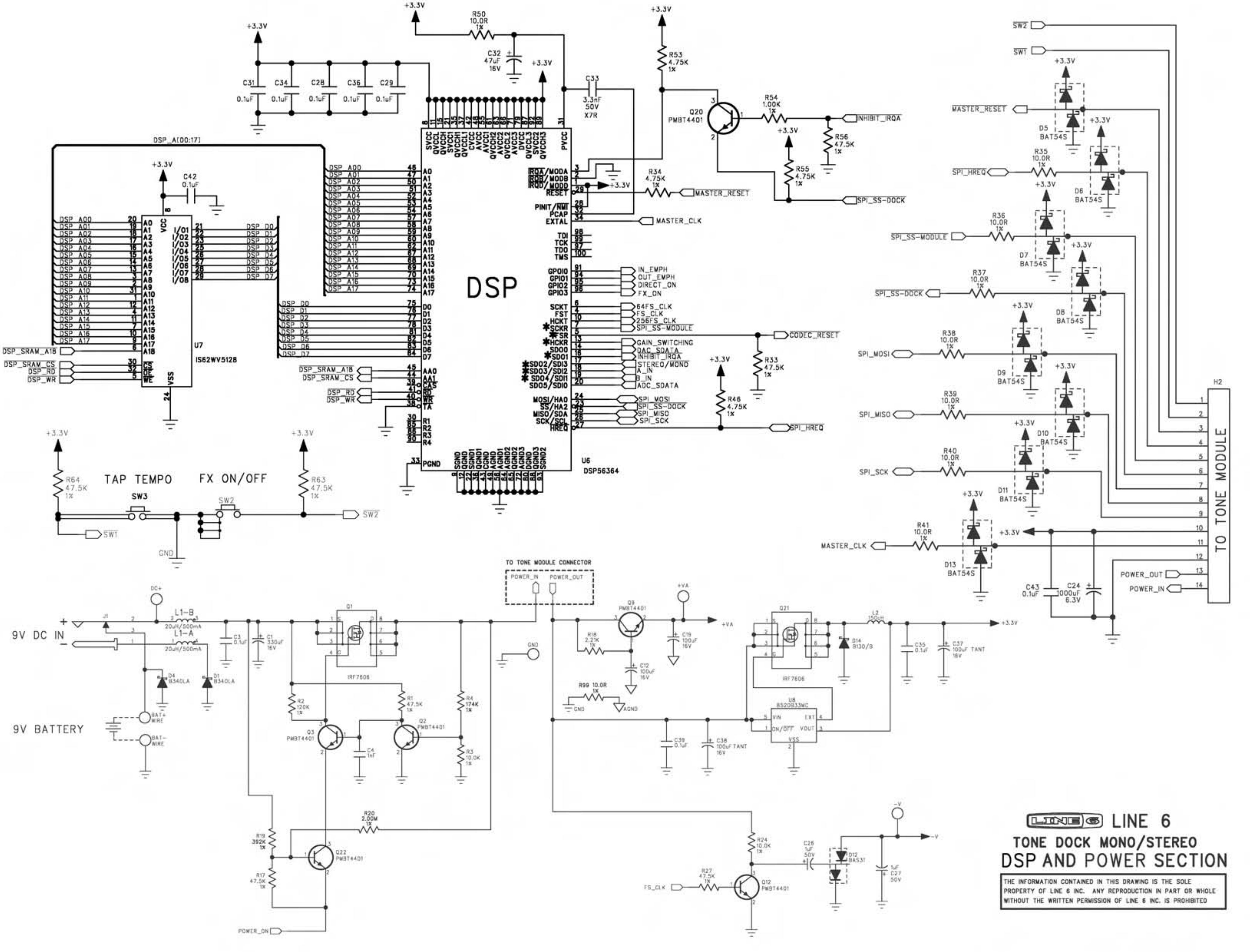
C:\Program Files\Line 6\TCDDK v1.0\DSP Sample Code

This DSP equalizer example is written in assembly for the DSP56300 and can be edited and compiled using Symphony Studio (or any 3rd party DSP56300 compiler).

For the MCU, a code template is installed by default in the following path:

C:\Program Files\Line 6\TCDDK v1.0\MCU Sample Code

This project provides the code that allows the MCU to perform as described earlier. You can use this as a template to modify the behavior of the MCU. Refer to the Users Guide for a detailed explanation of the code.

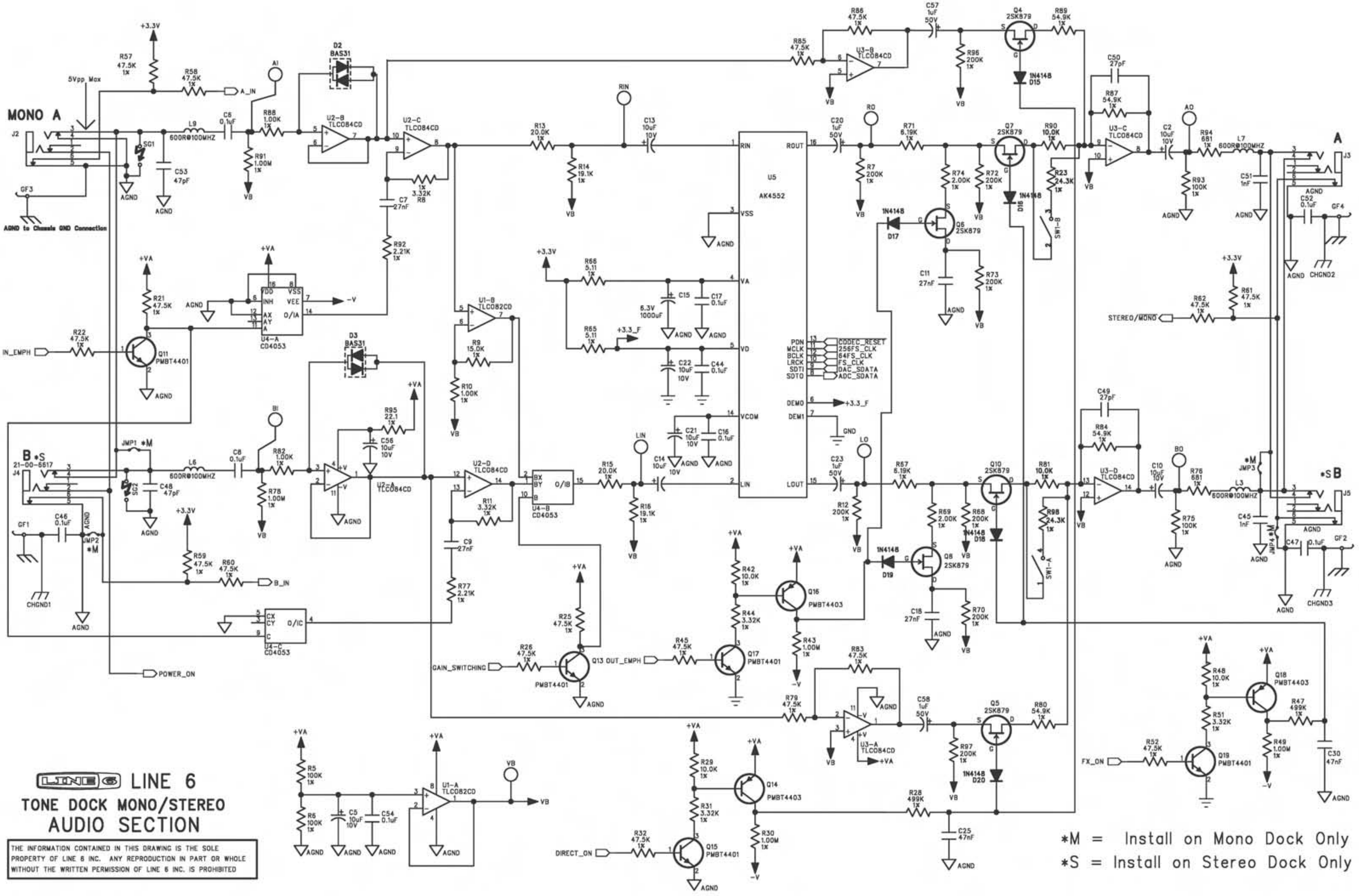


LINE 6

TONE DOCK MONO/STEREO

DSP AND POWER SECTION

THE INFORMATION CONTAINED IN THIS DRAWING IS THE SOLE PROPERTY OF LINE 6 INC. ANY REPRODUCTION IN PART OR WHOLE WITHOUT THE WRITTEN PERMISSION OF LINE 6 INC. IS PROHIBITED



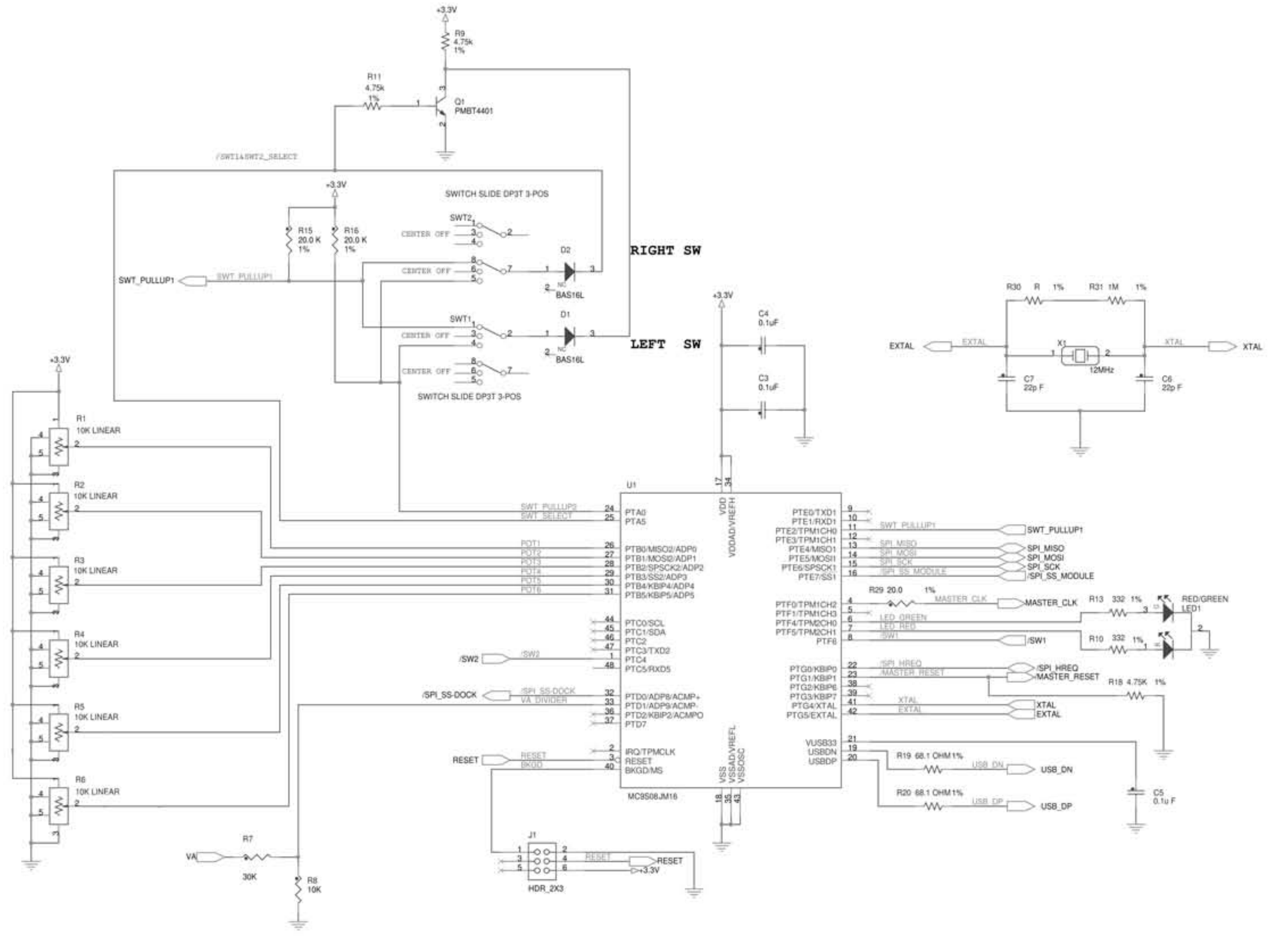
LINE 6

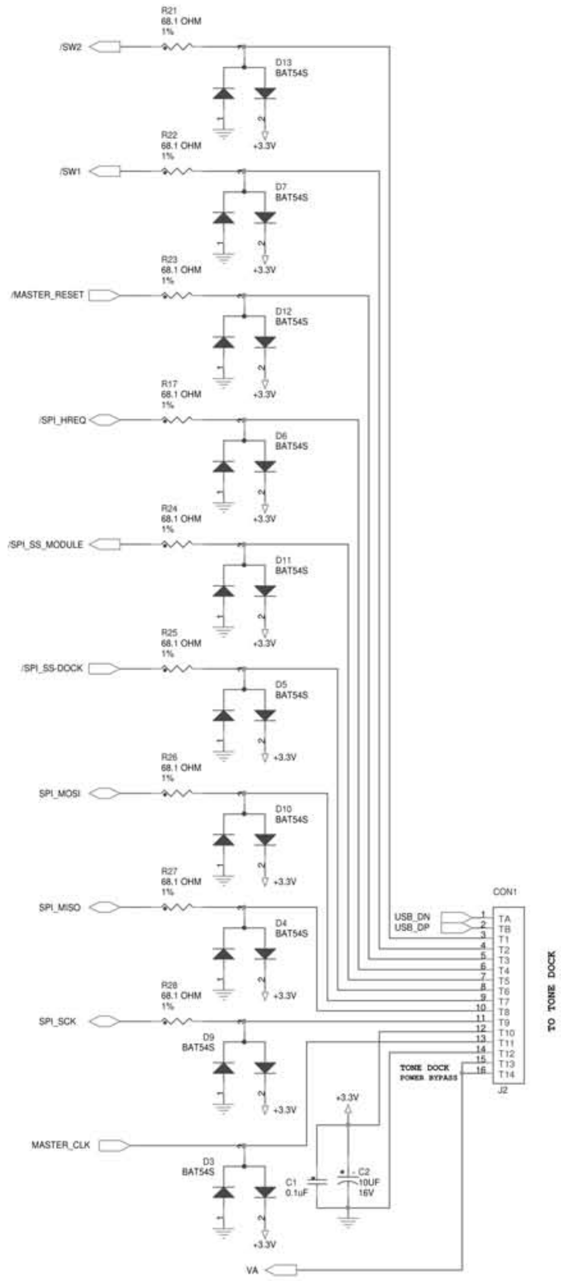
TONE DOCK MONO/STEREO AUDIO SECTION

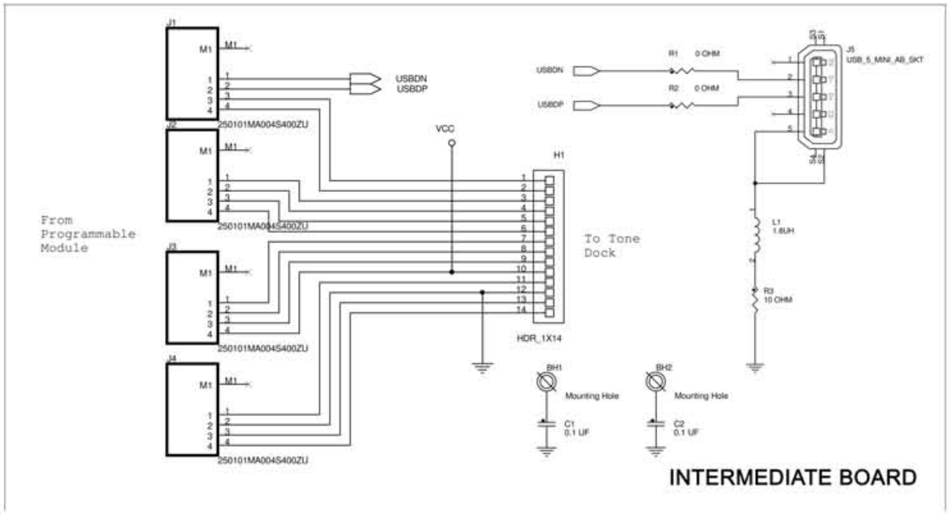
THE INFORMATION CONTAINED IN THIS DRAWING IS THE SOLE PROPERTY OF LINE 6 INC. ANY REPRODUCTION IN PART OR WHOLE WITHOUT THE WRITTEN PERMISSION OF LINE 6 INC. IS PROHIBITED

*M = Install on Mono Dock Only

*S = Install on Stereo Dock Only







INTERMEDIATE BOARD

LINE 6

TONECORE DSP DEVELOPER KIT
PROGRAMMABLE MODULE

THE INFORMATION CONTAINED IN THIS DRAWING IS THE SOLE PROPERTY OF LINE 6 INC. ANY REPRODUCTION IN PART OR WHOLE WITHOUT THE WRITTEN PERMISSION OF LINE 6 INC. IS PROHIBITED