Orion-I Beacon Manual

8/11/2018

Joe Haas, KE0FF joeh-*at*-rollanet-*dot*-org

The Orion-I is a PLL synthesizer based on the ADF-4351 PLL chip. It utilizes a SiLabs C8051F520 microcontroller to control the synthesizer function via the PLL registers. The microcontroller has additional resources available and these have been utilized to create a Morse Code message system that can be used to produce a stand-alone, on-off keyed, repeating message keyer (a.k.a., a beacon). This manual presents the operating particulars for this system. Refer to the Orion-I Operation Manual for information on details regarding mechanical installation and PLL channel data configuration.

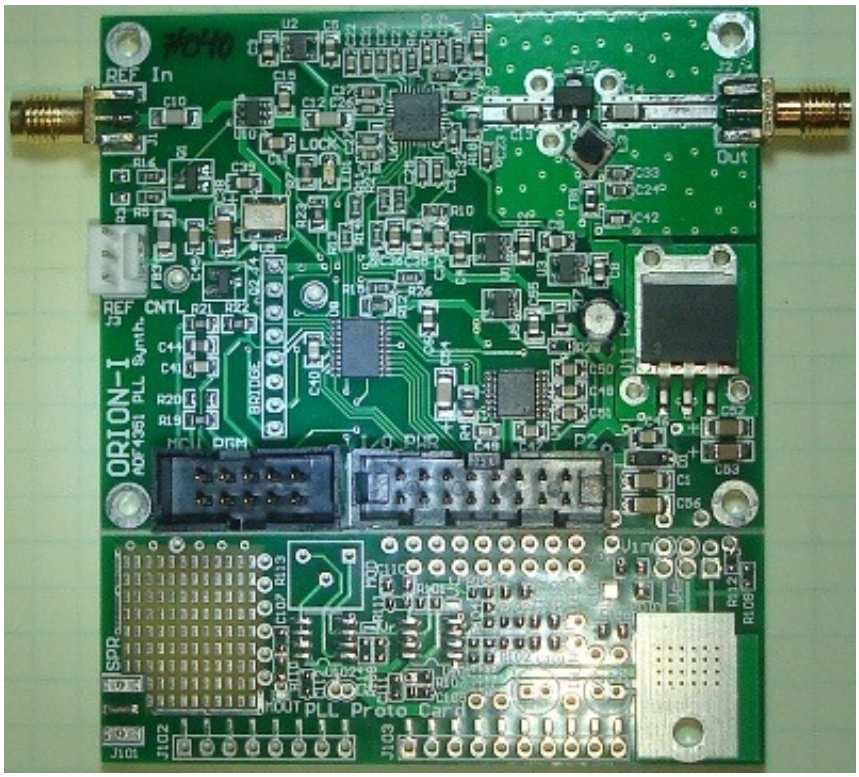

Figure 1. Orion-I PLL Synthesizer

Operation

The beacon software is derived from the basic PLL software and thus has many similarities to the user interface. However, some of the Orion resources and commands have been modified or eliminated for the beacon application. Also, there are some new commands to support the requirements of the message keyer. The P2 pinout below illustrates a change to the wiring of the main connector. For the beacon software, FSEL7-4 (P2-15) are configured to be outputs. As such, they should not be connected the same as the other FSEL signals (3-0). FSEL7 is used as an external keying output which may be used to on-off key a PA stage and/or apply wave-shaping to the modulated output. It could also be used to drive an audio oscillator to produce FM modulation.

P2 Pinout			
+Vin	1	2	GND
RS-232_RXD	3	4	+3.3V out
RS-232 TXD	5	6	GND
/KEY_INPUT	7	8	FSEL0 (CH b0)
(CH b1) FSEL1	9	10	FSEL2 (CH b2)
(CH b3) FSEL3	11	12	FSEL4 (GP out b0)
(GP out b1) FSEL5	13	14	FSEL6 (GP out b2)
KEY OUTPUT	15	16	GND

Note: All logic signals are 3.3V logic and will NOT tolerate higher voltage logic

Chan	FSEL[3:0]
00	0000
01	0001
02	0010
03	0011
04	0100
05	0101
06	0110
07	0111
08	1000
09	1001
10	1010
11	1011
12	1100
13	1101
14	1110
15	1111

The real-time frequency control of the PLL featured in the original Orion software is not available with the beacon software. Instead, the PLL channel is sampled at power-up only or if the “i" command is entered via the serial port.

The +5V output (P2-4) can power external circuits, but every mA supplied to this pin increases the heat-load on U11. The max current that can be supplied is about 200 mA, but this should generally be limited to less than 10 mA unless a thermal solution is verified for U11. Failure to follow these guidelines could result in failure of the U11 regulator, and possible damage to other components.

Frequency Selection (Channels)

The Orion beacon software features 16 PLL channels. This is reduced from the original software to create space for the Morse Code message. Unlike the basic PLL application, the channel selection uses a binary input to allow for the 16 different channels. The PLL registers and commands are the same as for the basic PLL application, and the user should refer to the Orion Manual for details on creating and updating channel data.

Optional /KEY_INPUT Function

For the beacon software, the /KEY_INPUT (This is P2-7, or /PTT as defined in the Orion Operating Manual) is treated differently than on the Orion PLL software. Here, this signal can be used to manually key the PLL output (/KEY_INPUT = GND). Once the /KEY_INPUT is grounded, the Morse message stops and the RF output and KEY output both go into "transmit" condition where will they stay until /KEY_INPUT is opened. The Morse message then re-starts from the beginning and normal beacon operation resumes.

Serial Port

The Orion MCU features a serial connection that is used to program the MCU channel sets and update the Morse Code message. An RS-232 transceiver is included on-board and this allows for an RS-232 serial connection to the Orion MCU. The wiring for this optional connection is described in the "UART Connections" section. At power-on, an initialization banner message is sent by Orion:

```
ADF4351 Beacon Exctr Ver 0.24, de ke0ff
16 CH, gnd-true BCD, PTT low = key out
Serial cmd enabled
CH 00
```

The reset banner provides the software version and a brief list of version features. Primarily, this message is important for identifying the current software version, and verifying that the UART TX connection to the user terminal is working. *Note: The banner message might change slightly with newer software versions.*

The beacon serial commands available are as follows:

While all of the following commands may be typed manually, the "M" and "C" commands are mainly intended to be used by either manually downloading a text file containing the commands, or by using a special-purpose PC program to send the commands after processing them from either a text or spreadsheet input. In this case, the terminal software should add a delay after each new-line of at least 50 ms.

All commands are terminated with <CR> (ASCII 0x0d, <ENTER> on most keyboards).
Serial port does <u>not</u> echo characters as they are entered.
Commands are case sensitive. Hex data may be upper or lower case.

?: List Command Help
Displays an abbreviated list of the available commands.

EC: Erase all PLL channels
EM: Erase Morse Message

Prompts the user: "Erase all, press Y to accept…" and waits 5 sec for input. Any character other than upper-case "Y", or a delay of more than 5 sec will cause this command to abort with no changes to the system. This command must be executed if the channel(s) to be programmed contain any data other than all 0xFF.

When the EM command is executed successfully, the software disables message transmission. The message remains disabled until a valid message is uploaded and the "i" command is issued, or the Orion power is cycled.

M: Program (memory) channel
`Mxx aaaaaaaa bbbbbbbb cccccccc dddddddd eeeeeeee ffffffff`

Programs channel "xx" ("xx" is BCD ASCII '00' thru '15') with "a..a" (R0), thru "f..f" (R5) values (each register field MUST be 8 ASCII HEX digits in length. Add leading '0' digits if needed). Data is represented as ASCII hex ('0' thru '9', 'A' thru 'F', or 'a' thru 'f'). Space, comma, or TAB (ASCII 0x09) characters may be included between channel data characters for clarity, but the serial buffer limit is 62 characters per command line, including the command and channel# characters. If any invalid data is received, the command is aborted (with error message) and the channel is left un-programmed.

Note: The "M" command processes one channel at a time. When uploading a file of programming commands, command lines that feature errors will not affect other command lines that are error-free.

Note: Unlike the Orion PLL software, CH00 on the beacon is available for use as a transmit channel.

zc: Compare CRC – PLL Channels
zm: Compare CRC – Morse Message
zc hhhh
zm hhhh

Compares the FLASH CRC for either the PLL channels or Morse message to the "hhhh" value provided in ASCII hex (the CRC must be 4 digits). Displays PASS if there is a match or FAIL otherwise. If FAIL, the global error flag is also set. *Note: A one (1) second delay is provided to allow terminal upload to finish. As such, this should be the last command in an upload file if the CRC is known beforehand. If a 1 second delay is not sufficient, the "Q" command may be used to manually interrogate the pass/fail status of the upload.*

cc: Display Channel Data CRC
cm: Display Morse Mesage CRC
`cc`
`cm`

Calculates a CRC-16 value (using a polynomial seed of 0x1021) based on the 384 bytes in the channel array stored in FLASH memory for the PLL channel data ("cc") or the Morse message data ("cm") and displays the result to the serial port.

The MCU calculates the displayed CRC is by sequentially calling a function called calcrc() (the code snippet below may be used to create an off-line CRC calculator if desired) with each byte in the array (for the channel data, 384 bytes), starting with CH00 R0 MS nybble (lowest address). "old_crc" = 0x0000 at the start of the calculation. *Note: The CRC16 result for all channels erased is 0x955C, and the erased message space CRC16 is 0x11AA.*

```
//---------------------------------------------------------------------
// calcrc() calculates incremental crcsum using defined poly
//       (xmodem poly = 0x1021)
//---------------------------------------------------------------------
U16 calcrc(U8 c, U16 oldcrc){
#define  POLY 0x1021      // xmodem polynomial

        U16 crc;
        U8      i;

        crc = oldcrc ^ ((U16)c << 8);
        for (i = 0; i < 8; ++i){
                if (crc & 0x8000) crc = (crc << 1) ^ POLY;
                else crc = crc << 1;
         }
         return crc;
}
```

r: Read Channel Registers
`rxx`

Displays PLL channel "xx" (xx is BCD ASCII '00' thru '15') in the same format as is used by the "M" command. Spaces are inserted between register fields for clarity.

r-: Read All Channel Registers
`r-`

Sequentially displays all PLL channels starting at CH00. The resulting output is in the "M" command syntax format and may be used to archive or clone an entire channel set for a given Orion board.

C: Program Morse Message
`Ciiiiaabbccdd...`

"iiii" = the message array index for the first byte of data, and "aa", "bb", (etc…) are the ASCII hex character pairs that represent the array bytes. *Note: The index, "iiii" is in BYTES, not characters. e.g., 6 bytes would actually be represented by 12 characters in the command line.* Due to the limits of the serial buffer, the program message command

should feature less than 62 characters. Subsequent data is transferred using multiple "C" commands. The Morse message consists of an internal array that holds 6 bytes of message configuration followed by up to 800 bytes of bit-mapped key data (each byte is transmitted MSb first). Before sending a new message, the existing message space must be erased using the EM command. A 0x18FF word terminates the Morse message. The format of the message is as follows:

0) KEY_OUT Polarity: 8-bit integer, "01" = logic high keys, "00" = logic low keys.
1) RAMP_DELAY: 8-bit integer that specifies the waveshape ramp in (ms)
2) DIT_TIME: 16-bit integer that specifies the duration of a DIT key-down (ms)
5) REPEAT_DELAY: 16-bit integer that specifies the time lapse between the end of the message and the re-start of the message (ms)
6) First message byte. Subsequent bytes are stored in successive address locations until a 0x18 0xFF byte pair is encountered (end of message)

0x18 is the embedded command semaphore. The byte that follows is the command/operand (high nybble/low nybble). The 0xFF operand specifies the end of message marker. The following commands are available:

0x18 0xFF	Specifies end of message
0x18 0xE0	Increment the GP output port bits (FSEL[6:4])
0x18 0xD0	Decrement the GP output port bits
0x18 0x0x	Set GP out port bits to "x" (x = 0 to 7)
0x18 0xCx	Select PLL channel "x" (x = A – P selects channels 00 – 15

The 0x18 semaphore MUST be aligned to a byte boundary, which means that extra message space (up to 7 elements) must be inserted to force the command to lie on a byte boundary. The message spreadsheet accounts for this, but users need to be aware of this limitation so that the semaphore may be inserted at a place in the message where the extra delay will be less noticeable.

Users who implement their own message pre-processing system must ensure that the 0x18 semaphore can not be inadvertently inserted into the message or unexpected results may be encountered. 0x18 is not a valid Morse element, so any message created to follow normal Morse timings would not result in this semaphore.

The primary purpose of the GP out and channel select commands is to allow for the variation of the RF output power during the message. The increment/decrement commands could feed a simple, 3-bit, D/A converter to allow for 4 or 8 power levels. Alternately, each bit may be manipulated with the Set GP out command to achieve 3 or 4 discrete power levels. The PLL channel data can set the RF output in roughly 3 dB steps over a 9 dB range which allows one to provide up to 4 discrete power levels using different PLL channels tuned to the same frequency (*Note: the behavior of the PLL power register setting tends to diminish with increasing frequency. For example, at 3.45 GHz, only about 5 dB of variation is feasible using the PLL registers*).

The GP out and channel select commands can also be used to create a multi-band beacon. For this implementation, the bands would be time-domain multiplexed, with a message transmitting on one band at a time, with band changes at the end of the Morse message.

While up to 16 different frequencies can be theoretically implemented, a realistic limit is 2 or 3. Still, this allows a single exciter to be used to drive multiple beacons.

R: Read Morse Message
`R`

Downloads the message in the same format as the Program Morse Message command. Useful for checking and cloning a message.

i: Re-initialize Beacon software
`i`

Causes Orion to restart the PLL system and re-initialize the Morse message and re-read the channel selection from FSEL[3:0]. Once the Morse message is erased, or if it is invalid upon power-on, the beacon software will not send any message information. After the new message is uploaded, the "i" command is used to re-initialize the system and start the message. A power-cycle will also accomplish the same effect.

e: Echo Command-Line (debug)
`e`

Echo command line. This is a debug command that will echo the characters on the command line and is useful in verifying UART operation.

Q: Query Error Status
`Q`

Displays FAIL if there was an error encountered in an "M" command. Otherwise, PASS is displayed.

QC: Clear Query Error Status
`QC`

Displays the current error status, then clears the status to PASS.

PLL Channel Data Configuration

The Orion Manual discusses the particulars of creating the data for a PLL channel. However, there are some details that are special to the beacon application that are presented here. It is recommended that the user utilize the ADF4351 Program published by Analog Devices to create the six registers needed for a PLL channel. Figure 2 illustrates the main screen of this program.

The default settings for all of the parameters should be utilized except for those parameters marked in the figure. The relatively obvious operating frequency, channel step, and reference frequency are at the left side of this figure. Less obvious, are the parameters circled on the right side. These should be set as shown in the figure (MTLD "Enabled", and RF Output power should be set as needed for the application).

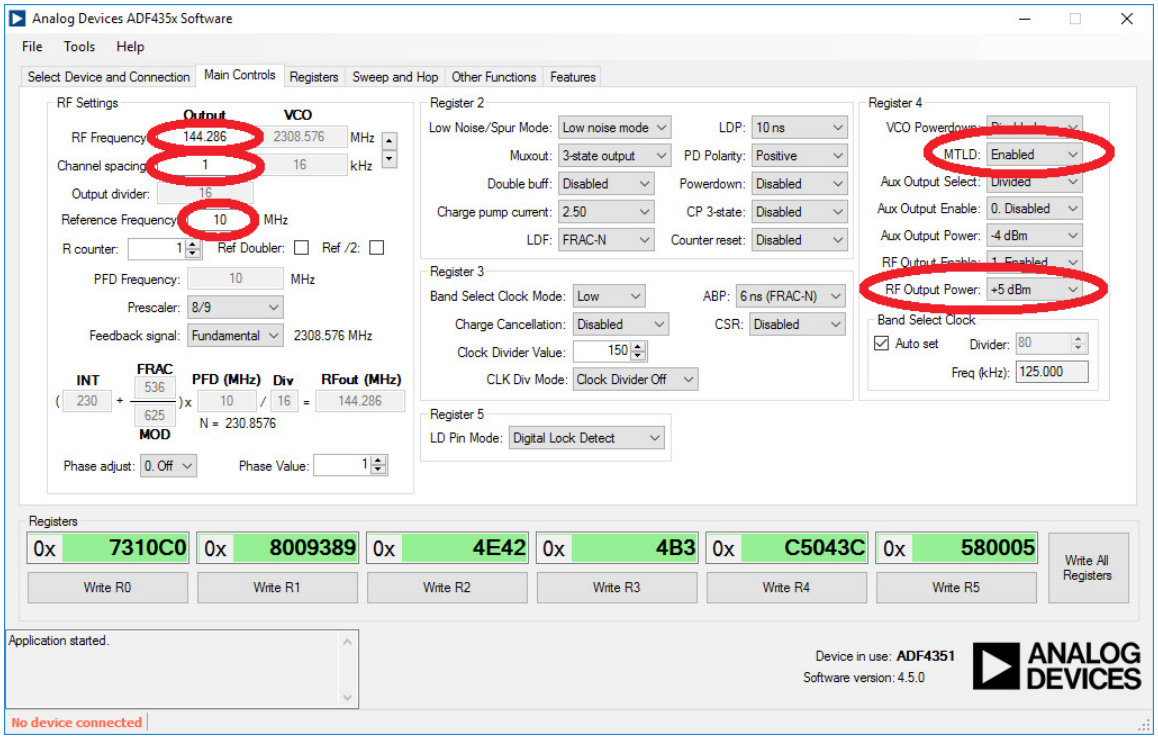

Figure 2. Analog Devices ADF4531 Register Calculator Program

The user should not adjust any of the other parameters unless they are well schooled in their use and application.

Note that the beacon software uses the VCO Powerdown bit of register 4 to turn the RF output on and off during Morse on-off keying (OOK). This bit should be cleared in the channel data to allow proper operation of this keying scheme.

Constructing an Upload File using the Orion Beacon Excel Spreadsheet

A spreadsheet is available for Microsoft Excel (2013 or later) that allows the user to enter message text and configuration data which is converted to the bit-mapped message format which can be saved as a text file of programming commands. This file can then be sent to the Orion MCU. This spreadsheet is available at:

www.rollanet.org/~joeh/projects/Orion/

under the "Beacon Message Programming Spreadsheet" link. Messages can contain approximately 4800 DIT periods (about 13 minutes of message elements at 15 WPM). A DIT occupies two (2) DIT-length elements (bits), while a DAH occupies four (4) DIT elements. The spreadsheet features several special character "prosigns" that can be invoked. These are detailed in the spreadsheet text.

Once the message file is produced, it can be uploaded to the Orion. *Note: The Orion software does not check for erased locations before programming. Thus, attempts to write data into a message that is not erased will produce unpredictable results.*

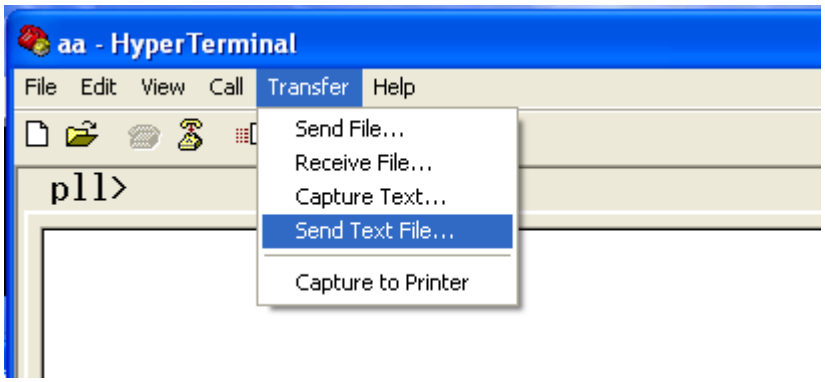

Figure 4. Hyperterm text file transfer selection

To write the file to memory, first connect the PC COM port (9600, N81) and apply power to the Orion board. After the power on message is observed, type "EM" and then "Enter" (CR). Type "Y" at the "`Erase all, press Y to accept…`" prompt. Next, issue the "QC" command to clear the error status. To effectively utilize a text upload, the terminal must be configured to produce a delay after each line has been sent. At least 50 ms is recommended. Once the line-delay is configured, perform a text file upload (for Hyperterm, see Figure 4).

Some terminal programs will display the terminal progress as the file uploads (Hyperterm does this). If so, the "`Chan pgmd!`" message should be observed after each "C" line is transferred.

Once the file transfer is complete, one should perform a "Q" command to verify that there were no errors encountered in the downloaded data. The CRC16 check ("cm" or

"zm" command) should also be performed. Note: The "Orion Beacon Message Workbook" spreadsheet will include a "zm" command with the pre-calculated CRC for the data in that file. This will display a "PASS" or "FAIL" message 1 second after the transfer finishes. Once it is verified that the data transferred without error, issue the "i" command (or cycle power) to re-start the PLL and Morse message.

UART Connections

The UART RXD and TXD connections are available at P2-3 and P2-5, respectively. To exercise this option, wire a DSUB-9, female connector using the table below:

RS-232 Pinout (D-SUB9 Female)

Shield GND (opt)	1	6	n/c
RS-232 TXD (P2-5)	2	7	RTS ---\ (optional: connect RTS and CTS)
RS-232 RXD (P2-3)	3	8	CTS ---/
n/c	4	9	n/c
GND (P2-6)	5		

For most PC serial connections, RTC/CTS are left disconnected. However, some terminal connections may require these signals. In this case, simply connect the D-SUB9 pin 7 to pin 8. Cable lengths should be no longer than needed to be accessible for connection to a standard RS-232 cable.

Appendix A

Installation Notes

See the Orion Operating Manual for details on mounting the Orion-I PLL board.

Morse Code Wave Shaping and RF Filtering

The Orion beacon can operate as a Morse code modulator with no additional circuits. However, the lack of wave-shaping will produce key-clicks in the adjacent spectrum. To correct this, a wave-shaping circuit which utilizes the KEY_OUT signal (P2-15) must be constructed and connected to the Orion RF output. The KEY_OUT signal activates 1ms before the RF output from the Orion is activated. However, it deactivates a few ms before the RF output is deactivated (this time delay is programmable in the CW message fields). This provides the correct timing to provide a trigger for external ramp circuits for an amplifier or voltage-controlled attenuator.

There are two relatively simple ways to produce Morse wave shaping. One is to ramp the power to a PA stage and the other is to use a PIN diode to act as a voltage-variable attenuator placed in series with the RF output (either at the Orion RF output, or at the output of one of the subsequent buffer or PA stages). This section will briefly present these two options. The exact circuit that might be implemented will depend on the design of the output stages.

Amplifier Power Ramp

The power-ramp option can use a P-CH FET to apply power to an RF amplifier stage. The gate of the FET is controlled with an RC circuit such that there is a short ramp time (5ms, typical) to the FET output. Another option is to use an op-amp and an emitter follower to produce the power ramp. This is the method discussed herein and is shown in Figure A1.

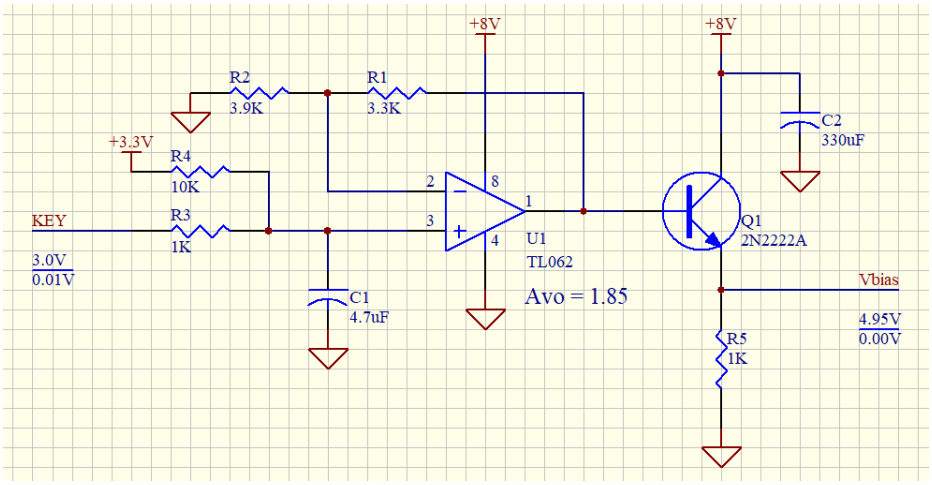

Figure A1. Amplifier power ramp wave-shaping circuit

KEY is active high (approx 3.0V when RF is on). The polarity of the key output is controlled by a field in the CW message). Setting the ramp delay to 0x04 in the Morse message is recommended for best results with the above circuit. R3 and C1 form an RC network with a risetime of approximately 10 ms. This is fed to the op-

amp configured as a non-inverting circuit with a gain of 1.85. R4 is needed to prevent 0V from being applied to the input of an non-rail-to-rail device. The resulting voltage varies from about 0.6V up to about 5.7Vat the base of Q1. After the inherent Vbe diode drop of Q1, Vbias will see voltages from about 0V up to about 5V with about 4ms of rise and fall time with a current capacity of about 100mA. This is sufficient to power the GVA-62 buffer at U7 of the Orion board.

Vbias is applied to the Orion buffer amplifier (U7) by either cutting the 5V power trace that feeds FB1 (the best location for this cut is neat U3) or by removing FB1 and replacing it with a small leaded inductor (1uH range). A wire is run from the ramp circuit to the cut trace or leaded inductor.

The prototype area of the Orion makes an excellent location for the above circuit. A source of 8V power can be achieved by installing an LM317 regulator at U14 (this is described in the Orion Operation Manual). If this circuit is used to supply an external driver or PA, care should be exercised in the selection of Q1 to meet the current requirements of the amplifier (a heat sink is likely required for currents much above 100mA) and also make sure that the transistor has a high enough value for beta so that the opamp is not loaded too heavily.

PIN Attenuator Ramp

The PIN diode attenuator uses a similar scheme except that the ramped voltage is used to control the impedance of the PIN diode to affect a "soft-switch" of the RF. Figure A2 illustrates an example of this type of wave-shaping circuit

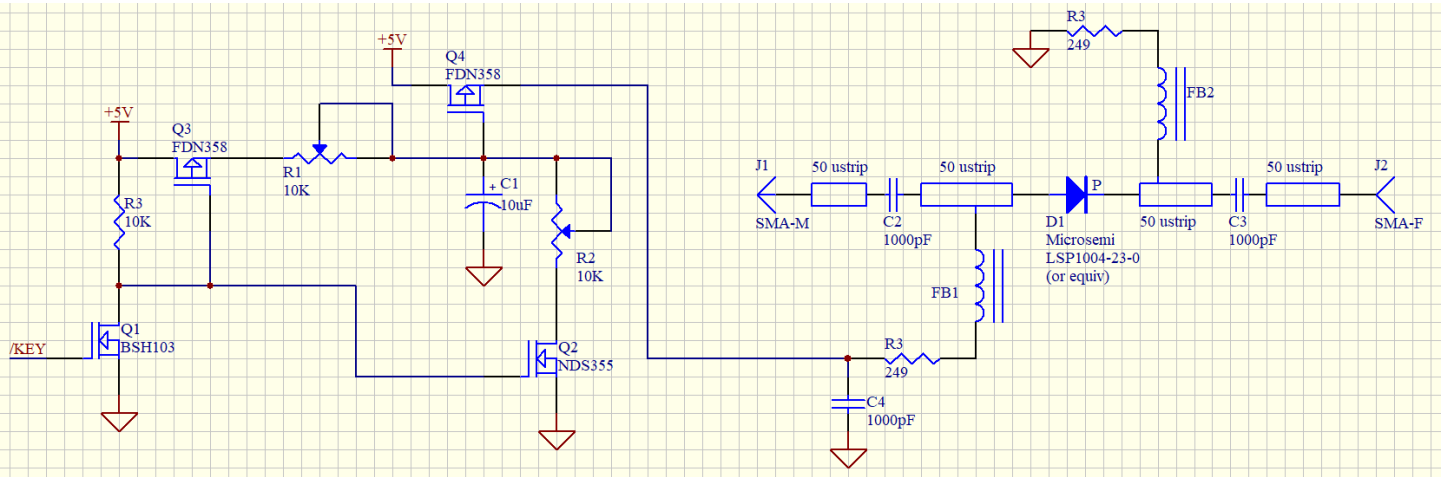

Figure A1. PIN diode (attenuator) wave-shaping circuit

/KEY is active low (0V when RF is on. The polarity of the key output is controlled by a field in the CW message). R2 adjusts the rise-time of the RF, while R1 adjusts the fall-time. POT values of 10K and C1 = 10uF give a maximum RC time constant on the order of 100ms. The actual transition time is affected by the Vgs curve of Q4. The idea is to operate Q4 in a linear mode during the transitions, and have it be saturated while RF is on, and cut-off when RF is off.

Adjustment is best accomplished on a spectrum analyzer set to the operating frequency and zero-span. In this configuration, the display of the analyzer will resemble that of an oscilloscope, where the vertical position of the trace represents the RF amplitude, and the horizontal position represents time. A test message of all dits (alternating "10" bits in the message space), with no character or word spaces is also helpful when making this adjustment.

The RAMP_DELAY field of the message must also be adjusted. Unfortunately, there is not an easy adjustment method for this value. When the circuit of Figure A1 sees a /KEY transition from "0" to "1", R1 will start to charge C2. However, since Vgs(th) is generally rather low for most FETS, there won't be much change to Q4 Vs until the voltage at C1 rises to about Q4Vgs(th). It is at this point that the shaping effects begin. A best guess at a rise-time value is about 3RC, once R1 is adjusted to give the desired shape. If the RAMP_DELAY is too large, the cadence of the Morse elements will be affected. If it is too small, the RF will drop too quickly, regardless of the setting of R1. Ultimately, cut-and-try is the best way to get an accurate value. Start with a small (5 ms) value and gradually increase just until the slope of the RF drops uniformly to at about 20 dBc (anything more that that isn't likely to affect the bandwidth of the CW signal) or starts to level-off.

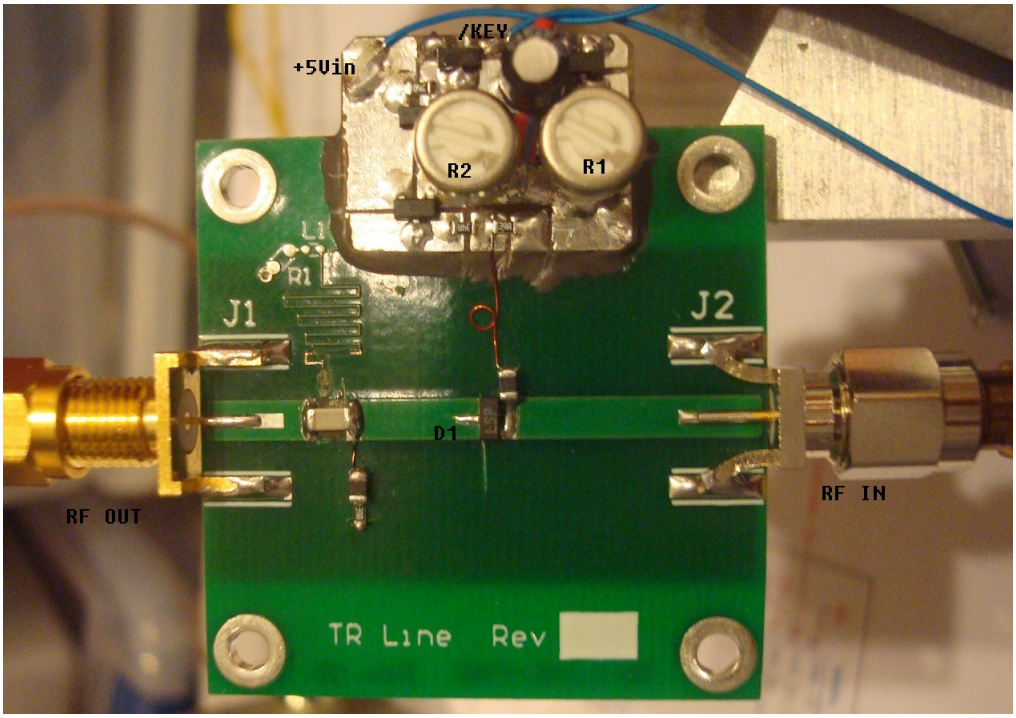

Figure A2. PIN diode (attenuator) wave-shaping circuit implementation

While the attenuator circuit can provide a good result, it does have some issues. First, the impedance of the RF path is disturbed during RF transitions. This can cause stability problems with the Orion buffer and/or a following buffer or PA stage. Secondly, the performance of the PIN diode will limit the upper frequency at which this is effective. The illustrated circuit performs well at 144 MHz, but is not usable at 3456 MHz (circuit layout likely has something to do with this as well).

Output Filtering

Output filtering is discussed in the Orion Operation Manual, but it is mentioned here as a reference. At least some degree of harmonic filtering should be implemented to purify the output of the Orion before it is applied to subsequent amplification stages. Ultimately, the output of the antenna is the driving motivator, but many antennae have a significant harmonic response, so it is generally considered good engineering practice to ensure that the output of the final PA is below the FCC limits for the band in question.