

CALIFORNIA STATE UNIVERSITY  
LOS ANGELES

Department of Electrical and Computer Engineering

EE-2449 Digital Logic Lab

EXPERIMENT 2

**NUMBER SYSTEMS AND WAVEFORMS**

Text: Mano and Ciletti, *Digital Design, 5<sup>th</sup> Edition*, Sec. 1-1 through 1-4 (for additional information on counters, refer to Sec. 6-3).

Required chips: **7493** ripple counter.

Oscilloscope: For information on the oscilloscopes in the lab, refer to "USING THE AGILENT DSO1012A OSCILLOSCOPE" at the end of this experiment. (There is also a manual, "Oscilloscope DSO1012A", on the lab support website, <http://www.calstatela.edu/ecst/ece/labs>, however, you probably won't need it.)

**2.1** Here is a quick review of number systems:

- Humans use decimal which is base 10 because we have 10 fingers.
- Digital logic systems, such as computers and microcontrollers, use binary which is base 2 because they are built using transistors. We saw in Experiment 1 that a transistor is an electrical switch that is on or off and thus has 2 states. A binary digit is known as a "bit" and 8 bits are called a "byte".
- Hexadecimal (base 16) and Octal (base 8) are shorthand for binary. Both are powers of two; i.e.  $16 = 2^4$  and  $8 = 2^3$ . Digits in a binary number (bits) can be grouped together to convert to either hexadecimal or octal. Groupings must start at the least significant bit of the number. To convert to hexadecimal, group by 4 bits and convert each grouping into the corresponding hex digit (where 10 through 15 are represented using letters A through F). To convert to octal, group by 3 bits and convert each grouping into the corresponding octal digit.

Binary (Base 2) Number Example:

$$(10100101)_2 = (1)*2^7 + (0)*2^6 + (1)*2^5 + (0)*2^4 + (0)*2^3 + (1)*2^2 + (0)*2^1 + (1)*2^0$$

Binary to Decimal (Base 10) Conversion Example:

$$(10100101)_2 \rightarrow (1\ 0\ 1\ 0\ 0\ 1\ 0\ 1)_2 \rightarrow \begin{matrix} 128 & 64 & 32 & 16 & 8 & 4 & 2 & 1 \\ (1) & (0) & (1) & (0) & (0) & (1) & (0) & (1) \end{matrix} \rightarrow 128 + 32 + 4 + 1 = 165_{10} = (1)*10^2 + (6)*10^1 + (5)*10^0$$

Binary to Hexadecimal (Base 16) Conversion Example (A=10, B=11, C=12, D=13, E=14, F=15):

$$(10100101)_2 \rightarrow (1010\ 0101)_2 \rightarrow \begin{matrix} 8 & 4 & 2 & 1 & 8 & 4 & 2 & 1 \\ (1) & (0) & (1) & (0) & (0) & (1) & (0) & (1) \end{matrix} \rightarrow A5_{16} = (10)*16^1 + (5)*16^0$$

Binary to Octal (Base 8) Conversion Example:

$$(10100101)_2 \xrightarrow{\begin{matrix} 421 & 421 & 421 \\ \underline{010} & \underline{100} & \underline{101} \end{matrix}} (010\ 100\ 101)_2 \rightarrow 245_8 = (2)*8^2 + (4)*8^1 + (5)*8^0$$

Complete the math in the above examples to verify that  $(10100101)_2$ ,  $165_{10}$ ,  $A5_{16}$ , and  $245_8$  are the same number just represented in a different base.

Question: Which base, hex or octal, is more commonly used today as a shorthand representation for binary numbers and why? To answer, refer to Table 2.1 and consider the following: 1) which base, hex or octal, will require fewer digits to represent a binary number? And 2) given that a byte is comprised of 8 bits, which base, hex or octal, can be more compactly used to represent an 8-bit (1-byte) binary number? Assume a byte value of 11111111 and convert to octal and hex. Write your answers, including explanations, in your lab journal.

| Decimal | Binary | Octal | Hex | Decimal | Binary  | Octal | Hex |
|---------|--------|-------|-----|---------|---------|-------|-----|
| 00      | 0000   | 00    | 00  | 56      | 111000  | 70    | 38  |
| 01      | 0001   | 01    | 01  | 57      | 111001  | 71    | 39  |
| 02      | 0010   | 02    | 02  | 58      | 111010  | 72    | 3A  |
| 03      | 0011   | 03    | 03  | 59      | 111011  | 73    | 3B  |
| 04      | 0100   | 04    | 04  | 60      | 111100  | 74    | 3C  |
| 05      | 0101   | 05    | 05  | 61      | 111101  | 75    | 3D  |
| 06      | 0110   | 06    | 06  | 62      | 111110  | 76    | 3E  |
| 07      | 0111   | 07    | 07  | 63      | 111111  | 77    | 3F  |
| 08      | 1000   | 10    | 08  | 64      | 1000000 | 100   | 40  |
| 09      | 1001   | 11    | 09  | 65      | 1000001 | 101   | 41  |
| 10      | 1010   | 12    | 0A  | 66      | 1000010 | 102   | 42  |
| 11      | 1011   | 13    | 0B  | 67      | 1000011 | 103   | 43  |
| 12      | 1100   | 14    | 0C  | 68      | 1000100 | 104   | 44  |
| 13      | 1101   | 15    | 0D  | 69      | 1000101 | 105   | 45  |
| 14      | 1110   | 16    | 0E  | 70      | 1000110 | 106   | 46  |
| 15      | 1111   | 17    | 0F  | 71      | 1000111 | 107   | 47  |

Table 2.1. Numbers 0 to 31 and 56 to 71 in decimal (base 10), binary (base 2), octal (base 8), and hexadecimal (base 16).

**2.2\*** (\* = lab work) Refer to Figure 2.1 below. The 7493 chip is a 4-bit "modulo-16" counter. "Modulo N" implies that the highest digit is N-1. In this case, N-1 = 15, so the 7493 counts from 0 to 15 (F in hex) and returns to 0 on the 16th count. The counter outputs QD, QC, QB, QA are displayed on the LED bargraph and the Ro(1) reset pin (pin 2) is grounded to allow the 7493 to count.

*(Note, that Ro(1) and Ro(2) reset inputs are connected internally such that if both of them are high, the counter is forced into a reset condition with count frozen at 0000. So it's necessary to ground either or both reset inputs (pins 2 and 3) to enable counting to proceed.)*

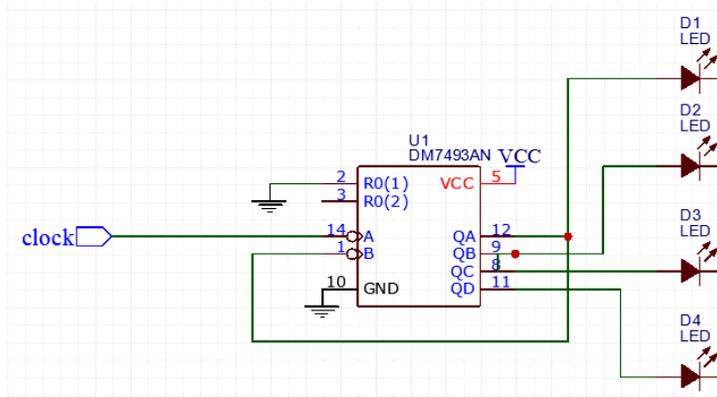
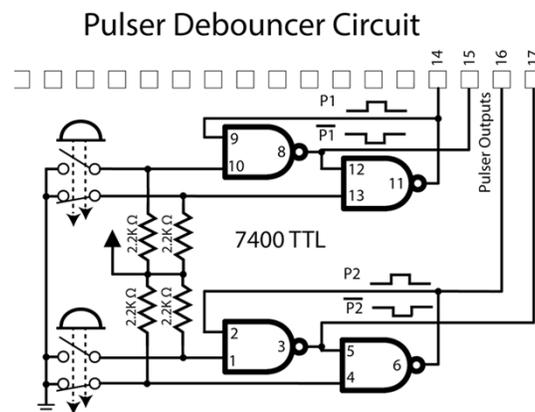


Figure 2.1. Wiring diagram for a Mod-16 counter.

Draw this circuit in your lab journal, making sure to include pin numbers. Then wire up the circuit on your breadboard as shown in Figure 2.2. For the clock, connect a pulser output (push button switch) to the counter's clock input at pin 14 (see the dark green wire in Fig. 2.2).

Note: you don't connect directly to a pulser--its output needs first to be "debounced". So it is the debounced output you will connecting to. In the following diagram, there are two pulsers at the left side. If you use the upper one, its "debounced" output is available at row 14 or 15; if you use the lower one, its debounced output is at row 16 or 17. (Debouncing is done by the four NAND gates to produce a clean pulse. Refer to pg.7 of the Intro section of the manual to learn why debouncing is needed and how it works.)



As you make connections, it is a good idea to put a check mark on your diagram next to each wire you add in order to know what part of the circuit you have completed and to ensure that you do not forget any wires.

Figure 2.2 illustrates how to place the 7493 chip on the board. It should straddle the center line of the breadboard section. The five holes along each horizontal row are connected together below the breadboard surface. A chip's pin will occupy one of the holes, so you can connect up to four wires to the same pin on that row (or connect two wires together if there is not a pin from a chip at that row). Notice the wiring in this circuit. Where possible the wires lay flat close to the board and do not cross one another. Avoid putting wires over the chip as it will be difficult to remove a bad chip without having to rewire the circuit. Taking the time to wire circuits neatly will save you a lot of time in troubleshooting your circuits (finding mistakes also known as "bugs").

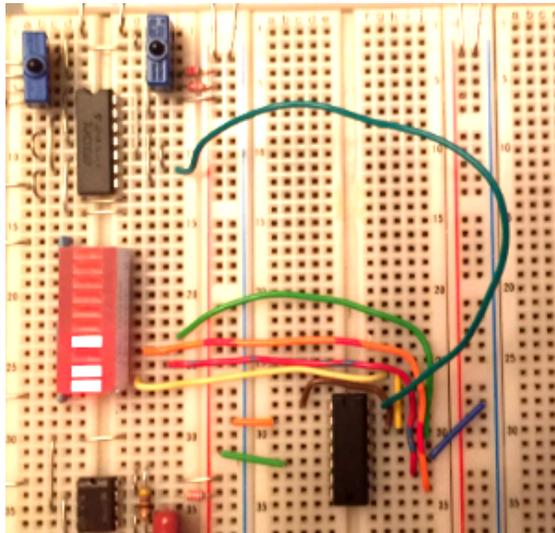


Figure 2.2. 7493 circuit with clock input connected to the normally-high pulser output and the counter's four outputs connected to the LED bargraph. Count showing on LED bargraph is  $1011_2$  (i.e.  $11_{10}$  or  $B_{16}$ ).

Pulse the clock repeatedly and verify that the counter cycles through the count 0000 to 1111. (Depending on the state of the counter when the chip powers up, the starting value will likely not be 0000, but eventually it will count up to 1111 and then roll over to 0000 so you can observe a complete count cycle.)

Now connect the timer output to the counter's clock input at pin 14 (as shown in Figure 2.3). Run the clock at its slowest rate so you can see the LEDs flash on and off. (For the slowest rate, the capacitor select switch--lower left--should be flipped opposite to the way it is shown.) Again, the LEDs should go through a full 16-count sequence:  $0000 \rightarrow 0001 \rightarrow 0010 \rightarrow \dots \rightarrow 1110 \rightarrow 1111 \rightarrow 0000 \rightarrow \dots$  etc..

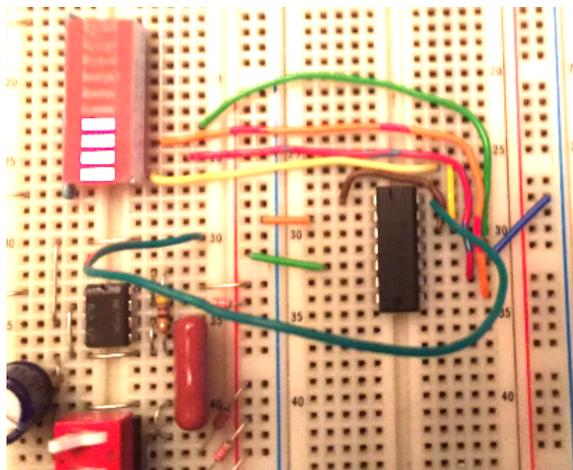


Figure 2.3. Counter circuit with clock input now connected to timer (row 30 on the breadboard).

(Note: if you haven't wired a circuit before, the photos in Figures 2.2 and 2.3 are provided to help. But, while you are connecting the wires to the chip, try as well to follow the wiring diagram in Figure 2.1 since for the rest of the experiments you will only be given, or will design your own, wiring diagrams.)

---

**2.3\*** Once you are satisfied that the counter is running correctly, switch the timer into its high range (flip the capacitor select switch back to the way it appears in Figure 2.3). The counter outputs should now be changing too fast to see their bit patterns on the LEDs which, as a result, appear to be on all the time. Instead, you will observe counter outputs using an oscilloscope. (See “Controlling the Oscilloscope Display” at the end of this experiment for information on using the Agilent DS01012A oscilloscope.)

Figure 2.4 shows the timing relationships between the counter outputs. The count values are shown at the bottom of the figure; each one represents the binary pattern in the corresponding vertical slice read from the bottom up; i.e. from QD up to QA. For example, count 7 corresponds to QD, QC, QB, QA = 0 1 1 1.

In this section, you will display the waveforms of Figure 2.4 two at a time on an oscilloscope.

- Bring QD to channel 1 and QC to channel 2.
- Set vertical sensitivity (Volts/Div) for QD and QC to 2 V/Div. Each waveform should swing approximately between 0 and 4 volts.
- Set up the scope to trigger on the *negative edge* (or slope) of channel 1 (i.e. QD).
- Adjust the scope-trace speed (Time/Div) so you see QD and QC displayed as in Figure 2.4 (i.e. a full cycle of QD and two full cycles of QC).
- Position the display with the Horizontal control so the point where QD goes low is on a vertical grid line near the left side of the screen as shown.
- Adjust Vertical position control for QD and QC so their 0 V indicators lie on horizontal grid lines.

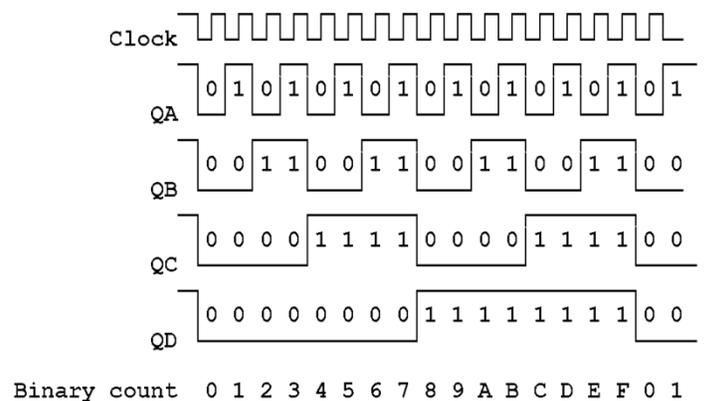


Figure 2.4. Expected waveforms for a mod-16 counter.

**Important:** in order to make it easy to read the binary count at any point in the display, do the following: adjust the scope's Time/Div setting until 16 counts--the time for one cycle of QD-- take up 8 divisions on the screen. Then each division will represent 2 counts. Thus QA will toggle each half-division, QB each division, QC each two division, and QD each four.

Verify that the timing relationship between QD and QC is as in Figure 2.4.

Repeat with QD and QB on the screen, then with QD and QA, and finally with QD and the clock. Don't change any scope setting while doing this--**QD should appear the same each time**. Show each display to your instructor (once you have verified that it works properly, you should be able to demonstrate it to the instructor quickly by moving the wire connected to channel 2 from QC to QB to QA).

---

**2.4** As discussed in the previous sections, the counter you are using is a "modulo-16" (mod-16 or hexadecimal) counter since it counts from 0 to 15 (F in hex) and returns naturally to 0 on the 16th count. In general, a counter that counts up to a maximum value of  $N-1$  and resets to 0 on the  $N$ th count, can be called a modulo- $N$  (or "mod- $N$ ") counter. Now, consider converting a 4-bit mod-16 counter into a mod- $N$  counter with  $N < 16$ . Such a counter needs to reset to 0 on a count lower than 16, but it will not do that unless it is forced to reset prematurely. For example, a mod-14 counter ( $N = 14$ ) should count up to a maximum of only 13 and then reset to 0 on the 14th count. The counter's reset inputs,  $Ro(1)$  and  $Ro(2)$ , can be used to force this to happen.

With the asynchronous 7493, the count must enter state  $N$  for just a few nanoseconds in order to be cleared. (As we'll see in later experiments, with a synchronous counter, the count never enters state  $N$ , but goes directly from state  $N-1$  to 0000.) In state  $N$ , certain  $Q$  outputs will be 1's. You can combine these in such a way that they make both reset inputs  $Ro(1)$  and  $Ro(2)$  go high and thereby trigger an immediate reset just after state  $N$  is entered.

For  $N=10_{10}$  (a decimal counter), simply connect  $QD$  to  $Ro(1)$  and  $QB$  to  $Ro(2)$ . In state  $10_{10}$  ( $1010_2$ ), these both go high. The internal logic responds by clearing the count ( $1010 \rightarrow 0000$ ). You don't see it enter state 10 since reset is almost instantaneous. Thus, the counter will seem to count from 0 to 9 (10 states) and then start counting up from 0 again.

**2.5\*** Design a mod- $N$  counter. Your instructor should give you a value for  $N$  from the set **5, 6, 8, 9, 12**. If not, you should choose your own value from this set. Draw the wiring diagram for your circuit in your lab manual (similar to Figure 2.1).

N =

As you connect your circuit, *be sure to remove any grounding wires you connected to  $Ro(1)$  and/or  $Ro(2)$  in section 2.1 above. If you don't, your circuit will never reset to 0 when the count reaches  $N$ . Instead, it will continue past  $N$  and reset as usual on the 16th count.*

Test your design with a pulser or slow clock and observe the count on the LED bar graph.

Next, bring the counter outputs to the scope as in Section 2.3. Use a fast clock and again **trigger on the falling edge of  $QD$** . This time, record  $QD-QC$ ,  $QD-QB$ , and  $QD-QA$  from the scope display.

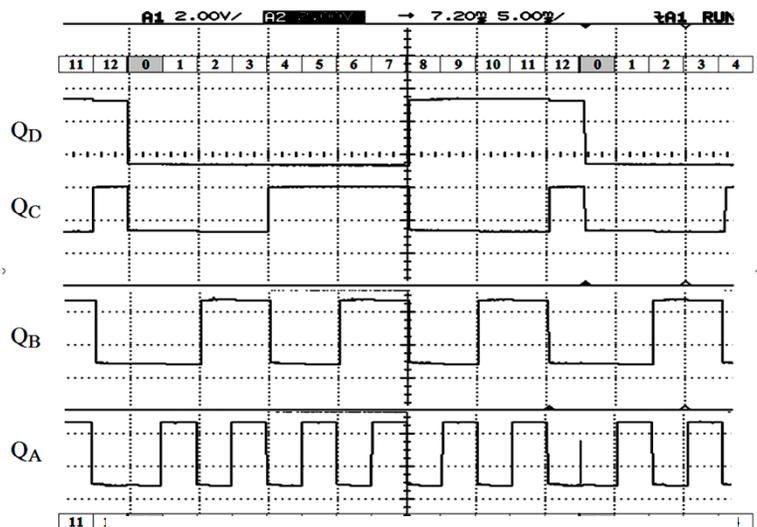


Figure 2.5. Expected waveforms for a mod-13 counter.

The diagram shown in Figure 2.5 is for  $N = 13$ . The display starts at the left shortly before QD goes low to trigger a new cycle of 13. The count proceeds from 0000 to 1100 (12) for a total of 13 full count periods. It then enters 1101 (13) just long enough to reset itself to 0000 again. In binary, the sequence 12-13-0 looks like  $1100 > 1101 > 0000$ . Notice that QA (lsb) goes to 1 as the count reaches 13. You can see this in Figure 2.5 at the end of count 12. QA shows a short spike as it brings the count to 13. It rises and then immediately falls, having reset all the Q's, including itself, to 0. This spike is much too narrow to be seen with the scope settings you will be using since it only lasts about 50 nanoseconds. But it is there (it must be, or what else would reset the counter?).

After you have designed and tested your Mod-N counter, make sure that the following are recorded in your lab journal:

- clearly **describe the connections** you made to reset the counter
- show the wiring diagram with the correct connections
- draw the waveforms you observe on the oscilloscope. As done in Figures 2.4 and 2.5, stack the waveforms for QA, QB, QC, and QD (even though you can only observe two at a time).
- Answer the following question: for  $N = 13$ , the count sequence  $1100 > 1101 > 0000$  shows why there is a spike in QA in Figure 2.5. For your value of  $N$ , take the same approach to show which Q should spike during the reset sequence.

### Going more in depth – The 7493 Ripple Counter (Optional Reading):

The 7493 consists of 4 flip-flops with J-K inputs unconnected. In a TTL chip, unconnected inputs behave as logic-1, so  $J=K=1$ . As a result, each flip-flop is configured as a "T" and will toggle every time it is clocked; i.e. when its clock voltage falls (the 7493 has a negative edge triggered clock input). Refer to the diagrams of the 7493 on the first and last pages of the last section of the manual, "Description of IC'S".

QB, QC, QD are chained together to form a 3-bit counter; the output of each is connected internally to the clock of the next (QB goes to QC's clock, etc.). For a 4-bit counter, QA must likewise go to QB's clock but there is no such connection inside the chip. Therefore the connection must be made *externally* (pin 12 to pin 1). In the resulting counter, **QD is the most significant bit (msb)**; i.e. the 4-bit counter output is [QD QC QB QA]. Clock pulses are input at pin 14.

The reset inputs Ro(1) and Ro(2) of the counter are connected internally to an AND gate. When both reset inputs are high, the AND's output goes high and resets the counter. Resetting is asynchronous; it does not wait for the clock and occurs almost immediately. (Note: if the reset inputs are left unconnected, the chip treats them as if they are high so the AND's output resets the counter to 0000 continuously. Therefore you must ground at least one of them if the chip is to count.)

## USING THE AGILENT DSO1012A OSCILLOSCOPE

Here are some general comments . The detailed instructions follow.

A) Triggering: you always want to trigger on the waveform of the most significant bit of the counter outputs. Let's call these XYZ. Then bring X to a scope input channel, say CH1, and select that as the source channel for the trigger. (The other variable will go to CH2.)

You also want to trigger on the *falling edge* of X since when  $X \rightarrow 0$ , it marks the start of a new period for both waveforms.

Finally, the trigger level needs to be somewhere in the middle of X's waveform. Either adjust the trigger-level line up or down with the Level knob or just push the knob to set the line halfway up X's waveform.

B) Preparing the screen image for transfer to a Word document: first press the Display button. Your options appear at the right of the screen.

- Increase the waveforms' intensity to 100% to make them more visible.
- Increase grid brightness to 100% for the same reason.
- Invert the screen colors so the waveforms and grid divisions appear dark against a white background.

C) To save screen image to your flash-drive, see instructions below.

---

### To set up triggering:

1. Press **Menu** button below Trigger Level knob.
2. In the menu at the right of the screen, press **Mode**.
3. Select **Edge** by repeatedly pressing **Mode** or by turning the "entry" knob. ↻
4. Next press **Source** and choose the channel you want to do the triggering, **CH1** or **CH2**.
5. Finally, press **Slope** and choose **Falling Edge**.

### To adjust the triggering level:

1. Turn the **Trigger Level** knob so that the horizontal line lies somewhere within the triggering waveform. Or just push the knob to set the level in the middle of that waveform.

### To set up the screen image:

1. Press **Display** .
2. In the menu at the right of the screen, press **Intensity** and turn the entry knob ↻ until 100%.
3. Then, go down the menu to **Gridbright** and, again, turn the entry knob until 100%.
4. Finally, press **Screen** to switch from Normal to **Inverted**. The screen should now show waveforms against a white background with gridlines clearly visible.

**To adjust trigger position:**

1. Turn the **Horizontal** right-hand knob to move the trigger indicator  to the first vertical grid line from the left edge of the screen. This will be the starting point of your display.

**To save screen image to flash drive:**

1. Insert flash drive.
2. Press **Stop** button. (By displaying only one sweep you eliminate noise fluctuations that may occur when display is running.)
3. Press **Save/Recall** button.
4. Press **Storage** button at right of screen.
5. Select **24-Bitmap** by repeatedly pressing **Storage** or turning the entry knob (image will be stored in this format).
6. Press the **External** button and then press **New File**.
7. Type out a filename:
  - Toggle between filename and keyboard fields by pressing .
  - In the keyboard field, select a character for your filename by turning entry knob.
  - Press the knob after each character is chosen. It should appear in the filename field.
  - You can enter characters in upper or lower case by toggling the lower right-hand key A/a, but it's probably not worth the bother.
  - In the filename field, use the entry knob to select a character you wish to delete. Then press the X button. Or delete all characters to the left of some starting point in the name by repeatedly pressing X.
8. Press the **Save** button. The file should be saved on your flash drive with the extension ".bmp".