

Experiments for the Lab9500

Chapter 1 – Lab9500 Hardware References

The Lab9500 Board has on-board I/O wired to specific pins on the chip. The compiler needs to be informed not only of the chip type, but the package type since the pinout varies according the package. All pins on the chip package, power included, are brought out to header footprints. This includes dedicated on-board I/O and uncommitted I/O. For using any of the on-board I/O, the compiler must be told in the source file what functions are connected to what pins.

Switch Inputs

For the switches, it is not recommended that you use the reference designators from the schematic for the switch identities. On the schematic, the three pushbuttons are S1, S2 and S3 and the DIP switch is S4. The DIP switch was originally an 8-position switch, but was changed to ten position. The sections of S4 I originally called S4_1 through S4_8. The two additional sections would be S4_9 and S4_10. There are two problems with this. First of all, the names are too long this way. Secondly, if the switches represent a hex nibble or a byte, the numbering should start with zero. So I recommend that the switches of S4 be numbered S0, S1, . . . S9, and the pushbuttons that are S1, S2 and S3 on the schematic can be called S11, S12 and S13.

Clearly, if the low eight switches were representing two hex nibbles, a more appropriate designation would be A3, A2, A1, A0 (S7, S6, S5, S4) and B3, B2, B1, B0 (A3, A2, A1, A0). Obviously, if a switch has a non-number use then a more appropriate symbolic name would be indicated. For example, a switch that indicated Night Mode might be called NightMode or NM.

The first registered experiment I tried on the Lab9500 was a four flip-flop divide-by-15 counter to generate 4Hz from the 60Hz line frequency. Since this was my time base, I called the flip-flops TB_A, TB_B, TB_C and TB_D. It seemed like a good idea at the time. But consider what a single minterm looks like:

$TB_D \& TB_C \& !TB_B \& TB_A$. If you have an equation with three or four product terms, you are going to go blind trying to see what the equation is. If we call the four flip-flops A, B, C and D, then the same minterm is $D\&C\&!B\&A$. Or if you choose to use the alternative PALASM notation, $D*C*/B*A$ which is even a little easier to see.

So I would make the following suggestions for defining symbol names. Make them as short as possible. One letter is great. While there is an historical and conventional meaning for A, B, C and D with A being the LSB. T, S, R and Q will not be nearly as recognizable as four binary weighted count bits. But there is no doubt what D3, D2, D1 and D0 stand for. So two character names, with the second being a number digit is also not bad. I do not recommend using the underscore as in TB_A.

Table 1. shows the I/O functions and the chip pins they are connected to. Note, that for the inputs, there is no switch corresponding to S10 on the board. However, if an external switch assembly is used in lieu of the DIP switch, J12 –11 is wired on the board to chip pin 32.

Input	Pin	Ref.Des
S0	39	S4
S1	38	S4
S2	36	S4
S3	35	S4
S4	27	S4
S5	24	S4
S6	20	S4
S7	19	S4
S8	33	S4
S9	34	S4
S10	32	NA
S11	7	S1
S12	10	S2
S13	11	S3

Output	Pin	Ref.Des
L0	50	LED1
L1	56	LED2
L2	57	LED3
L3	60	LED4
L4	61	LED5
L5	62	LED6
L6	63	LED7
L7	64	LED8
MR	49	LED10
SR	48	LED11
MA	45	LED12
SA	44	LED13
MG	43	LED14
SG	42	LED15

Table 1 - Assignment of I/O functions to chip I/O pins

Polarity Indication

ABEL follows the convention that a **one** or high voltage is true, and a **zero** or low voltage is false. This is quite fortunate. On the other hand, the connection of switch inputs may not agree with this. For example, we would like a true indication when a pushbutton switch is pushed. The pushbuttons, however, are wired in the conventional, traditional way. The switch is pulled high with a resistor and the switch connects the switch output to ground when switched. So, in fact, the switch gives a high indication when not pressed and a low indication when pressed.

We could accommodate this difference by using an inverted form in equations when we want to indicate a switch closure, but that is NOT recommended. Instead, in the input definitions we indicate that the switch is low-true, and when, for example, we want to reflect the fact that switch S11 is pressed (or true) in an equation, we can use S11 instead of !S11 (or /S11. The compiler, from the input definitions, knows when and where to invert signals internally so that the results are as indicated by our definitions.

By the same token, the LEDs on the board are also driven in the conventional, traditional manner, that is, the LED is pulled up to the positive supply via a resistor and the LED is turned on when the output driving it goes low. Accordingly, the LED outputs should be defined initially as low-true, and the user need not concern himself that the output actually must go low to turn the LED on. That is the compiler's problem and it always gets it right!

Note, that the DIP switches give a one indication when the toggle is up and a zero indication when the toggle is down, so the DIP switches don't need, indeed must not have, a low-true definition.

One input is not a switch. The AC wall transformer is opto-coupled and "squared-up" to give a 60Hz signal for use as a timebase or clock. Optionally, a PIC microcomputer can be used to give a 60Hz clock and many other choices as well. Jumpers determine the selection of the clock source.

Immediately following is a suitable, generic definition of the inputs:

```
"input pins
F60Hz pin 6;           "Frequency input from opto-isolator (or PIC)
S0,S1,S2,S3 pin 39,38,36,35;   "low four bits of DIP switch
S4,S5,S6,S7 pin 27,24,20,19;   "high four bits of DIP switch
S8,S9 pin 33,34;         "extra two bits on ext connector
!S11,!S12, !S13 pin 7,10,11;   "Pushbuttons, s1,s2 debounced;
```

Note, the double quote is a comment character in ABEL.

It might not be a bad idea to have a template in a file with the pin definitions in it. For a particular application, the original pin definitions might be copied and included as a comment, just to minimize the chance of errors in pin assignments. Immediately following the commented line would be the pin definitions as they apply to the particular application with more appropriate names assigned to switches and LEDs. Following are two lines with the first the commented out default definition. This helps to indicate which switch is used for which function.

```
"!S11,!S12, !S13 pin 7,10,11; "Pushbuttons, s1,s2 debounced;
!Up,!Dn,!MODE pin 7,10,11; Up,Down clock, & Mode (momentary) control
```

Accordingly, we know from the commented line that the UP clock is the pushbutton called S1 (or S11) etc. There is a problem in the commented line, though. We took the default input assignment and commented it out. This line, however, already had a comment within it. ABEL seems to have a problem with this. Therefore, when you comment out a line, remove any (") internal to the line to avoid compiler errors.

Definitions of output pins follow the same suggestions. First, use the original default output declaration so that the actual LED assignment is known. Then, if desired, comment out the default declaration and assign more application-meaningful names. Following are suggested default declarations.

```
"output pins
!L0,!L1,!L2,!L3 pin 50,56,57,60; "low 4 bits of 8 LEDs
!L4,!L5,!L6,!L7 pin 61,62,63,64; "high 4 bits of 8 LEDs
"Traffic Light LEDs: Main Red, Side Red, Main Amber, etc.
!MR,!SR,!MA,!SA,!MG,!SG pin 49,48,45,44,43,42;
```

This definition is shown as being commented out. Everything to the right of it is considered to be a comment.

It should be noted that there is no formal ABEL declaration for an input. The default for a pin declaration is an input. Note, that in the output declarations above, there is nothing in the definition to declare it as an output. ABEL is able to figure that out for itself. For example, if the following appears in the Equations declaration:

$$L0 = A\$B \# !A\&!B$$

ABEL knows that L0 is an output, and according to the pin definition, also knows to make L0 low when the equation evaluates as true.

Macrocells are numerous enough that a cell physically connected to an LED need not be a logic function itself, but just a monitoring of that function. This has the advantage, particularly in the case of the bank of 8 LEDs, that the outputs can represent various functions depending upon the state of some inputs, namely, the DIP switches. For example, with the full-featured traffic light, the LEDs might represent 1) the traffic light state machine 2) The long green time register 3) The short green time register and 4) The downcounter. Just two toggle switch inputs can select one of four possibilities.

ABEL Tutorial

The materials that come with the Lab9500 include an ABEL Tutorial that was obtained from the Internet, and was written by Jan Van Der Spiegel at the University of Pennsylvania. This tutorial was written with the cooperation of Xilinx, Inc. and was intended to be used with the Xilinx Foundation Series software. This software has been under development at Xilinx for some time, and recently has been offered free with several textbooks on Digital Design. It appears, however, that Xilinx intends to phase out the Foundation Series software and instead promote the WebPack software that is available free of charge, and downloadable from the Internet. Prof. Jan Der Spiegel's document was probably written just prior to the introduction of the Xilinx 9500 CPLD families. It is generally applicable to CPLD hardware, but in detail is probably more applicable to Xilinx FPGAs. It is certainly a good starting point and I appreciate the use of it. Because it does the job of explaining ABEL so well, there is no need to further document ABEL except for pointing specific application to the XC9500 family.