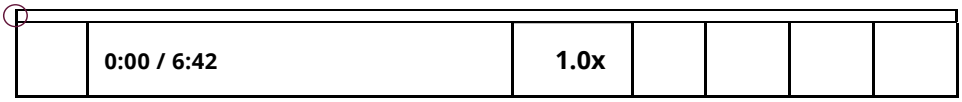
VIDEO 11.5B. CHAT TOOL PROGRAM WALK-THROUGH



DR. JONATHAN VALVANO: Next we'll show you

the software needed for the Chat tool.

We're going to add one more driver function, one more

input function, which is going to be non-blocking.

Non-blocking means that if the receive FIFO is empty

that means it has no data.

And so rather than waiting, like a busy-wait what will happen is

we will simply return a zero.

And that will mean that there is no data.

If the receive FIFO empty flag is zero, that means there is data,

and so we will input the data and return the data.

This will allow the main program to check

to see whether it has received any input from the other microcontroller

and processes it.

But if there is no data, it doesn't have to wait.

DR. RAMESH YERRABALLI: So this is different from the InChar

we wrote earlier which was a blocking I/O.

The difference is

that the blocking I/O would have sat in a loop waiting for the input

to be available.

And once it's available, it's going to return the data it reads.

DR. JONATHAN VALVANO: Absolutely.

DR. RAMESH YERRABALLI: OK, so what's next?

DR. JONATHAN VALVANO: This is the main program of our Chat tool.

We will initialize the PLL at 80 megahertz,

we'll use SysTick down here for wait, we'll initialization our UART,

and we'll initialize the port F buttons and LEDs.

The main body of the Chat tool, you can see, is a great big while loop.

The first thing we're going to do is check to see if switch one is pressed.

It switch one(SW1) is pressed, that means it's a zero,

we are going to select the next color in the wheel.

And you can see that is done with this line of code here,

which will add one to the color and then mask it with a seven so at all times

the value color goes to zero, one, two, all the way up to seven.

If the switch is not pressed, then we will go to the next step.

And we will check to see if switch two (SW2) is pressed.

If switch two is pressed, its time to send the message.

And so what will happen is we will encode the message and send it.

DR. RAMESH YERRABALLI: So switch one is our Chooser

and switch two is our Send button.

DR. JONATHAN VALVANO: The last thing we're going to do,

and if it's not pressed, we'll go to our third step, which

is to check to see whether we have any received messages.

And that was our input, non-blocking from the previous slide.

We saw with the non-blocking input we

had two choices, either we had no input

or we did have input.

If we have input we're going to decode that input.

In our case the encoding and decoding is very simple.

To encode the message, we took the number zero through seven

and added 30 to it to get the ASCII zero through seven.

To decode the message we're going to mask off the 30,

so again we have a number between zero and seven.

And we will set the color to that input value.

The fourth step is to output to the LED either the value

that we've selected with the press switch or the value

that we've read from the input.

And so if we don't have an input, we're going go here.

We're going to add a fifth step, and that is a 20 millisecond wait.

And this will remove the bounce of the switch.

And so these steps, one, two, three, four, and five

are repeated over and over again in the While loop.

DR. RAMESH YERRABALLI: So John, I understand the flowchart.

But I see that you have this variable called previous prevSW1

and prevSW2, what are those used for?

DR. JONATHAN VALVANO: They are used to make sure when I touch the switch,

it increments the color just once.

And the way I'm going to do it is in this if statement here.

I'm going to look for the touch, or the falling edge of switch one.

So the switch one will be touched when the switch one is zero.

But the previous value of switch one through the loop was a one.

So if the switch one is not pressed it goes through the loop

and there's a one in this variable.

And if I'm going around the loop, and I touch the switch,

it will become a zero.

So if switch one is a zero, and the previous time through the loop it's

a one, I've detected the first touch event.

So the color number is just increment once.

DR. RAMESH YERRABALLI: So otherwise the switch one would be zero

and the previous value of the switch would also

be a zero, that tells me that it's been pressed but not released yet.

DR. JONATHAN VALVANO: Yes.

And we do the same thing for switch two and it's previous two.

Because when I push the button, I only want to send one message.

Help

Solution: The operator selects the message to send by pushing the SW1/PF4 switch. While selecting the message the LED displays the message to be sent. Each time the operator pushes the SW1/PF4 switch, the system will cycle through the 8 possible colors on the LED. When the operator pushes the SW2/PF0 the message is sent. The message content is encoded as an ASCII character ‘0’ to ‘7’ (0x30 to 0x37) and sent via the UART. The driver from Program 11.8 is used. When a UART frame is received, the data is encoded and the‘0’ to ‘7’ data is displayed as the corresponding color on the LED. Figure 11.12 shows the hardware, which involves a 3-wire cable connecting the two LaunchPads.

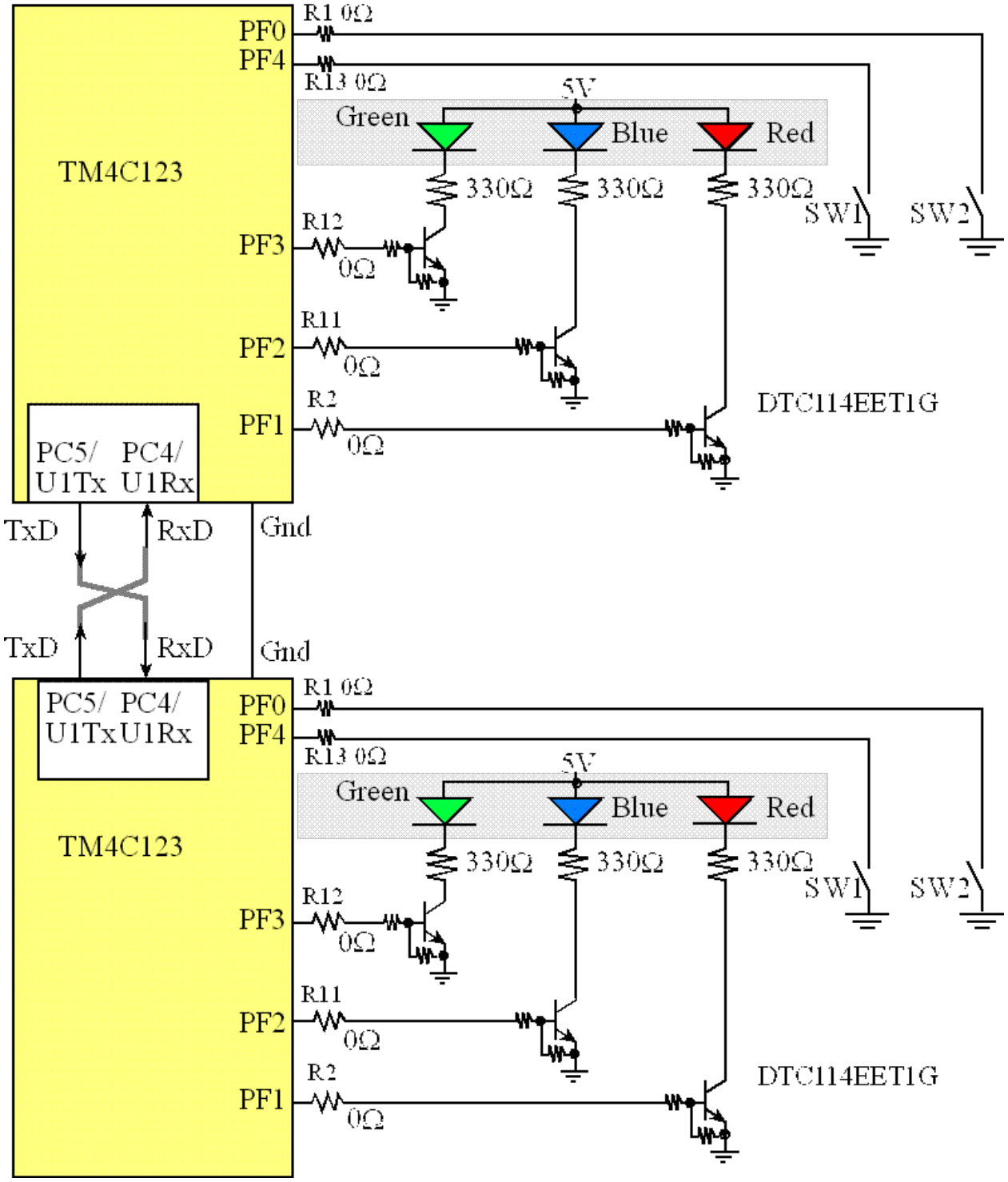


Figure 11.12. Distributed using two LaunchPads connected together by the UARTs.

```
//  red, yellow, green, light blue, blue, purple,  white,  dark
const long ColorWheel[8] = {0x02,0x0A,0x08,0x0C,0x04,0x06,0x0E,0x00};
int main(void){ unsigned long SW1,SW2;
  long prevSW1 = 0;        // previous value of SW1
  long prevSW2 = 0;        // previous value of SW2
  unsigned char inColor;   // color value from other microcontroller
```

```
  unsigned char color = 0; // this microcontroller's color value
  PLL_Init();              // set system clock to 80 MHz
  SysTick_Init();          // initialize SysTick
  UART_Init();             // initialize UART
  PortF_Init();            // initialize buttons and LEDs on Port F
  while(1){
    SW1 = GPIO_PORTF_DATA_R&0x10; // Read SW1
    if((SW1 == 0) && prevSW1){    // falling of SW1?
      color = (color+1)&0x07;     // step to next color
    }
    prevSW1 = SW1; // current value of SW1
    SW2 = GPIO_PORTF_DATA_R&0x01; // Read SW2
    if((SW2 == 0) && prevSW2){    // falling of SW2?
      UART_OutChar(color+0x30);   // send color as '0' - '7'
    }
    prevSW2 = SW2; // current value of SW2
    inColor = UART_InCharNonBlocking();
    if(inColor){ // new data have come in from the UART??
      color = inColor&0x07;     // update this computer's color
    }
    GPIO_PORTF_DATA_R = ColorWheel[color];  // update LEDs
    SysTick_Wait10ms(2);        // debounce switch
  }
}
```

Program 11.8. High-level communication network (C11_Network).