■ BY FRED EADY

INVASION OF THE CHIPKIT MAX32

With the introduction of the Arduino-speaking chipKIT Max32, about the only world-changing thing left for me to see in my lifetime is the arrival of aliens. There have been lots of intrusions into AVR-controlled Arduino airspace. However, the arrival of the 32-bit Microchip flagship microcontroller in the Arduino sector is akin to the Vulcan scout ship T'Plana Hath's first contact on Earth.

FIRST CONTACT

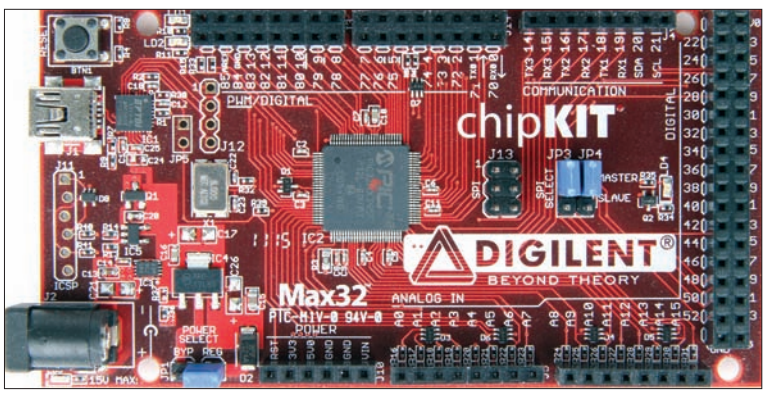
The chipKIT Max32 parked in **Photo 1** is capable of quickly ferrying Arduino shield payloads with minimal or zero modifications. The chipKIT Max32's warp drive is based on a Microchip PIC32MX795F512L 32-bit microcontroller clocked at 80 MHz. The PIC32MX795F512L is rich in on-chip peripherals. However, external components to support its on-chip peripherals are only required when a particular peripheral is put into use. For instance, the PIC32MX795F512L supports Ethernet and CAN. If you do not need Ethernet capability, the Ethernet PHY IC need not be mounted. The same goes for CAN. There is no need to include the MCP2551 CAN transceiver if the PIC's internal CAN hardware is to remain idle in your application.

Arduino is a programmer's paradise that shields the complexities of the hardware from the programmer. To that end, a specially modified Multi-Platform IDE (MPIDE) has been developed to assist in chipKIT Max32 sketch development. However, most "programmers" these days are also interested in what they are loading their precious code into. With that, we're going to board that chipKIT Max32 you see in **Photo 1** and check out the instrumentation.

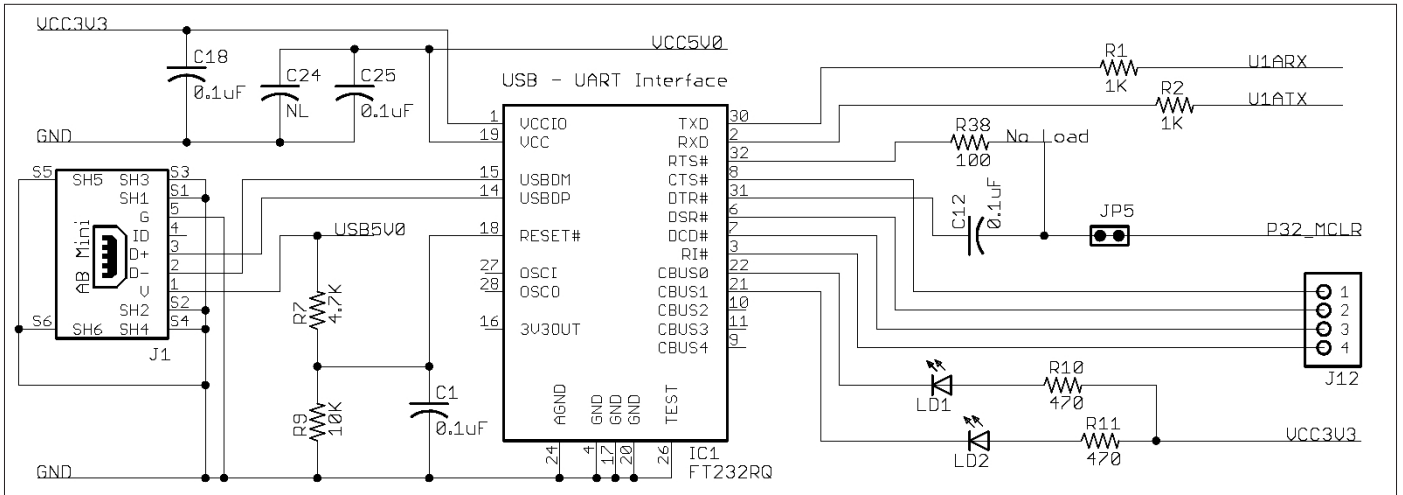
The chipKIT Max32's controlling microcontroller needs no introduction. The PIC32MX795F512L is the biggest, baddest, fastest microcontroller in the PIC family of devices. Every logical embedded peripheral a designer would need for an embedded application is integrated into the PIC32MX795F512L's silicon fabric.

This silicon embeds an intense USB subsystem. Thus, it would be logical to ascertain that the PIC32MX795F512L has no need for any help from an external USB-enabling peripheral IC. So, why is there an FTDI FT232RQ standing behind the chipKIT Max32's Mini-B USB connector? Answer me this. Does your brand new laptop have a nine-pin male RS-232 connector? The correct answer is "nope." Okay. Does your brand new laptop have a USB connector? The answer is "Yep. More than one." An RS-232 portal may not be physically mounted on your PC frame, however, RS-232 is not totally dead yet. So, if you don't have a regulation RS-232 interface on your laptop, how do you establish RS-232 type connectivity? Easy. You use one of the USB portals and a COM port driver. Okay. Then, why is that FTDI FT232RQ mounted on the chipKIT Max32 printed circuit board (PCB)? The answer lies in **Schematic 1**. Note that the FT232RQ's TXD and RXD signal lines are physically tied to the PIC32MX795F512L's UART1 pins.

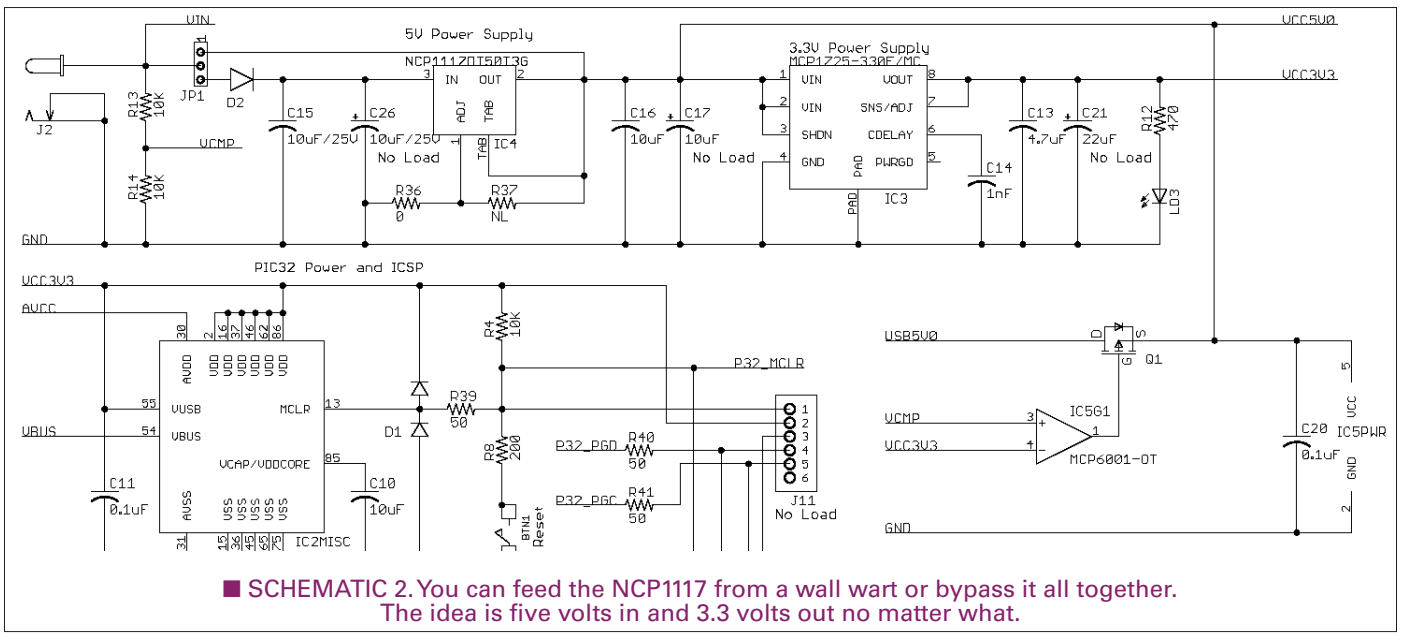
The chipKIT Max32's FT232RQ-based USB portal serves a dual purpose. From a communications standpoint, the Max32's USB portal provides a data path between the PIC and the MPIDE. Power for the Max32 can also be derived from the USB connection. Data activity is visually indicated by a pair of LEDs that sit directly behind the RESET button. The only other remarkable support components onboard the Max32 are the power supply electronics. The chipKIT designers took a page from the Vulcan play book. If it works and it has worked for a very long time, continue to use it. The 5.0 volt regulator in **Schematic 2** is a descendant of the caveman LM317 adjustable voltage regulator. In the case of the Max32, the Jetsons NCP1117 variant is a fixed voltage adaptation of its LM317 predecessor. The NCP1117's



■ PHOTO 1. It really doesn't matter if you lean towards Arduino or not. The chipKIT Max32 can work in the Arduino environment or fall back on its Microchip MPLAB/C32 C compiler roots. Either way, you have direct access to the resources of Microchip's most powerful microcontroller.



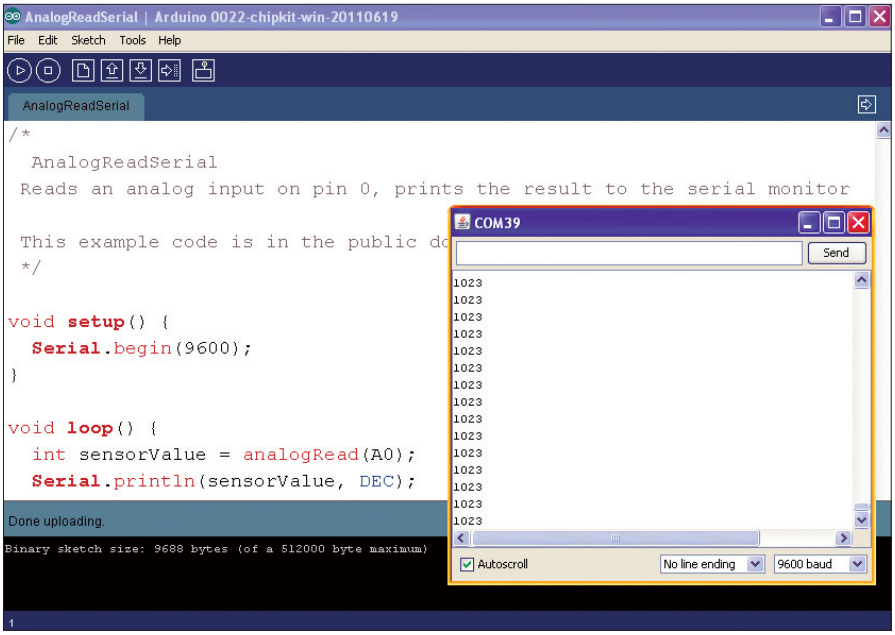
■ SCHEMATIC 1. There's no reason to put a regulation nine-pin RS-232 interface here if your PC has to be adapted to use it. The inclusion of the FT232RQ and a physical USB interface is a perfectly logical choice.



■ SCHEMATIC 2. You can feed the NCP1117 from a wall wart or bypass it all together. The idea is five volts in and 3.3 volts out no matter what.

job is to bring the raw chipKIT Max32 input voltage (7 to 15 VDC) down to an acceptable level (under six volts) for the MCP1725. The NCP1117 is a low dropout version of the standard LM317 and has a lower dropout voltage than its contemporary, the LM1117. In addition, the NCP1117 can supply 1.0 amperes of current versus the 800 mA offered by the LM1117. The "MCP" tells us that the 3.3 volt MCP1725 low dropout voltage regulator is a Microchip product. The MCP1725's entourage includes a minimum of relatively small and inexpensive stabilizing components. In fact, capacitors C17, C21, and C26 in **Schematic 2** are not mounted on the chipKIT Max32 PCB. The big picture is to use the NCP1117 to drop the input voltage to an acceptable level for presentation to the MCP1725 which will provide a very stable 3.3 volts at up to 500 mA for the Max32's PIC32MX795F512L and 3.3 volt peripherals.

The NCP1117's role depends on you. You can feed the NCP1117 from a wall wart, an external power supply, or totally ignore it by way of the POWER SELECT jumper. The 5.0 volts supplied by the NCP1117 to the MCP1725 can also be usurped by a host PC's USB portal. As you can see in **Schematic 2**, a Microchip MCP6001 comparator samples the voltages presented at the VCMP voltage divider junction and VCC3V3 output of the MCP1725. The output of the MCP6001 comparator drives the gate of a P-channel MOSFET which blocks or passes the USB portal's 5.0 volts to the VCC5V0 output. The comparator hookup makes sure that the external input voltage always wins the power sourcing battle when a USB power source is attached. If you're wondering why we need +5.0 volts other than to feed the MCP1725, it's the shields, Scotty, the shields. Many of the native AVR-based shields want to see a 5.0 volt power source. The chipKIT Max32 is designed to be compatible with Arduino shields. The Max32 digital I/O pins are five volt tolerant. However, the PIC32MX795F512L's analog-capable pins are not five volt tolerant. To keep the magic smoke contained within the PIC silicon, every PIC analog-capable pin is protected by a clamp diode and



■ SCREENSHOT 1. A jumper between the chipKIT Max32's A0 analog-to-digital input and +3.3 volts maxes out the PIC32MX795F512L's 10-bit analog-to-digital converter.

current limiting resistor. Regardless of the protection, the maximum analog input voltage remains at +3.3 volts. Scanning the Max32 landscape does not reveal any *Area 51* PIC32MX795F512L programming hardware. Instead, the standard set of six ICSP pads lie between the power connector and USB Mini-B interface. The ICSP pads are configured to interface directly to a PICkit3 programmer/debugger. If you accidentally find a way to eliminate the factory-loaded bootloader, the ICSP pads and an external PIC32MX795F512L programmer are the only way to reload the bootloader code. The ICSP portal is also your door to loading and debugging your home-grown PIC code.

The Max32 physically presents us with 83 I/O pins, 16 analog inputs, a 10 Mbps/100 Mbps Ethernet MAC, a USB 2.0 Full Speed OTG (On The Go) controller, a pair of CAN controllers, RS-232 modem control handshaking signals (DCD, CTS, etc.), an SPI portal interface, and a power portal. The length of the Max32 I/O access list translates into a gaggle of associated I/O pins and female headers. Fortunately for us, the Arduino boilerplate logically sorts them all out for us. The Max32 provides female headers that supply power to the shields and establish data paths between the shield electronics and the PIC32MX795F512L. The Arduino system dictates that the Max32 I/O pins range from zero to 85. Each I/O and analog input is clearly identified by silkscreen legend. We've walked around, on, and over most every major component that comprises the base chipKIT Max32. It's time to take our Max32 out of space dock and do some maneuvering.

OUT-OF-THE-BOX ARDUINO

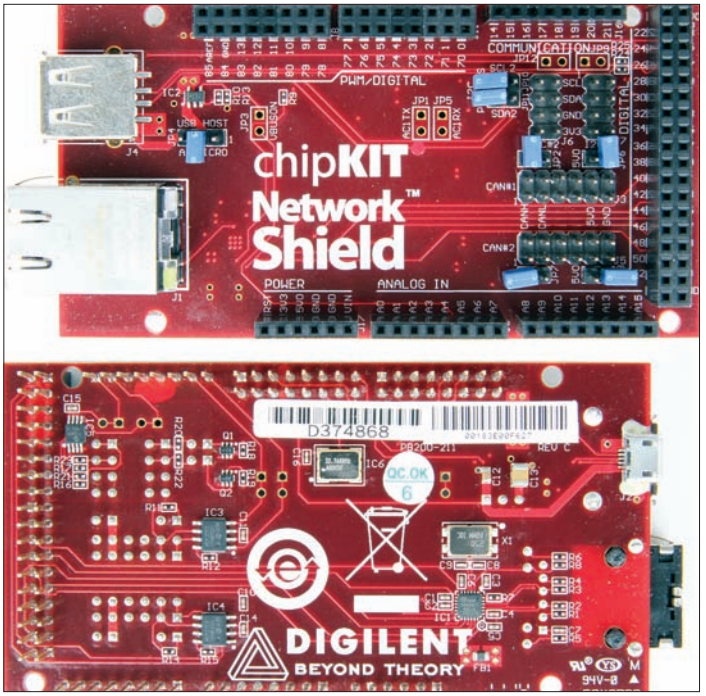
Your Max32 will come preloaded with a bootloader and a simple LED blinker application. I grew out of the LED blinker program rather quickly and decided to try out the MPIDE and Max32 with a simple analog-to-digital (A-to-D) application. So, I started the MPIDE, selected the Max32 as my target platform, and loaded the AnalogReadSerial example sketch.

Arduino is an intuitively obvious system. So, I won't insult your intelligence with a blow-by-blow description of how I loaded and ran the example program. Everything you need to see concerning my test run is displayed in **Screenshot 1**. I placed a jumper between the Max32's A0 A-to-D input and the 3.3 volt power pin on J10. The PIC32MX795F512L's A-to-D subsystem has 10 bits of resolution, and the 1023 decimal (0x3FF) result is what one would expect to see. I used the MPIDE's serial monitor tool to display the A-to-D readings in **Screenshot 1**.

Running those simple Arduino demos was enough to convince me that the PIC32MX795F512L could truly speak Arduino. I'll leave it to you to experiment with the Max32 at this level. If you're new to Arduino or the PIC32MX795F512L, I suggest stopping here, whipping out your Max32 hardware, firing up your copy of MPIDE, and doing some getting acquainted. You'll find lots of Max32-adapted Arduino example code within the MPIDE framework. Once you get comfortable with the Max32 and Arduino, you can pick up and continue from here.

THE ULTIMATE chipKIT Max32 SHIELD

Back in my corporate days, I was the communications specialist for our sales team. My sales pitches were purely technical and focused on the importance of setting up and maintaining a robust data network. I would always emphasize the point that no matter how big, how fast, or how expensive the CPU was, it was a useless piece of junk if it could not communicate with its peers and slaves via some sort of network. The same axiom holds true for small embedded devices like the Max32. With that networking thought in mind, get an eye full of **Photo 2**. Recall that Ethernet PHY IC we mentioned earlier? It's here. Remember that pair of MCP2551 CAN transceivers that were needed for the Max32 to participate in a CAN network mount? They're here too. The chipKIT Network Shield is an extension of the PIC32MX795F512L's internal communications subsystems. By fitting the Network Shield to the Max32's female headers, we gain total access to the PIC32MX795F512L's 10 Mbps/100 Mbps Ethernet engine and its CAN networking subsystem. The Ethernet PHY is made up of an SMSC LAN 8720 PHY IC and a Bel Stewart MagJack. The chipKIT Network Shield sources everything the PIC32MX795F512L's Ethernet MAC requires, right down to the 25 MHz crystal. The Ethernet



■ PHOTO 2. The chipKIT Network Shield complements the PIC32MX795F512L's internal communications subsystems by providing the physical means to access them.

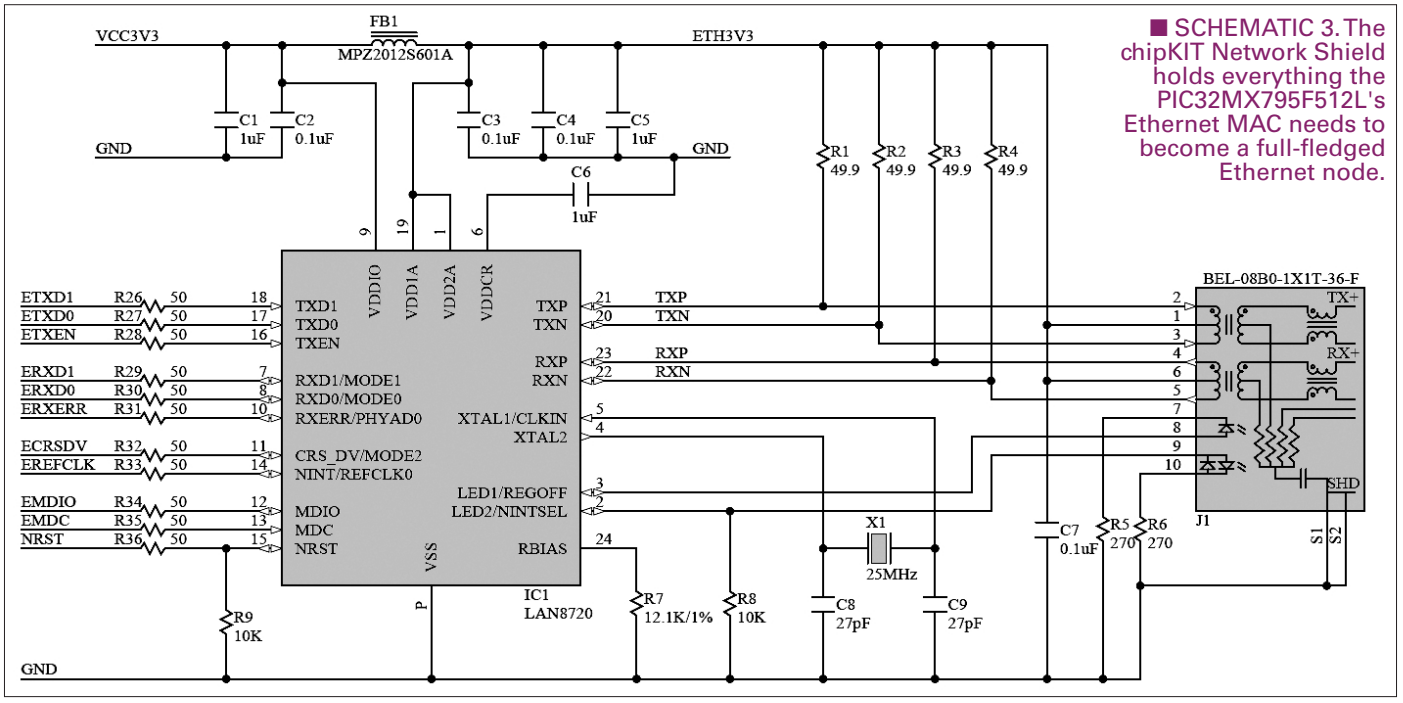
PHY portion of the Network Shield is graphically depicted in **Schematic 3**.

The MCP2551 CAN transceiver circuits are inked in **Schematic 4**. Note that the transceivers require a +5.0 volt supply. If you're CAN challenged, those 120Ω resistors are CAN termination resistors. The termination resistors reside at each end of the CAN network. Portals for accessing the I²C bus and an onboard 24LC256 I²C EEPROM are also revealed by the installation of the Network Shield. If you plan to use your Shield in an I²C network, you must use bus pull-up resistors in accordance with the I²C specification. Note the 2.2KΩ pull-up resistors on the 24LC256 SCL and SDA lines in **Schematic 5**.

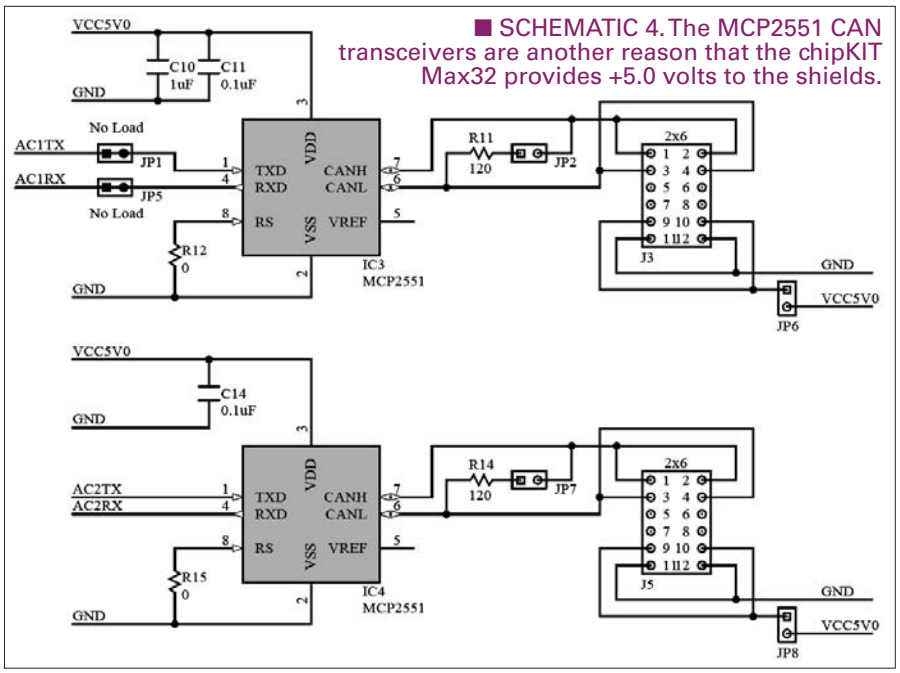
To utilize the PIC32MX795F512L's internal RTCC (Real Time Clock Calendar), all we really need from a hardware point of view is a 32.768 kHz clock source. The Network Shield is loaded with a 32.768 kHz crystal oscillator which we can use to keep up with the time and date. The Network Shield is equipped with a pair of USB interfaces. The Max32 can be configured as a USB host, a USB device, or a USB OTG device. When configured to act as a USB host, the Network Shield must have a means of supplying +5.0 volts at the USB host interface. Take a look at **Schematic 6**. The TPS2051B is a solid-state power switch that is under the control of the Max32's VBUSON signal. When the TP2051B's EN input is logically high, VCC5V0 flows to the TP2051B's OUT pin. The TP2051B's OUT pin voltage is manually directed to one of the USB interfaces via a jumper. An overcurrent condition is reported to the PIC32MX795F512L via the TP2051B's active-low OC pin. We've established one thing for certain: the Max32 loaded with a Network Shield is definitely not a piece of junk. The Max32/Network Shield combination has the ability to communicate with most any embedded or cloud-based computing platform. The Max32 can even talk to your car.

BILINGUAL SILICON

The Max32 was designed to support Arduino with a Microchip 32-bit microcontroller. However, the Max32 and the Network Shield combine to form a perfect PIC development platform. If you want to use the Max32 as a development tool, you'll need to take control of the



■ SCHEMATIC 3. The chipKIT Network Shield holds everything the PIC32MX795F512L's Ethernet MAC needs to become a full-fledged Ethernet node.



■ SCHEMATIC 4. The MCP2551 CAN transceivers are another reason that the chipKIT Max32 provides +5.0 volts to the shields.

Max32's FTDI-based USB portal. We'll do this using the PIC32MX UART peripheral library. The Arduino *Serial.begin(9600);* translates to this PIC32MX UART Peripheral Library snippet:

```
#define GetSystemClock()              80000000UL
#define GetPeripheralClock()          40000000UL
#define GetInstructionClock()
(GetSystemClock() / 2)

UARTConfigure(UART1A,
UART_ENABLE_PINS_TX_RX_ONLY);
UARTSetLineControl(UART1A, UART_DATA_SIZE_8_BITS
| UART_PARITY_NONE | UART_STOP_BITS_1);
UARTSetDataRate(UART1A, GetPeripheralClock(),
9600);
UARTEnable(UART1A, UART_ENABLE_RX_TX);
```

The PIC32MX UART Peripheral Library mnemonics are self-commenting and are as easy to read and follow as an Arduino sketch. Now that we have the Max32 UART's attention, sending a byte (0x55) of data is as easy as:

```
if(UARTTransmitterIsReady(UART1A))
      {
          UARTSendDataByte(UART1A, 0x55);
      }
```

Loosely translated, the PIC32MX UART Peripheral Library serial transmit looks like this Arduino statement:

```
Serial.write(0x55);
```

Odds are if you need to transmit, you may also need to receive. In Arduino speak, receiving data is as easy as:

```
int character;

if (Serial.available() > 0)
{
character = Serial.read()
}
```

Like the Arduino receive code, the Peripheral Library mnemonics are an easy read. There isn't much difference between the Arduino and the Peripheral Library code. As far as the PIC32MX795F512L is concerned, if you can code in C, you can code in Arduino, and vice versa:

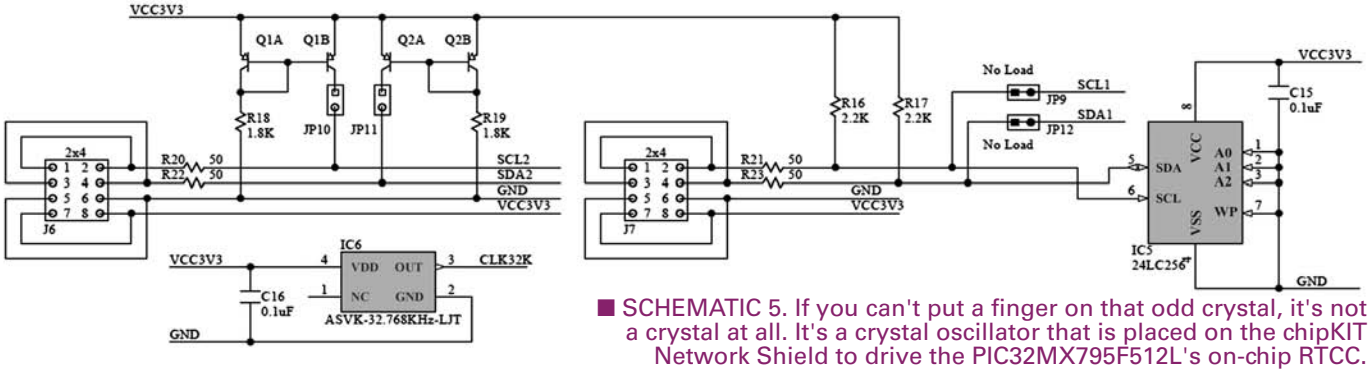
```
UINT8 character;

if(UARTReceivedDataIsAvailable(UART1A))
{
    character = UARTGetDataByte(UART1A);
}
```

Whether they are legacy RS-232 or USB-based, serial portals are staples in embedded computing. So, it's no surprise that Arduino has a strong set of serial routines in its bag.

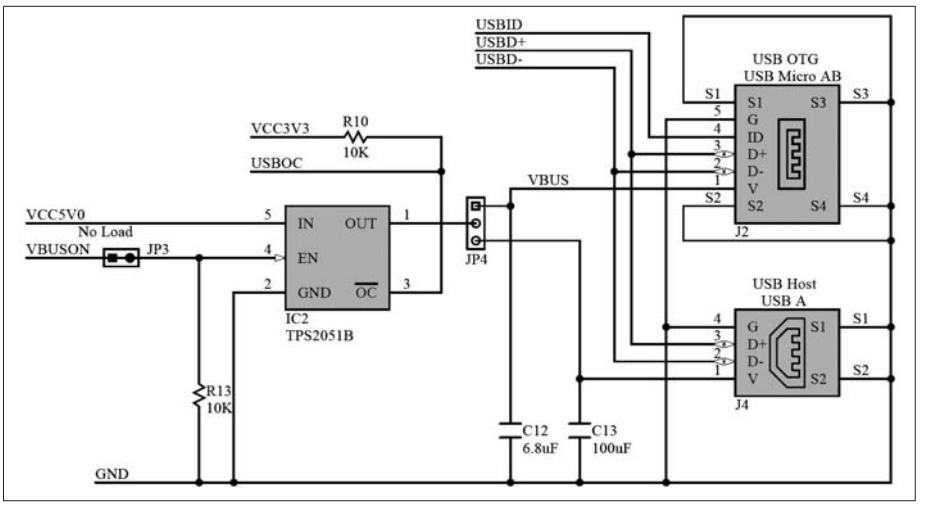
Some of the Max32 peripherals cannot be accessed bilingually using C and Arduino. However, at this very moment, there are guys and gals out there porting more and more PIC32MX code to the Arduino platform.

THE chipKIT Max32 AND THE MICROCHIP APPLICATION LIBRARIES

I decided to throw caution to the wind and load up my Max32 with a Microchip Application Libraries thumb drive application. Using my PICkit3, I blew up the Max32 factory bootloader and replaced it with the Mass Storage simple demo which is part of the Application Libraries.



■ SCHEMATIC 5. If you can't put a finger on that odd crystal, it's not a crystal at all. It's a crystal oscillator that is placed on the chipKIT Network Shield to drive the PIC32MX795F512L's on-chip RTCC.



■ SCHEMATIC 6. The PIC32MX795F512L contains all of the USB interface circuitry. All that we need to supply are the physical connectors. If our chipKIT Max32 is to act in the USB host capacity, throw in a switchable +5.0 volt power source with those connectors.

Here's the heart of the application:

```
//Opening a file in mode "w" will create the file if it
//  doesn't exist. If the file does exist it will delete
//  the old file and create a new one that is blank.
myFile = FSfopen("test.txt","w");

//Write some data to the new file.
FSfwrite("Bringing Arduino to the PIC32MX with chipKIT
Max32",1,50,myFile);


//Always make sure to close the file so that the data gets
//  written to the drive.
FSfclose(myFile);
```

My contribution to the Mass Storage application is rather obvious. Before compiling the Application Libraries thumb drive code, I set the USB host power jumper to feed the larger Type A USB connector on the Network Shield and plugged the Shield into the Max32. I used the Max32's Mini-B USB portal to power the Max32/Network Shield combination from my laptop's USB portal. Next, I opened the Mass Storage application with MPLAB, attached a PICkit3 to the Max32, and compiled the thumb drive application using Microchip's C32 C compiler. The Application Libraries Mass Storage application is bare bones and doesn't give any indication that it is working or not. However, when I attached the thumb drive to the Network Shield's Type A USB connector, the thumb drive's activity LED began to blink. After a few moments, the thumb drive's activity LED ceased to blink and remained illuminated. So, I detached the thumb drive and inserted it into my laptop's USB portal. Sure enough, there was a file created on the thumb drive called *test.txt*. I opened *test.txt* with a text editor application to find that the file contained the following: Bringing Arduino to the PIC32MX with chipKIT Max32. Enough said.

MORE chipKIT Max32 ARDUINO STUFF

We've shown the Max32 and Network Shield for what they are: excellent development tools for both Arduino and native 32-bit PIC32MX applications. If you've been on the fence about a Microchip presence in AVR land, check out the chipKIT Max32 forums. I think you'll be pleasantly surprised. You can access the chipKIT forums and purchase your Max32 equipment from the Digilent website (**www.digilent.com**).

Meanwhile, I've got some Arduino-to-chipKIT Max32 porting to do! **NV**