May 31, 2005

App Note 3553: CAN-to-Ethernet Using the DS80C400

This application note shows how to build a CAN-to-Ethernet solution using a DS80C400 microcontroller in a TINIm400/TINIs400 evaluation board. The example code allows the use of a TCP/IP connection to transfer data bidirectionally on the CAN and Ethernet networks. The complete project source is available for compilation in the Keil C environment.

Introduction

The DS80C400 is a highly capable, 8051-based microcontroller with Ethernet and CAN-bus network functionality. A Controller Area Network (CAN) bus is a popular high-reliability bus used in automobiles and industrial applications. The DS80C400 allows bridging between heterogeneous, networked manufacturing environments and eases integration among different tools. Bridging CAN to Ethernet enables remote network monitoring and control of systems not designed to interface with Ethernet.

This application note shows how to build a CAN-to-Ethernet solution using a TINIm400/TINIs400 Evaluation Kit, the Keil C compiler, and a simple C application.

Download: Source code used in this application note.

CAN Bus

The CAN messaging protocol requires the use of frames conforming to the CAN specification. These frames include a CRC, as well as bus arbitration bits to allow high-priority messages to override all others. The bit rate is determined by a combination of a time quantum (t_{qu}) and two time segments (t_{SEG1} and t_{SEG2}). The DS80C400 incorporates a single CAN controller, which provides operating modes that are fully compliant with the CAN 2.0B specification.

The CAN logical signaling may be placed on several types of busses, including copper and optical. In this example, we will use the differential bus defined by ISO11898. A transceiver must be used which matches the bus standard; we will use a MAX3051 3.3V CAN transceiver.

TINI/DS80C400

All hardware components necessary for this application note are found on the TINIm400 and TINIs400. The TINIm400 module provides the DS80C400 CPU, SRAM, and flash required for operation; the TINIs400 provides the socket for the module, as well as the Ethernet PHY, CAN transceiver, headers, and connectors for connection to Ethernet and CAN busses. We use the existing 14.7456MHz crystal and the clock multiplier built into the DS80C400 for an effective CPU frequency of 29.4912MHz. The CAN connector, J20, works with a standard DB9 to 5X2 connector, and the 120Ω CAN bus termination is enabled by bridging J19. Two TINIs400 boards can be connected to the same CAN network simply by connecting the corresponding CANH, CANL, and ground between the two boards. See: Schematics for the TINIs400 and TINIm400.

Example

In this example, we will use a bit rate of 50kbps, a common CAN bit rate. The corresponding setup values, given our 29.49MHz clock rate, are a prescaler of 14, t_{SEG1} of 13, and t_{SEG2} of 7. **Table 1** details standard bit rates available with this setup. Higher bit rates, up to 1Mbps, can be obtained by changing to a different CPU crystal (such as 15.0MHz). This would, however, add complexity to this example by requiring board modification. Please see

Application Note 2935 for more information on setting CAN bit rates on the DS80C400.

Table 1. Common Bit Rates Available Using 14.7456MHz Crystal.

Bit Rate	Prescaler	t_{qu}	t_{SEG1}	t_{SEG2}
10000	59	4.00us	16	8
20000	41	2.78us	13	4
50000	14	0.949us	13	7

On the Ethernet side, we use TCP for communication, as we want guaranteed delivery of the CAN packets over the network. We set up a bidirectional pipe for data to shuttle back and forth. Data from the Ethernet is packed to the maximum CAN frame size of eight bytes and sent across the CAN bus. The data in frames received from the CAN bus is then concatenated and sent across the Ethernet. At 50kbps, the CAN will be the bottleneck in the transfer. Two processes run on the protocol bridge, one handles CAN-to-Ethernet, and the other Ethernet-to-CAN. This allows simultaneous bidirectional transfers. The main process creates the CAN-to-Ethernet process, and then blocks in the accept() function while waiting for an incoming TCP connection. When the connection occurs, the main process awakens the companion process to allow both to work on incoming data. Once a network connection has disconnected, the companion process goes to sleep while the main process waits for the next incoming connection.

The error handling in this application is simplistic, and any adaptation into a reliable system should include analysis of all error and failure modes.

Loading and Running the Example

The source code and a prebuilt HEX file are included in the ZIP file. To load the HEX file, one needs the Microcontroller Tool Kit (MTK). Once MTK is installed and running, select `TINI/DS80C400` from the selection menu. Then select `Options->Configure Serial Port` and select the port connected to the development board. Also set the speed to 115200. Then, select `TINI->Reset`. You should see a prompt similar to this:

```
DS80C400/DS80C41X Silicon Software - Copyright (C) 2002-2004 Maxim Integrated Products
Detailed product information available at http://www.maxim-ic.com

Welcome to the TINI DS80C400/DS80C41X Auto Boot Loader 1.2.0
```

Load the HEX file with `File->Load HEX`. Once the HEX file is loaded, hit E to execute the application. You should now see something similar to this:

```
DS80C400 CAN to Ethernet Example
Built: May 27 2005 17:00:26

DS80C400/DS80C41X Silicon Software - Copyright (C) 2002-2004 Maxim Integrated Products
S/N: FB5E70038035DC89 MAC ID: 006035021122
DS80C400 Initialization Library Version 17
DS80C400 Kmem Library Version 5
DS80C400 Sock Library Version 10
DS80C400 Xnetstack Library Version 11
DS80C400 Task Library Version 8
IP    : 192 168 1 2
Subnet: 255 255 255 0
Prefix: 16
Gate  : 192 168 1 1
New task started with task id 146
```

```
Bound to port: 4004
Waiting for connection
```

Now using a simple telnet client, connect to port 4004 at the IP address displayed. Anything typed will be received and transmitted across the CAN bus. If any CAN traffic is received, it will be sent back through the same telnet connection.

Performance

This sample application can be deployed on two Ethernet-to-CAN bridge setups to allow feedback and to measure performance from Ethernet-to-CAN-to-Ethernet. Connect J7 of both boards into an Ethernet hub or switch. Connect pins 3, 4, and 5 of J20 on one board to the same numbered pins on the other. Status can be monitored through RS-232 serial connection to J12.

Because the DS80C400 at 29.49MHz can push about 160kBps through a TCP/IP connection over Ethernet, we will saturate the 50kbps CAN bus. Using an 11-bit ID and the full 8 bytes of data, we should have a total packet length of 95 bits. Our maximum theoretical throughput in this case would be about 4kBps, which was achieved during bench testing.

Conclusion

The DS80C400 is a fast microprocessor with I/O options that allow bridging of two popular communication busses. Ethernet-to-CAN allows non-CAN-enabled systems access to CAN-bus-enabled equipment through the common protocols of Ethernet and TCP/IP.

Relevant Links

- Microcontroller Home Page
- TINI Home Page
- High-Speed Microcontroller User's Guide
- DS80C400 User's Guide Supplement
- Dallas Semiconductor Microcontroller Support Forum

More Information

DS80C400: QuickView -- Full (PDF) Data Sheet -- Free Samples
DS80C410: QuickView -- Full (PDF) Data Sheet
MAX3051: QuickView -- Full (PDF) Data Sheet -- Free Samples