May 14, 2003

App Note 711: Creating Networked Multimedia Applications with the DS80C400

Impressive multimedia applications, including public address (PA) systems, networked doors, MP3 players, and security cameras, can be built using a low-cost, networked microcontroller. This article discusses how to use the DS80C400 networked microcontroller in example systems using audio and video.

Building a Networked PA System

Traditional PA systems (also called "overhead paging" or "public address" systems) use dedicated cabling and infrastructure. They are built on technology that was available before the transistor. This application note describes a system that can built to use a standard digital communications network instead of separate audio cabling.

A networked PA system can be made intelligent. For example, the paging system could interface to the building's access control system or a network server which knows the most likely location of an employee. A computerized PA system can repeat a message, freeing an operator to take more calls. The system could tap into the company-wide email system and allow for an email-to-voice service, or use a website where paging requests could be entered and then announced without further human intervention.

How do you build a networked PA system? First, start with at least one server that runs the web interface, email gateway, and has a microphone or something similar. We will call this server "master control." Next, you need a number of speaker modules. These modules are networked units with a digital-to-analog converter (DAC) to drive a speaker. The price for these speaker units must be kept low, and the units must be extremely easy to install in the field.

Figure 1 shows a network sketch with two buildings, seven speaker units, and one master control server. A router, not a bridge, connects the networks of the example buildings. (This has an important consequence for the software, described later in this article.) In our example, the DS80C400 networked microcontroller drives the speaker units. Even though a microcontroller does not have the processing power and memory resources of the latest PC systems, a PA system is neither bandwidth nor processing intensive. Uncompressed monaural audio sampled at 22.05kHz x 8 bits consumes less than 180kbps, and renders excellent voice quality. There is also no cost for hardware decompression.

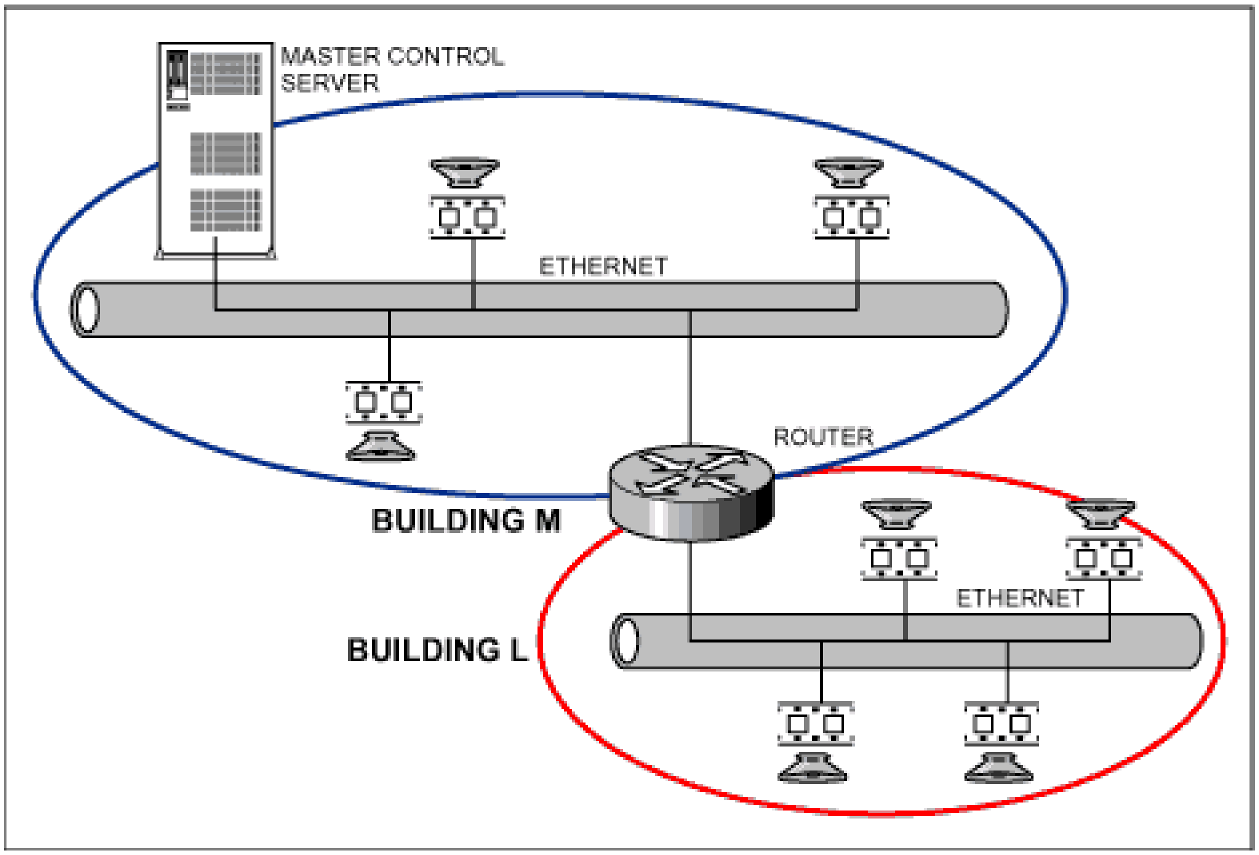

Figure 1. A network-based PA system can span multiple buildings.

Figure 2 illustrates the relative bandwidth requirements of a networked audio system. Even on an old (half duplex) 10Mb network with an effective 5Mbps bandwidth, the audio application uses less than 4% of that bandwidth. Most Ethernet networks today are at least 100Mbps.

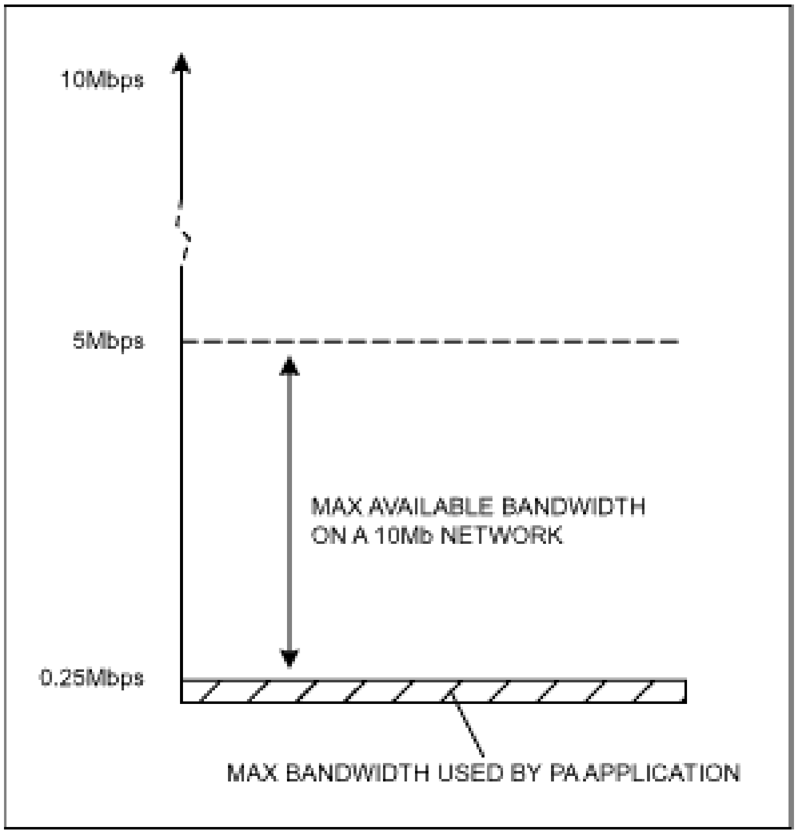

Figure 2. The bandwidth requirements for networked audio are very low compared to the available bandwidth of Ethernet networks.

Speaker Hardware
Apart from the DS80C400, a speaker module requires some memory (512kB of SRAM is sufficient), a network PHY, DAC, amplifier, and speaker. Refer to Application Note 609: Internet Speaker with the DS80C400 Silicon Software for a complete working description. The following technologies simplify installation and code distribution:

- DS80C400 NetBoot (see Networked Application Update on the DS80C400)
- DHCP to eliminate IP configuration, lowering installation and configuration cost
- Power-Over-Ethernet (see Power-Over-Ethernet) to simplify cabling and reduce material cost upon power-up, the DS80C400 ROM requests an IP address using DHCP, then queries the network for the latest version of the application. The application is then executed, and the system is ready to receive the audio data. Anew speaker module can be installed in the field by locating an unused network port and connecting the cable.

Networked Application Update on the DS80C400
The DS80C400 is equipped with network boot capabilities, a convenient way to load code into a fresh part. "NetBoot" can be invoked using the N command in the serial loader. It supports SRAM and flash memories.

The use for network booting goes beyond manually loading new parts. On a design without nonvolatile memory, such as flash, it is the easiest way to install an application after a power loss. NetBoot can also automatically check the network for code updates and install them (proceeding with the previously loaded "old" code if the network is unavailable). NetBoot uses the DHCP and TFTP network protocols to acquire an IP address and load program data. The network configuration can also be statically set and stored in a 1-Wire device. Address support includes IPv6 addresses.

The following steps are required to set up automatic network code update for the TINI400 evaluation module/socket. (Refer to the High-Speed Microcontroller User's Guide: DS80C400 Supplement at www.maxim-ic.com/microcontrollers for additional details, including the types of supported 1-Wire devices.)

1. Make sure a DHCP server is available on the network, or allocate a static IP address and record it in the 1-Wire device.
2. Install a TFTP server (e.g., tftpd) and either publish its IP address using DHCP or record the address in the 1-Wire device.
3. Upload the code onto the TFTP server in tbin2 format, under the file name TINI400.
4. Set the NetBoot jumper on the TINIs400.
5. Every time the system is reset, it will now automatically reload or update the code from the TFTP server.

Speaker Software

The software does extra work for easy hardware installation. Because there is a router between buildings (Figure 1), broadcast messages cannot get from one building to the other. Therefore, simple broadcast messages cannot be used. A new speaker must send multicast messages (see Multicasting) until master control confirms the speaker's location and parameters. A new speaker system does not know the location of the master control beforehand, and consequently sends out multicast messages asking master control to identify itself. If security is an issue, this exchange can be digitally signed to remove rogue systems claiming to be servers. Once configured using classic unicast messages, the speaker joins a multicast group and waits for audio packets. These audio packets are multicast from master control. The example software for the networked PA system was written in C (see Developing for the DS80C400). The following code can receive networked audio over a multicast socket.

```
#define MULTICAST_IP_MSB 239
#define MULTICAST_IP_2 19
#define MULTICAST_IP_3 0
#define MULTICAST_IP_LSB 22
#define MULTICAST_PORT 6789

 int s;                                                    /* socket handle */
 struct sockaddr address;                        * IP address */
 unsigned char xdata buffer[1600];

/* Step 1: Create the socket */
s = socket(PF_INET, SOCK_DGRAM, 0);
/* Step 2: Join the multicast group */
memset(address, '\0', sizeof(address));
address.sin_addr[12] = MULTICAST_IP_MSB;
address.sin_addr[13] = MULTICAST_IP_2;
address.sin_addr[14] = MULTICAST_IP_3;
address.sin_addr[15] = MULTICAST_IP_LSB;
address.sin_port = MULTICAST_PORT;
join(s, &address, sizeof(struct sockaddr));

/* Step 3: Listen for incoming packets */
memset(address, '\0', sizeof(address));
address.sin_port = MULTICAST_PORT;
bind(socket_handle, &address, sizeof(struct sockaddr));
/* Step 4: Now we're ready to receive data */
recvfrom(s, buffer, 1600, 0, &address, sizeof(struct sockaddr));
/* We actually received data! We could play it. */
printf("Data received:\r ");
...
```

With the exception of join(), these steps should be familiar to programmers who wrote code for TCP/IP networks. The function calls are very similar to Posix and Winsock. (Note: xdata is a Keil C keyword that tells the compiler where to allocate data.) On the DS80C400, join() and leave() initiate or terminate membership in a multicast group, respectively. All library calls - bind(), recvfrom(), etc. - in this and the following examples return status codes. Unlike the code in the abridged examples, it is preferable to check these return codes for errors and respond accordingly.

Power-Over-Ethernet
Attaching a device to the network can have a downside - it adds an extra cable for the network. Fortunately, the power supply can be integrated into spare wires of the Ethernet cable. There are several solutions to the problem, most commonly the IEEE802.3af standard calling for a 48V supply on pins 7, 8 (+) and 4, 5 (GND) of the 8-pin Ethernet connector. A 48V supply is commonly used in telephony applications, and thus often readily available in cabling cabinets.

For use with microcontrollers, the supply has to be regulated down to suitable levels. Refer to www.maxim-ic.com/appnoteindex for application notes describing the use of the MAX5910 and MAX5014 to build an efficient IEEE802.3af-compliant circuit.

Text-to-Speech

Using a microphone to capture live speech or playing canned audio stored on a network server is one way to use the PA system. Another use announces messages received in text form through email, from a web page, or by using the short message service on cell phones.

Retrofitting speech synthesis to the system is easy. The conversion can be done directly on the master control server, using a text-to-speech engine to generate waveform audio from the input text. The waveform can then be transmitted to the speakers like any other audio; changes to the speaker modules are not required.

Text-to-speech engines are widely available, and are an integral part of operating systems such as Mac OS X. A free Java speech engine can be found at freetts.sourceforget.net/. Commercial solutions sound more natural, so try the "Vocalizer" demo with American, British, and Australian accents on www.nuance.com.

Entertainment Grade Audio

CD audio poses a problem if the samples are transmitted in uncompressed form. Raw stereo samples at 44.1kHz x 16 bits would require 1.4Mbps of network bandwidth (or almost 30% of the 10Mb network), permanently exceeding the available bandwidth of many networks.

Compression algorithms such as MP3 can reduce the data rates, and therefore the network load ten-fold and make the system feasible. Paired with a hardware decompression chip, the DS80C400 can easily manage the task. In fact, a clock rate of about 36MHz is fast enough to seamlessly play 192kb MP3s. The TINI MP3 project at www.mp3elf.net is a network MP3 design based on Dallas micro-controllers. Schematics and a full-blown multimedia application are also available.

Multicasting

The DS80C400 ROM and the TINI runtime support multicasting. Unlike unicast packets sent from one source to only one destination, multicasting allows several destination hosts to receive the same data, saving bandwidth by eliminating duplicate traffic. Multicast differs from simple broadcast. Using the IGMP protocol, a host can subscribe to one or more multicast groups, and multicast traffic gets routed only to those parts of the network with at least one recipient. Unlike broadcast messages that require the bridging of networks, routers forward multicast messages.

On the network, multicast packets use a special class of destination IP addresses, also called the multicast group. From an application perspective, adding multicast support is trivial. A call to join() adds the host to the multicast group, and packets are received like any other UDP traffic. Good manners dictate to leave() a group when reception is no longer desired. (A host sends periodic membership reports after joining at least one multicast group. If the host crashes without leaving the groups, the group memberships eventually expire.) When choosing a multi-cast group for your own applications, make sure to avoid duplicate group allocations by following the guidelines in RFC 2365.

Images

A video source can also be connected to the DS80C400. Such a system could be very useful as a security camera, where an inexpensive camera takes a snapshot every second and transmits it over the network for display and storage. Post-processing on the server side can perform motion detection and alert security personnel.

Good camera choices are those with modern cell phones - they are not only small, but also in expensive and widely available. Most use a serial protocol for communication, but details vary among manufacturers. Be sure you have all required technical information before committing to a particular camera make and model.

Experimentation shows that the DS80C400 can transmit four frames of raw black and white video (240 x 180) per second without any hardware-assisted image compression and with some headroom that could be used for speech-quality audio. The following code sample opens a TCP connection to a network server to transmit a picture. (Even though we show the closing of the connection, the real application keeps the connection open to reduce overhead.)

```
int s;                              /* socket handle */
struct sockaddr address;      * IP address */
unsigned char xdata buffer[1600];

/* Step 1: Create the socket */
s = socket(PF_INET, SOCK_STREAM, 0);

/* Step 2: Fill in the target address 192.168.0.11 */
memset(address, '\0', sizeof(address));
address.sin_addr[12] = 192;
address.sin_addr[13] = 168;
address.sin_addr[14] = 0;
address.sin_addr[15] = 11;
address.sin_port = 8080; /* Target port */

/* Step 3: Connect to the server */
connect(s, &address, sizeof(address));

/* Step 4: Send the data in buffer */
send(s, buffer, sizeof(buffer), 0);

/* Step 5: Close the connection */
closesocket(s);
```

For those familiar with Unix network programming, closesocket() is close(). The DS80C400 version of the close() function is used by the file system. Like Windows systems, socket handles are not interchangeable with file handles on the DS80C400, and a separate function for sockets must be used.

The camera system clocks the DS80C400 at 73.7MHz, close to the 75MHz limit. The 73.7MHz frequency is generated using a fundamental mode crystal with an 18.432MHz frequency, and the PLL is integrated into the DS80C400 to multiply the frequency by four. This design reduces the overall system cost, while still allowing operation near the high end of the microcontroller's maximum frequency. In addition, 18.432MHz x 4 is also a good baud-rate generator for a synchronous serial communication.

Networked Door

It is easy to combine the concepts of a security camera with bi-directional audio, a button, and a buzzer. This system allows us to build a networked door (Figure 3). The applications are endless, especially when combined with access control and security logs.

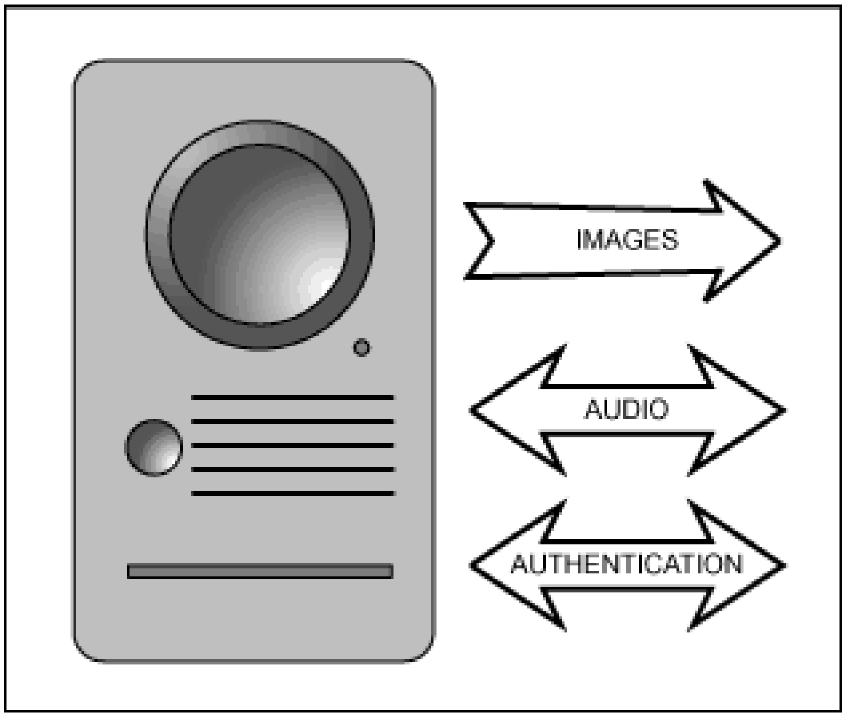

Figure 3. A networked door interface combines video, audio, and access control.

For the DS80C400 the button and buzzer are just peripherals that can be hooked up to a plain I/O port. In Keil C, I/O ports can be easily defined using sfr and sbit:

```
/* Define port 1 */
SFr p1 = 0x90;
/* Define P1.7 (port 1 is bit addressable) */
sbit p1_7 = p1^7;
```

```
/* Toggle P1.7 */
p1_7 = !p1_7;
```

Using iButtons and the 1-Wire master interface built into the DS80C400 makes adding authentication to the network door easy. (This interface is somewhat more complex to program, so Dallas Semiconductor provides libraries to simplify this task.) Refer to www.ibutton.com/TINI/applications/lock/ for an example showing how to connect a motorized door strike plate.

A final note: a system such as the networked door probably uses multiple processes (or tasks). The DS80C400 ROM contains a task scheduler. The following examples show how it can be used from C. Again, return codes should be checked in industrial-grade applications.

```
unsigned char pri, task;

/* Get the current task */
task = task_getcurrent();

/* The current task's priority */
pri = task_getpriority(0);

/* Decrease the priority */
task_setpriority(0, pri-1);

/* Sleep */
task_sleep(0, 0, 500);
```

The programming model also contains useful functions such as task_fork(), which creates a new task by duplicating the current task. task_kill() destroys a task and task_suspend() puts a task on hold. These and other features are described in the High-Speed Microcontroller User's Guide: DS80C400 Supplement on the website.

Conclusion

A small and inexpensive networked microcontroller can be the powerful heart of interesting and useful multimedia applications. We encourage readers to use the DS80C400 to reproduce or refine the ideas presented in this article. For free samples, visit www.maxim-ic.com.

Developing for the DS80C400
Software applications for the DS80C400 can be developed in a variety of ways. For rapid proto-typing, consider using Java and the TINI runtime environment. For applications where every cycle counts, hand-optimized assembly language is best.

For this article we use C. The Keil C compiler (www.keil.com) supports the 24-bit contiguous mode of the DS80C400 (refer to Application Note 606: Configuring Keil PK51 Tools to Support 24-Bit Contiguous Addressing Mode), allowing up to 16MB of code/data space. To use this mode, the extended versions of compiler and linker (CX51, LX51) are required. These tools are part of the Professional Developer's Kit (PK51).

Dallas Semiconductor provides C libraries that interface to the built-in DS80C400 network stack. The libraries and an illustrated step-by-step guide detailing how to create projects for the DS80C400 using the Keil development environment can be found on the Dallas Semiconductor ftp site, ftp://ftp.dalsemi.com/pub/tini/ds80c400. These libraries greatly simplify network programming. For example, the creation of a TCP connection is reduced to the universally known sequence of socket() and connect(). For technical support, join the TINI mailing list at lists.dalsemi.com.

More Information

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