White Paper

Embedded VoIP for Commercial and Industrial Applications

Freescale/Arcturus/Encore Software

freescale™ semiconductor

Contents

1 An Introduction to VoIP

VoIP stands for Voice over Internet Protocol. It is a process for sending audio signals, primarily voice, over a data network, such as the Internet. The audio is converted into a digital signal and compressed to reduce data throughput requirements. Then it's converted into packets and streamed across the network. At the receiving end, the data is decompressed and converted back to an audio signal.

There are a number of advantages to using VoIP instead of analog transmission, with the main one being that you do not need dedicated analog cables to carry the signal. In many situations an existing data network, wired or wireless, can be used for VoIP. It is also much easier to route the signals to different destinations because it's just a matter of changing the destination address instead of physically switching the analog circuit. Since the signals are converted from analog to digital, it is easier to maintain good audio quality, even in harsh industrial environments. Additional advantages include:

- The system may be able to use an existing application controller, such as a Freescale ColdFire® MCU, to minimize additional cost
- It's easy to record and archive calls on a computer system
- It's easy to connect to the public phone system through a gateway
- Networks can be designed to provide more than one route to a destination, providing inherent fault tolerance

VoIP communications typically occur between two endpoints. Data packets can be sent directly between them and may not need an intermediary server during a conversation. In many industrial applications the required endpoints are known and can be programmed directly into the systems. If more flexibility is required, a server can be used as a kind of electronic telephone book that stores a list of endpoints and their IP addresses and sets up the initial connection between the endpoints. Once communication is established, the server is not needed for the remainder of the call. If access to the public telephone network is required, a server may be used as a gateway.

2 Industrial VoIP

The factors propelling the adoption of commercial and industrial VoIP are significantly different from those that drive the same technology in the consumer market. For instance, companies face a different set of challenges when they try to implement audio applications in building systems or develop innovative new health care products to assist our aging population. To address this, we need to change our basic assumptions about VoIP services.

Consumer products are mostly commodity devices, with price the primary market driver followed by product features. They are the purview of manufacturers who look to fill a specific need at a precise moment in time. The very nature of this type of product means that the developers need to be prepared to dedicate considerable resources to a solution that can fulfill immediate requirements, typically at the edge of the adoption cycle and in advance of stiff competition. Little consideration is given for product lifespan, component selection or platform or software scalability.

While this model works well for VoIP applications in ultra-high volumes, meaningful access to the technology's core building blocks has not necessarily migrated to the industrial and commercial markets. Industrial developers recognize the benefits of VoIP, but they may not have the internal capacity to create a software team with signaling as well as low-level DSP vocoder integration experience, thus hindering their ability to develop a complete solution from the ground up.

That being said, there are still a number of compelling reasons for implementing VoIP in industrial and commercial applications.

- Better customer service—integrating support directly into the equipment (fast-serve restaurants, ATMs, manufacturing equipment)
- Consolidating infrastructure and reducing installation costs—running on a central data backplane (building systems)
- Providing more flexible service delivery—seamlessly integrating multiple locations (nursing and health care facilities, security systems, customer service)
- More flexible customer support options—centralizing customer service support and offering that support in multiple languages (chain restaurants)

Industrial customers are generally looking for complete VoIP solutions that can be easily integrated into their applications without the need for telephony expertise. It is also not practical to offer solutions with large up-front licensing costs and multiple complex licensing requirements.

A hospital bed or gurney with a patient communication device is a good example of an industrial VoIP application. Using wireless technology, a network connection with the patient can be maintained even when he or she is being moved around the hospital. The device can also include an intercom push button for a direct connection to the nurse's station. By using VoIP this could automatically be routed to different locations, or even different doctors and nurses, to accommodate shift changes. In addition, the device could function as a regular telephone through a gateway to the public phone system, thus allowing the patient to stay in touch with friends and family. The same industrial VoIP technology enables remote in-home health care monitoring that supplies the same data as in-hospital monitoring.

2.1 Protocols and Signaling

There are three main software components to a VoIP system: the signaling stack, the media transport and the audio subsystem. These need to be tightly synchronized by a master application that manages the state information for call flows, loads vocoders, enables different features and starts or stops media transport, audio and other services.

The dominant protocol for VoIP has evolved from legacy H.323 and MGCP signaling solutions to the lighter weight Session Initiation Protocol (SIP) RFC 3261. SIP is a request-response type architecture that looks very similar to HTTP. SIP itself does not include the media. Instead, SIP's role is to set up, change and tear down the media sessions. The media is carried independently using real-time protocol (RTP) RFC3550, which capitalizes on the speed of the User Datagram Protocol (UDP) to transport media streams using 10, 20 or 30 ms packets. In addition to SIP, a media negotiation protocol called Session Description Protocol (SDP) RFC4566 aids in the negotiation of the correct vocoders, packet size (ptime) and destination port addressing.

Several related components are also necessary: Network Time Protocol (NTP) to synchronize time stamps; Simple Traversal of UDP Networks (STUN) to traverse network address translation (NAT) firewalls and underpin support for networking TCP/IP; UDP; point-to-point protocol over Ethernet (PPPoE) and/or Dynamic Host Configuration Protocol (DHCP).

SIP's open architecture, its familiar nature and relative elegance has helped accelerate the technology into new markets and industries. SIP's reuse of a number of well established functional VoIP components, such as methods of media encoding, has provided strong compatibility with existing consumer network technologies. The result has been an opening up of a traditionally closed industry to a new generation of communication devices that includes commercial and industrial applications.

Table 1 (below) shows several signal compression algorithms and the resulting audio quality compared to the standard for long distance telephone calls (toll quality).

Vocoder	Quality	Codec Type
G.711	Toll Quality	Narrow Band
G.726	Near Toll Quality	Narrow Band
G.729AB	Below Toll Quality	Narrow Band
G.723.1	Below Toll Quality	Narrow Band
iLBC	Below Toll Quality	Narrow Band
G.722	Better Than Toll Quality	Wide Band

Other audio-quality and user-enhancement functions include:

- Acoustic echo cancellation for systems with a loudspeaker
- Automatic gain control for microphone input
- Line echo cancellation when connecting to a standard analog telephone set
- Caller ID
- Touch tone (DTMF) support

Acoustic echo cancellation is particularly challenging because it is highly dependent on the physical implementation of the equipment. Because of this, it is important to make key parameters, such as gain and echo path, easily adjustable by the system integrators so they can be optimized for a particular application.

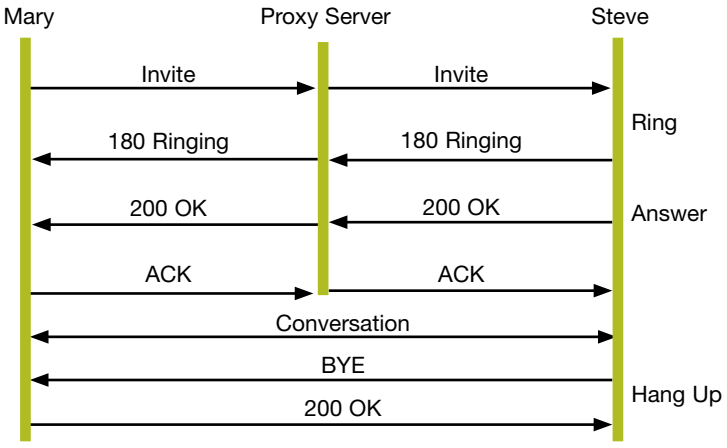
2.2 Understanding a SIP Dialog

Since SIP is a plain text protocol, tools like Wireshark (www.wireshark.org) are invaluable for analysis and debug. They are also excellent resources for understanding call flows and the interactions between elements. To help illustrate, consider the interaction of two UA endpoints constructing a basic call through a SIP server. The server may consist of several components, but for this example it is simply a proxy with a registrar server. A basic implementation like this can easily be downloaded and set up on a PC (Brekeke SIP server, party SIP or, if you're ambitious, even Asterisk) along with a PC SIP client, such as Zoiper.

In a simplified call dialog, the UA would first REGISTER itself with a SIP server, a challenge authentication would be given and credentials exchanged. The server would authorize the endpoint by providing a 200 OK message, and the endpoint would be bound to the server and ready to place a call. The user "Mary" would instruct the UA to place a call to user "Steve." The UA would encapsulate the requested location in an INVITE message using a SIP uniform resource identifier (URI). A SIP URI looks much like an email address (steve@destination.com), and this would be sent to the proxy server. The proxy would look up the location of the device by using the SIP registrar server and route the request to the correct destination.

In the meantime, the proxy would update the caller with progress messages: TRYING would indicate that the callee has not yet accepted the INVITE, and RINGING would mean the callee is available and the proxy is waiting for an answer. Once the call is answered a 200 OK is sent that contains the media session information, including the agreed upon vocoder and port locations to connect the media. At this point, the proxy server may no longer be involved in the call. Conversation can be enabled through a direct link between the caller and the callee until the communication is terminated. (See Fig. 1)

Figure 1: A Simplified VoIP Call Dialog



2.3 About RTP Media

RTP is a media transport mechanism that capitalizes on the speed of UDP to transmit media sessions. Data can be encapsulated inside the RTP packet in a number of standard formats based on the vocoder negotiated by the SDP. In VoIP these are typically the ITU standard G.7xx vocoder formats, and they are transmitted in packet-size increments of 10 ms. Since RTP relies on UDP, it is a best-effort protocol, meaning that re-transmit signaling does not occur, and if a packet is lost, nothing is done at the transport layer to accommodate it. Since RTP packets are relatively small and frequent, packet loss over a reliable network is negligible. However, each packet does not necessarily take the same network path, and packets may arrive at their destination out of sequence. Therefore, the dynamic jitter buffer arranges the packets in the correct order, monitors network latency and manages the balance between slow packet reception and an audible delay to the user.

2.4 VoIP Implementation Hurdles

While flexibility and openness are the SIP's core strengths, they can also be drawbacks. The RFCs very clearly define how certain tasks, such as registration, invites and basic call flows, need to be transacted. However, more complex call flows, such as broadcasts, conferences, transfers and even some ringback or reinvite functions, can be handled in multiple ways, requiring varying amounts of interaction between the endpoint and the server. The complexity issues are related to system interoperability. Simple call flows are usually not a problem, but more advanced call flows, features and services may not work across all infrastructure equipment. For device manufacturers that control the end-to-end system, such as a carrier, this is generally not an issue. However, for equipment installers that have multiple sites, each with an existing VoIP infrastructure, this can create a software management and interoperability nightmare.

A second issue with traditional VoIP is how the business model for licensing the technology is created. The traditional model targets mass market device manufacturers—a signaling stack port is licensed from one vendor, a vocoder port or audio subsystem is licensed from a second, and then the integration team brings the components together, writes a master application, tests and debugs the system and takes the product to market. This model ensures high-quality components and provides a direct channel of support, but it relies on the expertise of the integration team to guarantee a robust, interoperable solution, and the up-front license terms can be prohibitive.

Reacting to the rapid acceptance of open source technology, this model is changing, and signaling libraries and telephony applications are now publicly available. This is not the case for well-optimized or specialty vocoders that are required by industrial and commercial applications. The licenses for audio subsystems can exceed $50,000 (USD) per product in addition to runtime royalties.

It's also important to remember that VoIP is real-time. It can't be buffered like video media, and it can't retry like Web data. Because of this determinism, it's critical to consider the embedded system as a whole. For example, say at the very moment an occupant answers a door-access intercom call the DHCP lease for the device expires. Does the call end? Does the device renegotiate the lease? Does the interface get restarted, or does it even need to be? The product as a whole has to be designed to consider the potentially fatal impacts of related systems. DHCP, NTP, IPtables, routing, DNS and others are all necessary components of a complete product, but how they impact the rest of the system is dependant on the implementation.

2.5 Device Management and Other Considerations

In traditional telephony equipment, the server manages all the endpoints. An administrator logs onto the server, selects the device node they want to modify and changes its profile. With today's VoIP systems there is little provisioning and management cooperation between the SIP server and the endpoint. In addition to the base telephony setup, including unique account information, rules and restrictions, it is necessary to consider all other system requirements needed to support the application. These include network services, firewalls, routing and DNS. These lead to complex configuration requirements for each endpoint that could number in the hundreds or thousands. For a practical deployment, local device management for installers needs to co-exist with secure central management access for ongoing system configuration and mass deployments.

3 System Solution Overview

Several hardware and software components need to come together to build a VoIP system. The hardware components need to be tightly integrated to ensure clear software data paths. The hardware design rules need to mitigate noise and provide adequate isolation between analog and digital signals. A good reference design, along with a well-architected software package, can provide the building blocks around which the application can be developed. Building from a solid base platform can reduce project risk and decrease time to market.

3.1 Core OS Considerations

Operating system requirements can vary dramatically, depending on the end application. Simple applications can benefit from a compact real-time scheduler or a single-threaded monolithic deployment that doesn't require an OS. From an architecture standpoint, these models can be ideal solutions for media applications because they are inherently hard real-time systems. However, this type of architecture generally lacks protocol support and commonly available drivers. Scalability can also be a consideration, meaning that heavy software modifications may be required to enable different features or to target an alternative market niche. Developers may also have to deal with proprietary code, or tools may have limited capability or heavy license fees.

Linux® is a mature open source OS that offers a number of advantages, including a plethora of kernel services and file systems, plus broad compatibility with industry standards for security, networking and

peripheral devices. This helps ensure product scalability, allowing the OS to act as an abstraction layer that can enable the portability of applications, from low-cost processors suitable for simple endpoints to sophisticated high-end devices targeted at server-class equipment. In addition, most semiconductor companies offer a complete Linux distribution optimized for their products.

However, the open source solutions are not without their shortcomings. Linux and µClinux™ need software priority scheduling to ensure that a critical task is not given a lower priority than a non-critical task on the system. Improper schedule priorities can lead to dropped packets and latency issues.

3.2 The SIP Stacks

In a simplified form, a SIP network generally consists of various endpoints, such as handsets, intercom devices or media terminals. These devices form a network by interconnecting with each other through a server or by directly using a peer-to-peer capability. The server itself may or may not be interconnected over a private or public network to other SIP servers.

3.3 Endpoint Software Stack

Endpoints generally represent the user experience. They are the tactile product we interact with, like the steering wheel, throttle or brake pedal in a car. Endpoints rely on the coexistence of two user agent functions—the user agent client (UAC) and server (UAS). Both the client and server act independently based on the SIP dialog. For example, when placing a call from an endpoint, the UAC sends a request to the server and waits for a response. This is a typical client/server interaction. Turning this around and creating a call to an endpoint will result in a role reversal where the UAC is the one receiving and responding to messages. It's this dual role client/server nature that gives SIP its enhanced flexibility and peer-to-peer capability.

3.4 SIP Server Infrastructure

SIP server infrastructure is responsible for managing the endpoint status, processing messages, responding to requests and supplying routing information. Unlike an embedded SIP endpoint, the server infrastructure generally resides on a much more powerful system, such as a PC or communications application server. To help with these functions, the system is broken down into two main server components—the proxy and the registrar server. Since SIP is a peer-to-peer protocol, a server infrastructure is actually not required (SIP endpoints can quite happily chat with each other directly), but each server element does provide additional functionality, enhancing the capability of the system as a whole. It's important to note that both the proxy and registrar server may exist on the same machine or within the same software package.

- **The SIP proxy** plays the role of traffic controller and directs requests and responses to the correct SIP entity. It does this by interpreting the SIP message headers, rewriting them as necessary and forwarding the message to the destination. SIP proxies can act like a client or a server, depending on the message. SIP endpoints can communicate directly with one another. However, using a proxy means that each endpoint doesn't need to know every other SIP element on the network. Instead, they have a centralized way to access each other.
- **The SIP registrar** server maintains database entries that contain the current location and credentials for each endpoint. These endpoints can authenticate with the registrar server and record their location and names for the proxy to look up on behalf of other endpoints or SIP elements. With a registrar server, endpoints don't need to manage a database to track the locations of other endpoints. This allows endpoints to be more dynamic yet still reachable.

3.5 Endpoint and Server Management Requirements

There are three initial device management issues that must be considered before implementing an embedded VoIP communications system for commercial and industrial applications:

- The local needs of an installer, who will need to set up the initial configuration and validate that everything is working properly, must be addressed
- There needs to be a method that will allow a less experienced administrator to change configuration parameters
- There needs to be a way to efficiently update the software

When outlining the device management requirements, it's important to remember that many devices may exist in one location and there may be many interconnected locations.

It's also important to look at device management from the endpoint perspective and consider its dynamic resource requirements, such as idle state versus in call, keeping in mind the device's overall capacity and any affect that management may have on it. For example, invasive management methods, such as SNMP, common among customer premises equipment (CPE), force changes on the device with little consideration to the device's current status. It relies on the operator to view the system and ensure any changes will not adversely affect the user experience.

This model works well in the carrier environment, where expert operators regularly manage like devices. In the industrial space this responsibility will probably be shared by three individuals: the installer, who will set up the system, the administrator (likely IT staff), who will manage it and the user, who will interact with it. This shared responsibility means that every precaution needs to be taken to ensure that the device will protect itself from error events and misconfigurations and have a reliable way to easily recover. This also means that authentication is an absolute requirement. The public nature of most networks dictates that encryption should ensure that access is granted only to controlled users and that any connections are kept private.

4 Audio Subsystem

The audio subsystem prerequisites can vary considerably, depending on the application implementation and its requirements. For example, a simple one-way audio loud-speaker used for paging may only require a single vocoder combined with some form of half-duplex (one-way) digital-to-analog audio conversion. More complex systems may need to provide specific features:

- Background echo cancellation
- Audible status tones to indicate the progress of the call
- Full duplex audio support for emergency push-to-call panels in elevators
- The ability to broadcast to hundreds of endpoints simultaneously for fire and alarm systems
- The signaling capability of DTMF tones to maintain compatibility with legacy equipment in control applications used to support large equipment
- Half-duplex audio support for applications interoperating with two-way radios

4.1 Voice Processing Subsystem

The audio subsystem includes primarily vocoders and echo canceller modules, which have to be highly optimized in terms of CPU load and memory usage. There are a number of vocoder standards available with varying bit rates and voice quality. Selecting the correct vocoders for a particular system depends on the application/system requirements and system resource availability. The following section defines vocoder standards used in commercial VoIP applications and includes a quick reference table that summarizes and compares the specifications of various vocoders.

- **ITU-T G.711**—specifies PCM MU-Law and A-Law encoding for compressing speech at 8 KHz to 64 kbps. Appendix-II of this specification defines the silence compression techniques, such as voice activity detection (VAD) and comfort noise, to reduce the average bit rate transmitted during the silence intervals. Appendix-I of this specification details the packet loss concealment (PLC) functionality. The PLC handles packet losses that occur during the transmission over the data network.
- **ITU-T G.726**—is a waveform coder that works on the principle of adaptive differential pulse code modulation (ADPCM) and is used to compress speech at 8 KHz to 16, 24, 32 and 40 kbps. This vocoder also supports silence compression techniques and PLC
- **ITU-T G.729AB**—is a low bit-rate vocoder that works on the principle of conjugate structure algebraic code-excited linear prediction (CD-ACELP) for compressing speech at 8 KHz to 8 kbps. It operates at 10 ms frames with a total algorithmic delay of 15 ms. Annex-A of this specification defines silence compression techniques. This vocoder also supports PLC.
- **ITU-T G.723.1**—is a low-bit-rate dual-rate speech coder based on the principle of multi-pulse maximum likelihood quantization (MP-MLQ) and algebraic code-excited linear prediction (ACELP) with a total algebraic delay of 37.5 ms. It compresses speech at 8 KHz to 5.3 or 6.3 kbps.
- **iLBC**—is a vocoder based on the principle of block independent linear predictive coding (BI-LPC) that compresses speech at 8 KHz to 13.3 or 15.2 kbps.
- **ITU-T G.722**—is a wideband speech coder based on the principle of UB-band adaptive differential pulse code modulation (SB-ADPCM) with a total algorithmic delay of less than 1 ms. It compresses speech at 16 KHz to 64, 56 and 48 kbps.

Table 2 summarizes the specifications of various vocoders.

Vocoder	Bit Rate (kbps)	Frame Length	Algorithm	Algorithm Delay (ms)
G.711	64	Any	PCM	0.125
G.726	16, 24, 32, 40	Any	ADPCM	1
G.729AB	8	10 ms	CS-ACELP	15
G.723.1	5.3, 6.3	30 ms	MP-MLQ and ACELP	37.5
iLBC	13.3, 15.2	20 ms, 30 ms	BI-LPC	20, 30
G.722	64, 56, 48	10 ms	SB-ADPCM	10

4.2 Voice Quality

Some of the factors that affect the voice quality in a VoIP system are described below.

- **Bit rate:** Normally, increasing the bit rate improves voice quality.
- **Packet (frame loss):** Packets may be dropped on an IP network due to network congestion. Packet loss may be random or burst, based on the network conditions. Voice quality dramatically degrades as the packet loss increases. Packet loss concealment algorithms are built as part of vocoder standards, using history buffers to synthesize the speech for the lost frames.
- **Network jitter:** Normally, the packets are generated at regular intervals (packetization period) at the transmitter end. However, each packet experiences different delay while traversing the IP network, depending on the network conditions. This variation in the delay is called jitter. Jitter is alleviated using an adaptive jitter buffer, in which a playout buffer is used to store the packets and play them out in sequence. The downside of the jitter buffer is the increased end-to-end delay.
- **Echo:** This can be in the form acoustic echo, which is due to coupling between microphone and speaker, or it can be electric (line) echo due to hybrid circuits, as in the case of 4-wire-to-2-wire conversion, which is required in VoIP gateways. The echo is noticeable in VoIP systems due to higher end-to-end delay. Acoustic echo cancellers and line echo cancellers are used to cancel acoustic and electric echo respectively.

4.3 Voice Quality Measurement

There are essentially two methods for assessing voice quality: subjective and objective. Subjective methods employ human listeners to evaluate all aspects of voice quality. ITU-T P.800 defines mean opinion score (MOS) as an important metric for subjective determination of voice quality.

PESQ (perceptual evaluation of speech quality), defined by ITU-T P.862, is an objective method for perceptual voice quality measurement. PESQ uses a sensory model to compare the original unprocessed signal with the degraded signal from the network. The resulting quality score is comparable to the subjective MOS that is measured according to P.800. PESQ takes into account the different impairments, such as coding distortion, error, delay, packet loss, etc. PESQ scores normally range from 1.0 (poor quality) to 4.5 (high quality).

Table 3 provides the typical PESQ values for various vocoders.

Vocoder	PESQ
G.711	4.3–4.4
G.726	4.0–4.2
G.729AB	3.5–3.7
G.723.1	3.3–3.5
iLBC	3.5–3.7
G.722	4.0–4.2

4.4 Selecting a Vocoder

Vocoder selection in any VoIP system depends primarily on the following factors.

- Voice quality
- Network bandwidth
- Algorithm delay
- CPU load

Table 4 provides the characteristics of different vocoders with respect to the above factors:

Vocoder	Voice Quality	Bandwidth	Algorithm Delay	CPU Load
G.711	High	High	Low	Low
G.726	Good	Medium	Low	Low
G.729AB	Medium	Low	High	High
G.723.1	Low	Low	High	High
iLBC	Low	Low	High	High
G.722	High	High	Low	Low

4.5 Echo Canceller

Echo cancellers are required for VoIP systems because of high one-way end-to-end delay. When this delay is short (less than 25 ms), the echo is not noticeable. However, since the end-to-end delay is usually higher in a VoIP system, echo is one its major drawbacks.

Echo cancellers must perform the following general functions:

- Cancel the echo as quickly as possible at the beginning of the call
- Dynamically track the echo path changes
- Provide robust double-talk detection to avoid undesirable breaks in voice communications when both ends are active
- Operate well in the presence of background noise

An echo canceller uses an adaptive filtering algorithm to predict the echo path then generates a close replica of that path and subtracts it from the signal. The result is an echo-free signal.

Normally, there are two types of echo cancellers:

- **Line echo cancellers (LECs)** are designed to cancel echoes resulting from the reflections in the telephone hybrid circuit. There are generally one or two noticeable reflections from the hybrid, which are usually delayed by less than 32 ms. Normally, the echo characteristics do not change frequently, and therefore the LEC design is simpler than the acoustic echo canceller.
- **Acoustic echo cancellers (AECs)** are designed to cancel the echo that results from the acoustic coupling between the microphone and speaker. The acoustic echo cancellation process is considerably more complex than line echo cancellation, as outlined below:
  - In general, the impulse response of the acoustic path is longer
  - Acoustic echo path is non-stationary because of the dynamic acoustic properties of any given physical space
  - Acoustic echo includes both linear echo from the acoustic signal and non-linear echo from speaker non-linearity
  - Acoustic echo is influenced by the design of the enclosure

Due to these complex acoustic path characteristics, it is important to make key parameters, such as gain, echo tail length, buffer offset and non-linear processing elements, easily adjustable by the system integrators so they can be optimized for a particular application.

5 Freescale ColdFire MCF53281 Embedded VoIP Solution

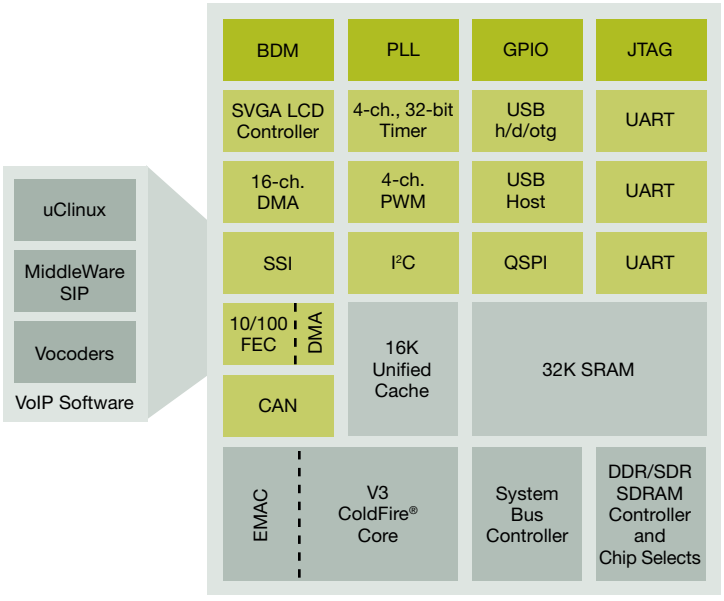
5.1 The ColdFire Advantage

Most low- to mid-range 32-bit processors do not have sufficient performance for audio (voice) processing. And most low- to mid-range DSPs do not have enough control capability to both control an application and manage a network connection. This generally necessitates the use of both a 32-bit CPU and a DSP for VoIP applications.

Most of Freescale's ColdFire embedded processors include an enhanced multiply accumulate unit (EMAC), which enables them to process VoIP audio and control the VoIP application as well as manage a network connection. Using a ColdFire processor instead of a CPU/DSP combo results in a simpler system with a lower total cost.

In addition to the EMAC, ColdFire processors, such as the MCF53281 device, include a rich on-chip peripheral set that is suitable for various VoIP applications. The MCF53281 MCU provides both 10/100 Ethernet and SSI synchronous serial support, which are required for high-speed digital audio communication between a host processor and a D/A or A/D codec. External peripheral devices, such as keypad scanners, I/O controllers or EEPROMs, can be connected to the host processor via QSPI or I²C. Interrupts, GPI/O and serial UARTS are also available. What's more, the MCF53281 controller contains an SVGA LCD controller, which makes it ideal for touch panel applications.

Figure 2: MC53281 Block Diagram



5.2 MCF53281 Software Bundle

The Freescale MCF53281 software bundle includes all the components required to deploy a full-featured VoIP system for voice-enabled industrial applications. The solution uses a hybrid of open source and proprietary elements that include complete VoIP and device management software with APIs and example applications that can be modified or used as is. These include:

- uClinux 2.6.21 or greater (drivers, kernel, userland applications collection, network services)
- Arcturus Management Middleware (complete device management, secure Web user interface [Web UI] and remote provisioning)
- Arcturus voice and media middleware
- Encore software voice processing subsystem

The software licensing costs associated with the proprietary components are bundled together as part of the MCF53281 VoIP processor cost. This means that there are no additional fees to pay for access to the software and no prohibitive NRE charges. The MCF53281KIT can be used as a reference platform to develop a hardware product or the MCF53281 VoIP modules can be used for prototype and small run applications. Freescale and Arcturus provide on-going training, a dedicated support site and direct e-mail support.

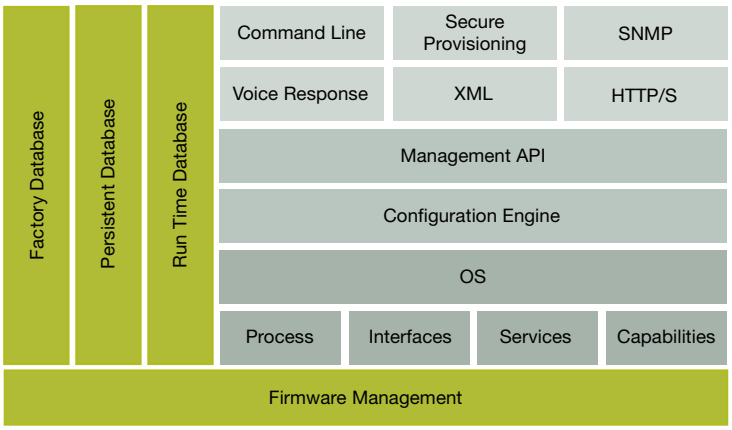
5.3 MCF53281 Management Middleware and Configuration Tools

The management middleware provided as part of the MCF53281 package hooks into the processes, services and interfaces internal to the Linux 2.6 kernel sysfs as well as the drivers, scripts and applications within the system. Through these hooks the middleware can change the configuration or report settings/statistics of any process, service or interface. The system relies on three database realms:

- A FACTORY database that contains the default values of the device, including serial number, MAC address and other default recoverable state information
- A PERSISTENT database that contains account information, personal settings and other user configured device parameters
- A RUNTIME database that is resident in RAM only and contains all real-time (current) device configuration settings, statistics, packet counters, interface information and others

The management middleware API talks directly to the database realms, which, in turn, signal a change to the management engine. The management engine contains a system of dependencies that are structured to take a macro view of the device and implement changes in a controlled way so as not to adversely affect the operation of the device.

Figure 3: Management Stack Diagram



A practical example of this is to consider the DHCP leasing reference from earlier in this paper. Under middleware control, if a call is in progress, the middleware engine will change the mode of the DHCP client to prevent the endpoint from losing its lease. At completion of the call the middleware engine will return the DHCP mode back to normal and the lease will be renegotiated if required. This control effectively delays the renegotiation and prevents the interface from going down, which would effectively kill the media session. While this is a fairly simplistic example, similar controls are provided for telephony and other network settings. The management API itself uses a simple set/get architecture compatible with SNMP. All database values are stored using the industry standard management information base (ASN1 MIB II) compliant format.

The MCF53281 package also includes tools that make use of the middleware API and implement various methods of device management, including secure remote provisioning and a Web-based user interface.

The Web UI is a feature-rich SSL-enabled method of configuring the device locally or remotely. It has direct access to the API and includes set up wizards, network tables and packet counters as well as diagnostic tools and feature, account and administrative settings. Since the Web UI is HTML, the source can easily be viewed or modified as required.

In addition to the Web UI tool, a remote provisioning tool is provided to help manage up to several thousand devices. This tool uses the unique credentials inside the device (username, passwords, serial number, firmware version) to authenticate with a remote Web server and establish an encrypted SSL connection for file transfer. The file itself is a script that can be executed through the management API and has access to the same resources as any other management tool.

The provisioning file may contain just enough information to set the device up generically or to provide a unique device profile or reference to obtain updated firmware. Since the file is obtained through an HTTPS connection, a fairly common Web server can be used as the provisioning infrastructure, which provides cost-effective flexibility and a more reliable transport than TFTP and other systems. The server itself can serve static files or exploit the unique credentials to auto-generate dynamic provisioning files on the fly.

5.4 Other Management Tools and Services

Two other management tools are provided—a command line client that uses the simple set/get management middleware API and a collection of MDNS-enabled applications that are compatible with Zeroconf requirements, discovery and configuration. These applications include DHCP, with the capability for server INFORM functions to support IP address retention, and advanced fall back modes for lease failure. Discovery is used to automatically find a provisioning server and to easily find the Web UI (using a Bonjour-enabled browser).

5.5 Firmware Management

Firmware upgrades are handled through the Web user interfaces or the provisioning system. Several local strategies are used on the device to structure and manage firmware in order to optimize updates and increase robustness.

At the lowest level of the middleware is a sophisticated bootloader that fully supports the flash device. It can read, write and erase flash as well as support persistent objects, environment variables and a garbage collection algorithm for wear leveling. The bootloader flash support has the ability to define partition information internal to the flash file system. The partitions are structured in such as way as to allow for mutually exclusive kernel, Web and rootfs images. This has two benefits. First it allows targeted firmware updates, which minimizes failures due to down time and network traffic. Second, it provides the capacity for the bootloader to manage multiple kernel and userland images, thus creating a fallback framework to recover from firmware failure or other events. This framework is particularly useful for remote devices or for those in difficult to access locations.

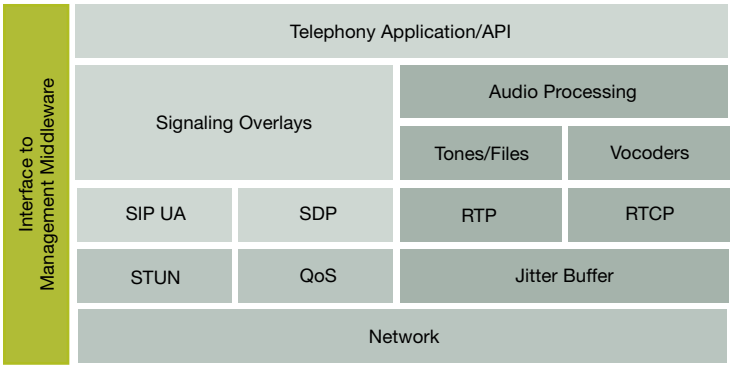
As the device boots, the bootloader decompresses the CRAMFS kernel image into RAM and jumps to its location to execute. The bootloader passes a kernel argument that defines the rootfs start address.

The processor's watchdog control and reset state information along with the bootloader's partitioning capability provide all the necessary components needed to arbitrate among multiple firmware images based on various failure modes. While this model of firmware recovery does require additional flash resources to store the redundant partitions, it provides better flexibility and remains more economical than storing a number of monolithic images. For applications in health care or industrial control where power fluctuations and external stimuli are common, the additional flash is certainly justified, considering how much it adds to the robustness of a critical recovery system.

5.6 MCF53281 Voice and Media Middleware

The voice and media middleware consists of a signaling stack that uses the open library components oSIP and oRTP. A runtime configurable signaling abstraction layer resides above the libraries to maintain call flow compatibility across different infrastructures. This allows deployment of a solution that is interoperable with SIP equipment from different vendors without having to manage various firmware loads. The telephony application is a multi-threaded, multi-context application that contains the internal logic for call handling, state information, voice response and outgoing call rules.

Figure 4: VoIP Stack Diagram



The phone application includes a loadable digit map that defines how the system should handle special feature codes, such as *70 (the North American standard to disable call waiting). It also contains logic that determines when the user's input should be packaged as an INVITE and sent on the network, such as the standard NANPA (North American Number Plan Association) 11-digit dialing (1-xxx-xxx-xxxx). All user configurable settings are available through the management middleware, including account information, vocoder preferences, speed dial settings, intercom and IVR. The negotiation, loading and unloading of the audio path, signaling conformance, device driver and vocoder are handled transparently by the middleware. Call control and command operation are enabled through a simple API.

There are two parts to the voice middleware API—a daemon application called atemul and a reference application called atcmd. The atemul application opens a channel into the middleware telephony application, and atcmd creates a command line interface. The interface uses a familiar AT command set similar to a modem. Source code is provided to help integrate atcmd with the application, or it can be used as is and attached to any interface to receive commands directly from an external source. An interface is provided through the voice and media middleware API into the management middleware to change a subset of telephony settings, such as volume controls, and obtain device information directly from the telephony application.

In addition to the telephony application, a number of related components are integrated to support the full duplex voice operation. These include NAT traversal using NTP client and server for accurate time stamping, VLAN for traffic shaping, STUN and QoS. The configurable settings for these applications are available via the management middleware database API.

The voice and media middleware supports most standard telephony features, such as do not disturb, call waiting, disable call waiting, caller-ID block, reject anonymous calls and attended and unattended call transfers with ring back support. To help with system development and configuration, a simple interactive voice response (IVR) is tied into the management middleware. This system uses telephony API calls to announce the device's IP address, phone number, last incoming call and other simple functions, such as play file command. This feature is particularly useful when needing to learn the IP address of the internal configuration Web server. By default, it is also programmed to respond to an external pushbutton that is connected via GPIO.

To help develop emergency assistance buttons, health care monitoring or other push-to-call applications, the voice middleware has two additional predefined GPIO signals that correspond to speed dial settings 1# and 2#, configurable from the Web user interface. This makes a push-to-call demo possible out-of-the-box by assigning a valid SIP account to the device and simply configuring the speed dials.

In addition to point-to-point communication, the middleware supports one-to-many modes of operation suitable for overhead paging, announcements or intercom functions. The software utilizes a subscribe-type architecture where endpoints are configured to send and/or listen to one of up to 99 broadcast audio groups. The audio streams are carried over multicast UDP and use an independent command packet to set up, maintain and terminate the session. The session is transparently encoded, packetized, transmitted and decoded as part of the middleware, which is designed to operate alongside any SIP infrastructure and co-exist with an endpoint configuration.

6 Summary

In order to meet the needs of next-generation products, it is critical to give the tools of innovation to the developers who innovate. This may seem straightforward enough, but in the traditional model for accessing VoIP software, the core components have been isolated by complicated business models, burdensome licensing costs and protocols that require specialized expertise. While this model may not be prohibitive for large OEMs and telecoms, it has stymied adoption in commercial and industrial markets where two-way digital voice communications can dramatically impact the way they do business.

Patient/nurse intercom stations no longer need to be hard wired to one fully staffed central desk, and industrial VoIP can enable health care monitoring to extend into homes or mobile applications. Building systems can benefit from consolidated infrastructures and gain the robustness that networks and protocols provide. Fast-serve restaurants can offer multi-language support for their drive-through customers via a centralized contact center. And operators who work in sensitive areas or with specialized equipment can have direct access to dedicated voice support to address glitches before they result in costly down time.

While no single solution will ever meet the needs of all applications, a simplified business model with access to the core software building blocks, board level hardware and dedicated support resources will help tear down the barriers for developing voice products in commercial and industrial applications.

How to Reach Us:

Home Page:
www.freescale.com

e-mail:
support@freescale.com

USA/Europe or Locations Not Listed:
Freescale Semiconductor
Technical Information Center, CH370
1300 N. Alma School Road
Chandler, Arizona 85224
1-800-521-6274
480-768-2130
support@freescale.com

Europe, Middle East, and Africa:
Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
support@freescale.com

Japan:
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku,
Tokyo 153-0064, Japan
0120 191014
+81 3 5437 9125
support.japan@freescale.com

Asia/Pacific:
Freescale Semiconductor Hong Kong Ltd.
Technical Information Center
2 Dai King Street
Tai Po Industrial Estate,
Tai Po, N.T., Hong Kong
+800 2666 8080
support.asia@freescale.com

For Literature Requests Only:
Freescale Semiconductor
Literature Distribution Center
P.O. Box 5405
Denver, Colorado 80217
1-800-441-2447
303-675-2140
Fax: 303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com

Document Number: VOIPWP
REV 0

freescale semiconductor