Automotive Infotainment Cookbook User's Guide

Note the following details of the code protection feature on Microchip devices:

- Microchip products meet the specification contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is one of the most secure families of its kind on the market today, when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods used to breach the code protection feature. All of these methods, to our knowledge, require using the Microchip products in a manner outside the operating specifications contained in Microchip's Data Sheets. Most likely, the person doing so is engaged in theft of intellectual property.
- Microchip is willing to work with the customer who is concerned about the integrity of their code.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of their code. Code protection does not mean that we are guaranteeing the product as "unbreakable."

Code protection is constantly evolving. We at Microchip are committed to continuously improving the code protection features of our products. Attempts to break Microchip's code protection feature may be a violation of the Digital Millennium Copyright Act. If such acts allow unauthorized access to your software or other copyrighted work, you may have a right to sue for relief under that Act.

For information regarding Microchip's Quality Management Systems, please visit www.microchip.com/quality.

Trademarks

ISBN: 978-1-5224-5259-1

Table of Contents

NOTES:

Preface

NOTICE TO CUSTOMERS

All documentation becomes dated, and this manual is no exception. Microchip tools and documentation are constantly evolving to meet customer needs, so some actual dialogs and/or tool descriptions may differ from those in this document. Please refer to our web site (www.microchip.com) to obtain the latest documentation available.

Documents are identified with a "DS" number. This number is located on the bottom of each page, in front of the page number. The numbering convention for the DS number is "DSXXXXXA", where "XXXXX" is the document number and "A" is the revision level of the document.

INTRODUCTION

This chapter contains general information. Topics discussed in this chapter include:

- Document Layout
- Term and Abbreviations
- Term Definitions
- Recommended Reading
- Customer Support
- Revision History

DOCUMENT LAYOUT

This document is organized as follows:

- Chapter 1, Introduction – This chapter gives an overview on the Automotive Infotainment Cookbook.
- Chapter 2, Architectures – This chapter discusses architectural considerations for automotive infotainment systems and shows various ECU examples. It further shows the basic network architecture and explains the node types used in an INICnet™ technology 50utp network.
- Chapter 3, Hardware – This chapter discusses the basic ECU structure. It gives details about the network front-end circuitry, the INIC and the application.
- Chapter 4, Software – This chapter gives an overview on UNICENS™, low-level software support, network configuration and device-to-device communication examples.
- Chapter 5, Half-Duplex Diagnosis – This chapter explains how the half-duplex diagnosis is performed.
- Chapter 6, Use Cases – This chapter lists some application use-cases.
- List of Figures
- List of Tables
- Index

TERM AND ABBREVIATIONS

This document uses the following terms and abbreviations:

TERMS AND ABBREVIATIONS

Term/Abbreviation	Description
ADAS	Advanced Driver Assistance Systems
ADC	Analog-Digital Converter
AIC	Automotive Infotainment Cookbook
AMP	Audio Amplifier (only used in tables)
AFE	Analog Front End
ANC	Active Noise Cancellation
AV	Audio Video
AVB	Audio Video Bridging
DAC	Digital-Analog Converter
DC	Direct Current
DSP	Digital Signal Processor
DUT	Device Under Test
ECU	Electronic Control Unit
EMI	Electro-Magnetic Interference
ESD	Electrostatic Discharge
HU	Head Unit (only used in tables)
IPC	Instrument Panel Cluster
INIC	Intelligent Network Interface Controller
IP	Internet Protocol
MII	Media Independent Interface
PBE	Power Bridging Equipment
PCM	Pulse Code Modulation
PD	Powered Devices
PoDL	Power over Data Line
PoC	Proof of Concept
PoE	Power over Ethernet
RPC	Remote Procedure Call
PSE	Power Sourcing Equipment
RFI	Requests For Information
RFQ	Requests For Quotation
RPM	Revolutions Per Minute
SoC	System-on-a-Chip
STP	Switch-To-Power
TCCP	Trivial Control Channel Protocol
UNICENS	Unified Centralized Network Stack
UTP	Unshielded Twisted Pair

TERM DEFINITIONS

This document uses the following term definitions:

TERM DEFINITIONS

Term	Definition
bPHY	Balanced media physical layer, used for INICnet technology 50utp networks
DUT	ECU that has the DUT role during the half-duplex diagnosis phase.
INICnet technology 50utp network	INIC network with 50Mbps, using UTP
PHY	Physical layer
Remote Node	Name for any node in the UNICENS network that is not the Root Node. There is no differentiation between Slim Nodes and Smart Nodes. A Remote Node does not run the UNICENS management library.
Root Node	Node in a UNICENS network that controls the network (discover, configure, control, connect). This node is the only type of node which must run the UNICENS management library.
Slim Node	Remote Node in a UNICENS network that has no micro-controller. Its hardware consists of an INIC plus application-specific components such as an audio CODEC. A Slim Node is remote controlled by the Root Node; receiving and sending of Ethernet/IP data is not available.
Smart Node	Remote Node in a UNICENS network that has a micro-controller or SoC that provides local intelligence. For all communication except Ethernet/IP data, running the UNICENS protocol library is mandatory.
Network Descriptor	XML-based configuration file used to describe the network configuration, the nodes and the connections between them. It can be either edited text-based or by use of a graphical tool.
Tester	ECU that has the tester role during the half-duplex diagnosis phase.
UNICENS Daemon	Network management software that has the UNICENS management library integrated
UNICENS management library	Library that provides features to manage the startup, shutdown, error handling and configuration of the network and the nodes.
UNICENS protocol library	Sample code that provides mechanisms to perform ECU to ECU communication and ECU control, e.g., the TCCP.

RECOMMENDED READING

[1] Physical Layer Reference Design application note

[2] INIC Hardware data sheet

[3] INIC Power Management application note

[4] bPHY Conformity application note

To obtain documents, go to http://www.microchip.com/support and submit a technical support case (for details type “5951” into the search dialogue field).

CUSTOMER SUPPORT

Users of Microchip products can receive assistance through several channels:

- Distributor or Representative
- Local Sales Office
- Field Application Engineer (FAE)
- Technical Support

Customers should contact their distributor, representative or field application engineer (FAE) for support. Local sales offices are also available to help customers. A listing of sales offices and locations is included in the back of this document.

Technical support is available through the web site at:

http://www.microchip.com/support.

REVISION HISTORY

Revision C (November 2019)

Removed confidentiality status from document

Revision B (September 2019)

- General:
  - Restructured document
  - Updated document with links to GitHub resources
  - Term and Abbreviations: added new entries
  - Term Definitions: reworked and added new entries
  - Recommended Reading: added new entries; updated support address
- Chapter 1, Introduction: now considers INICnet technology and OS81216 INIC
- Chapter 2, Architectures
  - updated description
  - Figure 2-1, Figure 2-6-Figure 2-9: reworked
- Chapter 3, Hardware
  - Figure 3-1: reworked
  - Section 3.1.1, Network Front-End: reworked, including Figure 3-2 and Figure 3-3
  - Section 3.1.2, INIC: reworked
  - Section “Measurement Points” removed
  - Section “EHC” removed
  - Section “Application Logic” reworked and renamed to Application
- Chapter 4, Software
  - Section 4.1, UNICENS: reworked
  - Figure 4-5, Figure 4-8: reworked/updated
  - Section 4.4, Device-to-Device Communication: reworked; added new information and Figure 4-9
- Chapter 5, Half-Duplex Diagnosis: added
- Section 6.1, Example System With Audio Amplifier/Section 6.2, Example System Without Audio Amplifier (Cost-Down Solution): reworked figures and adjusted bandwidth values in tables
- Section 6.3, Example System With Digital Mic. Arrays (In-Car Communication)/ Section 6.4, Example System With Mic. Connector Modules (In-Car Communication): added

Revision A (January 2018)

This is the initial draft release of this document.

NOTES:

Chapter 1. Introduction

The Automotive Infotainment Cookbook (AIC) is a mnemonic device for all those charged with bringing networked infotainment and/or telematics features into a car model or a car platform. Once the feature set is defined, walking through this document helps to clearly communicate the requirements to potential Tier1s for requesting either Proof of Concepts (PoCs) or for issuing Requests For Information (RFI) or Requests For Quotation (RFQ).

Starting from architectural considerations, this document continues by having a look at specific focal points regarding hardware and software. It finally gives an overview on typical applications in the infotainment area.

This document describes networked systems using Microchip's INICnet technology that allows for transporting audio, video and Ethernet/IP over a single cable. INICnet technology spans the two lowest layers of the OSI model, where Microchip's Intelligent Network Interface Controllers (INICs) cover the Data Link Layer functionality. In addition to INICnet technology, the networked systems make use of Microchip's Unified Centralized Network Stack (UNICENS). In particular they utilize:

- OS81210, OS81212, OS81214, OS81216 INICs
- UNICENS Revision 2.x
- Physical layer of type Unshielded Twisted Pair (UTP)
- Speed grade of 50 Mbps

NOTES:

Chapter 2. Architectures

2.1 SYSTEM

2.1.1 Market Requirements

Under the hood of today's cars, we find diverse network architectures with various network technologies such as LIN, CAN, CAN-FD, Ethernet, INICnet technology and USB being deployed in the respective application domains.

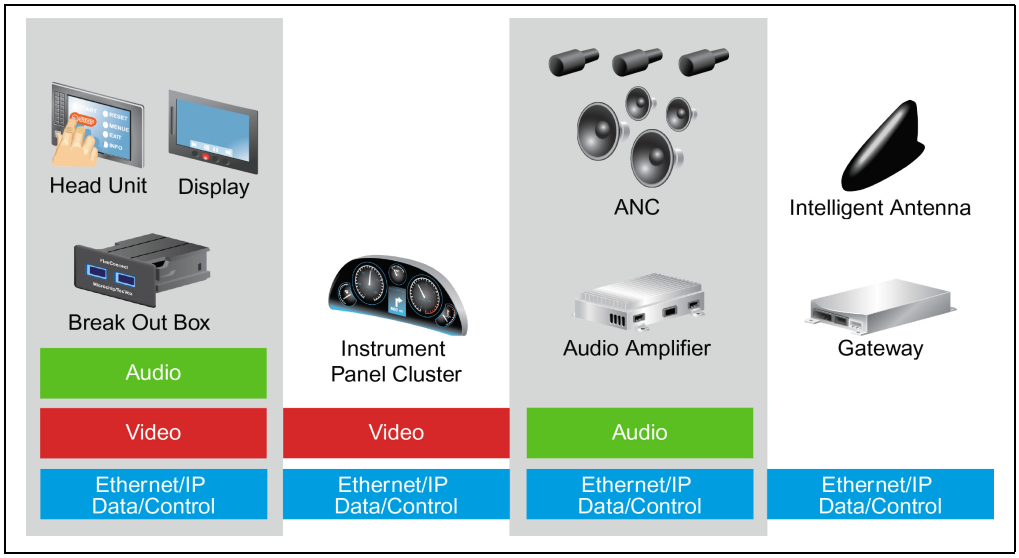
When traveling, passengers and drivers usually experience features based upon classic infotainment applications including digital audio and video transmission, internet access, telephone-based communication and navigation systems. These classic applications are complemented by advanced audio, acoustics applications using microphone arrays for Active Noise Cancellation (ANC), in-car communication, sound zones, hands-free phone enhancement and others.

For the classic infotainment applications, background services like gateways are required, which build bridges between the different in-vehicle networks and allow for information and communication to travel across all networks. Other applications provide more maintenance-related features such as a software update in the field.

For supporting driving and enhancing safety, there are Advanced Driver Assistance Systems (ADAS) that provide distributed sensors for park distance control, active cruise control and similar applications. Advanced camera applications e.g., for rear view, top view and image recognition complement the ADAS features.

Above applications are implemented in automotive grade hardware Electronic Control Units (ECUs). Figure 2-1 shows a selection of ECUs along with the data types their applications are handling typically.

FIGURE 2-1: EXAMPLE FOR AUTOMOTIVE GRADE HARDWARE ECUs AND SUPPORTED DATA TYPES



As shown in Figure 2-1, Ethernet/IP data and control data are used by all ECUs. Head Units (HUs), video displays and so-called break out boxes (“multi socket” kind of ECUs to connect smart phones or other non-automotive equipment) typically also process audio and video streams – for some applications even in real time.

Since classic Instrument Panel Clusters (IPCs) are meanwhile developed into continuously growing graphical displays, their way of operation has changed. The content presented to the driver is created in the Head Unit and from there transferred to the display. Hence, Instrument Panel Clusters mainly require video and data connections.

All audio-related applications handle audio streams of different complexity; classic Amplifiers are a typical example for this. Achieving a low and deterministic latency over the entire signal path from the source through transport and all processing stages is crucial for advanced audio, acoustic applications.

Getting a car connected with the outside world is a requirement that arises for example from today's habits of being online 24/7. Thus, simple antenna modules currently evolve to multi-purpose tuners, providing wireless access for all kinds of mobile consumer ECUs to the car infotainment. These applications mainly communicate via Ethernet/IP data, similar to gateway ECUs.

The increased interaction between all these in-car applications and the need for the entire system to share a limited number of user interfaces and resources require a network that provides the basis for the needed connectivity. From the market perspective it is essential to serve different target audiences. Hence, offering equipment variation is a must, which leads to an additional and very important requirement for in-car networks: Scalability.

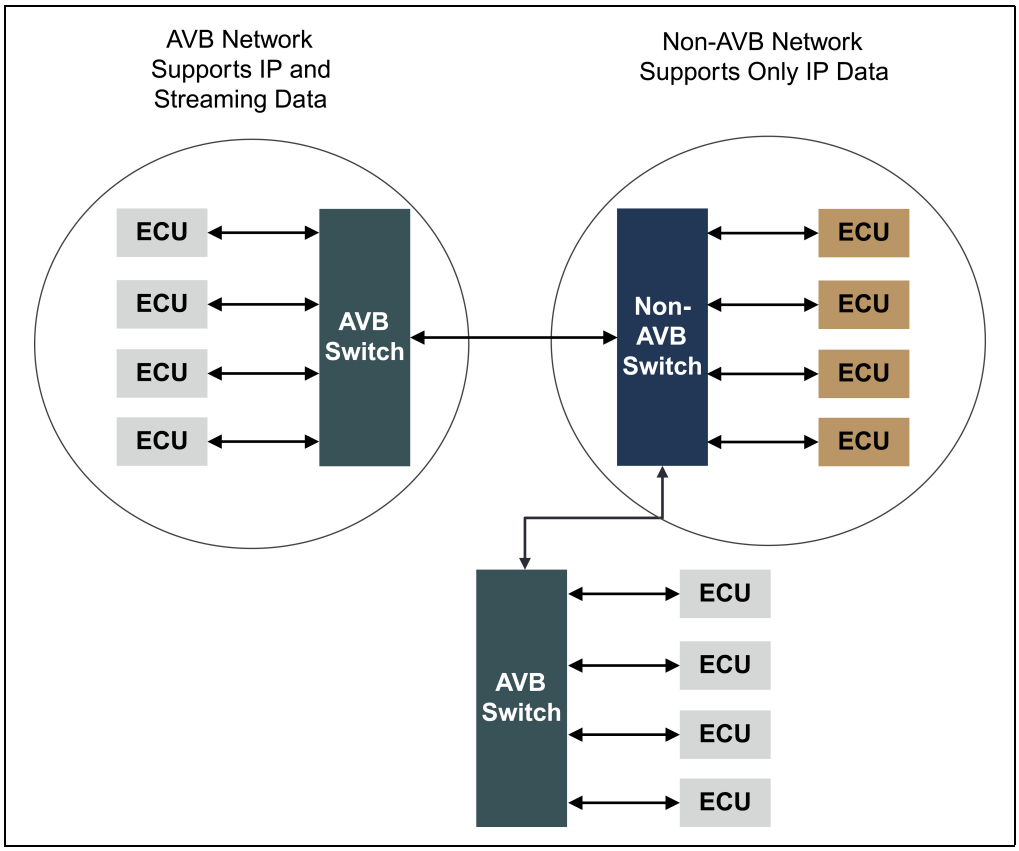
2.1.2 Typical Network Solution from Industry

The common network solution for automotive infotainment is currently based on Automotive Ethernet (100/1000BASE-T1) global standards in conjunction with TCP/IP protocol, extended by AVB (Audio Video Bridging IEEE 802.1). AVB is a collection of several standards needed to accomplish proper audio and video transport over Ethernet.

Ethernet/AVB supports IP data transport “by nature”. However, to fulfill the requirements for real time transport of streaming data, such as time synchronization and bandwidth reservation, this solution needs dedicated AVB-enabled switches and AVB software stacks.

By combining multiple switches, the system can accommodate multiple different types of networks or sub-systems. Some of these networks and their ECUs might be AVB enabled, because they transport and process audio and video data. Other networks might just support “classic” IP data, but are not AVB enabled.

FIGURE 2-2: AVB – NON-AVB NETWORKING



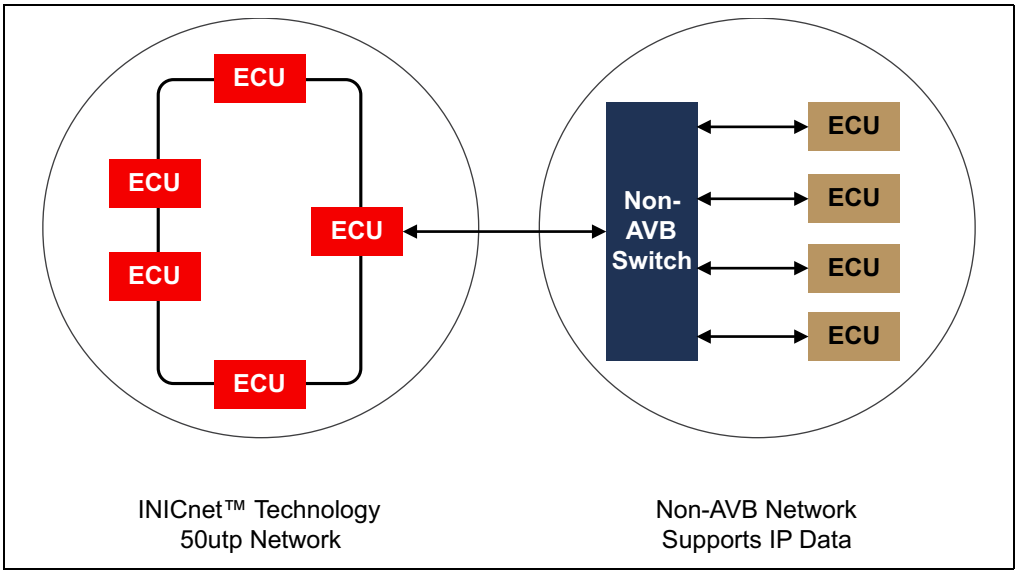
2.1.3 Microchip's Optimized INICnet Technology 50utp Network Solution

Microchip's INICnet technology 50utp network solution is well aligned to and optimized for the special requirements of real-time audio and video transport. In addition, this solution supports:

- Seamless Ethernet/IP data exchange throughout the entire system
- Ethernet/IP protocol can be used as the standard communication protocol
- Transport of streaming data (audio and video) and time-critical as well as latency-critical applications
- Switched Ethernet (non-AVB) and INICnet technology 50utp network topologies coexist where appropriate
- A simple software-based solution for configuring the INICnet technology 50utp network is available

Figure 2-3 shows an example for the approach that combines non-AVB Ethernet with Microchip's INICnet technology 50utp network. Bottom line, this approach reduces costs by removing the need for specific AVB-enabled switches. Although no AVB software stacks are present, timing-related requirements are fulfilled. For an infotainment system this solution is highly flexible and scalable, since several Ethernet networks and INICnet technology 50utp networks can be combined as needed.

FIGURE 2-3: INICnet TECHNOLOGY NETWORK AND ETHERNET NETWORK CO-EXISTENCE



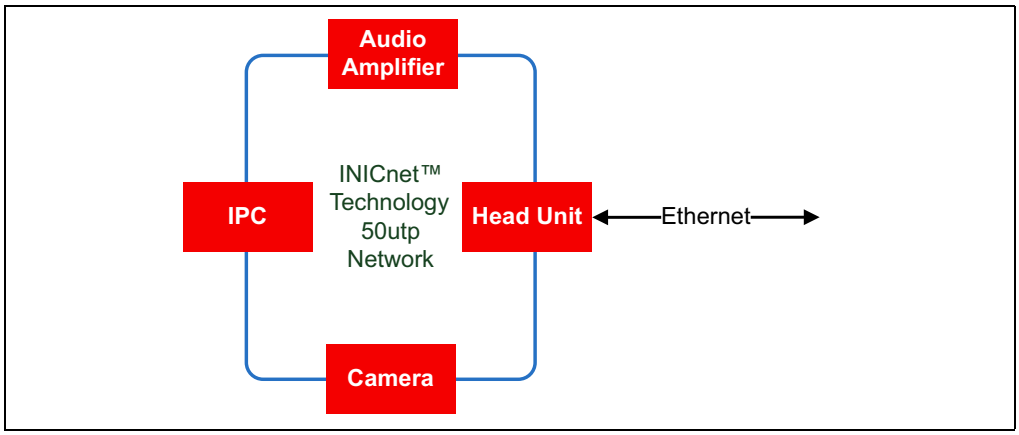
2.1.4 System Proposals

This section introduces two infotainment systems. While the first example is meant to be used as entry-level system, the second example aims at a more advanced mid-/high-level equipment variant.

2.1.4.1 ENTRY-LEVEL EQUIPMENT VARIANT

The entry-level proposal assumes that all features can be distributed among four ECUs. Although the Head Unit could also incorporate an Audio Amplifier (AMP), this example uses an external Audio Amplifier to allow for more variants, see Figure 2-4.

FIGURE 2-4: SYSTEM PROPOSAL – ENTRY-LEVEL VARIANT



The ECUs are as follows:

Head Unit:

Controls the entire system and also provides Ethernet access to all ECUs, e.g., for internet access or firmware update. It incorporates the main infotainment functionality, and sources audio/video data streams. The Head Unit has a digital video display (may be a touchscreen) for control and display purposes.

Instrument Panel Cluster:

Processes Ethernet/IP data to display speedometer and odometer data, basic navigational information, plus vehicle status information.

Audio Amplifier:

Handles the output of all audio streams.

Camera (rear view):

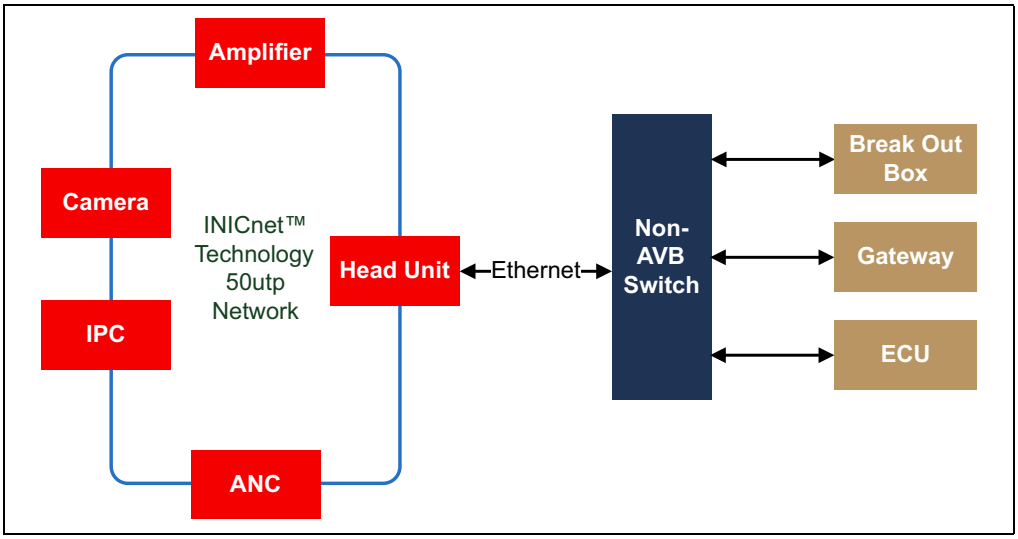
Sources a low-latency video stream to be displayed on the Head Unit or Instrument Panel Cluster.

This entry-level system can be implemented by using Microchip's INICnet technology 50utp network, which supports all data types without the requirement for an Ethernet switch.

2.1.4.2 MID-/HIGH-LEVEL EQUIPMENT VARIANT

The mid-/high-level system proposal benefits from Microchip's INICnet technology 50utp network as described in Section 2.1.3. As depicted in Figure 2-5, the ECUs that send and receive audio and video data streams plus Ethernet/IP data are grouped together in an INICnet technology 50utp network. This part of the mid-/high-level system proposal follows the principles already shown in Section 2.1.4.1.

FIGURE 2-5: SYSTEM PROPOSAL – MID-/HIGH-LEVEL VARIANT



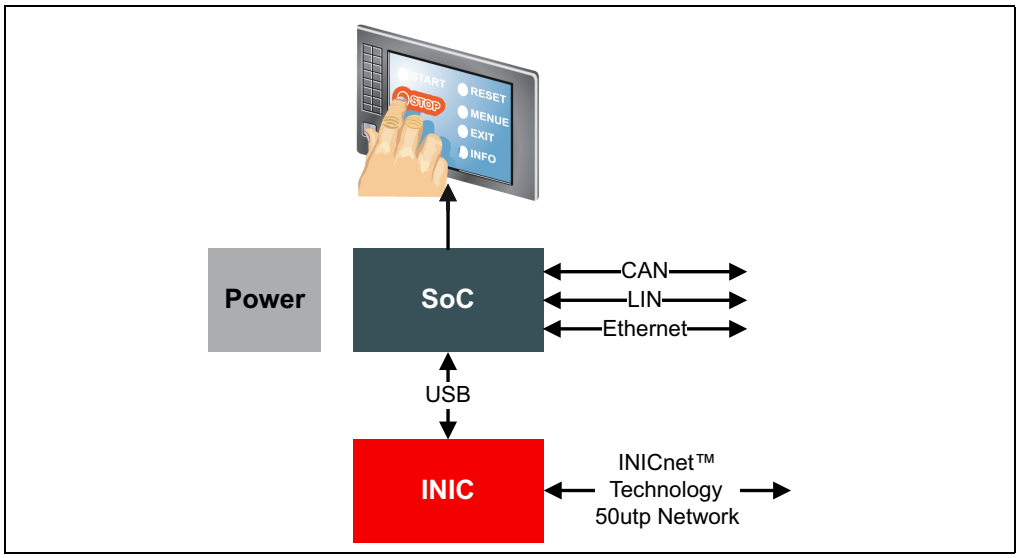
By using a standard non-AVB Ethernet connection, a standard non-AVB switch is connected to the Head Unit. This switch is the basis for the non-AVB network that typically groups together ECUs and gateways such as CAN/LIN as well as break out boxes, etc. This also provides for the required scalability.

2.2 ECU EXAMPLES

Each ECU, which is connected to an INICnet technology 50utp network, uses one of Microchip's INICs. The INIC takes care of all low-level network management tasks and accomplishes very flexible data routing for all data types between the network and one or more of its standard interfaces, such as I^2C, I^2S, SPI, MediaLB® and USB.

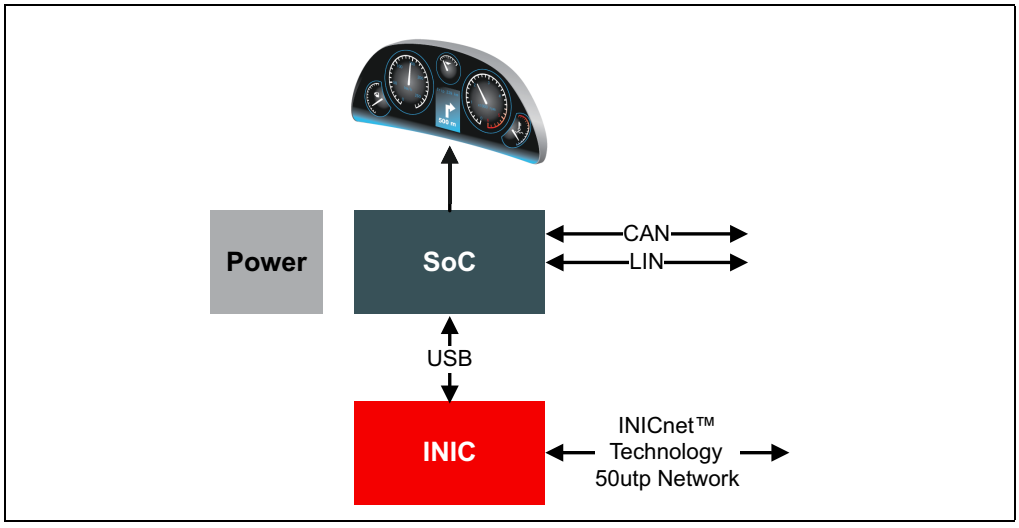
A Head Unit typically processes data and signals coming from CAN/LIN sub systems, but also via non-AVB automotive Ethernet connections. The results of the data processing will either be displayed on the screen of the Head Unit or might go back as commands to the subsystems connected to the System-on-a-Chip (SoC). The SoC is hooked up also to the INICnet technology 50utp network via USB and an INIC, thus having access to all data in the system.

FIGURE 2-6: ECU EXAMPLE – HEAD UNIT



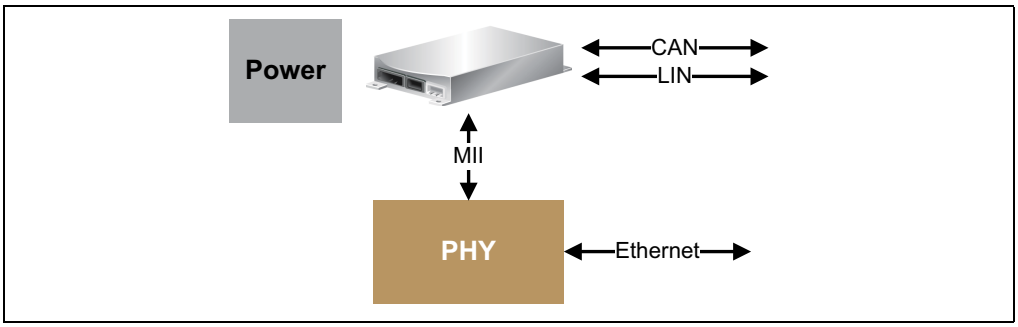
The internal connections in an Instrument Panel Cluster can be realized in a similar way as done in a Head Unit, however, there is not always a need for Ethernet connections.

FIGURE 2-7: ECU EXAMPLE – INSTRUMENT PANEL CLUSTER



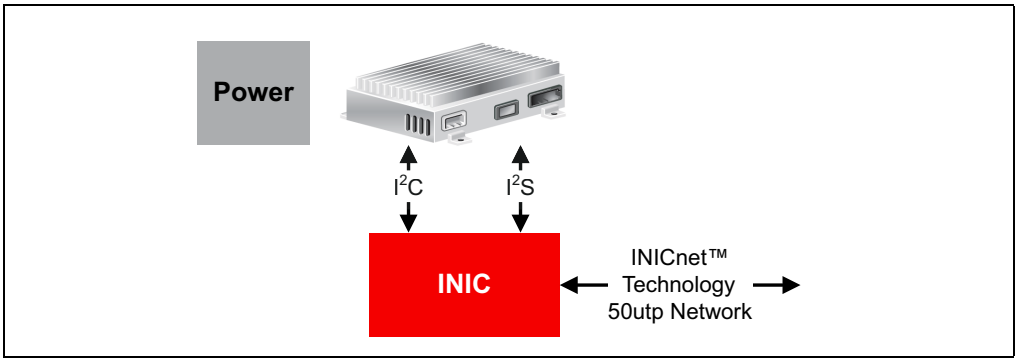
Looking at a “pure” gateway ECU, there is typically no need to connect it to the INICnet technology 50utp network, because it does not process audio or video data coming from there. Instead, CAN/LIN/Ethernet connections and the respective data are playing an important role. In this case, the SoC is connected directly via a Media Independent Interface (MII) to the Ethernet-PHY.

FIGURE 2-8: ECU EXAMPLE – GATEWAY



On the other hand, there is typically no data originating from CAN/LIN/Ethernet, which is processed by a standard Audio Amplifier. However, this ECU needs connection to the INICnet technology 50utp network for the low-latency exchange of audio data. A Digital Signal Processor (DSP) might be connected to the INIC via I^2C and I^2S.

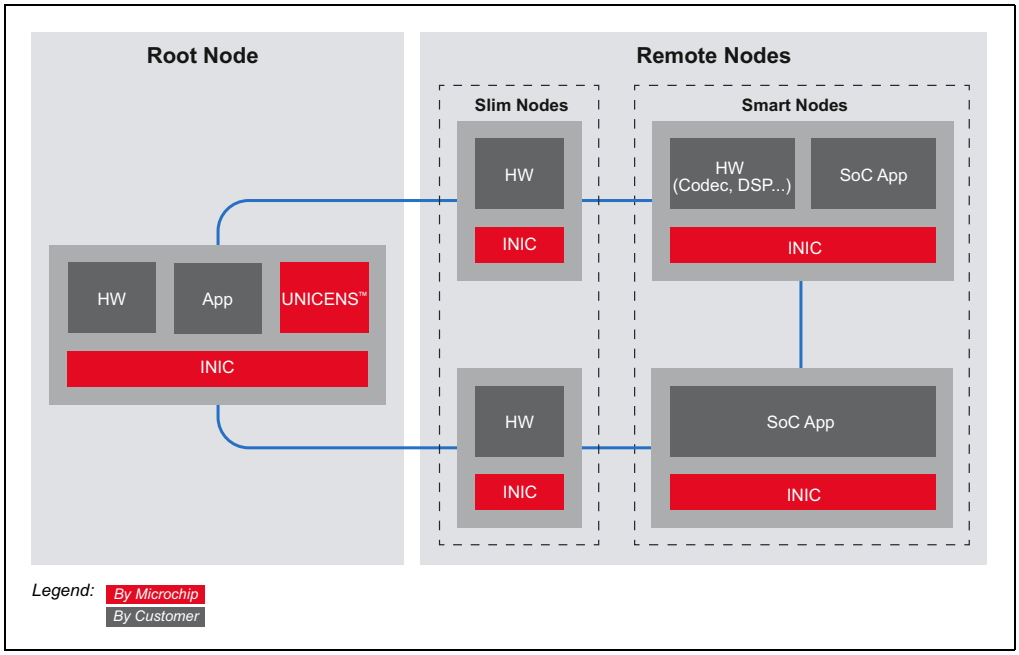
FIGURE 2-9: ECU EXAMPLE – AUDIO AMPLIFIER



2.3 BASIC NETWORK ARCHITECTURE

This section discusses the node types used in an INICnet technology 50utp network.

FIGURE 2-10: EXAMPLE – BASIC NETWORK ARCHITECTURE



The kinds of nodes shown in Figure 2-10 are explained below:

- Root Node – Node in the INICnet technology 50utp network that is required to implement the UNICENS management library. With INIC and UNICENS integrated on this node, network discovery, configuration and connections can occur.
- Remote Node – This node refers to any ECU in the INICnet technology 50utp network, other than the Root Node. It does not implement the UNICENS management library. Remote Nodes can be either a Slim Node or Smart Node.
  - Slim Node – A Remote Node that integrates INIC and application-specific components (e.g., audio CODEC, DAC); however, it contains no local intelligence (no micro-controller and software). It does not run UNICENS and is remote-controlled by the Root Node. Slim Nodes can be very lean ECUs, such as small microphones in an ANC system.
  - Smart Node – A Remote Node that has both INIC and additional local processing power (e.g., SoC, µC), which provide local intelligence (this means they have their own micro-controller to run customer applications, such as video displays, gateways).

2.4 IMPLEMENTATION CONSIDERATIONS

A network which is managed by UNICENS can be implemented on any operating system. Root Node and Smart Nodes can run on different operating systems. However, there are certain advantages when using Linux® as a basis for the Root Node as well as for Smart Nodes, such as:

- Microchip's example code has a comprehensive feature set.
- The provided kernel driver makes use of the existing infrastructure offered by a usual Linux system such as ALSA and networking interface. In addition, it offers standard interfaces for applications, see Section 4.2.
- It has an extensive customer basis.
- It is the most comfortable solution in combination with MPLAB® Network Creator.
- UNICENS Daemon is available as example integration.

The UNICENS Daemon takes care of all network management tasks, including:

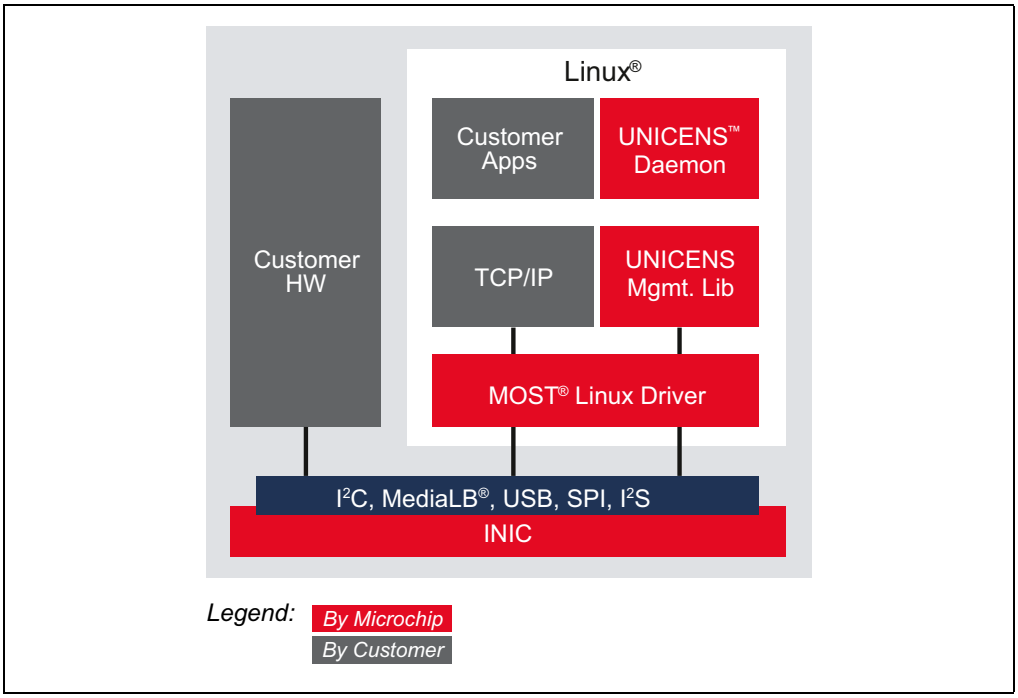
- evaluation of the Network Descriptor,
- node discovery,
- device configuration and control and
- bandwidth reservation and connection management for audio and video streams.

Software drivers are available for selected operating systems in order to eliminate integration efforts, see Section 4.2.

If the Root Node does not run Linux or another supported operating system, customers need to integrate the UNICENS management library by themselves into their application and must implement the above mentioned tasks.

Additional customer applications can simply use the built-in Linux TCP/IP stack for the transmission of Ethernet/IP data via the packet channel of the INICnet technology 50utp network.

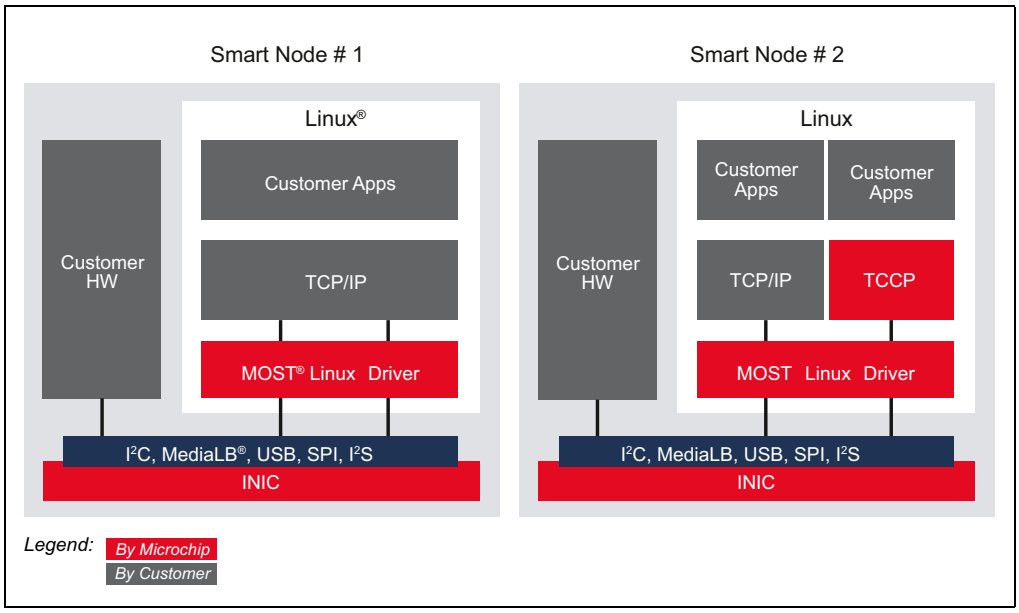
FIGURE 2-11: UNICENS ROOT NODE



When it comes to the communication between Smart Nodes, or between Smart Nodes and Root Node, there are the following options:

- Customer applications as shown below in Smart Node #1 can use TCP/IP for the transmission of Ethernet/IP data via the INICnet technology 50utp network, similarly as in the Root Node.
- Additionally, as shown in Smart Node #2, a customer application can use the Microchip supplied Trivial Control Channel Protocol (TCCP) library or any other Remote Procedure Call (RPC) technique. This enables control communication over USB or I^2C interface with the INIC and the INICnet technology 50utp network, without using a TCP/IP protocol stack.

Figure 2-12: UNICENS Smart Nodes Examples



In a Slim Node there exists no micro-controller, thus no TCP/IP stack, no UNICENS protocol library and no application are running. A Slim Node communicates simply via the INIC firmware functions with the Root Node to report any state changes or user interactions, e.g., when the user presses a button on the Head Unit. Moreover, a Slim Node can receive and send streaming data, which is configured by the Root Node. To configure the peripheral hardware on a Slim node, remote I^2C is available.

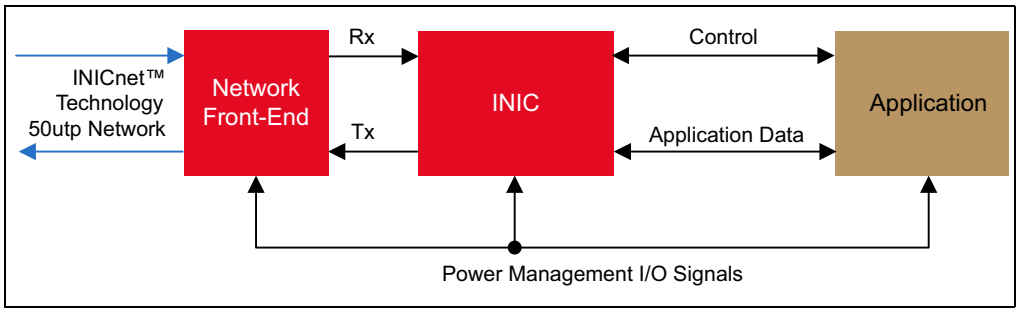
NOTES:

Chapter 3. Hardware

3.1 INIC-BASED ECU

The fundamental hardware architecture of an ECU, using an INIC for the network interface, is provided in Figure 3-1. This figure illustrates a generic ECU with basic connections between logical areas (actual implementations may vary from this diagram).

FIGURE 3-1: MAIN COMPONENTS OF AN INIC-BASED ECU



The logical areas and/or hardware blocks of the ECU are:

- Network front-end (passive): Interface between INIC and network physical layer
- INIC: Intelligent Network Interface Controller that manages all the lower software layers of the network protocol
- Application, which can consist of:
  - application hardware (codec, DSP, etc.)
  - microcontroller
  - power management (for details refer to the INIC Power Management application note [3].

3.1.1 Network Front-End

The network front-end is the circuitry between the INIC and the connector at the edge of the board. This is also known as the Analog Front End or AFE and consist of network termination, ESD protection and EMI mitigation.

3.1.1.1 INTERFACE TO INIC

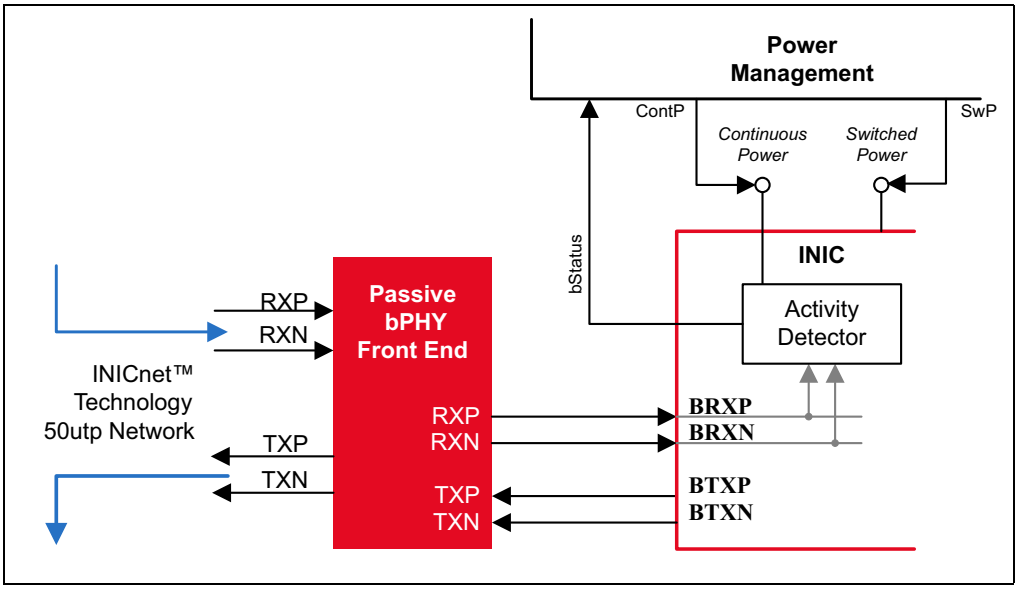
The INIC integrates a differential interface to the network front-end circuitry, supporting a balanced media physical layer (bPHY) interface, which is optimized for UTP copper wire.

The INIC bPHY interface consists of:

- BTXP – Positive transmitter output
- BTXN – Negative transmitter output
- BRXP – Positive receiver input
- BRXN – Negative receiver input

An internal network activity detection circuit monitors activity on the receiver. When waking from network activity is a desired feature of the network node, the internal INIC activity detection circuit must be powered by a continuous supply, as illustrated in Figure 3-2.

FIGURE 3-2: INIC ACTIVITY DETECTION CIRCUITRY



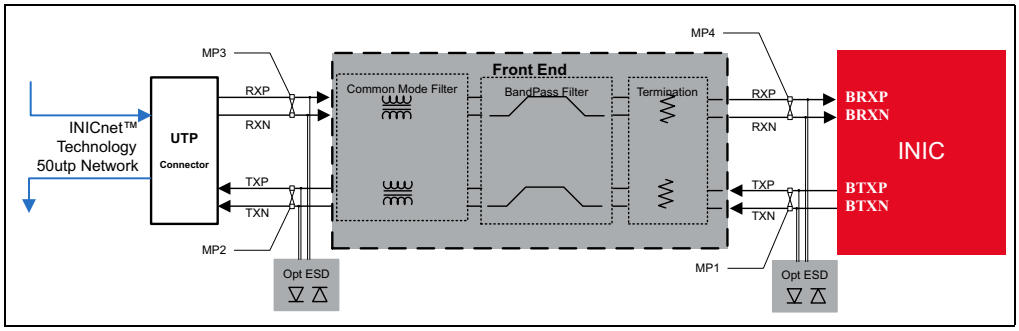
3.1.1.2 FRONT-END CIRCUITRY

The AFE is defined by all the components between the INIC bPHY pins and the connector at the edge of the board. The purpose of this circuitry is to provide the required network termination for EMI and ESD protection. A simplified block diagram of the AFE is shown below.

The termination resistors are located near the bPHY pins on the INIC. This is followed by a bandpass filter, which rejects out of band noise and provides DC blocking. A common mode filter is used to reduce electromagnetic emissions and increase the overall system immunity to EMI. Optional ESD protection may be needed either near the INIC pins or the bPHY connector or both, depending on the OEM ESD requirements.

Differential measurement points near the connector as shown by MP2 and MP3 below may be required by the OEM for conformity. It is highly recommended to include them even if they are not required. It is also recommended to include differential measurement points near the Rx and Tx pins of INIC as shown by MP1 and MP4 below. For detailed information refer to the bPHY Conformity application note [4].

FIGURE 3-3: FRONT-END CIRCUITRY



Microchip provides reference schematics that may be appropriate for certain applications. Contact Microchip for the Physical Layer Reference Design application note [1] for the INIC of interest, which describes the front-end circuitry implementation in greater detail.

3.1.1.3 NETWORK, CONNECTORS AND CABLES

Microchip does not require, nor recommend, any specific connector or cable for INICnet technology 50utp network applications. The bPHY interface is designed to support unshielded twisted pair (UTP) cable with a nominal characteristic impedance of 90-140 Ω. The maximum link length is 15 m, where the link is defined as all the cable segments, couplers and connectors between the Tx and Rx interfaces of two nodes. The number of interconnects is not specified, only that each link shall meet the attenuation requirements shown in Table 3-1.

TABLE 3-1: CABLE AND NETWORK REQUIREMENTS

Cable Requirements	Value
Characteristic impedance:	100 Ω -10/+40 Ω
Attenuation, max.:	-1 dB at 1 MHz -2.6 dB at 10 MHz -4.9 dB at 33 MHz -7.2 dB at 66 MHz
Length, max.:	15 m
Network Requirements	**Value**
Number of network ECUs, max.:	20

The cable characteristics above allow the use of a variety of automotive cables and connectors in use today, including those used for CAN FD, INICnet technology 50utp, FlexRay™ and 100Base-T1 Automotive Ethernet.

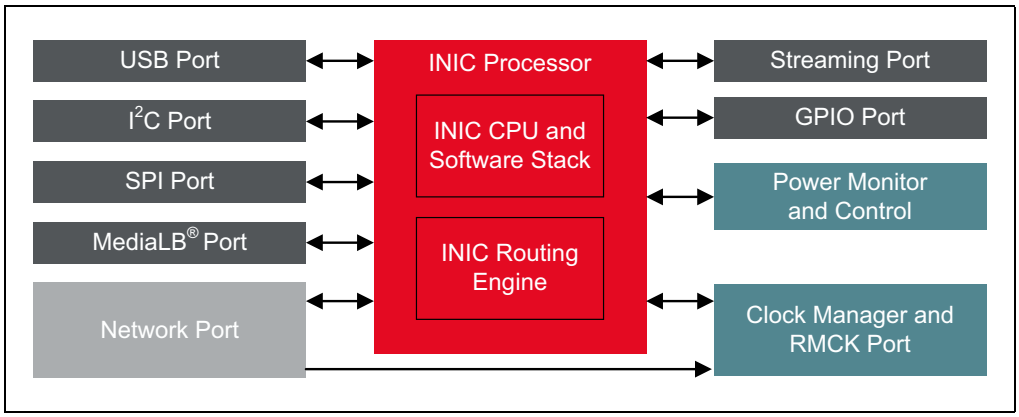
3.1.2 INIC

The INIC encapsulates the functionality required to implement a high speed synchronous network capable of transferring audio, video, Ethernet and control data amongst other INIC-based ECUs on the network. INIC manages network startup. Managing network startup on chip provides a quick and stable network startup and helps protect network operation from errant application software.

The INIC architecture uses a message-based control interface instead of the register-based interface used with traditional network PHYs.

The INIC block diagram below shows that INIC is a device that bridges standard data-transfer interfaces of an application to the INICnet technology 50utp network, and from there back to the standard application interfaces. For example, Ethernet packets can come into an INIC via USB in one ECU, go across the network and go out the SPI Port of an INIC in another ECU.

FIGURE 3-4: INIC BLOCK DIAGRAM



The interfaces colorized in dark gray are typical industry-standard interfaces for CPUs and audio CODECs. See the relevant INIC data sheet [2] for details on connecting these standard interfaces. The Network Port is unique to INIC and is the port used to connect INICs together to form an INICnet technology 50utp network. The clock manager block is used both internally and externally to synchronize all the INICs, and to synchronize the application's audio to the network.

Note: Not all INIC derivatives contain every block shown in Figure 3-4.

3.1.2.1 NETWORK PORT

The Network Port in combination with the network front-end is the interface used to connect INICs together to form an INICnet technology 50utp network. The interface and front-end create a differential balanced media physical layer (bPHY). INIC sends data to this interface in frames at a rate of 48,000 frames/s. INICnet technology 50utp devices use a frame size of 128 bytes yielding a data rate of 128 bytes/frame * 48,000 frames/s * 8 bits/byte = 49.152 Mbits/s or ~50Mbits/s. The integrated Network Port supports the electrical transmission of this data between two or more compatible network devices using low-cost UTP wiring.

3.1.2.2 CONFIGURATION OF INIC

The configuration of an INICnet technology 50utp network based on UNICENS is done via the network, initiated by the Root Node. For this, a local control interface over I^2C, MediaLB, SPI or USB 2.0 is supported.

Table 3-2 shows which control interfaces are available on INIC derivatives.

TABLE 3-2: SUPPORTED CONTROL INTERFACES

INIC Part #	I^2C Port	SPI	MediaLB 3-Pin	USB 2.0
OS81210	X	X	X	X
OS81212	X	X	X	—
OS81214	X	X	—	—
OS81216	X	—	—	—

3.1.2.3 RESET

It must be made sure that the INIC is brought out of reset after the power supplies have stabilized. An application reset must not affect the INIC. Therefore, if an application micro-controller is used, the INIC's **RESET** pin must be high impedance during micro-controller reset.

3.1.3 Application

As shown in Figure 3-1, an ECU consists of a network front-end, an INIC and the application. The application is that portion of the device, which is needed for entire device functionality.

An application can include a:

- Streaming data source, such as a DSP, codec, wireless module, microphone or ADC. It streams data to the network via the INIC. The streaming data source is connected over I^2S, USB or MediaLB, see also Table 3-3.
- Streaming data sink, such as DSP, codec, wireless module, amplifier or DAC. It receives data from the network via the INIC. The streaming data sink is connected over I^2S, USB or MediaLB, see also Table 3-3.
- Device-to-device communication handler (SOME/IP, TCP/IP...). It manages the used protocols and induces the proper device function. In case device-to-device communication needs to be handled, an application micro-controller is needed, which is connected to the INIC over I^2C, USB, SPI or MediaLB.
- Power management that consists of hardware or is realized through an additional power controller. It controls the power supply of a network device or the complete network and handles different power states if needed. For details on the INIC's power management refer to the INIC Power Management application note [3].

Table 3-3 gives an overview of INIC hardware interfaces and the data types they support:

TABLE 3-3: HARDWARE INTERFACES AND SUPPORTED DATA TYPES

Hardware Port	Control	Packet	Sync	AVPacketized	QoSPacket	DiscFramePhase
Network	X	X	X	X	X	X
MediaLB 3-Pin	X	X	X	X	X	X
Streaming	—	—	X	—	—	—
SPI	X	X	—	—	—	—
I^2C	X	—	—	—	—	—
USB	X	X	X	X	—	—

3.2 DESIGN REVIEWS

Suppliers should take advantage of the free hardware design review service that Microchip offers. An applications engineer will check both schematics and layout to ensure that the recommended design guidelines are followed.

NOTES:

Chapter 4. Software

Current car infotainment systems often follow a decentralized approach. In Ethernet-based systems a micro-controller is needed on each node, running a TCP/IP stack and application software on top to communicate with other nodes. To satisfy low-latency requirements an additional AVB-stack needs to be implemented.

Decentralized systems are very powerful and can be exactly tailored to customer needs. However, their development requires a lot of in-depth knowledge on lower communication layers as well as on higher layers. It must be also taken into account that specification, development and implementation of network- and connection management functionality for non-standardized decentralized systems are drivers for extended development time and high costs.

4.1 UNICENS

UNICENS is a new technology that separates application development and network management by defining a standardized abstraction layer between low-level hardware-specific software and applications. This implies that engineers can fully concentrate on their applications without knowing anything about the underlying physical layer. Network configuration can be done in a simplified way in a minimum of time. System engineers from OEMs and Tier1s can easily configure automotive networks.

The UNICENS management library is mandatory in the Root Node only. After starting-up the INICnet technology 50utp network, the library

- discovers all available nodes,
- performs centralized network management,
- allocates bandwidth for streaming data connections on the network and
- connects audio/video sources and sink devices (as configured in a central configuration file/structure).

The UNICENS management library is available[1] free of charge in C/C++ source code to be integrated into the customer's applications. In addition, the software is also available free of charge as compilable Android™, Linux or QNX® application, called UNICENS Daemon[2]. UNICENS Daemon is the network management software that has the UNICENS management library already integrated.

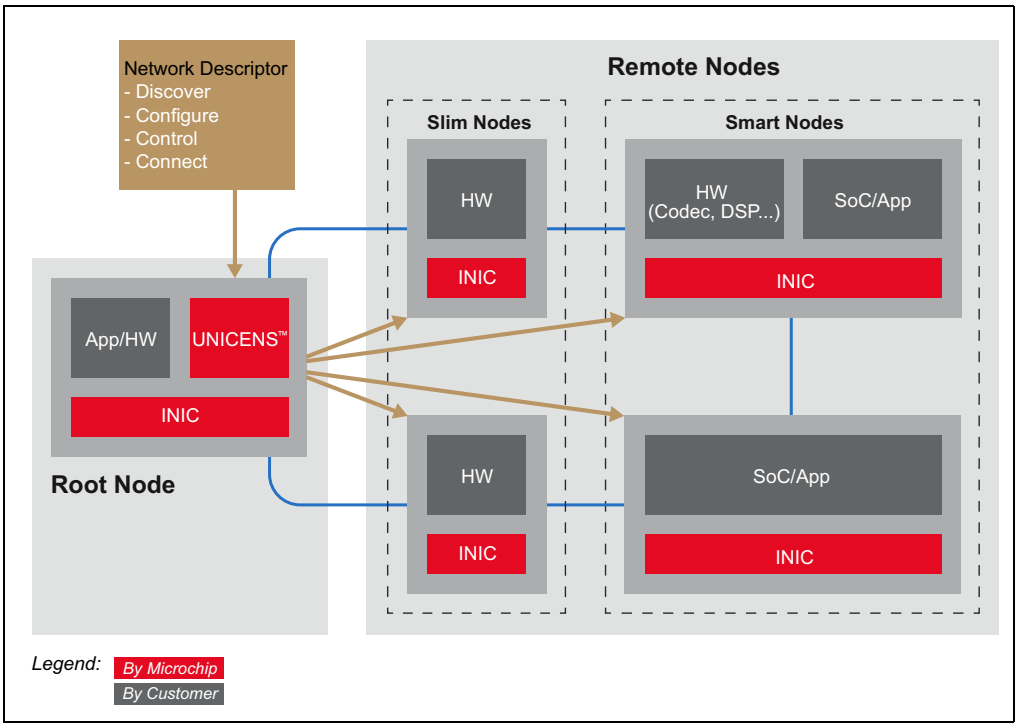
The Network Descriptor contains a list of commands to configure each node in the system (Root Node and Remote Nodes) including special scripts that can configure peripherals on Remote Nodes via I^2C. The Network Descriptor can be a superset of all possible network configurations that may cover a range of system options such as base-, standard- and premium configurations. The UNICENS daemon will inform the

1. Available for free download on GitHub: https://github.com/MicrochipTech/unicens
2. Available for free download on GitHub: https://github.com/MicrochipTech/unicens-linux-daemon

Root Node application of only the resources actually found in the network. This means that it is not necessary to create independent Network Descriptors for every system configuration that is possible.

The hardware configuration of the Remote Nodes and the setup of AV connections between the nodes is done by UNICENS, which evaluates the Network Descriptor.

FIGURE 4-1: UNICENS NETWORK



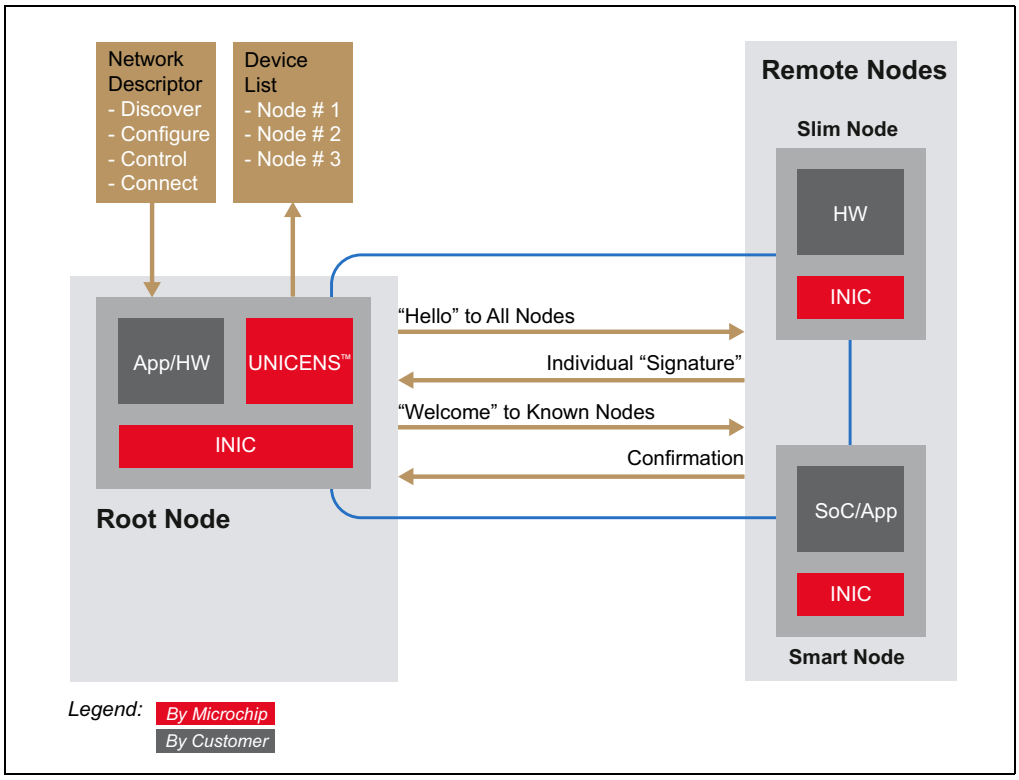
After network startup, UNICENS performs node discovery to compare the actual list of nodes with the Network Descriptor. Typically, node discovery is done every 5 seconds. Unknown nodes will be ignored by UNICENS.

Node discovery is done as follows:

1. UNICENS sends a “Hello” message to all nodes, see Figure 4-2.
2. Each node responds with its individual “Signature”.
3. Each node receives an individual “Welcome” from UNICENS, followed by a confirmation of the node.

Remote Nodes are configured and controlled according to the Network Descriptor.

FIGURE 4-2: NODE DISCOVERY



4.1.1 ANC System Example

Figure 4-3 shows an example of a special-purpose subsystem, which is used for advanced audio, acoustics applications.

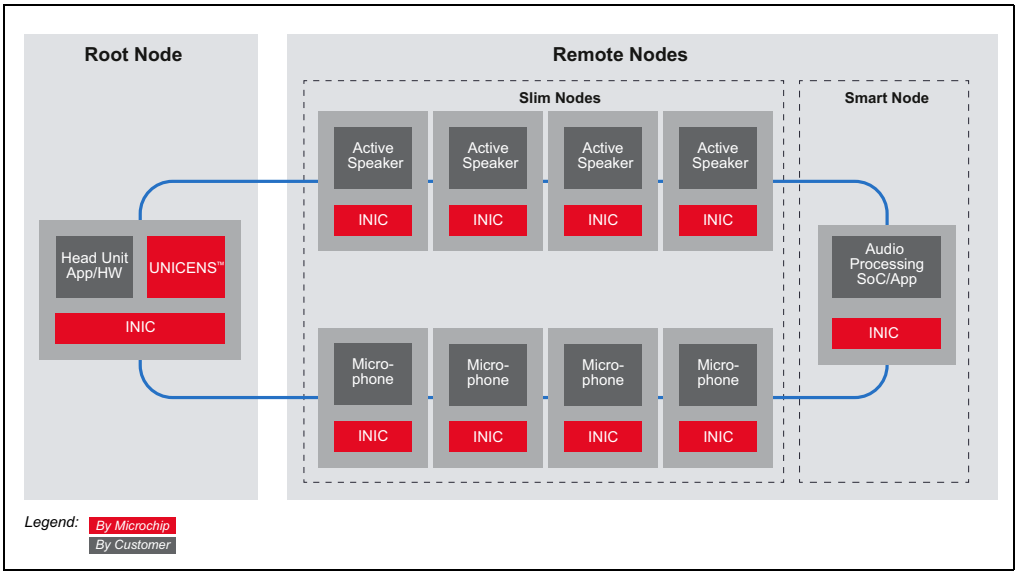
The INICnet technology 50utp network connects several Microphones and active Speakers, which could all be realized as cost-effective Slim Nodes, with a central Head Unit – the Root Node – and a so-called audio processing device, which is a Smart Node with own DSPs and application.

Network configuration, bandwidth reservation and connection management are very simple in this case and could even be realized in a static way in the Network Descriptor:

- Four audio streams, one going from each Microphone as an input to the audio processing device, and
- Four audio streams originating as an output of the audio processing device, going to the four active Speakers.

To select and control the audio application mode, e.g., pure ANC (fast counter wave generation and transport) or enhanced in-car communication (mixing one of the Microphone inputs to another seat's active Speaker output), the Head Unit can send commands to the audio processing device via the INICnet technology 50utp network.

FIGURE 4-3: ANC SYSTEM



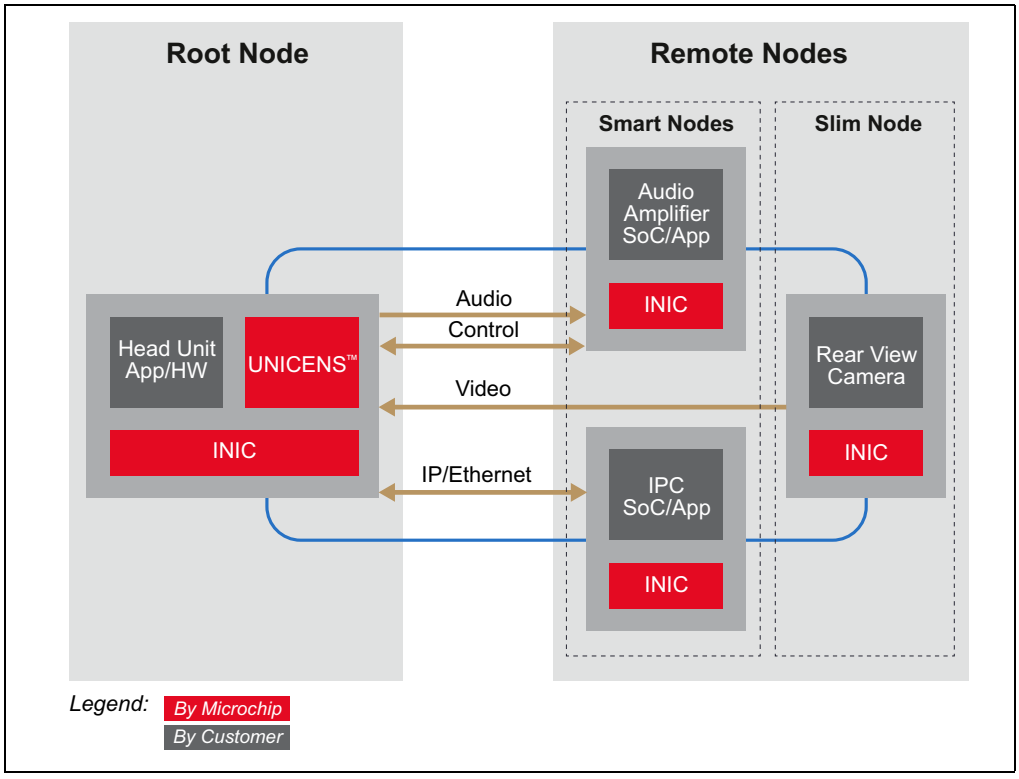
4.1.2 Simple System Example

The example shown in Figure 4-4 takes up the entry-level variant shown in Figure 2-4 and discusses some possible use-cases, which are:

- Display the picture of the rear view Camera on the Head Unit's display: This requires a (static) video connection from the Camera (Slim Node) to the Head Unit (Root Node).
- Listen to music: The sound is sourced by the Head Unit and received by the Audio Amplifier – a Smart Node – which has its own audio processing capabilities and control functions.
- Display driving information: Information is received via Ethernet/IP data from the Head Unit and displayed on the Instrument Panel Cluster screen. Also, the Instrument Panel Cluster device is a Smart Node having its own processor and application.

Note that all kinds of data including audio, video, Ethernet/IP and control data, are transmitted over the same single physical channel of the INICnet technology 50utp network. This saves the need for additional cabling and connectors compared to traditional wiring, which might use CAN for control, analog lines for audio, LVDS for video, etc.

FIGURE 4-4: SIMPLE SYSTEM



4.2 LOW-LEVEL SOFTWARE SUPPORT

Before UNICENS can configure an INIC and the application can use it, the system requires low-level software that supports them. Microchip has developed a ready-to-use solution that helps in minimizing integration effort.

4.2.1 Drivers

When Root Node or Smart Nodes are running an operating system, a suitable device driver is usually required. Drivers are currently available for Android, Linux, QNX, real-time operating systems and main loop architectures. They provide support for the INIC-net technology 50utp network almost out of the box and allow starting a project from scratch within hours.

Microchip's drivers for the INIC have been developed to enable a seamless integration of the INICnet technology 50utp network technology in these operating systems. They provide a strong abstraction layer between the physical network (technology specific) and the applications (technology independent).

FIGURE 4-5: TECHNOLOGY INDEPENDENT APPLICATIONS – EXAMPLE

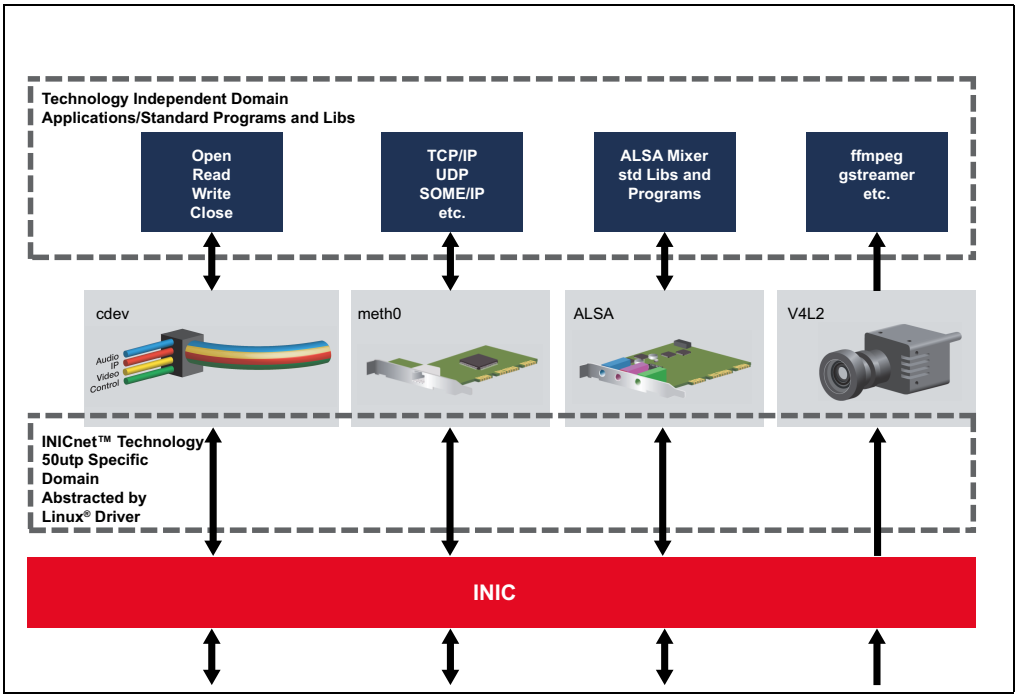


Figure 4-5 shows how the drivers, developed for an INICnet technology 50utp network, are completely abstracting the application from the underlying network technology by presenting the INICnet technology 50utp network to the application with using standard mechanisms and interfaces that are natively provided by the operating system.

4.2.2 Bare-Metal

When no operating system is used, it is not possible to use a standard driver. For such situations, Microchip has developed stand-alone low-level support code[1], written in pure C/C++. This bare-metal solution can be integrated very easily into the customers' own C/C++ code to provide full support for the INICnet technology 50utp network.

1. Available for free download on GitHub: https://github.com/MicrochipTech/unicens-bare-metal-sam-v71

4.3 NETWORK CONFIGURATION

After starting-up the INICnet technology 50utp network, UNICENS discovers all available nodes and performs centralized network management tasks as configured in the Network Descriptor. The Network Descriptor defines the:

- configuration for all hardware of the Remote Nodes, especially for the Slim Nodes,
- configuration of all remote INICs,
- bandwidth allocation on the network for streaming data and
- audio and video connections between nodes.

The Remote Nodes and their applications simply profit from the bandwidth reservation and connection management for all AV streams which have been accomplished by UNICENS in a standardized and reliable manner.

4.3.1 Network Descriptor Example

Figure 4-6 shows a simple example that focuses on a small system having only two devices, a Root Node and an Auxiliary I/O device, realized as Slim Node. Figure 4-7 shows the Network Descriptor in XML-file format as typically used by Microchip's UNICENS Daemon.

FIGURE 4-6: AUDIO EXAMPLE

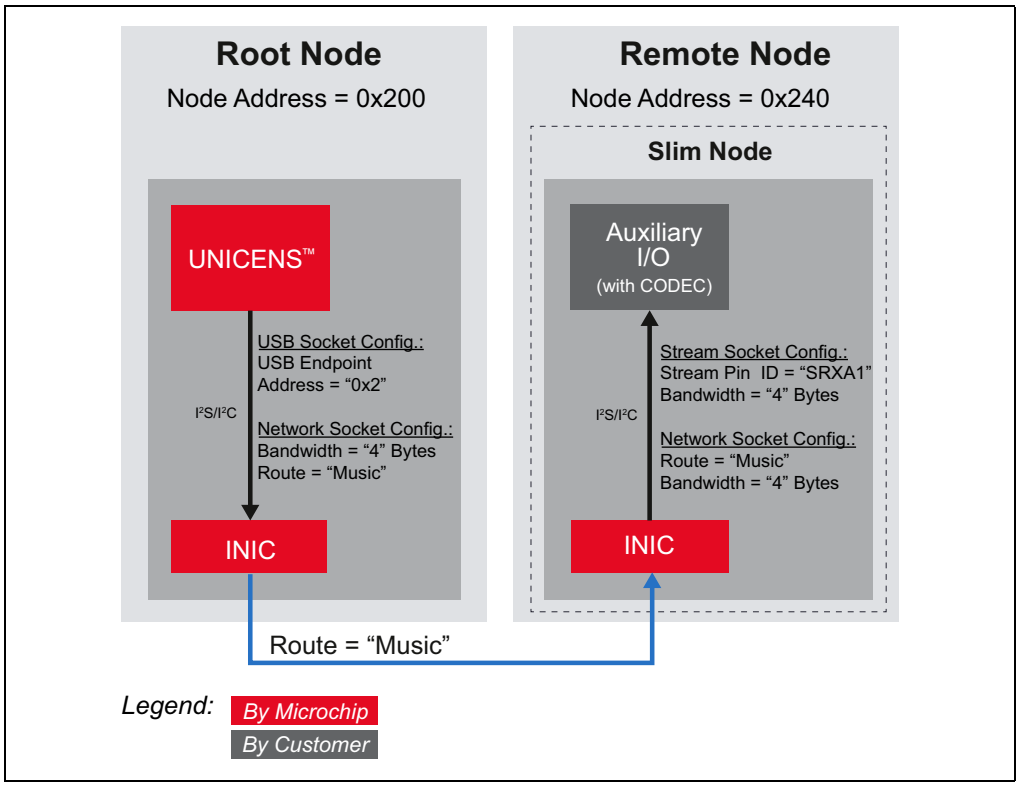


FIGURE 4-7: XML EXAMPLE

```
<!-- UNICENS Root Node  -->
<Node Address="0x200">
    <SyncConnection MuteMode="NoMuting">
        <USBSocket EndpointAddress="0x2" FramesPerTransaction="188"/>
        <NetworkSocket Bandwidth="4" Route="Music"/>
    </SyncConnection>
</Node>

<!-- 1st Auxiliary I/O, Slim Node -->
<Node Address="0x240" Script="aux-io-config">
    <StreamPort DataAlignment="Left16Bit" ClockConfig="64Fs"/>
    <SyncConnection MuteMode="NoMuting">
        <NetworkSocket Bandwidth="4" Route="Music"/>
        <StreamSocket StreamPinID="SRXA1" Bandwidth="4"/>
    </SyncConnection>
</Node>

<!-- Script for Auxiliary I/O -->
<Script Name="aux-io-config">
    <I2CPortCreate Speed="FastMode"/>
    <I2CPortWrite Address="0x18" Payload="00 0f 02 01 00 00 02 a5 df 03
</Script>
```

The first section of the XML file describes the configuration of the Root Node:

An audio stream originated from its application goes via USB Endpoint “2” to the local INIC, which in turn sends out the data, identified by the Route named “Music”, via the INICnet technology 50utp network.

The second section describes the configuration of the Auxiliary I/O Slim Node:

After the referenced script has been processed, the remote INIC at address “0x240” is configured to receive the data stream, which comes via the Network Socket Route “Music” and goes to the local hardware via the INIC's Streaming Port.

The third section shows the referenced Script:

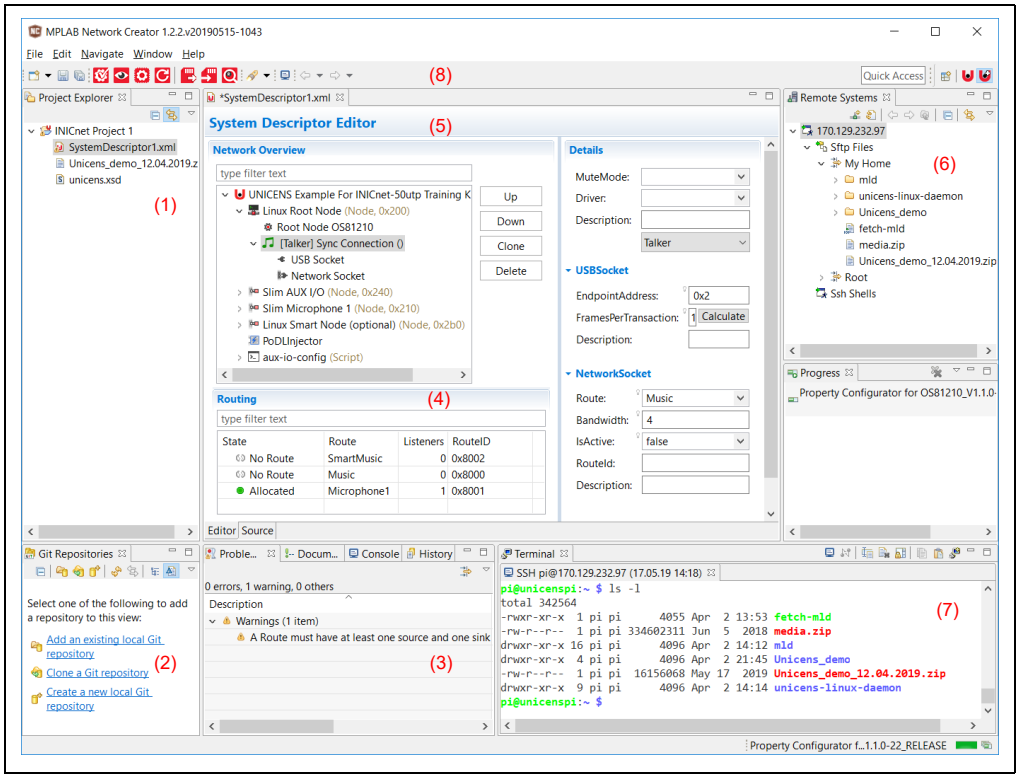
The I^2C Port in the Remote Node is activated, and subsequently some payload/commands are sent from the Root Node to the Remote Node, where the I^2C data is sent to configure the CODEC.

Audio and video streams on the INICnet technology 50utp network can be easily identified and managed by UNICENS but also by the customer's application, simply by specifying the configured Route name, in this example it is “Music”.

4.3.2 MPLAB Network Creator

Microchip provides a plug-in for Eclipse which is available free of charge to customers for the configuration of the UNICENS Daemon.

FIGURE 4-8: MPLAB NETWORK CREATOR – GUI EXAMPLE



The plug-in covers all actions, which are necessary when configuring a UNICENS network. MPLAB Network Creator offers various features, such as:

(1) Project Explorer window – Used for project administration

(2) Git Repositories window – Used to access the GitHub repository

(3) Problems window – Shows errors and warnings that were generated during code validation

(4) Routing overview – Shows details on the routing state of streams

(5) System Descriptor Editor – Smart editor that allows for easy and intuitive UNICENS Daemon configuration. It is used to customize and administrate configuration objects, such as network nodes, connection between nodes and scripts.

(6) File transfer to any Root or Smart node, which runs an FTP server

(7) Terminal access via ssh-client

(8) Tool bar that includes the main navigation components of MPLAB Network Creator, e.g., the generation of configuration files for programming the INICs in the system.

4.4 DEVICE-TO-DEVICE COMMUNICATION

Up-to-date in-vehicle infotainment systems need to have extensive communication capabilities. On one hand they consist of a number of well-defined and built-in nodes joined together by a single network. This part of an infotainment system is static and may vary only by the initial selection of the equipment level for a particular car. On the other hand, they are highly dynamic, as they embrace the whole of consumer devices that can be connected at any time. Thus, an in-vehicle infotainment system may involve multiple wired and wireless networks connected to each other.

Managing communication between nodes of different types located in different networks is not a trivial endeavor. This may lead to applications being very complex and hard to maintain. To be able to handle such a complex scenario, Microchip recommends using a messaging system, which fulfills the following requirements:

- It shall be independent from the transport layer.
- The content integrity shall be secured by a checksum (i.e., CRC).
- It shall support version control of the communication protocol; important for adding features over time, while supporting older versions.
- It should support automated code generation.

Microchip has developed sample code that demonstrates how to manage complex device-to-device communication. This sample code is provided free of charge as open source code to simplify and accelerate development of up-to-date infotainment systems. On application layer, the sample code provides mechanisms to fully abstract device-to-device communication from the underlying physical layer. Regardless of the network specifics, applications establish communication and transfer content between nodes in a standardized way. This keeps complexity of the applications low, but at the same time increases code readability and maintainability.

To help customers quick starting their project, Microchip provides some sample implementations of messaging systems to fit different device classes and project environments:

SOME/IP

- Fulfills the requirements above
- Well-known in automotive industry
- Free of charge tool chain
- Powerful definition language and abstract exchange format
- Code generators for different programming languages
- Tools available to trace and disassemble traffic and to implement rest-bus simulation (see https://www.k2l.de)
- Quite heavy in size for an MCU
- Examples are available at: https://github.com/MicrochipTech/some-ip

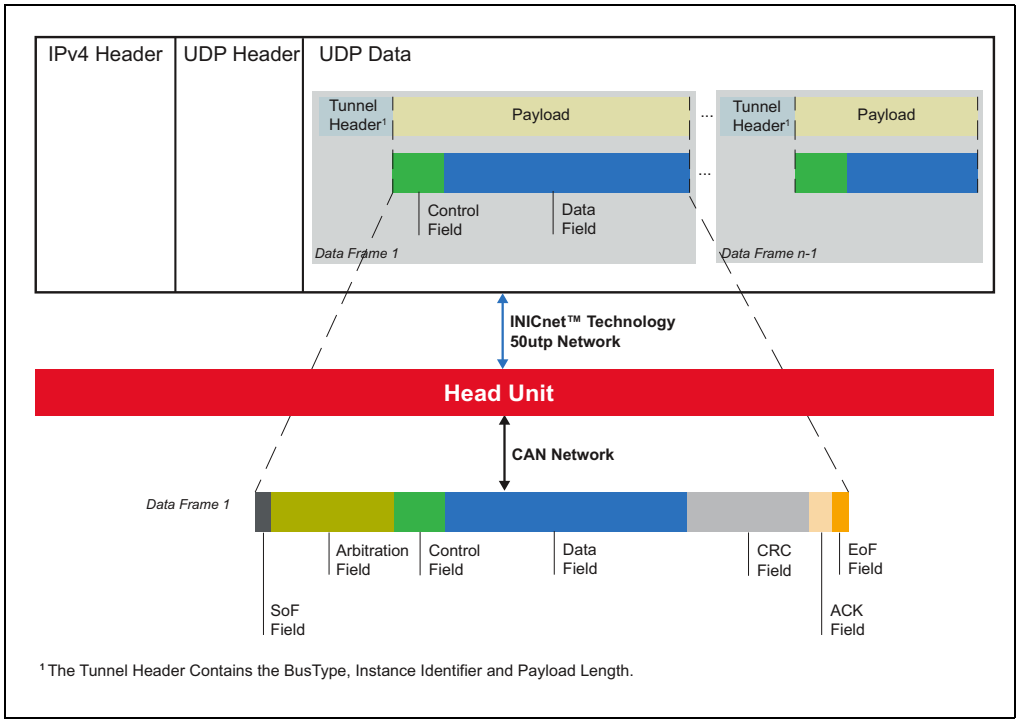
Microchip Messaging System

- Fulfills the requirements above
- Light weight and simple to use
- Supports code generators, but currently not available
- Examples are available at: https://github.com/MicrochipTech/microchip-messaging-system

Tunneling of existing Protocols (example: CAN)

- Fulfills some of the requirements above
- Light weight and simple to use
- Well known base protocol in automotive industry
- Legacy ECUs may be maintained
- Simple to implement for lower message rates

FIGURE 4-9: EXAMPLE – TUNNELING OF CAN PROTOCOL



NOTES:

Chapter 5. Half-Duplex Diagnosis

In a network structure, communication interruptions can occur due to a broken cable connection, e.g. a ring break. To unveil such physical defects, the INIC provides the capability to examine the connection status between network nodes in network architectures that are based on a dual-simplex structure by performing a half-duplex diagnosis.

The half-duplex diagnosis does not require external hardware triggers and cabling like for CAN, LIN, ECL. It uses a specific API function for calling the diagnosis. The registered callback will then deliver the diagnosis result.

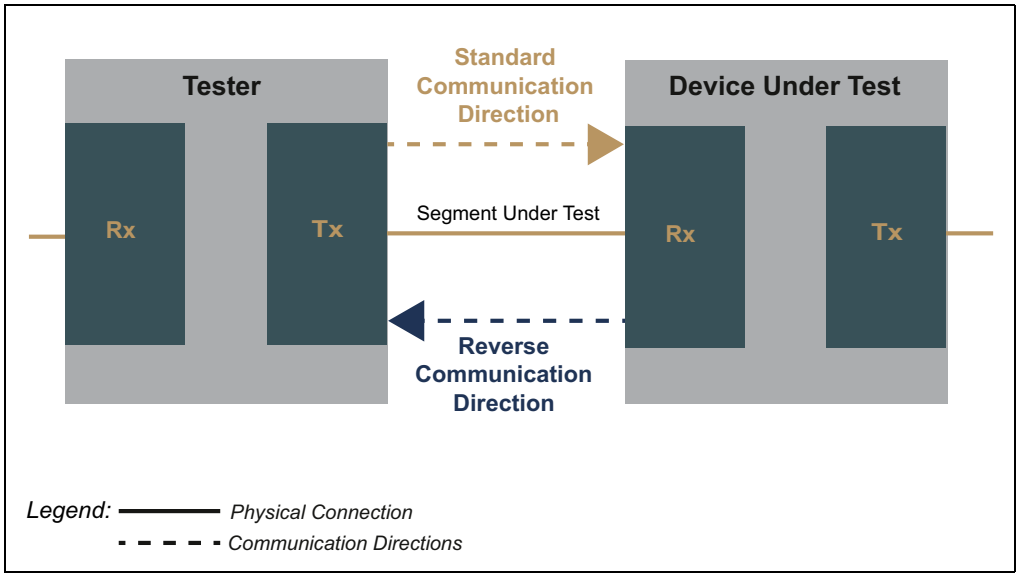
Under the hood, the system acts as described below.

The half-duplex diagnosis approach simply switches the communication direction, which is the logical inversion of the standard communication direction, see Figure 5-1. This implies that for reverse communication Tx will be used as Rx and Rx will be used as Tx.

Once the half-duplex diagnosis mode is entered, the communication channel can be either used in standard communication direction or in reverse communication direction. The switching between the directions is handled by timers. Timer values are distributed via control messages, which are sent as broadcast messages during runtime.

Half-duplex diagnosis always focuses on a specific connection segment between two network nodes. In order to test a certain segment, the nodes get assigned the role of a Tester and a Device Under Test (DUT).

FIGURE 5-1: HALF-DUPLEX DIAGNOSIS MODE – SEGMENT UNDER TEST



Whether a device becomes a Tester or a DUT is defined by a broadcast message that is sent out to the network and based on the node position of a certain device. If the DUT can be addressed in the standard communication direction, it will become the Timing-

Master during the reverse communication phase and start the network in reverse direction. If it cannot be addressed, due to e.g., a cable link interruption such as a ring break, it will not be able to start the network during the reverse communication phase.

During reverse communication, the Tester, which is now in TimingSlave mode, waits for the signal from the DUT and behaves as follows:

- If the DUT has generated a signal and started the network in reverse communication direction, the Tester sends a “Segment is OK” message to the Root Node.
- If the DUT has not generated a signal, e.g., due to a ring break, the Tester becomes the TimingMaster after a specified period of time and starts the network in reverse communication direction. In this case, the Tester sends a “Segment is NotOK” message to the Root Node.

The half-duplex diagnosis uses a segment-by-segment testing approach. The diagnosis always starts with the verification of the first segment, which is between the Root Node (Tester) and the first TimingSlave (DUT). If this segment was verified as fully functional – this means the Tester has sent back the result message “Segment is OK” – the next segment gets examined. This is done by shifting the Tester and DUT roles to the next two devices; now the segment to be tested is between the first TimingSlave (Tester) and the following TimingSlave (DUT). The shifting of the segment under test is continued until the diagnosis is finished. With this algorithm, the communication path from the current Tester device to the Root Node is ensured.

The Root Node always can receive a diagnosis result message from the Tester, which indicates the status of the segment that is under test (OK or NotOK). If the Root Node does not receive any message from the Tester, the diagnosis phase was interrupted irregularly. In this case, the diagnosis phase must be canceled, and a new diagnosis session must be started.

The diagnosis process ends if

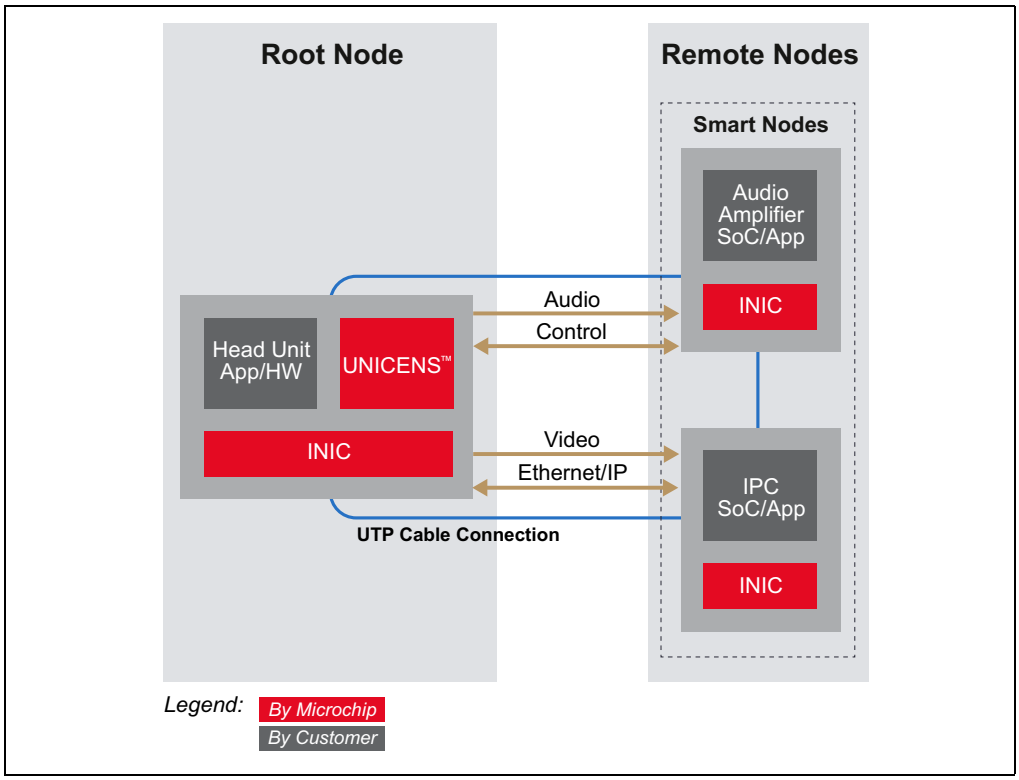
- a DUT does not start the network in reverse communication direction (ring break detected) and therefore a “Segment is NotOK” message is sent back from the Tester to the Root Node,
- the Root Node was found (ring closed) or
- the Root Node has not received the result message from the Tester.

Chapter 6. Use Cases

6.1 EXAMPLE SYSTEM WITH AUDIO AMPLIFIER

Figure 6-1 shows a system set up that incorporates a Head Unit, an Audio Amplifier and an Instrument Panel Cluster.

FIGURE 6-1: EXAMPLE SYSTEM WITH AUDIO AMPLIFIER



Head Unit:

Includes a powerful SoC that is capable of providing most of the desired functionality. It renders:

- The GUI for the local display to select media content, sets up the navigation, adjusts car settings
- The Instrument Panel Cluster image to be shown on the Instrument Panel Cluster ECU display.
  For this purpose, it collects information from other domains available on CAN or Ethernet such as Revolutions Per Minute (RPM), vehicle speed and temperature.

Audio Amplifier:

Receives all audio streams from the system. Apart from the Booster that is driving the Speakers, it incorporates a digital mixer. This mixer takes care of the different static circumstances including:

- Down mixing of multi-channel audio to the available number of Speakers
- Crossover and equalization depending on the equipped Speaker types (e.g., size of subwoofer), positions (e.g., door, trunk, A-pillar) and vehicle type (e.g., sedan, van)
- User settings such as tone, balance, fading

The Audio Amplifier also handles dynamic events such as:

- Volume adjustment depending on vehicle speed
- Audio ducking in case of navigation announcements or alert sounds

Instrument Panel Cluster:

Mainly decodes the video stream received from the Head Unit and renders it to its local display e.g., showing a navigation stream. Due to the typical location of the IPC, it might be useful to collect button information from the steering wheel via LIN for example. This data can be forwarded over the IP channel to the Head Unit. In addition, the IP channel can be used for software updates of the Instrument Panel Cluster from the Head Unit.

Bandwidth Usage

The bandwidth usage values in the table below and the following bandwidth examples are based on the following INICnet technology 50utp parameters:

- Total network bandwidth: 128 bytes/frame
- Administration bandwidth: 12 bytes
- Configuration control channel bandwidth: 4 bytes
- Bandwidth available for streaming- and packet data: 116 bytes/frame

The bandwidth usage is as follows:

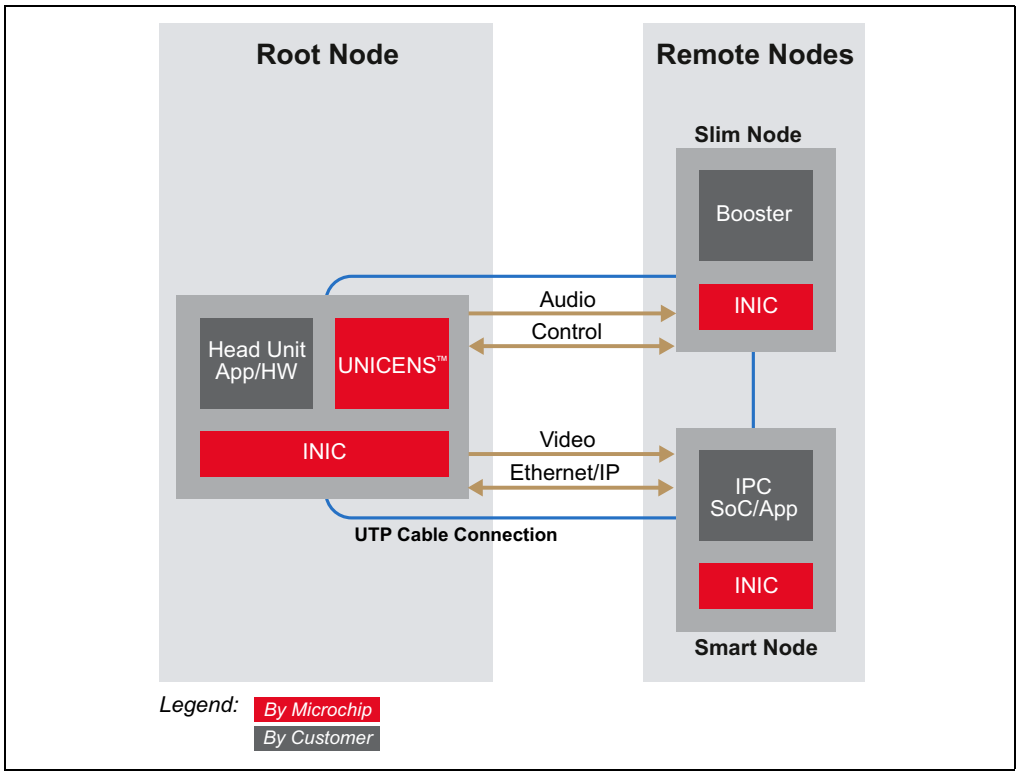
TABLE 6-1: EXAMPLE SYSTEM WITH AUDIO AMPLIFIER

Channel	From	To	Type	Bandwidth		Comment
				[Mbit/s]	[Bytes/ Frame]	
7.1 Audio 24 bit	HU	AMP	Synchronous	9.2	24	PCM audio
Warning Sounds	IPC	AMP	Synchronous	1.5	4	—
Cluster Video	HU	IPC	A/V Packetized	10	26	—
Navigation Sound	HU	AMP	Synchronous	1.5	4	—
Phone	HU	AMP	Synchronous	0.7	2	—
IP channel	All	All	Packet	21.5	56	Uses remaining bandwidth; shared

6.2 EXAMPLE SYSTEM WITHOUT AUDIO AMPLIFIER (COST-DOWN SOLUTION)

Figure 6-2 shows a cost-down example system set up that incorporates a Head Unit, a Booster, and an Instrument Panel Cluster. Compared to the previous example, this setup includes more functionality in the Head Unit to reduce costs in the Audio Amplifier; the Audio Amplifier loses its DSP and turns into a very simple Booster.

FIGURE 6-2: EXAMPLE SYSTEM WITHOUT AUDIO AMPLIFIER



Head Unit:

Is based upon a powerful SoC that:

- Renders the GUI for the local display to select media content, sets up the navigation, adjusts car settings
- Renders the Instrument Panel Cluster image to be shown on the Instrument Panel Cluster ECU display.
  For this purpose, it collects information from other domains available on CAN or Ethernet such as RPM, vehicle speed, temperature
- Sources all audio streams to the system
- Incorporates a digital mixer. This mixer takes care of the different static circumstances including:
  - Down mixing of multi-channel audio to the available number of Speakers
  - Crossover and equalization depending on the equipped Speaker types (e.g., size of subwoofer), positions (e.g., door, trunk, A-pillar) and vehicle type (e.g., sedan, van)
  - User settings such as tone, balance, fading

The Head Unit also handles dynamic events such as

- Volume adjustment depending on vehicle speed
- Audio ducking in case of navigation announcements or alert sounds

Due to the low latency of the INICnet technology 50utp network, the resulting audio mix then can be sent to the Booster.

Booster:

Only amplifies the digital audio streams it receives from the INICnet technology 50utp network. The Booster is a so-called Slim Node, which characteristically has a very lean design. The Booster does not incorporate an MCU or a DSP, which makes it very cost-effective. Depending on the level of equipment, the number of channels per Booster could vary. In many cases it will be cheaper to have multiple Boosters for mid- and high-end lines. Instead of connecting each Speaker with a long, heavy and expensive loud-speaker cable, a light-weight UTP connection would save costs and weight when the Boosters are placed close to the Speakers. Multiple Boosters can be manufactured as identical parts. To distinguish them in the INICnet technology 50utp network, their INIC can be configured end of line.

Most of the Audio Amplifier circuits can execute a self-, cable- and Speaker-diagnosis. To access these features from the Head Unit, the remote I^2C functionality of Microchip's INICs can be used.

Instrument Panel Cluster:

Mainly decodes the video stream received from the Head Unit and renders it to its local display e.g., showing a navigation stream. Due to the typical location of the IPC, it might be useful to collect button information from the steering wheel via LIN for example. This data can be forwarded over the IP channel to the Head Unit. In addition, the IP channel can be used for software updates of the Instrument Panel Cluster from the Head Unit.

Bandwidth Usage

The bandwidth usage for the cost-down system setup is as follows:

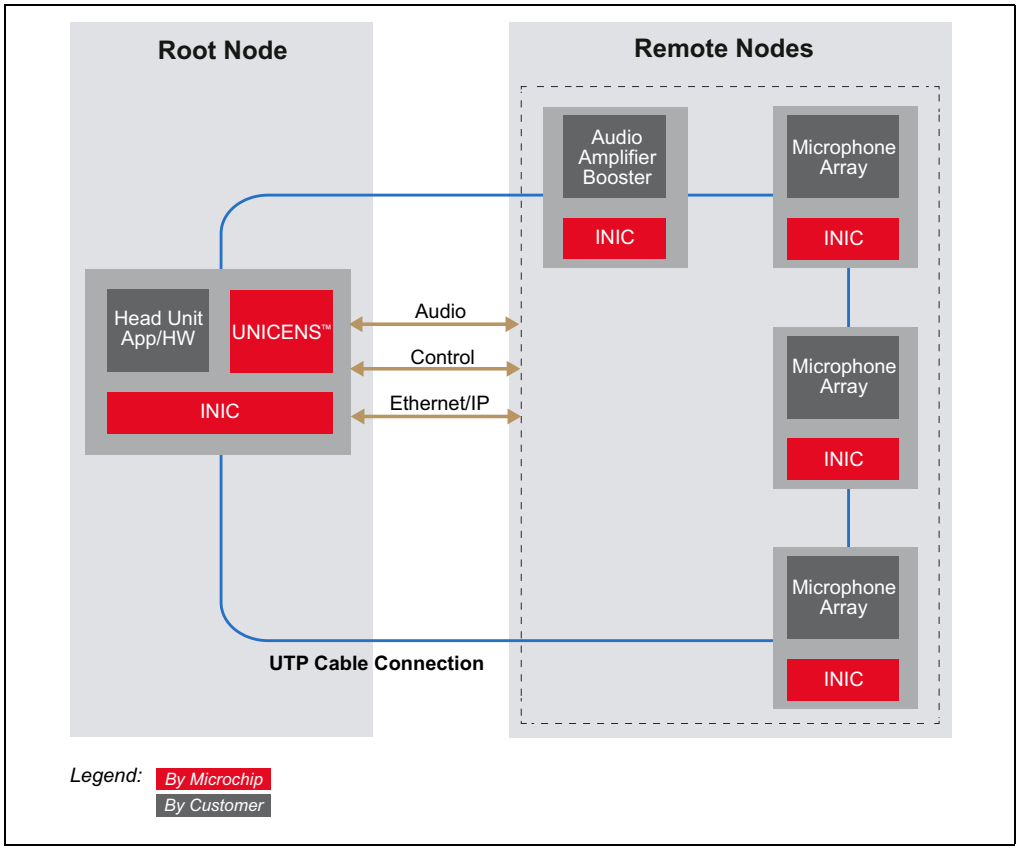
TABLE 6-2: EXAMPLE SYSTEM WITHOUT AUDIO AMPLIFIER

Channel	From	To	Type	Bandwidth		Comment
				[Mbit/s]	[Bytes/ Frame]	
7.1 Audio 24 bit	HU	Booster	Synchronous	9.2	24	PCM audio, can be received by multiple Boosters
Warning Sounds	IPC	HU	Synchronous	1.5	4	—
Cluster Video	HU	IPC	A/V Packetized	10	26	—
IP channel	All	All	Packet	20	52	Shared
Spare	—	—	Unassigned	3.8	10	Unused bandwidth, can be used for future upgrades

6.3 EXAMPLE SYSTEM WITH DIGITAL MIC. ARRAYS (IN-CAR COMMUNICATION)

Figure 6-3 shows an example system set up that incorporates a Head Unit and three Microphone Arrays (one in front row, second one for first rear seat row and third one for second rear seat row) to realize the in-car communication and voice applications such as voice assistance, sound zones, beam forming and hands free communication.

FIGURE 6-3: EXAMPLE SYSTEM WITH DIGITAL MICROPHONE ARRAYS



Head Unit:

Includes a powerful SoC that is capable of providing most of the desired functionality. It renders the GUI for voice applications. It does the DSP processing required for in-car communication and voice applications such as voice assistance, sound zones, beam forming and hands free communication.

Audio Amplifier:

Receives all audio streams from the system and incorporates a digital mixer. This mixer takes care of the different static circumstances, including:

- Down mixing of multi-channel audio to the available number of Speakers
- Crossover and equalization depending on the equipped Speaker types (e.g., size of subwoofer), positions (e.g., door, trunk, A-pillar) and vehicle type (e.g., sedan, van)
- User settings such as tone, balance, fading

The Audio Amplifier also handles dynamic events such as:

- Volume adjustment depending on vehicle speed
- Audio ducking in case of navigation announcements or alert sounds

Microphone Arrays:

These are digital MEMS microphone arrays. Each array can consist of 2-8 MEMS microphones with a defined geometrical spacing. These microphones can have either PDM or I^2S interface.

Bandwidth Usage

The bandwidth usage is as follows:

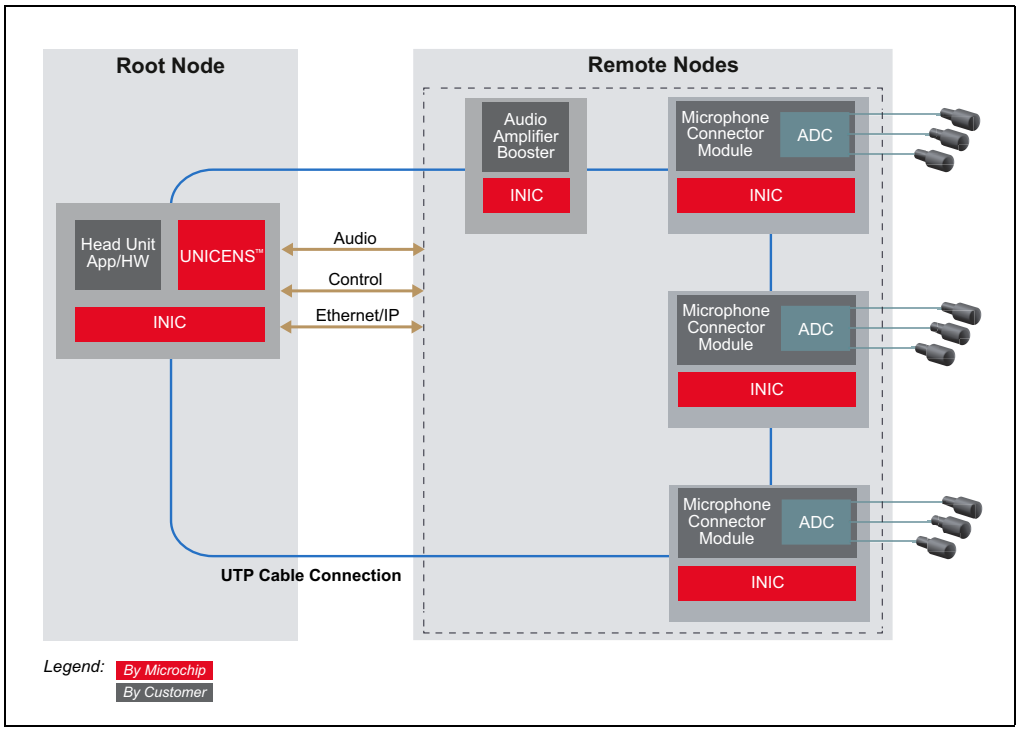
TABLE 6-3: EXAMPLE SYSTEM WITH DIGITAL MICROPHONE ARRAYS

Channel	From	To	Type	Bandwidth		Comment
				[Mbit/s]	[Bytes/ Frame]	
7.1 Audio 24 bit	HU	AMP	Synchronous	9.2	24	PCM audio
Voice	MICs	AMP	Synchronous	6.9	18	Three MEMS micro-phones per array
IP channel	All	All	Packet	20	52	Shared
Spare	—	—	Unassigned	8.4	22	Unused bandwidth, can be used for future upgrades

6.4 EXAMPLE SYSTEM WITH MIC. CONNECTOR MODULES (IN-CAR COMMUNICATION)

Figure 6-4 shows an example system set up that incorporates a Head Unit and three Microphone Connector Modules (one in front row, second one for first rear seat row and third one for second rear seat row). To each module a number of analog microphones (depending on the resolution and number of audio channels of ADC) can be connected to realize in-car communication and voice applications such as voice assistance, sound zones, beam forming and hands free communication.

FIGURE 6-4: EXAMPLE SYSTEM WITH MICROPHONE CONNECTOR MODULES



Head Unit:

Includes a powerful SoC that is capable of providing most of the desired functionality. It renders the GUI for voice applications. It does the DSP processing required for in-car communication and voice applications such as voice assistance, sound zones, beam forming and hands free communication.

Audio Amplifier:

Receives all audio streams from the system and incorporates a digital mixer. This mixer takes care of the different static circumstances, including:

- Down mixing of multi-channel audio to the available number of Speakers
- Crossover and equalization depending on the equipped Speaker types (e.g., size of subwoofer), positions (e.g., door, trunk, A-pillar) and vehicle type (e.g., sedan, van)
- User settings such as tone, balance, fading

The Audio Amplifier also handles dynamic events such as:

- Volume adjustment depending on vehicle speed
- Audio ducking in case of navigation announcements or alert sounds

Microphone Connector Module:

These are ECUs with Analog-Digital Converters (ADCs), which connect the analog microphones to the digital bus. Each ECU can have more than eight analog microphones depending on the resolution and the number of channels available by the ADC.

Bandwidth Usage

The bandwidth usage is as follows:

TABLE 6-4: EXAMPLE SYSTEM WITH MICROPHONE CONNECTOR MODULES

Channel	From	To	Type	Bandwidth		Comment
				[Mbit/s]	[Bytes/ Frame]	
7.1 Audio 24 bit	HU	AMP	Synchro-nous	9.2	24	PCM audio
Voice	MIC Connector Module	AMP	Synchro-nous	6.9	18	Three analog microphones connected to each MIC connector module
IP channel	All	All	Packet	20	52	Shared
Spare	—	—	Unassigned	8.4	22	Unused bandwidth, can be used for future upgrades

List of Figures

NOTES:

List of Tables

NOTES:

Index

Linux® is the registered trademark of Linus Torvalds in the U.S. and other countries.

QNX® is the registered trademark of Blackberry Limited Corporation.

Android™ is a trademark of Google LLC.

MPLAB Network Creator is based on Eclipse. For the Eclipse Public License refer to https://www.eclipse.org/legal/epl-v20.html

Worldwide Sales and Service

AMERICAS

Corporate Office
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 480-792-7200
Fax: 480-792-7277
Technical Support:
http://www.microchip.com/support
Web Address:
www.microchip.com

Atlanta
Duluth, GA
Tel: 678-957-9614
Fax: 678-957-1455

Austin, TX
Tel: 512-257-3370

Boston
Westborough, MA
Tel: 774-760-0087
Fax: 774-760-0088

Chicago
Itasca, IL
Tel: 630-285-0071
Fax: 630-285-0075

Dallas
Addison, TX
Tel: 972-818-7423
Fax: 972-818-2924

Detroit
Novi, MI
Tel: 248-848-4000

Houston, TX
Tel: 281-894-5983

Indianapolis
Noblesville, IN
Tel: 317-773-8323
Fax: 317-773-5453
Tel: 317-536-2380

Los Angeles
Mission Viejo, CA
Tel: 949-462-9523
Fax: 949-462-9608
Tel: 951-273-7800

Raleigh, NC
Tel: 919-844-7510

New York, NY
Tel: 631-435-6000

San Jose, CA
Tel: 408-735-9110
Tel: 408-436-4270

Canada - Toronto
Tel: 905-695-1980
Fax: 905-695-2078

ASIA/PACIFIC

Australia - Sydney
Tel: 61-2-9868-6733

China - Beijing
Tel: 86-10-8569-7000

China - Chengdu
Tel: 86-28-8665-5511

China - Chongqing
Tel: 86-23-8980-9588

China - Dongguan
Tel: 86-769-8702-9880

China - Guangzhou
Tel: 86-20-8755-8029

China - Hangzhou
Tel: 86-571-8792-8115

China - Hong Kong SAR
Tel: 852-2943-5100

China - Nanjing
Tel: 86-25-8473-2460

China - Qingdao
Tel: 86-532-8502-7355

China - Shanghai
Tel: 86-21-3326-8000

China - Shenyang
Tel: 86-24-2334-2829

China - Shenzhen
Tel: 86-755-8864-2200

China - Suzhou
Tel: 86-186-6233-1526

China - Wuhan
Tel: 86-27-5980-5300

China - Xian
Tel: 86-29-8833-7252

China - Xiamen
Tel: 86-592-2388138

China - Zhuhai
Tel: 86-756-3210040

ASIA/PACIFIC

India - Bangalore
Tel: 91-80-3090-4444

India - New Delhi
Tel: 91-11-4160-8631

India - Pune
Tel: 91-20-4121-0141

Japan - Osaka
Tel: 81-6-6152-7160

Japan - Tokyo
Tel: 81-3-6880- 3770

Korea - Daegu
Tel: 82-53-744-4301

Korea - Seoul
Tel: 82-2-554-7200

Malaysia - Kuala Lumpur
Tel: 60-3-7651-7906

Malaysia - Penang
Tel: 60-4-227-8870

Philippines - Manila
Tel: 63-2-634-9065

Singapore
Tel: 65-6334-8870

Taiwan - Hsin Chu
Tel: 886-3-577-8366

Taiwan - Kaohsiung
Tel: 886-7-213-7830

Taiwan - Taipei
Tel: 886-2-2508-8600

Thailand - Bangkok
Tel: 66-2-694-1351

Vietnam - Ho Chi Minh
Tel: 84-28-5448-2100

EUROPE

Austria - Wels
Tel: 43-7242-2244-39
Fax: 43-7242-2244-393

Denmark - Copenhagen
Tel: 45-4450-2828
Fax: 45-4485-2829

Finland - Espoo
Tel: 358-9-4520-820

France - Paris
Tel: 33-1-69-53-63-20
Fax: 33-1-69-30-90-79

Germany - Garching
Tel: 49-8931-9700

Germany - Haan
Tel: 49-2129-3766400

Germany - Heilbronn
Tel: 49-7131-72400

Germany - Karlsruhe
Tel: 49-721-625370

Germany - Munich
Tel: 49-89-627-144-0
Fax: 49-89-627-144-44

Germany - Rosenheim
Tel: 49-8031-354-560

Israel - Ra'anana
Tel: 972-9-744-7705

Italy - Milan
Tel: 39-0331-742611
Fax: 39-0331-466781

Italy - Padova
Tel: 39-049-7625286

Netherlands - Drunen
Tel: 31-416-690399
Fax: 31-416-690340

Norway - Trondheim
Tel: 47-7288-4388

Poland - Warsaw
Tel: 48-22-3325737

Romania - Bucharest
Tel: 40-21-407-87-50

Spain - Madrid
Tel: 34-91-708-08-90
Fax: 34-91-708-08-91

Sweden - Gothenberg
Tel: 46-31-704-60-40

Sweden - Stockholm
Tel: 46-8-5090-4654

UK - Wokingham
Tel: 44-118-921-5800
Fax: 44-118-921-5820