Application Note: AN01001

Adding TCP/IP to AVB

This application note describes how to add TCP/IP and a Webserver to the AVB Endpoint Software on a XMOS Multicore Microcontroller.

Details of prerequisites and required software components are provided, together with instructions to build an example application that combines AVB Endpoint (Audio Listener) functionality with a simple webserver.

For the discovery and control of AVB networks the use of the AVB standard 1722.1 protocol is always recommended. However some applications, such as Automotive SOME/IP or an audio endpoint with a built-in webserver, also require layer 3 functionality - which this note facilitates.

Required tools and libraries

- xTIMEcomposer Tools - Version 13.2.0
- XMOS AVB Stack - Version 6.0.4
- Ethernet/TCP Module - Version 3.2.1rc1.a

Required hardware

The example code provided with the application note has been implemented and tested on the XMOS AVB-LC Kit[1] connected to a MacBook Pro 9,2.

Prerequisites

- This document assumes familiarity with the XMOS xCORE architecture, the XMOS tool chain and the xC language. Documentation related to these aspects which are not specific to this application note are linked to in the references appendix.
- Basic knowledge of Ethernet Networks and the applicable Protocols (TCP/IP, AVB)
- Reading the **AVB Endpoint Quickstart Guide** and **AVB System Requirements Guide**.
- For descriptions of XMOS related terms found in this document please see the XMOS Glossary[2].

[1] http://www.xmos.com/products/reference-designs/avb-lc
[2] http://www.xmos.com/published/glossary

1 Overview

1.1 Introduction

Adding layer 3 functionality to the AVB Audio Endpoint software can be easily achieved by integrating the Simple HTTP Demo with the AVB Endpoint Software. The steps to do this are described in the section Code changes. The resulting application is illustrated in the next section Block diagram.

1.2 Block diagram

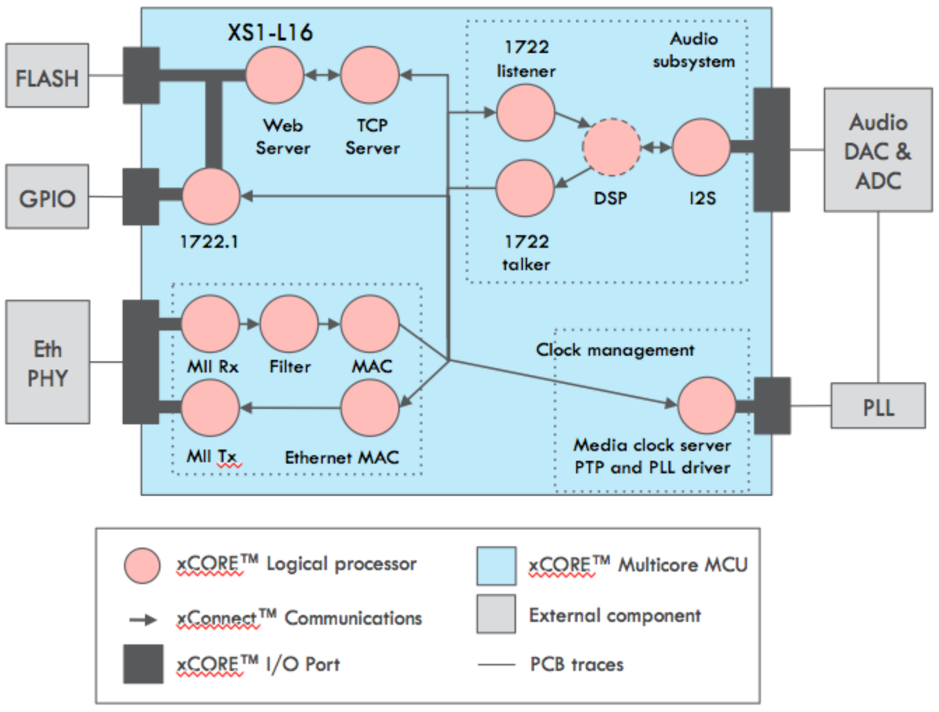


Figure 1: Core diagram of AVB with integrated Webserver

2 Application note for Adding TCP/IP to AVB

2.1 Download and combine the Software Components

- Download the AVB Endpoint Software from https://www.xmos.com/support/reference%20designs
- Download the Ethernet/TCP Module from https://www.xmos.com/support/xsoftip
- Create a directory for the application. Unzip the AVB Endpoint Software into the directory
- Unzip Ethernet/TCP Module in a separate directory. Copy the sc_xtcp into the application directory.

2.2 Setup Eclipse Workspace

- Open xTIMEcomposer
- Switch Workspace to the application directory
- Import the directory: File->Import, Existing Projects Into Workspace, Select root directory.

Do **not** select Copy projects into workspace.

2.3 Code changes

The following sections describe the code changes required to integrate a Webserver on TCP/IP with the AVB Endpoint software. This new functionality can be enabled with the #define AVB_WITH_TCP.

2.3.1 Adding the webserver to the AVB application

In the project explorer, open the app_simple_webserver. Copy the src directory and paste it into the directory app_avb_lc_demo/src. Rename the pasted directory to tcp_webserver_simple. Open the the main.xc file in tcp_webserver_simple. Copy the webserver task:

```
on tile[0]: xhttpd(c_xtcp[0]);
```

into the “par {” block in the main.xc file of the AVB application in app_avb_lc_demo/src like so:

```
#ifdef AVB_WITH_TCP
  on tile[0]: xhttpd(c_xtcp[0]);
#endif
```

Add the declaration for the channel array which is used by the Webserver to communicate with the TCP/IP server. Add this code before the “par {” block:

```
#ifdef AVB_WITH_TCP
  chan c_xtcp[1];
#endif
```

2.3.2 Adding the TCP/IP server to the AVB application

Now add the TCP/IP server task xtcp_server_uip to the application. Note: This is a TCP/IP server on the network. But with respect to the firmware architecture it is a client to the avb_ethernet_server. The TCP/IP server task should be added next to the xhttpd task described in the previous chapter. The code added to the "par {" block of main.xc in app_avb_lc_demo/src should now be this:

```
#ifdef AVB_WITH_TCP
    on tile[1] : xtcp_server_uip(c_mac_rx[MAC_RX_TO_TCP],
            c_mac_tx[MAC_TX_TO_TCP], c_xtcp, 1,
            ipconfig);

    on tile[0]: xhttpd(c_xtcp[0]);
#endif
```

Note that the channel array c_xtcp now connects the TCP/IP server with the webserver for data exchange.

2.3.3 Updating the enums for MAC rx and tx channels

The channels used by clients to exchange data with the avb_ethernet_server are enumerated. They have to be updated in main.xc to add the channels for for connecting the xtcp_server_uip task:

```
enum mac_tx_chans {
  MAC_TX_TO_MEDIA_CLOCK = 0,
#if AVB_DEMO_ENABLE_TALKER
  MAC_TX_TO_TALKER,
#endif
  MAC_TX_TO_SRP,
  MAC_TX_TO_1722_1,
  MAC_TX_TO_AVB_MANAGER,
#ifdef AVB_WITH_TCP
  MAC_TX_TO_TCP,
#endif
  NUM_MAC_TX_CHANS
};

enum mac_rx_chans {
  MAC_RX_TO_MEDIA_CLOCK = 0,
#if AVB_DEMO_ENABLE_LISTENER
  MAC_RX_TO_LISTENER,
#endif
  MAC_RX_TO_SRP,
  MAC_RX_TO_1722_1,
#ifdef AVB_WITH_TCP
  MAC_RX_TO_TCP,
#endif
  NUM_MAC_RX_CHANS
};
```

2.3.4 Updating the MAC filter

The avb_ethernet_server filters Ethernet Packets (according to the Ethertype field) and forwards them to the dedicated clients. This is controlled by the the mac_custom_filter() function. This function has to be extended in order to make the avb_ethernet_server task forward applicable packets to the xtcp_server_uip task. Update the mac_custom_filter function in avb_mac_filter.h as follows to accomplish this:

```
  switch (etype) {
#ifdef AVB_WITH_TCP
    case 0x0608:
    case 0x0008:
      result = MAC_FILTER_ARPIP;
      break;
#endif
```

Now packets of type TCP/IP, UDP, ARP, ICMP, etc will be forwarded to the xtcp_server_uip client.

In addition, the index of the xtcp_server_uip (called MAC_FILTER_ARPIP as used above) has to be added to the enum mac_clients in the same file:

```
enum mac_clients {
  MAC_FILTER_1722=0,
  MAC_FILTER_PTP,
  MAC_FILTER_AVB_CONTROL,
  MAC_FILTER_AVB_SRP,
#ifdef AVB_WITH_TCP
  MAC_FILTER_ARPIP,
#endif
  NUM_FILTER_CLIENTS
};
```

2.3.5 Define various configuration parameters

Static configuration parameters have to be updated to accomodate the TCP/IP server integration.

xtcp_client_conf.h:

Create a file xtcp_client_conf.h in app_avb_lc_demo/src to define the configuration and integration parameters of the TCP/IP server. This file will be automatically imported. It should have this content:

```
#define XTCP_VERBOSE_DEBUG (1)

#define UIP_PACKET_SPLIT_THRESHOLD 8
#define XTCP_SEPARATE_MAC 1 // The existing MAC in avb_ethernet_server is used and connected to connect
  ↪ xtcp_server_uip
#define ETHERNET_USE_XTCP_FILTER 0 // The existing AVB MAC filter is used and extended.

#define XTCP_USE_QUICKSTART 1
#define XTCP_QUICKSTART_ETHERNET_INTERFACE 1
```

app_config.h:

The source code for the TCP/IP integration is implemented under #ifdef. It has to be enabled with:

```
#define AVB_WITH_TCP
```

The AVB Talker is disabled when AVB_WITH_TCP is defined. To accomplish this conditionally, replace this line:

```
#define AVB_DEMO_ENABLE_TALKER 1
```

With this:

```
#ifdef AVB_WITH_TCP
#define AVB_DEMO_ENABLE_TALKER 0
#else
#define AVB_DEMO_ENABLE_TALKER 1
#endif
```

Adding TCP/IP and webserver functionality to the AVB Endpoint software consumes additional system resources, both cores and memory. The extra memory requirements of the TCP/IP and webserver functionality mean that, when enabled, only AVB Audio Endpoint listener functionality is available - talker functionality must be disabled.

app_conf.h:

Increase the number of MAX_ETHERNET_CLIENTS to 5:

```
#define MAX_ETHERNET_CLIENTS   5
```

2.3.6 Customise the simple webpage

The simple Webserver that is being integrated with AVB will create a simple website with the message:

```
"Hello World!"
```

This can be customised in httpd.c to a different message. E.g.:

```
<body>This page is generated from a simple TCP/IP webserver integrated with AVB</body></html>
```

2.3.7 Makefile changes

The Layer 3 TCP/IP (or alternatively UDP) functionality is provided by the sc_xtcp component. To make use of the TCP/IP functionality, add the following line to the Makefile in app_avb_lc_demo:

```
USED_MODULES += module_webserver
```

To make Header files in tcp_webserver_simple accessible, this line has to be added:

```
INCLUDE_DIRS += src/tcp_webserver_simple
```

2.3.8 Remove the default webserver application

Now that we have integrated the app_simple_webserver into app_avb_lc_demo, we have to delete the main.xc file in app_avb_lc_demo/tcp_webserver_simple. Otherwise there would be a compiler error from two conflicting main() functions.

APPENDIX A - Demo Hardware Setup

This application example is designed to run on a single AVB LC Kit Board connected to a MacBook Pro. OSX supports AVB natively. This enables to use the same Ethernet Port to stream Audio to the AVB Endpoint and access the web server via TCP/IP simultaneously.

Windows doesn't support AVB natively but Windows users can connect the Windows PC and both boards of the AVB AUDIO ENDPOINT kit to an AVB Switch. One AVB AUDIO ENDPOINT should be configured with the standard AVB Audio Endpoint software and the other with the software described in this application note. Audio may be streamed between the AVB AUDIO ENDPOINTs and the webserver will be accessible to the Windows PC.

Setting up Audio Streaming between the Boards is described in the **AVB Endpoint Quick Start Guide**.

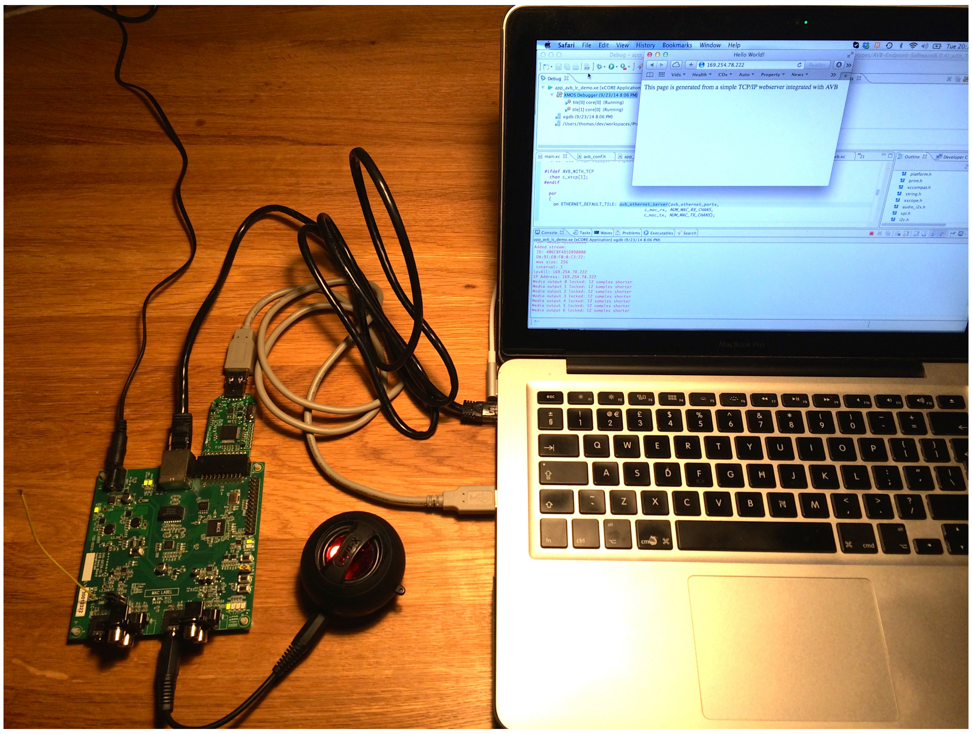


Figure 2: Hardware Setup. The Screen of the Mac shows xSCOPE Realtime Trace and the Webpage from the Webserver

The hardware should be configured as displayed above for this demo on the Mac:

- Both the power and debug adapter should be connected to the board
- Debug adapter connected to the MacBook Pro or another USB Host
- Ethernet cable connected to the MacBook Pro

APPENDIX B - Launching the demo device

Once the demo example has been built either from the command line using xmake or via the build mechanism of xTIMEcomposer studio we can execute the application on the AVB Endpoint board.

Once built there will be a `bin` directory within the project which contains the binary for the xCORE device. The xCORE binary has a XMOS standard .xe extension.

B.1 Launching from the command line

From the command line we use the `xrun` tool to download code to the xCORE device. If we change into the bin directory of the project we can execute the code on the xCORE microcontroller as follows:

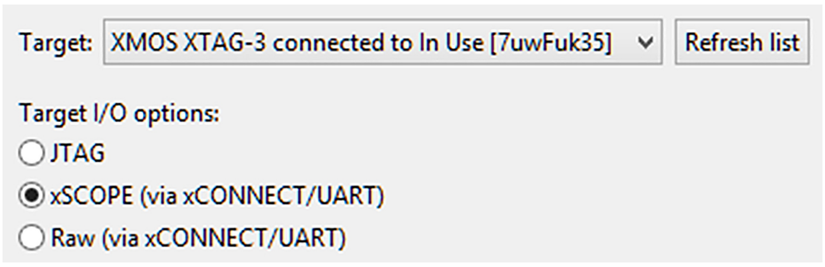
```
> xrun --xscope app_avb_lc_demo.xe          <-- Download and execute the xCORE code
```

Once this command has executed you should see a real time xscope trace. See Chapter Expected Realtime Stacktrace

B.2 Launching from xTIMEcomposer Studio

From xTIMEcomposer Studio we use the run mechanism to download code to xCORE device. Select the xCORE binary from the bin directory, right click and then follow the instructions below.

- Select **Run As**, **Run Configuration.**
- Double Click **xCORE Application** to create a new Run Configuration
- Enable xSCOPE in Target I/O options:



Target: XMOS XTAG-3 connected to In Use [7uwFuk35]

Target I/O options:
- JTAG
- [✓ xSCOPE (via xCONNECT/UART)]
- Raw (via xCONNECT/UART)

Figure 3: xTIMEcomposer xSCOPE configuration

- Click **Apply** and then **Run.**

B.3 Accessing the Webserver

The Stacktrace will print the IP address which is assignd by DHCP. E.g.:

```
IP Address: 169.254.78.222
```

You can use the printed IP address to access the Webserver with a Browser. The Webserver hosts a simple webpage that prints the message:

```
This page is generated from a simple TCP/IP webserver integrated with AVB
```

You can also ping the Server on the IP address.

B.4 Selecting the AVB Endpoint for Audio Playback

Follow the instructions in Chapter **3 Apple Mac OS X Support** in **AVB System Requirements Guide**. Once you Play Audio to the AVB Endpoint using some Media Player, you will see messages on the console reporting that the Media Clock is locked. See *Expected Realtime Stacktrace*.

B.5 Expected Realtime Stacktrace

This is the expected stack that is printed with xscope without interference with the Realtime execution of the Application:

```
** Starting simple webserver integrated with AVB **
PTP Port 0 Role: Master
Address: 0.0.0.0
Gateway: 0.0.0.0
Netmask: 0Setting clock source: INPUT_STREAM_DERIVED
.0.0.0
PTP Port 0 Role: Master
PTP Port 0 Role: Master
Joined SRP domain (VID 2, port 0)
Joined SRP domain (VID 2, port 0)
Joined SRP domain (VID 2, port 0)
Joined SRP domain (VID 2, port 0)
PTP Port 0 Role: Master
PTP Port 0 Role: Slave
PTP sync locked
1722.1 Controller 0FE2A33C6 acquired entity
CONNECTING Listener sink #0 -> Talker stream 406C8F481D890000, DA: 91:E0:F0:0:C3:22
Listener sink #0 chan map:
  0 -> 0
  1 -> 1
  2 -> 2
  3 -> 3
  4 -> 4
  5 -> 5
  6 -> 6
  7 -> 7
1722 router: Enabled map for stream 406C8F481D890000 (link_num:0, hash:0)
MSRP: Register attach request 406C8F48:1D890000
Added stream:
 ID: 406C8F481D890000
Added stream:
 ID: 406C8F481D890000
 DA:91:E0:F0:0:C3:22:
 max size: 256
 interval: 1
ipv4ll: 169.254.78.222
IP Address: 169.254.78.222

Media output 0 locked: 12 samples shorter
Media output 1 locked: 12 samples shorter
Media output 2 locked: 12 samples shorter
Media output 3 locked: 12 samples shorter
Media output 4 locked: 12 samples shorter
Media output 5 locked: 12 samples shorter
Media output 6 locked: 12 samples shorter
Media output 7 locked: 12 samples shorter
```

APPENDIX C - References

XMOS Tools User Guide

http://www.xmos.com/published/xtimecomposer-user-guide

XMOS xCORE Programming Guide

http://www.xmos.com/published/xmos-programming-guide

AVB System Requirements Guide

http://www.xmos.com/published/avb-system-requirements-guide

AVB Endpoint Quick Start Guide

http://www.xmos.com/published/avb-quick-start-guide

Ethernet TCP/IP Component Programming Guide

http://www.xmos.com/published/ethernet-tcpip-component-programming-guide

AVnu Alliance

http://www.avnu.org

Copyright © 2016, All Rights Reserved.

Xmos Ltd. is the owner or licensee of this design, code, or Information (collectively, the “Information”) and is providing it to you “AS IS” with no warranty of any kind, express or implied and shall have no liability in relation to its use. Xmos Ltd. makes no representation that the Information, or any particular implementation thereof, is or will be free from any claims of infringement and again, shall have no liability in relation to any such claims.