

AUDIO4-5410

Portable DSP Data Acquisition

Rev. D and E

Product of Domain Technologies, Inc.

Audio4-5410 User's Guide (D/E)

September, 2003

Domain Technologies, Inc.

811 East Plano Pkwy, Suite 115

Plano, Texas 75074

Tel.: (972) 578-1121

Fax: (972) 578-1086

E-mail: support@domaintec.com

Web page: <http://www.domaintec.com>

Disclaimer of Warranty

This software package is provided on an 'AS IS' basis and without warranty. In no event shall Domain Technologies be liable for incidental or consequential damages arising from the use of this software. This disclaimer of warranty extends to LICENSEE, to LICENSEE's customers or users of products and is in lieu of all warranties whether expressed, implied, or statutory, including implied warranties of merchantability or fitness for a particular purpose. Domain Technologies does not warrant that software furnished hereunder is free of infringement of any third party patents, copyrights, or trade secrets.

Table Of Contents

CHAPTER 1	Introduction	5
	Overview of the AUDIO4-5410	5
	Key features of the AUDIO4-5410	8
	Functional overview of the AUDIO4-5410	9
	Installation instructions	10
CHAPTER 2	Operation of the AUDIO4-5410	11
	AUDIO4-5410 board	11
	External connectors of the AUDIO4-5410	11
	Light emitting diodes (LEDs)	12
	Internal connectors	12
	Default jumper shunt locations	13
	AUDIO4-5410 memory	13
	I/O space	16
	Host Port Interface Option Connector	16
	McBSP1 interface connector.	17
	Codec A power down options - J15	17
	Input signal access headers - J5 and J7	18
	Codec secondary input headers - J6 and J8	18
	DSP operating conditions.	19
	Oscillator selection	19
	Stereo Audio Codecs	20
	Codec control	21
	Control port serial data format	22
	Headphones output	24
CHAPTER 3	Software	25
	Code Composer interface.	25
	User program - host side	25
	Emulnit	26
	EmuCall	27
	Access to the DSP user program	28
	USB monitor	29
	User program - DSP side	29
	PC test application	32

	General options	33
	Flash programming options	33
	DSP application control options	34
	COFF conversion utility.	35
CHAPTER 4	Schematics	37
	PLD Equations	48

1.1 - Overview of the AUDIO4-5410

The AUDIO4-5410 is a DSP (Digital Signal Processor) based system with the USB (Universal Serial Bus) interface to the host computer system, and 4 input/output audio channels.

The AUDIO4-5410 system can be used as a development platform for the DSP algorithms, or as an evaluation module for audio processing algorithms. AUDIO4-5410 allows for real-time transfer of the digital data to/from the host PC over the high-speed USB connection.

System is shipped with the driver for the Code Composer Studio, providing path for the debugger to access the DSP directly through the USB connection. Included sample programs illustrate initialization of the analog subsystem, sending control commands to the user application, and transfer of the digital audio data over the USB connection.



FIGURE 1.1. Audio4-5410 board - LINE IN configuration

The default configuration allows to connect 4 input channels with standard LINE IN signal levels. Optionally system can have installed microphone daughter board providing preamp and bias voltage for the electret microphones.



FIGURE 1.2. Microphone pre amp and bias

Circuit on the microphone daughter board replaces “line-in” circuitry. It is installed in place of jumper shunts, connecting input jacks to line-in circuit. Daughterboard provides adjustable bias voltage, which can be disconnected from each of four input lines.

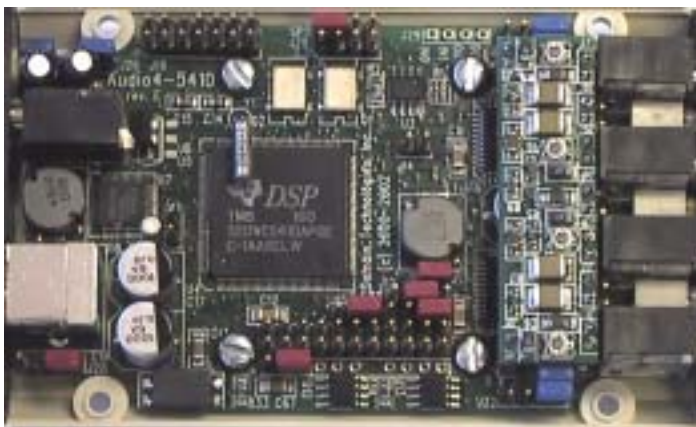


FIGURE 1.3. Audio4-5410 board with optional microphone daughter board

To accommodate standard electret microphones, quad microphone adaptor needs to be used. It allows to connect microphone signal and bias either to tip, ring, or both.



FIGURE 1.4. Quad microphone adaptor

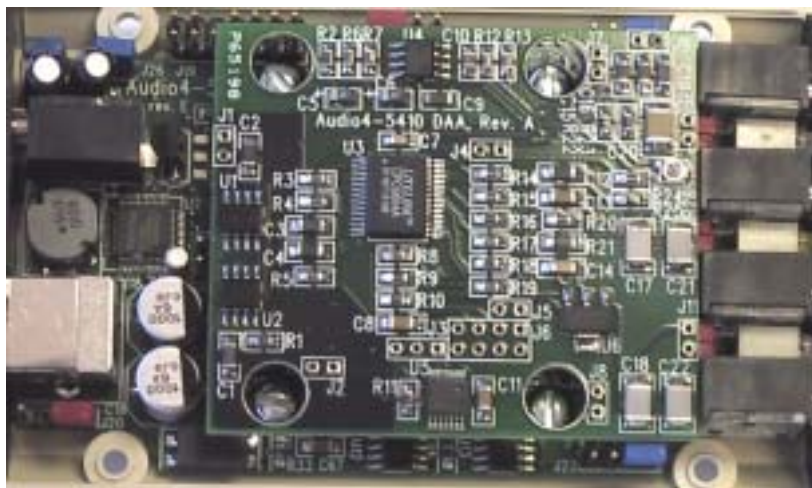


FIGURE 1.5. Audio4-5410 board with optional ETA-5410 adapter

Experimental Telephony Adapter (ETA-5410) converts system into telephone system with standard current loop 2 wire interface, with single line microphone input and single line output. One of two codecs (A) is disabled, and its control signals

are used to control DAA interface circuits. DAA's transmit and receive channels are connected to the left channel of the codec B. The right channel input of the codec B is connected to the preamp/bias circuit for the electret microphone. The right channel output of the codec B is connected to the line B out jack and right channel of the headphones connector.

1.2 - Key features of the AUDIO4-5410

- TMS320VC5410 processor with 64 K words of RAM, operating at 100 MHz
- 128 K words (16 bit) of flash ROM
- high speed (12 MHz) USB controller with fast I/O access
- two 16-bit stereo codecs – CS4218, sampling rate from 7.35 to 48 kHz
- two input and two output 3.5 mm stereo jacks for line level audio I/O
- one 3.5 mm stereo headphone jack
- bus powered operation, no external power supply needed
- optional Host Port Interface access through 22 pin 0.1" header
- optional quad microphone input adaptor
- 14 pin 0.1" header for optional JTAG access to the DSP

1.3 - Functional overview of the AUDIO4-5410

Figure below shows a block diagram of the AUDIO4-5410 system. The major components of the system include DSP, 2 stereo codecs, system ROM and the USB interface.

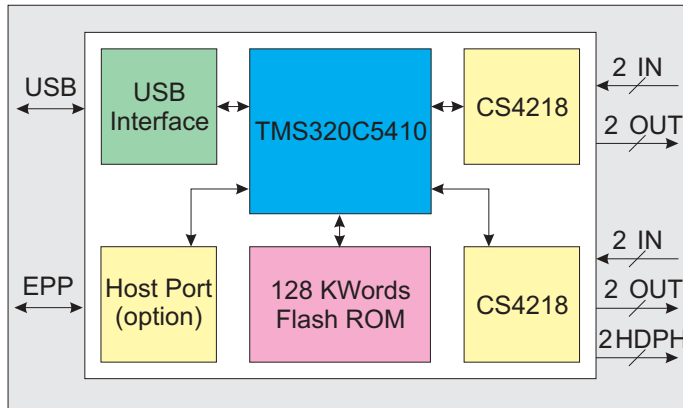


FIGURE 1.6. Audio4-5410 block diagram

The USB interface is controlled directly by the DSP. The USB interface chip is mapped into the DSP's I/O memory space, and requests service through one of the interrupts. The USB control code resides in the external ROM, so the DSP is ready to respond to the USB initialization requests from the host computer, immediately after power-up.

Optionally user application code can be loaded into the DSP's RAM from the external ROM, allowing for the stand-alone operation.

Two stereo codecs operate in the 32 bit per frame Master Mode. Each codec is connected to the separate serial port of the DSP, and can be programmed to operate at a different sampling frequency. System features two oscillators for the analog subsystem: 12.288 MHz and 11.2896 MHz. First oscillator allows to select 8 sampling frequencies from 8-48 kHz range. Second oscillator allows for 8 sampling frequencies from 7.32-44.1 kHz range. The control information is sent to each codec from the third DSP serial port, operating in the SPI protocol mode. The codec control features include 0-22.5 dB input gain control and 0-46.5 dB output attenuation.

The system external ROM is mapped in the high 32 K words of the program memory space, and can be also accessed from the high data memory space. 3 K words of the flash ROM are occupied by the USB control code, and rest can be used for storing user program or data. Host Port Interface can be accessed through the 22 pin header. Optional interface adapter allows to fully control the AUDIO4-5410 system form the PC's printer port, with the USB interface inactive.

1.4 - Installation instructions

AUDIO4-5410 can be installed on the PC computer with Windows 98, Me, 2000 or XP. Windows 95 and Windows NT do not support USB devices.

After connecting AUDIO4-5410 system to the USB port of the host computer, Windows will attempt to install device drivers for the USB peripheral. Drivers for the USB interface of the AUDIO4-5410 are located on the CD in the respective directories:

- for Windows 2000/XP: D:\Win2000
- for Windows 98/ME: D:\Win98

Simple test application is located on the CD in the \bin directory:
audio54.exe

System operation can be verified with the following sequence of commands:

- 1 – open connection to the device
- H – stop execution of the user application
- L – load codec54.out to the DSP
- G – start execution of the codec54.out code
- R – start recording audio data to the PC
- S – stop recording
- P – play back recorded data
- Q – exit test program

Detailed description of the test application audio54.exe can be found in chapter "PC test application" on page 32.

The runtime software can be installed from the startup menu, or with the setup.exe in the \audio54 directory.

2.1 - AUDIO4-5410 board

Six of the connectors are accessible from the outside of the box. There are also optional headers and configuration jumpers on top of the board.

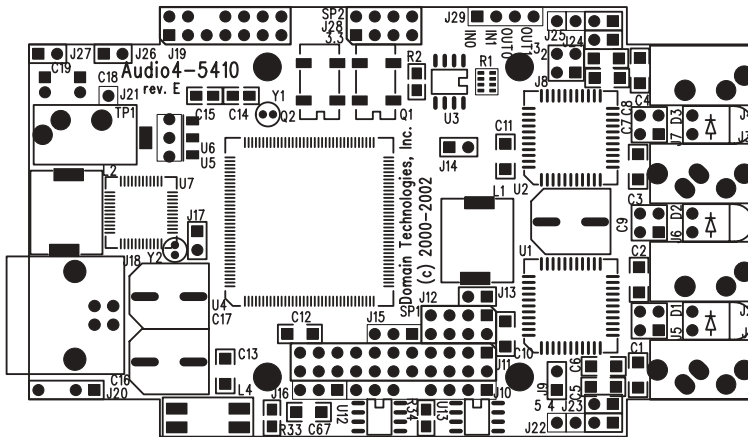


FIGURE 2.1. Audio4-5410 board

2.2 - External connectors of the AUDIO4-5410

- J1 – Codec A stereo input
- J2 – Codec A stereo output
- J3 – Codec B stereo input
- J4 – Codec B stereo output
- J18 – USB connector
- J21 – Codec B amplified stereo headphone output

2.3 - Light emitting diodes (LEDs)

- D1 – red/green bi-color, application controlled
- D2 – yellow, power indicator
- D3 – red/green bi-color, application controlled

2.4 - Internal connectors

- J5 – Input A access
- J6 – Codec A secondary input
- J7 – Input B access
- J8 – Codec A secondary input
- J9 – 5VA daughter board power supply
- J10 – ISP programming header for the PLD
- J11 – Host Port Interface
- J12 – McBSP1 access header
- J13 – 5VA bypass option
- J14 – 3.3V digital supply
- J15 – Codec A power down control
- J16 – MC/MP selector for DSP initialization
- J17 – 3.3V digital supply
- J19 – JTAG emulation header
- J20 – External power supply

2.5 - Default jumper shunt locations

Figure below shows location of jumpers shunts installed in the factory.

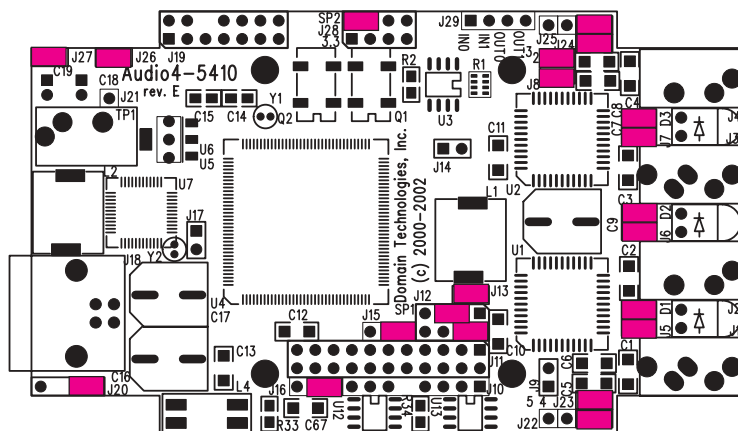


FIGURE 2.2. Audio4-5410 default jumper shunts locations

2.6 - AUDIO4-5410 memory

The TMS320VC5410 processor has 64 K words of the internal RAM. In addition to this, the system is equipped with 16 bit wide flash ROM. Four blocks of 32 K words of the ROM are accessible to the system. First block is used for USB monitor code and USB constant data structures. User can store application parameters or user code in range 0x8200..0xBFFF. Additional three 32 K word blocks are available for user application: 0x18000..0x1FFFF, 0x28000..0x2FFFF, 0x38000..0x3FFFF.

For the USB operation, external ROM needs to be mapped into high program memory space: 0x8000..0xFFFF.

For the operation with the optional Host Port Interface (HPI) adaptor to the PC's printer port, external ROM needs to be disabled, so the DSP can be reset and bootstrap directly through the HPI. Selection of the external ROM mapping after system reset, is done by setting the jumper on the three pin header (J7). J7 controls the state of the DSP's MP/-MC pin.

USB control program has reserved space for its operation. The user application needs to follow rules described in the chapter “User program - DSP side” on page 29. The interrupt vectors are relocated to the address 0x7F80. User application can overwrite interrupt vectors during load, but cannot modify INT0, TRAP B and TRAP C interrupt vectors during execution. Interface between USB monitor and user application is located in the communication area at 0x7F00..0x7F7F. Following detailed map of the program/data memory for the standard operation. Extended flash ROM pages can be accessed from the user program.

Address	Program	Data
0x0000 0x005F	Reserved	Memory mapped registers
0x0060 0x007F	Reserved	Scrach-Pad RAM
0x0080 0x1FFF	On-Chip DARAM	
0x2000 0x7BFF	On-Chip SARAM1	
0x7C00 0x7CFF	USB monitor stack	
0x7D00 0x7EFF	USB monitor data area	
0x7F00 0x7F7F	USB/Application interface area	
0x7F80 0x7FFF	Interrupt vectors	
0x8000 0x81FF	USB monitor static data	On-Chip SARAM2 (DROM = 1) External ROM (DROM = 0)
0x8200 0xBFFF	ROM for user application auto initialization	
0xC000 0xFFFF	USB monitor code	

USB monitor can also operate in the RAM mode. Most frequently monitor functions are executed from the on-chip RAM. This increases memory requirements, but improves system performance. In the RAM monitor mode, additional 3 K words are reserved for the monitor execution. Following table illustrated monitor’s

memory usage. All communication data structures are located at the identical addresses, as in the ROM operation mode.

Address	Program	Data
0x0000 0x005F	Reserved	Memory mapped registers
0x0060 0x007F	Reserved	Scrach-Pad RAM
0x0080 0x1FFF	On-Chip DARAM	
0x2000 0x6FFF	On-Chip SARAM1	
0x7000 0x70FF	USB monitor stack	
0x7100 0x7CFF	USB monitor code--	
0x7D00 0x7EFF	USB monitor data area	
0x7F00 0x7F7F	USB/Application interface area	
0x7F80 0x7FFF	Interrupt vectors	
0x8000 0x81FF	USB monitor static data	On-Chip SARAM2 (DROM = 1) External ROM (DROM = 0)
0x8200 0xBFFF	ROM for user application auto initialization	
0xC000 0xFFFF	USB monitor code	

Current mode of the monitor operation is reported by the interface library. Request for the name of the interface library:

```
ver = EmuError(0,EMU_DLL_QUERY_VERSION);
```

will return following strings for ROM and RAM mode of operation:

A4USB32 rev. 1.xx.xx (ROM)

A4USBD32 rev. 1.xx.xx (RAM)

2.7 - I/O space

I/O ports provide access to the USB interface controller, and to the control latches for the audio subsystem. Lower 32 I/O addresses are dedicated to the USB controller chip. Whole access to the USB controller is done from the USB monitor code. Only exception is for the fast bulk data transfer done by the user application. In this case user application needs to read or write to the FIFO's of the USB controller, but whole transfer setup is still done by the user monitor. Details of the data transfers are in the chapter "PC test application" on page 32.

User application controls analog subsystem through 6 I/O ports with two or three active bits each:

I/O address	Bit 2	Bit 1	Bit 0
0x0020		LED 0 Red	LED 0 Green
0x0021		LED 1 Red	LED 1 Green
0x0022	Codec SELA2	Codec SELA1	Codec SELA0
0x0023	Codec SELB2	Codec SELB1	Codec SELB0
0x0024	Codecs Pwr Dwn	Codec A CS	Codec B CS
0x0025		Osc 1 Enable	Osc 2 Enable

2.8 - Host Port Interface Option Connector

Audio4-5410 can be accessed also through the optional Host Port Interface adapter. The interface header has all signals required for the host port bootstrap procedure. Printer port adaptor is available as a optional item from Domain Technologies.

Pin #	Signal	Pin #	Signal
1	HD0	2	HD1
3	HD2	4	HD3
5	HD4	6	HD5
7	HD6	8	HD7

Pin #	Signal	Pin #	Signal
9	-HCS	10	HCNTL0
11	HRW	12	HCNTL1
13	HRDY	14	-HDS0
15	-HINT	16	-HDS1
17	HBIL	18	-HAS
19	-INT2	20	-RESET
21	Gnd	22	3.3V

2.9 - McBSP1 interface connector.

DSP's serial port #1 can be used to connect to the user peripherals. Signals of the McBSP1 are connected to the codec A. If the codec A is set in the power down state, those signals can be connected to the external peripherals, or used as general purpose I/O (GPIO). To force power down condition on the codec A, jumper needs to be placed on pins 2 and 3 of the J8. For normal operation jumper is installed on pins 1 and 2 - DSP has control on the codec's power down feature. Power down control signal is common for both codecs.

Pin	Signal
1	3.3 V
2	BCLKR1
3	BCLKX1
4	BFSR1
5	BFSX1
6	BDR1
7	BDX1
8	Gnd

2.10 - Codec A power down options - J15

Power down option of the codec A can be controlled from the DSP, or by forcing the power down mode with the J8. For normal operation, jumper should be installed on pins 1 and 2 of the J8. If the signals of the McBSP1 are needed as

GPIO signals (J7), power down of the codec A should be forced by installing jumper on pins 2 and 3 of J8.

Pin	Signal
1	-PDN from DSP
2	-PDN to codec
3	Gnd

2.11 - Input signal access headers - J5 and J7

For normal operation both headers have two jumpers installed on pins 1-2 and 3-4. Headers are used to connect external input jacks to the microphone daughter-board. For daughter board connection pins 2 and 4 are only used.

Pin	Signal
1	Right channel to codec
2	Right input from jack
3	Left channel to codec
4	Left input from jack

2.12 - Codec secondary input headers - J6 and J8

For normal operation both headers have two jumpers installed on pins 1-2 and 3-4, connecting bypass capacitors to unused codec inputs. For microphone daughter board installation, headers provide access to the secondary input of the codecs. For daughter board connection only pins 2 and 4 are used.

Pin	Signal
1	Right channel bypass capacitor
2	Right channel codec input
3	Left channel bypass capacitor
4	Left channel codec input

2.13 - DSP operating conditions.

Audio4-5410 features TMS320VC5410 Digital Signal Processor with 64 K words of internal RAM, operating at 96 MHz. Processor input clock is shared with the clock supplied to the USB controller – 48 MHz. After power up, the processor enables on-chip PLL to multiply the clock frequency by 2.

DSP interfaces to the external flash ROM through the memory bus. Memory interface logic is implemented within the Lattice PLD (ispLSI2032VE100). Decoded is only –MSTRB signal, with –PS and –DS ignored. So access to the flash ROM is possible from the page 0 or 1 (program or data memory space).

Other peripherals are located within I/O space: USB controller occupies lower 32 addresses (0x0000..0x001F), and control latches next 6 I/O addresses (0x0020..0x0025).

Interrupt line -INT0 is driven by the USB controller, line –INT2 is connected to the optional Host Port Interface header, to facilitate HPI bootstrap.

2.14 - Oscillator selection

AUDIO4-5410 features one oscillator (48 MHz) used as the clock input for the DSP and USB controller and two oscillators in the analog subsection. The 48 MHz oscillator is always active, and cannot be disabled.

The analog section oscillators are driving the same connection to the analog converters. When disabled, the oscillators are in the STANDBY mode, which reduces power consumption. Enabling the oscillator requires 10 ms delay to get it in the stable operating condition.

Two low bits at the I/O address 0x0025 control oscillator enable signals

Bit 1	Bit0	Oscillator mode
0	0	Both disabled
0	1	Enabled 12.288 MHz
1	0	Enabled 11.2896 MHz
1	1	Reserved (both enabled)

2.15 - Stereo Audio Codecs

AUDIO4-5410 features two Codecs CS4318 from Cirrus Logic (Crystal Semiconductor)

Codec Features:

- Complete CMOS stereo audio input and output system
- Dual ADCs and DACs using 64x oversampling
- Input anti-aliasing and output smoothing filters
- Programmable input gain (0 to 22.5 dB)
- Programmable output attenuation (0 to 46.5 dB)
- Sample frequencies from 4 to 50 kHz
- Low distortion
- THD < 0.02% for DAC
- THD < 0.02% for ADC
- Low power dissipation: 80 mA typical
- Power-down mode: 1 mA typical

The CS4218 stereo audio codec is a monolithic CMOS device for computer multimedia, automotive, and portable audio applications. It performs A/D and D/A conversion, filtering, level setting, creating four audio inputs, and two audio outputs for a digital computer system. The digital interfaces of left and right channels are multiplexed into a single serial data bus with word rates up to 50 kHz per channel.

The ADCs and DACs use DS modulation with 64x oversampling, and include digital-decimation and output-smoothing filters on-chip to eliminate the need for external anti-aliasing filters.

Codecs are connected to the separate serial ports, so each can be set to the different sampling rate. Codecs operating mode is set to Master Serial Mode 4, 32 bits per frame. Data is transferred to/from the DSP serial port as two channels per frame with MSB first.

Control data is sent to the codec's the Serial Control Port from the third serial port of the DSP operating in the SPI mode.

Sampling rate selection is available through the I/O port address 0x0022 (Codec A) and 0x0023 (Codec B).

MF6:F1 SELx1	MF7:F2 SELx2	MF8:F3 SELx3	N	Fs (kHz)	
				12.288 MHz	11.2896 MHz
0	0	0	256	48.00	44.10
0	0	1	384	32.00	29.40
0	1	0	512	24.00	22.05
0	1	1	640	19.20	17.64
1	0	0	768	16.00	14.70
1	0	1	1024	12.00	11.025
1	1	0	1280	9.60	8.82
1	1	1	1536	8.00	7.35

2.16 - Codec control

The control information is entered through the separate port that can be asynchronous to the audio port and only needs to be updated when changes in the control data are needed. After a reset or power down, the control port must be written once to initialize it if the port will be accessed to read or write control bits. This initial write is considered a “dummy” write since the data is ignored by the codec. A second write is needed to configure the codec as desired. Then, the control port only needs to be written to when a change is desired, or to obtain the status information. The control port does not function if the master clock is not operating. When the control port is used asynchronously to the audio port, the noise performance may be slightly degraded due to the asynchronous digital noise.

2.17 - Control port serial data format

Data to the control port is sent as a 32 bit word, MSB first. The received status data is also received as 32 bit word, MSB first.

Bit(s)	Symbol	Description
7 – 0	Unused	
11 – 8	RG3 – RG0	Right Input Gain 1.5 dB increments. 0000 = No gain (0 dB) 1111 = 22.5 dB gain
15 – 12	LG3- LG0	Left Input Gain 1.5 dB increments. 0000 = No gain (0 dB) 1111 = 22.5 dB gain
16	ISR	Input Mux, Right Select 0 = RIN1 1 = RIN2
17	ISL	Input Mux, Left Select 0 = LIN1 1 = LIN2
18	MUTE	Mute DAC outputs 0 = Outputs ON 1 = Outputs Muted
23 – 19	RA4 – RA0	Right Output Attenuation 1.5 dB increments. 00000 = no atten. (0 dB) 11111 = 46.5 dB atten.
28 – 24	LA4 – LA0	Left Output Attenuation 1.5 dB increments. 00000 = no atten. (0 dB) 11111 = 46.5 dB atten.
29	DO1	Digital Output 1 0 = output LOW 1 = output HIGH
30	MASK	Mask for the interrupt generated when status changed
31	Unused	

Bit(s)	Symbol	Description
1 – 0	Unused	
2	ADV	ADC valid data 0 = Invalid ADC data 1 = Valid ADC data
3	DI1	Digital Input 1 0 = Input LOW 1 = Input HIGH
4	RCL	ADC Right Clipping 0 = Normal 1 = Clipping
5	LCL	ADC Left Clipping 0 = Normal 1 = Clipping
7 – 6	ER1 – ER0	Error bits
11 – 8	Unused	
15 – 12	VER3 – VER0	CS4218 Version Number 0000 = Rev A 1000 = Rev B and later
17 – 16	ER1 – ER0	Error bits
18	Unused	
19	DI1	Digital Input 1 0 = Input LOW 1 = Input HIGH
20	RCL	ADC Right Clipping 0 = Normal 1 = Clipping
21	LCL	ADC Left Clipping 0 = Normal 1 = Clipping
22	ADV	ADC valid data 0 = Invalid ADC data 1 = Valid ADC data
31 – 23	Unused	

2.18 - Headphones output

Headphones amplifier is driven by the analog output signal from Codec B. Digital Output signal from Codec B can be used to shutdown the headphones amplifier to reduce power consumption. State of the DO1 pin is controlled by the control word sent into Serial Control Port.

3.1 - Code Composer interface.

User software for the Audio4-5410 system can be debugged with Texas Instruments Code Composer development system. Code Composer Studio is available as an optional item, either from Domain Technologies or from Texas Instruments distributor. The Audio4-5410 package contains a necessary driver, enabling access from the Code Composer debug subsystem to the monitor code on the Audio4-5410. The driver needs to be installed and selected through the Code Composer Setup application.

The Audio4-5410 debug driver is supplied as a "dtiusb54.dvr". The driver needs to access two dynamically loaded libraries (dll):

- dtia4usb.dll - main control for Domain's USB devices
- xdti5400.dll - 5400 disassembler needed for calculation of the instruction length.

The dtia4usb.dll is also used by the user application accessing Audio4-5410 system. It provides all necessary control for both applications: debugger and user program, so they can access Audio5-5410 device through the Universal Serial Bus simultaneously.

3.2 - User program - host side

Audio54.exe is a sample user program for the PC (host) side. This is a console type Win32 application. The same API can be also used by windows type Win32 programs. All access from the user program to the Dynamic Library functions is done through following 5 functions:

```
short EmuInit (TRG_INFO *init);
short EmuCall (short handle, short cmd, TRG_CMD *cmdArg);
short EmuClose (short handle);
char *EmuError (short handle, int error);
```

```
short EmuVersion (void);
```

3.2.1 - Emulnit

Connection to the Audio4-5410 is done with the Emulnit function call. Structure passed as parameter includes all the information needed by the device driver to initialize the target devices. Not all members of the structure are used, some are needed for other types of the interfaces (for example multi-DSP debugger). The structure format is as follows:

```
typedef struct{
short      structSize; /* to verify compatibility*/
short      procType;   /* target processor to access*/
short      ifType;     /* interface type to the target*/
char       *procName;  /* name of the processor found*/
short      scanDefLen; /* size of the scanDef table (bytes)*/
pSCAN_DEF  scanDef;    /* structure of multiprocessor */
short      comPort;    /* com port number to use (-1: all)*/
short      comBaud;    /* com baud to connect at*/
short      ioPort;     /* port address to access*/
short      scanNum;    /* scan-chain # */
char       *dllName;   /* name of this DLL (returned)*/
long       dllVer;     /* rev # of this DLL (returned)*/
long       fwVer;      /* rev # of firmware (returned)*/
char       ipStr[IP_STR_LEN]; /* IP address with port #*/
short      dspCount;   /* number of DSPs in the scan chain*/
short      dspNum;     /* DSP # we are talking to*/
short      resetReq;   /* reload firmware request*/
}TRG_INFO, *pTRG_INFO;
```

Required structure elements:

```
trgInfo.structSize = sizeof (trgInfo);
trgInfo.procType = TMS5410;
trgInfo.ifType = AUDIO54USER;
trgInfo.ioPort = usbNum;
trgInfo.dspCount = 1;
```

All other elements of the structure need to be set to 0. After successful initialization, the Emulnit will return dll's name, version, firmware version, and name of the processor found. The definition of the used constants and data types is within the include file `trg_def.h`

Structure element `ioPort` represents the USB device number, which user program wants to initialize. In case of multiple devices connected to the host computer, this number selects the required device. If this number is greater than 127, it will be converted to 0. Device number 0 will select the first not used device. Only single user application can talk to any USB device. If the application needs to talk to multiple USB devices, it has to issue multiple `EmuInit` calls.

Audio4-5410 device can be also accessed remotely, over the TCP/IP network. Communication with the remote device is done through the `BoxServer` application controlling the USB device, and responding to the sockets messages.

To verify if the device is initialized, `EmuError` function needs to be called with the returned handle, and parameter 0 (return last error). If it returns `NULL`, the initialization was successful, otherwise error string indicates the error cause. In case of error, function `EmuClose` needs to be called to release device handle to the driver.

3.2.2 - `EmuCall`

All interaction with the device is done through the `EmuCall` function. First parameter (handle) identifies the device. The Second parameter is a command identification number. Third parameter is a pointer to the structure holding all the other parameters for the function call. The structure format is as follows:

```
typedef struct trgcmd {
SPACE  Type;          /* Memory space to access*/
ADDR   Addr;          /* Memory address to access*/
SPACE  Type2;         /* Space for copy destination*/
ADDR   Addr2;         /* Address for copy destination*/
long   Count;         /* Count of words to access*/
DATABUF*Buf;         /* input/output data buffer*/
long   WrLen;         /* Count of words in Buf to write*/
long   RdLen;         /* Count of words in Buf to read*/
}TRG_CMD, *pTRG_CMD;
```

Detailed description of all `EmuCall` commands can be found in the `DSPLib` user manual.

3.3 - Access to the DSP user program

Access to the user application on the DSP is done by sending a command to the monitor with command id 0x00..0x6F. Commands 0x70..0xFF are reserved for the monitor operation, and are not passed to the user application. Command id number is encoded as a byte (8 bits) - the transfer unit for the USB interface. Command header is defined as follows:

```
typedef struct usb_data {  
    BYTE    usbCmd;  
    BYTE    usbLen;  
    short   usbSeq;  
    BYTE    usbData[MAX_USB54_DATA_LEN + 4];  
}USB_DATA, *pUSB_DATA;
```

The maximum size of the data packet sent as a command to the USB device is 64 bytes (0x40), so the maximum number of bytes sent as a payload is 60. Structure element "len" defines the number of 16-bit words transferred in the payload area. Command interface transfers only 16-bit words as a payload, and that's how it is decoded on the DSP side.

There are two EmuCall commands dedicated for the communication with the user application:

- **TRG_USER_CMD** - transfers data to/from the USB with the above structure. Element usbCmd is initialized with the value passed to the EmuCall as cmdArg->Type. Structure element usbLen receives the value from cmdArg->WrLen. Element usbSeq is initialized automatically by the EmuCall service with the auto incremented sequence number. Then data is copied from the cmdArg->Buf to the usbData, by copying low 16-bits from each 32-bits sent for the host user application. This feature allows for sharing data structures between the PC and DSP, where elements are defined as integers. If the USB device is required to send a response to the caller, this will be indicated by a non-zero value of the cmdArg->RdLen. This will cause to receive 4+RdLen*2 bytes, and will fill the 32-bit words of the cmdArg->Buf with 16-bit words received from the DSP.
- **TRG_XFER_DATA** - transfers byte data to/from the user application. This call allows to transfer bigger blocks of data without any 16 bit to 32 bit word translation. The maximum size of the block to transfer is 0xFFFFC bytes. Data needs to be transferred in the multiples of 4 bytes. The length of the block to transfer is passed as a number of 32-bit words (number of bytes / 4) in the cmdArg->WrLen (for transfer from the PC to the USB) and in the cmdArg->RdLen (for

transfer from the USB to the PC). All other elements of the structure TRG_CMD are ignored.

3.4 - USB monitor

USB monitor is executed out of the device's ROM. It controls all data transfers from and to the host PC.

Audio4-5410 USB interface features 5 endpoints:

- Endpoint 0 is dedicated as a bi-directional configuration endpoint and is used exclusively by the operating system.
- Endpoint 1 is used as pipe to transfer big blocks of unformatted data from the PC to the user application. If the user application is not loaded, or it fails to initialize service for endpoint 1 receive, data will be ignored by the USB monitor. The maximum size of the transfer block is 0xFFFC bytes.
- Endpoint 2 serves as a data pipe for unformatted data transfer from the DSP user application to the PC. If the user application is not loaded, or it fails to initialize service for endpoint 1 receive, data will be ignored by the USB monitor.
- Endpoint 3 is shared between USB monitor and DSP user application for receiving commands from the PC. Size of the command block is limited to 64 bytes, and is defined by the second byte of the header. Specified length indicates the number of 16 bit words following 4 byte command header.
- Endpoint 4 is shared between USB monitor and DSP user application for sending responses back to the host PC. Size of the response block is limited to 64 bytes, with 4 bytes reserved for the header.

All endpoints (except Endpoint 0) are configured as bulk transfer endpoints. Bulk transfer guarantees delivery over the USB bus, in contrary to the isochronous transfer, where data can be lost, if the USB bus is out of the bandwidth.

3.5 - User program - DSP side

User program can be loaded in to the DSP memory at any location except the areas reserved for the USB monitor (between 0x7C00 and 0x7F7F). Interrupt vectors are located at 0x7F80, and can be overwritten during the application load. USB monitor will restore all used interrupt vectors (INT0, TRAP 0xB and TRAP 0xC).

All commands sent from the PC over the USB Endpoint 3 will be filtered by the USB monitor. Commands 0x00 to 0x6F will be redirected to the user command service function. User application needs to initialize at least its entry point to the command service routine. All the interface information of the USB monitor is contained within the following data structure. The structure is located at 0x7F00. This is the protected area of the USB monitor, and is not restored after user program is loaded.

```
typedef struct app_interface {
UINT   size;
void   (*UserCmd)(pUINT ptr);/* pointer to user command service */
void   (*UserEp1Rx)(BYTE count);/* pointer to user data receive */
void   (*UserEp2Tx)(BYTE count);/* pointer to user data transmit */
void   (*UserEp4Tx)(BYTE count);/* pointer to user ctrl transmit */
void   (*UsbRecvData)(pUINT ptr, UINT len, void (*notify)(BYTE));
        /* monitor entry to start receiving data */
void   (*UsbSendData)(pUINT ptr, UINT len, void (*notify)(BYTE));
        /* monitor entry to start sending data */
void   (*UsbSendCtrl)(pUINT ptr, UINT len, void (*notify)(BYTE));
        /* monitor entry to start sending ctrl */
UINT   monitorFlags;        /*flags for interrupts enable */
void   (*BreakEnter)(UINT pc);
        /* user procedure called on breakpoint entry */
void   (*BreakExit)(void);
        /* user procedure called on return after break */
UINT   stdoutStart; /*start of the buffer for text output */
UINT   stdoutEnd;   /*end address of the circular buffer */
UINT   stdoutHead;  /* head pointer (DSP update) */
UINT   stdoutTail;  /* tail pointer (PC update) */
int    (*stdoutProc)(char *str);/* monitor access to the stdout*/
} APP_INT, *pAPP_INT;
```

USB monitor initializes the first and the last four entries in the communication structure.

Structure element "size" is used to verify the compatibility of the monitor code and the user application.

UserCmd - set by the user application. This function will receive all commands sent over Endpoint 3, with the command id 0x00 to 0x6F. The function parameter indicates the location of the command data read from the endpoint 3 - 1 byte per memory location.

UserEp1Rx - set by the user application. This function is called when data is received over Endpoint 1. None of the data is read out of the USB controller, and it needs to read from the USB controller FIFO by the user code. If this function pointer is NULL, data will be read automatically by the USB monitor. Data will be stored at the ptr set by the call to the `UsbRecvData`. Monitor stores each word as a sequence of two bytes read from the USB FIFO - LSB, MSB.

UserEp2Tx - set by the user application. This function is called when data is ready to be sent over the Endpoint 2 to the PC. Data has to be written directly to the USB controller's transmit FIFO. If this function pointer is NULL, data will be transferred automatically by the USB monitor. Data will be read from the ptr set by the call to the `UsbSendData`. Monitor sends each word as a sequence of two bytes - LSB, MSB.

UserEp4Tx - set by the user application. This function is called when data is ready to be sent over the Endpoint 4 to the PC. Data has to be written directly to the USB controller's transmit FIFO. If this function pointer is NULL, data will be transferred automatically by the USB monitor. Monitor transfers each word as a sequence of two bytes - LSB, MSB.

UsbRecvData - set by USB monitor. Call to this function initializes receiving data through the Endpoint 1. User code can provide own function to read data from the USB FIFO, or monitor can store received bytes at the buffer location specified by the first parameter. Second parameter indicates the total number of bytes to be transferred. Optionally user application can receive notification when the transfer is finished. USB monitor will call the notify function with the Endpoint number, which just finished transfer.

UsbSendData - set by USB monitor. Call to this function initializes sending data through the Endpoint 2 to the host PC. User code can provide own function to write data to the USB FIFO, or monitor can write bytes to the USB FIFO from the buffer location specified by the first parameter. Second parameter indicates total number of bytes to be transferred. Optionally user application can receive notification, when the transfer is finished. USB monitor will call the notify function with the Endpoint number, which just finished transfer.

UsbSendCtrl - set by USB monitor. Call to this function initializes sending control data block through the Endpoint 4 to the host PC. The maximum size for this transfer is 64 bytes. User code can provide an own function to write data to the USB FIFO, or monitor can write bytes to the USB FIFO from the buffer location

specified by the first parameter. Second parameter indicates total number of bytes to be transferred. Optionally user application can receive notification when the transfer is finished. USB monitor will call the notify function with the Endpoint number, which just finished transfer.

MonitorFlags - used by the USB monitor. Flag provides overrides to the standard breakpoints handling.

BreakEnter - address of the procedure to be called by the monitor on entry into breakpoint service. This can be used to implement profiling, or additional control of the user peripherals. Monitor will pass the address of the breakpoint to the user procedure.

BreakExit - procedure called by the user monitor on exit from the debug mode - resume after software breakpoint.

stdoutStart - start of the circular buffer for text output from the DSP application to the host PC. This format of passing data from the DSP application, allows for real-time access to the host PC. If the PC application does not support this feature, information will be lost, but DSP will continue execution.

stdoutEnd - end address of the circular buffer for the stdout text output

stdoutHead - head pointer to the circular buffer. This pointer is updated only by the DSP, and cannot be written by the host.

stdoutTail - tail pointer of the circular buffer. This pointer can be updated only by the PC - host application.

stdoutProc - address of the USB monitor procedure storing text string into the circular buffer. This operation, can be also executed by the user code.

Sample user application for the DSP is provided with the Audio4-5410 system.

3.6 - PC test application

Audio54.exe is a sample test application for the Audio4-5410 system. This is a console type Win32 program illustrating access to the USB monitor for program loading, memory and register access, flash ROM monitor and firmware updates,

and access to the DSP user application. All options are available by selecting a single character from the menu. The user interface looks as follows:

```
Audio4-5410 Test rev. 1.03
Audio4-USBm DLL Win 32-bit (dbg)
A4USBD32 rev. 1.07.86 (dbg)(ROM)   Audio4-5410 S/N: 0008 (Fw 3.17)
  1 - Open USB1           G - Run Target           8 - Set serial
  2 - Open USB2           H - Halt Target         9 - Load FW
  3 - Open USB3           V - Get Version         0 - Load User
  4 - Open USB4           N - Get Control         ? - Dbg Status
  C - Close USB           U - Inc Out Vol         F - Inc Inp Vol
  Q - Exit test           D - Dec Out Vol         J - Dec Inp Vol
  P - Play testx.pcm      L - Load codec54.out   E - Change Osc
  R - Rec testx.pcm      K - Read Ptrs           5 - Enable RAM
  S - Stop oper           Z - Inc Rate            6 - Enable ROM
  A - PlayLoop            X - Dec Rate            7 - Test MIPs
  T - Toggle dpy         M - Play Sinewave      / - Store DAT to ROM
  B - PC Loopback        W - Toggle Input
```

3.6.1 - General options

Test options for the test program control and access to the USB monitor:

- 1,2,3,4 - open connection to the USB device
- C - close connection to the USB device
- Q - exit test program
- G - go, run target code
- H - halt, stop execution of the target code
- L - load user application from the TI's COFF file. Default name is codec54.out, but it can be set with the command line option -Ffilename, while invoking the audio54.exe
- ? - display debug status of the user application. This state can be changed by the debugger (Code Composer) executed in parallel.

3.6.2 - Flash programming options

Audio54.exe allows for the flash ROM initialization and modification. Following options execute complex operation on the interface. Any flash ROM modification will reload the DSP with the program allowing the flash erasing and programming. This will temporarily disable USB monitor in the ROM. ROM control options:

- 8 - set serial number of the Audio4-5410 device. This number is displayed on the third line of the user interface. Serial number is used to identify multiple

devices attached to the same host computer. It has to be different for every device, and it should match the label on the bottom of the device.

- 9 - load firmware from the file a5410rom.dat The .dat file contains ROM image created from the a5410rom.out COFF. 32 K bytes of the ROM will be erased, and a new monitor image will be loaded.
- 0 - load user program. Option will ask for the file name to be loaded. It needs to be .dat file created with the out2dat.exe. Monitor code is preserved, and image of the user code is stored in the ROM. This image will be used by the monitor code to initialize DSP RAM during the power-up sequence.
- / - store DAT to ROM. Option allows to store DAT file into the flash ROM between addresses 0x18000 and 0x1FFFF.

3.6.3 - DSP application control options

Options for the control of the DSP user program "codec54.out":

- V - get user program version
- N - get value of the control words. Control words represent values used for the codec's gain control
- U - increment output volume. Modifies control words, by selecting lower attenuation for the output signal. Control words for both codecs are changed.
- D - decrement output volume. Modifies control words, by selecting higher attenuation for the output signal. Control words for both codecs are changed.
- F - increment input volume. Modifies control words, by selecting higher gain for the input signal. Control words for both codecs are changed.
- J - decrement input volume. Modifies control words, by selecting lower gain for the input signal. Control words for both codecs are changed.
- K - read pointers for the play and record buffers. Data structure is defined in the DSP user code.
- P - read data from the file test1.bin and send to codec 1 (stereo output B and headphones), and from the file test2.bin, to be sent to codec 2 (stereo output A).
- R - receive stereo data form codec 1 and 2, and write to the file test1.bin and test2.bin.
- A - play data from the files test1.bin and test2.bin in a loop
- M - generate sine wave patterns for codec 1 (stereo output B and headphones)
- S - stop play, record, or sine wave generation operation.
- E - toggle oscillator selection
- Z - increase sampling rate

- X - decrease sampling rate
- T - toggle status display for play and record operation
- B - enable digital loop back at the host computer side. It will "record" data from the USB device and send it back as a "play" data.
- W - toggle input will program codecs to use signal from primary or secondary inputs. Primary inputs are used for line in level signals. Secondary inputs can be connected to the optional microphone daughter board.

3.7 - COFF conversion utility.

Out2dat.exe is used to create two formats of the data.

First is used to load the USB monitor code into the flash. Data is organized as a sequence of two words - address/value. This instructs loader code where to store each word in the flash ROM. Sequence of address/value words is preceded with a standard Code Composer DAT header, so file can be used with the Code Composer's File I/O feature.

Second format contains sections of the data in a format understandable by the USB monitor loader. Sections of data are loaded into ROM at address 0x8200, and can be loaded into DSP RAM during the power-up sequence, and automatically started. Each section of data has a triple word header: length, address, space. If the length is equal to zero, this indicates end of the data to load into RAM, and following word is a entry point of the user application. For the non-zero length, header is followed by "length" words of data. Address defines starting point to load the data, and space corresponds to the memory area to load the data: 0 - program, 1 - data, 2 - I/O.

Conversion program is invoked from the console window with 2 or 3 parameters:

```
out2dat filename.out filename.dat [fwFlag]
```

- fileName.out - name of the TI's COFF file to be converted
- fileName.dat - name of the DAT file to be created
- fwFlag - if specified, program will create DAT file in the first format, otherwise .dat file will be created in the second format.

“Block Diagram” on page 38

“USB, Host Port Interace” on page 39

“CodecA, McBSPs” on page 40

“Codec B, Headphones output” on page 41

“Memory, bus control, PLD” on page 42

“Power, power-on reset” on page 43

“Microphone daughterboard” on page 44

“Quad microphone adaptor” on page 45

““ETA-5410 daughter board” on page 46

“5V analog charge pump” on page 47

“PLD Equations” on page 48

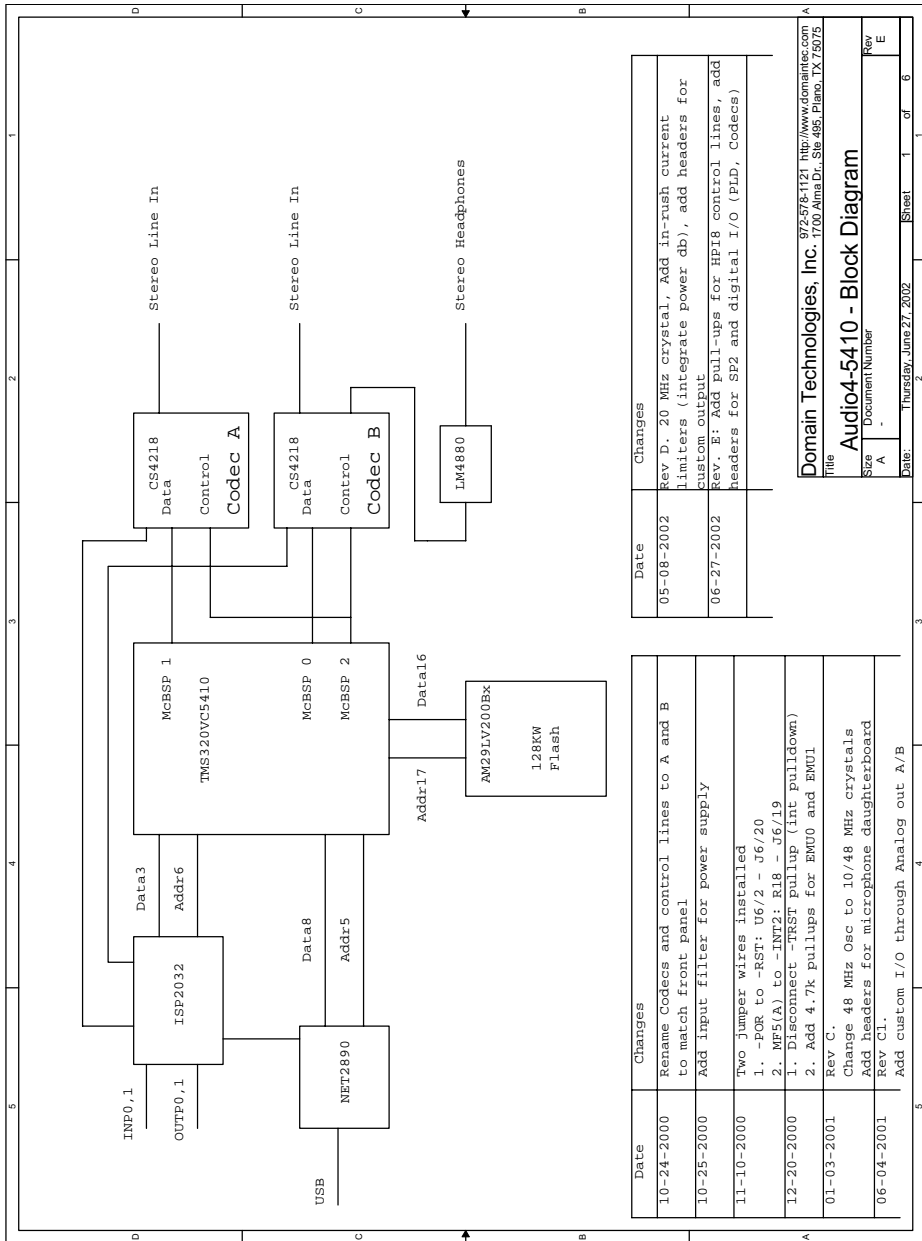


FIGURE 4.1. Block Diagram

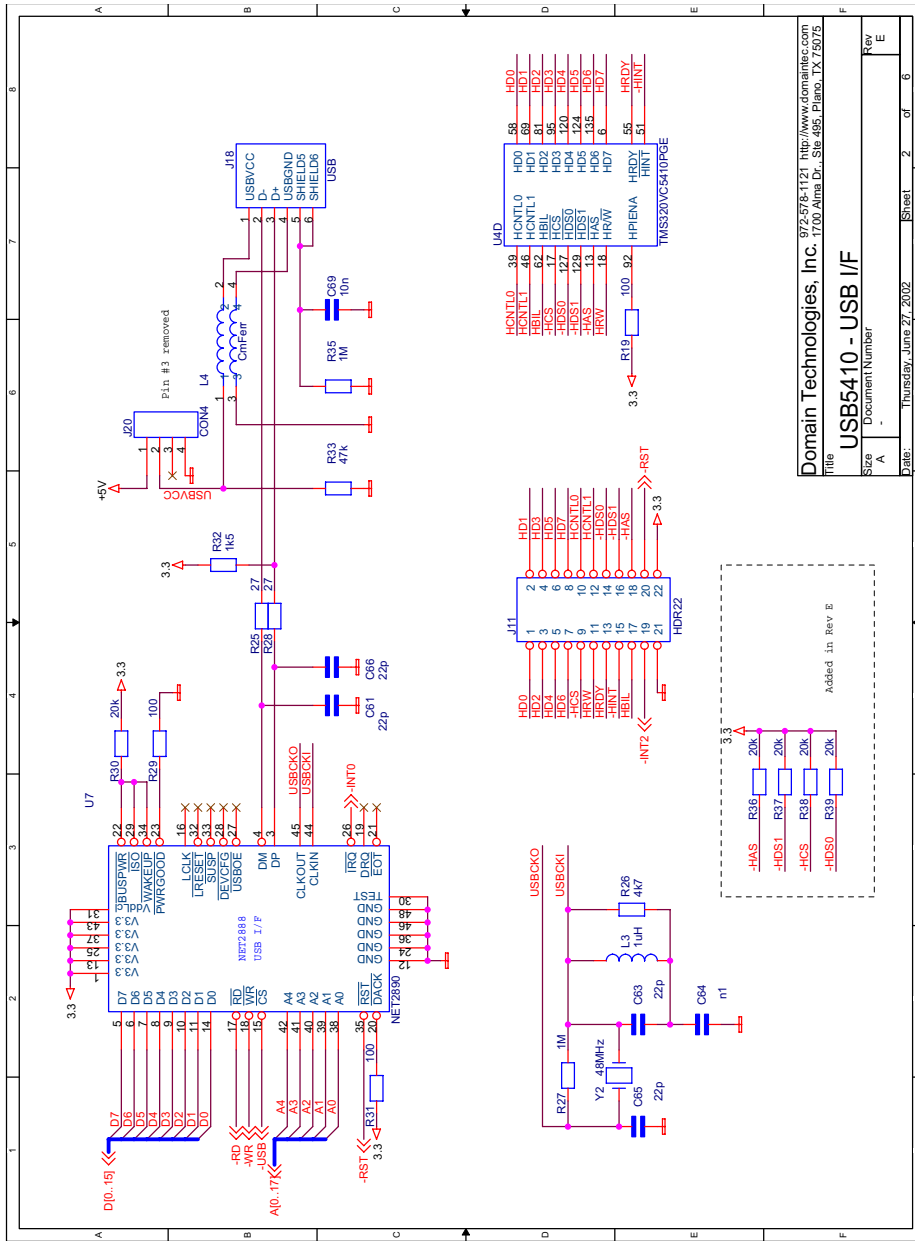


FIGURE 4.2. USB, Host Port Interface

Domain Technologies, Inc. 0925765-1191 <http://www.domaintech.com>
 17100 Almeta Dr., Ste 405, Plano, TX 75075

Title: **USB5410 - USB I/F**

Size: _____ Document Number: _____
 Date: Thursday, June 27, 2002 Sheet: 2 of 6

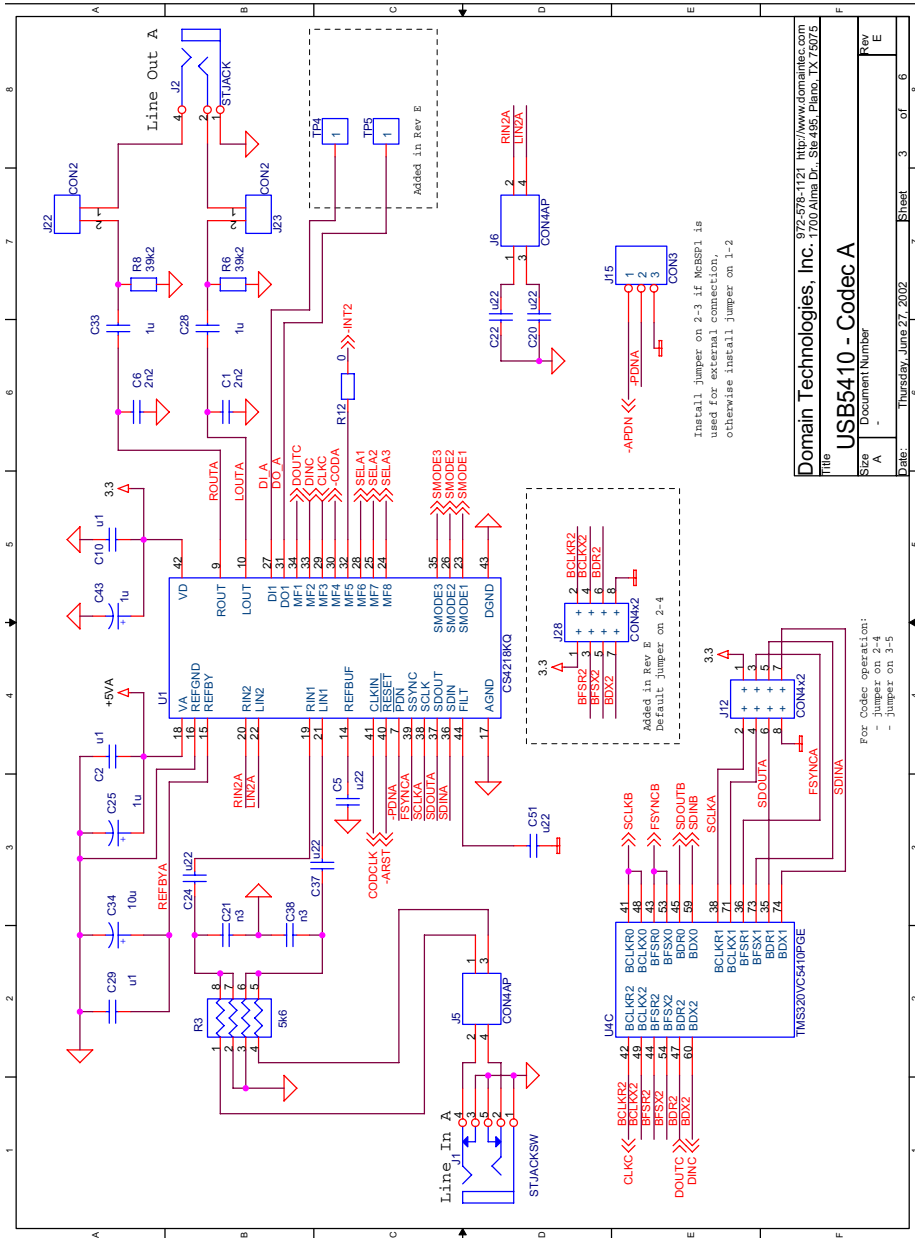
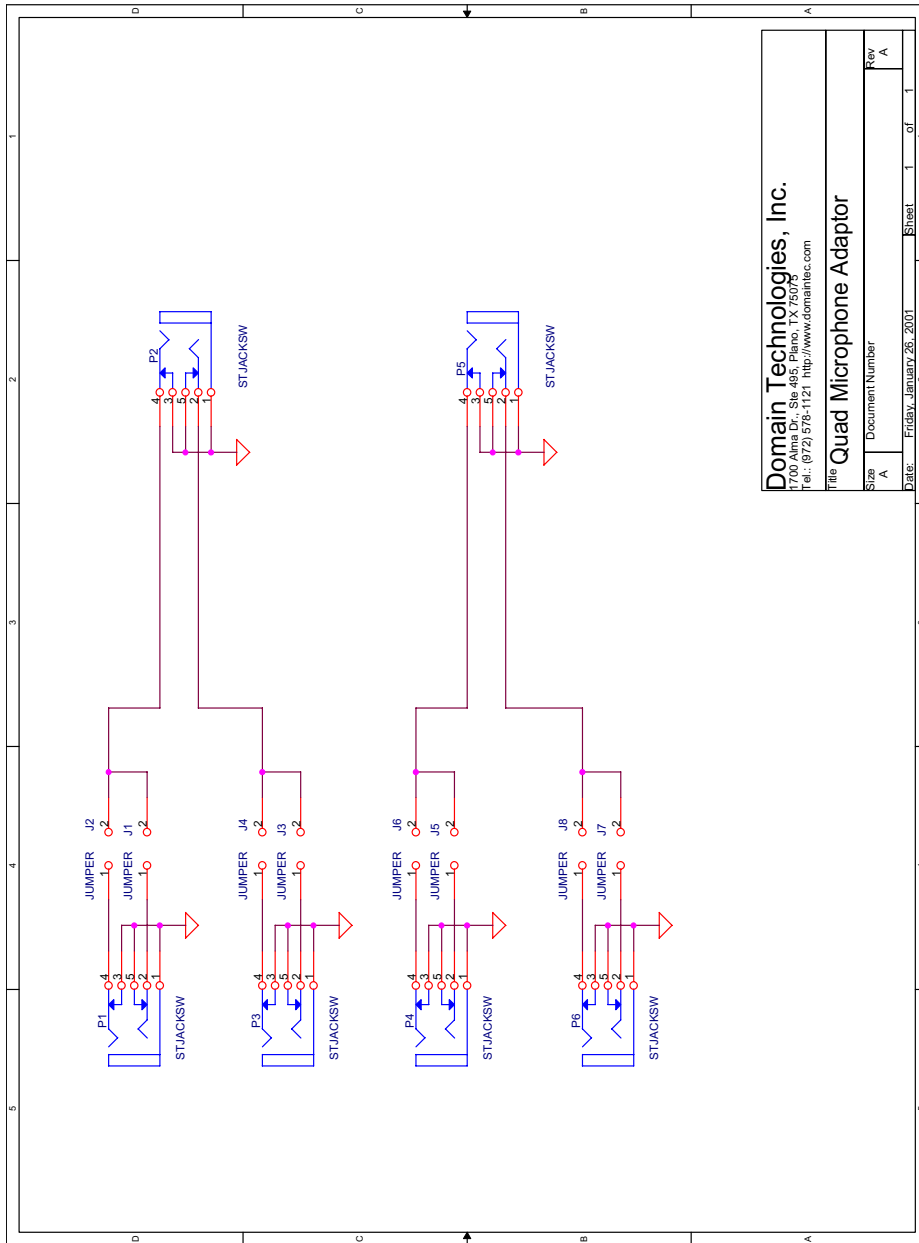


FIGURE 4.3. Codec A, McBSPs

Domain Technologies, Inc. 972-578-1121 http://www.domaintec.com	
Title	USB5410 - Codec A
Size	Document Number
A	
Rev	E
Date	Thursday, June 27, 2002
Sheet	3 of 6



Domain Technologies, Inc.
 1700 Alma Dr., Ste 495, Plano, TX 75075
 Tel.: (972) 576-1121 <http://www.domaintec.com>

Title: Quad Microphone Adaptor

Size	Document Number	Rev
A		A
Date:	Friday, January 26, 2001	Sheet 1 of 1

FIGURE 4.8. Quad microphone adaptor

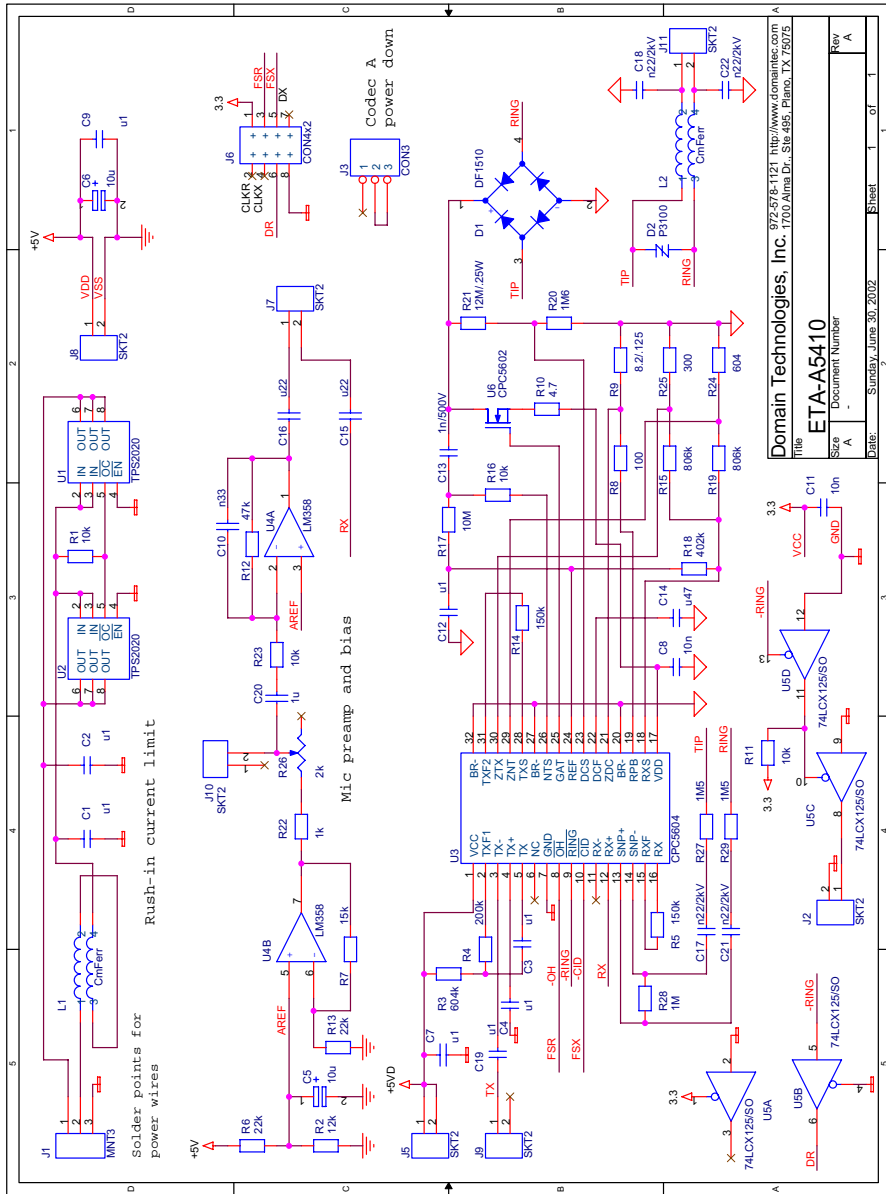


FIGURE 4.9. ETA-5410 daughter board

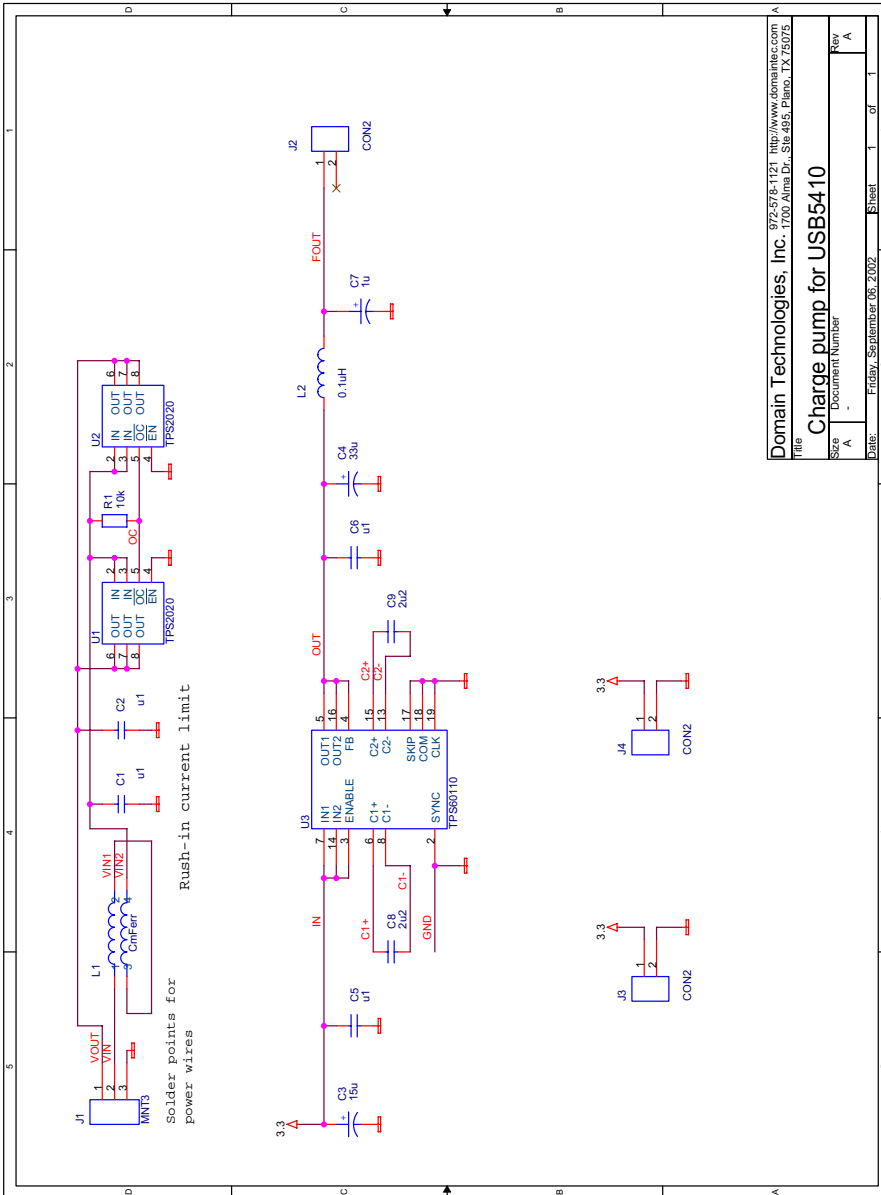


FIGURE 4.10. 5V analog charge pump

4.1 - PLD Equations

```

MODULE usb5410
TITLE 'USB 5410'
    !MSTRB,!IOSTRB          pin 15,16;
    A5,A2,A1,A0            pin 12,11,10,9;
    RW                     pin 19;
    !ROM,!USB              pin 4,3;
    !RD,!WR                 pin 13,14;
    E12M,E10M              pin 2,1      istype 'reg,buffer';
    LED0R,LED0G            pin 26,25   istype 'reg,buffer';
    LED1R,LED1G            pin 24,23   istype 'reg,buffer';
    SMODE3,SMODE2,SMODE1  pin 44,43,42;
    !APDN                   pin 41      istype 'reg,buffer';
    !CODA                   pin 38      istype 'reg,buffer';
    !CODB                   pin 37      istype 'reg,buffer';
    SELA1,SELA2,SELA3      pin 36,35,34 istype 'reg,buffer';
    SELB1,SELB2,SELB3      pin 33,32,31 istype 'reg,buffer';
    D0,D1,D2                pin 20,21,22;
    LATCH_LED0 = IOSTRB & A5 & !RW & !A2 & !A1 & !A0; "0x20 - LED0.
    LATCH_LED1 = IOSTRB & A5 & !RW & !A2 & !A1 & A0; "0x21 - LED1.
    LATCH_SELA = IOSTRB & A5 & !RW & !A2 & A1 & !A0; "0x22 - SELA.
    LATCH_SELB = IOSTRB & A5 & !RW & !A2 & A1 & A0; "0x23 - SELB
    LATCH_ACTRL= IOSTRB & A5 & !RW & A2 & !A1 & !A0; "0x24 - cod ctr
    LATCH_OSC = IOSTRB & A5 & !RW & A2 & !A1 & A0; "0x25 - osc ctrl

```

Equations

```

USB      = IOSTRB & !A5; "Address decode -cs to usb dev
ROM      = MSTRB;
WR       = !RW;
RD       = RW;
LED0R    := D1;
LED0G    := D0;
LED0R.C  = !LATCH_LED0;
LED0G.C  = !LATCH_LED0;
LED1R    := D1;
LED1G    := D0;
LED1R.C  = !LATCH_LED1;
LED1G.C  = !LATCH_LED1;
SMODE3   = 1; //SM4, Master, 32 BPF
SMODE2   = 0;
SMODE1   = 0;
SELA1    := D2;
SELA2    := D1;
SELA3    := D0;
SELA1.C  = !LATCH_SELA;
SELA2.C  = !LATCH_SELA;
SELA3.C  = !LATCH_SELA;
SELB1    := D2;
SELB2    := D1;
SELB3    := D0;
SELB1.C  = !LATCH_SELB;
SELB2.C  = !LATCH_SELB;
SELB3.C  = !LATCH_SELB;
APDN     := D2;
CODA     := D1;
CODB     := D0;
APDN.C   = !LATCH_ACTRL;
CODA.C   = !LATCH_ACTRL;
CODB.C   = !LATCH_ACTRL;
E12M     := D1;
E10M     := D0;
E12M.C   = !LATCH_OSC;
E10M.C   = !LATCH_OSC;

```

END