# AN1250
# APPLICATION NOTE

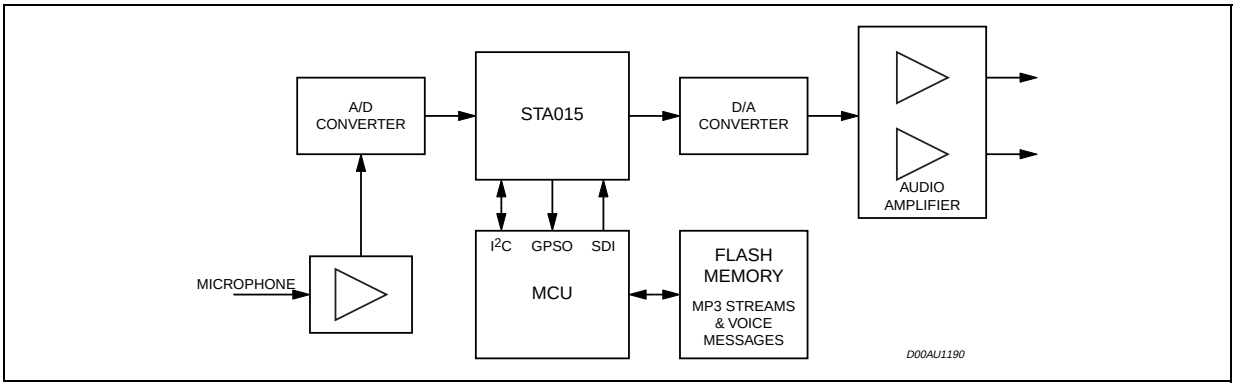
# STA014/STA015 MPEG LAYER III DECODER AND ADPCM CODEC

**by Marco Veneri**

## 1. INTRODUCTION

The STA015 device is an MPEG Layer III audio decoder with ADPCM compression / decompression capabilities and BYPASS mode for auxiliary audio sources post-processing

**Figure 1. Typical Application**




The Table 1 summarize the available sample frequencies and bitrates. The device is able to recognize bitstream parameters and automatically configure the embedded PLL in order to generate the required clock frequencies for the digital to analog converter. Different reference crystal frequencies are supported.

**Table 1. MPEG Layer III standard**

| | Specification | Sampling Rates (KHz) | Bitrates (kbit/s) |
|---|---|---|---|
| MPEG1 | ISO/IEC 11172-3 | 32 – 44.1 – 48 | Free, 32, 40, 48, 56, 64, 80, 96, 112, 128, 160, 192, 224, 256, 320 |
| MPEG2 | ISO/IEC 13818-3.2 | 16 – 22.05 – 24 | Free, 8, 16, 24, 32, 40, 48, 56, 64, 80, 96, 112, 128, 144, 160 |
| MPEG2.5 | ISO/IEC 13818-3.2 | 8 – 11.025 - 12 | Free, 8, 16, 24, 32, 40, 48, 56, 64 |

The ADPCM engine includes different compression algorithms:

- DVI (32 kbps)
- G723_24 (24 kbps)
- G721 (32 kbps)
- G723_40 (40 kbps)

Both mono and stereo channels are supported with sample frequency of 8, 16 or 32 KHz. A special frame format

can be selected for the compressed output data stream in order to reduce sensitiveness to bitstream errors while decoding and to enable fast-forward/backward capabilities. Embedded in the device a digital volume attenuator and a dual-band equalizer are available, eliminating the hassle and additional cost of an external audio processor. The device is fully operated using the standard I2C bus: interface configuration, PLL setup, operation mode and volume / tones settings can be easily programmed writing to some specific register. It's also possible to access a small 'DSP development RAM' (256 words) in order to modify the firmware code adding, eventually specific functions or requirements.

Please contact the local ST branch to get additional information about

## 2. ADPCM DECODING

ADPCM decoding is always accomplished using the SDI input interface: data must be transmitted to the device in the same way (and with the same configuration options) as an MP3 stream. Before starting the decoding process the ADPCM engine must be properly configured in order to meet encoded bitstream syntax and characteristics. The configuration phase involves the following registers:

- ADPCM_FRAMSE_SIZE (addr. BDh)
- ADPCM_SAMPLE_FREQ (addr. 53h)
- ADPCM_CONFIG (addr. B8h)

When the decoding phase is ongoing it's possible to adjust the volume and tone control of the output signal using the following registers:

- TREBLE_FREQUENCY_LOW (addr. 77h)
- TREBLE_FREQUENCY_HIGH (addr. 78h)
- TREBLE_ENHANCE (addr. 7Bh)

- BASS_FREQUENCY_LOW (addr. 79h)
- BASS_FREQUENCY_HIGH (addr. 7Ah)
- BASS_ENHANCE (addr. 7Ch)

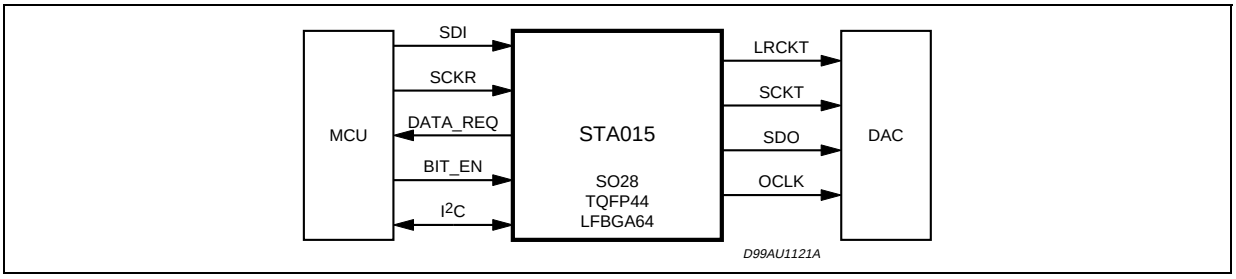
- TONE_ATTEN (addr. 7Dh) (this single control is used for both left and right channels)

## 3. ADPCM ENCODING

STA014/STA015 can be configured in order to get the data to be compressed from two different interfaces: SDI and ADC.ADC interface can be programmed in order to accept most analog to digital converter formats, including different word and slot length. The oversampling clock for the converter is generated by STA014/STA015 and will be automatically switched to the proper frequency according to the programmed sample rate and oversampling ratio. SDI interface is the same input interface used to feed MP3 bitstream. It's based on a simple 2-wire bus (SCKR and SDI) plus a feedback line (DATA_REQ) used to return embedded FIFO buffer status. Data must be sent MSb first and in 2's complement. To retrieve encoded data two different interfaces can be used as well: I2C and GPSO.This result in 4 different possible configurations to operate the device as an ADPCM encoding engine: each mode will be briefly explined in the following pages.

### 3.1 Mode 0 (from SDI to I2C)

**Figure 2.** Mode 0 block diagram



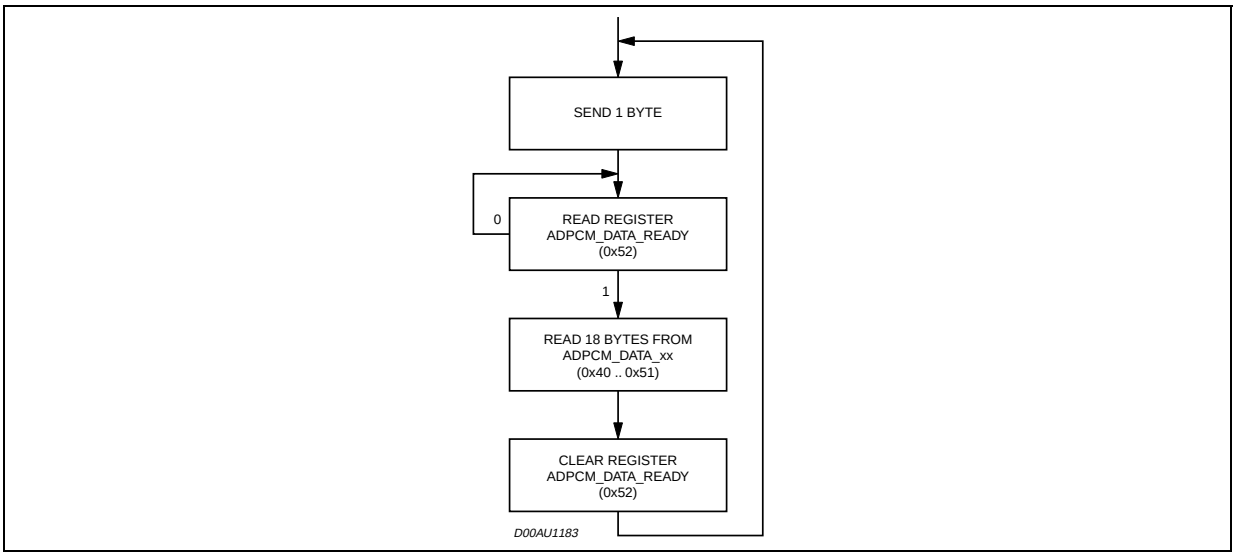

When STA014/STA015 is configured in mode 0 (default mode) it is pin to pin compatible with its predecessor STA013. The SDI input interface must be selected and encoded ADPCM data must be retrieved through a set of dedicated I2C registers using a polling technique based on the ADPCM_DATA_READY register. Both DVI and G726 pack algorithms are supported but the frame mode can be activated only for DVI: in order to be compatible with STA013 the encoder should be configured for 8 kHz, 16 bit mono input signal, as shown in Table 2.

**Table 2. Mode 0 (8kHz, 16 bit mono) configuration**

| Address | Name | Value |
|---|---|---|
| 16 | SOFT_RESET | 1 |
| 83 | ADPCM_SAMPLE_FREQ | 2 |
| 77 | CHIP_MODE | 2 (encoder mode)<br>3 (decoder mode) |
| 19 | PLAY | 0 (encoder mode)<br>1 (decoder mode) |
| 114 | RUN | 1 |

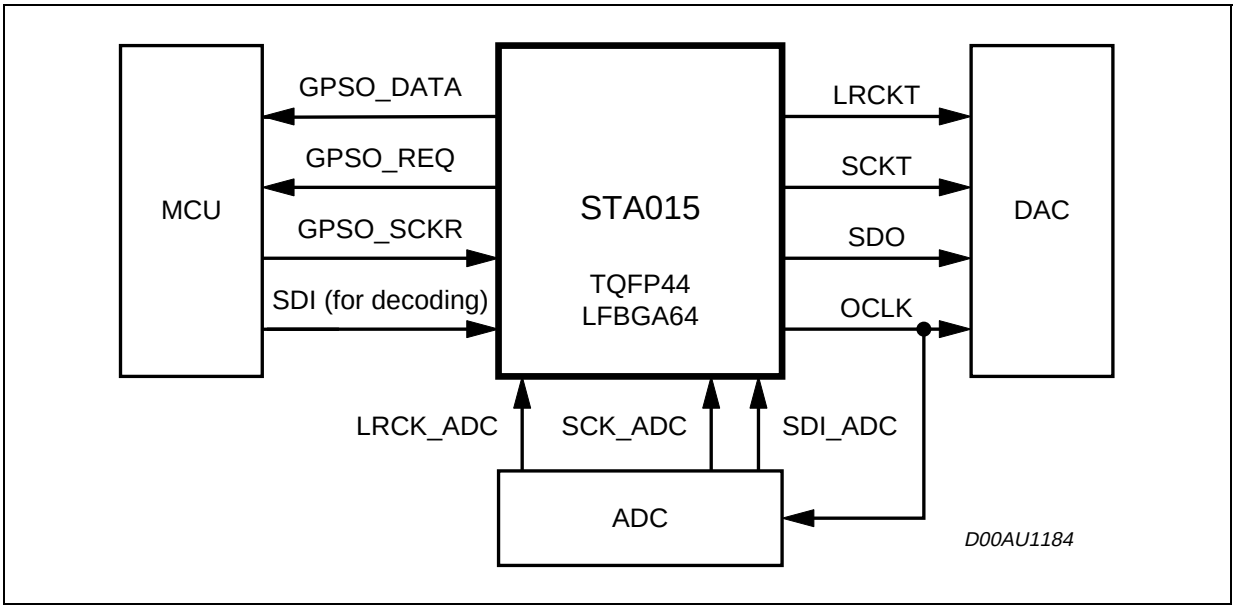
Mode 0 flow chart show the flow diagram that must be used in order to send uncompressed samples and retrieve encoded data.

**Figure 3. Mode 0 flow chart**




### 3.2 Mode 1 (from ADC to GPSO)

**Figure 4. Mode 1 block diagram**




When the device is configured in mode 1 the ADC interface is selected for incoming data and the GPSO interface for encoded output data. The GPSO_REQ signal should be connected to an interrupt source of the microcontroller; MCU should then provide the GPSO clock to retrieve encoded data.

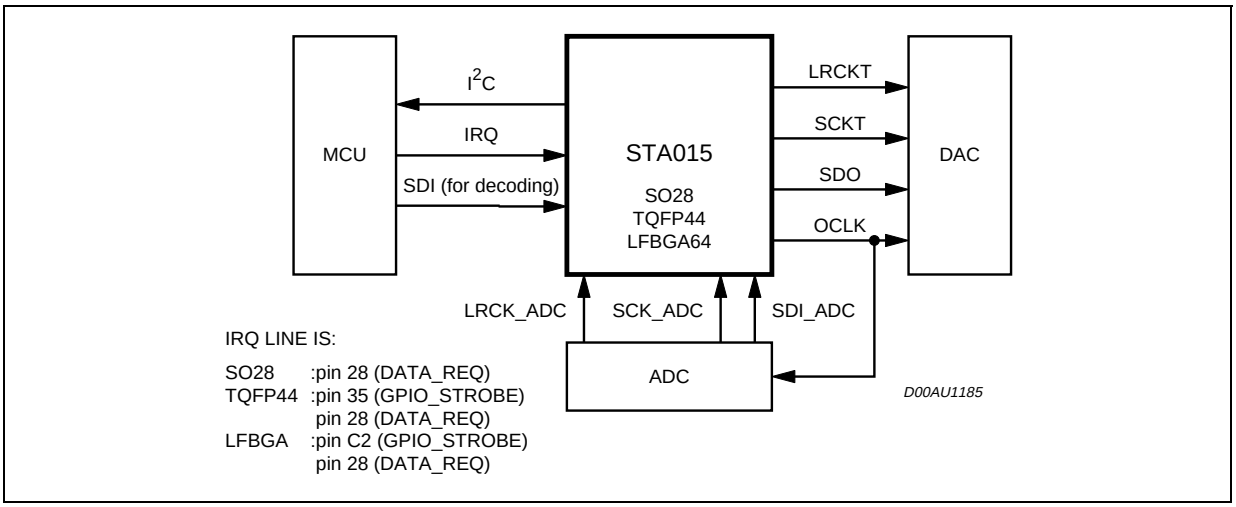
The configuration shown in Table 3 can be used to setup the device with G721 codec and for an 8kHz, 16 bit stereo input stream with frame packing capability.

**Table 3. Mode 1 (G721, 8kHz, 16 bit stereo + frame) configuration**

| Address | Name | Value |
|---|---|---|
| 16 | SOFT_RESET | 1 |
| 185 | GPSO_ENABLE | 1 |
| 187 | ADC_ENABLE | 1 |
| 192 | ADC_WLEN | 15 |
| 83 | ADPCM_SAMPLE_FREQ | 2 |
| 184 | ADPCM_CONFIG | 11 |
| 77 | CHIP_MODE | 2 (encoder mode)<br>3 (decoder mode) |
| 19 | PALY | 0 (encoder mode)<br>1 (decoder mode) |
| 114 | RUN | 1 |

### 3.3 Mode 2 (from ADC to I2C)

**Figure 5. Mode 2 block diagram**




This functional mode will enable ADC input interface for incoming data and I2C for encoded output data. Once 18 new ADPCM encoded data bytes are available an interrupt is issued and the ADPCM_DATA_READY is set to 1. The MCU must start reading the 18 ADPCM_DATA_xx registers as soon as the interrupt signal has been received and, after that, must execute the following two operations:

– clear ADPCM_DATA_READY (0x52)
– write 1 in CMD_INTERRUPT register (0x16)

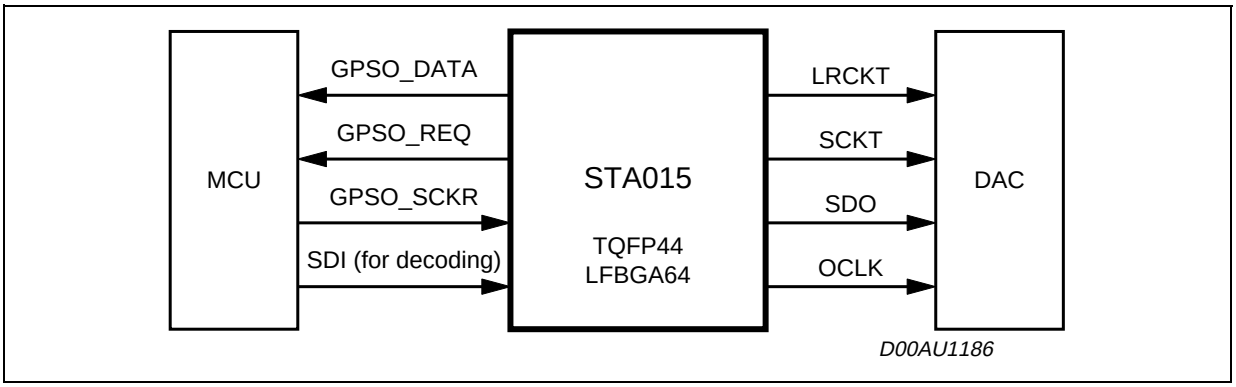
The interrupt pulse length may be programmed up to 128 DSP clock cycles using the ADPCM_INT_CFG (0xBE) register. The polarity of the interrupt signal can be programmed as well. For IRQ pin assignment refer to Figure 5.

**Table 4. Mode 2 (G721, 16kHz, 8 bit mono) configuration**

| Address | Name | Value |
|---|---|---|
| 16 | SOFT_RESET | 1 |
| 187 | ADC_ENABLE | 1 |
| 192 | ADC_WLEN | 7 |
| 83 | ADPCM_SAMPLE_FREQ | 10 |
| 184 | ADPCM_CONFIG | 8 |
| 77 | CHIP_MODE | 2 (encoder mode)<br>3 (decoder mode) |
| 19 | PALY | 0 (encoder mode)<br>1 (decoder mode) |
| 114 | RUN | 1 |

### 3.4 Mode 3 (from SDI to GPSO)

**Figure 6. Mode 3 block diagram**




Configuring the device in mode 3 incoming data are feed via SDI interface and encoded data are retrieved via GPSO interface. Since the SDI input interface is not well suited to receive a stereo signal only mono encoding is supported.

**Table 5. Mode 3 (G721, 16kHz, 8 bit mono) configuration**

| Address | Name | Value |
|---|---|---|
| 16 | SOFT_RESET | 1 |
| 185 | GPSO_ENABLE | 1 |
| 83 | ADPCM_SAMPLE_FREQ | 10 |
| 184 | ADPCM_CONFIG | 8 |
| 77 | CHIP_MODE | 2 (encoder mode)<br>3 (decoder mode) |
| 19 | PALY | 0 (encoder mode)<br>1 (decoder mode) |
| 114 | RUN | 1 |

## 4. ADPCM FRAME PACKING

The STA014/STA015 can pack ADPCM encoded sample into frames for fast error recovering and fast forward capability. The frame size may be adjusted to match a trade-off between the bit rate overhead and the frame length. Any of the 4 algorithms may use the frame capability in stereo or mono encoding: bit AFM_EN of ADPCM_CONFIG register enable or disable this feature. The frame size (in bytes) can be computed as follow:

Frame size = (AFS * 90) + 108

Where AFS is the value programmed into the ADPCM_FRAME_SIZE register.

The frame always starts with a 12 bytes common header for both DVI and G726 pack and is followed by either:

– 6 bytes when DVI algorithm is selected
– 96 bytes when G726 pack algorithm is selected

The 12 bytes header structure has the layout shown in Table 6 Frame header structure. Table 7 gives an example of the relationship between the ADPCM frame size and the corresponding bitrate overhead at the encoder output, assuming a compression ratio equal to 4 and a 16kHz 16 bit mono input bitstream.

**Table 6. Frame header structure**

| Index | Description | Content |
|---|---|---|
| 0..4 | Sync word | 0x5354445649 |
| 5..7 | Frame counter | |
| 8 | Frame size | ADPCM_FRAME_SIZE |
| 9 | Sample size | ADC_WLEN |
| 10 | Sample frequency | ADPCM_SAMPLE_FREQ |
| 11 | ADPCM configuration | ADPCM_CONFIG |

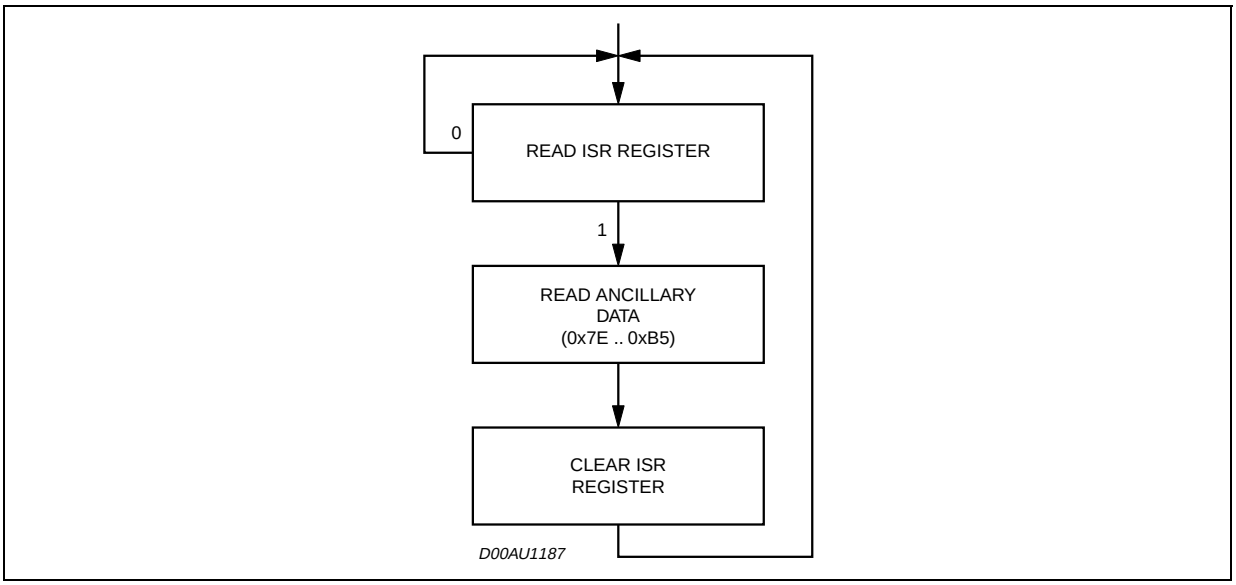
**Table 7. Bitrate overhead**

| Value | Header Size | Frame Size | Input Bitrate (kbit/s) | Output Bitrate (Kbit/s) | Bitrate Overhead | Audio Frame |
|---|---|---|---|---|---|---|
| 20 | 108 bytes | 1908 | 256 | 68 | 6.00 % | 225 ms |
| 40 | 108 bytes | 3708 | 256 | 68 | 3.00 % | 450 ms |

## 5. ANCILLARY DATA

Ancillary data are optional user data that can be included into MP3 frames. STA014/STA015 contains 56 consecutive 8-bit registers corresponding to the maximum number of ancillary data that can be contained in an MPEG frame: those registers may be read via I2C bus in order to retrieve this auxiliary information. The ANCCOUNT_L/ANCCOUNT_H registers contain the number of ancillary data bits available within the current MPEG frame. Moreover a status bit (see bit 0 of ISR register) is used to signal data availability: this flag should be continuously polled by MCU in order to retrieve data when available. After all data has been retrieved MCU should reset this bit in order to inform the embedded DSP and allow new data buffering. If this operation is not performed an error is reported in the ERROR_CODE register.

**Figure 7. Ancillary data retrieving**




## 6. EVALUATION BOARD LAYOUT

### 6.1 LPT Interface

In order to allow proper communication between the 3V evaluation board and a standard LPT interface the STA014/STA015 evaluation kit includes also the LPT interface board shown in Figure 7.

Basically the interface will perform a signal level translation between 3V and 5V in both directions.

Both 5V and 3V regulators are available on the interface: moreover the regulated supplies are available on the J2 connector in order to supply also the STA014/STA015 evaluation board.

The board is delivered with both voltage regulators mounted: as described in LPT interface layout “Configurations” note this mode requires a single 7V power supply.

**Figure 8. LPT interface layout**

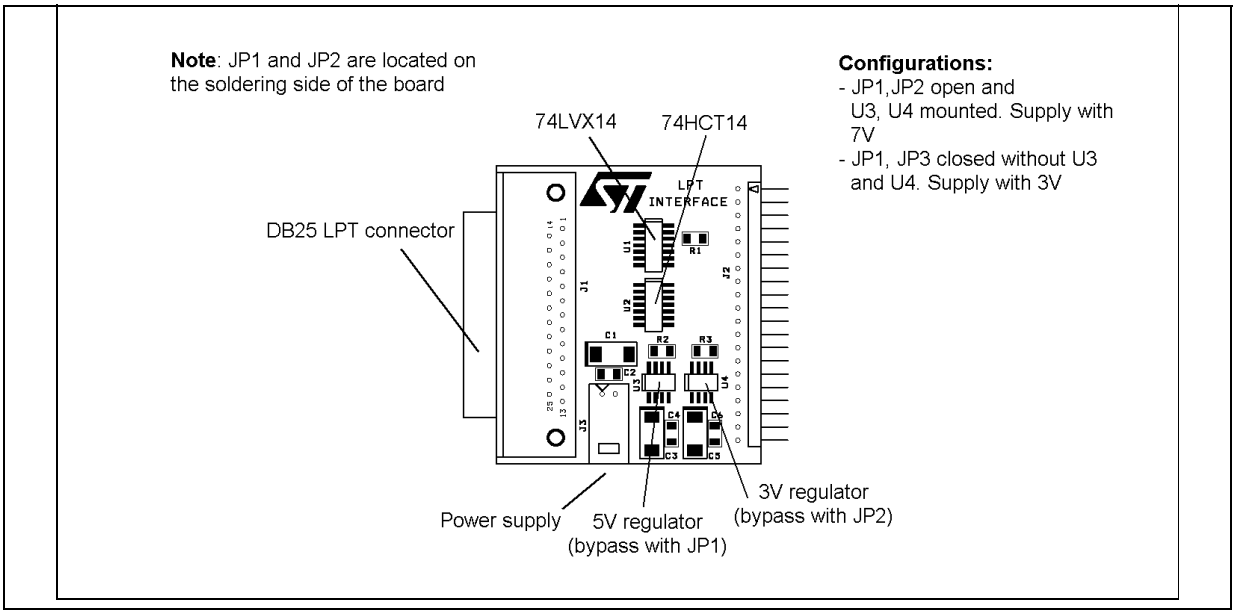
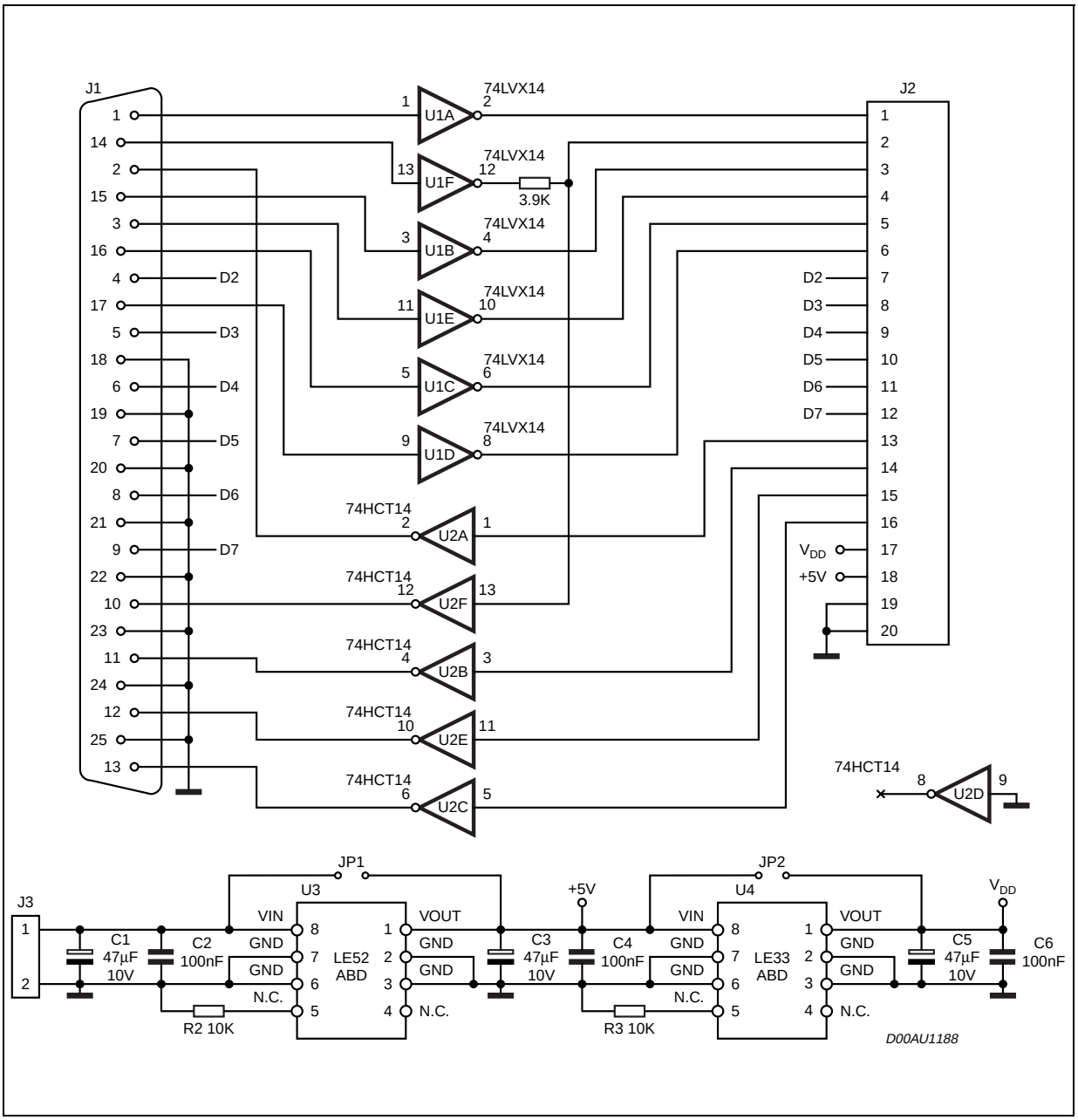


**Figure 9. LPT interface schematic**

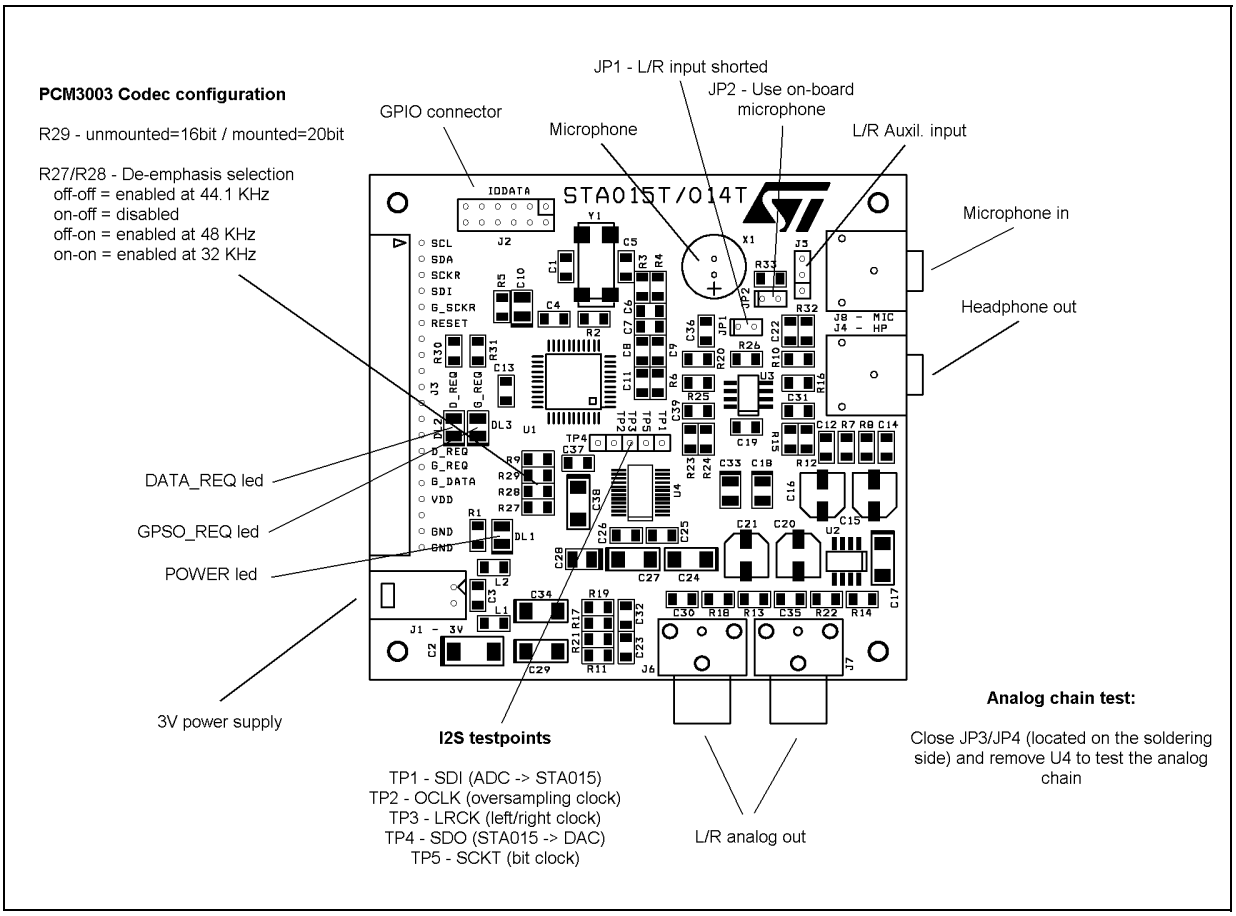



### 6.2 STA014/STA015 Evaluation board

The STA014/STA015 evaluation board layout is shown in Figure 10. If the board is operated without the ST provided LPT interface a single 3V power supply must be used (J1 connector): DL1 led will indicate supply presence.

All the STA014/STA015 useful signals and control lines are available on the J3 20 pin connector: no buffering exist between device and connector pins.

**Figure 10. STA014/STA015 eval. board layout**




The codec device PCM3003 can be configured as 16/20 bit and the embedded de-emphasis feature can be used in order to perform MP3 bitstream equalization. To configure this options three resistors are available: R27, R28 and R29 (see related note in Figure 10. No active control has been implemented in the STA014/STA015 control panel.

To encode from the onboard microphone jumper JP2 must be closed. Jumper JP1 can be used to shorten left and right input channels together ( this is mandatory if recording in mono mode since, in this case, the input channel is the right one).

The microphone input gain can be adjusted changing the value of R12, R16 (for right channel) and R23, R25 (for left channel) according to the following formula:

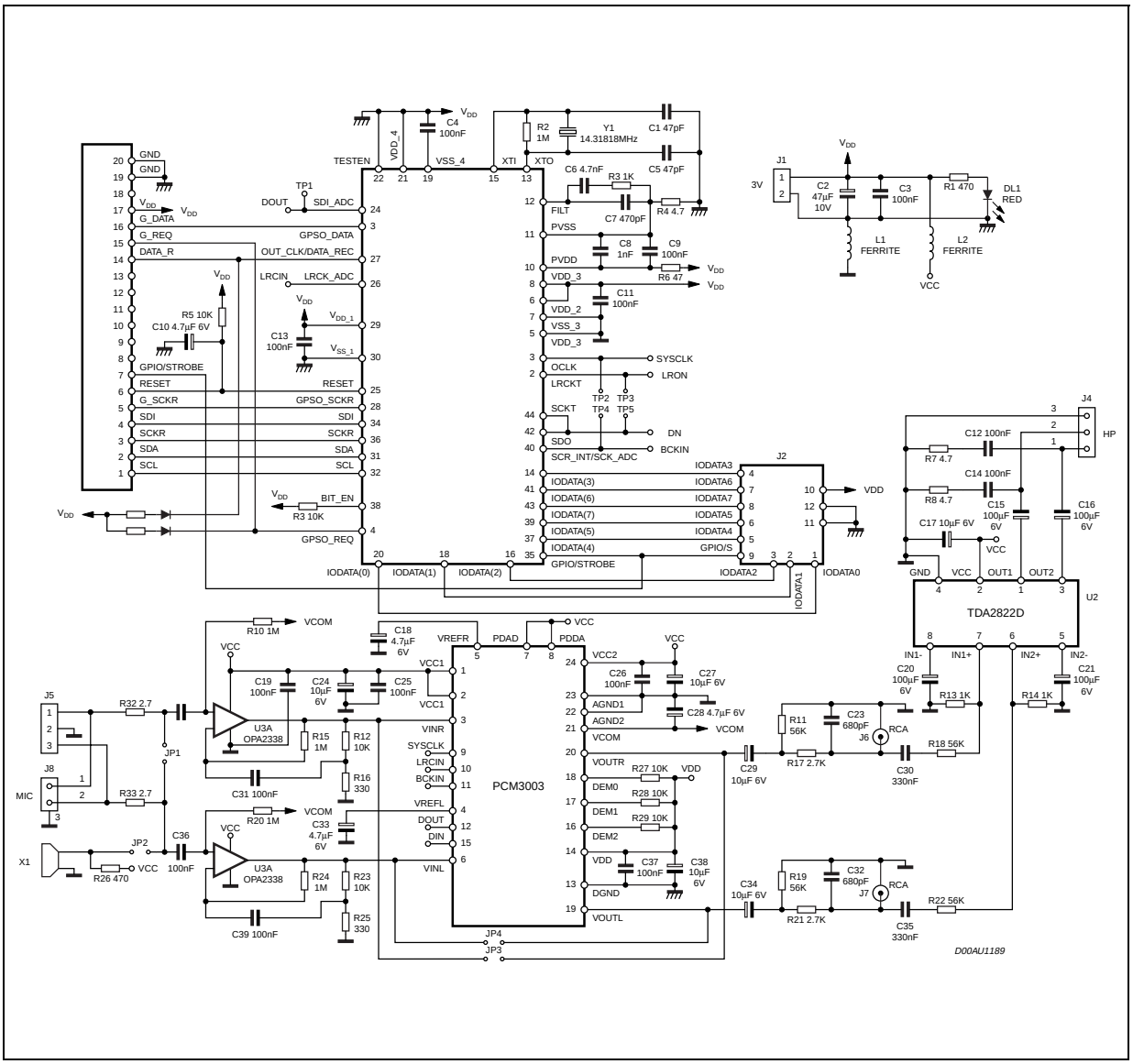
GainL = 1 + R12/R16 (right channel)

GainR = 1 + R23/R25 (left channel)

Entering from an external source it could be useful to short the R12 and R23 resistors in order to obtain unity gain, provided the external signal is already preamplified.

Please note that signal saturation at the input stage of the device could produce a very noisy signal when encoding in DVI ADPCM algorithm and could prevent any successful encoding if G72x algorithms are used.

**Figure 11. STA014/STA015 evaluation board schematic**




## 7. EVALUATION BOARD SETUP

To setup the evaluation kit some simple operations are required:

1. Connect STA014/STA015 evaluation board with the LPT interface board
2. Connect J1 DB25 connector to the PC LPT port
3. Supply the kit with a 7V regulated voltage applied to J3 connector (on LPT interface board)
4. Power up the kit (the DL1 led should light)
5. Start the STA014/STA015 control panel software

In order to perform a quick connection check the TEST button on the 'Register I/O' tab can be used.

Should the test fail check the PC parallel port settings: most PC, infact, can configure their LPT interface in different modes (SPP/EPP/ECP) and it could happen that some of these configurations are not well suited to control our evaluation kit

Moreover, in order to be able to operate the PC parallel port at maximum speed, it's important to properly setup the 'I/O recovery time' in the machine BIOS; this setting is tipycally available in the 'Chipset Features' page. The minimum available value must be set for both 8 and 16 bits I/O recovery time; choose NA (not available) option if present.

Improperly settings of these options could prevent the LPT interface to work up to the minimum required speed in order to correctly decode 320kbit/s MP3 streams and to encode/decode highest bitrate ADPCM messages.

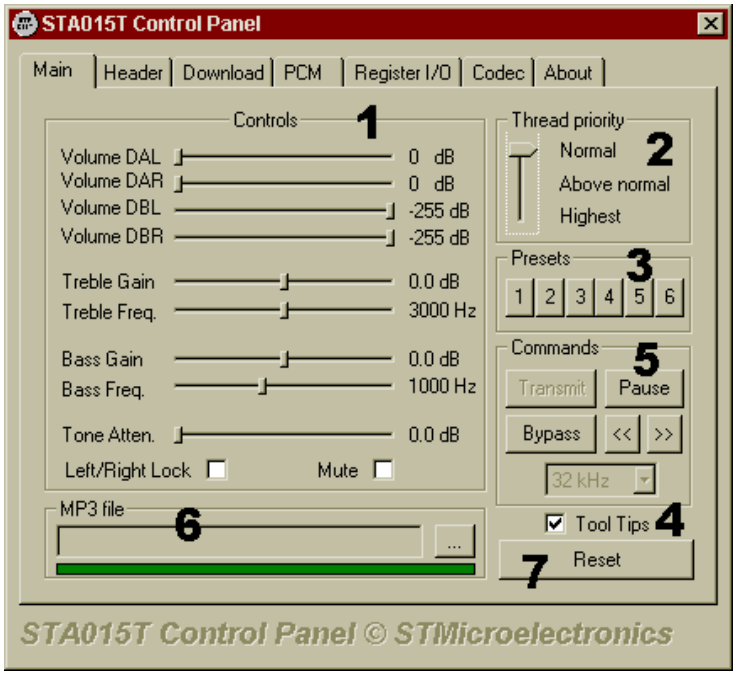
## 8 CONTROL SOFTWARE

In order to control and evaluate the STA014/STA015 board a dedicated control panel has been developed: it runs on Win95/98 operating system and allows both MP3 and ADPCM capabilities to be evaluated.

Hereby is a quick description of the various controls available on it.

### 8.1 Main page

**Figure 12. Main page**




§ 1 : The 'Controls' section groups the MP3 audio controls such as volumes, tones and global attenuation. The left/right volume sliders can be linked together checking the 'Left/Right Lock' box. The dual band equalizer can be adjusted both in frequency and enhancement. Special care must be taken before enhancing bass/treble: infact, in order to avoid any distortion, the 'Tone Attenuation' slider must be previously set to the maximum value of bass/treble desired enhancement.

§ 2 : Using this 3 position slider it's possible to change the priority of the MP3 bitstream transmitting thread. Higher priority will avoid audio gaps when switching to other applications but, of course, will slow down the machine.

§ 3 : Some default control presets are already stored in those 6 memories. To store current controls position just press down one of those buttons until it's automatically released.

§ 4 : Use this check-box to globally enable or disable the tooltip function

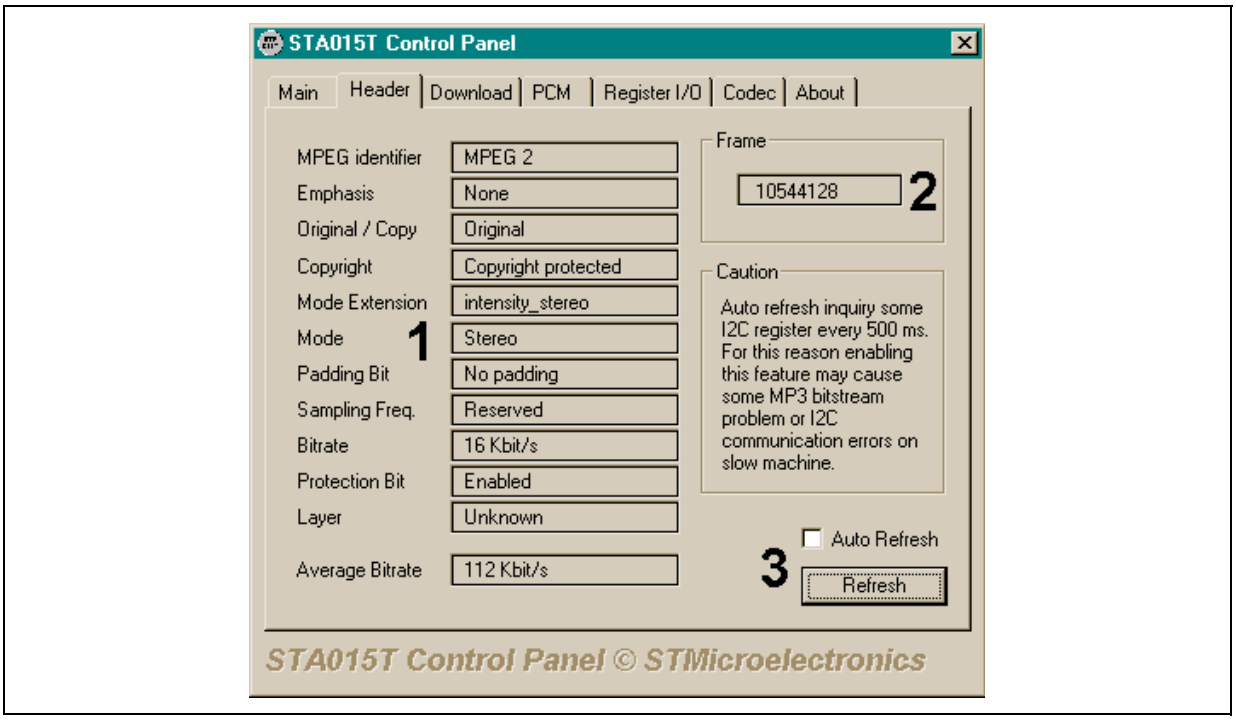
§ 5 : Here you can find the basic controls to handle the MP3 file playback or to select the BYPASS mode

§ 6 : Using the '…' button it's possible to browse for the desired MP3 file. In order to start decoding a bitstream it's also possible to drag & drop the desired file into the filename box.

§ 7 : Using this button it's possible to issue a device SOFT_RESET in order to restore default settings. To issue an hardware reset use the relative button in the 'Register I/O' page

### 8.2 Header page

**Figure 13. Header page**



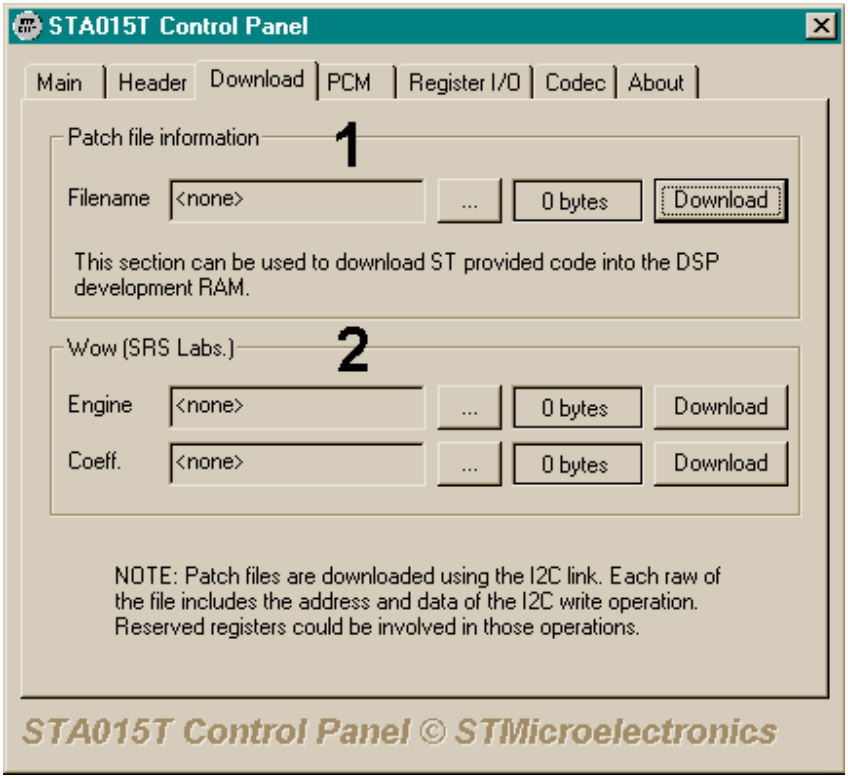

§ 1 : All this information comes from the frame header of the decoding MP3 bitstream. Each time a new frame has been decoded by the device both the header information and the frame counter (see next point) are updated into the device memory: the header page can be updated using the 'Refresh' button.

§ 2 : The frame counter reports the number of frames decoded since the playback process started.

§ 3 : Using the 'Auto Refresh' function it's possible to automatically update the header information and the frame counter every 500 ms. On slower PC this could cause some audio noise or distortion due to the possible bitstream interruption during the page update process.

### 8.3 Download page

**Figure 14. Download page**

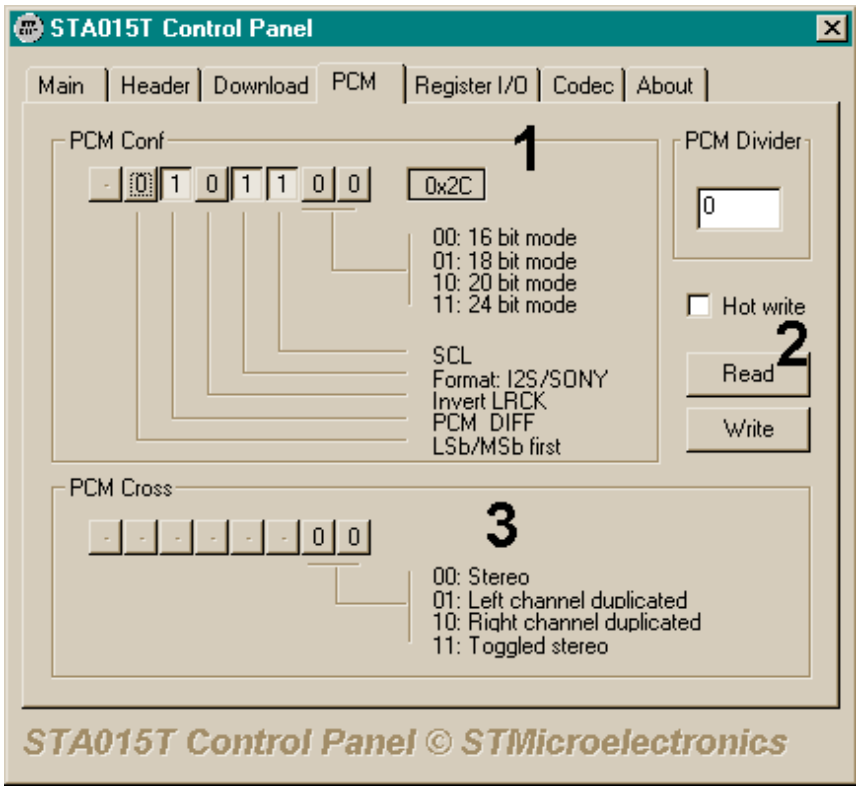



§ 1 : The device also embeds a small development RAM in order to add/develop new audio features a/o processing algorithms. Development of new features is intended for ST only and not for customers. Customers will only be allowed to use expansion files when available and officially released by ST. Expansion files can be downloaded using this section: each file is downloaded using the I2C link and different reserved registers.

§ 2 : This section just duplicate the previous one and offers handy controls when using expansion kits which include more than one file.

### 8.4 PCM page

**Figure 15. PCM page**



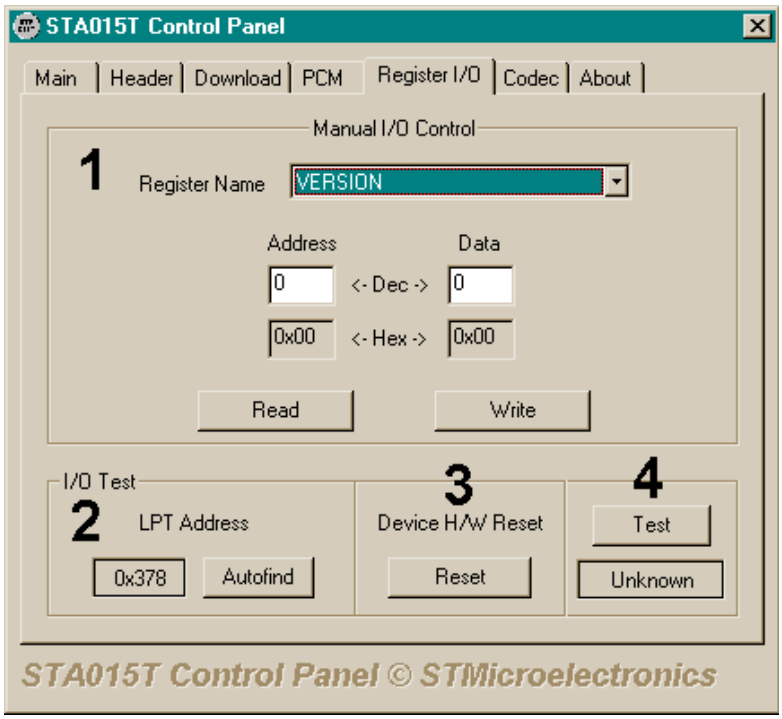

§ 1: Using the PCM page it's possible to change the serial output interface of the decoder in order to fit different D/A converter protocols. This page has been added just to show the various configuration options but, of course, there's no need to use it with the ST provided STA014/STA015 evaluation kit. The output interface of the decoder, infact, is already configured for the actual D/A converter used on the board.

§ 2: Instead of using the read/write buttons it's possible to enable the 'Hot write' function in order to change the device configuration on the fly every time a bit of the PCM configuration register is changed.

§ 3: This 2 bit register (PCMCROSS) can be used to select the decoder output mode (basically stereo/ mono).

### 8.5 Register I/O

**Figure 16. Register I/O page**




.§ 1 : Using the controls in this section it's possible to manually access all the device control/configuration registers. Address and data are showed both in decimal and hexadecimal notation. The register can also be selected using the upper side selection list-box in which symbolic names are used.

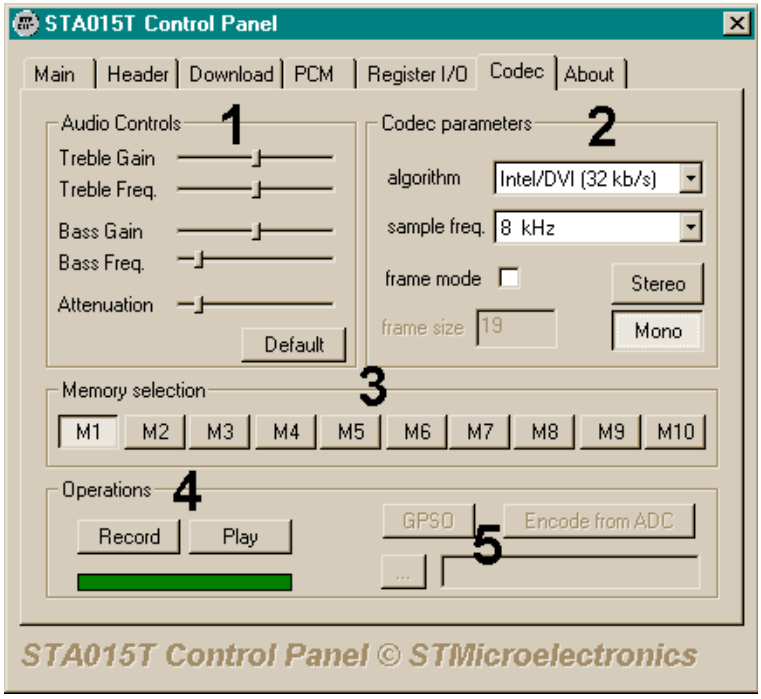
§ 2 : To find the LPT port address just use the 'Autofind' button. Tipically there should be no need to use it, since the parallel port address is automatically detected by the software every time you run it.

§ 3 : To perform an hardware reset of the device the 'Reset' button can be used. This operation, of course, will initialize all the device registers with their default values: therefore it should not be performed while decoding an MP3 file or while running the ADPCM engine.

§ 4 : In order to check the I2C connection and, basically, the correct setup of the evaluation kit a 'Test' button has been included in this page. It just reads some device registers and compare the result with the expected one.

### 8.6 Codec page

**Figure 17. Codec page**




§ 1 : The 'Audio Control' section offers a duplicate set of the audio controls available on the main page. The only difference is that the 'Tone Attenuation' register is automatically set to the maximum value of the desired tones enhancement. Of course this controls can be used when decoding an MP3 file as well: infact they just rely on the same set of I2C registers.

§ 2 : This section can be used to select the encoding/decoding ADPCM algorithm. User selected settings are automatically stored in the 'STA014/STA015T.ini' file located in the Windows system directory and are linked with each memory button: selecting a different memory will retrieve codec parameters used to store that memory. After the memory has been selected anyway, it's possible, to change the decoding parameters (increasing, for instance, the sampling frequency it's possible to make a fast playback of the message). This feature will not work if the message was recorded using the 'frame mode': in this case, infact, the decoding parameters comes from the header information inside the bitstream itself instead of coming from the I2C register settings. Some problem could arise when encoding/decoding the higher bitrates message (for instace, using the 32KHz Stereo mode with G723_40 algorithm), especially on slow machines: due to the PC parallel port speed limitation, infact, some communication problem could affect the encoded/decoded message. In any case the full set of option is still available in order to show all the device features and capabilities and to allow an exhaustive evaluation of the product. Again, in order to make the communication link between the PC and the STA014/STA015 board more robust it's stronly adviced to always enable the 'frame mode' operation.

§ 3 : This is the memory selection bank. As explained at the previous point every time a memory is selected the ADPCM codec parameters are retrieved from the 'STA014/STA015T.ini' file (if exist). Each memory can store up to 500 Kbytes.

§ 4 : Record/Play button are used to start/stop the encoding or decoding operation. The green filling-bar show the status of the memory or, when decoding, the current playback position.

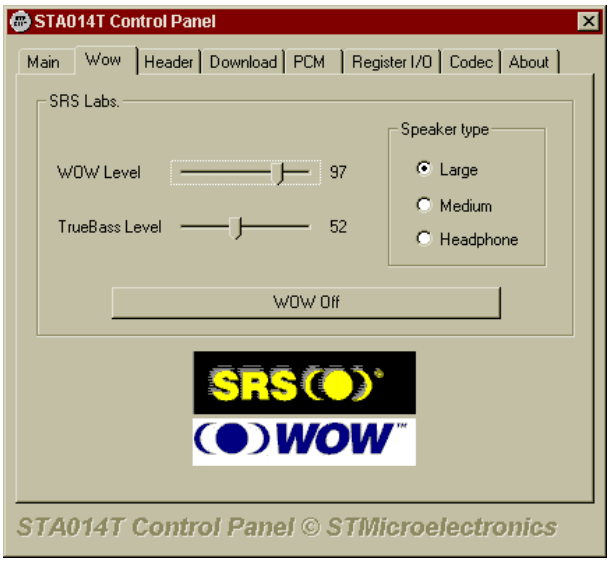
§ 5 : This controls are not enabled in the current release of the software.

Note: 1. 32KHz stereo encoding is only available in DVI ADPCM mode, as stated on the datasheet.

### §8.7 WOW (SRS Labs.) Page

The STA014T control panel also includes an additional page dedicated to the WOW post-processing feature controls. As shown in the Figure 18 the available controls allow to set the WOW and TrueBass enhancement level and the speaker type (large, medium or relative to headphones). Of course this controls will be effective only if the WOW post-processing feature has been turned on.

**Figure 18. WOW page**




## 9. PLL CONFIGURATION

### 9.1 PLL Overview

All the internal reference clocks required to operate the device and to output decoded audio samples are generated using the embedded digital PLL block. Depending on the external reference clock used (typically coming from en external crystal) some PLL registers must be initialized after device power-on: after that the DSP firmware is able to automatically decode any MPEG bitstream with any sample rate or bitrate without further configuration or initialization. The output I2S clocks (LRCKT and SCKT) will be automatically tuned according to the decoded MPEG bitstream: if the ADPCM mode is enabled the output frequency will depend on the choosed sample rate.

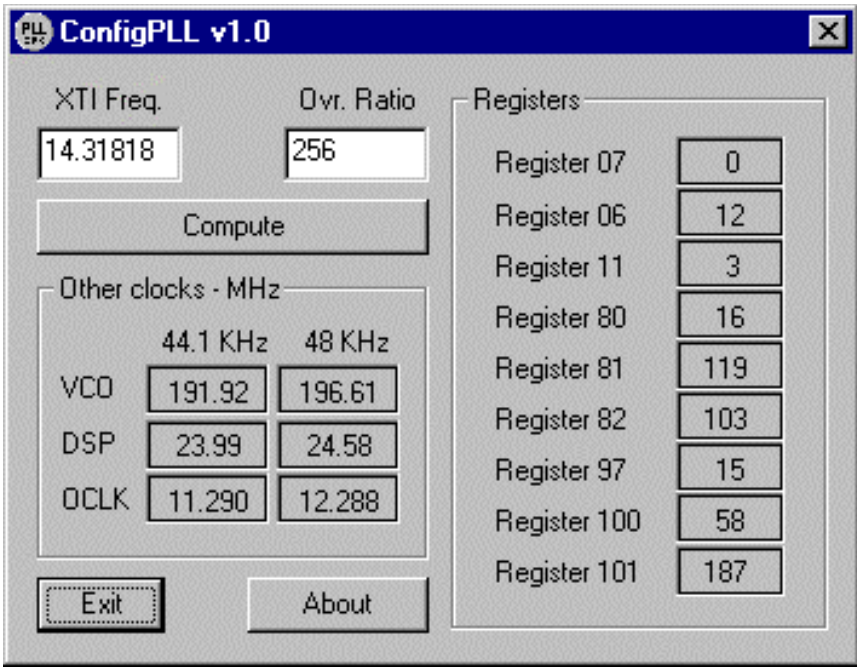
### 9.2 PLL Configuration Utility

In order to make the PLL configuration task as easy as possible a small software utility is available. As shown in Figure 19 no technical skills are required to operate the software. Only two input data are required to compute PLL register values:

- XTI Frequency: this is the external reference frequency (tipically coming from a crystal device). A wide range of values can be used with no loss in decoder performance (suggested values are between 8 MHz and 24 MHz)

- DAC Oversampling Ratio: this value depends on the choosed D/A converter. Typical values are 128, 256 or 384.

**Figure 19. Config PLL utility**




## 10 POWER CONSUMPTION HINTS

In order to reduce the device power consumption here are some general guidelines:

- If the device reference frequency comes from an external oscillator then the XTO pad can be disabled: in order to do this bit XTODIS of register PLLCTL must be set to 1. This bit is set to 0 on HW reset.

- If the device is not decoding the DSP can be set in idle mode setting PLAY and MUTE registers to 0. This configuration will also stop the PCM output interface clock signals saving a little bit more power.

- The power consumption can also be reduced using the reserved register 0x11. When the device is not decoding writing 31 in register 0x11 will enter an idle mode. For normal operation this register must be restored with the default value coming from the PLL configuration tables included in the datasheet or computed by the ConfigPLL utility.