MMCODECS

EQ ON TMS320C55xxTMS320C55xx

Error! Unknown document property name.

Software Released	Release version	Release Date
EQ	2.3	29th-Jan-04

Prepared by: Srividya M. S.

Approved by: Anita Shet

TABLE OF CONTENTS

1 EQUALIZER ALGORITHM 3

1.1 OVERVIEW 3
1.2 IMPLEMENTATION RESOURCE USAGE 3
1.3 CONFIGURING BAND EDGE FREQUENCIES 3

2 EQ HIGH LEVEL DESIGN 5

2.1 IEQ INTERFACE DESIGN 5
2.2 XDAIS INTERFACE APIS 5

3 EQ MODULE DOCUMENTATION 8

3.1 EQ HIGH LEVEL INTERFACE 8
3.2 EQ LOW LEVEL INTERFACE 11

4 EQ DATA STRUCTURE DOCUMENTATION 12

4.1 EQ_OBJ STRUCT REFERENCE 12
4.2 EQ_TI_PARAMS STRUCT REFERENCE 13
4.3 EQ_TI_STATUS STRUCT REFERENCE 14
4.4 EQ_TI_BAND STRUCT REFERENCE 15

5 EQ FILE DOCUMENTATION 17

5.1 EQ_TI_PRVT.H FILE REFERENCE 17

1 EQUALIZER ALGORITHM

1.1 Overview

An audio equalizer is an algorithm that allows the volume level of an audio signal to be adjusted separately in a different frequency ranges or bands. After sample rate conversion the digital samples are passed to the equalization module. This equalizer is a N band graphical equalizer that operates on one stream of samples at a time.

Samples are delivered to the equalizer in a Buffer in blocks of 16 samples or 64 samples. The samples are then passed through the N stages of equalizations and returned to the system in the same Buffer.

The EQ module has the following features:

- Supports N bands of Equalization
- Gains adjustable from –15db to +15 db in 1db steps
- Distortion free flat spectrum at 0 dB gain
- No unpleasant audible artifacts during gain changes
- 16-bit PCM input and output audio data
- Center frequencies and bandwidths are configurable off-line
- Implemented by 2^{nd} order IIR sections in cascade.

1.2 Implementation Resource Usage

1.2.1 Performance

Detailed Memory and MIPS requirement is available at SRC_datasheet.doc attached along with the release.

1.2.2 Testing

The Equalizer module was tested for all dB gains that are supported.

1.3 Configuring Band Edge Frequencies

The center frequencies and bandwidths of the equalizer bands can be configured off-line using the included Matlab script eq_ti_coeffs. Any number of sets of coefficients may be included in the system.

Each instance of the equalizer may be configured with a different set of band edges at initialization using EQ_TI_algInit (), or during operation using EQ_TI_setStatus (), by modifying the coeffs and deltaCoeffs fields of EQ_TI_Params and EQ_TI_Status.

The eq_ti_coeffs Matlab script can be used as follows:

File:
config\eq_ti_coeffs.m

Usage:

eq_ti_coeffs (BPCF, BPBW, BPGains)
Here, the parameters are:

BPCF - an Nx1 vector of band center frequencies, in Hz.
BPBW - an Nx1 vector of bandwidths, in octaves.
BPGains - an Nx1 vector of input gains, in dB (-15dB to 15dB).

Coefficients for N bands are produced, where N is the length of the BPCF parameter vector. Each band is implemented as a 2nd order IIR filter. The sample rate is fixed at 44100 Hz. Other sample rates can be accommodated by scaling the center frequencies in BPCF.

The output is written to the file eqcoeffs.s54, a TMS320C54x assembly source file. The file defines the symbols `EQ_TI_DefCoeffs` and `EQ_TI_DefDeltaCoeffs`. These may be referenced in application code as:

```
extern long EQ_TI_DefCoeffs[], EQ_TI_DefDeltaCoeffs[];
```

If multiple sets of coefficients are to be included in a system, each eqcoeffs.s54 file should be renamed to avoid filename conflicts, and edited to rename the symbols `EQ_TI_DefCoeffs` and `EQ_TI_DefDeltaCoeffs` to avoid linker namespace conflicts.

This EQ algorithm has the following features:

1. Data memory can be allocated at run time.
2. The library is in FAR mode
3. Fully validated on c5416 PG 2.2 silicon.

2 EQ HIGH LEVEL DESIGN

2.1 IEQ Interface Design

The given algorithm implements IEQ interface, which is derived from IALG interface. The module name is **EQ** and vendor name is **TI**.These API's are logically part of XDAIS interface framework and provided to simplify the use of given algorithm.

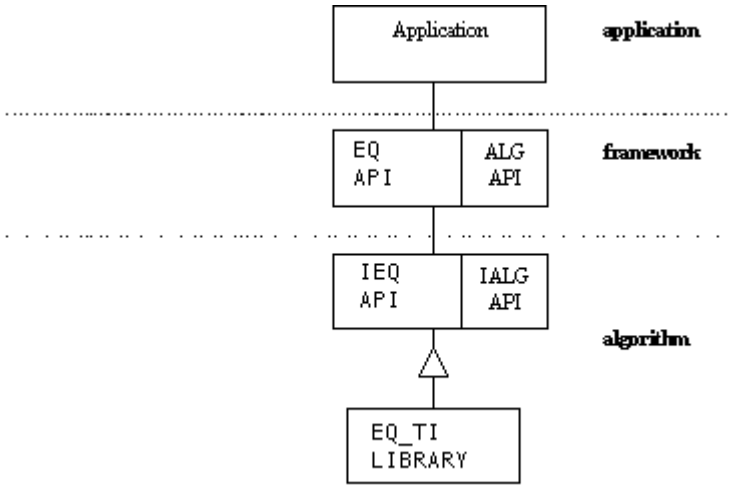

Figure 1: IEQ XDAIS Interface Design

The following is the list of the API that will be supported as part of XDAIS of the **EQ** interface design.

2.2 xDAIS Interface APIs

- Int EQ_TI_algNumAlloc (void);
- Int EQ_TI_algAlloc (const IALG_Params * Params,
  IALG_Fxns ** fxns,
  IALG_MemRec * memTab);
- Int EQ_TI_algInit(EQ_TI_Handle, const IALG_MemRec *,
  EQ_TI_Handle, const EQ_TI_Params *);
- Int EQ_TI_algFree (IALG_Handle Handle,
  IALG_MemRec * mem);

2.2.1 EQ_TI_algNumAlloc

Prototype:

Int EQ_TI_algNumAlloc(void);

Description:

This functions returns number of memory blocks to be assigned for the data memory.

2.2.2 EQ_TI_algAlloc

Prototype:

Int EQ_TI_algAlloc (const IALG_Params * Params, IALG_Fxns ** fxns, IALG_MemRec * memTab);

Description:

This function fills up the field of structure IALG_MemRec. The fields of this structure contain information about alignment, size and attribute of the different memory blocks.
The framework needs to call this function in order to find out the memory requirements of the EQ. The memory requirements will be filled in the IALG_MemRec structure array. Each IALG_MemRec structure will have the memory requirements for one block. The memTab array should have n elements if n blocks of memory have to be allocated. The base address for each of the blocks should be determined by the framework. So the base address field will not be filled in by this function.

```
typedef struct IALG_MemRec {
   Int              size;       /* size in MAU of allocation */
   Int              alignment;  /* alignment requirement (MAU) */
   IALG_MemSpace    space;      /* allocation space */
   IALG_MemAttrs    attrs;      /* memory attributes */
   Void             *base;      /* base address of allocated buf */
} IALG_MemRec;
```

The IALG_algNumAlloc function should be called before this function is called. This is necessary because the system needs to allocate memory for the MemRec structure array, so passing it as a parameter to this function.

EQ module requires two memory blocks, for EQ_TI_Obj structure and for EQ_TI_Band structure.
Memory block description for EQ_TI_Obj structure: --
The size field of memtab is assigned the size of EQ_TI_Obj,
Allignment is 0

IALG_DARAM0 is assigned to space attribute.
And the memory attribute is specified for persistence.

Memory block description for EQ_TI_Band structure :--

The size field of memtab is assigned the size of EQ_TI_Band multiply by number of bands,
Allignment is 0

IALG_DARAM0 is assigned to space attribute.
And the memory attribute is specified for persistence.

Memory block description for tempbuf: -
The size field is assigned 12 bytes.
Allignment is 0.
IALG_SARAM0 is assigned to space attribute for tempbuf.

2.2.3 EQ_TI_algInit

Prototype:

Int EQ_TI_algInit (EQ_TI_Handle, const IALG_MemRec *,

EQ_TI_Handle, const EQ_TI_Params *);

Description:

This function initializes all EQ parameters.
"handle" is the pointer to the current instance. Using the memRec structure array, module will set up its internal variables, to the memory blocks allocated for it.

2.2.4 EQ_TI_algFree

Prototype:

Int EQ_TI_algFree (IALG_Handle Handle, IALG_MemRec * mem)

Description:

After the end of the processing, this function has to be called to free the memory allocated for an instance of EQ.

2.2.5 EQ_create

Prototype:
IEQ_Handle EQ_create (const IEQ_Fxns *fxns, const EQ_Params *prms);

Parameters:

- fxns is a pointer to the v-table for the EQ implementation. This field is initialized by EQ_Obj ().
- prms is a pointer to the Creation Parameters for this instance.

Description:
This dynamically creats an instance of the algorithm.

2.2.6 EQ_delete

Prototype:

Void EQ_delete (IEQ_Handle handle)

Description:

This deletes the given dynamically created objects referenced by handle. If Handle is NULL, then EQ_delete simply returns.

3 EQ MODULE DOCUMENTATION

3.1 EQ High Level Interface

Data Structure

- EQ_TI_Obj

Typedefs

- Typedef struct EQ_TI_Obj *EQ_TI_Handle;

Functions

- Void EQ_TI_apply (EQ_TI_Handle handle, Int *in, Int *out)
- Void EQ_TI_getStatus (EQ_TI_Handle, EQ_TI_Status *status)
- Void EQ_TI_setStatus (EQ_TI_Handle, EQ_TI_Status *status)
- Void EQ_TI_getGains (EQ_TI_Handle handle, Int *gains)
- Void EQ_TI_setGains (EQ_TI_Handle handle, Int const *gains)
- Int EQ_TI_getGain (EQ_TI_Handle handle, Int band)
- Void EQ_TI_setGain (EQ_TI_Handle handle, Int band, Int gain)

Detailed Description

The EQ_TI_Obj structure contains all information about an instance of the equalizer. In the IEQ_Fxns field, it contains pointers to all the available EQ functions. It also contains all appropriate EQ variables such as buffer size and individual band information.
One instance of the EQ structure is required for each channel of digital audio that requires filtering. Thus in a system with stereo mode there are two instances of the equalizer modules and thus two pointers to EQ Object Structures, eqleft and eqright which are dynamically created during runtime by calling EQ_create function.

Following is the EQ_TI_Obj structure: -
.

```
typedef struct EQ_TI_Obj {
/*IALG */
const IEQ_Fxns fxns; /Function table */
/*EQ */
Int bufferSize; /*Length of input and output data blocks */
```

Typedef Definition

typedef struct IEQ_Obj *IEQ_Handle, EQ_Handle

This pointer type is used to reference all EQ instance objects.

Function Description

Void EQ_TI_apply (IEQ_Handle handle, Int *in, Int *out)

This is the function used to perform equalization operation. This function calls filter bank function and performs equalization operation. If required, it updates filter coefficients depending on dB gain, and uses them in filter bank.

Parameters

handle Pointer to an active object instance.
in Pointer to input buffer .
out Pointer to output buffer. For in-place operation it can be same as the input buffer .

Void EQ_TI_getStatus (IEQ_Handle, IEQ_Status *status)

This function will return run-time parameters of the algorithm.

Parameters

handle Pointer to an active object instance.
status Pointer to the IEQ_Status

Void EQ_TI_setStatus (IEQ_Handle, const IEQ_Status *status)

This function will set the run-time parameters of the algorithm.

Parameters

handle Pointer to an active object instance.
status Pointer to the IEQ_Status

Int EQ_TI_getGains (IEQ_Handle handle)

Fills an array with the current equalizer gain values of all frequency bands.

Parameters

handle Pointer to an active object instance.
gains Pointer to particular band gain

Void EQ_TI_setGains (IEQ_Handle handle, Int gain)

Set the equalizer gain value for n band.

Parameters

handle Pointer to an active object instance.

gains Pointer to particular band gain

Int EQ_TI_getGain (IEQ_Handle handle, Int band);

Return the equalizer gain value of each band.

Parameters

handle Pointer to an active object instance.
band It is the band number.

Void EQ_TI_setGain (IEQ_Handle handle, Int band, Int gain);

Set the equalizer gain value for each band

Parameters

handle Pointer to an active object instance.
band It is the band number.
gain It is the particular band gain.

3.2 EQ Low Level Interface

Data Structure

- Struct IEQ_Fxns

Members

- Void (*apply) (IEQ_Handle handle, Int *in, Int *out);
- Void (*getStatus) (IEQ_Handle handle, IEQ_Status *status);
- Void (*setStatus) (IEQ_Handle handle, const IEQ_Status *status);
- Void (*getGains) (IEQ_Handle handle, Int *gains);
- Void (*setGains) (IEQ_Handle handle, Int const *gains);
- Int (*getGain) (IEQ_Handle handle, Int band);
- Void (*setGain) (IEQ_Handle handle, Int band, Int gain)

Detailed Description

IEQ is the low-level algorithm interface to the equalizer. IEQ defines types that implement the EQ High Level Interface. Its main purpose is to support the function pointers that would allow the equalizer module to be accessed at run-time.
IEQ is a superset of the eXpressDSP interface IALG. This section describes only the features specific to IEQ.

Typedefs

typedef EQ_Handle, IEQ_Handle

Another name for EQ_Handle.

typedef EQ_Params, IEQ_Params

Another name for EQ_Params.

typedef EQ_Status, IEQ_Status

Another name for EQ_Status.

typedef EQ_Obj, IEQ_Obj

Another name for EQ_Obj.

4 EQ DATA STRUCTURE DOCUMENTATION

4.1 EQ_Obj Struct Reference

include <iEQ.h>

Data Fields

- const IEQ_Fxns *fxns; /* Function table */
  /* EQ */
- Int bufferSize; /* Length of input and output data blocks */
- Int numBands; /* Number of equalizer bands */
- EQ_TI_Band *pBands;
- Uns EQChangeFlag;
- LgInt *tempbuf;

Detailed Description

This structure formally declares an Equalizer algorithm instance. Almost every function of the API accepts a pointer to an object of this type, so identify an instance to operate on. This is an opaque structure, used only to define the pointer type.

Fields

const IEQ_Fxns *fxns

A pointer to the v-table for the EQ implementation. This field is initialized by EQ_Obj ().

Int bufferSize

This will give input and output buffer size in number of samples. The size must be multiple of 64 samples.

Int numBands

Number of equalizer bands. This may be any value greater than zero, and no more than fourteen.

EQ_TI_Band *pBands

This is a pointer to EQ_TI_Band structure.

Uns EQChangeFlag

This defines the change flags for each band.

LgInt *tempbuf

This is a pointer of type long, which is used as temporary buffer.

4.2 EQ_TI_Params Struct Reference

#include <ieq.h>

Data Fields

- Int size;
- Int bufferSize
- Int numBands
- Int const *dBGains
- EQ_TI_Coeff const *coeffs
- EQ_TI_Coeff const *deltaCoeffs

Detailed Description

This structure defines the creation parameters for an instance of the equalizer algorithm. The application fills the structure to describe the desired properties of the instance before calling the creation functions EQ_TI_algAlloc () and EQ_TI_algInit (). The configurable creation parameters include I/O Buffer size, number of frequency bands,
And band center frequencies and bandwidths

Fields

Int Size

This will give size of this structure. This application will initialize this value and then it will call the functions EQ_TI_algAlloc () and EQ_TI_algInit (). It is used internally to support compatibility across versions of the API

Int bufferSize

This will give input and output buffer size in number of samples. The size must be multiple of 64 samples.

Int numBands

Number of equalizer bands. This may be any value greater than zero, and no more than fourteen.

Int const *dBGains

Initial gain settings. This is a pointer to an array of numBands values, giving the gain or attenuation of each band in dB. Each band takes a value from –15(15 dB attenuation) to +15(15 dB boost) in steps of 1.

EQ_TI_Coeff const *coeffs

This is pointer to filter coefficients. This pointer is initialized in EQ_TI_algInit ();

EQ_TI_Coeff const *deltaCoeffs

This is pointer to filter delta-coefficients. This pointer is initialized in EQ_TI_algInit();

4.3 EQ_TI_Status Struct Reference

#include <ieq.h>

Data Fields

- Int size
- Int bufferSize
- Int numBands
- EQ_TI_Coeff *coeffs
- EQ_TI_Coeff *deltaCoeffs

Detailed Description

This structure gives status of the parameters. This structure defines the parameters that can be read at run-time, using EQ_TI_getStatus () and some parameters may also be changed after initialization using EQ_TI_setStatus ().

Fields

Int size

This will give size of this structure. This application will initialize this value and then it will call the functions EQ_TI_getStatus () and EQ_TI_setStatus (). It is used internally to support compatibility across versions of the API.

Int bufferSize

This will give input and output buffer size in number of samples. The size must be multiple of 64 samples. This value is set at initialization, and normally fixed. However if desired it may be changed at any time.

Int numBands

This gives number of equalizer bands.

EQ_TI_Coeff *coeffs

This is the pointer to the user defined coefficient table. If user want to use the user defined coefficients then this pointer has to be set as pointer to user defined coefficient table.

EQ_TI_Coeff *deltaCoeffs

This is the pointer to user defined coefficient table. If user want to use the user defined coefficients then this pointer has to be set as pointer to user defined coefficient table.

4.4 EQ_TI_Band Struct Reference

#include <ieq.h>

Data Fields

- Int XState[2];
- Int dBGain
- Int padd
- Int const * pGain;
- EQ_TI_Coeff const *pCoeff;
- EQ_TI_Coeff const *pDeltaCoeff;
- LgInt WState[2];
- Int Gain;
- LgInt Beta;
- LgInt BGamma;

Description

This structure is used for calculation of gain of particular band. This also contains pointers to coefficient tables. By using Beta, Gain, Bgamma, particular band gain is calculated. It also contains some temporary variables, which are used for calculation of particular band gain.

Fields

Int XState [2]

This is a temporary variable used for calculation of particular band gain.

Int dBGain

This contains every band gain.

Int const *pGain

This is the pointer to each band gain.

EQ_TI_Coeff const *pCoeff

This is the pointer to coefficient table.

EQ_TI_Coeff const *pDeltaCoeff

This is the pointer coefficient delta table.

LgInt WState [2]

This is a temporary variable used for calculation of particular band gain.

Int Gain

This is used for gain calculation of particular band.

LgInt Beta

This is used for gain calculation of particular band.

LgInt Bgamma

This is used for gain calculation of particular band.

5 EQ FILE DOCUMENTATION

5.1 eq_ti_prvt.h File Reference

#include “ieq.h”

Data structures

- EQ_TI_Coeff
- Struct EQ_TI_Params
- Struct EQ_TI_Band
- Struct EQ_TI_Obj

Typedefs

- typedef EQ_TI_Obj * EQ_TI_Handle
- typedef IEQ_Status EQ_TI_Status
- extern const IEQ_Fxns EQ_TI_IEQ

Functions

- Int EQ_TI_algNumAlloc(void);
- Int EQ_TI_algAlloc (const IALG_Params * Params, IALG_Fxns ** fxns, IALG_MemRec * memTab);
- Int EQ_TI_algFree (IALG_Handle Handle, IALG_MemRec * mem);
- Int EQ_TI_algInit (IALG_Handle, const IALG_MemRec *, IALG_Handle, const IALG_Params *)
- Void EQ_TI_apply (IEQ_Handle handle, Int *in, Int *out)
- Void EQ_TI_getStatus (IEQ_Handle, IEQ_Status *status)
- Void EQ_TI_setStatus (IEQ_Handle, IEQ_Status *status)
- Void EQ_TI_getGains (IEQ_Handle handle, Int *gains)
- Void EQ_TI_setGains (IEQ_Handle handle, Int const *gains)
- Int EQ_TI_getGain (IEQ_Handle handle, Int band)
- Void EQ_TI_setGain (IEQ_Handle handle, Int band, Int gain)

Description

This header file declares EQ High Level Interface. This header file contains all typedefs and defines all constants and function declared by all implementation of the abstract interface for EQ module

Appendix A – Document change history

Ver.No.	Editor/Author	Date dd-mmm-yy	Reviewer	Changes made
V1.1	Suresh A	20th Dec, 2002		EQ module supports 5 bands
V1.2	Suresh A	14th Aug 2003		Updated according to new format
V1.3	Suresh A	19th Sept 2003		Updated after 5 band to N band conversion
V1.4	Srividya M. S.	29th Jan 2004		Updated the template

---------- **End Of Document** ----------