Implementation of G.726 ADPCM on TMS320C62xx DSP

Literature Number: BPRA066
Texas Instruments Europe
October 1997

IMPORTANT NOTICE

Texas Instruments (TI) reserves the right to make changes to its products or to discontinue any semiconductor product or service without notice, and advises its customers to obtain the latest version of relevant information to verify, before placing orders, that the information being relied on is current.

TI warrants performance of its semiconductor products and related software to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are utilized to the extent TI deems necessary to support this warranty. Specific testing of all parameters of each device is not necessarily performed, except those mandated by government requirements.

Certain applications using semiconductor products may involve potential risks of death, personal injury, or severe property or environmental damage ("Critical Applications").

TI SEMICONDUCTOR PRODUCTS ARE NOT DESIGNED, INTENDED, AUTHORIZED, OR WARRANTED TO BE SUITABLE FOR USE IN LIFE-SUPPORT APPLICATIONS, DEVICES OR SYSTEMS OR OTHER CRITICAL APPLICATIONS.

Inclusion of TI products in such applications is understood to be fully at the risk of the customer. Use of TI products in such applications requires the written approval of an appropriate TI officer. Questions concerning potential risk applications should be directed to TI through a local SC sales office.

In order to minimize risks associated with the customer's applications, adequate design and operating safeguards should be provided by the customer to minimize inherent or procedural hazards.

TI assumes no liability for applications assistance, customer product design, software performance, or infringement of patents or services described herein. Nor does TI warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right of TI covering or relating to any combination, machine, or process in which such semiconductor products or services might be or are used.

Contents

List of Figures

List of Tables

Implementation of G.726 ADPCM on TMS320C62xx DSP

1. Overview

ADPCM (Adaptive Differential Pulse Code Modulation) is a widely used voice coding standard for communications links within the worldwide telecommunications network. Within the ITU (International Telecommunication Union) several standards are defined, of which the two most commonly used are G.721 and G.726. G.726 is a superset of G.721, and is used in the rest of this document. The G.726 standard is used for Digital Cordless Telephony, Radio/Wireless Local Loop, and Pair-Gain.

The Digital Cordless Telephone standards for CT2, DECT and PHS all specify that G.726 is the algorithm to be used for 32kbps voice channels.

In RLL (Radio Local Loop) or WLL (Wireless Local Loop) G.726 ADPCM is also widely used in both proprietary systems and in cordless standard based systems.

For Pair Gain, G.726 is used for satellite and other international links and compression on private voice circuits between sites of large organizations.

Texas Instruments TMS320C62xx processors are well suited to performing voice compression for multi-channel G.726 applications, and this application note puts emphasis on base-station design for all of the above systems.

This application report is designed to be used with the Texas Instruments G.726 TMS320C62xx software package and the ITU recommendation G.726.

2. What is G.726 ADPCM

ADPCM is a voice compression algorithm which works in the time domain by predicting the next time sample based on the spectrum and amplitude of the previous data. A block diagram showing the principal of an ADPCM voice coder is shown below. The adaptive predictor is used to predict the spectral quantities of the next sample, the difference between this prediction and the real signal is then quantized using an adaptive filter which base its quantization levels on the past accuracy of recent previous samples. This value is then used to update both the predictor and the quantizer for the next sample. The decoder works in a similar manner, except that there is an additional stage at the end called synchronous

coding adjustment. This mimics the effects of tandeming several voice vocoders in sequence and makes an adjustment to the output to reduce the risk of an ADPCM voice coder further down the line making a different decision about the ADPCM value used for compression.

G.726 is a standard ADPCM algorithm specified by the International Telecommunication Union (ITU) for reducing the 64 kbps A-Law or μ-Law logarithmic data of a normal telephone line to any of 16, 24, 32 or 40kbps. The full mathematical specification is ITU copyright and can be found in the G.726 specification published by the ITU. G.721 is an older ITU standard for ADPCM that only supported 32kbps. G.721 (1988) is fully bit compatible G.726

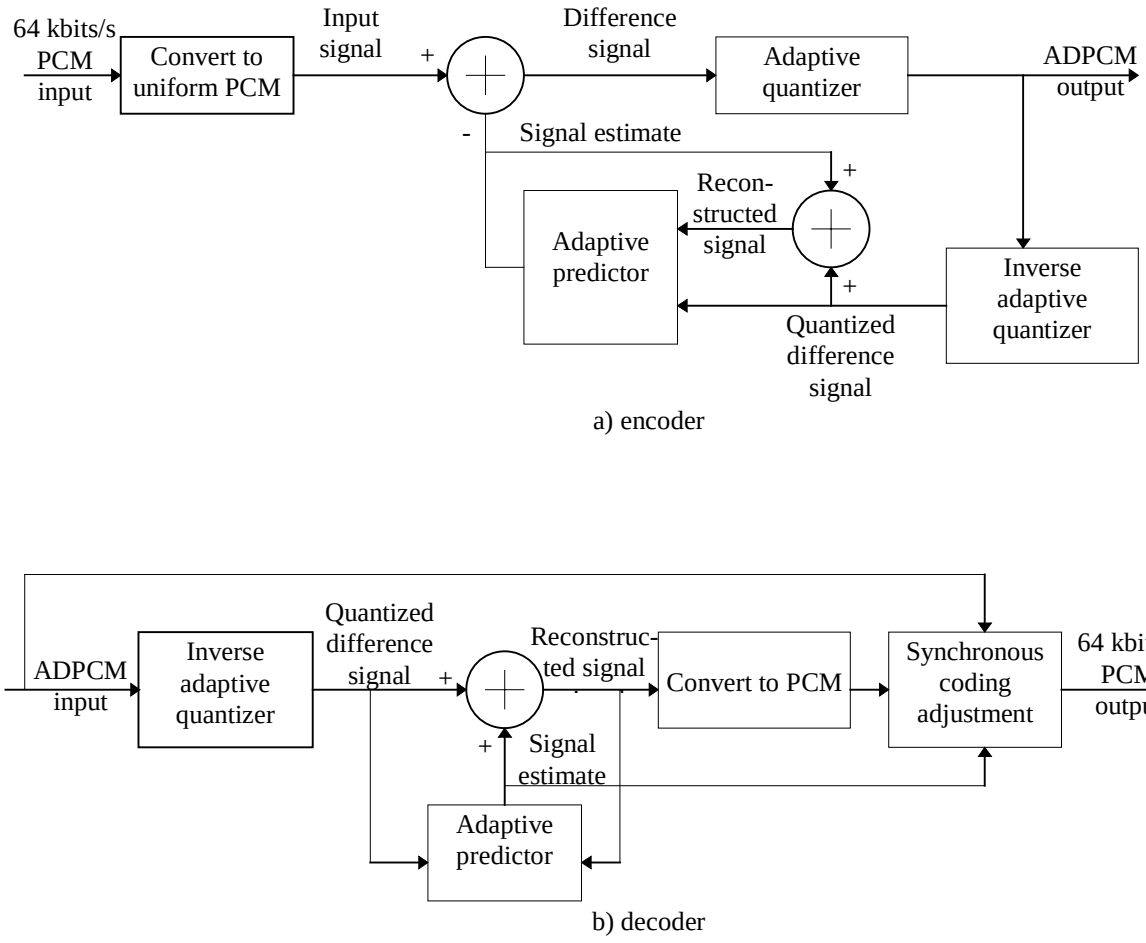

Figure 1: ADPCM encoder and decoder

3. Description

The Texas Instruments G.726 code for the TMS320C62xx is a fully implementation of the ITU standard that has been tested against all the test vectors supplied by the ITU for all modes of operation. The code is designed for implementing multi-channel voice vocoders and each subroutine performs 2 channels-worth of voice coding.

3.1 RAM requirements

152 bytes of RAM are required per channel.

3.2 ROM requirements

The program and constant memory requirements vary between 5260 and 5836 for a single rate (16, 24, 32 or 40kbps channel), or 22Kbytes for a full C multi-rate implementation.

Table 1: ROM requirements for different rates

Section	Memory (Module)	Memory (Include All)
All	928	
16 kbps	4544	5474
24 kbps	4640	5568
32 kbps	4800	5728
40 kbps	5152	6080
C test vector	1472	
Multi-rate + C		22368

3.3 MIPs requirements

This code was developed using version 1.0 of the simulator, from which the following benchmarks were obtained.

Table 2: Cycle and MIPs counts for 2 Channels

Rate	A-Law cycles	μ-Law cycles	Linear[1] cycles	A-Law MIPs	μ-Law MIPs	Linear MIPs
16	562	560	524	.004	.004	.004
24	566	564	526	.005	.005	.004
32	**574**	**572**	**530**	**.005**	**.005**	**.004**
40	595	593	542	.005	.005	.004

4. Software Routines

Whilst all of the G.726 code is written in 'C6x assembler, they are all written within the guidelines for 'C6x C-callable routines, and can be used with Texas Instruments C compiler. For use with other compilers or assembler code, it may be necessary to check which registers are 'parent' protected and which are 'child' protected. Texas Instruments C compiler uses parent protection for registers a0..9 and b0..9 with child protection for a10..15 and b10..15. This means that registers a0..9 and b0..9 are not protected within the assembler routines. In parent protection, registers are saved by the calling routine before the subroutine call; in child protection, registers are saved by the child routine after the subroutine call.

4.1 C-code and C-callable

All of the main G.726 subroutines are C-callable. Where necessary, these routines disable interrupts to protect multiple assignment code. In all cases, the maximum time for which interrupts are disabled is below 275 cycles. The test vector example program and the C functions are explained below.

4.1.1 read_io.c

This file provides an example of how to call the various G.726 assembler routines from a C environment. This program is the one that was used to apply the G.726 test vectors. All benchmark figures above were obtained by compiling this program with the '-k -as -g -o2' options with timing measured from a breakpoint at 'expand…' and 'if (LineNo>…'. Lines of the form: 'label =*(short *)address' are defined addresses in the simulator where test vector or input control can be found. Note the use of local variables and the use of sub-routine pointers (code, decode, expand, compress); these optimizations give a significant improvement in the

[1] Please Note Linear mode is not the same as using the internal companding of the 'C62xx, as there is no synchronous coding adjustment. Linear mode is a simplification to G.726 that is allowed to be used only when output is directly to a linear D/A. This allowance is by means of system level waivers.

MIPs achieved as they allow the assembler to optimize them into registers and avoid the need for a ‘switch’ in the main loop. Note also that the output values are read not written. This is a function of test vector verification. The variable read into mode contains information on the test sequence to be tested. In particular the upper 2 bytes contain the length of the test sequence, the 3rd byte the rate and the 4th byte the coding law.

4.1.2 g726.h

This file provides the external definitions of the C-callable assembler functions. All of these functions except perform 2 channels of functionality. The first two parameters are the input parameters and the second two parameters are pointers to where the output parameters should be written. The parameter format is (short in1,short in2,short *out1,short *out2). The G.726 voice coding routines have a further two parameters which give the channel number of the data frame which contains local data for that channel.

To minimize memory stalls in the 'C62xx Internal Data RAM, all calls to these routines should use an even and an odd channel number. If called with 2 even or 2 odd channel numbers, there will be large number of memory stalls, considerably reducing the performance of the algorithm.

It is possible to switch data rates between A- and μ-law in mid-channel so long as both ends of the communication channel do so at the same time. For instance, in a pair gain application both ends can dynamically alter the bit rate according to traffic demand. The predictor and quantizer will cope.

4.1.2.1 G726_all_reset()

This resets all the channels defined at assembler time in the G.726 control routine, and is defined in the common control routine. It works for all modes (16, 24, 32 or 40kbps, A or μ-law). It is defined in G726_all.asm.

4.1.2.2 G726_reset(short channel, boolean decode, boolean encode)

This resets one voice channel. The two boolean parameters specify whether the encode, decode or both direction are to be reset. It works for all modes (16, 24, 32 or 40kbps, A or μ-law). It is defined in G726_all.asm.

4.1.2.3 lin2alaw(short in1,short in2,short *out1,short *out2)

This routine converts two 15 bit two’s complement linear values to A-law values. It is defined in G726_all.asm. Conversion is as per G.711, except the input is multiplied by 4.

4.1.2.4 alaw2lin(short in1,short in2,short *out1,short *out2)

This routine converts two A-law values to 14 bit two's complement linear values. It is defined in G726_all.asm. Conversion is as per G.711, except the output is multiplied by 2.

4.1.2.5 lin2ulaw(short in1,short in2,short *out1,short *out2)

This routine converts two 15 bit two's complement linear values to μ-law values. It is defined in G726_all.asm. Conversion is as per G.711, except the input is multiplied by 2.

4.1.2.6 ulaw2lin(short in1,short in2,short *out1,short *out2)

This routine converts two μ-law values to 14 bit two's complement linear values. It is defined in G726_all.asm. Conversion is as per G.711.

4.1.2.7 G726_cod16(short in1,short in2,short *out1,short *out2,short ch1,short ch2)

This routine codes two 14 bit linear values to two 2 bit 16kbps ADPCM values. This routine can be used with either A or μ-law. It also requires two channel numbers, which should be odd and even to minimize memory stalls. Interrupts are disabled during this routine.

4.1.2.8 G726_dec16a(short in1,short in2,short *out1,short *out2,short ch1,short ch2)

This routine decodes two 2 bit 16kbps ADPCM values into two 15 bit linear values using A-law synchronous coding adjustment. It should not be used with μ-law. It also requires two channel numbers, which should be odd and even to minimize memory stalls. Interrupts are disabled during this routine.

4.1.2.9 G726_dec16u(short in1,short in2,short *out1,short *out2,short ch1,short ch2)

This routine decodes two 2 bit 16kbps ADPCM values into two 15 bit linear values using μ-law synchronous coding adjustment. It should not be used with A-law. It also requires two channel numbers, which should be odd and even to minimize memory stalls. Interrupts are disabled during this routine.

4.1.2.10 G726_dec16(short in1,short in2,short *out1,short *out2,short ch1,short ch2)

This routine decodes two 2 bit 16kbps ADPCM values into two 14 bit linear values without using synchronous coding adjustment. This mode is not defined in the ITU recommendations but the removal of synchronous coding adjustment from G.726 is allowed as an option in some standards that recommend G.726. It is also useful when further digital processing that may change the PCM value is to be used: synchronous coding adjustment is only a benefit in this case. Interrupts are disabled during this routine.

4.1.2.11 G726_cod24(short in1,short in2,short *out1,short *out2,short ch1,short ch2)

This routine codes two 14 bit linear values to two 3 bit 24kbps ADPCM values. This routine can be used with either A- or μ-law. It also requires two channel numbers, which should be odd and even to minimize memory stalls. Interrupts are disabled during this routine.

4.1.2.12 G726_dec24a(short in1,short in2,short *out1,short *out2,short ch1,short ch2)

This routine decodes two 3 bit 24kbps ADPCM values into two 15 bit linear values using A-law synchronous coding adjustment. It should not be used with μ-law. It also requires two channel numbers, which should be odd and even to minimize memory stalls. Interrupts are disabled during this routine.

4.1.2.13 G726_dec24u(short in1,short in2,short *out1,short *out2,short ch1,short ch2)

This routine decodes two 3 bit 24kbps ADPCM values into two 15 bit linear values using μ-law synchronous coding adjustment. It should not be used with A-law. It also requires two channel numbers, which should be odd and even to minimize memory stalls. Interrupts are disabled during this routine.

4.1.2.14 G726_dec24(short in1,short in2,short *out1,short *out2,short ch1,short ch2)

This routine decodes two 3 bit 24kbps ADPCM values into two 14 bit linear values without using synchronous coding adjustment. This mode is not defined in the ITU recommendations but the removal of synchronous coding adjustment from G.726 is allowed as an option in some standards that recommend G.726. It is also useful when further digital processing that may change the PCM value is to be used: synchronous coding adjustment is only a benefit in this case. Interrupts are disabled during this routine.

4.1.2.15 G726_cod32(short in1,short in2,short *out1,short *out2,short ch1,short ch2)

This routine codes two 14 bit linear values to two 4 bit 32kbps ADPCM values. This routine can be used with either A- or μ-law. It also requires two channel numbers, which should be odd and even to minimize memory stalls. Interrupts are disabled during this routine.

4.1.2.16 G726_dec32a(short in1,short in2,short *out1,short *out2,short ch1,short ch2)

This routine decodes two 4 bit 32kbps ADPCM values into two 15 bit linear values using A-law synchronous coding adjustment. It should not be used with μ-law. It also requires two channel numbers, which should be odd and even to minimize memory stalls. Interrupts are disabled during this routine.

4.1.2.17 G726_dec32u(short in1,short in2,short *out1,short *out2,short ch1,short ch2)

This routine decodes two 4 bit 32kbps ADPCM values into two 15 bit linear values using μ-law synchronous coding adjustment. It should not be used with A-law. It also requires two channel numbers, which should be odd and even to minimize memory stalls. Interrupts are disabled during this routine.

4.1.2.18 G726_dec32(short in1,short in2,short *out1,short *out2,short ch1,short ch2)

This routine decodes two 4 bit 32kbps ADPCM values into two 14 bit linear values without using synchronous coding adjustment. This mode is not defined in the ITU recommendations but the removal of synchronous coding adjustment from G.726 is allowed as an option in some standards that recommend G.726. It is also useful when further digital processing that may change the PCM value is to be used: synchronous coding adjustment is only a benefit in this case. Interrupts are disabled during this routine.

4.1.2.19 G726_cod40(short in1,short in2,short *out1,short *out2,short ch1,short ch2)

This routine codes two 14 bit linear values to two 5 bit 40kbps ADPCM values. This routine can be used with either A- or μ-law. It also requires two channel numbers, which should be odd and even to minimize memory stalls. Interrupts are disabled during this routine.

4.1.2.20 G726_dec40a(short in1,short in2,short *out1,short *out2,short ch1,short ch2)

This routine decodes two 5 bit 40kbps ADPCM values into two 15 bit linear values using A-law synchronous coding adjustment. It should not be used with μ-law. It also requires two channel numbers, which should be odd and even to minimize memory stalls. Interrupts are disabled during this routine.

4.1.2.21 G726_dec40u(short in1,short in2,short *out1,short *out2,short ch1,short ch2)

This routine decodes two 5 bit 40kbps ADPCM values into two 15 bit linear values using μ-law synchronous coding adjustment. It should not be used with A-law. It also requires two channel numbers, which should be odd and even to minimize memory stalls. Interrupts are disabled during this routine.

4.1.2.22 G726_dec40(short in1,short in2,short *out1,short *out2,short ch1,short ch2)

This routine decodes two 5 bit 40kbps ADPCM values into two 14 bit linear values without using synchronous coding adjustment. This mode is not defined in the ITU recommendations but the removal of synchronous coding adjustment from G.726 is allowed as an option in some standards that recommend G.726. It is also useful when further digital processing that may change the PCM value is to be used: synchronous coding

adjustment is only a benefit in this case. Interrupts are disabled during this routine.

4.2 G.726 assembler code

All of the G.726 core subroutines are written in highly optimized assembly code to give very efficient performance in terms of MIPs. Several of the subroutines contain multiple assignment code and disable interrupts. These routines also enable interrupts at the end. In all cases the maximum time for which interrupts are disabled is below 275 cycles.

4.2.1 G.726 control routines

These routines generate a voice coder and decoder for each of the G.726 data rates. The subroutines in G726_all.asm are common to all of the data rates and must be included with any rate. Each subroutine has one of 4 register usage types:

Type 1

Entry:	b3	return address
Exit:	a0..6/b0..6	undefined
	a7..15/b7..15	unchanged

Type 2

Entry:	b3	return address
	a4/b4	Two PCM input format values to code
	a6/b6	addresses to put Two new PCM formats
Exit:	a0..5/b0..5	undefined
	a6..15/b6..15	unchanged

Type 3

Entry:	b3	return address
	a4/b4	Two linear PCM/ADPCM input values to code
	a6/b6	addresses to put Two coded ADPCM/PCM values
	a8/b8	channel numbers of local data structure
Exit:	a0..9/b0..9	undefined
	a10.15/b10..15	unchanged

Type 4

Entry:	b3	return address
	a4	channel number to reset
	b4/a6	boolean to reset decode/encode sections
Exit:	a0..6/b0..6	undefined
	a7..15/b7..15	unchanged

Calls to type 1 functions affect all channels. Calls to type 2 functions are full register functions that do not affect any channel RAM. Calls to type 3 and 4 functions are channel specific.

4.2.1.1 g726_16.asm

This module defines the functions which need to be assembled together to produce the 16kbps ADPCM voice coder. In particular, it sets adpcm=16 so that the copied routines generate 16kbps ADPCM code, and defines the subroutines G726_cod16 (type 3), G726_dec16 (type 3), G726_dec16a (type 3), G726dec16u (type 3), for the coder, linear decoder, A-law decoder and μ-law decoder respectively.

4.2.1.2 g726_24.asm

This module defines the functions which need to be assembled together to produce the 24kbps ADPCM voice coder. In particular, it sets adpcm=24 so that the copied routines generate 24kbps ADPCM code, and defines the subroutines G726_cod24 (type 3), G726_dec24 (type 3), G726_dec24a (type 3), G726dec24u (type 3), for the coder, linear decoder, A-law decoder and μ-law decoder respectively.

4.2.1.3 g726_32.asm

This module defines the functions which need to be assembled together to produce the 32kbps ADPCM voice coder. In particular, it sets adpcm=32 so that the copied routines generate 32kbps ADPCM code, and defines the subroutines G726_cod32 (type 3), G726_dec32 (type 3), G726_dec32a (type 3), G726dec32u (type 3), for the coder, linear decoder, A-law decoder and μ-law decoder respectively.

4.2.1.4 g726_40.asm

This module defines the functions which need to be assembled together to produce the 40kbps ADPCM voice coder. In particular, it sets adpcm=40 so that the copied routines generate 40kbps ADPCM code, and defines the subroutines G726_cod40 (type 3), G726_dec40 (type 3), G726_dec40a (type 3), G726dec40u (type 3), for the coder, linear decoder, A-law decoder and μ-law decoder respectively.

4.2.1.5 g726_all.asm

This modules defines the functions which are common to all G726 voice coder rates and sets the amount of data space to be reserved for channel RAM via the “channels .equ x” line. The subroutines G726_all_reset (type 1), G726_reset (type 4), lin2alaw (type 2), alaw2lin (type 2), lin2ulaw (type 2) and lin2ulaw (type 2) are defined in this module.

4.2.2 G.726 implementation routines

These modules contain the assembly code that performs the mathematical functions that form the heart of the G.726 algorithm. Due to the highly parallel and pipe-lined nature of the TMS320C62xx several G.726 ‘functions’ are often being executed in parallel. Where this is happening each line of code is commented with a two part comment, the first of which is a reference to the function in the G.726 algorithm specification and the second a description of the sub-function or status within the function. A block diagram of the G.726 encoder and decoder is shown Figure 2

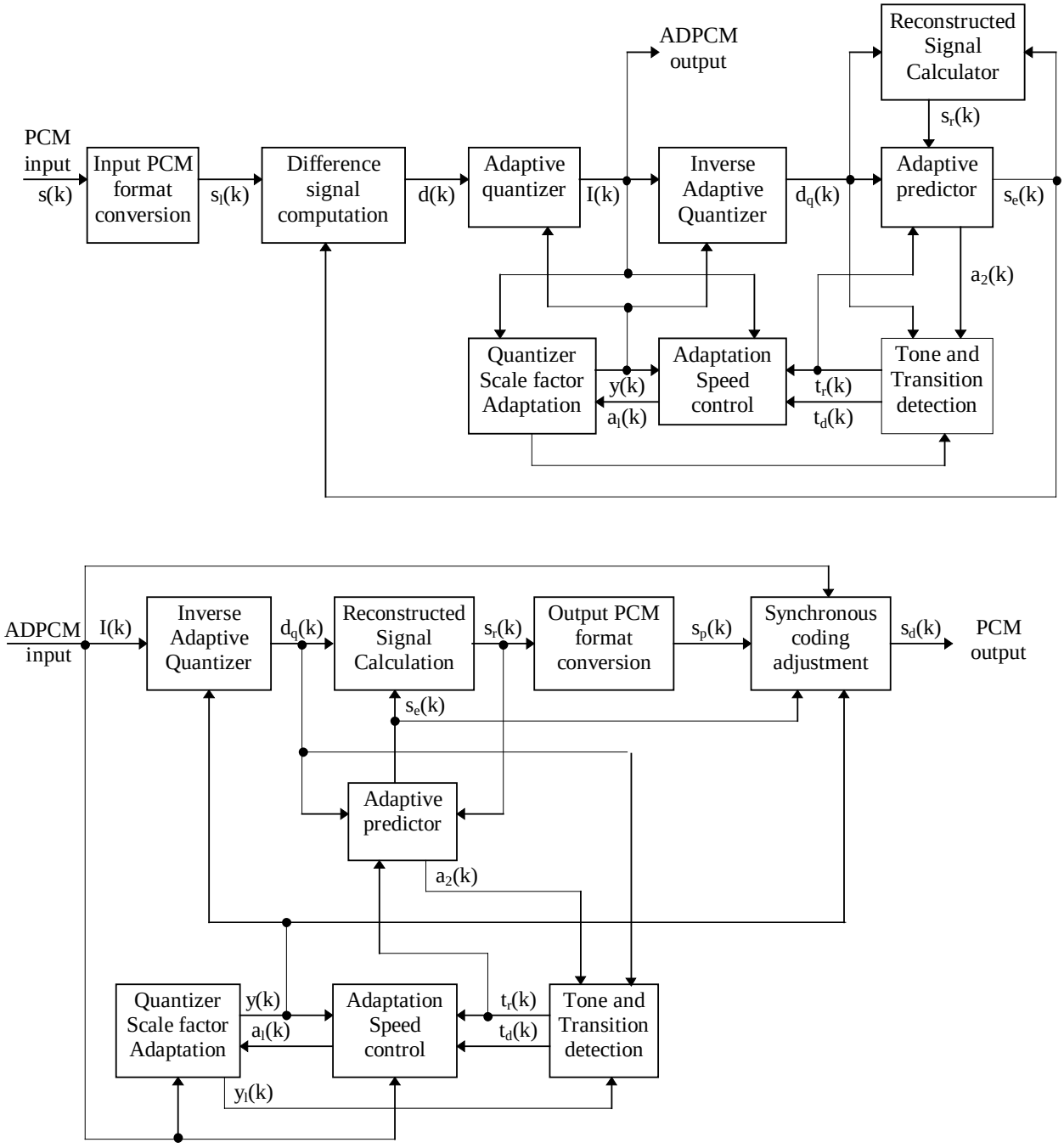

Figure 2: Encoder-Decoder block schematic

4.2.2.1 g726struct.asm

This module defines the data structure used by all the G.726 variants and the appropriate byte, short or long offset. In order to keep the existing

memory stall optimization, this block must be exactly 8n+4 bytes long (currently n=9). A separate block is required for each coder or decoder.

4.2.2.2 g726_all.asm

This module reserves the data RAM for all the channels as defined in the variable “Channels” in the control routine. If multiple rates are required this should only be included in one control routine (32kbps in the example code).

It defines the C-callable sub-routines G726_all_reset and G726_reset, which reset all or one of these channels. The Sub-routine G726reset is called by both G726_all_reset and G726_reset. Whilst it is not C-callable it may have uses. Registers a0..6 and b0..6 are undefined after calling this routine.

It also defines the A-law and μ-Law voice compression and decompression algorithms defined in G711, except that decompression always produces a 14 bit two’s complement number and compression always takes a 15 bit two’s complement number. 4 C-callable subroutines are generated, lin2alaw, alaw2lin, lin2ulaw, and lin2ulaw, performing linear to A-law, A-law to linear, linear to μ-Law and μ-Law to linear conversions respectively.

4.2.2.3 g726coddec.asm

This module must be included via a .copy or .include directive, from a master assembler program in which the variable ‘ADPCM’ has been predefined to be 16, 24, 32 or 40. It will then produce 4 C-callable subroutines (type 3) G726_cod, G726_deca, G726_decu and G726_dec to perform ADPCM coding, ADPCM decoding without synchronous coding adjustment, ADPCM decoding with A-Law synchronous coding adjustment and ADPCM decoding with μ-Law synchronous coding adjustment. These sub-routines may be renamed in the master assembler program which contains them. Synchronous coding adjustment is performed in the decode routines where appropriate but most of the ADPCM coding and decoding is performed by calling 3 subroutines called quantize, iquant, and update in different orders for coding and decoding. The subroutines quantize and iquant are defined in g726quant.asm, whilst the subroutine update is defined in g726predict.asm.

4.2.2.4 g726quant.asm

This module must be included via a .copy or .include directive, from a master assembler program in which the variable ‘ADPCM’ has been predefined to be one of 16, 24, 32 or 40. It contains the subroutines that perform the quantizer, inverse quantizer and part of the quantization level adaption.

4.2.2.5 g726predict.asm

This module must be included via a .copy or .include directive, from a master assembler program in which the variable ‘ADPCM’ has been predefined to be one of 16, 24, 32 or 40. It contains a subroutine that performs the prediction, prediction adaption and the remainder of the quantization adaption.

References

1. ITU Recommendation G.726 - 40, 32, 24, 16 kbps Adaptive Differential Pulse Code Modulation (ADPCM), 1990
2. ITU Recommendation G.721 - 32 kbps Adaptive Differential Pulse Code Modulation (ADPCM), 1988
3. TMS320C62xx CPU and Instruction Set (SPRU189A)
4. TMS320C62xx Programmers Guide (SPRU198).