Digital Voice Echo Canceler Implementation on the TMS320C5x

Application Report

Kevin McCoy
DNA Enterprises

Mansoor A. Chishtie
Digital Signal Processing Applications — Semiconductor Group

SPRA142
October 1994

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.

Introduction

This voice echo canceler implementation on the TMS320C5x is based on a similar implementation on the TMS320C2x [1]. This application report outlines the differences between the two implementations and highlights the specific ’C5x features that support an efficient echo canceler implementation.

This application report extends the ’C2x report with a description of the ’C5x implementation of the algorithm. It is highly recommended that you read both reports to get complete details on the theory and the algorithm used for adaptive filtering and echo cancellation. Although the basic algorithm is the same, the ’C5x implementation is considerably different from that of the ’C2x to take advantage of the ’C5x architecture. These performance improvement techniques are discussed in detail in this application report.

The hardware platform used for testing the ’C5x echo canceler software consists of a ’C5x software development system (SWDS) and an analog front end (AFE) board. The SWDS is a plug-in IBM PC AT card, which is used to debug and run ’C5x code in real time. It has all the necessary hardware hooks to allow an efficient message-passing scheme between the ’C5x and the host PC. The AFE board acts as an analog interface to the ’C5x SWDS. It is made up of two codecs, two telephone hybrid transformers, and clock generation logic for the near-end and the far-end line interfaces.

Although the software is designed to run on an SWDS-AFE platform, very little modification is required to adapt the program to a different target board[1]. The current implementation simulates the following functions in software:

- Near-end round-trip delay
- Far-end round-trip delay
- Near-end echo generation

The near-end round-trip delay directly affects the performance of the echo canceler. This is the time delay of the tail circuit (see Table 3 for details) and is simulated in software in order to analyze the echo canceler performance. The far-end round trip delay is the delay of the forward circuit. The echo generation is implemented in software.

In addition to these simulations, a message-passing scheme is supported by the ’C5x to interface to the host PC via the SWDS hardware. This allows you to monitor the echo canceler performance in real time.

These features are provided to fine-tune the software performance according to each applications requirement. They can be turned off by using software switches (see Table 1 on page 197) during assembly time.

’C5x Device Features Used in This Implementation

The ’C5x architecture is based on the industry-standard TMS320C25 architecture. The ’C5x assembly language is a superset of the TMS320C25 assembly language. However, the ’C5x has an enhanced pipelined architecture that allows it to execute instructions at 50 ns or 25 ns — more than twice the speed of the ’C2x. In addition, the ’C5x has a more powerful set of instructions that allows highly efficient algorithm implementation. Many of these enhanced features are used in this echo canceler implementation.

The rest of this section highlights various features of the ’C5x architecture that distinguish it from the ’C2x family. All code examples are taken from the echo canceler software, but the general comments are equally applicable to any DSP algorithm.

1 Editor’s note: This may be necessary since the ’C5x SWDS is no longer available from Texas Instruments Incorporated. An alternative development platform is the ’C5x evaluation module (EVM).

Dual Mapping of On-Chip Memory

The ’C5x has 1056 words of on-chip dual-access memory, 512 words more than the ’C25. While this type of memory is more efficient to use, it is expensive in terms of silicon real estate. Another type of on-chip memory available on ’C5x devices is single-access memory. The ’C53 and ’C51 have 3K/1K words of single-access memory, while the ’C50 has 8K words. This memory block can be mapped simultaneously in program and data spaces. This dual-mapping feature is very useful for adaptive FIR filters, such as the echo path transversal filter. The multiply/accumulate loops require FIR coefficients in the program space, but the same coefficient table is also accessed in data space to update the transversal filter coefficients. Placing this coefficient table in single-access memory and utilizing its dual-mapping feature make the transversal filter implementation more efficient. Note that the data-move operation (DMOV instruction) works on the single-access RAM (SARAM) block, as well.

Zero-Overhead Loops

The ’C5x features zero-overhead loops, as opposed to the 3-cycle overhead of the ’C25 BANZ (branch on AR not zero) loops. This makes ’C5x looped code as efficient as inline implementation. The code in Example 1 illustrates the use of block repeats in the filter taps update algorithm:

Example 1. Zero-Overhead Loops UPDATE.ASM

```
            lacl        num_a_iter_2
            samm        brcr            ;no. of iterations
            rptb        $block_end–1
                  lacc  *,16,arl        ;start of loop
                  mpya  *+,ar2
                  sach  *0–             ;end of loop
$block_end:
```

In the ’C25 implementation, the same algorithm was coded inline.

Dynamic Addressing of Coefficient Tables

The multiply/accumulate instruction (MAC) on ’C25/’C5x devices fetches input samples of an FIR filter from data memory and takes the filter coefficients from the program memory. This achieves single-cycle, multiply/accumulate operation by simultaneously fetching two operands from memory. Most ’C25/’C5x FIR computations are carried out this way. On the ’C25, the coefficient table address can be specified only in the direct addressing mode. This is adequate for most applications, except where the coefficient table address is determined in runtime. For such cases, the ’C5x provides a register-indirect mode of addressing on multiply/accumulate operations.

Example 2. Echo Estimation Routine FIR.ASM

```
            lacl        last_a          ;update coefficient
            samm        bmar            ;  table address
            lacc        one,14
            zpr                         ;clear preg
            rpt         num_a_1         ;repeat
                  madd  *_              ;multiply/accumulate
            apac                        ;last product
            sach        est_echo,1      ;save echo estimate
```

This feature is used in the echo estimation routine, as shown in Example 2. The block-move-address register (BMAR), a dedicated CPU register, points to the location of the coefficient table in program

memory. This feature is useful when code reuse is a consideration. For the code shown in Example 2, it is particularly important because the length and the location of the transversal filter coefficients are determined in runtime.

Use of Nested Loops

Complex applications like voice echo cancellation often need nested loops. For instance, the block update algorithm for echo filter taps requires two nested loops: an inner loop to compute a time-averaged correlation error for each coefficient in the block and an outer loop to update the coefficient. This can easily be accomplished on the ’C5x by nesting a single-instruction repeat (RPT) inside the block-repeat (RPTB) loop.

Example 3. Coefficient Update Routine TAPINC.ASM

```
            lt          cun0   ;
            lar         ar2, #inc0
            rptb        $calc_INCs–1        :outer loop
               lacc     one, 15
               mpy      *+
               rpt      #14                 ;inner loop
                  mac   pun0+1,*+           ;compute error
               mar      *,ar2
               lta      cun0
               sach     *+,0,arl            ;save coeff update
$calc_INCs:
```

When a single-instruction repeat (RPT) loop cannot be used, block-repeat loops can be nested with delayed-branch loops such as branch-on-AR-not-zero-delayed (BANZD). Up to eight such BANZD loops can be nested, each using an auxiliary register as the loop counter. In ’C25 implementation, the same algorithm is coded in-line.

Maxima/Minima Search

The ’C5x features special instructions to efficiently find minimum (or maximum) value in a data array. Each element in the array can be 32 or fewer bits wide. A signed comparison is made between the accumulator and the accumulator buffer, and the smaller (or greater) of the two values updates the accumulator buffer. This feature is advantageous in the near-end speech detection algorithm.

Example 4. Near-End Speech Detection Routine NESPDET.ASM

```
            lacl        num_m_1   ;
            samm        brcr              ;repeat count
            zap
            sacb                          ;initialize accb for search
            rptb        $max
               lacc     *–,0,ar2          ;get partial maxima M(k)’s
               sacl     *–,0,ar1
               crgt                       ;save largest M(k) in accb
$max:
            sacl        max_m             ;largest M(k) –> max_m
```

The code loop shown in Example 4 performs two functions:

- It finds the largest far-end speech sample (or its power estimate) from a set of the num_m most recent samples.
- It implements a time window spanning the echo path delay range.

On the TMS320C2x, the same algorithm must be implemented with conditional branches. The built-in ’C5x support for search algorithms generates faster and more elegant code.

Circular Buffers

Another ’C5x advantage over the ’C2x is its support for circular addressing. Two independent circular buffers of any size are supported by the ’C5x address generation unit. They can be used to implement FIFO buffers and queues. In this echo canceler application, the two circular buffers are used to hold far-end and near-end receive samples and implement variable delay for near-to-far and far-to-near signal paths.

Another important use of circular addressing is in FIR filter implementations. The conventional way of performing FIR computation on ’C2x/’C5x devices is via a multiply/accumulate with data-move (MACD) operation. In the case of a ’C5x, circular addressing can replace a data-move operation to update filter taps. This is a faster implementation if the filter taps reside in the on-chip single-access memory or the external data memory. The echo simulation filter employs this technique, as shown in Example 5.

Example 5. Echo Simulation Filter EFILT.ASM

```
        mar     *,ar5;
        lar     ar5,efilt_ptr    ;get echo filter taps address
        zap
        rpt     #(filt_len–1)    ;multiply/accumulate
           mac  echo_filt_end,*+ ;  with circular addressing
        apac                     ;add final product
        add     one,14           ;round output
        sach    sim_echo_out,1   ;save as Q15 result
```

Delayed Branches and Conditional Execution

The ’C2x has a three-deep instruction pipeline. This allows it to perform more operations in parallel by overlapping various phases of instructions. The ’C5x features a four-deep instruction pipeline to attain even higher performance. Since deeper pipelines take more cycles to flush, the ’C5x supports special types of branches and calls to avoid this overhead. Normal ’C5x branches take four machine cycles, while a similar instruction on a ’C2x takes only three cycles. However, all ’C5x instructions that cause a pipeline flush support a delayed option that reduces the overhead to only two machine cycles. Moreover, in the special case in which only one or two instructions are skipped over, you can use an even faster instruction, XC (conditional execute), which takes only one machine cycle.

The code shown in Example 6 illustrates the use of delayed branches and conditional execute instructions.

Example 6. Use of Delayed Branches NESPDET.ASM

```
        bd      $chk_hang       ;delayed branch
           sacl  max_m
           lacc  absy0f          ;branch executes here
        sub     max_m
        lar     arl,last_m_1
        xc      2,gt            ;if acc<=0 then skip next two
           lacc  absy0f          ;   instructions
           sacl  max_m
        lacc    num_m_1
        samm    brcr
```

Barrel Shifters

Both the ’C2x and ’C5x DSP families support a 16-bit input prescalar and an 8-bit output postscalar in hardware. This is necessary for efficient fractional arithmetic and bit manipulation. In addition to these barrel-shifters at the input and output paths, the ’C5x family also features a 16-bit right barrel shifter on the accumulator. This complements left barrel shifting provided by the input prescalar. The code in Example 7 illustrates the use of barrel shifters.

Example 7. Code Excerpt From MULAW.ASM

```
   .
   .
   .
        lact   temp_B2   ;Shift left biased linear into ACC
        bsar   16        ;Shift right ACC by 16
        add    #0E0h
        sub    treg1,4   ;Shift left by 4 and subtract
   .
   .
   .
```

The lact instruction uses the left barrel shifter to transfer data to the accumulator, and the input shift is determined by the treg1 register. The following instruction, bsar, performs a 16-bit right barrel shift on the accumulator contents.

Memory-Mapped Registers

Both the ’C2x and the ’C5x have accumulator-based internal architecture. In ’C2x devices, all arithmetic operations are performed on the accumulator. There is no data path between the accumulator and other CPU registers, including the auxiliary register set. Therefore, a temporary data memory location must be used to transfer data between the arithmetic logic unit (ALU) and the address generation unit (AGU).

The ’C5x architecture is considerably enhanced; it provides a direct data path between the accumulator and the rest of the CPU registers by mapping them into local data memory. It also supports direct memory-to-register data transfer on all its internal registers. The code in Example 8 illustrates the use of ’C5x memory-mapped registers.

Example 8. Taps Update Routine UPDATE. ASM

```
update taps:
    splk      #16,indx      ;init. index register
    lar       AR1,#INCO     ;init. aux register 1
    lacc      ADA0
    sub       H
    sacl      ar2           ;init. aux register 2
    lacc      beta_gain     ;get variable beta_gain factor
    samm      treg1         ;init. temp register 1
    lacl      num_a_2
    samm      brcr          ;init. repeat count
    lact      IABSY
    samm      treg0         ;init. temp register 2
    mpy       *+,ar2
    rptb      $block_end–1
        lacc  *,16,arl
        mpya  *+,ar2
        sach  *0–
$block_end
```

Parallel Logic Unit

The ’C5x bit manipulation unit runs independently from its arithmetic logic unit. It allows logical operations on any on-chip or off-chip memory location (including memory-mapped registers) without modifying the accumulator (ACC) or accumulator buffer (ACCB). This feature, in conjunction with the memory mapping of the CPU registers, provides ’C5x programmers more flexibility to modify auxiliary registers to implement software queues and FIFOs. Additionally, the read-modify-write operation performed by the parallel logic unit (PLU) instructions may also be used for semaphore update. The section of code in Example 9 is taken from the echo canceler program. It services the serial port receive interrupt by reading the received data, transmitting new data and setting appropriate flags to communicate with the background program. Notice in particular the use of PLU instructions for setting software flags.

Example 9. Serial Port ISR ECHOISR.ASM

```
rint_isr:
    ldp    #DRR_data
    smmr   drr,#DRR_data    ;get serial receive data
    lmmr   drr,#DXR_data    ;send serial transmit data
    opl    #RXDATA,sp_flag  ;mark serial data received
    apl    #TXDATA,sp flag  ;mark serial port data sent
    opl    #ERINT,intr_flag ;mark rint in intr_flag
    reti
```

Code and Data Requirement

The echo canceler software implementation gives you maximum control over its performance and behavior. Various system parameters, such as the echo filter length, echo cancellation enable/disable mode, and filter adaption enable/disable mode, are represented by memory variables rather than by hard-coding in software. This lets you either:

- Modify these parameters in realtime by the use of supervisory software, as illustrated in the SWDS demo program, or
- Set up these parameters in the initialization stage.

Table 1 lists these user-defined system parameters along with their default values. To modify the default value parameters, edit the echoequ.inc file.

Table 1. User-Defined System Parameters

Number	Variable Name	Description	Type	Default [range]
1	pd_wait	Program/data wait states	const	0h
2	echo_taps	Transversal echo filter taps	const	512 [16-512]
3	sim_echo	Simulated echo disable/enable	const	1 [0/1]
4	host_comm	Host PC communications disable/enable	const	1 [0/1]
5	control_flags†	Bit 0: echo cancellation disable/enable Bit 1: residual suppression disable/enable Bit 2: coeff adaptation disable/enable	variable	1 [0/1] 1 [0/1] 1 [0/1]

† The control_flags variable is active only when host_comm is set to 1. Edit the echoinit.asm file to modify this memory variable.

Table 2 indicates the processor loading and the code size of each software module for a 512-tap implementation. It also indicates where each module is located in program memory. Most of the time-critical subroutines are located in the on-chip single-access random-access memory (SARAM). The auxiliary functions, such as the host PC mailbox, are executed from external memory.

Table 2. Program Module Requirements

Number	Module Name	Description	CPU Cycles†	Code Size	Code Location‡
1	ECHO.ASM	Main module — variable declarations.	–	2	ROM
2	ECHOINIT.ASM	Initialization module.	–	218	ROM
3	ECHOISR.ASM	Interrupt services routines.	17	56	ROM
4	CYCLE.ASM	Get new samples. Convert μ-law to linear. Poll host PC mailbox.	67	71	SARAM
5	EFILT.ASM	AR for the echo simulation. Update delay buffers.	50	21	SARAM
6	FIR.ASM	Estimate echo. Compute error.	546	21	SARAM
7	RESID.ASM	Residual error suppressor.	17	16	SARAM
8	MULAW.ASM	Linear-to-PCM conversion.	41	28	SARAM
9	PCALC.ASM	Power estimate of y(n) and o(n).	39	19	SARAM
10	NESPDET.ASM	Near-end speech detection.	47	83	SARAM
11	ONORM.ASM	Output normalization for coefficient update.	55	32	ROM
12	TAPINC.ASM	Tap increment.	791	32	ROM
13	UPDATE.ASM	FIR filter tap update.	153	27	ROM
14	UTIL.ASM	Process host PC commands. Write monitored variables.	–	233	ROM
15	MAILBOX.ASM	Host PC mailbox.	–	41	ROM
			Total cycles for 512-tap filter = 1825	Total code size = 900 words	

† Only for the modules that are in the main cycle. Cycle count given for 512 taps transversal echo filter.
‡ ROM = 'C51 on-chip, read-only memory or external memory.
SARAM = 'C51 on-chip, single-access RAM.

Data Allocation

The 'C51 has 1056 words of dual-access and 1024 words of single-access on-chip memory. It also has 8K words of on-chip, read-only memory. The on-chip data memory is allocated to various modules of the echo canceler software according to their specific requirements. Table 3 lists the size and the location of various data variables for a 512-tap implementation.

The coefficients of the echo transversal filter are placed in the on-chip, single-access memory because of its dual-mapping capability. Note that these coefficients are accessed in both program and data spaces by two different modules.

The 1024 words of dual-access memory are used for data storage. Reference samples of the far-end talker reside in this memory block. This makes efficient use of multiply-accumulate-with-data-move-type operations.

To simulate delay paths between near-end and far-end speakers, two long buffers of 2K words each are maintained in external data memory. Another buffer that holds host PC messages resides in external memory. Since all three buffers are in noncritical paths and would eventually be deleted from the final implementation, they are placed in external memory.

Table 3. 512-Tap Implementation Data Variables

On-Chip Single-Access Memory: 528 Words	
16 words	Normalized outputs Un0 – Un15
512 words	Transversal echo filter coefficients A0 – A15
On-Chip Dual-Access Memory: 655 Words	
62 words	System variables
33 words	Local maxima M(k) for near-end speech detection
32 words	Coefficient increment INC(k)
528 words	Reference samples Y(k)
External Data Memory	
2304 words	Near-to-far sample delay buffer (optional)
2304 words	Far-to-near sample delay buffer (optional)
2048 words	Message buffer for PC communications (optional)

Code Benchmarks

The two most computationally intensive routines of this echo canceler application are:

- The transversal echo filter routine FIR.ASM, and
- The mean square error (MSE) computation routine TAPINC.ASM.

The computational requirement for these two routines depends on the length of the echo transversal filter.

Table 4 shows the relationship between the processor loading and the length of the transversal filter. For a 512-tap filter, the ’C5x takes only 92 microseconds to process each sample. With an input sampling rate of 128 microseconds, this leaves the processor with ample time for system overhead. In fact, a 50-ns ’C5x processor can implement about 750 echo filter taps within a 128-microsecond sampling period. In other words, one 50-ns ’C5x DSP can handle 96 ms of the tail-end circuit delay.

Table 4 shows code benchmarks for a hardware platform that consists of the ’C51 software development system (SWDS) with an analog front end (AFE) board, a zero-wait-state external data/program memory, a 50-ns instruction cycle rate, a 128-µs input sampling period, and PC communication disabled.

Table 4. Code Benchmarks

Number	Echo Filter Taps	Time Required to Process One Sample
1	32	26.0 µs
2	48	28.1 µs
3	64	30.0 µs
4	80	32.4 µs
5	96	34.6 µs
6	128	38.9 µs
7	256	56.7 µs
8	512	91.6 µs

Echo Canceler Demonstration on a ’C5x SWDS

The primary hardware platform for testing the ’C5x echo canceler software (for code benchmarks) was a ’C5x SWDS. The AFE board communicates with the ’C5x DSP via its serial port and has codecs and hybrid transformers for near-end and far-end telephone interfaces. An AFE board schematic is shown in the appendix of this report.

You can run the demonstration software on any ’C5x SWDS board by downloading the echo.out file to the board and running the echodemo.exe file on the host PC. To do this, type the following two commands at the DOS prompt:

```
c51load echo.out
echodemo.exe
```

You can control the various system parameters — such as tail-circuit delay, transversal filter taps, echo cancellation mode, and adaptation mode — in real time by running the echodemo.exe program.

Conclusion

This implementation of a single-channel voice echo canceler on a TMS320C51 highlights the powerful and versatile architecture of that DSP. This particular algorithm was first coded on a TMS32020. Coding the same algorithm on a TMS320C51 shows that the resulting performance improvement is not merely due to the faster instruction rate on the 'C5x. Performance is improved by more than a factor of two when enhanced 'C5x architecture is fully utilized. The 'C5x features used in this implementation are discussed in detail. The processor loading and the code and data size of each software module are listed. Several auxiliary functions that are used for testing and evaluation purposes are discussed. The details of a demonstration package that consists of a 'C51 SWDS, an analog front-end board, a 'C5x DSP, and PC software are given.

Acknowledgements

Texas Instruments acknowledges the efforts of the DNA Enterprises' project team: Kevin McCoy, Mark Sissom, and Paul Kniffen.

Code Availability

The associated program files are available from the Texas Instruments TMS320 Bulletin Board System (BBS) at (713) 274-2323. Internet users can access the BBS via anonymous ftp at *ti.com.*

References

1. "Digital Voice Echo Canceler With a TMS32020", *Digital Signal Processing Applications*, Volume 1, Texas Instruments, 1986, pp. 415−454.

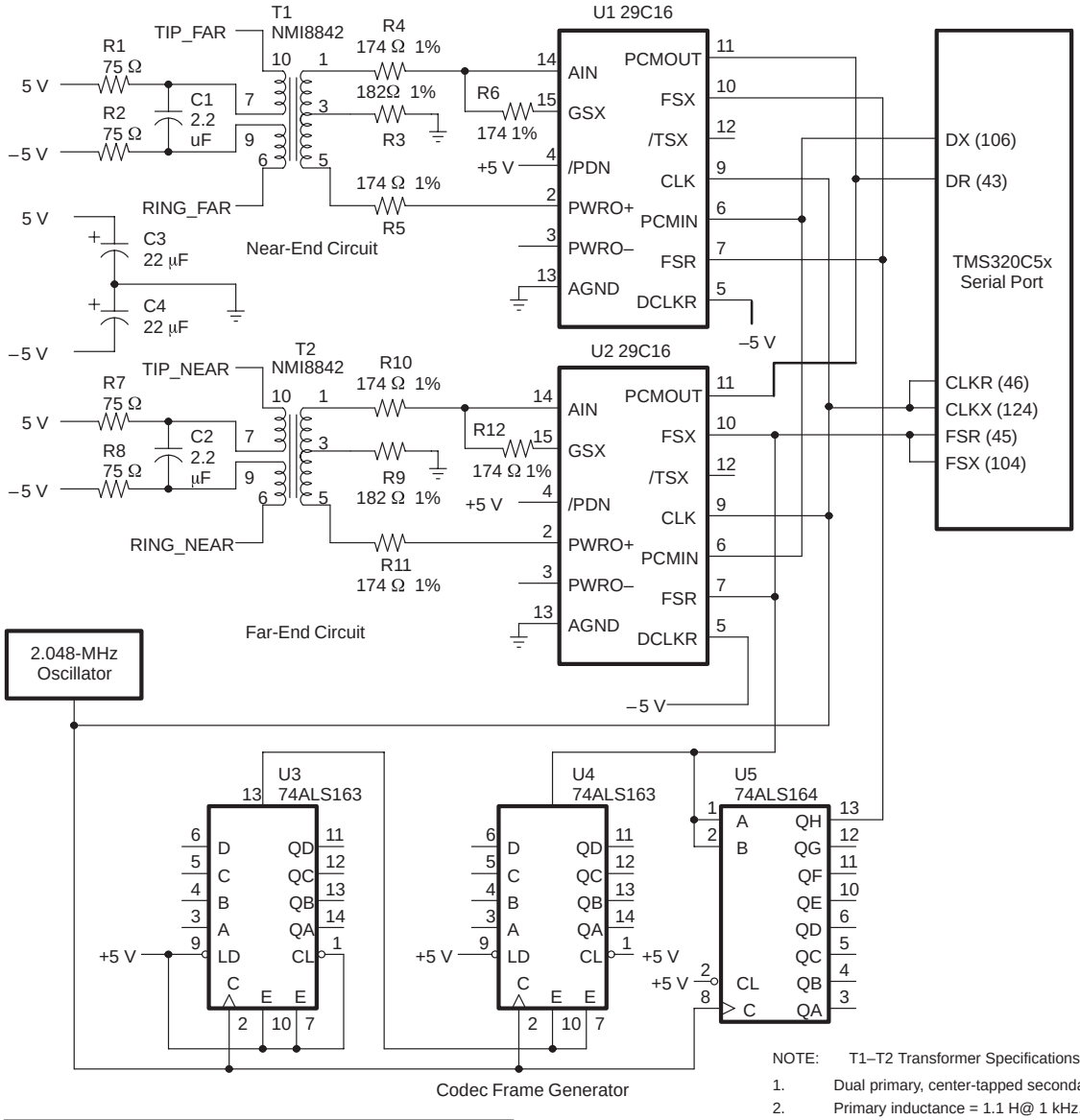
2. *TMS320C5x User's Guide*, Texas Instruments, 1991.

3. *TMS320C5x Software Development System Technical Reference,* Texas Instruments, 1990.

4. *TMS320C5x C Source Debugger User's Guide,* Texas Instruments, 1991.

5. *TMS320 Fixed-Point DSP Assembly Language Tools User's Guide,* Texas Instruments, 1990.

Appendix: Schematic of the Dual-Telephone Interface for the TMS320C51 SWDS



DNA Enterprises. Inc.
Dual-Telephone Interface for the TMS320C51 SWDS

NOTE: T1–T2 Transformer Specifications

1. Dual primary, center-tapped secondary
2. Primary inductance = 1.1 H@ 1 kHz. 65 mA
3. Saturation current > 70 mA
4. 1 : 1 : 1 : 1 turns ratio
5. Average winding resistance = 62.5 Ω
6. Primary winding resistances should match within 1%.
7. Wind 1/2 secondary first; tape, then wind primary bifilar; tape, then wind rest of secondary.
8. Coefficient of coupling (K) >= 0.998
9. Pri-to-pri and pri-to-sec Hipot = 500 V (This is a single-coil, hybrid transformer.)