Application Report
SPRAC13–January 2016

TI DSP Benchmarking

Mark Nadeski

ABSTRACT

This application report provides benchmarks for the C674x DSP core, the C66x DSP core and the ARM® Cortex®-A15 core. This document also shows how to reproduce these benchmarks on specific hardware platforms. For more benchmarks, see TI's DSP Benchmarks Page.

Contents

1 Processor Core Benchmarks

The benchmarks in Table 1 are for a single core.

Table 1. Processor Core Benchmarks (1)

Processor Core	C66x DSP Core		C674x DSP Core		ARM Cortex-A15		
Hardware Platform Used	C6657 EVM		C6748 LCDK		AM5728 EVM		
Devices Featuring Benchmarked Core	C66x DSPs 66AK2x DSPs Sitara AM57x SoC's		OMAP-L138 C674x DSPs		66AK2x DSPs Sitara AM57x SoC's		
	C66x Execution Time		C674x Execution Time		ARM Cortex-A15 Execution Time2		
Function Benchmarked	C66x Cycles	C66x µS @ 1 GHz	C674x Cycles	C674x µS @ 456 Mhz	Cortex - A15 Cycles (3)	Cortex-A15 µS@1GHz (3)	Associated TI Library
Complex FFT (256 pts) - SP floating point (2)	1782	1.78	2401	5.27	8644	8.64	FFTLIB for C66x DSPLIB for C674x
Complex FFT (1k pts) - SP floating point (2)	6269	6.27	10950	24.01	43916	43.92	
Real block FIR - fixed point 128 samples, 16 coeff	262	0.26	386	0.85	2152	2.15	DSPLIB
Real block FIR - SP floating point 128 samples, 16 coeff	1345	1.35	1406	3.08	6971	6.97	DSPLIB
Real block FIR - SP floating point 256 samples, 16 coeff	2625	2.63	2735	6.00	13879	13.88	DSPLIB
Complex block FIR - SP floating point 64 samples, 16 coeff	1334	1.33	2221	4.87	13039	13.04	DSPLIB
Complex block FIR - SP floating point 128 samples, 16 coeff	2646	2.65	4397	9.64	26072	26.07	DSPLIB
Real Matrix SGEMM 16x16	2405	2.41	3505	7.69	14662	14.66	DSPLIB
Complex Matrix SGEMM 16x16	4113	4.11	10884	23.87	26388	26.39	DSPLIB
Matrix Math DGEMM 16x16	5061	5.06	–	–	14669	14.67	DSPLIB
Autocorrelation - fixed point N=32 , IMG_corr_3x3_i16s_c16s	140	0.14	189	0.41	946	0.95	IMGLIB
ArcTan2 - SP floating point	24	0.02	31	0.07	49	0.05	MATHLIB
Log10 - Single precision	14	0.01	18	0.04	56	0.06	MATHLIB
Square Root - single precision float	6	0.01	6	0.01	5	0.01	MATHLIB

(1) All benchmarks measured with data located in L2 SRAM.

(2) C66x FFT code benchmarked is an optimized version of the FFT kernel code from FFTLIB using L2 memory.

(3) A15 benchmarks with data in OCMC RAM. Data and program cache enabled. Compiler flags used for ARM Neon optimizations are –mfpu =vfpv4 –mfloat-abi = hard -03. The A15 outputs not verified for accuracy and precision. No hand written intrinsics used in the code.

2 C674x DSP Benchmarking

This section details what is needed to reproduce the C674x core benchmarks from Table 1.

- Benchmarks performed using a C6748 LCDK.
- Code Composer Studio ™ CCSv5.4
- Libraries
  - DSP library dsplib_c674x_3_4_0_0
  - Math library mathlib_c674x_3_1_1_0
  - Image library imglib_c64Px_3_2_0_1
- Enable Cache prior to running the benchmark. The Cache enable example code is available as part of the BIOS SDK http://www.ti.com/tool/bioslinuxmcsdk under pdk_OMAPL138_1_01_00_02\packages\ti\csl\example\omapl138-lcdk\cache\src directory.

- Data is placed in L2SRAM
- The Benchmarking projects are available as part of the libraries and contain all required build options. The little endian elf version of the projects is used for benchmarking.
- Modify the linker command files in the library projects as shown below.

2.1 DSP Library Projects Used for Benchmarking

- DSPF_sp_fft_SPxSP_674_LE_ELF – Complex FFT – Single precision
- DSP_fir_r8_h16_674_LE_ELF – Real Block FIR - Fixed point
- DSPF_fir_gen_674_LE_ELF – Real Block FIR – Single precision
- DSPF_fir_cmplx_674_LE_ELF – Complex Block FIR – Single precision
- DSPF_sp_mat_mul_gemm_674_LE_ELF – General real matrix multiply – Single precision
- DSPF_sp_mat_mul_gemm_cmplx_674_LE_ELF – General complex matrix multiply – single precision

2.1.1 Modified DSPLIB Linker Command File

```
-c
-heap  0x1000
-stack 0x1000

-l ../../../../../lib/dsplib.lib
-l ../../../../../lib/dsplib_cn.lib

MEMORY
{
    L2SRAM (RWX) : org = 0x800000, len = 0x100000
    MSMCSRAM (RWX) : org = 0xc000000, len = 0x200000
}

SECTIONS
{
    .kernel: {
       *.obj (.text:optimized) { SIZE(_kernel_size) }
    }

    .text: load >> L2SRAM
    .text:touch: load >> L2SRAM

    GROUP (NEAR_DP)
    {
    .neardata
    .rodata
    .bss
    } load > L2SRAM

    .far: load >> L2SRAM
    .fardata: load >> L2SRAM
    .data: load >> L2SRAM
    .switch: load >> L2SRAM
    .stack: load > L2SRAM
    .args: load > L2SRAM align = 0x4, fill = 0 {_argsize = 0x200; }
    .sysmem: load > L2SRAM
    .cinit: load > L2SRAM
    .const: load > L2SRAM START(const_start) SIZE(const_size)
    .pinit: load > L2SRAM
    .cio: load >> L2SRAM
    xdc.meta: load >> L2SRAM, type = COPY
}
```

2.2 Math Library Projects Used for Benchmarking

- log10sp_674LE_LE_ELF – log10 – Single precision
- rsqrtsp_674LE_LE_ELF – square root – Single precision
- atan2sp_674LE_LE_ELF – arctan2 – Single precision

2.2.1 Modified MATHLIB Linker Command File

```
-l ../../../../../lib/mathlib.lib

MEMORY
{
  L2SRAM (RWX)  : org = 0x0800000, len = 0x080000
  MSMCSRAM (RWX): org = 0xc000000, len = 0x200000
}

SECTIONS
{
  .kernel_asm: {
    mathlib*<*.o*>  (.text:optasm) { SIZE(_kernel_asm_size) }
  }

  .kernel_vec: {
    mathlib*<*.o*>  (.text:optvec) { SIZE(_kernel_vec_size) }
  }

  .kernel_ci: {
    mathlib*<*.o*>  (.text:optci) { SIZE(_kernel_ci_size) }
  }

  .text:            load >> L2SRAM
  .text:touch:      load >> L2SRAM

  GROUP (NEAR_DP)
  {
    .neardata
    .rodata
    .bss
  } load > L2SRAM

  .init_array: load >> L2SRAM
  .far:        load >> L2SRAM
  .fardata:    load >> L2SRAM
  .data:       load >> L2SRAM
  .switch:     load >> L2SRAM
  .stack:      load >  L2SRAM
  .args:       load >  L2SRAM align = 0x4, fill = 0 {_argsize = 0x200; }
  .sysmem:     load >  L2SRAM
  .cinit:      load >  L2SRAM
  .const:      load >  L2SRAM START(const_start) SIZE(const_size)
  .pinit:      load >  L2SRAM
  .cio:        load >> L2SRAM
   xdc.meta:   load >> L2SRAM, type = COPY
}
```

2.3 IMG Library Projects Used for Benchmarking

- IMG_corr_3x3_i16s_c16s_64P_LE_ELF – Image correlation 16-bit fixed point

2.3.1 Modified IMGLIB Linker Command File

```
-heap  0x8000
-stack 0xC000

-stack 0x2000
-heap 0x10000
-l ../../../../../lib/imglib.lib
-l ../../../../../lib/imglib_cn.lib
-l ../../../../../lib/common.lib


MEMORY
{
    L1DRAM      o = 0x11F04000 l = 0x000c000
    L2_SRAM     o = 0x00800000 l = 0x40000
    DDR         o = 0x80000000 l = 0x1000000
}

SECTIONS
{
  .csl_vect > L2_SRAM

  .kernel
  {
     imglib*<*.o*> (.text:optimized)
  }  SIZE(_kernel_size) > L2_SRAM

  .fardata: > L2_SRAM
  .neardata > L2_SRAM
  .rodata   > L2_SRAM
  .sysmem   > L2_SRAM
  .text     > L2_SRAM
  .bss      > L2_SRAM
  .switch   > L2_SRAM
  .cinit    > L2_SRAM
  .data     > L2_SRAM
  .const    > L2_SRAM
  .far      > L2_SRAM
  .stack    > L2_SRAM
  .cio      > L2_SRAM
  .reset    > L2_SRAM
}
```

3 C66x DSP Benchmarking

This section details what is needed to reproduce the C66x core benchmarks from Table 1.

- C66x benchmarks measured on a C6657 EVM
- Code composer studio CCSv5.4 on single core of C6657
- Code composer studio CCSv6.1 on single core of C66x DSP on AM57x EVM
- CCSv5 library projects are imported on the CCSv6
  - For importing the projects, see the CCS User's Guide (http://processors.wiki.ti.com/index.php/Importing_Projects_into_CCS)
- Libraries
  - DSP library dsplib_c66x_3_4_0_0
  - Math library mathlib_c66x_3_1_1_0
  - Image library imglib_c66x_3_2_0_1
  - FFT library fftlib_k2hx_2_0_0_2

- Data is placed in L2SRAM
- The Benchmarking projects are available as part of the libraries and contain all required build options. The little endian elf version of the projects is used for benchmarking.
- Cache is enabled by the Gel files loaded by the CCS for DSPs.
- Modify the linker command files in the library projects as shown in Section 3.1.

3.1 DSP and FFT Library Projects Used for Benchmarking

- FFTlib fft_sp_cmplx_notwid_br – Complex FFT – Single precision
- DSP_fir_r8_h16_66_LE_ELF – Real Block FIR - Fixed point
- DSPF_fir_gen_66_LE_ELF – Real Block FIR – Single precision
- DSPF_fir_cmplx_66_LE_ELF – Complex Block FIR – Single precision
- DSPF_sp_mat_mul_gemm_66_LE_ELF – General real matrix multiply – Single precision
- DSPF_sp_mat_mul_gemm_cmplx_66_LE_ELF – General complex matrix multiply – Single precision
- DSPF_dp_mat_mul_gemm_66_LE_ELF – General matrix multiply – Double precision

3.1.1 Modified DSPLIB Linker Command File

```
-heap  0x8000
-stack 0xC000

-l ti/dsplib/lib/dsplib.lib
-l ti/dsplib/lib/dsplib_cn.lib
-l ti/mathlib/lib/mathlib.lib

MEMORY
{
  L2SRAM (RWX)  : org = 0x0800000, len = 0x080000
  MSMCSRAM (RWX): org = 0xc000000, len = 0x200000
  DDR (RWX)     :org = 0x80000000, len = 0x2000000
}

SECTIONS
{

  .kernel: {
    dsplib*<*.o*> (.text:optimized) { SIZE(_kernel_size) }
  }


  .text:            load >> L2SRAM
  .text:touch:      load >> L2SRAM

  GROUP (NEAR_DP)
  {
    .neardata
    .rodata
    .bss
  } load > L2SRAM
  .my_sect_ddr  >> L2SRAM
  .init_array: load >> L2SRAM
  .far:        load >> L2SRAM
  .fardata:    load >> L2SRAM
  .neardata    load >> L2SRAM
  .rodata      load >> L2SRAM
  .data:       load >> L2SRAM
  .switch:     load >> L2SRAM
  .stack:      load >  L2SRAM
  .args:       load >  L2SRAM align = 0x4, fill = 0 {_argsize = 0x200; }
  .sysmem:     load >  L2SRAM
  .cinit:      load >  L2SRAM
  .const:      load >  L2SRAM START(const_start) SIZE(const_size)
```

```
  .pinit:      load >  L2SRAM
  .cio:        load >> L2SRAM
   xdc.meta:   load >> L2SRAM, type = COPY
```

3.2 Math Library Projects Used for Benchmarking

- log10sp_66_LE_ELF – log10 – Single precision
- rsqrtsp_66_LE_ELF – square root – Single precision
- atan2sp_66_LE_ELF – arctan2 – Single precision

3.2.1 Modified MATHLIB Linker Command File

```
-heap  0x8000
-stack 0xC000

-l ../../../../../lib/mathlib.lib

MEMORY
{
  L2SRAM (RWX)  : org = 0x0800000, len = 0x080000
  MSMCSRAM (RWX): org = 0xc000000, len = 0x200000
}

SECTIONS
{
  .kernel_asm: {
    mathlib*<*.o*>  (.text:optasm) { SIZE(_kernel_asm_size) }
  }

  .kernel_vec: {
    mathlib*<*.o*>  (.text:optvec) { SIZE(_kernel_vec_size) }
  }

  .kernel_ci: {
    mathlib*<*.o*>  (.text:optci) { SIZE(_kernel_ci_size) }
  }

  .text:            load >> L2SRAM
  .text:touch:      load >> L2SRAM

  GROUP (NEAR_DP)
  {
    .neardata
    .rodata
    .bss
  } load > L2SRAM

  .init_array: load >> L2SRAM
  .far:        load >> L2SRAM
  .fardata:    load >> L2SRAM
  .data:       load >> L2SRAM
  .switch:     load >> L2SRAM
  .stack:      load >  L2SRAM
  .args:       load >  L2SRAM align = 0x4, fill = 0 {_argsize = 0x200; }
  .sysmem:     load >  L2SRAM
  .cinit:      load >  L2SRAM
  .const:      load >  L2SRAM START(const_start) SIZE(const_size)
  .pinit:      load >  L2SRAM
  .cio:        load >> L2SRAM
   xdc.meta:   load >> L2SRAM, type = COPY
}
```

3.3 IMG Library Projects Used for Benchmarking

- IMG_corr_3x3_i16s_c16s_66_LE_ELF – Image correlation 16bit fixed point

3.3.1 Modified IMGLIB Linker Command File

```
-heap  0x8000
-stack 0xC000
-l ../../../../../lib/imglib.lib
-l ../../../../../lib/imglib_cn.lib
-l ../../../../../lib/common.lib


MEMORY
{
  L2SRAM (RWX)  : org = 0x0800000, len = 0x080000
  MSMCSRAM (RWX): org = 0xc000000, len = 0x200000
}

SECTIONS
{
  .kernel: {
    imglib*<*.o*> (.text:optimized) { SIZE(_kernel_size) }
  }

  .text:            load >> L2SRAM
  .text:touch:      load >> L2SRAM

  GROUP (NEAR_DP)
  {
    .neardata
    .rodata
    .bss
  } load > L2SRAM

  .init_array: load >> L2SRAM
  .far:        load >> L2SRAM
  .fardata:    load >> L2SRAM
  .neardata    load >> L2SRAM
  .rodata      load >> L2SRAM
  .data:       load >> L2SRAM
  .switch:     load >> L2SRAM
  .stack:      load >  L2SRAM
  .args:       load >  L2SRAM align = 0x4, fill = 0 {_argsize = 0x200; }
  .sysmem:     load >  L2SRAM
  .cinit:      load >  L2SRAM
  .const:      load >  L2SRAM START(const_start) SIZE(const_size)
  .pinit:      load >  L2SRAM
  .cio:        load >> L2SRAM
   xdc.meta:   load >> L2SRAM, type = COPY
}
```

4 ARM Cortex-A15 Benchmarking

This section details what is needed to reproduce the ARM Cortex-A15 benchmarks from Table 1.

4.1 Library Projects Used for Benchmarking

- Benchmarks measured on an AM572x EVM
- Code composer studio CCSv6.1 on single core of A15 on AM572x EVM
- Reference C code is used from the DSP Library projects listed in previous section for creating A15 projects
  - C66x DSP library for fft, fir, gemm, sgemm, dgemm
    - Single precision complex FFT reference C code is taken from DSP library and not from the FFT library.
  - C66x Math library for arctan2, log10sp, rsqrtsp
  - C66x IMG library for image correlation
- Data in placed OCMC RAM, data and program cache enabled
- Neon Optimized flags enabled -mfpu = vfpv4 -mfloat-abi = hard and optimization level -03
- No hand written Intrinsics used in the code

The example code for enabling cache and MMU is below. The dependent header files are available as part of the CSL in the Platform development kit of AM57x BIOS SDK pdk_am57xx_1_0_0.

The path for the CSL folder is pdk_am57xx_1_0_0\packages\ti\csl.

4.1.1 Modified IMGLIB Linker Command File

```
#include <stdio.h>
#include <ti/csl/tistdtypes.h>
#include <ti/csl/csl_a15Aux.h>
/* ========================================================================== */
/*                       Internal Function Declarations                       */
/* ========================================================================== */

#define MMU_PAGETABLE_ALIGN_SIZE                          (16U * 1024U)


/** \brief Page tables to hold physical to virtual address mapping. The start
           address of the page table must be aligned at 16K boundary */
    CSL_A15MmuLongDescObj mmuObj
     __attribute__((aligned(MMU_PAGETABLE_ALIGN_SIZE)));

    CSL_A15MmuLongDescAttr mmuAttr0;
    CSL_A15MmuLongDescAttr mmuAttr1;
    CSL_A15MmuLongDescAttr mmuAttr2;



void systemInit(void)
{
    uint32_t phyAddr = 0U;

    mmuObj.numFirstLvlEntires = CSL_A15_MMU_LONG_DESC_LVL1_ENTIRES;
    mmuObj.numSecondLvlEntires = CSL_A15_MMU_LONG_DESC_LVL2_ENTIRES;
    mmuObj.mairEntires = CSL_A15_MMU_MAIR_LEN_BYTES;
    mmuObj.mairAttr[0] = 0x44U;
    mmuObj.mairAttr[1] = 0x0U;
    mmuObj.mairAttr[2] = 0xFFU;
    CSL_a15InitMmuLongDesc(&mmuObj);

    CSL_a15InitMmuLongDescAttrs(&mmuAttr0);
    CSL_a15InitMmuLongDescAttrs(&mmuAttr1);
    CSL_a15InitMmuLongDescAttrs(&mmuAttr2);
```

```
    mmuObj.mairAttr[0] = 0x44U;
    mmuObj.mairAttr[1] = 0x0U;
    mmuObj.mairAttr[2] = 0xFFU;
    CSL_a15SetMmuMair(0, mmuObj.mairAttr[0]);
    CSL_a15SetMmuMair(1, mmuObj.mairAttr[1]);
    CSL_a15SetMmuMair(2, mmuObj.mairAttr[2]);

    mmuAttr0.type = CSL_A15_MMU_LONG_DESC_TYPE_BLOCK;
    mmuAttr0.accPerm = 0U;
    mmuAttr0.shareable = 2U;
    mmuAttr0.attrIndx = 1U;

    for (phyAddr = 0x40000000U; phyAddr < 0x60000000U; phyAddr += 0x00200000U)
    {
        CSL_a15SetMmuSecondLevelLongDesc(&mmuObj, (void *)phyAddr, (void *)phyAddr,
    &mmuAttr0);
    }

    mmuAttr1.type = CSL_A15_MMU_LONG_DESC_TYPE_BLOCK;
    mmuAttr1.accPerm = 0U;
    mmuAttr1.shareable = 2U;
    mmuAttr1.attrIndx = 2U;

    for (phyAddr = 0x80000000; phyAddr < 0xA0000000; phyAddr += 0x00200000U)
    {
        CSL_a15SetMmuSecondLevelLongDesc(&mmuObj, (void *)phyAddr, (void *)phyAddr,
    &mmuAttr1);
    }

    mmuAttr2.type = CSL_A15_MMU_LONG_DESC_TYPE_BLOCK;
    mmuAttr2.accPerm = 0U;
    mmuAttr2.attrIndx = 0U;

    for (mmuAttr2.phyAddr[0U] = 0xA0000000; mmuAttr2.phyAddr[0U] < 0xB0000000;
        mmuAttr2.phyAddr[0U] += 0x00200000U)
    {
        CSL_a15SetMmuSecondLevelLongDesc(&mmuObj, (void *)phyAddr, (void *)phyAddr,
    &mmuAttr2);
    }
    CSL_a15EnableCache();
    CSL_a15InvAllDataCache();
    CSL_a15InvAllInstrCache();
    CSL_a15EnableMmu();
}
```

4.1.2 Modified Linker Command File (AM572x.lds)

```
MEMORY
{
    OCMC_RAM1 :  o = 0x40300000,  l = 0x00080000  /* 512kB L3 OCMC RAM1 */
    OCMC_RAM2 :  o = 0x40400000,  l = 0x00100000  /* 1MB L3 OCMC RAM2 */
    OCMC_RAM3 :  o = 0x40500000,  l = 0x00100000  /* 1MB L3 OCMC RAM3 */
    DDR0 :       o = 0x80000000,  l = 0x40000000  /* 1GB external DDR Bank 0 */
    DDR1 :       o = 0xC0000000,  l = 0x40000000  /* 1GB external DDR Bank 1 */
}

ENTRY(Entry)

SECTIONS
{
    .rsthand :
    {
        . = ALIGN(0x10000);
        KEEP(*(.isr_vector))
        *startup_ARMCA15.o (.text)
```

```
} > OCMC_RAM1

. = ALIGN(4);
.text :
{
    *(.text*)

    KEEP(*(.init))
    KEEP(*(.fini))

    /* .ctors */
    *crtbegin.o(.ctors)
    *crtbegin?.o(.ctors)
    *(EXCLUDE_FILE(*crtend?.o *crtend.o) .ctors)
    *(SORT(.ctors.*))
    *(.ctors)

    /* .dtors */
     *crtbegin.o(.dtors)
     *crtbegin?.o(.dtors)
     *(EXCLUDE_FILE(*crtend?.o *crtend.o) .dtors)
     *(SORT(.dtors.*))
     *(.dtors)

    *(.rodata*)

    KEEP(*(.eh_frame*))
} > OCMC_RAM1

.ARM.extab :
{
    *(.ARM.extab* .gnu.linkonce.armextab.*)
} > OCMC_RAM1

__exidx_start = .;
.ARM.exidx :
{
    *(.ARM.exidx* .gnu.linkonce.armexidx.*)
} > OCMC_RAM1
__exidx_end = .;


.data :
{
    . = ALIGN(4);
    __data_start__ = .;
    *(vtable)
    *(.data*)

    . = ALIGN(4);
    /* preinit data */
    PROVIDE_HIDDEN (__preinit_array_start = .);
    KEEP(*(.preinit_array))
    PROVIDE_HIDDEN (__preinit_array_end = .);

    . = ALIGN(4);
    /* init data */
    PROVIDE_HIDDEN (__init_array_start = .);
    KEEP(*(SORT(.init_array.*)))
    KEEP(*(.init_array))
    PROVIDE_HIDDEN (__init_array_end = .);


    . = ALIGN(4);
    /* finit data */
    PROVIDE_HIDDEN (__fini_array_start = .);
```

```
            KEEP(*(SORT(.fini_array.*)))
            KEEP(*(.fini_array))
            PROVIDE_HIDDEN (__fini_array_end = .);

            . = ALIGN(4);
            /* All data end */
            __data_end__ = .;

        } > OCMC_RAM1

        .bss :
        {
            . = ALIGN(4);
            __bss_start__ = .;
            *(.bss*)
            *(COMMON)
            __bss_end__ = .;
        } > OCMC_RAM1

        .heap (NOLOAD):
        {
            FILL(0xDEADBEEF)
            . = ALIGN(4);
            __end__ = .;
            end = __end__;
            __HeapBase = __end__;
            *(.heap*)
            . = . + HEAPSIZE;
            __HeapLimit = .;
        } > OCMC_RAM1

        .stack (NOLOAD):
        {

            FILL(0xBAD0BAD0)
            . = ALIGN(4);
            __StackLimit = . ;
            *(.stack*)
            . = . + STACKSIZE;
            __StackTop = . ;
                    __StackBase = . ;
        } > OCMC_RAM1
        PROVIDE(__stack = __StackTop);

    }
```

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, enhancements, improvements and other changes to its semiconductor products and services per JESD46, latest issue, and to discontinue any product or service per JESD48, latest issue. Buyers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All semiconductor products (also referred to herein as “components”) are sold subject to TI’s terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its components to the specifications applicable at the time of sale, in accordance with the warranty in TI’s terms and conditions of sale of semiconductor products. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by applicable law, testing of all parameters of each component is not necessarily performed.

TI assumes no liability for applications assistance or the design of Buyers’ products. Buyers are responsible for their products and applications using TI components. To minimize the risks associated with Buyers’ products and applications, Buyers should provide adequate design and operating safeguards.

TI does not 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 relating to any combination, machine, or process in which TI components or services are used. Information published by TI regarding third-party products or services does not constitute a license to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of significant portions of TI information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

Resale of TI components or services with statements different from or beyond the parameters stated by TI for that component or service voids all express and any implied warranties for the associated TI component or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

Buyer acknowledges and agrees that it is solely responsible for compliance with all legal, regulatory and safety-related requirements concerning its products, and any use of TI components in its applications, notwithstanding any applications-related information or support that may be provided by TI. Buyer represents and agrees that it has all the necessary expertise to create and implement safeguards which anticipate dangerous consequences of failures, monitor failures and their consequences, lessen the likelihood of failures that might cause harm and take appropriate remedial actions. Buyer will fully indemnify TI and its representatives against any damages arising out of the use of any TI components in safety-critical applications.

In some cases, TI components may be promoted specifically to facilitate safety-related applications. With such components, TI’s goal is to help enable customers to design and create their own end-product solutions that meet applicable functional safety standards and requirements. Nonetheless, such components are subject to these terms.

No TI components are authorized for use in FDA Class III (or similar life-critical medical equipment) unless authorized officers of the parties have executed a special agreement specifically governing such use.

Only those TI components which TI has specifically designated as military grade or “enhanced plastic” are designed and intended for use in military/aerospace applications or environments. Buyer acknowledges and agrees that any military or aerospace use of TI components which have ***not*** been so designated is solely at the Buyer's risk, and that Buyer is solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI has specifically designated certain components as meeting ISO/TS16949 requirements, mainly for automotive use. In any case of use of non-designated products, TI will not be responsible for any failure to meet ISO/TS16949.

Products		**Applications**	
Audio	www.ti.com/audio	Automotive and Transportation	www.ti.com/automotive
Amplifiers	amplifier.ti.com	Communications and Telecom	www.ti.com/communications
Data Converters	dataconverter.ti.com	Computers and Peripherals	www.ti.com/computers
DLP® Products	www.dlp.com	Consumer Electronics	www.ti.com/consumer-apps
DSP	dsp.ti.com	Energy and Lighting	www.ti.com/energy
Clocks and Timers	www.ti.com/clocks	Industrial	www.ti.com/industrial
Interface	interface.ti.com	Medical	www.ti.com/medical
Logic	logic.ti.com	Security	www.ti.com/security
Power Mgmt	power.ti.com	Space, Avionics and Defense	www.ti.com/space-avionics-defense
Microcontrollers	microcontroller.ti.com	Video and Imaging	www.ti.com/video
RFID	www.ti-rfid.com		
OMAP Applications Processors	www.ti.com/omap	**TI E2E Community**	e2e.ti.com
Wireless Connectivity	www.ti.com/wirelessconnectivity		

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265