Call progress tones monitor for TMS320C6201† Overview Guide

www.radisys.com

World Headquarters
5445 NE Dawson Creek Drive • Hillsboro, OR
97124 USA
Phone: 503-615-1100 • Fax: 503-615-1121
Toll-Free: 800-950-0044

International Headquarters
Gebouw Flevopoort • Televisieweg 1A
NL-1322 AC • Almere, The Netherlands
Phone: 31 36 5365595 • Fax: 31 36 5365620

007-01013-0002
November 2000

November 2000

Call progress monitor for TMS320C6201

The Call progress monitor for TMS320C6201† performs CPT detection for high capacity trunks. It detects only in-band audible tones; it does not address any tones which may be out-of band. You can use this Call progress monitor software in a multi-tasking environment.

The Call progress monitor for TMS320C6201 executes on blocks of 240 linear 16-bit PCM samples (30ms frames at 8000Hz sampling rate).

For information about installing and using the Call progress monitor software, see the topics listed below.

Installation

To make the Call progress monitor software fully functional, follow these steps:

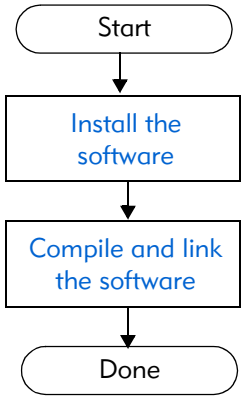

Figure 1. Installing and building the Call progress monitor software

Requirements

Before you can install this software, you need the following:

- A Microsoft† Windows† 95, Windows 98, or Windows NT† compatible PC platform.
- The following software installed on your computer:
  - TMS320C6x Code Generation Tools Release 3.01 for Windows 95, Windows 98, or Windows NT

    —or—

    Code Composer Studio Tools, version 1.2, for Windows 95, Windows 98, or Windows NT.
  - NMAKE utility (available from Microsoft).
  - PKUNZIP utility.

Install the software

To install the Call progress monitor software:

1. Create a directory on the target system. For example:

   ```
   mkdir c:\cpm
   ```

2. Copy the product files from the CD-ROM to the directory you just created.

Use a copy utility that retains the directory structure, such as Xcopy.

Compile and link the software

To compile and link the Call progress monitor software:

1. Open an MS DOS command prompt window and go to the target directory, then to the \c6xsim sub-directory.
2. If you've already installed the TI tools, you can build the C6x version of the Call progress monitor by entering this command at the MS DOS command prompt:

   ```
   NMAKE -f cpmbuild.mak
   ```

Code organization

When installed, the Call progress monitor software files reside in this directory structure:

Directory	Extension	File type	Description
\cpm	.pdf	Manual	Includes this guide, in PDF format.
\cpm\c6xsim		CPM example test program and working directory	
	.c	C files	Includes cpmc6c.c, the main test program.
	.c6x	Output file	The tonefile.c6x file includes test results.
	.cmd	Linker command files	The cpmc6x.cmd file controls object modules, linking options, and memory maps for the Call progress monitor module.
	.lib	Library files	The cpm.lib file includes object modules used to build DSP executables.
	.lin	Input files	The tonefile.lin file includes PCM test vectors.
	.mak	Build file	The cpmc6x.mak file[1] controls the build process for cpmc6x.out, the test program.
\cpm\include	.h	Header files	Header files used to build the Call progress monitor. Applications can share these files as well.

[1] This file is tested to work with the Microsoft NMAKE utility, and can be modified to work with other make utilities.

Application programmer interface

The CPT detector software is designed for use in a multi-tasking environment.

The functional prototypes for the CPT detector modules are as follows.

The API for the CPT detector module consists of the following functions. All the design is C callable and accepts buffers passed to the module (i.e. no stack variables, no direct access of globals).

CreateCPTdetect

Configures appropriate parameters for the detector and initializes variables.

Syntax

```
CreateCPTdetect(CPTdetectStateVar *pTdVar)
```

Parameters

pTdVar Pointer to the CPM state variable structure with sizeof (CPTdetStateVar), 32-bit alignment, persistent (input/output).

CPTdetMain

Syntax

```
CPTdetMain (
CPTdetStateVar *pTdVar,
CPTdetTbl *pTdCoeff,
Word16 *pInBuf,
Word16 *pScratch
)
```

Parameters

pTdVar Pointer to the CPM state variable structure with sizeof (CPTdetStateVar), 32-bit alignment, persistent (input/output).

pTdCoeff Pointer to the CPM state table structure with sizeof (CPTdetTbl), 32-bit alignment, shared (input).

pInBuf Pointer to the 16-bit PCM input samples with size of 240 bytes, 32-bit alignment, non-persistent (input).

pScratch Pointer to the scratch memory with size of 2400 bytes, 32-bit alignment, non-persistent (input).

Run-time Initializations

These structure elements are initialized one time at start of function, after calling CreateCPTdetect:

pTdVar->keyState
32-bit structure variable to enable up to 32 CPM tone detection's. Bit 0 is used for 1st tone, Bit 1 is used for 2nd tone, …Bit 31 is used for 32nd tone. By default all the bits in the *pTdVar->keyState* (0xffffffff) are enabled.

pTdVar->ToneNum0a,0b,…7a,7b
16-bit structure variables defined for each tone bin number (tone frequency/31.25 & round it to nearest integer) of the single tone. CPM module supports up to 8 combined dual tones. For default, refer to include\cpmdef.h

pTdVar->ToneNum8,9,…31

16-bit structure variables defined for each tone bin number (tone frequency/31.25 & round it to nearest integer) of the single tone. CPM module supports up to 24 single tones. For default, refer to include\cpmdef.h

pTdVar->FrDeLow[32]
16-bit structure for 32 variables array defined for allowable (low) frequency deviation for single/combined tones (((sampling frequency/16)/ (absolute difference of two frequencies) - 2). For default, refer to include\cpmdef.h

pTdVar->FrDeHigh[32]
16-bit structure for 32 variables array defined for allowable (high) frequency deviation for single/combined tones (((sampling frequency/16)/ (absolute difference of two frequencies) - 2). For default, refer to include\cpmdef.h

pTdVar->MaxOnTime[32]
16-bit structure for 32 variables array defined as a count of maximum ON time (mark) of a single/combined tones (Max. ON time in ms/10ms). For default, refer to include\cpmdef.h

pTdVar->MinOnTime[32]
16-bit structure for 32 variables array defined as a count of minimum ON time (mark) of a single/combined tones (Min. ON time in ms/10ms). For default, refer to include\cpmdef.h

pTdVar->MaxOffTime[32]
16-bit structure for 32 variables array defined as a count of maximum OFF time (space) of a single/combined tones (Max. OFF time in ms/10ms). For default, refer to include\cpmdef.h

pTdVar->MinOffTime[32]
16-bit structure for 32 variables array defined as a count of minimum OFF time (space) of a single/combined tones (Min. OFF time in ms/10ms). For default, refer to include\cpmdef.h

I/O parameters for every 30ms frame

pInBuf Short Pointer to the TDM Input buffer (16-bit PCM samples) size of 480 bytes, 32 bit alignment

pTdVar->keyState
32-bit structure variable to enable up to 32 CPM tone detection's. Bit 0 is used for 1st tone, Bit 1 is used for 2nd tone, …Bit 31 is used for 32nd tone. By default all the bits in the *pTdVar->keyState* (0xffffffff) are enabled. Application has an option to set a specific CPM tones by making the corresponding bit fields '1' in *pTdVar->keyState*. Then the CPM state machine looks for the application specified tones for detection.

pTdVar->key
32-bit structure variable to store the CPM detected output. Detected output consists of key number as specified in the cpmdef.h. Also refer to *Testing and performance specifications* for more details.

pTdVar->ToneDuration

This structure variable used detect the CPM tone duration. *pTdVar->ToneDuration* is a Short (Word16) variable defined as a count of 10ms intervals. For actual time duration multiply this count with 10ms. For examples if *pTdVar->ToneDuration* is 5, then the actual tone duration is 50ms. This duration available to the application as soon as *pTdVar->ToneStatus* becomes TRUE.

Testing and performance specifications

Call progress monitor tests on the c6x evaluation board

The c6x call progress monitor implementation must satisfy all the TIA/EIA-464-B specification. These tests are performed using the c6x evaluation board and standard file I/O. Results should match the results of the RadiSysfixed-point C simulation.

Call progress monitor initialization

CPM uses the following constants (cpmdef.h) which specifies frequencies, cadences, and tolerance for detection. User has an option to set a specific CPM tones by making the corresponding bit fields '1' in the structure variable (*pVar->keyState*). Then CPM state machine looks for the user specified tones for detection.

Configuring CPM

1. CPM state machine optimized to make it simple by restricting it to a lower level. That means it just has a capability to detect & continue to detect the tone with specified cadences. For example it has a capability to detect 500ms phase reversals in case of data modem tone. Its up to the higher level state machine (user wrapper code) to decide how many phase reversals one should look far before declaring it as a data modem tone.
2. User has an option to select 8 possible cadences for detection from a maximum 3 possible combined tones at each time.
3. TONE_KEY1,TONE_KEY2,TONE_KEY3 are subset of TONE_KEY0. i.e., the frequencies are same as TONE_KEY0, but different cadences. User has an option to only change cadences on the others.
4. TONE_KEY5 is a subset of TONE_KEY4. i.e., the frequencies are same as TONE_KEY4, but different cadences. User has an option to only change cadence between the two.
5. Similarly TONE_KEY7 is a subset of TONE_KEY6. i.e., the frequencies are same as TONE_KEY6, but different cadences. User has an option to only change cadence between the two.
6. User has an option to select 21 possible cadences for detection from a maximum of 18 possible single tones at each time.

7. TONE_KEY9,TONE_KEY10,TONE_KEY11 are subset of TONE_KEY8. i.e., same frequency as TONE_KEY8, but different cadences. User has an option to only change cadences on others.
8. TONE_KEY18 dedicated to detect 2100Hz fax tone with specified cadences.
9. TONE_KEY19 dedicated to detect 2100Hz data modem tone phase reversals with specified cadences.
10. Others are user programmable for single tone detects with specified cadences.

General CP tone representations for detection

Tone	Description
TONE_KEYxx	Tone number corresponding to bit field in structure variable: `(pVar->keyState)`
TONE_COMxA	First tone bin number: `(tone frequency/31.25 rounded to nearest integer) of the combined tone`
TONE_COMxB	Second tone bin number: `(tone frequency/31.25 rounded to nearest integer) of the combined tone`
COMx_FD_LO	Allowable (low) frequency deviation for combined tones: `(((sampling frequency/16)/(absolute difference of two frequencies) - 2)`
COMx_FD_HI	Allowable (high) frequency deviation for combined tones: `(((sampling frequency/16)/(absolute difference of two frequencies) + 2)`
COMx_MARK_MAX	Maximum mark time for combined tones: `(mark time+tolerance/10)`
COMx_MARK_MIN	Minimum mark time for combined tones: `(mark time-tolerance/10)`
COMx_SPACE_MAX	Maximum space time for combined tones: `(space time+tolerance/10)`
COMx_SPACE_MIN	Minimum space time for combined tones: `(space time-tolerance/10)`
TONE_SINxx	Tone bin number: `(tone frequency/31.25 rounded to nearest integer) of the combined tone`
SINxx_FD_LO	Allowable (low) frequency deviation for single tones: `(TONE_SINxx * 8 - 2)`
SINxx_FD_HI	Allowable (high) frequency deviation for single tones: `(TONE_SINxx * 8 + 2)`
SINx_MARK_MAX	Maximum mark time for single tones: `(mark time+tolerance/10)`
SINx_MARK_MIN	Minimum mark time for single tones: `(mark time-tolerance/10)`
SINx_SPACE_MAX	Maximum space time for single tones: `(space time+tolerance/10`

Tone	Description
SINx_SPACE_MIN	Minimum space time for single tones: (space time-tolerance/10)

Definitions for 8 possible combined tones

```
/* 0th combination (350+440Hz dial tone) and key state position is BIT0 */

#define TONE_KEY0        0x1   /* KEY NUMBER 0*/
#define TONE_COM0A       11    /* bin number for 350Hz */
#define TONE_COM0B       14    /* bin number for 440Hz */
#define COM0_FD_LO       5     /* <= Must accept thrs: 500/(440-350)-2 */
#define COM0_FD_HI       7     /* >= Must reject thrs: 500/(440-350)+2 */
#define COM0_MARK_MAX    32767 /* Max mark time by 10ms */
#define COM0_MARK_MIN    18    /* min mark time by 10ms */
#define COM0_SPACE_MAX   0     /* Max space time by 10ms */
#define COM0_SPACE_MIN   0     /* min space time by 10ms */

/* 1st combination (350+440Hz Recall Dial tone) and key state position is
BIT1 */

#define TONE_KEY1        0x2   /* KEY NUMBER 1*/
#define TONE_COM1A       11      /* bin number for 350Hz */
#define TONE_COM1B       14    /* bin number for 440Hz */
#define COM1_FD_LO       5     /* <= Must accept thrs: 500/(440-350)-2 */
#define COM1_FD_HI       7     /* >= Must reject thrs: 500/(440-350)+2 */
#define COM1_MARK_MAX    12    /* Max mark time by 10ms */
#define COM1_MARK_MIN    8     /* min mark time by 10ms */
#define COM1_SPACE_MAX   12    /* Max space time by 10ms */
#define COM1_SPACE_MIN   8     /* min space time by 10ms */

/* 2nd combination (350+440Hz Conformation tone) and key state position is
BIT2 */

#define TONE_KEY2        0x4   /* KEY NUMBER 2*/
#define TONE_COM2A       11    /* bin number for 350Hz */
#define TONE_COM2B       14    /* bin number for 440Hz */
#define COM2_FD_LO       5     /* <= Must accept thrs: 500/(440-350)-2 */
#define COM2_FD_HI       7     /* >= Must reject thrs: 500/(440-350)+2 */
#define COM2_MARK_MAX    120   /* Max mark time by 10ms */
#define COM2_MARK_MIN    80    /* min mark time by 10ms */
#define COM2_SPACE_MAX   12    /* Max space time by 10ms */
#define COM2_SPACE_MIN   8     /* min space time by 10ms */

/* 3rd combination (350+440Hz Stutter Dial tone) and key state position is
BIT3 */

#define TONE_KEY3        0x8   /* KEY NUMBER 3*/
#define TONE_COM3A       11    /* bin number for 350Hz */
#define TONE_COM3B       14    /* bin number for 440Hz */
#define COM3_FD_LO       5     /* <= Must accept thrs: 500/(440-350)-2 */
#define COM3_FD_HI       7     /* >= Must reject thrs: 500/(440-350)+2 */
#define COM3_MARK_MAX    127   /* Max mark time by 10ms */
#define COM3_MARK_MIN    122   /* min mark time by 10ms */
#define COM3_SPACE_MAX   27    /* Max space time by 10ms */
#define COM3_SPACE_MIN   22    /* min space time by 10ms */

/* 4th combination (480+620Hz Busy tone) and key state position is BIT4 */

#define TONE_KEY4        0x10  /* KEY NUMBER 4*/
#define TONE_COM4A       15    /* bin number for 480Hz */
#define TONE_COM4B       20    /* bin number for 620Hz */
```

```
#define COM4_FD_LO      3    /* <= Must accept thrs: 500/(620-480)-2 */
#define COM4_FD_HI      5    /* >= Must reject thrs: 500/(620-480)+2 */
#define COM4_MARK_MAX   55   /* Max mark time by 10ms */
#define COM4_MARK_MIN   45   /* min mark time by 10ms */
#define COM4_SPACE_MAX  55   /* Max space time by 10ms */
#define COM4_SPACE_MIN  45   /* min space time by 10ms */

/* 5th combination (480+620 Reorder tone) and key state position is BIT5
*/

#define TONE_KEY5       0x20 /* KEY NUMBER 5*/
#define TONE_COM5A      15   /* bin number for 480Hz */
#define TONE_COM5B      20   /* bin number for 620Hz */
#define COM5_FD_LO      3    /* <= Must accept thrs: 500/(620-480)-2 */
#define COM5_FD_HI      5    /* >= Must reject thrs: 500/(620-480)+2 */
#define COM5_MARK_MAX   27   /* Max mark time by 10ms */
#define COM5_MARK_MIN   22   /* min mark time by 10ms */
#define COM5_SPACE_MAX  27    /* Max space time by 10ms */
#define COM5_SPACE_MIN  22   /* min space time by 10ms */

/* 6th combination (440+480 Audible ring tone) and key state position is
BIT6 */

#define TONE_KEY6       0x40 /* KEY NUMBER 6*/
#define TONE_COM6A      14   /* bin number for 440Hz */
#define TONE_COM6B      15   /* bin number for 480Hz */
#define COM6_FD_LO      8    /* <= Must accept thrs: 500/(480-440)-2 */
#define COM6_FD_HI      18   /* >= Must reject thrs: 500/(480-440)+2 */
#define COM6_MARK_MAX   120  /* Max mark time by 10ms */
#define COM6_MARK_MIN   80   /* min mark time by 10ms */
#define COM6_SPACE_MAX  330  /* Max space time by 10ms */
#define COM6_SPACE_MIN  270  /* min space time by 10ms */
#define COM6A_MARK_MAX  220  /* Max mark time by 10ms */
#define COM6A_MARK_MIN  180  /* min mark time by 10ms */
#define COM6A_SPACE_MAX 440  /* Max space time by 10ms */
#define COM6A_SPACE_MIN 360  /* min space time by 10ms */

/* 7th combination (440+480Hz Special Audible ring tone) and key state
position is BIT7 */

#define TONE_KEY7       0x80 /* KEY NUMBER 7*/
#define TONE_COM7A      14   /* bin number for 440Hz */
#define TONE_COM7B      15   /* bin number for 480Hz */
#define COM7_FD_LO      8    /* <= Must accept thrs: 500/(480-440)-2 */
#define COM7_FD_HI      18   /* >= Must reject thrs: 500/(480-440)+2 */
#define COM7_MARK_MAX   32767/* Max mark time by 10ms */
#define COM7_MARK_MIN   80   /* min mark time by 10ms */
#define COM7_SPACE_MAX  0    /* Max space time by 10ms */
#define COM7_SPACE_MIN  0    /* min space time by 10ms */
```

Definitions for single tones

```
/* 1st single tone (440Hz Call Waiting tone) and  the key state position
is BIT8  */

#define TONE_KEY8       0x100/* KEY NUMBER 8*/
#define TONE_SIN8       14   /* bin number for 440Hz */
#define SIN8_FD_LO      108  /* <= Must accept thrs: TONE_SIN8*8-2 */
#define SIN8_FD_HI      116  /* >= Must reject thrs: TONE_SIN8*8+2 */
#define SIN8_MARK_MAX   30   /* Max mark time by 10ms */
#define SIN8_MARK_MIN   10   /* min mark time by 10ms */
#define SIN8_SPACE_MAX  1004 /* Max space time by 10ms */
```

```
#define SIN8_SPACE_MIN   996   /* min space time by 10ms */

/* 2nd single tone (440Hz Busy Verification tone)  and  the key state
position is BIT9    */

#define TONE_KEY9        0x200/* KEY NUMBER 9*/
#define TONE_SIN9        14    /* bin number for 440Hz */
#define SIN9_FD_LO       108   /* <= Must accept thrs: TONE_SIN8*8-2 */
#define SIN9_FD_HI       116   /* >= Must reject thrs: TONE_SIN8*8+2 */
#define SIN9_MARK_MAX    200   /* Max mark time by 10ms */
#define SIN9_MARK_MIN    150   /* min mark time by 10ms */
#define SIN9_SPACE_MAX   0     /* Max space time by 10ms */
#define SIN9_SPACE_MIN   0     /* min space time by 10ms */

/* 3rd single tone (440Hz Executive Override tone) and  the key state
position is BIT10     */

#define TONE_KEY10       0x400/* KEY NUMBER 10*/
#define TONE_SIN10       14    /* bin number for 440Hz */
#define SIN10_FD_LO      108   /* <= Must accept thrs: TONE_SIN8*8-2 */
#define SIN10_FD_HI      116   /* >= Must reject thrs: TONE_SIN8*8+2 */
#define SIN10_MARK_MAX   400   /* Max mark time by 10ms */
#define SIN10_MARK_MIN   200   /* min mark time by 10ms */
#define SIN10_SPACE_MAX  0     /* Max space time by 10ms */
#define SIN10_SPACE_MIN  0     /* min space time by 10ms */

/* 4th single tone (440Hz Intercept tone) and  the key state position is
BIT11       */

#define TONE_KEY11       0x800/* KEY NUMBER 11*/
#define TONE_SIN11       14    /* bin number for 440Hz */
#define SIN11_FD_LO      108   /* <= Must accept thrs: TONE_SIN8*8-2 */
#define SIN11_FD_HI      116   /* >= Must reject thrs: TONE_SIN8*8+2 */
#define SIN11_MARK_MAX   30    /* Max mark time by 10ms */
#define SIN11_MARK_MIN   16    /* min mark time by 10ms */
#define SIN11_SPACE_MAX  0     /* Max space time by 10ms */
#define SIN11_SPACE_MIN  0     /* min space time by 10ms */

/* 5th single tone (620Hz Intercept tone) and  the key state position is
BIT12         */

#define TONE_KEY12       0x1000/* KEY NUMBER 12*/
#define TONE_SIN12       19    /* bin number for 620Hz */
#define SIN12_FD_LO      155   /* <= Must accept thrs: TONE_SIN8*8-2 */
#define SIN12_FD_HI      159   /* >= Must reject thrs: TONE_SIN8*8+2 */
#define SIN12_MARK_MAX   30    /* Max mark time by 10ms */
#define SIN12_MARK_MIN   16    /* min mark time by 10ms */
#define SIN12_SPACE_MAX  0     /* Max space time by 10ms */
#define SIN12_SPACE_MIN  0     /* min space time by 10ms */

/* 6th single tone (1400Hz Off Hook) and  the key state position is BIT13
*/

#define TONE_KEY13       0x2000/* KEY NUMBER 13*/
#define TONE_SIN13       44    /* bin number for 1400Hz */
#define SIN13_FD_LO      356   /* <= Must accept thrs: TONE_SIN8*8-2 */
#define SIN13_FD_HI      364   /* >= Must reject thrs: TONE_SIN8*8+2 */
#define SIN13_MARK_MAX   12    /* Max mark time by 10ms */
#define SIN13_MARK_MIN   8     /* min mark time by 10ms */
#define SIN13_SPACE_MAX  12    /* Max space time by 10ms */
#define SIN13_SPACE_MIN  8     /* min space time by 10ms */

/* 7th single tone (2060Hz Off Hook) and  the key state position is BITk14
*/
```

```
#define TONE_KEY14       0x4000/* KEY NUMBER 14*/
#define TONE_SIN14       66    /* bin number for 2060Hz */
#define SIN14_FD_LO      524   /* <= Must accept thrs: TONE_SIN8*8-2 */
#define SIN14_FD_HI      531   /* >= Must reject thrs: TONE_SIN8*8+2 */
#define SIN14_MARK_MAX   12    /* Max mark time by 10ms */
#define SIN14_MARK_MIN   8     /* min mark time by 10ms */
#define SIN14_SPACE_MAX  12    /* Max space time by 10ms */
#define SIN14_SPACE_MIN  8     /* min space time by 10ms */

/* 8th single tone  (2450Hz Off Hook) and  the key state position is BIT15
*/

#define TONE_KEY15       0x8000/* KEY NUMBER 15*/
#define TONE_SIN15       78    /* bin number for 2450Hz */
#define SIN15_FD_LO      625   /* <= Must accept thrs: TONE_SIN8*8-2 */
#define SIN15_FD_HI      629   /* >= Must reject thrs: TONE_SIN8*8+2 */
#define SIN15_MARK_MAX   12    /* Max mark time by 10ms */
#define SIN15_MARK_MIN   8     /* min mark time by 10ms */
#define SIN15_SPACE_MAX  12    /* Max space time by 10ms */
#define SIN15_SPACE_MIN  8     /* min space time by 10ms */

/* 9th single tone  (2600Hz Off Hook) and  the key state position is BIT16
*/

#define TONE_KEY16       0x10000/* KEY NUMBER 16*/
#define TONE_SIN16       83    /* bin number for 2600Hz */
#define SIN16_FD_LO      660   /* <= Must accept thrs: TONE_SIN8*8-2 */
#define SIN16_FD_HI      668   /* >= Must reject thrs: TONE_SIN8*8+2 */
#define SIN16_MARK_MAX   12    /* Max mark time by 10ms */
#define SIN16_MARK_MIN   8     /* min mark time by 10ms */
#define SIN16_SPACE_MAX  12    /* Max space time by 10ms */
#define SIN16_SPACE_MIN  8     /* min space time by 10ms */

/* 10th single tone (1100Hz fax TX tone) and  the key state position is
BIT17       */

#define TONE_KEY17       0x20000/* KEY NUMBER 17*/
#define TONE_SIN17       35    /* bin number for 1100Hz */
#define SIN17_FD_LO      276   /* <= Must accept thrs: TONE_SIN8*8-2 */
#define SIN17_FD_HI      284   /* >= Must reject thrs: TONE_SIN8*8+2 */
#define SIN17_MARK_MAX   32767 /* Max mark time by 10ms */
#define SIN17_MARK_MIN   28    /* min mark time by 10ms */
#define SIN17_SPACE_MAX  0     /* Max space time by 10ms */
#define SIN17_SPACE_MIN  0     /* min space time by 10ms */

/* 11th single tone (2100Hz fax rx tone) and  the key state position is
BIT18     */

#define TONE_KEY18       0x40000/* KEY NUMBER 18*/
#define TONE_SIN18       67    /* bin number for 2100Hz */
#define SIN18_FD_LO      532   /* <= Must accept thrs: TONE_SIN8*8-2 */
#define SIN18_FD_HI      540   /* >= Must reject thrs: TONE_SIN8*8+2 */
#define SIN18_MARK_MAX   32767 /* Max mark time by 10ms */
#define SIN18_MARK_MIN   18    /* min mark time by 10ms */
#define SIN18_SPACE_MAX  0     /* Max space time by 10ms */
#define SIN18_SPACE_MIN  0     /* min space time by 10ms */

/* 12th single tone (2100Hz data modem tone) and  the key state position
is BIT19    */

#define TONE_KEY19       0x80000/* KEY NUMBER 19*/
#define TONE_SIN19       67    /* bin number for 2100Hz */
#define SIN19_FD_LO      532   /* <= Must accept thrs: TONE_SIN8*8-2 */
```

```
#define SIN19_FD_HI      540   /* >= Must reject thrs: TONE_SIN8*8+2 */
#define SIN19_MARK_MAX   32767 /* Max mark time by 10ms */
#define SIN19_MARK_MIN   48    /* min mark time by 10ms */
#define SIN19_SPACE_MAX  0     /* Max space time by 10ms */
#define SIN19_SPACE_MIN  0     /* min space time by 10ms */

/* 13th single tone (1004Hz line test tone) and  the key state position is
BIT20      */

#define TONE_KEY20       0x100000/* KEY NUMBER 20*/
#define TONE_SIN20       32    /* bin number for 1004Hz */
#define SIN20_FD_LO      252   /* <= Must accept thrs: TONE_SIN8*8-2 */
#define SIN20_FD_HI      260   /* >= Must reject thrs: TONE_SIN8*8+2 */
#define SIN20_MARK_MAX   32767 /* Max mark time by 10ms */
#define SIN20_MARK_MIN   20    /* min mark time by 10ms */
#define SIN20_SPACE_MAX  0     /* Max space time by 10ms */
#define SIN20_SPACE_MIN  0     /* min space time by 10ms */

/* 14th single tone (2010Hz SS7 tone) and  the key state position is BIT21
*/

#define TONE_KEY21       0x200000/* KEY NUMBER 21*/
#define TONE_SIN21       64    /* bin number for 2010Hz */
#define SIN21_FD_LO      512   /* <= Must accept thrs: TONE_SIN8*8-2 */
#define SIN21_FD_HI      516   /* >= Must reject thrs: TONE_SIN8*8+2 */
#define SIN21_MARK_MAX   32767 /* Max mark time by 10ms */
#define SIN21_MARK_MIN   20    /* min mark time by 10ms */
#define SIN21_SPACE_MAX  0     /* Max space time by 10ms */
#define SIN21_SPACE_MIN  0     /* min space time by 10ms */

/* 15th single tone  and  the key state position is BIT22    */

#define TONE_KEY22       0x400000/* KEY NUMBER 22*/
#define TONE_SIN22       29    /* bin number for 913.8Hz */
#define SIN22_FD_LO      230   /* <= Must accept thrs: TONE_SIN22*8-2 */
#define SIN22_FD_HI      234   /* >= Must reject thrs: TONE_SIN22*8+2 */
#define SIN22_MARK_MAX   26    /* max mark time by 10ms */
#define SIN22_MARK_MIN   24    /* min mark time by 10ms */
#define SIN22_SPACE_MAX  0     /* max space time by 10ms */
#define SIN22_SPACE_MIN  0     /* min space time by 10ms */

/* 16th single tone  and  the key state position is BIT23    */

#define TONE_KEY23       0x800000/* KEY NUMBER 23*/
#define TONE_SIN23       32    /* bin number for 985.2Hz */
#define SIN23_FD_LO      254   /* <= Must accept thrs: TONE_SIN23*8-2 */
#define SIN23_FD_HI      258   /* >= Must reject thrs: TONE_SIN23*8+2 */
#define SIN23_MARK_MAX   26    /* max mark time by 10ms */
#define SIN23_MARK_MIN   24    /* min mark time by 10ms */
#define SIN23_SPACE_MAX  0     /* max space time by 10ms */
#define SIN23_SPACE_MIN  0     /* min space time by 10ms */

/* 17th single tone  and  the key state position is BIT24    */

#define TONE_KEY24       0x1000000/* KEY NUMBER 24*/
#define TONE_SIN24       44    /* bin number for 1370.6Hz */
#define SIN24_FD_LO      349   /* <= Must accept thrs: TONE_SIN24*8-2 */
#define SIN24_FD_HI      353   /* >= Must reject thrs: TONE_SIN24*8+2 */
#define SIN24_MARK_MAX   26    /* max mark time by 10ms */
#define SIN24_MARK_MIN   24    /* min mark time by 10ms */
#define SIN24_SPACE_MAX  0     /* max space time by 10ms */
#define SIN24_SPACE_MIN  0     /* min space time by 10ms */

/* 18th single tone  and  the key state position is BIT25    */
```

```
#define TONE_KEY25       0x2000000/* KEY NUMBER 25*/
#define TONE_SIN25       46    /* bin number for 1428.5Hz */
#define SIN25_FD_LO      364   /* <= Must accept thrs: TONE_SIN25*8-2 */
#define SIN25_FD_HI      368   /* >= Must reject thrs: TONE_SIN25*8+2 */
#define SIN25_MARK_MAX   26    /* max mark time by 10ms */
#define SIN25_MARK_MIN   24    /* min mark time by 10ms */
#define SIN25_SPACE_MAX  0     /* max space time by 10ms */
#define SIN25_SPACE_MIN  0     /* min space time by 10ms */

/* 19th single tone  and  the key state position is BIT26    */

#define TONE_KEY26       0x4000000/* KEY NUMBER 26*/
#define TONE_SIN26       57    /* bin number for 1776.7Hz */
#define SIN26_FD_LO      453   /* <= Must accept thrs: TONE_SIN26*8-2 */
#define SIN26_FD_HI      457   /* >= Must reject thrs: TONE_SIN26*8+2 */
#define SIN26_MARK_MAX   26    /* max mark time by 10ms */
#define SIN26_MARK_MIN   24    /* min mark time by 10ms */
#define SIN26_SPACE_MAX  0     /* max space time by 10ms */
#define SIN26_SPACE_MIN  0     /* min space time by 10ms */

/* 20th single tone  and  the key state position is BIT27    */

#define TONE_KEY27       0x8000000/* KEY NUMBER 27*/
#define TONE_SIN27       64    /* bin number for xxxxHz */
#define SIN27_FD_LO      512   /* <= Must accept thrs: TONE_SIN27*8-2 */
#define SIN27_FD_HI      516   /* >= Must reject thrs: TONE_SIN27*8+2 */
#define SIN27_MARK_MAX   20    /* max mark time by 10ms */
#define SIN27_MARK_MIN   18    /* min mark time by 10ms */
#define SIN27_SPACE_MAX  0     /* max space time by 10ms */
#define SIN27_SPACE_MIN  0     /* min space time by 10ms */

/* 21th single tone  and  the key state position is BIT28    */

#define TONE_KEY28       0x10000000/* KEY NUMBER 28*/
#define TONE_SIN28       64    /* bin number for xxxxHz */
#define SIN28_FD_LO      512   /* <= Must accept thrs: TONE_SIN28*8-2 */
#define SIN28_FD_HI      516   /* >= Must reject thrs: TONE_SIN28*8+2 */
#define SIN28_MARK_MAX   20    /* max mark time by 10ms */
#define SIN28_MARK_MIN   18    /* min mark time by 10ms */
#define SIN28_SPACE_MAX  0     /* max space time by 10ms */
#define SIN28_SPACE_MIN  0     /* min space time by 10ms */

/* 22nd single tone and the key state position is BIT29    */

#define TONE_KEY29            0x20000000/* KEY NUMBER 29*/
#define TONE_SIN29       64    /* bin number for xxxxHz */
#define SIN29_FD_LO      512   /* <= Must accept thrs: TONE_SIN29*8-2 */
#define SIN29_FD_HI      516   /* >= Must reject thrs: TONE_SIN29*8+2 */
#define SIN29_MARK_MAX   20    /* max mark time by 10ms */
#define SIN29_MARK_MIN   18    /* min mark time by 10ms */
#define SIN29_SPACE_MAX  0     /* max space time by 10ms */
#define SIN29_SPACE_MIN  0     /* min space time by 10ms */

/* 23rd single tone and the key state position is BIT30    */

#define TONE_KEY30       0x40000000/* KEY NUMBER 30*/
#define TONE_SIN30       64    /* bin number for xxxxHz */
#define SIN30_FD_LO      512   /* <= Must accept thrs: TONE_SIN30*8-2 */
#define SIN30_FD_HI      516   /* >= Must reject thrs: TONE_SIN30*8+2 */
#define SIN30_MARK_MAX   20    /* max mark time by 10ms */
#define SIN30_MARK_MIN   18    /* min mark time by 10ms */
#define SIN30_SPACE_MAX  0     /* max space time by 10ms */
#define SIN30_SPACE_MIN  0     /* min space time by 10ms */
```

```
/* 24th single tone  and  the key state position is BIT31    */

#define TONE_KEY31       0x80000000/* KEY NUMBER 31*/
#define TONE_SIN31       64    /* bin number for xxxxHz */
#define SIN31_FD_LO      512   /* <= Must accept thrs: TONE_SIN31*8-2 */
#define SIN31_FD_HI      516   /* >= Must reject thrs: TONE_SIN31*8+2 */
#define SIN31_MARK_MAX   20    /* max mark time by 10ms */
#define SIN31_MARK_MIN   18    /* min mark time by 10ms */
#define SIN31_SPACE_MAX  0     /* max space time by 10ms */
#define SIN31_SPACE_MIN  0     /* min space time by 10ms */
```

TMS320C6X implementation test results

Test results on call progress monitor

All the tests have been performed on call progress monitor through the standard file IO using both the fixed-point C simulation as well as c6x emulator. It was confirmed that the c6x implementation was a bit-exact match of the fixed-point C simulation. The results of these tests are presented in what follows. All the files input PCM files are available upon request.

Test	Frequency (Hz)	Freq. Dev (%)	SNR (dB)	Twist (dB)	SigLev (dBm)	Input file	Detect (Y/N)
Dial Tone	350+440	0,0	60	0	-6	350_440dtone.pcm	YES
		0,+0.5				350_440dtone_fd1.pcm	YES
		0,-0.5				350_440dtone_fd2.pcm	YES
		+0.5,0				350_440dtone_fd3.pcm	YES
		-0.5,0,				350_440dtone_fd4.pcm	YES
		-0.5,-0.5				350_440dtone_fd5.pcm	YES
		+0.5,+0.5				350_440dtone_fd6.pcm	YES
	350+440						
		0,0	15			350_440dtone_snr.pcm	YES
			60		-30	350_440dtone_dnr.pcm	YES
			60	3		350_440dtone_tw1.pcm	YES
				-3		350_440dtone_tw2.pcm	YES
Recall Dial Tone	350+440	0,0	60	0	-6	350_440rtone.pcm	YES
Confirmation Tone	350+440	0,0	60	0	-6	350_440ctone.pcm	YES
Stutter Tone	350+440	0,0	60	0	-6	350_440stone.pcm	YES

Test	Frequency (Hz)	Freq. Dev (%)	SNR (dB)	Twist (dB)	SigLev (dBm)	Input file	Detect (Y/N)
Busy Tone	480+620	0,0	60	0	-6	480_620btone.pcm	YES
		0,+0.5				480_620btone_fd1.pcm	YES
		0,-0.5				480_620btone_fd2.pcm	YES
		+0.5,0				480_620btone_fd3.pcm	YES
		-0.5,0,				480_620btone_fd4.pcm	YES
		-0.5,-0.5				480_620btone_fd5.pcm	YES
		+0.5,+0.5				480_620btone_fd6.pcm	YES
		0,0	15			480_620btone_snr.pcm	YES
			60		-30	480_620btone_dnr.pcm	YES
			60	3		480_620btone_tw1.pcm	YES
				-3		480_620btone_tw2.pcm	YES
Reorder Tone	480+620	0,0	60	0	-6	480_620rtone.pcm	YES
Audible Ring Tone	440+480	0,0	60	0	-6	440_480atone.pcm	YES
Special Audible Ring Tone	440+480	0,0	60	0	-6	440_480sptone.pcm	YES
		0,+0.5				440_480sptone_fd1.pcm	YES
		0,-0.5				440_480sptone_fd2.pcm	YES
		+0.5,0				440_480sptone_fd3.pcm	YES
		-0.5,0,				440_480sptone_fd4.pcm	YES
		-0.5,-0.5				440_480sptone_fd5.pcm	YES
		+0.5,+0.5				440_480sptone_fd6.pcm	YES
		0,0	15			440_480sptone_snr.pcm	YES
			60		-30	440_480sptone_dnr.pcm	YES
			60	3		440_480sptone_tw1.pcm	YES
				-3		440_480sptone_tw2.pcm	YES

Test	Frequency (Hz)	Freq. Dev (%)	SNR (dB)	Twist (dB)	SigLev (dBm)	Input file	Detect (Y/N)
Intercept Tone	440	0	60	N/A	-6	440itone.pcm	YES
		±0.5				440itone_fd.pcm	YES
			15			440itone_snr.pcm	YES
					-30	440itone_dnr.pcm	YES
Call Waiting Tone	440	0	60	N/A	-6	440ctone.pcm	YES
Busy Verification Tone	440	0	60	N/A	-6	440btone.pcm	YES
Executive Override Tone	440	0	60	N/A	-6	440etone.pcm	YES
Intercept Tone	620	0	60	N/A	-6	620itone.pcm	YES
		±0.5				620itone_fd.pcm	YES
			15			620itone_snr.pcm	YES
					-30	620itone_dnr.pcm	YES
Tone during Off Hook	1400	0	60	N/A	-6	1400tone.pcm	YES
		±0.5				1400tone_fd.pcm	YES
			15			1400tone_snr.pcm	YES
					-30	1400tone_dnr.pcm	YES
Tone during Off Hook	2060	0	60	N/A	-6	2060tone.pcm	YES
		±0.5				2060tone_fd.pcm	YES
			15			2060tone_snr.pcm	YES
					-30	2060tone_dnr.pcm	YES

Test	Frequency (Hz)	Freq. Dev (%)	SNR (dB)	Twist (dB)	SigLev (dBm)	Input file	Detect (Y/N)
Tone during Off Hook	2450	0	60	N/A	-6	2450tone.pcm	YES
		±0.5				2450tone_fd.pcm	YES
			15			2450tone_snr.pcm	YES
					-30	2450tone_dnr.pcm	YES
Tone during Off Hook	2600	0	60	N/A	-6	2600tone.pcm	YES
		±0.5				2600tone_fd.pcm	YES
			15			2600tone_snr.pcm	YES
					-30	2600tone_dnr.pcm	YES
Data Modem Tone	2100 with phase shift of 180°	0	60	N/A	-6	2100dtone.pcm	YES
		-15 Hz				2087dtone.pcm	YES
		+15 Hz				2112dtone.pcm	YES
			15			2100dtone_snr.pcm	YES
	0°				-30	2100dtone_dnr.pcm	YES
	+110°	0	60		-6	2100tone.pcm	YES
	-110°					2100d110.pcm	(no detect)
	+155°					2100d_110.pcm	YES
	+205°					2100d155.pcm	(no detect)
						2100d205.pcm	YES
							YES

Test	Frequency (Hz)	Freq. Dev (%)	SNR (dB)	Twist (dB)	SigLev (dBm)	Input file	Detect (Y/N)
Fax Detect Rx tone	2100	0	60	N/A	-6	2100tone.pcm	YES
		±12.5 Hz				2100tone_fd.pcm	YES
			15			2100tone_snr.pcm	YES
					-30	2100tone_dnr.pcm	YES
Data Modem Tone	2100	0	60	N/A	-6	2100dtone.pcm	YES
		±15 Hz				2100dtone_fd.pcm	YES
			15			2100dtone_snr.pcm	YES
					-30	2100dtone_dnr.pcm	YES
Line test tone	1004	0	60	N/A	-6	1004tone.pcm	YES
		±0.5				1004tone_fd.pcm	YES
			15			1004tone_snr.pcm	YES
					-30	1004tone_dnr.pcm	YES
SS7 tone	2010	0	60	N/A	-6	2010tone.pcm	YES
		±0.5				2010tone_fd.pcm	YES
			15			2010tone_snr.pcm	YES
					-30	2010tone_dnr.pcm	YES

MCPS measurements for call progress monitor

The MCPS (million cycles per second) measurements for the c6x implementation of the CPM (with hand-optimization of most of the routines) are 0.30 MCPS per channel.

Memory measurements for call progress monitor

c6x implementation of the cpm needs 23.748 Kbytes of program memory, and 3.280 Kbytes of data memory. Out of 3.280 Kbytes of data memory, the variables structure is of 1.696 Kbytes which should be aligned to 32-bit word boundary and tables structure is of 1.584 Kbytes which needs 16 bit alignment. In addition to the above data memory, cpm also needs a scratch data memory of 2.528 Kbytes.

Where to get more information

About the Call progress monitor for TMS320C6201

You can find out more about the Call progress monitor from these sources:

- **Readme file:** Lists features and issues that arose too late to include in other documentation.
- **World Wide Web:** RadiSys maintains an active site on the World Wide Web. The site contains current information about the company and locations of sales offices, new and existing products, contacts for sales, service, and technical support information. You can also send e-mail to RadiSys using the web site.

When sending e-mail for technical support, please include information about both the hardware and software, plus a detailed description of the problem, including how to reproduce it.

To access the RadiSys web site, enter this URL in your web browser:
`http://www.radisys.com`

Requests for sales, service, and technical support information receive prompt response.

About related products

SP6040

The RadiSys SP6040 (SPIRIT™-6040 CompactPCI† board) is a high-performance intelligent I/O subsystem designed for telecom and datacom applications. Based on Texas Instruments† devices, the SP6040 has a 200MHz DSP engine with an I/O connector that provides interface-to-digital network interfaces (DNI) such as E1/T1 and ATM. The SP6040 contains up to four TMS320C6201 digital signal processors.

SP6020

The RadiSys SP6020 (SPIRIT-6020 board) features two Texas Instruments TMS320C6201B fixed-point DSPs that run at 200MHz and serve as the main processing engines. Each 'C6x can deliver up to 1600 MIPS of processing power, contingent on the available parallelism within the application code. The DSPs are used for voice and data processing, compression, and decompression.

TASK-6000

TASK-6000 software is a tool set that provides a framework for composing, executing and dynamically configuring optimized real-time TMS320C62x DSP applications. The Voice Gateway Platform (VGP), available from RadiSys, is used as the reference hardware platform.

For detailed information about TASK-6000 products, see the following publications:

TASK-6000 User's Manual, (07-0992-xx)

TASK-6000 Installation Guide, (07-1000-xx)

TI tools

Software tools from Texas Instruments used to build DSP executables.

For more information about Texas Instruments products, enter this URL in your web browser:
`http://www.ti.com/dsp`