DOCUMENTO 5

ANEXOS

Índice de los anexos

ANEXO 1
CÓDIGO DESARROLLADO

Índice del código desarrollado

Capítulo 1. Archivo fingerprint.h

```
/*************************************************************************
*
*  CREADOR: Juan Manuel Fernández Muñoz
*
*************************************************************************
*************************************************************************
*
*  NOMBRE: fingerprint.h
*
*
*  DESCRIPCIÓN:      Archivo de cabecera del algoritmo de reconocimiento de
*                    huellas dactilares basado en correlación de imágenes.
*
*************************************************************************/

#ifndef _FINGERPRINT_H
#define _FINGERPRINT_H

// Definición de constantes globales.
#define PI 3.14159265358979323846     /* Constante que contiene un valor
                                         aproximado de la constante matemática
                                         pi.*/
#define NF    128          /* Constante que indica el número de filas que
                              tendrán las matrices complejas.*/
#define NC    128          /* Constante que indica el número de columnas que
                              tendrán las matrices complejas.*/
#define NFi   200          /* Constante que indica el número de filas que
                              tendrán las imágenes.*/
#define NCi   152          /* Constante que indica el número de columnas que
                              tendrán las imágenes.*/
#define NUMUSER     5      /* Constante que indica el número de usuarios que
                              puede haber registrados en la base de datos.*/

// Nuevo tipo que define los datos de cada usuario.
typedef struct
{
   unsigned char imageuser[NFi][NCi];  // Imagen de la huella del usuario.
   int userenroll;                     /* Variable que indica si el número
                                          de usuario está registrado ya.*/
}user;

// Localización en memoria de las variables usadas.
#pragma DATA_SECTION (image, ".picture");
#pragma DATA_SECTION (imagecapture, ".picture");
#pragma DATA_SECTION (imagenroll, ".picture");
#pragma DATA_SECTION (database, ".picture");
#pragma DATA_SECTION (fftenroll, ".fft");
#pragma DATA_SECTION (fftcapture, ".fft");
#pragma DATA_SECTION (tw, ".fft");
#pragma DATA_SECTION (twi, ".fft");
#pragma DATA_SECTION (hist, ".info");
#pragma DATA_SECTION (simil, ".info");
#pragma DATA_SECTION (coord, ".info");
#pragma DATA_SECTION (phasepolar, ".info");

// Definición de variables usadas.
unsigned char image[NFi][NCi];              // Imagen recibida del sensor.
```

```
unsigned char imagecapture[NFi][NCi];       /* Copia de la imagen capturada
                                               con la que trabajaremos.*/
unsigned char imagenroll[NFi][NCi];   /* Copia de la imagen de la base de
                                         datos con la que trabajaremos.*/
user database[NUMUSER];               /* Definición de una base de datos de
                                         NUMUSER miembros.*/
float fftenroll[NF][2*NC];            /* Matriz compleja para la FFT 2D de
                                         la imagen de la base de datos.*/
float fftcapture[NF][2*NC];           // Matriz compleja para la FFT 2D de
                                         la imagen capturada.*/
float tw[NC];                         // Tabla twiddle para la FFT 2D.
float twi[NC];                        // Tabla twiddle para la IFFT 2D.
unsigned int hist[256];               /* Vector que almacena el histograma
                                         de la imagen.*/
float simil;                          /* Varible que contendrá el valor de
                                         comparación de las imágenes.*/
int coord[2];                         /* Vector para almacenar el
                                         desplazamiento entre imágenes.*/
float phasepolar[2*NF];    /* Tabla con valores de cosenos y senos
                              necesarios para la transformación a coordenadas
                              polares.*/

// Variables globales usadas por el fichero “botones.gel”.
unsigned int enrollFP=0, verifyFP=0, exitFP=0;

// Funciones necesarias para el control del fichero "botones.gel".
void dummyEnroll(void);
void dummyVerify(void);

// Funciones para tratar la imagen en el dominio espacial.
void gen_phase_table(float* , int);
void copy_image(unsigned char [][NCi], unsigned char [][NCi], int, int);
void histogram(unsigned char [][NCi], unsigned int*, int, int);
void preprocessing(unsigned char [][NCi], unsigned int*, int, int);
void complex_image(unsigned char [][NCi], float [][2*NC], int, int, int,
int);

void desp_image(unsigned char [][NCi], int*, int, int, int);
void trans_polar_image(unsigned char [][NCi], float [][2*NC], int*, float*,
int, int, int, int);

// Funciones para obtener la FFT 2D y la IFFT 2D.
void gen_twiddle(float*, int);
void copy_vector(float*, float*, int);
void complex_conj_vector(float*, int);
void bit_rev(float*, int);
void trasp_matrix(float [][2*NC], int, int);
#include "DSPF_sp_cfftr2_dit.h"
void fft2d(float [][2*NC], float*, float*, int, int);
void divide_ifft2d(float [][2*NC], int, int);

// Funciones para tratar la imagen en el dominio frecuencial.
void poc_function(float [][2*NC], float [][2*NC], int, int);
void poc(float [][2*NC], float [][2*NC], int, int);
float add_max_poc(float [][2*NC], int, int);
void high_peak_poc(float [][2*NC], int*, int, int);

#endif
```

Capítulo 2. Archivo fingerprint.c

```
/*************************************************************************
*
*  CREADOR: Juan Manuel Fernández Muñoz
*
*************************************************************************
*************************************************************************
*
*  NOMBRE: fingerprint.c
*
*
*  DESCRIPCIÓN:      Algoritmo de reconocimiento de huellas dactilares
*                    basado en correlación de imágenes.
*
*************************************************************************/

// Ficheros de cabecera.
#include <csl.h>
#include <csl_mcbsp.h>
#include <stdio.h>
#include <math.h>

#include "dsk6713.h"
#include "fpc1010.h"
#include "spi.h"
#include "fingerprint.h"

// Función principal (main).
void main(void)
{
  unsigned char     *imageptr;        // Puntero para imagen del sensor.
  int n;                                        // Número del usuario.

  // Inicialización de la placa.
  DSK6713_init();

  // Inicialización del módulo de leds.
    DSK6713_LED_init();

  // Iniciamos la funcionalidad de la librería CSL (Chip Support library).
  CSL_init();

  /* El McBSP0 es usado como interfaz SPI entre el DSP y el sensor FPC1010.
  Por defecto es usado para la comunicación entre el DSP y el codec AIC23.
  Habilitamos el McBSP0 para el sensor con la siguiente función.*/
  DSK6713_rset(DSK6713_MISC , MCBSP1SEL);

  // Inicializamos la comunicación SPI.
  SPI_init(2884615,0);

  imageptr = &image[0][0];      /* Puntero al comienzo de la matriz donde
                                almacenaremos la imagen del sensor.*/

  // Inicializando número de usuarios registrados a cero.
  for (n=0; n<NUMUSER; n++)
  {
        database[n].userenroll = 0;
  }
```

```
// Generación de tablas.
gen_twiddle(tw, NC);
copy_vector(tw, twi, NC);
complex_conj_vector(twi, NC>>1);
bit_rev(tw, NC>>1);
bit_rev(twi, NC>>1);
gen_phase_table(phasepolar , NF);

// Apagamos los leds del sensor escribiendo 1 en DC_CNTL1 y DC_CNTL0.
DSK6713_rset(DSK6713_DC_REG,3);

printf("Inicialización correcta.\n");

// Bucle principal.
waitForFp:

  while (!enrollFP && !verifyFP)
  {
       if(exitFP)
       {
             printf ("Saliendo del programa.\n");
             exit();
       }
}

// Apagamos los leds del sensor escribiendo 1 en DC_CNTL1 y DC_CNTL0.
DSK6713_rset(DSK6713_DC_REG,3);

// El sensor debe ser inicializado siempre antes de leer la imagen.
sensorInit();

// Leyendo imagen.
printf ("Capturando imagen.\n");
readImage(imageptr);
printf ("Lectura terminada. Puede retirar el dedo.\n");

if (verifyFP)
{
      // Preprocesado imagen capturada.
      histogram (image, hist, NFi, NCi);
      preprocessing(image, hist, NFi, NCi);
      copy_image(image, imagecapture, NFi, NCi);

      // Petición del número de usuario.
      waitNumUser1:
      printf("Introduzca su número de usuario: ");
      scanf("%d", &n);
      if (n<0 || n>=NUMUSER || database[n].userenroll==0)
      {
            printf("Número de usuario no válido.\n");
            goto waitNumUser1;
      }
      copy_image(database[n].imageuser, imagenroll, NFi, NCi);

      // Cálculo del desplazamiento.
      complex_image(imagecapture, fftcapture, NFi, NCi, NF, NC);
      fft2d(fftcapture, tw, tw, NF, NC);
      complex_image(imagenroll, fftenroll, NFi, NCi, NF, NC);
      fft2d(fftenroll, tw, tw, NF, NC);
```

```
        poc(fftcapture, fftenroll, NF, NC);
        fft2d(fftcapture, twi, twi, NF, NC);
        divide_ifft2d(fftcapture, NF, NC);
        high_peak_poc(fftcapture, coord, NF, NC);
        printf("Desplazamiento entre imágenes: %d filas, %d columnas.\n",
        coord[0], coord[1]);

        // Obtención de la zona común.
        desp_image(imagecapture, coord, 0, NFi, NCi);
        desp_image(imagenroll, coord, 1, NFi, NCi);

        // Transformación a coordenadas polares.
        trans_polar_image(imagecapture, fftcapture, coord, phasepolar, NFi,
        NCi, NF, NC);
        trans_polar_image(imagenroll, fftenroll, coord, phasepolar, NFi,
        NCi, NF, NC);

        // Valor de similitud.
        fft2d(fftcapture, tw, tw, NF, NC);
        fft2d(fftenroll, tw, tw, NF, NC);
        poc(fftcapture, fftenroll, NF, NC);
        fft2d(fftcapture, twi, twi, NF, NC);
        divide_ifft2d(fftcapture, NF, NC);
        simil = add_max_poc(fftcapture, NF, NC);
        printf("Resultado de la comparación = %f\n", simil);

        // Decisión.
        if (simil>=0.12)
        {
                printf("Identificación satisfactoria.\n");
                /* Encendemos el led verde escribiendo 0 en DC_CNTL1 para
                indicar la coincidencia entre huellas.*/
                DSK6713_rset(DSK6713_DC_REG,1);
        }
        else
        {
                printf("Identificación fallida.\n");
                /* Encendemos el led rojo escribiendo 0 en DC_CNTL0 para
                indicar la no coincidencia entre huellas.*/
                DSK6713_rset(DSK6713_DC_REG,2);
        }

        dummyVerify();
}

if (enrollFP)
{
        // Preprocesado de la imagen capturada.
        histogram (image, hist, NFi, NCi);
        preprocessing(image, hist, NFi, NCi);

        // Petición del número de usuario.
        waitNumUser2:
        printf("Introduzca el número de usuario elegido: ");
        scanf("%d", &n);
        if (n<0 || n>=NUMUSER)
        {
                printf("Número de usuario no válido.\n");
                goto waitNumUser2;
        }
```

```
            database[n].userenroll = 1;
            copy_image(image, database[n].imageuser, NFi, NCi);

            printf("Registro satisfactorio.\n");

            dummyEnroll();
      }

       goto waitForFp;
}

// Función para controlar el botón Registrar.
void dummyEnroll(void)
{
    enrollFP=0;
}

// Función para controlar el botón Verificar.
void dummyVerify(void)
{
    verifyFP=0;
}

/* Función que genera una tabla con valores de cosenos y senos en el
   vector phase. Esta tabla será usada por  la función trans_polar_image
   y su objetivo es reducir el tiempo de ejecución de dicha función.
   Parámetros de entrada:
   phase: vector de tamaño 2*nf. En los elementos pares se almacenarán
          los valores de los cosenos, y en los impares los correspondientes
          a los senos.
   nf: número de filas de la matriz que contendrá el resultado de la
          función trans_polar_image.*/
void gen_phase_table(float* phase, int nf)
{
   float tempp, invp, incp;
   int p;

   tempp = 0;
   invp = 1.0/nf;
   incp = (PI/2)*invp;

   for (p=0; p<nf; p++)
   {
          phase[2*p] = cos(tempp);
          phase[2*p+1] = sin(tempp);

          tempp += incp;
   }
}

/* Copia el contenido de la imagen imi en la imagen imo, ambas de tamaño
   nfxnc.
   Parámetros de entrada:
          imi: imagen de entrada.
          imo: imagen que contendrá la copia.
          nf: número de filas de la imagen a copiar.
          nc: número de columnas de la imagen a copiar.*/
void copy_image(unsigned char imi[][NCi], unsigned char imo[][NCi], int nf,
int nc)
{
```

```
   int f, c;

   for (f=0; f<nf; f++)
   {
         for (c=0; c<nc; c++)
         {
               imo[f][c] = imi[f][c];
         }
   }
}

/* Función que calcula el histograma de la imagen im, el cual es almacenado
   en h.
   Parámetros de entrada:
         im: imagen de entrada.
         h: puntero al vector que contendrá el histograma.
         nf: número de filas de la imagen.
         nc: número de columnas de la imagen.*/
void histogram(unsigned char im[][NCi], unsigned int* h, int nf, int nc)
{
   int f, c;

   for (f=0; f<256; f++)
   {
         h[f] = 0;
   }

   for (f=0; f<nf; f++)
   {
         for (c=0; c<nc; c++)
         {
               h[im[f][c]]++;
         }
   }
}

/* Función que aplica un preprocesado a la imagen im. El objetivo de éste es
   eliminar un poco de ruido , realzar las crestas (debido a que son las que
   aportan la información de la huella) y oscurecer los valles.
   Parámetros de entrada:
         im: imagen de entrada.
         h: puntero al vector que contiene el histograma de dicha imagen.
         nf: número de filas de la imagen.
         nc: número de columnas de la imagen.*/
void preprocessing(unsigned char im[][NCi], unsigned int* h, int nf, int
nc)
{
   int f, c, start, final, i, end, min_hist;
   float b, m, x;

   i = 0;
   end = 0;
   min_hist = 10;

   while (end==0)
   {
         if (h[i]<min_hist)
         {
               i++;
         }
```

```
        else
        {
                start = i;
                end = 1;
        }
    }

    while (end==1)
    {
        if (h[i]>min_hist)
        {
                i++;
        }
        else
        {
                final = i-1;
                end = 0;
        }
    }

    for (f=0; f<nf; f++)
    {
        for (c=0; c<nc; c++)
        {
                if (im[f][c]<start)
                {
                        im[f][c] = start;
                }
                if (im[f][c]>final)
                {
                        im[f][c] = final;
                }
        }
    }

    m =start-final;
    m = 255/m;
    b = final;
    b = -m*final;

    for (f=0; f<nf; f++)
    {
        for (c=0; c<nc; c++)
        {
                x = im[f][c];
                im[f][c] = m*x + b;
        }
    }
}

/* Esta función coge una ventana de tamaño nfxnc del centro de la imagen im,
   la cual tiene un tamaño nfixnci, y la pasa a la matriz compleja x para
   posteriormente aplicarle la FFT 2D.
   Parámetros de entrada:
        im: imagen de entrada.
        x: matriz que contendrá la información de la imagen en números
                complejos.
        nfi: número de filas de im.
        nci: número de columnas de im.
        nf: número de filas de x.
```

```
        nc: número de columnas de x.*/
void complex_image(unsigned char im[][NCi], float x[][2*NC], int nfi, int
nci, int nf, int nc)
{
   int f, c;
   int df, dc;

   df = (nfi-nf)/2;
   dc = (nci-nc)/2;

   for (f=0; f<nf; f++)
   {
         for (c=0; c<nc; c++)
         {
               x[f][2*c] = im[f+df][c+dc];
               x[f][2*c+1] = 0;
         }
   }
}

/* Función para generar la tabla twiddle.
   Parámetros de entrada:
         w: tabla donde se guardarán los valores twiddle necesarios para
               la FFT.
         n: número de elementos complejos a los que se les hará la FFT.*/
void gen_twiddle(float* w, int n)
{
    double delta = 2 * PI / n;
    int i;

    for(i = 0; i < n/2; i++)
    {
        w[2 * i + 1] = sin(i * delta);
        w[2 * i] = cos(i * delta);
    }
}

/* Función que copia un vector x en el vector y, ambos de tamaño n.
   Parámetros de entrada:
         x: vector que contiene los elementos a copiar.
         y: vector que recibirá los valores de x.
         n: número de elementos del vector x.*/
void copy_vector(float* x, float* y, int n)
{
   int i;

   for (i=0; i<n; i++)
   {
         y[i] = x[i];
   }
}

/* Función que realiza el complejo conjugado a todos los elementos del
   vector x de n elementos complejos.
   Parámetros de entrada:
         x: vector complejo al que se le quiere hacer el complejo conjugado.
         n: número de elementos complejos del vector.*/
void complex_conj_vector(float* x, int n)
{
   int i;
```

```
   for (i=0; i<n; i++)
   {
         x[2*i+1] = -x[2*i+1];
   }
}

/* Función que realiza un reordenamiento de los n elementos complejos del
   vector x mediante la operación del bit inverso.
   Parámetros de entrada:
         x: vector al que se le hará dicha transformación.
         n: número de elementos complejos del vector.*/
void bit_rev(float* x, int n)
{
   int i, j, k;
   float rtemp, itemp;

   j = 0;

   for(i=1; i < (n-1); i++)
   {
         k = n >> 1;
         while(k <= j)
         {
               j -= k;
               k >>= 1;
         }
         j += k;
         if(i < j)
         {
               rtemp = x[j*2];
               x[j*2] = x[i*2];
               x[i*2] = rtemp;
               itemp = x[j*2+1];
               x[j*2+1] = x[i*2+1];
               x[i*2+1] = itemp;
         }
   }
}

/* Dada la matriz compleja x, esta función realiza la traspuesta a la misma.
   Parámetros de entrada:
         x: matriz compleja a la que se le hará dicha transformación.
         nf: número de filas.
         nc: número de columnas.*/
void trasp_matrix(float x[][2*NC], int nf, int nc)
{
   float tempr, tempi;
   int f, c, b;

   b = 0;

   for (f=0; f<nf; f++)
   {
         for (c=b; c<nc; c++)
         {
               tempr = x[f][2*c];
               tempi = x[f][2*c+1];
               x[f][2*c] = x[c][2*f];
               x[f][2*c+1] = x[c][2*f+1];
```

```
                x[c][2*f] = tempr;
                x[c][2*f+1] = tempi;
        }
        b++;
    }
}

/* Función que realiza la FFT 2D a la matriz compleja x de tamaño nfxnc.
   Parámetros de entrada:
        x: matriz a la que se le hará la transformación.
        tw_f: tabla twidle correspondiente a las filas.
        tw_c: tabla twidle correspondiente a las columnas.
        nf: número de filas.
        nc: número de columnas.*/
void fft2d(float x[][2*NC], float* tw_f, float* tw_c, int nf, int nc)
{
    int f, c;
    float* ptr;
    short nffft, ncfft;

    nffft = nf;
    ncfft = nc;
    for (f=0; f<nf; f++)
    {
        ptr = &x[f][0];
        DSPF_sp_cfftr2_dit(ptr, tw_f, ncfft);
        bit_rev(ptr, nc);
    }
    trasp_matrix(x, nf, nc);
    for (c=0; c<nc; c++)
    {
        ptr = &x[c][0];
        DSPF_sp_cfftr2_dit(ptr, tw_c, nffft);
        bit_rev(ptr, nf);
    }
    trasp_matrix(x, nf, nc);
}

/* Función que realiza la división por el producto nfxnc (número de
   elementos) a todos los elementos de la matriz compleja x. Es aplicada
   después de realizar una IFFT 2D.
   Parámetros de entrada:
        x: matriz compleja a la que se le hará dicha transformación.
        nf: número de filas.
        nc: número de columnas.*/
void divide_ifft2d(float x[][2*NC], int nf, int nc)
{
    int f, c;
    float inv;

    inv = nf*nc;
    inv = 1.0/inv;

    for (f=0; f<nf; f++)
    {
        for (c=0; c<nc; c++)
        {
            x[f][2*c] = x[f][2*c]*inv;
            x[f][2*c+1] = x[f][2*c+1]*inv;
        }
```

```
   }
}

/* Realiza la función POC (Phase Only Correlation, del inglés) al par de
   elementos indicados por f (fila) y c (columna) de las matrices complejas
   x e y. El resultado se guardará en el mismo elemento de la matriz x.
   Parámetros de entrada:
         x: matriz compleja 1.
         y: matriz compleja 2.
         f: fila del elemento al que se le aplica la función.
         c: columna del elemento al que se le aplica la función.*/
void poc_function(float x[][2*NC], float y[][2*NC], int f, int c)
{
   float tempr, tempi, tempabs;

   if ((x[f][2*c]==0) && (x[f][2*c+1]==0))
   {
         if ((y[f][2*c]==0) && (y[f][2*c+1]==0))
         {
               tempr = 1;
               tempi = 0;
         }
         else
         {
               tempr = 0;
               tempi = 0;
         }
   }
   else if ((y[f][2*c]==0) && (y[f][2*c+1]==0))
   {
         tempr = 0;
         tempi = 0;
   }
   else
   {
         tempr = x[f][2*c]*y[f][2*c] + x[f][2*c+1]*y[f][2*c+1];
         tempi = x[f][2*c+1]*y[f][2*c] - x[f][2*c]*y[f][2*c+1];
         tempabs = sqrt(tempr*tempr + tempi*tempi);
         tempr = tempr/tempabs;
         tempi = tempi/tempabs;
   }
   x[f][2*c] = tempr;
   x[f][2*c+1] = tempi;
}

/* Función que realiza la función POC (Phase Only Correlation, del inglés)
   a las matrices complejas x e y, ambas de tamaño nfxnc y previamente
   convertidas al dominio de Fourier, devolviendo el resultado en la
   matriz x. Para ello hace uso de la función poc_function y reduce el
   número de operaciones a realizar aprovechando la propiedad de simetría
   de la transformada de Fourier 2D.
   Parámetros de entrada:
         x: matriz compleja 1. En esta matriz se guardará el resultado de la
               función.
         y: matriz compleja 2.
         nf: número de filas.
         nc: número de columnas.*/
void poc(float x[][2*NC], float y[][2*NC], int nf, int nc)
{
   int f, c;
```

```
	poc_function(x, y, 0, 0);
	poc_function(x, y, 0, nc/2);
	poc_function(x, y, nf/2, 0);
	poc_function(x, y, nf/2, nc/2);

	for (f=1; f<nf/2; f++)
	{
		c=0;
		poc_function(x, y, f, c);
		x[nf-f][0] = x[f][2*c];
		x[nf-f][1] = -x[f][2*c+1];
		c=nc/2;
		poc_function(x, y, f, c);
		x[nf-f][nc] = x[f][2*c];
		x[nf-f][nc+1] = -x[f][2*c+1];
	}

	for (c=1; c<nc/2; c++)
	{
		f=0;
		poc_function(x, y, f, c);
		x[0][2*nc-2*c] = x[f][2*c];
		x[0][2*nc-2*c+1] = -x[f][2*c+1];
		f=nf/2;
		poc_function(x, y, f, c);
		x[nf/2][2*nc-2*c] = x[f][2*c];
		x[nf/2][2*nc-2*c+1] = -x[f][2*c+1];
	}

	for (f=1; f<nf/2; f++)
	{
		for (c=1; c<nc/2; c++)
		{
			poc_function(x, y, f, c);
			x[nf-f][2*nc-2*c] = x[f][2*c];
			x[nf-f][2*nc-2*c+1] = -x[f][2*c+1];
		}
	}

	for (f=nf/2+1; f<nf; f++)
	{
		for (c=1; c<nc/2; c++)
		{
			poc_function(x, y, f, c);
			x[nf-f][2*nc-2*c] = x[f][2*c];
			x[nf-f][2*nc-2*c+1] = -x[f][2*c+1];
		}
	}
}

/* Función que encuentra los dos valores más altos dados por la aplicación
   de la función POC a dos matrices complejas y devuelve su suma.
   Parámetros de entrada:
		x: matriz compleja con el resultado de la función POC a la que se
			le ha hecho la IFFT 2D previamente para convertirla al
			dominio espacial.
		nf: número de filas.
		nc: número de columnas.*/
float add_max_poc(float x[][2*NC], int nf, int nc)
```

```
{
	int f, c;
	float max1, max2;

	max1 = 0;
	max2 = 0;

	for (f=0; f<nf; f++)
	{
		for (c=0; c<nc; c++)
		{
			if (x[f][2*c] > max1)
			{
				max1 = x[f][2*c];
			}
			else if (x[f][2*c] > max2)
			{
				max2 = x[f][2*c];
			}
		}
	}
	max1 += max2;
	return max1;
}

/* Función que encuentra la posición del pico más alto devuelto por la
   función POC.
   Parámetros de entrada:
		x: matriz compleja con el resultado de la función POC a la que se
			le ha hecho la IFFT 2D previamente.
		nf: número de filas.
		nc: número de columnas.*/
void high_peak_poc(float x[][2*NC], int* coor, int nf, int nc)
{
	int f, c, i, j;
	float max;

	i = 0;
	j = 0;
	max = 0;

	for (f=0; f<nf; f++)
	{
		for (c=1; c<nc; c++)
		{
			if (x[f][2*c]>max)
			{
				max = x[f][2*c];
				i = f;
				j = c;
			}
		}
	}

	if (x[0][0]/10>max)
	{
		i = 0;
		j = 0;
	}
	else
```

```
	{
		if (i>nf/2)
		{
			i = -(nf-i);
		}
		else
		{
			i = i;
		}
		if (j>nc/2)
		{
			j = nc-j;
		}
		else
		{
			j = -j;
		}
	}
	coor[0] = i;
	coor[1] = j;
}

/* Función que recorta la imagen im según el desplazamiento calculado
   previamente respecto a otra imagen indicado por el vector coor. Con
   esto se extrae la zona común de ambas imágenes.
   Parámetros de entrada:
		im: imagen que será recortada.
		coor: desplazamiento entre huellas.
		enr: = 0 => se refiere a la imagen capturada; = 1 => se refiere a
			la imagen de la base de datos.
		nf: número de filas de las imágenes.
		nc: número de columnas de las imágenes.*/
void desp_image(unsigned char im[][NCi], int* coor, int enr, int nf, int
nc)
{
	int f, c, df, dc, absdf, absdc;

	if (enr==1)
	{
		if (coor[0]>=0 && coor[1]>=0)
		{
			df = 0;
			dc = coor[1];
			absdf = coor[0];
			absdc = coor[1];
		}
		else if (coor[0]>=0 && coor[1]<0)
		{
			df = 0;
			dc = 0;
			absdf = coor[0];
			absdc = -coor[1];
		}
		else if (coor[0]<0 && coor[1]>=0)
		{
			df = -coor[0];
			dc = coor[1];
			absdf = -coor[0];
			absdc = coor[1];
		}
```

```
        else
        {
                df = -coor[0];
                dc = 0;
                absdf = -coor[0];
                absdc = -coor[1];
        }
    }
    else
    {
        if (coor[0]>=0 && coor[1]>=0)
        {
                df = coor[0];
                dc = 0;
                absdf = coor[0];
                absdc = coor[1];
        }
        else if (coor[0]>=0 && coor[1]<0)
        {
                df = coor[0];
                dc = -coor[1];
                absdf = coor[0];
                absdc = -coor[1];
        }
        else if (coor[0]<0 && coor[1]>=0)
        {
                df = 0;
                dc = 0;
                absdf = -coor[0];
                absdc = coor[1];
        }
        else
        {
                df = 0;
                dc = -coor[1];
                absdf = -coor[0];
                absdc = -coor[1];
        }
    }

    for (f=0; f<nf-absdf; f++)
    {
        for (c=0; c<nc-absdc; c++)
        {
                im[f][c] = im[f+df][c+dc];
        }
        for (c=nc-absdc; c<nc; c++)
        {
                im[f][c] = 0;
        }
    }
    for (f=nf-absdf; f<nf; f++)
    {
        for (c=0; c<nc; c++)
        {
                im[f][c] = 0;
        }
    }
}
```

```
/* Función para pasar a la matriz compleja x la imagen im en coordenadas
   polares atendiendo a su desplazamiento.
   Parámetros de entrada:
         im: imagen de entrada.
         x: matriz compleja a la que se le pasará la información (filas =>
               fase; columnas => radio).
         coor: desplazamiento entre imágenes.
         phase: dicho vector contiene valores de senos y cosenos usados en
               esta función para reducir su tiempo de ejecución.
         nfi: número de filas de im.
         nci: número de columnas de im.
         nf: número de filas de x.
         nc: número de columnas de x.*/
void trans_polar_image(unsigned char im[][NCi], float x[][2*NC], int* coor,
float* phase, int nfi, int nci, int nf, int nc)
{
   int m, p, f, c, absdf, absdc, mmax;
   float tempf, tempc, tempm;
   float invm, incm;

   tempm=0;

   if (coor[0]>=0)
   {
         absdf = coor[0];
   }
   else
   {
         absdf = -coor[0];
   }
   if (coor[1]>=0)
   {
         absdc = coor[1];
   }
   else
   {
         absdc = -coor[1];
   }

   if ((nfi-absdf)>(nci-absdc))
   {
         mmax = nci-absdc;
   }
   else
   {
         mmax = nfi-absdf;
   }

   invm = 1.0/nc;
   incm = mmax*invm;

   for (m=0; m<nc; m++)
   {
         for (p=0; p<nf; p++)
         {
               tempf = tempm*phase[2*p+1];
               tempc = tempm*phase[2*p];
               f = tempf;
               c = tempc;
```

```
                x[p][2*m] = im[f][c];
                x[p][2*m+1] = 0;
            }

            tempm += incm;
        }
    }
```

Capítulo 3. Archivo botones.gel

```
/*****************************************************************************
*
*  CREADOR: Juan Manuel Fernández Muñoz
*
*****************************************************************************
*****************************************************************************
*
*  NOMBRE: botones.gel
*
*
*  DESCRIPCIÓN:      Archivo creado para el control del algoritmo
*                    desarrollado mediante botones virtuales.
*
*****************************************************************************/

menuitem "Botones";

hotmenu Registrar()
{
   enrollFP = 1;

   GEL_Go(dummyEnroll);
   GEL_Run();
}

hotmenu Verificar()
{
   verifyFP = 1;

   GEL_Go(dummyVerify);
   GEL_Run();
}

hotmenu Salir()
{
   exitFP = 1;
}
```

ANEXO 2

DOCUMENTACIÓN TMS320C6713 DSK

SPECTRUM DIGITAL INCORPORATED

TMS320C6713 DSK

Technical Reference

2003 ***DSP Development Systems***

TMS320C6713 DSK
Technical Reference

506735-0001 Rev. B
November 2003

SPECTRUM DIGITAL, INC.
12502 Exchange Drive, Suite 440 Stafford, TX. 77477
Tel: 281.494.4505 Fax: 281.494.5310
sales@spectrumdigital.com www.spectrumdigital.com

IMPORTANT NOTICE

Spectrum Digital, Inc. reserves the right to make changes to its products or to discontinue any product or service without notice. Customers are advised to obtain the latest version of relevant information to verify that the data being relied on is current before placing orders.

Spectrum Digital, Inc. warrants performance of its products and related software to current specifications in accordance with Spectrum Digital's standard warranty. Testing and other quality control techniques are utilized to the extent deemed necessary to support this warranty.

Please be aware that the products described herein are not intended for use in life-support appliances, devices, or systems. Spectrum Digital does not warrant nor is Spectrum Digital liable for the product described herein to be used in other than a development environment.

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

WARNING

This equipment is intended for use in a laboratory test environment only. It generates, uses, and can radiate radio frequency energy and has not been tested for compliance with the limits of computing devices pursuant to subpart J of part 15 of FCC rules, which are designed to provide reasonable protection against radio frequency interference. Operation of this equipment in other environments may cause interference with radio communications, in which case the user at his own expense will be required to take whatever measures necessary to correct this interference.

Contents

About This Manual

This document describes the board level operations of the TMS320C6713 DSP Starter Kit (DSK) module. The DSK is based on the Texas Instruments TMS320C6713 Digital Signal Processor.

The TMS320C6713 DSK is a table top card to allow engineers and software developers to evaluate certain characteristics of the TMS320C6713 DSP to determine if the processor meets the designers application requirements. Evaluators can create software to execute onboard or expand the system in a variety of ways.

Notational Conventions

This document uses the following conventions.

The TMS320C6713 DSK will sometimes be referred to as the DSK.

Program listings, program examples, and interactive displays are shown is a special italic typeface. Here is a sample program listing.

equations
!rd = !strobe&rw;

Information About Cautions

This book may contain cautions.
This is an example of a caution statement.
A caution statement describes a situation that could potentially damage your software, or hardware, or other equipment. The information in a caution is provided for your protection. Please read each caution carefully.

Related Documents

Texas Instruments TMS320C67xx DSP CPU Reference Guide
Texas Instruments TMS320C67xx DSP Peripherals Reference Guide

Table 1: Manual History

Revision	History
A	Alpha Release

Chapter 1

Introduction to the TMS320C6713 DSK

Chapter One provides a description of the TMS320C6713 DSK along with the key features and a block diagram of the circuit board.

Topic **Page**

1.1 Key Features

The C6713 DSK is a low-cost standalone development platform that enables users to evaluate and develop applications for the TI C67xx DSP family. The DSK also serves as a hardware reference design for the TMS320C6713 DSP. Schematics, logic equations and application notes are available to ease hardware development and reduce time to market.

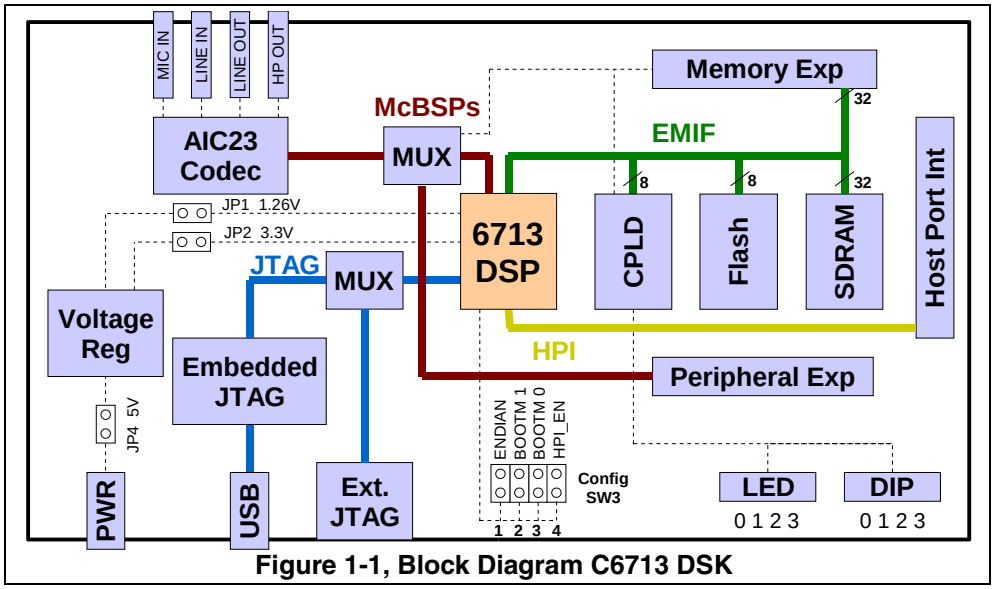


Figure 1-1, Block Diagram C6713 DSK

The DSK comes with a full compliment of on-board devices that suit a wide variety of application environments. Key features include:

- A Texas Instruments TMS320C6713 DSP operating at 225 MHz.
- An AIC23 stereo codec
- 16 Mbytes of synchronous DRAM
- 512 Kbytes of non-volatile Flash memory (256 Kbytes usable in default configuration)
- 4 user accessible LEDs and DIP switches
- Software board configuration through registers implemented in CPLD
- Configurable boot options
- Standard expansion connectors for daughter card use
- JTAG emulation through on-board JTAG emulator with USB host interface or external emulator
- Single voltage power supply (+5V)

1.2 Functional Overview of the TMS320C6713 DSK

The DSP on the 6713 DSK interfaces to on-board peripherals through a 32-bit wide EMIF (External Memory InterFace). The SDRAM, Flash and CPLD are all connected to the bus. EMIF signals are also connected daughter card expansion connectors which are used for third party add-in boards.

The DSP interfaces to analog audio signals through an on-board AIC23 codec and four 3.5 mm audio jacks (microphone input, line input, line output, and headphone output). The codec can select the microphone or the line input as the active input. The analog output is driven to both the line out (fixed gain) and headphone (adjustable gain) connectors. McBSP0 is used to send commands to the codec control interface while McBSP1 is used for digital audio data. McBSP0 and McBSP1 can be re-routed to the expansion connectors in software.

A programmable logic device called a CPLD is used to implement glue logic that ties the board components together. The CPLD has a register based user interface that lets the user configure the board by reading and writing to its registers.

The DSK includes 4 LEDs and a 4 position DIP switch as a simple way to provide the user with interactive feedback. Both are accessed by reading and writing to the CPLD registers.

An included 5V external power supply is used to power the board. On-board switching voltage regulators provide the +1.26V DSP core voltage and +3.3V I/O supplies. The board is held in reset until these supplies are within operating specifications.

Code Composer communicates with the DSK through an embedded JTAG emulator with a USB host interface. The DSK can also be used with an external emulator through the external JTAG connector.

1.3 Basic Operation

The DSK is designed to work with TI's Code Composer Studio development environment and ships with a version specifically tailored to work with the board. Code Composer communicates with the board through the on-board JTAG emulator. To start, follow the instructions in the Quick Start Guide to install Code Composer. This process will install all of the necessary development tools, documentation and drivers.

After the install is complete, follow these steps to run Code Composer. The DSK must be fully connected to launch the DSK version of Code Composer.

1) Connect the included power supply to the DSK.

2) Connect the DSK to your PC with a standard USB cable (also included).

3) Launch Code Composer from its icon on your desktop.

Detailed information about the DSK including a tutorial, examples and reference material is available in the DSK's help file. You can access the help file through Code Composer's help menu. It can also be launched directly by double-clicking on the file c6713dsk.hlp in Code Composer's docs\hlp subdirectory.

1.4 Memory Map

The C67xx family of DSPs has a large byte addressable address space. Program code and data can be placed anywhere in the unified address space. Addresses are always 32-bits wide.

The memory map shows the address space of a generic 6713 processor on the left with specific details of how each region is used on the right. By default, the internal memory sits at the beginning of the address space. Portions of the internal memory can be reconfigured in software as L2 cache rather than fixed RAM.

The EMIF has 4 separate addressable regions called chip enable spaces (CE0-CE3). The SDRAM occupies CE0 while the Flash and CPLD share CE1. CE2 and CE3 are generally reserved for daughtercards.

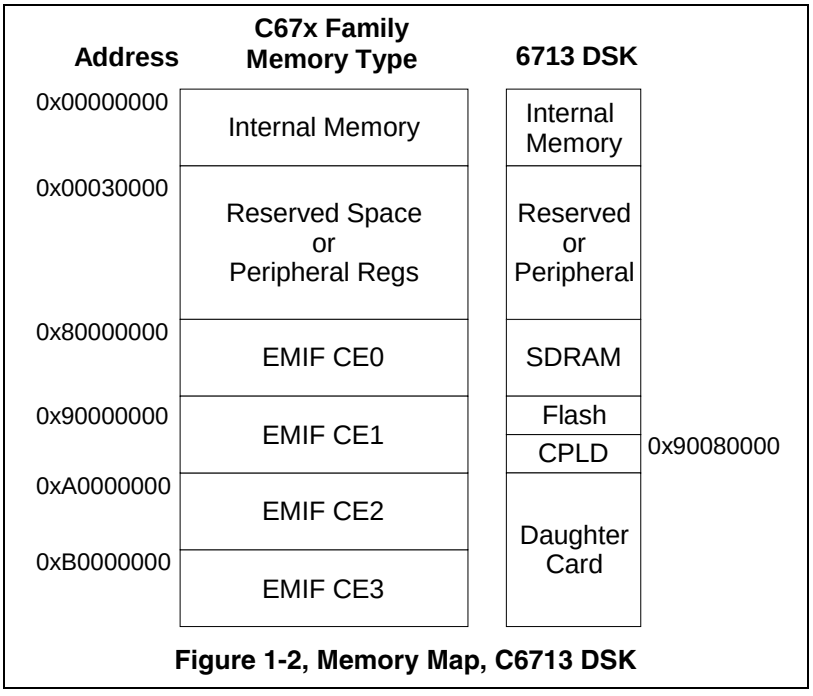


Figure 1-2, Memory Map, C6713 DSK

1.5 Configuration Switch Settings

The DSK has 4 configuration switches that allows users to control the operational state of the DSP when it is released from reset. The configuration switch block is labeled SW3 on the DSK board, next to the reset switch.

Configuration switch 1 controls the endianness of the DSP while switches 2 and 3 configure the boot mode that will be used when the DSP starts executing. Configuration switch 4 controls the on-chip multiplexing of HPI and McASP signals brought out to the HPI expansion connector. By default all switches are off which corresponds to EMIF boot (out of 8-bit Flash) in little endian mode and HPI signals on the HPI expansion connector.

Table 1: Configuration Switch Settings

Switch 1	Switch 2	Switch 3	Switch 4	Configuration Description
Off				Little endian (default)
On				Big endian
	Off	Off		EMIF boot from 8-bit Flash (default)
	Off	On		HPI/Emulation boot
	On	Off		32-bit EMIF boot
	On	On		16-bit EMIF boot
			Off	HPI enabled on HPI pins (default)
			On	McASP1 enabled on HPI pins

1.6 Power Supply

The DSK operates from a single +5V external power supply connected to the main power input (J5). Internally, the +5V input is converted into +1.26V and +3.3V using separate voltage regulators. The +1.26V supply is used for the DSP core while the +3.3V supply is used for the DSP's I/O buffers and all other chips on the board. The power connector is a 2.5mm barrel-type plug.

There are three power test points on the DSK at JP1, JP2 and JP4. All I/O current passes through JP2 while all core current passes through JP1. All system current passes through JP4. Normally these jumpers are closed. To measure the current passing through remove the jumpers and connect the pins with a current measuring device such as a multimeter or current probe.

It is possible to provide the daughter card with +12V and -12V when the external power connector (J6) is used.

Chapter 2

Board Components

This chapter describes the operation of the major board components on the TMS320C6713 DSK.

Topic **Page**

2.1 CPLD (Programmable Logic)

The C6713 DSK uses an Altera EPM3128TC100-10 Complex Programmable Logic Device (CPLD) device to implement:

- 4 Memory-mapped control/status registers that allow software control of various board features.
- Control of the daughter card interface and signals.
- Assorted "glue" logic that ties the board components together.

2.1.1 CPLD Overview

The CPLD logic is used to implement functionality specific to the DSK. Your own hardware designs will likely implement a completely different set of functions or take advantage of the DSPs high level of integration for system design and avoid the use of external logic completely.

The CPLD implements simple random logic functions that eliminate the need for additional discrete devices. In particular, the CPLD aggregates the various reset signals coming from the reset button and power supervisors and generates a global reset.

The EPM3128TC100-10 is a 3.3V (5V tolerant), 100-pin QFP device that provides 128 macrocells, 80 I/O pins, and a 10 ns pin-to-pin delay. The device is EEPROM-based and is in-system programmable via a dedicated JTAG interface (a 10-pin header on the DSK). The CPLD source files are written in the industry standard VHDL (Hardware Design Language) and included with the DSK.

2.1.2 CPLD Registers

The 4 CPLD memory-mapped registers allows users to control CPLD functions in software. On the 6713 DSK the registers are primarily used to access the LEDs and DIP switches and control the daughter card interface. The registers are mapped into EMIF CE1 data space at address 0x90080000. They appear as 8-bit registers with a simple asynchronous memory interface. The following table gives a high level overview of the CPLD registers and their bit fields:

The table below shows the bit definitions for the 4 registers in CPLD.

Table 1: CPLD Register Definitions

Offset	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	USER_REG	USR_SW3 R	USR_SW2 R	USR_SW1 R	USR_SW0 R	USR_LED3 R/W 0(Off)	USR_LED2 R/W 0(Off)	USR_LED1 R/W 0(Off)	USR_LED0 R/W 0(Off)
1	DC_REG	DC_DET R	0	DC_STAT1 R	DC_STAT0 R	DC_RST R 0(No reset)	0	DC_CNTL1 R/W 0(low)	DC_CNTL0 R/W 0(low)
4	VERSION	CPLD_VER[3.0] R				0	BOARD VERSION[2.0] R		
6	MISC	SCR_5 R/W 0	SCR_4 R/W 0	SCR_3 R/W 0	SCR_2 R/W 0	SCR_1 R/W 0	FLASH_PAGE R/W 0 (Flash A19=0)	McBSP1 ON/OFF Board R/W 0 (Onboard)	McBSP0 ON/OFF Board R/W 0 (Onboard)

2.1.3 USER_REG Register

USER_REG is used to read the state of the 4 DIP switches and turn the 4 LEDs on or off to allow the user to interact with the DSK. The DIP switches are read by reading the top 4 bits of the register and the LEDs are set by writing to the low 4 bits.

Table 2: CPLD USER_REG Register

Bit	Name	R/W	Description
7	USER_SW3	R	User DIP Switch 3(1 = Off, 0 = On)
6	USER_SW2	R	User DIP Switch 2(1 = Off, 0 = On)
5	USER_SW1	R	User DIP Switch 1(1 = Off, 0 = On)
4	USER_SW0	R	User DIP Switch 0(1 = Off, 0 = On)
3	USER_LED3	R/W	User-defined LED 3 Control (0 = Off, 1 = On)
2	USER_LED2	R/W	User-defined LED 2 Control (0 = Off, 1 = On)
1	USER_LED1	R/W	User-defined LED 1 Control (0 = Off, 1 = On)
0	USER_LED0	R/W	User-defined LED 0 Control (0 = Off, 1 = On)

2.1.4 DC_REG Register

DC_REG is used to monitor and control the daughter card interface. DC_DET detects the presence of a daughter card. DC_STAT and DC_CNTL provide simple communications with the daughter card through readable status lines and writable control lines.

The daughter card is released from reset when the DSP is released from reset. DC_RST can be used to put the card back in reset.

Table 3: DC_REG Register

Bit	Name	R/W	Description
7	DC_DET	R	Daughter Card Detect (1= Board detected)
6	0	R	Always zero
5	DC_STAT1	R	Daughter Card Status 1 (0=Low, 1 = High)
4	DC_STAT0	R	Daughter Card Status 0 (0=Low, 1 = High)
3	DC_RST	R/W	Daughter Card Reset (0=No Reset, 1 = Reset)
2	0	R	Always zero
1	DC_CNTL1	R/W	Daughter Card Control 1(0 = Low, 1 = High)
0	DC_CNTL0	R/W	Daughter Card Control 0(0 = Low, 1 = High)

2.1.5 VERSION Register

The VERSION register contains two read only fields that indicate the BOARD and CPLD versions. This register will allow your software to differentiate between production releases of the DSK and account for any variances. This register is not expected to change often, if at all.

Table 4: Version Register Bit Definitions

Bit #	Name	R/W	Description
7	CPLD_VER3	R	Most Significant CPLD Version Bit
6	CPLD_VER2	R	CPLD Version Bit
5	CPLD_VER1	R	CPLD Version Bit
4	CPLD_VER0	R	Least Significant CPLD Version Bit
3	0	R	Always zero
2	DSK_VER2	R	Most Significant DSK Board Version Bit
1	DSK_VER1	R	DSK Board Version Bit
0	DSK_VER0	R	Least Significant DSK Board Version Bit

2.1.6 MISC Register

The MISC register is used to provide software control for miscellaneous board functions. On the 6713 DSK, the MISC register controls how auxiliary signals are brought out to the daughter-card connectors.

McBSP0 and McBSP1 are usually used as the control and data ports of the on-board AIC23 codec. The power-on state of these bits (both 0s) represents that situation. Set the corresponding McBSP select bit to use the McBSP with a daughter card instead.

The Flash and CPLD share CE1 which means that the highest DSP address bit (A21) is used to differentiate between the two. The FLASH_PAGE bit is driven to the Flash as a replacement for that address line which is connected to A19 of the Flash. On a standard DSK, the on-board Flash is not large enough for this bit to be significant. FLASH_PAGE is only useful if the board is re-populated with a larger pin-compatible Flash chip.

The scratch bits are unused. They can be set to any value.

Table 5: MISC Register

Bit	Name	R/W	Description
7	SCRATCH_5	R/W	Scratch bit 5
6	SCRATCH_4	R/W	Scratch bit 4
5	SCRATCH_3	R/W	Scratch bit 3
4	SCRATCH_2	R/W	Scratch bit 2
3	SCRATCH_1	R/W	Scratch bit 1
2	FLASH_PAGE	R/W	Flash address bit 19
1	MCBSP1SEL	R/W	McBSP1 on/off board (0 = on-board, 1 = off-board)
0	MCBSP0SEL	R/W	McBSP0 on/off board (0 = on-board, 1 = off-board)

2.2 AIC23 Codec

The DSK uses a Texas Instruments AIC23 (part #TLV320AIC23) stereo codec for input and output of audio signals. The codec samples analog signals on the microphone or line inputs and converts them into digital data so it can be processed by the DSP. When the DSP is finished with the data it uses the codec to convert the samples back into analog signals on the line and headphone outputs so the user can hear the output.

The codec communicates using two serial channels, one to control the codec's internal configuration registers and one to send and receive digital audio samples. McBSP0 is used as the unidirectional control channel. It should be programmed to send a 16-bit control word to the AIC23 in SPI format. The top 7 bits of the control word should specify the register to be modified and the lower 9 should contain the register value. The control channel is only used when configuring the codec, it is generally idle when audio data is being transmitted,

McBSP1 is used as the bi-directional data channel. All audio data flows through the data channel. Many data formats are supported based on the three variables of sample width, clock signal source and serial data format. The DSK examples generally use a 16-bit sample width with the codec in master mode so it generates the frame sync and bit clocks at the correct sample rate without effort on the DSP side. The preferred serial format is DSP mode which is designed specifically to operate with the McBSP ports on TI DSPs.

The codec has a 12MHz system clock. The 12MHz system clock corresponds to USB sample rate mode, named because many USB systems use a 12MHz clock and can use the same clock for both the codec and USB controller. The internal sample rate generate subdivides the 12MHz clock to generate common frequencies such as 48KHz, 44.1KHz and 8KHz. The sample rate is set by the codec's SAMPLERATE register. The figure below shows the codec interface on the C6713 DSK.

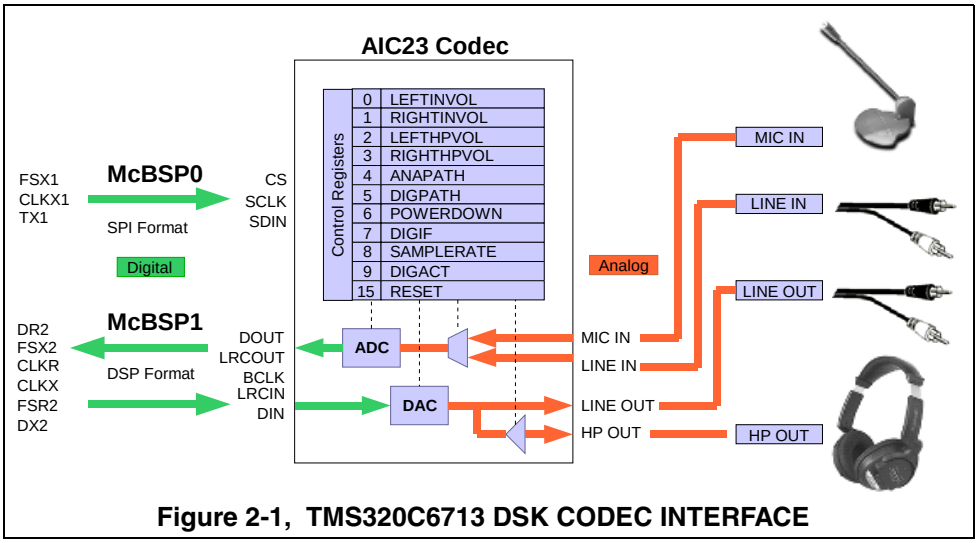


Figure 2-1, TMS320C6713 DSK CODEC INTERFACE

2.3 Synchronous DRAM

The DSK uses a 128 megabit synchronous DRAM (SDRAM) on the 32-bit EMIF. The SDRAM is mapped at the beginning of CE0 (address 0x80000000). Total available memory is 16 megabytes. The integrated SDRAM controller is part of the EMIF and must be configured in software for proper operation. The EMIF clock is derived from the PLL settings and should be configured in software at 90MHz. This number is based on an internal PLL clock of 450MHz required to achieve 225 MHz operation with a divisor of 2 and a 90MHz EMIF clock with a divisor of 5.

When using SDRAM, the controller must be set up to refresh one row of the memory array every 15.6 microseconds to maintain data integrity. With a 90MHz EMIF clock, this period is 1400 bus cycles.

2.4 Flash Memory

Flash is a type of memory which does not lose its contents when the power is turned off. When read it looks like a simple asynchronous read-only memory (ROM). Flash can be erased in large blocks commonly referred to as sectors or pages. Once a block has been erased each word can be programmed once through a special command sequence. After than the entire block must be erased again to change the contents.

The DSK uses a 512Kbyte external Flash as a boot option. It is visible at the beginning of CE1 (address 0x90000000). The Flash is wired as a 256K by 16 bit device to support the DSK's 16-bit boot option. However, the software that ships with the DSK treats the Flash as an 8-bit device (ignoring the top 8 bits) to match the 6713's default 8-bit boot mode. In this configuration, only 256Kbytes are readily usable without software changes.

2.5 LEDs and DIP Switches

The DSK includes 4 software accessible LEDs (D7-D10) and DIP switches (SW1) that provide the user a simple form of input/output. Both are accessed through the CPLD USER_REG register.

2.6 Daughter Card Interface

The DSK provides three expansion connectors that can be used to accept plug-in daughter cards. The daughter card allows users to build on their DSK platform to extend its capabilities and provide customer and application specific I/O. The expansion connectors are for memory, peripherals, and the Host Port Interface (HPI)

The memory connector provides access to the DSP's asynchronous EMIF signals to interface with memories and memory mapped devices. It supports byte addressing on 32 bit boundries. The peripheral connector brings out the DSP's peripheral signals like McBSPs, timers, and clocks. Both connectors provide power and ground to the daughter card

The HPI is a high speed interface that can be used to allow multiple DSPs to communicate and cooperate on a given task. The HPI connector brings out the HPI specific control signals.

Most of the expansion connector signals are buffered so that the daughter card cannot directly influence the operation of the DSK board. The use of TI low voltage, 5V tolerant buffers, and CBT interface devices allows the use of either +5V or +3.3V devices to be used on the daughter card.

Other than the buffering, most daughter card signals are not modified on the board. However, a few daughter card specific control signals like DC_RESET and DC_DET exist and are accessible through the CPLD DC_REG register. The DSK also multiplexes the McBSP0 and McBSP1 of on-board or external use. This function is controlled through the CPLD MISC register.

Chapter 3

Physical Description

This chapter describes the physical layout of the TMS320C6713 DSK and its connectors.

Topic **Page**

3.1 Board Layout

The C6713 DSK is a 8.75 x 4.5 inch (210 x 115 mm.) multi-layer board which is powered by an external +5 volt only power supply. Figure 3-1 shows the layout of the C6713 DSK.

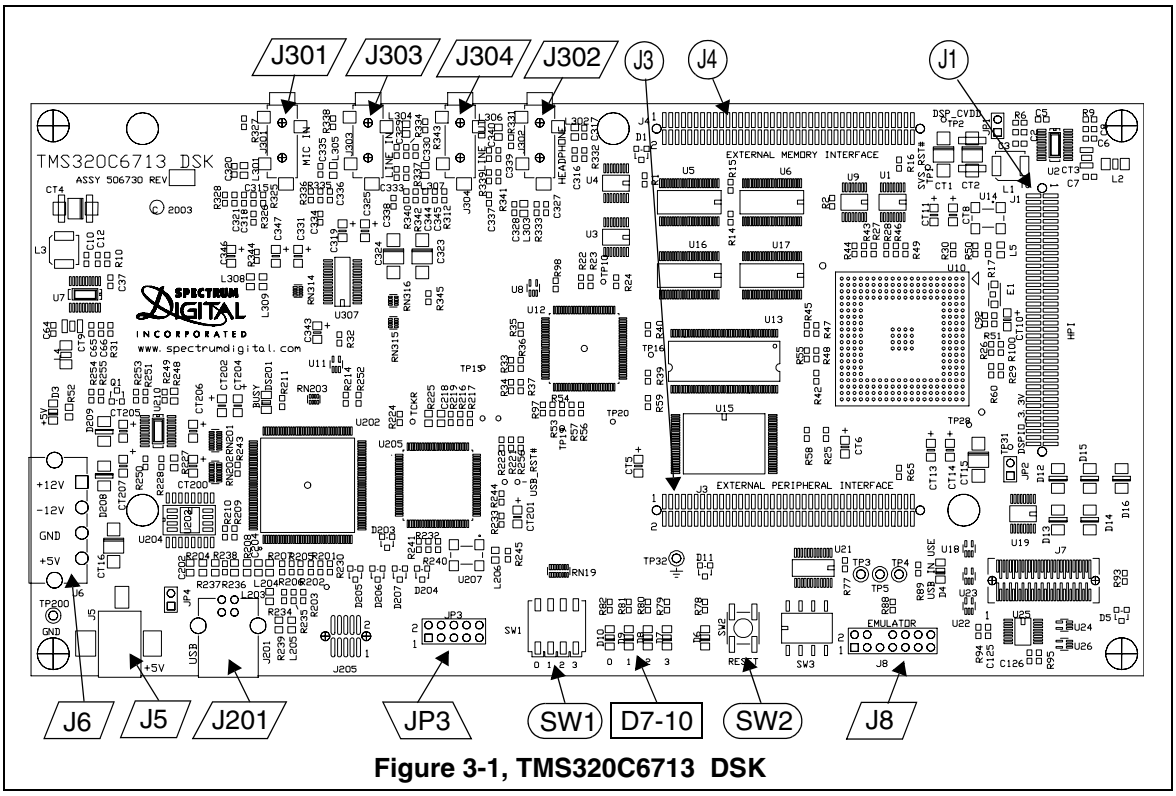


Figure 3-1, TMS320C6713 DSK

3.2 Connector Index

The TMS320C6713 DSK has many connectors which provide the user access to the various signals on the DSK.

Table 1: TMS320C6713 DSK Connectors

Connector	# Pins	Function
J4	80	Memory
J3	80	Peripheral
J1	80	HPI
J301	3	Microphone
J303	3	Line In
J304	3	Line Out
J303	3	Headphone
J5	2	+5 Volt
J6 *	4	Optional Power Connector
J8	14	External JTAG
J201	5	USB Port
JP3	10	CPLD Programming
SW3	8	DSP Configuration Jumper

Note: "*" Not populated

3.3 Expansion Connectors

The TMS320C6713 DSK supports three expansion connectors that follow the Texas Instruments interconnection guidelines. The expansion connector pinouts are described in the following three sections.

The three expansion connectors are all 80 pin 0.050 x 0.050 inches low profile connectors from Samtec or AMP. The Samtec SFM Series (surface mount) connectors are designed for high speed interconnections because they have low propagation delay, capacitance, and cross talk. The connectors present a small foot print on the DSK. Each connector includes multiple ground, +5V, and +3.3V power signals so that the daughter card can obtain power directly from the DSK. The peripheral expansion connector additionally provides both +12V and -12V to the daughter card. The recommended mating connector, whose part number is TFM-140-32-S-D-LC, is a surface mount connector that provides a 0.465" mated height.

Note: I is on an Input pin
O is on an Output pin
Z is on a High Impedance pin

3.3.1 J4, Memory Expansion Connector

Table 2: J4, Memory Expansion Connector

Pin	Signal	I/O	Description	Pin	Signal	I/O	Description
1	5V	Vcc	5V voltage supply pin	2	5V	Vcc	5V voltage supply pin
3	AEA21	O	EMIF address pin 21	4	AEA20	O	EMIF address pin 20
5	AEA19	O	EMIF address pin 19	6	AEA18	O	EMIF address pin 18
7	AEA17	O	EMIF address pin 17	8	AEA16	O	EMIF address pin 16
9	AEA15	O	EMIF address pin 15	10	AEA14	O	EMIF address pin 14
11	GND	Vss	System ground	12	GND	Vss	System ground
13	AEA13	O	EMIF address pin 13	14	AEA12	O	EMIF address pin 12
15	AEA11	O	EMIF address pin 11	16	AEA10	O	EMIF address pin 10
17	AEA9	O	EMIF address pin 9	18	AEA8	O	EMIF address pin 8
19	AEA7	O	EMIF address pin 7	20	AEA6	O	EMIF address pin 6
21	5V	Vcc	5V voltage supply pin	22	5V	Vcc	5V voltage supply pin
23	AEA5	O	EMIF address pin 5	24	AEA4	O	EMIF address pin 4
25	AEA3	O	EMIF address pin 3	26	AEA2	O	EMIF address pin 2
27	ABE3#	O	EMIF byte enable 3	28	ABE2#	O	EMIF byte enable 2
29	ABE1#	O	EMIF byte enable 1	30	ABE0#	O	EMIF byte enable 0
31	GND	Vss	System ground	32	GND	Vss	System ground
33	AED31	I/O	EMIF data pin 31	34	AED30	I/O	EMIF data pin 30
35	AED29	I/O	EMIF data pin 29	36	AED28	I/O	EMIF data pin 28
37	AED27	I/O	EMIF data pin 27	38	AED26	I/O	EMIF data pin 26
39	AED25	I/O	EMIF data pin 25	40	AED24	I/O	EMIF data pin 24
41	3.3V	Vcc	3.3V voltage supply pin	42	3.3V	Vcc	3.3V voltage supply pin
43	AED23	I/O	EMIF data pin 23	44	AED22	I/O	EMIF data pin 22
45	AED21	I/O	EMIF data pin 21	46	AED20	I/O	EMIF data pin 20
47	AED19	I/O	EMIF data pin 19	48	AED18	I/O	EMIF data pin 18
49	AED17	I/O	EMIF data pin 17	50	AED16	I/O	EMIF data pin 16
51	GND	Vss	System ground	52	GND	Vss	System ground
53	AED15	I/O	EMIF data pin 15	54	AED14	I/O	EMIF data pin 14
55	AED13	I/O	EMIF data pin 13	56	AED12	I/O	EMIF data pin 12
57	AED11	I/O	EMIF data pin 11	58	AED10	I/O	EMIF data pin 10
59	AED9	I/O	EMIF data pin 9	60	AED8	I/O	EMIF data pin 8
61	GND	Vss	System ground	62	GND	Vss	System ground
63	AED7	I/O	EMIF data pin 7	64	AED6	I/O	EMIF data pin 6
65	AED5	I/O	EMIF data pin 5	66	AED4	I/O	EMIF data pin 4
67	AED3	I/O	EMIF data pin 3	68	AED2	I/O	EMIF data pin 2
69	AED1	I/O	EMIF data pin 1	70	AED0	I/O	EMIF data pin 0
71	GND	Vss	System ground	72	GND	Vss	System ground
73	AARE#	O	EMIF async read enable	74	AAWE#	O	EMIF async write enable
75	AAOE#	O	EMIF async output enable	76	AARDY	I	EMIF asynchronous ready
77	ACE3#	O	Chip enable 3	78	ACE2#	O	Chip enable 2
79	GND	Vss	System ground	80	GND	Vss	System ground

3.3.2 J3, Peripheral Expansion Connector

Table 3: J3, Peripheral Expansion Connector

Pin	Signal	I/O	Description	Pin	Signal	I/O	Description
1	12V	Vcc	12V voltage supply pin	2	-12V	Vcc	-12V voltage supply pin
3	GND	Vss	System ground	4	GND	Vss	System ground
5	5V	Vcc	5V voltage supply pin	6	5V	Vcc	5V voltage supply pin
7	GND	Vss	System ground	8	GND	Vss	System ground
9	5V	Vcc	5V voltage supply pin	10	5V	Vcc	5V voltage supply pin
11	N/C	-	No connect	12	N/C	-	No connect
13	N/C	-	No connect	14	N/C	-	No connect
15	N/C	-	No connect	16	N/C	-	No connect
17	N/C	-	No connect	18	N/C	-	No connect
19	3.3V	Vcc	3.3V voltage supply pin	20	3.3V	Vcc	3.3V voltage supply pin
21	CLKX0	I/O	McBSP0 transmit clock	22	CLKS0	I	McBSP0 clock source
23	FSX0	I/O	McBSP0 transmit frame sync	24	DX0	O	McBSP0 transmit data
25	GND	Vss	System ground	26	GND	Vss	System ground
27	CLKR0	I/O	McBSP0 receive clock	28	N/C	-	No connect
29	FSR0	I/O	McBSP0 receive frame sync	30	DR0	I	McBSP0 receive data
31	GND	Vss	System ground	32	GND	Vss	System ground
33	CLKX1	I/O	McBSP1 transmit clock	34	CLKS1	I	McBSP1 clock source
35	FSX1	I/O	McBSP1 transmit frame sync	36	DX1	O	McBSP1 transmit data
37	GND	Vss	System ground	38	GND	Vss	System ground
39	CLKR1	I/O	McBSP1 receive clock	40	N/C	-	No connect
41	FSR1	I/O	McBSP1 receive frame sync	42	DR1	I	McBSP1 receive data
43	GND	Vss	System ground	44	GND	Vss	System ground
45	TOUT0	O	Timer 0 output	46	TINP0	I	Timer 0 input
47	N/C	-	No connect	48	EXT_INT5	I	External interrupt 5
49	TOUT1	O	Timer 1 output	50	TINP1	I	Timer 1 input
51	GND	Vss	System ground	52	GND	Vss	System ground
53	EXT_INT4	I	External interrupt 4	54	N/C	-	No connect
55	N/C	-	No connect	56	N/C	-	No connect
57	N/C	-	No connect	58	N/C	-	No connect
59	RESET	O	System reset	60	N/C	-	No connect
61	GND	Vss	System ground	62	GND	Vss	System ground
63	CNTL1	O	Daughtercard control 1	64	CNTL0	O	Daughtercard control
65	STAT1	I	Daughtercard status 1	66	STAT0	I	Daughtercard status
67	EXT_INT6	I	External interrupt 6	68	EXT_INT7	I	External interrupt 7
69	ACE3#	O	Chip enable 3	70	N/C	-	No connect
71	N/C	-	No connect	72	N/C	-	No connect
73	N/C	-	No connect	74	N/C	-	No connect
75	DC_DET#	Vss	System ground	76	GND	Vss	System ground
77	GND	Vss	System ground	78	ECL KOUT	O	EMIF Clock
79	GND	Vss	System ground	80	GND	Vss	System ground

3.3.3 J1, HPI Expansion Connector

Table 4: J1, HPI Expansion Connector

Pin	Signal	I/O	Description	Pin	Signal	I/O	Description
1	N/C	-	No connect	2	N/C	-	No connect
3	GND	Vss	System ground	4	HPI_RESETn	I	HPI reset input
5	CLKOUT3	O	Clock output3	6	N/C	-	No connect
7	GND	Vss	System ground	8	GND	Vss	System ground
9	HD1/AXR1[7]	I/O	HPI data 1	10	N/C	-	No connect
11	HD3/AMUTE1	I/O	HPI data 3	12	HD0/AXR1[4]	I/O	HPI data 0
13	HD5/AHCLKX1	I/O	HPI data 5	14	HD2/AFSX1	I/O	HPI data 2
15	HD7/GP0[3]	I/O	HPI data 7	16	HD4/GP0[0]	I/O	HPI data 4
17	GND	Vss	System ground	18	HD6/AHCLKR1	I/O	HPI data 6
19	HD8/GP0[8]	I/O	HPI data 8	20	GND	Vss	System ground
21	HD10/GP0[10]	I/O	HPI data 10	22	HD9/GP0[9]	I/O	HPI data 9
23	HD12/GP0[12]	I/O	HPI data 12	24	HD11/GP0[11]	I/O	HPI data 11
25	HD14/GP0[14]	I/O	HPI data 14	26	HD13/GP0[13]	I/O	HPI data 13
27	GND	Vss	System ground	28	HD15/GP0[15]	I/O	HPI data 15
29	HDS2z/AXR1[5]	I/O	Host data strobe 2	30	GND	Vss	System ground
31	GND	Vss	System ground	32	HASz/ACLKX1	I/O	Host address strobe
33	HDS1z/AXR1[6]	I/O	Host data strobe 1	34	GND	Vss	System ground
35	GND	Vss	System ground	36	HCNTL0/AXR1[3]	I/O	Host control 1
37	HCSz/AXR1[2]	I/O	Host chip select	38	GND	Vss	System ground
39	GND	Vss	System ground	40	HHWIL/AFSR1	I/O	Host half-word select
41	HCNTL1/AXR1[1]	I/O	Host control 1	42	GND	Vss	System ground
43	GND	Vss	System ground	44	HINTz/GP0[1]	I/O	Host interrupt
45	HRDYZ/ACLKR1	I/O	Host Ready	46	GND	Vss	System ground
47	GND	Vss	System ground	48	N/C	-	No connect
49	HR/Wz/AXR1[0]	I/O	Host R/W strobe	50	N/C	-	No connect
51	N/C	-	No connect	52	N/C	-	No connect
53	N/C	-	No connect	54	N/C	-	No connect
55	N/C	-	No connect	56	GND	Vss	System ground
57	N/C	-	No connect	58	N/C	-	No connect
59	N/C	-	No connect	60	N/C	-	No connect
61	GND	Vss	System ground	62	N/C	-	No connect
63	N/C	-	No connect	64	N/C	-	No connect
65	N/C	-	No connect	66	N/C	-	No connect
67	N/C	-	No connect	68	SCL0	I/O	I2C0 Clock
69	N/C	-	No connect	70	GND	Vss	System ground
71	GND	Vss	System ground	72	SDA0	I/O	I2C0 Data
73	N/C	-	No connect	74	GND	Vss	System ground
75	GND	Vss	System ground	76	N/C	-	No connect
77	N/C	-	No connect	78	GND	Vss	System ground
79	GND	Vss	System ground	80	CLKOUT2/GP0[2]	I/O	GP I/O 0 bit 2

3.4 Audio Connectors

The C6713 DSK has 4 audio connectors. They are described in the following sections.

3.4.1 J301, Microphone Connector

The input is a 3.5 mm. stereo jack. Both inputs are connected to the microphone so it is monaural. The signals on the plug are shown in the figure below.

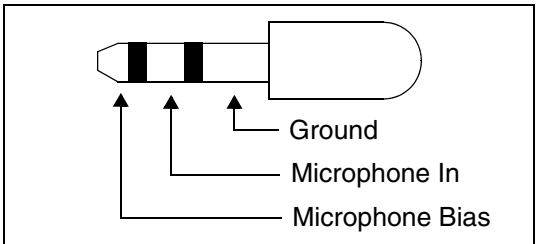


Figure 3-2, Microphone Stereo Jack

3.4.2 J303, Audio Line In Connector

The audio line in is a stereo input. The input connector is a 3.5 mm stereo jack. The signals on the mating plug are shown in the figure below.

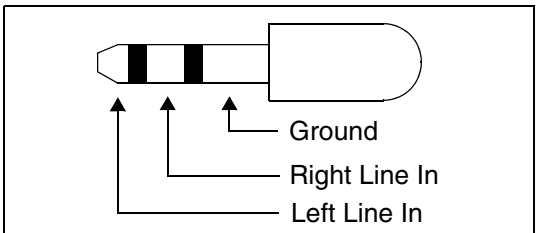


Figure 3-3, Audio Line In Stereo Jack

3.4.3 J304, Audio Line Out Connector

The audio line out is a stereo output. The output connector is a 3.5 mm stereo jack. The signals on the mating plug are shown in the figure below.

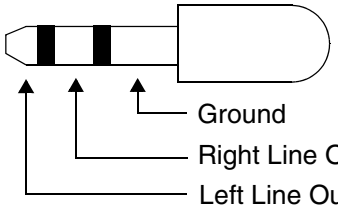


Figure 3-4, Audio Line Out Stereo Jack

3.4.4 J303, Headphone Connector

Connector J4 is a headphone/speaker jack. It can drive standard headphones or a high impedance speaker directly. The standard 3.5 mm jack is shown in the figure below.

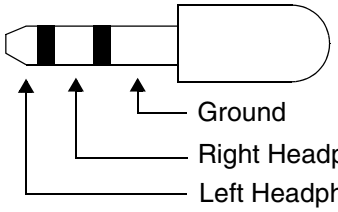


Figure 3-5, Headphone Jack

3.5 Power Connectors

The C6713 DSK has 2 power connectors. They are described in the following sections.

3.5.1 J5, +5 Volt Connector

Power (+5 volts) is brought onto the TMS320C6713 DSK via the J5 connector. The connector has an outside diameter of 5.5 mm. and an inside diameter of 2.5 mm. The A diagram of J5 is shown below.

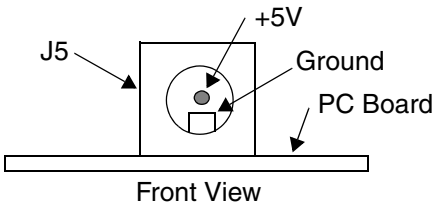


Figure 3-6, TMS320C6713 DSK Power Connector

3.5.2 J6, Optional Power Connector

Connector J6 is an optional power connector. It will operate with the standard personal computer power supply. To populate this connector use a Molex #15-24-4041. The table below shows the voltages on the respective pins.

Table 5: J6, Optional Power Connector

Pin #	Voltage Level
1	+12 Volts
2	-12 Volts
3	Ground
4	+5 Volts

WARNING !
Do not plug into J5 and J6 at the same time.

3.6 Miscellaneous Connectors

The C6713 DSK has 3 additional connectors to aid the user in developing with this product. They are described in the following sections.

3.6.1 J201, USB Connector

Connector J201 provides a Universal Serial Bus (USB) Interface to the embedded JTAG emulation logic on the DSK. This allows for code development and debug without the use of an external emulator. The signals on this connector are shown in the below.

Table 6: J201, USB Connector

Pin #	USB Signal Name
1	USBVdd
2	D+
3	D-
4	USB Vss
5	Shield
6	Shield

3.6.2 J8, External JTAG Connector

The TMS320C6713 DSK is supplied with a 14 pin header interface, J8. This is the standard interface used by JTAG emulators to interface to Texas Instruments DSPs. The pinout for the connector is shown in the figure below.

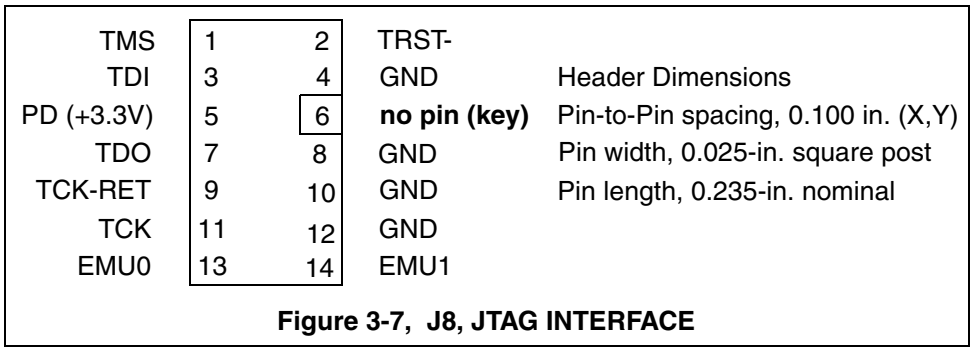


Figure 3-7, J8, JTAG INTERFACE

3.6.3 JP3, PLD Programming Connector

This connector interfaces to the Altera CPLD, U12. It is used in the in the factory for the programming of the CPLD. This connector is not intended to be used outside the factory.

3.7 System LEDs

TheTMS320C6713 DSK has four system light emitting diodes (LEDs). These LEDs indicate various conditions on the DSK. These function of each LED is shown in the table below.

Table 7: System LEDs

Reference Designator	Color	Function	On Signal State
D4	Green	USB Emulation in use. When External JTAG Emulator is used this LED is off.	1
D3	Green	+5 Volt present	1
D6	Orange	RESET Active	1
DS201	Green	USB Active, Blinks during USB data transfer	1

3.8 Reset Switch

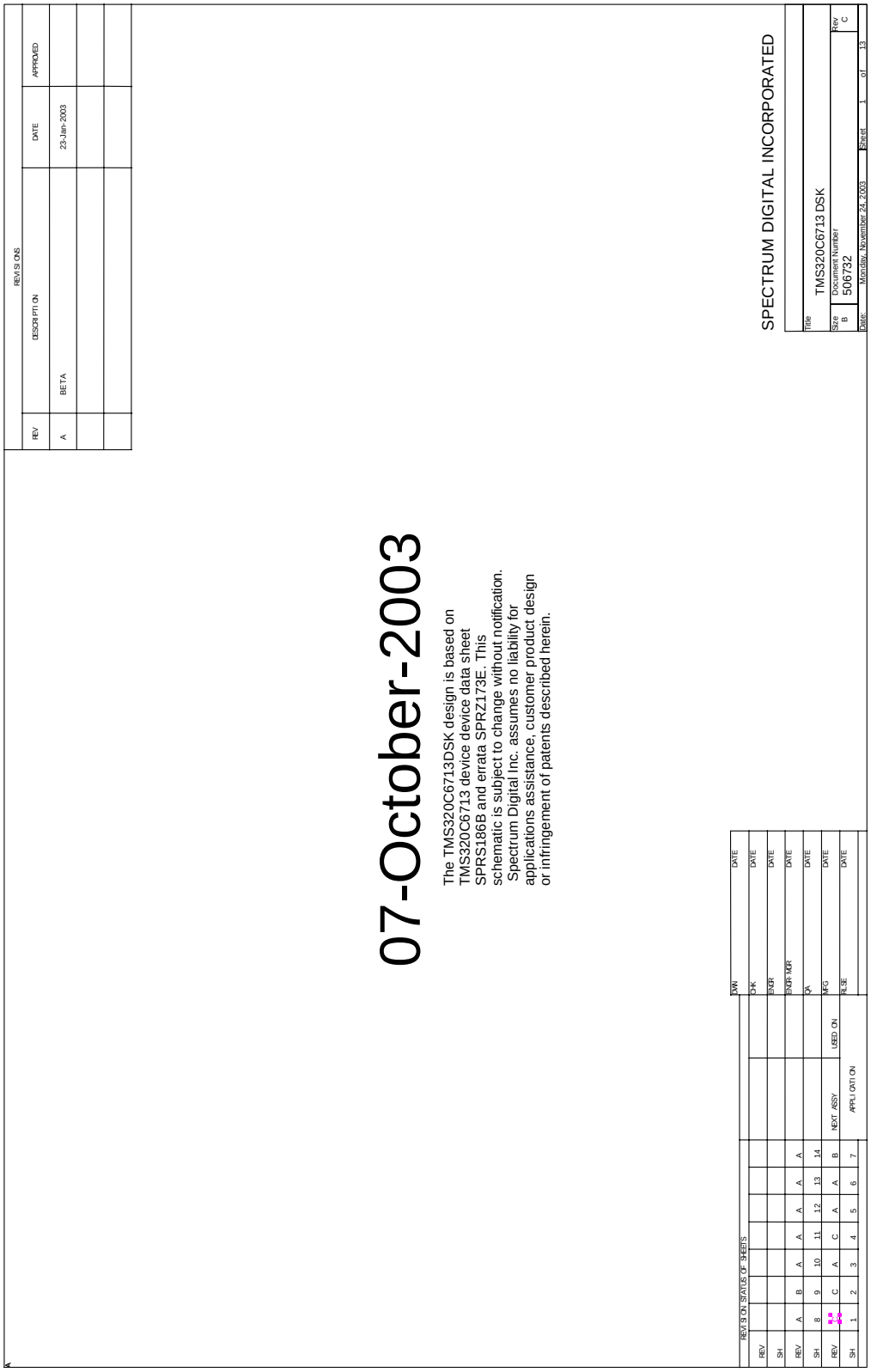
There are three resets on the TMS320C6713 DSK. The first reset is the power on reset. This circuit waits until power is within the specified range before releasing the power on reset pin to the TMS320C6713.

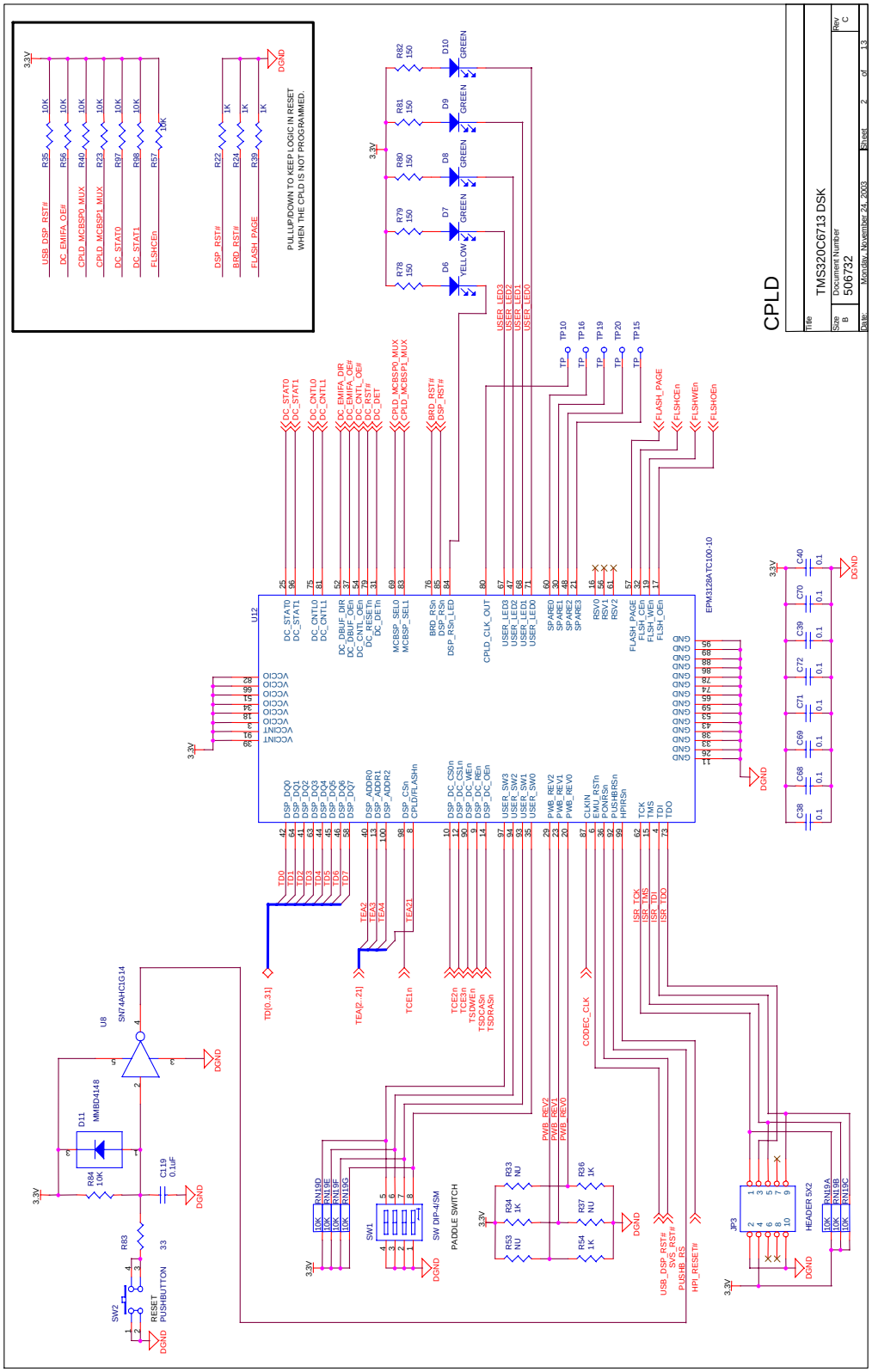
External sources which control the reset are push button SW2, and the on board embedded USB JTAG emulator.

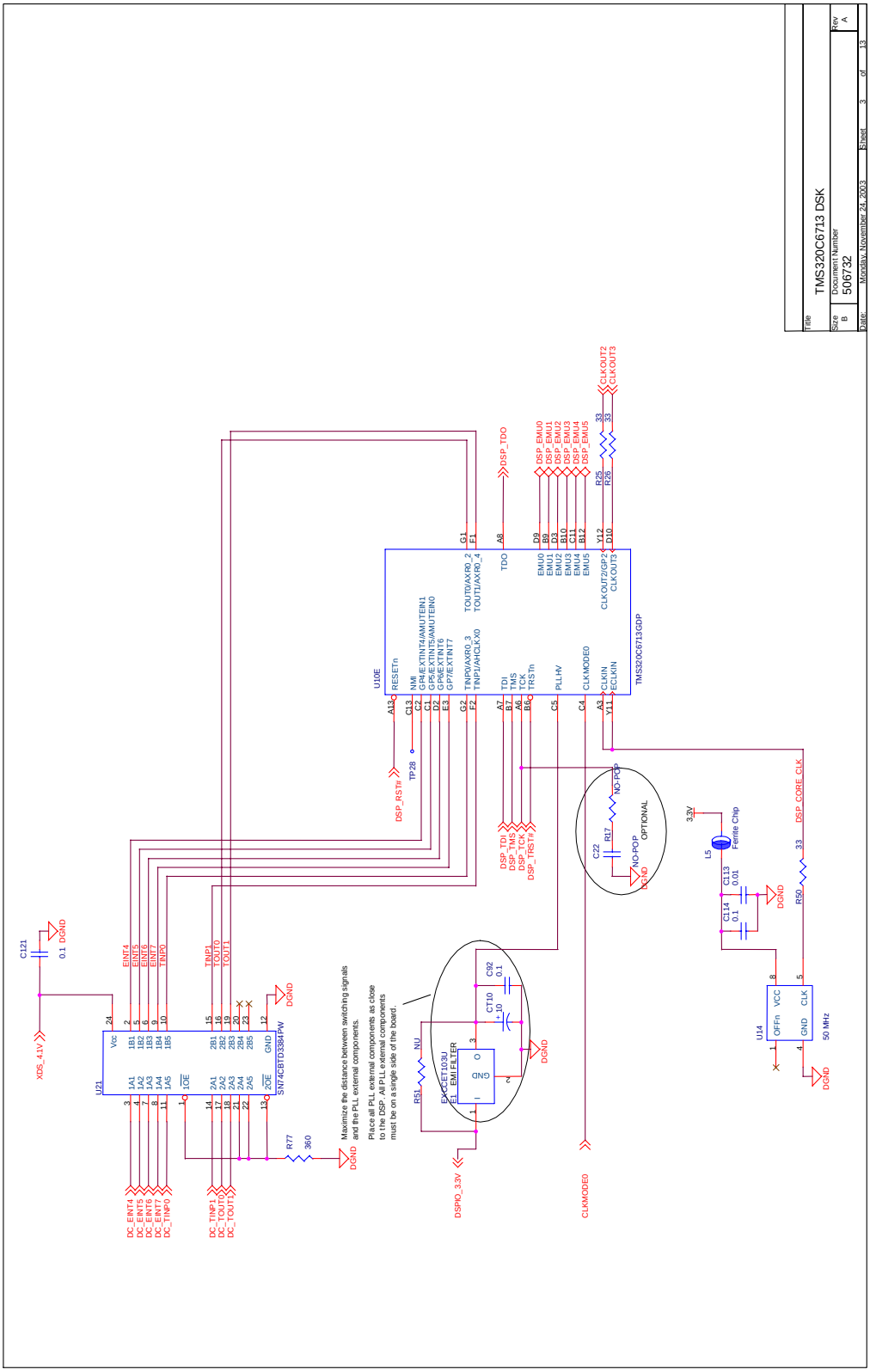
Appendix A

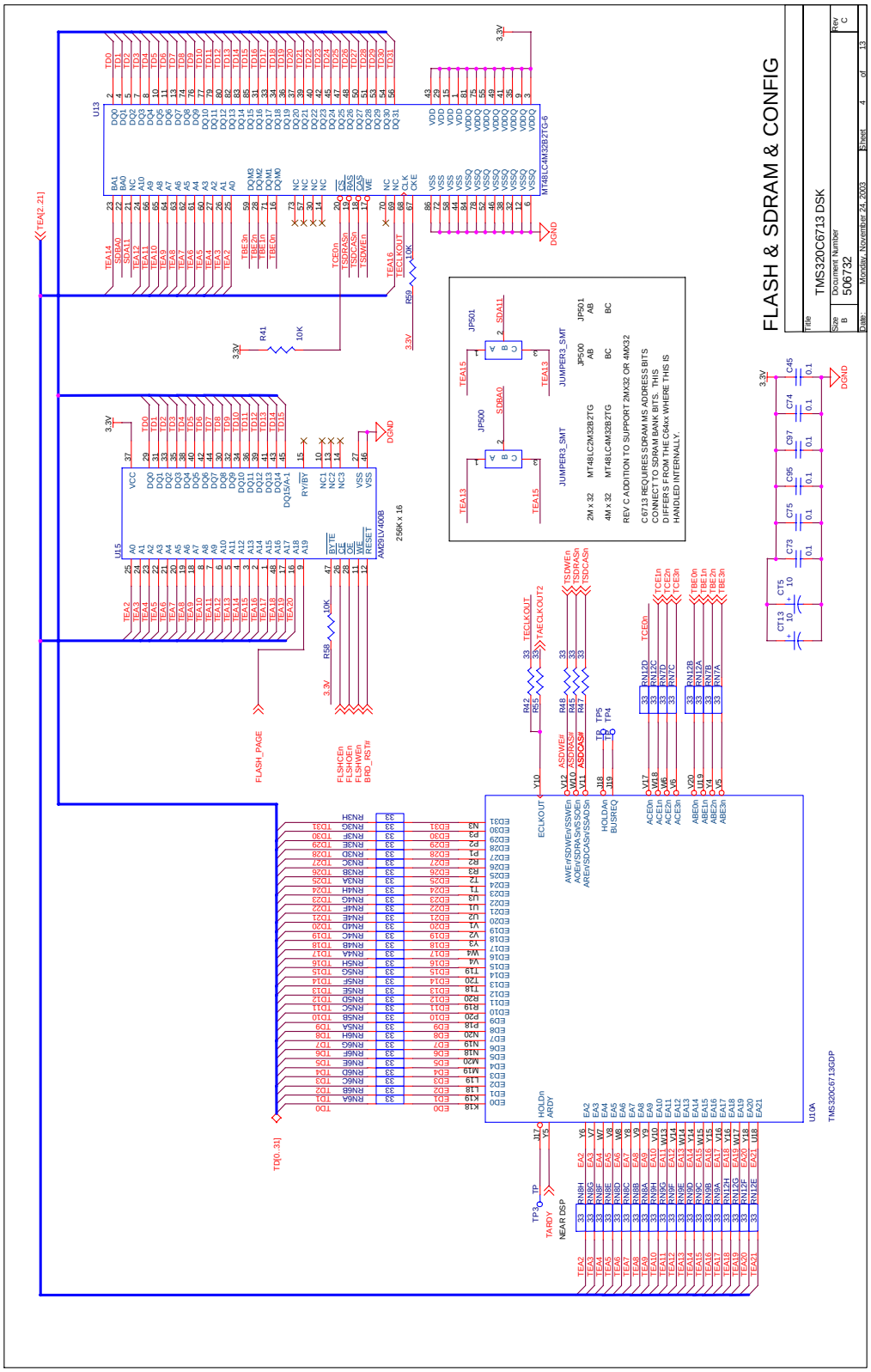
Schematics

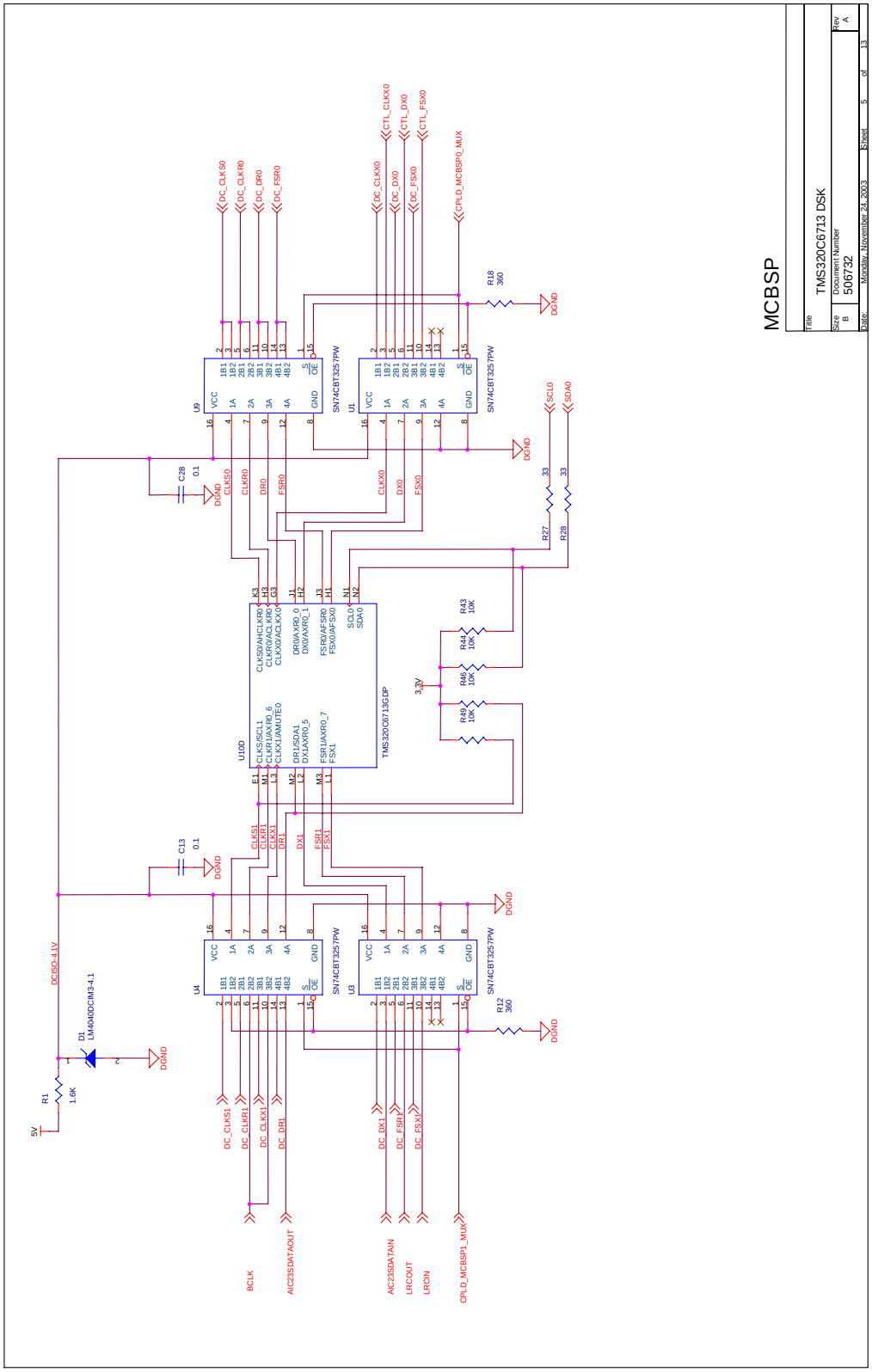
This appendix contains the schematics for the TMS320C6713 DSK. Board components with designators over 200 (e.g. DS201, R211) are part of Spectrum Digital's embedded JTAG emulator and are not included in these schematics.

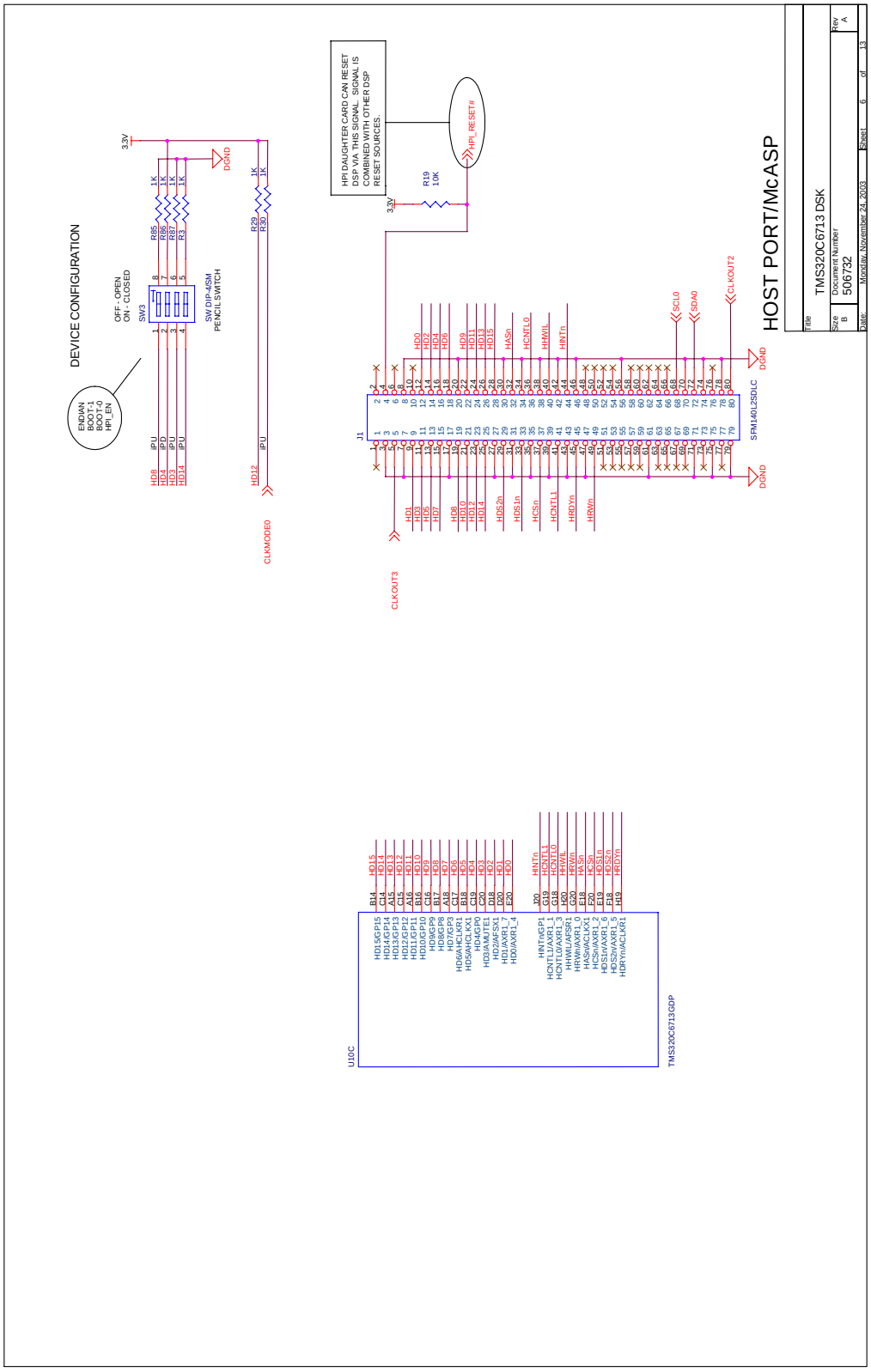

REVISIONS
REV
DESCRIPTION
DATE
APPROVED
A
BETA
23-Jan-2003
07-October-2003
The TMS320C6713DSK design is based on TMS320C6713 device device data sheet SPRS186B and errata SPRZ173E. This schematic is subject to change without notification. Spectrum Digital Inc. assumes no liability for applications assistance, customer product design or infringement of patents described herein.
SPECTRUM DIGITAL INCORPORATED
Title
TMS320C6713 DSK
Size
B
Document Number
506732
Rev
C
Date:
Monday, November 24, 2003
Sheet
1
of
13

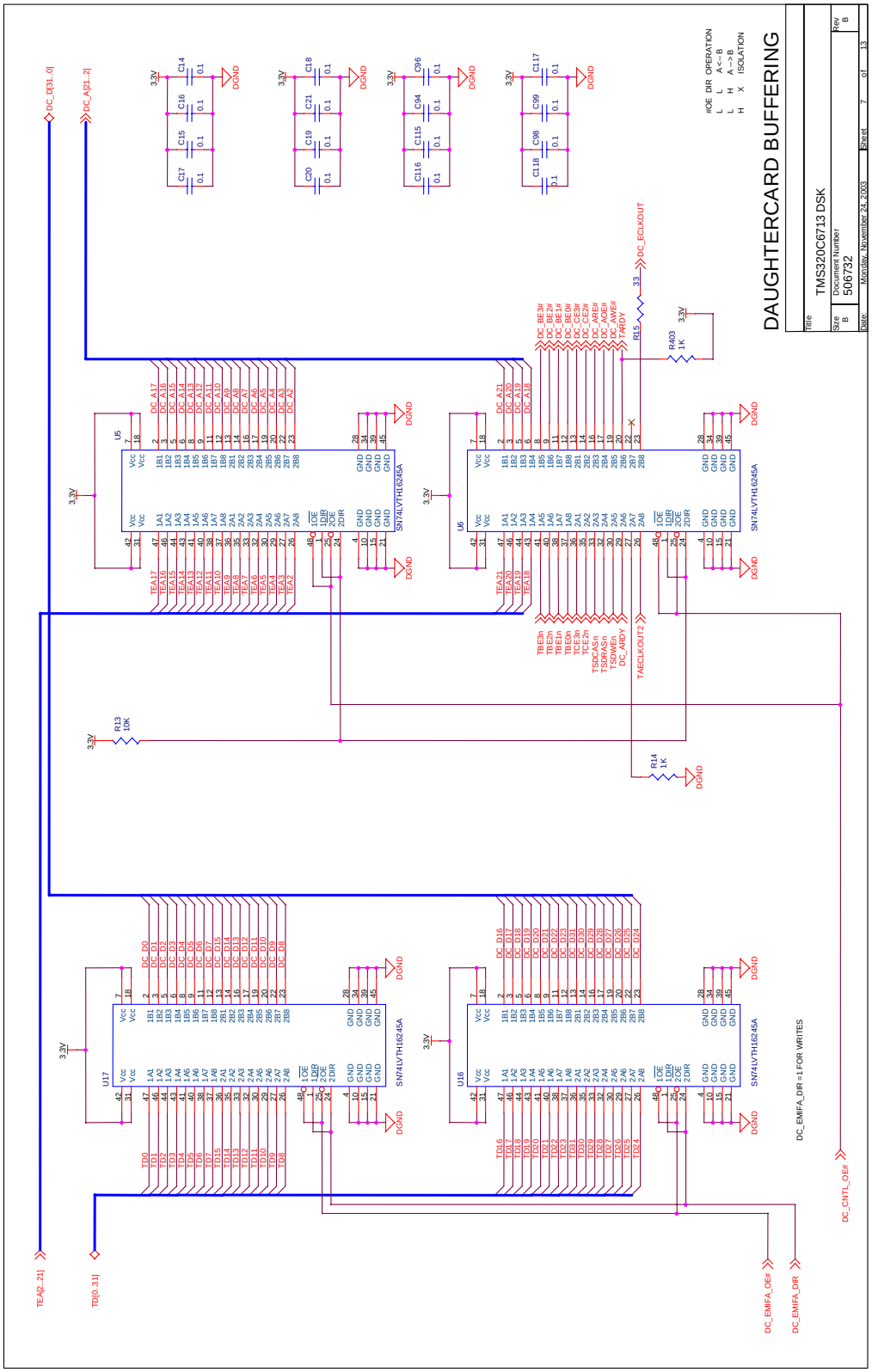

CPLD
TMS320C6713 DSK
506732
PULLUP/DOWN TO KEEP LOGIC IN RESET
WHEN THE CPLD IS NOT PROGRAMMED.
EPM3128A TC100-10
SW DIP-4/SM
PADDLE SWITCH
SN74AHC1G14
MMBD4148
RESET
PUSHBUTTON
HEADER 5X2

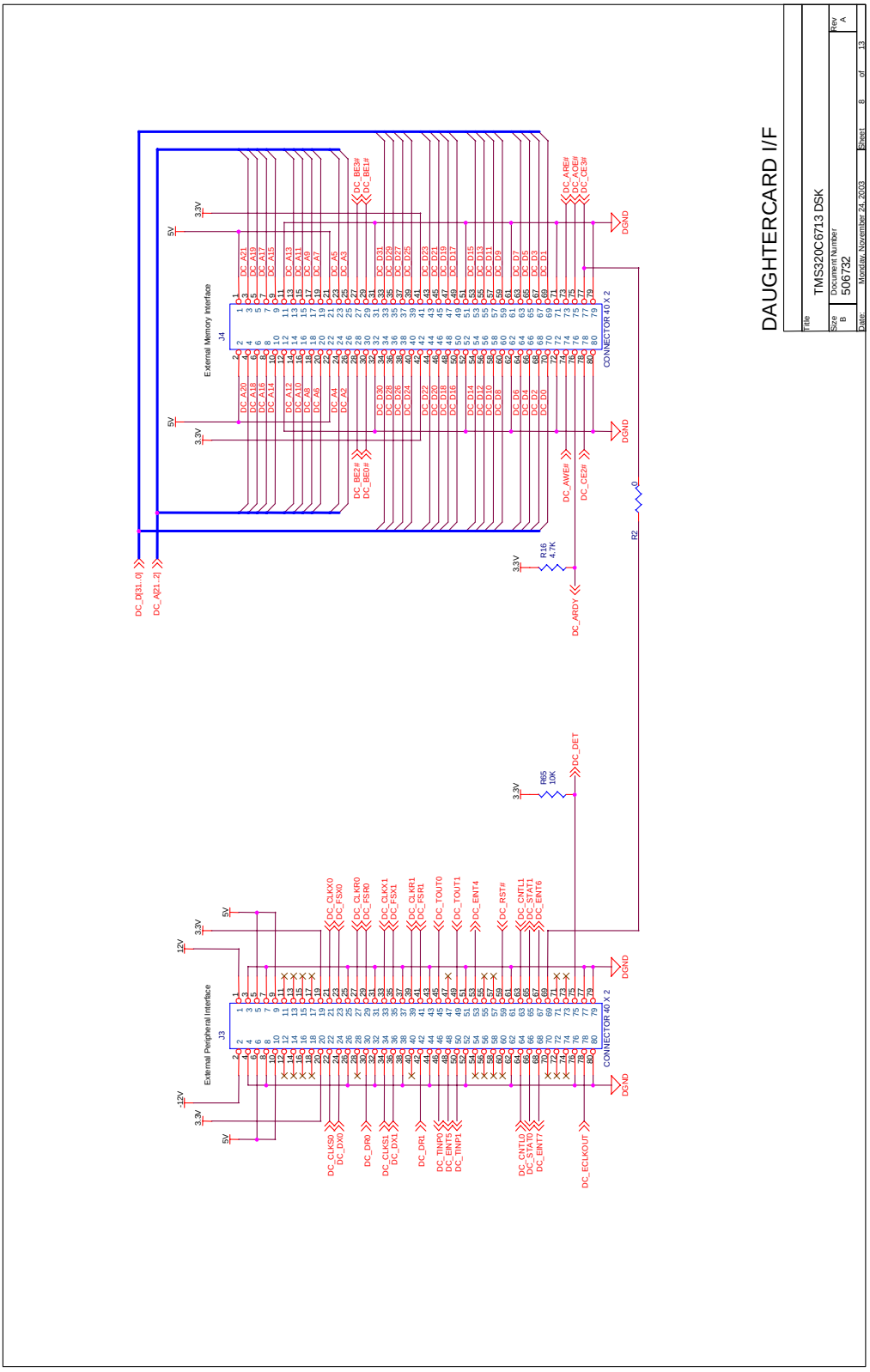

TMS320C6713 DSK
Document Number
506732
Monday, November 24, 2003
Sheet 3 of 13
Rev A
U10E
TMS320C6713GDP
RESETn
NMI
GP4/EXTINT4/AMUTEIN1
GP5/EXTINT5/AMUTEIN0
GP6/EXTINT6
GP7/EXTINT7
TINP0/AXR0_3
TINP1/AHCLKX0
TOUT0/AXR0_2
TOUT1/AXR0_4
TDI
TMS
TCK
TRSTn
PLLHV
CLKMODE0
CLKIN
ECLKIN
CLKOUT2/GP2
CLKOUT3
TDO
EMU0
EMU1
EMU2
EMU3
EMU4
EMU5
DSP_RSTn
DSP_TDO
DSP_TDI
DSP_TMS
DSP_TCK
DSP_TRSTn
DSP_EMU0
DSP_EMU1
DSP_EMU2
DSP_EMU3
DSP_EMU4
DSP_EMU5
CLKOUT2
CLKOUT3
DSP_CORE_CLK
U21
SN74CBT03384PW
XDS_4.1V
DC_EINT4
DC_EINT5
DC_EINT6
DC_EINT7
DC_TINP0
DC_TINP1
DC_TOUT0
DC_TOUT1
EINT4
EINT5
EINT6
EINT7
TINP0
TINP1
TOUT0
TOUT1
R77
360
R51
NU
EXCET103U
E1_EMULATOR
C92
0.1
CT10
+ 10
DSPIO_3.3V
CLKMODE0
C22
R17
NO-POP
OPTIONAL
3.3V
L5
Ferrite Chip
C113
0.01
C114
0.1
R50
33
U14
OSC
50 MHz
C121
0.1
DGND
Maximize the distance between switching signals and the PLL external components.
Place all PLL external components as close to the DSP. All PLL external components must be on a single side of the board.

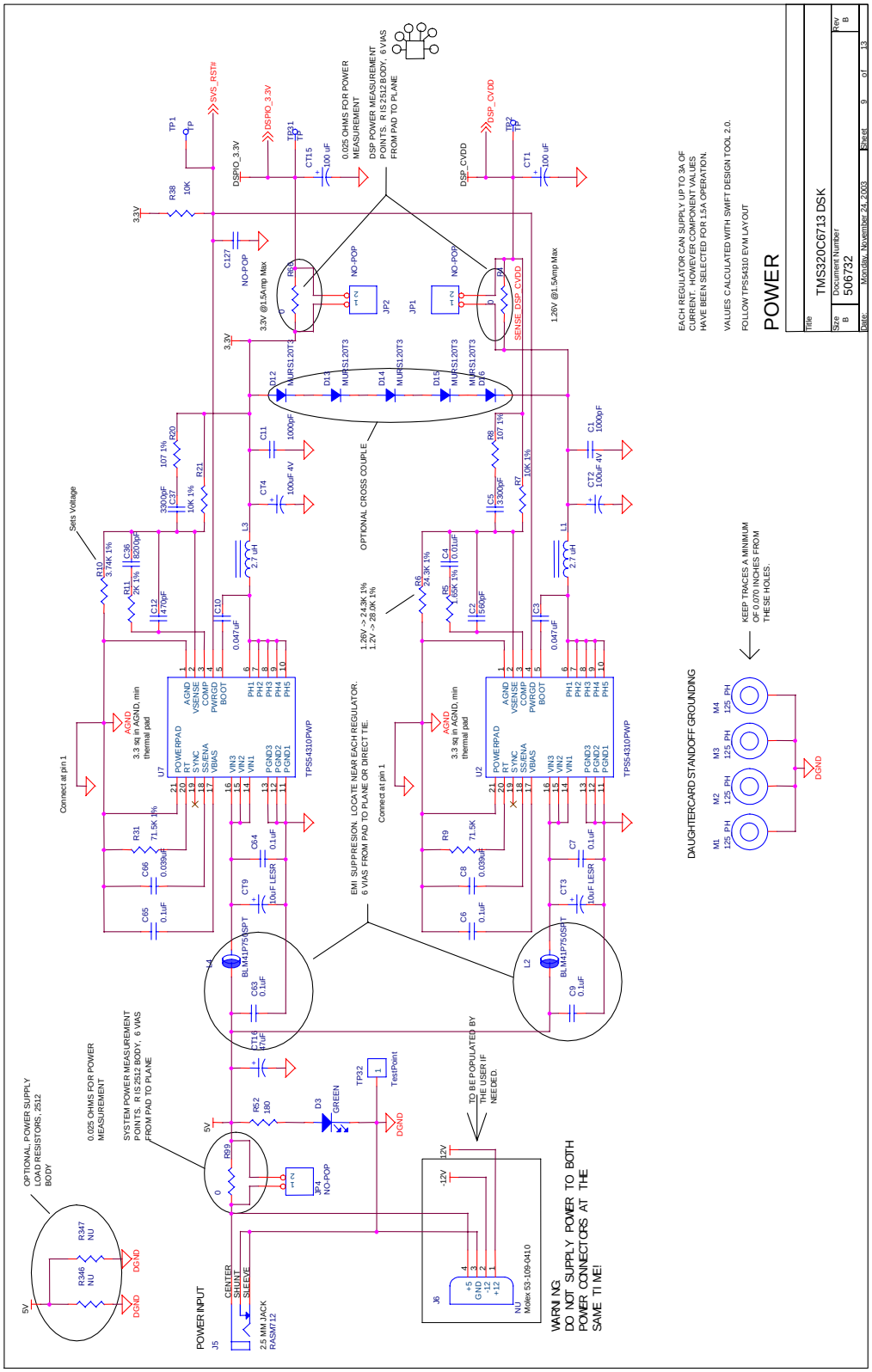

FLASH & SDRAM & CONFIG
TMS320C6713 DSK
506732
Monday, November 24, 2003
C
B
4
13
U13
MT48LC4M32B2TG-6
U15
AM29LV400B
256K x 16
U10A
TMS320C6713GDP
JUMPER3_SMT
JP500
JP501
SDA11
SDRA0
TEA13
TEA15
2M x 32 MT48LC2M32B2TG AB
4M x 32 MT48LC4M32B2TG BC
REV C ADDITION TO SUPPORT 2MX32 OR 4MX32
C6713 REQUIRES SDRAM MS ADDRESS BITS
CONNECT TO SDRAM BANK BITS. THIS
DIFFERS FROM THE C64xx WHERE THIS IS
HANDLED INTERNALLY.
FLASH_PAGE
TEA[2..21]
TD[0..31]
TCE0n
TSDRASn
TSDCASn
TSDWEn
TECLKOUT
ECLKOUT
HOLDn
ARDY
TARDY
NEAR DSP
3.3V
DGND
10K
R41
R59
R58
R42
R55
R48
R45
R47
RN12D
RN12C
RN7C
RN12B
RN12A
RN7B
RN7A
TBE0n
TBE1n
TBE2n
TBE3n
TCE1n
TCE2n
TCE3n
C45
C74
C37
C96
C75
C73
CT5
CT13
0.1
10

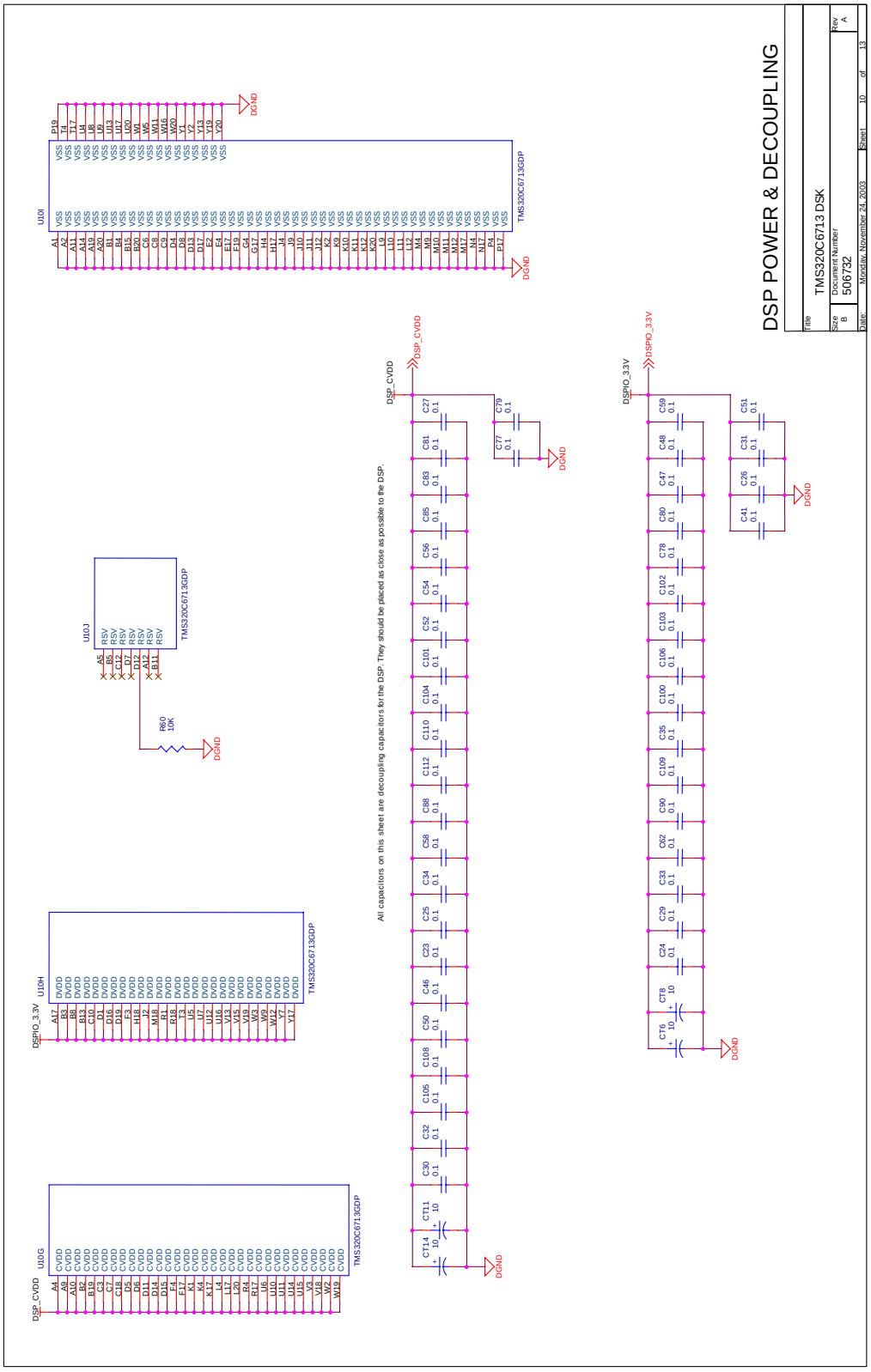

MCBSP
Title
TMS320C6713 DSK
Size
B
Document Number
506732
Rev
A
Date:
Monday, November 24, 2003
Sheet
5
of
13
SN74CBT3257PW
TMS320C6713GDP
DGND
3.3V
5V
DC_CLKS0
DC_CLKR0
DC_DR0
DC_FSR0
CTL_CLKX0
CTL_DX0
CTL_FSX0
CPLD_MCBSP0_MUX
DC_CLKS1
DC_CLKR1
DC_CLKX1
DC_DR1
DC_DX1
DC_FSR1
DC_FSX1
CPLD_MCBSP1_MUX
BCLK
AIC23SDATAOUT
AIC23SDATAIN
LRCOUT
LRCIN
SCL0
SDA0

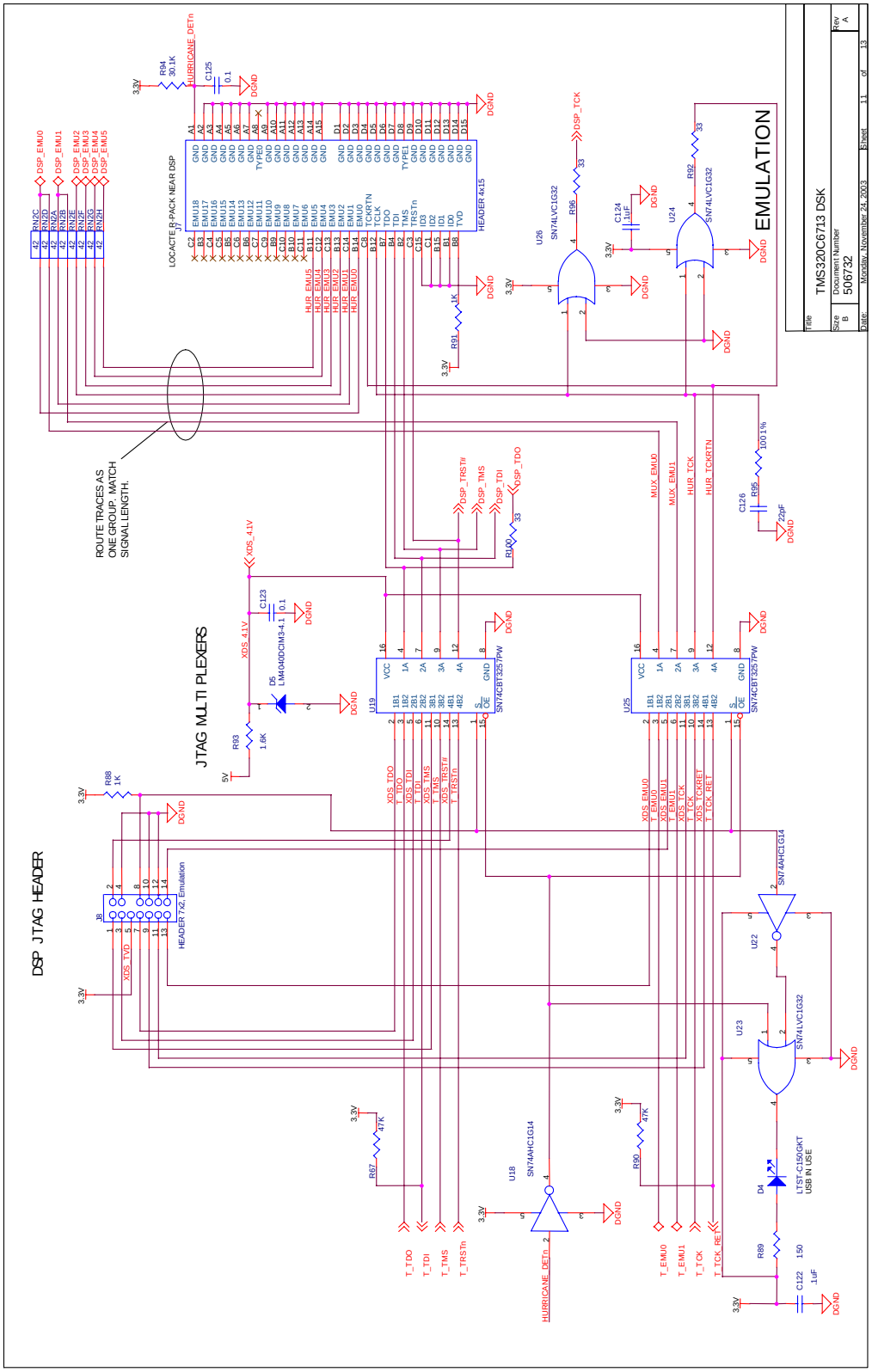

HOST PORT/McASP
TMS320C6713 DSK
Document Number
506732
Monday, November 24, 2003
DEVICE CONFIGURATION
ENDIAN
BOOT-1
BOOT-0
HPI_EN
OFF - OPEN
ON - CLOSED
SW3
SW DIP-4/SM
PENCIL SWITCH
HPI DAUGHTER CARD CAN RESET
DSP VIA THIS SIGNAL. SIGNAL IS
COMBINED WITH OTHER DSP
RESET SOURCES.
HPI_RESET#
CLKOUT3
CLKMODE0
CLKOUT2
SCL0
SDA0
SFM140L2SDLC
TMS320C6713GDP
U10C
DGND
3.3V

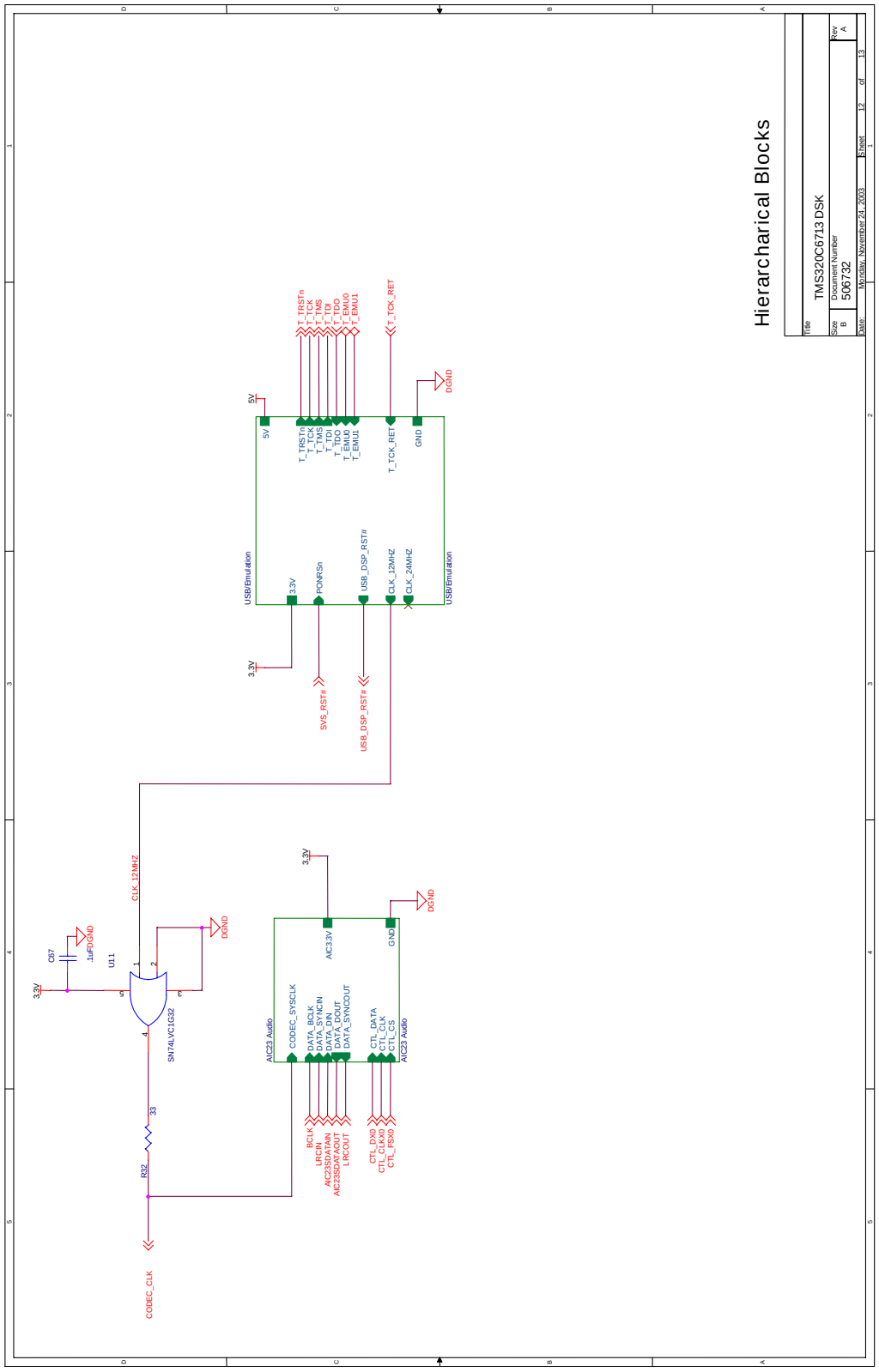

DAUGHTERCARD BUFFERING
TMS320C6713 DSK
Document Number
506732
Monday, November 24, 2003
DC_EMIFA_DIR =1 FOR WRITES
SN74LVTH16245A


DAUGHTERCARD I/F
Title
TMS320C6713 DSK
Size
B
Document Number
506732
Rev
A
Date:
Monday, November 24, 2003
Sheet
8
of
13
External Memory Interface
J4
CONNECTOR 40 X 2
External Peripheral Interface
J3
CONNECTOR 40 X 2
R16
4.7K
R65
10K
R2
0
DC_DET
DC_ARDY
DC_CE#
DC_AWE#
DC_BE#
DC_BE0#
DC_BE1#
DC_BE2#
DC_BE3#
DC_D[31..0]
DC_A[21..2]
DGND
5V
3.3V
12V


POWER
TMS320C6713 DSK
Document Number
506732
Rev B
Monday, November 24, 2003
Sheet 9 of 13
EACH REGULATOR CAN SUPPLY UP TO 3A OF CURRENT. HOWEVER COMPONENT VALUES HAVE BEEN SELECTED FOR 1.5A OPERATION.
VALUES CALCULATED WITH SWIFT DESIGN TOOL 2.0.
FOLLOW TPS54310 EVM LAYOUT
KEEP TRACES A MINIMUM OF 0.070 INCHES FROM THESE HOLES.
DAUGHTERCARD STANDOFF GROUNDING
M1 125 PH
M2 125 PH
M3 125 PH
M4 125 PH
DGND
SYS_RST#
DSPIO_3.3V
DSP_CVDD
3.3V
5V
12V
-12V
TP1
TP21
TP2
TP32
TestPoint
R38 10K
CT15 100 uF
CT1 100 uF
0.025 OHMS FOR POWER MEASUREMENT
DSP POWER MEASUREMENT POINTS. R IS 2512 BODY, 6 VIAS FROM PAD TO PLANE
C127 NO-POP
R99 0
JP4 NO-POP
JP2 NO-POP
JP1 NO-POP
R4 0
SENSE_DSP_CVDD
3.3V @1.5Amp Max
1.26V @1.5Amp Max
D12 MURS120T3
D13 MURS120T3
D14 MURS120T3
D15 MURS120T3
D16 MURS120T3
OPTIONAL CROSS COUPLE
C11 1000pF
CT4 100uF 4V
C1 1000pF
CT2 100uF 4V
L3 2.7 uH
L1 2.7 uH
Sets Voltage
R10 3.74K 1%
C36 1800pF
R11 2K 1%
C12 470pF
C37 3300pF
R20 107 1%
R21 10K 1%
R8 107 1%
R7 10K 1%
C5 3300pF
R6 24.3K 1%
R5 69K 1%
C4 0.01uF
C2 560pF
1.26V -> 2.43K 1%
1.2V -> 28.0K 1%
C10 0.047uF
C3 0.047uF
U7
U2
TPS54310PWP
POWERPAD
AGND
3.3 sq in AGND, min thermal pad
Connect at pin 1
R31 71.5K 1%
C66 0.039uF
C65 0.1uF
R9 71.5K
C8 0.039uF
C6 0.1uF
C54 0.1uF
C7 0.1uF
CT9 10uF LESR
CT3 10uF LESR
EMI SUPPRESION. LOCATE NEAR EACH REGULATOR. 6 VIAS FROM PAD TO PLANE OR DIRECT TIE.
L4 BLM41P750SPT
L2 BLM41P750SPT
C55 0.1uF
C9 0.1uF
CT14 47uF
R52 180
D3 GREEN
TO BE POPULATED BY THE USER IF NEEDED.
SYSTEM POWER MEASUREMENT POINTS. R IS 2512 BODY, 6 VIAS FROM PAD TO PLANE
0.025 OHMS FOR POWER MEASUREMENT
OPTIONAL POWER SUPPLY LOAD RESISTORS, 2512 BODY
R347 NU
R346 NU
POWER INPUT
J5
CENTER
SHUNT
SLEEVE
2.5 MM JACK RASM712
J6
+5
GND
-12
+12
NU
Molex 53-109-0410
WARNING
DO NOT SUPPLY POWER TO BOTH POWER CONNECTORS AT THE SAME TIME!


DSP POWER & DECOUPLING
Title
TMS320C6713 DSK
Size
B
Document Number
506732
Rev
A
Date:
Monday, November 24, 2003
Sheet
10
of
13
All capacitors on this sheet are decoupling capacitors for the DSP. They should be placed as close as possible to the DSP.
DSP_CVDD
DSPIO_3.3V
DGND
U10I
U10J
U10H
U10G
TMS320C6713GDP
R60
10K
CT14
CT11
CT6
CT8
10

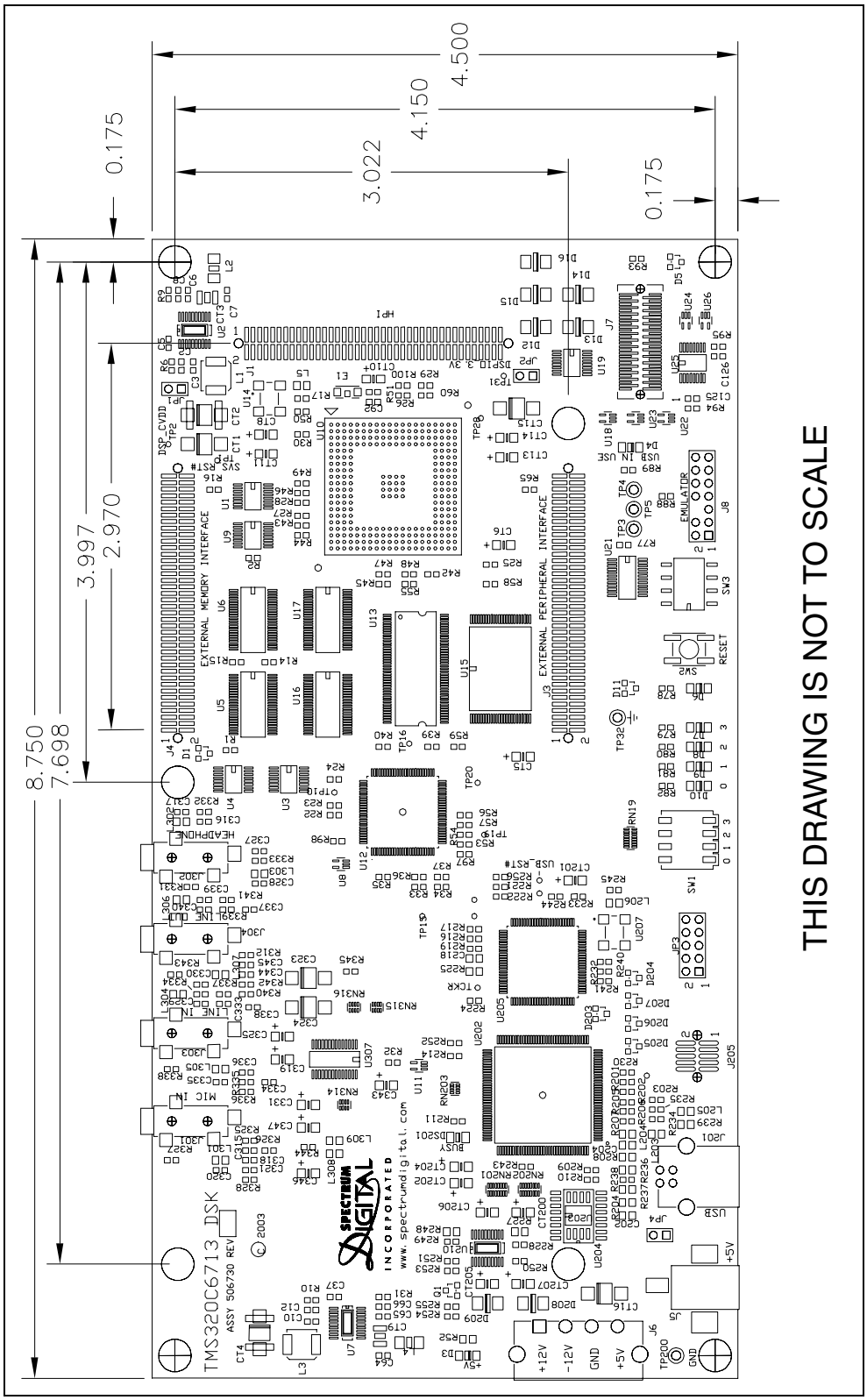

DSP JTAG HEADER
JTAG MULTIPLEXERS
EMULATION
TMS320C6713 DSK
Document Number
506732
Monday, November 24, 2003
ROUTE TRACES AS ONE GROUP. MATCH SIGNAL LENGTH.
LOCACTE 8-PACK NEAR DSP
HEADER 4x15
HEADER 7X2, Emulation
SN74CBT3257PW
SN74AHC1G14
SN74LVC1G32
LTST-C150GKT
USB IN USE
LM4040DCIM3-4.1
HURRICANE_DETn
DSP_TRST#
DSP_TMS
DSP_TDI
DSP_TDO
DSP_TCK
MUX_EMU0
MUX_EMU1
HUR_TCK
HUR_TCKRTN
XDS_4.1V
5V
3.3V
DGND


Hierarcharical Blocks
Title
TMS320C6713 DSK
Size
B
Document Number
506732
Rev
A
Date:
Monday, November 24, 2003
Sheet
12
of
13
USB/Emulation
5V
T_TRSTn
T_TCK
T_TMS
T_TDI
T_TDO
T_EMU0
T_EMU1
T_TCK_RET
GND
DGND
3.3V
PONRSn
USB_DSP_RST#
CLK_12MHZ
CLK_24MHZ
SYS_RST#
AIC23 Audio
CODEC_SYSCLK
DATA_BCLK
DATA_SYNCIN
DATA_DIN
DATA_DOUT
DATA_SYNCOUT
AIC3.3V
CTL_DATA
CTL_CLK
CTL_CS
BCLK
LRCIN
AIC23SDATAIN
AIC23SDATAOUT
LRCOUT
CTL_DIO
CTL_CLKIO
CTL_RSIO
C67
.1uF
U1.1
SN74LVC1G32
R32
33
CODEC_CLK

Appendix B

Mechanical Information

This appendix contains the mechanical information about the TMS320C6713 DSK produced by Spectrum Digital.


4.500
4.150
3.022
0.175
0.175
8.750
7.698
3.997
2.970
THIS DRAWING IS NOT TO SCALE
TMS320C6713 DSK
ASSY 506730 REV
© 2003
SPECTRUM DIGITAL INCORPORATED
www.spectrumdigital.com
EXTERNAL MEMORY INTERFACE
EXTERNAL PERIPHERAL INTERFACE
EMULATOR
RESET
HEADPHONE
MIC IN
LINE IN
USB
+5V
+12V
-12V
GND
+5V
GND

Printed in U.S.A., November 2003
506735-0001 Rev. B

ANEXO 3
DOCUMENTACIÓN FPC1010 DAUGHTER CARD

SPECTRUM DIGITAL INCORPORATED

FPC1010 Finger Print Sensor Daughter Card

Technical Reference

2003 DSP Development Systems

FPC1010 Finger Print Sensor Daughter Card Technical Reference

506865-0001 Rev. A
July 2003

SPECTRUM DIGITAL, INC.
12502 Exchange Drive, Suite 440 Stafford, TX. 77477
Tel: 281.494.4505 Fax: 281.494.5310
sales@spectrumdigital.com www.spectrumdigital.com

IMPORTANT NOTICE

Spectrum Digital, Inc. reserves the right to make changes to its products or to discontinue any 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.

Spectrum Digital, Inc. warrants performance of its products and related software to current specifications in accordance with Spectrum Digital's standard warranty. Testing and other quality control techniques are utilized to the extent deemed necessary to support this warranty.

Please be aware that the products described herein are not intended for use in life-support appliances, devices, or systems. Spectrum Digital does not warrant nor is liable for the product described herein to be used in other than a laboratory development environment. Use in any other environment voids the warranty.

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

WARNING

This equipment is intended for use in a laboratory test environment only. It generates, uses, and can radiate radio frequency energy and has not been tested for compliance with the limits of computing devices pursuant to subpart J of part 15 of FCC rules, which are designed to provide reasonable protection against radio frequency interference. Operation of this equipment in other environments may cause interference with radio communications, in which case the user at his own expense will be required to take whatever measures may be required to correct this interference.

Contents

About This Manual

This document describes the operations of the FPC1010 Finger Print Sensor Daughter Card. The card is designed to be used with EVMs and DSKs that meet the daughter card specification defined by Texas Instruments for it DSPs and microcontrollers.

The FPC1010 Finger Print Sensor Daughter Card with a DSK or EVM is a table top development system that allows engineers to evaluate finger print verification techniques and familiarize themselves with finger print basics.

Notational Conventions

This document uses the following conventions.

The FPC1010 Finger Print Sensor Daughter Card will sometimes be referred to as the Finger Print Sensor.

Program listings, program examples, and interactive displays are shown is a special italic typeface. Here is a sample program listing.

equations
!rd = rw &! strb;

Information About Cautions

This book may contain cautions.
This is an example of a caution statement.
A caution statement describes a situation that could potentially damage your software, or hardware, or other equipment. The information in a caution is provided for your protection. Please read each caution carefully.

Table 1: Manual History

Revision	History
A	Initial Release

Chapter 1

Introduction to the FPC1010 Finger Print Sensor Daughter Card

This chapter provides you with a description of the FPC1010 Finger Print Sensor Daughter Card along with the key features and a block diagram of the card.

Topic **Page**

1.0 Overview of the FPC1010 Finger Print Sensor Daughter Card

The FPC1010 Finger Print Sensor Daughter Card is a daughter card designed to be used with a DSK or EVM that meet TI's daughter card specification for the evaluation of finger print verification technologies.

1.1 Key Features of the FPC1010 Finger Print Sensor Daughter Card

The FPC1010 Finger Print Sensor Daughter Card has the following features:

- 200 x 152 Pixel Capacitive Finger Print Sensor
- 3.3 volt operation
- Mode selection switch
- User selection switch
- Six status LEDs

1.2 Functional Overview of the FPC1010 Finger Print Sensor Daughter Card

Figure 1-1 shows a block diagram of the basic configuration for the FPC1010 Finger Print Sensor Daughter Card.

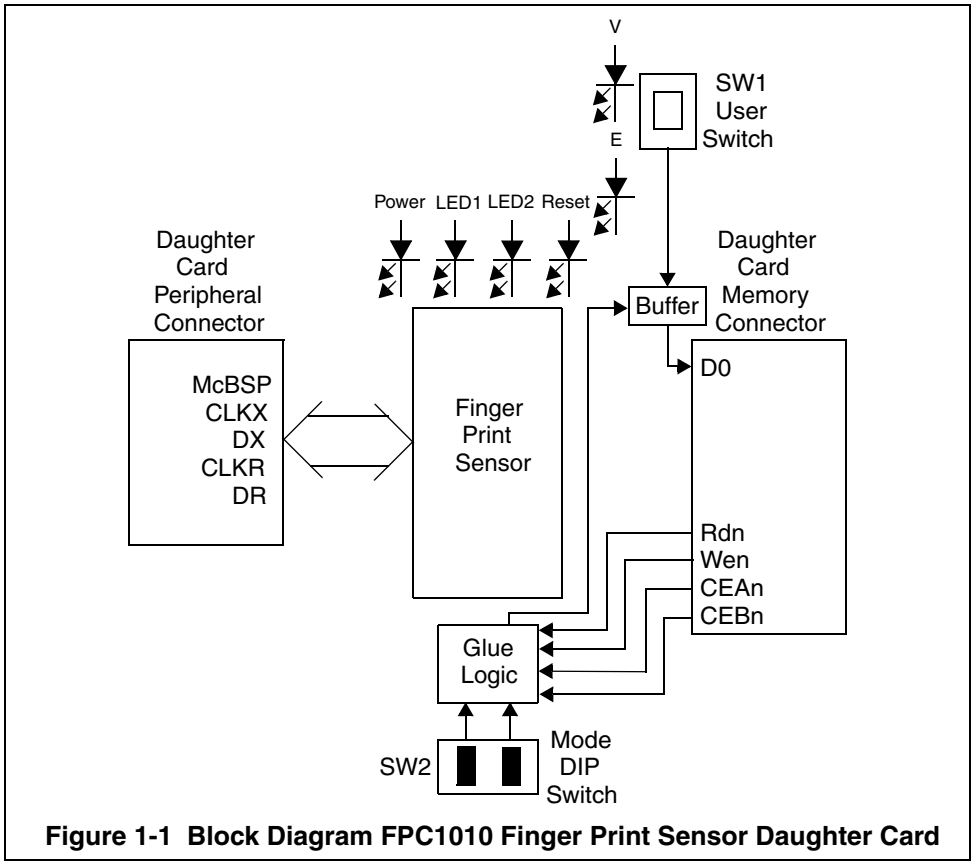


Figure 1-1 Block Diagram FPC1010 Finger Print Sensor Daughter Card

1.3 Sensor Interface.

The FPC1010 Finger Print Sensor Daughter Card uses a capacitive sensor with an SPI type interface. The daughter card interfaces to EVM's and DSK's via the McBSP on the expansion bus. The TOUT signal is used to reset the sensor.

Note 1: The Revision A TMS320VC5509 EVM board from Spectrum Digital has a different McBSP routing then the standard DSK interface a swizzler multiplex is provided on the daughter card and is controller by SW2-2.

Note 2: The C6711 DSK requires a small modification before installing daughter cards. The user must physically connect pins1-2 together on jumper JP1, by either populating the jumper or wiring the 2 pins together.

Chapter 2

Operation of the FPC1010 Finger Print Sensor Daughter Card

This chapter describes the operation of the FPC1010 Finger Print Sensor Daughter Card along with the key interfaces and an outline of the circuit board.

2.0 The FPC1010 Finger Print Sensor Daughter Card Operation

This chapter describes the FPC1010 Finger Print Sensor Daughter Card, its key components, and how they operate. It also provides information on the sensor's interfaces.The following topics are discussed:

- Sensor Inputs
- Configuration Switch
- User Switch Interface
- LEDs

2.1 The FPC1010 Finger Print Sensor Daughter Card

The FPC1010 Finger Print Sensor Daughter Card is a daughter card which allows evaluation and demonstration of finger print sensor techniques. Figure 2-1 shows the top view of the of the daughter card.

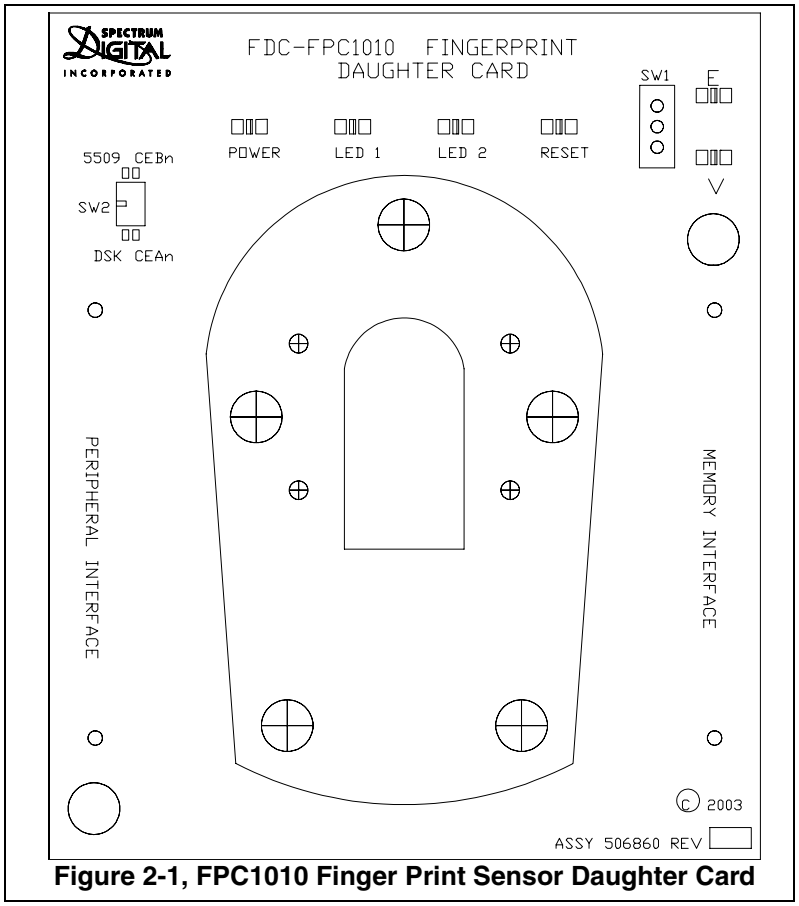


Figure 2-1, FPC1010 Finger Print Sensor Daughter Card

2.2 System Configuration

Figure 2-2 shows a typical use of the FPC1010 Finger Print Sensor Daughter Card when mounted on a DSK.

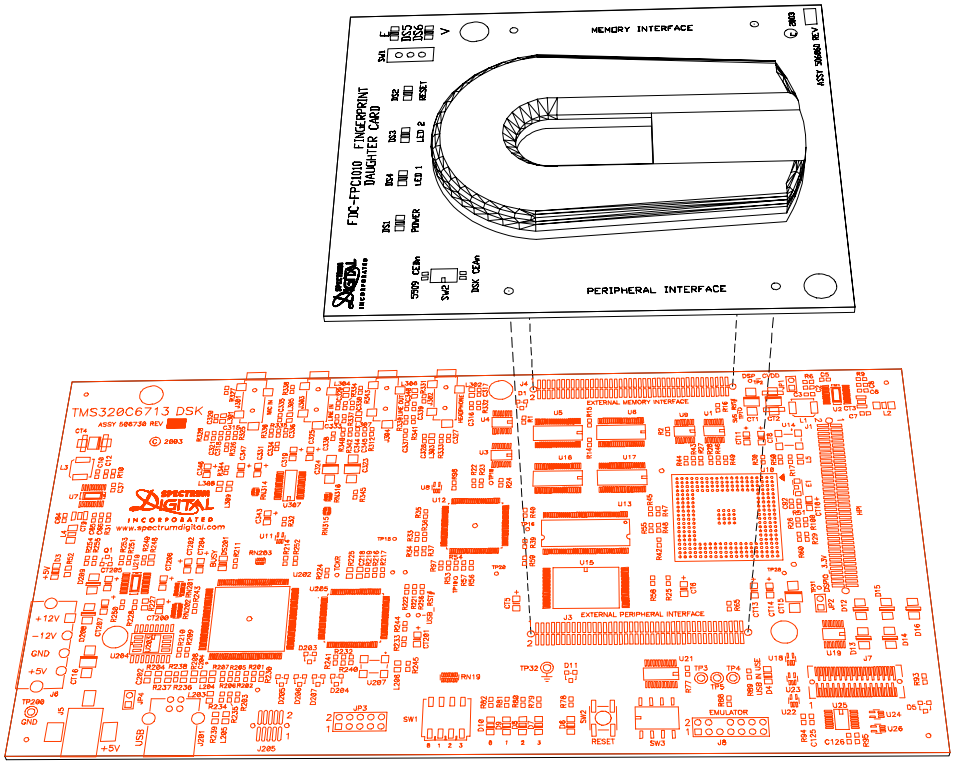


Figure 2-2, FPC1010 Finger Print Sensor Daughter Card System Configuration

2.2.1 Connecting the FPC1010 Finger Print Sensor Daughter Card

To connect the FPC1010 Finger Print Sensor Daughter Card in the configuration shown in figure 2-2 follow the procedure listed below.

1) Remove all power from all components in the system (emulator, DSK, etc.).

2a) If using the TMS320VC5509 EVM set the following jumpers on the EVM to the documented positions below.

Table 1: VC5509 EVM Jumper Settings

Jumper #	Position
JP4	1-2
JP5	1-2
JP6	2-3
JP7	1-2
JP8	1-2
JP10	2-3
JP11	1-2
JP12	1-2

2b) If using the TMS320C6711 DSK short the following jumper on the DSK to the documented position below.

Table 2: C6711 DSK Jumper Settings

Jumper #	Position
JP1	1-2

3) Plug the FDC daughter card on to the EVM or DSK. Note pin 1 orientation.

4) Set the two configuration switches. The suggested default is shown below.

Table 3: Configuration Switch Settings

Target	SW2-1	SW2-2
C6711 DSK	On	On
C6713 DSK	On	On
VC5510 DSK	On	On
VC5509 EVM	Off	Off

The board silkscreen provides a quick reference to switch settings.

5) Apply power to the target.

2.3 Mode Select Switches

The Mode Select Switch is used to select either the DSK or VC5509 EVM McBSP interface and which memory chip enable is used to interface to the user slide switch.

Table 4: Memory Chip Enable

SW2-1	User Switch Chip Enable
Off	CEBn Chip Enable, Pin 78
On	CEAn Chip Enable, Pin 77

Table 5: McBSP Routing

SW2-2	McBSP Routing
Off	5509 EVM McBSP Select
On	DSK McBSP Select

2.4 User Select Switch

A slide switch is available on the daughter card for user programs. Typically this switch is used for enrolling new finger print entries or verifying current entries in the database The switch is read on data bit 0 of one of the two expansion chip enable areas. The chip select should be configured for asynchronous memory 32 bits wide. The mapping for CEAn and CEBn are shown in the table below.

Table 6: CEAn and CEBn Mapping

Signal	Pin #	EVM5509	DSK 5510	DSK 6713	DSK 6711
CEAn	77	CE1n *	CE3n	CE3n	CE3n
CEBn	78	CE3n	CE2n	CE2n	CE2n

Note: This position is NOT recommended since it can conflict with the on chip VC5509 boot loader

The memory address corresponding to the chip select is supplied in the table below.

Table 7: Target DSP Memory Mapping

Target Board	Chip Enable	Memory Address
EVM VC5509	CEAn	0x0040 0000
EVM VC5509	CEBn	0x00C0 0000
DSK C5510	CEAn	0x0060 0000
DSK C5510	CEBn	0x0040 0000
DSK C6711	CEAn	0xB000 0000
DSK C6711	CEBn	0xA000 0000
DSK C6713	CEAn	0xB000 0000
DSK C6713	CEBn	0xA000 0000

2.5 LEDs

The FPC1010 Finger Print Sensor Daughter Card has six (6) leds to indicate status.

Table 8: FPC1010 Finger Print Sensor Daughter Card LEDs

LED #	Color	Function	Controlled By DSK	Controlled By VC5509 EVM
DS1	Green	Power	Power	Power
DS2	Yellow	Reset	TOUT1 Peripheral Pin 45	GPIO2 Peripheral Pin 66
DS3	Green	LED2 ( )	Control 1 Peripheral Pin 63	GPIO3 Peripheral Pin 63
DS4	Red	LED1 ( )	Control 0 Peripheral Pin 64	GPIO1 Peripheral Pin 64
DS5	Green	E (Enroll)	SW1 - Up	SW1 - Up
DS6	Green	V (Verify)	SW1 - Down	SW1 - Down

2.6 FPC1010 Finger Print Sensor Daughter Card Interface Signals

The FDC daughter card uses very few interface signals. The tables below indicate the signals used.

The I/O direction is referenced from the CPU card that the daughter card plugs into.

2.6.1 Memory Interface

Table 9: Memory Interface

Pin	Signal	I/O	Description	Pin	Signal	I/O	Description
1				2			
3				4			
5				6			
7				8			
9				10			
11	GND	Vss	System ground	12	GND	Vss	System ground
13				14			
15				16			
17				18			
19				20			
21				22			
23				24			
25				26			
27				28			
29				30			
31	GND	Vss	System ground	32	GND	Vss	System ground
33				34			
35				36			
37				38			
39				40			
41	3.3V	Vcc	3.3V voltage supply pin	42	3.3V	Vcc	3.3V voltage supply pin
43				44			
45				46			
47				48			
49				50			
51	GND	Vss	System ground	52	GND	Vss	System ground
53				54			
55				56			
57				58			
59				60			
61	GND	Vss	System ground	62	GND	Vss	System ground
63				64			
65				66			
67				68			
69				70	DC_D0	I/O	EMIF data pin 0
71	GND	Vss	System ground	72	GND	Vss	System ground
73	DC_RDn	O	EMIF async read enable	74		O	
75				76			EMIF asynchronous ready
77	DC_CEAn	O	Chip enable A	78	DC_CEBn	O	Chip enable B
79	GND	Vss	System ground	80	GND	Vss	System ground

2.6.2 Peripheral Interface

Table 10: Peripheral Interface

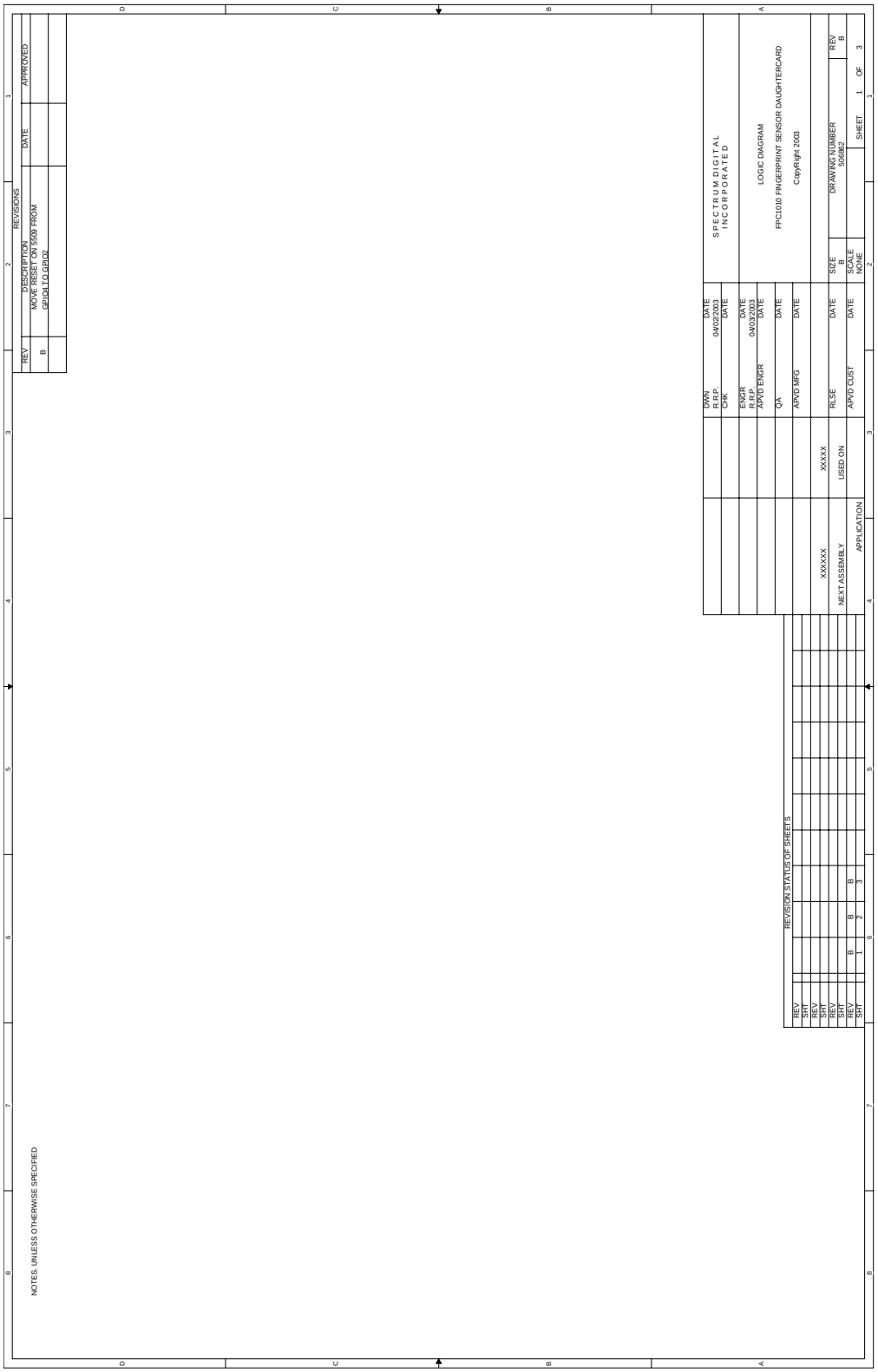
Pin	Signal	I/O	Description	Pin	Signal	I/O	Description
1				2			
3	GND	Vss	System ground	4	GND	Vss	System ground
5				6			
7	GND	Vss	System ground	8	GND	Vss	System ground
9				10			
11				12			
13				14			
15				16			
17				18			
19	3.3V	Vcc	3.3V voltage supply pin	20	3.3V	Vcc	3.3V voltage supply pin
21	CLKX0 *	I/O	McBSP0 transmit clock	22			
23				24	DX0 *	O	McBSP0 transmit data
25	GND	Vss	System ground	26	GND	Vss	System ground
27				28			
29				30	DR0 *	I	McBSP0 receive data
31	GND	Vss	System ground	32	GND	Vss	System ground
33				34			
35				36			
37	GND	Vss	System ground	38	GND	Vss	System ground
39				40			
41				42			
43	GND	Vss	System ground	44	GND	Vss	System ground
45	TOUT0	O	Timer 0 output	46			
47				48			
49	TOUT1	O	Timer 1 output	50			
51	GND	Vss	System ground	52	GND	Vss	System ground
53				54			
55				56			
57				58			
59				60			
61	GND	Vss	System ground	62	GND	Vss	System ground
63	CNTL1/GPIO3	O	Daughter card control 1	64	CNTL0/GPIO1	O	Daughter card control
65				66	STAT0/GPIO2	I	Daughter card status
67				68			
69				70			
71				72			
73				74			
75	DC_DET#	Vss	System ground	76	GND	Vss	System ground
77	GND	Vss	System ground	78			
79	GND	Vss	System ground	80	GND	Vss	System ground

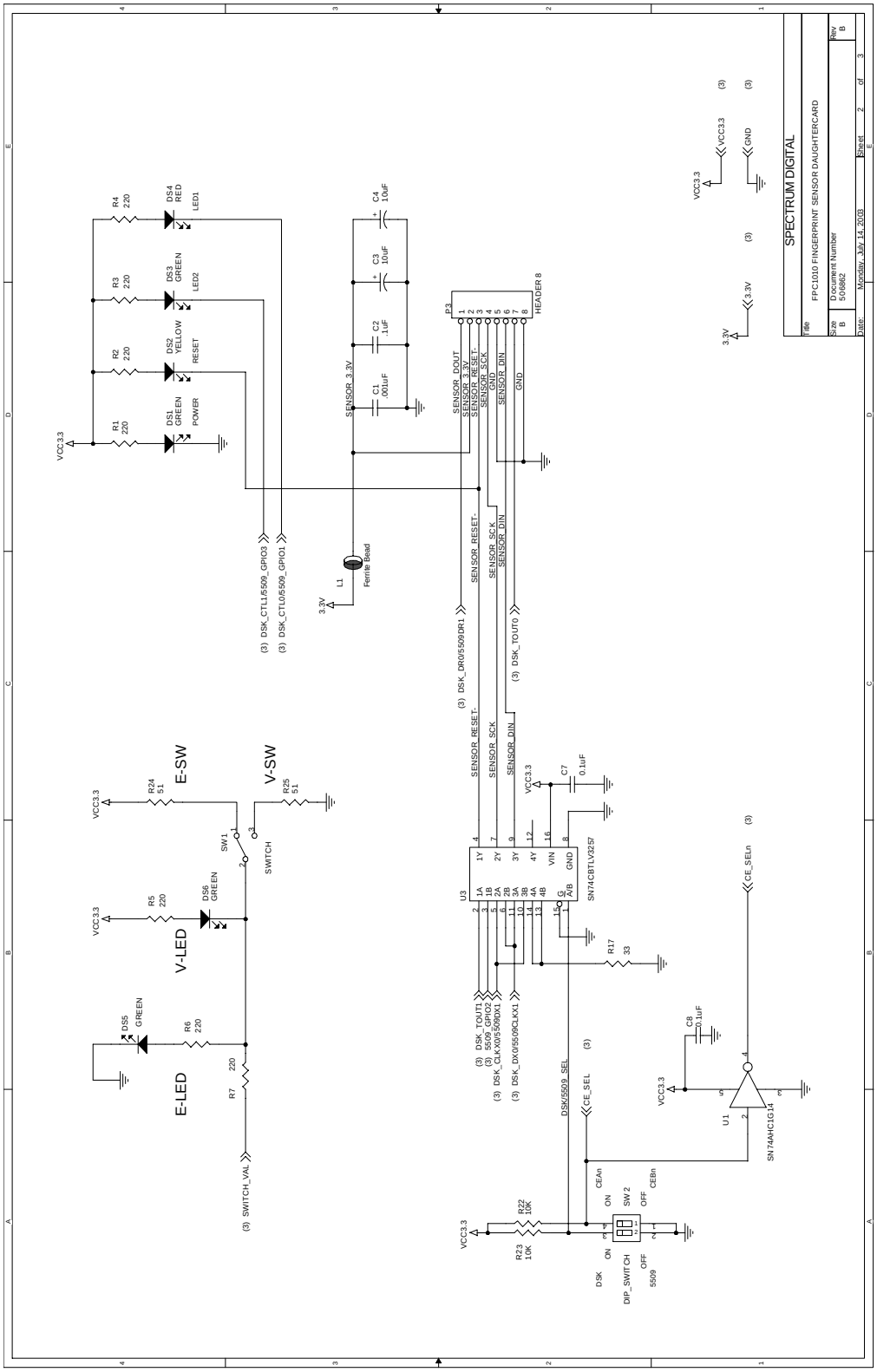
***** Pin order is different on VC5509 EVM from standard daughter card interface

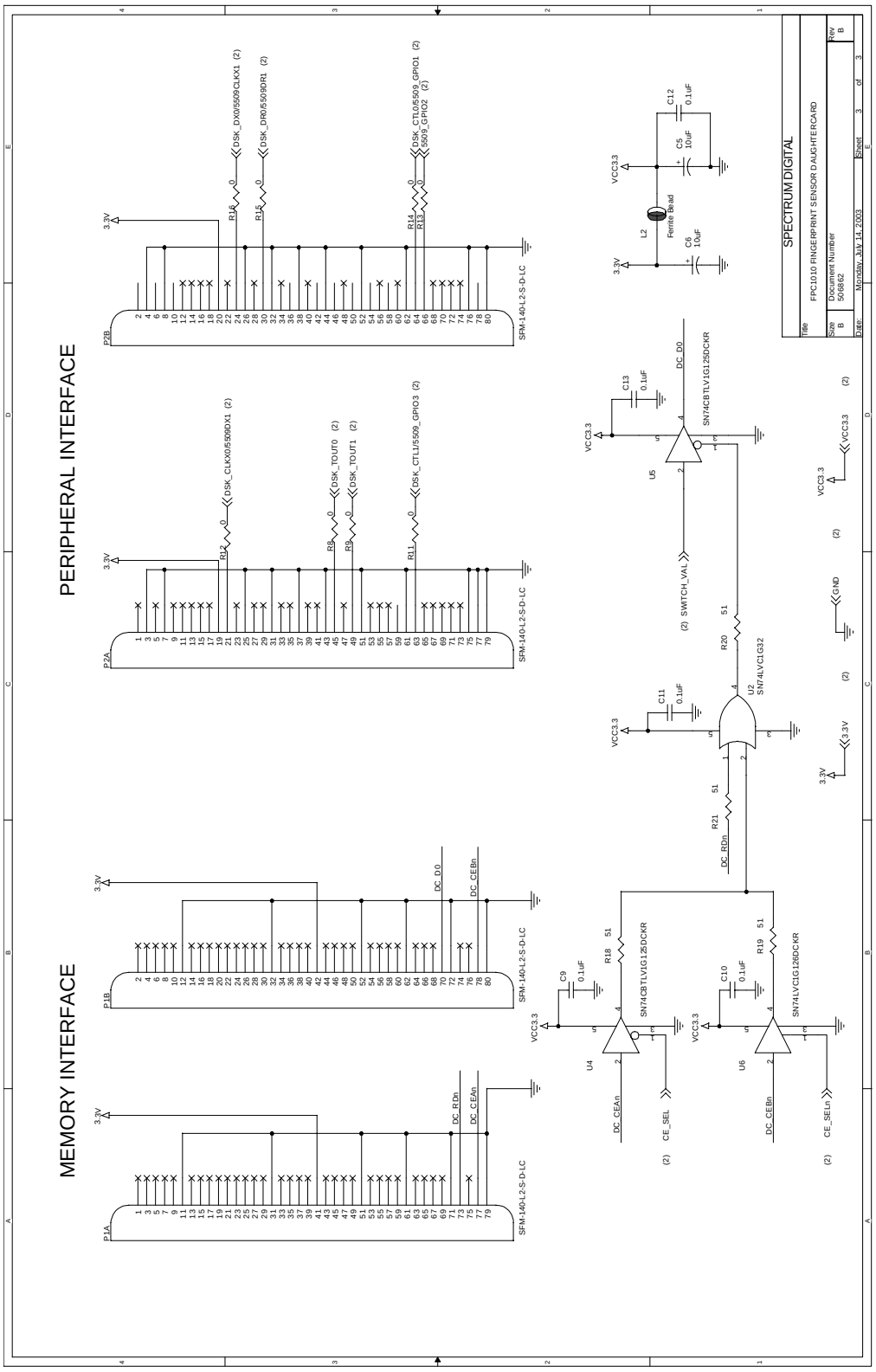
Appendix A

FPC1010 Finger Print Sensor Daughter Card Schematics

This appendix contains the schematics for the FPC1010 Finger Print Sensor Daughter Card. The schematics are drawn in OrCad.


REVISIONS
REV
DESCRIPTION
DATE
APPROVED
B
MOVE RESET ON 1509 FROM
GPIO4 TO GPIO2
NOTES, UNLESS OTHERWISE SPECIFIED
SPECTRUM DIGITAL
INCORPORATED
LOGIC DIAGRAM
FPC1010 FINGERPRINT SENSOR DAUGHTERCARD
CopyRight 2003
DRAWING NUMBER
500882
REV
B
SIZE
B
SCALE
NONE
SHEET 1 OF 3
DWN
R.P.P.
04/02/2003
CHK
ENGR
R.P.P.
04/03/2003
APVD ENGR
QA
APVD MFG
APVD CUST
RLSE
USED ON
XXXXX
NEXT ASSEMBLY
XXXXXX
APPLICATION
REVISION STATUS OF SHEETS
REV
SHT
1
2
3


SPECTRUM DIGITAL
FPC1010 FINGERPRINT SENSOR DAUGHTERCARD
Document Number
506862
Monday, July 14, 2008
HEADER 8
SENSOR_DOUT
SENSOR_3.3V
SENSOR_RESET-
SENSOR_SCK
SENSOR_DIN
GND
SN74CBTLV3257
SN74AHC1G14
DIP_SWITCH
SW 2
E-SW
V-SW
E-LED
V-LED
SWITCH
Ferrite Bead
DS1 GREEN POWER
DS2 YELLOW RESET
DS3 GREEN LED2
DS4 RED LED1
DS5 GREEN
DS6 GREEN
VCC3.3
3.3V
GND


PERIPHERAL INTERFACE
MEMORY INTERFACE
SPECTRUM DIGITAL
FPC1010 FINGERPRINT SENSOR DAUGHTERCARD
Document Number
506862
Monday, July 14, 2003
SFM-140-L2-S-D-LC
SN74CBTLV1G125DCKR
SN74LVC1G32
SN74LVC1G126DCKR
DSK_CLKX0/5509DX1 (2)
DSK_DX0/5509CLKX1 (2)
DSK_DR0/5509DR1 (2)
DSK_CTL0/5509_GPIO1 (2)
5509_GPIO2 (2)
DSK_TOUT0 (2)
DSK_TOUT1 (2)
DSK_CTL1/5509_GPIO3 (2)
DC_D0
DC_CEBn
DC_RDn
DC_CEAn
CE_SEL
CE_SELn
SWITCH_VAL
Ferrite Bead
VCC3.3
3.3V
GND

Appendix B

FPC1010 Finger Print Sensor Daughter Card Bill of Materials

This appendix contains the bill of materials for the FPC1010 Finger Print Sensor Daughter Card.

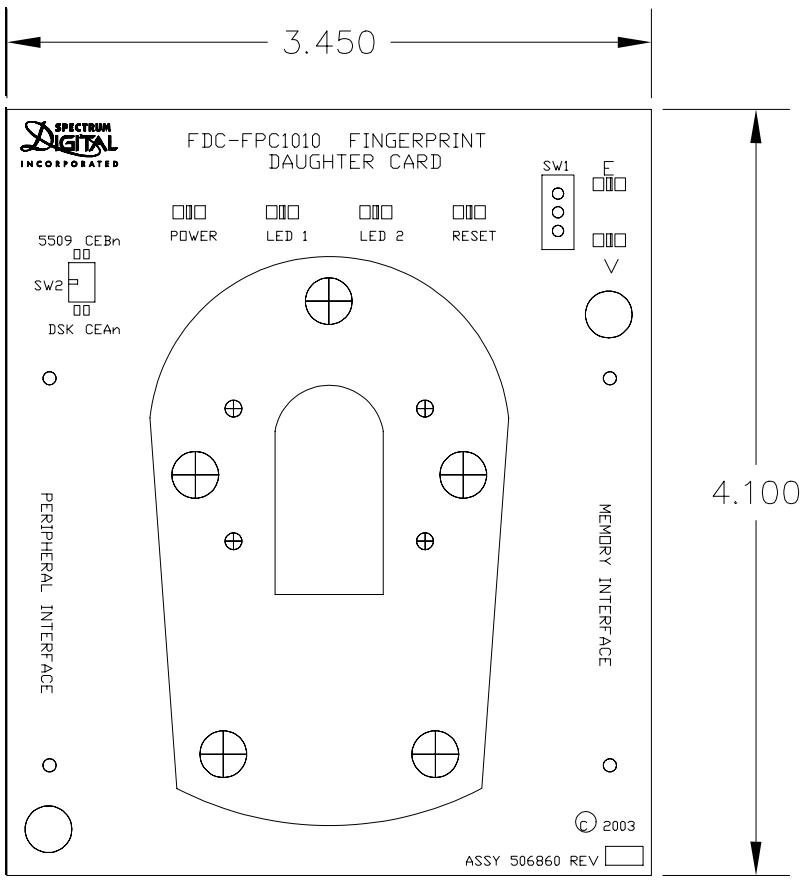
Table 1: Bill of Materials for FPC1010 Finger Print Sensor Daughter Card

Qty	Title	Mfr Name	Reference(m)	Mfr P/N
1	PWB,FINGERPRINT SYSTEM DAUGHTER CARD	SPECTRUM DIGITAL		506861-0001
1	LOGIC,FINGERPRINT SYSTEM DAUGHTER CARD	SPECTRUM DIGITAL		506862-0001
1	PWB,FINGERPRINT SYSTEM SHIM	SPECTRUM DIGITAL		506856-0001
1	IC,QSOP16,LOW VOLTAGE,QUAD 2:1 MULTIPLEXER/DEMULTI-PLEXER	TEXAS INSTRUMENTS	U3	SN74CBTLV3257PWR
3	IC,SO5,SINGLE BUS BUFFER GATE	TEXAS INSTRUMENTS	U4, U5, U6	SN74CBTLV1G125DCKR
1	IC,SO5,SINGLE SCHMITT-TRIGGER INVERTER GATE	TEXAS INSTRUMENTS	U1	SN74AHC1G14DCKR
1	IC,SO5,SINGLE 2-INPUT POSITIVE-OR GATE	TEXAS INSTRUMENTS	U2	SN74LVC1G32DCKR
4	LED,SMT 1206,GREEN	LITEON	DS1, DS3, DS5, DS6	LTST-C150GKT
1	LED,SMT 1206,RED	LITEON	DS4	LTST-C150CKT
1	LED,SMT 1206,YELLOW	LUMEX, INC	DS2	SML-LX1206YC-TR
2	FERRITE BEAD,SMT 1806, 75 OHMS	MURATA ELECTRONICS	L1, L2	BLM41P750S
4	CAP,TANT,SMT 1206,10uF,6.3V	PANASONIC	C3, C4, C5, C6	ECS-T0JY106R
8	CAP,CER,SMT 0603,1uF,6.3V ,X5R,+/-10%	PANASONIC	C2, C7, C8, C9, C10, C11, C12, C13	ECJ-1VB0J105K
1	CAP,CER,SMT 0603,.001uF, 50V,+/-10%,X7R	AVX CORPORATION	C1	06035C102KAT2A
8	RES,SMT 0603,0 OHM, 1/16 WATT	KOA SPEER ELECTRONICS, INC.	R8, R9, R11, R12, R13, R14, R15, R16	RM73Z1J000
7	RES,SMT 0603,51 OHM, 5%,1/16 WATT	PANASONIC	R17, R18, R19, R20, R21, R24, R25	ERJ-3GEYJ510V
7	RES,SMT 0603,220 OHM, 5%,1/16 WATT	PANASONIC	R1, R2, R3, R4, R5, R6, R7	ERJ-3GSYJ221V
2	RES,SMT 0603,10K OHM, 5%,1/16 WATT	KOA SPEER ELECTRONICS, INC.	R22, R23	RM73B1JT103J
1	CONN,SMT,RECEPTACLE,8 POS.,1mm SPC.	MOLEX	P3	52207-0890
2	CONN,SMT,VERTI-CAL,HEADER,40X2,.44 HIGH	SAMTEC, INC.	P1, P2	TFM-140-32-S-D-LC
1	SWITCH,SLIDE,SPDT	NKK	SW1	SS12SDP2
1	SWITCH,DIP,SMT,HALF-PITCH, 2 POS.	C&K/UNIMAX, INC.	SW2	TDA02H0SK1
1	SENSOR,FINGERPRINT, 200 x 152 PIXELS	Fingerprint Cards	XP3	FPC1010C
1	HOUSING,TOP COVER,FINGER-PRINT SENSOR	Fingerprint Cards	XP3	220-FPC9302A

Appendix C

FPC1010 Finger Print Sensor Daughter Card Mechanicals

This appendix contains the dimensions of the FPC1010 Finger Print Sensor Daughter Card.



This drawing is not to scale !

Figure A-1, FPC1010 Finger Print Sensor Daughter Card

[redacted]

Printed in U.S.A., July 2003
506865-0001 Rev. A