Detecting Errors in the Dual Port TPU RAM (DPTRAM) Module

by Jeff Loeliger
Rev. 0, 10 September 2001

1 Overview

The integrity of the data in the dual port TPU RAM (DPTRAM) module, when it is in TPU emulation mode, is ensured through the use of a RAM signature value. This allows a program to detect if the data has been corrupted in any way. This application note looks at how this information can be used and explains how to calculate the signature value.

2 Detailed Description

The integrity of the data in the DPTRAM module can be essential for the correct operation of the program on the MPC555. This is even truer if the DPTRAM is being used by the TPU3 in emulation mode. In emulation mode, the TPU3 reads instructions from the DPTRAM. Any error in the DPTRAM could cause the TPU3 to execute incorrect instructions. The DPTRAM module uses a multiple input signature calculator (MISC) to detect errors in the DPTRAM data.

There are two categories of error that can occur in the DPTRAM: hard errors and soft errors. The MISC signature can detect both types of errors. Hard errors are errors that are caused by a hardware fault in the device. This type of error cannot be fixed and the device must be replaced. Hard errors can be caused by mechanical or electrical extremes applied to the device. Soft errors happen when the value in the RAM cell changes but the cell itself is not damaged. Electromagnetic radiation or electrical noise typically causes soft errors. Since the RAM cell is not damaged, soft errors can be fixed by simply writing the correct data back into the RAM cell.

Normally, when a program is running from RAM, it has a background task that calculates a checksum or cycle redundancy check (CRC). When the DPTRAM is in emulation mode the TPU3 completely controls the RAM. The host CPU cannot read the data and therefore cannot directly detect errors in the RAM. The TPU3 uses Harvard architecture with separate instruction and data memories. The TPU3 can only read instructions from its instruction memory so it cannot calculate a checksum or CRC to check the integrity of the RAM. To solve this problem signature generator hardware, the MISC was added to the DPTRAM module. The signature hardware generates a unique value, called a signature, which can be used to validate the integrity of the contents of the RAM data.

When the TPU3 is in emulation mode and the MISC is enabled (MISEN = 1 in DPTMCR) a MISC value is calculated. The MISC hardware reads the RAM locations in reverse address order and calculates a unique 64-bit signature. The MISC hardware reads the RAM locations when the TPU3 modules are not reading them. The TPU3 module does not access RAM when idle and during a time slot transition. After a MISC value is calculated the MISF bit in the DPTRMCR register is set. The MISC hardware continuously calculates a MISC value. The value stored in the MISC registers (MISRH and MISRL) is updated even if the host CPU has not read it.

The MISC generator is based on the following polynomial:

$$G(x) = 1 + x + x^{2} + x^{22} + x^{31} = 0x80400007$$

The MISC signature generation starts by clearing the MISC value to 0. It then steps through each address reading 32 bit values and following the algorithm below:

```
If the least significant bit is 1 then
                 MISC = MISC right shifted by 1 bit
                 MISC = MISC XOR 0x80400007
else
                 MISC = MISC right shifted by 1 bit
end if
MISC = MISC XOR RAM data
```

3 Calculating MISC

There are two approaches to using the MISC value. If only soft errors are a problem, then this first value generated by the MISC calculator can be used as a reference and subsequent values can be compared to it. The problem with this approach is that if there is a hard error or an error occurs before the first MISC value is calculated, as it will not be detected. The second approach is to calculate the MISC value so that all MISC calculated by the MISC hardware can be checked.

The program in listing1 (tpu3misc.c) calculates a MISC value from an S-record file. The program is written for the PC using GNU GCC 2.95.2 but should work on most computers. The program has several options that are used to describe the size and contents of the RAM. To keep S-records short, they do not necessarily contain continuous data. If, for example, it contains some data followed by a gap and then more data, the gap will not be stored in the S-record. The MISC value is calculated using all of the values in RAM so the tpu3misc program must know the value of RAM locations that are not represented in the S-record. The value is specified using the '-f'' option. It is essential that the same value is used when the data is created for the RAM. If a C array of the S-record is created, then the same fill value must be used. The default fill value is 0. The size of the DPTRAM module must be specified so the signature value covers the entire memory. The module is 6 Kbytes on the MPC555 but may be a different size on new devices. The default size for the RAM is 6 Kbytes.

The program in listing 2 shows how the MISC value can be read and checked. The program loads a copy of the TPU3 microcode ROM into the DPTRAM. It then enables the TPU3 emulation mode and MISC hardware and awaits a valid MISC signature. The program uses the standard MPC555 header files and a copy of the TPU3 ROM that is shown in listing 3.

4 Conclusion

The tpu3misc program allows the MISC signature to be calculated so the MISC values can be checked. The MISC signature hardware provides an effective solution for monitoring the integrity of the data in the DPTRAM module while it is in TPU emulation mode.

5 Code Listings

5.1 Listing 1

```
/**************************************************************************/
/* FILE NAME: tpu3misc.c                      COPYRIGHT (c)  2000         */
/* VERSION:  1.0                                  All Rights Reserved     */
/*                                                                        */
/* DESCRIPTION:                                                           */
/* This program reads an s-record file and calculates a MISC value for    */
/* the file. The s-record must be an s19 file starting at 0.              */
/*                                                                        */
/*========================================================================*/
/* COMPILER: gcc 2.95.2                       AUTHOR: Jeff Loeliger       */
/*                                                                        */
/* UPDATE HISTORY                                                         */
/* REV      AUTHOR      DATE       DESCRIPTION OF CHANGE                  */
/* ----  -----------  ---------    ---------------------                  */
/* 1.00  J. Loeliger  25/Jul/00    Initial version of program.            */
/**************************************************************************/

/*****************
 * include files *
 *****************/
#include <stdio.h>
#include <stdlib.h>
#include "string.h"

/********************
 * global constants *
 ********************/
#define POLY 0x80400007   /*this comes from the polynomial: 1+x+x^2+x^22+x^31 */
#define TRUE 1
#define FALSE 0

/********************
 * global variables *
 ********************/
FILE *input_file;
char *program_name;
unsigned char *sram;                              /* 8 bit data from s-record */
unsigned int *data;                               /* 32 bit data in host computer format */
unsigned int DPTRAM_size = 6;                     /* size of the DPTRAM module */
                                                  /* default size is 6k        */
unsigned int DPTRAM_fill = 0;                     /* value to put in used DPRTAM locations */
                                                  /* default value is 0                    */
unsigned int verbose = FALSE;                     /* default is verbose mode off */

/***********************
 * function prototypes *
 ***********************/
void usage (void);
void read_file (FILE * input_file);
int calc_misc (void);
int get_hex_nib (void);
int get_hex_byte (void);
int get_hex_word (void);
int get_hex_3bytes (void);
int get_hex_long (void);

/****************
 * main program *
 ****************/
int main (int argc, char *argv[])
{
    unsigned int misc = 0;
    int i;
```

```
/* save program name for future use */
    program_name = argv[0];

/* loop for each option
 * stop when we run out of arguments
 * or we get an argument without a dash
 */
    while ((argc > 1) && (argv[1][0] == '-')) {
        /*argv[1][1] is the actual option character */
        switch (argv[1][1]) {
        case 'h':
            usage ();
            return (1);
            break;
        case 'f':
            sscanf (&argv[2][0], "%x", &DPTRAM_fill);
            ++argv;
            --argc;
            break;
        case 's':
            sscanf (&argv[2][0], "%x", &DPTRAM_size);
            ++argv;
            --argc;
            break;
        case 'v':
            verbose = TRUE;     /* turn verbose mode on */
            break;
        default:
            printf ("ERROR unknown option %s\n", argv[1]);
            usage ();
            return (1);
        }
        ++argv;
        --argc;
    }

/* check for filename */
    if (argc == 1) {
        printf ("ERROR a filename must be specified.\n");
        usage ();
        return (1);
    }

/* allocate memory for arrays */
    sram = malloc ((DPTRAM_size * 1024));
    data = calloc ((DPTRAM_size * 256), sizeof (unsigned int));

    if (sram == NULL || data == NULL) {
        printf ("ERROR can not allocate enough memory.\n");
        return (1);
    }

/* fill memory with default value */
    memset (sram, DPTRAM_fill, (DPTRAM_size * 1024));


/* check filename */
    if ((input_file = fopen (argv[1], "r")) == NULL) {
        printf ("ERROR can not open file %s\n", argv[1]);
    }
    else {

         read_file (input_file);

         fclose (input_file);

         printf ("%s version 1.0 by Jeff Loeliger - 25 July 2000 Copyright  2000.\n\n", program_name);
      printf ("The MISC value calculation used a RAM fill value of 0x%X and a size of %uk.\n", DPTRAM_fill,
DPTRAM_size);
```

```
        printf ("DPTRAM MISC value for the file %s is 0x%X.\n", argv[1], calc_misc ());
    };

};

/********************************************************************************
FUNCTION        : void usage()
PURPOSE         : This function displays the usage information.
INPUTS NOTES    : none
RETURNS NOTES   : none
GENERAL NOTES   : This function uses the program_name global variable.
********************************************************************************/
void usage (void)
{
    printf ("Usage: %s [options] filename\n", program_name);
    printf ("Calculate MISC value for DPTRAM and TPU3.\n\n");

    printf ("Options:\n");
    printf ("  -f n  fill value (32 bit) for unused area of DPTRAM (0 is default)\n");
    printf ("  -h    help, display this help and exit.\n");
    printf ("  -s n  size, where n is the size of the DPTRAM in kbytes (6 for 6k is default)\n");
    printf ("  -v    verbose, list intermediate values of calculation\n\n");
}

/********************************************************************************
FUNCTION        : void read_file (FILE * input_file)
PURPOSE         : This function reads the input file and creates a binary
                    image in the sram array.
INPUTS NOTES    : FILE structure
RETURNS NOTES   : none
GENERAL NOTES   : This function only reads S19 files. This is the only function
                   that needs to be changed to support other s-record formats.
********************************************************************************/
void read_file (FILE * input_file)
{
    char line[100];
    int address;
    int type;
    int length;
    int i;

    while (!feof (input_file)) {

        if (toupper (fgetc (input_file)) != 'S') {
            /* ignore line if it does not start with S */
            fgets (line, sizeof (line), input_file);
        }
        else {

            type = get_hex_nib ();
            if (type == 1) {    /* only process S1 lines */

                length = get_hex_byte ();
                address = get_hex_word ();

                for (i = 0; i < (length - 3); i++) {
                    sram[address] = get_hex_byte ();
                    address = address + 1;
                }
            }
            fgets (line, sizeof (line), input_file);
        }
    }

};
/********************************************************************************
FUNCTION        : void calc_misc()
PURPOSE         : This function calculates the MISC value.
INPUTS NOTES    : none
RETURNS NOTES   : MISC value
```

```
GENERAL NOTES   :
*******************************************************************************/
int calc_misc (void)
{
    int j;                    /* loop counters */
    int i;

    unsigned int misc = 0;


    /*first move byte array in sram to long words in data */
    /*This changes the big endian data in the s-record to 32 bit integers in the host */
    /*  because the value is calculated it will work with a big or little endian host */
    for (i = 0; i < ((DPTRAM_size*1024)-1); i += 4) {
        data[i / 4] = (sram[i]<<24) + (sram[i + 1] << 16) + (sram[i + 2]<<8) + (sram[i + 3]);
    }

    for (j = ((DPTRAM_size*1024)/4)-1; j >= 0 ; j--) {
        if (verbose)
            printf ("Address: %04X data: %08X %i, carry=%1X, ",j*4, data[j], data[j], misc & 0x1);
        if (misc & 0x1) {
            misc >>= 1;
            misc ^= POLY;
        }
        else {
            misc >>= 1;
        }
        misc ^= data[j];
        if (verbose)
            printf ("New Signature is: %08X\n", misc);
    }
    return (misc);
};


/*******************************************************************************
FUNCTION        : int get_hex_nib()
PURPOSE         : This function reads a hex nibble (4 bits) from an input file.
INPUTS NOTES    : none
RETURNS NOTES   : hex value of ASCII nibble
GENERAL NOTES   : none
*******************************************************************************/
int get_hex_nib (void)
{
    int c;

    c = toupper (fgetc (input_file));
    if (!isxdigit (c))
        return (0);
    if (isdigit (c))
        return (c - '0');
    else
        return (c - 'A' + 10);
}

/*******************************************************************************
FUNCTION        : int get_hex_byte()
PURPOSE         : This function reads a hex byte (8 bits) from an input file.
INPUTS NOTES    : none
RETURNS NOTES   : hex value of ASCII byte
GENERAL NOTES   : none
*******************************************************************************/
int get_hex_byte (void)
{
    int x;

    x = get_hex_nib () << 4;
    return (x + get_hex_nib ());
}
```

```
/*******************************************************************************
FUNCTION        : int get_hex_word()
PURPOSE         : This function reads a hex word (16 bits) from an input file.
INPUTS NOTES    : none
RETURNS NOTES   : hex value of ASCII word
GENERAL NOTES   : none
*******************************************************************************/
int get_hex_word (void)
{
    int x;

    x = get_hex_byte () << 8;
    return (x + get_hex_byte ());
}
```

5.2 Listing 2

```
/***************************************************************************/
/* FILE NAME: misctest.c                      COPYRIGHT (c)  2000 */
/*                                                                         */
/* INCLUDE FILES: mpc555.h, stdlih.h, string.h, tpu3rom.h                 */
/* VERSION: 1.0                                                            */
/*=========================================================================*/
/*                                                                         */
/* DESCRIPTION: This program loads a copy of the TPU3 ROM into the DPTRAM */
/*     and reads the MISC value.                                           */
/*=========================================================================*/
/* COMPILER: Diab Data        VERSION: 4.3b                                */
/*                                                                         */
/* AUTHOR: Jeff Loeliger                  CREATION DATE:  28/Jul/00       */
/* LOCATION: East Kilbride, Scotland.                                      */
/*                                                                         */
/* UPDATE HISTORY                                                          */
/* REV      AUTHOR      DATE       DESCRIPTION OF CHANGE                   */
/* ---   -----------  ---------    ---------------------                   */
/* 1.0   J. Loeliger  28/Jul/00    Initial version of function.            */
/***************************************************************************/

#include "mpc555.h"
#include "stdlib.h"
#include "string.h"
#include "tpu3rom.h"

unsigned int misch, miscl;

void main ()
{

/*****************************************************
 * Configure and copy new TPU microcode into DPTRAM *
 *****************************************************/
    DPTRAM.RAMBAR.R = 0xffa0;                      /* enable DPTRAM */

    memset ((void *)0x302000, 0x00, 6144);
    memcpy ((void *)0x302000, tpu3rom, 4096);


    DPTRAM.DPTMCR.B.MISEN = 1;                     /* enable MISC calculation*/

    TPU_A.TPUMCR.B.EMU = 1;                        /* put TPU A into emulation mode */

    while (DPTRAM.DPTMCR.B.MISF == 0);             /* wait for valid MISC */

    misch = DPTRAM.MISRH.R;                        /* read 64 bit MISC value */
    miscl = DPTRAM.MISRL.R;                        /* MISC value shoud be 0x9F27EF98 */

    while(1);
};
```

5.3 listing 3 – tpu3rom.h

```
const unsigned long tpu3rom[1024] = {
0x3FFFFFFE,0x7FFFFEFE,0xAC04FFC0,0x7FFFFFCF,0x7FF9FEFE,0xAC1FFFFF,0xA40DFFFF,0x673BFFFF,
0xFFFFFFFF,0xB400FEFF,0xD02FFFFF,0xA40DFFFF,0x7FFBFFFF,0x1CFE080F,0x367C380F,0x9014FEFF,
0x1FFFF013,0x30FDD013,0x8C14FEFF,0x7FFFFFFB,0xA41DFFFF,0x1FFFF007,0x20DFD007,0x22DC0FFF,
0x841DFEFF,0x3EDFF007,0x3FFFF817,0x7FFFFFFB,0x3EFFF00F,0x3CFFF00B,0x5C5DCFFE,0xA436FFFF,
0x7FFBFFFF,0xD01D3FFF,0xAC0BFFFF,0xA436FFFF,0x673BFFFF,0xB000FFFF,0xB400FEFF,0xD01D1FFF,
0xB400FFFF,0xD01D1FFF,0xAC1FFFFF,0xA40DFFFF,0x673BFFFF,0xFFFFFFFF,0xB400FFFF,0xD00D3FFF,
0xAC0BFFFF,0xA436FFFF,0x673BFFFF,0xB000FEFF,0xD028FFFF,0x7FFBFFFE,0x7FFDFEFE,0xBFFF417F,
0x00CDF003,0x903BFFDF,0x5859FFD7,0x943DFFC7,0x7FF9FECA,0x11EDF007,0x92424EFF,0x181FF80B,
0x3A1FFFFF,0xBFFF7FFF,0xE6DE09FF,0x362DF007,0x1C5EB003,0x9248F3FC,0x226DFFFF,0x7FFFF5FF,
0x984DFFFF,0x3667FFFF,0x924DFEFF,0x145E7203,0x345E3FFF,0x5FF9FEFA,0xBC3840BF,0x002FF007,
0x200FFFFF,0x15FC480B,0x8459FFFF,0xE07E05FF,0x16DFFA03,0x280CF007,0x9058FEFF,0x5C5873FE,
0x5C5875FE,0xAE55FFFF,0xE6DE09FF,0xB25EFEFF,0x229DEFFF,0x9455FEFF,0x3CFFF007,0x7FF9FFC2,
0xBC6740BD,0xBC67407D,0x9C01FFCC,0x185FF807,0x31FCCFFF,0x8CBCFFD7,0x9472FEDF,0x7FF9FEC2,
0x9A6BFFFF,0x307FD807,0x307FE807,0x1FFFF00B,0x01FDF003,0x9464FFFF,0x31FCCFFF,0x8CBCFFFF,
0x9472FEFF,0x98BCFEFF,0x9A80FEFF,0x101FF00B,0x203EDFFF,0x15FC480F,0x947DFFFF,0x8C80FEFF,
0x227FFFFF,0x21FCCFFF,0x8681FFFF,0x20DFD00B,0xC0A2F80F,0x360DFFFF,0x9081FEFF,0xD0A8FFFF,
0x20DFE00B,0x00DDE813,0x869FFFFF,0x160DF80F,0xE07E3FFF,0xB296FEFF,0xE63FE1FF,0x928FFEFF,
0xE1FDE7FF,0x8E94FFC7,0xE1FDCFFF,0x8E94FFFF,0xE07E47FF,0xE1FDB7FF,0xE07E4FFF,0x8C94FFFF,
0x9E97FFFF,0xE07FA7FF,0xD297FFFF,0xE07F77FF,0xE7FC61FF,0x9497FFFF,0x367EBFFF,0x17EDF203,
0x929BFFCF,0x223FFFFF,0x203FFFFF,0x32FFFFFF,0xFF5E0FFF,0x1E1FF80F,0x320A3FFD,0x36EFFFFF,
0x33FC0FFF,0x90A7FEFF,0x160DF00B,0x92A8FEFF,0x327E580F,0xD2A8FFFF,0x20DFE00B,0x30E7F00F,
0x1C1E7813,0x362DFFFF,0x1F47D017,0x90ADFEFF,0x2547FFFF,0x9AB2FFFF,0x3075F817,0x3065F817,
0x90B2FEFF,0x367FD817,0x8EB5F4FF,0x255DEFFF,0x7FFFF3FF,0x5259FEFF,0xE73E21FF,0x9401FFFF,
0xAEB2FEFF,0x3675FFFF,0xD2B2FFFF,0x3675FFFF,0x9A80FFFF,0x103FD00B,0x201FFFFF,0xD275FFFF,
0x361E4FFF,0xBFFFFFC8,0x7FFBFEFF,0x7FFDFFFE,0xA4C9FEFF,0x3FFFF013,0x08FFF80F,0x5C5CFFFB,
0xB0CBFEFD,0xA0CBFEFC,0x70FBDFFF,0x3FFFF016,0xA4C3FFFF,0x3E7FF817,0x5C583FC6,0xBFFF05C4,
0x7FF9FED6,0xBFFF4144,0x58583EDA,0x9AEFFEFF,0x3EFFF003,0x5C58FEFF,0x1FFFF00B,0x11FDF80F,
0x8C01FFFF,0x7FFFF5CB,0x367FD813,0x30FFC017,0x3FFFF00A,0x1C5EF013,0x9AF4FEFF,0x30FDE013,
0x8CEEFFFF,0x94E9FFFF,0x1FFFF017,0x1FFFF803,0x30EDF017,0x90E7FFFF,0x5FF9F4FE,0x367FD803,
0x5FF9F2FE,0xACEEFEFF,0xB2E4FEFF,0x1FFFF003,0xD2E4FFFF,0x30FFDFFF,0x5FF9F2C6,0x1C5EF00F,
0xACF2FEFF,0x30FFDFFF,0x366FF013,0x5C58F0CE,0x8EFCFFFF,0x1E1FF817,0xB6F9FFFF,0x5669FEFF,
0x3FFFF816,0x367FC817,0x1FFFF803,0x367FD802,0xB4FEFFFF,0x3E0FCFFF,0xAD05FEFF,0x1667F003,
0xB102FEFF,0x30FFDFFF,0x31EDFFFF,0x9105FEFF,0x321FCFFF,0x1FFFF00F,0x655805C7,0x366FFFFD,
0x367E780B,0x7FFFFFFA,0x035FF007,0xBF24FEFF,0x385FFFFF,0x16FFFA03,0x035FF00B,0x103EF817,
0x143CC013,0x8F19FEFF,0x101EC007,0x20DDE007,0x8D16FEFF,0x22DFE007,0xD530FFFF,0xD323F813,
0x367E4817,0x35FC4FFF,0x8123FFFF,0x201FDFFF,0x22DC4007,0x951FFFFF,0x32DFF007,0xD530FFFF,
0x1FFFF813,0x3FFFF817,0x367E7813,0xAD26FEFC,0xB126FECD,0x7FFFFFC7,0xC730F80F,0x1C3EF007,
0x30EDFFFF,0x932C42FF,0x255DEFFF,0xBFFF45FF,0x5459FFFF,0x9728FEFF,0xE73E21FF,0x7FFFFEFA,
0xE06611FF,0x267EFFFF,0xCBFFF203,0x021FF817,0xBD3606C4,0xBFFF07CC,0x6739FEFF,0x3EFFC017,
0xB53AFFFF,0x3EFFF017,0xAD3CFEFF,0xE73FE1FF,0xB13EFEFF,0xE73FE1FF,0x1E27D00F,0xC544F817,
0xC544F817,0x9D43FEFF,0xC544F817,0x371FF202,0x17FDF813,0x8D47FEFF,0x30FEBFFF,0x3427FFFF,
0xDBFFFFFF,0xE73E21FF,0xBF4D464C,0x185FF007,0x09FFF007,0x3C7FFA03,0x20DFD007,0x22DC0FFF,
0x9765FEFF,0x107FF003,0xD167F807,0xBD5546BC,0xBFFF477C,0xAF65FFC7,0x1E5FF203,0x105FF003,
0x101FF00F,0x023FF80F,0xB365FEFF,0x1C5E7203,0x367FDFFF,0x303C8203,0x32EE8FFF,0x8F63FEFF,
0x307EBFFF,0x32EFFFFF,0x307FFFFF,0xB566FFFF,0x3C7EF807,0x5C583EFE,0x3C7EF80B,0x5C583EFA,
0xE1E401C7,0x8F6CFEF8,0x7859FEFF,0x7A59FEFF,0x3C7FF807,0xD56FFFFF,0x525CB5FA,0x163FF00B,
0x101DF80F,0x8776FFFF,0x36FEB013,0x37FC4FFF,0x8577FFFF,0xD9FF1FFF,0x545CF18A,0x545CF14A,
0xA56CFFFF,0xD16DFFFF,0xA57DFEFF,0x9D7DFEFF,0x3C7FF807,0xD56FFFFF,0x545CF3FA,0x505DF3FE,
0xEA01EA01,0xEA0102BD,0x82C442D0,0x42D082C4,0xEA01EA01,0xEA01EA01,0x82C442D0,0x42D082C4,
0xFA00FA00,0x029762A7,0x3AB63ABC,0x72AD929E,0x92B562A8,0x92B562A8,0x92B862A8,0x72AC929E,
0x32963296,0x32951288,0x3291428E,0x3293428E,0x12011201,0x12011201,0x3291428E,0x3293428E,
0x297A2978,0x0168F800,0x917FF97C,0x316C316C,0xF801F801,0xF801F801,0xF801F801,0xF801F801,
0x026D026D,0x12001269,0x126A3A83,0x126A3A83,0x12014A76,0x12014A76,0x12014A76,0x12014A76,
0x12001200,0x12431200,0x42488265,0x42468265,0x12011201,0x12011201,0x42488265,0x42468265,
0xFA01FA01,0xEA22EA1D,0x62236237,0x62236237,0xFA01FA01,0xFA01FA01,0xFA01FA01,0xFA01FA01,
0xA953A953,0x8954894A,0xA955A94C,0x8955894C,0xE801E801,0xE801E801,0xA955A94C,0x8955894C,
0xF801F801,0xE934E935,0xE936E936,0xE936E936,0xF801F801,0xF801F801,0xE936E936,0xE936E936,
0xE800E800,0x010A010C,0x090D090D,0x090D090D,0x010A010C,0x010A010C,0x090D090D,0x090D090D,
0x12191219,0x121AFA01,0x320A320A,0x320A320A,0xFA01FA01,0xFA01FA01,0xFA01FA01,0xFA01FA01,
0xF801F801,0xE8CF20D1,0x28D328DD,0x28D328DD,0xE801E801,0xE801E801,0xE801E801,0xE801E801,
0xE801E801,0x40C1E801,0xA0C460CC,0xA0C460CC,0xE801E801,0xE801E801,0xA0C460CC,0xA0C460CC,
0xE860E860,0xE8610062,0x20682068,0x20682068,0xE801E801,0xE801E801,0x20682068,0x20682068,
0x304E3037,0x304E3037,0x004F004F,0x004F004F,0xF801003D,0xF801003D,0xF801003D,0xF801003D,
0xF836F836,0xF8360002,0x402A4030,0x40054022,0xF801F801,0xF801F801,0x402A4030,0x40054022,
0x3FFFFFFE,0x7FFFFEFE,0x231FFFFF,0xE31E21FF,0xE31E41FF,0xE31E61FF,0xE31E81FF,0xE31EA1FF,
0xE31EC1FF,0xE31EE1FF,0x035FF80F,0x367FD80F,0x007FF00F,0x103FF00B,0x9E11FFFF,0x11FC8A03,
0x3C7FFFFF,0x8413FFFF,0xD01CF817,0xAE16FEFF,0x3FFFF813,0xC402F004,0xB21BFFFF,0x7FFFFFFB,
0xBC1CF0FF,0xBC1BFEC0,0xBFFFFFC8,0x3E7FF80F,0x7FF9FEFE,0xBFFF07C4,0x6739FEFF,0x3EFFC00F,
0xB422FFFF,0x3EFFF00F,0x387FF80A,0x6739FFFF,0x103DF813,0xB628FEFF,0x10FFCA03,0x35FDCFFF,
0x8C01FEFF,0x303CF00F,0x1CFFF817,0xB233FFFF,0x367FD203,0xB633FEFF,0x35FDCFFF,0xAC33FEFF,
0xE73E21CF,0xE73FE1FF,0xBC33F2FD,0x8E42FEDF,0x1E1FD203,0x30FE4203,0x7FFFFFD6,0x7E03DFFF,
0xAE3FFFFF,0x063FF817,0x9A3DF6FF,0xE73E21C7,0x30FFCFFF,0x353FF00F,0xBE3FFEFC,0x367FDFFF,
0x9A35FEFF,0x1207F203,0x30FE7202,0x60F9FED7,0xBFFFFFC8,0x3E7FF80E,0xD24BFFFF,0x3C7FF80B,
0xAE4BFEFF,0x3C7FF80B,0xBE4DF2CF,0x101FF013,0x425CF5C7,0x167E400F,0x307CF80F,0xA652FEFF,
0x1FFFF003,0xA261FEFF,0x035FF007,0x9062FFFF,0x20DFD007,0x7FF9FFFF,0x22DC0FFF,0x8400FEFF,
0x3EDFF007,0x3EFFF00F,0x9A00FFFF,0x7FFFFFD3,0x3FFFF817,0xB05FFFFF,0x7FFFFFD6,0xC402F000,
0x3FFFF816,0x9068FEC7,0x1E7FF013,0x60DDDFC7,0xD05D7013,0xA068FEFF,0x3C7FF80B,0x4258F5C6,
0x7FFDFFFE,0xBFFFFFD8,0x3C7FF817,0x7FF9FECA,0x7FFFFFFA,0x387FFDDB,0x3A7FFDDF,0xAC6CFFFF,
0xBFFFFFC0,0xCFFF5007,0x1FFFF80B,0x1FFFFA03,0x565C3FFF,0x3CFFF012,0x16FFFA03,0x023FF003,
0x9A7DFED3,0x1FFFF00F,0x1FFFF80F,0x1FFFFA03,0x565DFFFF,0x3FFFF813,0x227FFFFF,0x1E7FF203,
0xFF5E0FFF,0x366A3FFD,0xDFFFE807,0x161DE013,0x8086F6FF,0xC069F003,0x30FEF013,0x505DFFD2,
0xAC8FFFFF,0xBFFF0FFC,0x585FFFFF,0xFFFFFFFF,0xB08FFEC3,0xCFFF300B,0x5C5E31FF,0xB693FFFF,
0x1FFFF007,0x70E9FEFB,0x3FFFF006,0x70E9FEFB,0x30FFC006,0xBC9140BF,0xBC93407F,0xE1E401C7,
0x8E9BFED0,0x7859FFFF,0x3A5FFFFF,0xB2ADFEFF,0x1FFFF00F,0xCFFF3013,0x3C7FF803,0x307FFFFF,
0x027FF203,0x101FFA03,0x3FFFF817,0x163FF00B,0x121E300F,0x525C34CB,0x347E3806,0x9C00FEFF,
0xCFFFE817,0xB600FFFF,0x36FE3007,0x505DF3D6,0x7FFFFEDF,0x3C7FF803,0x107EF00B,0x5C5C35C3,
0xACB4FEFF,0x1FFFF013,0x235FFFFF,0xD402FFFC,0x3FFFF806,0x7FFFFEDE,0x98BCFEFF,0x1FFFF013,
0x007FF017,0xCFFFEA03,0x167E300F,0x367E3807,0x565DF3D6,0x11FDF80F,0x8EC1FFD8,0x31EDF80B,
0x7FFFFFD7,0x92C4FEC7,0x185FF013,0x7FFFFFCF,0x9AC7FFFF,0x3065F813,0x306DF813,0xB2CEFEFF,
0x3C1FFFFF,0xE73E21FF,0x90CC4EFF,0xBCCDFE7D,0xBFFFFFBD,0xE73FE1FF,0x1FFFF807,0x5258FEFE,
0x1C1FF813,0xAEDAFEFF,0x30FDE00B,0xE73FE1FF,0xBFFF0FFF,0xB6DAFEFD,0xE73E21FF,0x36FFD013,
0x98DAFFFF,0x367FC813,0x94CEFEFF,0x7FF9FEFA,0xFE9E03FF,0xFEDE03FF,0xD001F003,0x119FF00B,
0xBFFFFFF9,0x387FF813,0x673FFEFB,0x3A7FF817,0xB4E6FFFF,0x3E7FF80E,0x3E7FC80E,0xD2E0FFFF,
0x11BFF00B,0x00000000,0x00000000,0x00000000,0x00000000,0x00000000,0x00000000,0x00000000,
0x00000000,0x00000000,0x00000000,0x00000000,0x00000000,0x00000000,0x00000000,0x00000000,
0x00000000,0x00000000,0x00000000,0x00000000,0x00000000,0x00000000,0x00000000,0x00000000,
0x00000000,0x00000000,0x00000000,0x00000000,0x00000000,0x00000000,0x00000000,0x00000000,
0x00000000,0x00000000,0x00000000,0x00000000,0x00000000,0x00000000,0x00000000,0x00000000,
0x00000000,0x00000000,0x00000000,0x00000000,0x00000000,0x00000000,0x00000000,0x00000000,
0x00000000,0x00000000,0x00000000,0x00000000,0x00000000,0x00000000,0x00000000,0x00000000,
0x00000000,0x00000000,0x00000000,0x00000000,0x00000000,0x00000000,0x00000000,0x00000000,
0x00000000,0x00000000,0x00000000,0x00000000,0x00000000,0x00000000,0x00000000,0x00000000,
0x00000000,0x00000000,0x00000000,0x00000000,0x00000000,0x00000000,0x00000000,0x00000000,
0x00000000,0x00000000,0x00000000,0x00000000,0x00000000,0x00000000,0x00000000,0x00000000,
0x00000000,0x00000000,0x00000000,0x00000000,0x00000000,0x00000000,0x00000000,0x00000000,
0x00000000,0x00000000,0x00000000,0x00000000,0x00000000,0x00000000,0x00000000,0x00000000,
0x00000000,0x00000000,0x00000000,0x00000000,0x00000000,0x00000000,0x00000000,0x00000000,
0x00000000,0x00000000,0x00000000,0x00000000,0x00000000,0x00000000,0x00000000,0x00000000,
0x00000000,0x00000000,0x00000000,0x00000000,0x00000000,0x00000000,0x00000000,0x00000000,
0x00000000,0x00000000,0x00000000,0x00000000,0x00000000,0x00000000,0x00000000,0x00000000,
0x00000000,0x00000000,0x00000000,0x00000000,0x00000000,0x00000000,0x00000000,0x00000000,
0x00000000,0x00000000,0x00000000,0x00000000,0x00000000,0x00000000,0x00000000,0x00000000,
0xEA01EA01,0xEA0102BD,0x82C442D0,0x42D082C4,0xEA01EA01,0xEA01EA01,0x82C442D0,0x42D082C4,
0x42E042E0,0x82DFA2E7,0xFA00FA00,0xFA00FA00,0x42E042E0,0x42E042E0,0x42E042E0,0x42E042E0,
0x32963296,0x32951288,0x3291428E,0x3293428E,0x12011201,0x12011201,0x3291428E,0x3293428E,
0x297A2978,0x0168F800,0x917FF97C,0x316C316C,0xF801F801,0xF801F801,0xF801F801,0xF801F801,
0x026D026D,0x12001269,0x126A3A83,0x126A3A83,0x12014A76,0x12014A76,0x12014A76,0x12014A76,
0xFADCFADC,0xFA01FA01,0xFA01FA01,0xFA01FA01,0xFA01FA01,0xFA01FA01,0xFA01FA01,0xFA01FA01,
0xFA01FA01,0xEA22EA1D,0x62236237,0x62236237,0xFA01FA01,0xFA01FA01,0xFA01FA01,0xFA01FA01,
0x00000000,0x00000000,0x00000000,0x00000000,0x00000000,0x00000000,0x00000000,0x00000000,
0xF801F801,0xE934E935,0xE936E936,0xE936E936,0xF801F801,0xF801F801,0xE936E936,0xE936E936,
0xE800E800,0x010A010C,0x090D090D,0x090D090D,0x010A010C,0x010A010C,0x090D090D,0x090D090D,
0x12191219,0x121AFA01,0x320A320A,0x320A320A,0xFA01FA01,0xFA01FA01,0xFA01FA01,0xFA01FA01,
0xF801F801,0xE8CF20D1,0x28D328DD,0x28D328DD,0xE801E801,0xE801E801,0xE801E801,0xE801E801,
0xE801E801,0x40C1E801,0xA0C460CC,0xA0C460CC,0xE801E801,0xE801E801,0xA0C460CC,0xA0C460CC,
0xE860E860,0xE8610062,0x20682068,0x20682068,0xE801E801,0xE801E801,0x20682068,0x20682068,
0x304E3037,0x304E3037,0x004F004F,0x004F004F,0xF801003D,0xF801003D,0xF801003D,0xF801003D,
0xF836F836,0xF8360002,0x402A4030,0x40054022,0xF801F801,0xF801F801,0x402A4030,0x40054022,
};
```

How to Reach Us:

Home Page:
www.freescale.com

E-mail:
support@freescale.com

USA/Europe or Locations Not Listed:
Freescale Semiconductor
Technical Information Center, CH370
1300 N. Alma School Road
Chandler, Arizona 85224
+1-800-521-6274 or +1-480-768-2130
support@freescale.com

Europe, Middle East, and Africa:
Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
support@freescale.com

Japan:
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku,
Tokyo 153-0064
Japan
0120 191014 or +81 3 5437 9125
support.japan@freescale.com

Asia/Pacific:
Freescale Semiconductor Hong Kong Ltd.
Technical Information Center
2 Dai King Street
Tai Po Industrial Estate
Tai Po, N.T., Hong Kong
+800 2666 8080
support.asia@freescale.com

For Literature Requests Only:
Freescale Semiconductor Literature Distribution Center
P.O. Box 5405
Denver, Colorado 80217
1-800-441-2447 or 303-675-2140
Fax: 303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com

rs

freescale™
semiconductor

Order Number AN2192/D