Motion Engineering, Inc.

33 South La Patera Lane
Santa Barbara, CA 93117
ph (805) 681-3300
fax (805) 681-3311
technical@motioneng.com

Release Note

DSP-Series
20MHz DSP Configuration

Option H001-00?? Firmware Version 2.4F4 Software Version 2.5 Revised 8/9/99

1.0 Introduction

The latest hardware revisions of Motion Engineering's DSP-Series controllers (PCX/DSP, STD/DSP, LC/DSP, 104/DSP, and VME/DSP) can be optionally configured for 20 MHz DSP operation. Standard DSP-Series controllers operate at 10 MHz. This document describes the new configuration.

This option includes special firmware which contains the code used by the DSP for 20 MHz operation and a sample program which demonstrates the new configuration.

2.0 20 MHz Operation

Standard DSP-Series controllers use a 10 MHz DSP clock from the FPGA. In 20 MHz mode, the FPGA creates a 20 MHz clock for the DSP. 20 MHz operation may be used when faster DSP sample rates are necessary. The following tables list the maximum DSP sample rates for 10 and 20 MHz operation.

Axes	10 MHz sample rates	20 MHz sample rates
1	7100 (Hz)	13700 (Hz)
2	4700	9100
3	3500	6800
4	2700	5400
5	2300	4500
6	2000	3900
7	1700	3400
8	1500	3000

3.0 Requirements

A.

The 20 MHz configuration is applicable to the latest hardware versions of DSP-series controllers. 20 MHz configuration is supported by the following hardware revisions (or newer):

Controller	*Rev*
PCX/DSP	3
STD/DSP	8
104/DSP	6
LC/DSP	8
CPCI/DSP	1
PCI/DSP	3
V6U/DSP	3

B.

The controller's DSP must have the **KP-80** designation, not **KP-40**. See below:

KP-40, not compatible

KP-80, correct version

C.

DSP-Series controllers have a PROM for every 4 axes which contains the FPGA logic code. The first EPROM (socket U1) contains the logic for axis 0 to 3 and the second EPROM (socket U54) contains the logic for axes 4 to 7. Several different FPGA models and manufacturers are used on the DSP-Series controllers.

Each FPGA model/manufacturer is only compatible with a specific PROM. If the controller is ordered with the special option, then the correct PROM will be installed at the factory. When ordering a special PROM for field installation, be sure to specify the controller's Serial Number. This will guarantee that the correct PROM will be sent with the order.

Configuring for 20 MHz operation on a standard board requires the standard 1.33 and 2.33 PROM (one for every four axes). To replace the 8 pin PROM, remove the controller from the computer. Locate the socketed 8 pin

PROM(s). Determine its orientation (the notch in the PROM indicates pin 1). Gently remove the PROM. A small flat blade screwdriver can be used to pry the chip from the socket if a chip remover is unavailable. Press the new PROM(s) into the socket. Be sure all the pins are seated properly and that chip is oriented the same as the one that was removed. Install the controller back into the computer and turn on the power. Each FPGA has an LED to indicate status. If the LED is green the PROM installation was successful. If the LED is red, turn off the power and check the PROM(s) orientation.

D.

If you have purchased your controller in conjunction with the 20 MHz option , your controller has already been configured at the factory with the special firmware. If you are purchasing the 20 MHz option to install onto an existing controller, follow these instructions to load the special firmware.

The new firmware must be downloaded to the controller by using the CONFIG program. The CONFIG program downloads firmware to the controller, configures the DAC offsets, and performs some basic tests of the axes.

NOTE!

Before running CONFIG, disconnect all of the cables from the DSP-Series controller and turn off the power to any external devices (amplifiers, etc.). All previous configurations stored in the DSP's boot memory will be lost when CONFIG is executed.

1. To download the firmware, switch to the directory where CONFIG.EXE is stored and type the following command at the DOS prompt:

   ```
   CONFIG -F 8AXIS.ABS
   ```

2. The CONFIG program will first load the standard **8AXIS.ABS** firmware and run some tests in order to determine the number of axes on the card and their internal offsets. Then the firmware will be downloaded and the internal offsets will be set.

3. If your board is configured for a base address other than the default (0x300), you will need to set an environment variable called *DSP* to the appropriate address. For example, to set the DSP base address setting to 0x280, type the following command at the DOS prompt:

   ```
   set DSP=base:0x280
   ```

The SETUP and CONFIG programs will then automatically read the *DSP* variable and access the board at address 0x280. Note that you may also insert this command into the AUTOEXEC.BAT file for automatic execution at system boot time.

.

4.0 Distribution Disk

One distribution disk is shipped with the 20 MHz configuration package:

File(s)	*Description*
CLOCK.C	Sample program source to demonstrate 20 MHz configuration.
8AXIS.ABS	Firmware version XX.XX for 20 MHz

operation

If the DSP-Series controller was ordered with the 20 MHz option, then the special firmware was installed at the factory and no further installation steps are necessary. If you need to load the new firmware, execute the following command at a DOS prompt: **`config -d 8axisb.abs`**. This will load the 20 MHz firmware onto the controller.

5.0 Implementation

Once the new firmware has been downloaded and the new PROM installed, an axis can be configured for 20 MHz operation using the following commands: (see CLOCK.C):

```
int16 set_dsp_clock_mode(int16 mode);

int16 get_dsp_clock_mode(void);
```

6.0 Sample Program

```
/* CLOCK.C

:Configures the DSP to use a 20Mhz clock.


Warning!  This is a sample program to assist in the integration of the
 DSP-Series controller with your application.  It may not contain all
 of the logic and safety features that your application requires.

Written for Version 2.5
*/

/* Revision Control System Information
  $Source$
  $Revision$
  $Date$

  $Log$
*/


# include <stdio.h>
# include <stdlib.h>
# include <dos.h>
# include "idsp.h"

# define  MEI_20MHZ           1
# define  MEI_10MHZ           0

# define  MEI_ADDRESS            0
# define  MEI_DATA           2
# define  MEI_RESET_ENABLE    4
# define  MEI_RESET_DISABLE      5
# define  MEI_20MHZ_ENABLE    8
# define  MEI_10MHZ_ENABLE    9

# define  MEI_PM_CONFIG       (0x3E0 * 4)
# define  MEI_PM_CONFIG_WAITS    (MEI_PM_CONFIG + 0x17 * 4)
# define  MEI_PM_CONFIG_SYS      (MEI_PM_CONFIG + 0x18 * 4)
# define  MEI_PM_CONFIG_REBOOT   (MEI_PM_CONFIG + 0x19 * 4)

# define  BM_PAGE_0           0x0000       /* (starting addresses) */
# define  BM_PAGE_1           (2 * BM_PAGE)
```

```
void error(int error_code)
{
  char buffer[MAX_ERROR_LEN];

  switch (error_code)
  {
    case DSP_OK:
        /* No error, so we can ignore it. */
        break ;

    default:
        error_msg(error_code, buffer) ;
        fprintf(stderr, "ERROR: %s (%d).\n", buffer, error_code) ;
        exit(1);
        break;
  }
}


int16 set_dsp_clock_mode(int16 mode)
{
  unsigned i, sum = 0;

  int16 iobase = dspPtr->iobase;

/* Disable the DSP. */
  DSP_OUTB(MEI_RESET_ENABLE + iobase, 0);

/* Configure the clock for proper operation. */
  switch (mode)
  {
  /* 10MHz operation. */
    case 0:
        DSP_OUTB(MEI_10MHZ_ENABLE + iobase, 0) ;
        DSP_OUT(MEI_ADDRESS + iobase, MEI_PM_CONFIG_WAITS | PCDSP_BM);
        DSP_OUT(MEI_DATA + iobase, 0x12);
        DSP_OUT(MEI_ADDRESS + iobase, MEI_PM_CONFIG_WAITS + 1 | PCDSP_BM);
        DSP_OUT(MEI_DATA + iobase, 0x4B);
        DSP_OUT(MEI_ADDRESS + iobase, MEI_PM_CONFIG_SYS | PCDSP_BM);
        DSP_OUT(MEI_DATA + iobase, 0x00);
        DSP_OUT(MEI_ADDRESS + iobase, MEI_PM_CONFIG_SYS + 1 | PCDSP_BM);
        DSP_OUT(MEI_DATA + iobase, 0x19);
        DSP_OUT(MEI_ADDRESS + iobase, MEI_PM_CONFIG_REBOOT | PCDSP_BM);
        DSP_OUT(MEI_DATA + iobase, 0x0E);
        DSP_OUT(MEI_ADDRESS + iobase, MEI_PM_CONFIG_REBOOT + 1 | PCDSP_BM);
        DSP_OUT(MEI_DATA + iobase, 0x59);
        break;

  /* 20MHz operation. */
    case 1:
        DSP_OUTB(MEI_20MHZ_ENABLE + iobase, 0) ;
        DSP_OUT(MEI_ADDRESS + iobase, MEI_PM_CONFIG_WAITS | PCDSP_BM);
        DSP_OUT(MEI_DATA + iobase, 0x24);
        DSP_OUT(MEI_ADDRESS + iobase, MEI_PM_CONFIG_WAITS + 1 | PCDSP_BM);
        DSP_OUT(MEI_DATA + iobase, 0x95);
        DSP_OUT(MEI_ADDRESS + iobase, MEI_PM_CONFIG_SYS | PCDSP_BM);
        DSP_OUT(MEI_DATA + iobase, 0x00);
        DSP_OUT(MEI_ADDRESS + iobase, MEI_PM_CONFIG_SYS + 1 | PCDSP_BM);
        DSP_OUT(MEI_DATA + iobase, 0x1A);
        DSP_OUT(MEI_ADDRESS + iobase, MEI_PM_CONFIG_REBOOT | PCDSP_BM);
        DSP_OUT(MEI_DATA + iobase, 0x0E);
        DSP_OUT(MEI_ADDRESS + iobase, MEI_PM_CONFIG_REBOOT + 1 | PCDSP_BM);
        DSP_OUT(MEI_DATA + iobase, 0x5A);
        break;
  }

  /* Recompute the checksum. */
  for(i = BM_PAGE_0; i < (BM_PAGE+BM_PAGE_0); i++)
  { DSP_OUT(MEI_ADDRESS + iobase, i | PCDSP_BM);
    sum += (DSP_IN(MEI_DATA + iobase) & 0xFF);
  }

  for(i = BM_PAGE_1; i < BM_CHECKSUM; i++)
  { DSP_OUT(MEI_ADDRESS + iobase, i | PCDSP_BM);
```

```
    sum += (DSP_IN(MEI_DATA + iobase) & 0xFF);
  }

/* Write the checksum */
  DSP_OUT(MEI_ADDRESS + iobase, BM_CHECKSUM | PCDSP_BM);
  DSP_OUT(MEI_DATA + iobase, sum & 0xFF);
  sum >>= 8 ;
  DSP_OUT(MEI_ADDRESS + iobase, (BM_CHECKSUM + 1) | PCDSP_BM);
  DSP_OUT(MEI_DATA + iobase, sum & 0xFF);

/* ***Note*** We don't store this to non-volatile boot memory. */

/* Wait at least 2ms for the PLL to catch up. */
  delay(3) ;

/* Restart the DSP. */
  DSP_OUTB(MEI_RESET_DISABLE + iobase, 0);

  return DSP_OK ;
}



int16 get_dsp_clock_mode(void)
{
  int16 clock_id, base, mode;

  base = dspPtr->iobase;

  /* Check to see if the board is running at 20MHz, by writing the address
    of the DSP's signature word into the ID address.  If the controller is
    running at 20MHz, the ID low byte (0x8) will be aliased to the
    address low byte (0x0) and the ID high byte (0x9) will be aliased to
    the address high byte (0x1).

      If the board is running at 20MHz and we write the DSP's signature address,
    we will be able to read the DSP's signature word from the data register. */

  DSP_OUT((int16)(base + MEI_20MHZ_ENABLE), (unsigned16)(PCDSP_DM | DM_SIGNATURE));
  clock_id = DSP_IN((int16)(base + MEI_DATA));


  if (clock_id == PCDSP_SIGNATURE)
  {
    mode = MEI_20MHZ;
  }
  else
  {
    mode =  MEI_10MHZ;
  }

  return mode;

}



int main()
{
  int error_code, mode;

  error_code = do_dsp();   /* initialize communication with the controller */
  error(error_code);    /* any problems initializing? */

  set_dsp_clock_mode(MEI_20MHZ); /* set clock mode to 20Mhz */

  printf("Clock mode is %d \n", get_dsp_clock_mode()); /* read current clock mode */


    return 0;
}
```