

Innovative Integration

GRABBER44 Hardware/Software Manual

Innovative Integration, Inc. makes no warranty of any kind with regard to this material, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. Innovative Integration, Inc. shall not be liable for errors contained herein or for incidental or consequential damages in connection with the furnishing, performance, or use of this material.

GRABBER44 is a trademark of Innovative Integration, Inc. All other brand and product names are trademarks or registered trademarks of their respective companies.

Innovative Integration products may not be used in life support applications without express written consent of the corporation.

Copyright 1998 by Innovative Integration, Inc.

This document contains information proprietary to Innovative Integration, Inc. Any reproduction, disclosure, or unauthorized use of this document, either in whole or in part, is expressly forbidden without prior permission in writing from:

Innovative Integration, Inc.

5785 Lindero Canyon Road
Westlake Village, CA 91362

Phone	(818) 865-6150
Fax	(818) 879-1770
Email	techsprt@innovative-dsp.com

1. Introduction

The GRABBER 44 is a TIM40 standard compliant processor module based around the Texas Instruments TMS320c44 processor. The module implements a single processor with 512 Kwords of local SRAM and 4Mbits of Flash EEPROM and implements 3 ultra high speed A/D's for capturing and processing transients and other high speed signals.

1.1 Installation

The GRABBER 44 requires a TIM40 compatible carrier card for proper operation. The GRABBER 44 is installed in on of the carrier board's TIM connector sites. Consult the hardware installation instructions provided by the host board's manufacturer (i.e. II's PC 44 hardware manual) before installing the GRABBER 44.

Typically, the GRABBER 44 is to be used with an Innovative Integration PC44 or PCI44 and should be installed in the appropriate TIM site that each GRABBER 44 is labeled with (note: this site may be changed by re-burning the flash on the GRABBER 44 with talker for a different site (see host manual)).

When installing the GRABBER 44:

1. **Turn power to the computer off.**
2. Remove the screws holding the two cards together and carefully separate the two cards pulling them apart while keeping both cards parallel. **DO NOT ROTATE THE CARDS WITH RESPECT TO ONE ANOTHER AS THIS WILL DAMAGE THE CARDS.**
3. Carefully align the module with the TIM connectors over the host cards TIM connectors. The connectors are keyed and the module may only be installed in one direction.
4. Press the module into the hosts TIM site being careful not to use excessive force. **DO NOT EXERT FORCE ON THE SEMICONDUCTOR DEVICES.** Press only on the printed circuit board, or with some foam, press directly on the back side of the TIM connectors. Be sure to seat the module completely.
5. Re-connect the mezzanine card to the module again being careful not to rotate the cards with respect to one another.
6. Re-install the screws to secure the two cards together.

Power and cabling connections:

1. Before installing the host card into the computer, decide what type of clock distribution will be necessary if any (see Figure 1).
 - To use sample clock on board, connect Internal clock output to clock input.
 - To use a single on board clock for two cards, connect card with desired sample clock (card 1) as above and card 2 clock input should be tied to card 1's external clock output.
 - To use an external clock, tie external clock to clock input line.
2. Connect analog inputs to corresponding channels on the GRABBER 44 (see Figure 1).
3. Connect the power connectors to the cable provided, and connect the cable to a free power supply connector or inline with any other power connector in the host computer.
4. With the power to the computer still off, carefully insert the card into the appropriate slot.
5. Turn on the system and begin.

2. Hardware Discussion

The Grabber 44 is comprised of five major sections (see figure 2 below):

1. 3 Analog input sections using Ultra Low Distortion Op Amps.
2. 3 12 bit 40 MHz A/D's.
3. 3 12 x 16k deep FIFO's.
4. Onboard Logic for clock phase selection and peripheral control.
5. Dedicated TMS320C44 DSP with access to Global bus and com ports.

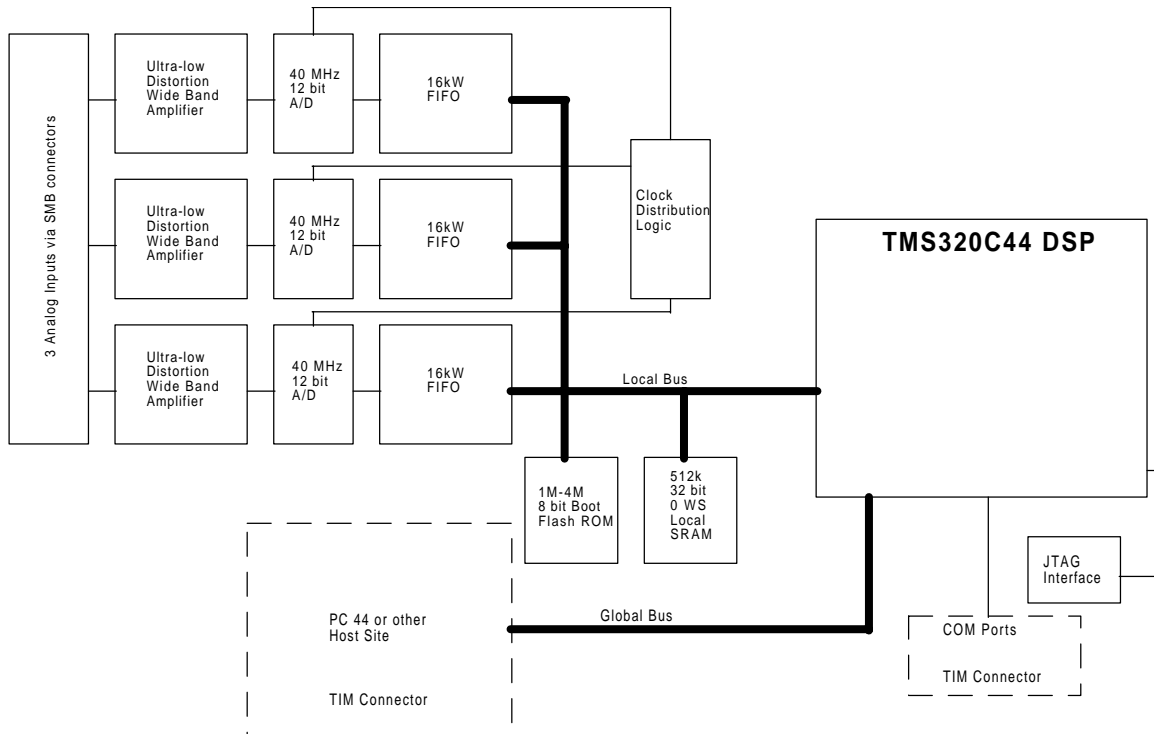


Figure 2: GRABBER 44 Hardware Block Diagram

The Grabber 44 module is a 3 channel, ultra high speed data acquisition TIM40 compatible module with its own dedicated C44 DSP which allows the user to have a great deal of flexibility and processing power. Up to two of these modules may be mounted on a PC44 or PCI44 and any of these modules may directly or transparently access its own 512k x 32 of local memory, plus all of the host DSP's global memory. Also, data may be routed via global bus or COM port to any other processor. The sampling clock may also be routed to each individual A/D at either 0°, 45°, or 90° thus effectively increasing the sample frequency of the system.

2.2 Analog input section

The analog input section (see figure 3) for each channel is comprised of two unity gain amplifiers. The first for buffering the signal and the second to level shift the signal for input to the A/D. Both are ultra low distortion wide band amplifiers to provide an undistorted input to the A/D. The large signal (4Vp-p) bandwidth of the amplifiers is 150 MHz. The input to the board is via 3 SMB plugs mounted to the module. Each input is terminated with a 50 Ohm resistor.

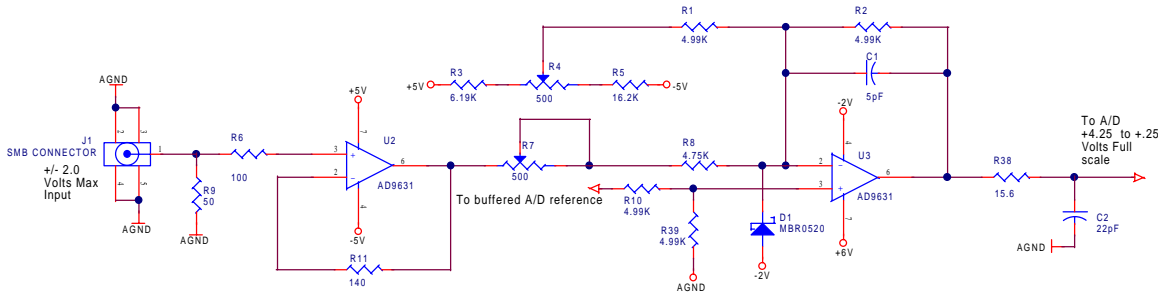


Figure 3. Analog input section for one channel.

Full scale input to the A/D is +4.25 Volts to +.25 Volts corresponding to an analog input at the connector (SMB type) of +/- 2 Volts. The gain of the analog section may be changed, but care must be taken not to reduce the bandwidth of the amplifier to the point the input wave is distorted. Also, two pots are provided to adjust linearity and offset (see Table 1).

Adjustment	Channel 1	Channel 2	Channel 3
Gross Gain Change	R2 & R39	R13 & R41	R24 & R43
Offset Adjustment	R4	R15	R26
Gain Adjustment	R7	R18	R29

Table 1.

2.3 A/D's & FIFO's

The 12 bit 40 MHz A/D's include a 12 bit quantizer, wideband track & hold and are connected to accept a single ended 4 Volt pk-pk input. The A/D's are configured to continually convert incoming data unless the clock to that particular channel is turned off (see software section). The clock used to convert data should have a minimum high time of 13ns. This means that for sampling rates greater than 38 MHz, the duty cycle must be greater than 50% (see data sheet for A/D in the appendix).

The data output from the A/D is then written to the FIFO connected to that A/D on a write command and continues to fill the FIFO until the FIFO is full or the write has been deasserted. The FIFO's are each 16k deep and can be read with 0 wait states from the processor. The FIFO's can be collectively programmed to give an interrupt to the processor on either a FIFO Full Flag, a FIFO Half Full Flag, or a FIFO Not Empty Flag, but the FIFO will continue to fill until the write is deasserted. Output data from the FIFO's is arranged on the upper 12 bits of the lower half of the data bus (D15 MSB - D4 LSB) for each FIFO to ease handling of the data in software.

2.4 Clock Phase Selection

There is a PLL on board that will output 0°, 45°, & 90° phase shifted clocks based on an input clock between 20-40 MHz. Each clock can be independently routed via logic to any A/D. For lower frequency clocks, a clock at four times the desired sampling rate must be input to the module. This clock is then divided down (see software section) and the 0°, 45°, & 90° phase shifted clocks will again be available to each A/D. The input clock may either be a clock module on the board or piped in externally via an SMB connector.

2.4.1 Low Skew Between Multiple Modules

If low skew measurements are required between modules, matched length cables may be used to feed in an external clock, or a single clock module may be used on one board and fed back to itself and the other module with the matched length cables. This will allow the user to take simultaneous samples on up to 6 channels.

3.0 Software

Software written for the GRABBER 44 should be built with the same tools and peripheral libraries that come with II's PC 44 and PCI 44. Each GRABBER 44 comes from the factory with talker and boot up code burned on the flash for a specific site on the host card. This code may be modified and re-burned into the flash using II's burn utility if necessary (see Software Manual for Host card).

Table 2 below shows the memory mapped addresses of the various peripheral functions of the card as well as how each address is set up. A header file GRABBER.H is included with the card setting up these C mnemonics at the addresses listed.

Function	Mnemonic	Address	Operation
A/D Clock Phase Selection for Channel 1	ADC_1_PHASE	0x00480000	Write 0-0°, 1-45°, 2-90°, 3-OFF
A/D Clock Phase Selection for Channel 2	ADC_2_PHASE	0x00440000	Write 0-0°, 1-45°, 2-90°, 3-OFF
A/D Clock Phase Selection for Channel 3	ADC_3_PHASE	0x00400000	Write 0-0°, 1-45°, 2-90°, 3-OFF
Data output format	ADC_MODE	0x00500000	Write 0- Offset Binary, 1- Two's Complement
Set FIFO interrupt Flags	FIFO_MODE	0x00540000	Write 1- Not Empty, 2- Half Full, 3- Full
Read or Write to FIFO Channel 1	FIFO_1	0x00640000	Write 1- Fill FIFO, 0-Stop Fill, Read- Data
Read or Write to FIFO Channel 2	FIFO_2	0x00600000	Write 1- Fill FIFO, 0-Stop Fill, Read- Data
Read or Write to FIFO Channel 3	FIFO_3	0x005c0000	Write 1- Fill FIFO, 0-Stop Fill, Read- Data
Select type of input clock (1x or 4x)	COUNTER_LOAD	0x004c0000	Write 0xFFFFFFFF for divide by 4, 0 - for no division of the input clock
Clear the FIFO's	FIFO_RESET	0x00580000	Write any value to initiate

Table 2. Memory Map of Grabber 44.

3.1 Sample Application

The GRABBER 44 system comes with a sample application program to test the hardware's functionality and demonstrate the use of the card. GRBBRTST.C may be used to test the basic functionality of the board as well as provide an example of how to use the board. The program allows the user to fill the FIFO's and receive an interrupt at one of three locations: NOT EMPTY, HALF FULL, or FULL. and then download that data into local SRAM and measure the min., max., mean and standard deviation. The program gives the user the option to do this at one of three clock phases (0, 45, & 90) for each A/D and choose whether or not to divide the input clock by four.

```

/*
 *
 * GRBBRTST.C
 *
 * Demonstrate data acquisition from 3 internal, free-running
 * A/D converters to 3 internal FIFO's to local SRAM using one
 * of three different phases of sample clock. A statistical analysis
 * is then done on each channels sampled data.
 *
 */

#include "periph.h"
#include "grabber.h"
#include "stdio.h"
#include "intpt40.h"

```

```

#include "dsp.h"

void          c_int03(); /* FIFO 1 Flag */
void          c_int04(); /* FIFO 2 Flag */
void          c_int06(); /* FIFO 3 Flag */

#define SIZE 16384      /* FIFO Depth */

volatile double buffer1[SIZE], buffer2[SIZE], buffer3[SIZE];
volatile short point = 0, fifo_flag, array;

STATISTICS statistics1;
STATISTICS statistics2;
STATISTICS statistics3;

main()
{
    int col, row;
    volatile short latched_data;
    short then, rate;
    char dummy_char;

    memset(&buffer1,0,SIZE);
    memset(&buffer2,0,SIZE);
    memset(&buffer3,0,SIZE);

    cpu(cpu_num());
    MHZ = 50;

    enable_cache();
    enable_interrupts();
    enable_monitor();

    *LBCR = 0x37a50050;      /* Set lstrb0 to 0 wait state */

    *ADC_MODE = 1;          /* 0-Offset Binary, 1-Two's Complement*/

    *FIFO_1 = 0;            /* Disables FIFO reads */
    *FIFO_2 = 0;            /* Disables FIFO reads */
    *FIFO_3 = 0;            /* Disables FIFO reads */

    *FIFO_RESET = 1;        /* Clears the FIFO's */

    do
    {
        *FIFO_RESET = 1;      /* Clears all FIFO's */

        clrscr();
        gotoxy(28, 0);
        bold();
        printf("\7Grabber 44 Test\n\n");
        normal();
    }
}

```

```

cursor(OFF);

printf("\n\nDo you wish to divide sample clk by four? ");

dummy_char = toupper(getchar());

if (dummy_char == 'Y')
    {
        *COUNTER_LOAD = 0; /* A counter load of any thing but 0xffffffff divides
        * clock by 4 */
    }
else
    {
        *COUNTER_LOAD = 0xffffffff;
    }

printf("\n\nEnter ADC 1 clock phase (0=0, 1=45, 2=90): ");
scanf("%d",&ADC_1_PHASE); /* 3 turns off clock to that channel */

printf("\n\nEnter ADC 2 clock phase (0=0, 1=45, 2=90): ");
scanf("%d",&ADC_2_PHASE);

printf("\n\nEnter ADC 3 clock phase (0=0, 1=45, 2=90): ");
scanf("%d",&ADC_3_PHASE);

printf("\n\nEnter FIFO flag mode (1=NEF, 2=HF, 3=FF): ");
scanf("%d",&fifo_flag);
*FIFO_MODE = fifo_flag;

if(fifo_flag = 3)
    array = 16384;
else if(fifo_flag = 2)
    array = 8192;
else
    array = 0;

printf("\n\nPress Any Key to sample channel #1. ");
getchar();

set_iiof(0, 0x9); /* Clears flags & enables interrupt 0 - FIFO 1 */

install_int_vector((void *)c_int03, 03);
*FIFO_1 = 1; /* Latches and begins filling FIFO on channel #1 */

while(point != array); /* Wait until FIFO flag occurs */

deinstall_int_vector(03);
point = 0;

printf("\n\nCOMPLETE!!");

if(array != 1)
    {
        cursor(ON);
        buffer_statistics(&statistics1, buffer1, array);
    }

```

```

        printf("\n\n Delta: %9.3f", statistics1.delta);
        printf( "\n Mean: %9.3f", statistics1.mean);
        printf( "\nStd Dev: %9.3f", statistics1.sdev);
        printf( "\nMinimum: %9.3f", statistics1.minimum);
        printf( "\nMaximum: %9.3f", statistics1.maximum);
        cursor(OFF);
    }

printf("\nPress Any Key to sample channel #2. ");
getchar();

        set_iiof(1, 0x9);          /* Clears flags & enables interrupt 1 - FIFO 2 */

install_int_vector((void *)c_int04, 04);
*FIFO_2 = 1;          /* Latches and begins filling FIFO on channel #2 */

while(point != array);    /* Wait until FIFO flag occurs */

deinstall_int_vector(04);
point = 0;

printf("\nCOMPLETE!!");

if(array != 1)
{
    cursor(ON);
    buffer_statistics(&statistics2, buffer2, array);

    printf("\n\n Delta: %9.3f", statistics2.delta);
    printf( "\n Mean: %9.3f", statistics2.mean);
    printf( "\nStd Dev: %9.3f", statistics2.sdev);
    printf( "\nMinimum: %9.3f", statistics2.minimum);
    printf( "\nMaximum: %9.3f", statistics2.maximum);
    cursor(OFF);
}

printf("\nPress Any Key to sample channel #3. ");
getchar();

        set_iiof(3, 0x9);          /* Clears flags & enables interrupt 3 - FIFO 3 */

install_int_vector((void *)c_int06, 06);
*FIFO_3 = 1;          /* Latches and begins filling FIFO on channel #3 */

while(point != array);    /* Wait until FIFO flag occurs */

deinstall_int_vector(06);
point = 0;

printf("\nCOMPLETE!!");

if(array != 1)
{
    cursor(ON);

```

```

        buffer_statistics(&statistics3, buffer3, array);

        printf("\n\n Delta: %9.3f", statistics3.delta);
        printf( "\n Mean: %9.3f", statistics3.mean);
        printf( "\nStd Dev: %9.3f", statistics3.sdev);
        printf( "\nMinimum: %9.3f", statistics3.minimum);
        printf( "\nMaximum: %9.3f", statistics3.maximum);
        cursor(OFF);
    }

    printf("\n\nHit ESC key to stop, any other to test again.");
    dummy_char = getchar();

}while(dummy_char != 0x1b)

monitor();
} /* End of main() */

/*
 * c_int03() services IIOF 0 which occurs at Full, Half full, or on
 * a single data word being read in depending on value of *FIFO_MODE.
 * It dumps the data in the FIFO to local SRAM.
 */
void c_int03()
{
    *FIFO_1 = 0;
    point = 0;
    while(point < array)
    {
        buffer1[point++] = (double)((*FIFO_1 << 16) >> 20);
    }
}

/*
 * c_int04() services IIOF 1 which occurs at Full, Half full, or on
 * a single data word being read in depending on value of *FIFO_MODE.
 * It dumps the data in the FIFO to local SRAM.
 */
void c_int04()
{
    *FIFO_2 = 0;
    point = 0;
    while(point < array)
    {
        buffer2[point++] = (double)((*FIFO_2 << 16) >> 20);
    }
}

/*
 * c_int06() services IIOF 3 which occurs at Full, Half full, or on

```

* a single data word being read in depending on value of *FIFO_MODE.
* It dumps the data in the FIFO to local SRAM.

*

*/

```
void c_int06()
```

```
{
```

```
    *FIFO_3 = 0;
```

```
    point = 0;
```

```
    while(point < array)
```

```
    {
```

```
        buffer3[point++] = (double)((*FIFO_3 << 16) >> 20);
```

```
    }
```

```
}
```

The program can be loaded either with II's Terminal program or with a JTAG controller and a Debugger like "Code Composer" (for the standard IO to work in a debugger Terminal must be running) (see host card software manual).